ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE FORMACIÓN DE TECNÓLOGOS

CONSTRUCCIÓN DE UN SISTEMA INALÁMBRICO QUE CONTROLARA A UN PROTOTIPO DE CARRITO DESDE UN COMPUTADOR.

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

JOSÉ LUIS MALDONADO RODRÍGUEZ

doc.leo@hotmail.com

DARÍO JAVIER SOLÍS ALULIMA

djsa83@gmail.com

DIRECTOR: TLGO. HUGO FABIÁN SVIERCOVICH DOBRONSKI

Fabian.sviercovich@epn.edu.ec

QUITO, ENERO, 2012

DECLARACIÓN

Nosotros, Maldonado Rodríguez José Luis y Solís Alulima Darío Javier declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentado 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 correspondiente a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normativa institucional vigente.

---------- ----------

José Luis Maldonado. Darío Solís Alulima.

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Darío Javier Solís Alulima y José Luis Maldonado Rodríguez, bajo mi supervisión.

_____

Tlgo. Fabián Sviercovich

Director del Proyecto

AGRADECIMIENTOS

Agradezco a Dios por cada día de vida los cuales me permiten realizar mis metas y objetivos aprendiendo de mis errores y siempre consiente de tratar de ya no volver a repetirlos para ser mejor como persona y como profesional.

Agradezco a mis padres y mis hermanos por el apoyo y sobre todo el amor que he recibido y que siempre lo recibiré incondicionalmente toda mi vida, por ser un ejemplo de lucha y amor.

Agradezco a mis verdaderos amigos que se preocuparon por ayudarme a realizar este presente proyecto con sus concejos y conocimientos Santiago, Javier y mi compañero y amigo de proyecto José Luis.

Agradezco a todos los que formaron parte de mi formación profesional mis compañeros, profesores y amigos de la Escuela Politécnica Nacional en especial de la Escuela de Formación de Tecnólogos, a mi profesor tutor Tlgo. Fabián Sviercovich por su sabiduría y concejos para el presente proyecto y a los ingenieros Patricio Carrasco y Alcivar Costales

Darío.

DEDICATORIA

Dedico en especial el presente proyecto a mis padres Carlos y Flor María por su infinito y eterno amor junto con sus concejos los cuales me han servido y servirán para toda mi vida, mis hermanos Byron por ser un ejemplo de firmeza y mi hermano menor Carlitos que gracias a el nunca olvide ser un niño cuando ya era grande.

Y también dedico a una persona que llego cuando menos lo esperaba con su amor incondicional para darme las fuerzas, ánimos, amor y creyendo en mi para culminar con este proyecto, mi novia LiLy.

Darío.

ÍNDICE

CAPÍTULO II

CONSTRUCCIÓN DEL HARDWARE.

CAPÍTULO III

DESARROLLO DEL SOFTWARE

CAPITULO IV

SISTEMA DE CONTROL INALÁMBRICO DESDE EL COMPUTADOR.

CAPITULO V

PRUEBAS Y RESULTADOS

CAPITULO VI

CONCLUSIONES Y RECOMENDACIONES

ÍNDICE DE FIGURAS.

ÍNDICE DE TABLAS.

RESUMÉN

La construcción de un sistema inalámbrico que controlará a un prototipo de carrito desde un computador, se lo realizo mediante un método de investigación científica y tecnológica actual, para poder determinar los mejores elementos que se utilizara para la construcción del prototipo.

Como se planteo el proyecto, se desea realizar un control remoto mediante una computadora o cualquier equipo portátil los cuales deben cumplir con algunos requerimientos básicos como es permitir una conexión USB y un sistema operativo que permita la instalación de paquetes de computación necesarios para el funcionamiento de nuestro sistema de control inalámbrico como es el programa **VISUAL BASIC 6.0** y los software necesarios para la programación de los microcontroladores.

Continuando con la investigación también se determino todos los elementos electrónicos a utilizarse y las herramientas necesarias, dentro de la adquisición de elementos electrónicos se determino el dispositivo inalámbrico a usarse, el cual es un producto nuevo dentro del mercado de dispositivos inalámbricos llamados módulos inalámbricos XBEE, su alcance es mayor que muchos dispositivos que ya se encuentran en el mercado durante algún tiempo y a más de eso es un dispositivo bidireccional muy confiable.

También se utilizaron los microcontroladores ATMEGA 164 y ATMEGA 48 que son microcontroladores más avanzados tecnológicamente para garantizar y obtener circuitos electrónicos confiables y compactos.

Para la construcción de los circuitos se utiliza tres tipos de software: Orcad Layout, el Bascon AVR y el progisp los cual son muy importantes para el desarrollo y desenvolvimiento de este proyecto.

INTRODUCCIÓN

El presente trabajo se lo realizo con una investigación previa utilizando todos los conocimientos adquiridos dentro de nuestra formación profesional e investigando las nuevas tecnologías presentes en el mercado.

El proceso de la construcción del prototipo a ser controlado mediante un sistema inalámbrico desde un computador se lo realizo dentro de una planificación previa determinada con el objetivo de optimizar los recursos y el tiempo, teniendo como primer objetivo la realización de las pruebas para poder determinar los circuitos definitivos a realizar, esto se obtuvo armando los circuitos en el protoboard y analizando y corrigiendo los errores obtenidos en el proceso de determinar los circuitos.

Una vez determinado los circuitos definitivos se realiza los esquemáticos de cada uno mediante el software Orcad Layout y así obtener las placas electrónicas de cada uno de los circuitos es decir los circuitos emisor y receptor, este proceso se lo explica paso a paso dentro del proyecto.

En el desarrollo de los programas para el proyecto se utiliza el lenguaje Basic de programación, del cual se obtiene el archivo hexadecimal (HEX) después de haberlo compilado, para luego poder cargar el programa al microcontrolador. En el ATMEGA 164 y ATMEGA 48 se utiliza un programador para AVRs llamado progisp, una de las ventajas de este software es la de poder manipular los fusibles de configuración de los microcontrolador según desee el programador.

CAPÍTULO I

CONCEPTOS GENERALES

1.1 MANDO A DISTANCIA

Un **mando a distancia** o **control remoto** es un dispositivo electrónico usado para realizar una operación remota (o telemando) sobre una máquina.

El término se emplea generalmente para referirse al mando a distancia (llamado por lo general simplemente "el mando" o, en Latinoamérica, "el control") para el televisor u otro tipo de aparato electrónico casero, como DVD, Hi-Fi, computadoras, y para encender y apagar un interruptor, la alarma, o abrir la puerta del estacionamiento. Los mandos a distancia para esos aparatos son normalmente pequeños objetos (fácilmente manipulables con una mano) con una matriz de botones para ajustar los distintos valores, como por ejemplo, el canal de televisión, el número de canción y el volumen. De hecho, en la mayoría de dispositivos modernos el mando contiene todas las funciones de control, mientras que el propio aparato controlado sólo dispone de los controles más primarios. La mayoría de estos controles remotos se comunican con sus respectivos aparatos vía señales de infrarrojo (IR) y sólo unos pocos utilizan señales de radio. En los vehículos modernos las clásicas llaves incorporan ahora mandos a distancia con diversas funciones. Su fuente de energía suele ser pequeñas pilas de tipo *AA*, *AAA* o *de botón*.

1.1.1 HISTORIA

Uno de los primeros ejemplos de control remoto (mando a distancia en España) fue desarrollado en 1893 por Nikola Tesla y descrito en su patente número 613809, titulada *Método de un aparato para el mecanismo de control de vehículo o vehículos en movimiento*.

En 1903, Leonardo Torres Quevedo presentó el *telekino* en la Academia de Ciencias de París, acompañado de una memoria y haciendo una demostración

experimental. En ese mismo año obtuvo la patente en Francia, España, Gran Bretaña y Estados Unidos. El *telekino* consistía en un autómata que ejecutaba órdenes transmitidas mediante ondas hertzianas; constituyó el primer aparato de radiodirección del mundo, y fue un pionero en el campo del control remoto. El 25 de septiembre de 1906, en presencia del Rey y ante una gran multitud, demostró con éxito el invento en el puerto de Bilbao al guiar un bote desde la orilla; más tarde intentaría aplicar el *telekino* a proyectiles y torpedos, pero tuvo que abandonar el proyecto por falta de financiación.

El primer modelo de avión por control remoto voló en 1932. Durante la Segunda Guerra Mundial, se llevó a cabo el uso de tecnología de control remoto para propósitos militares; uno de los resultados de esto fue el misil alemán Wasserfall. El primer artilugio diseñado para controlar remotamente un televisor fue desarrollado por Zenith Radio a principios de 1950s. El control —extraoficialmente llamado "Lazy Bones" (~*para vagos*)— usaba un cable para conectarse al televisor. Para mejorar el engorroso sistema, se creó un control remoto sin cables en 1955. El mando, llamado "Flashmatic", funcionaba enviando un rayo de luz a una célula fotoeléctrica. Desafortunadamente, las células no distinguían entre la luz del mando y la luz de otras fuentes. El Flashomatic también requería que se apuntara el mando a distancia al receptor con precisión.

En 1956, Robert Adler desarrollo el "Zenith Space Command" (*Mando del espacio cenit*), un control sin cables. Era mecánico y usaba ultrasonidos para cambiar el canal y el volumen. Cuando el usuario pulsaba un botón del mando a distancia, hacía un chasquido y golpeaba una barra, de ahí el término para denominarlo en EE.UU."clicker" (*chasqueador*). Cada barra emitía una frecuencia diferente y los circuitos en el televisor detectaban el ruido. La invención del transistor hizo posible controles electrónicos más baratos, que contenía un cristal piezoeléctrico que era alimentado por una corriente eléctrica oscilatoria a una frecuencia cercana o mayor a la del umbral superior de audición humana, aunque todavía audible para perros. El receptor contenía un micrófono unido a un circuito que estaba

configurado a la misma frecuencia. Algunos problemas de este método eran que el receptor podía ser activado accidentalmente por ruidos que ocurrieran de forma natural y, algunas personas, especialmente mujeres jóvenes, podían oír las agudas señales ultrasónicas. Hubo incluso un incidente memorable, en el cual un xilófono cambiaba los canales de ese tipo de televisores, ya que algunos de los armónicos del instrumento eran iguales a la frecuencia ultrasónica del mando a distancia.

El impulso para un tipo más complejo de control remoto vino a finales de los 70 con el desarrollo del servicio de teletexto Ceefax por BBC. La mayoría de los controles que existían por entonces tenían un número limitado de funciones, a veces sólo cuatro: canal siguiente, canal anterior, subir o bajar el volumen.

Figura 1. 1 Control remoto de un televisor Philips de 1978.

Este tipo de controles no satisfacía las necesidades de televisores con teletexto donde las páginas se identificaban con un número de tres dígitos. Un control remoto para seleccionar páginas de teletexto necesitaría botones para cada número del cero al nueve, así como otras funciones, como por ejemplo, cambiar del texto a la imagen (y viceversa) y los controles normales de un televisor: volumen, canal, brillo, intensidad del color, etc. Los primeros televisores con teletexto usaban mandos con cable para elegir las páginas, pero su uso

continuado, requerido para el teletexto, indicó rápidamente la necesidad de un dispositivo sin cables. Así que ingenieros de la BBC comenzaron conversaciones con uno o dos fabricantes de televisores, lo cual llevó a los primeros prototipos sobre 1977-78, ya pudiendo controlar un mayor número de funciones. ITT fue una de las compañías, la cual más tarde daría su nombre al Protocolo ITT de comunicaciones infrarrojas.

A principios de los años 80, cuando se desarrollaron los semiconductores para emitir y recibir radiación infrarroja, los mandos a distancia fueron gradualmente cambiando a esta tecnología que, en 2006, todavía es ampliamente usada. También existen tecnologías de radio, como los Sistemas de audio Bose y aquellas basadas en Bluetooth.

A principios de los años 2000, la cantidad de electrodomésticos que hay en la mayoría de los hogares había aumentado notablemente. De acuerdo con la Asociación de electrónica de consumo, el americano medio dispone de cuatro mandos a distancia. Para manejar un *"home theater"* se pueden llegar a necesitar seis mandos, incluidos uno para el receptor del cable o satélite, el video, el reproductor de DVD, el televisor y el amplificador de audio. Se necesita usar varios de estos mandos de manera secuencial, pero, como no existe un diseño común aceptado, el proceso se hace más intrincado. Muchos especialistas, como Jakob Nielsen, renombrado especialista en usabilidad, y Robert Adler, el inventor del mando actual, señalan lo confuso, difícil de manejar y frustrante que se ha convertido lidiar con múltiples mandos a distancia. En ese sentido, los diseñadores de mando a distancia de TiVo han reemplazado las clásicas columnas de botones en un rectángulo negro por un diseño en forma de maní que ha sido bien recibida por sus usuarios. El diseño, que ha provocado varias imitaciones, probablemente signifique un cambio en la forma en la que los diseñadores de aparatos electrónicos ven un control.

1.1.2 TIPOS DE MANDOS A DISTANCIA

Existen mandos o radiocontroles para muchos otros dispositivos: modelos a escala de aviones, helicópteros, y otros modelos por radiocontrol son juguetes bastante populares. Muchos robots se controlan remotamente, especialmente aquellos que han sido diseñados para llevar a cabo tareas peligrosas; así como algunos de los más nuevos cazas de combate se maniobran por control remoto.

Figura 1.2 Mando a distancia para un temporizador fotográfico.

Además, un mando universal combina diversos controles en uno, normalmente con alguna clase de interruptor o botón para seleccionar el aparato controlado. Los mandos universales varían desde modelos básicos baratos a un mando como el modelo de 700$ con Linux de Sony. El primer modelo de control remoto universal fue desarrollado por William Russell McIntyre a mediados de los 60, mientras trabajaba en Philips. Al diseño del software de McIntyre se le fueron otorgadas patentes, ya que fue el primer control remoto que podía apuntarse a un aparato electrónico y aprender sus controles operativos.

El siguiente paso en los mandos a distancia son los paneles de control doméstico. Estos controles remotos no sólo funcionan en televisores o sistemas de entretenimiento, sino que permiten controlar otros aparatos eléctricos tales como cortinas electrónicas, interruptores de la luz y cámaras de seguridad. Algunos de

los últimos paneles de control domésticos permiten la transmisión de audio así como tomar fotografías.

A veces se usan armas de fuego por control remoto para cazar pájaros y otros animales. En 2005, el estado de Virginia, en Estados Unidos, prohibió esta práctica.

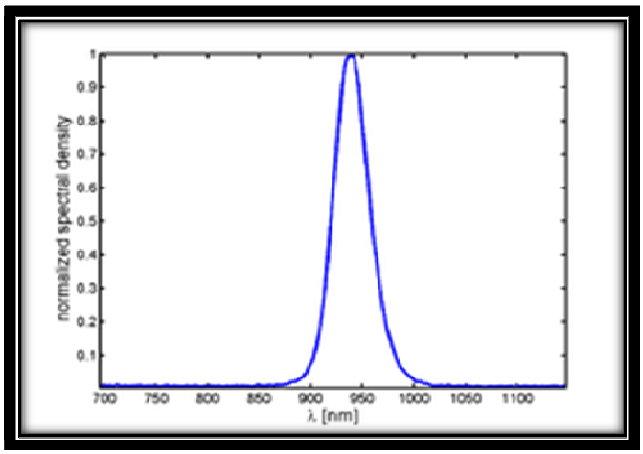


Figura 1.3 El espectro de emisión de un típico mando es cercano al infrarrojo.

La mayoría de mandos a distancia para aparatos domésticos utilizan diodos de emisión cercana a infrarrojo para emitir un rayo de luz que alcance el dispositivo. Esta luz es invisible para el ojo humano, pero transporta señales que pueden ser detectadas por el aparato.

Un mando a distancia de un sólo canal permite enviar una señal portadora, usada para accionar una determinada función. Para controles remoto multicanales, se necesitan procedimientos más sofisticados; uno de ellos consiste en modular la señal portadora con señales de diferente frecuencia. Después de la demodulación de la señal recibida, se aplican los filtros de frecuencia apropiados para separar las señales respectivas. Hoy en día, se suelen usar métodos digitales.

Por lo general un mando a distancia está compuesto por:

- **Una carcasa**.
- **Un circuito impreso** donde se encuentran los componentes electrónicos.
- **Una fuente de alimentación**, generalmente dos baterías de 1,5 voltios.
- **Una botonera**.

1.1.3 FUNCIONAMIENTO

Los botones tienen en su parte posterior un material que conduce la electricidad. Cuando se presiona el botón, este material hace contacto con el circuito impreso y cierra el circuito que corresponde al botón. Un pequeño circuito integrado reconoce la señal y determina qué botón fue presionado; con base a esa información envía una señal al resonador de cuarzo (cristal); éste la devuelve con una frecuencia determinada. Ese impulso es transmitido a un LED que lo envía convertido en radiación infrarroja. El receptor (por ejemplo, un televisor) puede reconocer el botón pulsado midiendo la frecuencia de la radiación.

1.1.4 APLICACIONES

- **Industria**

El control remoto es usado para operar subestaciones, centrales hidroeléctricas reversibles y plantas HVDC. Para estos sistemas se suelen usar PLCs de baja frecuencia.

- **Aplicaciones militares**

Figura 1.4 El astronauta Leroy Chiao manipula el *Canadarm2* o Sistema de manipulación remota de la Estación Espacial Internacional (SSRMS).

El uso control remoto de vehículos militares data de comienzos del siglo XX. El Ejército rojo usaba tele tanques, controlados remotamente, durante los años 1930 y los comienzos de la Segunda Guerra Mundial. También experimentaron con aviones por control remoto.

- **Astronáutica**

La tecnología por control remoto también es usada en los viajes al espacio. Por ejemplo, en el programa Ruso Lunokhod, los vehículos eran accionados por control remoto. El control remoto directo de naves, carros y aparatos espaciales a mayores distancias desde la tierra no era práctico ya que se generaba un gran retardo de señal.

1.2 MODULOS INALÁMBRICOS XBEE[1]

Figura 1.5 Modulo Inalámbrico XBEE

1.2.1 DEFINICIÓN

XBee es el nombre comercial de Digi International para una familia de módulos de factor de forma compatible con la radio. Las radios XBee primero se introdujeron bajo la marca MaxStream en 2005 y se basa en el estándar 802.15.4-2003 diseñado para punto a punto y comunicaciones punto a multipunto con velocidades de 250kbps.

Dos modelos se introdujeron inicialmente, la primera con menor potencia XBee 1mW y la de mayor potencia de 100mW XBee-PRO.

Desde la introducción inicial, una serie de nuevas radios XBee se han introducido y todos los XBees están comercializados y vendidos bajo la marca Digi.

Las radios XBee se pueden usar con conexiones mínimas de alimentación (3,3 V), GND, DOUT y DIN (UART), con otras líneas que se recomienda Reset y Sleep. Además, las familias XBee poseen mayor control de flujo I / O, A / D y líneas del indicador

[1]http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/xbee-series1-module.jsp#overview

construido adentro una versión de la XBees llamado XBee programable tiene un procesador adicional a bordo para el código de usuario. El XBee programables y una nueva superficie (SMT), la versión de las radios XBee fueron introducidas en 2010.

A partir de mayo de 2011, la familia de radio XBee consiste en:

- XBee 802.15.4 (también conocida como la Serie 1 de hardware) - El primer punto a punto (PTP), punto a multipunto (PTM) de radio que ejecuta el protocolo IEEE 802.15.4
- XBee-PRO 802.15.4 (también conocido como Serie 1) - Una versión de mayor potencia.
- ZB XBee (también conocido como la serie 2) - Un módulo XBee que incorpora el ZigBee PRO Feature Set protocolo de red de malla
- XBee-PRO ZB (también conocido como la serie 2) - Una versión de mayor potencia
- XBee ZB SMT-A XBee de montaje en superficie que ejecute el protocolo ZigBee
- XBee-PRO ZB SMT - Una versión de mayor potencia XBee SE-Un módulo XBee ZB, que incorpora el grupo de seguridad del perfil de ZigBee Smart Energy pública
- XBee PRO SE - Una versión de mayor potencia
- XBee PRO 900 - A 900 MHz de propiedad de PTP y radio PTM
- XBee PRO 868 - 868 MHz La propiedad PTP y PTM radio para Europa

- XBee PRO XSC - Una radio de 900 MHz compatible a través del aire con la MaxStream (ahora Digi) radios xstream
- XBee-PRO DigiMesh 900 - un radio de 900 MHz con la adición de un protocolo de red de malla para sleep (DigiMesh)
- XBee DigiMesh 2.4 - Igual que el anterior, pero a 2,4 GHz XBee-PRO DigiMesh 2.4 - Una versión de mayor potencia.

1.2.2 LAS ANTENAS Y LOS MODOS DE DATOS

La mayoría de los XBees vienen con varias opciones de antena, aunque no todas las variantes tienen exactamente los mismos conectores. El conector U. FL es común a todas las familias, y todas las variantes vienen con una antena de chip o una antena integrada, PCB rastro. Otras antenas incluyen un látigo integrado de ¼ de onda y un conector para antenas RPSMA mamparo

El XBees puede funcionar en un modo transparente de datos o en un paquete en función de la API (Interfaz de Aplicación programadores) de modo. En los datos de modo transparente entrada en la DIN (datos de) pin se transmiten directamente a las radios intención de recibir, sin ninguna modificación. Paquete de entrada puede dirigirse directamente a un objetivo o de difusión a varios objetivos. Esta modalidad se utiliza principalmente en los casos en que un protocolo existente no puede tolerar los cambios en el formato de datos. Los comandos AT se usan para controlar la configuración de la radio. En el modo API de los datos se envuelve en una estructura de paquetes que permite abordar, de ajuste de parámetros y la retroalimentación de entrega de paquetes.

1.2.2.1 AUI (Conector para antena externa)

Los módulos XBee nos ayuda con conectividad inalámbrica ya que proveen 2 formas amigables de comunicación: Transmisión serial transparente (modo AT) y el modo API que provee muchas ventajas. Los módulos Xbee pueden ser configurados desde el PC utilizando el programa X-CTU o bien desde tu microcontrolador. Los Xbee pueden comunicarse en arquitecturas punto a punto, punto a multi punto o en una red mesh. La elección del módulo XBee correcto pasa por escoger el tipo de antena (chip, alambre o conector SMA) y la potencia de transmisión (2mW para 300 pies o 60mW para hasta 1 milla).

Los módulos Xbee pueden ser utilizados con adaptadores Xbee Explorer Serial o Xbee Explorer USB. Aquellos microntroladores que trabajan con 5V necesitarán de una interfaz (Xbee regulated) para comunicarse con los módulos XBee.

Figura 1.6 Adaptadores Xbee Explorer Serial

Debido a que los módulos Xbee tienen una separación de pines de 2mm se puede utilizar tarjetas adaptadoras. Estas tarjetas permiten conectar los módulos Xbee en cualquier protoboard estándar con separación de 0.1 pulgadas. Si se va a comunicar un módulo Xbee con un PC se aconsejable utilizar el adaptador

USB el cual permitir configurar el módulo XBee fácilmente y probar la configuración antes de utilizar el módulo en una red punto a punto.

Los módulos Xbee son económicos, poderosos y fáciles de utilizar. Algunas sus principales características son:

- Buen Alcance: hasta 300ft (100 mts) en línea vista para los módulos Xbee y hasta 1 milla (1.6 Km) para los módulos Xbee Pro.
- 9 entradas/salidas con entradas analógicas y digitales.
- Bajo consumo <50mA cuando están en funcionamiento y <10uA cuando están en modo sleep.
- Interfaz serial.
- 65,000 direcciones para cada uno de los 16 canales disponibles. Se pueden tener muchos de estos dispositivos en una misma red.
- Fáciles de integrar.

Existen 2 series de estos módulos. La serie 1 y la serie 2 o también conocida como 2.5. Los módulos de la Serie 1 y la Serie 2 tienen el mismo pin-out, sin embargo, NO son compatibles entre sí ya que utilizan distintos chipset y trabajan con protocolos diferentes.

La serie 1 está basada en el chipset Freescale y está pensado para ser utilizado en redes punto a punto y punto a multipunto. Los módulos de la serie 2 están basados en el chipset de Ember y están diseñados para ser utilizados en aplicaciones que requieren repetidores o una red mesh. Ambos módulos pueden ser utilizados en los modos AT y API.

1.2.3 CARACTERÍSTICAS

MaxStream fabrica más de 70 tipos de módulos Xbee con diferentes antenas, potencia y capacidades.

Muchas de las características de los módulos Xbee tales como velocidad de transmisión y canales por ejemplo pueden ser configurados utilizando el software X-CTU o directamente desde un microcontrolador.

Los módulos Xbee son económicos, poderosos y fáciles de utilizar. Algunas sus principales características son:

- Buen Alcance: hasta 300ft (100 mts) en línea vista para los módulos Xbee y hasta 1 milla (1.6 Km) para los módulos Xbee Pro.
- 9 entradas/salidas con entradas analógicas y digitales.
- Bajo consumo <50mA cuando están en funcionamiento y <10uA cuando están en modo sleep.
- Interfaz serial.
- 65,000 direcciones para cada uno de los 16 canales disponibles. Se pueden tener muchos de estos dispositivos en una misma red.
- Fáciles de integrar.

Existen 2 series de estos módulos. La serie 1 y la serie 2 o también conocida como 2.5. Los módulos de la Serie 1 y la Serie 2 tienen el mismo pin-out, sin embargo, NO son compatibles entre sí ya que utilizan distintos chipset y trabajan con protocolos diferentes.

La serie 1 está basada en el chipset Freescale y está pensado para ser utilizado en redes punto a punto y punto a multipunto. Los módulos de la serie 2 están basados en el chipset de Ember y están diseñados para ser utilizados en

aplicaciones que requieren repetidores o una red mesh. Ambos módulos pueden ser utilizados en los modos AT y API.

1.2.4 APLICACIONES COMUNES:

- Sistemas de Seguridad & Controles de Iluminación
- Automatización de Casas (DOMOTICA)
- Aparatos domésticos & Alarmas de Incendio/CO2
- Monitorización de sistemas remotos
- Colección de datos de un sensor en sistemas embebidos.

1.3 XBEE-PRO [2]

Figura 1.7 Modulo XBEE-PRO

Este modulo no necesita configuración para las comunicaciones fuera de la caja de RF

Huella XBee común para una variedad de módulos de RF

Rápido 250 kbps RF velocidad de datos para el nodo final 2,4 GHz para el despliegue en todo el mundo

Los modos de suspensión compatible con la vida de la batería

XBee y XBee-PRO 802.15.4 OEM RF módulos son soluciones integradas de servicios celulares de punto final de conectividad a los dispositivos. Estos módulos utilizan el protocolo IEEE 802.15.4 de redes para un rápido punto-a-multipunto o de redes peer-to-peer. Están diseñados para aplicaciones de alto rendimiento que requieren baja latencia y tiempo de comunicación predecibles.

[2]http://www.xbee.cl/index.html

1.4 MICROCONTROLADOR [3]

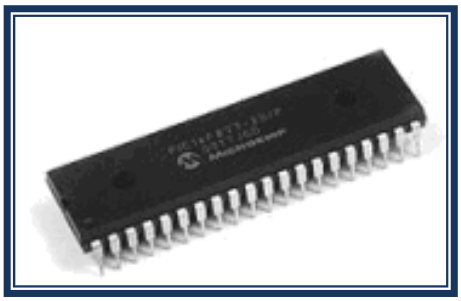


Figura 1.8 El microcontrolador.

Un **microcontrolador** es un circuito integrado que incluye en su interior las tres unidades funcionales de una computadora: unidad central de procesamiento, memoria y periféricos de entrada y salida.

1.4.1 CARACTERÍSTICAS

Son diseñados para reducir el costo económico y el consumo de energía de un sistema en particular. Por eso el tamaño de la unidad central de procesamiento, la cantidad de memoria y los periféricos incluidos dependerán de la aplicación. El control de un electrodoméstico sencillo como una batidora, utilizará un procesador muy pequeño (4 u 8 bit) por que sustituirá a un autómata finito. En cambio un reproductor de música y/o vídeo digital (mp3 o mp4) requerirá de un procesador de 32 bit o de 64 bit y de uno o más Códec de señal digital (audio y/o vídeo). El control de un sistema de frenos ABS (Antilock Brake System) se basa normalmente en un microcontrolador de 16 bit, al igual que el sistema de control electrónico del motor en un automóvil.

[3] **http://es.wikipedia.org/wiki/Microcontrolador**

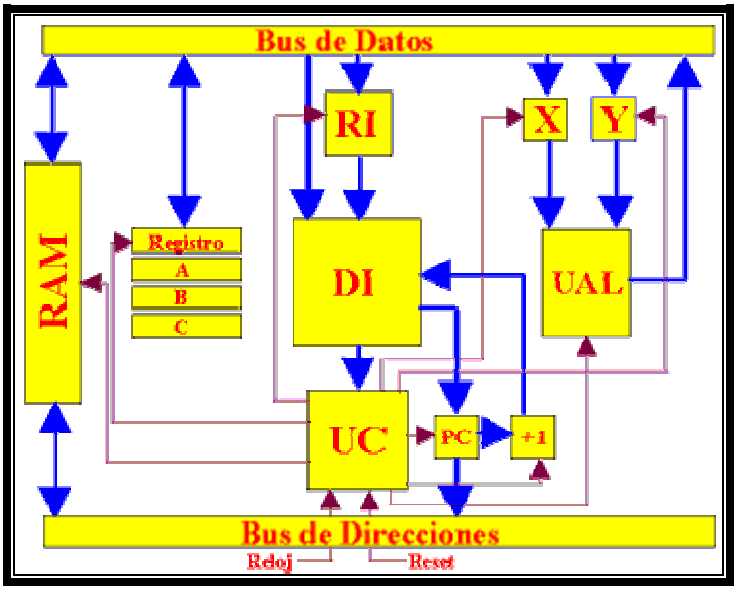


Figura 1.9 Esquema de un Microcontrolador.

Los microcontroladores representan la inmensa mayoría de los chips de computadoras vendidos, pueden encontrarse en casi cualquier dispositivo electrónico como automóviles, lavadoras, hornos microondas, teléfonos, etc.

Un microcontrolador difiere de una unidad central de procesamiento normal, debido a que es más fácil convertirla en una computadora en funcionamiento, con un mínimo de circuitos integrados externos de apoyo. La idea es que el circuito integrado se coloque en el dispositivo, enganchado a la fuente de energía y de información que necesite, y eso es todo. Un microprocesador tradicional no le permitirá hacer esto, ya que espera que todas estas tareas sean manejadas por otros chips. Hay que agregarle los modulos de entrada y salida (puertos) y la memoria para almacenamiento de información.

Por ejemplo, un microcontrolador típico tendrá un generador de reloj integrado y una pequeña cantidad de memoria de acceso aleatorio y/o ROM/EPROM/EEPROM/flash, significando que para hacerlo funcionar, todo lo que

se necesita son unos pocos programas de control y un cristal de sincronización. Los microcontroladores disponen generalmente también de una gran variedad de dispositivos de entrada/salida, como convertidores de analógico a digital, temporizadores, UARTs y buses de interfaz serie especializados, como I^2C y CAN. Frecuentemente, estos dispositivos integrados pueden ser controlados por instrucciones de procesadores especializados. Los modernos microcontroladores frecuentemente incluyen un lenguaje de programación integrado, como el lenguaje de programación BASIC que se utiliza bastante con este propósito.

Los microcontroladores negocian la velocidad y la flexibilidad para facilitar su uso. Debido a que se utiliza bastante sitio en el chip para incluir funcionalidad, como los dispositivos de entrada/salida o la memoria que incluye el microcontrolador, se ha de prescindir de cualquier otra circuitería.

1.4.2 ESTRUCTURA BÁSICA DE UN MICROCONTROLADOR

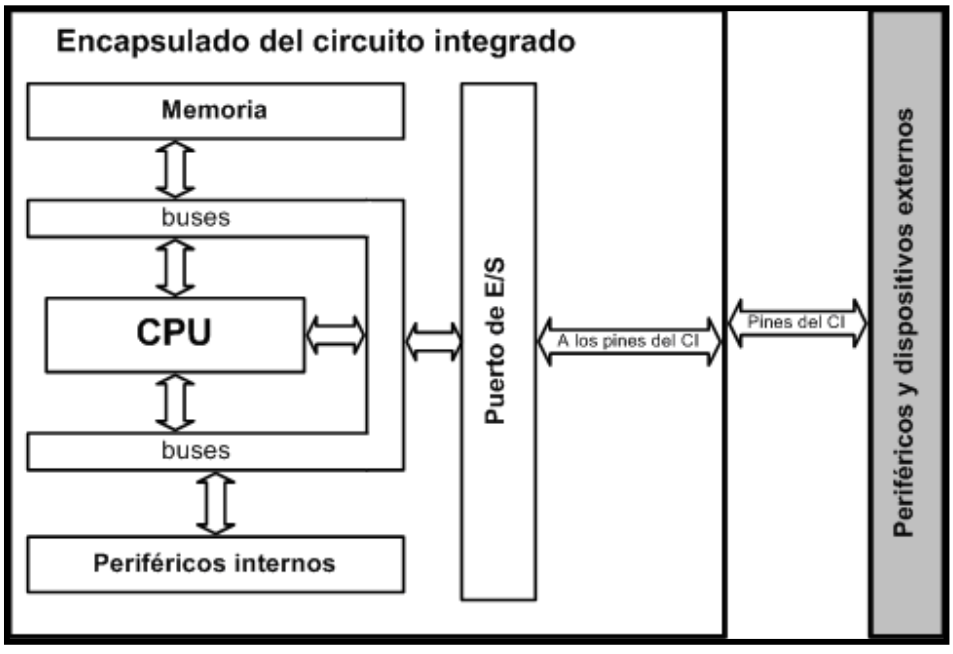


Figura 1.10 Estructura básica de un microcontrolador

En la figura 1.10 vemos al microcontrolador dentro de un encapsulado de circuito integrado, con su procesador (CPU), buses, memoria, periféricos y puertos de entrada/salida. Fuera del encapsulado se ubican otros circuitos para completar periféricos internos y dispositivos que pueden conectarse a los pines de entrada/salida. También se conectarán a los pines del encapsulado la alimentación, masa, circuito de completamiento del oscilador y otros circuitos necesarios para que el microcontrolador pueda trabajar.

1.4.3 ARQUITECTURAS DE COMPUTADORA

Básicamente existen dos arquitecturas de computadoras, y por supuesto, están presentes en el mundo de los microcontroladores: Von Neumann y Harvard. Ambas se diferencian en la forma de conexión de la memoria al procesador y en los buses que cada una necesita.

La arquitectura Von Neumann es la que se utiliza en las computadoras personales, para ella existe una sola memoria, donde coexisten las instrucciones de programa y los datos, accedidos con un bus de dirección, uno de datos y uno de control.

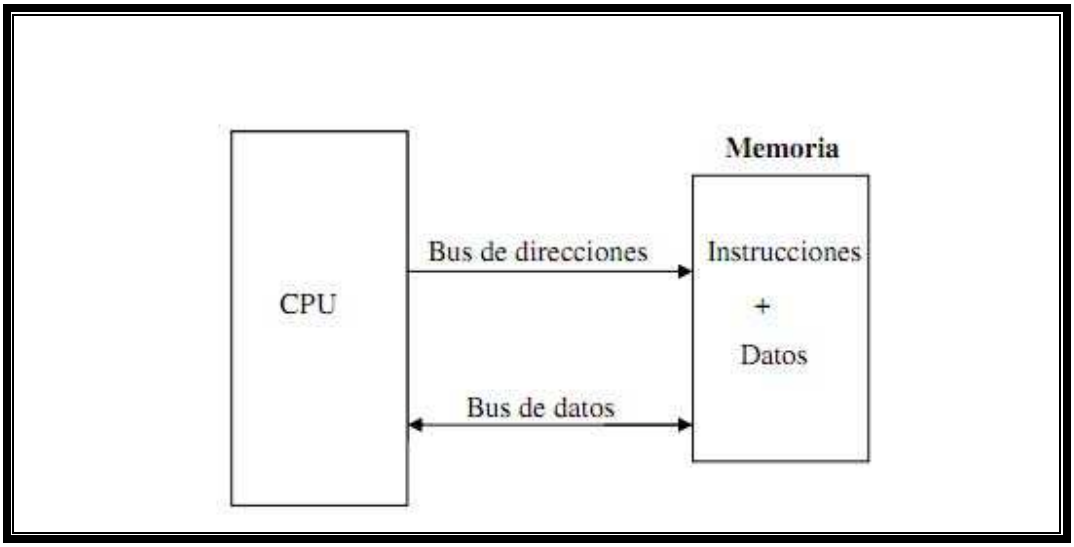


Figura 1.11 Arquitectura de un computador Von Neumann

Debemos comprender que en una PC, cuando se carga un programa en memoria, a éste se le asigna un espacio de direcciones de la memoria que se divide en segmentos, de los cuales típicamente tenderemos los siguientes: código (programa), datos y pila. Es por ello que podemos hablar de la memoria como un todo, aunque existan distintos dispositivos físicos en el sistema (HDD, RAM, CD, FLASH).

En el caso de los microcontroladores, existen dos tipos de memoria bien definidas: memoria de datos (típicamente algún tipo de SRAM) y memoria de programas (ROM, PROM, EEPROM, flash u de otro tipo no volátil). En este caso la organización es distinta a las de las PC, porque hay circuitos distintos para cada memoria y normalmente no se utilizan los registros de segmentos, sino que la memoria está segregada y el acceso a cada tipo de memoria depende de las instrucciones del procesador.

A pesar de que en los sistemas integrados con arquitectura Von Neumann la memoria esté segregada, y existan diferencias con respecto a la definición tradicional de esta arquitectura; los buses para acceder a ambos tipos de memoria son los mismos, del procesador solamente salen el bus de datos, el de direcciones, y el de control. Como conclusión, la arquitectura no ha sido alterada, porque la forma en que se conecta la memoria al procesador sigue el mismo principio definido en la arquitectura básica.

Esta arquitectura es la variante adecuada para las PC, porque permite ahorrar una buena cantidad de líneas de E/S, que son bastante costosas, sobre todo para aquellos sistemas como las PC, donde el procesador se monta en algún tipo de zócalo alojado en una placa madre. También esta organización les ahorra a los diseñadores de motherboards (tarjeta madre) una buena cantidad de problemas y reduce el costo de este tipo de sistemas.

Algunas familias de microcontroladores como la INTEL-51 y la Z80 implementan este tipo de arquitectura, fundamentalmente porque era la utilizada cuando aparecieron los primeros microcontroladores.

La otra variante es la arquitectura Harvard, y por excelencia la utilizada en supercomputadoras, en los microcontroladores, y sistemas integrados en general. En este caso, además de la memoria, el procesador tiene los buses segregados, de modo que cada tipo de memoria tiene un bus de datos, uno de direcciones y uno de control.

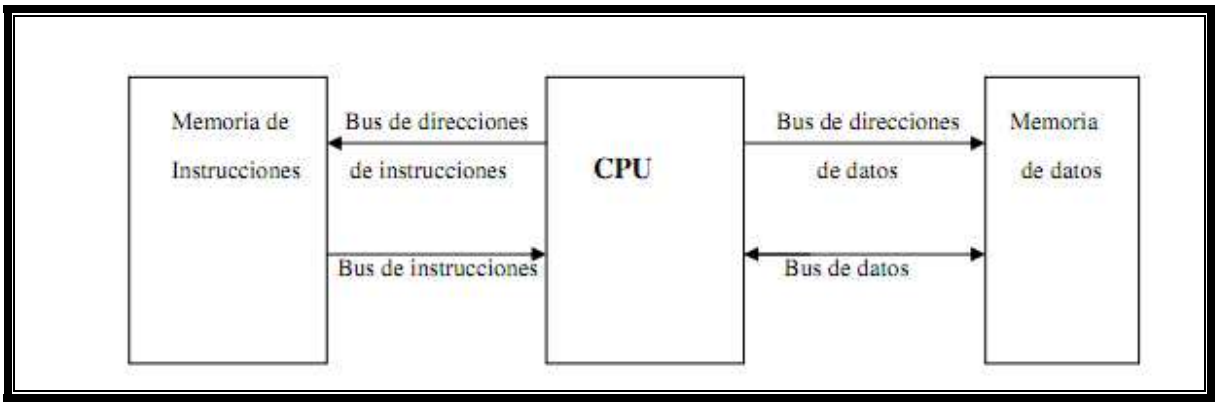


Figura 1.12 Arquitectura de un computador Harvard

La ventaja fundamental de esta arquitectura es que permite adecuar el tamaño de los buses a las características de cada tipo de memoria; además, el procesador puede acceder a cada una de ellas de forma simultánea, lo que se traduce en un aumento significativo de la velocidad de procesamiento, típicamente los sistemas con esta arquitectura pueden ser dos veces más rápidos que sistemas similares con arquitectura Von Neumann.

La desventaja está en que consume muchas líneas de E/S del procesador; por lo que en sistemas donde el procesador está ubicado en su propio encapsulado, solo se utiliza en supercomputadoras. Sin embargo, en los microcontroladores y otros sistemas integrados, donde usualmente la memoria de datos y programas comparten el mismo encapsulado que el procesador, este inconveniente deja de

ser un problema serio y es por ello que encontramos la arquitectura Harvard en la mayoría de los microcontroladores.

Por eso es importante recordar que un microcontrolador se puede configurar de diferentes maneras, siempre y cuando se respete el tamaño de memoria que este requiera para su correcto funcionamiento.

1.4.3.1 Registros

Son un espacio de memoria muy reducido pero necesario para cualquier microprocesador, de aquí se toman los datos para varias operaciones que debe realizar el resto de los circuitos del procesador. Los registros sirven para almacenar los resultados de la ejecución de instrucciones, cargar datos desde la memoria externa o almacenarlos en ella.

Mientras mayor sea el número de bits de los registros de datos del procesador, mayores serán sus prestaciones, en cuanto a poder de cómputo y velocidad de ejecución, ya que este parámetro determina la potencia que se puede incorporar al resto de los componentes del sistema, por ejemplo, no tiene sentido tener una ALU de 16 bits en un procesador de 8 bits.

1.4.3.2 Unidad de control

Esta unidad es de las más importantes en el procesador, en ella recae la lógica necesaria para la decodificación y ejecución de las instrucciones, el control de los registros, la ALU, los buses y cuanta cosa más se quiera meter en el procesador.

La unidad de control es uno de los elementos fundamentales que determinan las prestaciones del procesador, ya que su tipo y estructura, determina parámetros tales como el tipo de conjunto de instrucciones, velocidad de ejecución, tiempo del ciclo de máquina, tipo de buses que puede tener el sistema, manejo de

interrupciones y un buen número de cosas más que en cualquier procesador van a parar a este bloque.

Por supuesto, las unidades de control, son el elemento más complejo de un procesador y normalmente están divididas en unidades más pequeñas trabajando de conjunto. La unidad de control agrupa componentes tales como la unidad de decodificación, unidad de ejecución, controladores de memoria cache, controladora de buses, controladora de interrupción, entre otros elementos, dependiendo siempre del tipo de procesador.

- **Unidad aritmético-lógica**

Como los procesadores son circuitos que hacen básicamente operaciones lógicas y matemáticas, se le dedica a este proceso una unidad completa, con cierta independencia. Aquí es donde se realizan las sumas, restas, y operaciones lógicas típicas del álgebra de Boole.

Actualmente este tipo de unidades ha evolucionado mucho y los procesadores más modernos tienen varias ALU, especializadas en la realización de operaciones complejas como las operaciones en coma flotante. De hecho en muchos casos le han cambiado su nombre por el de “coprocesador matemático”, aunque este es un término que surgió para dar nombre a un tipo especial de procesador que se conecta directamente al procesador más tradicional.

Su impacto en las prestaciones del procesador es también importante porque, dependiendo de su potencia, tareas más o menos complejas, pueden hacerse en tiempos muy cortos, como por ejemplo, los cálculos en coma flotante.

1.4.3.3 Buses

Son el medio de comunicación que utilizan los diferentes componentes del procesador para intercambiar información entre sí, eventualmente los buses o una parte de ellos estarán reflejados en los pines del encapsulado del procesador.

En el caso de los microcontroladores, no es común que los buses estén reflejados en el encapsulado del circuito, ya que estos se destinan básicamente a las E/S de propósito general y periféricos del sistema.

Existen tres tipos de buses:

- Dirección: Se utiliza para seleccionar al dispositivo con el cual se quiere trabajar o en el caso de las memorias, seleccionar el dato que se desea leer o escribir.
- Datos.
- Control: Se utiliza para gestionar los distintos procesos de escritura lectura y controlar la operación de los dispositivos del sistema.

1.4.3.4 Conjunto de instrucciones

Se define a las operaciones básicas que puede realizar el procesador, que conjugadas y organizadas forman lo que conocemos como software. El conjunto de instrucciones vienen siendo como las letras del alfabeto, el elemento básico del lenguaje, que organizadas adecuadamente permiten escribir palabras, oraciones y cuanto programa se le ocurra.

Existen dos tipos básicos de repertorios de instrucciones, que determinan la arquitectura del procesador: CISC y RISC.

CISC, del inglés Complex Instruction Set Computer, Computadora de Conjunto de Instrucciones Complejo. Los microprocesadores CISC tienen un conjunto de instrucciones que se caracteriza por ser muy amplio y que permiten realizar operaciones complejas entre operandos situados en la memoria o en los registros internos. Este tipo de repertorio dificulta el paralelismo entre instrucciones, por lo que en la actualidad, la mayoría de los sistemas CISC de alto rendimiento,

convierten las instrucciones complejas en varias instrucciones simples del tipo RISC, llamadas generalmente microinstrucciones.

Dentro de los microcontroladores CISC podemos encontrar a la popular familia INTEL -51 y la Z80, aunque actualmente existen versiones CISC-RISC de estos microcontroladores, que pretenden aprovechar las ventajas de los procesadores RISC a la vez que se mantiene la compatibilidad hacia atrás con las instrucciones de tipo CISC.

RISC, del inglés Reduced Instruction Set Computer, Computadora con Conjunto de Instrucciones Reducido. Se centra en la obtención de procesadores con las siguientes características fundamentales:

- Instrucciones de tamaño fijo.
- Pocas instrucciones.
- Sólo las instrucciones de carga y almacenamiento acceden a la memoria de datos.
- Número relativamente elevado de registros de propósito general.

Una de las características más destacables de este tipo de procesadores es que posibilitan el paralelismo en la ejecución, y reducen los accesos a memoria. Es por eso que los procesadores más modernos, tradicionalmente basados en arquitecturas CISC implementan mecanismos de traducción de instrucciones CISC a RISC, para aprovechar las ventajas de este tipo de procesadores.

Los procesadores de los microcontroladores PIC son de tipo RISC.

1.4.3.5 Memoria

La memoria en los microcontroladores debe estar ubicada dentro del mismo encapsulado, esto es así la mayoría de las veces, porque la idea fundamental es mantener el grueso de los circuitos del sistema dentro de un solo integrado.

En los microcontroladores la memoria no es abundante, aquí no encontrará Gigabytes de memoria como en las computadoras personales. Típicamente la memoria de programas no excederá de 16 K-localizaciones de memoria no volátil para La memoria RAM está destinada al almacenamiento de información temporal que será utilizada por el procesador para realizar cálculos u otro tipo de operaciones lógicas. En el espacio de direcciones de memoria RAM se ubican además los registros de trabajo del procesador y los de configuración y trabajo de los distintos periféricos del microcontrolador. Es por ello que en la mayoría de los casos, aunque se tenga un espacio de direcciones de un tamaño determinado, la cantidad de memoria RAM de que dispone el programador para almacenar sus datos es menor que la que puede direccionar el procesador.

El tipo de memoria utilizada en las memorias RAM de los microcontroladores es SRAM, lo que evita tener que implementar sistemas de refrescamiento como en el caso de las computadoras personales, que utilizan gran cantidad de memoria, típicamente alguna tecnología DRAM. A pesar de que la memoria SRAM es más costosa que la DRAM, es el tipo adecuado para los microcontroladores porque éstos poseen pequeñas cantidades de memoria RAM.

En el caso de la memoria de programas se utilizan diferentes tecnologías, y el uso de una u otra depende de las características de la aplicación a desarrollar, a continuación se describen las cinco tecnologías existentes, que mayor utilización tienen o han tenido:

- ROM de máscara. En este caso no se “graba” el programa en memoria sino que el microcontrolador se fabrica con el programa, es un proceso similar al de producción de los CD comerciales mediante masterización.

El costo inicial de producir un circuito de este tipo es alto, porque el diseño y producción de la máscara es un proceso costoso, sin embargo, cuando se necesitan varios miles o incluso cientos de miles de microcontroladores para una aplicación determinada, como por ejemplo, algún electrodoméstico, el costo inicial

de producción de la máscara y el de fabricación del circuito se distribuye entre todos los circuitos de la serie y, el costo final de ésta, es bastante menor que el de sus semejantes con otro tipo de memoria.

- **OTP One Time Programmable**. Este tipo de memoria, también es conocida como PROM o simplemente ROM.

Los microcontroladores con memoria OTP se pueden programar una sola vez, con algún tipo de programador. Se utilizan en sistemas donde el programa no requiera futuras actualizaciones y para series relativamente pequeñas, donde la variante de máscara sea muy costosa, también para sistemas que requieren serialización de datos, almacenados como constantes en la memoria de programas.

- **EPROM Erasable Programmable Read Only Memory.**

Los microcontroladores con este tipo de memoria son muy fáciles de identificar porque su encapsulado es de cerámica y llevan encima una ventanita de vidrio desde la cual puede verse la oblea de silicio del microcontrolador.

Se fabrican así porque la memoria EPROM es reprogramable, pero antes debe borrase, y para ello hay que exponerla a una fuente de luz ultravioleta, el proceso de grabación es similar al empleado para las memorias OTP. Al aparecer tecnologías menos costosas y más flexibles, como las memorias EEPROM y FLASH, este tipo de memoria han caído en desuso, se utilizaban en sistemas que requieren actualizaciones del programa y para los procesos de desarrollo y puesta a punto.

- **EEPROM Electrical Erasable Programmable Read Only Memory**. Fueron el sustituto natural de las memorias EPROM, la diferencia fundamental es que pueden ser borradas eléctricamente, por lo que la

ventanilla de cristal de cuarzo y los encapsulados cerámicos no son necesarios.

Al disminuir los costos de los encapsulados, los microcontroladores con este tipo de memoria se hicieron más baratos y cómodos para trabajar que sus equivalentes con memoria EPROM. Otra característica destacable de este tipo de microcontrolador es que fue en ellos donde comenzaron a utilizarse los sistemas de programación en circuito o ICSP (In Circuit Serial Progamming) que evitan tener que sacar el microcontrolador de la tarjeta que lo aloja para hacer actualizaciones al programa.

- **FLASH.** En el campo de las memorias reprogramables para microcontroladores, son el último avance tecnológico en uso a gran escala, y han sustituido a los microcontroladores con memoria EEPROM.

A las ventajas de las memorias flash se le adicionan su gran densidad respecto a sus predecesoras lo que permite incrementar la cantidad de memoria de programas a un costo muy bajo. Pueden además ser programadas con las mismas tensiones de alimentación del microcontrolador, el acceso en lectura y la velocidad de programación es superior, disminución de los costos de producción, entre otras. Lo más habitual es encontrar que la memoria de programas y datos está ubicada toda dentro del microcontrolador, de hecho, actualmente son pocos los microcontroladores que permiten conectar memoria de programas en el exterior del encapsulado. Las razones para estas “limitaciones” están dadas porque el objetivo fundamental es obtener la mayor integración posible y conectar memorias externas consume líneas de E/S que son uno de los recursos más preciados de los microcontroladores. A pesar de lo anterior existen familias como la INTEL 51 cuyos microcontroladores tienen la capacidad de ser expandidos en una variada gama de configuraciones para el uso de memoria de programas externa. En el caso de los PIC, estas posibilidades están limitadas sólo a algunos microcontroladores de la gama alta.

Cuando se requiere aumentar la cantidad de memoria de datos, lo más frecuente es colocar dispositivos de memoria externa en forma de periféricos, de esta forma se pueden utilizar memorias RAM, FLASH o incluso discos duros como los de las PC, mientras que para los cálculos y demás operaciones que requieran almacenamiento temporal de datos se utiliza la memoria RAM interna del microcontrolador. Esta forma de expandir la memoria de datos está determinada, en la mayoría de lo casos, por el tipo de repertorio de instrucciones del procesador y porque permite un elevado número de configuraciones distintas, además del consiguiente ahorro de líneas de E/S que se logra con el uso de memorias con buses de comunicación serie.

1.4.3.6 Interrupciones

Las interrupciones son esencialmente llamadas a subrutina generadas por los dispositivos físicos, al contrario de las subrutinas normales de un programa en ejecución. Como el salto de subrutina no es parte del hilo o secuencia de ejecución programada, el controlador guarda el estado del procesador en la pila de memoria y entra a ejecutar un código especial llamado "manejador de interrupciones" que atiende al periférico específico que generó la interrupción. Al terminar la rutina, una instrucción especial le indica al procesador el fin de la atención de la interrupción. En ese momento el controlador restablece el estado anterior, y el programa que se estaba ejecutando antes de la interrupción sigue como si nada hubiese pasado. Las rutinas de atención de interrupciones deben ser lo más breves posibles para que el rendimiento del sistema sea satisfactorio, porque normalmente cuando una interrupción es atendida, todas las demás interrupciones están en espera.

Los procesos de atención a interrupciones tienen la ventaja de que se implementan por hardware ubicado en el procesador, así que es un método rápido de hacer que el procesador se dedique a ejecutar un programa especial para atender eventos que no pueden esperar por mecanismos lentos.

En términos generales, un proceso de interrupción y su atención por parte del procesador, tiene la siguiente secuencia de acciones:

1. En el mundo real se produce el evento para el cual queremos que el procesador ejecute un programa especial, este proceso tiene la característica de que no puede esperar mucho tiempo antes de ser atendido o no sabemos en qué momento debe ser atendido.

2. El circuito encargado de detectar la ocurrencia del evento se activa, y como consecuencia, activa la entrada de interrupción del procesador.

3. La unidad de control detecta que se ha producido una interrupción y "levanta" una bandera para registrar esta situación; de esta forma si las condiciones que provocaron el evento desaparecen y el circuito encargado de detectarlo desactiva la entrada de interrupción del procesador, ésta se producirá de cualquier modo, porque ha sido registrada.

4. La unidad de ejecución termina con la instrucción en curso y justo antes de comenzar a ejecutar la siguiente comprueba que se ha registrado una interrupción

5. Se desencadena un proceso que permite guardar el estado actual del programa en ejecución y saltar a una dirección especial de memoria de programas, donde está la primera instrucción de la subrutina de atención a interrupción.

6. Se ejecuta el código de atención a interrupción, esta es la parte "consciente" de todo el proceso porque es donde se realizan las acciones propias de la atención a la interrupción y el programador juega su papel.

7. Cuando en la subrutina de atención a interrupción se ejecuta la instrucción de retorno, se desencadena el proceso de restauración del procesador al estado en que estaba antes de la atención a la interrupción.

Como podemos observar, el mecanismo de interrupción es bastante complicado, sin embargo tiene dos ventajas que obligan a su implementación: la velocidad y su capacidad de ser asincrónico. Ambas de conjunto permiten que aprovechemos al máximo las capacidades de trabajo de nuestro procesador.

Los mecanismos de interrupción no solo se utilizan para atender eventos ligados a procesos que requieren atención inmediata sino que se utilizan además para atender eventos de procesos asincrónicos.

Las interrupciones son tan eficaces que permiten que el procesador actúe como si estuviese haciendo varias cosas a la vez cuando en realidad se dedica a la misma rutina de siempre, ejecutar instrucciones una detrás de la otra.

1.4.3.7 Periféricos

Cuando observamos la organización básica de un microcontrolador, señalamos que dentro de este se ubican un conjunto de periféricos, cuyas salidas están reflejadas en los pines del microcontrolador. A continuación describiremos algunos de los periféricos que con mayor frecuencia encontraremos en los microcontroladores.

- **Entradas y salidas de propósito general**

También conocidos como puertos de E/S, generalmente agrupadas en puertos de 8 bits de longitud, permiten leer datos del exterior o escribir en ellos desde el interior del microcontrolador, el destino habitual es el trabajo con dispositivos simples como relés, LED, o cualquier otra cosa que se le ocurra al programador.

Algunos puertos de E/S tienen características especiales que le permiten manejar salidas con determinados requerimientos de corriente, o incorporan mecanismos especiales de interrupción para el procesador.

Típicamente cualquier pin de E/S puede ser considerada E/S de propósito general, pero como los microcontroladores no pueden tener infinitos pines, ni siquiera todos los pines que queramos, las E/S de propósito general comparten los pines con otros periféricos. Para usar un pin con cualquiera de las características a él asignadas debemos configurarlo mediante los registros destinados a ellos.

- **Temporizadores y contadores**

Son circuitos sincrónicos para el conteo de los pulsos que llegan a su para poder conseguir la entrada de reloj. Si la fuente de un gran conteo es el oscilador interno del microcontrolador es común que no tengan un pin asociado, y en este caso trabajan como temporizadores. Por otra parte, cuando la fuente de conteo es externa, entonces tienen asociado un pin configurado como entrada, este es el modo contador.

Los temporizadores son uno de los periféricos más habituales en los microcontroladores y se utilizan para muchas tareas, como por ejemplo, la medición de frecuencia, implementación de relojes, para el trabajo de conjunto con otros periféricos que requieren una base estable de tiempo entre otras funcionalidades. Es frecuente que un microcontrolador típico incorpore más de un temporizador/contador e incluso algunos tienen arreglos de contadores. Como veremos más adelante este periférico es un elemento casi imprescindible y es habitual que tengan asociada alguna interrupción. Los tamaños típicos de los registros de conteo son 8 y 16 bits, pudiendo encontrar dispositivos que solo tienen temporizadores de un tamaño o con más frecuencia con ambos tipos de registro de conteo.

- **Conversor analógico/digital**

Como es muy frecuente el trabajo con señales analógicas, éstas deben ser convertidas a digital y por ello muchos microcontroladores incorporan un conversor

analógico-digital, el cual se utiliza para tomar datos de varias entradas diferentes que se seleccionan mediante un multiplexor.

Las resoluciones más frecuentes son 8 y 10 bits, que son suficientes para aplicaciones sencillas. Para aplicaciones en control e instrumentación están disponibles resoluciones de 12bit, 16bit y 24bit Ejemplo: microcontroladores "analógicos". También es posible conectar un convertidor externo, en caso de necesidad.

1.4.3.8 Puertos de comunicación

Este periférico está presente en casi cualquier microcontrolador, normalmente en forma de UART (Universal Asynchronous Receiver Transmitter) o USART (Universal Synchronous Asynchronous Receiver Transmitter) dependiendo de si permiten o no el modo sincrónico de comunicación.

El destino común de este periférico es la comunicación con otro microcontrolador o con una PC y en la mayoría de los casos hay que agregar circuitos externos para completar la interfaz de comunicación. La forma más común de completar el puerto serie es para comunicarlo con una PC mediante la interfaz EIA-232 (más conocida como RS-232), es por ello que muchas personas se refieren a la UART o USART como puerto serie RS-232, pero esto constituye un error, puesto que este periférico se puede utilizar para interconectar dispositivos mediante otros estándares de comunicación. En aplicaciones industriales se utiliza preferiblemente RS-485 por su superior alcance en distancia, velocidad y resistencia al ruido.

- **SPI**

Este tipo de periférico se utiliza para comunicar al microcontrolador con otros microcontroladores o con periféricos externos conectados a él, por medio de una interfaz muy sencilla. Hay solo un nodo controlador que permite iniciar cualquier

transacción, lo cual es una desventaja en sistemas complejos, pero su sencillez permite el aislamiento galvánico de forma directa por medio de optoacopladores.

- **I2C**

Cumple las mismas funciones que el SPI, pero requiere menos señales de comunicación y cualquier nodo puede iniciar una transacción. Es muy utilizado para conectar las tarjetas gráficas de las computadores personales con los monitores, para que estos últimos informen de sus prestaciones y permitir la autoconfiguración del sistema de video.

- **USB**

Los microcontroladores son los que han permitido la existencia de este sistema de comunicación. Es un sistema que trabaja por polling (monitoreo) de un conjunto de periféricos inteligentes por parte de un amo, que es normalmente un computador personal. Cada nodo inteligente está gobernado inevitablemente por un microcontrolador.

- **Ethernet**

Es el sisma más extendido en el mundo para redes de área local cableadas. Los microcontroladores más poderosos de 32 bit se usan para implementar periféricos lo suficientemente poderosos como para que puedan ser accesados directamente por la red. Muchos de los enrutadores caseros de pequeñas empresas están construidos en base a un microcontrolador que hace del cerebro del sistema.

- **Can**

Este protocolo es del tipo CSMA/CD con tolerancia a elevados voltajes de modo común y orientado al tiempo real. Este protocolo es el standar mas importante en la industria automotriz (OBD). También se usa como capa física del "field bus" para el control industrial.

1.4.3.9 Otros puertos de comunicación

Hay una enorme cantidad de otros buses disponibles para la industria automotriz (linbus) o de medios audiovisuales como el i2s, IEEE_1394. Es usuario se los encontrará cuando trabaje en algún area especializada.

- **Comparadores**

Son circuitos analógicos basados en amplificadores operacionales que tienen la característica de comparar dos señales analógicas y dar como salida los niveles lógicos ‘0’ o ‘1’ en dependencia del resultado de la comparación. Es un periférico muy útil para detectar cambios en señales de entrada de las que solamente nos interesa conocer cuando está en un rango determinado de webetas

- **Modulador de ancho de pulsos**

Los PWM (Pulse Width Modulator) son periféricos muy útiles sobre todo para el control de motores, sin embargo hay un grupo de aplicaciones que pueden realizarse con este periférico, dentro de las cuales podemos citar: inversión DC/AC para UPS, conversión digital analógica D/A, control regulado de luz (dimming) entre otras.

- **Memoria de datos no volátil**

Muchos microcontroladores han incorporado este tipo de memoria como un periférico más, para el almacenamiento de datos de configuración o de los procesos que se controlan. Esta memoria es independiente de la memoria de datos tipo RAM o la memoria de programas, en la que se almacena el código del programa a ejecutar por el procesador del microcontrolador.

1.5 MICROCONTROLADORES AVR[4]

Los AVR son una familia de microcontroladores RISC de Atmel. La arquitectura de los AVR fue concebida por dos estudiantes en el Norwegian Institute of Technology, y posteriormente refinada y desarrollada en Atmel Norway, la empresa subsidiaria de Atmel, fundada por los dos arquitectos del chip.

El AVR es una CPU de arquitectura Harvard. Tiene 32 registros de 8 bits. Algunas instrucciones sólo operan en un subconjunto de estos registros. La concatenación de los 32 registros, los registros de entrada/salida y la memoria de datos conforman un espacio de direcciones unificado, al cual se accede a través de operaciones de carga/almacenamiento. A diferencia de los microcontroladores PIC, el stack se ubica en este espacio de memoria unificado, y no está limitado a un tamaño fijo.

El AVR fue diseñado desde un comienzo para la ejecución eficiente de código C compilado. Como este lenguaje utiliza profundamente punteros para el manejo de variables en memoria, los tres últimos pares de registros internos del procesador, son usados como punteros de 16 bits al espacio de memoria externa, bajo los nombres X, Y y Z. Esto es un compromiso que se hace en arquitecturas de ocho bits desde los tiempos de Intel 8008, ya que su tamaño de palabra original de 8 bits (256 localidades accedidas) es pobre para direccionar. Por otro lado, hacer que todo el banco superior de 16 registros de 8 bits tenga un comportamiento alterno como un banco de 8 registros de 16 bits, complicaría mucho el diseño, violando la premisa original de su simplicidad. Además, algunas instrucciones tales como 'suma inmediata' faltan, ya que la instrucción 'resta inmediata' con el complemento dos puede ser usada como alternativa.

[4]http://es.wikipedia.org/wiki/AVR

El set de instrucciones AVR está implementado físicamente y disponible en el mercado en diferentes dispositivos, que comparten el mismo núcleo AVR pero tienen distintos periféricos y cantidades de RAM y ROM: desde el microcontrolador de la familia Tiny AVR ATtiny11 con 1KB de memoria flash y sin RAM (sólo los 32 registros), y 8 pines, hasta el microcontrolador de la familia Mega AVR ATMEGA 2560 con 256KB de memoria flash, 8KB de memoria RAM, 4KB de memoria EEPROM, conversor análogo digital de 10 bits y 16 canales, temporizadores, comparador analógico, JTAG, etc. La compatibilidad entre los distintos modelos es preservada en un grado razonable.

Los microcontroladores AVR tienen una cañería ('pipeline' en inglés) con dos etapas (cargar y ejecutar), que les permite ejecutar la mayoría en un ciclo de reloj, lo que los hace relativamente rápidos entre los microcontroladores de 8-bit.

El set de instrucciones de los AVR es más regular que la de la mayoría de los microcontroladores de 8-bit (por ejemplo, los PIC). Sin embargo, no es completamente ortogonal:

- Los registros punteros X, Y y Z tienen capacidades de direccionamiento diferentes entre sí.
- Los registros 0 al 15 tienen diferentes capacidades de direccionamiento que los registros 16 al 31.
- Las registros de I/O 0 al 31 tienen distintas características que las posiciones 32 al 63.
- La instrucción CLR afecta los 'flag', mientras que la instrucción SER no lo hace, a pesar de que parecen ser instrucciones complementarias (dejar todos los bits en 1, y dejar todos los bits en 0 respectivamente).
- Los códigos de operación 0x95C8 y 0x9004 hacen exactamente lo mismo (LPM).

- Como los PIC, tiene una comunidad de seguidores (ejemplificadas por el foro de internet AVRFreaks), principalmente debido a la existencia de herramientas de desarrollo gratuitas o de bajo costo. Estos microcontroladores están soportados por tarjetas de desarrollo de costo razonable, capaces de descargar el código al microcontrolador, y por una versión de las herramientas GNU. Esto último es posible por su uniformidad en el acceso al espacio de memoria, propiedad de la que carecen los procesadores de memoria segmentada o por bancos, como el PIC o el 8051 y sus derivados.

	AT89S51	*PIC16F877*	*ATMEGA16*
I/O	32	33	32
FLASH	4K	8K	16 K
RAM	128 BYTE	368 BYTE	1 KBYTE
EEPROM	ND	256 BYTE	512 BYTE
PWM	ND	2	4
ADC	ND	8 10BITS	8 10 BITS
TIMER 8BITS PRESCALER	ND	2 PRESCALER	2
TIMER 16BITS	2	1 PRESCALER	1 PRESCALER
IN CIRCUIT	SI	SI	SI
UART	1	1	1
SPI	ND	1	1
I2C	ND	1	1
CM	F/12	F/4	F
1K$	1	4	

Tabla1.1 Característica de algunos microcontroladores utilizados en el medio.

1.6 CONTROL DE MOTORES DE CORRIENTE CONTINÚA

Como ya sabes, el motor eléctrico transforma energía eléctrica en energía mecánica, gracias a la interacción de campos electromagnético

1.6.1 CONTROL DE LA MARCHA Y PARO EN UN SOLO SENTIDO

1.6.1.1 Mediante relés

En este circuito se pueden distinguir dos partes: la parte de mando, representada mediante línea fina y la parte de fuerza, dibujada con trazo grueso. La parte de mando está compuesta por un circuito de control que gobierna un relé. Este circuito de control puede ser manual (pulsador, interruptor, etc...) o automático (circuito electrónico con sensores, temporizadores etc...).

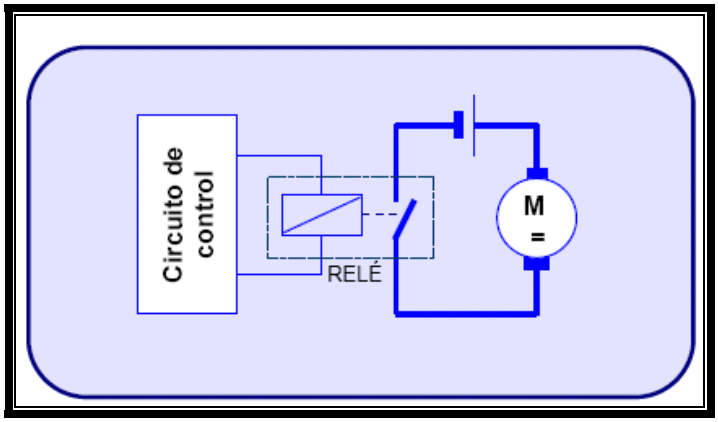


Figura 1.13 Circuito de control de la marcha y paro en un solo sentido mediante relés

1.6.1.2 Mediante un transistor de mediana potencia:

En este caso la conexión del motor la realiza un transistor adecuado a la potencia del motor. Este montaje también permite la regulación de la velocidad mediante un ajustable que regula la corriente de base del transistor.

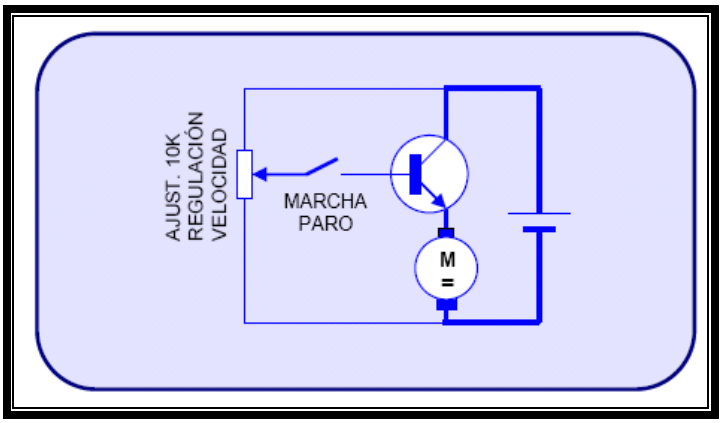


Figura 1.14 Circuito de control de la marcha y paro en un solo sentido mediante transistor de mediana potencia.

1.6.2 CONTROL DEL SENTIDO DE GIRO

Para invertir el sentido de la marcha de un motor de corriente continua, es necesario invertir el sentido de la corriente que circula por su inducido. Esto se consigue invirtiendo la polaridad aplicada a sus terminales, lo cual se puede realizar manualmente, usando dos conmutadores de tres salidas accionados al unísono, o haciendo uso de los circuitos que se indican a continuación:

1.6.2.1 Usando un relé de dos contactos conmutados:

El sentido de marcha se controla mediante un interruptor en serie con la bobina del relé. Si el interruptor está abierto, el relé no está excitado por lo que el positivo de la pila llegará al terminal superior del motor, por lo que girará en un sentido

determinado. Si accionamos el interruptor el interruptor, el relé se excita e invierte la polaridad que llega al motor. El interruptor de marcha paro es necesario para evitar que el motor esté girando siempre.

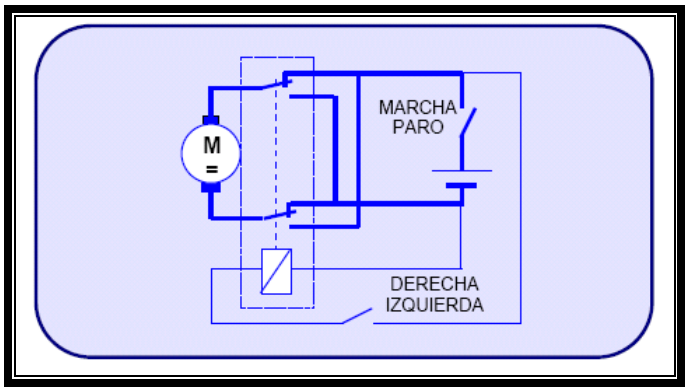


Figura 1.15 Circuito de control del sentido de giro, mediante un relé de dos contactos conmutados.

1.6.2.2 Mediante dos relés:

En este circuito, R1 gobierna el sentido de giro y R2 la marcha y parada del motor. Si accionamos el pulsador “derecha” se excita R1 que a través de sus contactos conecta el “+” del motor con el “+” de la pila. R2 se excita a través del diodo. Al pulsar “izquierda” sólo se excita R2, pues a R1 no le puede llegar tensión a través del diodo. R1 en reposo conecta al “+” del motor con el “-“de la pila, lo que hace que el motor gire a izquierdas.

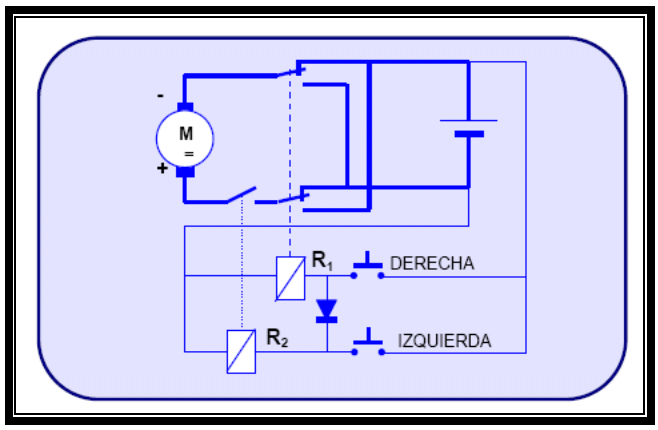


Figura 1.16 Circuito de control del sentido de giro, mediante dos relés.

1.6.2.3 Mediante un puente de transistores en “H”

Este circuito está formado por cuatro transistores dispuestos como en la figura. La corriente de base está limitada por una resistencia R adecuada, aunque en la mayoría de los casos su valor está comprendido entre 1 y 3K. Si pulsamos derecha, conducen los transistores T1 y T4 que permiten el paso de la corriente a través del motor de izquierda a derecha, obligándolo a girar a derechas. Si accionamos el pulsador izquierda, conducen T2 y T3 y el motor gira en sentido contrario.

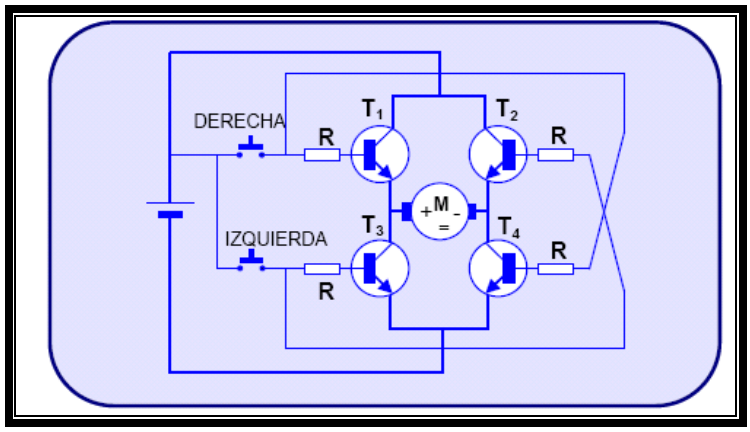
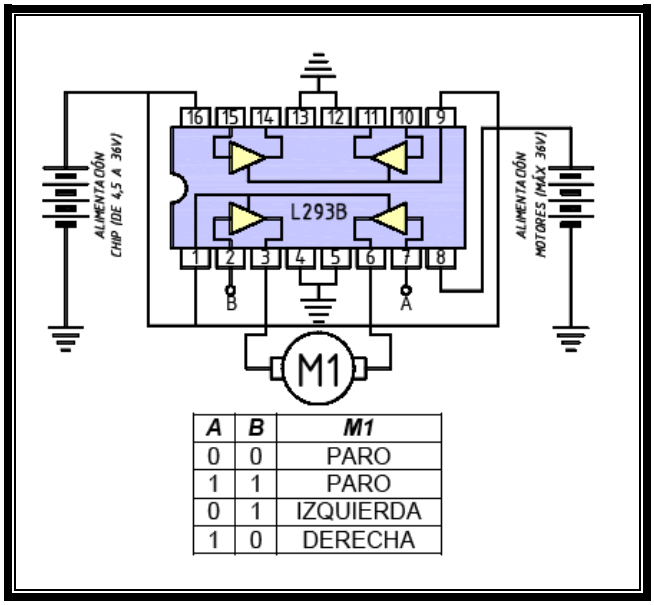


Figura 1.17 Circuito de control del sentido de giro, mediante un puente de transistores en “H”.

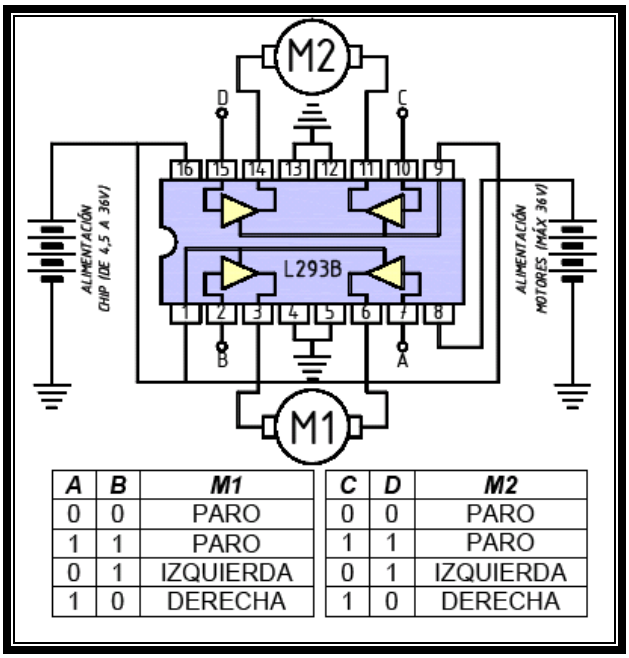
1.6.2.4 Mediante un circuito integrado

Existen circuitos integrados que realizan la misma función que un puente en “H”. Uno de estos chips es el L293B, al cual no hay que conectarle más que el motor, la alimentación y las señales de control. Entre las características más interesantes del L293B se encuentran la protección contra sobre temperaturas, la alta inmunidad al ruido, la alimentación separada de las cargas y la capacidad de proporcionar una corriente de salida de 1 A por canal. Además, posee dos canales, por lo que puede gobernar simultáneamente la marcha de dos motores.



A	B	M1
0	0	PARO
1	1	PARO
0	1	IZQUIERDA
1	0	DERECHA

Figura 1.18 Circuito de control del sentido de giro de un motor, mediante circuito integrado L293B.



A	B	M1
0	0	PARO
1	1	PARO
0	1	IZQUIERDA
1	0	DERECHA

C	D	M2
0	0	PARO
1	1	PARO
0	1	IZQUIERDA
1	0	DERECHA

Figura 1.19 Circuito de control del sentido de giro de dos motores, mediante circuito integrado L293B

CAPÍTULO II

CONSTRUCCIÓN DEL HARDWARE.

2.1 INTRODUCCIÓN

En el presente capitulo se detalla la construcción de los circuitos emisor y receptor, en el capitulo anterior se detallo los elementos y dispositivos principales que se utilizara en la construcción del sistema de control inalámbrico por lo que en este capítulo se describe paso a paso la realización y la obtención de cada circuito del proyecto.

2.2. DIAGRAMA DE BLOQUES PARA EL SISTEMA DE CONTROL INALÁMBRICO

Para obtener y determinar los circuitos emisor y receptor para el control del sistema inalámbrico hemos realizado un estudio previo de los diferentes elementos y componentes a utilizarse, para lo cual a continuación se explicara mediante diagramas de bloques tanto para el circuito emisor como para el circuito receptor de manera muy general todos los sistemas de los cuales está compuesto el sistema de control inalámbrico y que se utilizara en el proyecto.

2.2.1. DIAGRAMA DE BLOQUES CIRCUITO EMISOR.

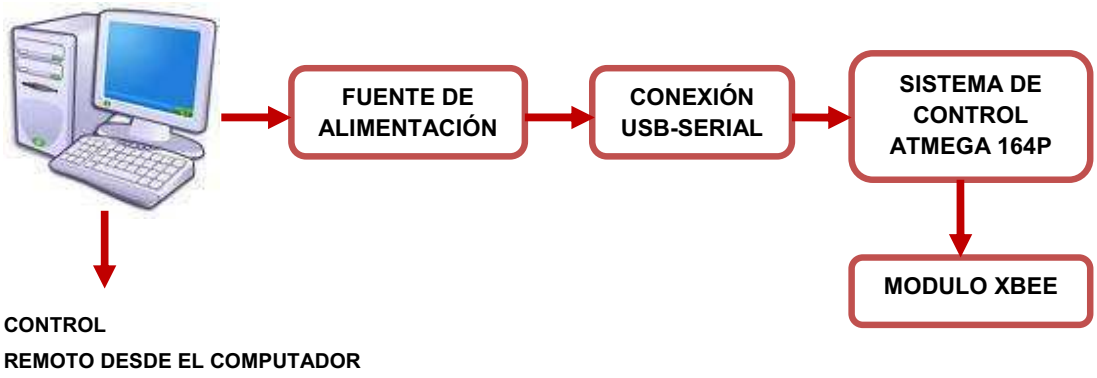


Figura 2.1 Diagrama de bloques circuito Emisor.

2.2.1.1. CONEXIÓN USB A SERIAL.

Para el proyecto a construir se determino previamente que la comunicación entre el computador y el circuito emisor será vía conexión USB, por lo que debemos tener en cuenta para la realizar el circuito este parámetro de gran importacia, y se determinó la implementación de un circuito que nos ayudara a realizar la comunicación USB a serial que servirá para la comunicación desde el computador al microcontrolador que tendrá el sistema de control, este circuito conocido a nivel tecnológico y el cual podemos encontrar su diagrama circuital y el programa a cargar en el microcontrolador a elegir el cual puede ser el ATMEGA 8/48/88, está compuesto como se puede observar en la figura 2.2 de un microcontrolador a elegir que en nuestro caso será el ATMEGA 8, un circuito oscilador y elementos como resistencias.

Con este circuito se obtendrá la comunicación USB a serial mediante software y más no por hardware, y esto se lo hace mediante el programa a utilizar el microcontrolador como se menciono anteriormente.

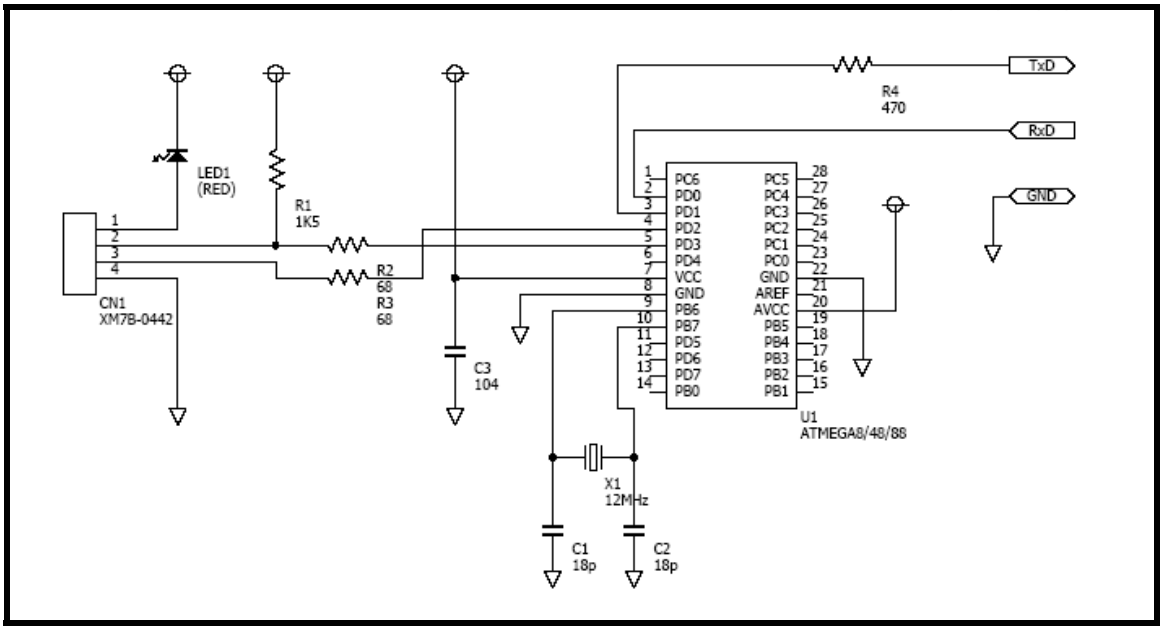


Figura 2.2. Diagrama circuito AVR-CDC (ATmega8/48/88)

2.2.1.2. SISTEMA DE CONTROL DEL CIRCUITO EMISOR.

Este es el sistema más importante del circuito emisor ya que este será como un administrador o controlador de acciones, el software del circuito emisor se lo implementara en un microcontrolador ATMEGA 164, se llegó a determinar este microcontrolador porque posee dos comunicaciones seriales por hardware lo cual es una gran ventaja para nuestro circuito que necesita dichas comunicaciones una para la comunicación entre el microcontrolador ATMEGA 8 que nos servirá para transformar la comunicación USB a serial el cual nos dará las instrucciones que se enviara desde el computador y la otra para la comunicación con el modulo

inalámbrico XBEE, el cual enviara las instrucciones al modulo inalámbrico del circuito receptor del carrito a control remoto.

La conexión del microcontrolador ATMEGA 164p con el circuito de comunicación USB a serial esta dado por los pines **PD.2 (INT0)** y **PD.3 (INT1)** del ATMEGA 164 con los pines PD0 (RXD) y PD1 (TXD) del microcontrolador ATMEGA 8 respectivamente como se puede observar en la figura 2.5.

Aunque el proceso de ejecución y control de acciones es serial, es decir ejecuta una sola acción a la vez (en tiempo del orden de micro-segundos), para la percepción humana resulta que el control y monitoreo se da simultáneamente. Además ciertas situaciones son manejadas por interrupción de hardware y software, es decir cualquier evento que genere una interrupción, es atendido automáticamente.

2.2.1.3. MODULO INALÁMBRICO XBEE.

Para la comunicación inalámbrica del proyecto realizado se selecciono el modulo inalámbrico XBEE, el cual se lo selecciono y determino por su alta confiabilidad dentro de la comunicación inalámbrica, este dispositivo trabaja con una frecuencia de operación de 2.4 GHz, tiene un alcance en línea de vista de hasta aproximadamente 80 metros y su configuración para su correcto funcionamiento es fácil de realizarlo con el software y hardware adecuado y su conexión se puede observar en la figura 2.5, en el que los pines **PD.0 (RXD)** y el **PD.1 (TXD)** del microcontrolador ATMEGA 164P va conectado a los pines del **DOUT** y **DIN** del modulo inalámbrico respectivamente.

2.2.1.4. CONEXIÓN DEL CIRCUITO RESET

También dentro del sistema de control determinamos un circuito para el RESET del microcontrolador.

La entrada MCLR permite reiniciar el estado del micro, llevándose a cabo dos acciones importantes:

- Se carga un 0 en el contador de programa, de forma que después de un reset siempre se ejecuta la instrucción que está en la posición 0 de la memoria de programa.
- Los registros de estado y control toman un estado conocido y determinado.

Existen dos circuitos muy usados para RESET los cuales se indican en la figura 2.3 y en la figura 2.4

- En este circuito el pulsador normalmente abierto está conectado en paralelo con C

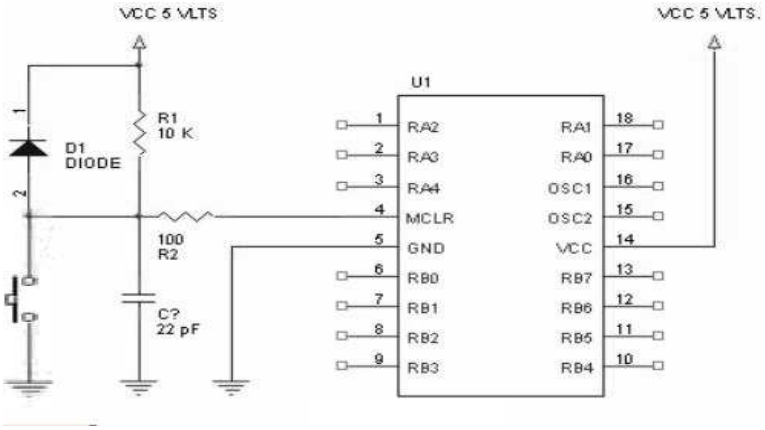


Figura 2.3. Circuito reset (1)

- En este circuito el pulsador normalmente abierto está conectado como un divisor de tensión.

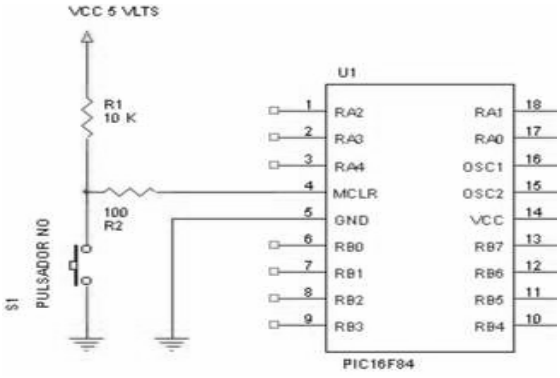


Figura 2. 4. Circuito reset (2)

Para el proyecto a realizar se determino el circuito de la figura 2.3 quedando su conexión final con el microcontrolador 164p como se observa en la figura 2.5

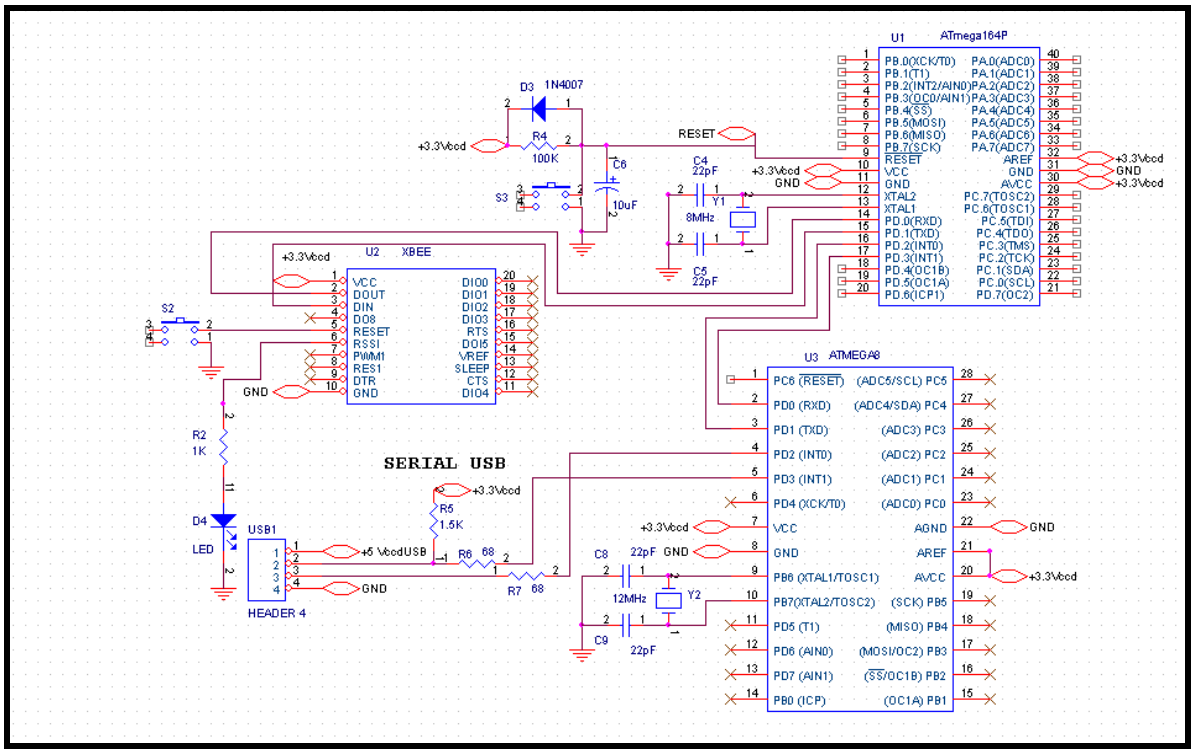


Figura 2.5 Diagrama circuital del sistema de control del circuito Emisor

2.2.1.5. FUENTE DE ALIMENTACIÓN DE VOLTAJE

Para la etapa de alimentación de todo el circuito emisor que está conformado como ya se explico anteriormente, por dos microcontroladores ATMEGA 8 y 164P respectivamente y un modulo inalámbrico XBEE como elementos principales, por lo que es necesario una fuente de alimentación de 3.3 voltios para todos estos elementos ya mencionados los cuales trabajan con el mismo voltaje por lo que tenemos una gran ventaja de solamente utilizar un regulador de voltaje para todos estos elementos, por lo cual emplearemos un regulador de voltaje LM1117T3.3, este circuito integrado entrega un voltaje fijo estable a pesar de que existan variaciones en el voltaje de entrada, es decir regula la tensión de alimentación a 3.3 V.

El circuito de alimentación se lo realizo como se puede observar en la figura 2.6 bajo el concepto y recomendaciones del Datasheets del circuito regulador, pero con una implementación adicional la cual es que se podrá seleccionar si se desea que el voltaje de entrada el cual va ser regulado para todo el circuito emisor sea desde el voltaje de conexión USB es decir 5 voltios o un voltaje externo adicional, esto se lo puede seleccionar a través de un jamper y se lo realizo bajo el criterio de garantizar el funcionamiento del circuito de una manera segura y no tener caídas de voltaje en el momento de el funcionamiento del circuito.

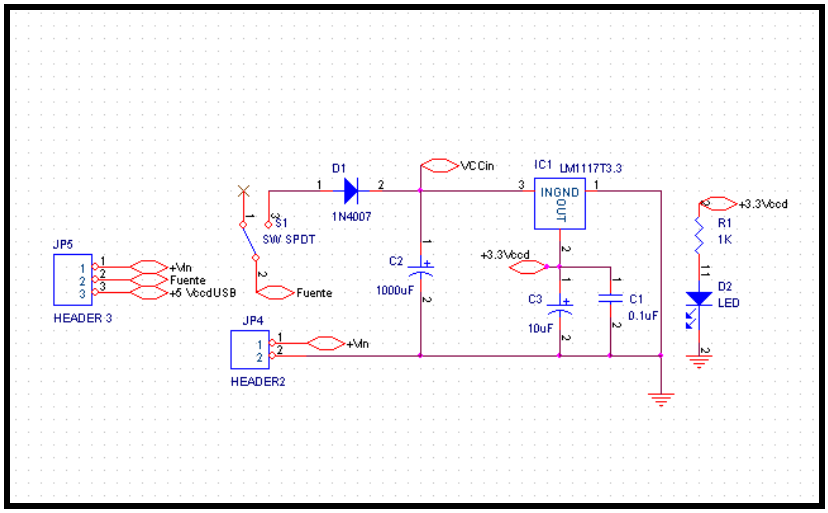


Figura 2.6. Fuente de Alimentación de Voltaje (Circuito Emisor)

2.2.2. DIAGRAMA DE BLOQUES CIRCUITO RECEPTOR.

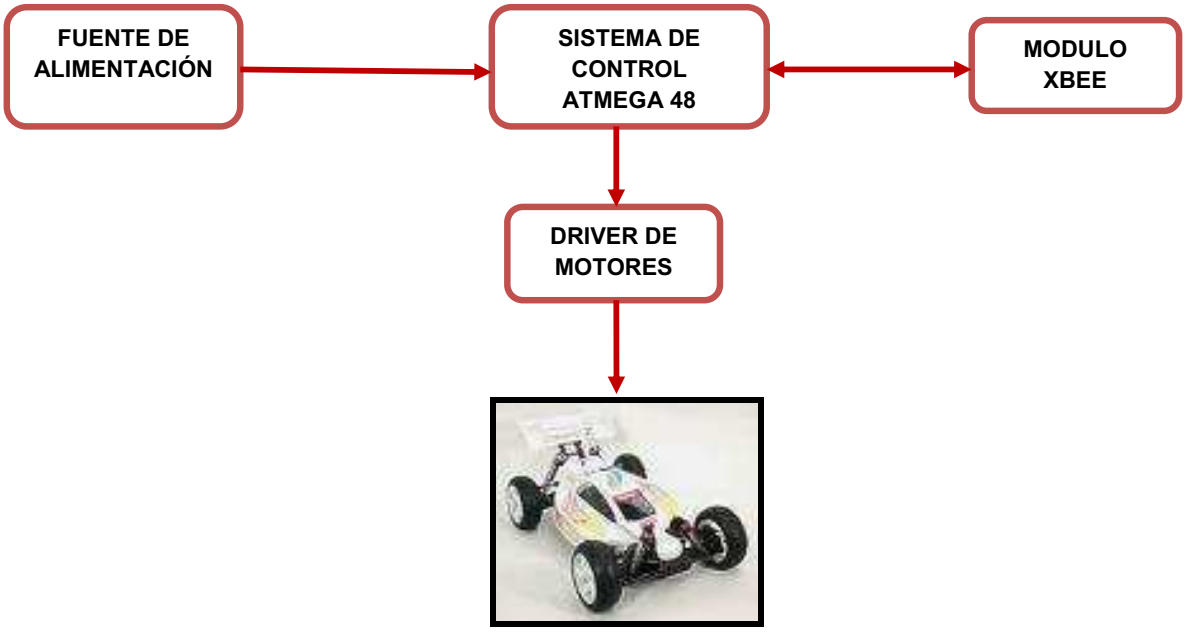


Figura 2.7 Diagrama de Bloques Circuito Emisor

2.2.2.1. SISTEMA DE CONTROL DEL CIRCUITO RECEPTOR.

Este es el sistema más importante del circuito receptor ya que este será como un administrador o controlador de acciones, el software del circuito receptor se lo implementara en un microcontrolador ATMEGA 48, este microcontrolador servirá para la comunicación de la recepción de las instrucciones a realizar que se emite desde el modulo XBEE del circuito receptor, el cual recibe las mismas instrucciones inalámbricamente del circuito emisor a través del modulo inalámbrico del circuito emisor.

También con el mismo microcontrolador realizamos la comunicación con el driver de motores para el carrito a control remoto que se lo detallara más adelante.

2.2.2.2. MODULO INALAMBRICO XBEE.

Los pines de conexión entre el microcontrolador ATMEGA 8 y el modulo inalámbrico XBEE como se puede observar en la figura 2.8, en el cual el **PD0 (RXD)** y **PD1 (TXD)** del microcontrolador del ATMEGA 8 va conectados a los pines **DOUT** y **DIN** del modulo inalámbrico respectivamente son los necesarios para realizar la comunicación entre estos dos elementos y así garantizar la comunicación inalámbrico.

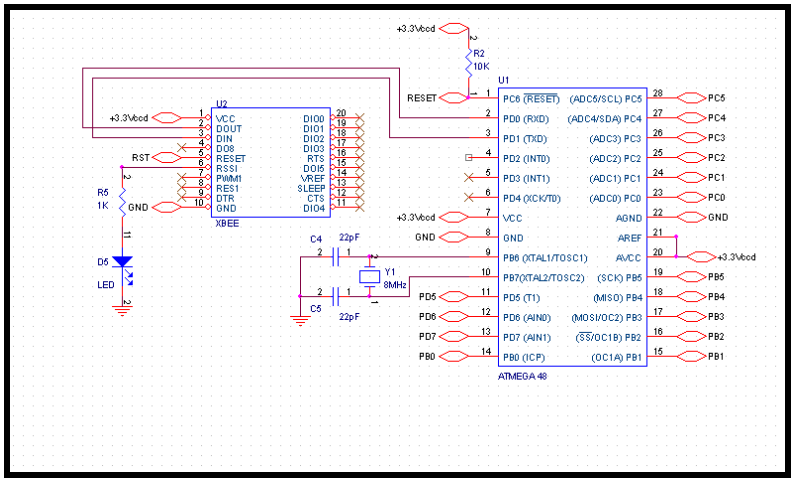


Figura 2.8. Conexión XBEE

2.2.2.3. DRIVER DE MOTORES.

En esta etapa se debe seleccionar el driver adecuado para el control de los motores del carrito a ser controlado, para lo cual se determino el uso del circuito integrado L293B el cual posee doble puente H independientes que nos servirá para los motores.

El rango de voltaje para su funcionamiento es de 3 a 5 voltios y se lo debe utilizar con un banco de diodos recomendados por el fabricante para que no afecte el driver de motores como se puede observar en la figura 2.9, a pesar que en el mercado tecnológico se encuentra drivers de motores en el que ya esta integrados este banco de diodos en un solo circuito integrado, se selecciono el mencionado driver porque maneja un rango de hasta 1 amperio.

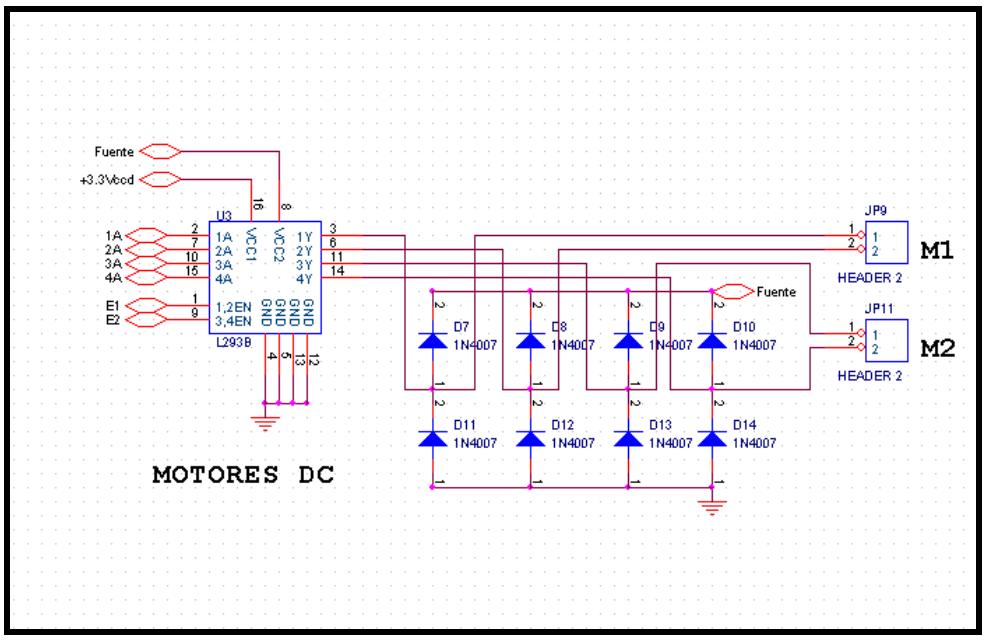


Figura 2.9. Driver de Motores

2.2.2.3.1. CONEXIÓN DEL DRIVER DE LOS MOTORES CON EL MICROCONTROLADOR ATMEGA 48.

Para la comunicación entre el driver de motores y el microcontrolador ATMEGA 48 se debe realizar las conexiones como se puede observar en la figura 2.10, en la que utilizamos los pines **PB2** y **PB1** del microcontrolador ATMEGA 48 para la conexión con los pines **1.2EN** y **3.4EN** del CI L293B, esta conexión nos servirá para controlar la velocidad de los motores mediante PWM (Modulación por Ancho de Pulso) que se lo puede realizar con el programa del microcontrolador que se realizara, ahí se determinara la velocidad con cada opción a realizarse.

Y la otra conexión a realizar como también se puede observar en la figura 2.10 es con los pines **1A, 2A, 3A** y **4A** del CI L293B y los pines **PD5 (T1), PD6 (AIN0), PD7 (AIN1)** y **PB0 (ICP)** del microcontrolador respectivamente, esta conexión nos servirá para poder determinar el sentido de giro de las ruedas basándonos en el datasheet y su tabla de control.

1A	2A	M1
1	0	GIRA HACIA DELANTE
0	1	GIRA HACIA ATRAS
0	0	PARA
1	1	PARA

3A	4A	M2
1	0	GIRA HACIA DELANTE
0	1	GIRA HACIA ATRAS
0	0	PARA
1	1	PARA

Tabla 2.1. Tabla de control de motores para el CI l293B

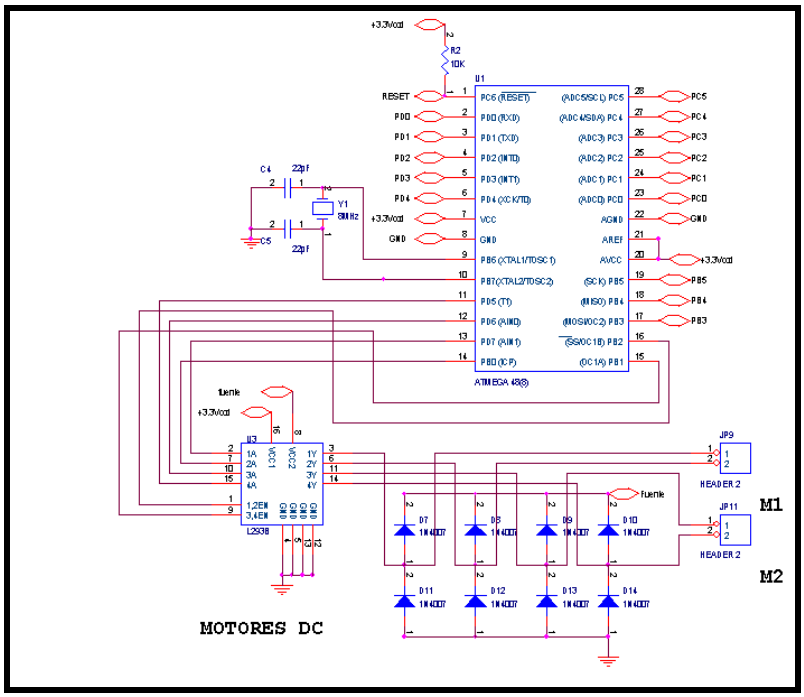


Figura 2.10. Conexión entre Driver de Motores con el Microcontrolador 48

2.2.2.4. FUENTE DE ALIMENTACIÓN VOLTAJE.

De igual manera como se lo hizo para el circuito emisor se determino los elementos principales a utilizar, un microcontrolador, un driver de motores y el modulo inalámbrico XBEE, todos estos trabajan con una fuente de alimentación de 3.3 Voltios, por lo que al igual que el circuito emisor se utilizara el regulador de voltaje LM1117T3.3 el cual regulara el voltaje de entrada que será de la batería del carrito a control remoto que es 7.5 Voltios, el diagrama del circuito es igual que la utilizada para el circuito emisor, y también viene implementado un jamper de selección de voltaje el cual podemos seleccionar la opción si fuera el caso que necesitaremos un voltaje externo que servirá únicamente para el motor del carrito si se desea que perdure el voltaje más tiempo del prototipo.

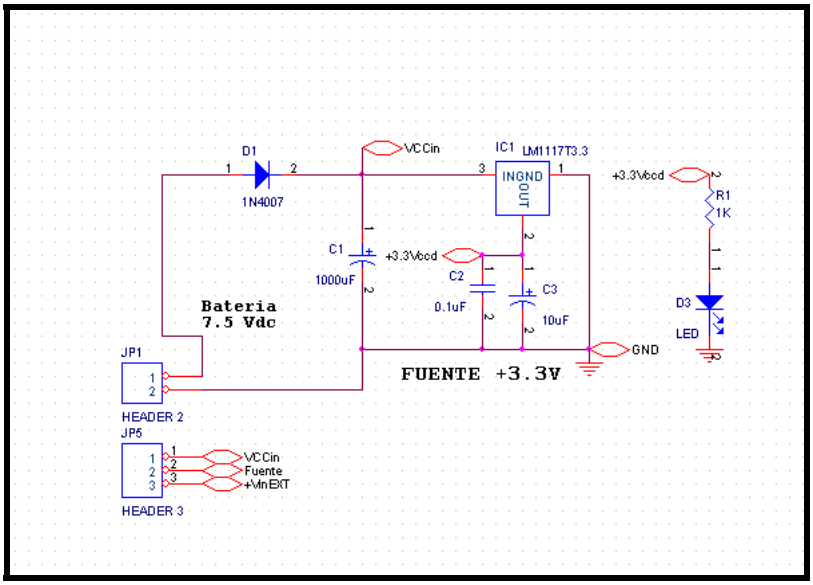


Figura 2.11. Fuente de Alimentación de Voltaje (Circuito Receptor)

2.2.3. CONEXIÓN DE OSCILADORES

Para el oscilador de los circuitos emisor y receptor se utilizo un cristal de 8Mhz, capacitores de 22pF y de tipo XT que se detallara más adelante.

La frecuencia de trabajo viene dada por el oscilador externo.

Los microprocesadores admiten cuatro tipos de osciladores:

- Oscilador RC: Oscilador de bajo costo formado por una resistencia y un condensador, cuyos valores determinan la frecuencia de oscilación. Proporciona una estabilidad mediocre.
- Oscilador HS: Basado en un cristal de cuarzo, alcanza una velocidad entre 4 y 10 MHz.

- Oscilador XT: Oscilador de cristal o resonador para frecuencias entre 100 KHz y 4 MHz.
- Oscilador LP: Oscilador de bajo consumo, con cristal o resonador para frecuencias entre 35 y 200 KHz.

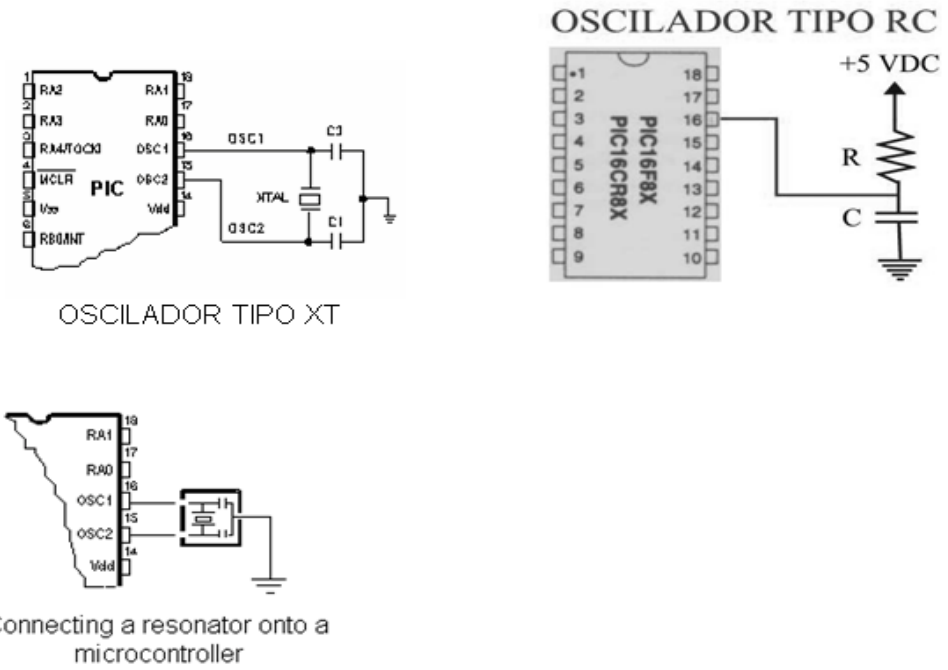


Figura 2. 1 Conexión de osciladores

Finalmente después de a ver determinado y seleccionado todos los parámetros que se necesitan para la construcción del sistema inalámbrico que controlara el carrito a control remoto, se obtiene los circuitos emisor (Ver figura 2.13) y receptor (Ver figura 2.14) con los que empezaremos la realización del hardware y después la realización del software para cada uno de los microcontroladores a utilizar en cada circuito

2.3. ELEMENTOS A UTILIZARSE PARA LA CONSTRUCCIÓN DE LOS CIRCUITOS EMISOR Y RECEPTOR.

2.3.1. Circuito Emisor

A continuación se detallan todos los elementos a utilizarse en la construcción del circuito emisor:

- Modulo XBEE
- Microcontrolador de ATMEL, ATMEGA8
- Microcontrolador de ATMEL, ATMEGA164
- Regulador LM1117T3.3
- 2 Condensadores de 0.1uF
- 1 Condensador de 1000uF
- 2 Condensadores de 10uF
- 4 Condensadores de 22pF
- 3 Resistencias de 1K
- 1 Resistencia de 100K
- 1 Resistencia de 1.5K
- 2 Resistencias de 68
- 2 Diodos 1N4007
- 3 Led
- Cristal de 8MHZ
- Cristal de 12MHZ
- USB Heander de 4 pines
- 7 Heander de 2 pines
- 1 Heander de 3 pines
- 2 Heander de 8 pines
- 1 Heander de 6 pines
- 1 Heander de 5 pines

- 3 Swith Push button

2.3.2 Circuito Receptor

A continuación se detallan todos los elementos a utilizarse en la construcción del circuito receptor:

- Modulo XBEE
- Microcontrolador de ATMEL, ATMEGA48
- Regulador LM1117T3.3
- L293B
- 1 Condensadores de 0.1uF
- 1 Condensador de 1000uF
- 1 Condensadores de 10uF
- 2 Condensadores de 22pF
- 5 Resistencias de 1K
- 1 Resistencia de 1K
- 1 Resistencias de 10K
- 9 Diodos 1N4007
- 5 Led
- Cristal de 8MHZ
- 8 Heander de 2 pines
- 2 Heander de 6 pines
- 2 Heander de 8 pines
- 1 Heander de 3 pines
- 1 Swith Push button

2.4. CIRCUITOS DEL SISTEMA INALÁMBRICO.

2.4.1 Circuito Emisor

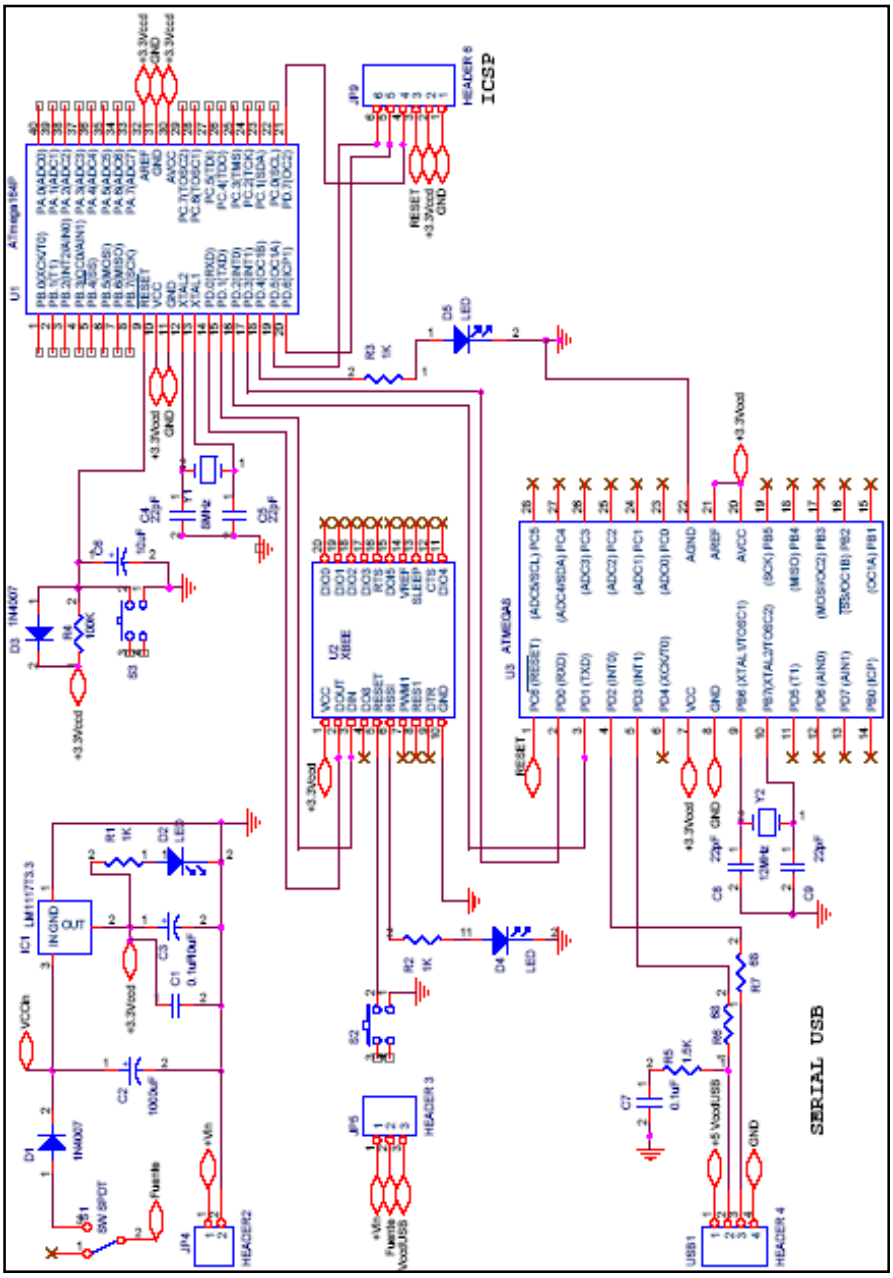


Figura 2.13. Diagrama Circuito Emisor

2.4.2 Circuito Receptor.

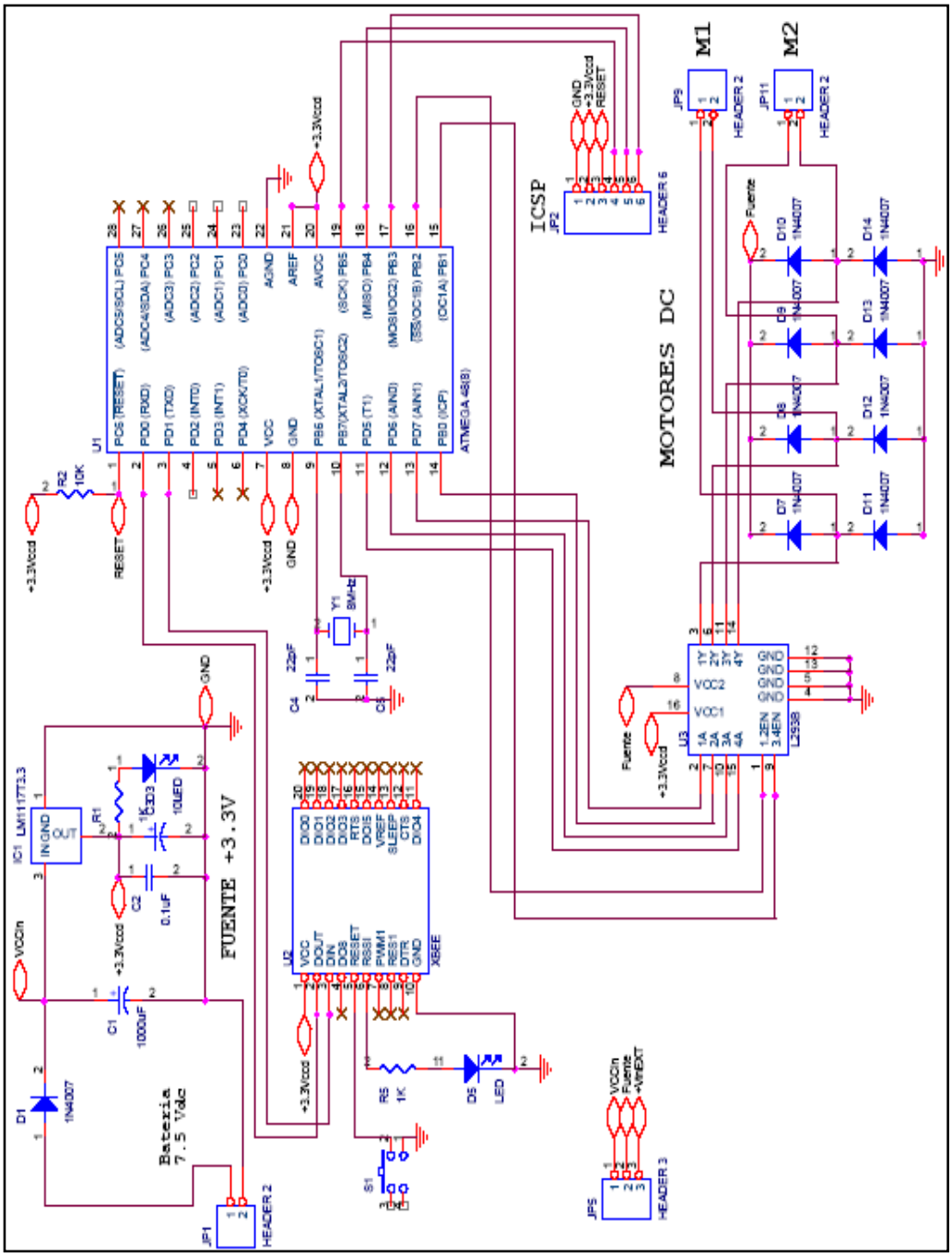


Figura 2.14. Diagrama Circuito Receptor

2.5 CIRCUITOS ESQUEMÁTICOS

En el software OrCad Layout se desarrollaron los circuitos esquemáticos e impreso del emisor y receptor de acuerdo a las dimensiones y espacio que se requiere tanto para el circuito emisor y el receptor que es el que se requiere con mas precisión ya que se lo colocara en el prototipo del carrito.

2.5.1 Circuito Emisor

En las siguientes tablas muestra cada uno de los elementos del circuito con su nomenclatura y símbolos.

SIMBOLOGIA	NOMBRE	TIPO/VALOR
U1	Microcontrolador	ATMEGA164
U2	Modulo Inalámbrica	XBEE
U3	Microcontrolador	ATMEGA8
IC1	Regulador	LM1117T3.3
C1, C7	Condensadores	0.1uF
C2	Condensador	1000uF
C3, C6	Condensadores	10uF
C4,C5,C8,C9	Condensadores	22pF
R1,R2,R3	Resistencias	1K
R4	Resistencia	100K
R5	Resistencia	1.5K
R6,R7	Resistencias	68
D1,D3	Diodos	1N4007
D2,D4,D5	Led	
Y1	Cristal	8MHZ
Y2	Cristal	12MHZ
JP1,JP2,JP3,	Zócalos para	2 pines

JP4,JP6, JP11,JP12	comprobación de voltajes principales del circuito	
JP5	Jamper de selección de voltaje USB/VEXT	3 pines
JP7,JP8	Pines libres para futuras aplicaciones	8 pines
JP9	Zócalos ICSP	6 pines
JP10	Pines libres para futuras aplicaciones	5 pines
S1,S2,S3	Swith	Push button

Tabla 2.2. Elementos que contiene el circuito Emisor

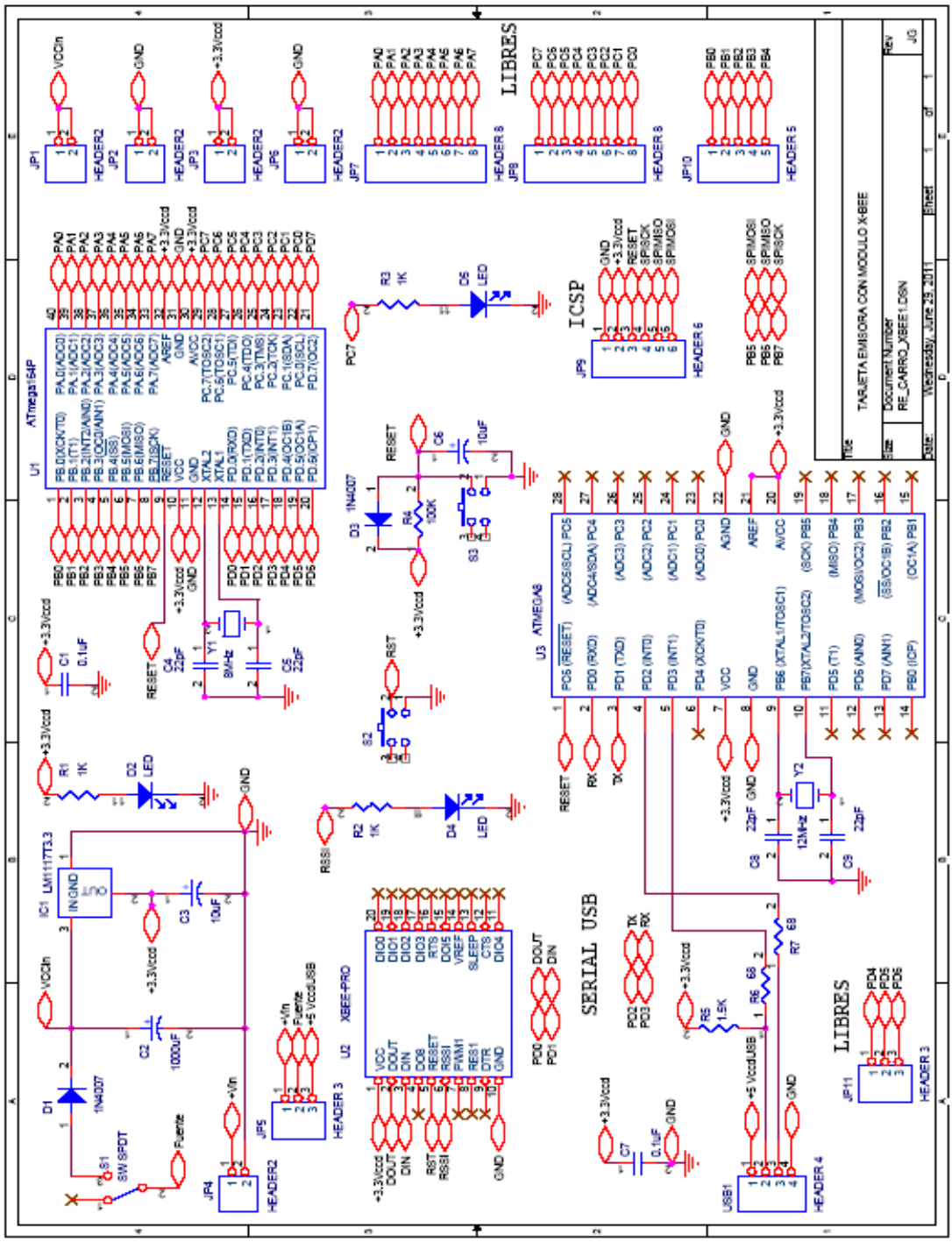


Figura 2.15. Circuito Esquemático OrCad Layout del circuito emisor

2.5.2 Circuito Receptor

En las siguientes tablas muestra cada uno de los elementos del circuito con su nomenclatura y símbolos.

SIMBOLOGIA	NOMBRE	TIPO/VALOR
U1	Microcontrolador	ATMEGA48
U2	Modulo inalámbrico	XBEE
U3		L293B
IC1	Regulador	LM1117T3.3
C2	Condensadores	0.1uF
C1	Condensador	1000uF
C3	Condensadores	10uF
C4,C5	Condensadores	22pF
R1,R3,R4,R5,R6	Resistencias	1K
R2	Resistencia	10K
D1,D7,D8,D9,D10, D11,D12, D13,D14	Diodos	1N4007
D2,D3,D4,D5,D6	Led	
Y1	Cristal	8MHZ
JP1,JP3,JP4, JP6,JP7,JP8, JP9,JP11	Zócalos para comprobación de voltajes principales del circuito	2 pines
JP5	Jamper de selección de voltaje BATERIA/VEXT	3 pines

JP10,JP2	Pines libres para futuras aplicaciones	6 pines
JP9	Zócalos ICSP	6 pines
JP10	Pines libres para futuras aplicaciones	5 pines
S1	Swith	Push button

Tabla 2.3. Elementos que contiene el circuito Receptor

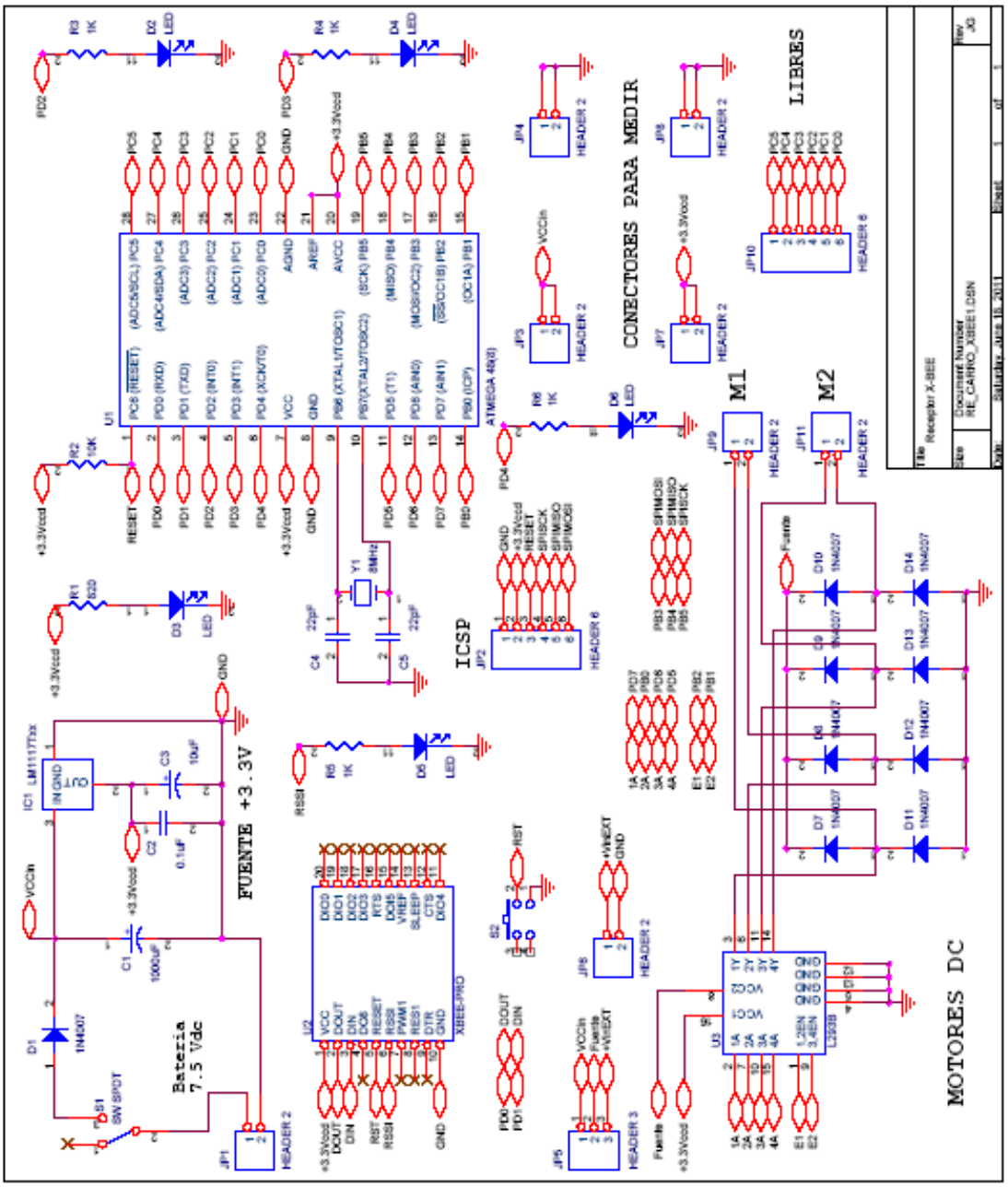


Figura 2.16. Circuito Esquemático OrCad Layout del circuito Receptor

2.6 DISEÑO DE LAS PLACAS DEL CIRCUITO IMPRESO (PCB)

Una vez realizado los circuitos esquemáticos del emisor y receptor, procedemos a realizar el rutiado en el Layaout de cada uno de los circuitos esquemáticos, una vez que ya tenemos el Layaout con todos los elementos procedemos a ubicarlos de acuerdo a nuestra conveniencia y finalmente obtenemos el circuito impreso final como se puede ver en la figura 2.17 del circuito emisor y la figura 2.18 del circuito receptor.

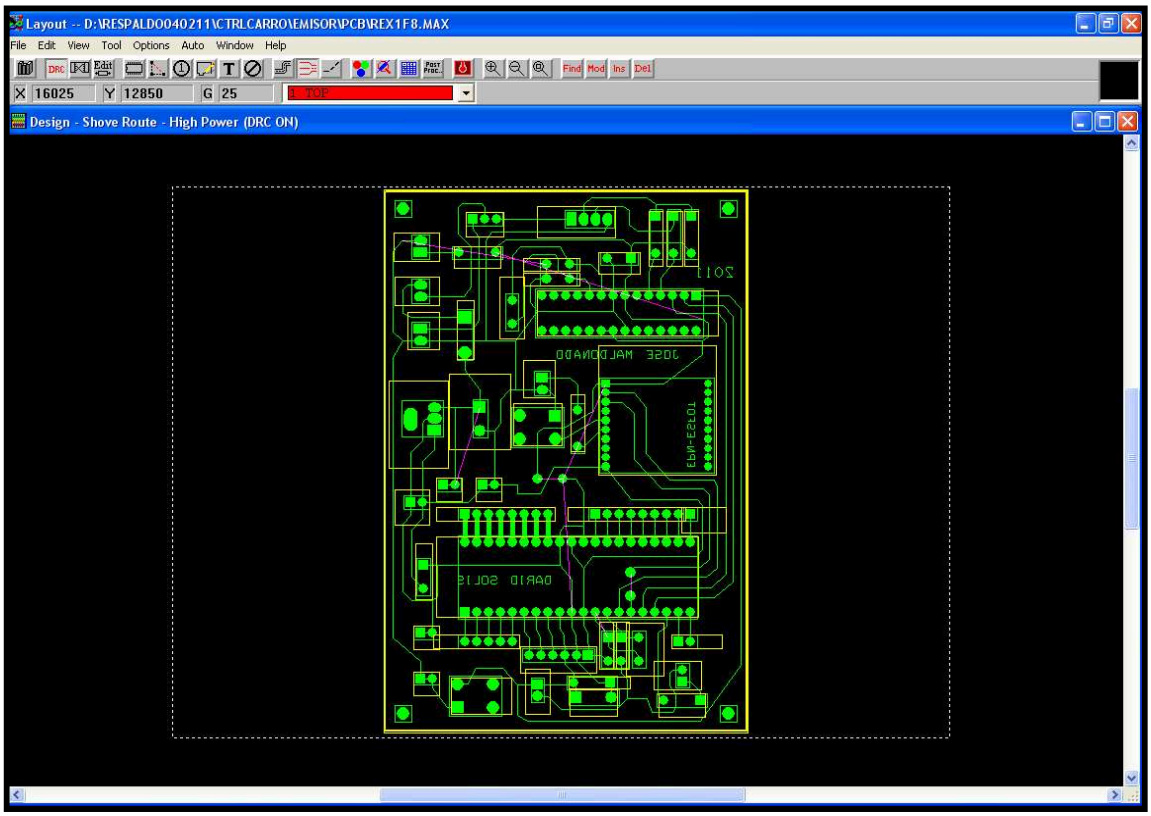


Figura 2.17. Circuito impreso del emisor

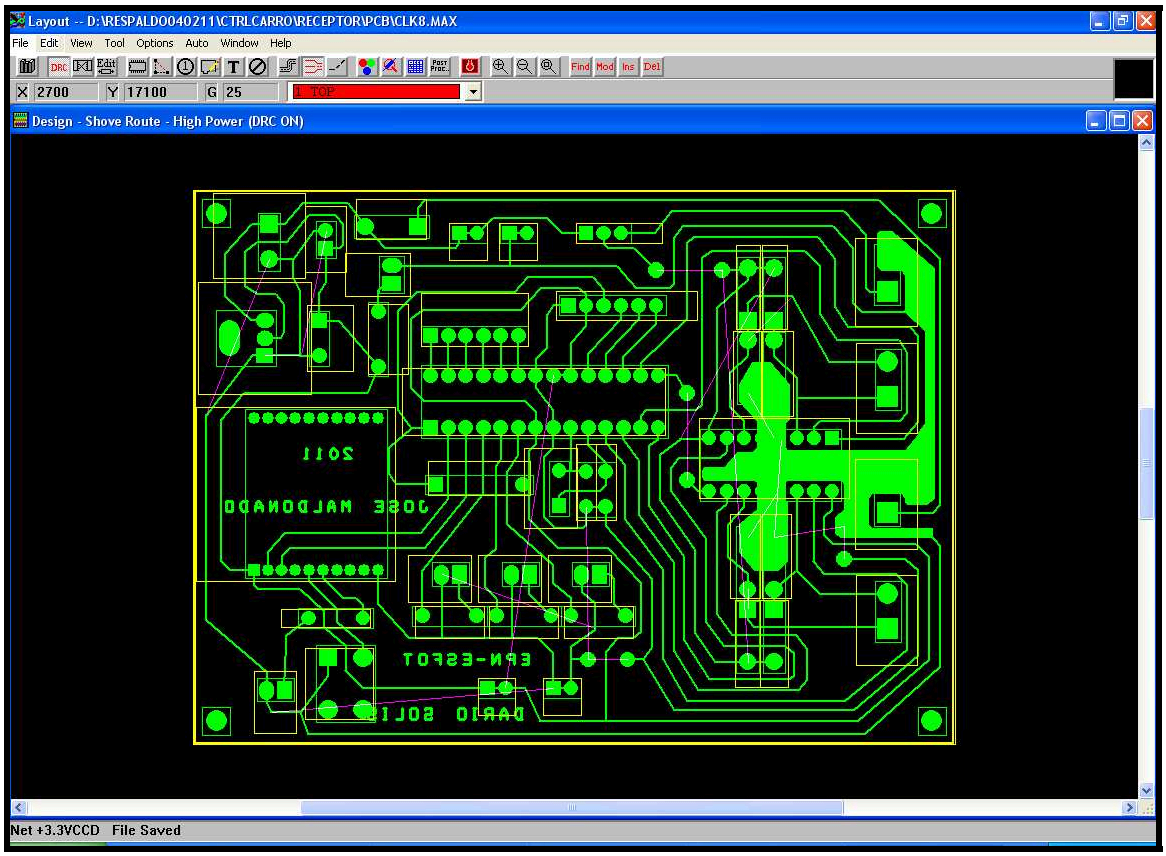


Figura 2.18. Circuito impreso del Receptor

2.7 CREACIÓN DE LOS CIRCUITO IMPRESOS

Una vez realizado los diagramas circuitales, realizamos los trazados de las pistas de los circuitos emisor y receptor con el programa Layaout, para eso debemos tomar en cuenta las medidas de cada placa, las cuales son 12.5cm de largo y 9 cm de ancho para la placa del circuito emisor y 11.5 cm de largo y 8.5 cm de ancho para el circuito receptor, también debemos tomar en cuenta los requerimientos electrónicos y eléctricos.

Una vez realizado la ubicación de cada uno de los elementos se procede a rutear y como resultado final se obtiene los diagramas de pistas como se observa en la figura 2.19 del circuito emisor y la figura 2.20 del circuito receptor.

Para imprimir solo las pistas y no los elementos se tiene que desactivar Top Cooper.

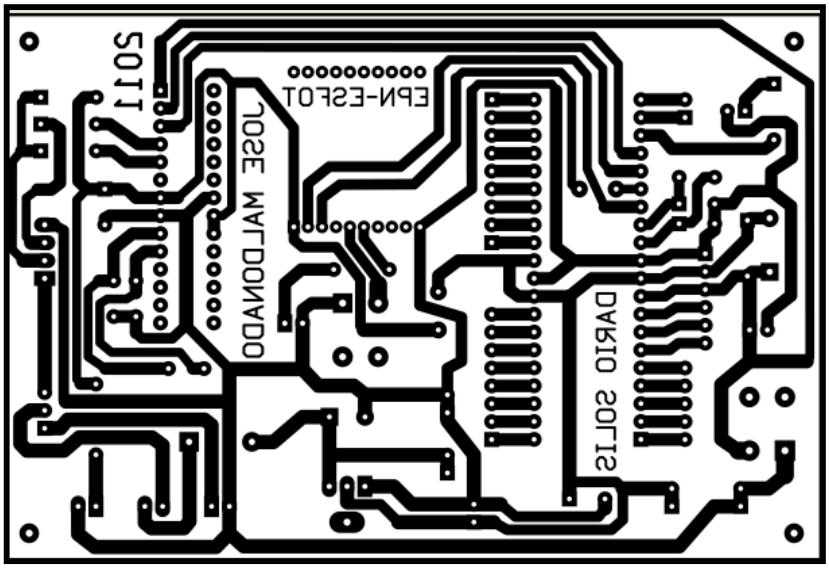


Figura 2.19. Ruteado del circuito Emisor

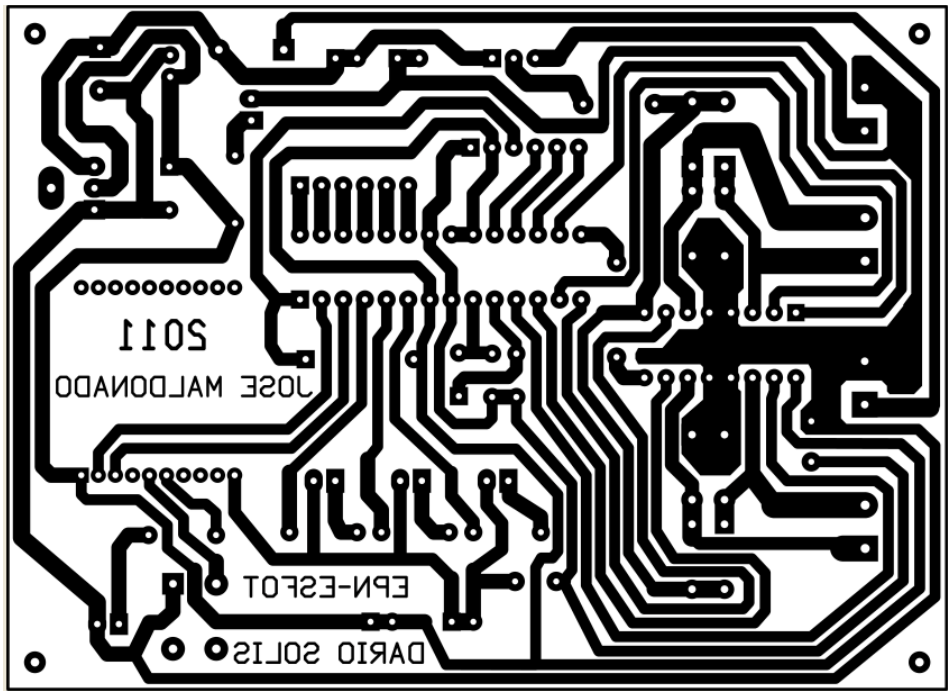


Figura 2.20. Ruteado del circuito Receptor

Para el diseño de cada una de las placas también realizamos el screen de los elementos, que es más que la información, datos y figuras que nos indica el lugar donde van a ubicarse los elementos electrónicos de cada placa impresa, se debe tomar en cuenta que se bebe poner en el efecto espejo y sin las pistas y para esto seleccionamos el Top Silk y Mirror, como se indica en la Figura 2.21 del circuito emisor y la Figura 2.22 del circuito receptor.

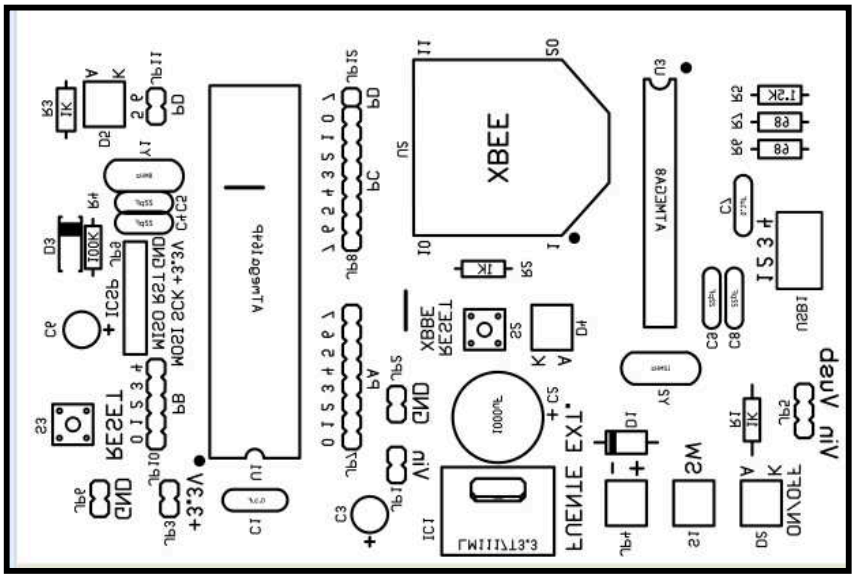


Figura 2.21. Screen del circuito Emisor

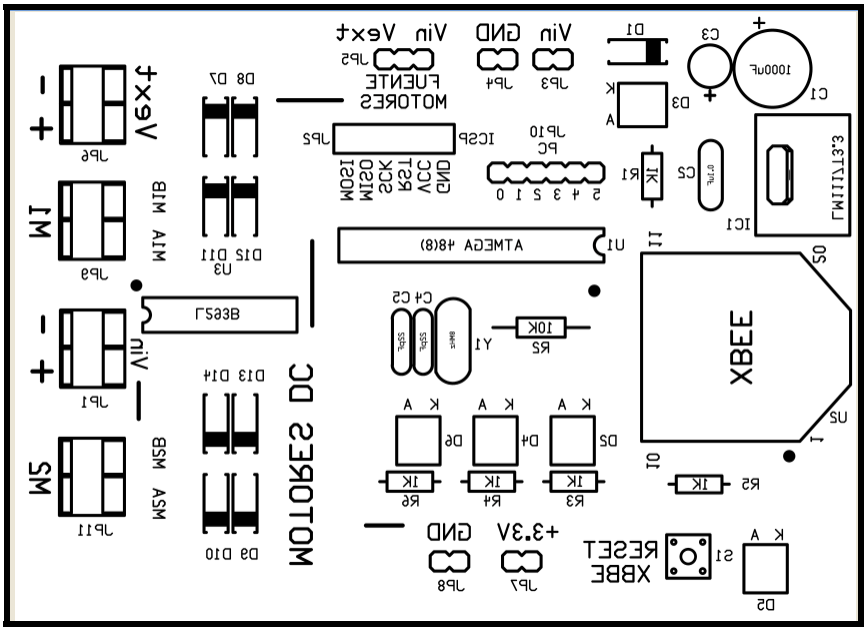


Figura 2.22.Screen del circuito Receptor

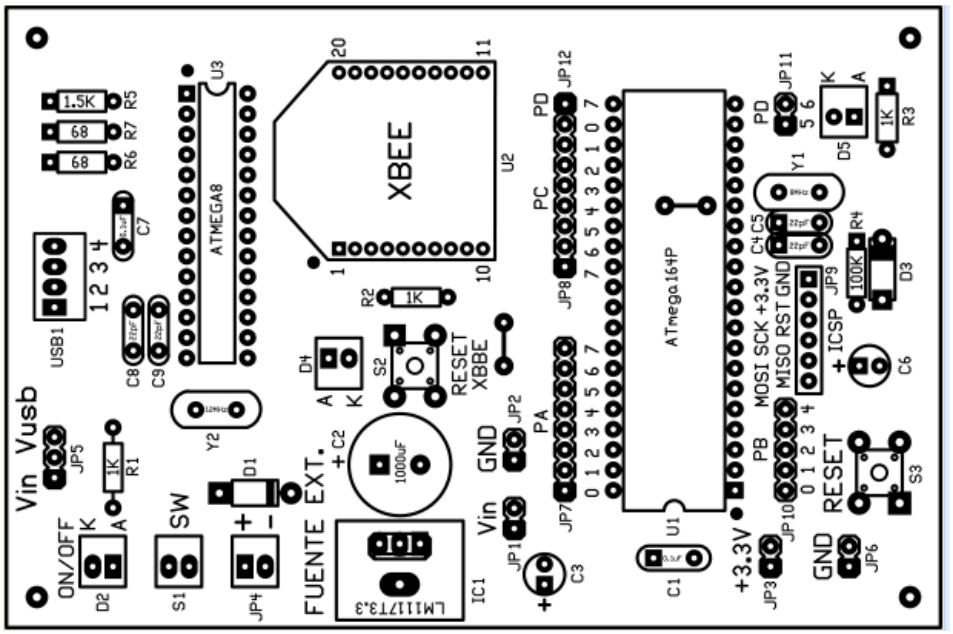


Figura 2.23. Diagrama posicional de los elementos placa del circuito Emisor

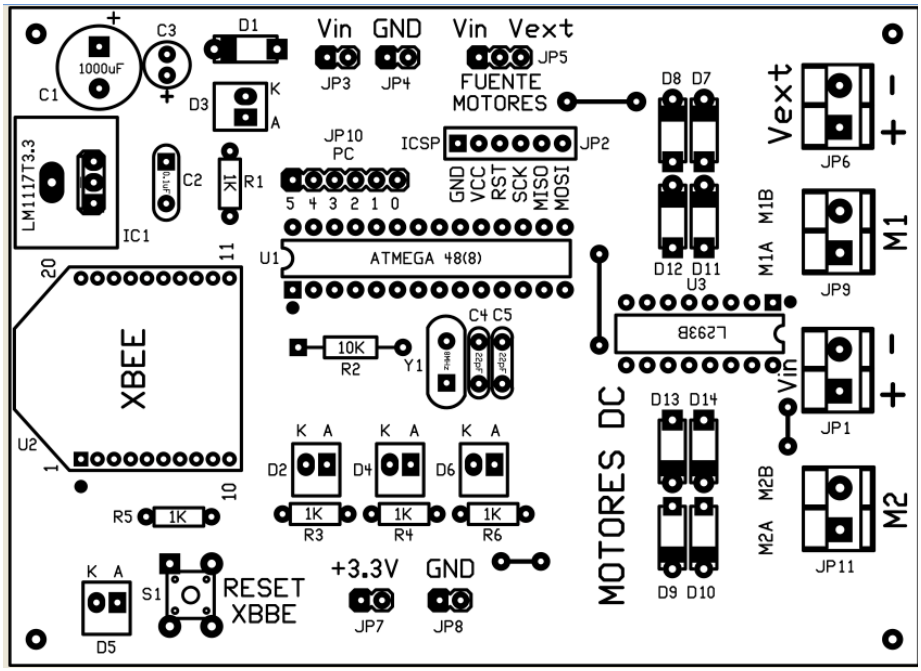


Figura 2.24. Diagrama posicional de los elementos placa del circuito Receptor

Después debemos imprimir los diseños tanto del circuito impreso como el screen de los elementos del circuito emisor y receptor con una impresora laser de excelente calidad en un papel termo transferible, blanco o azul, luego cortamos las placas a las medidas exactas y la limpiamos con mucho cuidado con lana de acero muy fina con el fin de no hacer ninguna clase de ralladura y evitando dejar huellas digitales en ella.

Recortamos los diseños de las fotocopias y colocamos el circuito impreso sobre el lado del cobre de la placa y el screen de los elementos en el lado de la baquelita en el lugar donde no hay cobre.

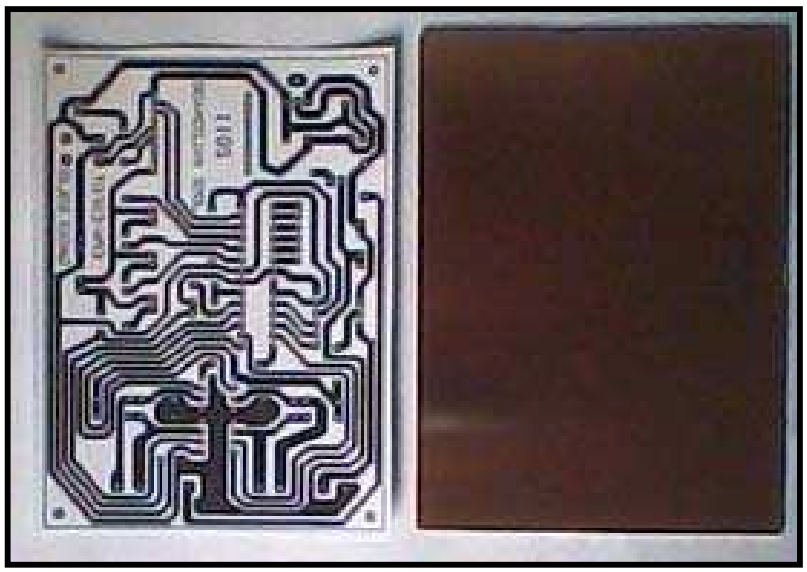


Figura 2.25. Paso previo para el proceso térmico

Realizamos el proceso térmico de cada placa en este caso utilizaremos una plancha al máximo de su temperatura y aplicamos sobre el papel alrededor de 40 segundos para fundir el tóner y adherirlo al cobre tal como se puede observar en la figura 2.26.

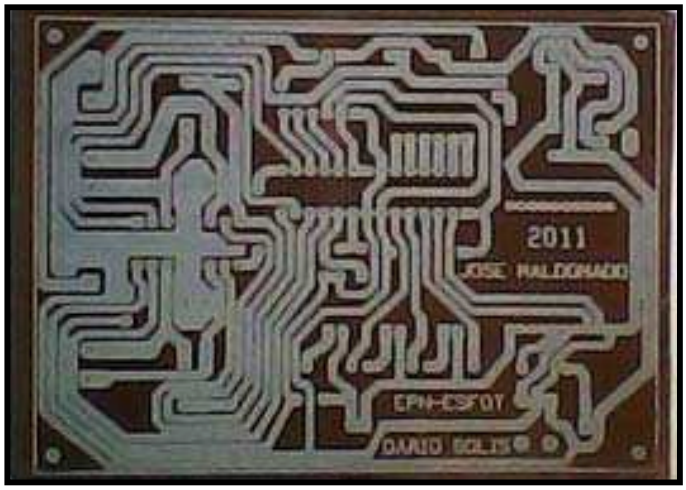


Figura 2.26. Resultado final del proceso térmico

Como se puede observar en la figura 2.26, las pistas del circuito están cubiertas con el tóner de la impresora, para retirar el cobre sobrante es decir lo que no está protegido por el tóner debemos realizar un baño de cloruro férrico durante unos 45 minutos, pero antes de poner las placas en la solución, procedemos a realizar las perforaciones donde estará ubicada cada elemento electrónico de las placas, para así poder corregir el entorno de cada orificio con el marcador.

Una vez que la solución elimina el cobre expuesto retiramos las placas y las lavamos con abundante agua quedando la placa solamente con las pistas deseadas quedando como se indica en la figura 2.27.

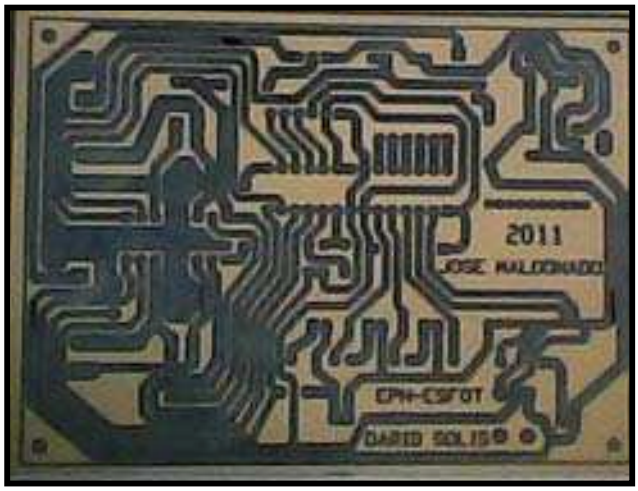


Figura 2.27. Placa después del baño de cloruro férrico

Y finalmente obtenemos las placas de los circuitos deseados como se ve en la figura 2.28, y la de los sreen de los elementos como se ve en la figura 2.29.

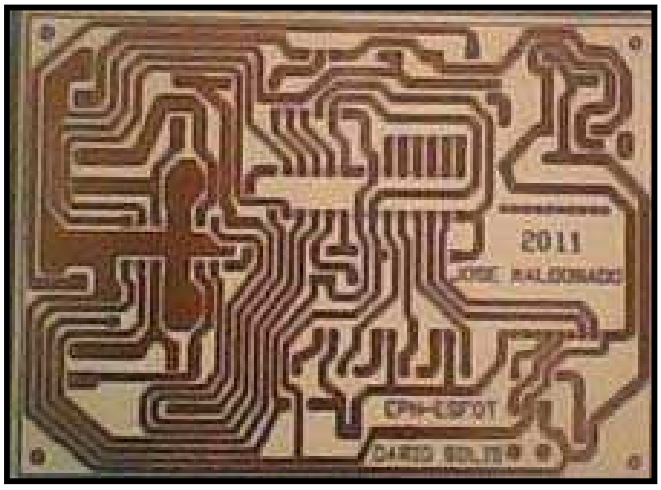


Figura 2.28. Circuito impreso final

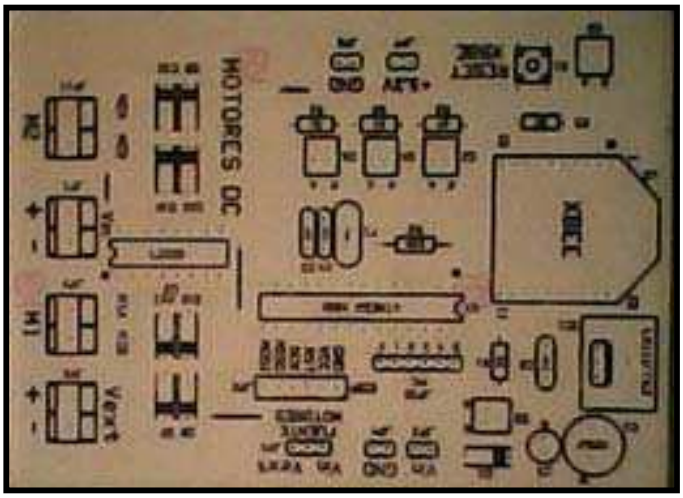


Figura 2.29. Circuito screen final

2.8 DESCRIPCIÓN DE LAS TARJETAS DE LOS CIRCUITOS DEL CONTROL INALAMBRICO.

2.8.1 Tarjeta Circuito Emisor.

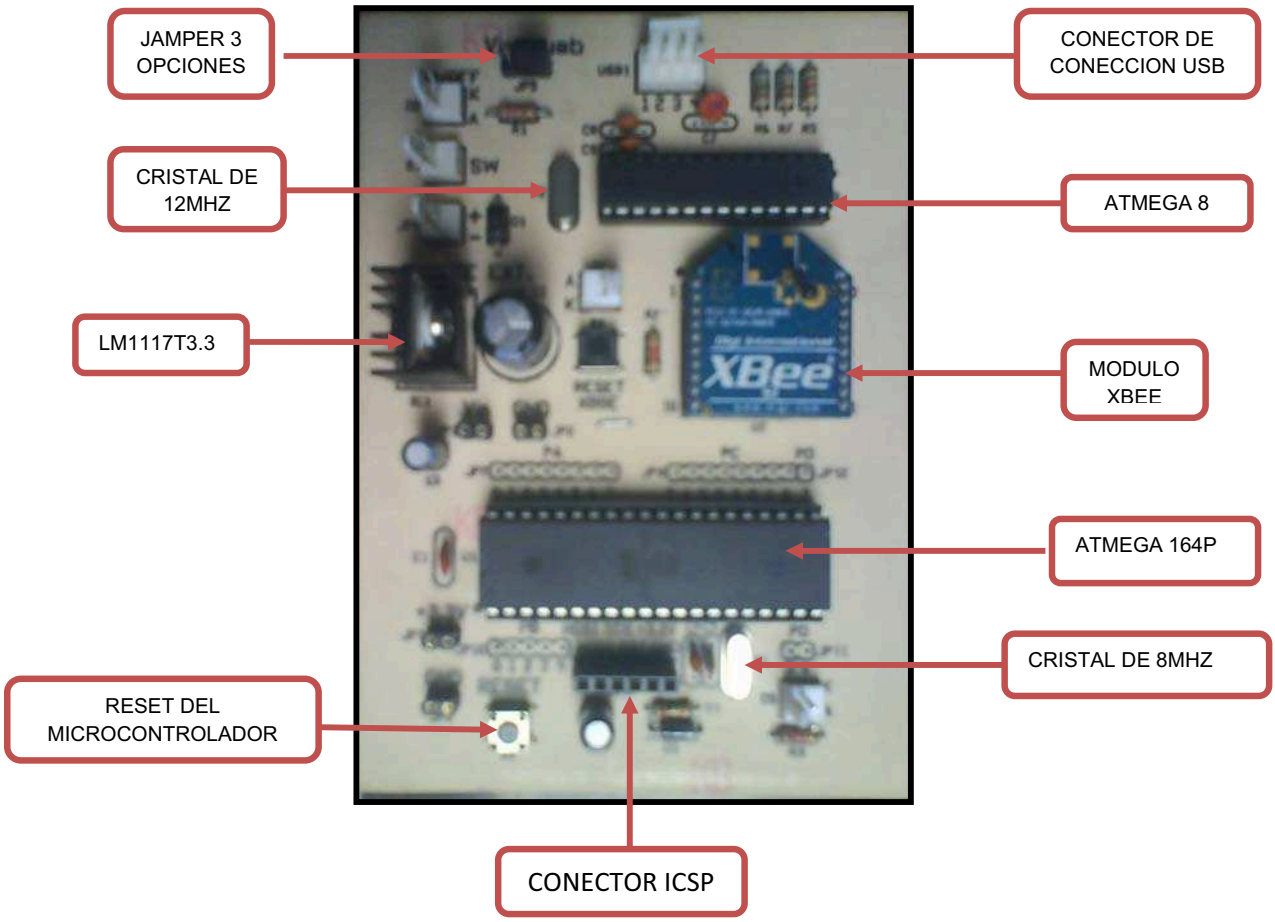


Figura 2.30. Circuito emisor

2.8.2 Tarjeta Circuito Receptor.

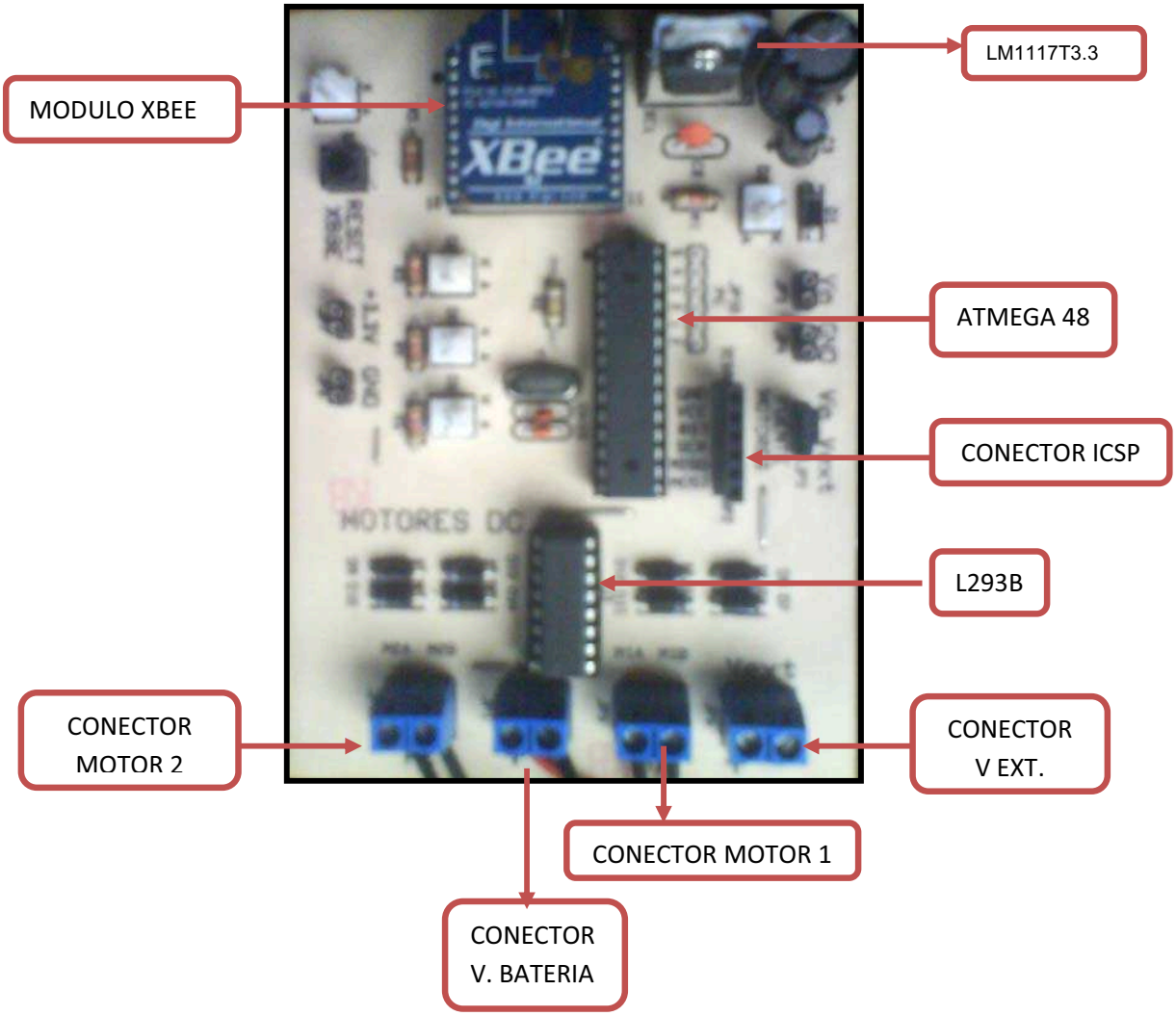


Figura 2.31. Circuito Receptor

2.9. IMPLEMENTACION DEL CIRCUITO EMISOR Y RECEPTOR.

Culminado la realización del hardware, debemos realizar la implementación de los circuitos emisor y receptor, para lo cual al circuito emisor se lo colocara en una caja cerrada con sus debidas implementaciones tanto para las conexiones internas como externas, como se puede ver en la figura 2.32.

Para el circuito receptor de nuestro proyecto se lo colocara en el carrito a control remoto, para esto previamente se hizo la adquisición del mencionado carrito a control remoto que se lo encuentra en el mercado de juguetes y entretenimientos, pero para nuestro proyecto se retiro su circuito de control original para poder implementar el nuestro como se puede ver en la figura 2.33. Y figura 2.34.

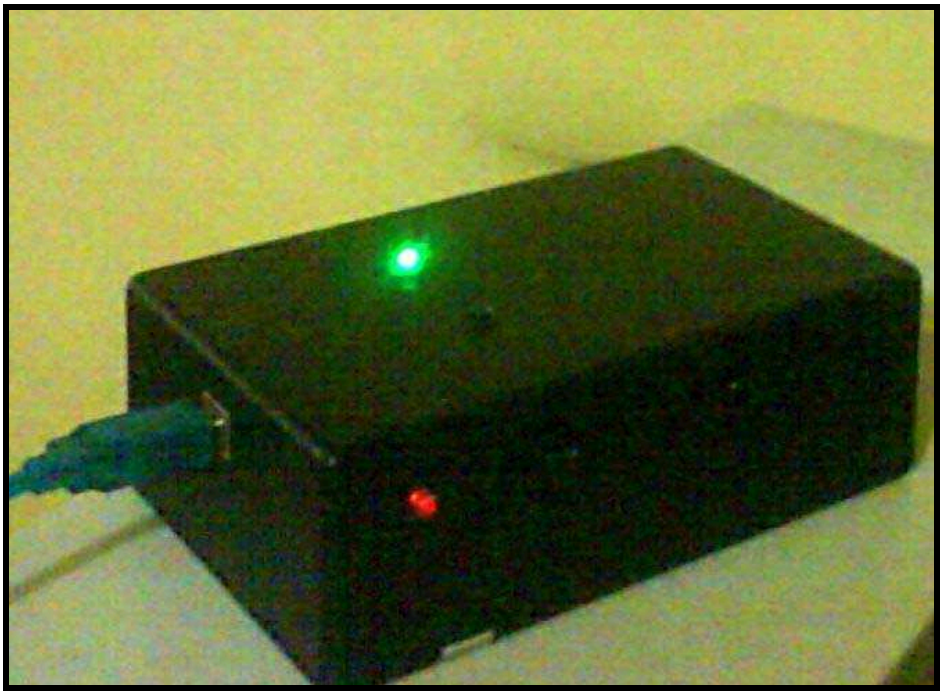


Figura 2.32. Implementación del Circuito Emisor

Figura 2.33. Implantación del Circuito Receptor (Vista Lateral)

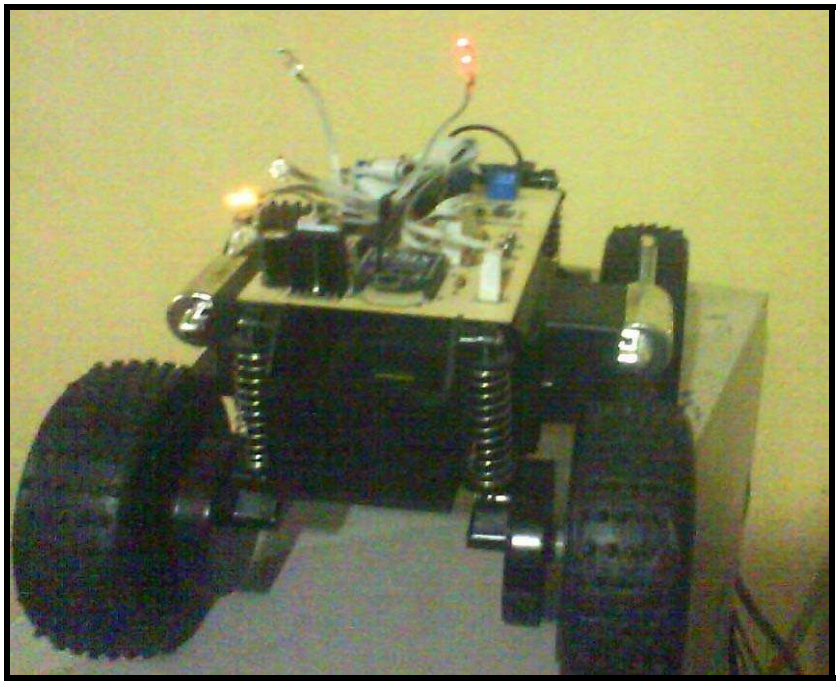


Figura 2.34. Implantación del Circuito Receptor (Vista Frontal)

CAPÍTULO III

DESARROLLO DEL SOFTWARE

3.1. INTRODUCCIÓN

En el presente capitulo trataremos sobre el desarrollo del software para los microcontroladores ATMEGA 148 del circuito emisor y ATMEGA 48 del circuito receptor del sistema de control inalámbrico, para lo cual los programas a utilizarse fueron creados en lenguaje de programación BASIC, estos fueron realizados en el programa BASCOM – AVR, y después compilados para crear el archivo HEX, el cual fue cargado mediante el programador para AVRs llamado PROGISP.

3.2 PASOS PARA CREAR UN PROGRAMA

- Escribir el programa en BASIC, crea un archivo BAS.
- Compilar el programa y ver si no contiene errores
- Si no tiene errores se crea un archivo ASM que es un archivo en ensamblador
- El archivo de ensamblador es entendible para la maquina en donde se está programando pero para el microcontrolador por lo cual crea un archivo en Hexadecimal HEX.
- El archivo HEX es entendible para el microcontrolador, este archivo es el que se graba en la memoria de programa por medio de un grabador o programador ISP.

3.3 HERRAMIENTAS PARA LA PROGRAMACIÓN

3.3.1. BASCOM AVR SOFTWARE[5]

Desde un punto de vista de desarrollo de software, debemos de escoger un lenguaje de programación simple y eficaz para lo cual se ha seleccionado el compilador BASIC de MCS-Electronic (Fabricante).

Se utilizará como herramienta de desarrollo el compilador BASCOM. AVR, el cual permite trabajar en un lenguaje de alto nivel.

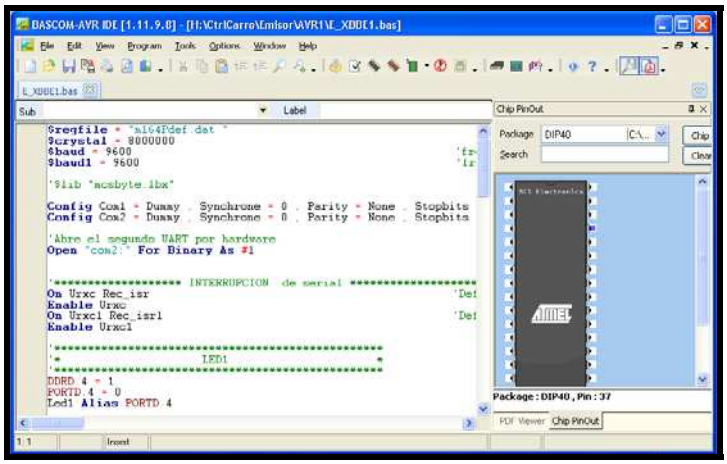


Figura 3.1 Pantalla Principal BASCOM AVR

3.3.1.1 PRINCIPALES SENTENCIAS DE BASCOM

Las siguientes declaraciones son soportadas en BASCOM:

Decisión y estructuras.

IF, THEN, ELSE, ELSEIF, END IF, DO, LOOP, WHILE, WEND, UNTIL, EXIT DO,

EXIT WHILE, FOR, NEXT, TO, DOWNTO, STEP, EXIT FOR, ON ..

GOTO/GOSUB, SELECT, CASE.

(5) **http://www.dontronics.com/basc-avr.html**

Entrada y salida

PRINT, INPUT, INKEY, PRINT, INPUTHEX, LCD, UPPERLINE,

LOWERLINE,DISPLAY ON/OFF, CURSOR ON/OFF/BLINK/NOBLINK, HOME,

LOCATE, SHIFTLCD LEFT/RIGHT, SHIFTCURSOR LEFT/RIGHT, CLS,

DEFLCDCHAR, WAITKEY, INPUTBIN, PRINTBIN, OPEN, CLOSE, DEBOUNCE,

SHIFTIN, SHIFTOUT, GETATKBD

Funciones lógica y numérica

AND, OR, XOR, INC, DEC, MOD, NOT, ABS, BCD, LOG, EXP, SQR, SIN, COS,

TAN, EXP.

I2C

I2CSTART, I2CSTOP, I2CWBYTE, I2CRBYTE, I2CSEND y I2CRECEIVE.

1WIRE.

1WWRITE, 1WREAD, 1WRESET, 1WIRECOUNT, 1WSEARCHFIRST,

1WSEARCHNEXT.

SPI

SPIINIT, SPIIN, SPIOUT, SPIMOVE.

Programación de interrupciónes

ON INT0/INT1/TIMER0/TIMER1/SERIAL, RETURN, ENABLE, DISABLE,

COUNTERx, CAPTUREx, INTERRUPTS, CONFIG, START, LOAD.

Manipulación de bit

SET, RESET, ROTATE, SHIFT, BITWAIT, TOGGLE.

Tipos de datos

DIM, BIT , BYTE , INTEGER , WORD, LONG, SINGLE, STRING , DEFBIT,

DEFBYTE, DEFINT, DEFWORD.

Misceláneos

REM, ' , SWAP, END, STOP, CONST, DELAY, WAIT, WAITMS, GOTO, GOSUB,

POWERDOWN, IDLE, DECLARE, CALL, SUB, END SUB, MAKEDEC,

MAKEBCD, INP,OUT, ALIAS, DIM , ERASE, DATA, READ, RESTORE, INCR,

DECR, PEEK, POKE, CPEEK, FUNCTION, READMAGCARD, SPC.

Directivas del Compilador

$INCLUDE, $BAUD y $CRYSTAL, $SERIALINPUT, $SERIALOUTPUT,

$RAMSIZE, $RAMSTART, $DEFAULT XRAM, $ASM-$END ASM, $LCD,

$EXTERNAL, $LIB.

Manipulación de Cadenas

STRING, SPACE, LEFT, RIGHT, MID, VAL, HEXVAL, LEN, STR, HEX, LTRIM,

RTRIM, TRIM, LCASE, UCASE, FORMAT, FUSING, INSTR.

3.3.1.2. FUNDAMENTOS DEL LENGUAJE

Los caracteres y el set de instrucciones de BASCOM son fusionados para formar etiquetas, palabras clave, variables y operadores. Estos son combinados para formar las declaraciones que constituyen un programa.

El set de instrucciones de BASCOM consiste en caracteres alfabéticos, numéricos y especiales.

Los caracteres alfabéticos en BASCOM son las letras mayúsculas (A-Z) y letras minúsculas (a-z) del alfabeto. Los caracteres numéricos son los dígitos 0-9.
Las letras A-H pueden ser usadas como parte de números hexadecimales.
Los caracteres de la Tabla 2.4, tienen significado especial en declaraciones y expresiones BASCOM:

Carácter	Nombre
ENTER	Termina la entrada de una línea
	Blanco (o espacio)
'	Apostrofe
*	Asterisco (símbolo de multiplicación)
+	Signo más
,	Coma
-	Signo menos
.	Periodo (punto decimal)
/	Slash (symbol de division)
:	Dos puntos
"	Comillas
;	Punto y coma
<	Menor que
=	Signo igual
>	Mayor que
\	Backslash
^	Exponente

Tabla 3.1. Caracteres especiales

3.3.1.3. LÍNEA DE PROGRAMA DE BASCOM

Las líneas de programa de BASCOM tienen la siguiente sintaxis:

[[línea - identificador]] [[Declaración]] [[: Declaración]].... [[Comentarios]]

3.3.1.4. USANDO IDENTIFICADORES DE LÍNEA

BASCOM soporta etiquetas de línea alfanuméricas, que podrían ser la combinación de 1 a 32 letras y dígitos, iniciando con una letra y terminando con dos puntos.

Los siguientes ejemplos son etiquetas de línea alfanuméricas correctas:
Alfa:
Pantalla:
Prueba3:

Las siguientes etiquetas de línea son equivalentes:
alfa:
Alfa:
ALFA:
Las etiquetas de línea podrían comenzar en cualquier columna. Los espacios en blanco no son admitidos entre una etiqueta alfabética y los dos puntos.

Una línea puede tener solamente una etiqueta. Cuando hay una etiqueta en la línea, ningún otro identificador puede ser usado en la misma línea. Así la etiqueta es el único identificador de una línea.

3.3.1.5. DECLARACIONES DE BASCOM

Una declaración de BASCOM puede ser "Ejecutable" o "No ejecutable".

Una sentencia ejecutable promueve el curso de la lógica del programa señalando la acción que se debe realizar.

La expresión no ejecutable lleva a cabo las tareas de asignar el almacenamiento de variables, la declaración y definición del tipo de variables.

Las siguientes declaraciones de BASCOM son ejemplos de declaraciones no ejecutables:

REM (empieza un comentario)

DIM

Un "Comentario" es una declaración no ejecutable, se usa para aclarar una operación del programa y su propósito. Un comentario es presentado por la sentencia de REM o un apostrofe (‘).

Las siguientes líneas son equivalentes:

```
PRINT "Bienvenido":      REM etiqueta que informa imprimir.
PRINT "Bienvenido”:      ' etiqueta que informa imprimir.
```

Más de una declaración de BASCOM puede ser puesta en una línea, pero los dos puntos (:) separan las declaraciones como se ilustra a continuación.

```
FOR numero = 1 TO 5:   PRINT “numero”:   NEXT numero
```

3.3.1.6. TIPOS DE DATOS

Cada variable en BASCOM tiene un tipo de datos que determina lo que puede ser guardado en la misma.

Bit (1/8 byte). Un bit puede tener un solo valor, 0 ó 1.

Byte (1 byte). Los bytes son guardados como números binarios de 8 bits sin signo, su valor se extiende de 0 a 255.

Integer (dos bytes). Los integers son guardados como números binarios de dieciséis bits con signo, su valor se extiende desde -32,768 a +32,767.

Word (dos bytes). Los words son almacenados como números binarios de dieciséis bits sin signo, se extienden en el valor de 0 a 65535.

Long (cuatro bytes). Los longs son guardados como números binarios de 32 bits con signo que se extienden en el valor de -2147483648 a 2147483647.

Single. Los singles son guardados como números binarios de 32 bits. Se extienden en el valor de 1.5 x 10^–45 a 3.4 x 10^38.

Double. Los doubles son guardados como números binarios de 64 bits con signo. Se extienden en el valor de 5.0 x 10^–324 a 1.7 x 10^308.

String (hasta 254 byte). Los strings son almacenados como bytes y terminan con un byte 0. Un string con una longitud de 10 bytes ocupará 11 bytes.

Las variables pueden ser guardadas internamente (default), externamente o en EEPROM.

3.3.1.7. VARIABLES

Una variable es un nombre que se refiere a un objeto ó a un número específico y debe ser asignada con un solo valor numérico (either, integer, byte, long, single ó bit).

La siguiente lista indica algunos ejemplos de las asignaciones a variables:

Un valor constante:

Uno = 5

C = 1.1

El valor de otra variable numérica:

abc = def

k = g

El valor obtenido combinando otras variables, constantes y operadores:

Temp = a + 5

Temp = c + 5
El valor obtenido al llamando de una función:
Temp = Asc (S)

3.3.1.8. NOMBRES DE VARIABLES [6]

El nombre de una variable BASCOM puede contener hasta 32 caracteres, estos pueden ser letras y números, pero el primer carácter siempre debe ser una letra.

Un nombre de una variable no puede ser una palabra reservada, pero las palabras reservadas compuestas son admitidas.

Las palabras reservadas incluyen todos los comandos de BASCOM, declaraciones, nombres de funciones, registros internos y nombres de operadores

Por ejemplo, la siguiente declaración no es permitida porque AND es una palabra reservada.
AND = 8
Sin embargo, la siguiente declaración es permitida:
ToAND = 8
[7] Ayuda del programa BASCOM AVR

Se puede especificar un número hexadecimal o binario con el prefijo &H o &B. Ejemplo:
a = &HA, a = &B1010 y a= 10
Antes de asignar una variable, se debe indicar al compilador sobre la misma con la declaración DIM.

[6]Ayuda del programa BASCOM AVR

Ejemplo: DIM numero As Bit,
DIM a as Integer
DIM k as Byte
DIM s as String * 10
El tipo string necesita un parámetro adicional que especifique la longitud.
También se puede usar DEFINT, DEFBIT, DEFBYTE, DEFWORD, DEFLNG o DEFSNG.

Por ejemplo, c de DEFINT dice al compilador que todas las variables que no son dimensionadas y que inician con el carácter c son del tipo integer.

3.3.1.9. EXPRESIONES Y OPERADORES

Este punto detalla cómo combinar, modificar, comparar, o conseguir información sobre las expresiones usando los operadores disponibles en BASCOM.

Una expresión puede ser una constante numérica, una variable, o un valor obtenido combinando constantes, variables, y otras expresiones con operadores.

Los operadores llevan a cabo operaciones matemáticas o lógicas sobre valores. Los operadores provistos por BASCOM pueden ser divididos en cuatro categorías, de la siguiente manera:

3.3.1.10. OPERADORES ARITMÉTICOS

Usados para efectuar cálculos y son +, - , *, \, / y ^.

Entero

La división de enteros es indicada por la barra invertida (\).

Ejemplo: Z = X \ Y

Modulo Aritmético

El modulo aritmético es denotado por el modulo operador MOD.

El modulo aritmético provee el resto, en vez del cociente, de una división de enteros.

Ejemplo: X = 10 \ 4: resto = 10 MOD 4

Exceso y división por cero

La división por cero, produce error.

Por el momento ningún mensaje es producido así que debe asegurarse que esto no ocurra.

3.3.1.11. OPERADORES DE RELACIÓN

Usados para comparar valores numéricos o de secuencia como se muestra en la Tabla 2.1. El resultado puede ser usado para tomar una decisión respecto al flujo del programa.

Operador	Prueba de relación	Expresión
=	Igualdad	X = Y
<>	Desigualdad	X <> Y
<	Menor que	X < Y
>	Mayor que	X > Y
<=	Menor que o igual a	X <= Y
>=	Mayor que o igual a	X >= Y

Tabla 3.2. Operadores de relación

3.3.1.12. OPERADORES LÓGICOS

Los operadores lógicos llevan a cabo pruebas sobre las relaciones, manipulaciones de bits u operadores Booleanos.

Hay cuatro operadores en BASCOM, mostrados en la Tabla 4.3

Operador	Significado
NOT	Complemento Lógico
AND	Conjunción
OR	Disyunción
XOR	OR Exclusivo

Tabla 3.3. Operadores lógicos

3.3.1.13. OPERADORES FUNCIONALES

Usados para complementar operadores simples.

Funciones Basic.[7]

DO LOOP

```
Do ’inicio del lazo
  Instrucciones
Loop ’regreso al inicio del salto
```

DO LOOP-UNTIL

```
Do
A=a+1
Loop until a=2 ’Cuando a=2 entonces sale del lazo
```

FOR NEXT

```
For a=inicio to fin step pasos
instrucciones
Next a
```

[7] **SOLIS Darío apuntes de curso AVR**

SELECT CASE

Select case variable

Case 1: instrucción

Case 2: instrucción

Case n: instrucción

End select

WHILE WEND

While condición

Instrucciones

Wend

Temporizaciones

WAIT	tiempo en segundos
WAITMS	tiempo en milisegundos
WAITUS	tiempo en microsegundos
SET	poner un 1 al pin
RESET	poner un 0 al pin

Configuración del cristal: $crystal=1000000 (en hertz)

Configuración del puerto serial: $baud=2400

3.3.1.14. CONFIGURACIÓN DE LOS PINES

Los pines del microcontrolador avr tienen 3 registros los cuales son:
Registro ddr, port, pin. El registro ddr nos configura al pin como entrada salida, port es el registro de salida y pin es el registro de entra.

Ddrb.0=0 entrada alta impedancia

Portb.0=0

Ddrb.0=0 entrada pull up

Portb.0=1

Ddrb.0=1 salida a cero 20 mA

Portb.0=0

Ddrb.0=1 salida a uno 20 mA

Portb.0=1

3.3.2. PROGISP (Versión 1.6.7)

Este programa se utiliza para grabar los programas generados por el BASCOM AVR, tanto en el microcontrolador ATMEGA 164 del circuito emisor, como el microcontrolador ATMEGA 48 del circuito receptor.

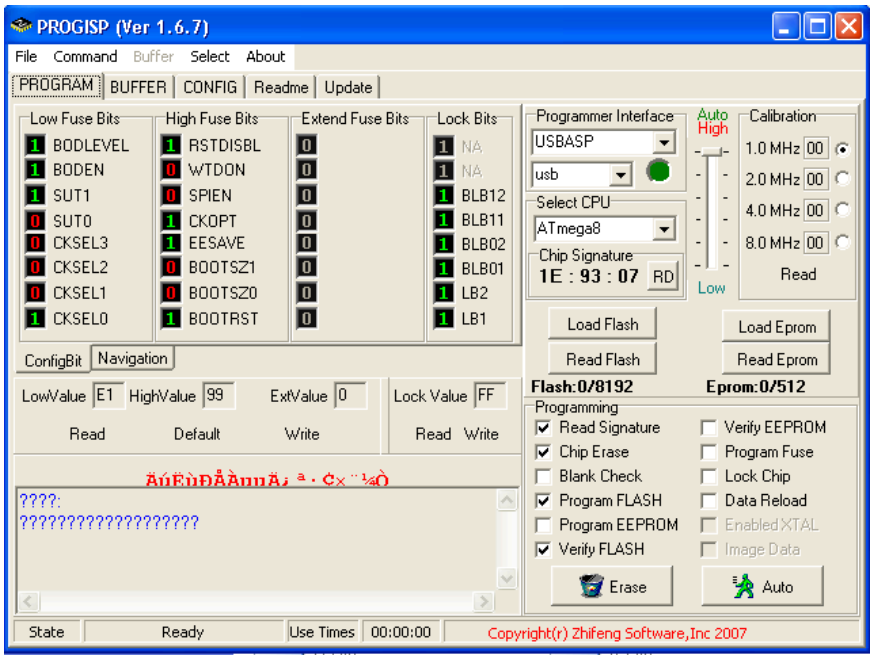


Figura 3.2. Pantalla principal PROGISP (Versión 1.6.7)

3.4. COMPILACIÓN DEL PROGRAMA

BASCOM AVR genera los archivos ".bas" los cuales serán compilados por el mismo programa de tal manera que se obtiene los archivos ".hex", estos archivos nos servirán para cargar los programas en los microcontroladores utilizando como software el **PROGISP (Versión 1.6.7)** y como hardware el **USBAS**

Dentro del programa BASCOM AVR realizamos la compilación de cada uno de los programas utilizando el icono de compilación como se puede observar en la figura 3.3 o también se lo puede compilar a través del teclado del computador con la tecla F7.

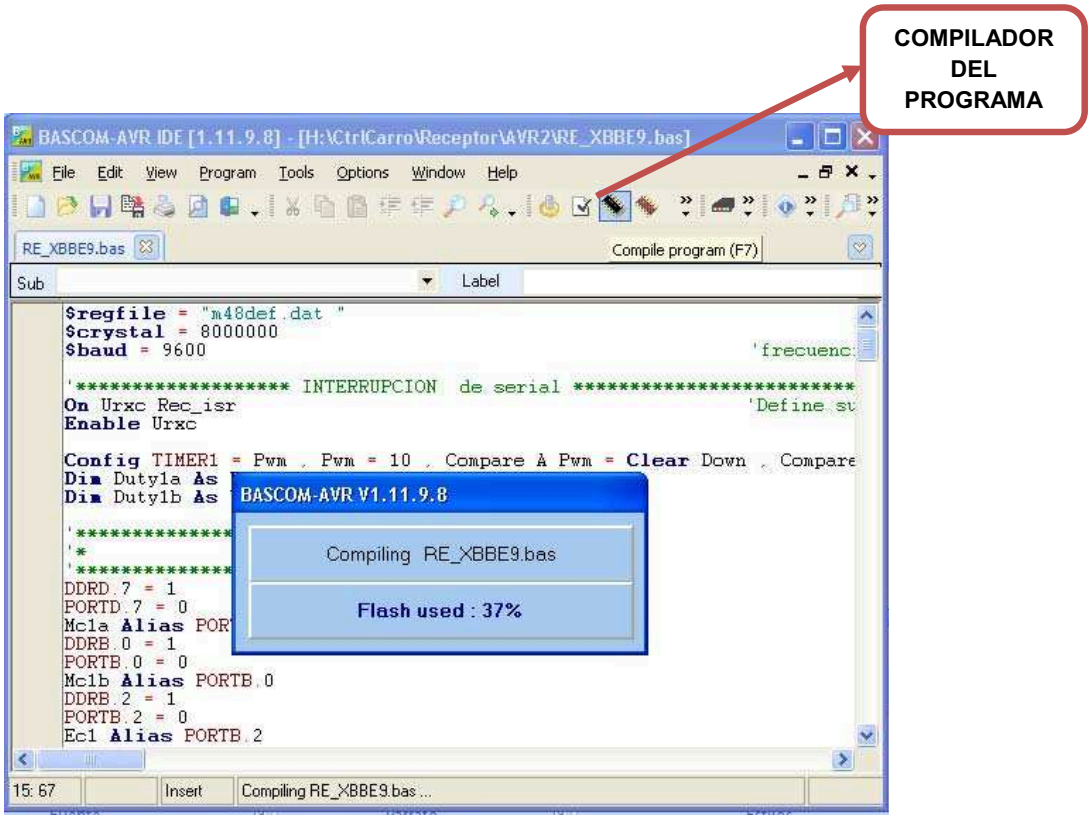


Figura 3.3. Compilación del programa realizado en BASCOM AVR

3.5. GRABACIÓN DE LOS PROGRAMAS EN LOS MICROCONTROLADORES

Como se explico anteriormente en este capítulo para poder grabar cada uno de los programas utilizaremos el programa PROGISP (Version 1.6.7) y el hardware USBAS.

Tenemos que configurar el programador PROGISP dentro del computador para los microcontroladores que utilizaremos tanto para el del circuito emisor que es el ATMEGA 164P, como para el circuito receptor que es el ATMEGA 48, para lo cual realizamos los siguientes procedimientos:

- Seleccionamos en la opción **Programmer Interface** (Interface del Programador), el tipo de Hardware con el que haremos el interfaz entre el programador y los microcontroladores, en este caso como ya mencionamos anteriormente es el USBASP, como se puede observar en la figura 3.4.

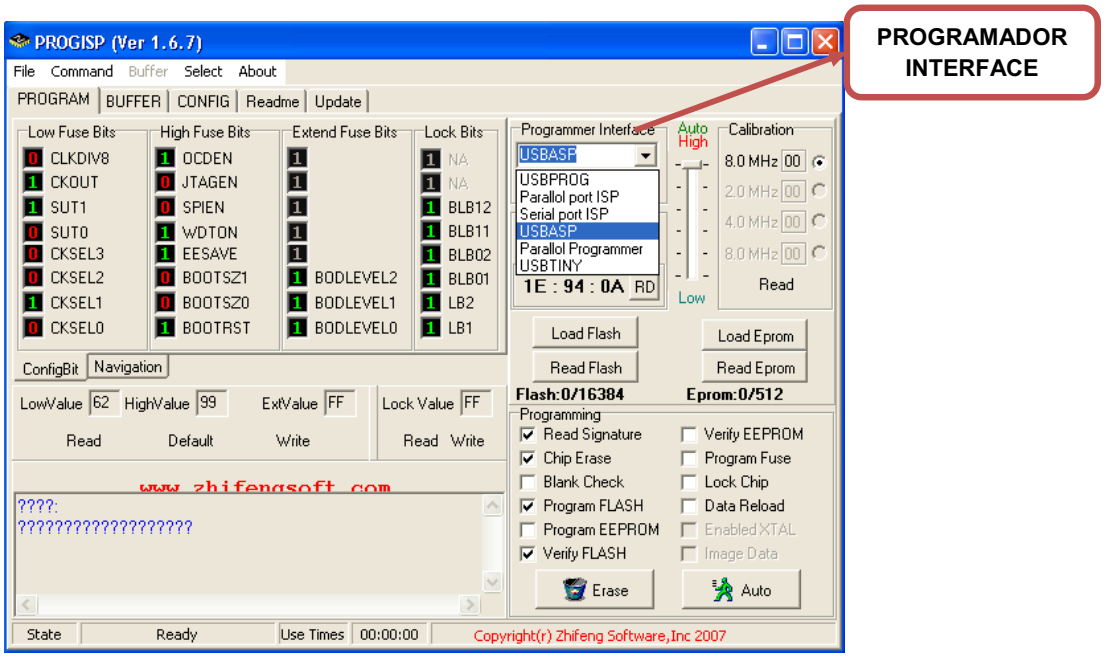


Figura 3.4. Selección del Interface del Programador

- Seleccionamos en la opción **Select CPU** (Selección CPU), como se puede observar en la figura 3.5 el tipo de microcontrolador a utilizarse.

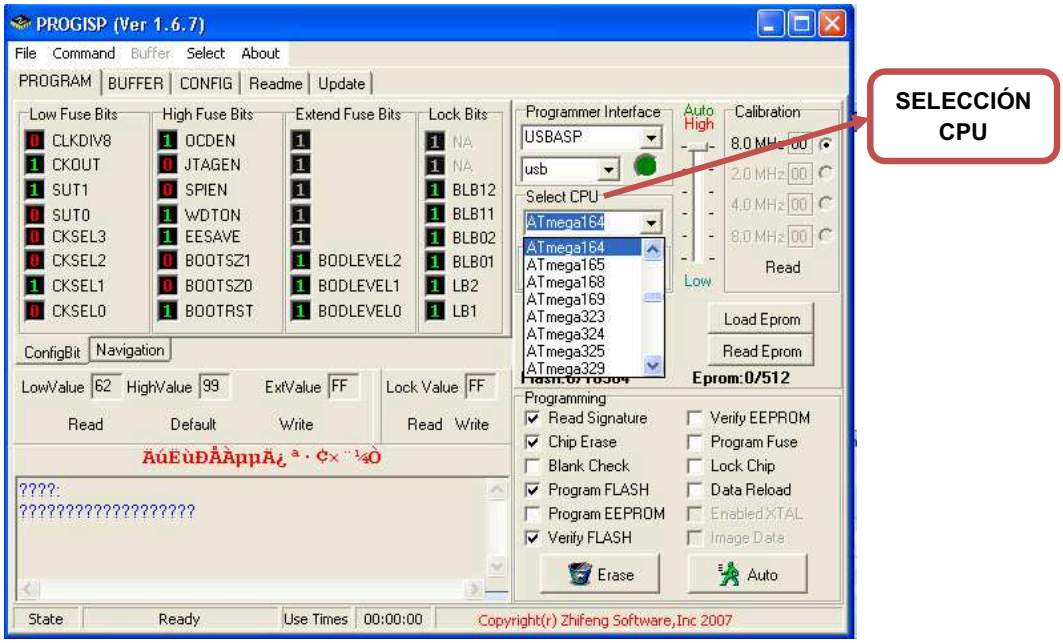


Figura 3.5. Selección del tipo de microcontroladores a utilizarse.

- Seleccionamos la opción **Navigation** (Navegación), en el cual encontraremos las diferentes opciones que por default en el programa esta activado, estos son llamados los fusibles de configuración, como se puede observar en la figura 3.6, para el microcontrolador ATMEGA 164.

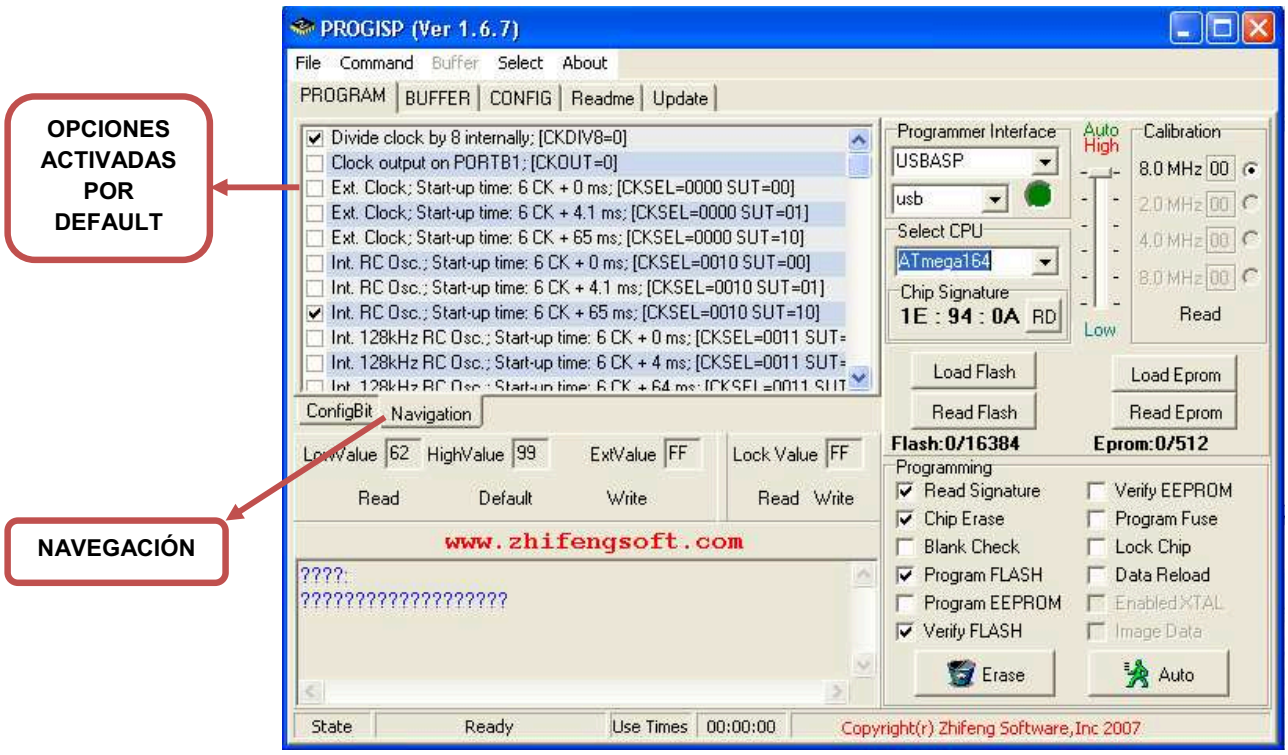


Figura 3.6. Opciones por default activados ATMEGA 164.

- Debemos activar los fusibles de cristal externo de 16K, 65 ms y desactivar la opción JTAG Interface Enabled, tanto para el microcontrolador ATMEGA 164, como para el ATMEGA 48, como se indica en la figura 3.7.

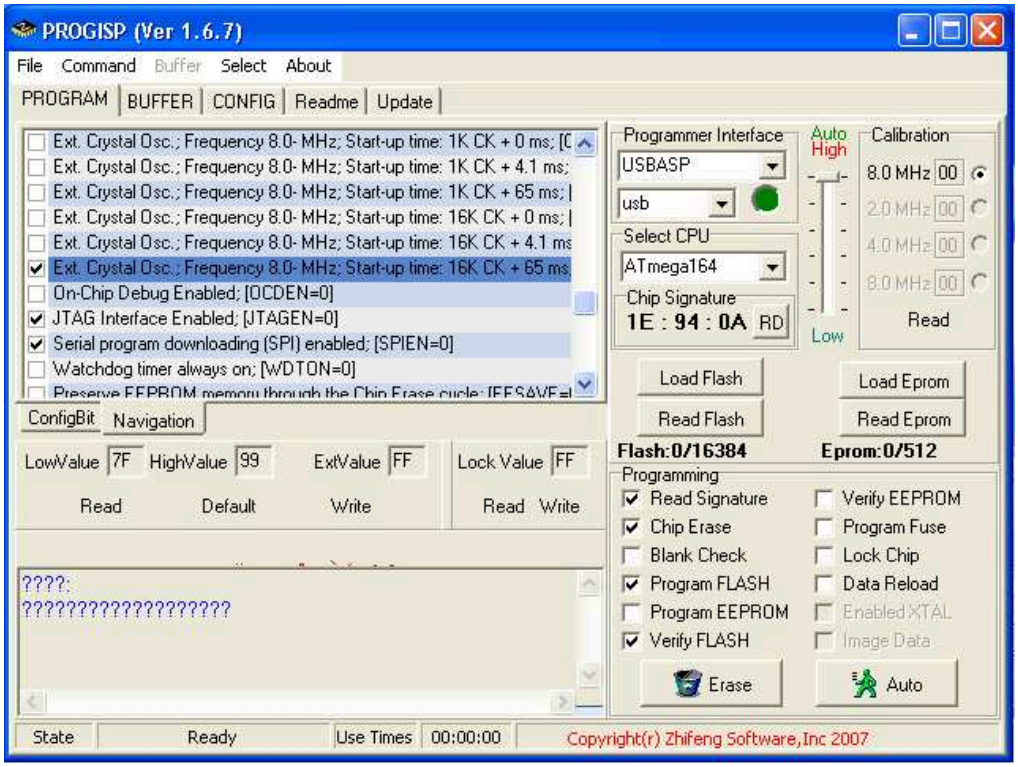


Figura 3.7. Cristal externo de 16K, 65 ms activado y la opción JTAG Interface Enabled desactivado del ATMEGA 164.

- Después seleccionamos la opción **Write** (Escribir), para guardar todos los cambios seleccionados y procedemos a grabar los programas en los microcontroladores utilizando la opción **Auto,** dichas opciones las podemos observar en la figura 3.8.

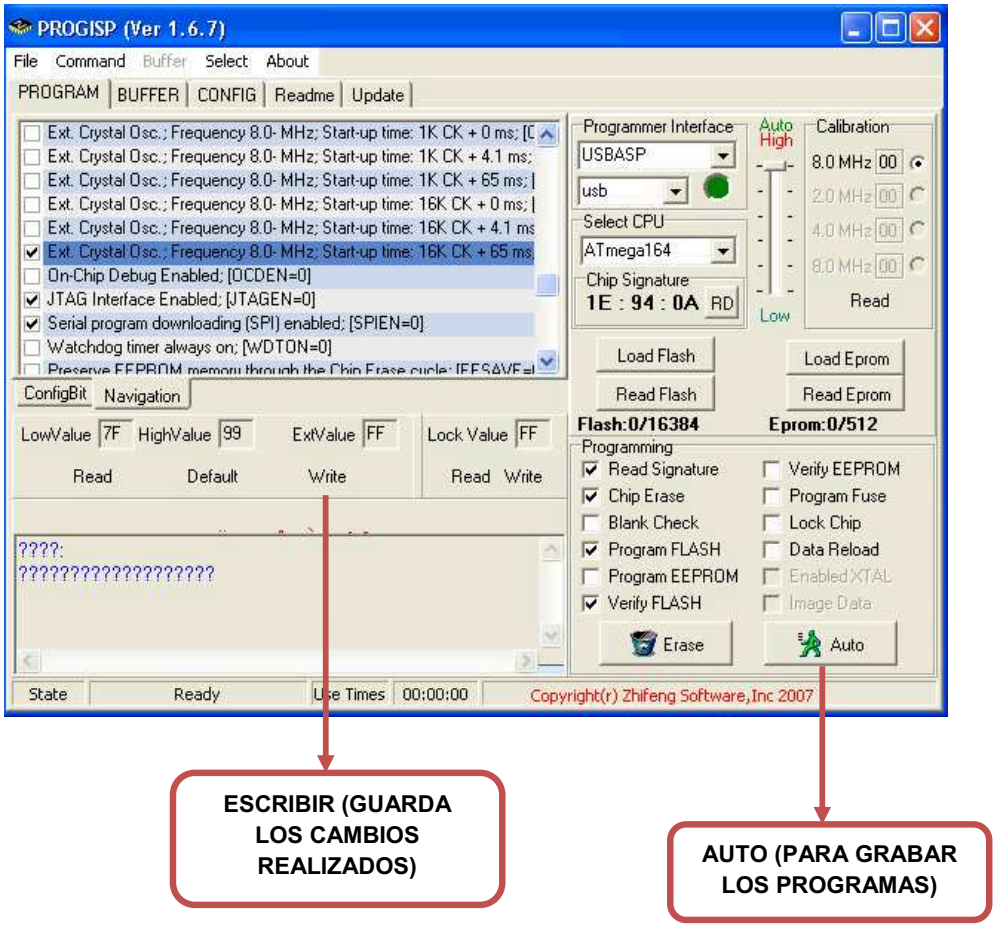


Figura 3.8. Opciones Write (escribir) y Auto del programa PROGISP

3.6. PROGRAMA DE LA TARJETA DEL CIRCUITO EMISOR (ATMEGA 164)

```
$regfile = "m164Pdef.dat "
$crystal = 8000000                          'frecuencia del cristal 8 Mhz
$baud = 9600                                'Velocidad del Com1
$baud1 = 9600                               'Velocidad del Com2

Config Com1 = Dummy , Synchrone = 0 , Parity = None , Stopbits = 1 , Databits
= 8 , Clockpol = 0
Config Com2 = Dummy , Synchrone = 0 , Parity = None , Stopbits = 1 , Databits
= 8 , Clockpol = 0

'Abre el segundo UART por hardware
Open "com2:" For Binary As #1


'****************** INTERRUPCION SERIAL ********************************
On Urxc Rec_isr                             'Define subrutina de Interrupcion
XBEE
Enable Urxc
On Urxc1 Rec_isr1                           'Define subrutina de Interrupcion
PC
Enable Urxc1

'************************************************
'*              LED1                *
'************************************************
Ddrd.4 = 1
Portd.4 = 0
```

```
    Led1 Alias Portd.4
    '*****************************************************************

    Dim B As Byte

    Dim Temporal1 As Byte
    Dim Temporal As String * 1
    Dim Serial2 As String * 20

    Waitms 500
    Cursor Off
    Cls

    Enable Interrupts

    '------------------------ PROGRAMA PRINCIPAL ---------------------------------
    Do
      Gosub Toggle_led1
    Loop
    '*****************************************************************
    Toggle_led1:
     Toggle Led1
     Waitms 20
Return
'*****************************************************************
        Rec_isr:                                    'INTERRUPCION DEL XBEE
        Disable Interrupts
        Disable Urxc
        B = Inkey()
        If B > 0 Then
```

```
        Temporal1 = B
        End If

        Do
        B = Inkey()                          'vacia el bufer del XBEE...
        Loop Until B = 0
        Temporal = Chr(temporal1)
        Print #1 , Temporal;
        Enable Interrupts
        Enable Urxc
        Return
'*************************************************************
        Rec_isr1:                            'INTERRUPCION DEL PC
        Disable Interrupts
        Disable Urxc1
        Input #1 , Serial2 Noecho
        Temporal = Mid(serial2 , 1 , 1)
        Print Temporal;
        Do
        B = Inkey(#1)
        Loop Until B = 0                     'vacia el bufer del micro....
        Enable Interrupts
        Enable Urxc1
        Return

End
```

3.7. PROGRAMA DE LA TARJETA DEL CIRCUITO RECEPTOR (ATMEGA 48)

```
    $regfile = "m48def.dat "
    $crystal = 8000000
    $baud = 9600                                'frecuencia del cristal 8 Mhz

    '****************** INTERRUPCION  de serial ******************************
    On Urxc Rec_isr                             'Define subrutina de Interrupcion
    XBEE
    Enable Urxc

Config Timer1 = Pwm , Pwm = 10 , Compare A Pwm = Clear Down ,
CompareB Pwm = Clear Down , Prescale = 8       ' configura el pwm al timer 1
    Dim Duty1a As Word
    Dim Duty1b As Word

'************************************************
    '*              Motor 1              *
'************************************************
Ddrd.7 = 1
Portd.7 = 0
Mc1a Alias Portd.7
Ddrb.0 = 1
Portb.0 = 0
Mc1b Alias Portb.0
Ddrb.2 = 1
Portb.2 = 0
Ec1 Alias Portb.2
```

```
'************************************************
'*                Motor 2                       *
'************************************************
Ddrd.6 = 1
Portd.6 = 0
Mc2a Alias Portd.6
Ddrd.5 = 1
Portd.5 = 0
Mc2b Alias Portd.5
Ddrb.1 = 1
Portb.1 = 0
Ec2 Alias Portb.1


'************************************************
'*                Led 1                         *
'************************************************
Ddrd.2 = 1
Portd.2 = 0
Led1 Alias Portd.2
'************************************************
'*                Led 2                         *
'************************************************
Ddrd.4 = 1
Portd.4 = 0
Led2 Alias Portd.4
```

```
'************************************************
'*                Led 3                 *
'************************************************
Ddrd.3 = 1
Portd.3 = 0
Led3 Alias Portd.3
'************************************************

Dim Flag1 As Bit                              'variable tipo bit
Dim B As Byte
Dim Temporal1 As Byte
Dim Contador As Word

Dim Temporal As String * 1
Dim Serial1 As String * 20

Const Delay1 = 300                             'carro

Flag1 = 0
Duty1a = 400
Duty1b = 400
Pwm1a = 0
Pwm1b = 0
Enable Interrupts                             'habilita la interrupcion serial del micro
al recibir datos por el xbee
'-------------------------------------------------------------------------------
Do
        Gosub Toggle_led1
        If Flag1 = 1 Then
            Print Chr(temporal1);
```

```
Flag1 = 0
Select Case Temporal1
  Case "a"
    Pwm1a = 1023
    Pwm1b = 1023
    Mc1a = 1
    Mc1b = 0
    'Ec1 = 1
    Mc2a = 1
    Mc2b = 0
    'Ec2 = 1
    Waitms Delay1
    Pwm1a = 0
    Pwm1b = 0
    Mc1a = 0
    Mc1b = 0
    'Ec1 = 0
    Mc2a = 0
    Mc2b = 0
    'Ec2 = 0
Case "b"
    Pwm1a = 1023
    Pwm1b = 1023
    Mc1a = 0
    Mc1b = 1
    'Ec1 = 1
    Mc2a = 0
    Mc2b = 1
    'Ec2 = 1
    Waitms Delay1
```

```
        Pwm1a = 0
        Pwm1b = 0
        Mc1a = 0
        Mc1b = 0
        'Ec1 = 0
        Mc2a = 0
        Mc2b = 0
        'Ec2 = 0
    Case "c"
        Pwm1a = 1023
        Pwm1b = 1023
        Mc1a = 1
        Mc1b = 0
        'Ec1 = 1
        Mc2a = 0
        Mc2b = 1
        'Ec2 = 1
        Waitms Delay1
'       Waitms Delay1
'       Waitms Delay1
        Pwm1a = 0
        Pwm1b = 0
        Mc1a = 0
        Mc1b = 0
        'Ec1 = 0
        Mc2a = 0
        Mc2b = 0
        'Ec2 = 0
    Case "d"
        Pwm1a = 1023
```

```
            Pwm1b = 1023
            Mc1a = 0
            Mc1b = 1
            'Ec1 = 1
            Mc2a = 1
            Mc2b = 0
            'Ec2 = 1
            Waitms Delay1
        '   Waitms Delay1
        '   Waitms Delay1
            Pwm1a = 0
            Pwm1b = 0
            Mc1a = 0
            Mc1b = 0
            'Ec1 = 0
            Mc2a = 0
            Mc2b = 0
            'Ec2 = 0
'-------------------------------------------------------------------------
        Case "e"
            Led2 = 1
            Led3 = 1
            Pwm1a = Duty1a
            Pwm1b = Duty1b
            Mc1a = 1
            Mc1b = 0
            'Ec1 = 1
            Mc2a = 1
            Mc2b = 0
            'Ec2 = 1
```

```
Case "f"
    Led2 = 1
    Led3 = 1
    Pwm1a = Duty1a
    Pwm1b = Duty1b
    Mc1a = 0
    Mc1b = 1
    'Ec1 = 1
    Mc2a = 0
    Mc2b = 1
    'Ec2 = 1
  Case "g"
    Led2 = 1
    Led3 = 0
    Pwm1a = 0
    Pwm1b = 1023
    Mc1a = 1
    Mc1b = 0
    'Ec1 = 1
    Mc2a = 0
    Mc2b = 0
    'Ec2 = 0
    Waitms Delay1
    'Waitms Delay1
    'Waitms Delay1
    Pwm1a = 0
    Pwm1b = 0
    Led2 = 0
    Led3 = 0
    Mc1a = 0
```

```
        Mc1b = 0
        'Ec1 = 0
        Mc2a = 0
        Mc2b = 0
        'Ec2 = 0
Case "h"
        Led2 = 0
        Led3 = 1
        Pwm1a = 1023
        Pwm1b = 0
        Mc1a = 0
        Mc1b = 0
        'Ec1 = 0
        Mc2a = 1
        Mc2b = 0
        'Ec2 = 1
        Waitms Delay1
        'Waitms Delay1
        'Waitms Delay1
        Led2 = 0
        Led3 = 0
        Pwm1a = 0
        Pwm1b = 0
        Mc1a = 0
        Mc1b = 0
        'Ec1 = 0
        Mc2a = 0
        Mc2b = 0
        'Ec2 = 0
```

```
    Case "i"
        Led2 = 0
        Led3 = 0
        Pwm1a = 0
        Pwm1b = 0
        Mc1a = 0
        Mc1b = 0
        'Ec1 = 0
        Mc2a = 0
        Mc2b = 0
        'Ec2 = 0
    Case "j"
         Duty1a = Contador
         Contador = Contador - 10
         If Contador <= 200 Then
         Duty1a = 200
         Duty1b = 200
  Else
    Duty1a = Contador
    Duty1b = Contador
  End If
Case "k"
   Duty1a = Contador
   Contador = Contador + 10
  If Contador >= 1000 Then
   Duty1a = 1000
    Duty1b = 1000
Else
    Duty1a = Contador
    Duty1b = Contador
```

```
      End If
        End Select
  End If
Loop
'****************************************************************
Toggle_led1:
    Toggle Led1
    Waitms 20
Return
'****************************************************************
Rec_isr:                                    'INTERRUPCION DEL uC
  Disable Interrupts
  Disable Urxc
  B = Inkey()
  If B > 0 Then
        Flag1 = 1
        Temporal1 = B
  End If
  Do
    B = Inkey()                             'vacia el bufer del micro....
  Loop Until B = 0
  Enable Urxc
  Enable Interrupts
Return
'****************************************************************
End
```

3.8. CONFIGURACION DE LOS MÓDULOS INALÁMBRICOS XBEE

Dentro de la adquisición de los módulos inalámbricos XBEE para el proyecto realizado, también debemos adquirir el programa **X-CTU** que es el software a utilizarse para configurar las opciones que necesitamos tener para el correcto funcionamiento de los módulos dentro del proyecto del control inalámbrico y como hardware el **XBee explorer USB** que permite conectar y utilizar cualquier módulo XBee directamente mediante un puerto USB. Es tan sencillo como conectar un módulo XBee, pinchar un cable mini USB al PC y tendremos acceso a los pines TX/RX del XBee.

En nuestro caso para el proyecto a realizar necesitamos configurar las siguientes opciones:

- Abrimos el programa X-CTU y una vez que nos haya detectado el puerto, en nuestro caso el puerto USB, debemos añadirlo en el programa a través de la sección **"User Com Ports"**(Puerto de comunicación de Usuario) colocamos el nombre que le hemos dado al usb como se puede observar en la Figura 3.9, el cual en este caso es el COM1.

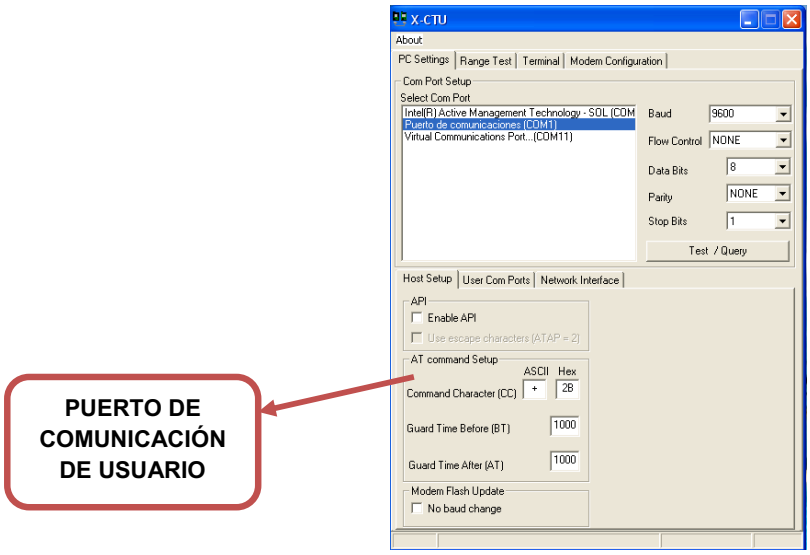
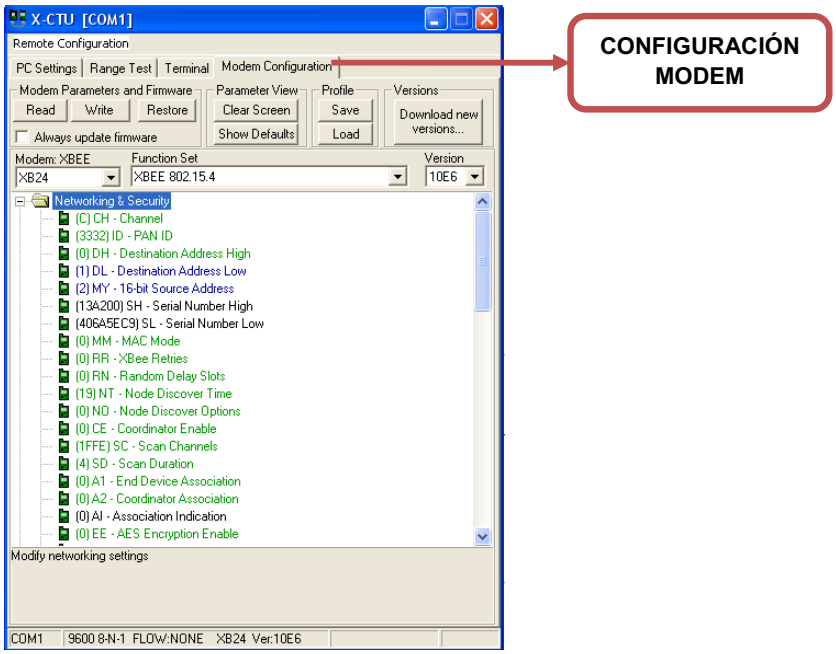


Figura 3.9. Selección del puerto de Comunicación de Usuario para configuración de los módulos XBEE

- Después de seleccionar la opción anterior mencionada establecemos comunicación con los módulos inalámbricos que se utilizara en los circuitos emisor y receptor respectivamente con la opción "**Modem Configuration"** (Configuración Modem), con la cual se despliega las múltiples opciones que nosotros podemos modificar y así poder obtener los resultados requeridos podemos observar que hay opciones con letras de color verde los cuales se aquellas que se las puede modificar

  Para nuestro proyecto solamente debemos modificar el DL y MY que son las opciones de destino y origen, el DL (1) y el MY (2) para el modulo inalámbrico del circuito emisor y DL (2) y el MY (1) para el modulo inalámbrico del circuito receptor que es la configuración de la aplicación que requerimos la cual es la aplicación punto a punto, determinamos una dirección de origen y una dirección de destino como podemos observar en la Figura 3.10.



Figuras 3.10. Configuración del Modulo XBEE.

CAPITULO IV

SISTEMA DE CONTROL INALAMBRICO DESDE EL COMPUTADOR.

4.1. INTRODUCCION

Después de haber realizado y explicado la construcción del Hardware y el Software del control del sistema inalámbrico del prototipo de carrito, tanto del circuito emisor como el del receptor, como punto final y para la culminación del proyecto propuesto en el presente capitulo trataremos acerca del control desde el computador de todo el sistema inalámbrico, explicaremos el programa a utilizarse y los requerimientos que necesitamos para aquello.

4.2. REQUERIMIENTOS A UTILIZARSE PARA EL CONTROL DEL SISTEMA INALAMBRICO DESDE EL COMPUTADOR.

Como primer paso para el control del sistema inalámbrico debemos seleccionar un programa adecuado que se instalara en cualquier computador e inclusive en cualquier computadora portátil, para lo cual hemos seleccionado un programa fácil de utilizar y programar el cual es el VISUAL BASIC 6.0.

A continuación se dará una breve introducción del programa **VISUAL BASIC 6.0** a utilizarse y el programa que se llevo a cabo para el control inalámbrico.

4.2.1. VISUAL BASIC 6.0 [7]

4.2.1.1 INTRODUCCION.

Visual Basic es un lenguaje de programación dirigido por eventos, desarrollado por el alemán Alan Cooper para Microsoft. Este lenguaje de programación es un dialecto de BASIC, con importantes agregados. Su primera versión fue presentada en 1991, con la intención de simplificar la programación

[7] **www.lawebdelprogramador.com**

utilizando un ambiente de desarrollo completamente gráfico que facilitara la creación de interfaces gráficas y, en cierta medida, también la programación misma.

La última versión fue la 6, liberada en 1998, para la que Microsoft extendió el soporte de este lenguaje hasta marzo de 2008.

En 2001 Microsoft propuso abandonar el desarrollo basado en la API Win32 y pasar a un framework o marco común de librerías, independiente de la versión del sistema operativo, .NET Framework, a través de Visual Basic .NET (y otros lenguajes como C Sharp (C#) de fácil transición de código entre ellos); fue el sucesor de Visual Basic 6.

Si bien Visual Basic es de propósito general, también permite el desarrollo de aplicaciones de bases de datos usando Data Access Objects, Remote Data Objects, o ActiveX Data Objects.

Visual Basic (Visual Studio) contiene un entorno de desarrollo integrado o IDE que incluye un editor de textos para edición del código, un depurador, un compilador (y enlazador) y un constructor de interfaz gráfica o GUI.

4.2.1.2. EL ENTORNO INTEGRADO DE DESARROLLO (IDE)

Cuando se inicia Visual Basic, se crea un proyecto nuevo con un formulario. El IDE de Visual Basic consta de los siguientes elementos:

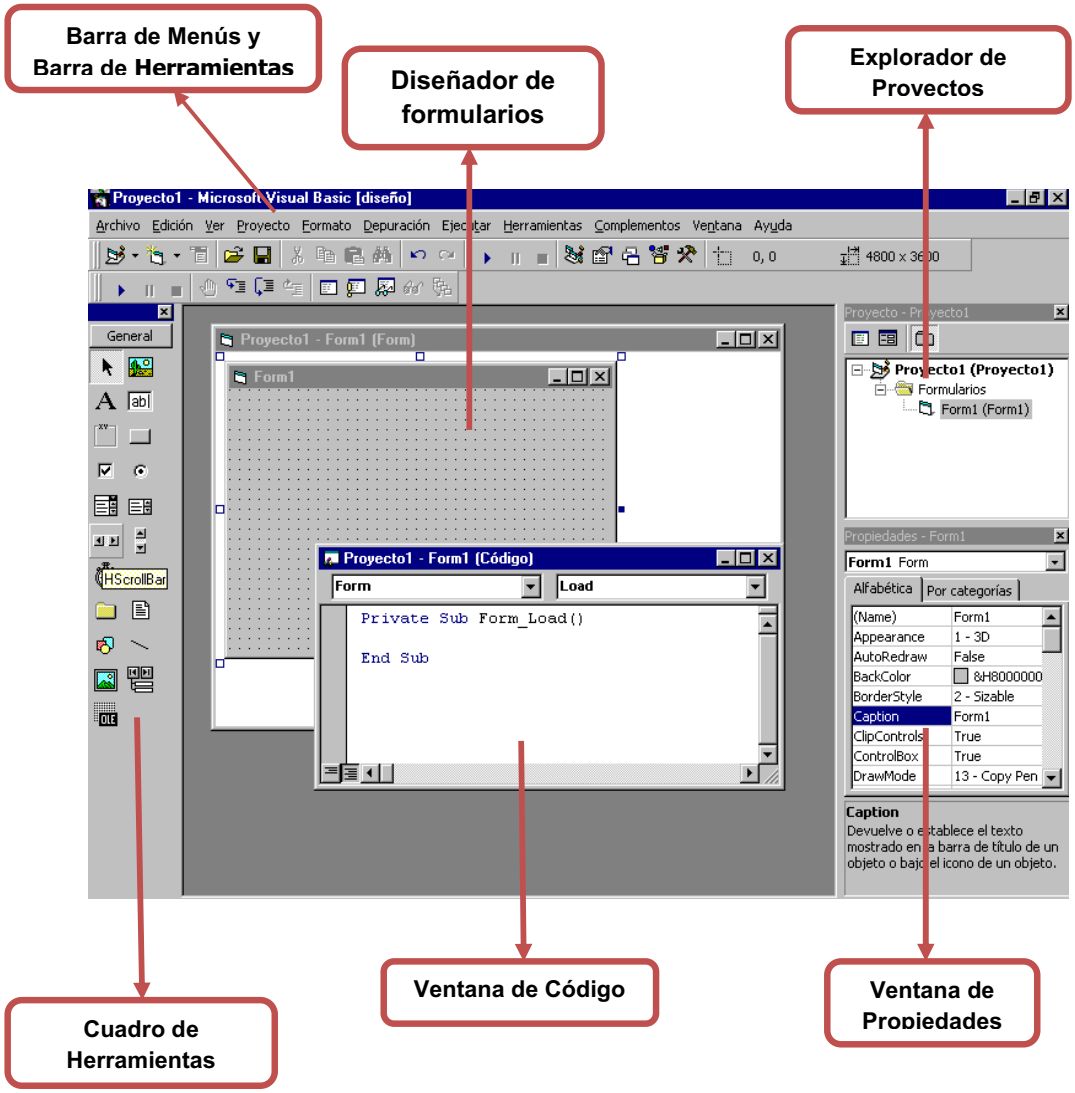


Figura 4.1. Entorno integrado de desarrollo (IDE)

Barra de Menús

Presenta los comandos que se usan para trabajar con Visual Basic. Además de los menús estándar **Archivo, Edición, Ver, Ventana y Ayuda,** contiene otros menús para tener acceso a funciones específicas de programación, como **Proyecto, Formato** o **Depuración.**

Barra de Herramientas

Permite un acceso directo (solo un clic) a muchas de las operaciones más frecuentes utilizadas durante el desarrollo de aplicaciones.

Cuadro de Herramientas

Contiene todos los objetos y controles que se pueden añadir a los formularios para crear aplicaciones.

Diseñador de Formularios

Funciona como una ventana en la que se puede personalizar el diseño de la interfaz de usuario (ventana) de una aplicación.

Explorador de Proyectos

Lista de los archivos (formularios, módulos, etc.) del proyecto actual. Un **Proyecto** es una colección de archivos que utiliza para construir una aplicación.

Ventana de Propiedades

Lista los valores de las propiedades del formulario o control seleccionado que pueden ser modificados durante el diseño del formulario o control.

Ventana de Código

Funciona como un editor para escribir el código (sentencias) de la aplicación.

4.2.1.3. OPERADORES

Aritméticos

^	Exponenciación
*	Multiplicación
/	División
	División entera
Mod	Residuo entero (Ejm: A Mod B)
+	Suma
-	Resta
&	Concatenación de cadenas

Comparación

=	Igual
<>	Distinto
➢	Menor que
<=	Menor o igual
>=	Mayor o igual
Like	Compara dos cadenas
*	Cero o más caracteres (Ejm: cad Like “ma*”)
?	Cualquier carácter
#	Cualquier dígito (0-9)
	[lista] cualquier carácter en lista
	[¡lista] cualquier carácter que no esta en lista
Is	Usado para comparar dos variables de referencia a objetos

Lógicos

And	“Y” lógico
Or	“O” lógico
Xor	“O” Exclusivo
Not	Negación

4.2.1.4. ESTRUCTURAS DE CONTROL

Las estructuras de control le permiten controlar el flujo de ejecución del programa. Tenemos dos tipos de estructuras de control:

- Estructuras de decisión
- Estructuras de bucle

4.2.1.4.1. Estructuras de Decisión

Los procedimientos de Visual Basic pueden probar condiciones y, dependiendo de los resultados, realizar diferentes operaciones. Entre las estructuras de decisión que acepta Visual Basic se incluyen las siguientes:

- *If...Then*
- *If...Then...Else*
- *Select Case*

If...Then

Use la estructura **If...Then** para ejecutar una o más instrucciones basadas en una condición. Puede utilizar la sintaxis de una línea o un bloque de varias líneas:

- ***If*** *condición* ***Then*** *Sentencias*
- ***If*** *condición* ***Then***

  Sentencias

  End If

Condición normalmente es una comparación, pero puede ser cualquier expresión que dé como resultado un valor numérico. Visual Basic interpreta este valor como **True** o **False**; un valor numérico cero es **False** y se considera **True** cualquier valor numérico distinto de cero. Si ***condición*** es **True,** Visual Basic ejecuta todas las *sentencias* que siguen a la palabra clave **Then.** Puede utilizar sintaxis de una línea o de varias líneas para ejecutar una sentencia basada en una condición, los siguientes dos ejemplos son equivalentes:

- ```
  If cualquierFecha < Now Then CualquierFecha = Now
  ```

- ```
  If cualquierFecha < Now Then
  CualquierFecha = Now
  End If
  ```

4.2.1.4.2 Estructuras de Repetición

Las estructuras de repetición o bucle le permiten ejecutar una o más líneas de código repetidamente. Las estructuras de repetición que acepta Visual Basic son:

- Do...Loop
- For...Next

- For Each...Next
- **Do...Loop**

Utilice el bucle **Do** para ejecutar un bloque de sentencias un número indefinido de veces. Hay algunas variantes en la sentencia **Do...Loop,** pero cada una evalúa una condición numérica para determinar si continúa la ejecución. Como ocurre con **If...Then,** la ***condición*** debe ser un valor o una expresión que dé como resultado **False** (cero) o **True** (distinto de cero).

En el siguiente ejemplo de **Do...Loop,** las **sentencias** se ejecutan siempre y cuando ***condición*** sea **True:**

Do While *condición*

Sentencias

Loop

Cuando Visual Basic ejecuta este bucle **Do,** primero evalúa ***condición.*** Si ***condición*** es **False** (cero), se salta todas las ***sentencias.*** Si es **True** (distinto de cero) Visual Basic ejecuta las ***sentencias***, vuelve a la instrucción **Do While** y prueba la condición de nuevo.

Por tanto, el bucle se puede ejecutar cualquier número de veces, siempre y cuando ***condición*** sea distinta de cero o **True.** Nunca se ejecutan las ***sentencias*** si ***condición*** es **False** inicialmente. Por ejemplo, este procedimiento cuenta las veces que se repite una cadena destino dentro de otra cadena repitiendo el bucle tantas veces como se encuentre la cadena de destino:

```
Function ContarCadenas  (cadenalarga,  destino)
    Dim posición, contador
```

```
        posición = 1
        Do While InStr (posición, cadenalarga,  destino)
            posición = InStr (posición, cadenalarga,  destino)+1
        contador  =  contador  +  1
        Loop
    ContarCadenas  =  contador
End Function
```

Si la cadena destino no está en la otra cadena, **InStr** devuelve 0 y no se ejecuta el bucle.

Otra variante de la instrucción **Do...Loop** ejecuta las ***sentencias*** primero y prueba la ***condición*** después de cada ejecución. Esta variación garantiza al menos una ejecución de ***sentencias:***

Do

Sentencias

Loop While *condición*

Hay otras dos variantes análogas a las dos anteriores, excepto en que repiten el bucle siempre y cuando ***condición*** sea **False** en vez de **True.**

Hace el bucle cero o más veces	**Hace el bucle al menos una vez**
Do Until *condición*	***Do***
Sentencias	*Sentencias*
Loop	***Loop Until*** *condición*

For...Next

Los bucles **Do** funcionan bien cuando no se sabe cuántas veces se necesitará ejecutar las ***sentencias*** del bucle. Sin embargo, cuando se sabe que se va a ejecutar las ***sentencias*** un número determinado de veces, es mejor elegir el bucle **For...Next.** A diferencia del bucle **Do,** el bucle **For** utiliza una variable llamada ***contador*** que incrementa o reduce su valor en cada repetición del bucle. La sintaxis es la siguiente:

For *contador = iniciar* ***To*** *finalizar [****Step*** *incremento]*

Sentencias

Next *[contador]*

4.3. PROGRAMA CREADO PARA EL CONTROL DEL SISTEMA INALÁMBRICO DESDE UN COMPUTADOR.

```
Private Sub Form_Load()
    Form1.Lblportico.Caption = "Com" & Form1.MSComm1.CommPort
    Form1.LblVelocidad.Caption = Left$(Form1.MSComm1.Settings,
Len(Form1.MSComm1.Settings) - 6)
    isButton1.Visible = False
    isButton2.Visible = False
    isButton3.Visible = False
    isButton4.Visible = False
    isButton5.Visible = False
    isButton6.Visible = False
    isButton7.Visible = False
    isButton8.Visible = False
    isButton9.Visible = False
    isButton13.Visible = False
    isButton14.Visible = False
End Sub
```

```
Private Sub isButton1_Click()
Dim z2, z1 As Single
    isButton1.Enabled = False
    Form1.Label4.Caption = "d"
    Form1.MSComm1.Output = "d" + Chr(13)
    Do
        Ret = DoEvents()
    Loop Until Form1.MSComm1.OutBufferCount = 0
    z2 = Timer
    Do
        z1 = Timer
    Loop While (z1 - z2) < 0.05
    isButton1.Enabled = True
End Sub

Private Sub isButton10_Click()
' Chequea si esta abierto o No el portico COM
Dim OpenFlag
Dim dato1 As String '
OpenFlag = Form1.MSComm1.PortOpen
```

```
    If OpenFlag Then
    Else
        Form1.MSComm1.PortOpen = Not Form1.MSComm1.PortOpen
    End If

' Desabilita botones de propiedades y de transmision
    isButton1.Visible = True
    isButton2.Visible = True
    isButton3.Visible = True
    isButton4.Visible = True
    isButton5.Visible = True
    isButton6.Visible = True
    isButton7.Visible = True
    isButton8.Visible = True
    isButton9.Visible = True
    isButton10.Enabled = False
    isButton12.Enabled = False
    isButton13.Visible = True
    isButton14.Visible = True
End Sub
```

```
Private Sub isButton11_Click()
'Cierra el pórtico serial COM a la salida del programa
    If Form1.MSComm1.PortOpen Then Form1.MSComm1.PortOpen = False
    End
End Sub

Private Sub isButton12_Click()
    Form2.Show vbModal
End Sub

Private Sub isButton13_Click()
Dim z2, z1 As Single
    isButton13.Enabled = False
    Form1.Label4.Caption = "j"
    Form1.MSComm1.Output = "j" + Chr(13) ' Subir
    Do
        Ret = DoEvents()
    Loop Until Form1.MSComm1.OutBufferCount = 0
    z2 = Timer
    Do
```

```
    z1 = Timer
  Loop While (z1 - z2) < 0.05
  isButton13.Enabled = True
End Sub

Private Sub isButton14_Click()
Dim z2, z1 As Single
  isButton14.Enabled = False
  Form1.Label4.Caption = "k"
  Form1.MSComm1.Output = "k" + Chr(13) ' Arriba
  Do
    Ret = DoEvents()
  Loop Until Form1.MSComm1.OutBufferCount = 0
  z2 = Timer
  Do
    z1 = Timer
  Loop While (z1 - z2) < 0.05
  isButton14.Enabled = True
End Sub
```

```
Private Sub isButton2_Click()
Dim z2, z1 As Single
    isButton2.Enabled = False
    Form1.Label4.Caption = "h"
    Form1.MSComm1.Output = "h" + Chr(13) ' Arriba
    Do
        Ret = DoEvents()
    Loop Until Form1.MSComm1.OutBufferCount = 0
    z2 = Timer
    Do
        z1 = Timer
    Loop While (z1 - z2) < 0.05
    isButton2.Enabled = True
End Sub

Private Sub isButton3_Click()
Dim z2, z1 As Single
    isButton3.Enabled = False
    Form1.Label4.Caption = "c"
    Form1.MSComm1.Output = "c" + Chr(13)
```

```
    Do
        Ret = DoEvents()
    Loop Until Form1.MSComm1.OutBufferCount = 0
    z2 = Timer
    Do
        z1 = Timer
    Loop While (z1 - z2) < 0.05
    isButton3.Enabled = True
End Sub

Private Sub isButton4_Click()
Dim z2, z1 As Single
    isButton4.Enabled = False
    Form1.Label4.Caption = "g"
    Form1.MSComm1.Output = "g" + Chr(13) ' Arriba
    Do
        Ret = DoEvents()
    Loop Until Form1.MSComm1.OutBufferCount = 0
    z2 = Timer
```

```
    Do
        z1 = Timer
    Loop While (z1 - z2) < 0.05
    isButton4.Enabled = True
End Sub

Private Sub isButton5_Click()
Dim z2, z1 As Single
    isButton5.Enabled = False
    Form1.Label4.Caption = "i"
    Form1.MSComm1.Output = "i" + Chr(13) ' Arriba
    Do
        Ret = DoEvents()
    Loop Until Form1.MSComm1.OutBufferCount = 0
    z2 = Timer
    Do
        z1 = Timer
    Loop While (z1 - z2) < 0.05
    isButton5.Enabled = True
End Sub
```

```
Private Sub isButton6_Click()
Dim z2, z1 As Single
    isButton6.Enabled = False
    Form1.Label4.Caption = "a"
    Form1.MSComm1.Output = "a" + Chr(13) ' Arriba
    Do
        Ret = DoEvents()
    Loop Until Form1.MSComm1.OutBufferCount = 0
    z2 = Timer
    Do
        z1 = Timer
    Loop While (z1 - z2) < 0.05
    isButton6.Enabled = True
End Sub
Private Sub isButton7_Click()
Dim z2, z1 As Single
    isButton7.Enabled = False
    Form1.Label4.Caption = "e"
    Form1.MSComm1.Output = "e" + Chr(13) ' Arriba
    Do
```

```
    Ret = DoEvents()
  Loop Until Form1.MSComm1.OutBufferCount = 0
  z2 = Timer
  Do
    z1 = Timer
  Loop While (z1 - z2) < 0.05
  isButton7.Enabled = True
End Sub

Private Sub isButton8_Click()
Dim z2, z1 As Single
  isButton8.Enabled = False
  Form1.Label4.Caption = "f"
  Form1.MSComm1.Output = "f" + Chr(13) ' Arriba
  Do
    Ret = DoEvents()
  Loop Until Form1.MSComm1.OutBufferCount = 0
  z2 = Timer
  Do
    z1 = Timer
```

```
    Loop While (z1 - z2) < 0.05
    isButton8.Enabled = True
End Sub

Private Sub isButton9_Click()
Dim z2, z1 As Single
    isButton9.Enabled = False
    Form1.Label4.Caption = "b"
    Form1.MSComm1.Output = "b" + Chr(13)
    Do
        Ret = DoEvents()
    Loop Until Form1.MSComm1.OutBufferCount = 0
    z2 = Timer
    Do
        z1 = Timer
    Loop While (z1 - z2) < 0.05
    isButton9.Enabled = True
End Sub
```

4.4. CONFIGURACIÓN DEL CIRCUITO EMISOR CON EL PROGRAMA DEL COMPUTADOR

Una vez realizado el programa en el VISUAL BASIC 6.0 el siguiente paso a seguir es la configuración de conexión del circuito emisor con la computadora o la portátil, según el requerimiento a elegir.

A continuación realizamos los siguientes pasos:

- Realizamos la conexión del circuito emisor con el computador con un cable USB, el cual debe ser conectado en un puerto USB de la computadora libre y a nuestra elección.

- Una vez realizada la conexión el computador detecta un nueva hardware como podemos ver en la Figura 4.2, el cual inmediatamente requiere ser instalado para lo cual utilizamos el driver respectivo para aquello.

Figura 4.2. Detección de nuevo hardware USB-232.

- Instalamos el drive requerido para que el computador instale un puerto serial virtual que nos servirá para la conexión entre el circuito emisor y la computadora como se puede observar en la figura 4.3.

- Una vez instalado el driver está listo para ser utilizado, y para poder verificar que se ha creado el puerto serial virtual correctamente, vamos a **Mi PC** hacemos click derecho en propiedades, en la cual se despliega la

pantalla de propiedades del sistemas , hacemos click en **Hardware**, de ahí en **Administrador de Dispositivos** y ahí podemos observar en **Puertos (COM&LPT)** el Puerto de comunicación Virtual que hemos instalado en nuestro caso es el **COM5**, como se puede observar en la figura 4.4, con el que configuraremos posteriormente cuando ya establezcamos comunicación mediante el programa realizado en el **VISUAL BASIC 6.0**

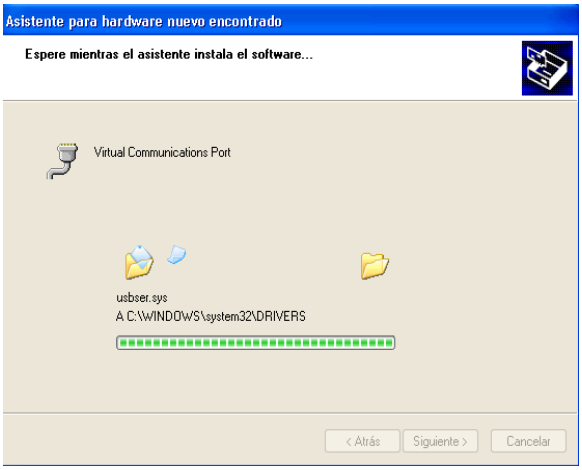


Figura 4.3. Instalación de los Driver para puerto de comunicación virtual

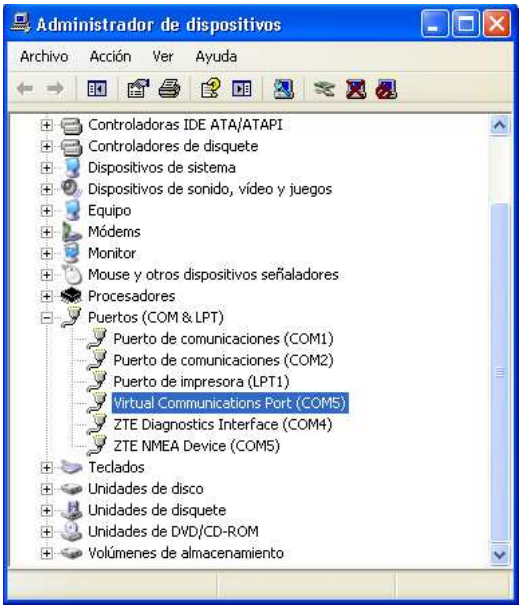


Figura 4.4. Verificación del puerto de comunicación virtual (COM5)

- Finalmente debemos configurar en el programa de comunicación que realizamos en el VISUAL BASIC 6.0 el puerto serial que creamos anteriormente que es el **COM5**

4.5. FORMAS DE ACCESO AL PROGRAMA DE COMUNICACIONES ENTRE EL COMPUTADOR Y EL CIRCUITO EMISOR DEL SISTEMA DE CONTROL INALÁMBRICO.

Existen dos formas de acceder al programa de comunicaciones, mediante el mismo programa **VISUAL BASIC 6.0** que debe estar instalado en la computadora o maquina portátil a utilizar como control o también con un archivo ejecutable creado desde el mismo **VISUAL BASIC 6.0.**

4.5.1. ACCESO DESDE EL PROGRAMA VISUAL BASIC.

Podemos establecer la comunicación entre el computador y el circuito emisor mediante el programa VISUAL BASIC 6.0 con el que creamos el mismo programa que controlara al carrito, para lo cual es necesario que el computador o equipo portátil a utilizar como control este instalado el programa.

Cumplido este requisito se podrá abrir el programa realizado en el mismo **VISUAL BASIC 6.0** en la cual se despliega la el programa y en la opción play (empezar) podemos visualizar la pantalla dinámica creada para el control del prototipo como se puede observar en la figura 4.5.

Y finalmente para establecer el control remoto desde el computador seleccionamos la opción **puerto serial** y cambiamos el puerto establecido por default por el creado por nosotros anteriormente es decir **COM5** como se observa en la figura 4.6., y hacemos click en la opción Comunicación en la cual se despliega finalmente la pantalla dinámica que se utilizara para el control del prototipo desde el computador como se observa en la figura 4.7.

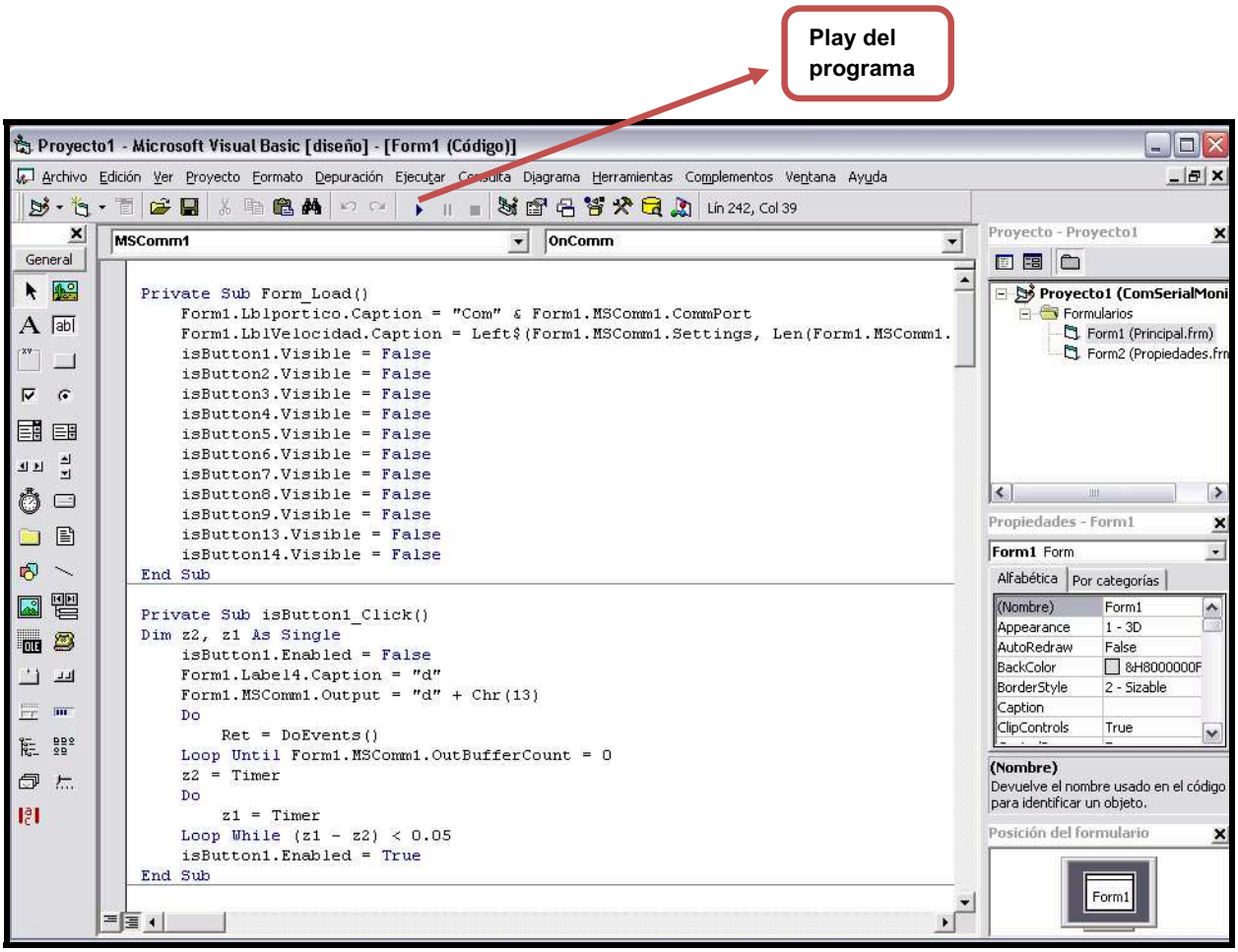


Figura 4.5. Pantalla dinámica de control

Figura 4.6. Selección de puerto serial instalado

Figura 4.7.Pantalla de control Remoto desde el computador

4.5.2. ACCESO MEDIANTE UN ARCHIVO EJECUTABLE.

Esta opción fue creada y establecida con el propósito de que en cualquier equipo de computación ya sea equipo de escritorio como en una portátil podamos controlar al prototipo sin necesidad de que estos equipos este previamente instalados el **VISUAL BASIC 6.0**, con el archivo ejecutable solo basta instalar mediante cualquier dispositivo de almacenamiento en el equipo a utilizar previamente instalado el puerto de comunicación virtual como se indico anteriormente y estar listo para servir de control remoto del prototipo.

El archivo ejecutable fue creado desde el mismo programa VISUAL BASIC 6.0 y los pasos a seguir para su instalación son los mismos que cualquier programa a instalar en un computador solamente que el icono a ser utilizado es el que se indica en la figura 4.8 , una vez que hacemos click en el mencionado icono se despliega la misma pantalla dinámica que sale en el **VISUAL BASIC 6.0** (Ver figura 4.5)**,** y de igual manera como en el anterior acceso debemos cambiar el puerto de comunicación virtual por el que creamos anteriormente el cual es el **COM5** (Ver figura 4.6), y obtenemos la misma pantalla dinámica para controlar el prototipo desde el computador (Ver figura 4.7).

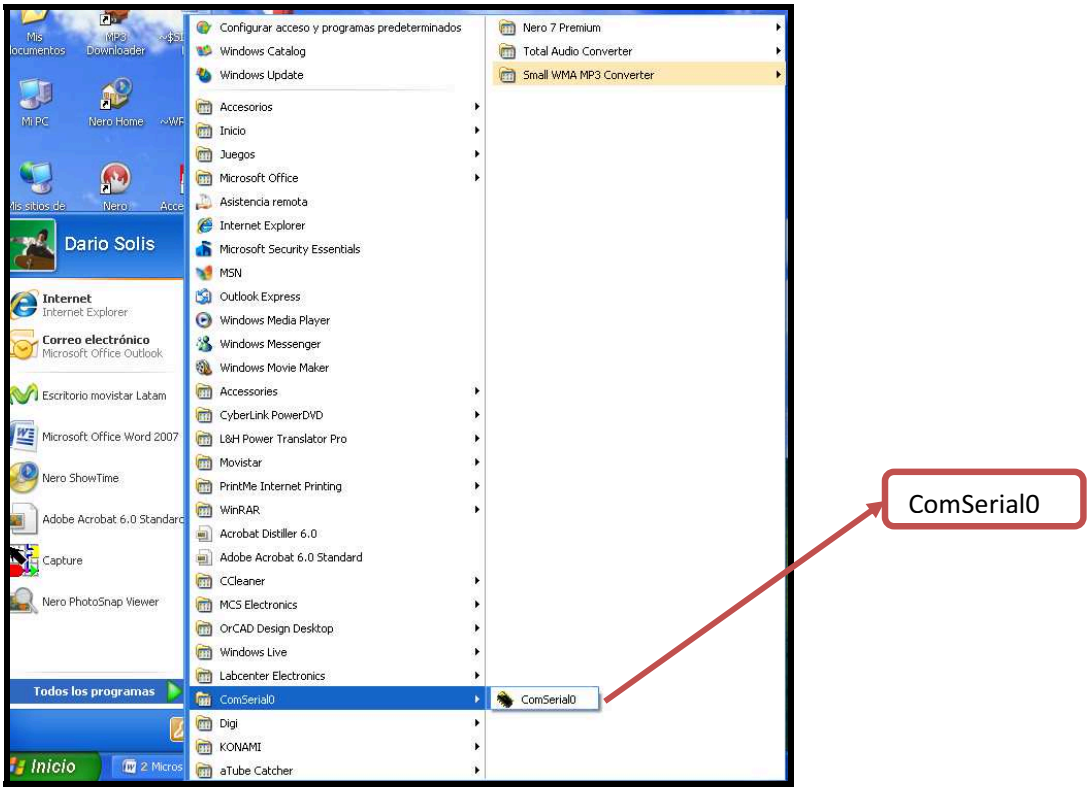


Figura 4.8. Icono ComSerial

CAPITULO V

PRUEBAS Y RESULTADOS

5.1. INTRODUCCIÓN.

Las pruebas y los resultados son de vital importancia en el ciclo de desarrollo tanto del hardware como del software, consistiendo en la revisión final de los requerimientos, análisis y diseño para finalmente realizar la implementación. El objetivo de las pruebas es encontrar fallas o errores para luego hacer una depuración del sistema y así asegurar que el proyecto ha sido desarrollado de acuerdo a los requerimientos y que todos los errores han sido detectados.

5.2. PRUEBAS REALIZADAS A LOS CIRCUITOS EMISOR Y RECEPTOR.

Antes de realizar el hardware y el software en las placas electrónicas y el programa realizado respectivamente tuvimos que para aquello realizar su montaje previamente en los protoboard y así poder determinar su validez o las modificaciones que se debería realizar para su correcto funcionamiento, con el montaje de los circuitos en el protoboard pudimos determinar el funcionamiento de una de las partes más importantes del proyecto que son los módulos inalámbrico XBEE verificando su transmisión y recepción , para aquello se coloco un led en cada circuito de tal manera que cuando se transmita una instrucción desde el emisor el led se encienda y de igual manera y al mismo tiempo el led del circuito receptor y así se da por hecho que si hay una correcta transmisión y recepción, este detalle también se lo realizo en los circuitos emisor y receptor finales.

De igual manera se realizo las pruebas con los programas de los microcontroladores ya que se coloco motores DC en el circuito receptor que se monto en el protoboard y así verificamos los sentidos de movimiento de los motores que posteriormente será los motores del carrito.

5.3. PRUEBAS DEL CONTROL REMOTO DESDE EL COMPUTADOR.

Como se puede observar en la figura 4.9 se tiene la pantalla del control remoto del prototipo realizado en la que podemos observar que se tiene varias opciones para el direccionamiento del carrito a control remoto a la derecha se encuentra las opciones que son para que la trayectoria del carrito sea ilimitado es decir que para que el carrito se detenga tenemos que oprimir o hacer click en el botón parar o también en los botones Derecha o izquierda que son para direccionar el lado en el que el carrito se va a dirigir, cabe mencionar que el alcance de nuestro prototipo de control inalámbrico es de aproximadamente 80 metros sin obstáculos.

En el lado izquierdo en cambio encontramos las opciones de control limitado, es decir que la trayectoria es controlada a un determinado tiempo. Este tiempo fue determinado en el respectivo programa.

En la parte inferior encontramos las opciones en las cuales podemos controlar la velocidad de la trayectoria del carrito

Figura 5.1. Pantalla de Control Remoto

Una vez que detallamos la pantalla del control de mando desde el computador a continuación realizamos las pruebas pertinentes para poder verificar que el carrito controlado inalámbricamente mediante el computador funcione como nos planteamos realizarlo, encontrando algunos detalles que no estaban funcionando correctamente que son los siguientes:

- Se tuvo que regular la velocidad del carrito es decir modificar el programa del microcontrolador del circuito receptor ya que la velocidad con la que se empezó a trabajar eran insuficiente como para movilizarse en la superficie.

- El giro hacia el lado derecha como izquierda estaba en sentido opuesto por lo que se pudo corregir en el programa realizado para el control de motores es decir el microcontrolador con que trabaja el circuito receptor.

- Se comprobó si el voltaje de alimentación del circuito emisor era suficiente del que recibe de la conexión del cable USB, lo cual se obtuvo como resultado positivo y que era suficiente para alimentar el modulo XBEE y los microcontroladores los cuales cabe mencionar tiene una alimentación de 3.3 V, pero si hubiese este problema dentro del circuito emisor se implemento un jamper de voltaje USB o voltaje externo.

- También se probo el tiempo de duración de la batería del carrito ya que se midió si presentaba alguna caída de voltaje en el momento de conectar el VCC del circuito con el que alimenta el modulo XBEE y el microcontrolador (3.3V) y el VCC de los motores del carrito, sin presentar ningún problema, pero si hubiese este problema dentro de la construcción del circuito receptor se implemento un jamper de voltaje externo de tal manera que si hubiese algún problema de caída de voltaje podríamos implementar un voltaje externo.

CAPITULO VI

CONCLUSIONES Y RECOMENDACIONES

6.1 CONCLUSIONES.

1. Los módulos inalámbricos XBEE presenta la confiabilidad necesaria para la comunicación inalámbrica la cual trabaja en una frecuencia de operación de 2.4GHz con un alcance aproximado de 80 metros.

2. El voltaje utilizado desde la conexión USB del circuito emisor fue suficiente para la alimentación de los microcontroladores y el modulo XBEE (3.3 V), por lo que también se puede concluir que no fue necesario la conexión externa de otra fuente de voltaje.

3. De igual manera para el circuito receptor y los motores del carrito no fue necesario la conexión de una fuente externa, ya que con la entrega de voltaje que se obtenía de la batería del carrito, fue suficiente para alimentar el microcontrolador y el modulo XBEE del circuito receptor y también para los motores del carrito a control remoto.

4. La rapidez de oscilación del cristal de cuarzo depende del tamaño del mismo, el cual entre más pequeño sea vibrará a una mayor frecuencia y tienen la ventaja de poder ser usados para frecuencias altas de hasta 1GHz.

5. Para poder tener una buena comunicación se utilizo los microcontroladores ATMEGA 164 y ATMEGA 48 debido a que estos presentaban buenas características como es, la cantidad de memoria flash, números de pines de entrada y salida, entre otras, lo que permitió una comunicación más ágil entre las tarjetas.

6. Con la opción **Write** (Escribir), de la pantalla principal del programa progisp se guarda los cambios y modificaciones de los fusibles a utilizarse en el momento de cargar los programas en los microcontroladores, ya que si no se utiliza esta opción al momento de cargar los programas, estos se cargaran con los fusibles que se tiene por default.

7. Con la implementación de los zócalos para **ICSP** en cada uno de los circuitos tanto emisor como receptor del sistema de control inalámbrico son muy importantes para el momento de cargar el programa de los microcontroladores con el hardware **USBAS,** ya que con esta implementación no es necesario que al momento de cargar el programa se deba retirar los microcontroladores de los zócalos de los circuitos realizados y así evitar que estos sufran daños y se quemen.

8. Con la implementación de los pines de medida en los circuitos emisor y receptor podemos tener un control del funcionamiento de los voltajes de los microcontroladores y así verificar en el momento que presente alguna falla.

9. Es muy importante el tipo de carrito a control remoto que se seleccione, ya que se debe tener en cuenta los sentidos de giro de los motores y que las ruedas sean aptas para cualquier terreno que se presente.

6.2. RECOMENDACIONES.

1. Se debe seleccionar todas las herramientas adecuadas para la elaboración de cualquier circuito electrónico en el que se utilice uno o varios microcontroladores como son emuladores, simuladores, compiladores, etc.

2. Antes de realizar las placas de los circuitos es decir el hardware del proyecto debemos realizar todas las pruebas de funcionamiento armando

los circuitos en el protoboard y de ahí verificar su confiabilidad y así realizar los circuitos impresos.

3. La implementación de los ya mencionados zócalos para cargar los programas (ICSP), en las placas de los circuitos ya que facilita cargar el programa a los microcontroladores evitando retirarlos a cada momento de sus zócalos respectivos y así tener más durabilidad.

4. Siempre en el software progisp de los AVRs se debe desactivar el fusible lock chip, ya que este es un seguro y al tenerlo activado solo se puede programar una vez. No se puede hacer cambios futuros en el microcontrolador, es decir el programa no puede ser manipulado por ninguna persona.

5. Se debe configurar correctamente los módulos XBEE según la función que se requiere, para lo cual se utiliza el programa **X-CTU,** adquirido cuando se compro los módulos y como hardware el **XBee explorer USB** para cargar las configuraciones**.**

6. La utilización del microcontrolador ATMEGA 164 tiene como gran ventaja en la realización del proyecto, ya que con el mencionado microcontrolador se puede realizar dos comunicaciones seriales por hardware, necesarios para el prototipo a realizarse ya que si utilizamos otro microcontrolador de menor capacidad tendríamos que utilizar dos de ellos y no se obtendría la optimización requerida y un circuito más compacto a pesar que quede sin utilizarse la mayoría de los pines del microcontrolador.

BIBLIOGRAFIA.

XBEE

http://www.digi.com/products/wireless-wired-embedded-solutions/zigbee-rf-modules/point-multipoint-rfmodules/xbee-series1-module.jsp#overview

XBEE-PRO

http://www.xbee.cl/index.html

MICROPROCESADORES

http://es.wikipedia.org/wiki/Microcontrolador

ATMEGA8

http://www.sc.ehu.es/sbweb/webcentro/automatica/web_avr/archivos/Otros%20AVRs/ATmega/ATmega8.htm

http://www.lulu.com/items/volume_38/588000/588200/1/print/SESION_1_ATMEGA8.pdf

ATMEGA 164

http://spanish.alibaba.com/product-gs/atmega48-20pu-290356744.html

http://www.datasheetcatalog.com/

CONTROL DE MOTORES DC

http://www.tecnologiaseso.es/pdf/electronicapdf/motores%20cc.pdf

http://es.wikipedia.org/wiki/Motor_de_corriente_continua

L293B

http://www.todopic.com.ar/utiles/l293b_driver_en_puente.pdf

http://www.datasheetcatalog.net/es/datasheets_pdf/L/2/9/3/L293B.shtml

LM1117T

http://pdf1.alldatasheet.es/datasheet-pdf/view/134371/ETC1/LM1117T.html

COMUNICACIONES SERIAL USB

http://es.wikipedia.org/wiki/Puerto_(inform%C3%A1tica)

http://es.kioskea.net/contents/pc/serie.php3

ANEXOS

ANEXO 1:

QUICK START GUIDE

XBEE/XBEE-PRO

Quick Start Guide

XBee™/XBee-PRO™ OEM Development Kits

Create long range wireless links in minutes!

Introduction

This Quick Start Guide provides step-by-step instruction on how to setup wireless links and test the modules' ability to transport data over varying ranges and conditions. This guide illustrates how to setup and run a point-to-point range test; then how to expand into a point-to-multipoint topology.

Range Test Setup

Required Components

(2) OEM RF Modules (any combination of XBee & XBee-PRO Modules)

(1) USB Interface Board* (for interfacing between an RF module & host PC)

(1) RS-232 Interface Board (for looping data back to the base from a remote)

(1) PC (Windows 2000 or XP) with an available USB (or RS-232*) port.
Required installations: X-CTU Software & USB drivers (Note: Drivers for LINUX and Mac OS X are provided on the CD, but the X-CTU Software will only run on Windows.)

Accessories (1 USB Cable, 1 Serial Loopback Adapter [RED] & 1 power supply)

** XBee Professional Developer Kits (XB24-PDK) contain four RS-232 boards. An RS-232 board (w/ RS-232 cable & power supply) can be used in lieu of the USB option.*

Software Installations

Install X-CTU Software

Double-click "setup_X-CTU.exe" file and follow prompts of the installation screens. This file is located on the MaxStream CD and under the 'Software' section of the following web page: www.maxstream.net/support/downloads.php

The X-CTU Software interface is divided into the four following tabs:

- PC Settings - Setup PC serial com ports to interface with the RF module
- Range Test - Test the range of wireless links under varying conditions
- Terminal - Read/Set RF module parameters and monitor data communications
- Modem Configuration - Read/Set RF module parameters

Software Installations (continued)

Install USB Drivers (Hardware USB Bus & Virtual Com Port drivers)

The following steps were recorded while using the Windows XP operating system.

1. Verify the MaxStream CD is inserted into the CD drive.
2. Connect the USB development board to a PC using USB cable.
   - After the module assembly is detected by the PC, a "Found New Hardware Wizard" dialog box should appear.
3. Select the 'No, not this time' option; then select the 'Next' button.
4. Select 'Install from a specific list or location (Advanced)' option; then select the 'Next' button.
5. a. Select the 'Search for the best driver in these locations' option.
   b. Check 'Search removable media (CD-ROM...)' box; then select 'Next'.
   - The "Windows Logo Testing" alert box appears.
6. Select the 'Continue Anyway' button.
7. Select the 'Finish' button.
8. Repeat steps 2 through 6 to install the next driver.
9. Reboot the PC if prompted to do so.

Hardware Setup

Setup Point-to-point Wireless Data Link

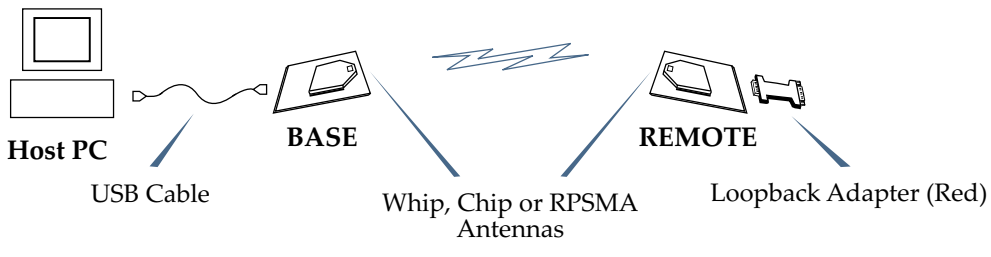
1. Mount XBee/XBee-PRO Modules to the USB & RS-232 development boards.
   - The module mounted to the USB board will be referred to as the "BASE".

   The module mounted to the RS-232 board will be referred to as the "REMOTE".
2. [Only if using modules that have the U.FL antenna connector]
   Connect the RF Cable Assembly to the U.FL antenna connector and RPSMA half-wave dipole antenna.
3. After installing the X-CTU Software and USB drivers, connect the BASE module assembly to the PC using a standard USB cable [Figure 1].
4. Attach the Serial Loopback Adapter [red] to the female DB-9 connector of the REMOTE module assembly.

   The Serial Loopback Adapter configures the REMOTE to function as a repeater by looping data back into the module for retransmission [Figure 1].
5. Power the REMOTE through the RS-232 development board's power connector using the power adapter included in the development kit.

   The BASE is powered through its USB connection.

Figure 1. Hardware Setup



Point-to-point Range Test

Use the "PC Settings" and "Range Test" tabs of the X-CTU Software to:

- Setup a PC Serial Com Port for communications with the BASE module assembly
- Determine the range capabilities of the XBee/XBee-PRO Modules

The out-of-box default configuration of the module is optimal for running this range test.

Run Range Test

1. Launch the X-CTU Software: *(Start --> Programs --> MaxStream --> X-CTU)*
2. Under the "PC Settings" tab [Figure 2], select the PC serial com port from the list that will be used to connect to the BASE module assembly. Before proceeding, verify the baud and data settings of the com port match those of the RF module.
3. Select the "Range Test" tab. [Figure 3]
4. (Optional) Check the "RSSI" checkbox to enable its display.
5. Click the 'Start' button to begin the range test.
6. Move the REMOTE (with red Serial Loopback Adapter) away from the BASE to find the maximum range of the wireless link.

Figure 2. PC Settings tab

② **PC Com Port**

NOTE: Failure to enter AT Command Mode is most commonly due to baud rate mismatch. Ensure the 'Baud' setting on the "PC Settings" tab matches the interface data rate of the module [BD (Interface Data Rate) parameter = 9600 bps by default].

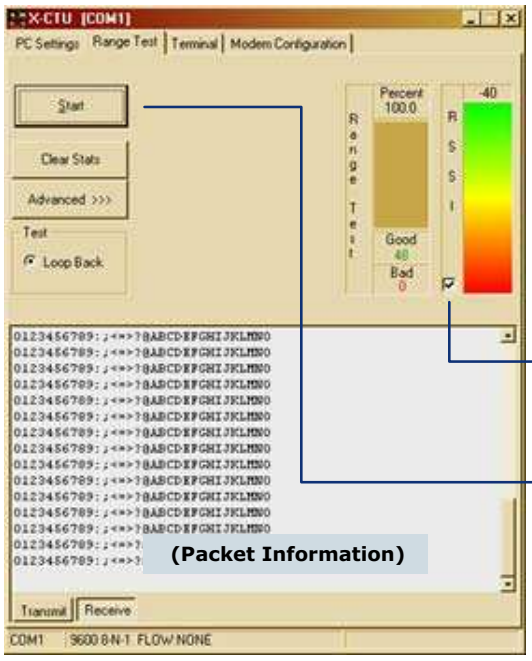


Figure 3. Range Test tab

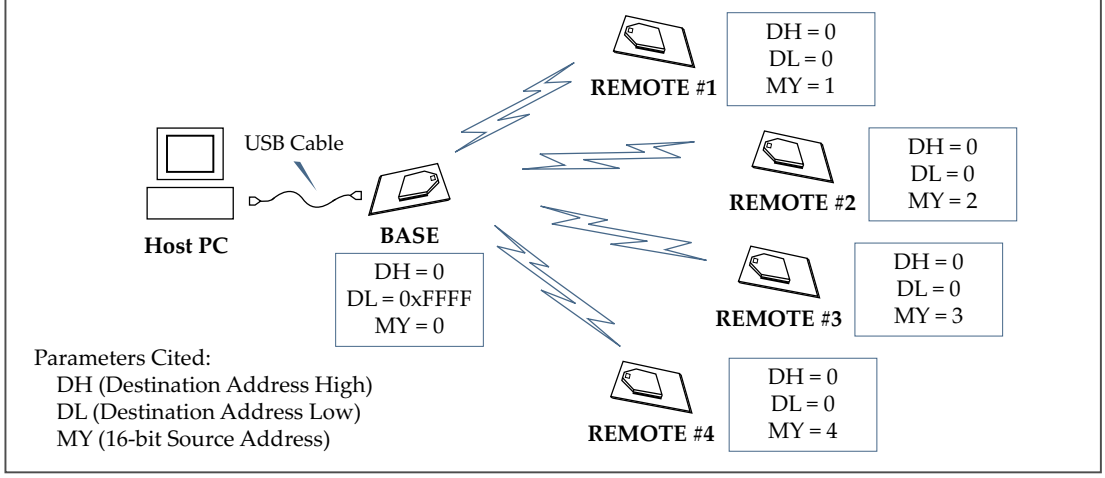
④ **RSSI** checkbox
RSSI stands for "Received Signal Strength Indicator"

⑤ **Start/(Stop)** button

Point-to-multipoint Networks

Point-to-multipoint topologies require that one BASE module be configured to operate in Broadcast Mode [see 'Addressing Considerations' section]. REMOTE modules can operate either in Broadcast or Unicast Mode. The figure below depicts a typical point-to-multipoint network that contains one BASE (Broadcast Mode) and four REMOTES (Unicast Mode).

Figure 4. Sample Point-to-multipoint Network Configuration



RF Module Configuration

To configure RF module parameters:

- Install X-CTU Software (& USB drivers if connecting to the host via a USB port)
- Setup a USB or RS-232 connection between the module and host PC [page 2]
- On the X-CTU "PC Settings" tab [Figure 2]: Verify com port and module settings match; then select the PC com port that will be used to connect to the BASE
- On the X-CTU "Modem Configuration" tab: Select the 'Read' button, modify parameters, then select the 'Write' button.

This is one of several configuration methods. Refer to the manual for more information.

Addressing Considerations

Unicast Mode

By default, XBee/XBee-PRO Modules are configured to operate in Unicast Mode - retries are enabled and receiving modules send an ACK (acknowledgement) of RF packet reception to the transmitter. If the transmitting module does not receive an ACK, it re-sends the RF data packet up to three times or until the ACK is received.

Default addressing parameters:

MY (16-bit Source Address) = 0
DL (Destination Address Low) = 0
DH (Destination Address High) = 0

Broadcast Mode

For one RF module to communicate to many modules, one module (BASE) must be configured to operate in Broadcast Mode. When in Broadcast Mode, retries and acknowledgements are disabled. Broadcast Mode is enabled by setting Destination Addresses as follows:

DL (Destination Address Low) = 0x00000000
DH (Destination Address High) = 0x0000FFFF

Refer to the product manual for more information regarding addressing.

Contact MaxStream (Office hours are 8am – 5pm U.S. Mountain standard time)

Phone: (801) 765-9885, Live Chat: www.maxstream.net, E-mail: rf-xperts@maxstream.net

ANEXO 2:

X-CTU CONFIGURATION & TEST UTILITY SOFTWARE

X-CTU
Configuration & Test Utility Software

User's Guide

Contents

Technical Support:

Online support: http://www.digi.com/support/eservice/login.jsp

Phone: (801) 765-9885

90001003_A

2008.08.20

Introduction

This User's Guide is intended to discuss the functions of Digi's X-CTU software utility. Each function will be discussed in detail allowing a better understanding of the program and how it can be used.

X-CTU is a Windows-based application provided by Digi. This program was designed to interact with the firmware files found on Digi's RF products and to provide a simple-to-use graphical user interface to them.

X-CTU is designed to function with all Windows-based computers running Microsoft Windows 98 SE and above. X-CTU can either be downloaded from Digi's Web site or an installation CD. When properly installed it can be launched by clicking on the icon on the PC desktop (see Figure 1) or selecting from the Start menu (see Figure 2).

Figure 1

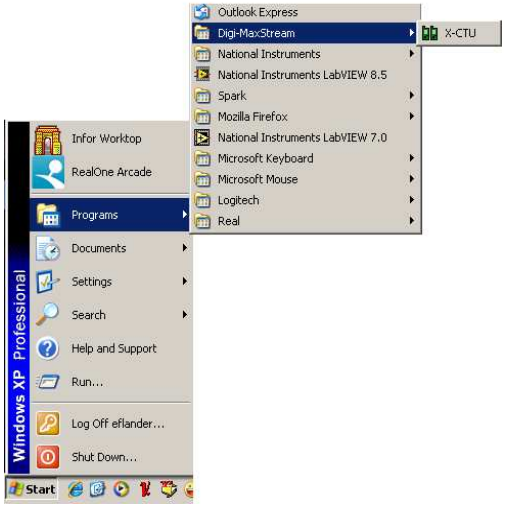


Figure 2

When launched, you will see four tabs across the top of the program (see Figure 3). Each of these tabs has a different function. The four tabs are:

PC Settings: Allows a customer to select the desired COM port and configure that port to fit the radios settings.

Range Test: Allows a customer to perform a range test between two radios.

Terminal: Allows access to the computers COM port with a terminal emulation program. This tab also allows the ability to access the radios' firmware using AT commands (for a complete listing of the radios' AT commands, please see the product manuals available online).

Modem Configuration: Allows the ability to program the radios' firmware settings via a graphical user interface. This tab also allows customers the ability to change firmware versions.

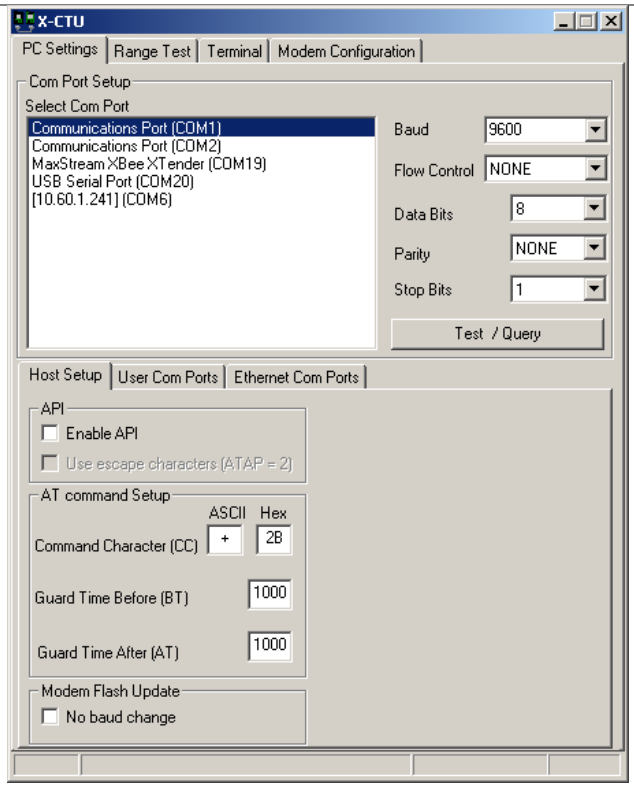


Figure 3

PC Settings Tab

When the program is launched, the default tab selected is the "PC Settings" tab. The PC Settings tab is broken down into three basic areas: The COM port setup, the Host Setup, and the User Com ports.

COM port setup:

The PC settings tab allows the user to select a COM port and configure the selected COM port settings when accessing the port. Some of these settings include:

Baud Rate:	Both standard and non-standard
Flow Control:	Hardware, Software (Xon/Xoff), None
Data bits:	4, 5, 6, 7, and 8 data bits
Parity:	None, Odd, Even, Mark and Space
Stop bit:	1, 1.5, and 2

To change any of the above settings, select the pull down menu on the left of the value and select the desired setting. To enter a non-standard baud rate, type the baud rate into the baud rate box to the left.

The **Test / Query** button is used to test the selected COM port and PC settings. If the settings and COM port are correct, you will receive a response similar to the one depicted in Figure 4 below.

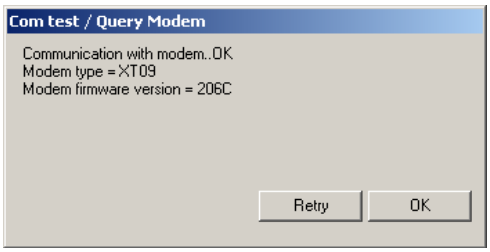


Figure 4

Host Setup:

The Host Setup tab allows the user to configure how the X-CTU program is to interface with a radio's firmware. This includes determining whether API or AT command mode will be used to access the module's firmware as well as the proper command mode character and sequence.

By default, the Host Settings are as follows:

API mode:	not enabled (Not checked)
Command mode Character:	+ (ACSII) 2B (Hex).
Before Guard Time:	1000 (1 Sec)
After Guard Time:	1000 (1 Sec)

This is the default value of our radios. If this is not the value of the AT, BT, or GT commands of the connected radio, enter the respective value here.

User COM ports:

The user COM port option allows the user to "Add" or "Delete" a user-created COM port. This is only for temporary use. Once the program has closed, the user-created COM port will disappear and is no longer accessible to the program.

Range Test Tab

The range test tab is designed to verify the range of the radio link by sending a user-specified data packet and verifying the response packet is the same, within the time specified. For performing a standard range test, please follow the steps found in most Quick Start or Getting Started Guides that ship with the product.

Packet Data and Size

By default, the size of the data packet sent is 32 bytes. This data packet specified can be adjusted in either size or the text sent.

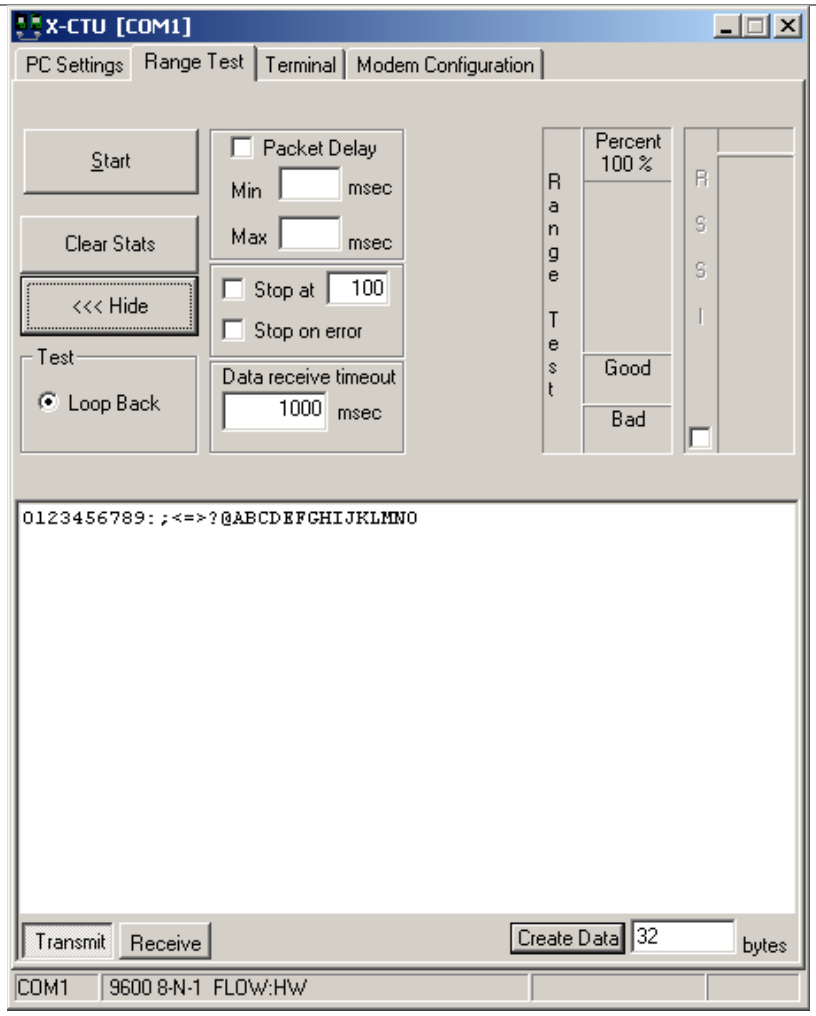


Figure 5

To modify the size of the packet sent, change the value next to the “Create Data” box and click on the “Create Data” button (see Figure 5). If you want to change the data sent, delete the text in the transmit window and place in your desired text.

By modifying the text, data packet size, packet delay and the data receive timeout; the user is able to simulate a wide range of scenarios.

RSSI:

The RSSI option of the X-CTU allows the user to see the RSSI (Received Signal Strength Indicator) of a received packet when performing a range test.

API Function:

The X-CTU also allows the user to test the API function of a radio during a range test.

To perform a range test with the API function of the radio, follow the steps outlined below:

1: Configure the Base with API enabled and a unique 16 bit or 64 bit source address.
2: Configure the remote radio with a unique source address and set the Destination address to equal the Base radio's source address.
3: Enable the API option of the X-CTU on the PC Settings tab and connect the base radio to the PC (See Figure 3).
4: Connect the red loopback adapter to the remote radio and place them a distance apart.
5: Enter either the 16 bit or 64 bit destination address of the remote radio into the Destination Address box on the Range Test tab (See figure 6).
6: Create a data packet of your choosing by typing in the data in the Transmit box
7: To start a Range test, click on Start.

You will notice the TX failures, Purge, CCA, and ACK messages will increment accordingly while the range test is performed.

To stop a range test, click on the Stop button.

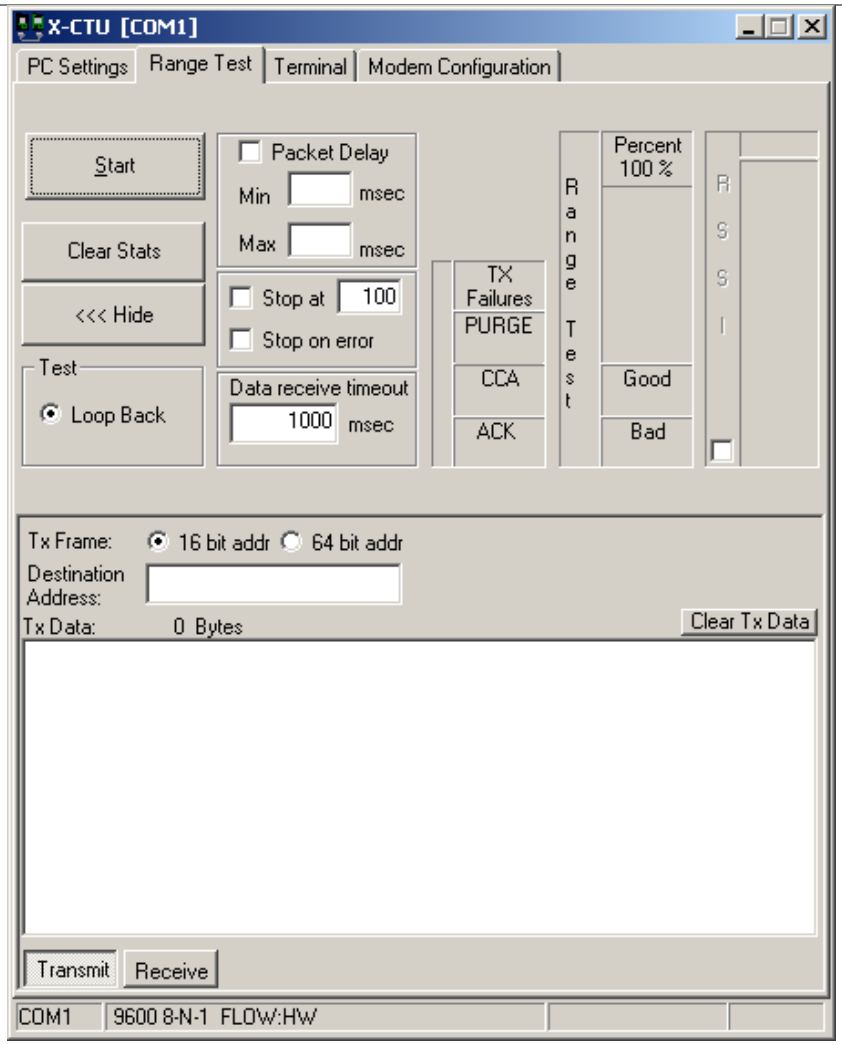


Figure 6

The Terminal Tab

The Terminal tab has three basic functions:

Terminal emulator

Ability to send and receive predefined data pacts (Assemble packet)

Ability to send and receive data in Hex and ASCII formats (Show/Hide hex)

The main terminal window

The main white portion of this tab is where most of the communications information will occur while using X-CTU as a terminal emulator. The text in blue is what has been typed in and directed out to the radio's serial port while the red text is the incoming data from the radio's serial port (see Figure 7).

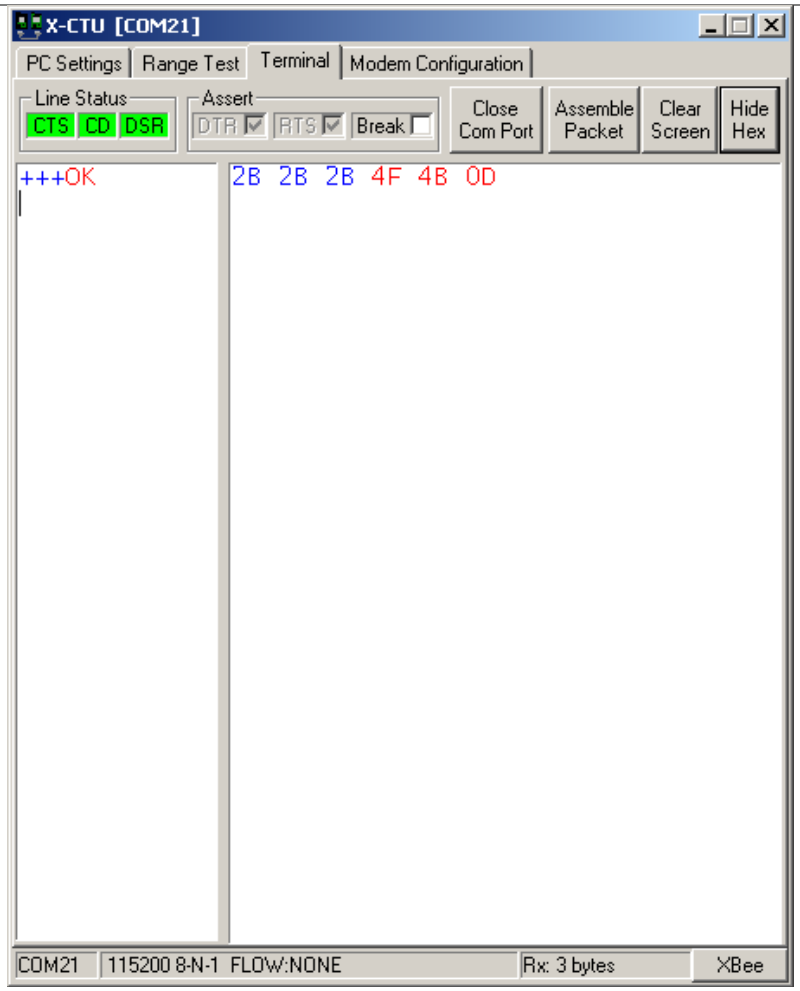


Figure 7

Assemble Packet

The Assemble Packet option on the Terminal tab is designed to allow the user to assemble a data packet in either ASCII or Hex characters. This is accomplished by selecting the Assemble packet window and choosing either ASCII (default) or Hex. Once selected, the data packet is assembled by typing in the desired characters as depicted in Figure 8.

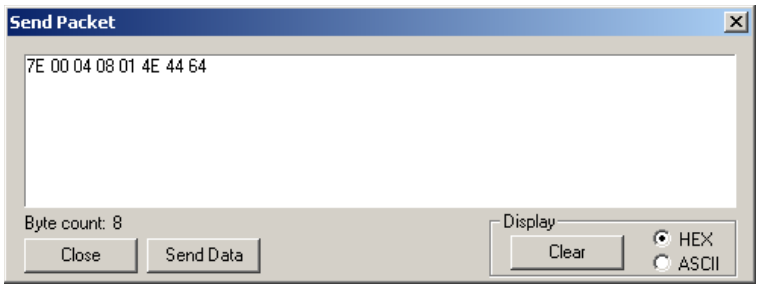


Figure 8

The **Line Status** indicators depicted in Figure 5 shows the status of the RS-232 hardware flow control lines. Green indicates the line is asserted while black indicates de-asserted.

The **Break** option is for engaging the serial line break. This can be accomplished by checking or asserting the Break option. Asserting the Break will place the DI line high and prevent data from being sent to the radio.

Modem Configuration tab

The Modem configuration tab has four basic functions:

1: Provide a Graphical User Interface with a radio's firmware
2: Read and Write firmware to the radio's microcontroller
3: Download updated firmware files from either the web or from a compressed file
4: Saving or loading a modem profile

Reading a radio's firmware

To read a radio's firmware, follow the steps outlined below:

1: Connect the radio module to the interface board and connect this assembly or a packaged radio (PKG) to the PC's corresponding port (IE: USB, RS232, Ethernet etc.).
2: Set the PC Settings tab (see Figure 3) to the radio's default settings.
3: On the Modem Configuration tab, select "Read" from the Modem Parameters and Firmware section (see Figure 9).

Making changes to a radio's firmware

Once the radio's firmware has been read, the configuration settings are displayed in three colors (see Figure 10):

Black – not settable or read-only
Green – Default value
Blue – User-specified

To modify any of the user-settable parameters, click on the associated command and type in the new value for that parameter. For ease of understanding a specific command, once the command is selected, a quick description along with its limits is provided at the bottom of the screen. Once all of the new values have been entered, the new values are ready to be saved to the radio's non-volatile memory.

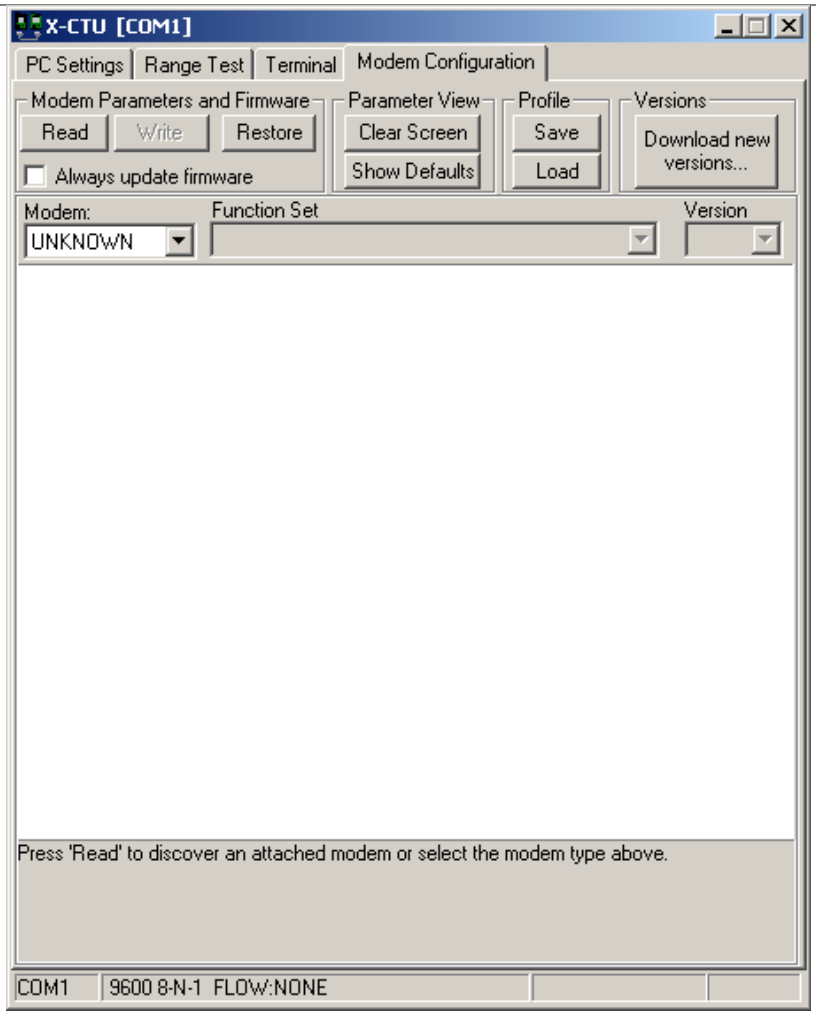


Figure 9

Writing firmware to the Radio

To write the parameter changes to the radio's non-volatile memory, click on the Write button located in the Modem Parameters and Firmware section (see Figure 10)

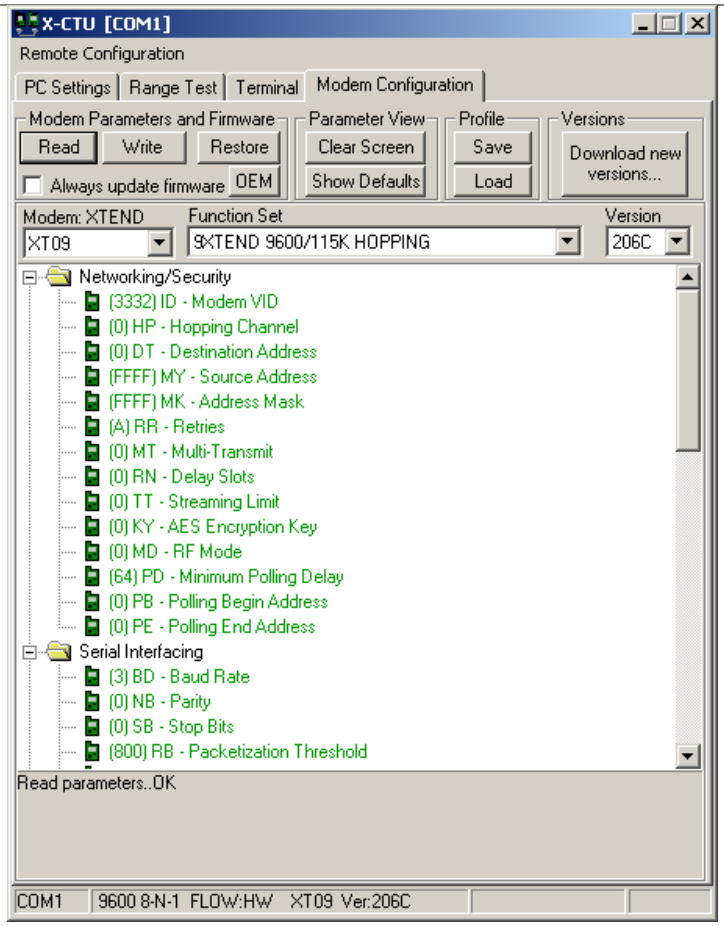


Figure 10

Downloading Updated Firmware Files

Another function of the Modem Configuration tab is allowing the user to download updated firmware files from either the web or install them from a disk or CD. This is accomplished by following the steps below:

1: Click on the Download New Versions... option under the Version section
2a: Click on Web for downloading new firmware files from the web
2b: Click on the File when installing compressed firmware files from a CD or saved file (see Figures 11 and 12)
    2bi: Browse to the location the file is saved at and click on Open (see Figure 13)
3: Click on OK and Done when prompted

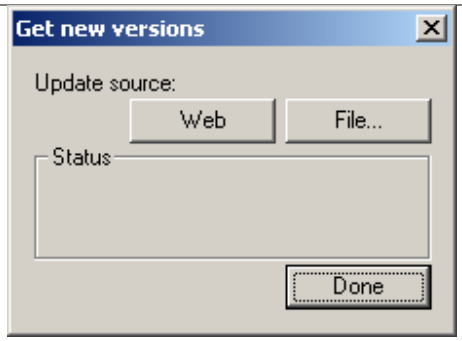


Figure 11

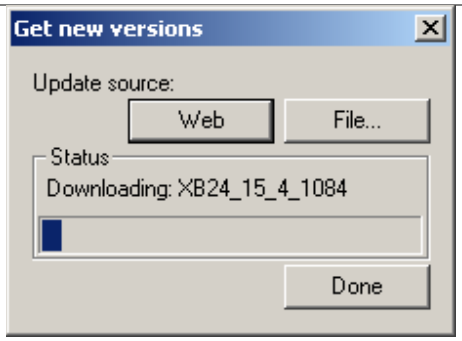


Figure 12

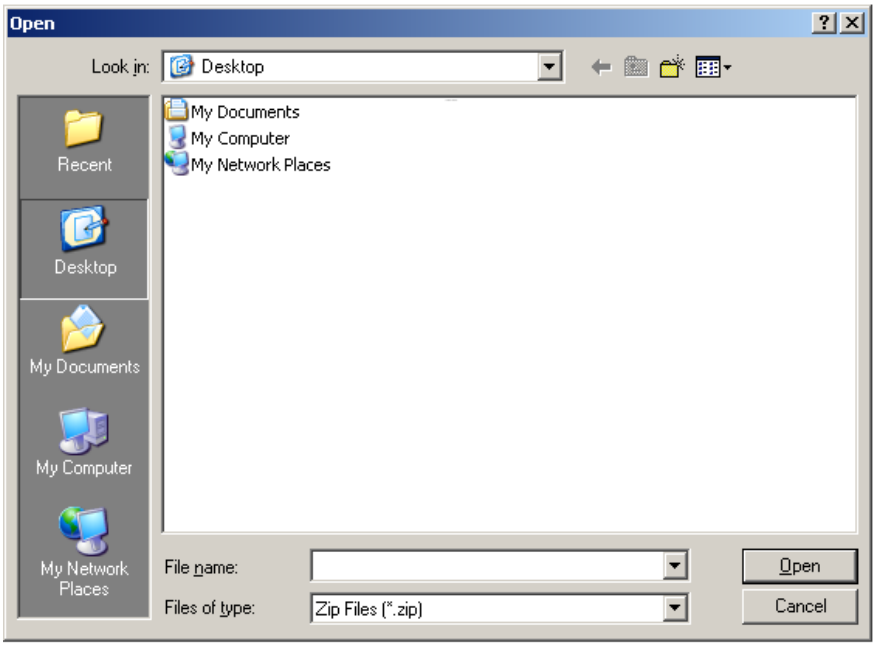


Figure 13

Modem Profiles

The X-CTU has the ability to save and write saved modem profiles or configuration to the radio. This function is useful in a production environment when the same parameters need to be set on multiple radios.

How to save a profile:

1: Set the desired settings within the radio's firmware as described in the Making changes to the radios firmware section
2: Click Save in the Profile section
3: Type in the desired name of this profile in the File Name box (see Figure 14)
4: Browse to the location where you wish to save your profile
5: Click Save

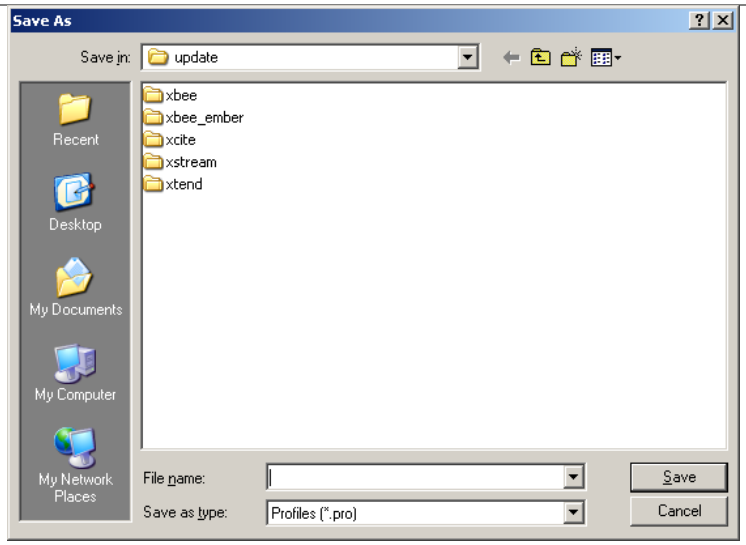


Figure 14

How to load a saved profile:

1: Click on Load from the profile section
2: Browse to the location of the file and click on the desired file (see Figure 15)
3: Click Open

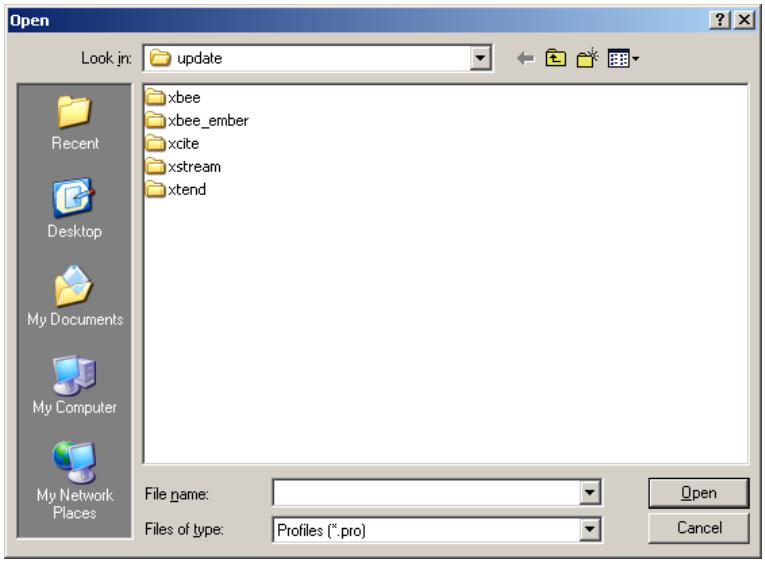


Figure 15

To save the loaded profile to the radio once you have loaded the file, follow the steps outlined in the Writing firmware to the radio section above.

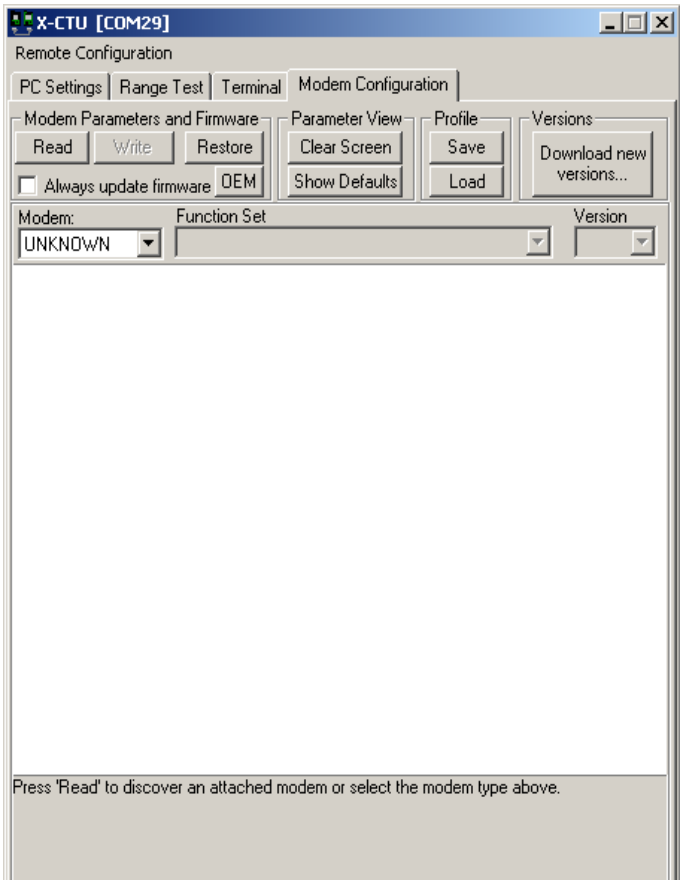
To find out how to load the saved profiles in a production environment from a DOS prompt, please follow the steps outlined in Digi's online Knowledgebase at http://www.maxstream.net/support/knowledgebase/article.php?kb=126

XBee 802.15.4 modules with firmware version 1xCx and above, XBee ZNet 2.5 modules, and XBee ZB modules offer the ability to be configured with over the air commands. With the addition of this new feature, the user is able to configure remote radio parameters with X-CTU or API packets. To use the remote configuration tool, the following is required:

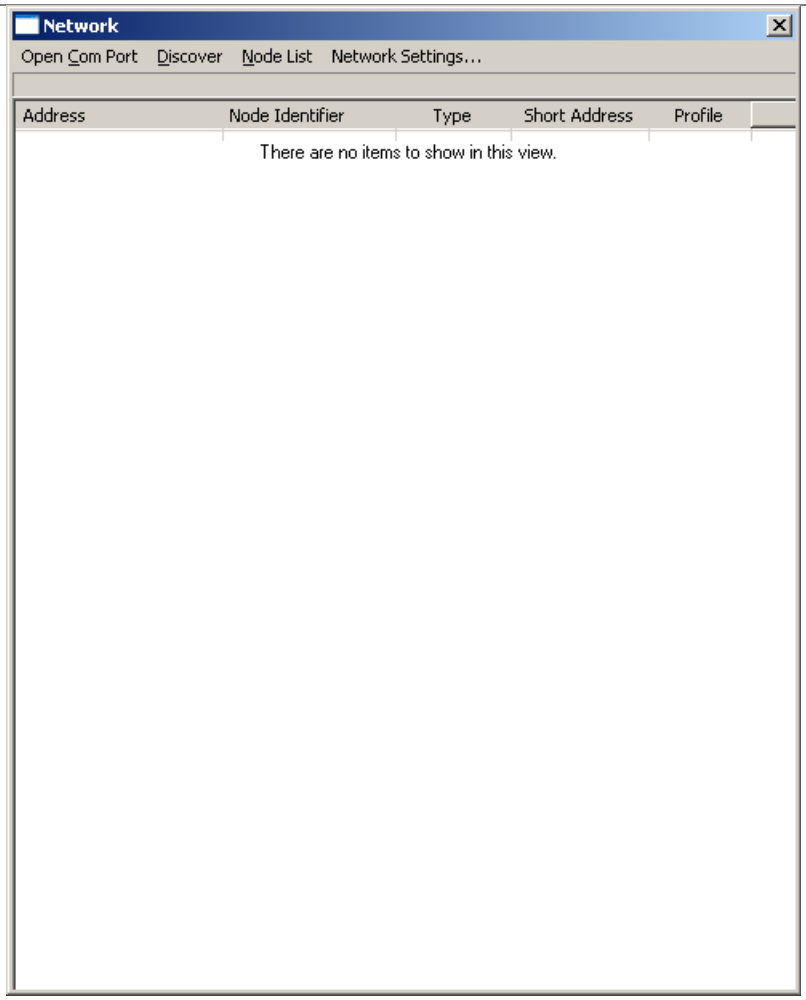
- The radio connected to the PC must be in API mode
- The remote radio must be associated or within range of the base radio

To access remote radios through X-CTU's Modem Configuration tab, perform the steps below:

- Enable API on the PC Settings tab
- Verify the COM port selection and settings
- On the Modem Configuration tab, select the Remote Configuration option on the top left corner of the program



- Select Open Com port
- Select Discover



- Select the desired modem from the discovered node list
- On the Modem configuration tab, select Read

The remote radio's configuration is now displayed on the Modem Configuration tab. At this point, the same options exist with respect to Read and Write parameter changes. Please note that the ability to change firmware versions is still limited to the radio's UART.

To clear the discovered node list, click on Node List and Clear.

The Node List option provides several additional options, including:

- Ability to print the discovered list
- Ability to remove a specific node from a list
- Ability to add additional nodes that have not been discovered
- Save the Node List
- Load a saved Node List
- Select/filter All, Routers, or End nodes

For specific questions related to the X-CTU configuration and test utility software, please contact our Support department, Mon – Fri, 8am – 5pm U.S. Mountain Time:

US and Canada Toll free:
(866)765-9885

Local or International calls:
(801) 765-9885

Online support: http://www.digi.com/support/eservice/login.jsp

ANEXO 3:

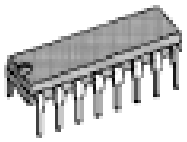
DESCRIPCIÓN DEL DRIVER DE MOTORES L293B

SGS-THOMSON MICROELECTRONICS

L293B

DRIVER PUSH-PULL DE 4 CANALES

- Corriente de salida de 1 A por canal.
- Corriente de salida de pico de 2 A por canal
- Señal para la habilitación de las salidas
- Alta inmunidad al ruido
- Alimentación para las cargas separada de la alimentación de control.
- Protección contra sobre - temperaturas.



DIP16

ORDERING NUMBER : L293B

DESCRIPCIÓN

El L293B es un driver de 4 canales capaz de proporcionar una corriente de salida de hasta 1A por canal. Cada canal es controlado por señales de entrada compatibles TTL y cada pareja de canales dispone de una señal de habilitación que desconecta las salidas de los mismos.

Dispone de una patilla para la alimentación de las cargas que se estén controlando, de forma que dicha alimentación es independiente de la lógica de control.

La Figura 2 muestra el encapsulado de 16 pines, la distribución de patillas y la descripción de las mismas.

Pin	Nombre	Descripción
1	Chip Enable 1	Habilitación de los canales 1 y 2
2	Input 1	Entrada del Canal 1
3	Output 1	Salida del Canal 1
4	GND	Tierra de Alimentación
5	GND	Tierra de Alimentación
6	Output 2	Salida del Canal 2
7	Input 2	Entrada del Canal 1
8	V_S	Alimentación de las cargas
9	Chip Enable 2	Habilitación de los canales 3 y 4
10	Input 3	Entrada del Canal 3
11	Output 3	Salida del Canal 3
12	GND	Tierra de Alimentación
13	GND	Tierra de Alimentación
14	Output 4	Salida del Canal 4
15	Input 4	Entrada del Canal 4

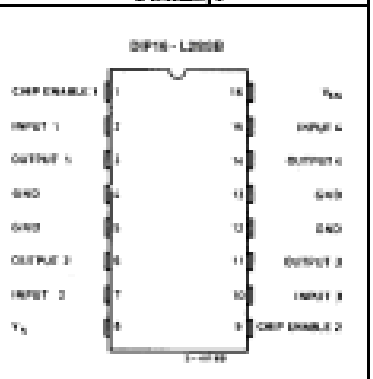


Figura 2 - Descripción de los Pines del L293B

DIAGRAMA DE BLOQUES

En la Figura 3, se muestra el diagrama de bloques del L293B. La señal de control **EN1** activa la pareja de canales formada por los drivers 1 y 2. La señal **EN2** activa la pareja de drivers 3 y 4. Las salidas **OUTN** se asocian con las correspondientes **OUTn**. Las señales de salida son amplificadas respecto a las de entrada tanto en tensión (hasta +Vss) como en corriente (máx. 1 A).

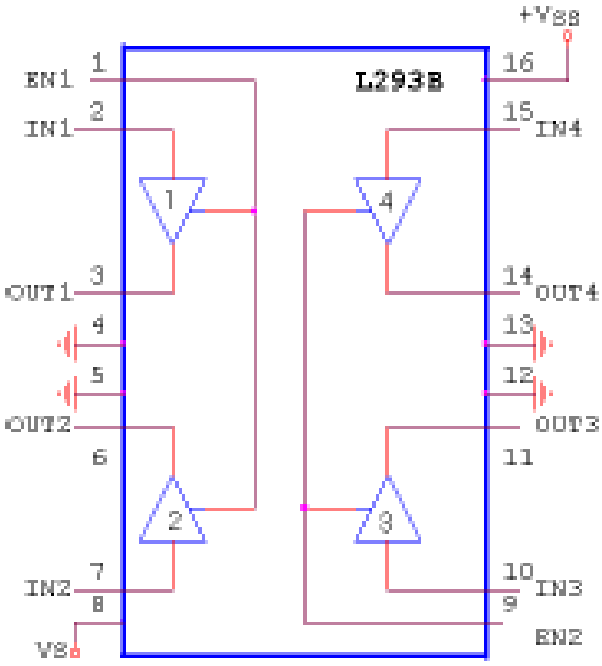


Figura 3.- Diagrama de bloques del L293B

La tabla de funcionamiento para cada uno de los driver es la siguiente:

V_{INn}	V_{OUTn}	V_{ENn}
H	H	H
L	L	H
H	Z	L
L	Z	L

Donde:

H = Nivel alto "1"
L = Nivel bajo "0"
Z = Alta Impedancia

PARAMETROS

RANGOS ABSOLUTOS MAXIMOS

Símbolo	Parámetro	Valor	Unidades
V_s	Tensión de alimentación para las cargas	36	V
V_{ss}	Tensión de alimentación de la lógica	36	V
V_i	Tensión de entrada	7	V
V_{inh}	Tensión de habilitación	7	V
I_{out}	Intensidad de pico de salida	2	A
P_{tot}	Potencia total de disipación	5	W
T_{stg}, T_j	Temperatura de almacenaje y de la unión	-40 a +150	ºC

CARACTERISTICAS ELECTRICAS
Para cada canal, V_s = 24V, V_{ss} = 5V, T_{amb} =25ºC

Símbolo	Parámetro	Condiciones de Test	Min	Típica	Max	Unidades
V_S	Tensión de alimentación de las cargas		V_{SS}		36	V
V_{SS}	Tensión de alimentación de la lógica		4.5		36	V
I_S	Corriente total de reposo	V_i = L I_O=0 V_{inh} =H V_i = H I_O=0 V_{inh} =H V_{inh} =L		2 16	6 24 4	mA
I_{SS}	Corriente total de reposo con señal de control	V_i = L I_O=0 V_{inh} =H V_i = H I_O=0 V_{inh} =H V_{inh} =L		44 16 16	60 22 24	mA
V_{IL}	Tensión de entrada a nivel bajo		-0.3		1.5	V
V_{IH}	Tensión de entrada a nivel alto	$V_{SS} \leq 7V$ $V_{SS} > 7V$	2.3 2.3		V_{SS} 7	V
I_{IL}	Corriente de entrada a nivel bajo	V_{IL} =1.5V			-10	µA
I_{IH}	Corriente de entrada a nivel alto	$2.3V \leq V_{IH} \leq V_{SS} - 0.6$		30	100	µA
V_{inhL}	Tensión de habilitación a nivel bajo		-0,3		1.5	V
V_{inhH}	Tensión de habilitación a nivel alto	$V_{SS} \leq 7V$ $V_{SS} > 7V$	2.3 2.3		V_{SS} 7	V
I_{inhL}	Corriente de habilitación a nivel bajo	V_{inhL} = 1.5V		-30V	-100	µA
I_{inhH}	Corriente de habilitación a nivel alto	$2.3V \leq V_{IH} \leq V_{SS} - 0.6$			±10	µA
V_{CEsatH}	Tensión de salida con la fuente saturada	I_O = 1 A		1.4	1.8	V
V_{CEsatL}	Tensión de salida con el sumidero saturado	I_O = 1 A		1.2	1.8	V

APLICACIONES.
En este apartado se muestran distintas configuraciones de conexión de motores al L293B

1. **GIRO DE 2 MOTORES EN UNICO SENTIDO.**
   En la Figura 4 se muestra el modo de funcionamiento de dos motores de corriente continua que giran en un único sentido.
   - El motor M1 se activa al poner a nivel bajo la entrada de control A.
   - El motor M2 se activa al poner a nivel alto la entrada de control B

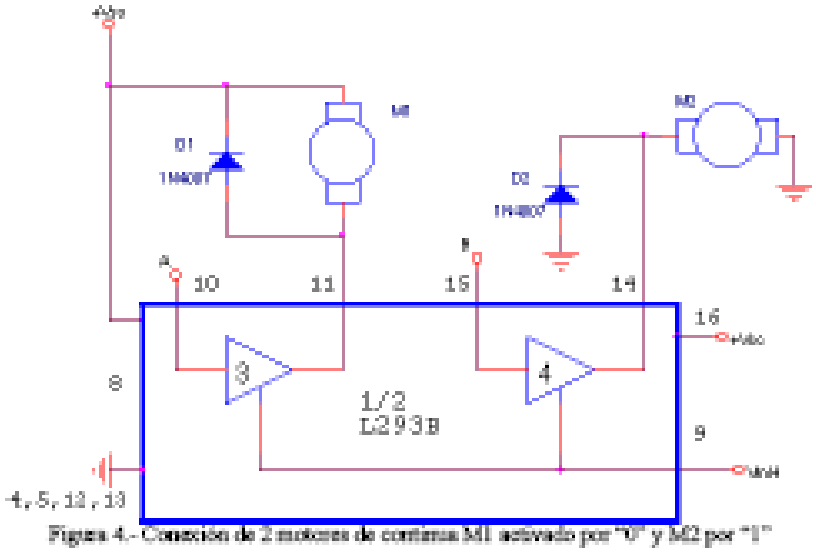


Figura 4.- Conexión de 2 motores de corriente M1 activado por "0" y M2 por "1"

Su tabla de funcionamiento es la siguiente:

V_{ss}	A	M1	B	M2
H	H	Parada rápida del motor	H	Giro
H	L	Giro	L	Parada rápida del motor
L	X	Motor desconectado, giro libre	X	Motor desconectado, giro libre

Tabla de verdad del circuito de la Figura 4

Los diodos D_1 y D_2, están conectados para proteger el circuito cuando se generan los picos de arranque de los motores. Si no se trabaja a máxima potencia de trabajo pueden eliminarse del circuito.

2. CONTROL DEL GIRO DE UN MOTOR EN LOS DOS SENTIDOS

El circuito de la Figura 5 permite controlar el doble sentido de giro del motor. Cuando la entrada C esta a nivel bajo y la D a nivel alto, el motor gira hacia la izquierda. Cambiando la entrada C a nivel alto y la D a nivel Bajo, se cambia el sentido de giro del motor hacia la derecha.

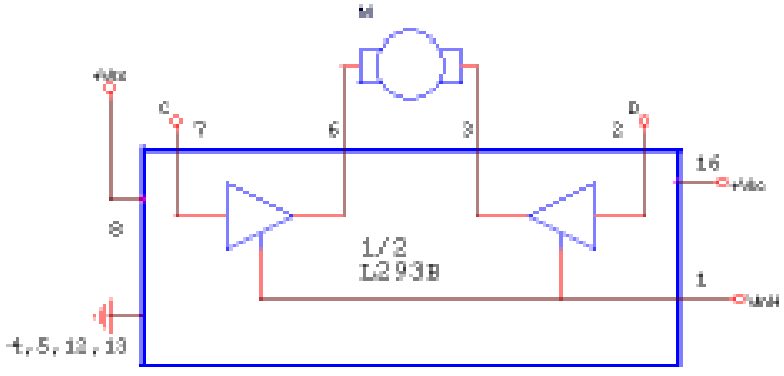


Figura 5.- Circuito de control para el doble giro de un motor de corriente continua

Si se quiere proteger el circuito contra posibles picos de corriente inversa cuando se arranca el motor, se recomienda conectar unos diodos tal y como se muestra en la Figura 6.

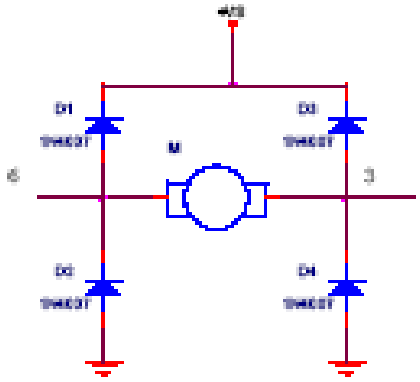


Figura 6.- Circuito de protección para el L293 para evitar sobre corrientes inversas al arrancar el motor

En este caso la tabla de funcionamiento es la siguiente:

V_{ss}	A	B	M
H	L	L	Parada rápida del motor
H	H	H	Parada rápida del motor
H	L	H	Giro a la Izquierda
H	H	L	Giro a la derecha
L	X	X	Motor desconectado, giro libre

3. CONTROL DE UN MOTOR PASO A PASO BIPOLAR

En la Figura 7 se muestra una forma de conectar un motor bipolar paso a paso. En este caso habrá que generar la secuencia adecuada al motor paso a paso para poder excitar de forma correcta sus bobinas.

La forma de proteger el circuito contra las corrientes que se producen en el momento de arranque del motor sería el mismo que el de la Figura 6 pero para cada una de las bobinas del motor paso a paso, es decir, utilizando 8 diodos.

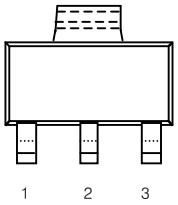
ANEXO 4:

DESCRIPCIÓN DEL REGULADOR LM1117

FEATURES

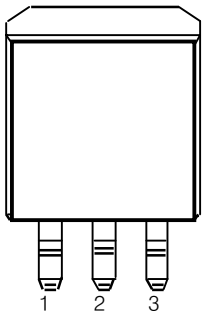
- Output Current up to 1 A
- Low Dropout Voltage (700mV at 1A Output Current)
- Three Terminal Adjustable or Fixed 1.5V, 1.8V, 2.5V, 2.85V, 3.0V, 3.3V, 5.0V
- 2.85V Device for SCSI-II Active Terminator
- 0.04% Line Regulaion, 0.1% Load Regulation
- Very Low Quiescent Current
- Internal Current andTerminal Limit
- Logic-Controlled Electronics Shutdown
- Surface Mount Package SOT-223 & TO-263 (D2-Pack)
- 100% Thermal Limit Burn-In

SOT-223 PKG (FRONT VIEW)



PIN FUNCTION
1. Adj/Gnd
2. Vout
3. Vin

TO-263 (D2 PKG, FRONT VIEW)



PIN FUNCTION
1. Adj/Gnd
2. Vout
3. Vin

APPLICATION

- Active SCSI Terminators
- Portable/Plan Top/Notebook Computers
- High Efficiency Linear Regulators
- SMPS Post Regulators
- Mother B/D Clock Supplies
- Disk Drives
- Battery Chargers

ORDERING INFORMATION

Device (Marking)	Package
LM1117S	SOT-223
LM1117S-XX	
LM1117T	TO-263 (D2)
LM1117T-XX	

(X=Output Voltage=1.5V, 1.8V, 2.5V, 2.85V, 3.0V, 3.3V, 5.0V, Adjustable=AD)

DESCRIPTION

The LM1117 is a low power positive-voltage regulator designed to meet 1A output current and comply with SCSI-II specifications with a fixed output voltage of 2.85V. This device is an excellent choice for use in battery-powered applications, as active terminators for the SCSI bus, and portable computers.
The LM1117 features very low quiescent current and very low dropout voltage of 700mV at a full load and lower as output current decreases. LM1117 is available as an adjustable or fixed 1.5V, 1.8V, 2.5V, 2.85V, 3.0V, 3.3V, and 5.0V output voltages.
The LM1117 is offered in a 3-pin surface mount package SOT-223 & TO-263. The output capacitor of 10μF or larger is needed for output stability of LM1117 as required by most of the other regulator circuits.

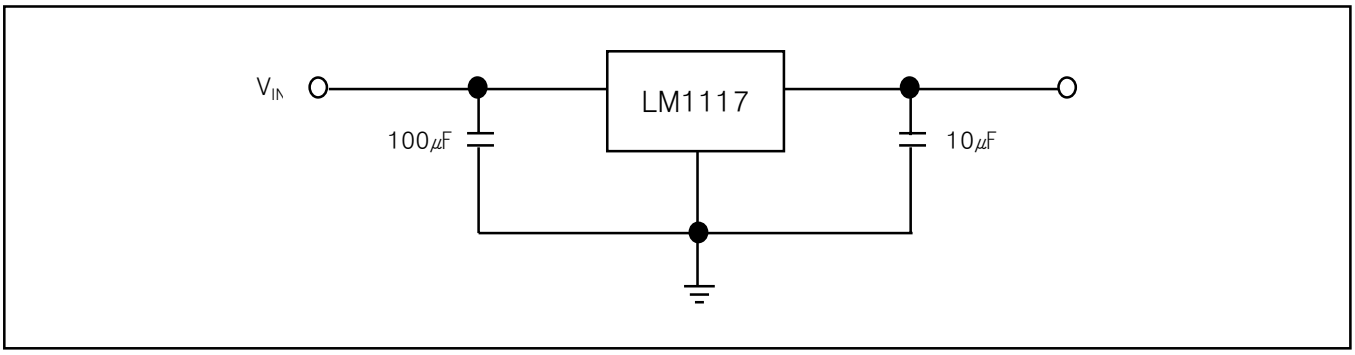
ABSOLUTE MAXIMUM RATINGS

CHARACTERISTIC	SYMBOL	MIN.	MAX.	UNIT
DC Input Voltage	V_{IN}		7	V
Lead Temperature (Soldering, 5 Seconds)	T_{SOL}		260	℃
Storage Temperature Range	T_{STG}	-65	150	℃
Operating Junction Temperature Range	T_{OPR}	0	125	℃

THERMAL DATA

PARAMETER	SYMBOL	SOT-223	TO-263	UNIT
Thermal Resistance Junction-Case	$R_{THJ-CASE}$	15	3	℃

TYPICAL APPLICATION



ELECTRICAL CHARACTERISTICS FOR LM1117 S/T-AD(ADJUSTABLE)

(Refer to the test circuits, T_J=0 to 125℃ C_O=10㎌ unless otherwise specified)

PARAMETER	SYMBOL	TEST CONDITION	MIN.	TYP.	MAX.	UNIT
Reference Voltage	V_{REF}	$V_{IN}-V_O$ = 2V, I_O = 100㎃, T_J = 25℃	1.238	1.25	1.262	V
Reference Voltage	V_{REF}	I_O = 10 to 1A, $V_{IN}-V_O$ = 1.4 to 10V	1.230		1.270	V
Line Regulation	ΔV_O	$V_{IN}-V_O$ = 1.5 to 13.75V, I_O = 10㎃		0.035	0.2	%
Load Regulation	ΔV_O	$V_{IN}-V_O$ = 3V, I_O = 10mA to 1A		0.1	0.4	%
Temperature Stability	ΔV_O			0.5		%
Long Term Stability	ΔV_O	1000 hrs, T_J = 125℃		0.3		%
Operating Input Voltage	V_{IN}				7	V
Adjustment Pin Current	I_{ADJ}	$V_{IN} \leq$ 15V, I_{Load} =10㎃		50	120	㎂
Adjustment Pin Current Change	ΔI_{ADJ}	$V_{IN}-V_O$ = 1.4 to 10V, I_O = 10mA to 1A		1	5	㎂
Minimum Load Current	$I_{O(MIN)}$	V_{IN} = 15V		1.7	5	㎃
Output Current	I_O	$V_{IN}-V_O$ = 5V, T_J=25℃	800	950	1200	㎃
Output Noise (%V_O)	$_E$N	B = 10㎐ to 10㎑, T_J = 25℃		0.003		%
Supply Voltage Rejection	SVR	I_O = 40㎃ , f = 120㎐, T_J = 25℃ $V_{IN}-V_O$ = 3V, V_{NIPPLE} = $1V_{PP}$	60	75		dB
Dropout Voltage	V_D	I_O = 100㎃ , V_{IN}=V_{OUT}+ 0.8V		1	1.1	V
		I_O = 500㎃ , V_{IN}=V_{OUT}+ 0.8V		1.00	1.15	V
		I_O = 1A , V_{IN}=V_{OUT}+ 0.8V		1.0	1.3	V
Thermal Regulation		T_A = 25℃ 30㎳ Pulse		0.003		%/W

ELECTRICAL CHARACTERISTICS FOR LM1117 S/T-1.5

(Refer to the test circuits, T_J=0 to 125℃ C_O=10μF unless otherwise specified)

PARAMETER	SYMBOL	TEST CONDITION	MIN.	TYP.	MAX.	UNIT
Output Voltage	V_O	V_{IN}= 4.5V, I_O = 10mA, T_J= 25℃	1.485	1.5	1.515	V
Output Voltage	V_O	I_O = 0 to 1A, V_{IN}= 3.9 to 10V	1.475		1.525	V
Line Regulation	ΔV_O	V_{IN} = 3.9 to 10V, I_O = 0mA		0.04	0.2	mV
Load Regulation	ΔV_O	V_{IN}= 3.9V, I_O = 0 to 1A		0.08	0.4	mV
Temperature Stability	ΔV_O			0.5		%
Long Term Stability	ΔV_O	1000 hrs, T_J = 125℃		0.3		%
Operating Input Voltage	V_{IN}	I_O = 100mA			7	V
Quiescent Current	I_D	$V_{IN} \leq$ 10V		5	10	mA
Output Current	I_O	V_{IN}= 7.5V, T_J= 25℃	800	950	1200	mA
Output Noise Voltage	$_EN$	B = 10Hz to 10kHz, T_J= 25℃		100		μV
Supply Voltage Rejection	SVR	I_O = 40mA , f = 120Hz, T_J= 25℃ V_{IN} = 5.5V, V_{NIPPLE} = $1V_{PP}$	60	75		dB
Dropout Voltage	V_D	I_O = 100mA		1	1.1	V
		I_O = 500mA		1.05	1.15	V
		I_O = 800mA		1.1	1.2	V
Thermal Regulation		T_A = 25℃ 30ms Pulse		0.003		%/W

ELECTRICAL CHARACTERISTICS FOR LM1117 S/T-1.8

(Refer to the test circuits, T_J=0 to 125℃ C_O=10μF unless otherwise specified)

PARAMETER	SYMBOL	TEST CONDITION	MIN.	TYP.	MAX.	UNIT
Output Voltage	V_O	V_{IN}= 4.5V, I_O = 10mA, T_J= 25℃	1.782	1.8	1.818	V
Output Voltage	V_O	I_O = 0 to 1A, V_{IN}= 3.9 to 10V	1.772		1.828	V
Line Regulation	ΔV_O	V_{IN} = 3.9 to 10V, I_O = 0mA		0.04	0.2	mV
Load Regulation	ΔV_O	V_{IN}= 3.9V, I_O = 0 to 1A		0.08	0.4	mV
Temperature Stability	ΔV_O			0.5		%
Long Term Stability	ΔV_O	1000 hrs, T_J = 125℃		0.3		%
Operating Input Voltage	V_{IN}	I_O = 100mA			7	V
Quiescent Current	I_D	$V_{IN} \leq$ 10V		5	10	mA
Output Current	I_O	V_{IN}= 7.5V, T_J= 25℃	800	950	1200	mA
Output Noise Voltage	$_EN$	B = 10Hz to 10kHz, T_J= 25℃		100		μV
Supply Voltage Rejection	SVR	I_O = 40mA , f = 120Hz, T_J= 25℃ V_{IN} = 5.5V, V_{NIPPLE} = $1V_{PP}$	60	75		dB
Dropout Voltage	V_D	I_O = 100mA		1	1.1	V
		I_O = 500mA		1.05	1.15	V
		I_O = 800mA		1.1	1.2	V
Thermal Regulation		T_A = 25℃ 30ms Pulse		0.003		%/W

ELECTRICAL CHARACTERISTICS FOR LM1117 S/T-2.5

(Refer to the test circuits, T_J=0 to 125℃ C_O=10μF unless otherwise specified)

PARAMETER	SYMBOL	TEST CONDITION	MIN.	TYP.	MAX.	UNIT
Output Voltage	V_O	V_{IN}= 4.5V, I_O = 10mA, T_J = 25℃	2.475	2.5	2.525	V
Output Voltage	V_O	I_O = 0 to 1A, V_{IN}= 3.9 to 10V	2.46		2.54	V
Line Regulation	ΔV_O	V_{IN} = 3.9 to 10V, I_O = 0mA		0.04	0.2	mV
Load Regulation	ΔV_O	V_{IN}= 3.9V, I_O = 0 to 1A		0.08	0.4	mV
Temperature Stability	ΔV_O			0.5		%
Long Term Stability	ΔV_O	1000 hrs, T_J = 125℃		0.3		%
Operating Input Voltage	V_{IN}	I_O = 100mA			7	V
Quiescent Current	I_D	V_{IN} ≤ 10V		5	10	mA
Output Current	I_O	V_{IN}= 7.5V, T_J = 25℃	800	950	1200	mA
Output Noise Voltage	$_EN$	B = 10Hz to 10kHz, T_J = 25℃		100		μV
Supply Voltage Rejection	SVR	I_O = 40mA , f = 120Hz, T_J = 25℃ V_{IN} = 5.5V, V_{NIPPLE} = $1V_{PP}$	60	75		dB
Dropout Voltage	V_D	I_O = 100mA		1	1.1	V
		I_O = 500mA		1.05	1.15	V
		I_O = 800mA		1.1	1.2	V
Thermal Regulation		T_A = 25℃ 30ms Pulse		0.01	0.1	%/W

ELECTRICAL CHARACTERISTICS FOR LM1117 S/T-2.85

(Refer to the test circuits, T_J=0 to 125℃ C_O=10μF unless otherwise specified)

PARAMETER	SYMBOL	TEST CONDITION	MIN.	TYP.	MAX.	UNIT
Output Voltage	V_O	V_{IN}= 4.85V, I_O = 10mA, T_J = 25℃	2.821	2.85	2.879	V
Output Voltage	V_O	I_O = 0 to 1A, V_{IN}= 4.25 to 10V	2.805		2.895	V
Line Regulation	ΔV_O	V_{IN} = 4.25 to 10V, I_O = 0mA		0.004	0.2	mV
Load Regulation	ΔV_O	V_{IN}= 4.25V, I_O = 0 to 1A		0.08	0.4	mV
Temperature Stability	ΔV_O			0.5		%
Long Term Stability	ΔV_O	1000 hrs, T_J = 125℃		0.3		%
Operating Input Voltage	V_{IN}	I_O = 100mA			7	V
Quiescent Current	I_D	V_{IN} ≤ 10V		5	10	mA
Output Current	I_O	V_{IN}= 7.85V, T_J = 25℃	800	950	1200	mA
Output Noise Voltage	$_EN$	B = 10Hz to 10kHz, T_J = 25℃		100		μV
Supply Voltage Rejection	SVR	I_O = 40mA , f = 120Hz, T_J = 25℃ V_{IN} = 5.85V, V_{NIPPLE} = $1V_{PP}$	60	75		dB
Dropout Voltage	V_D	I_O = 100mA		1	1.1	V
		I_O = 500mA		1.05	1.15	V
		I_O = 800mA		1.1	1.2	V
Thermal Regulation		T_A = 25℃ 30ms Pulse		0.003		%/W

ELECTRICAL CHARACTERISTICS FOR LM1117 S/T-3.0

(Refer to the test circuits, T_J=0 to 125℃ C_O=10μF unless otherwise specified)

PARAMETER	SYMBOL	TEST CONDITION	MIN.	TYP.	MAX.	UNIT
Output Voltage	V_O	V_{IN}= 5V, I_O = 10mA, T_J= 25℃	2.97	3	3.03	V
Output Voltage	V_O	I_O = 0 to 1A, V_{IN}= 4.5 to 10V	2.95		3.05	V
Line Regulation	ΔV_O	V_{IN} = 4.5 to 12V, I_O = 0mA		0.04	0.2	mV
Load Regulation	ΔV_O	V_{IN}= 4.5V, I_O= 0 to 1A		0.08	0.4	mV
Temperature Stability	ΔV_O			0.5		%
Long Term Stability	ΔV_O	1000 hrs, T_J = 125℃		0.3		%
Operating Input Voltage	V_{IN}	I_O = 100mA			7	V
Quiescent Current	I_D	$V_{IN} \leq$ 12V		5	10	mA
Output Current	I_O	V_{IN}= 8V, T_J= 25℃	800	950	1200	mA
Output Noise Voltage	$_EN$	B = 10Hz to 10kHz, T_J= 25℃		100		μV
Supply Voltage Rejection	SVR	I_O= 40mA , f = 120Hz, T_J= 25℃ V_{IN} = 6V, V_{NIPPLE} = 1V_{PP}	60	75		dB
Dropout Voltage	V_D	I_O = 100mA		1	1.1	V
		I_O = 500mA		1.05	1.15	V
		I_O = 800mA		1.1	1.2	V
Thermal Regulation		T_A = 25℃ 30ms Pulse		0.003	0.1	%/W

ELECTRICAL CHARACTERISTICS FOR LM1117 S/T-3.3

(Refer to the test circuits, T_J=0 to 125℃ C_O=10μF unless otherwise specified)

PARAMETER	SYMBOL	TEST CONDITION	MIN.	TYP.	MAX.	UNIT
Output Voltage	V_O	V_{IN}= 5.3V, I_O = 10mA, T_J= 25℃	3.267	3.3	3.333	V
Output Voltage	V_O	I_O = 0 to 1A, V_{IN}= 4.75 to 10V	3.247		3.353	V
Line Regulation	ΔV_O	V_{IN} = 4.75 to 15V, I_O = 0mA		0.04	0.2	mV
Load Regulation	ΔV_O	V_{IN}= 4.75V, I_O= 0 to 1A		0.08	0.4	mV
Temperature Stability	ΔV_O			0.5		%
Long Term Stability	ΔV_O	1000 hrs, T_J = 125℃		0.3		%
Operating Input Voltage	V_{IN}	I_O = 100mA			7	V
Quiescent Current	I_D	$V_{IN} \leq$ 15V		5	10	mA
Output Current	I_O	V_{IN}= 8.3V, T_J= 25℃	800	950	1200	mA
Output Noise Voltage	$_EN$	B = 10Hz to 10kHz, T_J= 25℃		100		μV
Supply Voltage Rejection	SVR	I_O= 40mA , f = 120Hz, T_J= 25℃ V_{IN} = 6.3V, V_{NIPPLE} = 1V_{PP}	60	75		dB
Dropout Voltage	V_D	I_O = 100mA		1	1.1	V
		I_O = 500mA		1.05	1.15	V
		I_O = 800mA		1.1	1.2	V
Thermal Regulation		T_A = 25℃ 30ms Pulse		0.003		%/W

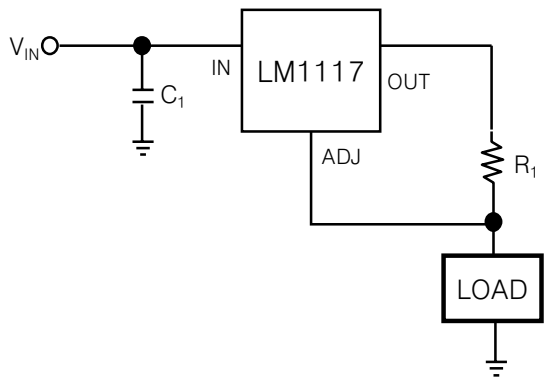
ELECTRICAL CHARACTERISTICS FOR LM1117 S/T-5.0

(Refer to the test circuits, T_J=0 to 125℃ C_O=10μF unless otherwise specified)

PARAMETER	SYMBOL	TEST CONDITION	MIN.	TYP.	MAX.	UNIT
Output Voltage	V_O	V_{IN}= 7V, I_O = 10mA, T_J= 25℃	4.95	5	5.05	V
Output Voltage	V_O	I_O = 0 to 1A, V_{IN}= 6.5 to 15V	4.92		5.08	V
Line Regulation	ΔV_O	V_{IN} = 6.5 to 15V, I_O = 0mA		0.04	0.2	mV
Load Regulation	ΔV_O	V_{IN}= 6.5V, I_O = 0 to 1A		0.08	0.4	mV
Temperature Stability	ΔV_O			0.5		%
Long Term Stability	ΔV_O	1000 hrs, T_J = 125℃		0.3		%
Operating Input Voltage	V_{IN}	I_O = 100mA			7	V
Quiescent Current	I_D	V_{IN} ≤ 15V		5	10	mA
Output Current	I_O	V_{IN}= 8.3V, T_J= 25℃	800	950	1200	mA
Output Noise Voltage	$_EN$	B = 10Hz to 10kHz, T_J = 25℃		100		μV
Supply Voltage Rejection	SVR	I_O = 40mA , f = 120Hz, T_J = 25℃ V_{IN} = 6.3V, V_{NIPPLE} = $1V_{PP}$	60	75		dB
Dropout Voltage	V_D	I_O = 100mA		1	1.1	V
		I_O = 500mA		1.05	1.15	V
		I_O = 800mA		1.1	1.2	V
Thermal Regulation		T_A = 25℃ 30ms Pulse		0.003		%/W

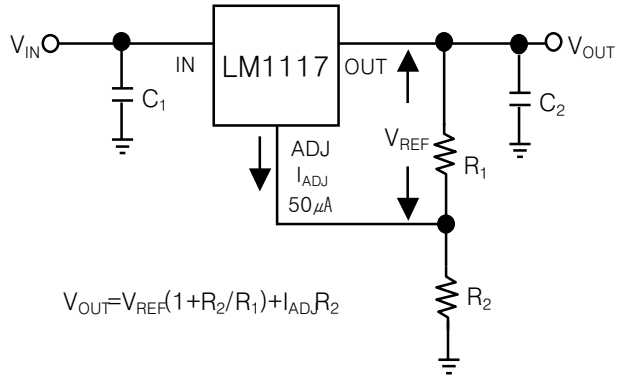
LM1117 (ADJUSTABLE) ELECTRICAL CHARACTERISTICS

Figure 1.



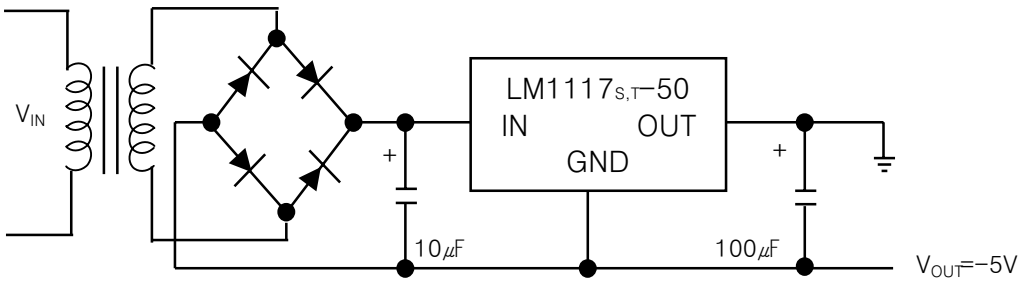
1A Current Output

Figure 2.



Typical Adjustable Regulator

Figure 3. Negative Supply



FLOATING INPUT

Figure 4. Active Terminator for SCSI-2 BUS

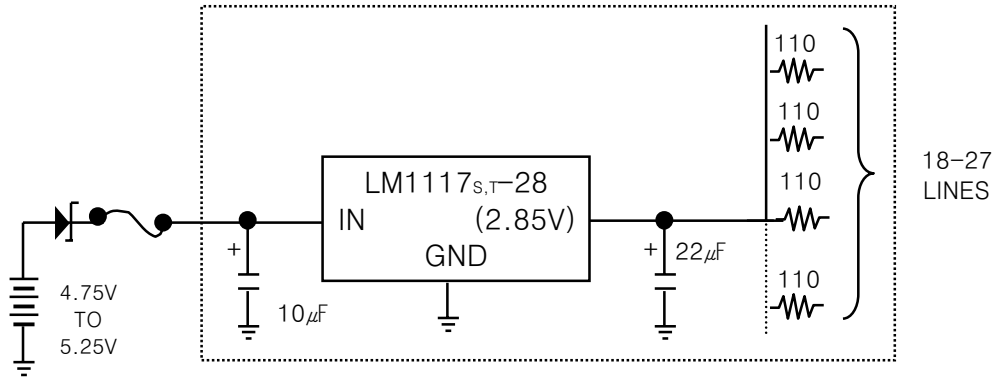


Figure 5. Voltage Regulator With Reference

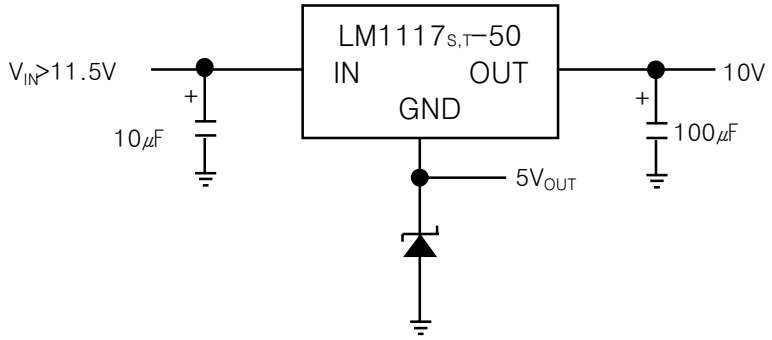
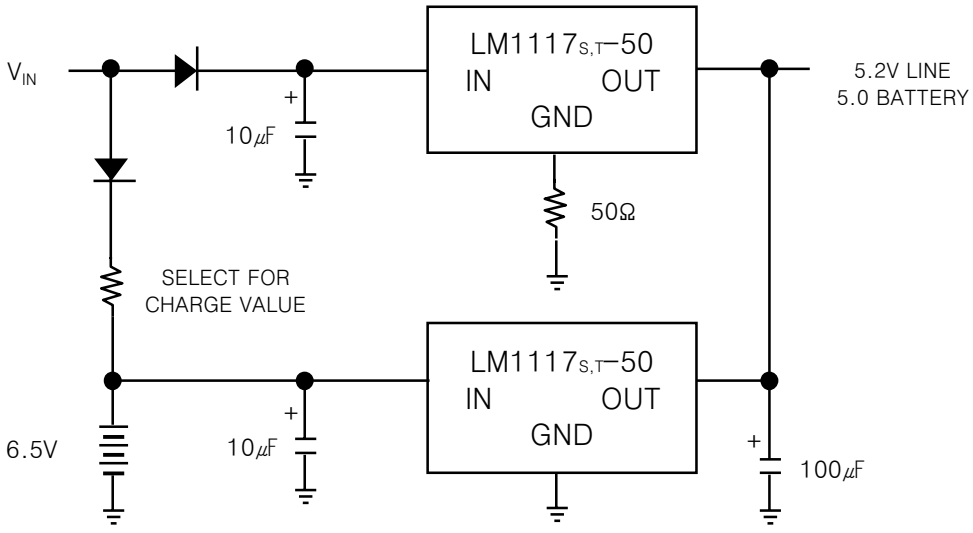
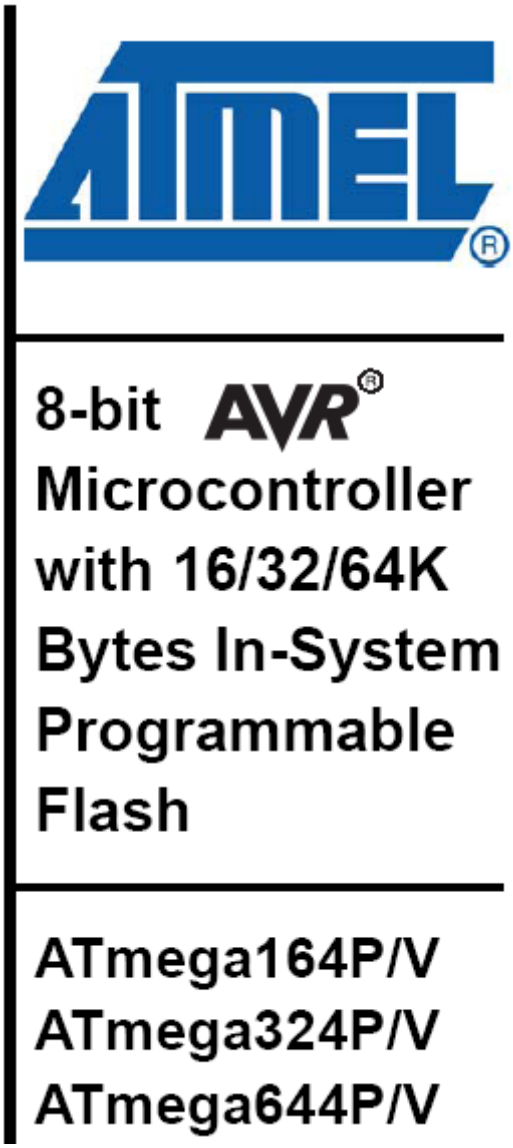


Figure 6. Battery Backed-up Regulated Supply



ANEXO 5:

DESCRIPCIÓN DEL MICROCONTROLADOR ATMEGA 164/644



Características:

- **Microcontrolador AVR de 8 bits de alto rendimiento y bajo consumo.**

- **Arquitectura Avanzada RISC**
  - 131 instrucciones. La mayoría de un solo ciclo de reloj de ejecución.
  - 32 registros de trabajo de 8 bits para propósito general.
  - Funcionamiento estático total.
  - Capacidad de procesamiento de unos 20 MIPS a 20 MHz.
  - Multiplicador por hardware de 2 ciclos

- **Memorias de programa y de datos no volátiles de alta duración**
  - 16/32/44 K bytes de FLASH auto programable en sistema
  - 512B/1K/2K bytes de EEPROM
  - 1/2/4K bytes de SRAM Interna
  - Ciclos de escritura/borrado: 10.000 en Flash / 100.000 en EEPROM

- Retención de Datos: 20 años a 85ºC / 100 años a 25ºC
- Sección opcional de código Boot con bits de bloqueo independientes. Programación en sistema del programa Boot que se encuentra dentro del mismo chip. Operación de lectura durante la escritura.
- Bloqueo programable para la seguridad del software.

❖ **Interfase JTAG**
- Capacidades de Boundary Scan de acuerdo con el estándar JTAG
- Soporte Extendido Debug dentro del chip
- Programación de FLASH, EEPROM, fusibles y bits de bloqueo a través de la interfase JTAG.

❖ **Características de los periféricos**
- Dos Timer/Contadores de 8 bits con prescalamiento separado y modo comparación.
- Un Timer/Contador de 16 bits con prescalamiento separado, modo comparación y modo de captura.
- Contador en Tiempo Real con Oscilador separado
- 6 Canales para PWM
- ADC de 10 bits y 8 canales
  Modo Diferencial con ganancia seleccionable a x1, x10 o x200.
- Interfase serie de dos hilos con byte orientado.
- Dos puertos Seriales USART Programables
- Interfaz Serial SPI maestro-esclavo
- Watchdog Timer programable con oscilador independiente, dentro del mismo chip.
- Comparador Analógico dentro del mismo Chip
- **Interrupt and Wake-up on Pin Change**

❖ **Características especiales del microcontrolador**
- Power-on Reset (en el encendido) y detección de Brown-out (pérdida de polarización) programable.
- Oscilador RC interno calibrado.
- Fuentes de interrupción externas e internas.
- 6 modos de descanso: Idle, Reducción de Ruido ADC, Power-save, Power-down, Standby y Standby extendido.

❖ **Encapsulados para Entradas/Salidas (E/S)**
- 32 líneas de E/S programables.
- PDIP de 40 pines, TQFP y QFN/MLF de 44 pines.

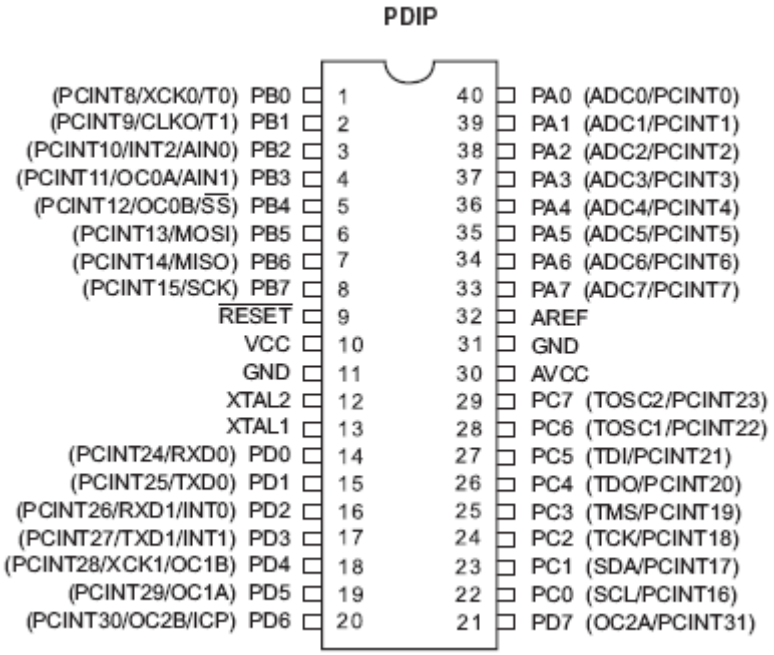
❖ **Voltajes de Operación**
- 1.8 – 5.5V para el ATMEGA 164P/324P/644PV
- 2.7 – 5.5V para el ATMEGA 164P/324P/644P

- **Velocidad de Funcionamiento**
  - ATMEGA 164P/324P/644PV: 0 – 4MHz @ **1.8 – 5.5V - 10MHz @ 2.7 – 5.5V**
  - ATMEGA 164P/324P/644PV: 0 – 10MHz @ **2.7 – 5.5V - 20MHz @ 4.5 – 5.5V**

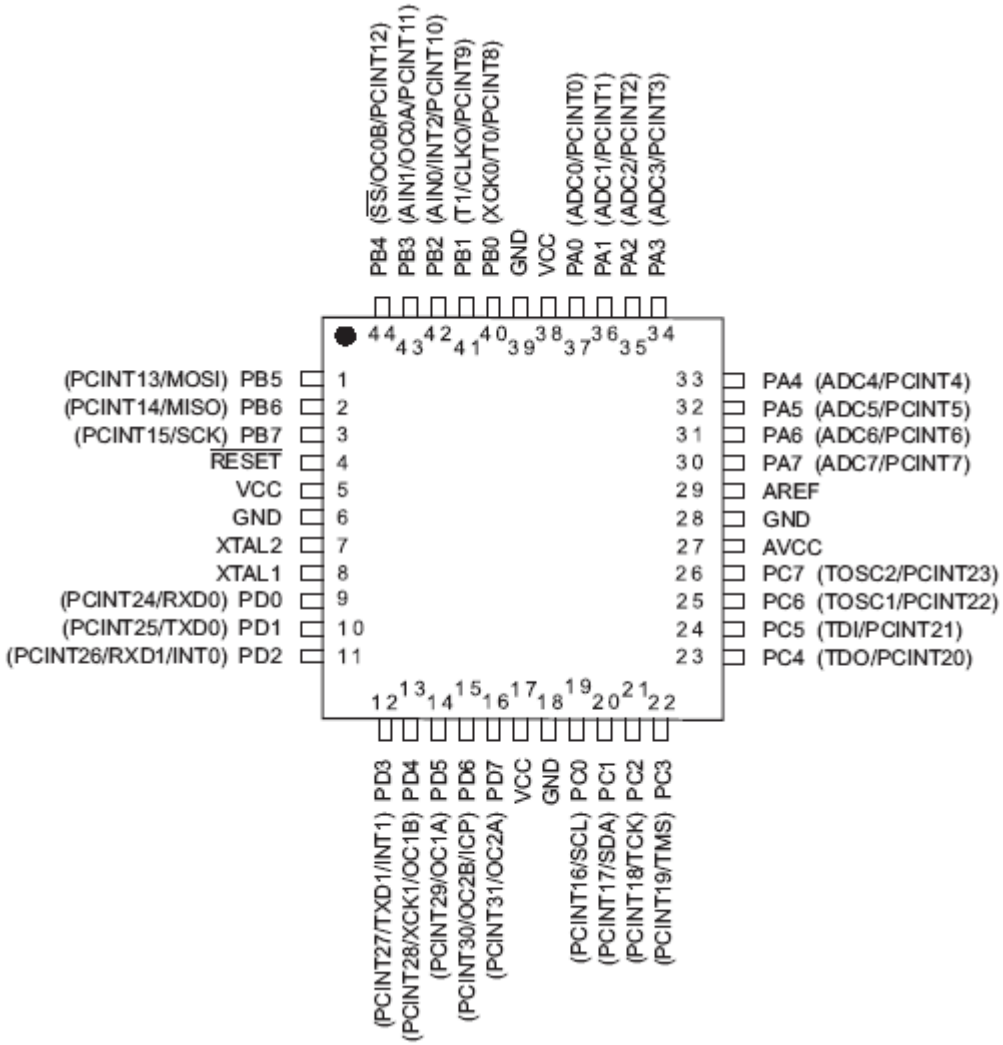
- **Consumo de energía a 1MHz, 1.8V, 25°C para el ATMEGA 164P/324P/644P**
  - Activo: 0.4mA
  - Modo Power-down: 0.1uA
  - Modo Power-Save: 0.6uA (Incluye RTC de 32 Khz)

1. CONFIGURACIÓN DE PINES

Figura 1-1. Pines de Salida del ATmega164P/324P/644P



TQFP/QFN/MLF

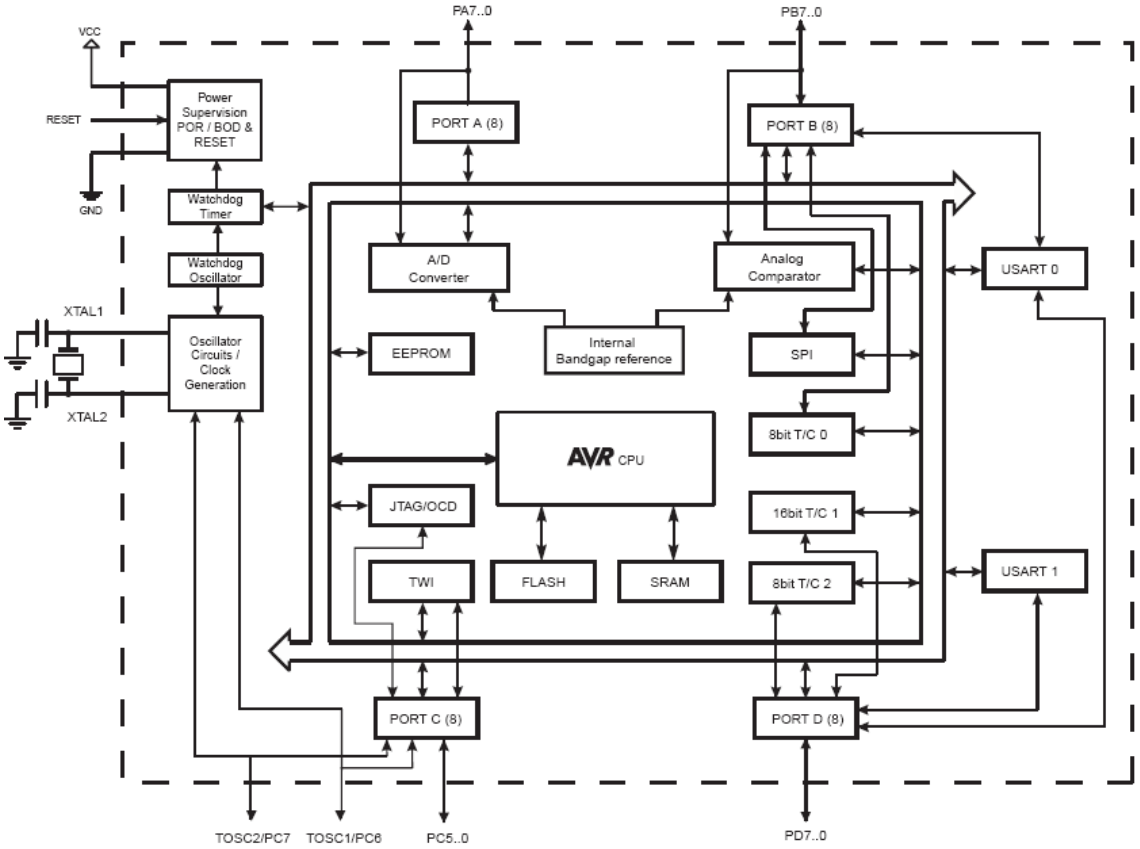


2. Revisión Global

El ATmega164P/324P/644P es un microcontrolador CMOS de 8 bits de bajo consumo basado en la arquitectura RISC mejorada. Sus instrucciones se ejecutan en un ciclo de máquina, el ATmega164P / 324P / 644P consigue transferencia de información alrededor de 1 MIPS por MHz admitido por el sistema, permitiendo al diseñador del sistema optimizar el consumo de energía versus la velocidad de procesamiento.

2.1 Diagrama de Bloque

Figura 2 -1. Diagrama de Bloques



El core (núcleo) AVR combina un conjunto de instrucciones RISC con 32 registros para uso de propósito general. Todos los 32 registros están directamente relacionados con la Unidad Aritmética Lógica (ALU), admitiendo dos registros independientes al ejecutarse una instrucción en un ciclo de máquina. El resultado de esta arquitectura es más eficiente, se consigue un caudal de flujo y transferencia hasta diez veces más rápido que microcontroladores CISC convencionales.

El ATmega164P / 324P / 644P provee las siguientes características: 16/32 / 64K bytes en el sistema de Flash Programable con capacidad de lectura y escritura de 512B/1K/2K bytes en la EEPROM, 1/2/4K bytes en la SRAM, 32 pines de E/S para propósito general, 32 registros de propósito general, Contador en Tiempo real (RTC), tres Timer/Contadores flexibles con modo de Comparación y PWM, 2 USARTs, un byte orientado a la Interfaz Serial de 2 hilos, 8 canales ADC de 10 bits con opción de entrada Diferencial con ganancia programable, Watchdog Timer programable con oscilador interno, un Puerto serial SPI, Interfase de prueba JTAG, también usado para acceder al sistema On-chip Debug y seis modos de programación seleccionable para ahorro de energía. El modo Idle detiene al CPU mientras permite a la SRAM, Timer/Contador, Puerto SPI y al sistema de interrupciones continuar funcionando.

El Modo Power-down guarda el contenido de los registros pero paraliza al oscilador, desactiva todas las otras funciones de chip hasta la próxima interrupción o mediante reseteo por hardware. En el Modo Power-save, el reloj asincrónico continúa corriendo, permitiendo tener actualizado al reloj mientras el resto de dispositivos están descansando. El Modo de Reducción del Ruido del ADC detiene al CPU y a todos los módulos de E/S excepto al Reloj Asincrónico y al ADC, para minimizar el ruido durante la conversión. En el Modo Standby, el oscilador Cristal/Resonador está corriendo mientras el resto de dispositivos están descansando. Estos permiten comenzar una rápida combinación con el consumo de baja energía.

En el Modo de Standby extendido, corre el Oscilador principal y el Reloj Asincrónico. Este elemento es hecho usando tecnología de alta densidad de memoria no volátil de ATMEL. El chip interno ISP de la FLASH permite a la memoria de programa ser reprogramada a través del puerto interno ISP mediante un programador convencional no volátil o mediante un programa interno en el dispositivo AVR. El programa de inicialización puede usar cualquier interfaz para descargar el programa de aplicación en la memoria flash. El programa en la sección Flash Boot es actualizado mientras continúa corriendo la sección de aplicaciones de la Flash, proporcionando una escritura-lectura verdadera de operación. Para combinar un CPU RISC de 8 bits en un sistema de Flash Auto-programable en un chip monolítico, el ATmega164P/324P/644P es un poderoso microcontrolador que provee una alta flexibilidad y solución de costos efectivos para cualquier aplicación de control.

El ATmega164P / 324P / 644P AVR es soportado con un juego completo de programas y herramientas de desarrollo del sistema incluyendo: compiladores de C, ensambladores de macro, depurador / simuladores de programa, emuladores de circuitos y equipos de evaluación.

2.2 Comparación entre el ATmega164P, ATmega324P and ATmega644P

Table 2-1. Differences between ATmega164P and ATmega644P

Device	Flash	EEPROM	RAM
ATmega164P	16 Kbyte	512 Bytes	1 Kbyte
ATmega324P	32 Kbyte	1 Kbyte	2 Kbyte
ATmega644P	64 Kbyte	2 Kbyte	4 Kbyte

2.3 Descripción de Pines

2.3.1 VCC
Alimentación de Voltaje Digital

2.3.2 GND
Tierra

2.3.3 Puerto A (PA7:PA0)

El puerto A sirve como entradas analógicas para el conversor Análogo Digital.
El puerto A también sirve como un puerto bidireccional de 8 bits con resistencias internas de pull up (seleccionables para cada bit). Los buffers de salida del puerto A tienen características simétricas controladas con fuentes de alta capacidad.

Los pines del puerto A están en tri-estado cuando las condiciones de reset están activadas o cuando el reloj no este corriendo. El puerto A también sirve para varias funciones especiales del ATmega164P/324P/644P como la Conversión Análoga Digital.

2.3.4 Port B (PB7:PB0)

El puerto B es un puerto bidireccional de 8 bits de E/S con resistencias internas de pull up.
Las salidas de los buffers del puerto B tienen características simétricas controladas con fuentes de alta capacidad.

Los pines del puesto B están en tri-estado cuando las condiciones de reset están activadas o cuando el reloj no esté corriendo. El puerto B también sirve para varias funciones especiales del ATmega164P/324P/644P como se menciona en las páginas iniciales.

2.3.5 Port C (PC7:PC0)

El puerto C es un puerto bidireccional de 8 bits de E/S con resistencias internas de pull up (seleccionadas por cada bit). Las salidas de los buffers del puerto C tienen características simétricas controladas con fuentes de alta capacidad.

Los pines del puesto C están en tri-estado cuando las condiciones de reset están activadas siempre y cuando el reloj no este corriendo. El puerto C también sirve para las funciones de Interfaz del JTAG, con funciones especiales del ATmega164P/324P/644P como se menciona en las páginas iniciales.

2.3.6 Port D (PD7:PD0)

El Puerto D es un puerto bidireccional de entradas y salidas con resistencias internas de pull up (seleccionadas por cada bit). Las salidas de los buffers del puerto D tienen características simétricas controladas con sumideros de fuentes de alta capacidad.
Los pines del Puerto D están en tri-estado cuando llega una condición de reset activa, siempre y cuando el reloj no esté corriendo**.**
El puerto D también sirve para varias funciones especiales del ATmega164P/324P/644P como se menciona en las páginas iniciales.

2.3.7 $\overline{\text{RESET}}$

Entrada del Reset. Un pulso de nivel bajo en este pin por períodos de pulso mínimo genera un reset, siempre y cuando el reloj no esté corriendo.

La longitud del pulso mínimo está especificada en las Características y Sistemas de Reset (Páginas 331 del Data Sheet). Pulsos cortos no son garantizados para generar un reset.

2.3.8 XTAL1

Entrada para el amplificador del oscilador invertido y entrada para el circuito de operación del reloj interno.

2.3.9 XTAL2

Salida del Oscilador amplificador de salida**.**

2.3.10 AVCC

AVCC es la alimentación de voltaje para el pin del Puerto F y el Conversor Análogo a Digital. Este debe ser conectado externamente a VCC, siempre y cuando el ADC no sea usado. Si el ADC es usado, este deberá ser conectado a VCC a través de un filtro paso bajo.

2.3.11 AREF

Está es la referencia para el pin de la conversión Análoga a Digital.

3. Recursos

Un set comprensible de herramientas, notas de aplicación y datasheet se puede descargar desde http://www.atmel.com/avr.

4. Acerca de los Ejemplos de Código

Este documento contiene unos simples ejemplos de código que brevemente muestran como usar varias partes del dispositivo. Sea consiente que no todos los vendedores de compiladores en C incluyen la definición de bits en el archivo de cabecera y el manejo de interrupciones en el compilador C es dependiente. Por favor confirme con la documentación del compilador C para más detalles.

Los ejemplos de código asumen que la parte de archivo de cabecera esta incluido antes de la compilación. Para las localidades de los registros de E/S en el mapa extendido de E/S, las instrucciones "IN", "OUT", "SBIS", "SBIC", "CBI", y "SBI" serían reemplazadas con instrucciones de acceso extendido E/S. Típicamente "LDS" y "STS" combinadas con "SBRS", "SBRC", "SBR", y "CBR".

5. Retención de Datos

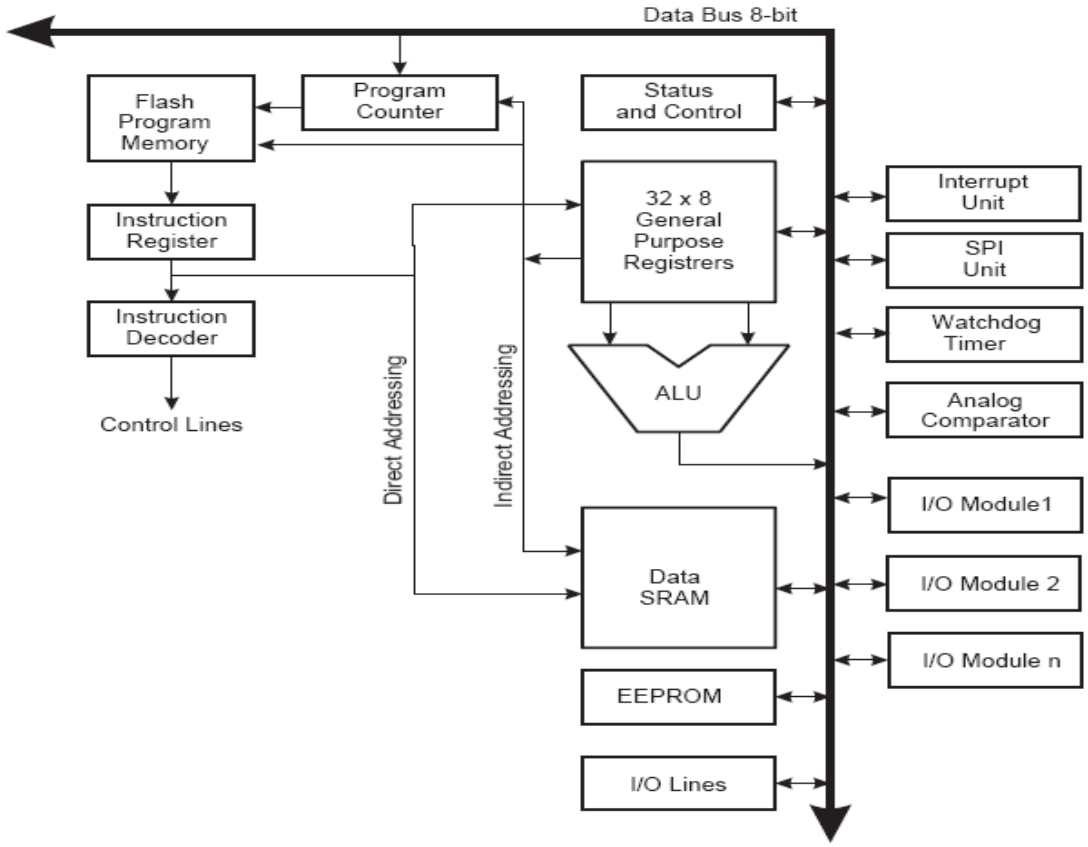
La fiabilidad de la calificación de resultados muestra que la velocidad de falla de un proyecto es mucho menor que 1 PPM en 20 años a 85°C ó 100 años a 25°C.

6. AVR CPU Core

6.1 VISIÓN GENERAL

Esta sección discute la arquitectura general del AVR. La principal función del AVR es asegurar la correcta ejecución del programa.

La CPU debe ser capaz de acceder a la memoria, llevar a cabo cálculos, control de periféricos y atención de interrupciones.



Para maximizar el rendimiento y el paralelismo, el AVR usa una arquitectura de Hardware con separador de memorias y buses para programa y datos. Las instrucciones en la memoria de programa son ejecutadas con un simple nivel de colas. Mientras una instrucción es ejecutada, la siguiente instrucción es ejecutada desde la memoria de programa. Este concepto permite que las instrucciones sean ejecutadas en cada ciclo de máquina. La memoria de programa está en la memoria Flash re-programable.
El Archivo del Registro (Register File) de rápido acceso contiene 32 registros de propósito general de 8 bits trabajando en un simple ciclo de reloj.
Esto permite una operación de ciclo simple en la Unidad Aritmética lógica. En una operación típica de la ALU, dos operandos están fuera del Archivo de Registro, la operación es ejecutada, y el resultado es guardado en el Archivo de Registro en un ciclo de máquina.

Seis de los 32 registros pueden ser usados como tres registros punteros de 16 bits de dirección, para direccionar los Datos y permitir los cálculos de direcciones diferentes.

Uno de estos tres punteros puede ser usado como un puntero de direcciones para tablas en la memoria de programa de la Flash. Estos registros de función adicionales son el X, Y y Z de 16 bits, descritos después en esta sección. La ALU soporta operaciones lógicas y aritméticas entre registros o entre constantes y registros.

Simples operaciones de registros pueden ser ejecutadas en la ALU. Después de una operación aritmética, el registro de estado es actualizado para reflejar información acerca de los resultados de la operación. El flujo del programa es provisto por un salto condicional e incondicional y llamado de interrupciones, capaz de direccionar espacios de direcciones completamente.

La mayoría de instrucciones del AVR tienen un formato simple de una palabra de 16 bits. Cada dirección de memoria contiene instrucciones de 16 o 32 bits. El espacio de memoria en la flash está dividido en dos secciones, la sección Baja del programa y la sección de aplicación de programa.

Ambas secciones están dedicadas para bloqueo de escritura y protección de lectura/escritura.
La instrucción SMP que se escribe en la Sección de la memoria Flash debe residir en la sección Baja del programa.

Durante los llamados de interrupción y subrutinas, la dirección de regreso del Contador de Programa (CP) es almacenado en la pila (stack).

La pila (stack) está localizada efectivamente en la SRAM (RAM estática) de datos y consecuentemente el tamaño de la pila está limitado solo por el tamaño total de la SRAM (RAM estática) y su uso.
Todos los programas a usarse deben inicializar en el SP (Stack Pointer) en la rutina del Reset (antes de que sea ejecutada una interrupción o una subrutina).
El puntero de pila (Stack pointer SP) es la lectura/escritura accesible en el espacio de E/S. La RAM estática de datos puede ser fácilmente penetrada a través de los cinco diferentes modos de direccionamiento soportados en la arquitectura de AVR.
El espacio de memoria en la arquitectura de los mapas de memoria de los AVR son todos lineales y regulares. Un módulo de interrupción flexible tiene sus registros de control en los espacios de E/S con una Habilitación de Interrupción Global en el Registro de Estado.

Todas las interrupciones tienen separado un vector de interrupciones en la tabla del vector de interrupciones. Las interrupciones tienen prioridad de conformidad con su vector de interrupciones. La dirección más baja del vector de interrupciones tiene alta prioridad.

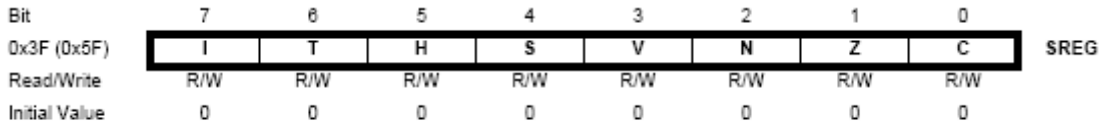
El espacio de memoria de E/S contiene 64 direcciones para las funciones periféricas de la CPU, el Registro de Control SPI y otras funciones de Entrada y Salida. La memoria de Entrada y Salida puede ser accedida directamente o como localidades de espacio de datos siguiendo estos Archivos de Registro:0x20 - 0x5F. En suma, el ATmega164P/324P/644P tiene espacios extendidos de entrada y salida desde la dirección 0x60 - 0xFF en la SRAM donde solo las instrucciones ST/STS/STD y LD/LDS/LDD pueden ser usadas.

6.3 Registro de Estado

El registro de estado contiene información acerca de los resultados de las instrucciones aritméticas más recientes ejecutadas.
Está información puede ser usada para alterar el flujo del programa en el funcionamiento de operaciones condicionales. Note que el Registro de Estado es actualizado después de todas las operaciones de la ALU, como especificaciones en el Set de Instrucciones de Referencia. En algunos casos esto retira la necesidad de usar comparación de instrucciones dedicadas, resultando un código más rápido y compacto.
El Registro de estado no es almacenado automáticamente cuando entra una rutina de interrupción y se restituye cuando regresa de una interrupción.

6.3.1 SREG – Registro de Estado

El registro de Estado AVR – SREG – está definido por:



Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – I: Habilitación de Interrupciones Globales**

El bit de habilitación de las interrupciones globales debe estar en uno para habilitar las interrupciones. La interrupción individual permite que el control sea llevado a cabo en registros de control distintos. Si el registro de Habilitación de interrupciones globales es borrado, ninguna de las interrupciones están activadas independiente de la configuración de una interrupción individual. El bit I es limpiado por hardware después de que una interrupción ha ocurrido, y es puesto en uno por la instrucción de RETI para habilitar interrupciones siguientes. El bit I también puede ser puesto en uno y borrado por la aplicación con las instrucciones SEI y CLI.

- **Bit 6 – T: Bit Copia de almacenamiento**

Los bits de instrucción de copia BLD (cargar bit) y BST (almacenar bit) usa el bit T como una fuente o destino para la operación del bit. Un bit desde los registros de los Archivos de Registro pueden ser copiados en el bit T mediante la instrucción SBT y un bit en T puede ser copiado dentro de un registro del Archivo de registros mediante la instrucción BLD.

- **Bit 5 – H: Half Carry Flag**

Half Carry es útil en la aritmética BCD.

- **Bit 4 – S: Bit de Signo, S = N ⊕ V**

El Bit S es una OR exclusiva entre la bandera negativa N y la bandera de desbordamiento V en Complemento a Dos.

- **Bit 3 – V: Bandera de Desbordamiento V en Complemento a Dos**

La bandera de desbordamiento en Complemento a Dos soporta el complemento a dos.

- **Bit 2 – N: Bandera Negativa**

La Bandera Negativa N indica un resultado negativo en una operación aritmética o lógica.

• **Bit 1 – Z: Bandera del Cero**
La bandera del cero indica si un resultado es cero en una operación aritmética o lógica.

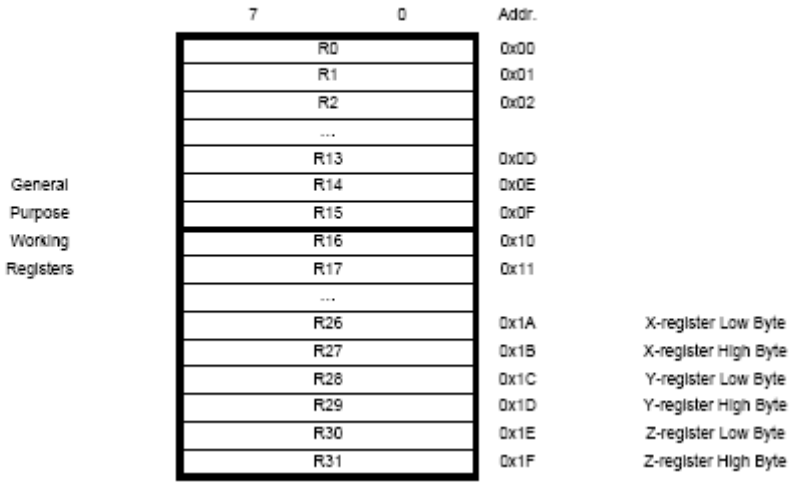
• **Bit 0 – C: Bandera del Carry**
La bandera del Carry indica un carry en una operación aritmética o lógica.

Archivo de registros de propósito General
El Archivo de Registros es optimizado por el Juego de instrucciones RISC del AVR.
Para lograr la actuación y flexibilidad requerida, los esquemas del entrada/salida se apoyan en los siguientes Archivos de Registros:

• Un operador de salida de 8 bits y una entrada resultante de 8 bits
• Dos operadores de salida de 8 bits y una entrada resultante de 8 bits
• Dos operadores de salida de 8 bits y una entrada resultante de 16 bits
• Un operador de salida de 16 bits y una entrada resultante de 16 bits

La siguiente Figura (6-2) muestra la estructura de los 32 Registros de Propósito General que trabajan en la CPU.

Figure 6-2. AVR CPU General Purpose Working Registers



La mayoría de las operaciones de instrucciones en el Archivo de Registro tiene acceso directo a todos los registros, y la mayoría de ellos son instrucciones de un ciclo.

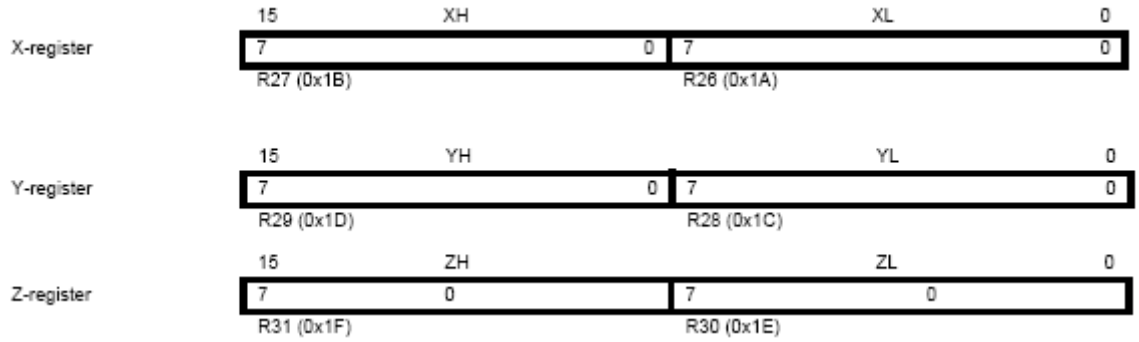
Como se muestra en la figura anterior a cada registro en la memoria de datos se le asigna una dirección, mapeados estos directamente dentro de las 32 localidades para el uso del espacio de Datos.

Aunque físicamente no son implementados en las localidades de la SRAM, esta memoria provee gran organización flexible para el acceso a estos registros, como los registros punteros X, Y y Z que se usan como índices de cualquier archivo de registro.

El registro X, el registro Y y el registro Z

Los registros R26… R31 tienen algunas funciones adicionales para uso de propósito general. Estos registros son punteros de 16 bits de dirección para direccionar indirectamente al espacio de datos.
Los tres registros de direccionamiento indirecto X, Y y Z están definidos como se muestra en la figura siguiente.

Figure 6-3. Los registros X, Y y Z



En los modos de direccionamiento directo estos registros de dirección tienen desplazamientos fijos, incrementos y decrementos automáticos.

6.5 Puntero de Pila (Stack Pointer)

El Stack–Pointer es usado para almacenar información temporal, para almacenar variables locales y para almacenar direcciones de regreso después de una interrupción o llamado de subrutinas.
Los registros del Puntero de Pila siempre apuntan a la parte superior de la pila.
Note que la pila es implementada como un crecimiento desde la localidad más alta a la localidad mas baja de la memoria. Esto implica que un comando PUSH decrementa el Puntero de Pila.
El puntero de pila para la SRAM es el área donde las interrupciones y subrutinas son localizadas. Estos espacios de pila en los datos de la SRAM deben ser definidos por el programa antes de que cualquier llamado de subrutina sea ejecutado o una interrupción sea habilitada.
El puntero de pila debe estar por encima de la localidad 0x0100. El valor inicial del puntero de pila es la última dirección de la SRAM interna. El puntero de pila es decrementado a uno cuando los datos presionan en la pila con la instrucción PUSH, y este es decrementado por tres cuando los datos llenan la pila con un regreso desde una subrutina RET o de regreso desde una interrupción RETI.

El Puntero de Pila del AVR es implementado como un registro de 8 bits en el espacio de E/S. El número de bits actualmente usados es dependientemente implementado. Note que el espacio de datos en algunas implementaciones en la arquitectura del AVR es tan pequeña que solo necesita el SPL. En este caso, el registro SPH no esta presente.

6.5.1 SPH y SPL – Puntero de Pila Alto y Puntero de Pila Bajo

Bit	15	14	13	12	11	10	9	8	
0x3E (0x5E)	–	–	–	SP12	SP11	SP10	SP9	SP8	SPH
0x3D (0x5D)	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	SPL
	7	6	5	4	3	2	1	0	
Read/Write	R	R	R	R/W	R/W	R/W	R/W	R/W	
	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0/0/1[1]	0/1/0[1]	1/0/0[1]	0	0	
	1	1	1	1	1	1	1	1	

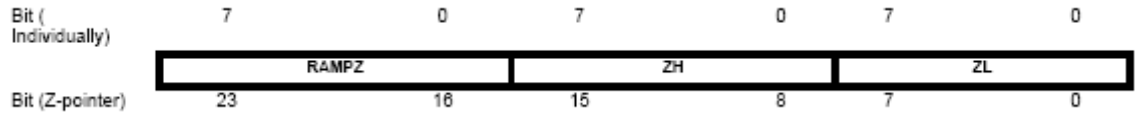
Note: 1Valores iniciales respectivamente para el ATmega164P/324P/644P.

6.5.2 RAMPZ – Z Extendido – Registro puntero para ELPM/SPM

Bit	7	6	5	4	3	2	1	0	
0x3B (0x5B)	RAMPZ7	RAMPZ6	RAMPZ5	RAMPZ4	RAMPZ3	RAMPZ2	RAMPZ1	RAMPZ0	RAMPZ
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Para las instrucciones ELPM/SPM, el puntero Z es una concatenación del RAMPPZ, ZH y ZL se muestran en la figura siguiente. Note que LPM no es afectado por la configuración del RAMPZ.

Figura. El puntero Z usado por ELPM y SPM.



El número actual de los bits es dependiente de la implementación. Al no usar los bits en una implementación se leerán como cero. Para compatibilidad con dispositivos futuros, asegurar el escribir estos bits como cero.

6.7 Reset e Interrupción Manual

El AVR provee diferentes Fuentes de interrupción. Cada de estas interrupciones y el Vector Separador de Reset tienen un Vector separador de programa en el espacio de la memoria de programa. Todas las interrupciones son bits habilitados de forma individual los cuales deben ser escritos a uno lógico junto con el bit Habilitador de Interrupciones Globales en el registro de Estado.

ANEXO 6:

DESCRIPCIÓN DEL MICROCONTROLADOR ATMEGA 48/88/168

Features

- High Performance, Low Power AVR® 8-Bit Microcontroller
- Advanced RISC Architecture
  - 131 Powerful Instructions – Most Single Clock Cycle Execution
  - 32 x 8 General Purpose Working Registers
  - Fully Static Operation
  - Up to 24 MIPS Throughput at 24 MHz
  - On-chip 2-cycle Multiplier
- Non-volatile Program and Data Memories
  - 4/8/16K Bytes of In-System Self-Programmable Flash (ATmega48/88/168)
    Endurance: 10,000 Write/Erase Cycles
  - Optional Boot Code Section with Independent Lock Bits
    In-System Programming by On-chip Boot Program
    True Read-While-Write Operation
  - 256/512/512 Bytes EEPROM (ATmega48/88/168)
    Endurance: 100,000 Write/Erase Cycles
  - 512/1K/1K Byte Internal SRAM (ATmega48/88/168)
  - Programming Lock for Software Security
- Peripheral Features
  - Two 8-bit Timer/Counters with Separate Prescaler and Compare Mode
  - One 16-bit Timer/Counter with Separate Prescaler, Compare Mode, and Capture Mode
  - Real Time Counter with Separate Oscillator
  - Six PWM Channels
  - 8-channel 10-bit ADC in TQFP and MLF package
  - 6-channel 10-bit ADC in PDIP Package
  - Programmable Serial USART
  - Master/Slave SPI Serial Interface
  - Byte-oriented 2-wire Serial Interface
  - Programmable Watchdog Timer with Separate On-chip Oscillator
  - On-chip Analog Comparator
  - Interrupt and Wake-up on Pin Change
- Special Microcontroller Features
  - Power-on Reset and Programmable Brown-out Detection
  - Internal Calibrated Oscillator
  - External and Internal Interrupt Sources
  - Five Sleep Modes: Idle, ADC Noise Reduction, Power-save, Power-down, and Standby
- I/O and Packages
  - 23 Programmable I/O Lines
  - 28-pin PDIP, 32-lead TQFP and 32-pad MLF
- Operating Voltage:
  - 1.8 - 5.5V for ATmega48V/88V/168V
  - 2.7 - 5.5V for ATmega48/88/168
- Temperature Range:
  - -40°C to 85°C
- Speed Grade:
  - ATmega48V/88V/168V: 0 - 6 MHz @ 1.8 - 5.5V, 0 - 12 MHz @ 2.7 - 5.5V
  - ATmega48/88/168: 0 - 12 MHz @ 2.7 - 5.5V, 0 - 24 MHz @ 4.5 - 5.5V
- Low Power Consumption
  - Active Mode:
    1 MHz, 1.8V: 240µA
    32 kHz, 1.8V: 15µA (including Oscillator)
  - Power-down Mode:
    0.1µA at 1.8V

8-bit AVR® Microcontroller with 8K Bytes In-System Programmable Flash

ATmega48/V
ATmega88/V
ATmega168/V

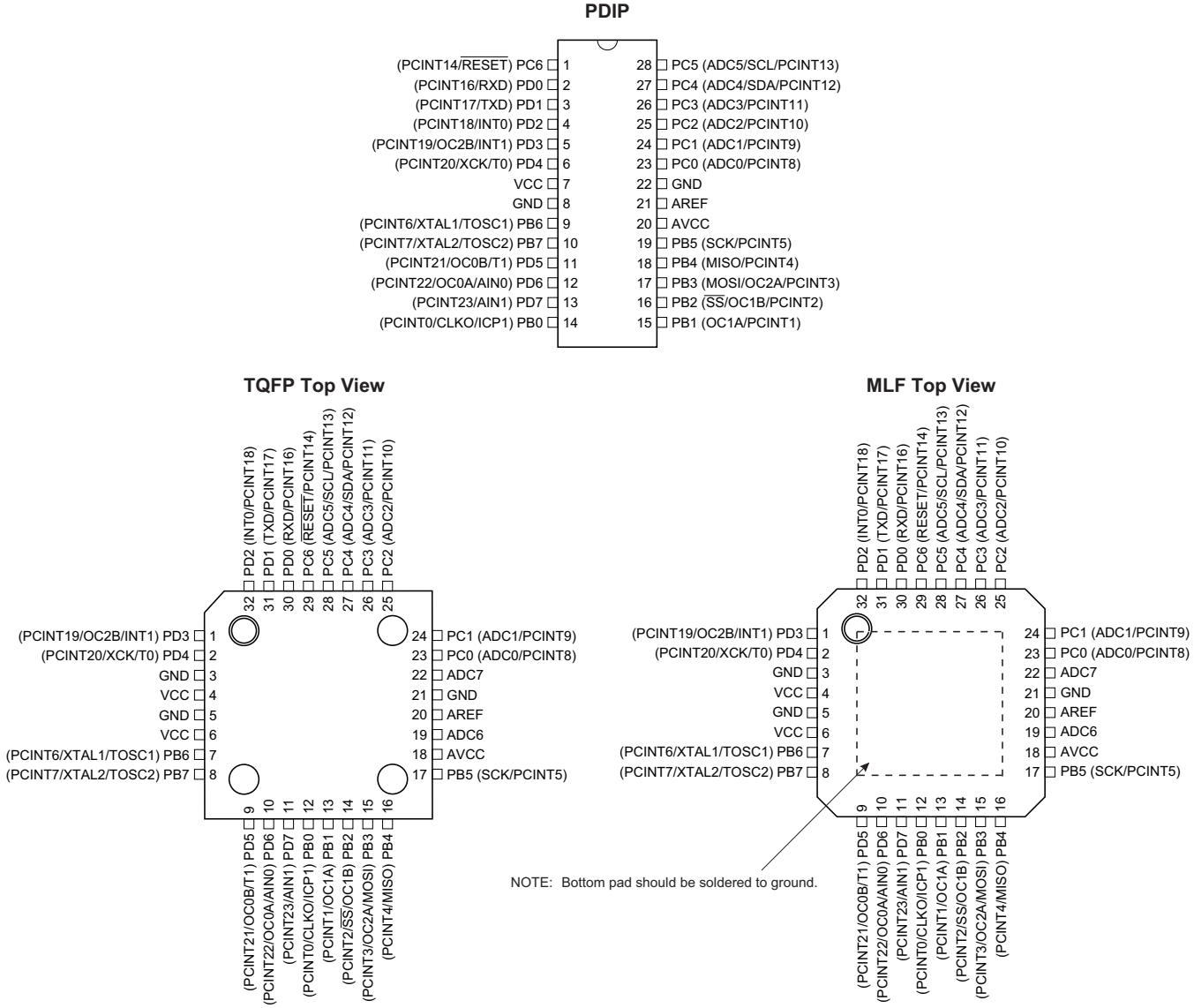
Preliminary Summary

Rev. 2545BS–AVR–01/04

Note: This is a summary document. A complete document is available on our Web site at www.atmel.com.

Pin Configurations

Figure 1. Pinout ATmega48/88/168



Disclaimer

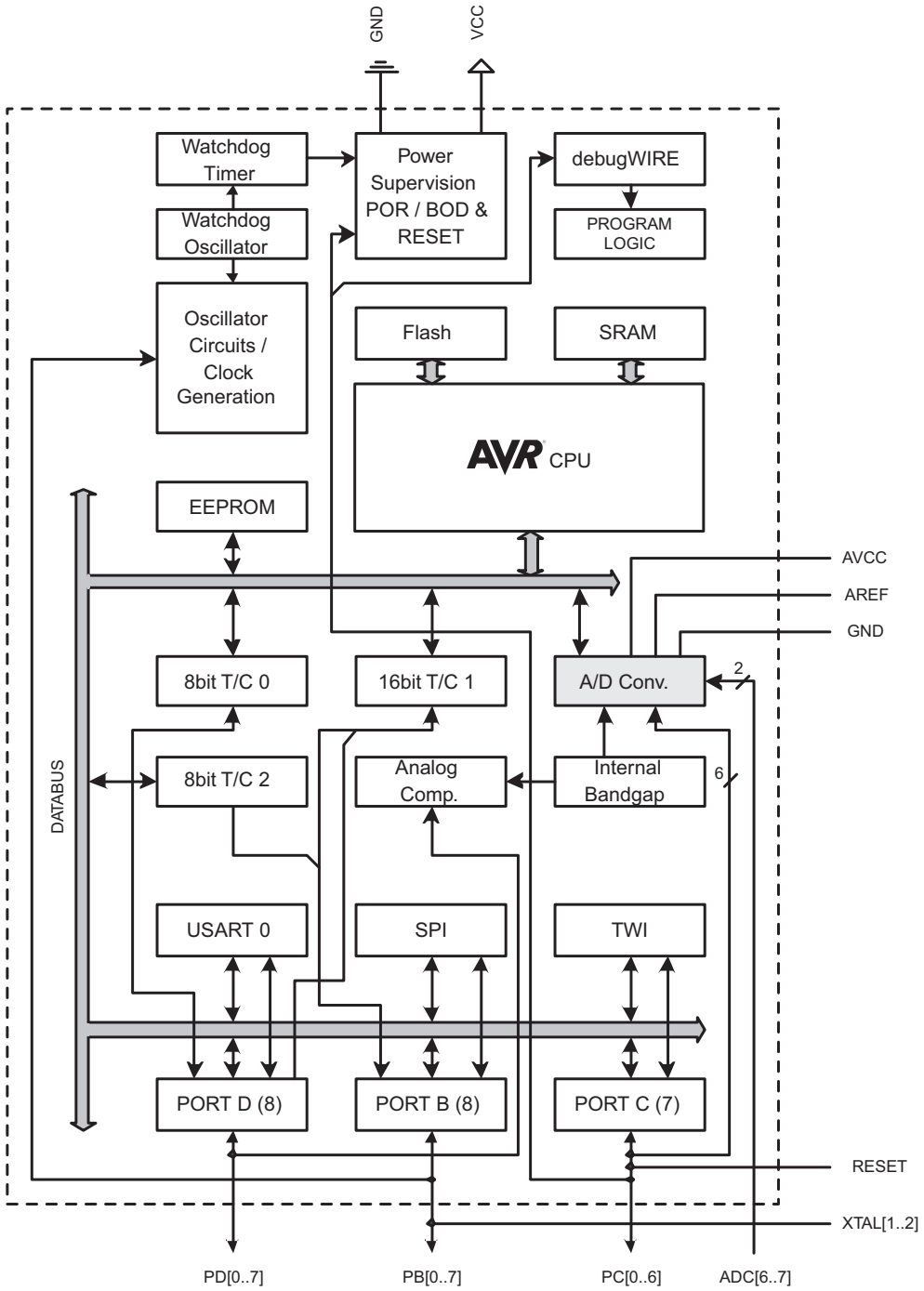
Typical values contained in this datasheet are based on simulations and characterization of other AVR microcontrollers manufactured on the same process technology. Min and Max values will be available after the device is characterized.

Overview

The ATmega48/88/168 is a low-power CMOS 8-bit microcontroller based on the AVR enhanced RISC architecture. By executing powerful instructions in a single clock cycle, the ATmega48/88/168 achieves throughputs approaching 1 MIPS per MHz allowing the system designer to optimize power consumption versus processing speed.

Block Diagram

Figure 2. Block Diagram



The AVR core combines a rich instruction set with 32 general purpose working registers. All the 32 registers are directly connected to the Arithmetic Logic Unit (ALU), allowing two independent registers to be accessed in one single instruction executed in one clock cycle. The resulting architecture is more code efficient while achieving throughputs up to ten times faster than conventional CISC microcontrollers.

The ATmega48/88/168 provides the following features: 4K/8K/16K bytes of In-System Programmable Flash with Read-While-Write capabilities, 256/512/512 bytes EEPROM, 512/1K/1K bytes SRAM, 23 general purpose I/O lines, 32 general purpose working registers, three flexible Timer/Counters with compare modes, internal and external interrupts, a serial programmable USART, a byte-oriented 2-wire Serial Interface, an SPI serial port, a 6-channel 10-bit ADC (8 channels in TQFP and MLF packages), a programmable Watchdog Timer with internal Oscillator, and five software selectable power saving modes. The Idle mode stops the CPU while allowing the SRAM, Timer/Counters, USART, 2-wire Serial Interface, SPI port, and interrupt system to continue functioning. The Power-down mode saves the register contents but freezes the Oscillator, disabling all other chip functions until the next interrupt or hardware reset. In Power-save mode, the asynchronous timer continues to run, allowing the user to maintain a timer base while the rest of the device is sleeping. The ADC Noise Reduction mode stops the CPU and all I/O modules except asynchronous timer and ADC, to minimize switching noise during ADC conversions. In Standby mode, the crystal/resonator Oscillator is running while the rest of the device is sleeping. This allows very fast start-up combined with low power consumption.

The device is manufactured using Atmel's high density non-volatile memory technology. The On-chip ISP Flash allows the program memory to be reprogrammed In-System through an SPI serial interface, by a conventional non-volatile memory programmer, or by an On-chip Boot program running on the AVR core. The Boot program can use any interface to download the application program in the Application Flash memory. Software in the Boot Flash section will continue to run while the Application Flash section is updated, providing true Read-While-Write operation. By combining an 8-bit RISC CPU with In-System Self-Programmable Flash on a monolithic chip, the Atmel ATmega48/88/168 is a powerful microcontroller that provides a highly flexible and cost effective solution to many embedded control applications.

The ATmega48/88/168 AVR is supported with a full suite of program and system development tools including: C Compilers, Macro Assemblers, Program Debugger/Simulators, In-Circuit Emulators, and Evaluation kits.

Comparison Between ATmega48, ATmega88, and ATmega168

The ATmega48, ATmega88 and ATmega168 differ only in memory sizes, boot loader support, and interrupt vector sizes. Table 1 summarizes the different memory and interrupt vector sizes for the three devices.

Table 1. Memory Size Summary

Device	Flash	EEPROM	RAM	Interrupt Vector Size
ATmega48	4K Bytes	256 Bytes	512 Bytes	1 instruction word/vector
ATmega88	8K Bytes	512 Bytes	1K Bytes	1 instruction word/vector
ATmega168	16K Bytes	512 Bytes	1K Bytes	2 instruction words/vector

ATmega88 and ATmega168 support a real Read-While-Write Self-Programming mechanism. There is a separate Boot Loader Section, and the SPM instruction can only execute from there. In ATmega48, there is no Read-While-Write support and no separate Boot Loader Section. The SPM instruction can execute from the entire Flash.

Pin Descriptions

VCC — Digital supply voltage.

GND — Ground.

Port B (PB7..0) XTAL1/ XTAL2/TOSC1/TOSC2 — Port B is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port B output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port B pins that are externally pulled low will source current if the pull-up resistors are activated. The Port B pins are tri-stated when a reset condition becomes active, even if the clock is not running.

Depending on the clock selection fuse settings, PB6 can be used as input to the inverting Oscillator amplifier and input to the internal clock operating circuit.

Depending on the clock selection fuse settings, PB7 can be used as output from the inverting Oscillator amplifier.

If the Internal Calibrated RC Oscillator is used as chip clock source, PB7..6 is used as TOSC2..1 input for the Asynchronous Timer/Counter2 if the AS2 bit in ASSR is set.

The various special features of Port B are elaborated in “Alternate Functions of Port B” on page 69 and “System Clock and Clock Options” on page 24.

Port C (PC5..0) — Port C is a 7-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The PC5..0 output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port C pins that are externally pulled low will source current if the pull-up resistors are activated. The Port C pins are tri-stated when a reset condition becomes active, even if the clock is not running.

PC6/$\overline{\text{RESET}}$ — If the RSTDISBL Fuse is programmed, PC6 is used as an I/O pin. Note that the electrical characteristics of PC6 differ from those of the other pins of Port C.

If the RSTDISBL Fuse is unprogrammed, PC6 is used as a Reset input. A low level on this pin for longer than the minimum pulse length will generate a Reset, even if the clock is not running. The minimum pulse length is given in Table 20 on page 41. Shorter pulses are not guaranteed to generate a Reset.

The various special features of Port C are elaborated in “Alternate Functions of Port C” on page 73.

Port D (PD7..0) — Port D is an 8-bit bi-directional I/O port with internal pull-up resistors (selected for each bit). The Port D output buffers have symmetrical drive characteristics with both high sink and source capability. As inputs, Port D pins that are externally pulled low will source current if the pull-up resistors are activated. The Port D pins are tri-stated when a reset condition becomes active, even if the clock is not running.

The various special features of Port D are elaborated in “Alternate Functions of Port D” on page 75.

AVCC — AVCC is the supply voltage pin for the A/D Converter, PC3..0, and ADC7..6. It should be externally connected to V_{CC}, even if the ADC is not used. If the ADC is used, it should be connected to V_{CC} through a low-pass filter. Note that PC6..4 use digital supply voltage, V_{CC}.

AREF — AREF is the analog reference pin for the A/D Converter.

ADC7..6 (TQFP and MLF Package Only)

In the TQFP and MLF package, ADC7..6 serve as analog inputs to the A/D converter. These pins are powered from the analog supply and serve as 10-bit ADC channels.

Register Summary

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xFF)	Reserved	–	–	–	–	–	–	–	–	
(0xFE)	Reserved	–	–	–	–	–	–	–	–	
(0xFD)	Reserved	–	–	–	–	–	–	–	–	
(0xFC)	Reserved	–	–	–	–	–	–	–	–	
(0xFB)	Reserved	–	–	–	–	–	–	–	–	
(0xFA)	Reserved	–	–	–	–	–	–	–	–	
(0xF9)	Reserved	–	–	–	–	–	–	–	–	
(0xF8)	Reserved	–	–	–	–	–	–	–	–	
(0xF7)	Reserved	–	–	–	–	–	–	–	–	
(0xF6)	Reserved	–	–	–	–	–	–	–	–	
(0xF5)	Reserved	–	–	–	–	–	–	–	–	
(0xF4)	Reserved	–	–	–	–	–	–	–	–	
(0xF3)	Reserved	–	–	–	–	–	–	–	–	
(0xF2)	Reserved	–	–	–	–	–	–	–	–	
(0xF1)	Reserved	–	–	–	–	–	–	–	–	
(0xF0)	Reserved	–	–	–	–	–	–	–	–	
(0xEF)	Reserved	–	–	–	–	–	–	–	–	
(0xEE)	Reserved	–	–	–	–	–	–	–	–	
(0xED)	Reserved	–	–	–	–	–	–	–	–	
(0xEC)	Reserved	–	–	–	–	–	–	–	–	
(0xEB)	Reserved	–	–	–	–	–	–	–	–	
(0xEA)	Reserved	–	–	–	–	–	–	–	–	
(0xE9)	Reserved	–	–	–	–	–	–	–	–	
(0xE8)	Reserved	–	–	–	–	–	–	–	–	
(0xE7)	Reserved	–	–	–	–	–	–	–	–	
(0xE6)	Reserved	–	–	–	–	–	–	–	–	
(0xE5)	Reserved	–	–	–	–	–	–	–	–	
(0xE4)	Reserved	–	–	–	–	–	–	–	–	
(0xE3)	Reserved	–	–	–	–	–	–	–	–	
(0xE2)	Reserved	–	–	–	–	–	–	–	–	
(0xE1)	Reserved	–	–	–	–	–	–	–	–	
(0xE0)	Reserved	–	–	–	–	–	–	–	–	
(0xDF)	Reserved	–	–	–	–	–	–	–	–	
(0xDE)	Reserved	–	–	–	–	–	–	–	–	
(0xDD)	Reserved	–	–	–	–	–	–	–	–	
(0xDC)	Reserved	–	–	–	–	–	–	–	–	
(0xDB)	Reserved	–	–	–	–	–	–	–	–	
(0xDA)	Reserved	–	–	–	–	–	–	–	–	
(0xD9)	Reserved	–	–	–	–	–	–	–	–	
(0xD8)	Reserved	–	–	–	–	–	–	–	–	
(0xD7)	Reserved	–	–	–	–	–	–	–	–	
(0xD6)	Reserved	–	–	–	–	–	–	–	–	
(0xD5)	Reserved	–	–	–	–	–	–	–	–	
(0xD4)	Reserved	–	–	–	–	–	–	–	–	
(0xD3)	Reserved	–	–	–	–	–	–	–	–	
(0xD2)	Reserved	–	–	–	–	–	–	–	–	
(0xD1)	Reserved	–	–	–	–	–	–	–	–	
(0xD0)	Reserved	–	–	–	–	–	–	–	–	
(0xCF)	Reserved	–	–	–	–	–	–	–	–	
(0xCE)	Reserved	–	–	–	–	–	–	–	–	
(0xCD)	Reserved	–	–	–	–	–	–	–	–	
(0xCC)	Reserved	–	–	–	–	–	–	–	–	
(0xCB)	Reserved	–	–	–	–	–	–	–	–	
(0xCA)	Reserved	–	–	–	–	–	–	–	–	
(0xC9)	Reserved	–	–	–	–	–	–	–	–	
(0xC8)	Reserved	–	–	–	–	–	–	–	–	
(0xC7)	Reserved	–	–	–	–	–	–	–	–	
(0xC6)	UDR0	USART I/O Data Register								180
(0xC5)	UBRR0H					USART Baud Rate Register High				184
(0xC4)	UBRR0L	USART Baud Rate Register Low								184
(0xC3)	Reserved	–	–	–	–	–	–	–	–	
(0xC2)	UCSR0C	UMSEL01	UMSEL00	UPM01	UPM00	USBS0	UCSZ01 /UDORD0	UCSZ00 / UCPHA0	UCPOL0	183/196
(0xC1)	UCSR0B	RXCIE0	TXCIE0	UDRIE0	RXEN0	TXEN0	UCSZ02	RXB80	TXB80	182
(0xC0)	UCSR0A	RXC0	TXC0	UDRE0	FE0	DOR0	UPE0	U2X0	MPCM0	180

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0xBF)	Reserved	–	–	–	–	–	–	–	–	
(0xBE)	Reserved	–	–	–	–	–	–	–	–	
(0xBD)	TWAMR	TWAM6	TWAM5	TWAM4	TWAM3	TWAM2	TWAM1	TWAM0	–	209
(0xBC)	TWCR	TWINT	TWEA	TWSTA	TWSTO	TWWC	TWEN	–	TWIE	206
(0xBB)	TWDR	2-wire Serial Interface Data Register								208
(0xBA)	TWAR	TWA6	TWA5	TWA4	TWA3	TWA2	TWA1	TWA0	TWGCE	208
(0xB9)	TWSR	TWS7	TWS6	TWS5	TWS4	TWS3	–	TWPS1	TWPS0	207
(0xB8)	TWBR	2-wire Serial Interface Bit Rate Register								206
(0xB7)	Reserved	–		–	–	–	–	–	–	
(0xB6)	ASSR	–	EXCLK	AS2	TCN2UB	OCR2AUB	OCR2BUB	TCR2AUB	TCR2BUB	148
(0xB5)	Reserved	–	–	–	–	–	–	–	–	
(0xB4)	OCR2B	Timer/Counter2 Output Compare Register B								147
(0xB3)	OCR2A	Timer/Counter2 Output Compare Register A								147
(0xB2)	TCNT2	Timer/Counter2 (8-bit)								147
(0xB1)	TCCR2B	FOC2A	FOC2B	–	–	WGM22	CS22	CS21	CS20	146
(0xB0)	TCCR2A	COM2A1	COM2A0	COM2B1	COM2B0	–	–	WGM21	WGM20	143
(0xAF)	Reserved	–	–	–	–	–	–	–	–	
(0xAE)	Reserved	–	–	–	–	–	–	–	–	
(0xAD)	Reserved	–	–	–	–	–	–	–	–	
(0xAC)	Reserved	–	–	–	–	–	–	–	–	
(0xAB)	Reserved	–	–	–	–	–	–	–	–	
(0xAA)	Reserved	–	–	–	–	–	–	–	–	
(0xA9)	Reserved	–	–	–	–	–	–	–	–	
(0xA8)	Reserved	–	–	–	–	–	–	–	–	
(0xA7)	Reserved	–	–	–	–	–	–	–	–	
(0xA6)	Reserved	–	–	–	–	–	–	–	–	
(0xA5)	Reserved	–	–	–	–	–	–	–	–	
(0xA4)	Reserved	–	–	–	–	–	–	–	–	
(0xA3)	Reserved	–	–	–	–	–	–	–	–	
(0xA2)	Reserved	–	–	–	–	–	–	–	–	
(0xA1)	Reserved	–	–	–	–	–	–	–	–	
(0xA0)	Reserved	–	–	–	–	–	–	–	–	
(0x9F)	Reserved	–	–	–	–	–	–	–	–	
(0x9E)	Reserved	–	–	–	–	–	–	–	–	
(0x9D)	Reserved	–	–	–	–	–	–	–	–	
(0x9C)	Reserved	–	–	–	–	–	–	–	–	
(0x9B)	Reserved	–	–	–	–	–	–	–	–	
(0x9A)	Reserved	–	–	–	–	–	–	–	–	
(0x99)	Reserved	–	–	–	–	–	–	–	–	
(0x98)	Reserved	–	–	–	–	–	–	–	–	
(0x97)	Reserved	–	–	–	–	–	–	–	–	
(0x96)	Reserved	–	–	–	–	–	–	–	–	
(0x95)	Reserved	–	–	–	–	–	–	–	–	
(0x94)	Reserved	–	–	–	–	–	–	–	–	
(0x93)	Reserved	–	–	–	–	–	–	–	–	
(0x92)	Reserved	–	–	–	–	–	–	–	–	
(0x91)	Reserved	–	–	–	–	–	–	–	–	
(0x90)	Reserved	–	–	–	–	–	–	–	–	
(0x8F)	Reserved	–	–	–	–	–	–	–	–	
(0x8E)	Reserved	–	–	–	–	–	–	–	–	
(0x8D)	Reserved	–	–	–	–	–	–	–	–	
(0x8C)	Reserved	–	–	–	–	–	–	–	–	
(0x8B)	OCR1BH	Timer/Counter1 - Output Compare Register B High Byte								129
(0x8A)	OCR1BL	Timer/Counter1 - Output Compare Register B Low Byte								129
(0x89)	OCR1AH	Timer/Counter1 - Output Compare Register A High Byte								129
(0x88)	OCR1AL	Timer/Counter1 - Output Compare Register A Low Byte								129
(0x87)	ICR1H	Timer/Counter1 - Input Capture Register High Byte								129
(0x86)	ICR1L	Timer/Counter1 - Input Capture Register Low Byte								129
(0x85)	TCNT1H	Timer/Counter1 - Counter Register High Byte								129
(0x84)	TCNT1L	Timer/Counter1 - Counter Register Low Byte								129
(0x83)	Reserved	–	–	–	–	–	–	–	–	
(0x82)	TCCR1C	FOC1A	FOC1B	–	–	–	–	–	–	128
(0x81)	TCCR1B	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	127
(0x80)	TCCR1A	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	125
(0x7F)	DIDR1	–	–	–	–	–	–	AIN1D	AIN0D	230
(0x7E)	DIDR0	–	–	ADC5D	ADC4D	ADC3D	ADC2D	ADC1D	ADC0D	245

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
(0x7D)	Reserved	–	–	–	–	–	–	–	–	
(0x7C)	ADMUX	REFS1	REFS0	ADLAR	–	MUX3	MUX2	MUX1	MUX0	241
(0x7B)	ADCSRB	–	ACME	–	–	–	ADTS2	ADTS1	ADTS0	244
(0x7A)	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	242
(0x79)	ADCH	ADC Data Register High byte								244
(0x78)	ADCL	ADC Data Register Low byte								244
(0x77)	Reserved	–	–	–	–	–	–	–	–	
(0x76)	Reserved	–	–	–	–	–	–	–	–	
(0x75)	Reserved	–	–	–	–	–	–	–	–	
(0x74)	Reserved	–	–	–	–	–	–	–	–	
(0x73)	Reserved	–	–	–	–	–	–	–	–	
(0x72)	Reserved	–	–	–	–	–	–	–	–	
(0x71)	Reserved	–	–	–	–	–	–	–	–	
(0x70)	TIMSK2	–	–	–	–	–	OCIE2B	OCIE2A	TOIE2	150
(0x6F)	TIMSK1	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	130
(0x6E)	TIMSK0	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	100
(0x6D)	PCMSK2	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	83
(0x6C)	PCMSK1	–	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	83
(0x6B)	PCMSK0	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	84
(0x6A)	Reserved	–	–	–	–	–	–	–	–	
(0x69)	EICRA	–	–	–	–	ISC11	ISC10	ISC01	ISC00	80
(0x68)	PCICR	–	–	–	–	–	PCIE2	PCIE1	PCIE0	
(0x67)	Reserved	–	–	–	–	–	–	–	–	
(0x66)	OSCCAL	Oscillator Calibration Register								30
(0x65)	Reserved	–	–	–	–	–	–	–	–	
(0x64)	PRR	PRTWI	PRTIM2	PRTIM0	–	PRTIM1	PRSPI	PRUSART0	PRADC	37
(0x63)	Reserved	–	–	–	–	–	–	–	–	
(0x62)	Reserved	–	–	–	–	–	–	–	–	
(0x61)	CLKPR	CLKPCE	–	–	–	CLKPS3	CLKPS2	CLKPS1	CLKPS0	33
(0x60)	WDTCSR	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	49
0x3F (0x5F)	SREG	I	T	H	S	V	N	Z	C	9
0x3E (0x5E)	SPH	–	–	–	–	–	(SP10) [5.]	SP9	SP8	11
0x3D (0x5D)	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	11
0x3C (0x5C)	Reserved	–	–	–	–	–	–	–	–	
0x3B (0x5B)	Reserved	–	–	–	–	–	–	–	–	
0x3A (0x5A)	Reserved	–	–	–	–	–	–	–	–	
0x39 (0x59)	Reserved	–	–	–	–	–	–	–	–	
0x38 (0x58)	Reserved	–	–	–	–	–	–	–	–	
0x37 (0x57)	SPMCSR	SPMIE	(RWWSB)[5.]	–	(RWWSRE)[5.]	BLBSET	PGWRT	PGERS	SELFPRGEN	260
0x36 (0x56)	Reserved	–	–	–	–	–	–	–	–	
0x35 (0x55)	MCUCR	–	–	–	PUD	–	–	IVSEL	IVCE	
0x34 (0x54)	MCUSR	–	–	–	–	WDRF	BORF	EXTRF	PORF	
0x33 (0x53)	SMCR	–	–	–	–	SM2	SM1	SM0	SE	35
0x32 (0x52)	Reserved	–	–	–	–	–	–	–	–	
0x31 (0x51)	MONDR	Monitor Data Register								
0x30 (0x50)	ACSR	ACD	ACBG	ACO	ACI	ACIE	ACIC	ACIS1	ACIS0	228
0x2F (0x4F)	Reserved	–	–	–	–	–	–	–	–	
0x2E (0x4E)	SPDR	SPI Data Register								160
0x2D (0x4D)	SPSR	SPIF	WCOL	–	–	–	–	–	SPI2X	160
0x2C (0x4C)	SPCR	SPIE	SPE	DORD	MSTR	CPOL	CPHA	SPR1	SPR0	158
0x2B (0x4B)	GPIOR2	General Purpose I/O Register 2								23
0x2A (0x4A)	GPIOR1	General Purpose I/O Register 1								23
0x29 (0x49)	Reserved	–	–	–	–	–	–	–	–	
0x28 (0x48)	OCR0B	Timer/Counter0 Output Compare Register B								
0x27 (0x47)	OCR0A	Timer/Counter0 Output Compare Register A								
0x26 (0x46)	TCNT0	Timer/Counter0 (8-bit)								
0x25 (0x45)	TCCR0B	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	
0x24 (0x44)	TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	–	–	WGM01	WGM00	
0x23 (0x43)	GTCCR	TSM	–	–	–	–	–	PSRASY	PSRSYNC	103/152
0x22 (0x42)	EEARH	(EEPROM Address Register High Byte) [5.]								18
0x21 (0x41)	EEARL	EEPROM Address Register Low Byte								18
0x20 (0x40)	EEDR	EEPROM Data Register								18
0x1F (0x3F)	EECR	–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	18
0x1E (0x3E)	GPIOR0	General Purpose I/O Register 0								23
0x1D (0x3D)	EIMSK	–	–	–	–	–	–	INT1	INT0	81
0x1C (0x3C)	EIFR	–	–	–	–	–	–	INTF1	INTF0	82

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x1B (0x3B)	PCIFR	–	–	–	–	–	PCIF2	PCIF1	PCIF0	
0x1A (0x3A)	Reserved	–	–	–	–	–	–	–	–	
0x19 (0x39)	Reserved	–	–	–	–	–	–	–	–	
0x18 (0x38)	Reserved	–	–	–	–	–	–	–	–	
0x17 (0x37)	TIFR2	–	–	–	–	–	OCF2B	OCF2A	TOV2	151
0x16 (0x36)	TIFR1	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1	130
0x15 (0x35)	TIFR0	–	–	–	–	–	OCF0B	OCF0A	TOV0	
0x14 (0x34)	Reserved	–	–	–	–	–	–	–	–	
0x13 (0x33)	Reserved	–	–	–	–	–	–	–	–	
0x12 (0x32)	Reserved	–	–	–	–	–	–	–	–	
0x11 (0x31)	Reserved	–	–	–	–	–	–	–	–	
0x10 (0x30)	Reserved	–	–	–	–	–	–	–	–	
0x0F (0x2F)	Reserved	–	–	–	–	–	–	–	–	
0x0E (0x2E)	Reserved	–	–	–	–	–	–	–	–	
0x0D (0x2D)	Reserved	–	–	–	–	–	–	–	–	
0x0C (0x2C)	Reserved	–	–	–	–	–	–	–	–	
0x0B (0x2B)	PORTD	PORTD7	PORTD6	PORTD5	PORTD4	PORTD3	PORTD2	PORTD1	PORTD0	79
0x0A (0x2A)	DDRD	DDD7	DDD6	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0	79
0x09 (0x29)	PIND	PIND7	PIND6	PIND5	PIND4	PIND3	PIND2	PIND1	PIND0	79
0x08 (0x28)	PORTC	–	PORTC6	PORTC5	PORTC4	PORTC3	PORTC2	PORTC1	PORTC0	79
0x07 (0x27)	DDRC	–	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0	79
0x06 (0x26)	PINC	–	PINC6	PINC5	PINC4	PINC3	PINC2	PINC1	PINC0	79
0x05 (0x25)	PORTB	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	79
0x04 (0x24)	DDRB	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	79
0x03 (0x23)	PINB	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	79
0x02 (0x22)	Reserved	–	–	–	–	–	–	–	–	
0x01 (0x21)	Reserved	–	–	–	–	–	–	–	–	
0x0 (0x20)	Reserved	–	–	–	–	–	–	–	–	

Note: 1. For compatibility with future devices, reserved bits should be written to zero if accessed. Reserved I/O memory addresses should never be written.

2. I/O Registers within the address range 0x00 - 0x1F are directly bit-accessible using the SBI and CBI instructions. In these registers, the value of single bits can be checked by using the SBIS and SBIC instructions.
3. Some of the Status Flags are cleared by writing a logical one to them. Note that, unlike most other AVRs, the CBI and SBI instructions will only operate on the specified bit, and can therefore be used on registers containing such Status Flags. The CBI and SBI instructions work with registers 0x00 to 0x1F only.
4. When using the I/O specific commands IN and OUT, the I/O addresses 0x00 - 0x3F must be used. When addressing I/O Registers as data space using LD and ST instructions, 0x20 must be added to these addresses. The ATmega48/88/168 is a complex microcontroller with more peripheral units than can be supported within the 64 location reserved in Opcode for the IN and OUT instructions. For the Extended I/O space from 0x60 - 0xFF in SRAM, only the ST/STS/STD and LD/LDS/LDD instructions can be used.
5. Only valid for ATmega88/168

Instruction Set Summary

Mnemonics	Operands	Description	Operation	Flags	#Clocks
ARITHMETIC AND LOGIC INSTRUCTIONS					
ADD	Rd, Rr	Add two Registers	Rd ← Rd + Rr	Z,C,N,V,H	1
ADC	Rd, Rr	Add with Carry two Registers	Rd ← Rd + Rr + C	Z,C,N,V,H	1
ADIW	Rdl,K	Add Immediate to Word	Rdh:Rdl ← Rdh:Rdl + K	Z,C,N,V,S	2
SUB	Rd, Rr	Subtract two Registers	Rd ← Rd - Rr	Z,C,N,V,H	1
SUBI	Rd, K	Subtract Constant from Register	Rd ← Rd - K	Z,C,N,V,H	1
SBC	Rd, Rr	Subtract with Carry two Registers	Rd ← Rd - Rr - C	Z,C,N,V,H	1
SBCI	Rd, K	Subtract with Carry Constant from Reg.	Rd ← Rd - K - C	Z,C,N,V,H	1
SBIW	Rdl,K	Subtract Immediate from Word	Rdh:Rdl ← Rdh:Rdl - K	Z,C,N,V,S	2
AND	Rd, Rr	Logical AND Registers	Rd ← Rd • Rr	Z,N,V	1
ANDI	Rd, K	Logical AND Register and Constant	Rd ← Rd • K	Z,N,V	1
OR	Rd, Rr	Logical OR Registers	Rd ← Rd v Rr	Z,N,V	1
ORI	Rd, K	Logical OR Register and Constant	Rd ← Rd v K	Z,N,V	1
EOR	Rd, Rr	Exclusive OR Registers	Rd ← Rd ⊕ Rr	Z,N,V	1
COM	Rd	One's Complement	Rd ← 0xFF – Rd	Z,C,N,V	1
NEG	Rd	Two's Complement	Rd ← 0x00 – Rd	Z,C,N,V,H	1
SBR	Rd,K	Set Bit(s) in Register	Rd ← Rd v K	Z,N,V	1
CBR	Rd,K	Clear Bit(s) in Register	Rd ← Rd • (0xFF - K)	Z,N,V	1
INC	Rd	Increment	Rd ← Rd + 1	Z,N,V	1
DEC	Rd	Decrement	Rd ← Rd – 1	Z,N,V	1
TST	Rd	Test for Zero or Minus	Rd ← Rd • Rd	Z,N,V	1
CLR	Rd	Clear Register	Rd ← Rd ⊕ Rd	Z,N,V	1
SER	Rd	Set Register	Rd ← 0xFF	None	1
MUL	Rd, Rr	Multiply Unsigned	R1:R0 ← Rd x Rr	Z,C	2
MULS	Rd, Rr	Multiply Signed	R1:R0 ← Rd x Rr	Z,C	2
MULSU	Rd, Rr	Multiply Signed with Unsigned	R1:R0 ← Rd x Rr	Z,C	2
FMUL	Rd, Rr	Fractional Multiply Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULS	Rd, Rr	Fractional Multiply Signed	R1:R0 ← (Rd x Rr) << 1	Z,C	2
FMULSU	Rd, Rr	Fractional Multiply Signed with Unsigned	R1:R0 ← (Rd x Rr) << 1	Z,C	2
BRANCH INSTRUCTIONS					
RJMP	k	Relative Jump	PC ← PC + k + 1	None	2
IJMP		Indirect Jump to (Z)	PC ← Z	None	2
JMP[1]	k	Direct Jump	PC ← k	None	3
RCALL	k	Relative Subroutine Call	PC ← PC + k + 1	None	3
ICALL		Indirect Call to (Z)	PC ← Z	None	3
CALL[1]	k	Direct Subroutine Call	PC ← k	None	4
RET		Subroutine Return	PC ← STACK	None	4
RETI		Interrupt Return	PC ← STACK	I	4
CPSE	Rd,Rr	Compare, Skip if Equal	if (Rd = Rr) PC ← PC + 2 or 3	None	1/2/3
CP	Rd,Rr	Compare	Rd – Rr	Z, N,V,C,H	1
CPC	Rd,Rr	Compare with Carry	Rd – Rr – C	Z, N,V,C,H	1
CPI	Rd,K	Compare Register with Immediate	Rd – K	Z, N,V,C,H	1
SBRC	Rr, b	Skip if Bit in Register Cleared	if (Rr(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBRS	Rr, b	Skip if Bit in Register is Set	if (Rr(b)=1) PC ← PC + 2 or 3	None	1/2/3
SBIC	P, b	Skip if Bit in I/O Register Cleared	if (P(b)=0) PC ← PC + 2 or 3	None	1/2/3
SBIS	P, b	Skip if Bit in I/O Register is Set	if (P(b)=1) PC ← PC + 2 or 3	None	1/2/3
BRBS	s, k	Branch if Status Flag Set	if (SREG(s) = 1) then PC←PC+k + 1	None	1/2
BRBC	s, k	Branch if Status Flag Cleared	if (SREG(s) = 0) then PC←PC+k + 1	None	1/2
BREQ	k	Branch if Equal	if (Z = 1) then PC ← PC + k + 1	None	1/2
BRNE	k	Branch if Not Equal	if (Z = 0) then PC ← PC + k + 1	None	1/2
BRCS	k	Branch if Carry Set	if (C = 1) then PC ← PC + k + 1	None	1/2
BRCC	k	Branch if Carry Cleared	if (C = 0) then PC ← PC + k + 1	None	1/2
BRSH	k	Branch if Same or Higher	if (C = 0) then PC ← PC + k + 1	None	1/2
BRLO	k	Branch if Lower	if (C = 1) then PC ← PC + k + 1	None	1/2
BRMI	k	Branch if Minus	if (N = 1) then PC ← PC + k + 1	None	1/2
BRPL	k	Branch if Plus	if (N = 0) then PC ← PC + k + 1	None	1/2
BRGE	k	Branch if Greater or Equal, Signed	if (N ⊕ V= 0) then PC ← PC + k + 1	None	1/2
BRLT	k	Branch if Less Than Zero, Signed	if (N ⊕ V= 1) then PC ← PC + k + 1	None	1/2
BRHS	k	Branch if Half Carry Flag Set	if (H = 1) then PC ← PC + k + 1	None	1/2
BRHC	k	Branch if Half Carry Flag Cleared	if (H = 0) then PC ← PC + k + 1	None	1/2
BRTS	k	Branch if T Flag Set	if (T = 1) then PC ← PC + k + 1	None	1/2
BRTC	k	Branch if T Flag Cleared	if (T = 0) then PC ← PC + k + 1	None	1/2
BRVS	k	Branch if Overflow Flag is Set	if (V = 1) then PC ← PC + k + 1	None	1/2
BRVC	k	Branch if Overflow Flag is Cleared	if (V = 0) then PC ← PC + k + 1	None	1/2

Mnemonics	Operands	Description	Operation	Flags	#Clocks
BRIE	k	Branch if Interrupt Enabled	if (I = 1) then PC ← PC + k + 1	None	1/2
BRID	k	Branch if Interrupt Disabled	if (I = 0) then PC ← PC + k + 1	None	1/2
BIT AND BIT-TEST INSTRUCTIONS					
SBI	P,b	Set Bit in I/O Register	I/O(P,b) ← 1	None	2
CBI	P,b	Clear Bit in I/O Register	I/O(P,b) ← 0	None	2
LSL	Rd	Logical Shift Left	Rd(n+1) ← Rd(n), Rd(0) ← 0	Z,C,N,V	1
LSR	Rd	Logical Shift Right	Rd(n) ← Rd(n+1), Rd(7) ← 0	Z,C,N,V	1
ROL	Rd	Rotate Left Through Carry	Rd(0)←C,Rd(n+1)← Rd(n),C←Rd(7)	Z,C,N,V	1
ROR	Rd	Rotate Right Through Carry	Rd(7)←C,Rd(n)← Rd(n+1),C←Rd(0)	Z,C,N,V	1
ASR	Rd	Arithmetic Shift Right	Rd(n) ← Rd(n+1), n=0..6	Z,C,N,V	1
SWAP	Rd	Swap Nibbles	Rd(3..0)←Rd(7..4),Rd(7..4)←Rd(3..0)	None	1
BSET	s	Flag Set	SREG(s) ← 1	SREG(s)	1
BCLR	s	Flag Clear	SREG(s) ← 0	SREG(s)	1
BST	Rr, b	Bit Store from Register to T	T ← Rr(b)	T	1
BLD	Rd, b	Bit load from T to Register	Rd(b) ← T	None	1
SEC		Set Carry	C ← 1	C	1
CLC		Clear Carry	C ← 0	C	1
SEN		Set Negative Flag	N ← 1	N	1
CLN		Clear Negative Flag	N ← 0	N	1
SEZ		Set Zero Flag	Z ← 1	Z	1
CLZ		Clear Zero Flag	Z ← 0	Z	1
SEI		Global Interrupt Enable	I ← 1	I	1
CLI		Global Interrupt Disable	I ← 0	I	1
SES		Set Signed Test Flag	S ← 1	S	1
CLS		Clear Signed Test Flag	S ← 0	S	1
SEV		Set Twos Complement Overflow.	V ← 1	V	1
CLV		Clear Twos Complement Overflow	V ← 0	V	1
SET		Set T in SREG	T ← 1	T	1
CLT		Clear T in SREG	T ← 0	T	1
SEH		Set Half Carry Flag in SREG	H ← 1	H	1
CLH		Clear Half Carry Flag in SREG	H ← 0	H	1
DATA TRANSFER INSTRUCTIONS					
MOV	Rd, Rr	Move Between Registers	Rd ← Rr	None	1
MOVW	Rd, Rr	Copy Register Word	Rd+1:Rd ← Rr+1:Rr	None	1
LDI	Rd, K	Load Immediate	Rd ← K	None	1
LD	Rd, X	Load Indirect	Rd ← (X)	None	2
LD	Rd, X+	Load Indirect and Post-Inc.	Rd ← (X), X ← X + 1	None	2
LD	Rd, - X	Load Indirect and Pre-Dec.	X ← X - 1, Rd ← (X)	None	2
LD	Rd, Y	Load Indirect	Rd ← (Y)	None	2
LD	Rd, Y+	Load Indirect and Post-Inc.	Rd ← (Y), Y ← Y + 1	None	2
LD	Rd, - Y	Load Indirect and Pre-Dec.	Y ← Y - 1, Rd ← (Y)	None	2
LDD	Rd,Y+q	Load Indirect with Displacement	Rd ← (Y + q)	None	2
LD	Rd, Z	Load Indirect	Rd ← (Z)	None	2
LD	Rd, Z+	Load Indirect and Post-Inc.	Rd ← (Z), Z ← Z+1	None	2
LD	Rd, -Z	Load Indirect and Pre-Dec.	Z ← Z - 1, Rd ← (Z)	None	2
LDD	Rd, Z+q	Load Indirect with Displacement	Rd ← (Z + q)	None	2
LDS	Rd, k	Load Direct from SRAM	Rd ← (k)	None	2
ST	X, Rr	Store Indirect	(X) ← Rr	None	2
ST	X+, Rr	Store Indirect and Post-Inc.	(X) ← Rr, X ← X + 1	None	2
ST	- X, Rr	Store Indirect and Pre-Dec.	X ← X - 1, (X) ← Rr	None	2
ST	Y, Rr	Store Indirect	(Y) ← Rr	None	2
ST	Y+, Rr	Store Indirect and Post-Inc.	(Y) ← Rr, Y ← Y + 1	None	2
ST	- Y, Rr	Store Indirect and Pre-Dec.	Y ← Y - 1, (Y) ← Rr	None	2
STD	Y+q,Rr	Store Indirect with Displacement	(Y + q) ← Rr	None	2
ST	Z, Rr	Store Indirect	(Z) ← Rr	None	2
ST	Z+, Rr	Store Indirect and Post-Inc.	(Z) ← Rr, Z ← Z + 1	None	2
ST	-Z, Rr	Store Indirect and Pre-Dec.	Z ← Z - 1, (Z) ← Rr	None	2
STD	Z+q,Rr	Store Indirect with Displacement	(Z + q) ← Rr	None	2
STS	k, Rr	Store Direct to SRAM	(k) ← Rr	None	2
LPM		Load Program Memory	R0 ← (Z)	None	3
LPM	Rd, Z	Load Program Memory	Rd ← (Z)	None	3
LPM	Rd, Z+	Load Program Memory and Post-Inc	Rd ← (Z), Z ← Z+1	None	3
SPM		Store Program Memory	(Z) ← R1:R0	None	-
IN	Rd, P	In Port	Rd ← P	None	1
OUT	P, Rr	Out Port	P ← Rr	None	1
PUSH	Rr	Push Register on Stack	STACK ← Rr	None	2

Mnemonics	Operands	Description	Operation	Flags	#Clocks
POP	Rd	Pop Register from Stack	Rd ← STACK	None	2
MCU CONTROL INSTRUCTIONS					
NOP		No Operation		None	1
SLEEP		Sleep	(see specific descr. for Sleep function)	None	1
WDR		Watchdog Reset	(see specific descr. for WDR/timer)	None	1
BREAK		Break	For On-chip Debug Only	None	N/A

Note: 1. These instructions are only available in ATmega168.

Ordering Information

ATmega48

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
12[3]	1.8 - 5.5	ATmega48V-12AI ATmega48V-12PI ATmega48V-12MI ATmega48V-12AJ[2] ATmega48V-12PJ[2] ATmega48V-12MJ[2]	32A 28P3 32M1-A 32A 28P3 32M1-A	Industrial (-40°C to 85°C)
24[3]	2.7 - 5.5	ATmega48-24AI ATmega48-24PI ATmega48-24MI ATmega48-24AJ[2] ATmega48-24PJ[2] ATmega48-24MJ[2]	32A 28P3 32M1-A 32A 28P3 32M1-A	Industrial (-40°C to 85°C)

Note: 1. This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.

2. Pb-free packaging alternative
3. See Figure 131 on page 293 and Figure 132 on page 293.

Package Type	
32A	32-lead, Thin (1.0 mm) Plastic Quad Flat Package (TQFP)
28P3	28-lead, 0.300” Wide, Plastic Dual Inline Package (PDIP)
32M1-A	32-pad, 5 x 5 x 1.0 body, Lead Pitch 0.50 mm Micro Lead Frame Package (MLF)

ATmega88

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
12[3]	1.8 - 5.5	ATmega88V-12AI ATmega88V-12PI ATmega88V-12MI ATmega88V-12AJ[2] ATmega88V-12PJ[2] ATmega88V-12MJ[2]	32A 28P3 32M1-A 32A 28P3 32M1-A	Industrial (-40°C to 85°C)
24[3]	2.7 - 5.5	ATmega88-24AI ATmega88-24PI ATmega88-24MI ATmega88-24AJ[2] ATmega88-24PJ[2] ATmega88-24MJ[2]	32A 28P3 32M1-A 32A 28P3 32M1-A	Industrial (-40°C to 85°C)

Note: 1. This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.
2. Pb-free packaging alternative
3. See Figure 131 on page 293 and Figure 132 on page 293.

Package Type	
32A	32-lead, Thin (1.0 mm) Plastic Quad Flat Package (TQFP)
28P3	28-lead, 0.300” Wide, Plastic Dual Inline Package (PDIP)
32M1-A	32-pad, 5 x 5 x 1.0 body, Lead Pitch 0.50 mm Micro Lead Frame Package (MLF)

ATmega168

Speed (MHz)	Power Supply	Ordering Code	Package	Operation Range
12[3]	1.8 - 5.5	ATmega168V-12AI ATmega168V-12PI ATmega168V-12MI ATmega168V-12AJ[2] ATmega168V-12PJ[2] ATmega168V-12MJ[2]	32A 28P3 32M1-A 32A 28P3 32M1-A	Industrial (-40°C to 85°C)
24[3]	2.7 - 5.5	ATmega168-24AI ATmega168-24PI ATmega168-24MI ATmega168-24AJ[2] ATmega168-24PJ[2] ATmega168-24MJ[2]	32A 28P3 32M1-A 32A 28P3 32M1-A	Industrial (-40°C to 85°C)

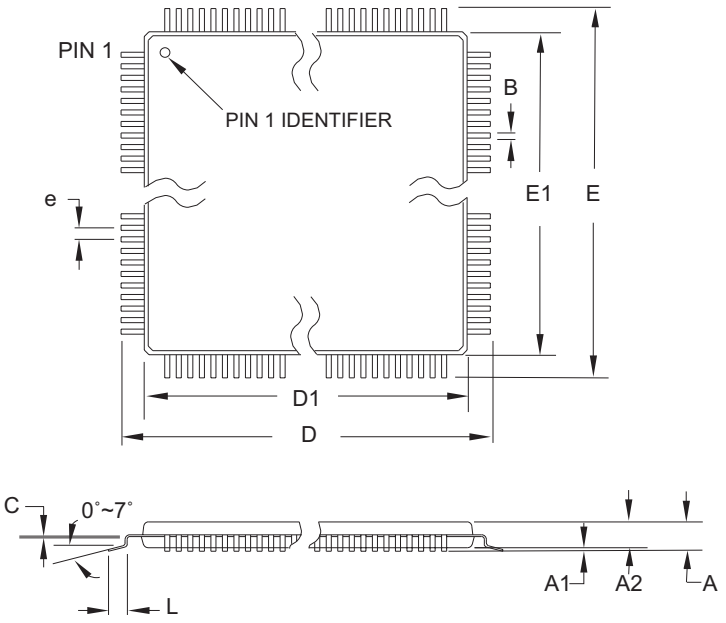
Note: 1. This device can also be supplied in wafer form. Please contact your local Atmel sales office for detailed ordering information and minimum quantities.

2. Pb-free packaging alternative
3. See Figure 131 on page 293 and Figure 132 on page 293.

Package Type	
32A	32-lead, Thin (1.0 mm) Plastic Quad Flat Package (TQFP)
28P3	28-lead, 0.300” Wide, Plastic Dual Inline Package (PDIP)
32M1-A	32-pad, 5 x 5 x 1.0 body, Lead Pitch 0.50 mm Micro Lead Frame Package (MLF)

Packaging Information

32A



COMMON DIMENSIONS
(Unit of Measure = mm)

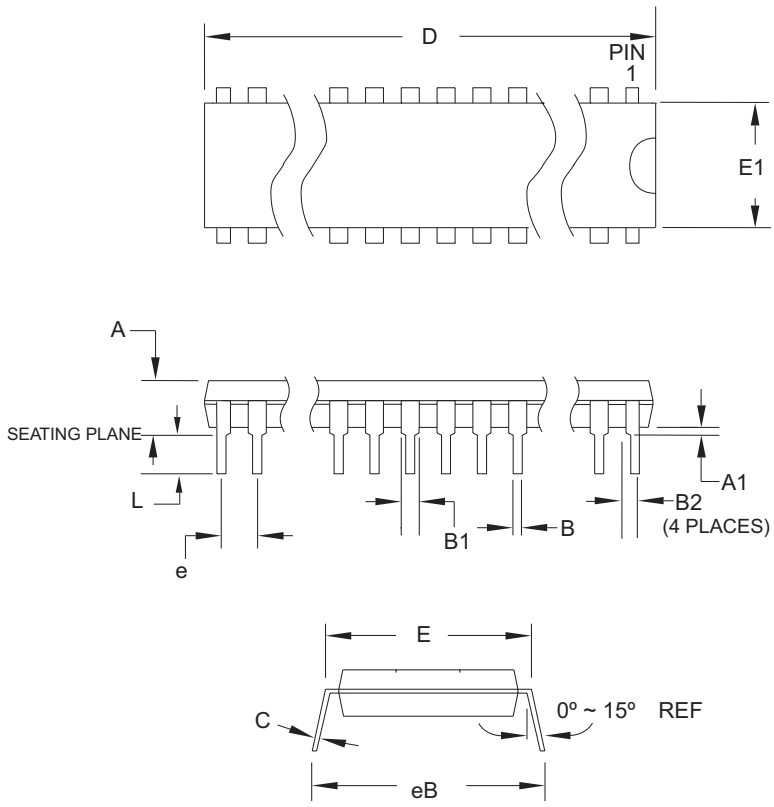
SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	1.20	
A1	0.05	–	0.15	
A2	0.95	1.00	1.05	
D	8.75	9.00	9.25	
D1	6.90	7.00	7.10	Note 2
E	8.75	9.00	9.25	
E1	6.90	7.00	7.10	Note 2
B	0.30	–	0.45	
C	0.09	–	0.20	
L	0.45	–	0.75	
e	0.80 TYP			

Notes:
1. This package conforms to JEDEC reference MS-026, Variation ABA.
2. Dimensions D1 and E1 do not include mold protrusion. Allowable protrusion is 0.25 mm per side. Dimensions D1 and E1 are maximum plastic body size dimensions including mold mismatch.
3. Lead coplanarity is 0.10 mm maximum.

10/5/2001

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**32A,** 32-lead, 7 x 7 mm Body Size, 1.0 mm Body Thickness, 0.8 mm Lead Pitch, Thin Profile Plastic Quad Flat Package (TQFP)	32A	B

28P3



COMMON DIMENSIONS
(Unit of Measure = mm)

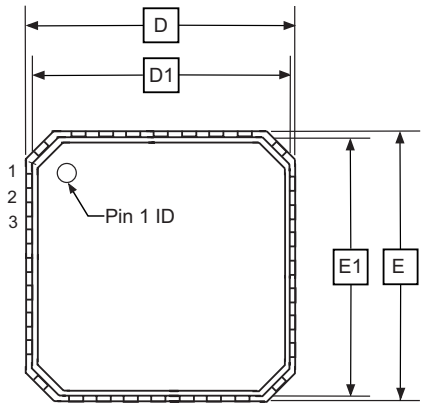
SYMBOL	MIN	NOM	MAX	NOTE
A	–	–	4.5724	
A1	0.508	–	–	
D	34.544	–	34.798	Note 1
E	7.620	–	8.255	
E1	7.112	–	7.493	Note 1
B	0.381	–	0.533	
B1	1.143	–	1.397	
B2	0.762	–	1.143	
L	3.175	–	3.429	
C	0.203	–	0.356	
eB	–	–	10.160	
e	2.540 TYP			

Note: 1. Dimensions D and E1 do not include mold Flash or Protrusion. Mold Flash or Protrusion shall not exceed 0.25 mm (0.010").

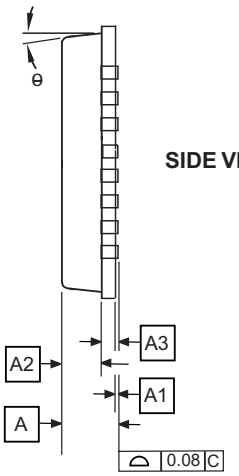
09/28/01

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**28P3**, 28-lead (0.300"/7.62 mm Wide) Plastic Dual Inline Package (PDIP)	28P3	B

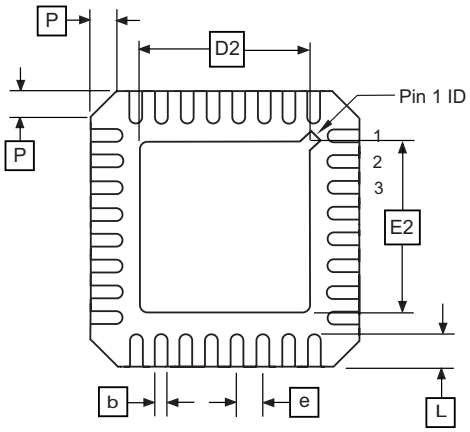
32M1-A



TOP VIEW



SIDE VIEW



BOTTOM VIEW

COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	0.80	0.90	1.00	
A1	–	0.02	0.05	
A2	–	0.65	1.00	
A3		0.20 REF		
b	0.18	0.23	0.30	
D		5.00 BSC		
D1		4.75 BSC		
D2	2.95	3.10	3.25	
E		5.00 BSC		
E1		4.75BSC		
E2	2.95	3.10	3.25	
e		0.50 BSC		
L	0.30	0.40	0.50	
P	–	–	0.60	
θ	–	–	12^{o}	

Notes: 1. JEDEC Standard MO-220, Fig. 2 (Anvil Singulation), VHHD-2.

01/15/03

	TITLE	DRAWING NO.	REV.
2325 Orchard Parkway San Jose, CA 95131	**32M1-A**, 32-pad, 5 x 5 x 1.0 mm Body, Lead Pitch 0.50 mm Micro Lead Frame Package (MLF)	32M1-A	C

Errata ATmega48

The revision letter in this section refers to the revision of the ATmega48 device.

Rev A

- **Wrong values read after Erase Only operation**
- **Watchdog Timer Interrupt disabled**
- **Start-up time with Crystal Oscillator is higher than expected**
- **High Power Consumption in Power-down with External Clock**
- **Asynchronous Oscillator does not stop in Power-down**

1. **Wrong values read after Erase Only operation**

   At supply voltages below 2.7 V, an EEPROM location that is erased by the Erase Only operation may read as programmed (0x00).

   Problem Fix/Workaround

   If it is necessary to read an EEPROM location after Erase Only, use an Atomic Write operation with 0xFF as data in order to erase a location. In any case, the Write Only operation can be used as intended. Thus no special considerations are needed as long as the erased location is not read before it is programmed.

2. **Watchdog Timer Interrupt disabled**

   If the watchdog timer interrupt flag is not cleared before a new timeout occurs, the watchdog will be disabled, and the interrupt flag will automatically be cleared. This is only applicable in interrupt only mode. If the Watchdog is configured to reset the device in the watchdog time-out following an interrupt, the device works correctly.

   Problem fix / Workaround

   Make sure there is enough time to always service the first timeout event before a new watchdog timeout occurs. This is done by selecting a long enough time-out period.

3. **Start-up time with Crystal Oscillator is higher than expected**

   The clock counting part of the start-up time is about 2 times higher than expected for all start-up periods when running on an external Crystal. This applies only when waking up by reset. Wake-up from power down is not affected. For most settings, the clock counting parts is a small fraction of the overall start-up time, and thus, the problem can be ignored. The exception is when using a very low frequency crystal like for instance a 32 kHz clock crystal.

   Problem fix / Workaround

   No known workaround.

4. **High Power Consumption in Power-down with External Clock**

   The power consumption in power down with an active external clock is about 10 times higher than when using internal RC or external oscillators.

   Problem fix / Workaround

   Stop the external clock when the device is in power down.

5. **Asynchronous Oscillator does not stop in Power-down**

   The Asynchronous oscillator does not stop when entering power down mode. This leads to higher power consumption than expected.

   Problem fix / Workaround

   Manually disable the asynchronous timer before entering power down.

Errata ATmega88

The revision letter in this section refers to the revision of the ATmega88 device.

Rev A

- **Wrong values read after Erase Only operation**

1. Wrong values read after Erase Only operation

At supply voltages below 2.7 V, an EEPROM location that is erased by the Erase Only operation may read as programmed (0x00).

Problem Fix/Workaround

If it is necessary to read an EEPROM location after Erase Only, use an Atomic Write operation with 0xFF as data in order to erase a location. In any case, the Write Only operation can be used as intended. Thus no special considerations are needed as long as the erased location is not read before it is programmed.

Errata ATmega168

The revision letter in this section refers to the revision of the ATmega168 device.

Rev A

- **Wrong values read after Erase Only operation**

1. **Wrong values read after Erase Only operation**

   At supply voltages below 2.7 V, an EEPROM location that is erased by the Erase Only operation may read as programmed (0x00).

   Problem Fix/Workaround

   If it is necessary to read an EEPROM location after Erase Only, use an Atomic Write operation with 0xFF as data in order to erase a location. In any case, the Write Only operation can be used as intended. Thus no special considerations are needed as long as the erased location is not read before it is programmed.

Datasheet Change Log

Please note that the referring page numbers in this section are referred to this document. The referring revision in this section are referring to the document revision.

Changes from Rev. 2545A-09/03 to Rev. 2545B-01/04

1. **Added PDIP to “I/O and Packages”, updated “Speed Grade” and Power Consumption Estimates in “Features” on page 1.**
2. **Updated “Stack Pointer” on page 11 with RAMEND as recommended Stack Pointer value.**
3. **Added section “Power Reduction Register” on page 37 and a note regarding the use of the PRR bits to 2-wire, Timer/Counters, USART, Analog Comparator and ADC sections.**
4. **Updated “Watchdog Timer” on page 46.**
5. **Updated Figure 55 on page 125 and Table 56 on page 126.**
6. **Extra Compare Match Interrupt OCF2B added to features in section “8-bit Timer/Counter2 with PWM and Asynchronous Operation” on page 132**
7. **Updated Table 19 on page 37, Table 102 on page 245, Table 118 to Table 121 on page 272 to 273 and Table 98 on page 236. Added note 2 to Table 115 on page 270. Fixed typo in Table 42 on page 81.**
8. **Updated whole “ATmega48/88/168 Typical Characteristics – Preliminary Data” on page 298.**
9. **Added item 2 to 5 in “Errata ATmega48” on page 20.**
10. **Renamed the following bits:**
    - SPMEN to SELFPRGEN,
    - PSR2 to PSRASY
    - PSR10 to PSRSYNC
    - Watchdog Reset to Watchdog System Reset.
11. **Updated C code examples containing old IAR syntax.**
12. **Updated BLBSET description in “Store Program Memory Control and Status Register – SPMCSR” on page 260.**

Atmel Corporation

2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 487-2600

Regional Headquarters

Europe
Atmel Sarl
Route des Arsenaux 41
Case Postale 80
CH-1705 Fribourg
Switzerland
Tel: (41) 26-426-5555
Fax: (41) 26-426-5500

Asia
Room 1219
Chinachem Golden Plaza
77 Mody Road Tsimshatsui
East Kowloon
Hong Kong
Tel: (852) 2721-9778
Fax: (852) 2722-1369

Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
Japan
Tel: (81) 3-3523-3551
Fax: (81) 3-3523-7581

Atmel Operations

Memory
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

Microcontrollers
2325 Orchard Parkway
San Jose, CA 95131, USA
Tel: 1(408) 441-0311
Fax: 1(408) 436-4314

La Chantrerie
BP 70602
44306 Nantes Cedex 3, France
Tel: (33) 2-40-18-18-18
Fax: (33) 2-40-18-19-60

ASIC/ASSP/Smart Cards
Zone Industrielle
13106 Rousset Cedex, France
Tel: (33) 4-42-53-60-00
Fax: (33) 4-42-53-60-01

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Scottish Enterprise Technology Park
Maxwell Building
East Kilbride G75 0QR, Scotland
Tel: (44) 1355-803-000
Fax: (44) 1355-242-743

RF/Automotive
Theresienstrasse 2
Postfach 3535
74025 Heilbronn, Germany
Tel: (49) 71-31-67-0
Fax: (49) 71-31-67-2340

1150 East Cheyenne Mtn. Blvd.
Colorado Springs, CO 80906, USA
Tel: 1(719) 576-3300
Fax: 1(719) 540-1759

Biometrics/Imaging/Hi-Rel MPU/
High Speed Converters/RF Datacom
Avenue de Rochepleine
BP 123
38521 Saint-Egreve Cedex, France
Tel: (33) 4-76-58-30-00
Fax: (33) 4-76-58-34-80

Literature Requests
www.atmel.com/literature

2545BS–AVR–01/04

ESCUELA DE FORMACION DE TECNOLOGOS
CARRERA DE: ELECTRONICA Y TELECOMUNICACIONES

ORDEN DE EMPASTADO

De acuerdo con lo estipulado en el artículo 83 del Reglamento del Sistema de Estudios de las Carreras de Formación Profesional y de Postgrados aprobado por el Consejo Politécnico en sesión del 16 de agosto de 2011 y una vez verificado el cumplimiento del formato de presentación establecido, se autoriza la impresión y encuadernación final del Proyecto de Titulación presentado por los señores:

DARIO JAVIER SOLIS ALULIMA
JOSE LUIS MALDONADO RODRIGUEZ

Fecha de autorización: Quito DM 18 de enero de 2012

Ingeniero Carlos Posso Játiva
DIRECTOR DE LA ESCUELA DE FORMACION DE TECNOLOGOS