ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN TECNOLÓGICA

PROGRAMADOR DEL PIC 16F628 UTILIZANDO EL PORTICO USB

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

MARTÍNEZ ROSERO SILVANA PATRICIA
SILVA ALVAREZ JENNY ELIZABETH

DIRECTOR: ING. ALCÍVAR COSTALES

Quito, Marzo, 2006

DECLARACIÓN

Nosotros, Silvana Patricia Martínez Rosero y Jenny Elizabeth Silva Alvarez, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

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

Silvana P. Martínez R. **Jenny E. Silva A.**

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Silvana Patricia Martínez Rosero y Jenny Elizabeth Silva Alvarez, bajo mi supervisión.

Ing. Alcívar Costales
DIRECTOR DE PROYECTO

DEDICATORIA

A Dios por habernos dado la guía, la fortaleza y paciencia para seguir adelante a pesar de las adversidades presentadas a lo largo de esta importante etapa de nuestra vida estudiantil.

A nuestros padres quienes nos apoyaron con entusiasmo y optimismo, depositando en nosotros su confianza para la culminación de nuestros estudios.

Jenny y Silvana

ÍNDICE

CAPÍTULO 3.

3. PROGRAMA PARA EL GRABADOR DEL PIC 16F628

CAPÍTULO 4.

4. CONSTRUCCIÓN DEL DISPOSITIVO

INTRODUCCIÓN

En la actualidad la tecnología evoluciona a grandes pasos, especialmente en el mundo de la computación, es así que nace el puerto USB (Universal Serial Bus) que como su nombre lo indica, se trata de un sistema de comunicación entre dispositivos electrónicos-informáticos que sólo transmite una unidad de información a la vez. De igual manera el microcontrolador es hoy en día básico para el imperio tecnológico, ya que dentro de unos años estaremos rodeados de microcontroladores y serán indispensables para la vida cotidiana. Por este motivo es importante el conocimiento y la utilización de estos circuitos integrados, que en un principio los llamaron microcomputadores y hoy en día los conocemos como poderosos microcontroladores, y es aquí donde surge el llamado PIC que lleva el liderazgo por su bajo costo, fácil programación y la gran cantidad de modelos a elegir según sea la necesidad.

Para la grabación de este microcontrolador se tienen programadores que utilizan el puerto serial y el puerto paralelo, pero como se dijo anteriormente la comunicación a través del puerto USB es mucho más rápida que las anteriores mencionadas. Por esta razón hemos orientado este trabajo de investigación a la elaboración de un programador de PICs y específicamente el PIC16F628, que utilice el pórtico USB por las grandes ventajas que presenta ante los puertos serie y paralelo.

La utilización del microcontrolador 18F2550 nos permitió lograr el objetivo que nos hemos propuesto, ya que cumple con las necesidades requeridas para este proyecto por su configuración y estructura.

RESUMEN

El objetivo de este trabajo es la realización de un Programador del PIC16F628 utilizando el pórtico USB para ser utilizado en el Laboratorio de Microprocesadores de la carrera de Tecnología en Electrónica y Telecomunicaciones. El trabajo escrito da a conocer la información que fue necesaria para el desarrollo del proyecto.

En el capítulo 1 se da una breve teoría en lo que respecta al puerto USB, las características más sobresalientes, su funcionamiento ventajas y limitaciones.

En el capítulo 2 se estudia las principales características del PIC16F628, su funcionamiento y se dará a conocer el lenguaje de programación utilizado.

En el capítulo 3 se habla sobre el programa que fue necesario utilizar en el PIC18F2550 para la interfaz del programador del PIC16F628.

En el capítulo 4 se indican los diagramas esquemático y circuital del dispositivo y la instalación paso a paso del software utilizado para el funcionamiento del programador.

Finalmente se presentan conclusiones y recomendaciones obtenidas luego de la realización del proyecto, a más de que se incluye la bibliografía y anexos que fueron necesarios para el desarrollo del mismo.

PRESENTACIÓN

Actualmente en el Laboratorio de Microprocesadores de la carrera de Tecnología en Electrónica y Telecomunicaciones de la Escuela Politécnica Nacional se está incrementando el estudio de los microcontroladores PIC, especialmente del 16F628; pero para este estudio y aprendizaje hay que realizar pruebas con distintos proyectos, por lo que es necesario tener un programador de fácil uso para el estudiante y de mayor velocidad que los tradicionales, para agilitar la grabación del los PICs con los programas necesarios en la realización de diferentes proyectos y aplicaciones.

De esta manera surgió la idea de hacer un Programador para el PIC16F628 utilizando el pórtico USB que es mucho más rápido y fácil de utilizar que los pórticos tradicionales como es el serial y el paralelo.

La realización de éste programador es posible gracias a la utilización del PIC18F2550 que sirve de interfase entre el programador de PICs por medio del puerto USB y el computador.

En este proyecto también se trata sobre el lenguaje de programación Picbasic Pro el cual permite programar de manera fácil y rápida una gran variedad de microcontroladores PIC, puesto que el lenguaje utilizado por este compilador se basa en el lenguaje BASIC que es mucho más entendible y manejable por los estudiantes que el lenguaje Ensamblador.

CAPITULO 1.

1. PUERTO USB

1.1 ANTECEDENTES [5]

Figura 1.1. Conector USB

Se ha señalado repetidamente que el PC adolece de una serie de deficiencias que podríamos llamar "congénitas", heredadas de un diseño deficiente en algunos aspectos, entre las que cabría destacar la escasez de determinados recursos tales como: líneas de interrupción IRQs, y canales de acceso directo a memoria DMA.

En ambos casos las capacidades del diseño inicial tuvieron que ser dobladas en 1984, tres años después de su lanzamiento, aprovechando la aparición de la gama AT.

La instalación de periféricos ha sido un constante problema para los ensambladores, que debían asignar los escasos recursos disponibles entre la creciente variedad de dispositivos que debían conectarse a los sistemas.

En este sentido, aunque el estándar PnP ("Plug and Play") vino a aliviar en parte las dificultades mecánicas de cambiar "jumpers" en las placas, el problema seguía ahí, ya que desde la aparición del AT el diseño del PC no había sufrido cambios sustanciales.

Como resultado de un intento de dotar al PC de un bus de alta velocidad que ofreciera las características ideales PnP de universalidad, facilidad de conexión y desconexión, incluso en caliente ("Hot Swappable"), y sobre todo, que consumiese pocos recursos, Intel y otros líderes de la industria diseñaron el Bus Universal Serie, más comúnmente conocido por su acrónimo inglés USB ("Universal Serial Bus"), que como su nombre lo indica, es un bus serie, bidireccional y de bajo costo, desarrollado tanto por industrias de computación como de telecomunicaciones.

Este dispositivo USB, permite adjuntar dispositivos periféricos a la computadora rápidamente, con la ventaja de que no se necesita reiniciar la computadora ni

volver a configurar el sistema. Los dispositivos con USB se configuran automáticamente tan pronto como se han conectado físicamente.

En las computadoras que cuentan con esta tecnología se puede observar dos conectores de este tipo. Además, se pueden unir dispositivos con USB en una cadena para conectar más de dos dispositivos a la computadora mediante otros periféricos USB (figura 1.2 y 1.3).

Figura 1.2. Hub USB

Figura 1.3. Conectores USB

1.2 HISTORIA [5]

Así como ha avanzado la tecnología en los distintos periféricos y elementos electrónicos, mejorando ya sea su resolución, sonido, etc. También los dispositivos de conexión han tenido que mejorar a la par de esto. En un principio se tenía la interfaz serie y paralelo, pero era necesario unificar todos los conectores creando uno más sencillo y de mayores prestaciones. Así nació el USB (Universal Serial Bus) que en sus primeras instancias (versión 1.0) fue diseñado para conectar periféricos como: módems, ratones, teclados, monitores, equipos estereofónicos, lectores de CD de baja velocidad a 4x o 6x, unidades de disquete, digitalizadores de imagen de baja resolución (scanner), teléfonos, conexiones ISDN, impresoras, unidades para almacenamiento en cinta, etc.

En resumen toda clase de dispositivos existentes y los que vayan a crearse aprovechando las ventajas USB; la única condición, era que el dispositivo no requiera de rangos de transmisión superiores a los 12 Mbps (también con la opción de transmisiones a 1,5 Mbps para dispositivos de baja velocidad, entre ellos el mouse) esto significa que las tarjetas de video, tarjetas de red a 100 Mbps y controladoras de discos duros particularmente, seguirían siendo tarjetas

conectadas al interior de la PC. Dadas estas velocidades el Universal serial bus, es capaz de soportar hasta 127 dispositivos conectados directamente a la PC o Host USB, y el resto se Irán conectando entre si de forma encadenada o bien empleando Hub USB.

Como se dijo anteriormente, la tecnología de conexión de los computadores ha tenido que modificar su características, para poder entregar mayor calidad y a la vez poder subsistir en el mercado de la tecnología, estas modificaciones, en el caso del USB se encuentran en la velocidad de transmisión, con la cual se plantea que en estos momentos los 12 Mbps, serian revisados para dar paso a una velocidad 20 y 30 veces mayor que esta, pudiendo así ofrecer compatibilidad con las aplicaciones de usuario más exigentes sin incrementar el costo o la complejidad con respecto al objetivo anterior.

A principio de 1999, el Grupo Promotor de USB 2.0 o USB de alta velocidad, compuesto por Compaq, HP, Intel, Lucent, Microsoft, NEC y Philips, anunció que la velocidad de USB 2.0 sería de 120 a 240 mega bits por segundo (Mbps), o 10 a 20 veces más rápida que la de USB 1.1. Este incremento más reciente eleva la velocidad ahora de 360 a 480 Mbps, o 30 a 40 veces más rápida que la de USB 1.1. La velocidad significativamente más alta es el resultado de análisis realizados por el Grupo Promotor de USB 2.0 que concluyen que la velocidad se puede incrementar sin costo o complejidad adicionales con respecto al anterior.

El primer ordenador que incluyó un puerto USB de forma estándar fue el iMac de Apple, presentado en Marzo de 1998, que utilizaba esta conexión para el teclado y el ratón. Por su parte el mundo del PC solo comenzó a utilizarlo cuando Microsoft introdujo los controladores correspondientes en la versión OSR 2.1 de Windows 95. Fue a partir de Windows 95C cuando los sistemas de MS incorporan de forma estándar soporte para este bus; en el ámbito de servidores la incorporación se produjo en Windows 2000.

USB es una nueva arquitectura de bus o un nuevo tipo de bus que forma parte de los avances plug and play y permite instalar periféricos sin tener que abrir la

máquina para instalarle hardware, es decir, basta con conectar dicho periférico en la parte posterior del computador y listo. Los primeros dispositivos que empezaron a utilizar masivamente este tipo de conexión fueron las cámaras de video-conferencia, aunque actualmente pueden encontrarse todo tipo de dispositivos con este tipo de conexión.

1.3 ESTÁNDARES [5]

La tecnología USB ha sido promovida principalmente por Intel, aunque le han seguido todos los grandes fabricantes, de forma que se ha convertido en un estándar importante. En sus comienzos los interesados en esta tecnología se agruparon en un foro, el USB Implementers Forum Inc., USB-IF, que agrupa a más de 460 compañías, y ha publicado diversas revisiones de la norma:

- **USB 0.9**: Primer borrador, publicado en Noviembre de 1995.
- **USB 1.0**: Publicada en 1996 establece dos tipos de conexión: La primera, denominada de Velocidad baja ("Low speed"), ofrece 1.5 Mbps, y está pensada para periféricos que no requieren un gran ancho de banda, como ratones o joysticks. La segunda, denominada de Velocidad completa ("Full speed"), es de 12 Mbps, y está destinada a los dispositivos más rápidos.
- **USB 1.1**: Publicada en 1998, añade detalles y precisiones a la norma inicial; es el estándar mínimo que debe cumplir un dispositivo USB.
- **USB 2.0**: Su versión final fue publicada en Abril del 2000; es una extensión de la norma compatible con las anteriores. Permite velocidades de hasta 480 Mbps, denominada de Alta velocidad ("High speed").

1.4 TOPOLOGÍA DEL BUS [4]

La forma física en la que los elementos se interconectan dentro del sistema USB, puede asemejarse a la topología estrella estratificada piramidalmente.

En un bus USB existen dos tipos de elementos: Anfitrión ("host") y dispositivos; a su vez, los dispositivos pueden ser de dos tipos: concentradores USB (“hubs”) y funciones USB (“periféricos”).

En la raíz o vértice de las capas, está el controlador anfitrión o host que controla todo el tráfico que circula por el bus.

En el centro de cada estrella se encuentra un hub USB, dispositivo que por un lado se conecta al computador o a otro hub USB y por otro lado, permite conectar al mismo varios dispositivos USB (“funciones”) o en su defecto nuevos hubs USB.

Esta disposición significa que los computadores con soporte para USB han de tener tan solo uno o dos conectores USB, pero ello no representa poder contar con tan solo dos dispositivos de esta clase, quien sabe un ratón y un teclado.

Muchos dispositivos USB han de traer conectores USB adicionales incorporados, por ejemplo un monitor puede tener 3 ó 4 conectores USB donde pueden ir el teclado, el ratón, y algún otro dispositivo. Por su parte el teclado puede tener otros más, y así sucesivamente hasta tener más de 127 dispositivos, todos funcionando simultáneamente.

A diferencia de otras arquitecturas, USB no es un bus de almacenamiento y envío, de forma que no se produce retardo en el envío de un paquete de datos hacia capas inferiores (Figura 1.4).

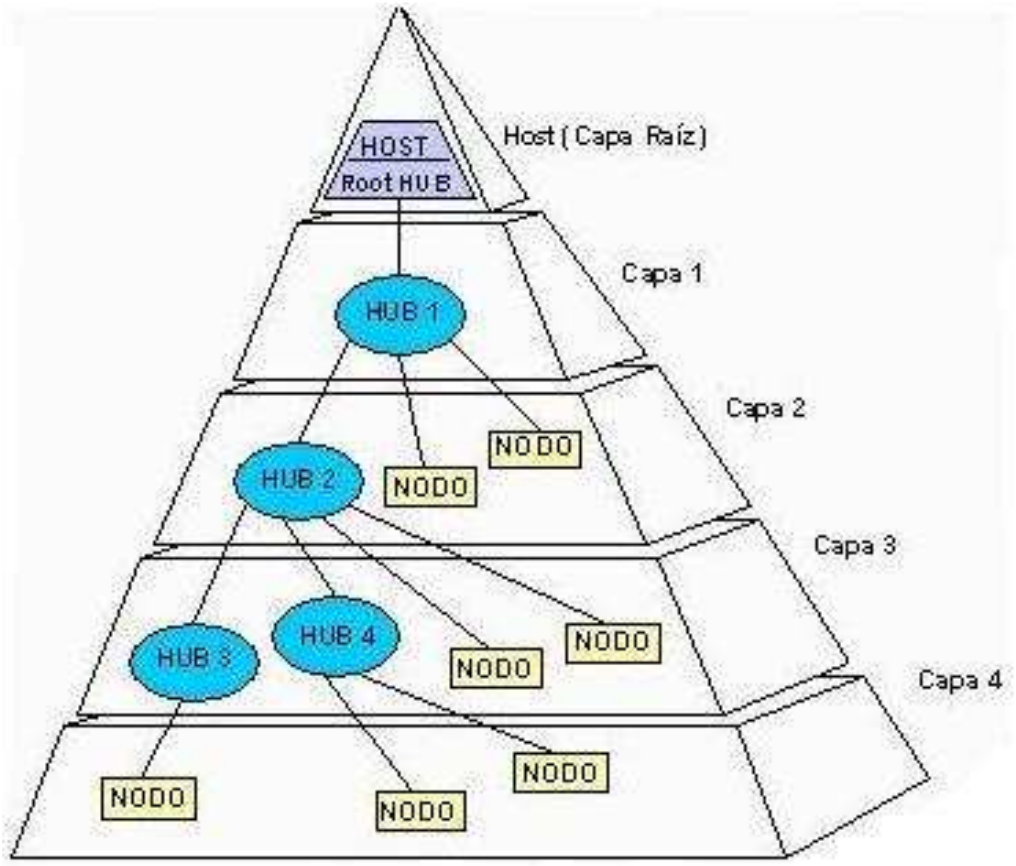


FIGURA 1.4. Estructura de capas del bus USB

1.4.1 ANFITRION (HOST) USB

Dentro de la terminología USB, el computador o la PC que soporta este tipo de bus, se denomina **Host USB**. A diferencia de los dispositivos y los hubs, existe tan solo un host dentro del sistema USB, que como ya dijimos es el computador mismo, particularmente una porción del mismo denominado Controlador USB del Host. Este tiene la misión de hacer de interfaz entre el computador mismo y los diferentes dispositivos. Existen algunas particularidades respecto a este controlador. Su implementación es una combinación de hardware y software todo en uno, es decir **Firmware**. Puede proveer de uno o dos puntos de conexión iniciales, denominados Hub raíz, a partir de los cuales y de forma ramificada irán conectándose los periféricos.

1.4.1.1 Host USB - Hardware y Software

El computador o Host USB trabaja con los diferentes dispositivos valiéndose del controlador de host compuesto por una parte de hardware y otra de software, de esta forma conjunta, el host es responsable al nivel de hardware, de los siguientes aspectos dentro del sistema USB:

- detectar tanto la conexión de nuevos dispositivos USB al sistema como la remoción de aquellos ya conectados, y por supuesto, configurarlos y ponerlos a disposición del usuario, tarea que involucra acciones por software.

- Administrar y controlar el flujo de datos entre el host y los dispositivos USB, es decir el movimiento de información generada por el mismo usuario.

- Administrar y regular los flujos de control entre el host y los dispositivos USB, es decir la información que se mueve con el objeto de mantener el orden dentro de los elementos del sistema.

- Recolectar y resumir estadísticas de actividad y estado de los elementos del sistema.

- Proveer de una cantidad limitada de energía eléctrica para aquellos dispositivos que pueden abastecerse con tan solo la energía proveniente del computador (teclado, ratón son dos ejemplos claros).

Por otra parte, a nivel de software las funciones del controlador de Host se incrementan y complican en los siguientes aspectos:

- Enumeración y configuración de los dispositivos conectados al sistema.

- Administración y control de transferencias isocrónicas de información.

- Administración y control de transferencias asincrónicas.

- Administración avanzada de suministro eléctrico a los diferentes dispositivos.

- Administración de la información del bus y los dispositivos USB.

1.4.2 DISPOSITIVOS USB [1]

Se consideran a los siguientes:

- Concentradores USB o Hubs, que proveen puntos adicionales al USB.
- Funciones USB o Periféricos, que proveen capacidades al sistema como una conexión ISDN, un joystick digital, parlantes, dispositivos dedicados.

1.4.2.1 Concentradores o Hubs [4]

Además del controlador, el PC también contiene el concentrador raíz. Este es el primer concentrador de toda la cadena que permite a los datos y a la energía pasar a uno o dos conectores USB del PC, y de allí a los 127 periféricos que, como máximo, puede soportar el sistema.
Los Hubs son elementos claves dentro del Sistema USB. Adicionalmente, simplifican de gran manera la sencillez de la interconexión de dispositivos al computador. Las Figuras1.5 y 1.6 muestran hubs USB disponibles en el mercado.

FIGURA 1.5. Hub USB

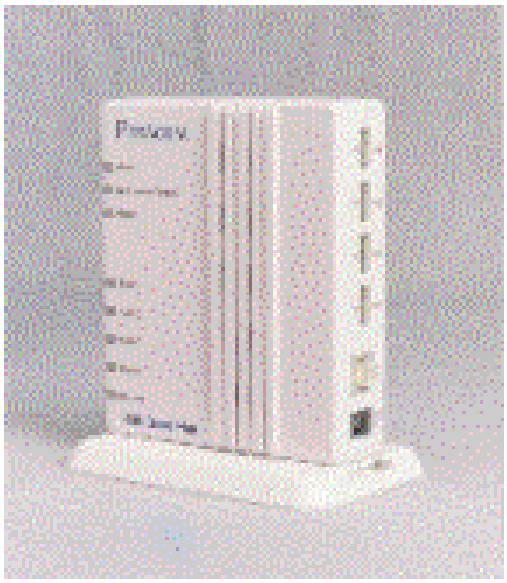


FIGURA 1.6. Hub USB

Bajo una óptica eléctrica e informática, los hubs son concentradores cableados que permiten múltiples conexiones simultáneas. Su aspecto más interesante es la concatenación, función por la que a un hub se le puede conectar otro y otro, ampliando la cantidad de puertos disponibles para periféricos

El hub USB tiene la capacidad de detectar si un periférico ha sido conectado a uno de sus puertos, notificando de inmediato al Controlador de Host en el computador, proceso que desata la configuración del equipo nuevo; adicionalmente, los hubs también son capaces de detectar la desconexión de un dispositivo, notificando al Controlador de Host que debe remover las estructuras de datos y programas de administración (drivers) del dispositivo retirado.

Otra de las funciones importantes de los hubs es la de aislar a los puertos de baja velocidad de las transferencias a alta velocidad, proceso sin el cual todos los dispositivos de baja velocidad conectados al bus entrarían en colapso. La protección de los dispositivos lentos de los rápidos ha sido siempre un problema serio dentro de las redes mixtas, como es USB.

El hub está compuesto por dos partes importantes: El Controlador del Hub y el Repetidor del Hub. El Repetidor del Hub tiene la función de analizar, corregir y

retransmitir la información que llega al hub, hacia los puertos del mismo. Mantiene una memoria consistente en varios registros de interfaz que le permiten sostener diálogos con el host y llevar adelante algunas funciones administrativas además de las meramente operativas; mientras que el Controlador de Hub puede asemejarse a una pequeña CPU de supervisión de las múltiples funciones que deben desempeñar un hub.

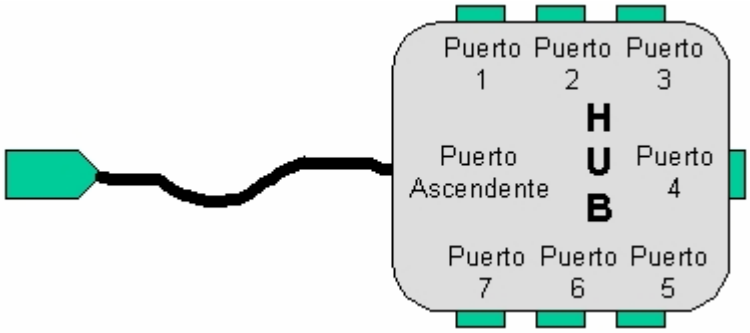


FIGURA 1.7. Esquema de un concentrador o Hub

1.4.2.1.1 Funcionamiento General del Hub USB

Las tarjetas madre de las computadoras “modernas” tienen normalmente dos conectores estandarizados que sirven para conectar dos dispositivos USB, pero para conectar hasta 127 dispositivos necesitamos utilizar HUBS (concentradores) USB con varios puertos, hasta llegar a totalizar como máximo 127 dispositivos, de modo que un dispositivo USB se puede conectar directamente al conector de la tarjeta madre o a un conector de HUB, sin variar para nada su funcionamiento. De hecho, algunos dispositivos pueden funcionar como hubs al tener conectores USB incorporados, como los teclados. También podemos conectar un dispositivo a un hub, que a su vez esté conectado a otro hub que está conectado al conector de la tarjeta madre y el funcionamiento del dispositivo será igual que estando conectado directamente al conector de la tarjeta madre. El cable de los dispositivos USB es un cable de 4 hilos con una longitud máxima de 5 metros por

dispositivo o HUB, con lo que los dispositivos conectados no tienen por qué estar amontonados encima de una mesa. [1]

1.4.2.2 Periféricos [6]

Dentro de la terminología USB, todos los dispositivos que pueden ser conectados a este bus, a excepción de los Hubs, se denominan Funciones. Son funciones típicas: el ratón, el monitor, altoparlantes, MODEM, etc.

Las funciones o dispositivos periféricos, son capaces de recibir y transmitir información, ya sea del usuario o de control. El común denominador de todas las funciones USB es su cable y el conector del mismo, diseñado y fabricado de acuerdo a las especificaciones del bus, por lo que no cabe preocuparse por la compatibilidad entre equipos de diferentes fabricantes; solamente hay que recordar las empresas que respaldan esta tecnología.

USB soporta periféricos de baja y media velocidad. Empleando dos velocidades para la transmisión de datos, de 1,5 y 12 Mbps con lo que se consigue una utilización más eficiente de sus recursos.

Los periféricos de baja velocidad tales como teclados, mouse, joystick, y otros periféricos para juegos, no requieren 12 Mbps. Empleando para ellos 1,5 Mbps, se puede dedicar más recursos del sistema a periféricos tales como monitores, impresoras, módems, scanner, equipos de audio, etc, que precisan de velocidades más altas para transmitir mayor volumen de datos o datos cuya dependencia temporal es más estricta.

En las figuras 1.8 y 1.10 se pueden ver cómo los hubs proporcionan conectividad a toda una serie de dispositivos periféricos.

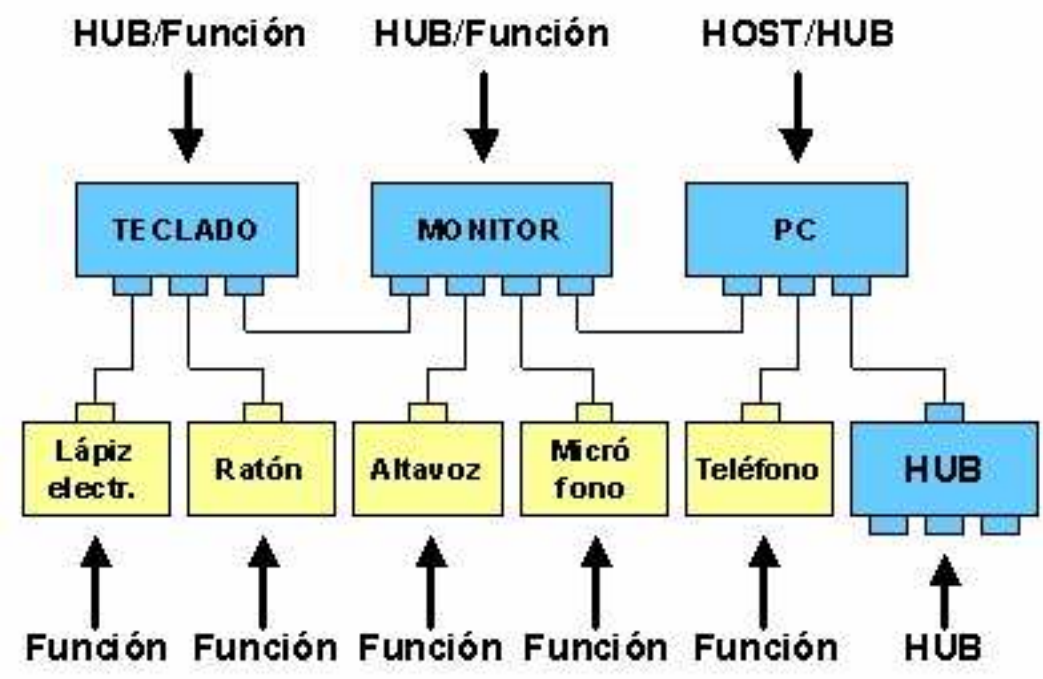


FIGURA 1.8. Posible esquema de conexiones del bus USB.

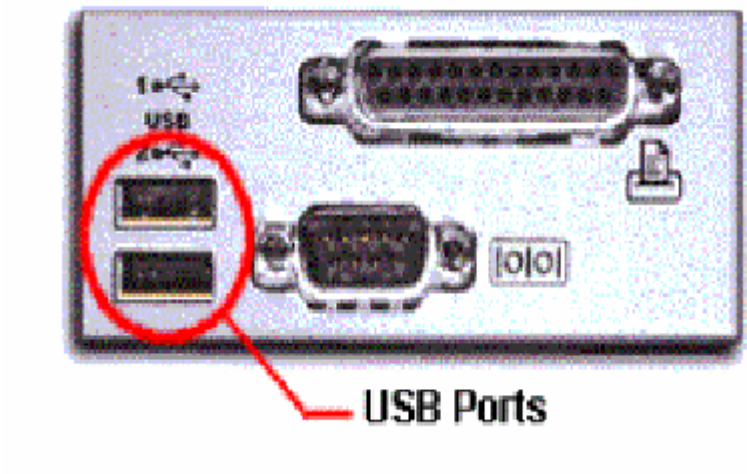


FIGURA 1.9. Puertos USB de una PC

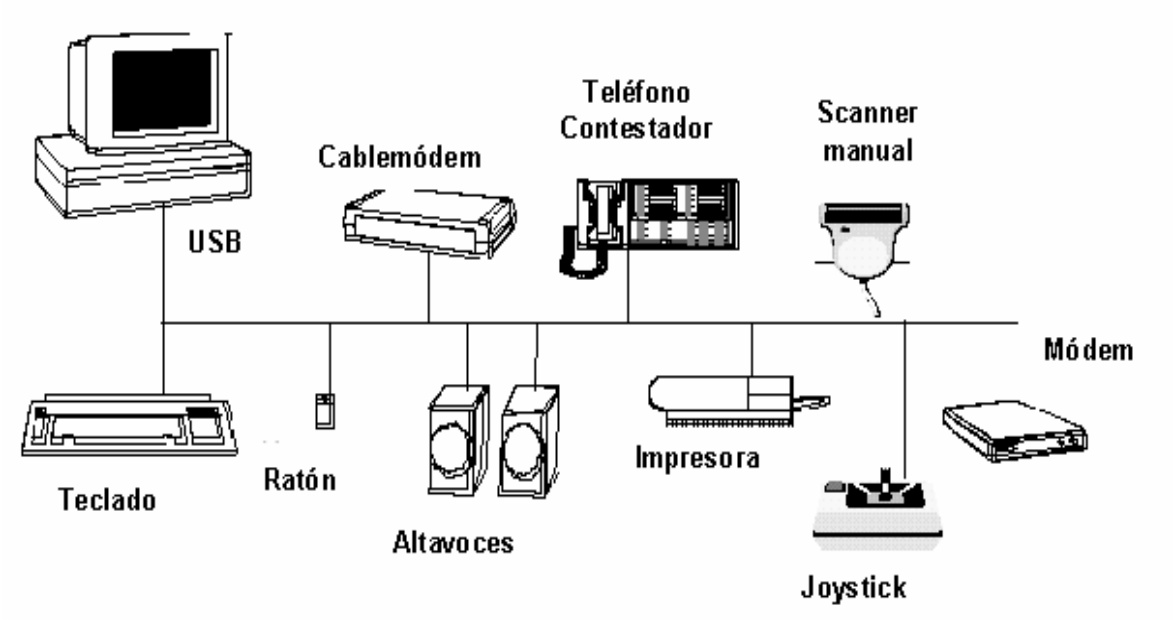


FIGURA 1.10. Dispositivos USB conectados a un PC [3]

1.5 FUNCIONAMIENTO [4]

Trabaja como interfaz para transmisión de datos y distribución de energía, que ha sido introducida en el mercado de PC´s y periféricos para mejorar las lentas interfaces serie (RS-232) y paralelo. Esta interfaz de 4 hilos, 12 Mbps y "plug and play", distribuye 5V para alimentación, transmite datos y está siendo adoptada rápidamente por la industria informática.

El protocolo de comunicación utilizado es de testigo, semejante a otros buses como los de las redes locales en anillo con paso de testigo y las redes FDDI (Fiber Distributed Data Interface). El controlador USB distribuye testigos por el bus. El dispositivo cuya dirección coincide con la que porta el testigo responde aceptando o enviando datos al controlador. Este también gestiona la distribución de energía a los periféricos que lo requieran.

El bus serie USB es síncrono, y utiliza el algoritmo de codificación NRZI ("Non Return to Zero Inverted"). En este sistema existen dos voltajes opuestos; una

tensión de referencia corresponde a un "1", pero no hay retorno a cero entre bits, de forma que una serie de unos corresponde a un voltaje uniforme; en cambio los ceros se marcan como cambios del nivel de tensión, de modo que una sucesión de ceros produce sucesivos cambios de tensión entre los conductores de señal.

El controlador USB instalado en el ordenador, denominado controlador de host, o concentrador raíz ("Root hub"), proporcionan un enlace entre el bus de la placa-base, por ejemplo PCI, y una o más conexiones iniciales con el exterior (generalmente 2 conectores del tipo "A").

A partir de estas y utilizando concentradores adicionales, pueden conectarse más dispositivos.

Actualmente la mayoría de las placas-base incluyen un controlador USB integrado en el chipset. Para sistemas antiguos que no dispongan de USB pueden instalarse tarjetas PCI (e incluso PC-CARD para portátiles) que incluyen un controlador y uno o dos conectores de salida.

Puesto que todos los periféricos comparten el bus y pueden funcionar de forma simultánea, la información es enviada en paquetes; cada paquete contiene una cabecera que indica el periférico a que va dirigido. Existen cuatro tipos de paquetes distintos: Token, Datos, Handshake, y Especial.

El máximo de datos por paquete es de 8, 16, 32 y 64 Bytes.

Se utiliza un sistema de detección y corrección de errores bastante robusto tipo CRC ("Cyclical Redundancy Check").

El funcionamiento está centrado en el host, todas las transacciones se originan en él; es el controlador host el que decide todas las acciones, incluyendo el número asignado a cada dispositivo (esta asignación es realizada automáticamente por el controlador "host" cada vez que se inicia el sistema, se añade, o elimina, un nuevo dispositivo en el bus), su ancho de banda, etc. Cuando se detecta un nuevo dispositivo es el host el encargado de cargar los drivers oportunos sin necesidad de intervención por el usuario.

El sistema utiliza varios tipos de transmisiones que resuelven todas las posibles situaciones de comunicación. Cada transmisión utiliza un mínimo de tres paquetes, el primero es siempre un Token que avisa al dispositivo que puede iniciar la transmisión.

1.5.1 TRANSMISIÓN ASINCRÓNICA

Las distintas formas de transmisión de datos a distancia siempre fueron seriales, ya que el desfase de tiempos ocasionada por la transmisión paralela en distancias grandes impide pensar en esta última como apta para cubrir longitudes mayores a algunos pocos metros.

Sobre ello, la transmisión serial ha topado con el problema de que la información generada en el transmisor sea recuperada en la misma forma en el receptor, para lo cual es necesario ajustar adecuadamente un sincronismo entre ambos extremos de la comunicación. Para ello, tanto el receptor como el transmisor deben disponer de relojes que funcionen a la misma frecuencia y posibilite una transmisión exitosa. Como respuesta a este problema surgió la transmisión asincrónica, empleada masivamente años atrás para la comunicación entre los equipos servidores conocidos como hosts y sus terminales.

En este modelo cabe entender que ambos equipos poseen relojes funcionando a la misma frecuencia, por lo cual, cuando uno de ellos desea transmitir, prepara un grupo de bits encabezados por un BIT conocido como de arranque, un conjunto de 7 u 8 bits de datos, un BIT de paridad (para control de errores), y uno o dos bits de parada. El primero de los bits enviados anuncia al receptor la llegada de los siguientes, y la recepción de los mismos es efectuada. El receptor conocerá perfectamente cuántos bits le llegarán, y da por recibida la información cuando verifica la llegada de los bits de parada. El esquema de los datos se muestra en la Figura 1.11.

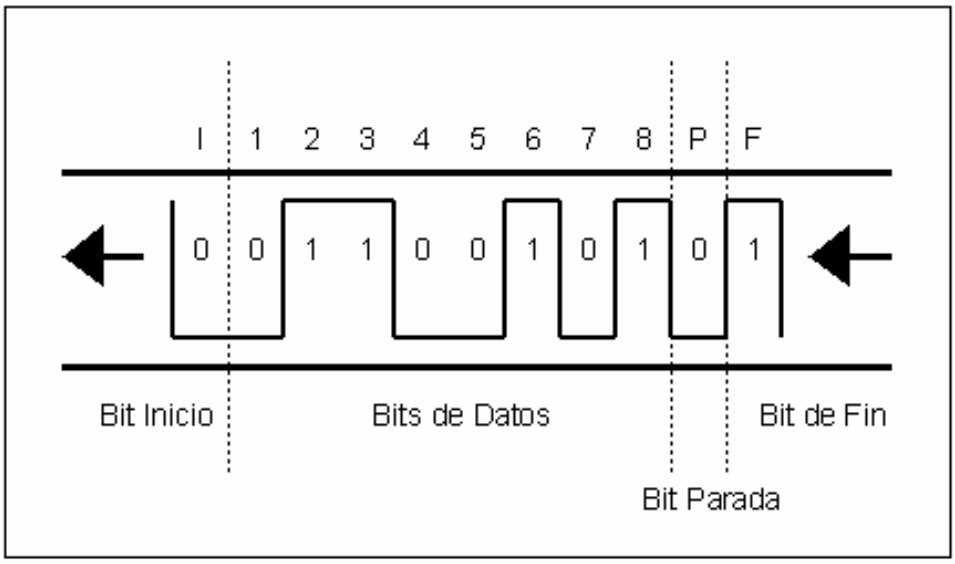


FIGURA 1.11. Esquema de la transmisión asincrónica

Se denomina transmisión asincrónica no porque no exista ningún tipo de sincronismo, sino porque el sincronismo no se halla en la señal misma, más bien son los equipos mismos los que poseen relojes o clocks que posibilitan la sincronización. La sincronía o asincronía siempre se comprende a partir de la señal, no de los equipos de transmisión o recepción.

1.5.2 TRANSMISIÓN SINCRÓNICA

En este tipo de transmisión, el sincronismo viaja en la misma señal, de esta forma la transmisión puede alcanzar distancias mucho mayores como también un mejor aprovechamiento de canal. En la transmisión asincrónica, los grupos de datos están compuestos por generalmente 10 bits, de los cuales 4 son de control. Evidentemente el rendimiento no es el mejor. En cambio, en la transmisión sincrónica, los grupos de datos o paquetes están compuestos por 128 bytes, 1024 bytes o más, dependiendo de la calidad del canal de comunicaciones.

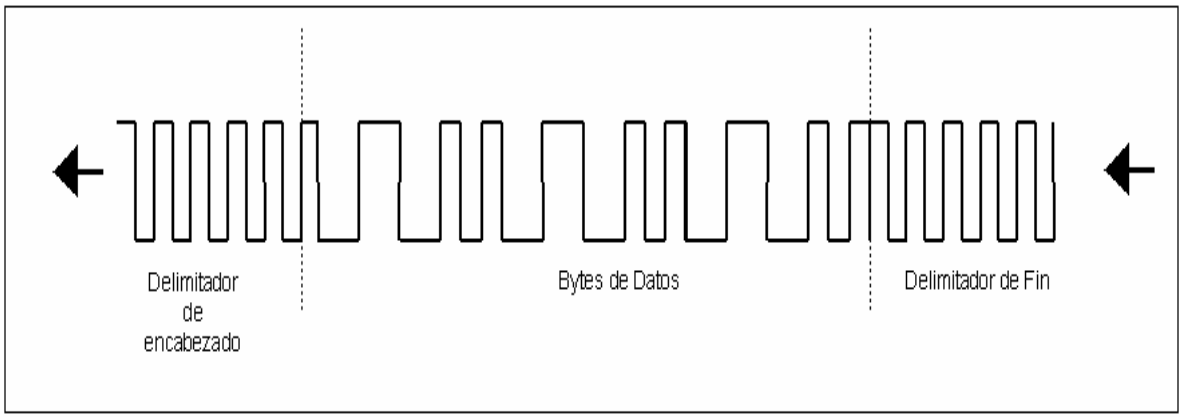


Figura 1.12. Esquema de la transmisión sincrónica

Las transmisiones sincrónicas ocupan en la actualidad gran parte del mundo de las comunicaciones seriales, especialmente las que emplean el canal telefónico.

1.5.3 TRANSMISIÓN ISOCRÓNICA

Inicialmente vale la pena aclarar el origen de este término tan extraño, ISO (algún) CRONOS (tiempo). La transmisión isocrónica ha sido desarrollada especialmente para satisfacer las demandas de la transmisión multimedia por redes, esto es integrar dentro de una misma transmisión, información de voz, video, texto e imágenes. La transmisión isocrónica es una forma de transmisión de datos en la cual los caracteres individuales están solamente separados por un número entero de intervalos, medidos a partir de la duración de los bits. Contrasta con la transmisión asincrónica en la cual los caracteres pueden estar separados por intervalos aleatorios. La transferencia isocrónica provee comunicación continua y periódica entre el host y el dispositivo, con el fin de mover información relevante a un cierto momento. La transmisión isocrónica se encarga de mover información relevante a algún tipo de transmisión, particularmente audio y video.

1.5.4 TRANSMISIÓN BULK

La transmisión Bulk (“Transmisión de pila de datos”) es una comunicación no periódica, explosiva típicamente empleada por transferencias que requieren usar todo el ancho de banda disponible o en su defecto son demoradas hasta que el ancho de banda completo esté disponible. Esto implica particularmente

movimientos de imágenes o video, donde se requiere de gran potencial de transferencia en poco tiempo.

1.5.5 TRANSMISIONES DE CONTROL

Es un tipo de comunicación exclusivamente entre el host y el dispositivo que permite configurar este último, sus paquetes de datos son de 8, 16, 32 o 64 bytes, dependiendo de la velocidad del dispositivo que se pretende controlar. Ocurre cuando un dispositivo se conecta por primera vez. En este momento el controlador de host envía un paquete "Token" al periférico notificándole el número que le ha asignado.

1.5.6 TRANSMISIONES DE INTERRUPCIÓN

Este tipo de comunicación está disponible para aquellos dispositivos que demandan mover muy poca información y poco frecuentemente. Tiene la particularidad de ser unidireccional, es decir del dispositivo al host, notificando de algún evento o solicitando alguna información. Su paquete de datos tiene las mismas dimensiones que el de las transmisiones de control.

1.6 CABLES Y CONECTORES [9]

1.6.1 CABLES

El cable de bus USB es de 4 hilos, y comprende 2 líneas de señal (datos) y 2 de alimentación, con lo que las funciones pueden utilizar un único cable.
Existen dos tipos de cable: apantallado para transmisiones a 12 Mbps y sin apantallar para transmisiones a 1,5 Mbps. En el primer caso el par de hilos de señal es trenzado; los de tierra y alimentación son rectos, y la cubierta de protección (pantalla) solo puede conectarse a tierra en el anfitrión. En el cable sin apantallar todos los hilos son rectos. Las conexiones para transmisiones a 15 Mbps y superiores exigen cable apantallado.

Se utilizan diámetros estándar para los hilos del bus. Para cada sección se autoriza una longitud máxima del segmento. En las siguientes tablas se muestran estas distancias y la disposición de pines y colores de identificación.

TABLA 1.1. Diámetros de los conductores

AWG	mm Ø	long. máx.
28	0.321	0.81 m
26	0.405	1.31 m
24	0.511	2.08 m
22	0.644	3.33 m
20	0.812	5.00 m

TABLA 1.2. Asignación de terminales

Pin	Nombre	Descripción	Color
1	VBUS	+ 5 V. CC	rojo
2	D-	Data -	azul
3	D+	Data +	amarillo
4	GND	Tierra	verde

El calibre de los conductores destinados a alimentación (VBUS y GND) de los periféricos varía desde 20 a 26 AWG, mientras que el de los conductores de señal (D+ y D-) es de 28 AWG. La longitud máxima de los cables es de 5 metros.

En la figura 1.13 se muestra un esquema del cable, con dos conductores para alimentación y los otros dos para señal, debiendo estos últimos ser trenzados o no según la velocidad de transmisión.

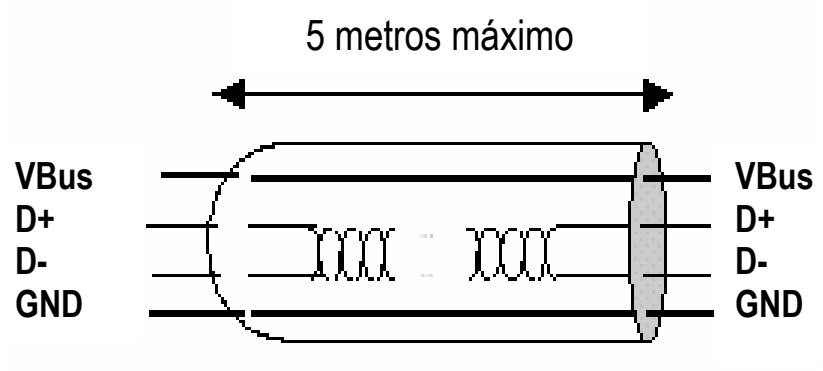


Figura 1.13. Esquema del cable para USB.

Es importante indicar que los cables USB tienen protectores de voltaje a fin de evitar cualquier daño a los equipos, son estos mismos protectores los que permiten detectar un dispositivo nuevo conectado al sistema y su velocidad de trabajo.

1.6.1.1 Características de los cables USB

Los cables USB deben cumplir con las siguientes características:

- El cable debe terminar en el un terminal con un molde de Plug Series A, y en el terminal opuesto con un molde de Plug Series B.
- El cable debe ser fabricado como Full Speed.
- La impedancia del cable debe reconocer la impedancia de drivers Full Speed.
- La máxima distancia del cable es determinado por la señal del par de atenuación, hasta un máximo de 5 metros.
- Se deben minimizar el retardo de propagación entre las dos señales del conductor.
- GND provee la tierra común entre el host y el dispositivo USB.
- El VBUS provee energía al dispositivo conectado.

1.6.2 CONECTORES [9]

En lo que respecta a los conectores hay que decir que son del tipo ficha (o conector) y receptáculo, y son de dos tipos: serie A y serie B. Ambos son polarizados (solo pueden insertarse en una posición) y utilizan sistemas de presión para sujetarse. Los de tipo **A** utilizan la hembra en el sistema anfitrión, y suelen usarse en dispositivos en los que la conexión es permanente (por ejemplo, ratones y teclados). Los de tipo **B** utilizan la hembra en el dispositivo USB (función), y se utilizan en sistemas móviles (por ejemplo, cámaras fotográficas, impresoras, scanner, módems o altavoces). En general podemos afirmar que la hembra de los conectores **A** están en el lado del host (PC) o de los concentradores (hubs), mientras que las de tipo **B** están del lado de los periféricos.

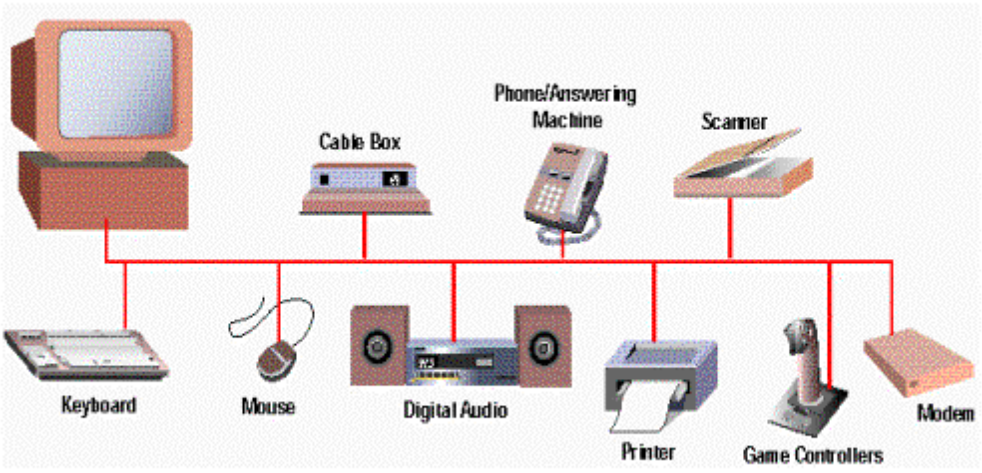


FIGURA 1.14. Utilización de los conectores tipo B

Cada conector tiene 4 cables (V+, V-, D+, D-) para proveer de energía a los dispositivos de bajo consumo.

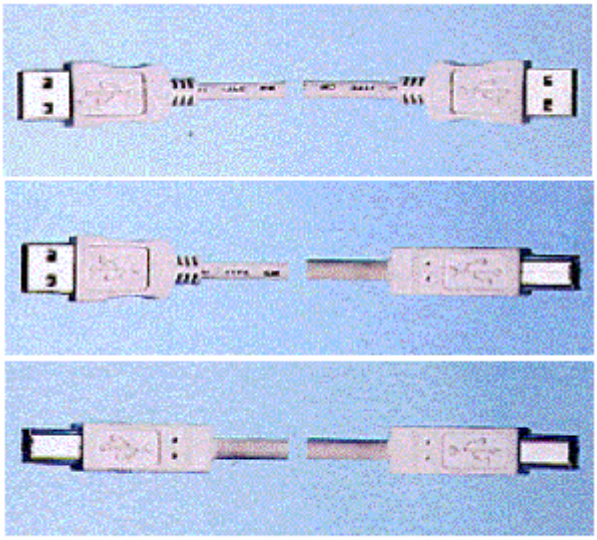


FIGURA 1.15. Conectores: Serie A – Serie A. Serie A – Serie B. Serie B - Serie B

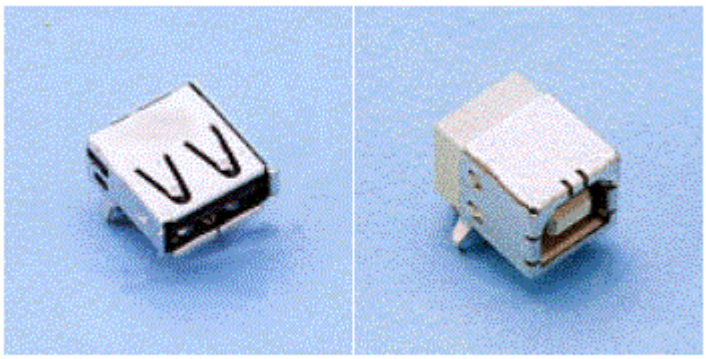


FIGURA 1.16. Ranuras Serie A y Serie B

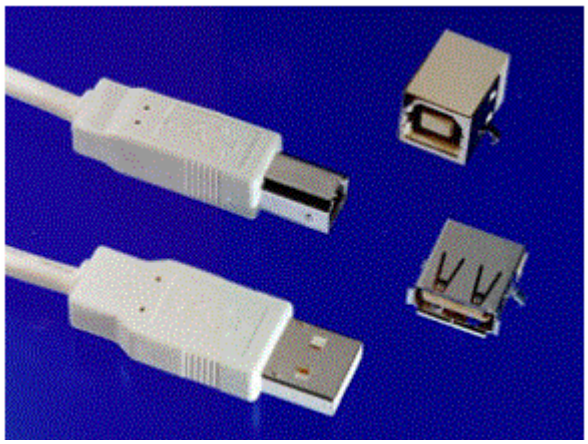


FIGURA 1.17. Conectores USB tipo a y tipo B

La descripción de medidas de construcción que utilizan las empresas dedicadas a la fabricación de dispositivos USB, para los conectores USB se muestra a continuación en las figuras 1.18 y 1.19.

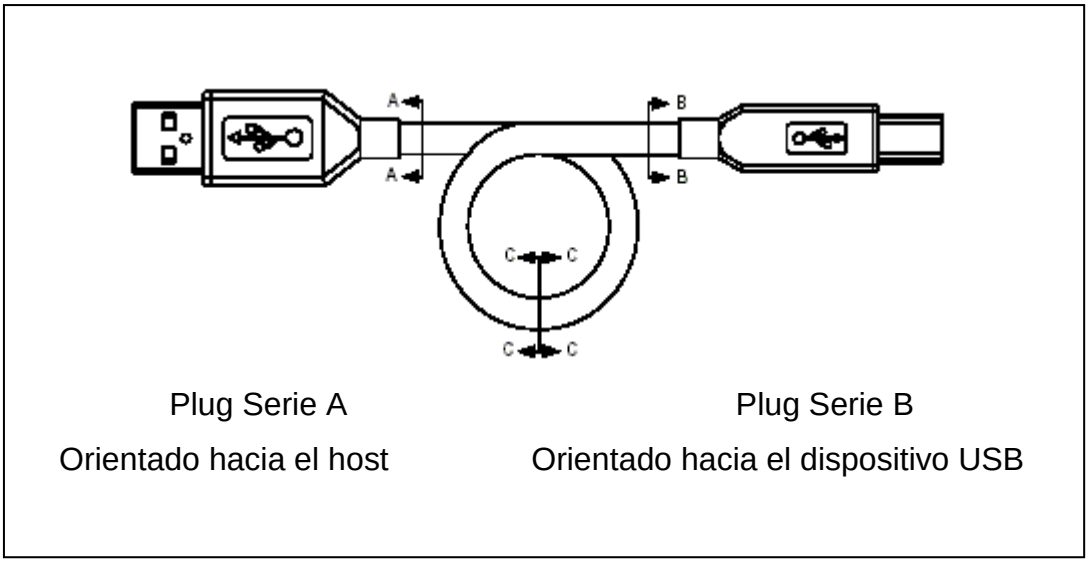


FIGURA 1.18. Orientación de los conectores

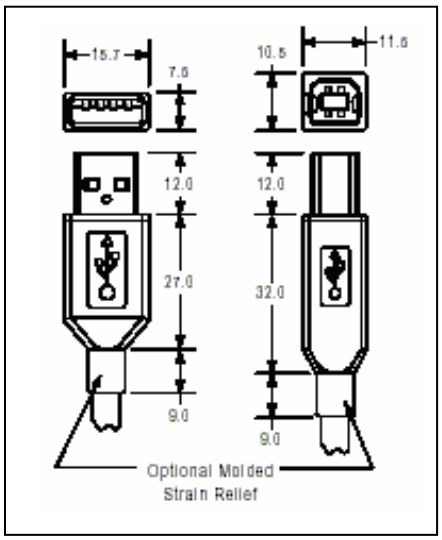


FIGURA 1.19. Medidas de construcción del cable USB

1.6.3 APLICACIONES DE LOS CABLES Y CONECTORES USB [4]

Tenemos aplicaciones actuales y posibilidades a futuro:

- Discos duros de estado sólido portátiles.
- Adaptadores de video para monitores de PC.
- Grabadores de audio y video sobre bus USB.
- Conexiones de PC a PC a través de puertos USB.
- Sustitución de los puertos serie y paralelo.

En las placas que se venden actualmente, especialmente si son en formato ATX, el conector del bus USB está presente como un estándar, a veces hasta por duplicado. Como se aprecia en la figura, es un pequeño rectángulo, del tamaño aproximado de una clavija telefónica (pero distinta de éstas).

1.7 IDENTIFICACIÓN Y DIAGNÓSTICO [5]

Windows dispone de un programa específico para ver los puertos USB reconocidos y sus parámetros de configuración, es el programa Usbview.exe, que puede encontrarse en el CD de instalación de Windows 98, en el directorio Tools\Reskit\Diagnos.

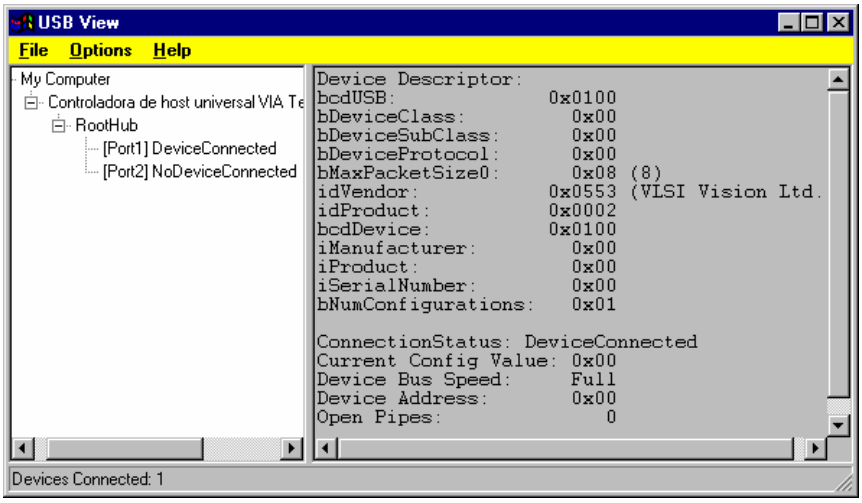


FIGURA 1.20. Resultado de conectar una cámara de video-conferencia en el puerto Nº 1

Por su parte, el foro de desarrolladores USB (USB-IF), ha publicado diversas herramientas para la comprobación de las características de los dispositivos USB, entre las que se encuentran las siguientes:

USBCV ("USB Command Verifier"). Es una herramienta de diagnóstico y verificación para comprobar que dispositivos USB de alta y baja velocidad se ajustan a las normas.

USB Check. Esta herramienta permite verificar dos enlaces; uno para comprobar dispositivos de alta velocidad funcionando a alta velocidad, y otro para comprobar dispositivos de velocidad completa y baja; también dispositivos de alta velocidad funcionando a velocidad completa.

USBHTT ("USB2 Hub Transaction Translator Test Suite"). Es una herramienta de verificación para concentradores USB 2.0.

USBHSET ("USB High Speed Electrical Test Tool Kit"). Este sistema de prueba contiene software y procedimientos diseñados para verificar diversos parámetros eléctricos, incluyendo la calidad de señal en dispositivos USB de alta velocidad. Incluye también procedimientos detallados para comprobación de controladores host, concentradores y funciones USB de alta velocidad. Este software también permite comprobar la calidad de señal en dispositivos de velocidad completa y baja, así como verificaciones de suministro energético.

Estas herramientas están disponibles para su descarga en la página web: www.usb.org/developers/tools.html#usbhset

1.8 DIAGRAMA DE CAPAS DEL USB [6]

En el diagrama de capas de la figura 1.21 se observa cómo fluye la información entre las diferentes capas a nivel real y a nivel lógico.

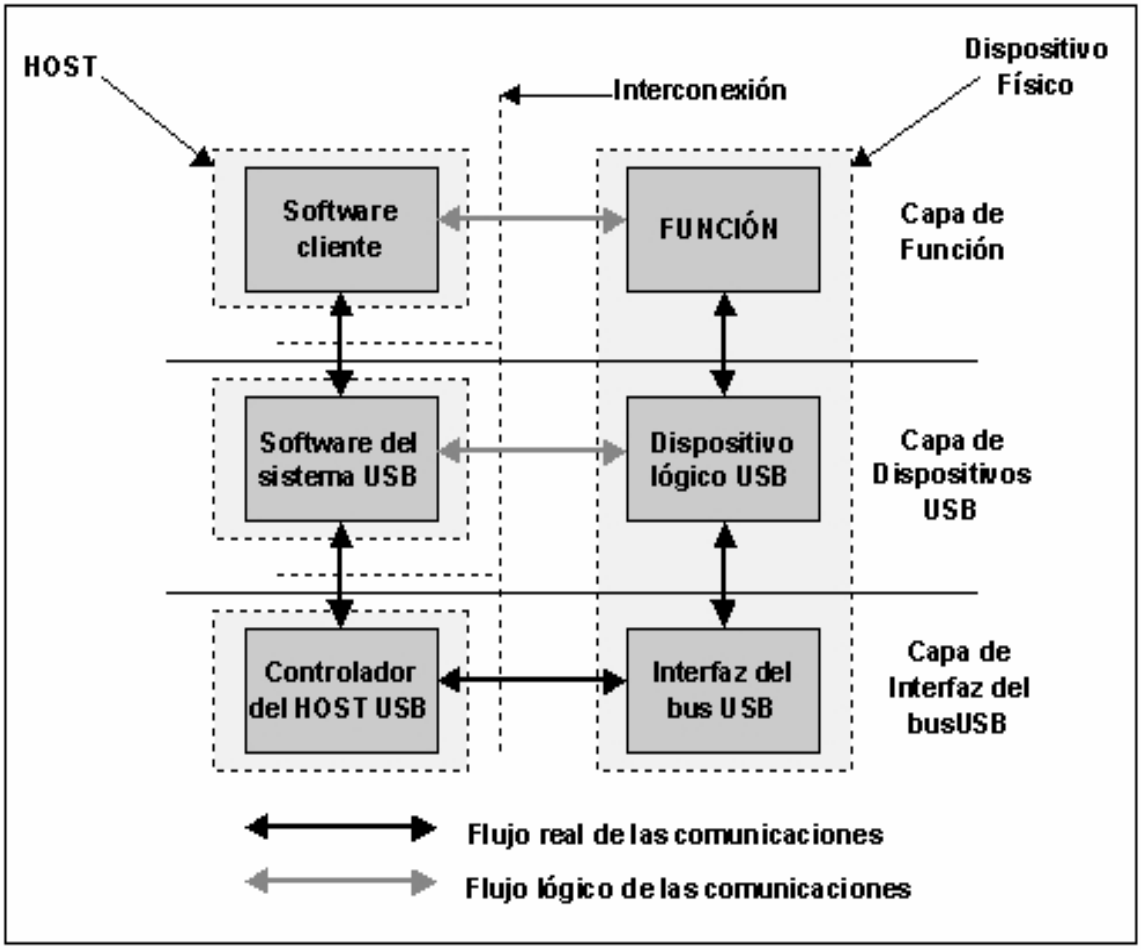


FIGURA 1.21. Capas del sistema de comunicación USB

El diagrama de la Figura anterior, ilustra el flujo de datos USB a partir de tres niveles lógicos: entre el Software Cliente y la Función, el Controlador USB y el dispositivo, y finalmente la capa física, donde la transmisión realmente sucede. Es importante entender que este modelo es muy parecido al OSI, el estándar de redes, y su comprensión radica en el hecho de que si bien existe un solo canal físico, pero los datos son manejados en cada punto por unidades homólogas o idénticas, tal como si estuviesen sosteniendo una comunicación directa. Por esta razón se las denomina Capas Lógicas.

El nivel superior lógico es el agente de transporte de datos que mueve la información entre el Software Cliente y el dispositivo. Existe un Software Cliente en el host, y un Software de Atención al mismo en cada una de las funciones o periféricos USB. A este nivel, el host se comunica con cada uno de los periféricos en alguna de las varias formas posibles de transmisión que soporta USB. El Software Cliente solicita a los dispositivos diversas tareas y recibe respuestas de ellos a través de esta capa.

La capa lógica intermedia es administrada por el Software de Sistema USB, y tiene la función de facilitarles las tareas particulares de comunicación a la capa superior, cabe decir, administra la parte del periférico con la que la capa superior desea comunicarse, maneja la información de control y comando del dispositivo, etc. Su objetivo es permitir a la capa superior concentrarse en las tareas específicas tendientes a satisfacer las necesidades del usuario, adicionalmente gestiona el control interno de los periféricos.

El acceso al bus es bajo la modalidad de Ficha o Token, lo que involucra siempre complejidad de protocolos, especialmente si agregamos dos velocidades posibles: 12Mbps ó 1.5Mbps. Todos estos algoritmos y procesos son administrados por el Host USB, reduciendo la complejidad del periférico, y lo más importante, el costo final de los dispositivos USB.

La capa física del modelo lógico USB comprende los puertos físicos, el cable, los voltajes y señales, el hardware y funcionamiento del hardware. Esta capa tiene el objetivo de liberar a las capas superiores de todos los problemas relacionados a la modulación, voltajes de transmisión, saltos de fase, frecuencias y características netamente físicas de la transmisión.

1.9 BENEFICIOS Y LIMITACIONES

1.9.1 BENEFICIOS DEL USB [4]

El trabajo involucrado dentro de la especificación USB es realmente completo, es un estudio realmente minucioso, que comprende aspecto tales como:

- Arquitectura del Bus.
- Definiciones de protocolos.
- Tipos de transacciones.
- Administración del bus.
- Señales eléctricas.
- Especificaciones electrónicas.
- Conectores.
- Formas de transmisión.
- Etc.

Pero todo esto se puede traducir en beneficios tangibles para el usuario, como los siguientes:

- Fácil expansión de periféricos en la PC, no debe hacer falta, más que conectar el periférico y emplearlo (sin abrir la computadora).
- Bajo costo para aplicaciones que demandan velocidades por los 12 Mbps, particularmente aplicaciones multimediales: micrófonos, parlantes, teléfonos, etc.
- Soporte completo para transmisión en tiempo real de voz, audio, y video.
- Flexibilidad de protocolos para transmisiones mixtas isocrónicas y asincrónicas (las cuales serán analizadas más adelante, ya que es el eje de transmisión de USB).
- Cómoda integración de dispositivos de tecnología y fabricantes diferentes.
- Soporte para plataformas diversas de la línea de las PC's compatibles (como ya se vio, algunos problemas para MACINTOSH)

- Posibilitar la producción de nuevos dispositivos capaces de aprovechar sus ventajas.

1.9.2 LIMITACIONES O DESVENTAJAS DEL USB [9]

- El ancho de banda debe repartirse entre los dispositivos, lo que no importa mucho si estamos conectando otro ratón, pero que nos indica que conectar 126 impresoras al mismo puerto USB e intentar imprimir en todas a la vez no es una buena idea. Sin embargo, parece un ancho suficiente para utilizar algunos dispositivos portátiles como las unidades Zip, mientras no intentemos usarlos a la vez que una impresora, un módem y un escáner USB (combinación ciertamente improbable). Por este motivo no puede tampoco manejar el ancho de banda requerido para video en vivo no comprimido, dado que un dispositivo USB no puede tener un ancho de banda de más de 6 Mbits/sec.

- Necesita de un PC que coordine su actividad a través de un controlador, pero esto lo hace de fácil fabricación y económico de implementar.

- El puerto USB suministra solamente 500 miliamperios de electricidad para los dispositivos conectados, que aunque es suficiente potencia para la mayoría de los dispositivos que se conectan a este puerto, resulta escaso cuando conectamos varios dispositivos sin fuente de alimentación propia. Lo que sí podemos hacer es comprar un HUB USB con toma de alimentación eléctrica, para proporcionar la potencia necesaria a aquellos dispositivos que lo requieran (especialmente escáneres e impresoras).

CAPITULO 2.

2. MICROCONTROLADOR 16F628

2.1 DESCRIPCIÓN [2]

Un microcontrolador es un circuito integrado, en cuyo interior posee toda la arquitectura de un computador, CPU, memorias RAM, EEPROM, y circuitos de entrada y salida.

Un microcontrolador de fábrica, no realiza tarea alguna, este debe ser programado para que realice desde un simple parpadeo de un led hasta una sofisticada automatización de una fábrica.

Un microcontrolador es capaz de realizar la tarea de muchos circuitos lógicos como compuertas AND, OR, NOT, AND, conversores A/D, D/A, temporizadores, decodificadores, etc, simplificando todo el diseño de una placa de reducido tamaño y pocos elementos.

Los microcontroladores PIC (Peripheral interfase Controller) son fabricados por la empresa MICROCHIP technology INC. Cuya central se encuentra en Chandler, Arizona, esta empresa se mantiene a la cabeza frente a los demás competidores debido a la gran variedad, gran velocidad, bajos costos, bajo consumo de potencia, y gran disponibilidad de herramientas para su programación.

Uno de los microcontroladores más populares en la actualidad es el PIC16F628, soporta 1000 ciclos de escritura en su memoria FLASH, y 1’000.000 ciclos en su memoria Eeprom, este está reemplazando rápidamente al popular PIC16F84A, pues presenta grandes ventajas frente a este último como son:

TABLA 2.1. Comparación entre el PIC16F84A y el PIC16F628

	PIC 16F84A	**PIC 16F628**
Memoria de programa	1024	2048
Memoria datos EEPROM	64	128
Memoria RAM	68	224
Pines de entrada/salida	13	16
Comparadores	---	2

Todas estas y otras ventajas más como oscilador interno RC de 4 MHz, MCLR programable, mayor resistencia, comunicación AUSART, etc. Lo hacen al PIC16F62X, como el microcontrolador ideal para estudiantes y aficionados, ya que al tener oscilador interno y el MCLR (Master Clear) sea opcional, es mucho más sencillo ponerlo en funcionamiento, basta con conectar al pin 14 a 5V y el pin 5 a tierra para que empiece a trabajar.

- El voltaje de alimentación del PIC 16F62X es desde 3V hasta 5.5V como máximo.
- Sus dos puertos el A y el B entregan un total de 200mA cada uno, es decir 25mA cada pin.
- En modo sumidero pueden soportar cada uno de sus puertos 200mA, es decir 25mA cada pin.

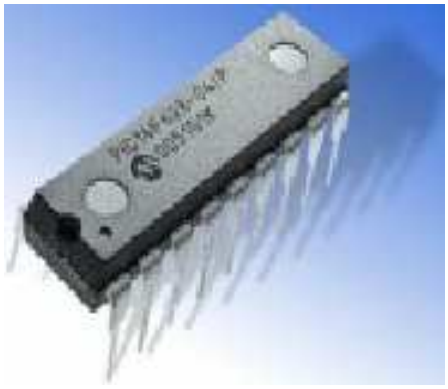


FIGURA 2.1. Presentación del PIC16F628

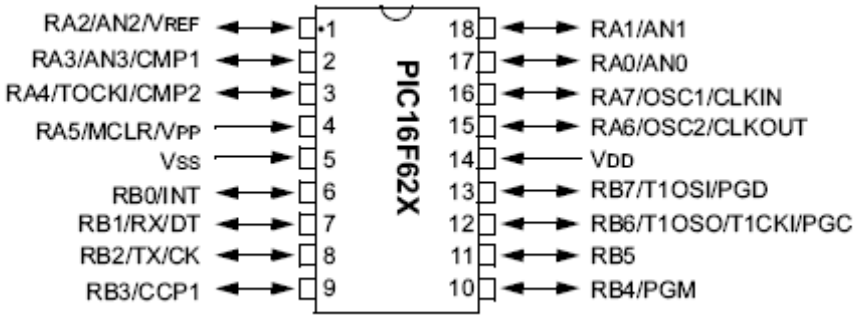


FIGURA 2.2. Diagrama de pines

2.2 FUNCIONAMIENTO [7]

El PIC16F628 de Microchip es un potente microcontrolador CMOS FLASH de 8 bits de arquitectura RISC capaz de operar con frecuencias de reloj hasta de 20 MHz (ciclos de instrucción de apenas 200 ns), fácil de programar y disponible en cápsulas DIP y SOIC de 18 pines**.**

Posee internamente un oscilador de 4 MHz y un circuito Power-On Reset (P.O.R) que eliminan la necesidad de componentes externos y expanden a 16 el número de pines que pueden ser utilizados como líneas I/O de propósito general.

Adicionalmente, el PIC16F628 proporciona una memoria de datos EEPROM de 128x8 , una memoria de programa FLASH de 2024x14, una memoria de datos RAM de propósito general de 224x8, un módulo CCP (captura/comparación/PWM), dos comparadores análogos, una referencia de voltaje programable y tres temporizadores. Estas y otras características lo hacen ideal en aplicaciones automotrices, industriales, y de electrónica de consumo, así como en equipos e instrumentos programables de todo tipo.

La distribución de pines del PIC16F628 es idéntica a la del PIC16F627, excepto que este último posee una memoria de programa FLASH de 1024x14. También es idéntica a la del PIC16F84, excepto que en el PIC16F628 se puede disponer de tres líneas I/O adicionales para el puerto A (RA7, RA6, RA5) y algunos pines

I/O están multiplexados con una función alterna para los diversos dispositivos periféricos que soporta el chip; por ejemplo, RB1 funciona también como la línea de recepción. En general, cuando un periférico está habilitado, esa línea no puede ser utilizada como un pin I/O de propósito general.

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. Los vectores de reset e interrupción están en las direcciones 0000h y 0004h, respectivamente.

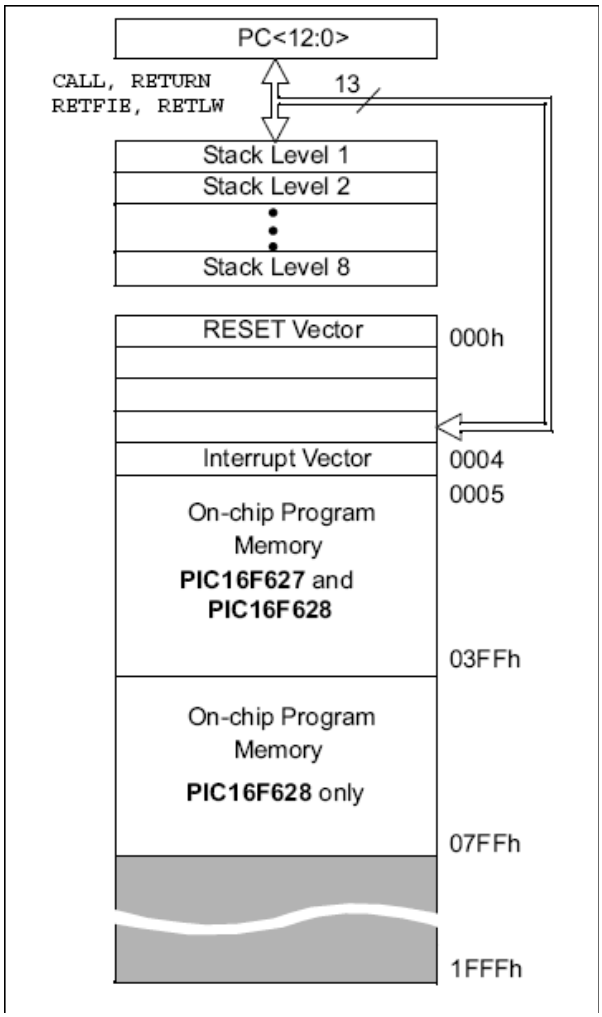


Figura 2.3. Organización de la memoria de programa Flash

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. Sin embargo, sólo están implementados 330 bytes, correspondiendo 224 al área de los registros de propósito general (GPR) y 36 al área de los registros de función especial (SFR).

Los restantes 70 bytes implementados son espejos de algunos SFR de uso frecuente, así como de los últimos 16 GPR del banco 0. Por ejemplo, las posiciones 0Bh, 8Bh, 10Bh 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. Ver figura 2.4.

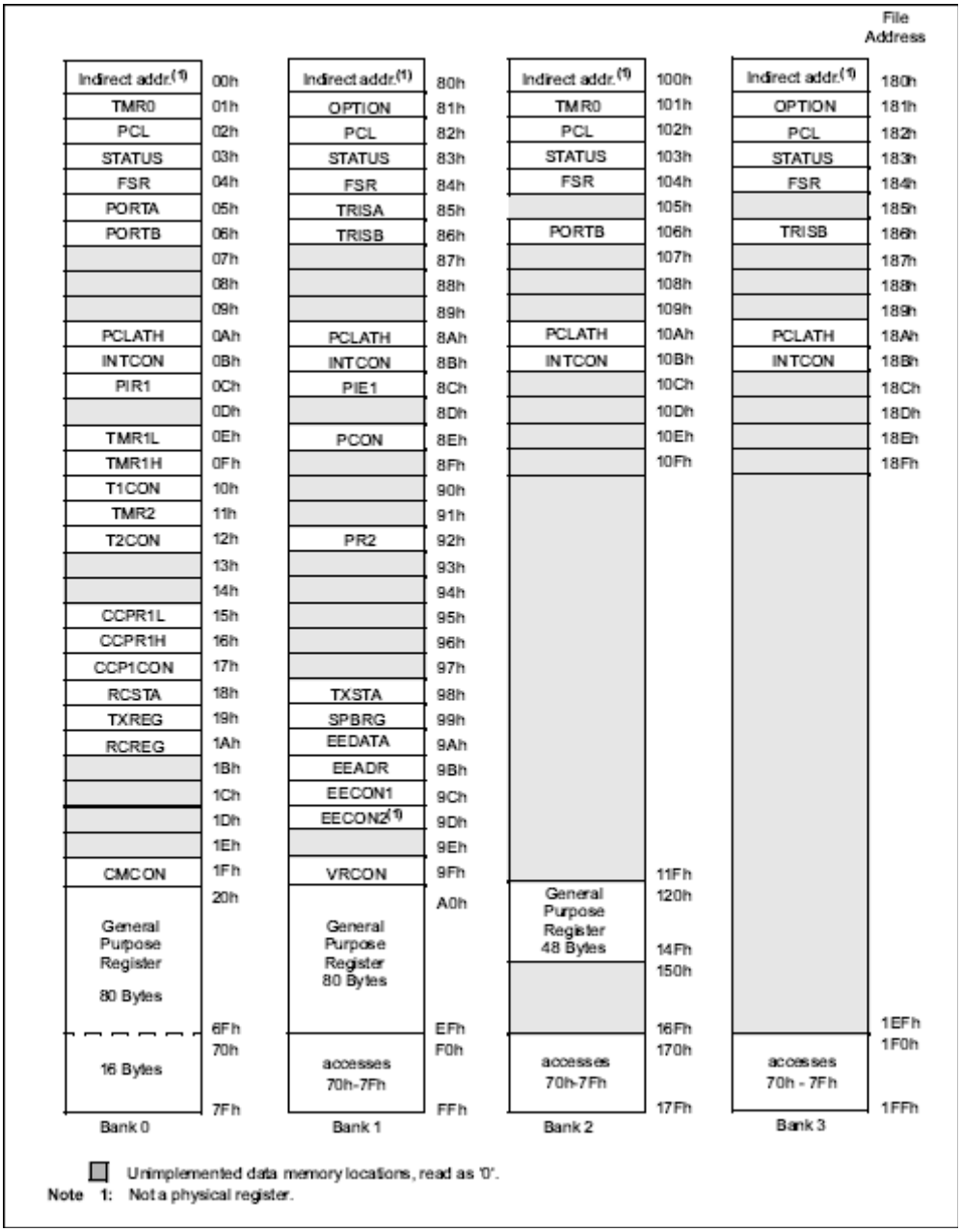


Figura 2.4. Organización de la memoria de datos

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, figura 2.5.

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

Figura 2.5. Selección de bancos

2.3 LENGUAJE DE PROGRAMACIÓN [8]

El compilador PicBasic Pro (PBP), es un lenguaje de programación de nueva generación que hace más fácil y rápido programar microcontroladores PIC de Microchip. El lenguaje Basic es mucho más fácil de leer y escribir que el lenguaje ensamblador usado por otros lenguajes de programación como el MPLAB desarrollado por Microchip Technology.

Por ser un compilador real, los programas se ejecutan mucho más rápido y pueden ser mayores que sus equivalentes. El PicBasic Pro (PBP) por defecto crea archivos que corren en un PIC con un reloj de 4 MHz.

El PBP produce un código que puede ser programado para una variedad de micro controladores PIC que tengan de 8 a 68 pines y varias opciones en el chip incluyendo convertidores A/D, temporizadores y puertos seriales.

Por default PBP genera ejecutables para ser cargados en un PIC16F628. Solo se requieren unos pocos componentes extra para poner el sistema en marcha: 2 capacitores de 22pf para el cristal de 4Mhz, y un resistor de 4.7K entre VCC y el pin MCLR cuando se utiliza oscilador externo.

Hay algunos microcontroladores PIC que no trabajaran con el PBP, por ejemplo las series PIC 16C5X incluyendo el PIC 16C54 Y PIC 15C58. Estos microcontroladores PIC están basados en el núcleo de 12 bits en lugar del núcleo más corriente de 14 bits. El PBP necesita alguna de las opciones que solamente están disponibles con el núcleo de 14 bits como el stack (pila) de 8 niveles.

Otros micros PIC de las series 12C67X, 16C55X, 16C6X, 16C7X y 16C9X son programables una vez (OTP) o tienen una ventana de cuarzo en su parte superior (JW) para permitir el borrado exponiéndolo a una luz ultravioleta durante varios minutos.

2.3.1 CARACTERÍSTICAS DEL PicBasic Pro [13]

- Permite ejecución más rápida y programas más largos que los interpretes Basic.
- Acceso directo a cualquier pin o registro.
- Paginado automático para banco mayor a 2K.
- Arreglos con bit, byte y Word.
- Instrucciones de condicionamiento: If, Then Else y Endif.
- Expresiones con jerarquías de procesamiento.
- Interrupciones en Basic y Assembler.
- Librerías BASIC Stamp I y II.
- Instrucciones Built-in LCD.
- Soporta osciladores desde 3.58MHz a 40MHz.
- Instrucciones de acceso a buses I2C incluyendo memorias EEPROMs serie.
- In-line assembler y soporte Call.
- Compatibilidad MPLAB / MPASM / ICE.
- Se ejecuta en DOS o Windows.
- Soporta todos los microcontroladores Microchip PICmicro.

Para programar en el lenguaje PicBasic Pro, se puede utilizar el editor de texto MicroCode Studio, lo que constituye una gran ayuda para la programación de Pics de Microchip como es el PIC 16F628.

MicroCode es un programa editor de texto como Bloc de notas de Windows, pero con la diferencia que este está hecho exclusivamente para facilitar la programación de los microcontroladores PIC, los procedimientos para programar

son muy sencillos, primero seleccionamos el modelo del PIC, escribimos el programa y lo guardamos bajo un nombre y por último se presiona el botón compilar, si el programa está bien escrito y sin fallas compilará y mostrará en la parte inferior izquierda el espacio que requiere en el pic y automáticamente se creara 3 archivos (*.mac/*.asm/*.hex) este último es el más importante para el pic y será el que se grabe dentro del microcontrolador.

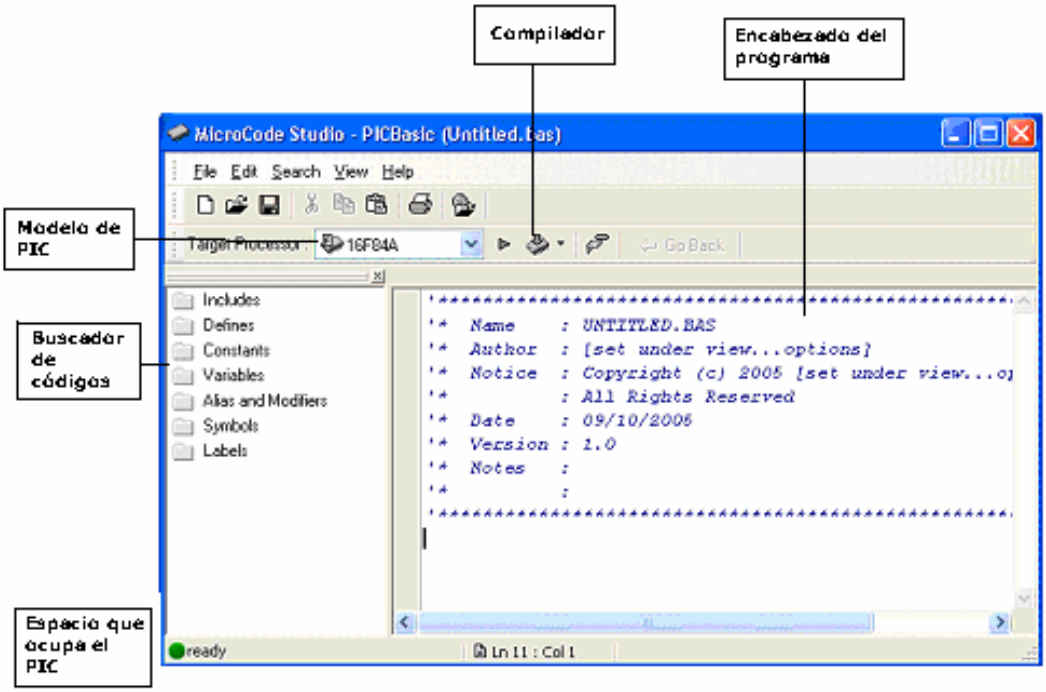


Figura 2.6. Partes de MicroCode

2.3.2 PicBasic Pro – SET DE INSTRUCCIONES [2]

@	Inserta una línea de código ensamblador
ASM...ENDASM	Inserta una sección de código ensamblador
BRANCH	GOTO computado(equiv. a ON..GOTO)
BRANCHL	BRANCH fuera de pagina (BRANCH largo)
BUTTON	Anti-rebote y auto-repetición de entrada en el pin especificado
CALL	Llamada a subrutina de ensamblador
CLEAR	Hace cero todas las variables
COUNT	Cuenta el numero de pulsos en un pin
DATA	Define el contenido inicial en un chip EEPROM

DEBUG	Señal asincrónica de salida en un pin fijo y baud
DISABLE	Deshabilita el procesamiento de ON INTERRUPT
DTMFOUT	Produce tonos en un pin
EEPROM	Define el contenido inicial en un chip EEPROM
ENABLE	Habilita el procesamiento de ON INTERRUPT
END	Detiene la ejecución e ingresa en modo de baja potencia
FOR...NEXT	Ejecuta declaraciones en forma repetitiva
FREQOUT	Produce hasta 2 frecuencias en un pin
GOSUB	Llama a una subrutina BASIC en la etiqueta especificada
GOTO	Continua la ejecución en la etiqueta especificada
HIGH	Hace alto la salida del pin
HSERIN	Entrada serial asincrónica (hardware)
HSEROUT	Salida serial asincrónica (hardware)
I2CREAD	Lee bytes de dispositivo I^{2C}
I2CWRITE	Graba bytes en dispositivo I2C
IF..THEN..ELSE..ENDIF	Ejecuta declaraciones en forma condicional
INPUT	Convierte un pin en entrada
(LET)	Asigna el resultado de una expresión a una variable
LCDOUT	Muestra caracteres en LCD
LOOKDOWN	Busca un valor en una tabla de constantes
LOOKDOWN2	Busca un valor en una tabla de constantes o variables
LOOKUP	Obtiene un valor constante de una tabla
LOOKUP2	Obtiene un valor constante o variable de una tabla
LOW	Hace bajo la salida de un pin
NAP	Apaga el procesador por un corto periodo de tiempo
ON INTERRUPT	Ejecuta una subrutina BASIC en un interrupt
OUTPUT	Convierte un pin en salida
PAUSE	Demora (resolución 1mseg.)
PAUSEUS	Demora (resolución 1 useg.)
PEEK	Lee un byte del registro

POKE	Graba un byte en el registro
POT	Lee el potenciómetro en el pin especificado
PULSIN	Mide el ancho de pulso en un pin
PULSOUT	Genera pulso hacia un pin
PWM	Salida modulada en ancho de pulso a un pin
RANDOM	Genera numero pseudo-aleatorio
RCTIME	Mide el ancho de pulso en un pin
READ	Lee byte de un chip EEPROM
RESUME	Continua la ejecución después de una interrupción
RETURN	Continua en la declaración que sigue al ultimo GOSUB
REVERSE	Convierte un pin de salida en entrada o uno de entrada en salida
SERIN	Entrada serial asincrónica (tipo BS!)
SERIN2	Entrada serial asincrónica (tipo BS2)
SEROUT	Salida serial asincrónica (tipo BS1)
SEROUT2	Salida serial asincrónica (tipo BS2)
SHIFTIN	Entrada serial sincrónica
SHIFTOUT	Salida serial sincrónica
SLEEP	Apaga el procesador por un periodo de tiempo
SOUND	Genera un tono o ruido blanco en un pin
STOP	Detiene la ejecución del programa
SWAP	Intercambia los valores de dos variables
TOGGLE	Hace salida a un pin y cambia su estado
WHILE..WEND	Ejecuta declaraciones mientras la condición sea cierta
WRITE	Graba bytes a un chip EEPROM
XIN	Entrada X - 10
XOUT	Salida X - 10

CAPITULO 3.

3. PROGRAMA PARA EL GRABADOR DEL PIC 16F628

Para la realización de este grabador, se tuvo que utilizar una interfaz que permita la comunicación entre el puerto USB del computador con el dispositivo a grabar.

La interfaz se lo realizó con la ayuda de otro microcontrolador, el cual debe servir como intermedio para permitir grabar, borrar y leer el PIC16F628; para llevar a cabo tal objetivo, se utilizó como interfaz de este programador el PIC18F2550, el cual entre sus principales características tiene incorporado pórticos para la transmisión y recepción de datos por medió de USB 2.0, pero puede usarse cualquiera de la serie 18Fxx5x.*

El programa implementado en el PIC18F2550, el software para programar el PIC, así como los controladores que permiten la comunicación entre la computadora a través del puerto USB y el grabador, fue descargado de la página web: http://perso.wanadoo.es/siscobf/winpic800.htm.

3.1 PROGRAMA PARA EL GRABADOR DEL PIC 16F628 [12]

El programa para la interfaz está editado en lenguaje C, que se presenta a continuación:

```
// IOCTLS.H -- IOCTL code definitions for mchpusb driver
// Copyright (C) 2003 by Microchip Technology, Inc.
// All rights reserved
#pragma once
#ifndef CTL_CODE
#pragma message("CTL_CODE undefined. Include winioctl.h or wdm.h")
#endif

//////////////////////////////////////////////////////////////////////
// Interface GUID for Microchip PIC18F4550 family driver

// {5354FA28-6D14-4E35-A1F5-75BB54E6030F}
```

* Ver Anexo 2

```
DEFINE_GUID(GUID_DEVINTERFACE_MCHPUSB, 0x5354fa28L, 0x6d14, 0x4e35, 0xa1, 0xf5, 0x75, 0xbb, 0x54, 0xe6, 0x03, 0x0f);

// Suffix strings for forming device names. Take the "device detail" path string
// returned by a SetupDiXxx enumeration and add on one of these, followed by
// a zero-based DECIMAL integer denoting the endpoint address. Use the resulting
// name string as the first argument in a call to CreateFile. Application
// code should use constructs like this to specify names in either Unicode or
// ANSI format: _T(MCHPUSB_PIPE_NAME). Driver code can use TEXT(MCHPUSB_PIPE_NAME), etc.
// For interrupt endpoints, using the "ASYNC" suffix means that you must use
// IOCTL_MCHPUPSB_WAIT_INTERRUPT to read input from the pipe. Using the EP
// suffix means you must use ReadFile or WriteFile, as appropriate to the endpoint's
// directionality.

#define MCHPUSB_PIPE_NAME                   "\\MCHP_EP"
#define MCHPUSB_ASYNC_PIPE_SUFFIX           "_ASYNC"

//////////////////////////////////////////////////////////////////////////////
//      IOCTL definitions.
//
//      Note that "input" and "output" refer to the parameter designations in calls to
//      DeviceIoControl, which are the opposite of common sense from the perspective of
//      an application making the calls.
//
//      IOCTL_MCHPUSB_GET_VERSION -- get driver revision level
//      Input:
//              None
//      Output:
//              32-bit revision level MMMMmmmm
//
//      IOCTL_MCHPUSB_GET_DESCRIPTOR -- get descriptor
//      Input:
//              GET_DESCRIPTOR_PARAMETER structure (q.v.)
//      Output:
//              As much of specified descriptor as will fit in the buffer
//
//      IOCTL_MCHPUSB_GET_CONFIGURATION_INFO -- get simplified information about configuration
//      Input:
//              None
//      Output:
//              DEVICE_INFO structure (q.v.)
//
//      IOCTL_MCHPUSB_SET_CONFIGURATION -- configure device
//      Input:
//              SET_CONFIGURATION_PARAMETER structure (q.v.)
//      Output:
//              None.
//      Note that this IOCTL will fail with STATUS_ACCESS_DENIED if any pipe handles are presently
//      open for endpoints that are not in the new configuration.
//
//      IOCTL_MCHPUSB_CONTROL_OUT -- issue control pipe command with output data
//      Input:
//              CONTROL_PARAMETER structure (q.v.)
```

```
//          Output:
//                    Data to be sent to device (at most 4096 bytes)
//          Note: you must use this IOCTL with care, so as not to put the device into a state other
//          than expected by the driver. E.g., sending a SET_ADDRESS request would be a really bad idea.
//
//          IOCTL_MCHPUSB_CONTROL_IN -- issue control pipe command with input data
//          Input:
//                    CONTROL_PARAMETER structure (q.v.)
//          Output:
//                    Buffer for data retrieved from device (at most 4096 bytes)
//
//          IOCTL_MCHPUSB_WAIT_INTERRUPT -- await input from interrupt endpoint
//          Input:
//                    None
//          Output:
//                    Interrupt data provided by the matching input transaction on the endpoint

#define IOCTL_MCHPUSB_GET_VERSION                                   CTL_CODE(FILE_DEVICE_UNKNOWN,
0x800, METHOD_BUFFERED, FILE_ANY_ACCESS)
#define IOCTL_MCHPUSB_GET_DESCRIPTOR                        CTL_CODE(FILE_DEVICE_UNKNOWN,          0x801,
METHOD_BUFFERED, FILE_READ_ACCESS)
#define IOCTL_MCHPUSB_GET_CONFIGURATION_INFO    CTL_CODE(FILE_DEVICE_UNKNOWN,                  0x802,
METHOD_BUFFERED, FILE_READ_ACCESS)
#define IOCTL_MCHPUSB_SET_CONFIGURATION                     CTL_CODE(FILE_DEVICE_UNKNOWN,          0x803,
METHOD_BUFFERED, FILE_WRITE_ACCESS)
#define IOCTL_MCHPUSB_CONTROL_OUT                                   CTL_CODE(FILE_DEVICE_UNKNOWN,
0x804, METHOD_IN_DIRECT, FILE_WRITE_ACCESS)
#define IOCTL_MCHPUSB_CONTROL_IN                                    CTL_CODE(FILE_DEVICE_UNKNOWN,
0x805, METHOD_OUT_DIRECT, FILE_READ_ACCESS)
#define IOCTL_MCHPUSB_WAIT_INTERRUPT                        CTL_CODE(FILE_DEVICE_UNKNOWN,          0x806,
METHOD_BUFFERED, FILE_READ_ACCESS)

//////////////////////////////////////////////////////////////////////////
// Parameter structure for IOCTL_MCHPUSB_GET_DESCRIPTOR:

typedef struct _GET_DESCRIPTOR_PARAMETER {
          UCHAR bType;                              // type of descriptor
          UCHAR bIndex;                             // index of descriptor
          USHORT wLangid;                           // language id for string descriptor
          } GET_DESCRIPTOR_PARAMETER, *PGET_DESCRIPTOR_PARAMETER;

#define USB_DEVICE_DESCRIPTOR_TYPE          0x01
#define USB_CONFIGURATION_DESCRIPTOR_TYPE   0x02
#define USB_STRING_DESCRIPTOR_TYPE          0x03

//////////////////////////////////////////////////////////////////////////
// Output pseudo-structures for IOCTL_MCHPUSB_GET_CONFIGURATION_INFO. Note that as much
// of the structure is filled in as there is room in the output buffer. To retrieve
// the entire structure, make one call with a 4-byte buffer to get the length, then
// make another call after allocating sufficient memory.

typedef struct _ENDPOINT_INFO {
   UCHAR bEndpointAddress;
```

```
    UCHAR bmAttributes;
    USHORT wMaxPacketSize;
    UCHAR bInterval;
            } ENDPOINT_INFO, *PENDPOINT_INFO;

#define NEXT_ENDPOINT_INFO(pei) ((pei) + 1)

#define USB_ENDPOINT_TYPE_CONTROL               0x00
#define USB_ENDPOINT_TYPE_ISOCHRONOUS           0x01
#define USB_ENDPOINT_TYPE_BULK                  0x02
#define USB_ENDPOINT_TYPE_INTERRUPT             0x03

typedef struct _INTERFACE_INFO {
    UCHAR bInterfaceNumber;
    UCHAR bAlternateSetting;
    UCHAR bNumEndpoints;
    UCHAR bInterfaceClass;
    UCHAR bInterfaceSubClass;
    UCHAR bInterfaceProtocol;
            _ENDPOINT_INFO Endpoints[1];            // placeholder for array of endpoint info structures
            } INTERFACE_INFO, *PINTERFACE_INFO;

#define NEXT_INTERFACE_INFO(pii) ((PINTERFACE_INFO) ((pii)->Endpoints + (pii)->bNumEndpoints))

typedef struct _CONFIGURATION_INFO {
            ULONG ulSize;                                       // size of config descriptor and all interface descriptors
            UCHAR bConfigurationValue;
            UCHAR bNumInterfaces;                       // number of interfaces in this configuration, including different
alternate settings
            _INTERFACE_INFO Interfaces[1];          // placeholder for array of interface info structures
            } CONFIGURATION_INFO, *PCONFIGURATION_INFO;

#define NEXT_CONFIGURATION_INFO(pci) ((PCONFIGURATION_INFO) ((PUCHAR) (pci) + (pci)->ulSize))

typedef struct _DEVICE_INFO {
            ULONG ulSize;                                                       // length of the entire structure
    UCHAR bDeviceClass;
    UCHAR bDeviceSubClass;
    UCHAR bDeviceProtocol;
    UCHAR bNumConfigurations;
    USHORT idVendor;
    USHORT idProduct;
    USHORT bcdDevice;
            _CONFIGURATION_INFO Configurations[1];          // placeholder for array of configuration info structures
            } DEVICE_INFO, *PDEVICE_INFO;

////////////////////////////////////////////////////////////////////////////
// Parameter pseudo-structure for IOCTL_MCHPUSB_SET_CONFIGURATION:

typedef struct _SET_CONFIGURATION_PARAMETER {
            UCHAR bConfigurationValue;    // desired configuration number
            UCHAR bNumInterfaces;                       // number of interfaces to enable
            struct _Interfaces {
```

```
                    UCHAR bInterfaceNumber;       // interface number
                    UCHAR bAlternateSetting;// alternate setting number
                    } Interfaces[1];                    // placeholder for array of interface specifications
          } SET_CONFIGURATION_PARAMETER, *PSET_CONFIGURATION_PARAMETER;

#define       SET_CONFIGURATION_PARAMETER_SIZE(numint)       (FIELD_OFFSET(SET_CONFIGURATION_PARAMETER,
Interfaces) + numint * sizeof(SET_CONFIGURATION_PARAMETER::_Interfaces))

//////////////////////////////////////////////////////////////////////////////
// Parameter structure for IOCTL_MCHPUSB_CONTROL_xxx requests

typedef struct _CONTROL_PARAMETER {
          UCHAR bmRequestType;
          UCHAR bRequest;
          USHORT wValue;
          USHORT wIndex;
          } CONTROL_PARAMETER, *PCONTROL_PARAMETER;
```

La compilación de este programa, es decir el archivo hexadecimal que tiene que ser grabado en el PIC18F2550, se encuentra en el archivo que se descargó anteriormente.

El programa implementado en el PIC18F2550 básicamente genera ondas de pulso por los puertos RA0 y RA1, con la finalidad de incrementar el valor del voltaje, de 5 voltios entregados por el puerto USB hasta un valor de 13 voltios, que constituyen el voltaje necesario para que el PIC16F628 entre en modo de programación.

Por otra parte los pórticos RB3 y RB4, son los encargados de que al momento de terminar la grabación del PIC16F628, permitan el reseteo del dispositivo, además de que tiene la finalidad de poder leer la memoria Flash del PIC16F628 desde la dirección 00H.

La transmisión de los datos se realiza a través de pórtico RB1 y la sincronización a través del pórtico RB0.

3.2 PRUEBAS

Para la realización de las pruebas se utilizó el WinPic800, que consiste en un software que permite grabar una gran variedad de microcontroladores PIC, en los cuales se encuentran el propio 18F2550 y el 16F628.

Este software, permite no solo la grabación por medio del puerto serial, sino que ofrece la opción de grabar a través del puerto paralelo, mediante conversor de USB-232 y por supuesto a través del puerto USB.

3.2.1 INSTALACIÓN DEL CONTROLADOR PARA EL GRABADOR USB, NECESARIO PARA LA COMUNICACIÓN ENTRE EL COMPUTADOR Y EL GRABADOR. [12]

- Detección del grabador por parte del sistema, figura 3.1.

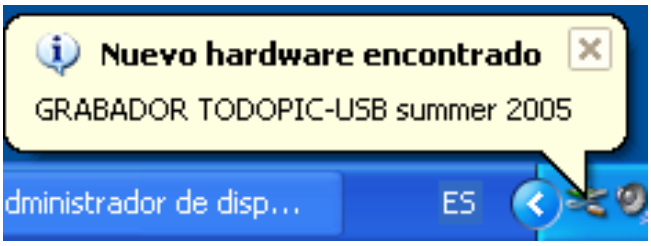


Figura 3.1. Detección del grabador

- Cuando el sistema ha reconocido el grabador de Pics, se procede a seguir los pasos para agregar un nuevo hardware. Se escoge la opción marcada en la figura 3.2 y se presiona < Siguiente >.

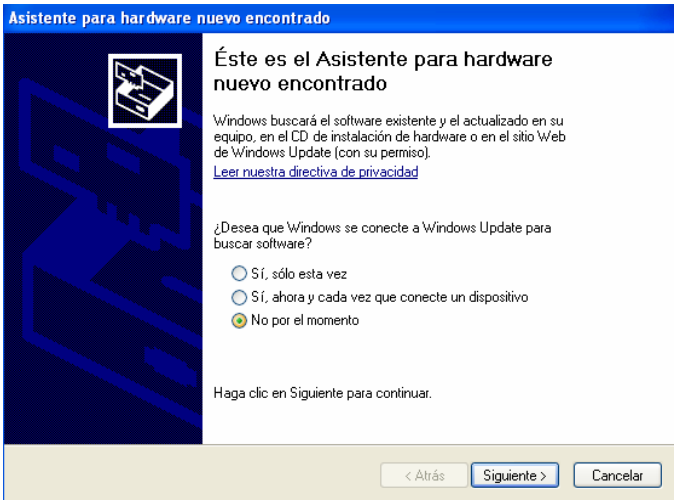


Figura 3.2. Asistente para hardware

- Escoger la opción: Instalar desde una lista o ubicación especificada, presionar <Siguiente>.

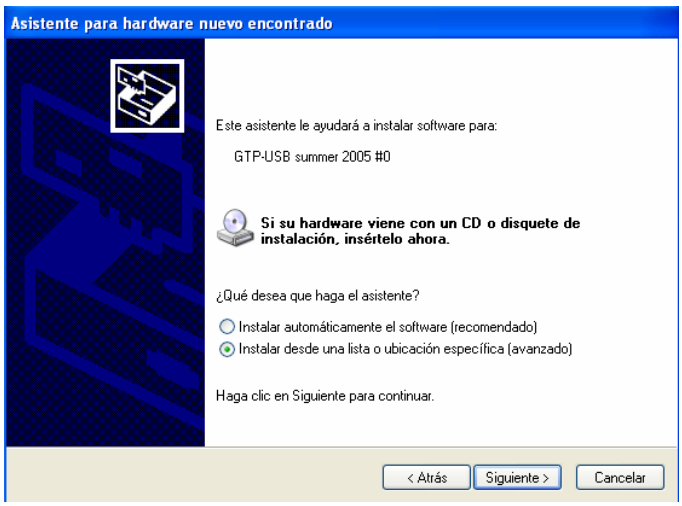


Figura 3.3. Instalación del controlador

- Ubicar la carpeta donde se localice el controlador, en este caso dentro de la carpeta C:/.....WinPic800/GTP-USB/Driver GTP-USB y presionar <Aceptar>.

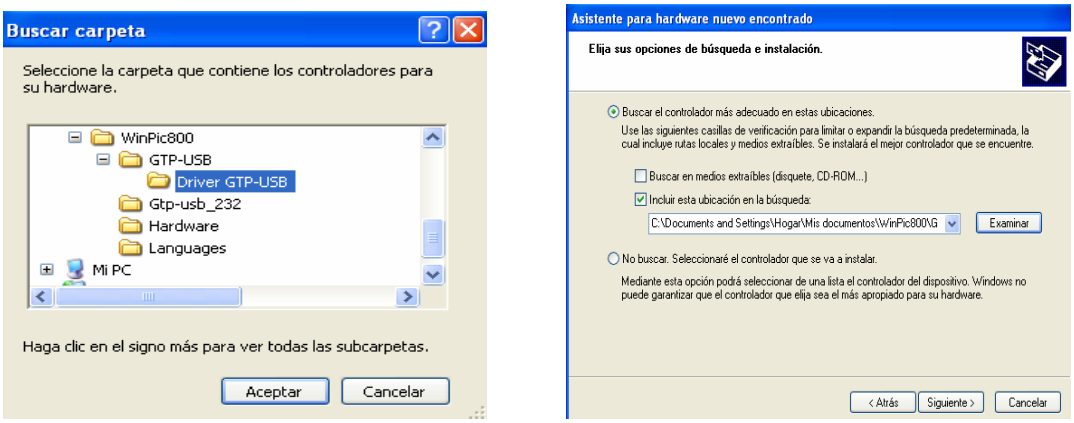


Figura 3.4. Ubicación del controlador

- En esta ventana presionar <Continuar>.

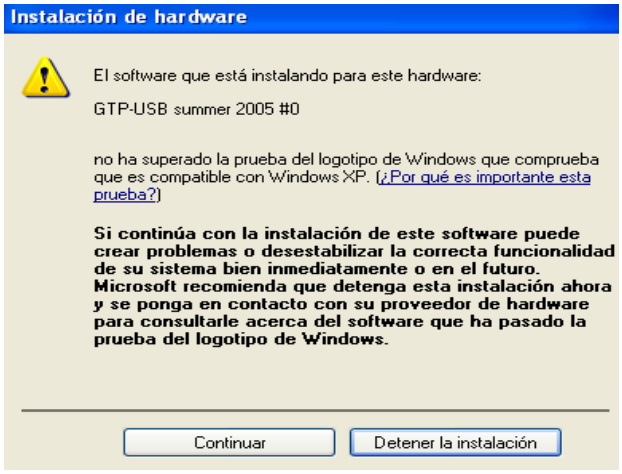


Figura 3.5. Instalación de hardware

- Esperar mientras los controladores son debidamente instalados.

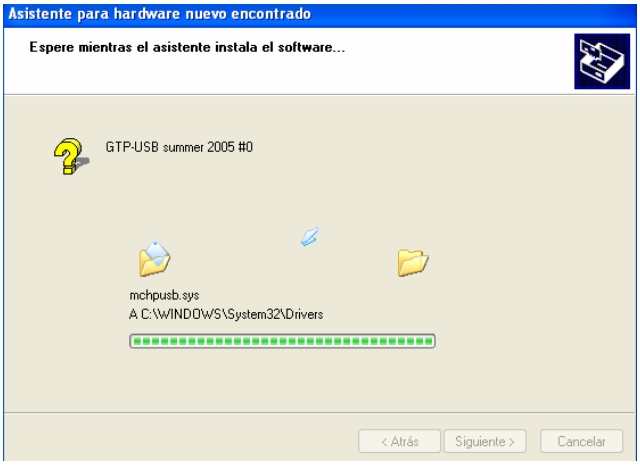


Figura 3.6. Asistente para hardware nuevo encontrado

- Una vez que el controlador sea correctamente instalado se presiona <Finalizar>.

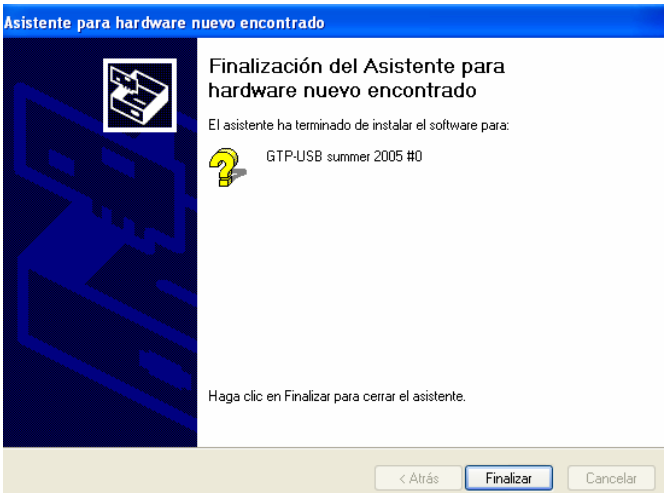


Figura 3.7. Finalización de la instalación

- Al conectar por primera vez el grabador USB al computador, saldrá un mensaje indicando que el sistema pudo detectar el dispositivo, tal como se indica en la figura 3.8; de esta manera se asegura el existe comunicación entre el computador y el grabador USB por medio de el interfaz.

Figura 3.8. Nuevo hardware encontrado

CAPITULO 4.

4. CONSTRUCCIÓN DEL DISPOSITIVO

4.1 PROTECCIONES

Debido a la sencillez del circuito, no es muy necesaria la utilización de protecciones, ya que el puerto USB entrega 5 voltios con una corriente máxima de salida de 500 miliamperios, que son más que suficientes para alimentar todo el dispositivo, además de contar dichos puertos con un fusible de 1 amperio de corriente máxima.

4.2 DISEÑO DEL CIRCUITO IMPRESO POR SOFTWARE

4.2.1 DIAGRAMA ESQUEMÁTICO

4.2.2 DIAGRAMA CIRCUITAL

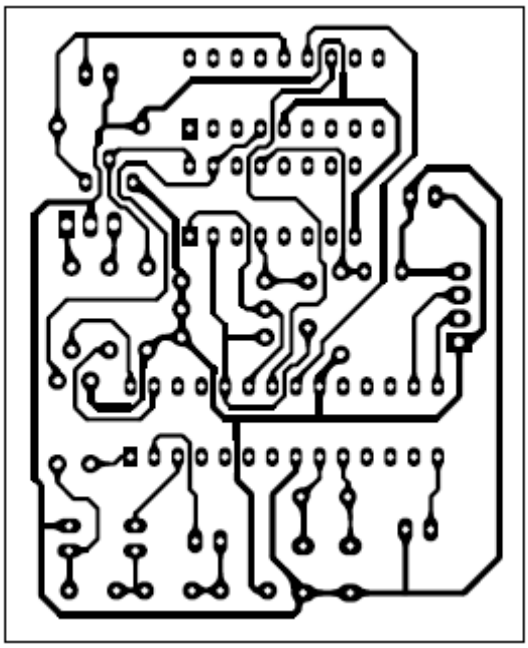


Figura 4.2. Circuito Impreso

4.3 PRUEBAS Y RESULTADOS

4.3.1 REALIZACIÓN DEL PROGRAMA CON EL MicroCode Studio PARA HACER PARPADEAR UN LED CON INTERVALOS DE 1 SEGUNDO.[2]

```
; ejemplo del parpadeo un led con intervalos de 1 segundo

Trisb.0 = 0
led var portb.0                 ;etiqueta para el puerto b.0
pepe:                           ;línea asignado con el nombre pepe
        high led                ;encender el led
                pause 1000      ;esperar 1000 milisegundos (1 segundo)
        low led                 ;apagar el led
                pause 1000      ;esperar 1000 milisegundos (1 segundo)
goto pepe                       ;ir a la línea que tenga el nombre pepe
end                             ;fin de las instrucciones
```

4.3.2 PRUEBA DE COMUNICACIÓN Y GRABACIÓN DEL GRABADOR USB PARA EL PIC16F628

- ✓ Prueba de comunicación y detección entre el computador y el grabador de PICs, ver figura 4.3.

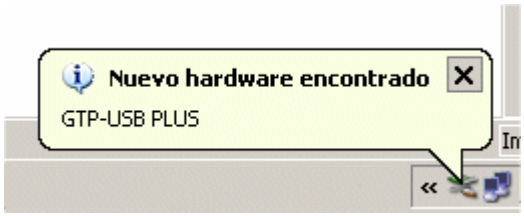


Figura 4.3. Ventana de nuevo hardware encontrado

- ✓ Prueba de detección del PIC 16F628 por medio del software WinPic800

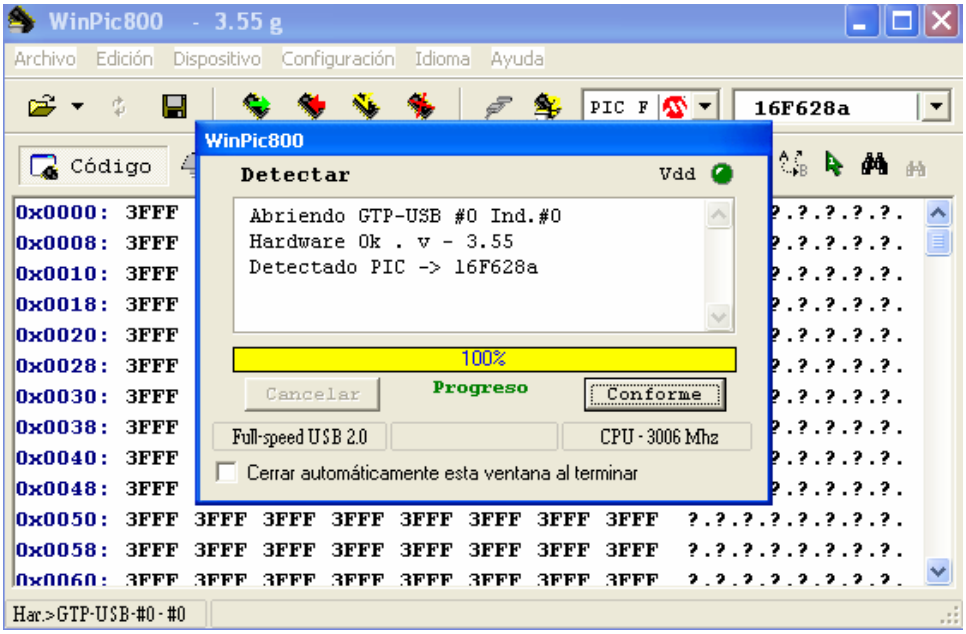


Figura 4.4. Detección del PIC utilizado

✓ Prueba de lectura del Pic 16F628 y el grabador por medio del software de programación.

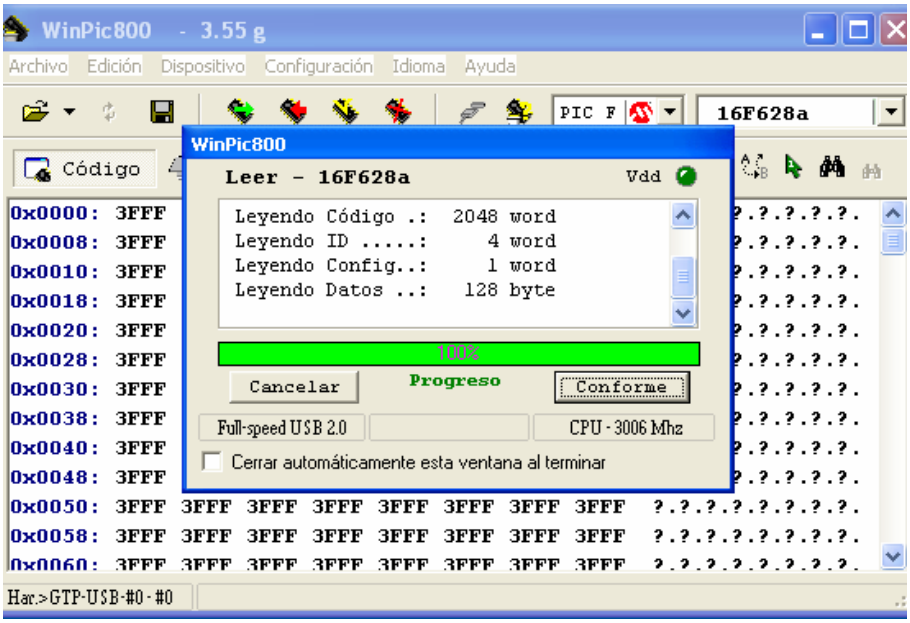


Figura 4.5. Lectura del PIC

✓ Prueba de grabación del PIC 16F628 por medio del software de programación

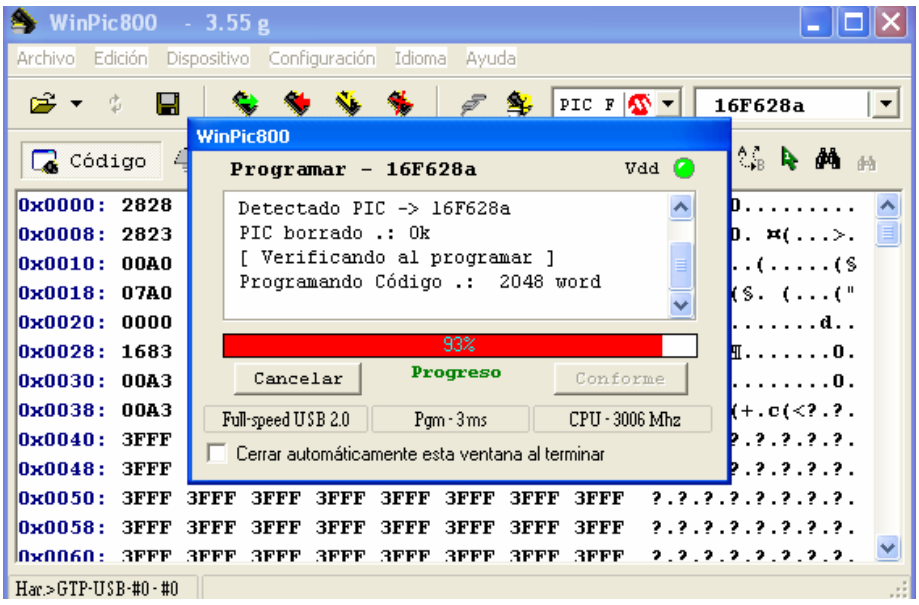


Figura 4.6. Grabación del PIC

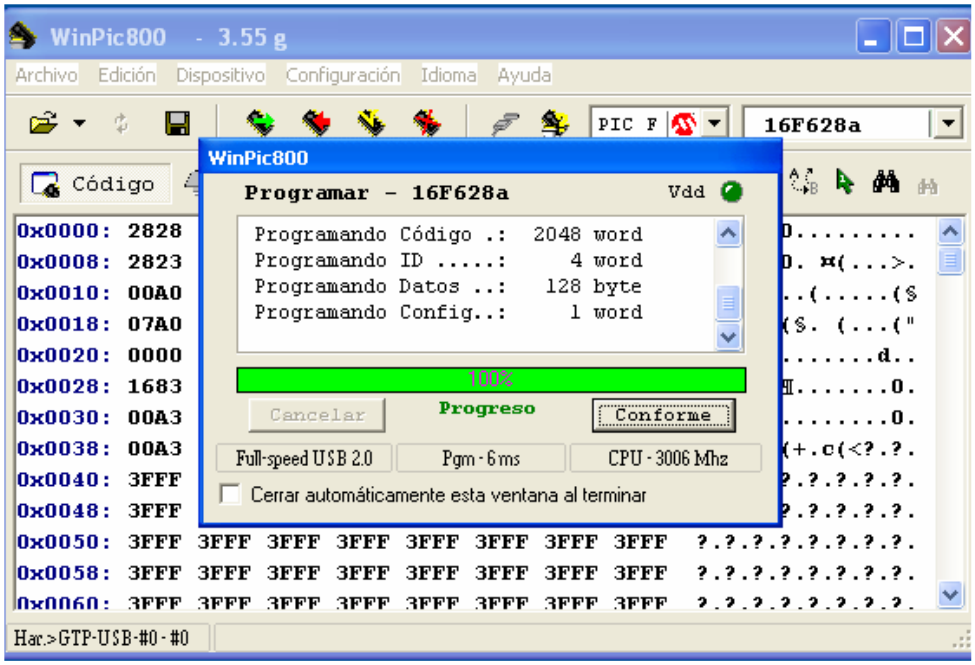


Figura 4.7. Lectura del nuevo programa grabado en el PIC

✓ Prueba de verificación del PIC 16F628 por medio del software de programación

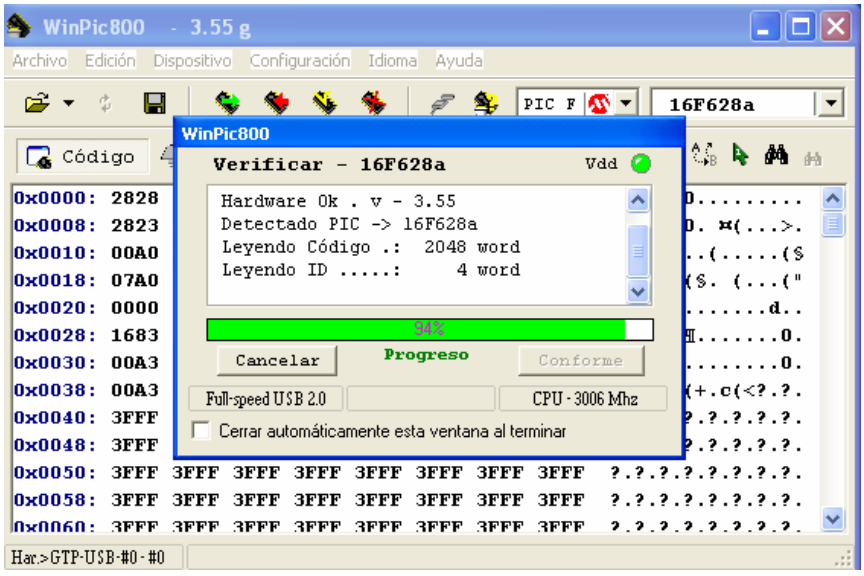


Figura 4.8. Proceso de verificación del PIC

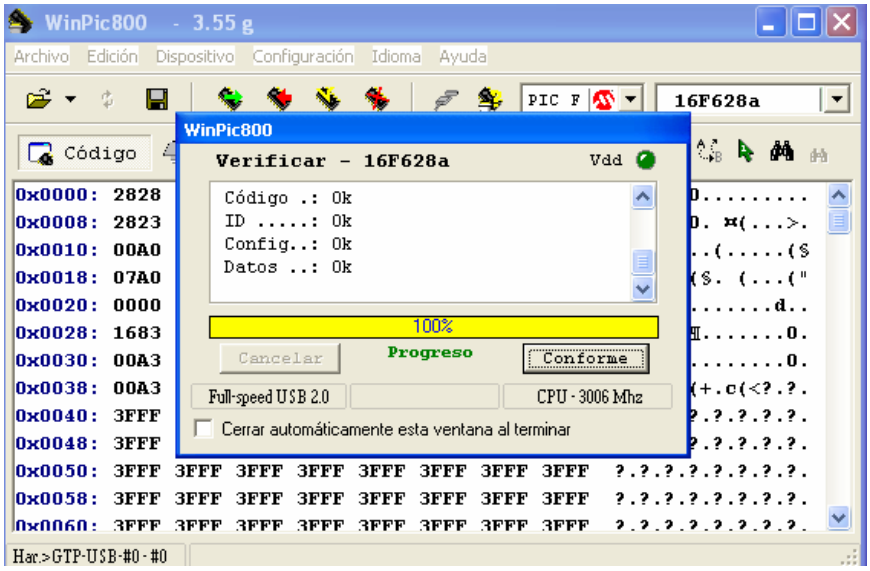


Figura 4.9. Proceso de grabación del PIC exitoso

CONCLUSIONES

El puerto USB y todos los dispositivos con conexión USB son Plug and Play ya que el dispositivo es detectado automáticamente al conectarlo al equipo (PC) y el sistema instala el driver adecuado o pide el disco de instalación, además no es necesario apagar, ni reiniciar, el equipo para instalar o desinstalar un dispositivo USB, permitiendo conservar recursos de memoria, pues cada dispositivo conectado requiere un driver residente, de este modo sólo se carga cuando se necesita y se descarga al desconectar el dispositivo.

Un puerto serie permite transmitir hasta 112,5 KB/s, el puerto paralelo entre 600KB/s y 15MB/s, pero el puerto USB es capaz de transmitir entre 1,5MB/s y 12MB/s, lo que lo convierte en la conexión ideal para módems, escáneres, CD-ROMs externos, dispositivos de copia de seguridad externos, etc.

Las comunicaciones asíncronas ponen más énfasis en garantizar el envío de datos, y menos en su temporización ("cuando" llegan); por su parte las comunicaciones isócronas son justamente lo contrario, ponen más énfasis en la oportunidad de la transmisión que en la velocidad. Esta sincronización es importante en situaciones como la reproducción de video, donde no debe existir desfase entre las señales de video y audio.

El PIC16F628 supera a los anteriores PICs en velocidad, tiene bajo consumo de potencia, y un amplio campo de disponibilidad de herramientas para ser programado, por esta razón es uno de los más utilizados en la actualidad.

En la actualidad la fabricación del PIC16F628, ha sido reemplazada por una nueva generación de microcontroladores, como es el caso del PIC16F628A, que tiene la ventaja de trabajar con dos voltajes de programación (Vpp), que son de 13 voltios o de 5 voltios, a diferencia de su predecesor el PIC16F628 que posee un voltaje de programación único de 13 voltios.

El PIC16F628 posee oscilador interno RC de 4 MHZ, además de tener la opción de ser configurado para trabajar con oscilador externo, MCLR programable, etc., esto hace que sea más sencillo de ponerlo a funcionar.

El grabador de PIC mediante el puerto USB, transfiere los datos 10 veces más rápido que el grabador serial ya conocido.

El programador WinPic 800 a diferencia del IcProg permite programar los microcontroladores no solamente a través de puerto serial y puerto paralelo, sino también con puerto de conversión de USB a 232 y puerto USB.

El lenguaje BASIC es un lenguaje de alto nivel, ya que es más entendible para el ser humano que el lenguaje Ensamblador, pero la desventaja de éste es que ocupa mucha más memoria que el lenguaje ensamblador, ya que éste último utiliza solamente instrucciones básicas para la programación de lo microcontroladores.

RECOMENDACIONES

No debemos olvidar a la hora de comprar un dispositivo USB que cada dispositivo puede funcionar como HUB, es decir, incluir uno o más conectores USB, de modo que podamos conectar un dispositivo a otro en cadena, y así, por ejemplo un teclado, puede incluir dos conectores USB, uno para el ratón y otro para el joystick, de igual modo el monitor puede servir de HUB y permitir conectar a él por ejemplo los altavoces, o el teclado, al cual a su vez se conectan el ratón y el joystick, etc. Hay que tener en cuenta que muchos dispositivos USB actuales no son más que conversiones de dispositivos existentes por lo que mucho aún no implementan su uso como HUBs, por lo que quizás valga la pena esperar un poco a que haya más dispositivos disponibles.

Para poder utilizar dispositivos USB, hay recordar que el sistema operativo instalado en nuestro equipo debe soportar este nuevo bus. Windows 95 en sus versiones OSR2.1 y OSR2.5 detecta el puerto USB y soporta dispositivos USB (la versión OSR2.0 también añadiendo el SUPLEMENTO USB), pero es realmente con el sistema operativo Windows 98 que los ya abundantes dispositivos USB no han dado problemas de instalación y funcionamiento, no dejando de lado las ultimas versiones de Windows NT.

Si queremos que los dispositivos USB funcionen, se debe asegurar que en la configuración de la main board o tarjeta madre se encuentre habilitada la opción ASSIGN USB IRQ – ENABLED, para su correcto funcionamiento. Tomando en cuenta que el programador de PICs es un dispositivo periférico que trabaja a través del puerto USB.

El PIC16F628 tiene tecnología CMOS, es decir consume muy poca corriente pero a la vez es susceptible a daños de estática, por esta razón se recomienda utilizar pinzas para su manipulación o a su vez utilizar una manilla antiestática.

No sobrepasar los niveles de corriente tanto de entrada como de salida; hay que recordar que el PIC16F628 puede entregar por cada uno de sus pines una corriente máxima de 25 miliamperios. Así mismo soporta una corriente máxima de 25 miliamperios.

BIBLIOGRAFÍA

1. CHINCHERO VILLACÍS Héctor Fernando, Dispositivo de adquisición de datos por puerto USB, Tesis de Grado, 2003.

2. REYES Carlos. (2004). Aprenda rápidamente a programar Microcontroladores PIC 16F62X, 16F81X, 12F6XX. Ecuador: Editorial Ayerve.

3. ANGULO José, ANGULO Ignacio. Microcontroladores PIC. Madrid: McGraw Hill. Tercera Edición.

REFERENCIAS

4. Tecnología USB
http://html.rincondelvago.com/tegnologia-usb.html

5. Antecedentes e Historia del USB
http://www.carsoft.com.ar/usb.htm

6. Diagrama de Capas y Controladores del USB
http://www.zator.com/Hardware/index.htm

7. Funcionamiento del PIC 16F628
file:///F:/boletin.php.htm

8. Set de instrucciones y manual del Compilador PicBasic Pro
http://www.frino.com.ar/pdf/manual_PBP.pdf

9. Tipos de conectores USB
http://www.ddmsa.com/index.html

10. Especificaciones técnicas y gráficas del microcontrolador PIC16F628
http://www.microchip.com

11. Especificaciones técnicas y gráficas del microcontrolador PIC18F2550
http://www.microchip.com

12. Descarga del Programador WinPic 800 y Controladores para el grabador USB
http://perso.wanadoo.es/siscobf/winpic800.htm

13. Características del PicBasic Pro
http://www.robodacta.com/detalles.asp

APENDICE A

SOFTWARE DE DESARROLLO DEL PROGRAMA MicroCode Studio PARA MICROCONTROLADORES PIC DE MICROCHIP

A.1 Descarga del programa gratuito MicroCode

A continuación se aprenderá a descargar los softwares necesarios para poder editar y compilar los programas para los microcontroladores PIC.

Ingresando a la página http://www.mecanique.co.uk procedemos a descargar el programa MicroCode. Una vez abierta la página web, damos un clic en "MicroCode Studio" como se ilustra en el siguiente gráfico:

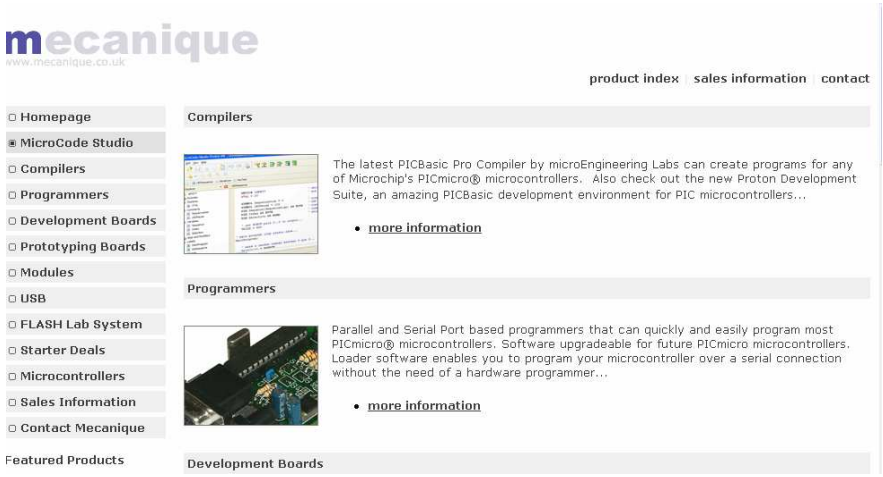


Figura A.1

En esta pantalla damos clic en USA, hosted by Reynolds Electronics y esperamos un momento para pasar a la siguiente pantalla.

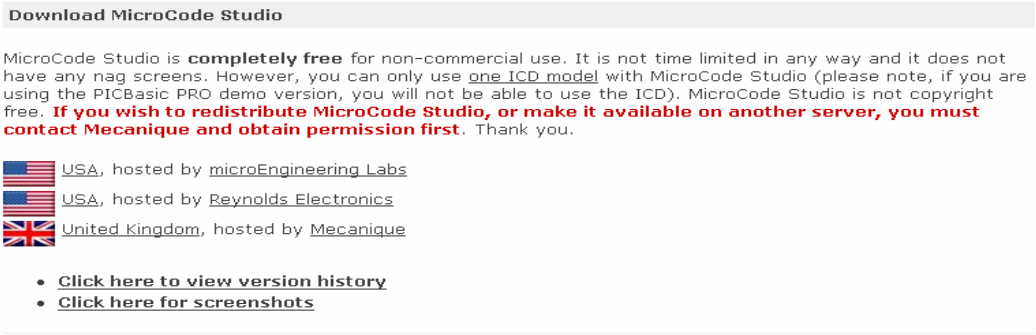


Figura A.2

A continuación nos deslizamos hasta “Download MicroCode Studio” y damos un clic en “Download MicroCode Studio 2.3.0.0”

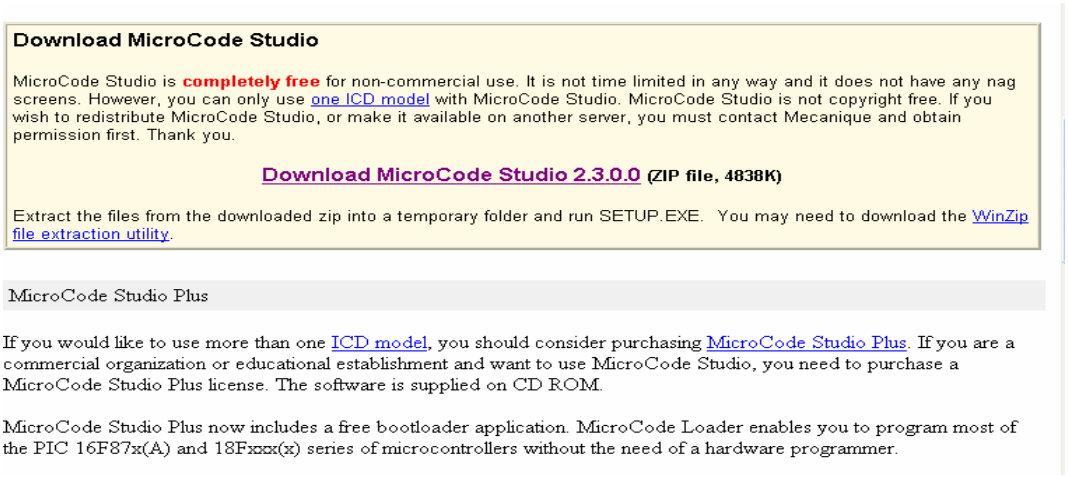


Figura A.3

Luego de unos instantes, veremos una pantalla de descarga, al igual que en el caso anterior sólo debemos escoger el lugar de destino donde se va a grabar el archivo seleccionado y dar un clic en Aceptar. Luego de unos instantes se descargará.

A.2 Descarga de Pbp 2.44 (Pic Basic Pro Ver 2.44)

Necesitamos la descarga de este compilador, ya que este programa se encarga de generar los archivos con extensión **.hex**, estos archivos son necesarios para poder grabar un microcontrolador PIC. Se puede adquirir este programa comprando, pero la página web, nos facilita un demo que podemos bajarnos de la página http://mecanique.co.uk. En los siguientes gráficos (Gráfico 3.19 y Gráfico 3.20) mostramos cual es el software a ser descargado.

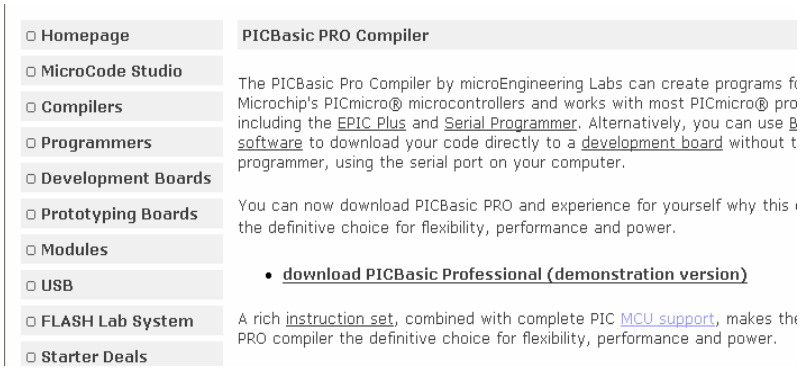


Figura A.4

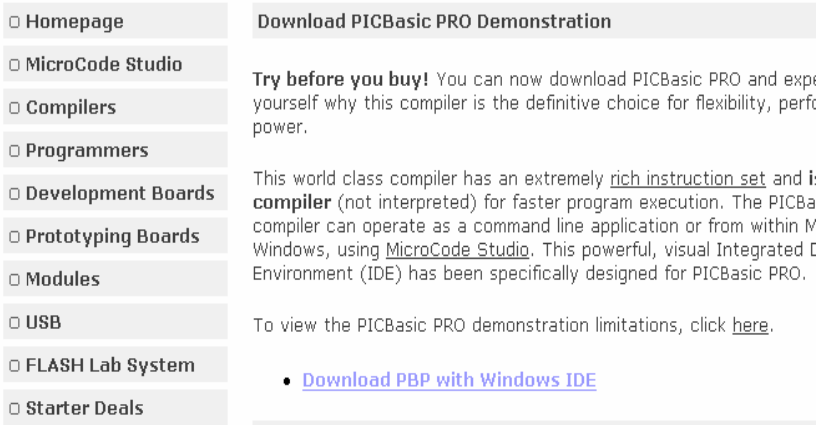


Figura A.5

A continuación esperamos la pantalla que nos confirme que esta descargando y guardamos el archivo.

Como podemos observar, se sigue los mismos pasos que en los casos anteriores.

A.3 Instalación del Software MicroCode Studio

La instalación del software es muy fácil, solo debemos buscar el archivo anteriormente descargado y lo ejecutamos. Lo próximo a seguir son los pasos que le indique el programa. Los gráficos (Gráficos 14; 15; 16; 17 y 18) que se muestran a continuación, indican lo fácil que es instalar el programa.

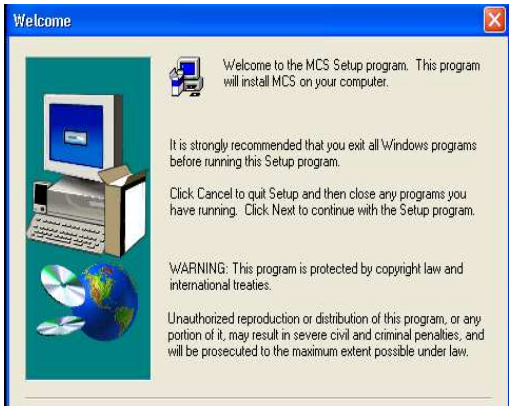


Figura A.6

Figura A.7

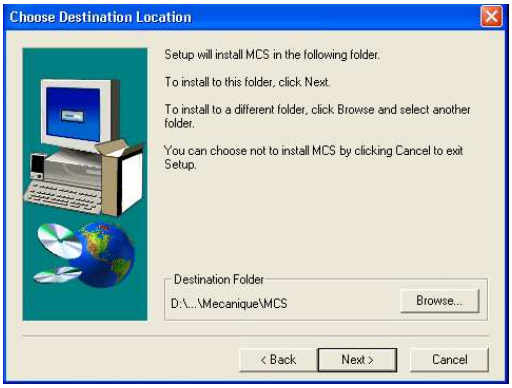


Figura A.8

Figura A.9

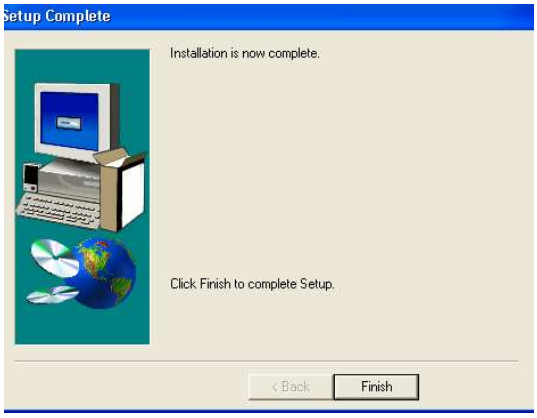


Figura A.10

A.4 Instalación de pbp 2.44

Este programa también se debe descomprimir al igual que el IC-Prog, si es la versión demo, solo podrá compilar 30 líneas de programa, caso contrario se deberá conseguir la versión completa de pbp 2.44.

A.5 Manejo del MicroCode Studio

El Microcode es un programa editor de texto, pero con la diferencia que está hecho exclusivamente para facilitar la programación de los microcontroladores PIC, los procedimientos para programar son muy sencillos, el seleccionar el modelo del PIC, escribir el programa y guardar con un nombre específico y por último, debemos compilar el programa. Solo si está bien escrito y sin fallas compilará, y se formarán tres archivos con las siguientes extensiones: **.mac/.asm/.hex**. Este último es el más importante para el PIC y es el que se grabará dentro del microcontrolador.

A continuación mostramos la pantalla del MicroCode Studio y observaremos las partes más importantes en esta pantalla.

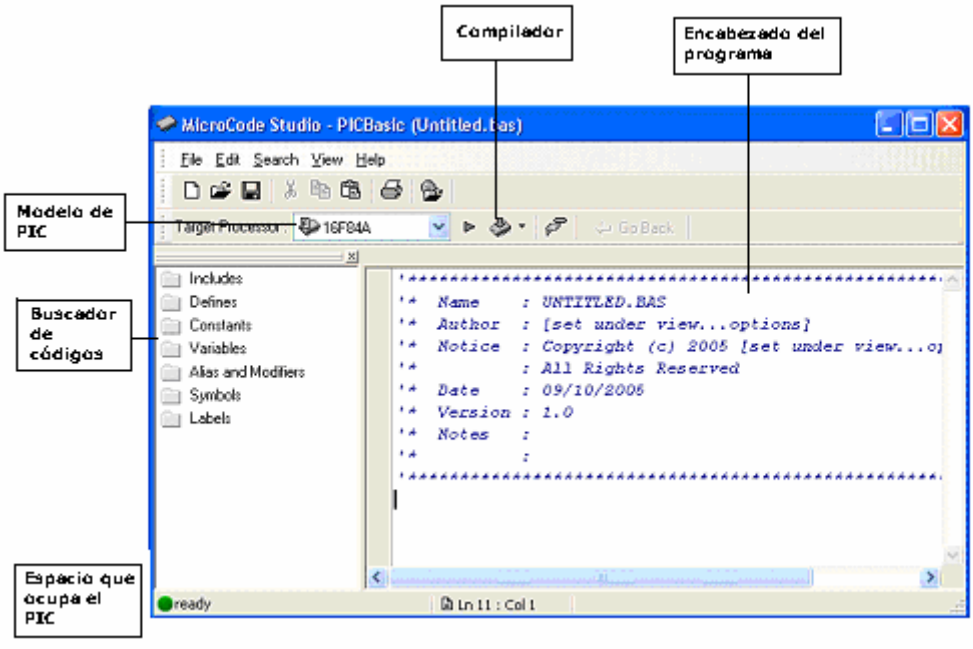


Figura A.11

ANEXOS 1

PIC16F628

MICROCHIP

PIC16F62X

FLASH-Based 8-Bit CMOS Microcontrollers

Devices Included in this Data Sheet:

- PIC16F627
- PIC16F628

Referred to collectively as PIC16F62X

High Performance RISC CPU:

- Only 35 instructions to learn
- All single cycle instructions (200 ns), except for program branches which are two-cycle
- Operating speed:
  - DC - 20 MHz clock input
  - DC - 200 ns instruction cycle

Device	Memory		
	FLASH Program	RAM Data	EEPROM Data
PIC16F627	1024 x 14	224 x 8	128 x 8
PIC16F628	2048 x 14	224 x 8	128 x 8

- Interrupt capability
- 16 special function hardware registers
- 8-level deep hardware stack
- Direct, Indirect and Relative addressing modes

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
  - Programmable input multiplexing from device inputs and internal voltage 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 (CCP) module
  - Capture is 16-bit, max. resolution is 12.5 ns
  - Compare is 16-bit, max. resolution is 200 ns
  - PWM max. resolution is 10-bit
- Universal Synchronous/Asynchronous Receiver/Transmitter USART/SCI
- 16 Bytes of common RAM

Special Microcontroller Features:

- Power-on Reset (POR)
- Power-up Timer (PWRT) and Oscillator Start-up Timer (OST)
- Brown-out Detect (BOD)
- Watchdog Timer (WDT) with its own on-chip RC oscillator for reliable operation
- Multiplexed $\overline{MCLR}$-pin
- Programmable weak pull-ups on PORTB
- Programmable code protection
- Low voltage programming
- Power saving SLEEP mode
- Selectable oscillator options
  - FLASH configuration bits for oscillator options
  - ER (External Resistor) oscillator
    - Reduced part count
  - Dual speed INTRC
    - Lower current consumption
  - EC External Clock input
  - XT Oscillator mode
  - HS Oscillator mode
  - LP Oscillator mode
- In-circuit Serial Programming™ (via two pins)
- Four user programmable ID locations

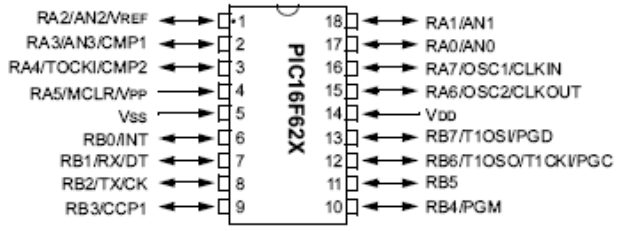
CMOS Technology:

- Low power, high speed CMOS FLASH technology
- Fully static design
- Wide operating voltage range
  - PIC16F627 - 3.0V to 5.5V
  - PIC16F628 - 3.0V to 5.5V
  - PIC16LF627 - 2.0V to 5.5V
  - PIC16LF628 - 2.0V to 5.5V
- Commercial, industrial and extended temperature range
- Low power consumption
  - < 2.0 mA @ 5.0V, 4.0 MHz
  - 15 µA typical @ 3.0V, 32 kHz
  - < 1.0 µA typical standby current @ 3.0V

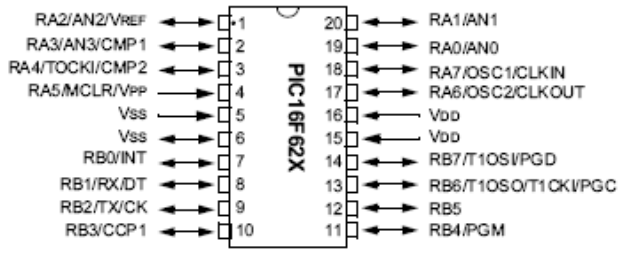
PIC16F62X

Pin Diagrams

PDIP, SOIC



SSOP



Device Differences

Device	Voltage Range	Oscillator	Process Technology (Microns)
PIC16F627	3.0 - 5.5	**(Note 1)**	0.7
PIC16F628	3.0 - 5.5	**(Note 1)**	0.7
PIC16LF627	2.0 - 5.5	**(Note 1)**	0.7
PIC16LF628	2.0 - 5.5	**(Note 1)**	0.7

Note 1: If you change from this device to another device, please verify oscillator characteristics in your application.

2.0 ARCHITECTURAL OVERVIEW

The high performance of the PIC16F62X family can be attributed to a number of architectural features commonly found in RISC microprocessors. To begin with, the PIC16F62X uses a Harvard architecture, in which, program and data are accessed from separate memories using separate buses. 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.

The Table below lists program memory (FLASH, Data and EEPROM).

TABLE 2-1: DEVICE DESCRIPTION

Device	Memory		
	FLASH Program	RAM Data	EEPROM Data
PIC16F627	1024 x 14	224 x 8	128 x 8
PIC16F628	2048 x 14	224 x 8	128 x 8
PIC16LF627	1024 x 14	224 x 8	128 x 8
PIC16LF628	2048 x 14	224 x 8	128 x 8

The PIC16F62X 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 PIC16F62X 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 PIC16F62X simple yet efficient. In addition, the learning curve is reduced significantly.

The PIC16F62X 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.

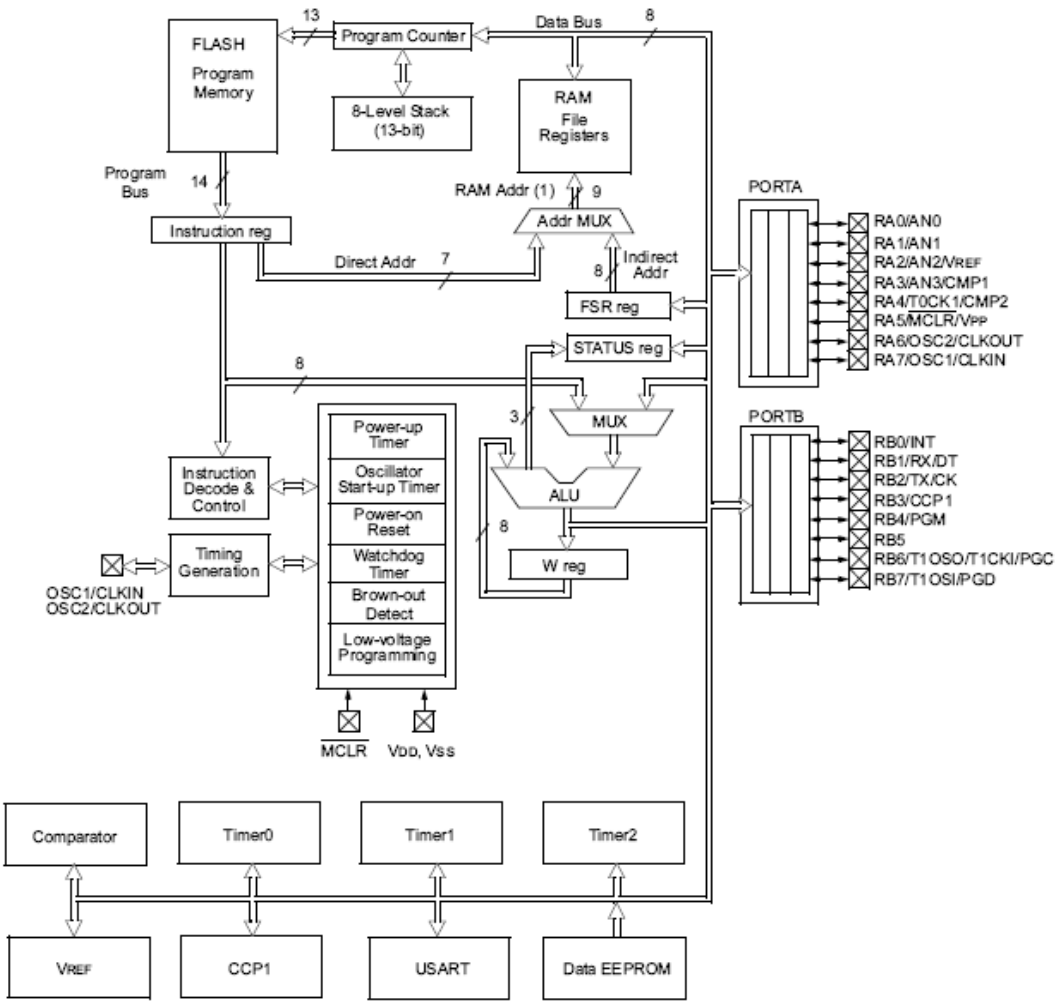
The W register is an 8-bit working register used for ALU operations. It is not an addressable register.

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 2-1, and a description of the device pins in Table 2-1.

Two types of data memory are provided on the PIC16F62X devices. Non-volatile EEPROM data memory is provided for long term storage of data such as calibration values, lookup table data, and any other data which may require periodic updating in the field. This data is 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. It is lost when power is removed.

FIGURE 2-1: BLOCK DIAGRAM



Note 1: Higher order bits are from the STATUS register.

TABLE 2-1: PIC16F62X 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/$\overline{\text{MCLR}}$/VPP	RA5	ST	—	Input port
	$\overline{\text{MCLR}}$	ST	—	Master clear
	VPP	—	—	Programming voltage input. 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.
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 ER/INTRC 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. ER 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.
	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

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 2-1: PIC16F62X PINOUT DESCRIPTION (CONTINUED)

Name	Function	Input Type	Output Type	Description
RB4/PGM	RB4	TTL	CMOS	Bi-directional I/O port. Can be software programmed for internal weak pull-up.
	PGM	ST	—	Low voltage programming input pin. Interrupt-on-pin change. 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. Wake-up from SLEEP on pin change. Can be software programmed for internal weak pull-up.
	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

2.1 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 2-2.

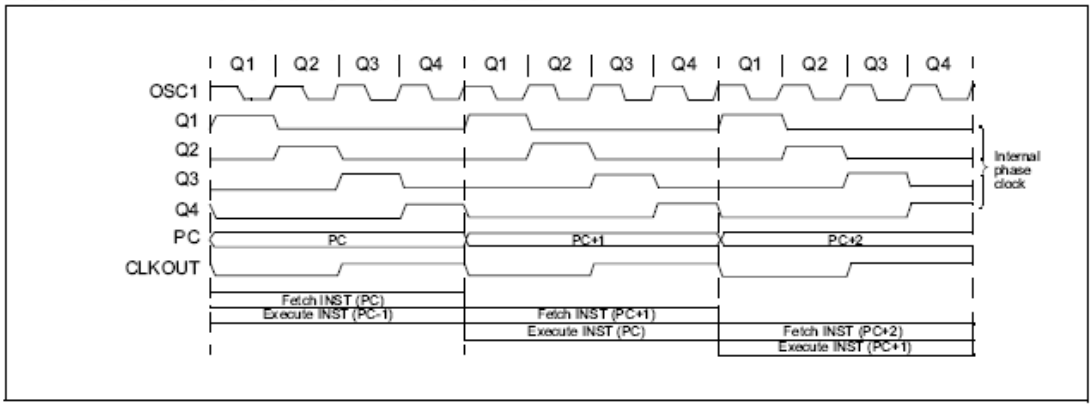
2.2 Instruction Flow/Pipelining

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 2-1).

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 2-2: CLOCK/INSTRUCTION CYCLE



EXAMPLE 2-1: INSTRUCTION PIPELINE FLOW

```
1. MOVLW 55h
2. MOVWF PORTB
3. CALL  SUB_1
4. BSF   PORTA, 3
```

Fetch 1	Execute 1				
	Fetch 2	Execute 2			
		Fetch 3	Execute 3		
			Fetch 4	Flush	
				Fetch SUB_1	Execute SUB_1

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.

ANEXO 2

PIC18F2550

MICROCHIP

PIC18F2455/2550/4455/4550

28/40/44-Pin High-Performance, Enhanced Flash USB Microcontrollers with nanoWatt Technology

Universal Serial Bus Features:

- USB V2.0 Compliant
- Low Speed (1.5 Mb/s) and Full Speed (12 Mb/s)
- Supports Control, Interrupt, Isochronous and Bulk Transfers
- Supports up to 32 endpoints (16 bidirectional)
- 1-Kbyte dual access RAM for USB
- On-chip USB transceiver with on-chip voltage regulator
- Interface for off-chip USB transceiver
- Streaming Parallel Port (SPP) for USB streaming transfers (40/44-pin devices only)

Power-Managed Modes:

- Run: CPU on, peripherals on
- Idle: CPU off, peripherals on
- Sleep: CPU off, peripherals off
- Idle mode currents down to 5.8 μA typical
- Sleep mode currents down to 0.1 μA typical
- Timer1 oscillator: 1.1 μA typical, 32 kHz, 2V
- Watchdog Timer: 2.1 μA typical
- Two-Speed Oscillator Start-up

Flexible Oscillator Structure:

- Four Crystal modes including High Precision PLL for USB
- Two External Clock modes, up to 48 MHz
- Internal oscillator block:
  - 8 user-selectable frequencies, from 31 kHz to 8 MHz
  - User-tunable to compensate for frequency drift
- Secondary oscillator using Timer1 @ 32 kHz
- Dual oscillator options allow microcontroller and USB module to run at different clock speeds
- Fail-Safe Clock Monitor
  - Allows for safe shutdown if any clock stops

Peripheral Highlights:

- High-current sink/source 25 mA/25 mA
- Three external interrupts
- Four Timer modules (Timer0 to Timer3)
- Up to 2 Capture/Compare/PWM (CCP) modules:
  - Capture is 16-bit, max. resolution 6.25 ns (TCY/16)
  - Compare is 16-bit, max. resolution 100 ns (TCY)
  - PWM output: PWM resolution is 1 to 10-bit
- Enhanced Capture/Compare/PWM (ECCP) module:
  - Multiple output modes
  - Selectable polarity
  - Programmable dead time
  - Auto-Shutdown and Auto-Restart
- Enhanced USART module:
  - LIN bus support
- Master Synchronous Serial Port (MSSP) module supporting 3-wire SPI™ (all 4 modes) and I^2C™ Master and Slave modes
- 10-bit, up to 13-channels Analog-to-Digital Converter module (A/D) with programmable acquisition time
- Dual analog comparators with input multiplexing

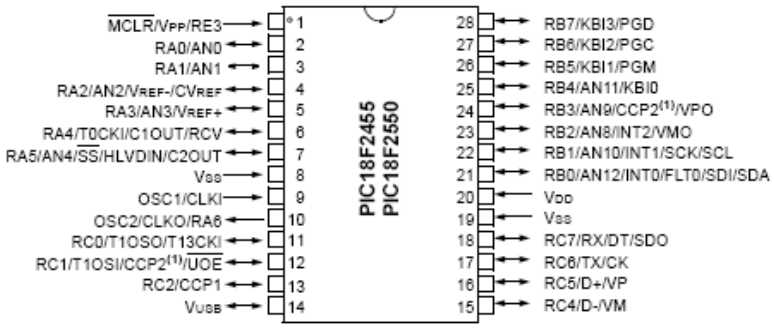
Special Microcontroller Features:

- C compiler optimized architecture with optional extended instruction set
- 100,000 erase/write cycle Enhanced Flash program memory typical
- 1,000,000 erase/write cycle Data EEPROM memory typical
- Flash/Data EEPROM Retention: > 40 years
- Self-programmable under software control
- Priority levels for interrupts
- 8 x 8 Single-Cycle Hardware Multiplier
- Extended Watchdog Timer (WDT):
  - Programmable period from 41 ms to 131s
- Programmable Code Protection
- Single-Supply 5V In-Circuit Serial Programming™ (ICSP™) via two pins
- In-Circuit Debug (ICD) via two pins
- Optional dedicated ICD/ICSP port (44-pin devices only)
- Wide operating voltage range (2.0V to 5.5V)

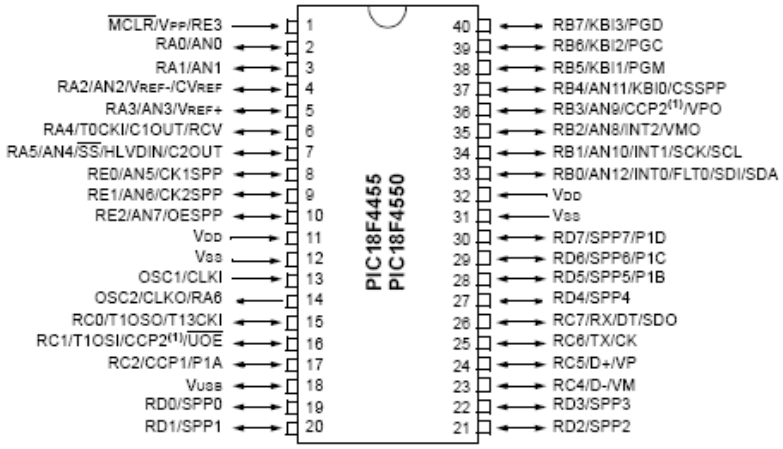
Device	Program Memory		Data Memory		I/O	10-bit A/D (ch)	CCP/ECCP (PWM)	SPP	MSSP		EAUSART	Comparators	Timers 8/16-bit
	Flash (bytes)	# Single-Word Instructions	SRAM (bytes)	EEPROM (bytes)					SPI™	Master I^2C™			
PIC18F2455	24K	12288	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F2550	32K	16384	2048	256	24	10	2/0	No	Y	Y	1	2	1/3
PIC18F4455	24K	12288	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3
PIC18F4550	32K	16384	2048	256	35	13	1/1	Yes	Y	Y	1	2	1/3

Pin Diagrams

28-Pin PDIP, SOIC



40-Pin PDIP



Note 1: RB3 is the alternate pin for CCP2 multiplexing.

1.0 DEVICE OVERVIEW

This document contains device specific information for the following devices:

- PIC18F2455
- PIC18F2550
- PIC18F4455
- PIC18F4550
- PIC18LF2455
- PIC18LF2550
- PIC18LF4455
- PIC18LF4550

This family of devices offers the advantages of all PIC18 microcontrollers – namely, high computational performance at an economical price – with the addition of high endurance, Enhanced Flash program memory. In addition to these features, the PIC18F2455/2550/4455/4550 family introduces design enhancements that make these microcontrollers a logical choice for many high-performance, power sensitive applications.

1.1 New Core Features

1.1.1 nanoWatt TECHNOLOGY

All of the devices in the PIC18F2455/2550/4455/4550 family incorporate a range of features that can significantly reduce power consumption during operation. Key items include:

- **Alternate Run Modes:** By clocking the controller from the Timer1 source or the internal oscillator block, power consumption during code execution can be reduced by as much as 90%.
- **Multiple Idle Modes:** The controller can also run with its CPU core disabled but the peripherals still active. In these states, power consumption can be reduced even further, to as little as 4% of normal operation requirements.
- **On-the-fly Mode Switching:** The power-managed modes are invoked by user code during operation, allowing the user to incorporate power-saving ideas into their application's software design.
- **Low Consumption in Key Modules:** The power requirements for both Timer1 and the Watchdog Timer are minimized. See **Section 28.0 "Electrical Characteristics"** for values.

1.1.2 UNIVERSAL SERIAL BUS (USB)

Devices in the PIC18F2455/2550/4455/4550 family incorporate a fully featured Universal Serial Bus communications module that is compliant with the USB Specification Revision 2.0. The module supports both low-speed and full speed communication for all supported data transfer types. It also incorporates its own on-chip transceiver and 3.3V regulator and supports the use of external transceivers and voltage regulators.

1.1.3 MULTIPLE OSCILLATOR OPTIONS AND FEATURES

All of the devices in the PIC18F2455/2550/4455/4550 family offer twelve different oscillator options, allowing users a wide range of choices in developing application hardware. These include:

- Four Crystal modes using crystals or ceramic resonators.
- Four External Clock modes, offering the option of using two pins (oscillator input and a divide-by-4 clock output) or one pin (oscillator input, with the second pin reassigned as general I/O).
- An internal oscillator block which provides an 8 MHz clock (±2% accuracy) and an INTRC source (approximately 31 kHz, stable over temperature and VDD), as well as a range of 6 user selectable clock frequencies, between 125 kHz to 4 MHz, for a total of 8 clock frequencies. This option frees an oscillator pin for use as an additional general purpose I/O.
- A Phase Lock Loop (PLL) frequency multiplier, available to both the high-speed crystal and external oscillator modes, which allows a wide range of clock speeds from 4 MHz to 48 MHz.
- Asynchronous dual clock operation, allowing the USB module to run from a high-frequency oscillator while the rest of the microcontroller is clocked from an internal low-power oscillator.

Besides its availability as a clock source, the internal oscillator block provides a stable reference source that gives the family additional features for robust operation:

- **Fail-Safe Clock Monitor:** This option constantly monitors the main clock source against a reference signal provided by the internal oscillator. If a clock failure occurs, the controller is switched to the internal oscillator block, allowing for continued low-speed operation or a safe application shutdown.
- **Two-Speed Start-up:** This option allows the internal oscillator to serve as the clock source from Power-on Reset, or wake-up from Sleep mode, until the primary clock source is available.

1.2 Other Special Features

- **Memory Endurance:** The Enhanced Flash cells for both program memory and data EEPROM are rated to last for many thousands of erase/write cycles – up to 100,000 for program memory and 1,000,000 for EEPROM. Data retention without refresh is conservatively estimated to be greater than 40 years.
- **Self-Programmability:** These devices can write to their own program memory spaces under internal software control. By using a bootloader routine, located in the protected Boot Block at the top of program memory, it becomes possible to create an application that can update itself in the field.
- **Extended Instruction Set:** The PIC18F2455/2550/4455/4550 family introduces an optional extension to the PIC18 instruction set, which adds 8 new instructions and an Indexed Literal Offset Addressing mode. This extension, enabled as a device configuration option, has been specifically designed to optimize re-entrant application code originally developed in high-level languages such as C.
- **Enhanced CCP Module:** In PWM mode, this module provides 1, 2 or 4 modulated outputs for controlling half-bridge and full-bridge drivers. Other features include auto-shutdown for disabling PWM outputs on interrupt or other select conditions and auto-restart to reactivate outputs once the condition has cleared.
- **Enhanced Addressable USART:** This serial communication module is capable of standard RS-232 operation and provides support for the LIN bus protocol. Other enhancements include Automatic Baud Rate Detection and a 16-bit Baud Rate Generator for improved resolution. When the microcontroller is using the internal oscillator block, the EUSART provides stable operation for applications that talk to the outside world without using an external crystal (or its accompanying power requirement).
- **10-bit A/D Converter:** This module incorporates programmable acquisition time, allowing for a channel to be selected and a conversion to be initiated, without waiting for a sampling period and thus, reducing code overhead.
- **Dedicated ICD/ICSP Port:** These devices introduce the use of debugger and programming pins that are not multiplexed with other micro-controller features. Offered as an option in select packages, this feature allows users to develop I/O intensive applications while retaining the ability to program and debug in the circuit.

1.3 Details on Individual Family Members

Devices in the PIC18F2455/2550/4455/4550 family are available in 28-pin and 40/44-pin packages. Block diagrams for the two groups are shown in Figure 1-1 and Figure 1-2.

The devices are differentiated from each other in six ways:

1. Flash program memory (24 Kbytes for PIC18FX455 devices, 32 Kbytes for PIC18FX550).
2. A/D channels (10 for 28-pin devices, 13 for 40/44-pin devices).
3. I/O ports (3 bidirectional ports and 1 input only port on 28-pin devices, 5 bidirectional ports on 40/44-pin devices).
4. CCP and Enhanced CCP implementation (28-pin devices have 2 standard CCP modules, 40/44-pin devices have one standard CCP module and one ECCP module).
5. Streaming Parallel Port (present only on 40/44-pin devices).

All other features for devices in this family are identical. These are summarized in Table 1-1.

The pinouts for all devices are listed in Table 1-2 and Table 1-3.

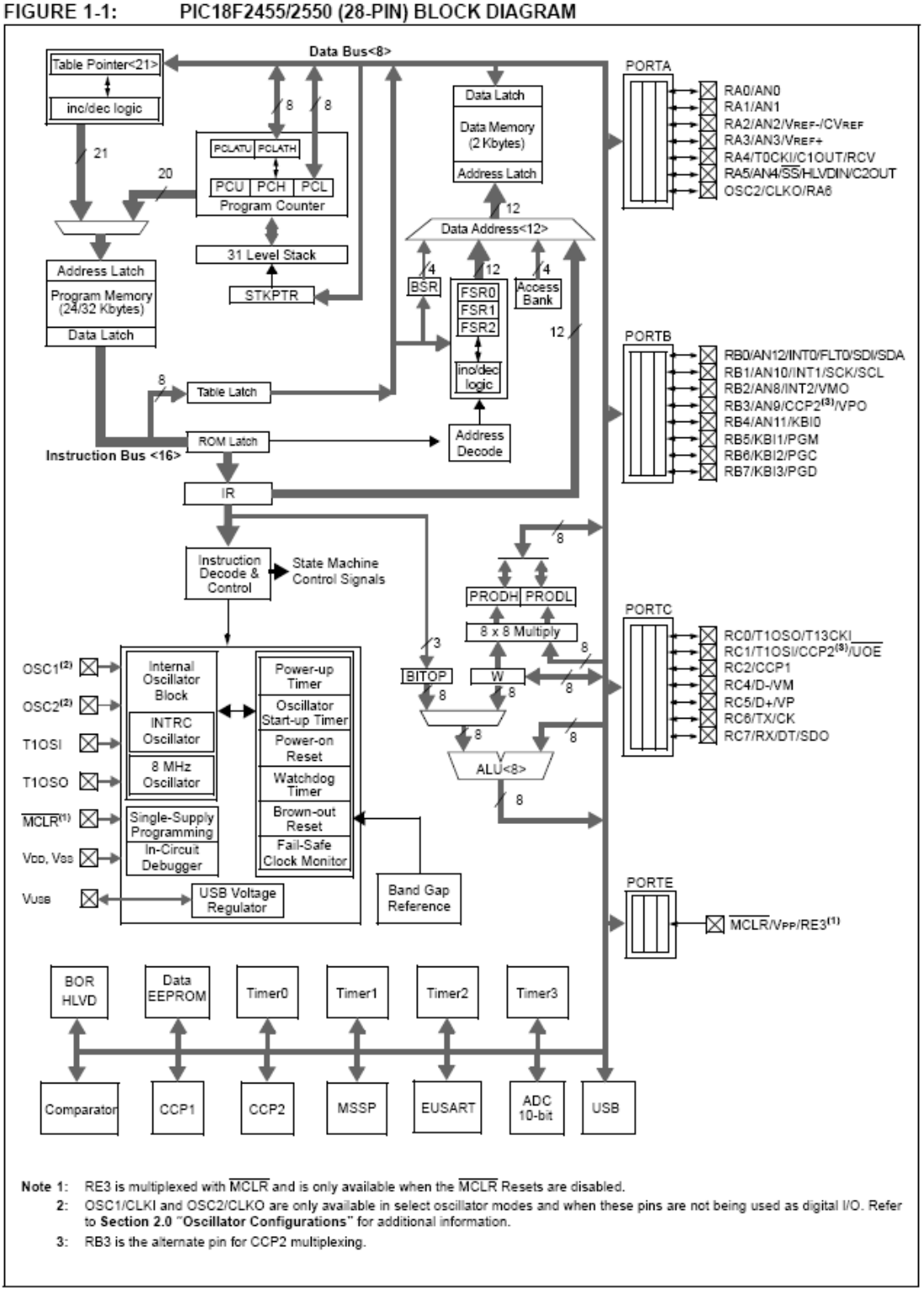
Like all Microchip PIC18 devices, members of the PIC18F2455/2550/4455/4550 family are available as both standard and low-voltage devices. Standard devices with Enhanced Flash memory, designated with an "F" in the part number (such as PIC18F2550), accommodate an operating VDD range of 4.2V to 5.5V. Low-voltage parts, designated by "LF" (such as PIC18LF2550), function over an extended VDD range of 2.0V to 5.5V.

TABLE 1-1: DEVICE FEATURES

Features	PIC18F2455	PIC18F2550	PIC18F4455	PIC18F4550
Operating Frequency	DC – 48 MHz	DC – 48 MHz	DC – 48 MHz	DC – 48 MHz
Program Memory (Bytes)	24576	32768	24576	32768
Program Memory (Instructions)	12288	16384	12288	16384
Data Memory (Bytes)	2048	2048	2048	2048
Data EEPROM Memory (Bytes)	256	256	256	256
Interrupt Sources	19	19	20	20
I/O Ports	Ports A, B, C, (E)	Ports A, B, C, (E)	Ports A, B, C, D, E	Ports A, B, C, D, E
Timers	4	4	4	4
Capture/Compare/PWM Modules	2	2	1	1
Enhanced Capture/ Compare/PWM Modules	0	0	1	1
Serial Communications	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART	MSSP, Enhanced USART
Universal Serial Bus (USB) Module	1	1	1	1
Streaming Parallel Port (SPP)	No	No	Yes	Yes
10-bit Analog-to-Digital Module	10 Input Channels	10 Input Channels	13 Input Channels	13 Input Channels
Comparators	2	2	2	2
Resets (and Delays)	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT	POR, BOR, RESET Instruction, Stack Full, Stack Underflow (PWRT, OST), MCLR (optional), WDT
Programmable Low-Voltage Detect	Yes	Yes	Yes	Yes
Programmable Brown-out Reset	Yes	Yes	Yes	Yes
Instruction Set	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled	75 Instructions; 83 with Extended Instruction Set enabled
Packages	28-pin PDIP 28-pin SOIC	28-pin PDIP 28-pin SOIC	40-pin PDIP 44-pin QFN 44-pin TQFP	40-pin PDIP 44-pin QFN 44-pin TQFP

PIC18F2455/2550/4455/4550

FIGURE 1-1: PIC18F2455/2550 (28-PIN) BLOCK DIAGRAM



Note 1: RE3 is multiplexed with MCLR and is only available when the MCLR Resets are disabled.

2: OSC1/CLKI and OSC2/CLKO are only available in select oscillator modes and when these pins are not being used as digital I/O. Refer to **Section 2.0 "Oscillator Configurations"** for additional information.

3: RB3 is the alternate pin for CCP2 multiplexing.

TABLE 1-2: PIC18F2455/2550 PINOUT I/O DESCRIPTIONS

Pin Name	Pin Number	Pin Type	Buffer Type	Description
	PDIP, SOIC			
MCLR/VPP/RE3	1			Master Clear (input) or programming voltage (input).
MCLR		I	ST	Master Clear (Reset) input. This pin is an active-low Reset to the device.
VPP		P		Programming voltage input.
RE3		I	ST	Digital input.
OSC1/CLKI	9			Oscillator crystal or external clock input.
OSC1		I	Analog	Oscillator crystal input or external clock source input.
CLKI		I	Analog	External clock source input. Always associated with pin function OSC1. (See OSC2/CLKO pins.)
OSC2/CLKO/RA6	10			Oscillator crystal or clock output.
OSC2		O	—	Oscillator crystal output. Connects to crystal or resonator in Crystal Oscillator mode.
CLKO		O	—	In select modes, OSC2 pin outputs CLKO which has 1/4 the frequency of OSC1 and denotes the instruction cycle rate.
RA6		I/O	TTL	General purpose I/O pin.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Alternate assignment for CCP2 when CCP2MX configuration bit is cleared.

2: Default assignment for CCP2 when CCP2MX configuration bit is set.

TABLE 1-2: PIC18F2455/2550 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number PDIP, SOIC	Pin Type	Buffer Type	Description
				PORTA is a bidirectional I/O port.
RA0/AN0	2			
RA0		I/O	TTL	Digital I/O.
AN0		I	Analog	Analog input 0.
RA1/AN1	3			
RA1		I/O	TTL	Digital I/O.
AN1		I	Analog	Analog input 1.
RA2/AN2/VREF-/CVREF	4			
RA2		I/O	TTL	Digital I/O.
AN2		I	Analog	Analog input 2.
VREF-		I	Analog	A/D reference voltage (low) input.
CVREF		O	Analog	Analog comparator reference output.
RA3/AN3/VREF+	5			
RA3		I/O	TTL	Digital I/O.
AN3		I	Analog	Analog input 3.
VREF+		I	Analog	A/D reference voltage (high) input.
RA4/T0CKI/C1OUT/RCV	6			
RA4		I/O	ST	Digital I/O.
T0CKI		I	ST	Timer0 external clock input.
C1OUT		O	—	Comparator 1 output.
RCV		I	TTL	External USB transceiver RCV input.
RA5/AN4/$\overline{SS}$/ HLVDIN/C2OUT	7			
RA5		I/O	TTL	Digital I/O.
AN4		I	Analog	Analog input 4.
$\overline{SS}$		I	TTL	SPI™ slave select input.
HLVDIN		I	Analog	High/Low-Voltage Detect input.
C2OUT		O	—	Comparator 2 output.
RA6	—	—	—	See the OSC2/CLKO/RA6 pin.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Alternate assignment for CCP2 when CCP2MX configuration bit is cleared.

2: Default assignment for CCP2 when CCP2MX configuration bit is set.

PIC18F2455/2550/4455/4550

TABLE 1-2: PIC18F2455/2550 PINOUT I/O DESCRIPTIONS (CONTINUED)

Pin Name	Pin Number PDIP, SOIC	Pin Type	Buffer Type	Description
				PORTB is a bidirectional I/O port. PORTB can be software programmed for internal weak pull-ups on all inputs.
RB0/AN12/INT0/FLT0/ SDI/SDA	21			
RB0		I/O	TTL	Digital I/O.
AN12		I	Analog	Analog input 12.
INT0		I	ST	External interrupt 0.
FLT0		I	ST	PWM Fault input (CCP1 module).
SDI		I	ST	SPI™ data in.
SDA		I/O	ST	I^2C™ data I/O.
RB1/AN10/INT1/SCK/ SCL	22			
RB1		I/O	TTL	Digital I/O.
AN10		I	Analog	Analog input 10.
INT1		I	ST	External interrupt 1.
SCK		I/O	ST	Synchronous serial clock input/output for SPI mode.
SCL		I/O	ST	Synchronous serial clock input/output for I^2C mode.
RB2/AN8/INT2/VMO	23			
RB2		I/O	TTL	Digital I/O.
AN8		I	Analog	Analog input 8.
INT2		I	ST	External interrupt 2.
VMO		O	—	External USB transceiver VMO output.
RB3/AN9/CCP2/VPO	24			
RB3		I/O	TTL	Digital I/O.
AN9		I	Analog	Analog input 9.
CCP2(1)		I/O	ST	Capture 2 input/Compare 2 output/PWM 2 output.
VPO		O	—	External USB transceiver VPO output.
RB4/AN11/KBI0	25			
RB4		I/O	TTL	Digital I/O.
AN11		I	Analog	Analog input 11.
KBI0		I	TTL	Interrupt-on-change pin.
RB5/KBI1/PGM	26			
RB5		I/O	TTL	Digital I/O.
KBI1		I	TTL	Interrupt-on-change pin.
PGM		I/O	ST	Low-Voltage ICSP™ Programming enable pin.
RB6/KBI2/PGC	27			
RB6		I/O	TTL	Digital I/O.
KBI2		I	TTL	Interrupt-on-change pin.
PGC		I/O	ST	In-Circuit Debugger and ICSP programming clock pin.
RB7/KBI3/PGD	28			
RB7		I/O	TTL	Digital I/O.
KBI3		I	TTL	Interrupt-on-change pin.
PGD		I/O	ST	In-Circuit Debugger and ICSP programming data pin.

Legend: TTL = TTL compatible input; CMOS = CMOS compatible input or output; ST = Schmitt Trigger input with CMOS levels; I = Input; O = Output; P = Power

Note 1: Alternate assignment for CCP2 when CCP2MX configuration bit is cleared.

2: Default assignment for CCP2 when CCP2MX configuration bit is set.

