ESCUELA SUPERIOR POLITECNICA DEL LITORAL

Facultad de Ingeniería en Electricidad y Computación

"DISEÑO E IMPLEMENTACION DE UNA TARJETA INTELIGENTE BASADA EN MICROCONTROLADORES, REPROGRAMABLE DESDE UN PC POR MEDIO DE UN SOFTWARE HECHO EN VISUAL BASIC."

TESIS DE GRADO

Previo a la obtención del título de:
INGENIERO EN ELECTRICIDAD
Especialización Electrónica

Presentado por:

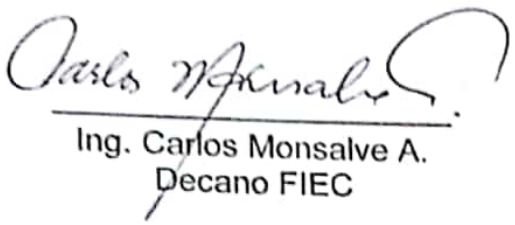
Ronald Alberto Ponguillo Intriago

Guayaquil-Ecuador
2003

AGRADECIMIENTOS

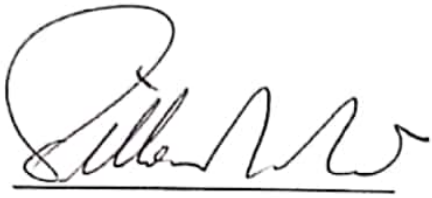
La mayor de las gratitudes para mi familia, quienes le apostaron todo a la vida por mí, y han sido mi sostén en todo momento.

Un agradecimiento especial para la Ingeniera Ludmila Gorenkova, quien ayudó de forma desinteresada para terminar este proyecto.

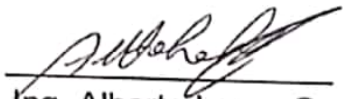


Ing. Carlos Monsalve A.
Decano FIEC

Ing. Ludmila Gorenkova L.
Directora de TESIS



Ing. Hugo Villavicencio V.
Miembro del Tribunal



Ing. Alberto Larco G.
Miembro del Tribunal

DECLARACION EXPRESA

"La responsabilidad por los hechos, ideas y doctrinas expuestos en esta tesis, me corresponden exclusivamente; y, el patrimonio intelectual de la misma, a la ESCUELA SUPERIOR POLITECNICA DEL LITORAL"

(Reglamentos de Exámenes y Títulos profesionales de la ESPOL)

Ronald Alberto Ponguillo Intriago

RESUMEN

El presente trabajo tiene como enfoque principal servir de apoyo para los diseñadores de proyectos con microcontroladores en la facultad, considerando que estos dispositivos controladores están siendo introducidos de a poco entre los profesores y estudiantes, de manera que en el mismo intento mostrar como se puede diseñar elementos de automatización basadas en herramientas de uso sencillo como un entorno de programación visual y microcontroladores.

El informe ha sido redactado en cinco capítulos, en los que se detallan las bases del diseño, considerando los aspectos del análisis del problema así como la solución propuesta, avanzando por la etapa de construcción, donde se muestran los detalles del diseño de cada parte en particular. En el capítulo 1 se hace una introducción a lo que es el proyecto, mostrando el esquema de funcionamiento, basado en diagrama de bloques donde se detalla las características y el funcionamiento de cada etapa, además se plantean varias propuestas para la resolución del mismo y se justifica la elección hecha. El capítulo 2 ya nos introduce en el diseño en si de la parte del hardware, donde se hacen las consideraciones para el diseño de la tarjeta, como también se establecen las características que esta deberá cumplir, además se establecen los criterios para la selección del microcontrolador y todos los detalles se resumen en un diagrama esquemático. Se plantea también en este capitulo el uso de los recursos internos del microcontrolador que han sido utilizados, y los requerimiento que debe cumplir la fuente de alimentación. El diseño del firmware es lo que se detalla en el capítulo 3, haciendo una exposición general de los requerimientos del programa de control, para luego pasar al detalle de cada subrutina que componen al mismo. El diseño del software se lo analiza en el capítulo 4, en este se hace la descripción de las herramientas de software adicionales que se

emplearon y se justifica la elección del paquete de programación que se empleó. Posteriormente en el capítulo 5 se encuentra material fotográfico que muestra el trabajo final, también se muestran algunos resultados de la simulación del microcontrolador y de las pruebas hechas al software, concluyendo con la presentación de la lista de materiales usados y los costos de los mismos. Al final de este reporte se encontrará varios apéndices que dan algunos detalles de las herramientas utilizadas como el formato hexadecimal de Intel en el apéndice A, la tarjeta de circuito impreso en el apéndice B y las características del microcontrolador utilizado en el apéndice C, para dejar el manual del usuario en el apéndice D. Finalmente se muestra el glosario y la bibliografía que sirvió de consulta para este trabajo.

INDICE GENERAL

INDICE DE FIGURAS

INDICE DE TABLAS

INTRODUCCIÓN

Con el avance tecnológico han surgido variedades de circuitos integrados con capacidad de programación. Esta es la tendencia del mundo moderno en el que casi todo lo que existe es digital y casi todo lo digital es programable. Entre los circuitos integrados programables uno que se ha destacado en los últimos años por su flexibilidad a la hora de usarlo es el microcontrolador, ya que lleva integrado un sinnúmero de recursos que reduce el tiempo de trabajo para el diseñador y el tamaño de los prototipos y productos finales. Su antecesor, el microprocesador, ha tenido su auge en las computadoras personales que han evolucionando con una velocidad asombrosa. Por otro lado el desarrollo del microcontrolador se debe en gran parte a la necesidad de comodidad del ser humano, que día a día requiere de cosas que le faciliten la vida y que la tecnología le puede ofrecer gracias a los sistemas automáticos controlados electrónicamente. Para facilitar el desarrollo de estos sistemas automáticos la electrónica moderna diseña y construye dispositivos semiconductores con capacidad de programación y reprogramación junto con las herramientas necesarias para en conjunto elaborar una solución especifica.

Si tuviésemos que enumerar las diferentes aplicaciones que se puede dar a los microcontroladores nos tomaría mucho tiempo debido a que su campo de aplicación es bastante amplio y variado. Además como aquí si cabe mencionar que el límite lo pone la imaginación, solamente diré que casi en todas las casas hoy en día se encuentra microcontroladores, por ejemplo en los equipos de audio y video, en los productos de línea blanca, en los automóviles, en las puertas de garaje automáticas, en controles electrónicos de encendido apagado de bombas de agua, solo por citar algo.

Cuando se habla de microcontroladores es posible mencionar un sinnúmero de fabricantes como Intel, AMD, Motorola, Texas Instruments y no se puede dejar de mencionar a Microchip, que es una de las empresas que mas microcontroladores vende en el mundo anualmente, y su éxito lo debe en gran parte a que entrega gratuitamente toda la información necesaria para que las personas interesadas aprendan a utilizar sus productos. Además de dar mucha información con los detalles de los recursos internos y su programación, sus microcontroladores son muy fáciles de programar ya que basados en una arquitectura RISC poseen muy pocas instrucciones.

En la realidad de nuestro medio se hace cada vez imprescindible contar con elementos que automaticen ciertos procesos que nos generen además de cierta comodidad una reducción en el consumo de energía, como por ejemplo un sistema que controle el encendido y apagado de las luces, o el funcionamiento de una bomba que llene un tanque elevado. Estas son tareas de automatización que no demandan una gran inversión y en cambio nos podrían generar un gran beneficio.

En este proyecto de tesis se presenta un prototipo bastante versátil que intenta ser una alternativa para las situaciones antes mencionadas, en el mismo que se podrá generar diversas soluciones a problemas sencillos pero de mucha utilidad y que además podrán ser generadas por personas que no necesitarán ser unos peritos en la materia, sino, tener solo unas cuantas nociones básicas al respecto.

La verdad del caso es que no se puede hacer una comparación en cuanto a prestaciones entre esta tarjeta y un PLC, ya que el PLC es mucho mas completo, pero para el tipo de aplicaciones que se pueden desarrollar con la tarjeta presentada en ésta tesis, resulta mas conveniente usar esta última por las siguientes razones:

El PLC más pequeño y básico no deja de costar alrededor de cien dólares, mientras que el costo de armar la tarjeta basada en microcontroladores, como se puede ver en el capítulo 5 no sobrepasa los sesenta y cinco dólares.

Por otro lado el PLC posee su software para programación de las aplicaciones de usuario al igual que lo hace el software TESIS_RAPI con la tarjeta hecha con microcontrolador, pero la diferencia de costos es enorme, ya que el software TESIS_RAPI por ser parte de una tesis no tiene precio y en cambio un software para programar PLC podría estar costando hasta varios miles de dólares, inversión que no se justifica para desarrollar aplicaciones pequeñas como las planteadas aquí.

CAPITULO 1

DESCRIPCIÓN DEL SISTEMA

1.1 INTRODUCCIÓN

En este capítulo se hace un bosquejo de la estructura que debe tener el proyecto, es decir, aquí se dan las ideas y se establecen las condiciones de funcionamiento en las que se basa el prototipo presentado en este trabajo.

1.2 ESQUEMA DE FUNCIONAMIENTO

Básicamente lo que se quiere es un dispositivo que sea capaz de realizar procesos de encendido/apagado de acuerdo a una secuencia programada que puede estar en función de los estados que tengan las entradas en determinado momento. Además se debe tener la capacidad de cambiar el programa directamente desde una computadora personal sin ayuda de un programador especializado adicional.

Un programa de computadora debe realizar las tareas de edición, compilación y transferencia de datos de los programas de usuario hasta la tarjeta prototipo.

Dentro de las posibles aplicaciones que se dará a este proyecto podrían estar incluidos cualquier proceso de control que demande una cantidad de cuatro entradas digitales del tipo todo o nada, ocho salida con relés que pueden manejar una corriente nominal de 5 amperios, temporización de hasta 255 segundos con pasos de un segundo

1.2.1 DIAGRAMA DE BLOQUES

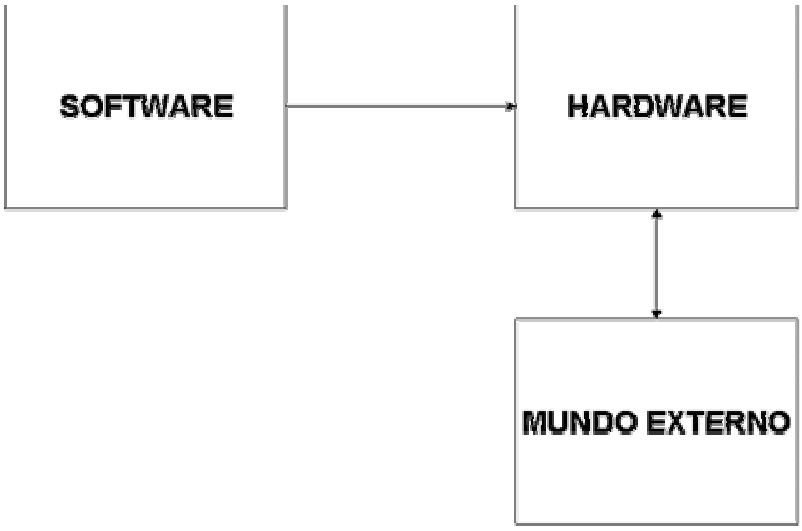


Figura 1-1 Diagrama de Bloques del Sistema

El diagrama de bloques mostrado representa cada una de las etapas que comprenden el proyecto, y como se puede ver consta de tres bloques que se detallan a continuación.

El bloque del software lo constituye un programa para PC que corre bajo el sistema operativo Windows y debe tener un ambiente visual agradable que mantenga muchas de las características de los programas que funcionan bajo Windows, como los menús comunes para abrir o guardar archivos que el usuario requiera para su trabajo, también una barra con las herramientas mas usadas y las opciones para salir del programa.

El programa deberá ser capaz de comunicarse con el hardware para transferir hasta el microcontrolador el código necesario para realizar la tarea que estime el usuario; estas tareas podrán ser codificadas dentro del editor que posee el software y compiladas en el mismo entorno, para ello se colocarán en el programa las opciones necesarias.

En lo que respecta al hardware debe implementarse el firmware de tal manera que posea funciones básicas como una rutina que le permita al microcontrolador comunicarse serialmente con la PC para poder recibir los datos que requieran ser grabados en la memoria de programa. Además debe tener la capacidad de hacer un control en tiempo real que le permita temporizar eventos, pero como las salidas son en base a relés no deberían implementarse temporizaciones muy pequeñas ya que debido a la inercia de los elementos mecánicos que éstos poseen su respuesta no puede ser tan rápida. Aunque los relés si responden a períodos menores a un segundo he considerado que la base de tiempo para las temporizaciones sea un segundo, debido a la facilidad en el manejo de las cantidades que se derivan de ésta y con esto también reducimos el ruido. De lo expuesto anteriormente se puede decir que se podrán tener temporizaciones que sean múltiplos de uno, pero solo hasta 255, debido a que el valor del temporizador debe ser cargado en un registro del microcontrolador y estos son de ocho bits, lo que no significa que el límite sea los 255 segundos ya que se pueden hacer los bucles que sean necesarios para aumentar este tiempo.

Otra de las características que se debe considerar a la hora de elaborar el proyecto es darle la capacidad de poder escribir en la memoria de programa, tomando en cuenta que no se deberá sobrescribir en la región donde residen las funciones del firmware, para no provocar conflictos en el funcionamiento.

En el hardware deberán colocarse también las respectivas interfaces para acoplar los niveles de voltaje RS232 de la PC con los niveles del microcontrolador, por otro lado, hay que establecer la circuitería para manejar cada relé en las salidas.

CAPITULO 2

DISEÑO DEL HARDWARE

2.1 INTRODUCCIÓN

Dentro del proceso de diseño del proyecto está una parte fundamental, que es el diseño del hardware que va a servir de controlador de los diferentes procesos que el usuario realice. Este diseño se lo ha dividido en varias secciones, en las cuales se analiza las características de cada una de las etapas, los criterios para la selección del microcontrolador a emplear y los recursos que serán utilizados de este. Se hace una explicación de los diferentes bloques que constituyen el hardware y se muestran los detalles del diseño en un diagrama esquemático del circuito.

2.2 CONSIDERACIONES PARA EL DISEÑO DE LA TARJETA

El objetivo de este proyecto es diseñar una tarjeta controladora basada en microcontroladores PIC, que se pueda reprogramar sin necesidad de poseer un programador especial de los tantos que se encuentran en el medio, sino que su programación la realice una PC y la subrutina para el caso implementada en el firmware del microcontrolador. Además deberá tener capacidad para manejar cuatro entradas y ocho salidas digitales, así como un temporizador de segundos reprogramable por el usuario. En las siguientes secciones y capítulos se establecerán las características en detalle de cada una de estas partes.

2.2.1 CARACTERÍSTICAS TÉCNICAS DE LA TARJETA

Durante el diseño de este proyecto era necesario establecer las capacidades y los límites del mismo de acuerdo al objetivo planteado. En esta sección se expondrán las características de cada una de las etapas concebidas en el mismo.

2.2.1.1 ENTRADAS

Son cuatro las entradas de la tarjeta ubicadas en el puerto A, que corresponden a los pines 2 a 5 que tienen las funciones RA0 hasta RA3 respectivamente. Cabe indicar que las entradas digitales son del tipo TON (Todo o Nada), que manejan 5V máximo y 0V como valor mínimo.

2.2.1.2 SALIDAS

Las salidas son manejadas por ocho relés de 12VDC y son controladas por el puerto B del microcontrolador. Como las salidas del puerto del PIC son digitales y manejan una corriente relativamente baja en comparación con la corriente de polarización de la bobina de los relés, es necesario colocar algún circuito para manejar los mismos. Típicamente estos circuitos se hacen con un transistor, un diodo y algunos resistores por cada relé, pero esto demandaría un buen espacio en la tarjeta de circuito impreso, así que pensé en usar un chip que tuviese estos elementos integrados como el ULN2803; además los relés tienen la capacidad de manejar 5 amperios nominales en sus contactos

2.2.1.3 ALIMENTACIÓN

En cuanto a la alimentación del circuito, esta se puede lograr con una fuente de DC con capacidad para entregar 12V y 1 amperio. Dentro de la tarjeta se baja el voltaje a 5VDC con un regulador de voltaje integrado como es el 7805, y este se aplica a la electrónica de control de la tarjeta.

2.2.2 CRITERIOS PARA LA SELECCIÓN DEL PIC

Para poder elegir el PIC que se iba a utilizar se debió tomar en consideración varios aspectos, basados en los requerimientos de recursos, consideraciones de precio versus prestaciones, disponibilidad en el mercado y proyecciones para futuras ampliaciones. A continuación hago un bosquejo de los principales criterios para esta selección:

1. Se necesitan 12 pines de E/S, 4 para entradas y 8 para salidas.
2. Requerimos 2 pines para comunicación serial con el PC, uno para transmisión y el otro para recepción.
3. Se debe tener la capacidad para generar temporización de un segundo manejada con interrupciones.
4. Es indispensable que el dispositivo tenga memoria de programa tipo FLASH o EEPROM, para poder grabar y borrar desde el software.
5. Debe poseer suficiente memoria para programa y datos

Me corresponde ahora detallar cada uno de los puntos expuestos arriba.

El primer punto se mencionó ya en 2.2.1.1, donde se explica que la tarjeta tiene cuatro entradas y ocho salidas, para ello necesito 12 pines.

Para poder comunicar la tarjeta con la PC serialmente a través del puerto RS232 de éste último, necesito bien sea emular su protocolo en el firmware del PIC o utilizar un dispositivo que establezca dicha comunicación serial. Este elemento es un USART (Universal Sinchronous Asynchronous Receiver Transmitter – Receptor Transmisor Sincrónico Asincrónico Universal), y es mas conveniente utilizar el recurso en hardware que emular su funcionamiento por dos razones fundamentales, la primera es que realizar estas funciones con código es más complicado y la segunda y mas importante es que dicho código crearía dependencia con las subrutinas a las que les da servicio y ello conlleva a consumo innecesario de ciclos de instrucción, lo que haría mas lento y menos eficiente el proceso.

En el tercer punto se analiza la capacidad de generar temporizaciones de un segundo lo que sería completamente ineficiente hacerlo a base de retardos usando bucles dentro del programa, además esto no tendría la precisión que puede darnos un temporizador integrado que se pueda manejar con interrupciones.

La base del proyecto es la capacidad que tiene este para reprogramarse sin necesidad de un hardware de programación, para lo cual el PIC seleccionado debe tener un tipo de memoria de programa que pueda ser escrito y borrado con un código implementado en la misma. Esta capacidad la ofrecen las memorias FLASH y EEPROM, esto es lo que se establece en el punto cuatro.
Por último en el punto cinco se plantea la necesidad de disponer de suficiente memoria para las aplicaciones del usuario

2.2.3 DIAGRAMA DE BLOQUES

En el siguiente diagrama de bloques se muestra cada una de las partes que constituyen el hardware del proyecto. Como bloque central está el

controlador que ha sido implementado con el PIC16F877, en el que se han incluido subrutinas para el manejo de las entradas digitales, otra para el manejo de la comunicación serial con la PC. Se incluye también el código que permite manipular las salidas y alguna rutina que da la opción de producir temporizaciones desde un segundo hasta 255 segundos, para aplicaciones en las que se requiera manejar intervalos de tiempo.

El bloque Acoplador de Señales corresponde a cuatro entradas digitales implementadas sobre el puerto A del microcontrolador. La comunicación con la PC se logra a través del USART integrado en el PIC y un circuito integrado MAX232 debido a que las señales del microcontrolador tienen valores entre cero y cinco voltios, y no son compatibles con los niveles que maneja la PC. Para manejar las salidas de la tarjeta se ha dispuesto de un circuito integrado ULN2803, que contiene ocho drivers para manejar el banco de ocho relés.

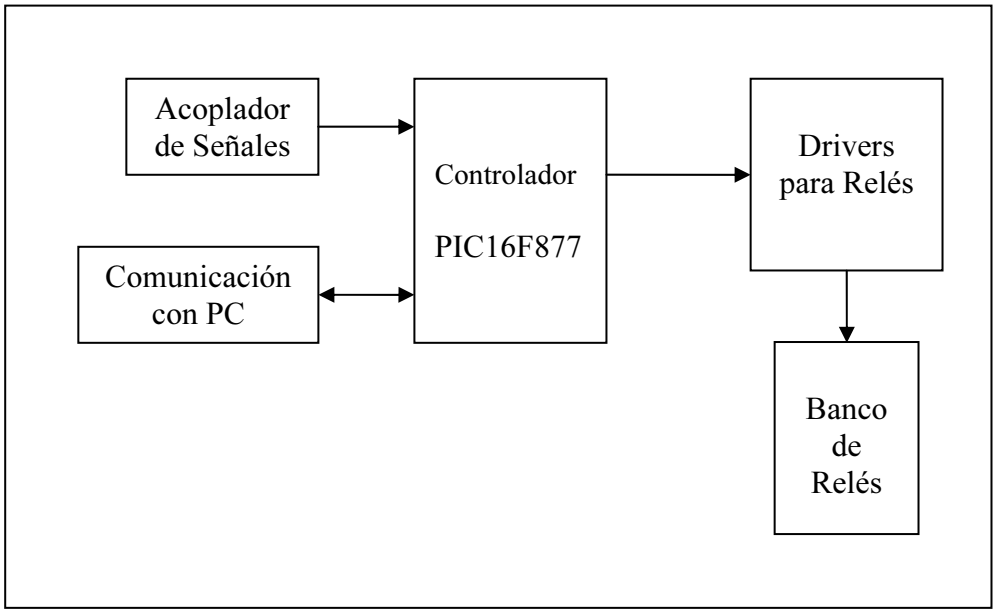


Fig. 2-1 Diagrama de Bloques de la Tarjeta

2.2.4 DIAGRAMA ESQUEMÁTICO

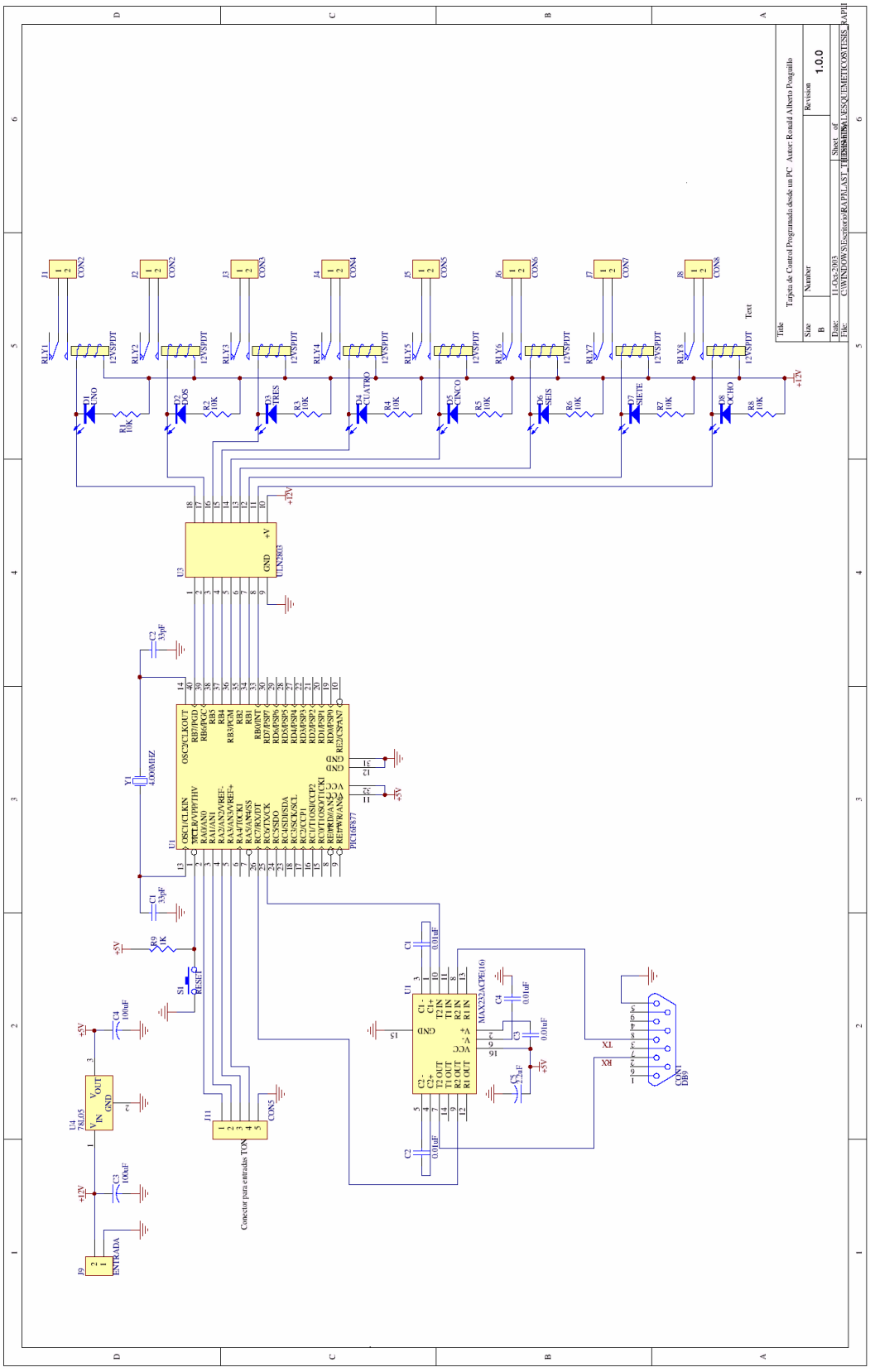


Figura. 2-2 Diagrama Esquemático del la Tarjeta Controladora

2.3 RECURSOS DEL PIC UTILIZADOS

El PIC16F877 para nuestro caso en particular dispone de una versión con 40 pines en los cuales se han implementado 5 puertos de E/S. Como la mayoría de estos pines tienen múltiples funciones, posee entre ellos multiplexados con los puertos de E/S Convertidor A/D de 10 bits, 3 Temporizadores, 2 módulos CCP, Puerto Serie SSP, interfaz Serie SCI y Puerto Paralelo Esclavo. De todos estos recursos este proyecto solamente emplea unos cuantos que se detallan en las siguientes secciones.

2.3.1 PUERTOS DE ENTRADA/SALIDA

Las patitas de comunicación de los microcontroladores se agrupan en conjuntos que a nosotros nos gusta llamar “puertos”, porque dejan entrar y salir la información al procesador. Dichos puertos deben soportar las líneas que precisan los distintos periféricos que hay integrados en la cápsula. Cuantos más periféricos dispone el modelo, exige más líneas de comunicación y mayor número de patitas, con mas multiplexado de señales.

En general, las cápsulas con 18 patitas, como las que contienen al PIC16C62X, PIC16C71 y PIC16C84, destinan 13 a los puertos de E/S, dejando las otras 5 para labores generales, como recibir la tensión de alimentación (V_{DD} y Vss), conectar el cristal de cuarzo (OSC1 y OSC2) y otra para el Reset/Tensión de Programación (MCLR/V_{PP}). Las cápsulas con 28 patitas, como la que contiene le PIC16C73, destinan 22 para los puertos de E/S y, finalmente, en las de 40 patitas (PIC16F877), 33 se dedican a las líneas de E/S.

Mientras que los microcontroladores encapsulados con 18 patitas solo suelen disponer de 2 Puertos de E/S (A y B), los de 40 patitas, llegan a tener 5 puertos de E/S (A, B, C, D y E).

La mayoría de las patitas de los Puertos son multifunción, es decir, soportan diferentes funciones según sean programadas. Así, por ejemplo, existen patitas que a veces funcionan como líneas de E/S digitales y otras como entradas o salidas de señales analógicas.

En el desarrollo de este proyecto sólo se han utilizado el Puerto A como entradas digitales, el Puerto B usado para generar las salidas que pasando a través de los drivers conmutan a los relés, y del Puerto C se tomaron los dos pines para la comunicación serial con la PC. A continuación hago una breve referencia de estos.

PUERTO A

Consta de 6 patitas o líneas (RA0-RA5). Todas, menos RA4, pueden actuar como E/S digitales o como canales de entrada para el Conversor AD. La patita RA4, además de E/S digital puede funcionar como entrada de reloj externo para el TMR0. La tarjeta diseñada utiliza cuatro pines del puerto A como entradas.

FIGURA 2-3: DIAGRAMA DE BLOQUES DE LOS PINES RA3:RA0 Y RA5

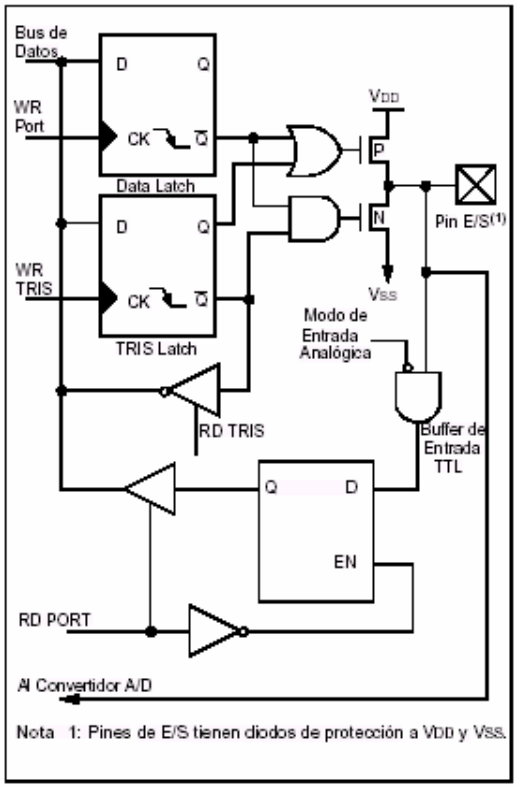
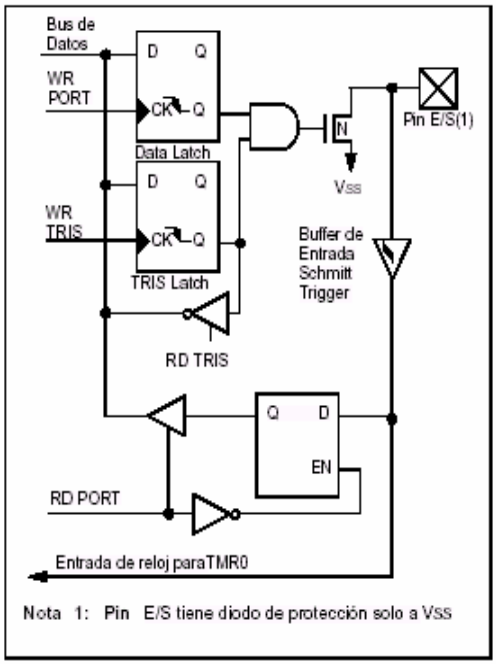


FIGURA 2-4: DIAGRAMA DE BLOQUES DEL RA4/T0CKI



PUERTO B

Las 4 líneas de mas peso del Puerto B (RB<3:0>) actúan como E/S digitales, según la programación del registro TRISB. Además pueden disponer de una carga pull-up interna si se programa la línea como entrada y el bit<7> (RBPO) del registro OPTION vale 0.

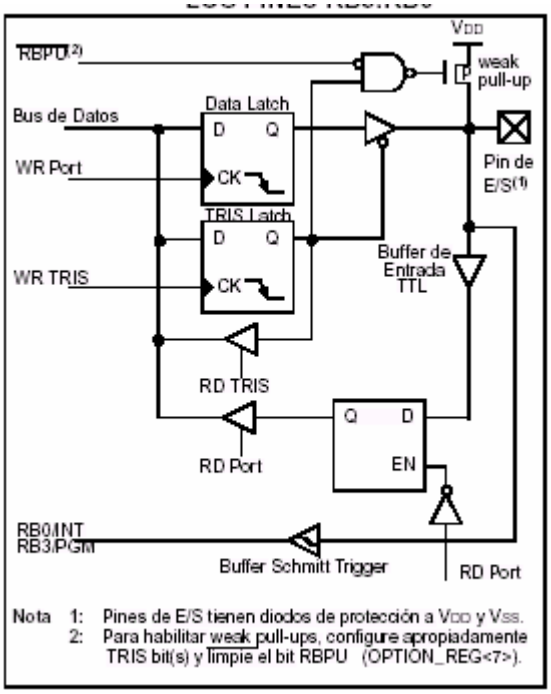


Figura 2-5 Diagrama de bloques de los Pines RB3:RB0

Las líneas RB<7:4> funcionan como las anteriores, pero además pueden provocar una interrupción si se programan como entradas y se produce el cambio de nivel lógico en alguna de ellas. En tal caso se activa el bit <0> (RBIF) de INTCON. La interrupción se anula al borrar el bit <3> (RBIE) de INTCON o al hacer una nueva lectura del Puerto B. El puerto B del PIC en la tarjeta de control está destinado al manejo de las salidas, que conectado a las entradas del driver ULN2803 realiza la conmutación de los relés

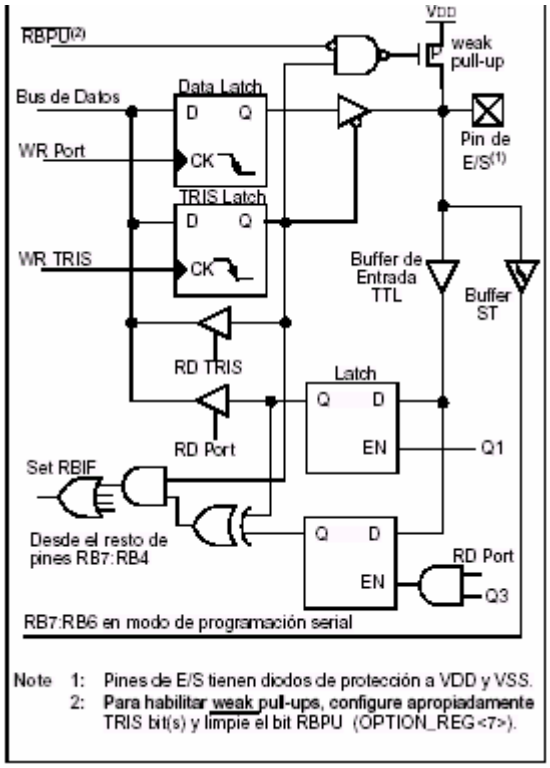


Figura 2-6 Diagrama de Bloques de los Pines RB7:RB4

PUERTO C

Es un puerto bidireccional de 8 bits, como el Puerto B. Cada patita actúa como E/S digital, según la programación de TRISC. Además, también puede actuar como entrada o salida de diversos periféricos internos. Figura 2-7.

A continuación se describe la nomenclatura de cada patita y sus funciones.

RC0/OSO/T1CKI: E/S digital. Salida para la conexión del cristal del oscilador externo para el TMR1. Entrada de reloj para el TMR1.

RC1/OSI/CCP2: E/S digital. Entrada para la conexión del cristal del oscilador externo del TMR1. E/S del modulo2 CCP para Captura/Comparación/PWM.

RC2/CCP1: E/S digital. E/S del modulo1 CCP para Captura/Comparación/PWM.

RC3/SCK/SCL: E/S digital. Reloj síncrono para los modos SPI e I^2C del puerto serie.

RC4/SDI/SDA: E/S digital. Entrada de datos serie en el modo SPI. E/S serie en el modo I^2C.

RC5/SDO: E/S digital. Salida de datos serie en el modo SPI.

RC6/TX/CK: E/S digital. Línea de transmisión asíncrona del canal serie. Reloj sincrono del canal serie.

RC7/RX/DT: E/S digital. Línea de recepción asíncrona del canal serie. Línea de datos del canal serie.

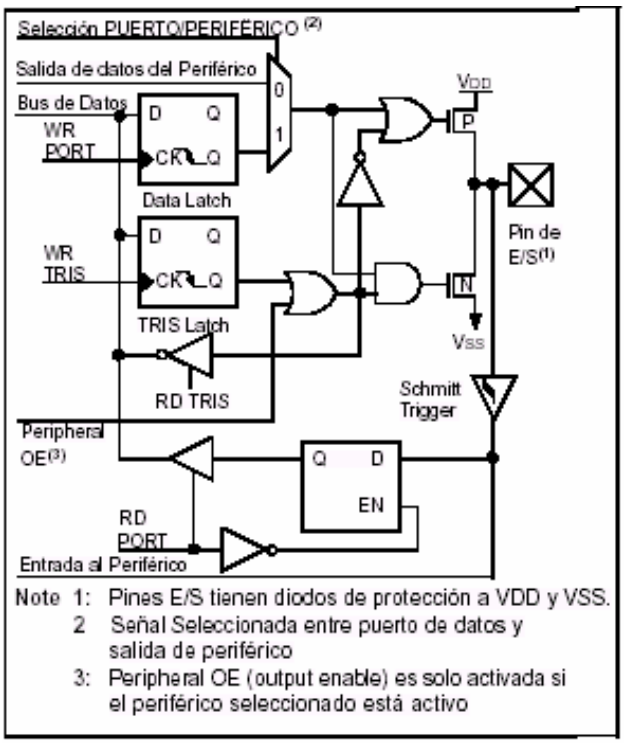


Figura 2-7. Conexiones de los pines del puerto C.

Para la realización de este proyecto se ha utilizado los pines RC7/RX/DT como pin de recepción en la comunicación asincrónica con la PC y el RC6/TX/CK para la transmisión de los datos del PIC hasta la PC.

2.3.2 MÓDULO DE COMUNICACIÓN SERIAL

El módulo Receptor Transmisor Sincrónico Asincrónico (USART) es uno de los dos módulos de I/O seriales del PIC (otro es el módulo

SSP). El USART también es conocido como una Interfase de Comunicación Serial o SCI. El USART puede configurarse como un sistema asíncrono full duplex que puede comunicarse con dispositivos periféricos como terminales CRT y computadoras personales, o puede configurarse como un sistema síncrono half duplex que puede comunicar computadoras, o puede configurarse como un sistema síncrono half duplex que puede comunicarse con dispositivos periféricos como circuitos integrados A/D o D/A, EEPROMs seriales, etc.

El USART puede configurarse en los modos siguientes:

- Asincrónico (full duplex)
- Síncrono - Maestro (half duplex)
- Síncrono - Esclavo (half duplex)

El bit SPEN (RCSTA <7>), y los bits de TRIS, tienen que ser puestos a uno para configurar los pines TX/CK y RX/DT para el USART. El registro específico TXSTA actúa como registro de estado y control del transmisor y el RCSTA hace lo mismo para el receptor.

Los baudios se establecen por el valor cargado en el registro SPBRG y el bit BRGH del registro TXSTA, con el que se puede elegir la velocidad alta (1) o baja (0) en el modo asíncrono.

$$BAUDIOS = F_{OSC} / (n(x + 1))$$

donde:

n = 4 en el modo síncrono

n = 16 en el modo asíncrono de alta velocidad

n = 64 en el modo asíncrono de baja velocidad

x = valor cargado en el registro SPBRG

y despejando nos queda

$$x = (F_{OSC} /(n^* \text{ Baudios})) – 1$$

A partir de esto podemos determinar el valor que cargaremos en el registro SPBRG

Mediante la programación de los bits del registro TXSTA y RCSTA se configura el modo de trabajo. Así, SPEN configura RC7/RX y RC6/TX como líneas de comunicación serie. El transmisor se activa con el bit TXEN. El dato a transmitir se carga en TXREG y luego pasa al registro transmisor TSR, cuando se haya transmitido el bit de stop del dato anterior. Entonces se activa el señalizador TXIF y si el bit de permiso esta activado se produce una interrupción.

Activando Tx8/9 se inserta el noveno bit almacenado en el bit <0> (TXD8) de TXSTA. El bit TRMT indica si el transmisor esta vacío o no. El dato se recibe por RSR y cuando se completa se pasa al registro RCREG para su posterior lectura, activándose el señalizador RCIF y si acaso fue habilitada se genera la interrupción.

Si se activa el bit RC8/9del RCSTA el noveno bit se deposita en el bit <0> (RCD8) del RCSTA. Los bits OERR y FERR indican error de desbordamiento y de trama, respectivamente.

Para este proyecto en particular he utilizado el USART del PIC en modo asincrónico con capacidad de comunicación full duplex.

A continuación se muestra un diagrama que explica el funcionamiento del Transmisor del USART y se detallan los pasos necesarios para configurar una transmisión asíncrona.

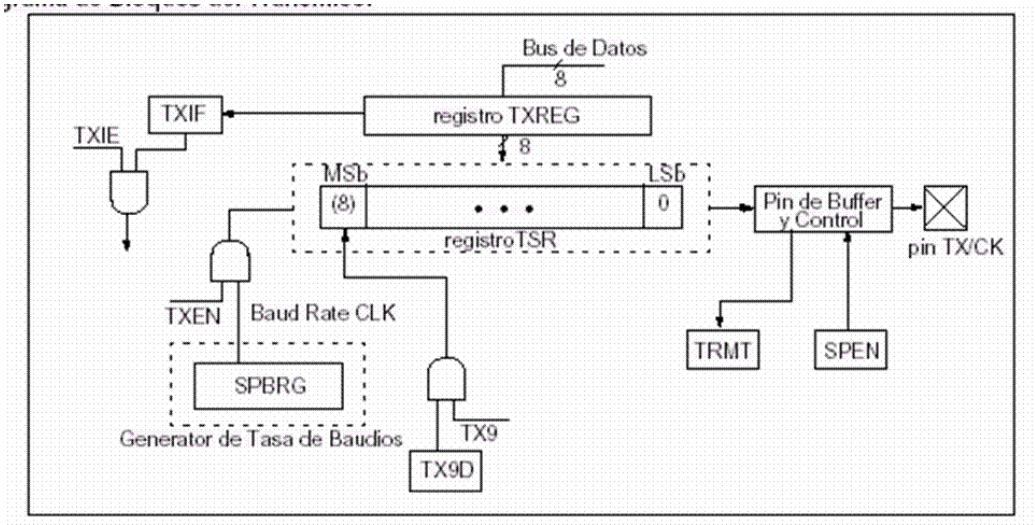


Figura 2-8 Diagrama del Transmisor Asíncrono

Pasos a seguir al preparar una Transmisión Asíncrona:

1. Inicialice el registro SPBRG con la tasa de baudio apropiada. Si es deseada una tasa de baudios de alta velocidad, ponga a uno el bit BRGH.
2. Habilite el puerto serie asíncrono limpiando el bit SYNC y poniendo a uno el bit SPEN.
3. Si se desean interrupciones, entonces ponga a uno los bits TXIE, GIE y PEIE.
4. Si se desea transmisión de 9-bits, entonces ponga a uno el bit TX9.
5. Habilite la transmisión poniendo a uno el bit TXEN que también pondrá a uno el bit TXIF.
6. Si se selecciona transmisión de 9-bits, el noveno bit debe cargarse en el bit TX9D.
7. Cargar los datos en el registro TXREG (inicia la transmisión).

En la figura 2-9 se muestra los diagramas de tiempo de la transmisión asíncrona maestra y en la figura 2-10 de la transmisión asíncrona maestra back to back.

USART: Transmisión Asincrónica Maestra

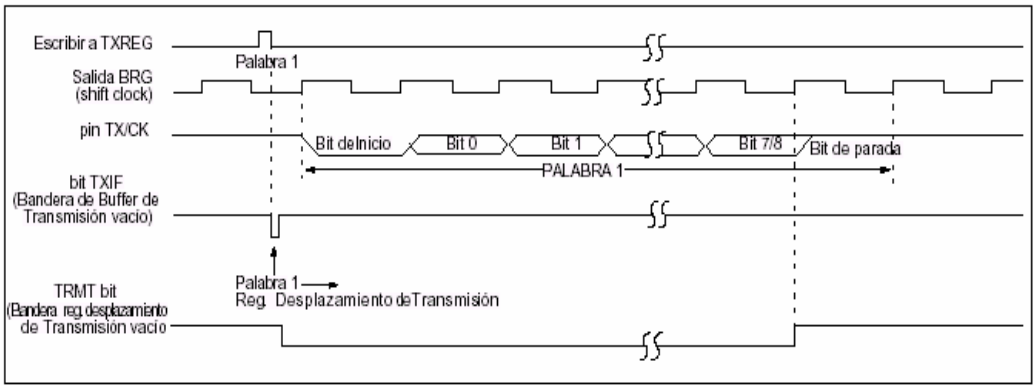


Figura 2-9 Diagrama de tiempo Transmisión Asíncrona Maestra

USART: Transmisión Asincrónica Maestra (Back to Back)

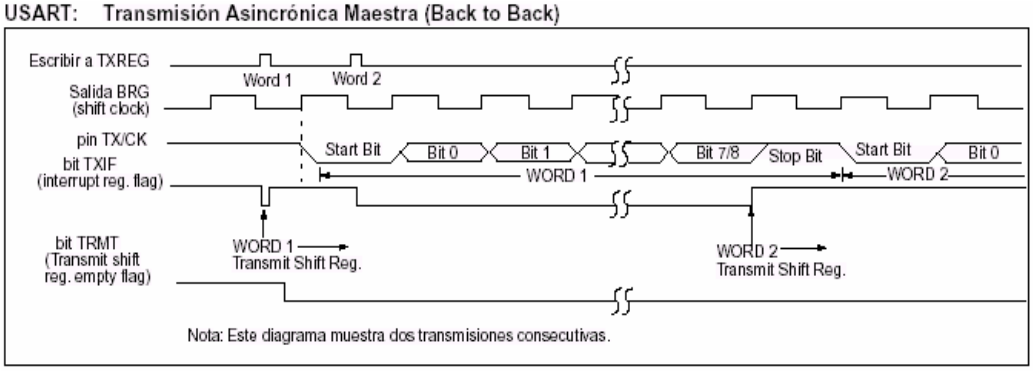


Figura 2-10 Diagrama de tiempo Transmisión Asíncrona Maestra (Back to Back)

En la siguiente figura se muestran los registros asociados con la transmisión asíncrona junto con los valores después de un reset.

USART: Registros Asociados con la Transmisión Asincrónica

Nombre	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Valor en: POR, BOR	Valor en el resto de Resets
PIR	TXIF (1)								0	0
RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
TXREG	TX7	TX6	TX5	TX4	TX3	TX2	TX1	TX0	0000 0000	0000 0000
PIE	TXIE (1)								0	0
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Leyenda: x = desconocido, - = no implementado se lee como '0'.

Nota 1: La posición de éste bit depende del dispositivo.

Figura 2-11 Registros Asociados con la Transmisión Asincrónica

Como se hizo para el caso de la transmisión, aquí se muestra un gráfico que explica el funcionamiento en bloques del módulo de recepción asincrónica, y a continuación los pasos necesarios para realizar la recepción.

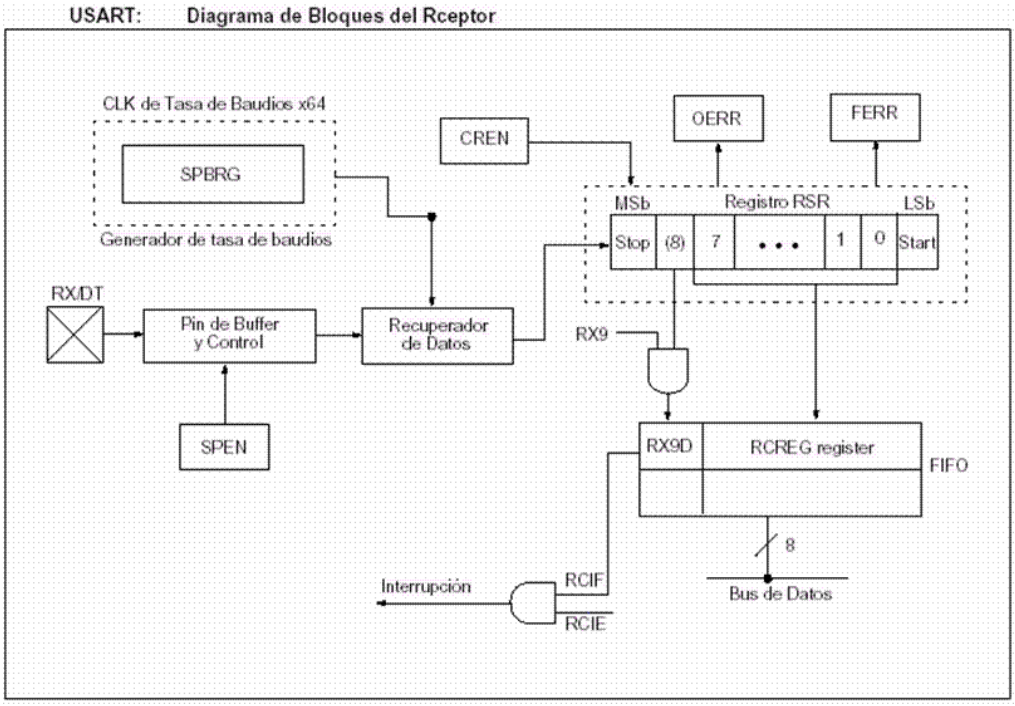


Figura 2-12 Diagrama de bloques del Receptor Asincrónico

Pasos a seguir para preparar una Recepción Asíncrona:

1. Inicialice el registro SPBRG con la tasa de baudios apropiada. Si se desea una tasa de baudios de alta velocidad, ponga a uno el bit BRGH.
2. Habilite el puerto serial asíncrono limpiando el bit SYNC, y poniendo a uno el bit SPEN.
3. Si se desean interrupciones, entonces ponga a uno los bits RCIE, GIE y PEIE.
4. Si se desea recepción de 9-bits, ponga a uno el bit RX9.
5. Habilite la recepción poniendo a uno el bit CREN.

6. La bandera RCIF se pondrá a uno cuando la recepción esté completa y una interrupción se generará si el bit RCIE fue puesto a uno.
7. Lea el registro RCSTA para obtener el noveno bit (si lo habilitó) y determine si ocurrió algún error durante la recepción.
8. Obtenga los 8-bits de datos recibidos leyendo el registro RCREG.
9. Si cualquier error ocurriera, borre el error limpiando el bit CREN.

Para entender un poco mejor este proceso de recepción se muestra a continuación el diagrama de tiempos correspondiente.

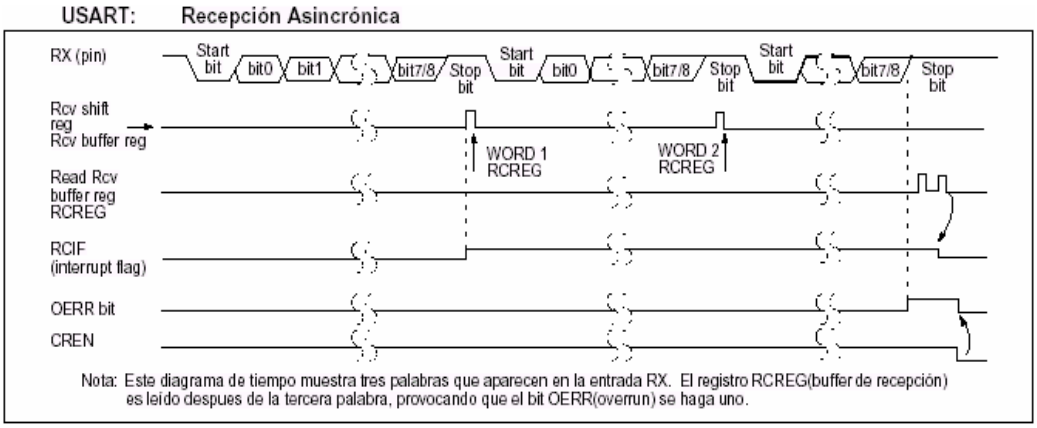


Figura 2-13 Diagrama de tiempo de la Recepción Asíncrona

Para terminar con la recepción en la figura 2-14 se muestran los registros que tienen relación con esta parte.

USART: Registros Asosciados con la Recepción Asincrónica

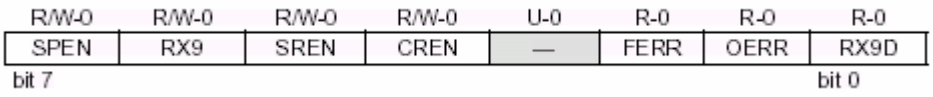
Nombre	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Valor en: POR, BOR	Valor en los demás Resets
PIR	RCIF (1)								0	0
RCSTA	SPEN	RX9	SREN	CREN	—	FERR	OERR	RX9D	0000 -00x	0000 -00x
RCREG	RX7	RX6	RX5	RX4	RX3	RX2	RX1	RX0	0000 0000	0000 0000
PIE	RCIE (1)								0	0
TXSTA	CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D	0000 -010	0000 -010
SPBRG	Baud Rate Generator Register								0000 0000	0000 0000

Leyenda: x= desconocido, -= no implementado, se lee como '0'.

Figura 2-14 Registros Asociados con la Recepción Asincrónica

Todos los elementos internos de los PICs son controlados mediante bits que se encuentran distribuidos en varios registros dentro de la memoria RAM de datos. En esta parte se hace hincapié en los registros y cada uno de los bits que tienen que ver con el módulo de comunicación serial.

Registro RCSTA: Registro de Estado y Control de Recepción



bit 7 **SPEN:** Bit habilitador del puerto serial
1 = Puerto Serial habilitado (Configura pines RX/DT y TX/CK como puerto serial)
0 = Puerto Serial Deshabilitado

bit 6 **RX9**: Bit habilitador de recepción de 9-bits
1 = Recepción de 9-bits seleccionada
0 = Recepción de 8-bits seleccionada

bit 5 **SREN**: Bit habilitador de recepción simple
Modo Asincrónico
No importa
Modo sincrónico - maestro
1 = Habilita recepción simple
0 = Deshabilita recepción simple
Este bit es limpiado después que la recepción está completa.
Modo sincrónico - esclavo
No es usado en este modo

bit 4 **CREN**: Bit habilitador de recepción continua
Modo Asincrónico
1 = Habilita recepción continua
0 = Deshabilita recepción continua

Modo Sincrónico

1 = Habilita la recepción continua hasta que el bit CREN es limpiado (CREN sobrescribe a SREN)

0 = Deshabilita recepción continua

bit 3 **No implementado:** Se lee como '0'

bit 2 **FERR**: Bit de error de trama

1 = Error de trama(Puede ser actualizado leyendo el registro RCREG y recibe el próximo byte válido)

0 = No hay error de trama

bit 1 **OERR**: Bit de error por desbordamiento

1 = Error por desbordamiento (Puede ser borrado, limpiando el bit CREN)

0 = No hay error por desbordamiento

bit 0 **RX9D:** 9no bit de dato recibido, puede ser bit de paridad.

Leyenda

R = Bit leíble W = Bit escribible

U = No implementado se lee como ‘0’ - n = Valor en reset POR

Registro TXSTA: Registro de Estado y Control de la Transmisión

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

bit 7 **CSRC:** Bit Selección de fuente de reloj

Modo Asincrónico

No importa

Modo Sincrónico

1 = Modo Maestro (Reloj generado internamente desde BRG)

0 = Modo esclavo (Reloj desde fuente externa)

bit 6 **TX9**: Bit habilitador de transmisión de 9-bits

1 = Selecciona transmisión de 9-bits

0 = Selecciona transmisión de 8-bits

bit 5 **TXEN**: Bit habilitador de la Transmisión

1 = Transmisión Habilitada

0 = Transmisión Deshabilitada

Nota: SREN/CREN sustituye a TXEN en modo SYNC.

bit 4 **SYNC**: Bit de selección de modo USART

1 = Modo Sincrónico

0 = Modo Asincrónico

bit 3 **No implementado:** Se lee como '0'

bit 2 **BRGH**: Bit de selección de tasa de baudios de alta velocidad

Modo Asincrónico

1 = Alta velocidad

0 = Baja velocidad

Modo Sincrónico

No es usado en este modo

bit 1 **TRMT**: Bit del estado del registro de transmisión

1 = TSR vacío

0 = TSR lleno

bit 0 **TX9D:** 9^{no} bit de dato. Puede ser bit de paridad.

Leyenda

R = Bit leíble W = Bit escribible

U = No implementado se lee como ‘0’ - n = Valor en reset POR

2.3.3 TEMPORIZADOR

Una exigencia en las aplicaciones de control es la regulación estricta de los tiempos que duran las diversas acciones que realiza el sistema. El dispositivo típico destinado a gobernar los tiempos recibe el nombre de temporizador o "timer" y, básicamente, consiste en un contador ascendente o descendente que determina un tiempo determinado entre el valor que se le carga y el momento en que se produce su desbordamiento o paso por 0.

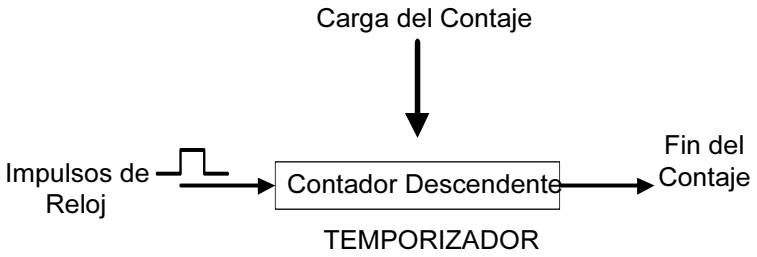


Figura 2-15 Esquema simplificado de un temporizador. En este caso se trata de un contador descendente, que, una vez cargado con un valor, se decrementa al ritmo de los impulsos de reloj hasta que llega a 0

Además del Perro Guardián con el que cuentan todos los PICs el PIC16F877 posee tres contadores/temporizadores llamados TIMER0, TIMER1 y TIMER2, siendo el TIMER1 de 16 bits y los otros de 8. Dentro de las especificaciones de este proyecto se debe generar una temporización de 1 segundo, cuyo valor se genera con más exactitud valiéndonos del TIMER2, su capacidad de generar interrupción y de una variable contador auxiliar en el programa del PIC, por ello he elegido usar este. En seguida paso a dar algunos detalles de su funcionamiento.

El TIMER2 solo esta incorporado en unos pocos modelos de la gama media porque se trata de un temporizador de 8 bits diseñado para usarse conjuntamente con el circuito de Modulación de Anchura de Impulsos (PWM).

Se incrementa al ritmo de los impulsos que se le aplica ($4.T_{OSC}$), que pueden ser divididos por 1, por 4 o por 16 mediante un Predivisor. Cuando el valor del registro TMR2 coincide con el del PR2 (Registro de Periodo) se genera un impulso en la salida EQ y TMR2 pasa a 00h. PR2 es un registro específico de lectura y escritura que cuando hay un Reset se carga con el valor FFh. Los impulsos producidos por EQ se aplican a un Post-divisor que puede dividirlos hasta 1:16, activando su salida al señalizador TMR2IF. El registro T2CON regula los principales parámetros de este temporizador. La salida EQ se puede utilizar como señal de reloj para el modulo de interfaz serie SSP.
En la figura que se muestra a continuación se puede ver el diagrama a bloques de su funcionamiento.

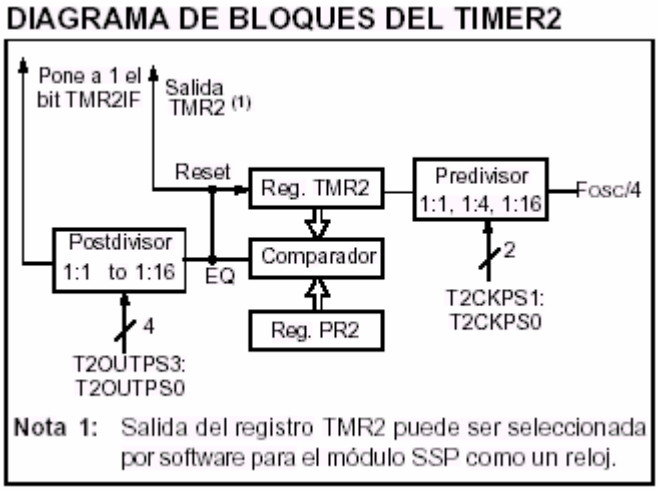


Figura 2-16 Diagrama de bloques del TIMER2

El reset borra al Predivisor y al Post-divisor. También lo hace al TMR2 cuando se ha generado como consecuencia del WDT, POR o MCLR. Cada vez que se escribe sobre el registro TMR2 o el T2CON se borran el Predivisor y el Post-divisor. Todo esto, junto con los registros involucrados con el TIMER2 cuando funciona como temporizador/contador se muestra en la figura 2-17.

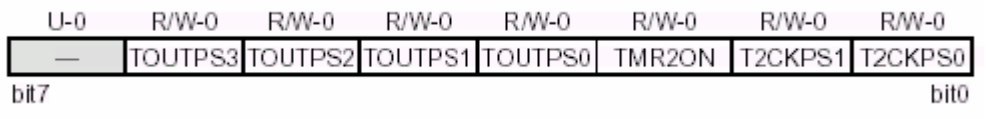
REGISTROS ASOCIADOS CON EL TIMER2 COMO UN TEMPORIZADOR/CONTADOR

Direccion	Nombre	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Valor en: POR, BOR	Valor en los otros resets
0Bh,8Bh, 10Bh,18Bh	INTCON	GIE	PEIE	T0IE	INTE	RBIE	T0IF	INTF	RBIF	0000 000x	0000 000u
0Ch	PIR1	PSPIF(1)	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF	0000 0000	0000 0000
8Ch	PIE1	PSPIE(1)	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE	0000 0000	0000 0000
11h	TMR2	Registro del módulo Timer2								0000 0000	0000 0000
12h	T2CON	—	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0	-000 0000	-000 0000
92h	PR2	Registro de PeriodoTimer2								1111 1111	1111 1111

Figura 2-17 Registros Asociados con el TIMER2 como Temporizador/Contador

Para concluir se pone a consideración del lector el registro de control del TIMER2 con la funcionalidad de cada uno de sus bits.

T2CON: REGISTRO DE CONTROL DEL TIMER2 (DIRECCIÓN 12h)



bit 7: **No implementado:** Se lee como '0'

bit 6-3: **TOUTPS3:TOUTPS0**: Bit de selección del Postdivisor

0000 =1:1 Postdivisor

0001 = 1:2 Postdivisor

0010 = 1:3 Postdivisor

•

•

•

1111 = 1:16 Postdivisor

bit 2: **TMR2ON**: Bit para encender el Timer2

1 = Timer2 encendido

0 = Timer2 apagado

bit 1-0: **T2CKPS1:T2CKPS0**: Bit de selección del Predivisor

00 = Predivisor es 1

01 = Predivisor es 4

1x = Predivisor es 16

Leyenda

R = Bit leíble W = Bit escribible

U = No implementado se lee como ‘0’ - n = Valor en reset POR

CAPITULO 3

DISEÑO DEL FIRMWARE DEL PIC

3.1 INTRODUCCIÓN

En este capítulo se aborda el diseño de las subrutinas que componen el firmware que residirá permanentemente en la memoria de programa del PIC, el cual podría comparárselo con un sistema operativo ya que permite al mundo externo comunicarse con la tarjeta de control y hacer uso de los recursos con los que éste cuenta como comunicación serial, temporización, puertos de entrada digitales y puerto para salida de control.

Para el desarrollo de cada subrutina se ha hecho un análisis de las necesidades del recurso que estas permitirían utilizar, basado en la solución propuesta para este proyecto, así como la configuración que deben tener cada uno de estos recursos para poder desarrollar la tarea que se requiere. Luego se establecen los algoritmos que nos permitirán resolver cada una de éstas temáticas, basados en diagramas de flujo. Finalmente se analiza algunos fragmentos de código que sirven para comprender mejor la solución aplicada y se muestra un bosquejo de la utilización de la memoria de programa.

3.2 DIAGRAMA DE FLUJO GENERAL

A continuación se presenta el diagrama de flujo general que muestra la estrategia implementada para el funcionamiento de la tarjeta. Se puede notar pasos comunes para la utilización de microcontroladores como la configuración de puertos de entrada/salida y estrategias de programación como la inicialización de variables. El proceso configurar USART hace posible la utilización de éste en modo de transmisión

asincrónica con ocho bit de datos, un bit de parada y 9600 baudios, y maneja la recepción de datos mediante la utilización de la interrupción generada para el caso. El timer se lo configura para que produzca la interrupción correspondiente cuando se desborde, así se controla el tiempo con mas precisión que si se lo hiciera mediante el sondeo del registro específico.

En la figura 3-1 se muestra el diagrama de flujo general del firmware del microcontrolador. En el se pueden ver cada uno de los procesos que hacen posible el funcionamiento de la tarjeta y que serán analizados con mas detalles después.

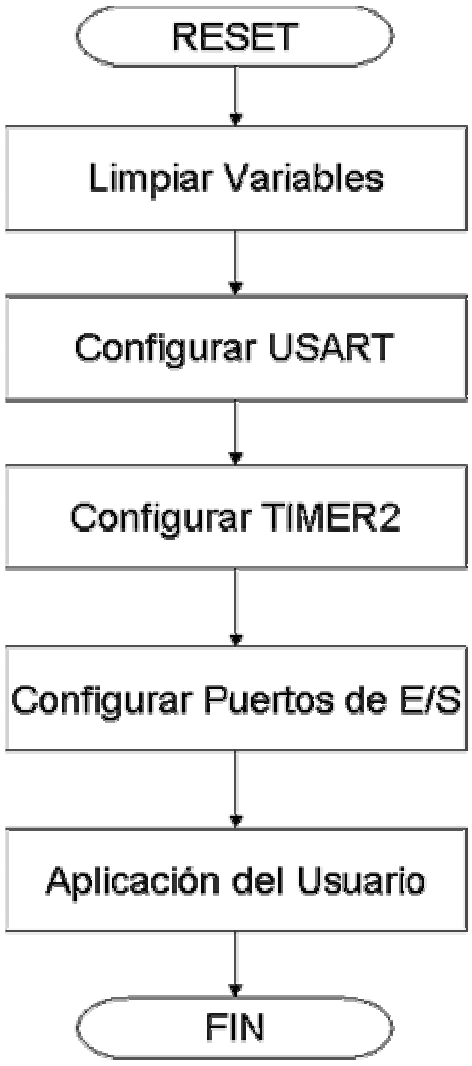


Figura 3-1 Diagrama de Flujo General del Firmware

3.3 RUTINA DE COMUNICACIÓN SERIAL

Otra parte fundamental en el desarrollo de este proyecto es la comunicación de la tarjeta con la computadora, lo cual se consigue elaborando una subrutina de comunicación serial tanto en la PC, como en el microcontrolador. En esta parte detallaré la rutina implementada en el PIC y la implementada en el software de la PC quedará para el siguiente capítulo.

Dentro de la funcionalidad de la tarjeta controladora se tomaron en consideración varios aspectos que están ligados a la comunicación con la PC, como por ejemplo la capacidad de reprogramar la tarjeta, poder dar instrucciones a esta directamente desde la PC o saber el estado de cada una de las entradas o salidas de la controladora. Debido a esto se generó el algoritmo de recepción serial por parte del PIC, el cual a su vez da paso a la ejecución de otras tres subtareas que son la rutina de autoprogramación, enviar reporte y recibir comandos, como se puede observar en la figura 3-2. Cabe indicar que para manejar la recepción asíncrona en el PIC se ha hecho uso de la interrupción respectiva, por tanto, cuando es generada una interrupción por el receptor del USART, el flujo del programa es interrumpido y el contador del programa apunta al vector de interrupciones, y en este se determina que elemento fue el produjo la interrupción mediante la comparación de sus bits de bandera. Una vez que se determinó que la interrupción fue provocada por el receptor serial, se procede a borrar el bit de bandera de éste y revisar los bits de bandera del registro FLAG que se implementó en el código del PIC para determinar cual es la tarea que se debe realizar.

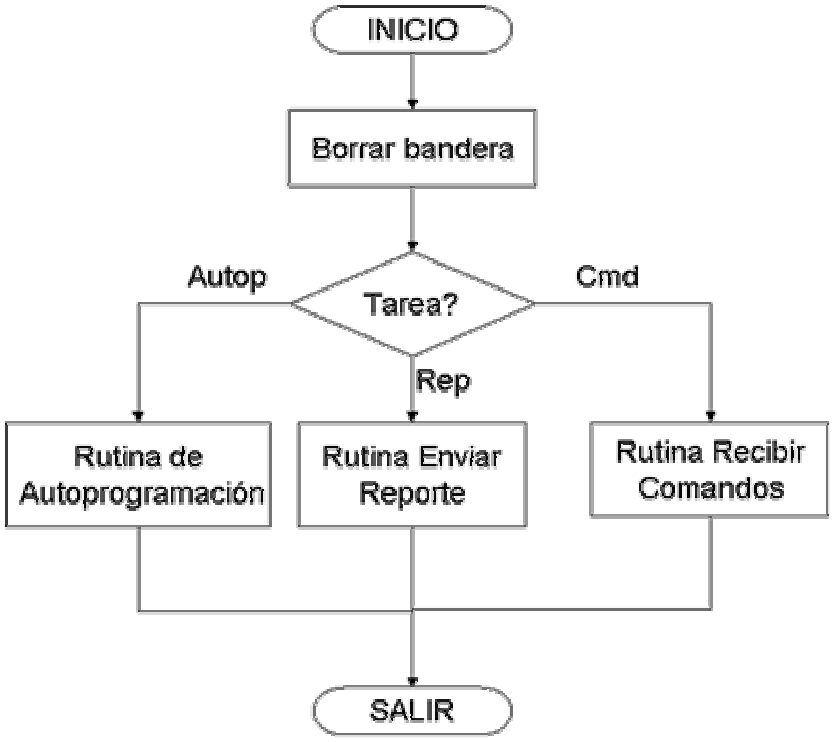


Figura 3-2 Rutina de Comunicación Serial

A continuación se muestra el código que corresponde al diagrama de flujo mostrado en la figura 3-2. En el se puede observa la variable FLAG, que es un registro definido en la cabecera del programa.

```
RECEPCION       Aquí debo supervisar la bandera
        banksel PIR1
        bcf     PIR1,RCIF   ; deshabilito la bandera
Recibir_Bandera             ; Recibo y decodifico bandera
        banksel FLAG
        btfsc   FLAG,FDECO  ;Verifico si la bandera ya fue
                            decodificada
        goto    TAREA           ; Si, realizar tarea
        pagesel Decodificar_FLAG
        goto    Decodificar_FLAG ; No, Decodificamos bandera
Decodificar_FLAG
        banksel RCREG
        movlw   0x80        ;
```

```
andwf     RCREG,w   ;
btfss     STATUS,Z  ; ¿Es autoprogramación?
goto      AutoP     ; Si
movlw     0x20      ;
andwf     RCREG,w   ;
btfss     STATUS,Z  ; ¿Es Reporte?
goto      Reprt     ; Si
movlw     0x10      ;
andwf     RCREG,w   ;
btfss     STATUS,Z  ; ¿Es Comando?
goto      Command   ;
movlw     0x08
andwf     RCREG,w   ;
btfss     STATUS,Z  ; ¿Es Fin de Tarea?
goto      Fin_T     ;
pagesel   FIN
goto      FIN       ;
```

Aquí la tarea es enviada por la PC serialmente y recogida en el registro RCREG del USART para su posterior decodificación.

3.4 RUTINA DE TEMPORIZACIÓN

Sin duda alguna cuando queremos realizar algún proceso de control la mayoría de las veces habrá momentos que requieran la posibilidad de manejar tiempos de retardo entre uno y otro evento, esto se puede conseguir en nuestro caso usando alguno de los temporizadores integrados que posee el PIC. En el capítulo anterior establecimos las razones para la elección del TIMER2, en este capítulo analizaremos la manera como lo utilizamos para generar una temporización de un segundo y la forma como puede ser usado dentro del programa de usuario. En realidad con el TIMER2 lo que se ha generado es una

temporización de 40ms, claro está trabajando en las condiciones siguientes: con un reloj del sistema de cuatro megahertz, poniendo el predivisor en diez y el postdivisor en 16, y cargando el registro PR2 con el valor 250. Haciendo el cálculo respectivo tendríamos lo siguiente:

Tci = Periodo del ciclo de instrucción

M = Valor del predivisor

N = Valor del postdivisor

$$T = Tci * M * N * PR$$

$$T = 1us * 10 * 16 * 250 = 40000\ us$$

Del cálculo anterior podemos observar que se obtuvo 40ms, pero la idea es realizar temporizaciones de 1s. Para conseguirlo nos valemos de un registro auxiliar que he definido como variable auxiliar para la temporización dentro del código del microcontrolador, en la cual se controla que cuente veinticinco veces el tiempo T que calculamos anteriormente quedándonos como resultado total de la temporización un segundo exacto. Hay que indicar que el incremento de esta variable auxiliar se lo realiza cuando el TIMER2 genera la interrupción correspondiente. En la figura 3-3 se muestra el diagrama de flujo de la subrutina que da servicio a la interrupción del TIMER2 para completar la temporización de un segundo.

Figura 3-3 Rutina de Temporización

El siguiente fragmento de código muestra como inicializar el TIMER2 para que funcione en base a los requerimientos planteados.

```
movlw   .250            ; 250
movwf   PR2             ;
banksel TMR2
clrf    TMR2            ;
banksel T2CON
bsf     T2CON,T2CKPS1   ; Predivisor 16
bsf     T2CON,T2CKPS0   ;   "       "
bsf     T2CON,TOUTPS3   ; Postdivisor 10
bcf     T2CON,TOUTPS2   ; Postdivisor 10
bcf     T2CON,TOUTPS1   ; Postdivisor 10
bsf     T2CON,TOUTPS0   ; Postdivisor 10
banksel PIR1
bcf     PIR1,TMR2IF     ;
banksel PIE1
bsf     PIE1,TMR2IE
bsf     INTCON,PEIE     ;
bsf     INTCON,GIE      ;
banksel T2CON
```

```
bsf         T2CON,TMR2ON        ; Enciende el TIMER2
```

Una vez que el TIMER2 se ha configurado de acuerdo al código mostrado anteriormente, el microcontrolador tiene la capacidad de realizar la subrutina mostrada en el diagrama de flujo de la figura 3-3, que en pocas palabras lo que hace es incrementar un contador cada vez que el TIMER2 produce una interrupción hasta que se halla completado un segundo, luego de lo cual se encera el contador para empezar con un nuevo conteo.

En cuanto a lo que tiene que ver con el manejo de esta temporización por parte del usuario, se ha diseñado una macro en la cual se necesita cargarle únicamente el número de segundos que se desea de retardo. Funcionalmente hablando esta macro llama a un procedimiento que es quien realmente realiza los bucles necesarios para esta tarea. Podría pensarse ¿por qué mezclar una macro con un procedimiento y no usar solamente uno de ellos?, bueno la respuesta está en hacer mas eficiente el trabajo debido a que por un lado cuando uno escribe una macro en alguna línea de un programa lo que hace el compilador es escribir en memoria todo el código de la macro en cada una de las líneas donde aparece su nombre, lo cual dependiendo del programa del usuario podría gastar mucha memoria innecesariamente. Debido a esto lo que he hecho es escribir la parte funcional del código dentro de un procedimiento común que es llamado desde la macro con una instrucción call, y a su vez la macro le pasa el parámetro que el usuario introdujo, logrando que todo el proceso sea transparente para el usuario y que únicamente su trabajo se limite a escribir algo como esto TIME .20, con lo cual se estaría generando dentro del programa de usuario un retardo de 20 segundos. A continuación se muestran el código de la macro y del procedimiento.

```
TIME macro      t
        banksel     CONT_1
        clrf        CONT_1
        movlw       t
        movwf       T
        pagesel     TIEMPO
        call        TIEMPO
        pagesel     PROCEDER
        endm

TIEMPO
        banksel     CONT_1
        movf        T,w
        subwf       CONT_1,w
        btfss       STATUS,Z
        goto        $-4
        banksel     CONT_1
        clrf        CONT_1
        return
```

3.5 RUTINA DE AUTOPROGRAMACIÓN DE LA MEMORIA DE PROGRAMA

Aunque todas las subrutinas tienen mucha importancia, la subrutina de auto programación de la memoria FLASH le da mayor fuerza al proyecto debido a que le proporciona cierta autonomía considerando que no se necesita de un programador adicional para poder cambiar el programa principal o programa de usuario. A continuación se muestra el diagrama de flujo que establece la estrategia para programar la memoria FLASH de código.

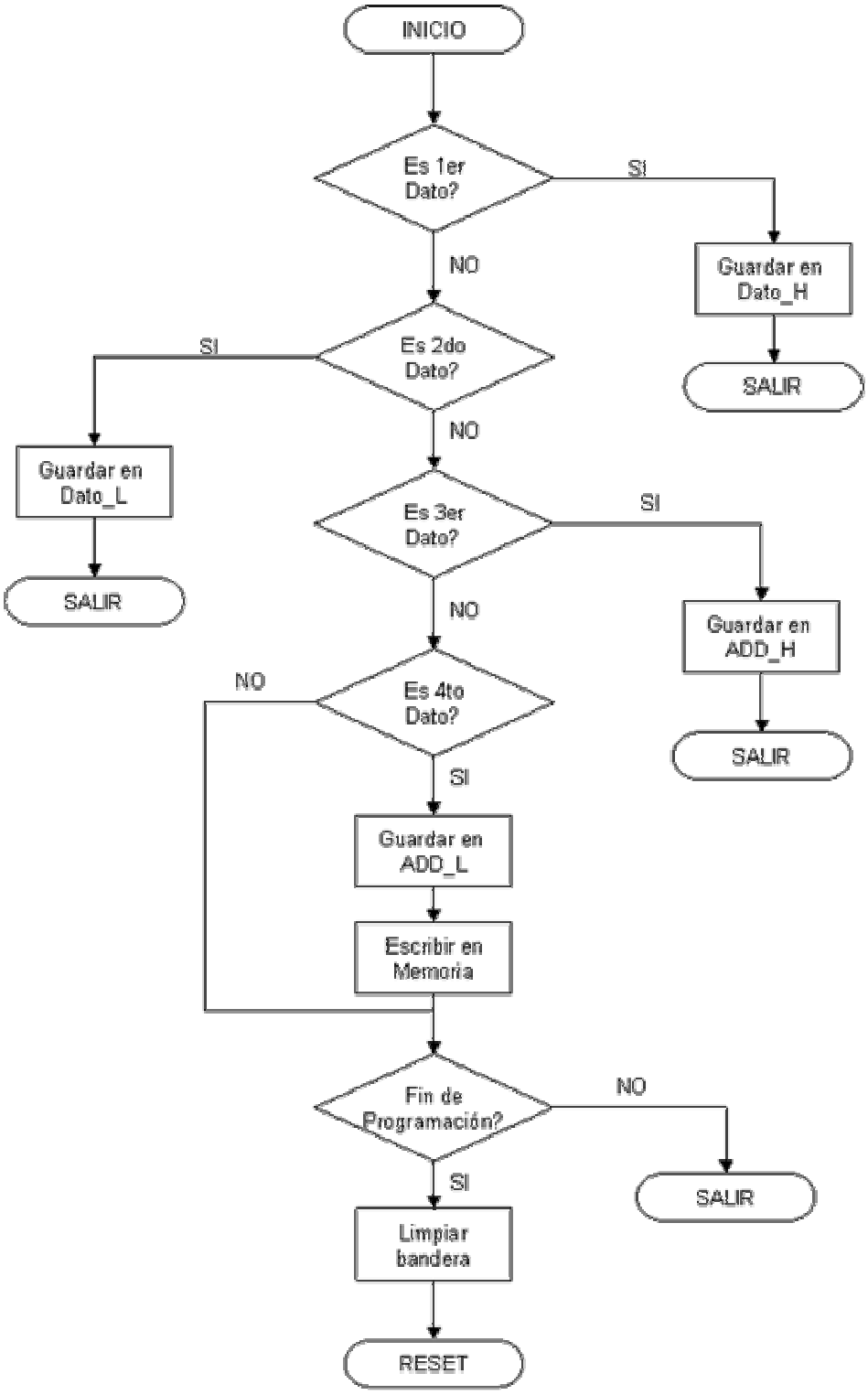


Figura 3-4 Rutina de Auto Programación de la Memoria de Programa

Esta es una de las tareas que se ejecuta con la interrupción de recepción del USART, luego de haber revisado el bit de bandera correspondiente. La idea se fundamenta en la propiedad de los

microcontroladores PIC de poder escribir tanto en la memoria de tipo EEPROM o FLASH utilizada para guardar tanto código como datos. Esto se consigue guardando el dato que se quiere grabar en dos registros del microcontrolador implementados para el efecto que son el EEDATH y EEDATA. Se utilizan dos registros para almacenar un dato de instrucción debido a que los registros que están implementados en este microcontrolador son de ocho bits y las instrucciones que maneja de catorce.

Adicionalmente hay que pasarle la dirección en la que se requiere guardar ese dato en otros dos registros que son EEADRH y EEADR, cargándose en EEADRH los cinco bits de más peso que forman parte de la mencionada dirección y en EEADR los ocho bits menos significativos. Luego que se tiene esto se debe enviar la secuencia 55h, AAh al registro EECON2, dicha secuencia es como una clave de seguridad cuando se quiere escribir en esta memoria que el fabricante ha establecido para prevenir posibles errores en el manejo de la misma.

Algo que hay que tener muy presente al momento de escribir en la memoria de código es deshabilitar las interrupciones de los diferentes elementos que se han configurado en alguna parte del programa, esto se consigue haciendo cero el bit GIE del registro INTCON mientras dura la escritura y volver a ponerlo en uno cuando el dato ha sido escrito. El siguiente código muestra la manera de llevar a cabo la escritura de una instrucción en la memoria de código.

```
banksel   ADDR_H
movf      ADDR_H,w      ; Se mete en EEADRH la parte alta de
banksel   EEADRH
movwf     EEADRH        ; la dirección a escribir
banksel   ADDR_L
```

```
movf    ADDR_L,w      ; En EEADR la parte baja de la
banksel EEADR
movwf   EEADR         ; dirección a escribir
banksel DATA_H
movf    DATA_H,w      ; En EEDATAH la parte alta del dato
banksel EEDATH
movwf   EEDATH        ; a escribir
banksel DATA_L
movf    DATA_L,w      ; parte baja del dato
banksel EEDATA
movwf   EEDATA        ; a escribir
banksel EECON1
bsf     EECON1,EEPGD      ; Se selecciona el acceso a la
                            memoria FLASH
bsf     ECON1,WREN    ; Se habilita la escritura en la FLASH
bcf     INTCON,GIE    ; Se bloquean las interrupciones
movlw   0x55          ;
movwf   EECON2        ; Se ingresa la clave
movlw   0xAA          ; de funcionamiento
movwf   EECON2        ;
bsf     EECON1,WR     ; Se da la orden de escritura
nop
nop
bsf     INTCON,GIE    ; Se habilita las interrupciones
bcf     EECON1,WREN   ; Se bloquea la escritura de nuevos
                        datos
```

3.6 RUTINA DE SERVICIO DE INTERRUPCIÓN

Se ejecuta cada vez que algún bit de interrupción se hace uno y es la encargada de dar las prioridades cuando se han configurado varias fuentes de interrupción y estas se accionan al mismo tiempo, y también

direcciona al contador del programa hasta la subrutina que va a dar servicio a la interrupción específica.

Esto es lo que se muestra en la figura 3-5 donde se puede notar que para cada ingreso al vector de interrupción lo primero que se hace es respaldar los valores que en ese momento tienen varios registros que podrían hacer variar el funcionamiento normal del programa al retornar de la subrutina. Dichos registros son el PCLATH que guarda el valor del byte alto de la dirección de la siguiente instrucción que se debe ejecutar, esto se hace para poder volver a la siguiente dirección absoluta de memoria desde donde fue interrumpido el programa principal cuando se termine de dar servicio a la rutina de interrupción correspondiente para cada caso. Otro registro que se necesita conservar es el registro de trabajo, ya que en el momento que se interrumpe el flujo del programa principal éste podría contener datos necesarios para realizar algún proceso. Asimismo es importante conservar las condiciones de estado del programa como por ejemplo determinar si en un momento dado hubo una operación matemática o de control que diera como resultado cero, o generado algún acarreo; esto se consigue guardando el contenido del registro de estado en alguna variable temporal. Al finalizar cada rutina de servicio de las interrupciones es necesario recuperar los valores de los registros mencionados anteriormente

Para el caso particular de este proyecto se han configurado dos fuentes de interrupción que son la que genera el receptor del USART y la del Timer2. Se puede ver en el diagrama de flujo que en ambos casos lo que se hace es revisar el bit de bandera correspondiente y enviar el flujo del programa a su subrutina de servicio.

El fragmento de código que se presenta a continuación son las líneas necesarias para respaldar los registros mencionados arriba.

```
movwf     W_temp          ; guarda W
movf      STATUS,W
clrf      STATUS          ; forza a la página 0
movwf     Status_temp     ; guarda registro de estado
movf      PCLATH,w
movwf     PCLATH_temp     ; guarda PCLATH
movf      FSR,w
movwf     FSR_temp        ; guarda FSR
clrf      PCLATH          ;
```

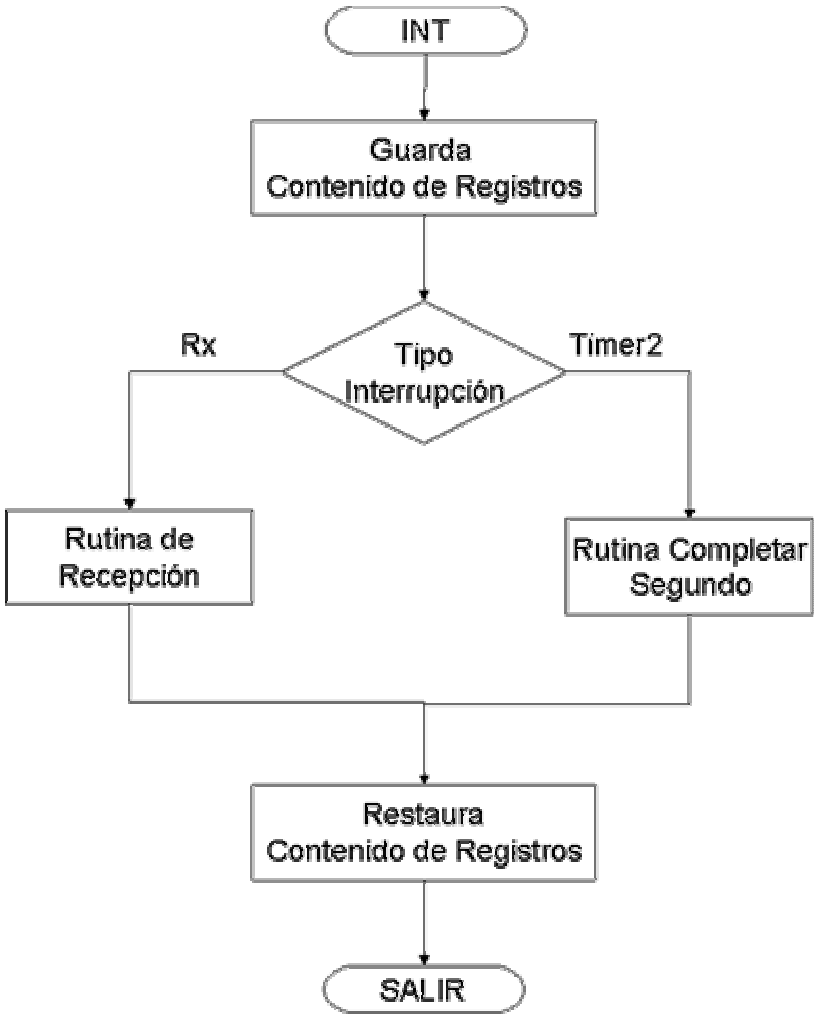


Figura 3-5 Rutina de Servicio de Interrupción

3.7 ÁREA DE MEMORIA PARA EL USUARIO

Por último para concluir este capítulo se presenta un esquema de lo que es el área de memoria en la que podrá trabajar el usuario. Dicha región de memoria está implementada a partir de la dirección 1000h hasta la 1FFFh, es decir el usuario tendrá una capacidad de 4096 casillas de memoria para escribir sus aplicaciones. En la figura 3-6 se muestra el mapa de la memoria de código, donde se ubica cada uno de los bloques funcionales del microcontrolador.

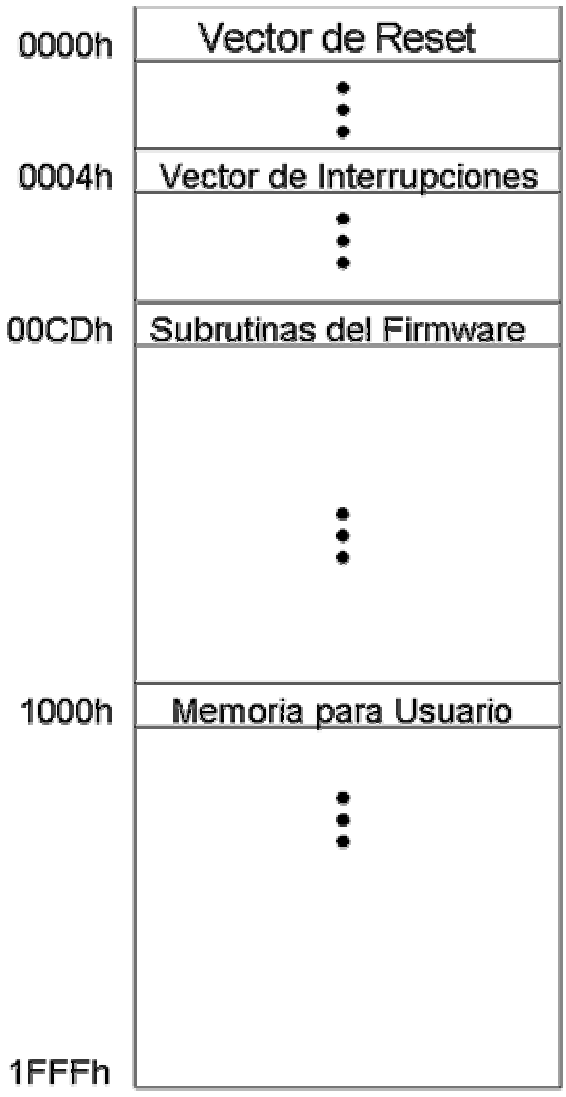


Figura 3-6 Mapa de la Memoria de Código

En este punto hay que notar que la memoria de programa puede ser segmentada mediante la directiva de compilación ORG, en otras palabras nosotros podemos obligar al compilador a que ensamble ciertas líneas de código en la dirección de memoria que se nos antoje. En este caso se ha establecido que el código que escriba el usuario de esta tarjeta el compilador lo ensamble a partir de la dirección 1000h con la siguiente línea:

```
#define        TAREAS       ORG  0x1000
```

Como se puede observar se hace uso de otra directiva de compilación #define, que hace una sustitución de texto, es decir cada vez que se escriba la palabra TAREAS será como escribir ORG 0x1000, lo cual el compilador interpretará como ensamblar código siguiente a partir de esta dirección. Como se explicó unas líneas mas arriba para escribir en la memoria de código había que enviar tanto el código de la instrucción como su dirección, esas direcciones y códigos son generados en la compilación y transmitidos por el PC hasta el microcontrolador pero esto se detallará un poco más en el capítulo siguiente.

CAPITULO 4

DISEÑO DEL SOFTWARE DEL PC

4.1 INTRODUCCIÓN

En este capítulo se tratará el tema del software que deberá realizar tareas como decodificar un archivo en formato hexadecimal Intel de ocho bits, extraer las instrucciones y direcciones donde deben ser grabadas, y con la subrutina adecuada deberá enviarlas serialmente hasta el microcontrolador También presenta un entorno para editar el código del usuario en lenguaje Ensamblador permitiéndole grabar sus trabajos, ensamblarlos para generar el archivo hexadecimal antes mencionado y generar un reporte de los errores que han ocurrido durante el ensamblado. Al final se podrá ver un menú de ayuda donde se presenta los pasos necesarios para utilizar el producto.

4.2 DIAGRAMA DE FLUJO GENERAL

El diagrama de flujo general del software del proyecto es presentado en las figuras 4-1a hasta 4-1f. En la primera figura se presenta las diferentes opciones de menús con que cuenta el programa, dejándose el desglose para mostrarlo en las cuatro figuras restantes. Siendo así que en la figura 4-1b se muestra las diferentes opciones que presenta el menú archivo, como son las funciones: nuevo, abrir, guardar, guardar como y salir. En la figura 4-1c se puede observar la opción para seleccionar el puerto de comunicaciones que se encuentra en el menú configuración. Siguiendo por la figura 4-1d podemos ver las opciones que realizan las tareas de compilar y transferir la información hasta el microcontrolador en el menú opciones, dejando para ser mostrado en la figura 4-1e una lista de las ventanas que han sido abiertas en el menú ventana. Por último se puede encontrar las

opciones acerca de TESIS_RAPI, acerca del autor y contenido en el menú ayuda mostrado en la figura 4-1f

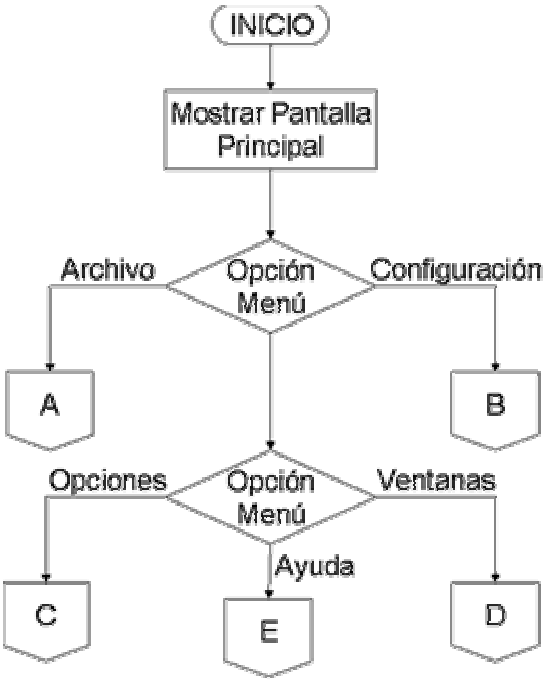


Figura 4-1a Diagrama de Flujo General.

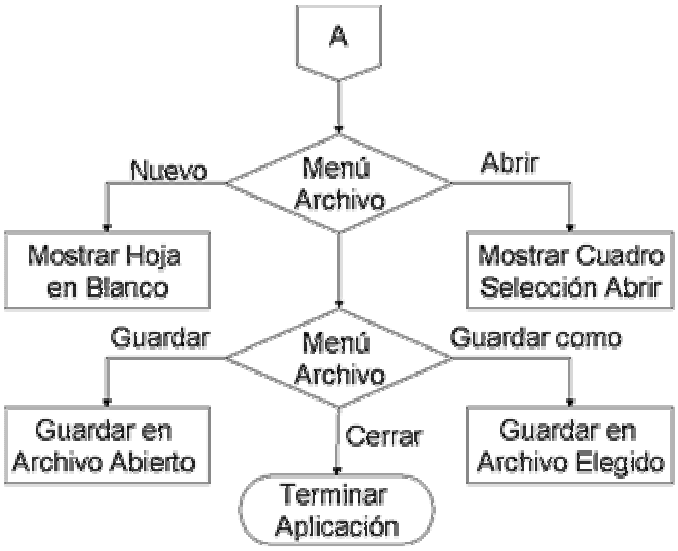


Figura 4-1b Diagrama de Flujo General. Opciones del menú Archivo

Figura 4-1c Diagrama de Flujo General. Opción del menú Configuración

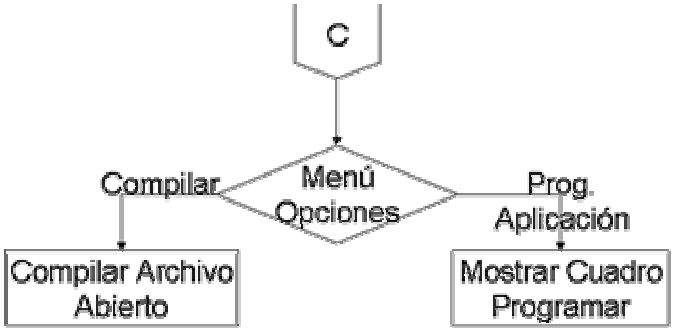


Figura 4-1d Diagrama de Flujo General. Opciones del menú Opciones

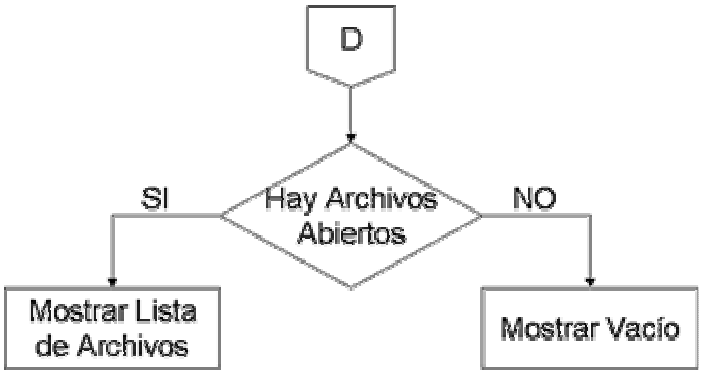


Figura 4-1e Diagrama de Flujo General. Menú Ventana

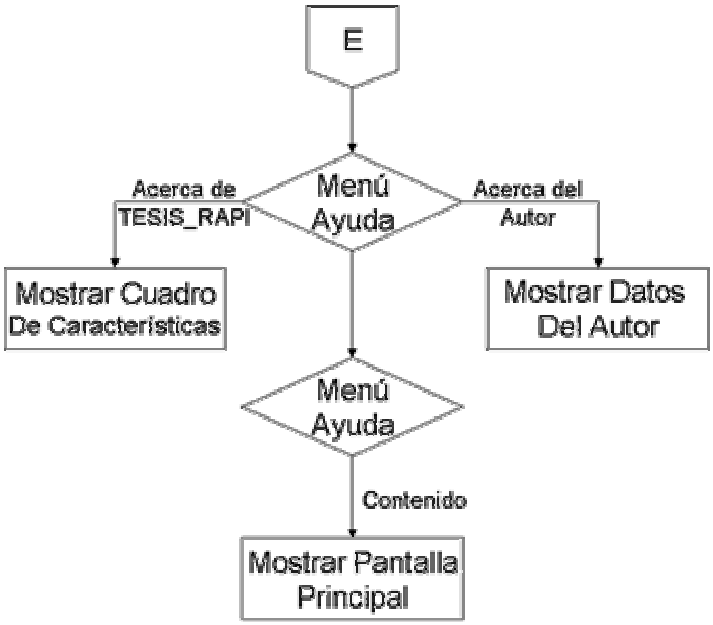


Figura 4-1f Diagrama de Flujo General. Opción del menú Ayuda

4.3 SELECCIÓN DEL PAQUETE DE PROGRAMACIÓN

Al plantear el desarrollo de un programa de computadora que sirva como interfaz entre la tarjeta y el usuario, lo primero que se pensó es que este debería ser amigable y muy fácil de usar; así que surgió la necesidad de hacerlo en un entorno gráfico compatible con la plataforma mas aceptada en el mercado. Partiendo de éstas premisas se tomaron en consideración varias alternativas para poder elegir la que mas convenga a nuestros intereses. A continuación se presenta una tabla con las diferentes opciones que se tomaron en cuenta.

PROGRAMA	PLATAFORMA
Visual C++	Windows
Visual Basic	Windows
C++	Windows
Java	Varias

Tabla 4-1 Lista de Programas tomados en cuenta

En la tabla 4-1 se pueden ver algunos paquetes de programación que sirvieron para la elección del que se usó para esta aplicación. Como se puede observar todos estos programas corren bajo Windows; esto se decidió en función de que es el sistema operativo mas utilizado y que todas las personas que tengan acceso en nuestro medio a una computadora seguramente están familiarizados con el entorno Windows.

Desde este punto de vista la elección debía ser orientada hacia una herramienta con entorno gráfico que herede las características del sistema operativo en mención, lo cual hace que nos alejemos de la posibilidad de utilizar C++, que aunque si se pueden generar este tipo de pantallas no es tan simple hacerlo. Siguiendo la senda de la simplicidad se puede encontrar obstáculos con el Visual C++, que aunque se pueden desarrollar menús y pantallas del tipo de Windows muy fácilmente con la ayuda del asistente, por otro lado el código de funcionalidad, como manejo de archivos y la decodificación del archivo hexadecimal se tornaban en tarea un tanto titánicas al momento de implementarlo. Hasta el momento nos queda solo la opción de Java y de Visual Basic, siendo este último la alternativa que escogí porque a pesar de que Java corre sobre varios sistemas operativos entre ellos Windows, hay que codificar cada uno de sus elementos. Por otro lado el Visual Basic es una herramienta muy fácil de usar, en el que se puede programar en base objetos ya diseñados por el fabricante y que solamente hay que darle ciertas propiedades de acuerdo al gusto del programador. Claro está, también hay que generar código de funciones y procedimientos propios de la aplicación pero esto se hace en lenguaje Basic que es uno de los lenguajes más fáciles de aprender y usar.

4.4 DESCRIPCIÓN DEL PROGRAMA

Básicamente el programa esta formado por cinco menús desplegables, que muestran las diferentes tareas que se pueden ejecutar; por ejemplo en el menú de archivo se puede observar que hay las opciones para crear un nuevo archivo .ASM de código de usuario, se lo puede guardar o abrir uno existente. En el menú de configuración se da la posibilidad al usuario para cambiar el puerto COM de la PC que se quiere utilizar, además uno tiene la oportunidad de compilar el código y luego transferir este hasta el microcontrolador con las opciones que se dan en el menú opciones. El menú ventanas inicialmente está vació y en este se van agregando los títulos de las ventanas nuevas o del reporte de los errores de la compilación, y por último se puede tener una referencia del programa y una pequeña guía de cómo utilizar este en el menú ayuda.

4.4.1 EDITOR DE ARCHIVOS FUENTES

Por la necesidad de independizar un poco este sistema de las herramientas tradicionales de Microchip decidí implementar una hoja para que el usuario pueda escribir directamente su código fuente aquí sin necesidad de buscar algún editor de textos o utilizar el entorno integrado MPLAB. Se puede tener acceso a este recurso haciendo clic en la opción nuevo del menú archivo.

Cabe indicar que el código de usuario antes de ser ensamblado se concatena con el código del firmware del microcontrolador para su posterior procesamiento, por lo que deben tener en cuenta las siguientes observaciones al momento de escribir el código de usuario. Primeramente no hay que incluir los archivos de cabecera, que normalmente se lo hace con la directiva #include ya que estos están considerados en la parte de código que no es accesible para el usuario,

y la otra observación es terminar siempre con la directiva end. Un ejemplo de lo sencillo que sería desarrollar código de usuario es el siguiente:

```
nop
nop
nop
nop
movlw  55
movwf  PORTB
nop
nop
nop
END
```

4.4.2 COMPILACIÓN CON MPASM Y MPLINK

La compilación es el proceso de traducir los mnemónicos del lenguaje, para nuestro caso ensamblador en código que pueda ser interpretado por la máquina que para nosotros es el microcontrolador. Para realizar esta tarea se plantean varias opciones, entre ellas construir nuestro propio compilador utilizando para ello los códigos de operación de cada una de las instrucciones mas la implementación de las directivas de compilación comunes. Pero todo esto es innecesario si se cuenta con herramientas muy buenas como las que provee gratuitamente el fabricante de los PIC en su entorno de desarrollo integrado MPLAB.

Para llevar a efecto el proceso de compilación Microchip ha diseñado dos herramientas que pueden usarse juntas o por separado, el MPASM y MPLINK. La primera aplicación es en si el ensamblador que traduce los mnemónicos a código hexadecimal que luego serán cargados en el PIC, y la segunda lo que hace es ordenar en memoria los fragmentos

de código que se generan cuando se escribe el código fuente en varios archivos. En la figura 4-2 se muestra un archivo siendo ensamblado con MPASM

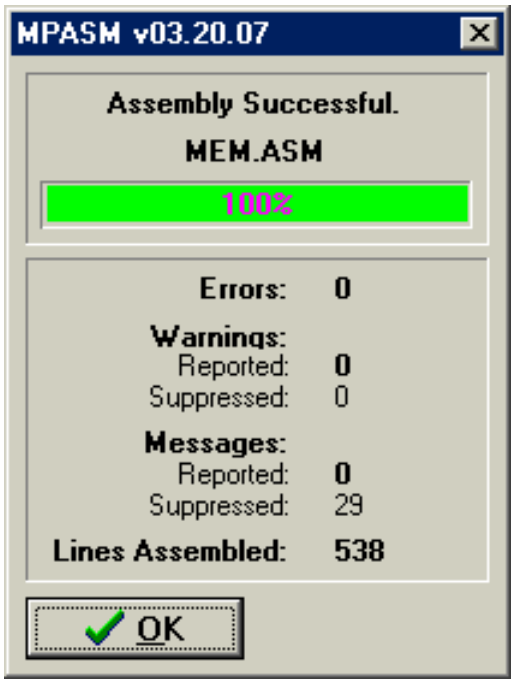


Figura 4-2 Compilación con MPASM

Vale la pena explicar aquí que el firmware del microcontrolador fue codificado en varios archivos .ASM, lo que significa que el compilador debe enlazar todos estos archivos y generar un único código hexadecimal para cargarlo en el PIC, lo cual como ya se mencionó es hecho con el MPLINK. En la figura 4-3 se puede ver el procesamiento de archivos con el MPLINK.

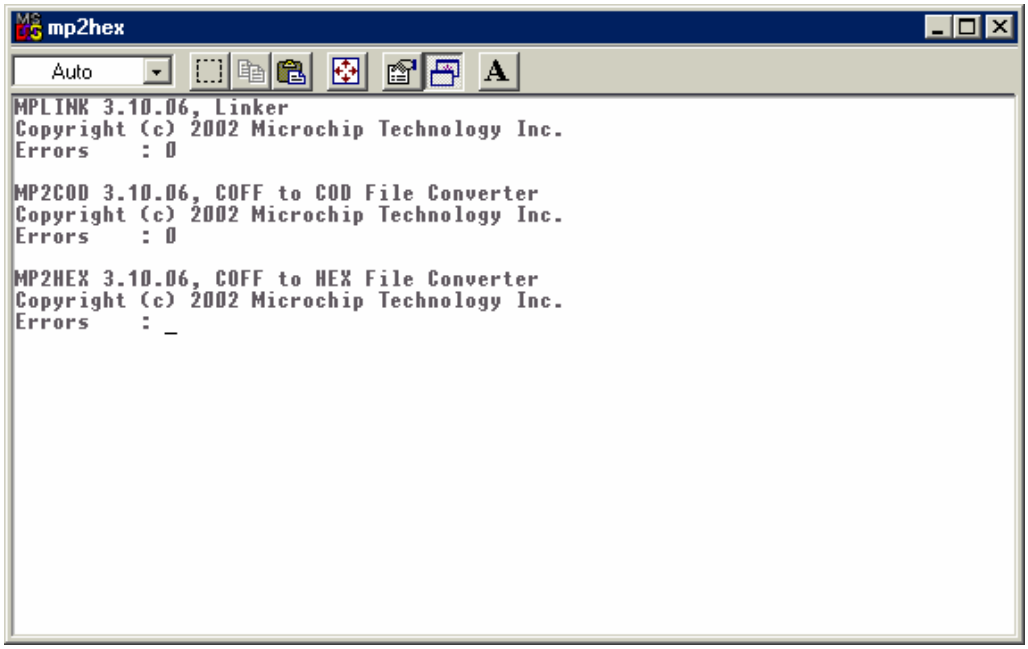


Figura 4-3 Procesamiento de archivos con MPLINK

La estrategia que se ha utilizado en este proyecto es llamar desde el programa principal tanto al MPASM como al MPLINK cuando se hace clic en la opción compilar del menú opciones y se le pasa los parámetros necesarios que entre otros son el nombre del archivo fuente generado por el usuario y los archivos que forman parte del firmware a los que el usuario no tiene acceso.

La forma de utilizar esta opción es abrir un archivo fuente con la opción abrir del menú archivo, o crear uno nuevo con la opción nuevo del mismo menú. Una vez que se tenga el código fuente listo bastará solamente con hacer un clic en la opción compilar del menú opciones y se verá como aparece la consola del MPASM y se muestra el proceso de la compilación, luego veremos la consola del MPLINK haciendo su trabajo. Terminado esto aparecerá una ventana mostrando los errores de compilación que hubiera. Si no se ha producido error en la compilación, debe generarse el archivo hexadecimal con el mismo nombre que el del archivo fuente.

4.4.3 TRANSFERENCIA DEL ARCHIVO HEXADECIMAL HASTA EL PIC

Cuando se ha terminado el proceso de compilación estaremos en capacidad de transferir el archivo hexadecimal generado hasta el microcontrolador. Dicho archivo se lo puede encontrar en la carpeta HEX que se abre por defecto cuando se hace clic sobre la opción Prog Aplicación del menú opciones, y tendrá por nombre el mismo que el del archivo fuente. Esto se puede observar en la figura 4-4 que se muestra a continuación

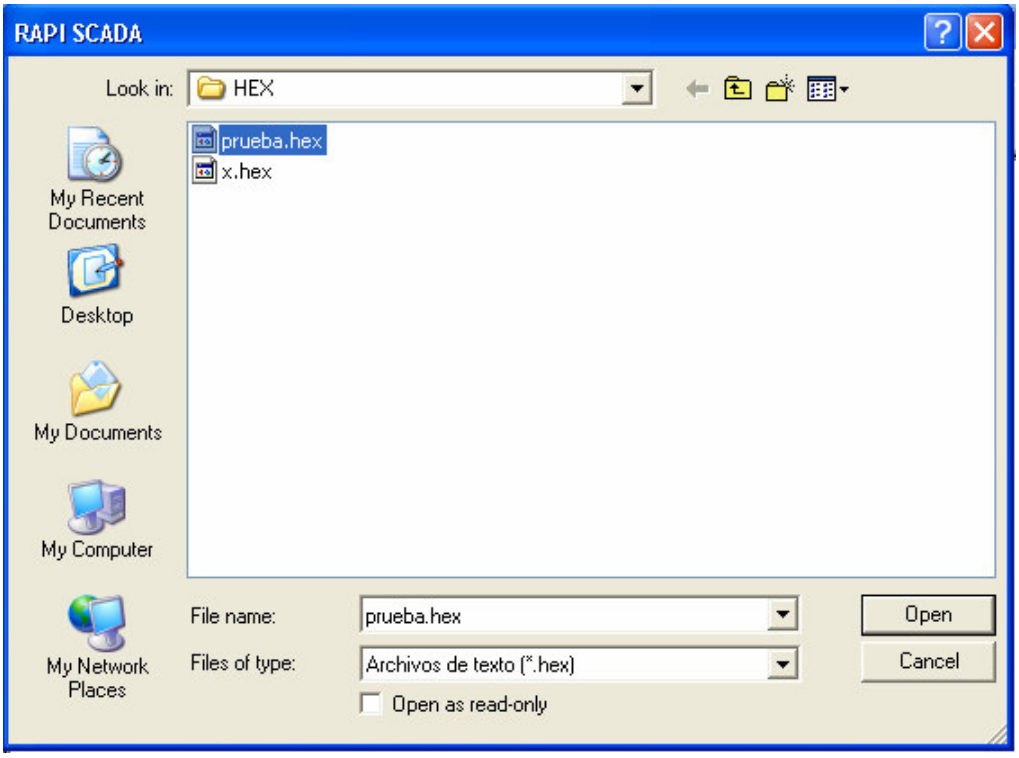


Figura 4-4 Cuadro abrir archivo hexadecimal. Se abre con la opción Prog Aplicación

En realidad la transferencia del código hexadecimal hasta el microcontrolador implica una serie de procesos que detallo a continuación. Primeramente se ejecuta la función manipular_archivo(), que lee el archivo .hex, lo decodifica, genera la tabla de datos con sus respectivas direcciones de memoria y selecciona la porción de este

archivo que será enviada hasta el microcontrolador, es decir los datos que se deben escribir a partir de la dirección 1000h. Cada tarea es ejecutada por un procedimiento especializado, por ejemplo, tabla_gen es el procedimiento que genera la tabla que contiene las instrucciones y la dirección de memoria correspondiente para cada una de ellas, que han sido decodificadas del archivo hexadecimal formato Intel que produjo el compilador. El procedimiento seleccionar_datos se encarga de tomar de la tabla únicamente los datos que deben colocarse desde la dirección 1000h hasta el final del archivo.

Inmediatamente se llama a la función programar_aplicacion(), quien ejecuta los procedimientos tanto para abrir como para cerrar el puerto de comunicaciones de la PC, y además envía todos los datos que necesita el microcontrolador para realizar la tarea que se le pide, es decir, le envía el valor del registro FLAG para la autoprogramación de la memoria de código, cada uno de los datos con sus respectivas direcciones y por último el valor del registro FLAG para fin de tarea que el PIC lo traducirá como Reset del sistema. A continuación se muestra el código para algunas de estas funciones.

```
Public Function manipular_archivo()
Dim cadena_hex As String
   MDI_Principal.Dialog1.FileName = ""
   MDI_Principal.Dialog1.InitDir = "C:\ASM_RAPI\HEX\"
   MDI_Principal.Dialog1.DefaultExt = "hex"
   MDI_Principal.Dialog1.Filter = "Archivos de texto (*.hex)|*.hex"
   MDI_Principal.Dialog1.ShowOpen
   Archivo = MDI_Principal.Dialog1.FileName
   If Archivo <> "" Then
      Open Archivo For Input As #1
      i = 1
      Do While Not EOF(1)
```

```
        ReDim Preserve linea(i)
        Line Input #1, cadena_hex
        linea(i) = cadena_hex
        i = i + 1
      Loop
      Close #1
      tabla_gen
      generar_mem
      seleccionar_datos
   End If
End Function

Public Function seleccionar_datos()
   Inicio = 4096    '
   fin = 8192      '
   m = 1

   For i = Inicio To fin
    ReDim Preserve valor_listo(m)
    valor_listo(m).ADD_H(0) = tabla_dat(i).ADD_H(0)
    valor_listo(m).ADD_L(0) = tabla_dat(i).ADD_L(0)
    valor_listo(m).D_H(0) = tabla_dat(i).D_H(0)
    valor_listo(m).D_L(0) = tabla_dat(i).D_L(0)
    m = m + 1
   Next
End Function

Public Function programar_aplicacion()
  flag(0) = 128
  final(0) = 8
   abrir_puerto
   For i = 1 To 256
```

```
        MDI_Principal.com_serial.Output = (flag)
        MDI_Principal.com_serial.Output = (valor_listo(i).D_H)
        MDI_Principal.com_serial.Output = (valor_listo(i).D_L)
        MDI_Principal.com_serial.Output = (valor_listo(i).ADD_H)
        MDI_Principal.com_serial.Output = (valor_listo(i).ADD_L)
    Next
        MDI_Principal.com_serial.Output = final
    cerrar_puerto
End Function
```

4.4.4 PRESENTACIÓN DE ERRORES DE COMPILACIÓN

Cuando alguien escribe un código de algún programa lo más probable es que se cometan errores involuntarios, por supuesto, como omitir algún caracter necesario para que lo pueda interpretar bien el compilador. Debido a esto se me ocurrió agregar al programa un formulario que muestre los mensajes de error que genera el programa compilador cuando esté procesando el archivo del usuario, estos errores son vistos en el programa como se muestra en la figura 4-5.

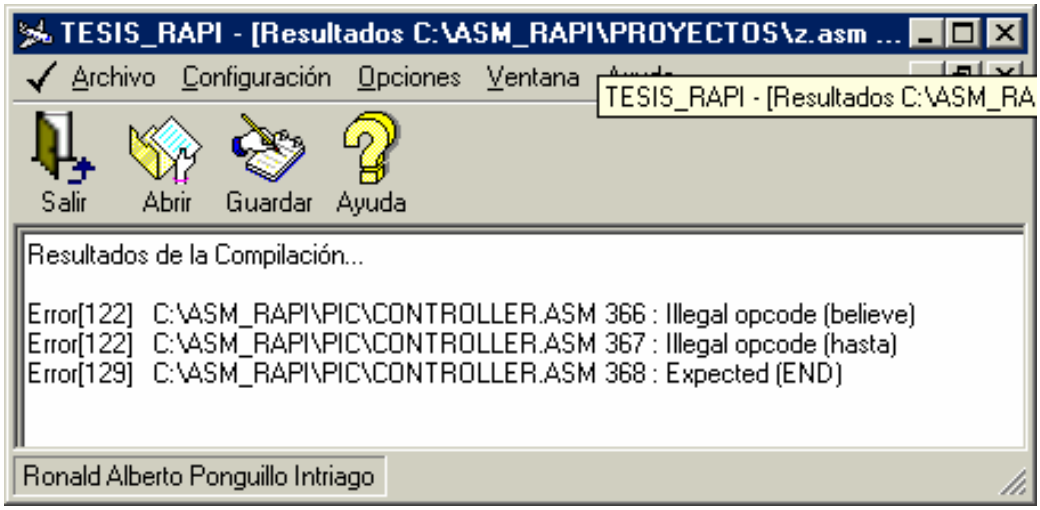


Figura 4-5 Mensajes de error

4.4.5 AYUDA

Para concluir este capítulo haré una reseña de lo que se podrá encontrar en el menú de ayuda del programa. Como se muestra en la figura 4-6 el menú ayuda contiene tres opciones, Acerca de TESIS_RAPI que muestra alguna información sobre el programa como el nombre del mismo, número de versión y una leyenda sobre los derechos del proyecto como se puede ver en la figura 4-7. También se considera algunos datos sobre el autor en la opción Acerca del Autor. Por último en la opción contenido se podrá encontrar un resumen de las opciones que posee el software, así como información relacionada con el uso de éste y la tarjeta.

Figura 4-6 Opciones del menú Ayuda

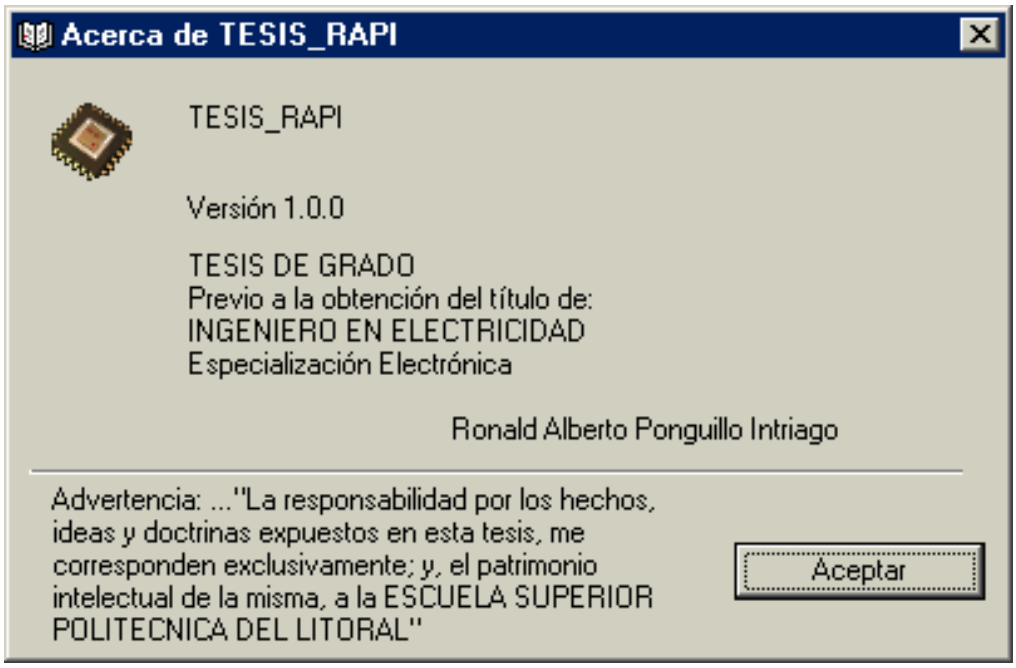


Figura 4-7 Cuadro Acerca de TESIS RAPI

CAPITULO 5

IMPLEMENTACIÓN DEL SISTEMA

5.1 MATERIAL FOTOGRÁFICO

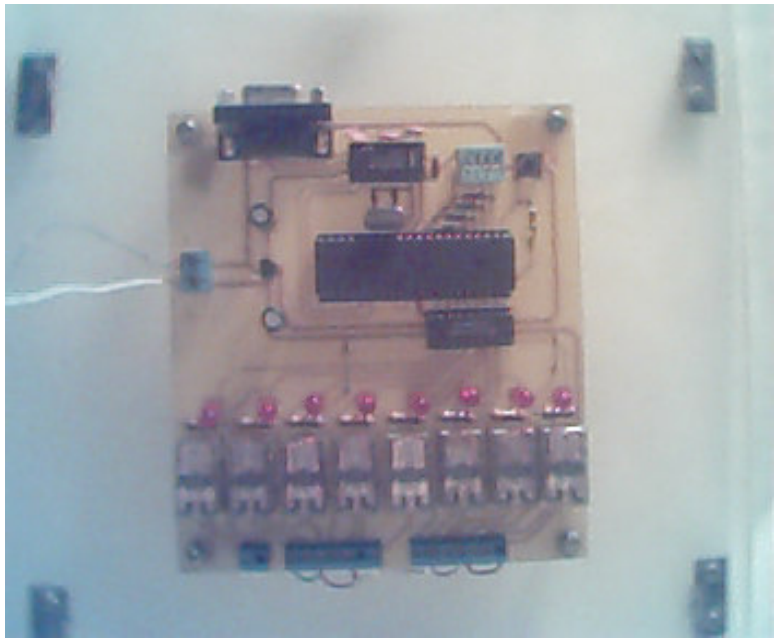


Figura 5-1 Foto de la Tarjeta Controladora

Figura 5-2 Foto del Cable de Comunicación

5.2 PRUEBAS REALIZADAS

En esta parte se hicieron pruebas tanto en el software como en el hardware, obteniendo algunos resultados que se detallan en sus respectivas secciones. La necesidad de realizar pruebas en diferentes ambientes radica en poder establecer cuales podrán ser las condiciones de trabajo del equipo, condiciones mínimas y condiciones críticas. Se debe tener en cuenta que las pruebas realizadas al hardware presentadas aquí son el resultado de la simulación de algunas partes en el entorno integrado MPLAB y no las pruebas reales hechas al equipo. La razón para hacer esto es que los resultados arrojados por el simulador de PIC siempre son lo más parecido a lo que pasa en la realidad y esto disminuye el tiempo de desarrollo en los proyectos. Por otro lado la tolerancia a la temperatura y otros factores ambientales se los puede estimar a partir de los datos técnicos de los elementos. En el software si se hicieron varias pruebas en diferentes equipos y con sistemas operativos variados, para dar una idea del funcionamiento del programa en distintas condiciones.

5.2.1 SIMULACIÓN DE ALGUNAS PARTES EN MPLAB

En este punto se mostrará la simulación de algunas partes del código del microcontrolador realizadas en el entorno de desarrollo integrado MPLAB.

En términos generales en el simulador se puede ejecutar el programa fuente línea por línea e ir viendo como van cambiando los registros del microcontrolador asociados con el proceso. Otra de las opciones que ofrece la simulación es la posibilidad de ver el tiempo que gasta cada instrucción, un procedimiento o cualquier segmento de código que se quiera. Si se quisiera saber como respondería el microcontrolador a entradas externas de tipo digital solamente el simulador integrado

MPSIM tiene una opción que permite introducir estos estímulos externos desde una pequeña ventana. También se tiene acceso a la pila, donde se puede ver la dirección del contador del programa que se almacena aquí en cada llamada a subrutina.

En la figura 5-3 podemos observar el entorno de desarrollo integrado MPLAB, donde se muestran la ventana del código principal, ventana de pila, de memoria de programa y una ventana llamada watch en la que se pueden observar como cambian los valores de los registros a medida que se va corriendo la simulación.

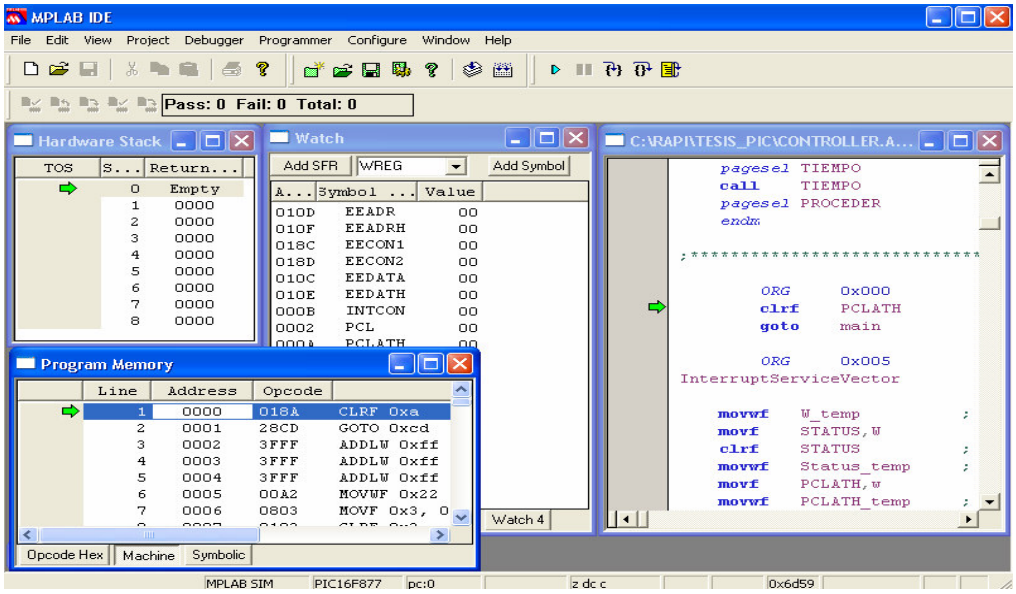


Figura 5-3 Opciones del Entorno de Desarrollo MPLAB

Para probar el funcionamiento de las entradas digitales antes de hacerlo en la tarjeta, utilicé la opción estímulos que se encuentra en el menú depurador, y coloqué un interruptor virtual en el pin RA2, un nivel de voltaje fijo alto en RA3 y un bajo en RA4 tal como se puede ver en la figura 5-4.

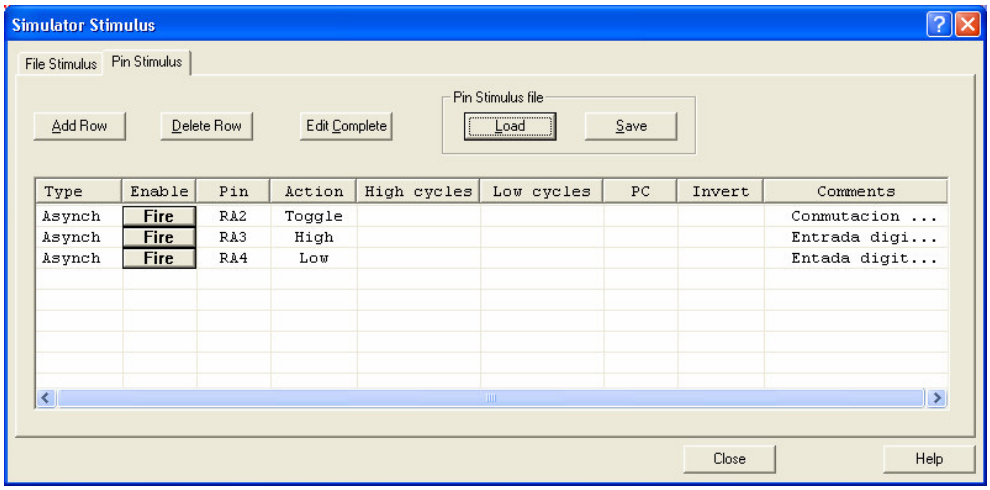


Figura 5-4 Estímulos digitales en entradas del Puerto A

Una vez colocados estos estímulos se puede ir corriendo el programa paso a paso y al llegar a la línea en que se quiere analizar la reacción del PIC ante una entrada digital, con el ratón se hace clic en el botón con la etiqueta Fire.

Una de las cosas que hay que tomar en cuenta, sobre todo cuando se han programado muchas subrutinas e interrupciones es revisar la pila, ya que la pila viene implementada en el hardware y está limitada a ocho niveles. Debido a que la pila es una memoria de tipo LIFO (el último en entrar es el primero en salir) y además tiene un funcionamiento circular, es decir, luego que llega al final el siguiente dato lo va a escribir en la primera posición, hay que tener cuidado de no desbordar la pila porque al salir de alguna subrutina, se tomaría una dirección equivocada de retorno y el flujo del programa se perdería.

En la figura 5-5 se puede ver que, en el momento en que el programa llama a la subrutina, en la pila se carga la dirección de retorno de la subrutina y en el contador del programa la dirección de la subrutina llamada.

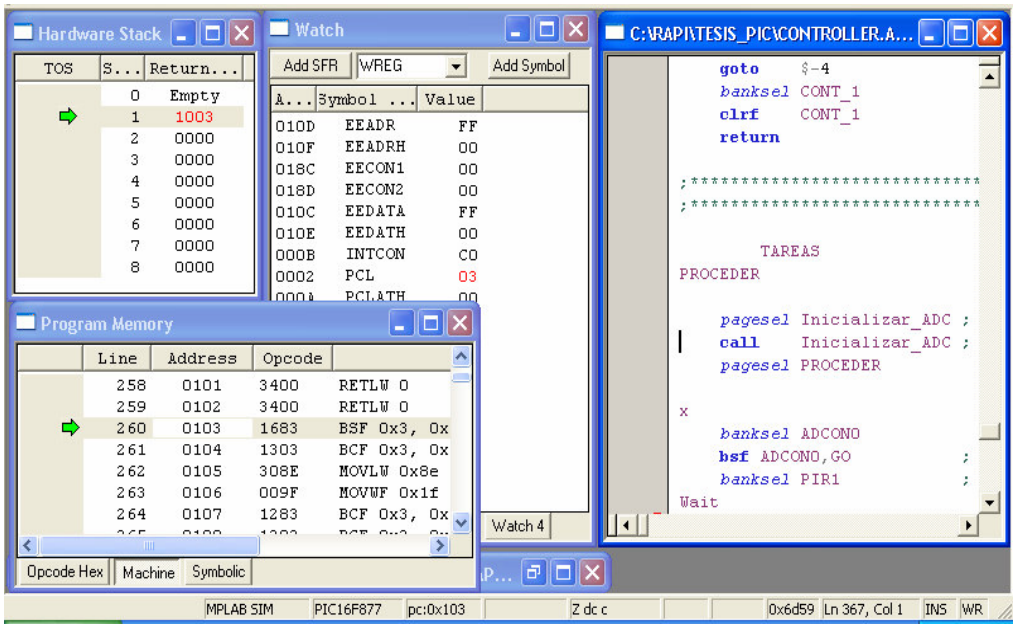


Figura 5-5 Estructura de funcionamiento de la Pila

Para verificar que la subrutina que genera el retardo de un segundo lo está haciendo adecuadamente, existe en el menú ver la opción inspección, que permite ver el tiempo que ocupa cada instrucción en el programa. Con esta herramienta hice un seguimiento de la subrutina que permite la temporización, y el valor que dio el simulador es 998 milisegundos.

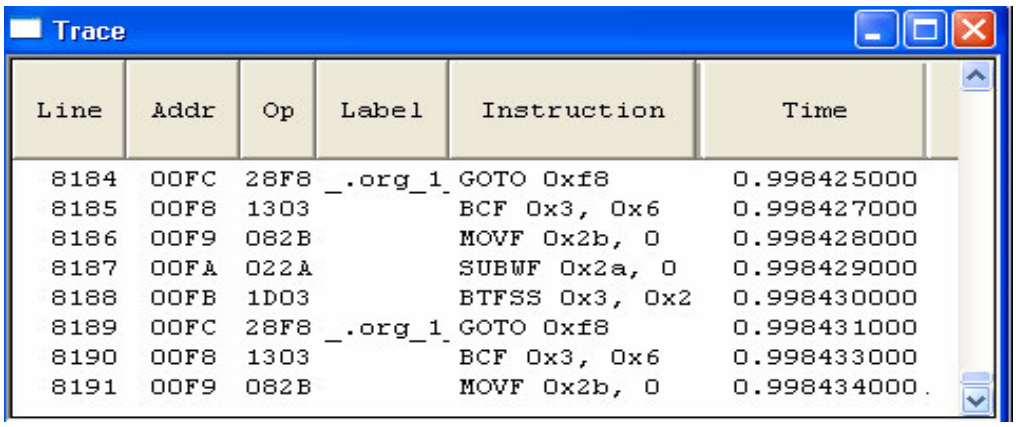


Figura 5-6 Ventana de Inspección

Es necesario aclarar que el simulador ejecuta las instrucciones del programa, pero no lo hace en tiempo real, esto significa que la

velocidad de la simulación puede variar de una PC a otra. Sin embargo los resultados son tan parecidos a lo que ocurre en la realidad que es muy confiable.

5.2.2 PRUEBAS HECHAS EN EL SOFTWARE

Con lo que respecta al software se lo ha instalado y corrido en varias máquinas con diferentes versiones de Windows. Estas pruebas se resumen en la siguiente tabla.

CARACTERISTICAS DEL PROCESADOR	SISTEMA OPERATIVO	OBSERVACIONES
AMD Atlon XP 2GHz	Windows XP	Funcionamiento OK condicionado[a]
Intel Pentium MMX 166MHz	Windows Me	Funcionamiento OK
Intel Pentium III 448MHz	Windows XP	Funcionamiento OK condicionado[a]
Intel Pentium 4 2GHz	Windows XP	Funcionamiento OK condicionado[a]
Intel Pentium 4 1.7GHz	Windows Me	Funcionamiento OK
Intel Pentium 133MHz	Windows 98	Funcionamiento OK

Tabla 5-1 Resumen de las Pruebas hechas al Software

[a] Cuando se instala el programa TESIS_RAPI, se copian algunas aplicaciones de Microchip como el MPASMWIN, MPLINK,_MPLINK, MP2COD, MP2HEX en la carpeta C:\ASM_RAPI. Estas aplicaciones fueron desarrolladas para correr bajo el sistema operativo MS-DOS, por lo tanto tiene restricciones al momento de correr en Windows XP. Así, si el administrador no ha habilitado la compatibilidad con versiones

anteriores de Windows, corresponde al usuario hacerlo. Es muy sencillo, si se instala TESIS_RAPI sobre Windows XP, ubíquese en la carpeta C:\ASM_RAPI y seleccione los archivos mencionados arriba; entones haga clic con el botón derecho del ratón y habilite la opción compatibilidad con Windows 98, Windows Me.

Aunque parece obvio he creído necesario hacer la siguiente observación:
El software fue programado con Visual Basic 6.0 que es un programa que genera código de 32 bits, por lo tanto la aplicación solamente correrá en máquinas con sistemas operativos que soporten 32 o mas bits, esa es la razón por la que no se realizaron pruebas bajo Windows 95 por ejemplo.

5.3 COSTOS

Para terminar con el capítulo se presenta una tabla con la lista de los materiales utilizados y sus respectivos precios en el mercado local.

№	COMPONENTE	VALOR/UNIDAD	VALOR/TOTAL
4	Capacitor 0.01uF	0,15	0,60
1	Resistor 1Kohm	0,05	0,05
1	Capacitor 2.2uF	0,10	0,10
1	Cristal 4.000MHZ	0,80	0,80
8	Resistor 10Kohm	0,05	0,40
8	Relay 12VSPDT	1,50	12,00
1	Regulador de Voltaje 7805	0,70	0,70
8	Diodo emisor de luz Rojo	0,10	0,80
2	Borneras de 7	1,20	2,40
1	Conector DB9 hembra	1,75	0,75
1	PIC16F877	14,00	14,00
1	C.I. MAX232ACPE	6,00	6,00
1	Botonera	0,20	0,20
1	C.I. ULN2803	2,75	2,75
1	Tarjeta de Circuito Impreso	9,80	9,80
1	Cable serial RS232	3,00	3,00
1	Fuente de 12VDC, 1A	7,00	7,00
1	Sócalo 40 pines DIP	0,45	0,45
1	Sócalo 18 pines DIP	0,12	0,12
1	Sócalo 16 pines DIP	0,10	0,10
1	Banco de switches de 2	0,80	0,80
		TOTAL	**$62,82**

Tabla 5-2 Lista de Precios

CONCLUSIONES Y RECOMENDACIONES

A pesar que el proyecto fue concebido y construido originalmente para trabajar con entradas digitales, existe la posibilidad de implementar entradas analógicas también. Este tema no ha sido considerado en el desarrollo de ésta tesis debido a que ya se había hecho la propuesta y el alcance del mismo. En el transcurso de la realización del proyecto recibí varias sugerencias acerca de si se podía implementar dicho tipo de entradas en el prototipo ya construido, sin necesidad de elaborar otra tarjeta con las modificaciones del caso, así que emprendí el análisis de esta situación.

Después de analizar el caso pude llegar a la siguiente conclusión:
Como el PIC que se ha utilizado en este trabajo posee un convertidor analógico/digital con ocho canales, y además, las cuatro entradas digitales que se implementaron coinciden con cuatro de los ocho canales analógicos, si se puede implementar también entradas analógicas en la tarjeta.

Cabe indicar, como es obvio, que si se usa un pin determinado para entrada digital, este no podrá usarse como entrada analógica y viceversa; de esta manera se puede tener hasta cuatro entradas analógicas si se programa la opción para usar referencia de voltaje interna.

Otra cosa que hay que tener en cuenta es que en el firmware del microcontrolador no se ha implementado la subrutina necesaria para darle esta capacidad a la tarjeta, ni el software tiene la posibilidad de recibir e interpretar los datos digitalizados. Pero aún así el usuario podría tranquilamente hacer alguna aplicación que implique la conversión de datos de analógico a digital y visualizarlos por algún medio que podría ser un monitor de computadora.

Aunque en teoría se puede tener temporizaciones con valores múltiplos de uno, menores o igual a 255 segundos, sugiero en el caso que se quiera hacer algún retardo de varios minutos que se utilicen lazos dentro del programa de usuario que llamen a la función TIME que realiza la temporización.

Otra sugerencia que yo haría a todos aquellos lectores de este trabajo es que le saquen provecho a cada una de las subrutinas elaboradas aquí y sobre todo se fijen en la forma utilizada para programar el PIC, que presenta cada subrutina para manipulación de recursos en un archivo ASM separado y que es mucho mas eficiente a la hora de hacer una depuración que escribir todo el código sobre un solo archivo; además he notado que casi nadie la conoce, o por lo menos no la usan. También se podría intentar hacer que el proyecto corra en otras plataformas como por ejemplo Linux, haciendo una versión del software en Java, si no se quiere programar en C, o en otra aplicación dedicada para entorno Linux. En lo que respecta al hardware, éste es un buen punto de partida para desarrollar una tarjeta que tenga capacidad de trabajar en red, sea esta de tipo industrial o por que no hacerla interactuar con Internet.

APÉNDICE A

FORMATO HEXADECIMAL VARIEDAD INTEL DE 8 BITS

El archivo hexadecimal contiene el programa compilado por el MPASM u otro compilador para PICs. Este archivo es leído por algún programador y es transferido al PIC. El formato de este archivo está documentado en el apéndice A del manual del MPASM (Hex File Format). Aquí explicaré el formato Intel INHX8M que a más de ser el configurado por defecto en el MPASM es el que se utiliza en el proyecto.

Este formato produce un archivo hexadecimal de 8 bits con una combinación de byte bajo - alto. Como cada dirección contiene solo 8 bits, todas las direcciones están duplicadas. Cada registro de datos comienza con un prefijo de 9 caracteres y termina con 2 caracteres de checksum. Cada registro tiene el siguiente formato:

:BBAAAATTHHHHHH....HHHHCC

Donde:

: Es el caracter dos puntos, y representa el comienzo de un registro.

BB Es un byte de dos dígitos hexadecimales que representan el doble de la cantidad de bytes de datos que contiene la línea.

AAAA Es una dirección de cuatro dígitos hexadecimales que representan la dirección de comienzo del registro de datos (multiplicada por dos).

TT Es un byte de dos dígitos hexadecimales que siempre es '00', excepto en el registro de final de archivo, en donde es '01'.

HH..HH Son pares de bytes de dos dígitos hexadecimales, de la forma byte bajo - byte alto, y representan los datos a grabar en el PIC.

CC Es un byte de dos dígitos hexadecimales que representan el checksum de verificación de errores del registro de datos. Se calcula como el complemento a dos de la suma de todos los bytes anteriores en el registro.

Para aclarar un poco esto propongo unos ejemplos:

Ejemplo 1. Considere la siguiente línea de un archivo .HEX

:10 0600 00 0A14 8A14 8207 2E34 3B34 2D34 2A34 3134 B0

El primer carácter siempre es **:** , los caracteres siguientes el doble de la cantidad de datos, los cuatro caracteres que siguen indican la dirección duplicada donde se escribirá el primer dato de este registro, luego hay dos caracteres 00 que indica que es una línea de datos o 01 que determina el fin del archivo hexadecimal. A continuación tomamos los datos que están agrupados en cuatro caracteres, siendo los dos primeros el byte bajo de la instrucción y los dos siguientes el byte alto; al final quedan dos caracteres que representan el checksum del registro que es calculado como la suma en complemento a dos de los bytes del registro.

En resumidas cuentas en esta línea de ejemplo dice:

Es una línea de datos que contiene 8 instrucciones que deben ser grabadas a partir de la dirección 300h y la primera tiene el valor 140Ah.

Ejemplo 2. Revisemos ahora el siguiente fragmento de archivo .HEX

:02 400E 00 F13F 80 --> Una sola instrucción en la posición 2007h

:08 0008 00 00008A0117088207BD --> 4 instrucciones a partir de la dirección 0004h

:00000001FF --> Indicador de fin de archivo HEX

APÉNDICE B

DISEÑO DEL PCB

B.1 ELECCIÓN DEL PAQUETE CAD

Una herramienta CAD (**C**omputer **A**ided **D**esign – Diseño Asistido por Computadora) es en los actuales momentos de vital importancia para los ingenieros y demás profesionales cuyos diseños los componen dibujos de gran precisión y complejidad.

Para nuestro caso en particular en la electrónica, existen en el mercado varios programas CAD para el diseño de Tarjetas de Circuito Impreso (PCB), algunos de los cuales formaron parte del grupo de selección para este proyecto, los cuales menciono a continuación:

PRODUCTO	EMPRESA
EAGLE	CadSoft
TRAXMAKER	Grupo Altium
ORCAD	Cadence
PROTEL	Grupo Altium

Tabla B-1 Algunos Paquetes CAD para PCB

De todos los programas mencionados tengo los demos. Para el caso de EAGLE el demo no permite grabar el proyecto si la tarjeta excede un tamaño básico predefinido 10cm x 10cm lo cual lo descalifica automáticamente. El TRAXMAKER que viene en el paquete del CIRCUITMAKER no tiene muchas librerías con los footprints que necesitaba y no da la opción para generar los nuestros. Con el ORCAD en cambio, los resultados son excepcionales, posee gran cantidad de librerías de footprints pero su demo es muy limitado como por ejemplo que no puede guardar un diseño con mas de 60 partes, o guardar

librerías con mas de 15 partes. Por último PROTEL que en mi criterio es el mejor programa para diseño electrónico que ha pasado por mis manos, ya que es bastante amigable y agradable su interfase de usuario, además es muy fácil de usar las herramientas y librerías que posee, como también crear nuevas librerías de usuario. Pero sobre todo lo que le dio mas peso al momento de elegirlo es que su demo viene completamente funcional, aunque tiene la limitación de correr por sólo 30 días ese tiempo es más que suficiente para crear el esquemático y el PCB, guardarlo e imprimirlo.

B.2 TARJETA DE CIRCUITO IMPRESO

Al finalizar el trabajo el resultado de la tarjeta de circuito impreso es el siguiente:

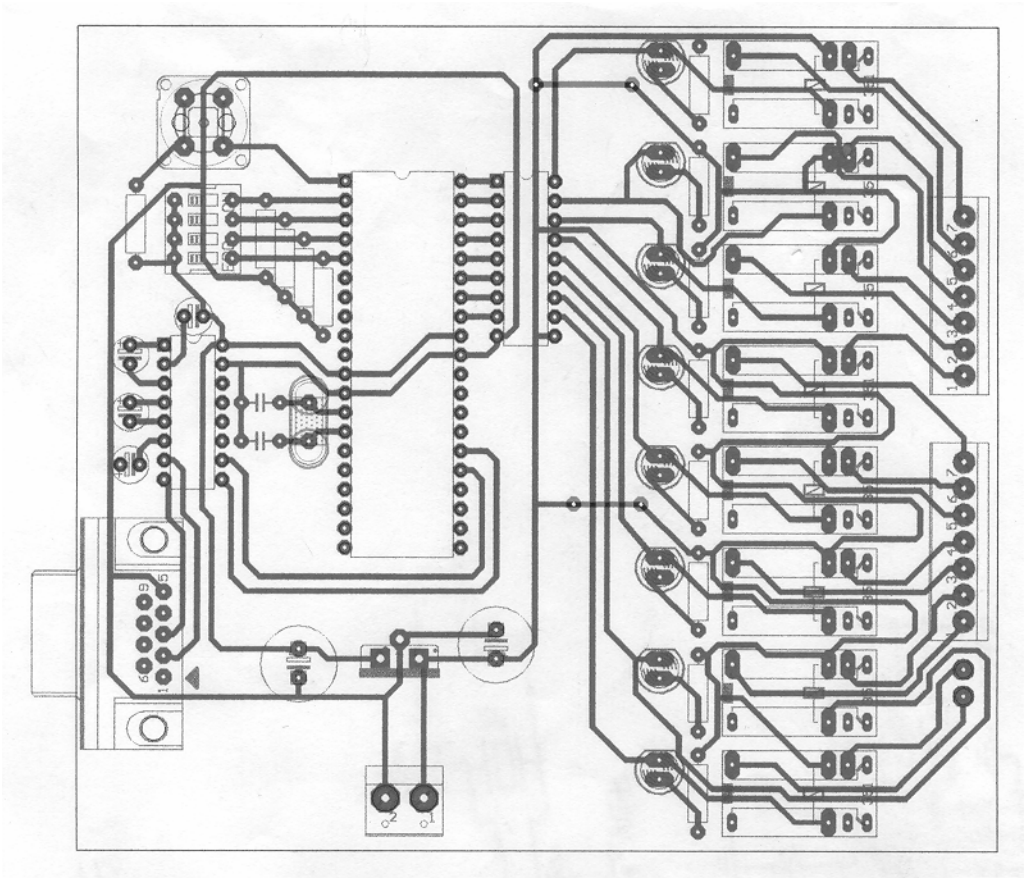


Figura B-1 Tarjeta de Circuito Impreso. Lado de Componentes

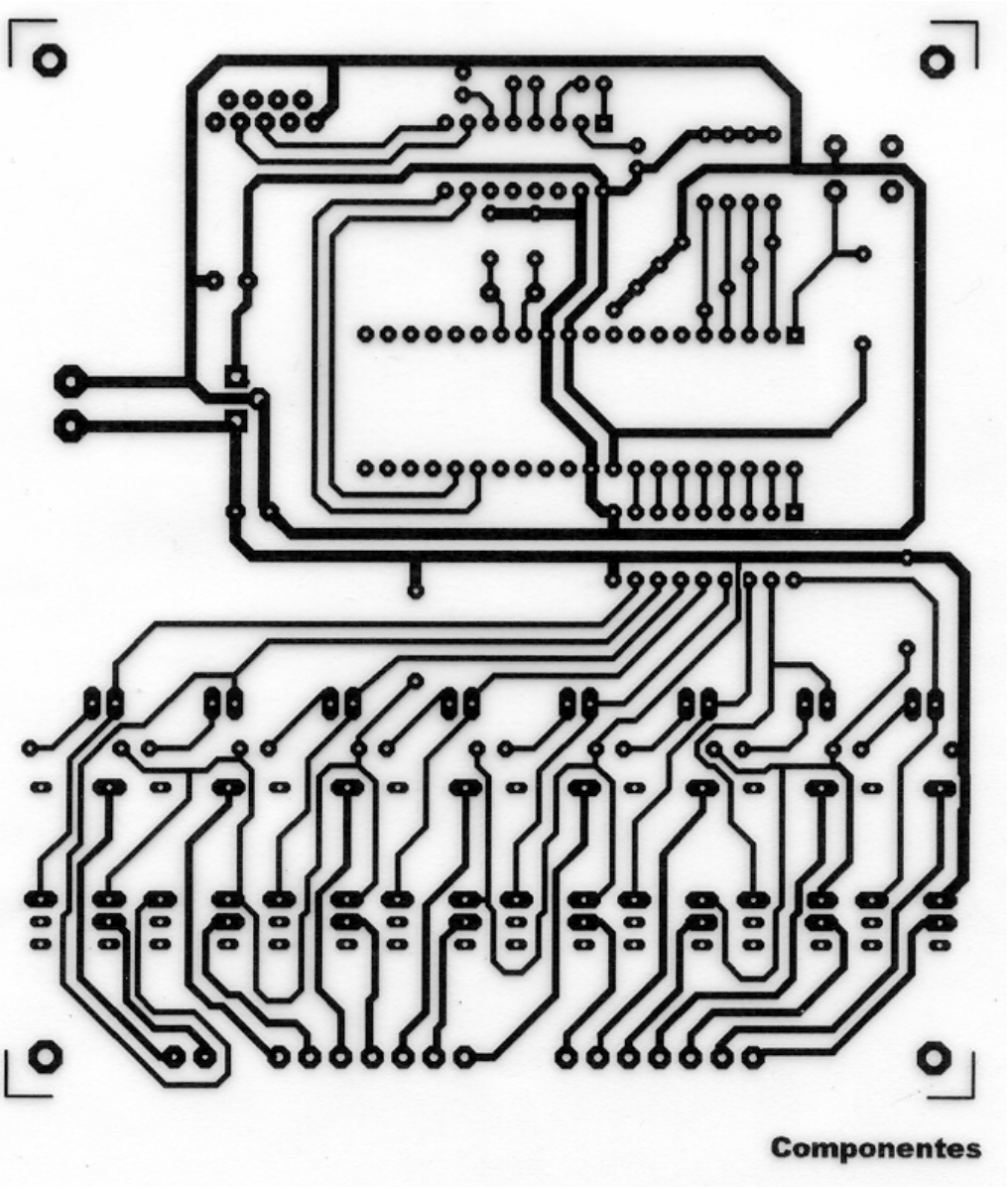


Figura B-2 Tarjeta de Circuito Impreso. Lado de las Pistas

APÉNDICE C

CARACTERISTICAS DEL PIC 16F877

Características del núcleo del Microcontrolador:

- CPU RISC de alto desempeño
- Sólo 35 instrucciones de una palabra para aprender
- Todas las instrucciones son de un ciclo excepto las de bifurcación que son de dos ciclos
- Velocidad de Operación :
  DC - 20 MHz entrada de reloj
  DC - 200 ns ciclo de instrucción
- 8K x 14 palabras de Memoria FLASH de Programa,
  368 x 8 bytes de Memoria de Datos (RAM)
  256 x 8 bytes de Memoria de Datos EEPROM
- Diagrama de Pines compatible con PIC16C73B/74B/76/77
- Capacidad de Interrupciones (hasta 14 fuentes)
- Ocho niveles de pila en hardware
- Modos de direccionamiento directo, indirecto y relativo
- Reset en el Encendido (POR)
- Temporizador de estabilización de la energía (PWRT) y Temporizador de estabilización del Oscilador (OST)
- Temporizador Perro Guardián (WDT) con su propio oscilador RC integrado
- Protección de Código programable
- Modo SLEEP para ahorro de energía
- Opciones para elegir el oscilador
- Memoria de Programa FLASH/EEPROM con tecnología CMOS de bajo consumo y alta velocidad
- Diseño completamente estático
- Programación serial en circuito (ICSP) por medio de dos pines
- Capacidad de programación en circuito con fuente de 5V

- Depuración en circuito a través de dos pines
- Procesador con acceso a leer/escribir memoria de programa
- Amplio rango de voltaje de operación: 2.0V to 5.5V
- Gran manejo de corriente: 25 mA
- Rangos de temperatura Comercial e Industrial
- Bajo cosumo de energía:
  -< 2 mA tipico @ 5V, 4 MHz
  -20 mA típico @ 3V, 32 kHz
  -< 1 mA típico en reposo

Características de Periféricos:

- Timer0: temporizador/contador de 8-bit con prescaler de 8-bit
- Timer1: temporizador/contador de16-bit con pre-escalador, puede ser incrementado durante sleep via cristal/reloj externo
- Timer2: temporizador/contador de 8-bit con registro de periodo de 8 bits, pre-escalador y post-escalador
- Dos módulos de Captura, Comparación, PWM
  -Captura es de 16-bit, máx. resolución es 12.5 ns
  -Comparación es de 16-bit, máx. resolución es 200 ns
  -PWM máx. resolución es de 10-bit
- Convertidor Analógico/Digital multicanal de 10 bits
- Puerto Serial Sincrónico (SSP) con SPI (Modo Maestro) e I 2 C (Maestro/Esclavo)
- Receptor Transmisor Universal Sincrónico Asincrónico (USART/SCI) con detección de dirección de 9-bits
- Puerto Paralelo Esclavo (PSP) 8-bits de ancho, con controles RD, WR y CS externo
- Circuito de detección de falla de energía para Reset por falla de energía (BOR)

CARACTERISTICAS PRINCIPALES	PIC16F877
Frecuencia de operación	DC - 20 MHz
Resets (y Retardos)	POR, BOR (PWRT, OST)
Memoria FLASH de Programa (Palabras de14-bits)	8K
Memoria de Datos (bytes)	368
Memoria de Datos EEPROM	256
Interrupciones	14
Puertas de E/S	Puertas A,B,C,D,E
Temporizadores	3
Módulos de Captura/Comparación/PWM	2
Comunicación Serial	MSSP, USART
Comunicación Paralela	PSP
Módulo Convertidor A/D de 10-bits	8 canales
Conjunto de Instrucciones	35 Instrucciones

Tabla C-1 Características del PIC16F877

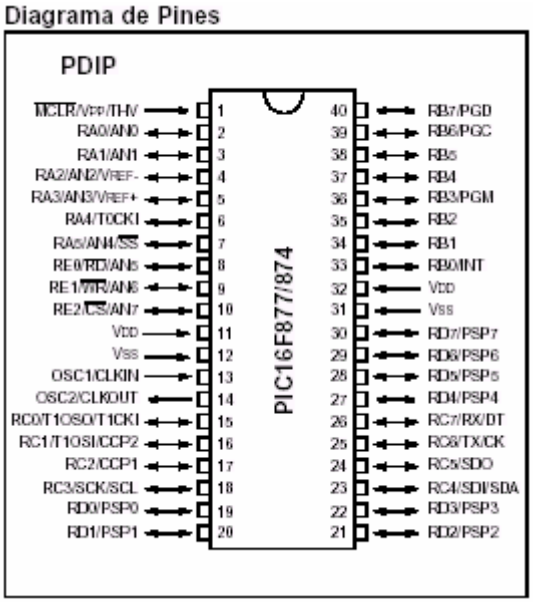


Figura C-1 Diagrama de Pines

APÉNDICE D

MANUAL DEL USUARIO

Este proyecto esta construído en dos partes: el software y el hardware. En la parte del hardware es importante tomar en cuenta las siguientes instrucciones para el uso.

Conecte la fuente de 12VDC en las borneras de alimentación de la tarjeta con la polaridad adecuada.

Conecte el cable de comunicación serial tanto en la terminal de la tarjeta como en la computadora.

Una vez cargado el programa de usuario puede desconectar el cable de comunicación serial.

Utilización del Software

En la parte que tiene que ver con el software hay más consideraciones a tomar en cuenta ya que es aquí realmente donde el usuario va a realizar su trabajo.

Como todo programa basado en Windows, TESIS_RAPI, posee una estructura de ventanas con barra de menú para las opciones de las tareas que se pueden realizar.

Como se puede observar en la pantalla principal del programa existen cinco menús.

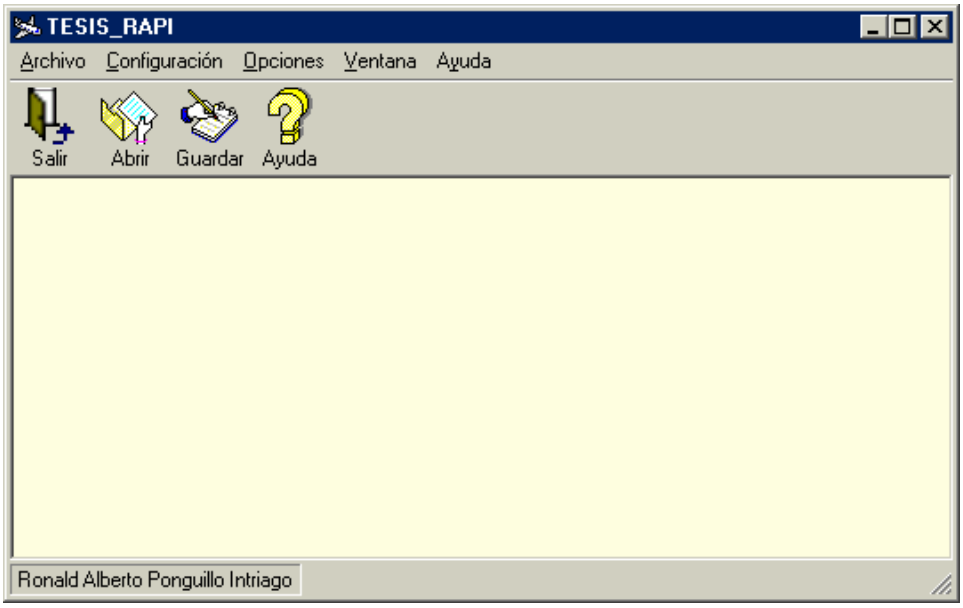


Figura D-1 Pantalla Principal del Programa

A continuación se presentan cada una de las opciones con las que cuenta este software.

Nuevo

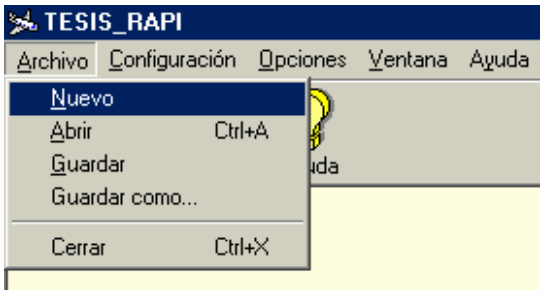


Figura D-2 Opción Nuevo, menú Archivo

La opción Nuevo se encuentra en el menú archivo y permite abrir una hoja en blanco para editar el código de la aplicación de usuario. Se puede tener acceso a esta opción con la combinación de teclas Alt+a, Alt+n.

Abrir

Esta opción permite abrir algún archivo .asm guardado con anterioridad.

Figura D-3 Opción Abrir, menú Archivo

La combinación de teclas para acceso opcional a esta opción es Alt+a, Alt+a. Cuando se selecciona la opción Abrir, aparece un cuadro de dialogo que permite al usuario escoger algún archivo fuente entre los existentes. Dicho cuadro es el siguiente.

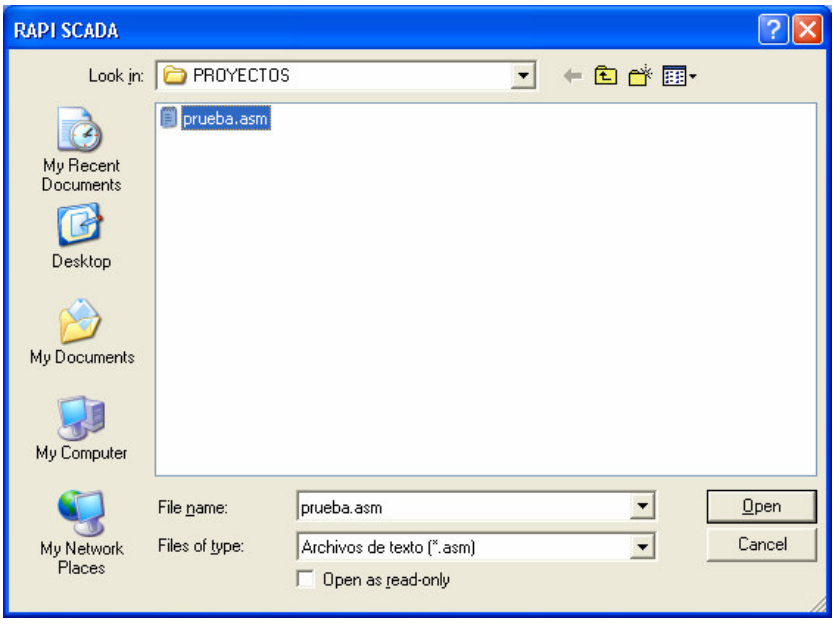


Figura D-4 Cuadro Abrir Archivo

Guardar

Con esta opción se guarda los cambios que se han realizado en un archivo determinado. Puede usarse para guardar los cambios que se van realizando en el archivo hasta terminarlo. En el caso que alguien cree un archivo nuevo, lo esté editando y decida guardar con esta opción, como aún el archivo no tiene nombre se abrirá el cuadro de dialogo necesario para que el usuario le asigne un nombre.

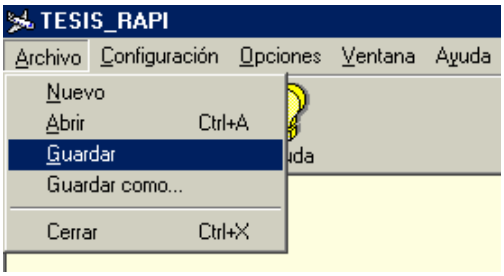


Figura D-5 Opción Guardar, menú Archivo

La combinación de teclas para acceso a esta opción es Alt+a, Alt+g.

Guardar como…

Como la mayoría de los programas bajo Windows, RAPI_TESIS también posee esta opción que da la posibilidad de guardar el archivo con el nombre que el usuario desee, en el directorio o unidad que prefiera mediante un cuadro de dialogo.

Figura D-6 Opción Guardar como.., menú Archivo

Esta opción no posee acceso alterno desde el teclado, así que la única manera de hacerlo es con el ratón.

Cerrar

Es la última opción del menú Archivo, y permite cerrar el programa. Otra forma de cerrar el programa, es haciendo clic en el botón  que se encuentra en la esquina superior derecha de la ventana.

Figura D-7 Opción Cerrar, menú Archivo

Se puede tener acceso rápido a esta opción desde el teclado con la combinación ctrl.+x.

Menú Configuración

En este menú se ha ubicado únicamente la opción para escoger cual de los puertos com del la PC será utilizado.

Puerto Com

Esta opción permite al usuario indicarle al PC en cual de los puertos com esta conectada la tarjeta.

Figura D-8 Opción Puerto Com, menú Configuración

Cabe aclarar que solamente se ha considerado la posibilidad de com1 y com2, ya que son los que mayormente se encuentran en la PC.

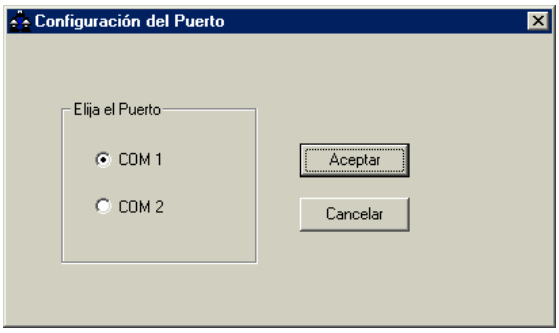


Figura D-9 Cuadro Configuración del Puerto

Cuando se elige esta opción aparece un cuadro de dialogo como el que se ve en la figura, y en el cual con el ratón se hace selección del com que se va a usar y luego hacer clic en el botón Aceptar.

Menú Opciones

En el menú opciones se puede encontrar dos funcionalidades del proyecto, la opción de compilar y la de programar aplicación.

Compilar

Con esta opción se llama a las aplicaciones MPASM y MPLINK para proceder al ensamblaje y construcción del archivo hexadecimal del proyecto que se encuentra actualmente abierto.

Figura D-10 Opción Compilar, menú Opciones

Para tener acceso a esta opción desde el teclado se puede utilizar la combinación Alt+o, Alt+c.

Si no hay archivo abierto y se selecciona esta opción no se ejecutará ningún proceso. En el evento en que se tuviera un archivo fuente listo y se ejecuta compilar aparecerán los siguientes cuadros de dialogo que el usuario debe dejar que terminen de ejecutarse, ya que se cierran automáticamente.

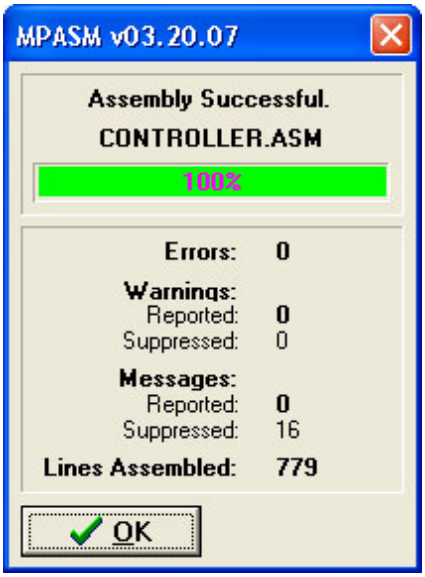


Figura D-11 Cuadro de reporte del ensamblador

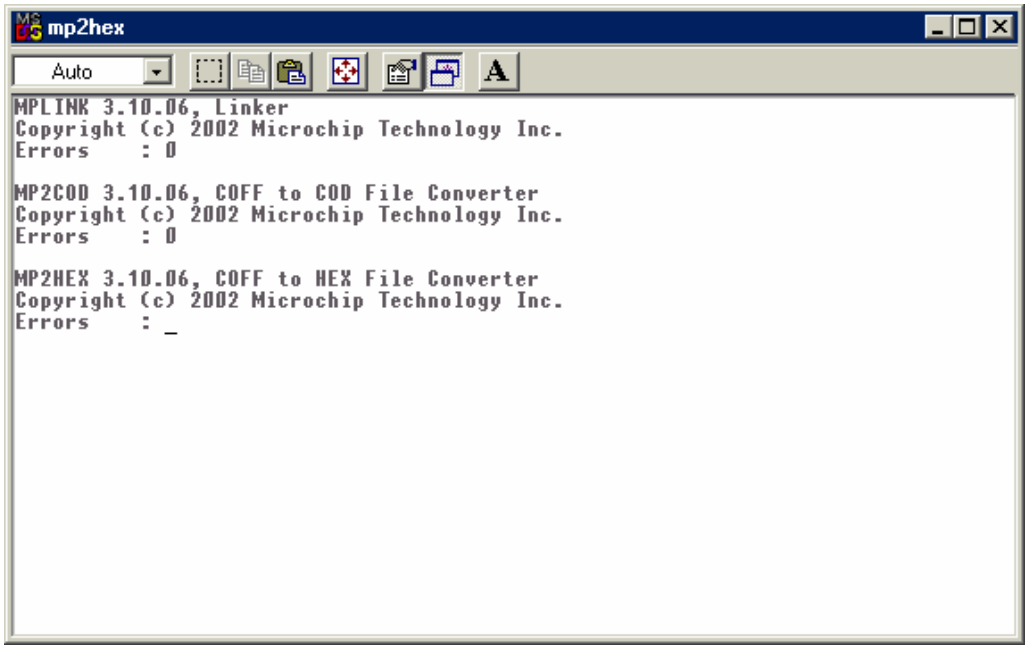


Figura D-12 Cuadro de reporte del enlazador

Al finalizar, cuando estos cuadros se cierran aparecerá una hoja presentando los errores de compilación si los hubiera.

ProgAplicacion

Cuando se selecciona esta opción aparece el cuadro de dialogo que permite elegir el archivo hexadecimal que será transferido hasta el PIC.

Figura D-13 Opción Prog Aplicacion, menú Opciones

Una vez escogido este archivo aparecerá el siguiente cuadro de confirmación.

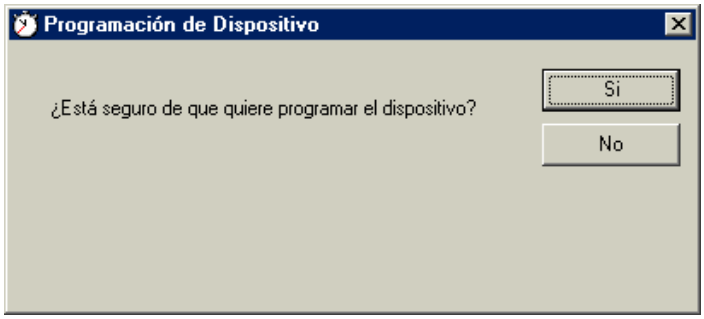


Figura D-14 Cuadro Programación de Dispositivo

Si en este punto hacemos clic en el botón No, el cuadro se cierra y no ha pasado nada, en cambio si elegimos Si se procede con la transmisión de los datos hasta el microcontrolador, y mientras se realiza este proceso se mostrará el siguiente mensaje

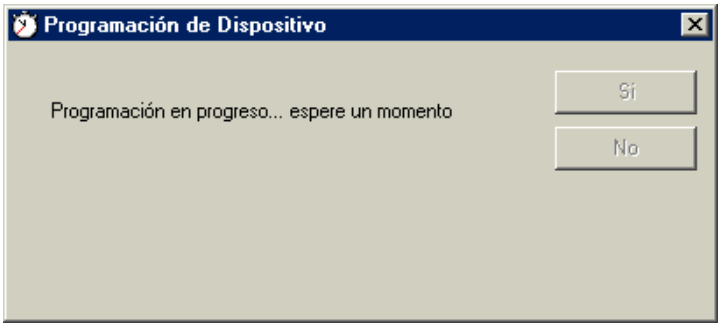


Figura D-15 Cuadro Programación en Progreso

Una vez terminada la comunicación con la tarjeta se genera el mensaje de finalización como se ve en la figura siguiente.

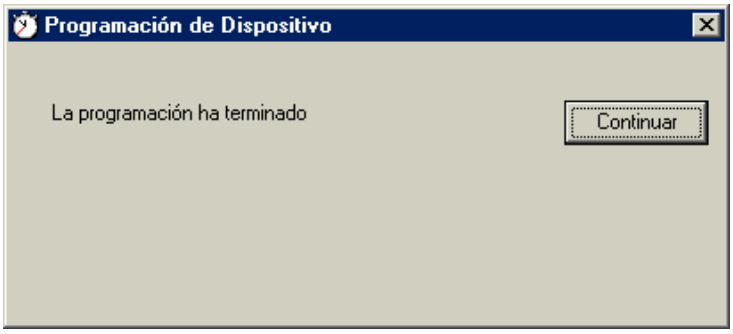


Figura D-16 Cuadro Programación Lista

Ya en este momento la tarjeta tendrá cargado el programa de la aplicación del usuario.

Menú Ventana

El menú ventana inicialmente no presenta ninguna opción y se muestra de la siguiente manera.

Figura D-17 Menú Ventana

Cuando el usuario abre un archivo, el nombre de éste se ubica tanto en la barra de título como en la persiana del menú Ventana de la siguiente manera.

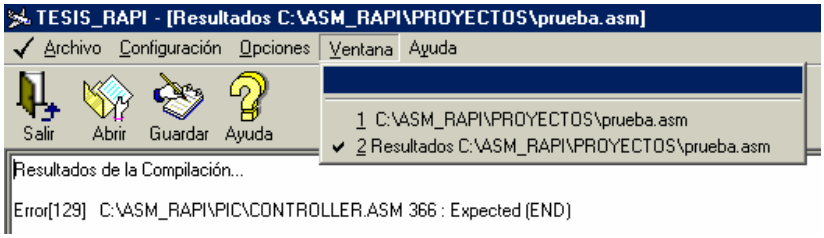


Figura D-18 Opción del menú Ventana

Lo mismo ocurre cuando luego de la compilación se genera el archivo de reportes, se puede visualizar su ruta tanto en la barra de títulos como en el menú Ventana. Este menú Ventana sirve entonces para cambiarse de un archivo a otro sin necesidad de minimizar o cerrar la ventana que tiene el enfoque del programa, solamente haciendo clic en el nombre de éste dentro del menú Ventana.

Menú Ayuda

Para seguir con la estructura de los programas basados en Windows, se ha colocado un menú de Ayuda con tres opciones.

Acerca de TESIS_RAPI

La opción Acerca de TESIS_RAPI, muestra un cuadro de dialogo con información acerca del programa, como número de versión, procedencia y derechos.

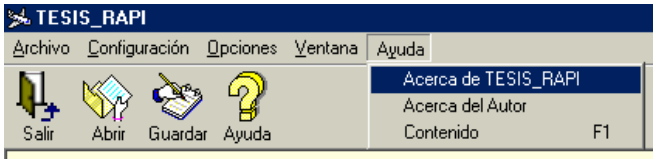


Figura D-19 Opción Acerca de TESIS_RAPI, menú Ayuda

Al hacer clic sobre esta opción aparece el siguiente cuadro de dialogo.

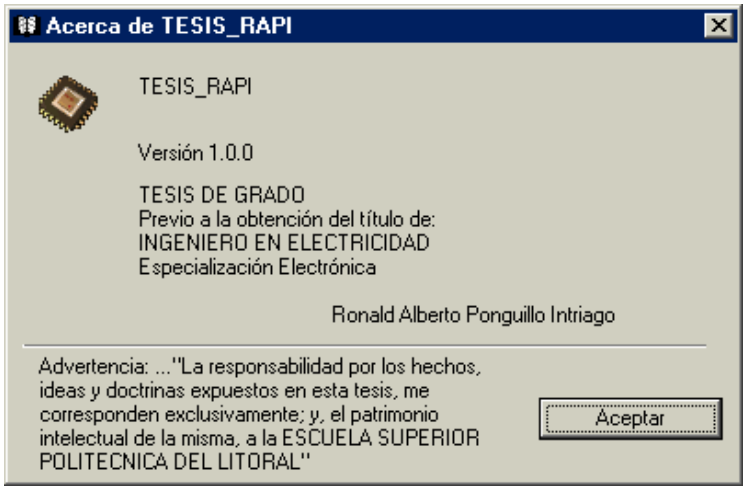


Figura D-20 Cuadro Acerca de TESIS_RAPI

Acerca del Autor

En esta opción se presenta alguna reseña del autor.

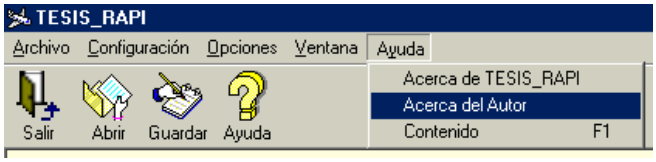


Figura D-21 Opción Acerca del Autor, menú Ayuda

Contenido

Haciendo clic sobre esta opción o accediendo a ella por medio de la tecla F1, se despliega en la pantalla un archivo de ayuda típico de los programas basados en Windows. En éste archivo se encontrará información acerca de las opciones del programa y algunos datos sobre como elaborar los archivos fuentes para aplicaciones de usuario.

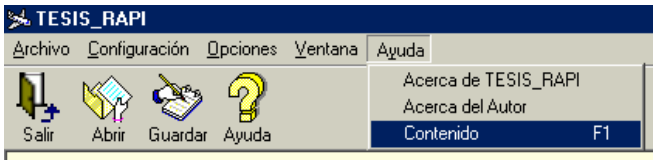


Figura D-22 Opción Contenido, menú Ayuda

Además se presenta algunos datos sobre la tarjeta y su uso.

GLOSARIO

ASM

Lenguaje Ensamblador. Mnemónicos que representan las instrucciones o código de máquina que instruyen al procesador para ejecutar determinadas labores.

CISC

Complex **I**nstruction **S**et **C**omputer. Chip procesador central que puede ejecutar un gran número de instrucciones, cada una de las cuales realiza una tarea completa; se compara con un procesador RISC, que tiene pocas instrucciones que se ejecutan mas rápidamente, aunque son programas más complejos.

COMPILADOR

Software que traduce un lenguaje de alto nivel a código de máquina.

CONVERTIDOR ANALÓGICO-DIGITAL (ADC)

Circuito que convierte una entrada analógica en su correspondiente salida digital.

CONVERTIDOR DIGITAL-ANALÓGICO (DAC)

Circuito que convierte una entrada digital en su correspondiente salida analógica

FIRMWARE

Programa de computador o datos que se almacenan permanentemente en un chip.

HARDWARE

Unidades físicas, componentes, circuitos integrados, discos y mecanismos que integran un computador o sus periféricos.

MICROCONTROLADOR

Circuito Integrado que contiene una unidad central de proceso CPU, memoria para programa y datos, temporizador, puertas de entrada/salida y demás elementos que sirvan para realizar circuitos de control.

MULTIPLEXADO

Proceso de selección de una entrada entre varias y transmisión de los datos seleccionados hacia un solo canal de salida.

MULTIPLEXOR

Circuito lógico que, dependiendo de los estados de sus entradas de selección, lleva uno de sus varios datos de entrada a su salida.

PIC

Microcontroladores fabricados por la empresa Arizona Microchip Inc.

PROTOTIPO

Primer modelo operativo de un programa o un dispositivo para su prueba y adaptación.

RISC

Reduced **I**nstruction **S**et **C**omputer. Diseño de un procesador, cuyo conjunto de comandos está compuesto por un pequeño y rápido número de instrucciones, que hacen más compleja la escritura de programas pero incrementan la velocidad de ejecución.

SOFTWARE

Programa o grupo de ellos que indica al equipo como operar y reaccionar ante diferentes eventos. El término incluye sistemas operativos, programas y aplicaciones.

TEMPORIZADOR

Circuito contador que es utilizado para generar períodos de mayor duración que el del reloj aplicado a éste.

TRANSMISIÓN ASINCRÓNICA

Método de transmisión que usa bits de arranque y bits de parada para regular el flujo de datos.

TRANSMISIÓN SINCRÓNICA

Método de transmisión que usa una señal de reloj para sincronizar los computadores qe envían y reciben información, con el fin de regular el fluo de datos, en lugar de usar bits de arranque y parada como en una transmisión asincrónica

USART

Universal **S**ynchronous **A**synchronous **R**eceiver **T**ransmitter. Circuito integrado para comunicación sincrónica o asincrónica serial.

BIBLIOGRAFÍA

Microcontroladores PIC, la solución en un chip
J. Ma. Angulo Usategui, E. Martín Cuenca y J. Angulo Martínez
Editorial Paraninfo, 1997

Aplicaciones de los microcontroladores PIC de Microchip
J. Ma. Angulo Usategui, E. Martín Cuenca y J. Angulo Martínez
Editorial McGRAW-HILL, 1998

Microchip PIC Microcontrollers
Data Book, Microchip Technology Inc.
Microchip, The embedded control solutions company, 1997

VISUAL BASIC - Guía del Estudiante.
Ing. Luis Suárez Bernaldo
San Sebastián de los Reyes (Madrid, España), Junio de 1998

MSDN (Microsoft Developer Network) Library, Visual Studio 6.0
Microsoft Corporation 1991-1998

Diccionario de Multimedia
Silmon Collin
McGRAW-HILL, 1996

DIRECCIONES DE INTERNET

www.lvr.com

Sitio web con información acerca de los puertos de comunicaciones de las PCs

www.epanorama.net

Sitio web con mucha información y enlaces sobre temas de electrónica y telecomunicaciones

www.microchip.com

Sitio web del fabricante de los microcontroladores PIC

www.protel.com

Sitio web del fabricante de la herramienta de desarrollo Protel

http://www.microsoft.com/msdn/

Sitio web de Microsoft Corporation, desarrollador del Visual Studio

http://www.maxim-ic.com/

Sitio web del fabricante del circuito de interfaz MAX232