ESCUELA POLITÉCNICA NACIONAL

ESCUELA DE INGENIERÍA

DISEÑO E IMPLEMENTACIÓN DE UN PLC SOBRE LA BASE DEL MICROCONTROLADOR PIC16F877

PROYECTO PREVIO A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y CONTROL

TANIA RODRÍGUEZ HIDALGO
FABIÁN RODRIGO ROMO RIVERA

DIRECTOR: DR. LUIS CORRALES

Quito, septiembre 2005

DECLARACIÓN

Nosotros, Tania Rodríguez Hidalgo y Fabián Rodrigo Romo Rivera, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que hemos consultado las referencias bibliográficas que se incluyen en este documento.

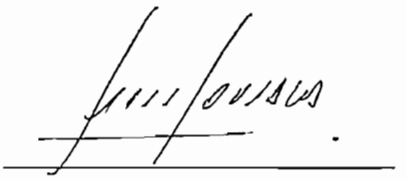
A través de la presente declaración cedemos nuestros derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Tania Rodríguez Hidalgo

Fabián Rodrigo Romo Rivera

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Tania Rodríguez Hidalgo y Fabián Rodrigo Romo Rivera, bajo mi supervisión.



Dr. Luis Corrales

DIRECTOR DE PROYECTO

AGRADECIMIENTOS

Agradecemos a la Escuela Politécnica Nacional, a los profesores del ex ICB y de la carrera de Ingeniería Electrónica y Control que nos guiaron por el camino para ser profesionales exitosos.

Al Doctor Luis Corrales por su acertada dirección y paciencia durante todo el tiempo de este trabajo.

Y a todas las personas y amigos que nos ayudaron y apoyaron de cualquier forma.

DEDICATORIA

Dedico esta Tesis y toda mi carrera a mis padres ya que sin su apoyo y comprensión no lo habría logrado, los amo mucho papi RAMI y mami GLADYS.

A mis hermanas Aleja y Natis que siempre fueron mi inspiración para seguir adelante.

A Fabián mi compañero de Tesis y enamorado quien me ha enseñado a querer la carrera y sin duda la compañía ideal en este trabajo.

A la familia Hidalgo, una Familia de mujeres valerosas que siempre han sido un ejemplo de trabajo y constancia.

Y por último a tres personas que ya no están mas conmigo. A mi Tío Raulito, a mi Tía Marujita y a mi Tía Ceci siempre estarán en mi corazón.

Tania Rodríguez Hidalgo

DEDICATORIA

Dedico este trabajo realizado a mis Padres Segundo y Georgina por todo su amor, apoyo, sacrificio y preocupación. A mi hermano Édison, por su paciencia y colaboración. A Tañita por su constancia ante las adversidades y su cariño en todo momento que trabajamos juntos.

También dedico este trabajo a mi abuelito, por el gran apoyo recibido durante toda mi formación en la Politécnica Nacional.

NEMO PERVENIT QUI NON LEGITIME CERTAVERIT.

Rodrigo Romo.

Fabián Rodrigo Romo Rivera

CONTENIDO

RESUMEN..... I

PRESENTACIÓN..... III

CAPITULO 1. ESTUDIO COMPARATIVO DE LOS CONTROLADORES LÓGICOS PROGRAMABLES..... 1

1.1 OBJETIVO GENERAL..... 1

1.2 DEFINICIÓN DE UN CONTROLADOR LÓGICO PROGRAMABLE (PLC)..... 1

1.3 VENTAJAS Y DESVENTAJAS DE UN CONTROLADOR LÓGICO PROGRAMABLE (PLC)..... 2

1.4 HARDWARE BÁSICO DE UN CONTROLADOR LÓGICO PROGRAMABLE..... 3

1.4.1 FUENTE DE ALIMENTACIÓN..... 4

1.4.2 UNIDAD CENTRAL DE PROCESO (CPU)..... 5

1.4.2.1 Procesador..... 6

1.4.2.2 Memoria..... 6

1.4.3 MÓDULOS DE ENTRADA..... 8

1.4.3.1 Módulos de entradas digitales..... 8

1.4.3.2 Módulos de entradas analógicas..... 9

1.4.4 MÓDULOS DE SALIDA..... 10

1.4.4.1 Módulos de salidas digitales..... 11

1.4.4.2 Módulos de salidas analógicas..... 12

1.4.5 MÓDULOS PERIFÉRICOS..... 14

1.5 VELOCIDAD DE PROCESAMIENTO..... 14

1.6 LENGUAJES DE PROGRAMACIÓN..... 15

1.6.1 DIAGRAMA DE CONTACTOS..... 15

1.6.2 DIAGRAMA DE FUNCIONES..... 16

1.6.3 LISTA DE INSTRUCCIONES..... 17

1.6.4 LENGUAJE SECUENCIAL..... 18

1.6.5 TEXTO ESTRUCTURADO..... 18

1.6.6 CONFIGURACIÓN DE PROGRAMA..... 18

1.7 OPERACIONES PARA PROGRAMACIÓN..... 19

1.8 CARACTERÍSTICAS A CONSIDERAR EN LA ELECCIÓN DE UN CONTROLADOR LÓGICO PROGRAMABLE..... 19
1.8.1 FACTORES CUANTITATIVOS..... 20
1.8.2 FACTORES CUALITATIVOS..... 20
1.9 CARACTERÍSTICAS DEL PLC DISEÑADO PARA ESTE PROYECTO..... 21
CAPITULO 2. ESTUDIO DE LAS CARACTERÍSTICAS DEL MICROCONTROLADOR UTILIZADO COMO CPU..... 22
2.1 ESTRUCTURA Y CARACTERÍSTICAS DEL MICROCONTROLADOR..... 22
2.1.1 CARACTERÍSTICAS FÍSICAS DEL MICROCONTROLADOR 16F877A..... 25
2.1.2 PROCESADOR RISC CON ARQUITECTURA HARVARD..... 26
2.1.3 ORGANIZACIÓN DE LA MEMORIA DE PROGRAMA..... 26
2.1.4 MEMORIA DE DATOS RAM..... 27
2.1.5 SET DE INSTRUCCIONES..... 28
2.2 PROGRAMACIÓN DEL MICROCONTROLADOR PIC16F877A..... 29
2.2.1 MODO DE PROGRAMACIÓN / VERIFICACIÓN..... 33
2.2.2 MODO BORRAR/PROGRAMAR..... 42
2.2.3 MODO SOLAMENTE PROGRAMAR..... 43
2.2.4 BORRADO MASIVO DEL MICROCONTROLADOR..... 44
CAPITULO 3. DISEÑO DEL HARDWARE DEL PLC..... 46
3.1 INTRODUCCIÓN..... 46
3.2 DISEÑO DEL CIRCUITO DE LA UNIDAD CENTRAL DE PROCESO (CPU)..... 47
3.3 DISEÑO DE LOS MÓDULOS DE ENTRADA DIGITALES Y ANALÓGICOS..... 54
3.3.1 DISEÑO DE LAS ENTRADAS DIGITALES..... 54
3.3.2 DISEÑO DE LAS ENTRADAS ANALÓGICAS..... 56
3.4 DISEÑO DE LOS MÓDULOS DE SALIDAS DIGITALES..... 59
3.5 DISEÑO DEL MÓDULO ESPECIAL..... 60
3.5.1 DISEÑO DEL CIRCUITO DE COMUNICACIÓN SERIAL RS-232..... 60

3.5.2 DISEÑO DE LAS SALIDAS PWM..... 61
3.6 DISEÑO DE LA TARJETA DE VISUALIZACIÓN DE ENTRADAS O SALIDAS..... 63
CAPITULO 4. DISEÑO DEL SOFTWARE DE SOPORTE..... 66
4.1 PRÓLOGO..... 66
4.2 INTERFAZ GRÁFICA DE USUARIO..... 68
4.2.1 INTERFAZ GRÁFICA PARA ELABORACIÓN DE PROYECTOS..... 69
4.2.1.1 Barra de Estado..... 69
4.2.1.2 Barra de Comentarios..... 69
4.2.1.3 Barra de Herramientas..... 69
4.2.1.4 Barra de Menús..... 89
4.2.2 INTERFAZ GRAFICA DEL SIMULADOR..... 97
4.2.3 INTERFAZ GRAFICA PARA DESCARGAR AL PLC..... 100
4.3 DESARROLLO DE LOS ALGORITMOS PARA EL MICROCONTROLADOR PIC..... 105
4.3.1 DISEÑO DE LA INTERFAZ GRÁFICA PARA LA PROGRAMACIÓN DEL PLC..... 106
4.3.2 DISEÑO DEL SIMULADOR PARA EL PLC..... 112
4.3.2.1 Funciones Tipo Booleanas..... 112
4.3.2.2 Funciones a Nivel de Bytes..... 121
4.3.2.3 Desarrollo del algoritmo para el Simulador..... 167
4.3.3 DISEÑO DEL PROGRAMADOR DEL MICROCONTROLADOR Y GENERACIÓN DEL CÓDIGO HEXADECIMAL..... 170
4.3.4 DESARROLLO DEL FIRMWARE EN EL MICROCONTROLADOR..... 174
CAPITULO 5. PRUEBAS Y RESULTADOS..... 182
5.1 PRUEBAS..... 182
5.2 RESULTADOS..... 200
CAPÍTULO 6. COMPARACIÓN DE COSTOS..... 202
6.1 COSTO DEL HARDWARE..... 202
6.2 COSTO DEL SOFTWARE..... 207

6.3 COMPARACIÓN CON EL COSTO DE OTROS PLCs..... 209
CAPÍTULO 7. CONCLUSIONES Y RECOMENDACIONES..... 212
7.1 CONCLUSIONES..... 212
7.2 RECOMENDACIONES..... 214
REFERENCIAS BIBLIOGRÁFICAS..... 215

ANEXOS

ANEXO A. Especificaciones de la Programación de la Memoria Flash de los PIC16F87XA

ANEXO B. Especificaciones del Opto aislador NTE3045

ANEXO C. Especificaciones del Driver amplificador de corriente ULN2001A

ANEXO D. Especificaciones del Transistor Amplificador de Potencia B676

ANEXO E. Diseño de las tarjetas del PLC

ANEXO F. Fotografías del PLC desarrollado y de las tarjetas que lo conforman

RESUMEN

El objetivo general del presente proyecto es diseñar y construir un PLC, sobre la base del microcontrolador PIC 16F877A.

El principal trabajo de este proyecto se circunscribió a la elaboración de un software que logre que el microcontrolador PIC se comporte como un PLC. Adicionalmente se desarrollo una interfaz para la elaboración de proyectos en lenguaje FBD, una interfaz de simulación que permite al usuario probar el comportamiento del programa realizado y una interfaz para descargar los proyectos realizados, a través del puerto paralelo.

Un subproducto muy importante que resultó durante el desarrollo de este proyecto, es la generación del código hexadecimal para el microcontrolador PIC, con lo cual se puede programar a éste independientemente del hardware de programación utilizado.

El hardware del PLC desarrollado tiene una estructura modular; es decir, está compuesto por varias tarjetas que pueden ser fácilmente reemplazadas según las necesidades del usuario. En total se diseñaron y construyeron 6 tarjetas: 2 de entradas digitales, 2 de salidas digitales, 1 de entradas analógicas y 1 con 2 salidas PWM más Comunicación Serial RS232. Las tarjetas son intercambiables y pueden usarse en cualquiera de los 4 puertos que posee el microcontrolador, a excepción de la tarjeta A/D que únicamente se puede utilizar en el Puerto A y E y la tarjeta con los PWMs y comunicación serial que solo se puede usar en el Puerto C.

Para protección del microcontrolador, se usó entradas digitales optoacopladas de 24V (1L). Las salidas son de tipo relé normalmente abierto, las entradas analógicas tienen un rango de trabajo de 0 a 10V, las salidas PWM son de 0 a 24 Vdc y la comunicación serial RS232 es de 8 bits, sin bit de paridad.

Todos los circuitos diseñados, simulados y descargados funcionaron de manera apropiada. Y únicamente las funciones temporizador Ton y Toff tienen un rango de error aproximadamente del 2% en el conteo del tiempo.

PRESENTACIÓN

Los microcontroladores son dispositivos muy relevantes en el campo de la electrónica. Actualmente se los utiliza en gran escala por sus diversas aplicaciones y funciones integradas que poseen. El presente documento trata sobre el diseño y la construcción de un PLC sobre la base del microcontrolador PIC16F877A al cual se lo ha dotado de programación en lenguaje FBD o diagrama de funciones. Se utilizó dicho microcontrolador porque resultó ser el más apropiado para cumplir con el rol del procesador de un PLC, siendo además un dispositivo que se lo puede conseguir localmente. El presente documento ha sido dividido en siete capítulos y anexos buscando una organización adecuada de la información

El Capítulo Primero describe las principales características de un PLC comercial, y las ventajas que presenta en la industria como herramienta de automatización. El objetivo de este estudio fue identificar las ventajas y desventajas que caracterizan a un PLC y que se intenta emular con el PIC. También se describe los diversos lenguajes de programación de los PLCs. Se termina este capítulo con las características generales que se dotaron al PLC diseñado y construido en este proyecto.

En el Capítulo Segundo se indica las principales ventajas del microcontrolador usado, así como los aspectos técnicos que deben ser tomados en cuenta para la programación de éste. Se presenta su arquitectura interna y el funcionamiento de sus distintos algoritmos de programación, etc.

El Capítulo Tercero menciona los pasos que se siguieron para el diseño y construcción del hardware del PLC partiendo de un diagrama de bloques hasta terminar en un equipo sobre todo modular.

El Capítulo Cuarto describe el desarrollo del software de programación del PLC que fue elaborado en Microsoft Visual Basic y que está compuesto por tres HMIs.

La primera es la interfaz gráfica de usuario, la segunda es la interfaz de simulación para predecir el comportamiento del PLC y finalmente la interfaz para programar al PLC vía puerto paralelo o generación del código hexadecimal para el microcontrolador PIC utilizado. Se añaden los diagramas de flujo de todas las interfaces antes mencionadas.

En el Capítulo Quinto se presenta los resultados de las pruebas realizadas con el PLC, para lo cual se presenta varios ejemplos en FBD, explicando brevemente el funcionamiento de cada uno. Este capítulo es de importancia para aquellas personas que no están interesadas en detalles de diseño o de construcción y solamente deseen resultados concretos.

En el Capítulo Sexto se hace una comparación de costos con el fin de establecer un valor para saber si el PLC desarrollado y construido es competitivo en el mercado nacional.

En el último capítulo y en base de las pruebas que se hicieron al PLC, se concluye con las ideas mas importantes del presente proyecto y además se recomienda las futuras ampliaciones que podría tener el software y como debe ser el manejo del PLC para su óptimo funcionamiento.

Los anexos contienen información complementaria de lo tratado en los capítulos y que sin duda ayudarán a los lectores de esta Tesis.

CAPÍTULO 1

ESTUDIO COMPARATIVO DE LOS CONTROLADORES LÓGICOS PROGRAMABLES

1.1 OBJETIVO GENERAL

El presente proyecto se fundamenta en diseñar e implementar un PLC de mediano costo y excelentes características basado en un microcontrolador. Una de las principales desventajas de un microcontrolador es la programación en lenguaje ensamblador que requiere de un adiestramiento especial, por lo cual se decidió desarrollar un lenguaje de programación gráfico a base de "BLOQUES DE FUNCIONES", para programar a este dispositivo.

1.2 DEFINICIÓN DE UN CONTROLADOR LÓGICO PROGRAMABLE (PLC)

Previo al origen de los PLCs, las máquinas y dispositivos de un proceso industrial eran manejadas por operadores humanos, quienes realizaban sus tareas sobre la base de la información que obtenían, muchas veces a través de sus sentidos y basados en su experiencia, hasta generar órdenes que se transmitían a la máquina a través de accionadores o actuadores. Todo este accionar tenía varias desventajas tanto para el proceso como para los operadores. El proceso se veía afectado por errores propios de los seres humanos, baja velocidad de la producción, baja flexibilidad al cambio de producto y la limitación para controlar procesos más complejos. En cuanto a los operadores siempre estaba el tema de la baja confiabilidad y seguridad.

Por otro lado, se tenía el gran inconveniente del cableado que se debía hacer para conectar armarios llenos de relés y contactores. Fue este entorno poco flexible, y las exigencias cada vez mayores de la automatización, junto a la aparición de los microprocesadores lo que impulsó el desarrollo de los controladores lógicos programables (PLC).

"Un PLC es un equipo electrónico de cableado interno, independiente del proceso a controlar; están diseñados para controlar en tiempo real, en un entorno industrial cualquiera, un proceso secuencial de la industria en general. Se adaptan a la máquina o instalación mediante un programa que define la función de las operaciones que se desea, entre los elementos de entrada y salida del PLC. "[*1] En la Figura 1.1 se muestra un Controlador Lógico Programable.

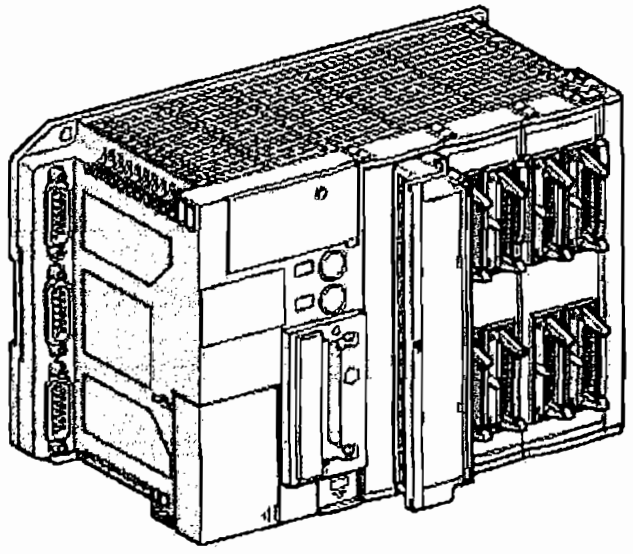


Figura 1.1. Vista frontal de un Controlador Lógico Programable.

1.3 VENTAJAS Y DESVENTAJAS DE UN CONTROLADOR LÓGICO PROGRAMABLE (PLC)

Para establecer parámetros de referencia y comparación, y así poder guiar de mejor manera el presente proyecto, a continuación se revisan las características más importantes de los PLCs comerciales

*1. CARROBLES, Marcial; RODRÍGUEZ, Felix. Manual de Mecánica Industrial, Autómatas y Robótica III. Editora Cultural, España, 2000

Las ventajas, que actualmente hacen a los PLCs uno de los recursos de control más utilizados son:

- Su fácil mantenimiento
- Utilidad en aplicaciones de control complejo
- Apertura al cambio y ampliaciones del automatismo
- Posibilidad de gobernar varias máquinas al mismo tiempo
- Alta confiabilidad en el ambiente industrial
- Mejora el control de la producción ya que pueden introducir sistemas automáticos de verificación
- Mínimo de espacio de ocupación, modulares y expandibles
- Permite la variación del proceso
- Capacidad de comunicación.
- Permite programar la producción.
- Se reducen las incidencias laborales

Las desventajas de un PLC son pocas, entre estas se pueden mencionar:

- Requieren buena regulación de voltaje
- Para pequeñas aplicaciones es costoso
- Se necesita preparación o adiestramiento para manejarlos

Sin embargo, algunos de estos inconvenientes se puede decir que han sido superados, por ejemplo, en la actualidad existe una gama de PLCs cuyo precio es bastante conveniente para una industria mediana, sin renunciar a sus ventajas.

1.4 HARDWARE BÁSICO DE UN CONTROLADOR LÓGICO PROGRAMABLE

Los PLCs pueden tener una estructura compacta o una estructura modular. En la estructura compacta todos lo elementos están en un solo bloque, mientras que en la estructura modular depende del fabricante o de las siguientes tendencias:
Según la tendencia americana se debe separar las E/S del resto del autómata; y según la europea cada módulo es una función (fuente de alimentación, CPU, E/S, etc.). Después de analizar las dos tendencias, se decidió que el trabajo presente

siga la estructura modular, por ofrecer más versatilidad como se indica en la Figura 1.2.

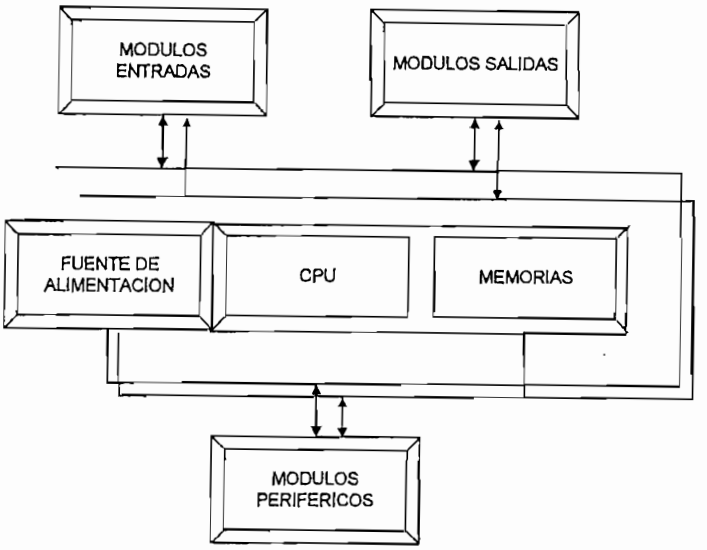


Figura 1.2. Estructura de un controlador lógico programable.

1.4.1 FUENTE DE ALIMENTACIÓN

Es habitual que los controladores lógicos programables deban alimentarse de las tensiones usadas en la industria: 24 Vcc en continua o a 110 Vca en alterna. La principal característica de estas fuentes es su grado de supresión de interferencias y la transformación de la energía eléctrica suministrada por la red de alimentación en las tensiones continuas exigidas por los componentes electrónicos e interfaces. Para evitar que en caso de falla de tensión se pierda la información almacenada en el PLC, la fuente de alimentación puede incluir una batería.

Para reducir el tamaño y peso del PLC, y además proteger al microprocesador de interferencias, obteniendo al mismo tiempo igual potencia que con una fuente lineal, se crearon las fuentes conmutadas o fuentes switching cuyo diseño varía de acuerdo a la potencia, aislamiento, costo y regulación de voltaje que se requiera.

Las características más importantes que estas fuentes poseen son:

- Tensión de entrada universal
- Encendido suave
- Protección de sobrecalentamiento
- Protección de sobre corriente
- Protección contra cortocircuito
- Protección contra sobre y bajo voltaje
- Voltaje de salida regulado
- Factor de potencia Corregido
- Tamaños estándar de la industria

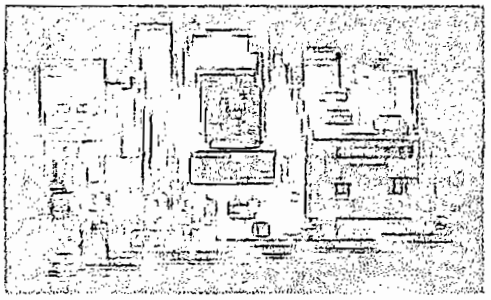


Figura 1.3. Foto de una fuente switching
Artesyn NFS40-7632
Entrada: 100-240VAC
Salida: +12VDC, +5VDC, -12VDC
Medidas: 12,7 cm x 7,6 x 3 cm.

Por sus características, se puede concluir que una de estas fuentes sería la más indicada para alimentar el PLC del presente proyecto.

1.4.2 UNIDAD CENTRAL DE PROCESO (CPU)

La Unidad Central de Proceso es la parte inteligente del sistema. Interpreta las instrucciones del programa de usuario y consulta el estado de las entradas. Dependiendo de dichos estados y del programa, ordena la activación de las salidas deseadas.

La CPU está conformada por los siguientes elementos:
- Procesador
- Memoria y
- Circuitos Auxiliares

1.4.2.1 Procesador

El procesador está constituido por el microprocesador y otros circuitos integrados, incluyendo la memoria ROM del sistema con el firmware.
Las distintas marcas de PLCs no tienen normas para la elección del microprocesador. Se puede escoger desde una variedad muy amplia, tomando como referencia el número de bits que el microprocesador maneja para sus operaciones. Este número va desde 8 bits hasta de 32 bits.
Por ejemplo, existe el microprocesador Motorola ColdFire de 32 bits, que constituye, una herramienta poderosa tanto en términos de programación, como en cuanto a su manejo. El uso de un microprocesador de 8 bits, como el que se empleará en este proyecto, no necesariamente lo hace menos capaz para realizar operaciones básicas.

1.4.2.2 Memoria

Para un PLC es un aspecto importante el tamaño de la memoria. En general, las unidades centrales incorporan una cantidad de memoria acorde con su capacidad de control y la potencia del conjunto de instrucciones con las que opera. Para adaptarse mejor a cada aplicación y por razones económicas, un mismo equipo suele presentarse con distintas opciones de cantidad de memoria o bien ofrecer la posibilidad de ampliación. Si bien un microprocesador viene con cierta cantidad de memoria, se puede ampliarla mediante circuitos integrados o bien mediante módulos de memoria. En todo caso, la posibilidad de expansión futura de la memoria debe existir para no encontrarse con la necesidad de sustituir toda una unidad central. Esta opción, sin embargo, encarece el precio del PLC y no siempre se la necesita. Tomando en consideración lo expuesto, el microcontrolador que se escogió y del cual se habla en el Capítulo 2, posee una capacidad de memoria suficiente para una aplicación de mediana extensión.

Los diferentes tipos o tecnologías de la memoria empleadas en los PLCs comerciales son los siguientes:

- La memoria RAM (Randon Acces Memory)
- La memoria ROM (Read Only Memory), también conocida como firmware
- Las memorias EPROM.
- Las memorias EEPROM
- Las combinaciones RAM + EEPROM

Los PLCs comerciales disponen de diferentes tipos de memorias tanto en capacidad, como expansibilidad y tipo. Como ejemplo se muestra en la Tabla 1.1 lo que ofrece Siemens.

	CPU 212	CPU 214
Memoria de programa Memoria de datos Módulos de memoria (opcional)	1 KB / aprox. 0.5 K líneas 512 palabras	4 KB / aprox. 2 K líneas 2,048 palabras 1 incluida (EEPROM);

Tabla 1.1. Características de las memorias para dos diferentes PLCs de Siemens.

En cambio, el microcontrolador que se pensó en utilizar tiene las siguientes características de memoria:

	PIC16F877A
Memoria de programa	8Kx14 FLASH
Memoria de datos	256x8 bytes EEPROM
Memoria RAM	368x8 bytes SRAM

Tabla 1.2. Características de las memorias del PIC16F877A.

Como se puede ver en la Tabla 1.2, la memoria de programa del microcontrolador es más que suficiente, comparado con la del PLC de la Tabla 1.1, mientras que la memoria de datos volátil es de tamaño mediano, pero suficiente para la aplicación que se va a desarrollar.

1.4.3 MÓDULOS DE ENTRADA

Los módulos de entrada son los encargados de transmitir el estado del proceso a la unidad central de Proceso (CPU); a estos módulos se cablearán los sensores. A los módulos de entrada se los define como dispositivos básicos, por donde se toma la información de los captadores (interruptores, finales de carrera, pulsadores, sensores inductivos, sensores capacitivos, sensores fotoeléctricos, etc.), los cuales se acoplan al bus de datos por medio de su conductor y conector correspondiente, o bien a través de un bastidor o rack que le proporciona dicha conexión al bus y soporte mecánico, todo esto dependiendo de las facilidades constructivas que se disponga. Sin embargo, en este punto cabe indicar que no existe una normalización, y queda más bien a la creatividad y recursos de los que se disponga.

Las entradas suelen ser fácilmente identificables debido a su numeración, o por su identificación de input o entrada, por sus bornes para acoplar los dispositivos de entrada-salida, y también por su indicación luminosa de activado por medio de un LED. En este proyecto sin embargo la nomenclatura resaltará la existencia de los puertos del microprocesador, con sus correspondientes nombres.

1.4.3.1 Módulos de entradas digitales

Los módulos de entrada digitales permiten conectar al PLC captadores de tipo todo o nada como: finales de carrera, pulsadores, etc.

Los módulos de entrada digital trabajan con señales de tensión, por ejemplo cuando por una vía llegan 24 voltios se interpreta como un "1" y cuando llegan cero voltios se interpreta como un "0"

El proceso de adquisición de la señal digital consta de varias etapas.

- Protección contra sobre tensiones
- Aislamiento galvánico o por opto acoplador.

Al hablar de este punto, hay que recordar que los PLCs que se comercializan actualmente poseen una buena robustez. Sin embargo, este proyecto tiene como idea central la programación de un microcontrolador en un lenguaje diferente del que comúnmente se lo programa para su empleo en un ambiente industrial. Aún así, esto quiere decir que no se debe descuidar y dejar sin las debidas protecciones al microcontrolador, tema que se detalla de mejor manera en el Capítulo 3.

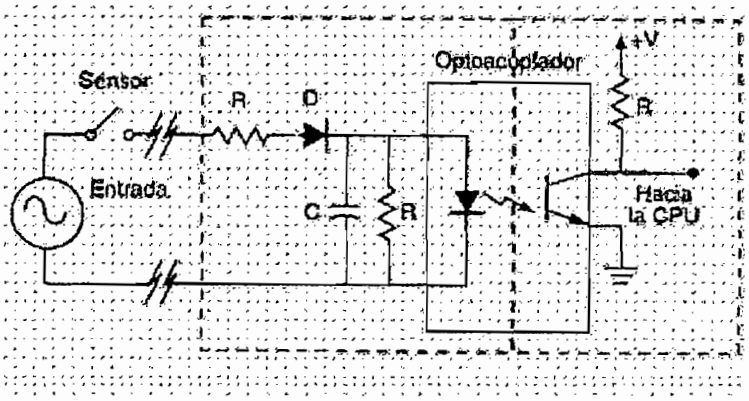


Figura 1.4. Ejemplo de una entrada digital que puede tener un PLC.

1.4.3.2 Módulos de entradas analógicas

Los módulos de entrada analógicas permiten que los PLCs trabajen con accionadores de mando analógico y lean señales de tipo analógico como puede ser la temperatura, la presión o el caudal.

Los módulos de entradas analógicas convierten una magnitud analógica en un número, con una precisión o resolución determinada (numero de bits), y cada cierto intervalo de tiempo (periodo de muestreo), dependiendo del microprocesador que se utilice en el PLC.

Los módulos de entrada analógica pueden leer tensión o intensidad. Para el presente proyecto se va ha utilizar el conversor A/D interno del microcontrolador que posee varios canales en dos de sus puertos, para leer una señal de voltaje.

1.4.4 MÓDULOS DE SALIDA

Los módulos de salida son los encargados de transmitir las órdenes a los actuadores (dispositivos de salida) estas órdenes las genera la Unidad Central de Proceso (CPU) en base a un algoritmo, previamente definido para ese proceso en concreto. Cada señal irá cableada a su correspondiente actuador.

La identificación de las salidas se realiza de forma similar a las entradas; en este caso aparecerá la indicación de output (salida) y en el caso del PLC que se desarrolla en este proyecto, según los puertos del microprocesador. También para estos módulos es necesario un LED de activado. Los tipos de salida que se pueden proporcionar son:

1. Salidas a relés (ca o cc). Este tipo de salida suele utilizarse cuando el consumo es alto (del orden de los amperios) y también donde las conmutaciones no sean demasiado rápidas. Ejemplos que se pueden citar son: cargas de contactores, electrovávulas, etc.
2. Salidas a triac (ca o cc). Se suelen utilizar para conmutaciones muy rápidas, las que el relé no es capaz de realizar. Su tiempo de vida útil es más largo que la del relé y es capaz de manejar valores de intensidad similares al relé.
3. Salida a transistores (cc). Este tipo de salida es conveniente cuando se utiliza cc, cuando las cargas sean del tipo de poco consumo, respuesta rápida y alto número de operaciones, como es el caso de circuitos electrónicos. Su tiempo de vida útil es superior a la del relé.

El análisis de confiabilidad y funcionamiento, conduce a los módulos de salida electromecánica por lo cual el PLC en construcción posee esta como única alternativa de salida digitales.

1.4.4.1 Módulos de salidas digitales

Un módulo de salida digital permite al PLC actuar sobre los accionadores que admiten ordenes de tipo todo o nada.

El valor binario de las salidas digitales se convierte en la apertura o cierre de un relé interno del PLC en el caso de módulos de salidas a relé.

En los módulos estáticos, los elementos que conmutan son los componentes electrónicos como transistores o triacs, y en los módulos electromecánicos son contactos de relés internos al módulo.

Los módulos de salidas estáticos al suministrar tensión, solo pueden actuar sobre elementos que trabajan todos a la misma tensión, en cambio los módulos de salida electromecánicos, al ser libres de tensión, pueden actuar sobre elementos que trabajen a tensiones distintas.

El proceso de envío de la señal digital consta de varias etapas:

- Aislamiento
- Circuito de mando (relé interno)
- Protección electrónica

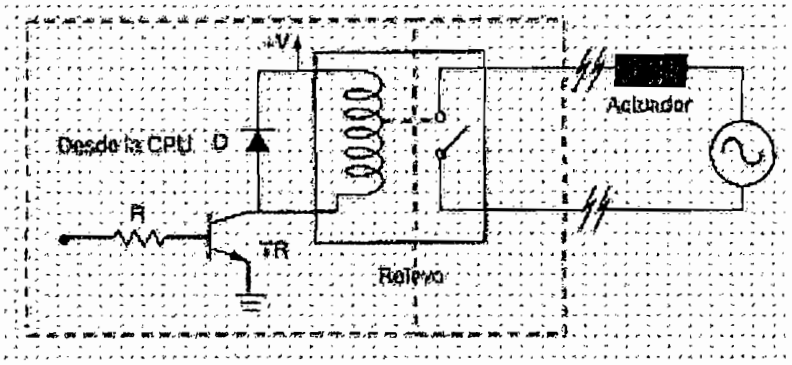


Figura 1.5. Una de las configuraciones de una salida digital que puede tener un PLC.

1.4.4.2 Módulos de salidas analógicas

Los módulos de salida analógica permiten que el valor de una variable numérica interna del PLC se convierta en tensión o intensidad. Lo que realiza es una conversión D/A, puesto que el PLC solo trabaja con señales digitales. Esta conversión se realiza con una precisión o resolución determinada (numero de bits) y cada cierto intervalo de tiempo (periodo muestreo).

Esta tensión o intensidad puede servir de referencia de mando para actuadores que admitan mando analógico como pueden ser los variadores de velocidad, las etapas de los tiristores de los hornos, reguladores de temperatura, etc. Permitiendo al PLC realizar funciones de regulación y control de procesos continuos.

El proceso de envío de la señal analógica consta de varias etapas:

- Aislamiento galvánico.
- Conversión D/A.
- Circuitos de amplificación y adaptación.
- Protección electrónica de la salida.

Sin embargo, este proceso en el PLC a diseñar no se implementará. El microprocesador, que en realidad es un sistema microcontrolado, incorpora varios periféricos en su encapsulado siendo uno de estos dos PWMs, los cuales no generan señales analógicas, pero si sus señales fueran filtradas actuarían como conversores D/A. Esta opción de filtrado no ofrece este trabajo y queda mas bien a criterio del usuario.

Como se ha visto, las señales analógicas sufren un gran proceso de adaptación tanto en los módulos de entrada como en los módulos de salida. Las funciones de conversión A/D y D/A que realiza son esenciales. Por ello los módulos de E/S analógicos se los consideran módulos de E/S especiales.

Para terminar con esta parte tan importante de un PLC, como son los módulos de entrada / salida (E/S), se indica algunos ejemplos que PLCs comerciales ofrecen:

- En la Tabla 1.3 se enumeran las características de los módulos de Entradas / salidas de dos modelos de PLCs Siemens.

	CPU 212	CPU 214
E/S integradas:		
- Entradas digitales	8	14
- Salidas digitales	6	10
- Potenciómetros analógicos	1	2
E/S conectables:	Máx. 64	Máx. 64
- E/S digitales	entradas y 64 salidas	entradas y 64 salidas
- E/S analógicas	16 entradas y 16 salidas	16 entradas y 16 salidas

Tabla 1.3. Características para el CPU 212 y CPU 214, de Siemens.

Todos los puertos del microcontrolador se pueden configurar como entradas o salidas dependiendo del programa realizado en la PC (software) y la tarjeta utilizada (por ejemplo: si un puerto se lo configura en el computador como salida deberá utilizar una tarjeta de salidas a relé). A continuación se indica los puertos que posee el microcontrolador.

	PIC16F877A
Entradas o Salidas digitales	33 (5 puertos)
Puerto A	*6 Entradas o salidas digitales y los 5 primeros canales* para el conversor A/D.
Puerto B	8 Entradas o salidas digitales (1 entrada de interrupción externa)
Puerto C	8 Entradas o salidas digitales (2 salidas PWM y terminales RX, TX para comunicación serial RS232)
Puerto D	8 Entradas o salidas digitales.
Puerto E	3 Entradas o salidas digitales y los 3 últimos canales para el conversor A/D.

Tabla 1.4. Características de los puertos del microcontrolador.

1.4.5 MÓDULOS PERIFÉRICOS

Los fabricantes de los Controladores Lógicos Programables ofrecen en sus catálogos módulos para realizar funciones como:

1. Captación de desplazamientos,
2. Conteo de pulsos cuya frecuencia sea demasiado alta,
3. Funciones de regulación
4. Módulos de entradas especiales de temperatura.
5. Capacidad de comunicación con otros PLC's y con otros sistemas.

Como se explicará en el capítulo II, el presente proyecto se desarrolla mediante un sistema microcontrolado, lo que constituye una restricción en la ampliación de periféricos pues, como tal, este ya dispone algunos de los más importantes como: un A/D con varios canales, PWMs y comunicación serial RS-232. Una forma de aumentar recursos o periféricos al sistema sería mediante comunicación I^2C, que posee el microcontrolador que se utiliza en este Proyecto, pero no se desarrollará esta aplicación en este proyecto.

1.5 VELOCIDAD DE PROCESAMIENTO

La velocidad o frecuencia de trabajo del microprocesador es un parámetro fundamental a la hora de establecer la velocidad de ejecución de las instrucciones y el consumo de energía. Comparado con el PLC Siemens **(13)**, CPU 212 y CPU 241, cada operación booleana es de 1.2 μs para todas sus instrucciones, menos las de salto que tardan le doble. La máxima velocidad con que puede trabajar el *microcontrolador* es de 20MHz, con lo cual se obtiene una velocidad por instrucción de 200ns. Sin embargo, para este proyecto se trabajará con un cristal de 4MHz, suficiente para controlar cualquier proceso.

1.6 LENGUAJES DE PROGRAMACIÓN

Los Controladores Lógicos Programables fueron creados para sustituir la lógica cableada de los armarios de relés. Por esta razón los tipos de lenguajes de programación fueron diseñados de forma que se adaptasen al personal familiarizado con la tecnología cableada. Actualmente, y con el objeto de lograr un estándar mundial y con proyección de futuro en los PLC, surgió la norma internacional IEC 1131; dicha norma ha sido también adoptada como norma europea y como norma alema, la DIN EN 61131.
La norma IEC 1131 consta de cinco partes:

- Definición de conceptos y características generales de un PLC
- Requisitos eléctricos, mecánicos y funcionales impuestos a un PLC.
- Lenguajes de programación de autómatas.
- Directivas para usuarios de autómatas en las diferentes fases de un proyecto.
- Bloques estándar para la comunicación de PLCs de diferentes fabricantes.

Los lenguajes de normalizados para programación de un Controlador Lógico Programable son:

- Esquema de contactos: Ladder diagram (LD)
- Esquema de funciones: Function Block diagram (FBD)
- Lista de instrucciones: Instruction List (IL)
- Lenguaje secuencial: Sequential Function Chart (SFC)
- Texto estructurado: Structured Text (ST)
- Configuración de Programa: Program Configuration.

1.6.1 DIAGRAMA DE CONTACTOS

Es el lenguaje más antiguo y nace de la utilización de los controladores programables para sustituir la lógica de relés. Se representan las distintas instrucciones mediante símbolos de circuitos eléctricos. Fue desarrollado inicialmente en Estados Unidos como lenguaje de PLC.

Es un lenguaje gráfico que utiliza símbolos que representan contactos abiertos, contactos cerrados, bobinas de relés, etc., identificados mediante una variable de entrada, salida, etc. Este lenguaje expresa las secuencias de control de forma gráfica, similar a la empleada en la tecnología cableada de relés. Es evidente, por tanto, que sea el lenguaje preferido por los electricistas acostumbrados a trabajar con mandos realizados con lógica cableada. Como ejemplos de símbolos utilizados, se puede citar los siguientes:

] [Contacto normalmente abierto

] / [Contacto normalmente cerrado

( ) Bobina de relé o señal de accionamiento.

(S) Señal de activación de un biestable RS.

(R) Señal de desactivación de un biestable

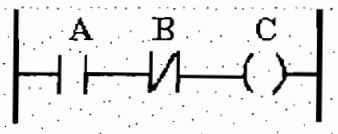


Figura 1.6. Esquema de lenguaje de programación, Diagrama de contactos.

1.6.2 DIAGRAMA DE FUNCIONES

Fue extendido en Europa, y es también un lenguaje gráfico, al igual que el de contactos, y en el se representan las instrucciones mediante símbolos de los circuitos electrónicos. Como ejemplo de estos símbolos se puede citar los siguientes:

& Función AND

>= Función OR

O Entrada Negada

= Salida

En este lenguaje las instrucciones se componen de la unión de los símbolos con los operandos adecuados.

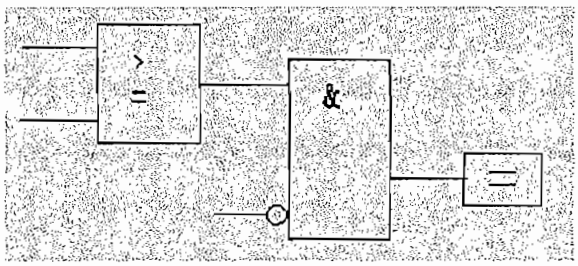


Figura 1.7. Esquema de lenguaje de programación, Diagrama de bloques.

1.6.3 LISTA DE INSTRUCCIONES

Lenguaje similar al ensamblador para lograr una programación optimizada en cuanto uso de memoria y tiempo de ejecución. Lenguaje de PLC desarrollado inicialmente en Alemania y extendido, sobre todo, en Europa. Solo está normalizado un reducido juego de instrucciones básico independiente del hardware. Utiliza caracteres alfanuméricos de las instrucciones para construir el programa. Es el que permite mayor variedad de instrucciones y operaciones. En este lenguaje, además de instrucciones de tipo booleano (and, or, negación, etc.), suele haber instrucciones de manejo del procesador como son:

- Manejo de los acumuladores y registros.
- Saltos condicionales e incondicionales.
- Operaciones aritméticas, etc.

Es fácil encontrar equipos que realizando el programa en diagrama de contactos o en diagrama de funciones se puede convertirlos a otras formas de representación. En cambio, no todas las instrucciones del lenguaje Lista de Instrucciones tienen traducción en diagrama de contactos o diagrama de funciones.

1.6.4 LENGUAJE SECUENCIAL

No es un lenguaje de programación propiamente dicho, sino una forma de representar la secuencia de ejecución de los distintos módulos de un programa. Orientado al estándar francés grafcet. El grafcet es un lenguaje gráfico compuesto de dos tipos de elementos: las etapas y transiciones. Con las etapas se representan las distintas acciones a realizar, mientras que con las transiciones regulan las condiciones de paso de una etapa a otra. Estos elementos se unen entre sí mediante líneas orientadas, alternando etapas y transiciones.

El grafcet no es un lenguaje de programación, sino una forma de representación de la secuencia de ejecución de los distintos módulos de un programa. Los elementos (Etapas y transiciones) se programan en cualquiera de los lenguajes anteriormente citados: diagrama de contactos, diagrama de funciones y lista de instrucciones.

1.6.5 TEXTO ESTRUCTURADO

Lenguaje de alto nivel para la realización de algoritmos y tareas de cálculo complejas. Basado en el lenguaje de programación Pascal con ampliaciones específicas para PLC.

1.6.6 CONFIGURACIÓN DE PROGRAMA

Programas realizados por la interconexión gráfica de bloques de función. Definición de diferentes niveles de ejecución (tareas). Hasta ahora utilizados en ingeniería de control de procesos.

De todos los lenguajes de programación antes mencionados, el lenguaje en el que se programa el PLC de este proyecto es FBD, el cual es familiar para cualquier persona que tenga conocimientos básicos de electrónica digital. Además esta fue la idea principal con la cual se planteó esta Tesis, ya que sería

la primera vez que se programe de esta forma, a un microcontrolador como el que se ha escogido.

1.7 OPERACIONES PARA PROGRAMACIÓN

Las Operaciones básicas para programar un Controlador Lógico Programable son las siguientes:

- Operaciones lógicas
- Operaciones de memoria
- Operaciones de tiempo
- Operaciones de computo
- Operaciones de comparación

Todas estas operaciones son las básicas y el PLC que se construye para este proyecto deberá por lo menos tener la mayoría de estas, además deberá tener otras que faciliten a los usuarios la elaboración de programas y proyectos, economizando un recurso muy importante, como lo es el tiempo.

1.8 CARACTERÍSTICAS A CONSIDERAR EN LA ELECCIÓN DE UN CONTROLADOR LÓGICO PROGRAMABLE

Durante todo este capítulo se enumeró las características generales y especificas que un Controlador Lógico Programable puede tener; sin embargo, también se conoce que las características dadas, no pertenecen a todos los PLCs.
Actualmente existen PLCs tan completos que pueden poseer en su mayoría todas o casi todas las características antes descritas, lo cual implica que su precio también será mucho mayor, que aquel que posee características básicas; sin embargo, esto no es importante sin un análisis sistemático completo de una serie de factores, que debe hacerse previo a la elección del Controlador Lógico Programable, no solo en base a las características actuales de la tarea de control,

sino también de acuerdo a las necesidades futuras en función de los objetivos de la empresa.

Cuando se trata de automatizar un sistema se debe tener primero una idea global del funcionamiento de la máquina o proceso que se quiere automatizar. Para hacer una mejor elección, se tendrá en cuenta dos factores: los factores cuantitativos y los factores cualitativos.

1.8.1 FACTORES CUANTITATIVOS

Se refieren a la capacidad del equipo para soportar todas aquellas especificaciones para el sistema de control y se pueden agrupar en las siguientes categorías:

- Entradas/ Salidas (E/S): cantidad, tipo, prestaciones, ubicación, etc.

- Tipo de control: control de una o varias máquinas, proceso, etc.

- Memoria: cantidad, tecnología, expansibilidad, etc.

- Software: conjunto de instrucciones, módulos de programa, etc.

- Periféricos: equipos de programación, dialogo hombre - maquina, etc.

- Físicos y ambientales: características constructivas, banda de temperatura.

1.8.2 FACTORES CUALITATIVOS

Una vez evaluados los factores correspondientes a las características técnicas y constructivas de los componentes de PLC y equipos periféricos, él numero de equipos posibles para una determinada aplicación. Es necesario tomar una decisión en basa a criterios comerciales y en general limitados al aspecto económico de la adquisición, pero se debe tener en cuenta otros aspectos que en

definitiva tendrán una mayor influencia a medio plazo. Es el momento de evaluar factores menos tangibles que se ocultan en las mismas características del equipo y en las del fabricante o el suministrador del PLC.

1.9 CARACTERÍSTICAS DEL PLC DISEÑADO PARA ESTE PROYECTO

De lo dicho anteriormente y recopilando estas características el PLC de este trabajo tiene:

- Fuente de alimentación 110Vac a 24Vdc/15Vdc.
- Microcontrolador con memoria flash de programa de capacidad de 8k de palabras de 14 bits cada una y memoria de datos no volátil EEPROM de 256 bytes.
- 2 tarjetas de entrada digitales de 9 entradas cada una, y con aislamiento por optoacopladores.
- 2 tarjetas de salidas digitales a relé normalmente abiertas de 24Vdc, de 9 salidas cada una.
- 1 tarjeta de entradas analógicas de 8 canales, de 0 a 10V.
- 1 tarjeta con conector DB9 para comunicación RS-232 half duplex, y dos salidas PWM.

CAPÍTULO 2

ESTUDIO DE LAS CARACTERÍSTICAS DEL MICROCONTROLADOR UTILIZADO COMO CPU

2.1 ESTRUCTURA Y CARACTERÍSTICAS DEL MICROCONTROLADOR

En este capítulo se justifica la selección del microcontrolador PIC16F877A y se procede a revisar sus características, procurando determinar si las mismas son suficientes para diseñar un PLC con el PIC escogido. Además se explica la forma de cómo se debe programar este microcontrolador, es decir que señales deben ser enviadas desde la PC al PIC, pero tomando en cuenta las características eléctricas de este, esta parte permite al lector de esta tesis un mejor entendimiento del Diseño del Software que se explica en el Capítulo 4.

La unidad central de proceso de un PLC normalmente está constituida por el procesador, memoria y circuitos auxiliares, como se habló en la Sección 1.3.2 del capítulo anterior, pero este PLC tendrá una estructura distinta, ya que la memoria y periféricos como: A/D, PWM, puerto serial, etc. forman parte del circuito integrado.

La elección del Microprocesador se basó en varios aspectos que se explican a continuación. Como procesador se decidió emplear un microcontrolador ya que este ofrece varias ventajas respecto a un microprocesador como son:

- Más resistencia a condiciones físicas externas.
- Recursos incorporados como conversores A/D, salidas PWM, memoria de datos no volátil, comunicación serial, etc.
- Protección ante fallos de alimentación.

- Código de protección programable.
- Programación de la memoria de programa más sencilla.
- Costo vs. prestaciones más accesibles.

Por el contrario, la desventaja de utilizar un microcontrolador es que es un sistema cerrado, cuyos recursos no se pueden modificar, por ejemplo: el número de A/Ds, la memoria de programa, etc.
Entre más de cincuenta fabricantes de microcontroladores que existen en el mundo es muy difícil seleccionar "el mejor". En realidad no existe, porque en cada aplicación son sus características específicas las que determinan el más conveniente. Sin embargo, los factores siguientes son determinantes en su elección: sencillez de manejo, buena información, precio, buen promedio de parámetros (velocidad, consumo, tamaño, alimentación, código compacto), entre otros.

Para el caso de este proyecto de titulación se escogió uno de los microcontroladores PIC de la Microchip, basados en la familiaridad y la experiencia que se adquirió con estos durante nuestra formación.

Pero Microchip dispone de cuatro familias o gamas de microcontroladores de donde escoger, estas son:

Gama Enana: PIC16F(C) XXX de ocho pines, es un PIC de reducido tamaño, de las siguientes características:

- Alimentación de 2.5 V a 5.5V, consumen menos de 2 mA y pueden trabajar hasta 2 Mhz, su aplicación está limitada a la aplicación de recursos que se comunican serialmente (I^2C).

Gama Baja o Básica: PIC16C5X, con instrucciones de 12 bits, se trata de una serie de PICs con recursos limitados, solo tienen manejo de instrucciones a nivel de bits. No tienen ningún otro recurso auxiliar.

Gama Media: PIC16F(C) XXX con instrucciones de 14 bits. Es la gama mas variada y completa de los PICs, tienen encapsulados desde 18 pines hasta 68 e integran abundantes periféricos externos.

Gama Alta: PIC17CXXX con instrucciones de 16 bits. Tiene un sistema de gestión de interrupciones vectorizadas muy potente, además de diversos periféricos externos y un multiplicador de hardware a gran velocidad. La característica más significativa es su arquitectura abierta.

Dentro de la gama media, Microchip presentó al mercado mundial los nuevos microcontroladores RISC con memoria de programa FLASH, que en general reúnen las mejores características de todas las gamas o familias, se trata de la serie PIC16F87X y PIC16F87XA. De estos nuevos PICs el que más recursos posee y mejor se adapta a las necesidades de este trabajo es el **PIC16F877A** cuyas características más sobresalientes son:

Como recursos Fundamentales:

- Procesador de arquitectura RISC avanzada.
- Juego de 35 instrucciones de 14 bits de longitud, todas ellas se ejecutan en un ciclo de instrucción, menos las instrucciones de salto.
- Frecuencia de trabajo hasta de 20 Mhz.
- Hasta 8k palabras de 14 bits para la memoria de código tipo flash.
- Hasta 368 bytes de memoria de datos RAM
- Hasta 256 bytes de memoria de datos no volátil EEPROM.
- Hasta 14 fuentes de interrupción interna y externa.
- Pila con 8 niveles.
- Modo de direccionamiento directo, indirecto y relativo
- Perro guardián (WDT).
- Código de protección programable.
- Modo SLEEP de bajo consumo.
- Programación serie en circuito con dos pines.
- Voltaje de alimentación comprendido entre 2 y 5.5V

- Bajo consumo (menos de 2mA a 5V y 5Mhz)

Como dispositivos Periféricos:

- Timer0: temporizador – contador de 8 bits, con predivisor.
- Timer1: temporizador – contador de 16 bits, con predivisor.
- Timer2: temporizador de 8 bits, con predivisor y postdivisor.
- Dos módulos de PWM, captura y comparación.
- Conversor A/D de 10 bits.
- Puerto serie sincrónico (SSP) con SPI e I^2C.
- Puerto serie asincrónico (USART).
- Puerta Paralela Esclava (PSP).
- Una interrupción externa.

2.1.1 CARACTERÍSTICAS FÍSICAS DEL MICROCONTROLADOR 16F877A

- Encapsulado tipo DIP de 40 pines.
- Cinco puertos configurables para entradas y/o salidas.
- Puerto A de 6 bits, puerto B, C, D de 8 bits y puerto E de 3 bits.
- Dos pines para el cristal oscilador de cuarzo.
- Dos pines para VDD (+5V) y dos pines para GND.
- Un pin para reset o entrada del voltaje de programación _ verificación.

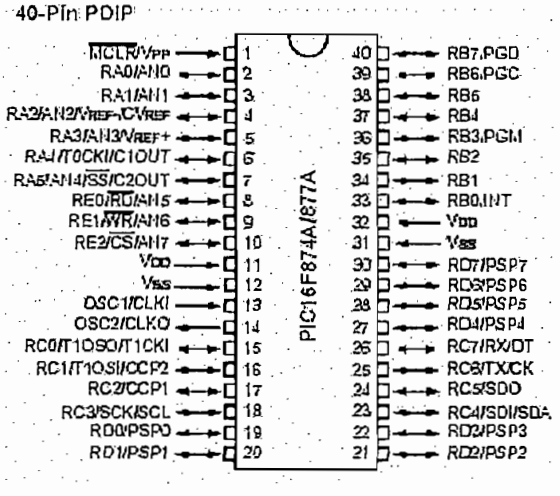


Figura 2.1. Encapsulado y disposición del microcontrolador PIC16F877A.

2.1.2 PROCESADOR RISC CON ARQUITECTURA HARVARD

Esta arquitectura (Figura 2.2) se característica por la independencia entre la memoria de código y la de datos, facilitando el trabajo en paralelo de las dos memorias. Lo que permite obtener altas cotas de rendimiento. La filosofía RISC se refleja en el reducido número de instrucciones que forman su repertorio. Solo constan de 35 instrucciones que se ejecutan en un ciclo de instrucción equivalente a 4 ciclos de reloj, excepto las de salto que necesitan dos.

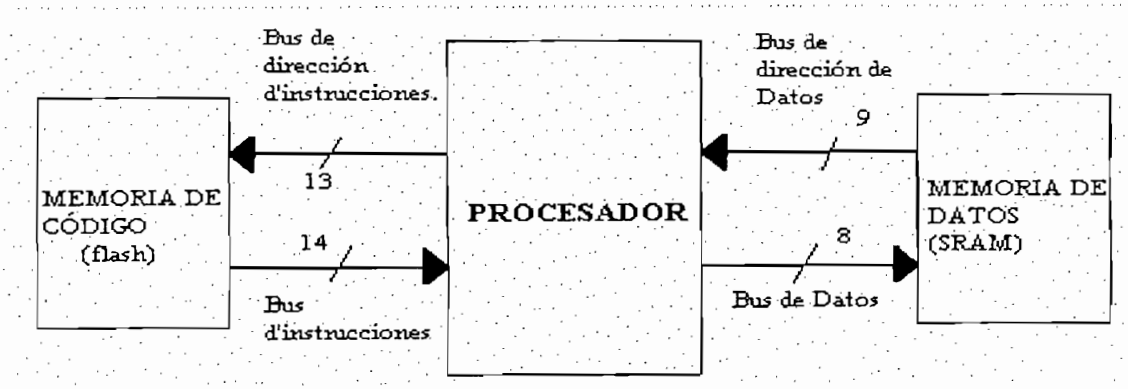


Figura 2.2. Arquitectura Harvard.

2.1.3 ORGANIZACIÓN DE LA MEMORIA DE PROGRAMA

Como ya se mencionó en un párrafo anterior, la memoria flash es donde se almacena el programa de la aplicación. En el PIC16F877A forma parte del mismo circuito integrado y tiene una capacidad de 8 k palabras de 14 bits cada una; dicha memoria está dividida en dos páginas de 4k cada una.

La pila tiene 8 niveles de profundidad y funciona automáticamente, no necesita instrucciones para guardar o sacar información de ella. El vector reset ocupa la dirección 0000h, y el vector de interrupción la posición 0004h. (Ver Figura 2.3).

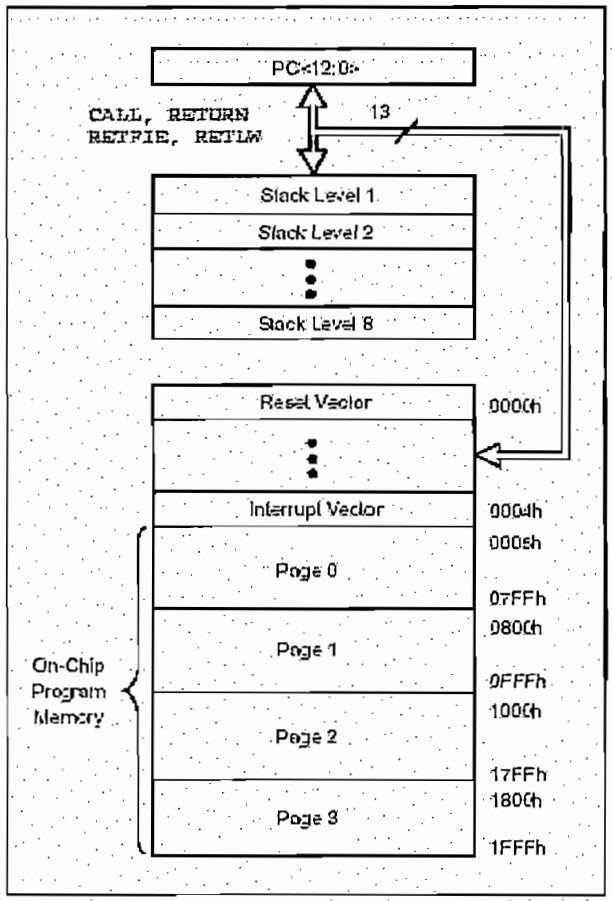


Figura 2.3. Mapa de la memoria de programa y pila.

2.1.4 MEMORIA DE DATOS RAM

La memoria RAM también forma parte del mismo encapsulado de este microcontrolador y su función es alojar los registros operativos fundamentales del funcionamiento del procesador y el manejo de todos sus periféricos, además de registros que el programador puede usar para el trabajo propio de la aplicación. La RAM consta de cuatro bancos con 128 bytes cada uno. En las posiciones iniciales de cada banco se encuentran los registros específicos que gobiernan al procesador y sus recursos. El total de esta memoria es de 368x8 bytes. Como ya se explicó en la Sección 1.3.2.2 esta cantidad es suficiente para la aplicación que se está desarrollando en este proyecto.

2.1.5 SET DE INSTRUCCIONES

Ninguna descripción de un microcontrolador está completa sin el set de instrucciones, en este caso consta de 35 instrucciones simples. Dichas instrucciones servirán para crear el código de programa de la aplicación desarrollada en FBD, lo cual será detallado mejor en el Capítulo 4: Diseño del Software. En la Figura 2.4 se muestra un resumen del Set de Instrucciones.

Mnemonic, Operands		Description	Cycles	14-Bit Opcode MSb	LSb	Status Affected	Notes
BYTE-ORIENTED FILE REGISTER OPERATIONS							
ADDWF	f, d	Add W and f	1	00 0111 dfff ffff		C,DC,Z	1,2
ANDWF	f, d	AND W with f	1	00 0101 dfff ffff		Z	1,2
CLRF	f	Clear f	1	00 0001 1fff ffff		Z	2
CLRW	-	Clear W	1	00 0001 0xxx xxxx		Z	
COMF	f, d	Complement f	1	00 1001 dfff ffff		Z	1,2
DECF	f, d	Decrement f	1	00 0011 dfff ffff		Z	1,2
DECFSZ	f, d	Decrement f, Skip if 0	1(2)	00 1011 dfff ffff			1,2,3
INCF	f, d	Increment f	1	00 1010 dfff ffff		Z	1,2
INCFSZ	f, d	Increment f, Skip if 0	1(2)	00 1111 dfff ffff			1,2,3
IORWF	f, d	Inclusive OR W with f	1	00 0100 dfff ffff		Z	1,2
MOVF	f, d	Move f	1	00 1000 dfff ffff		Z	1,2
MOVWF	f	Move W to f	1	00 0000 1fff ffff			
NOP	-	No Operation	1	00 0000 0xx0 0000			
RLF	f, d	Rotate Left f through Carry	1	00 1101 dfff ffff		C	1,2
RRF	f, d	Rotate Right f through Carry	1	00 1100 dfff ffff		C	1,2
SUBWF	f, d	Subtract W from f	1	00 0010 dfff ffff		C,DC,Z	1,2
SWAPF	f, d	Swap nibbles in f	1	00 1110 dfff ffff			1,2
XORWF	f, d	Exclusive OR W with f	1	00 0110 dfff ffff		Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS							
BCF	f, b	Bit Clear f	1	01 00bb bfff ffff			1,2
BSF	f, b	Bit Set f	1	01 01bb bfff ffff			1,2
BTFSC	f, b	Bit Test f, Skip if Clear	1 (2)	01 10bb bfff ffff			3
BTFSS	f, b	Bit Test f, Skip if Set	1 (2)	01 11bb bfff ffff			3
LITERAL AND CONTROL OPERATIONS							
ADDLW	k	Add Literal and W	1	11 111x kkkk kkkk		C,DC,Z	
ANDLW	k	AND Literal with W	1	11 1001 kkkk kkkk		Z	
CALL	k	Call Subroutine	2	10 0kkk kkkk kkkk			
CLRWDT	-	Clear Watchdog Timer	1	00 0000 0110 0100		$\overline{TO}$,$\overline{PD}$	
GOTO	k	Go to Address	2	10 1kkk kkkk kkkk			
IORLW	k	Inclusive OR Literal with W	1	11 1000 kkkk kkkk		Z	
MOVLW	k	Move Literal to W	1	11 00xx kkkk kkkk			
RETFIE	-	Return from Interrupt	2	00 0000 0000 1001			
RETLW	k	Return with Literal in W	2	11 01xx kkkk kkkk			
RETURN	-	Return from Subroutine	2	00 0000 0000 1000			
SLEEP	-	Go into Standby mode	1	00 0000 0110 0011		$\overline{TO}$,$\overline{PD}$	
SUBLW	k	Subtract W from Literal	1	11 110x kkkk kkkk		C,DC,Z	
XORLW	k	Exclusive OR Literal with W	1	11 1010 kkkk kkkk		Z	

Figura 2.4. Set de instrucciones del PIC16F877A.

2.2 PROGRAMACIÓN DEL MICROCONTROLADOR PIC16F877A

El microcontrolador PIC16F877A se programa de una forma serial sincrónica de tipo FULL DUPLEX, y con el puerto paralelo de la computadora que es el que se escogió.

Con el software desarrollado para esta tesis y que se explicará con más detalle en el Capítulo 4, se hará un manejo adecuado del puerto paralelo con el fin de generar las señales que el microcontrolador necesita para programar la memoria de programa, de datos y la palabra de configuración. Durante esta sección se hará una descripción completa de estas señales, mientras que en el capitulo siguiente se verá el diseño completo de este circuito.

Del puerto paralelo se usarán cinco pines para emitir dichas señales, un pin para la señal de transmisión, y otro para la recepción de datos hacia el microcontrolador, un pin para la señal de reloj y dos pines más para controlar las señales de voltaje de alimentación y voltaje de programación.

Los algoritmos de programación del PIC son únicamente dos y son los siguientes:

- El primero, recomendado por Microchip, utiliza la programación conocida como alto voltaje. En esta se mantiene los pines RB6 y RB7 en bajo mientras el pin MCLR pasa del voltaje VIL (0V) a VHH (13+-0.5V).
- El segundo método, llamado de bajo voltaje ICSPTM o LVP, se aplica VDD (+5V) a MCLR y se usa el pin I/O RB3 para entrar en este modo. Cuando RB3 pasa de voltaje 0 a VDD el PIC entra en este modo de programación.

De los dos algoritmos se escogió el método que aconseja MICROCHIP es decir, por alto voltaje puesto que en el segundo método se pierde el RB3 (bit 3 del puerto B), dejando a este solamente para el control de la programación; además, el segundo método solamente se lo utiliza en aquel hardware que no disponga de

voltaje de programación alto (13V). De acuerdo a esto, la explicación que sigue se basa en el método por alto voltaje que además se amplia en el PDF: FLASH Memory Programming Specification de la MICHOCHIP, y que se incluye como ANEXO A.

Como es de suponerse, la programación del PIC no es una sola ya que existen algunas variantes que dependen del programador y las necesidades que este tenga a la hora de programar.

A continuación se detallan algunos conceptos que son necesarios para un mejor entendimiento de la programación del PIC.

En primer lugar se debe considerar que la memoria de programa, está dividida en dos partes; la memoria de usuario desde la 0000h hasta 1FFFh (memoria donde se graba el programa que el usuario desea grabar en el microcontrolador) y la memoria de configuración (donde se graba las localidades ID y la palabra de configuración) que va desde la 2000h hasta la 2008h.

La memoria de datos, es una memoria EEPROM de 256 bytes de capacidad. Su manejo es importante para guardar información que el usuario desee conservar.

Localidades ID, son cuatro direcciones que van desde la 2000h hasta 2003h y es en donde el usuario puede guardar información de identificación (ID). Se recomienda solo usar los cuatro bits menos significativos de cada localidad ID. En algunos dispositivos las localidades ID se leen hacia fuera en un modo descifrado, después que el código de protección es habilitado.

Para estos dispositivos se recomienda que la localidad ID se escriba como "11 1111 1000 bbbb", donde el bbbb es información ID. En otros dispositivos las localidades ID se leen normalmente aún después de ser activado el código de protección.

Palabra de configuración, Los PIC 16F87XA tienen diferentes bits de configuración. Estos bits pueden setearse (se leen 0L) o ser inalterados (se leen 1L), para seleccionar diferentes configuraciones del dispositivo.

R/P-1	U-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	U-1	U-1	R/P-1	R/P-1	R/P-1	R/P-1
CP	-	DEBUG	WRT1	WRT0	CPD	LVP	BOREN	-	-	PWRTEN	WDTEN	FOSC1	FOSC0
bit 13													bit 0

Figura 2.5. Palabra de Configuración del Microcontrolador.

Lo que se explica a continuación las consecuencias que tiene poner 0L o 1L en cada uno de los 14 bits de la Palabra de Configuración para los PICs 16F877A, las variantes que existen dependen de la seguridad que se le quiera dar al programa, tipo de programación, tipo de oscilador, watch dog timer, entre otros, para un correcto funcionamiento del microcontrolador.

a) bit 13 CP: Bit de Código de Protección para la memoria Flash.

PIC16F877A/876A:

1= Código de Protección desactivado.

0 = Código de Protección desde 0000h a 1FFFh.

PIC16F874A/873A:

1= Código de Protección desactivado.

0 = Código de Protección desde 0000h a 0FFFh.

1000h a 1FFFh envueltas a 0000h a 0FFFh.

b) bit 12 **No está implementado:** Se lee como 1L

c) bit 11 Bit de Modo Depurador

1= Desactivado

0= Activado. La depuración se realiza desde MPLAB.

d) bits 10-9 Bit de Permiso de Escritura de la Memoria de Programa Flash

PIC16F877A/876A:

11= Protección de escritura apagada.

10= 0000h a 00FFh se protege, 0100h a 1FFFh puede ser modificada por el control EECON

01= 0000h a 07FFh se protege, 0800h a 1FFFh puede ser modificada por el control EECON

00= 0000h a 0FFFh se protege, 1000h a 1FFFh puede ser modificada por el control EECON

e) bit 8 Código de Protección de la Memoria de Datos EEPROM.

1= No hay protección en la EEPROM.

0= Protección de los datos en la EEPROM.

f) bit 7 Bit LVP: Habilitación de la programación en Bajo Voltaje.

1= El pin RB3 tiene el control de la habilitación de la grabación en Bajo Voltaje

0= El pin RB3 es una entrada o salida digital y la programación se realiza en Alto Voltaje.

g) bit 6 Bit BOREN: Permiso de reset por Caída de Tensión VDD.

1= BOR activada.

0= BOR desactivada.

h) bit 5-4 **No implementados:** Se leen como 1L

i) bit 3 Bit PWRTEN: Habilitación del temporizador de arranque de 72 ms.

1= PWRTEN habilitado.

0= PWRTEN deshabilitado.

j) bit 2 Bit WDTEN: Habilitación del Perro Guardián.

1= WDT habilitado.

0= WDT deshabilitado.

k) bit 1-0 Tipo de Oscilador:

11= Oscilador RC, 10= Oscilador HS, 01 = Oscilador XT y 00= Oscilador LP.

En la Figura 2.6, que se muestra el mapa de memoria de los PIC16F87XA, esta permite aclarar los conceptos explicados anteriormente, tales como la memoria de programa, la memoria de datos, las localidades ID, la palabra de configuración, en fin el tamaño de la memoria del microcontrolador y su distribución.

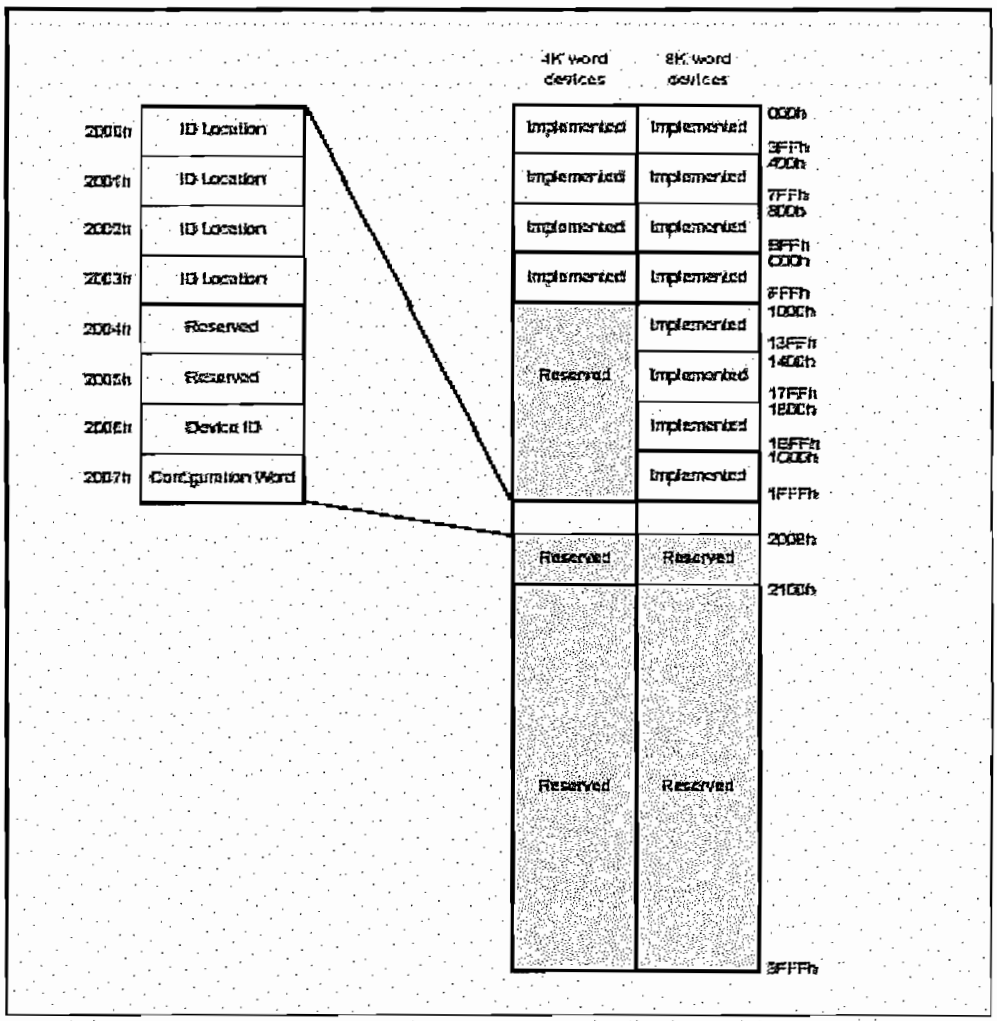


Figura 2.6. Mapa de Memoria del PIC16F87XA.

2.2.1 MODO DE PROGRAMACIÓN / VERIFICACIÓN

Con el afán de que este trabajo sirva como referencia para los interesados en la temática, a continuación se detallan aspectos que son útiles considerar para programar el PIC.

Como ya se dijo antes toda la programación se realizará mediante el algoritmo de alto voltaje. Una vez en este modo se puede acceder y programar la memoria de programa de usuario y su configuración en forma serial.

En el modo indicado RB6 y RB7 son entradas Schmitt Trigger. La secuencia para entrar en este modo pone toda otra lógica en RESET, esto significa que todas las I/O están en un estado de reset (entradas en alta impedancia). Un dispositivo reseteado limpia el PC (counter program) y lo pone en la dirección 0000h. El comando Incrementar Dirección (Increment Address) sirve para incrementar el PC. El comando Cargar Configuración (Load Configuration) puede poner al PC en 2000H.

La secuencia normal para programar las ocho palabras de memoria es:

1. En la Figura 2.7, se puede observar que previo a la palabra se debe usar el comando Cargar dato (Load Data) para cargar una palabra en la dirección de la memoria de programa.

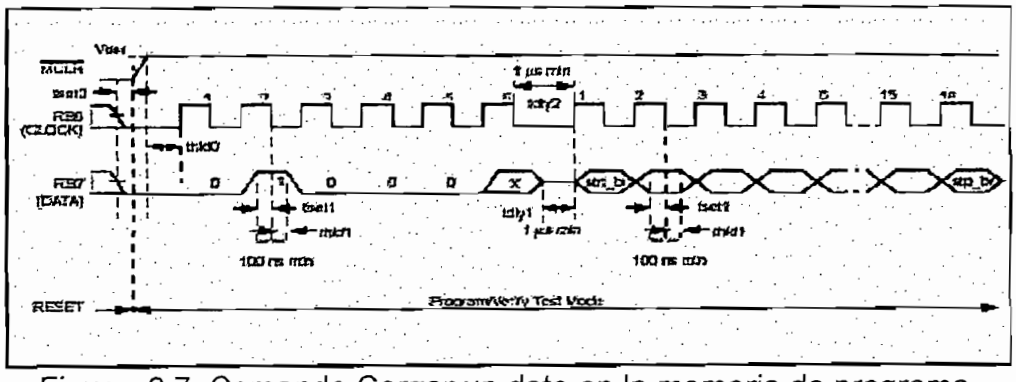


Figura. 2.7. Comando Cargar un dato en la memoria de programa.

2. Se emite un comando Incremento de dirección o Increment Address. Como se puede ver en la Figura 2.8.

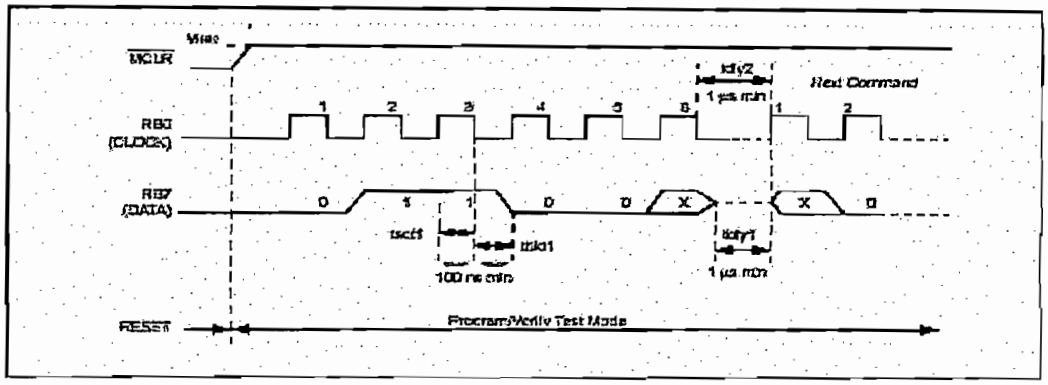


Figura 2.8. Comando Incrementar dirección.

3. Volver a cargar la siguiente palabra en la dirección de memoria de programa actual usando el comando Cargar Dato (Load Data) como se muestra en la Figura 2.7.
4. Se repite el paso 2 y paso 3 seis veces más, con lo cual se completan las primeras ocho localidades de la memoria de programa (se programa 8 palabras por vez, ya que las direcciones de este dispositivo se disponen de forma de una Matriz de nx8, donde n es un número entero correspondiente a la capacidad de la memoria de usuario).
5. Se emite el comando Empezar solamente a programar (Begin Programming Only) como se indica en la Figura 2.9.

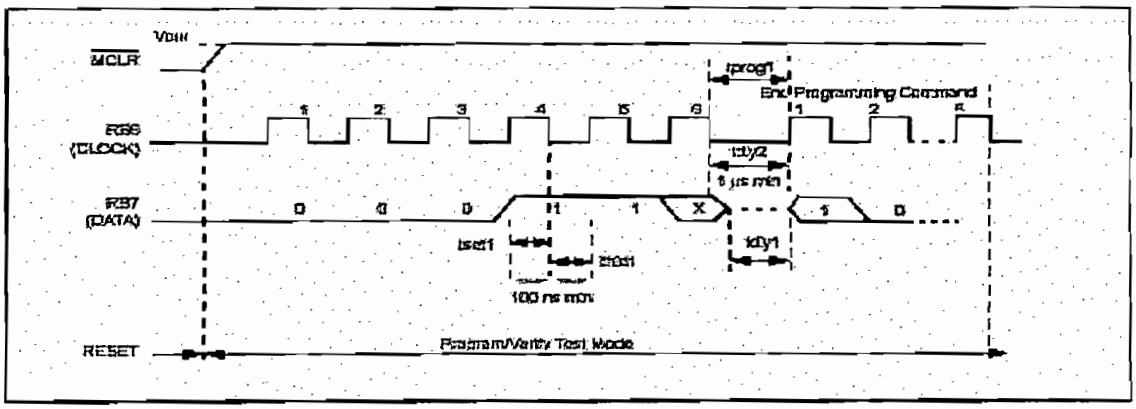


Figura 2.9. Comando Empezar solamente a programar.

6. Se espera un tprog (aproximadamente 1ms, de acuerdo con las especificaciones requeridas para este microcontrolador y resumidas en Tabla 2.2).

7. Se emite un comando Fin de programación (End Programming). Como se puede observar en la Figura 2.10.

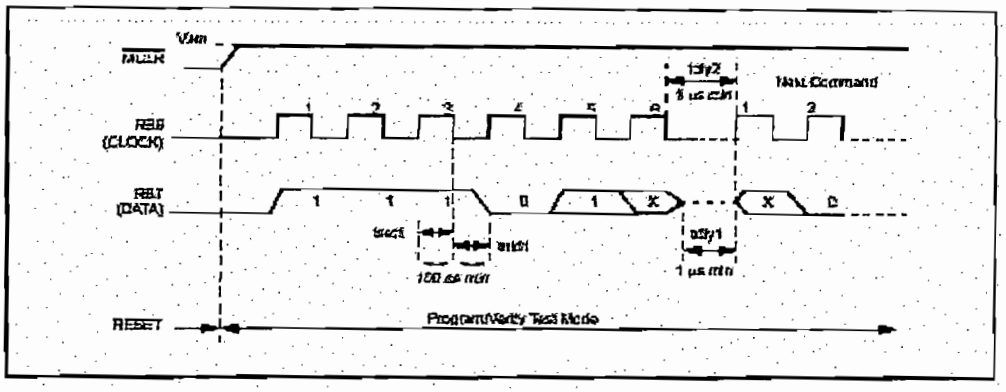


Figura 2.10. Comando Fin de programación.

8. Se incrementa la siguiente dirección.
9. Por último se repite esta secuencia como requerimiento para escribir el programa y la configuración de memoria.

La secuencia alternativa para programar una palabra de memoria es como se indica:

1. Se pone una palabra en la localidad de memoria actual usando el comando Cargar Datos (Load Data), igual que en la secuencia anterior (Figura 2.7).
2. Se envía el un comando Empezar solamente a programar (Begin Programming Only) de la Figura 2.9.
3. Se espera un tprog (de acuerdo con la Tabla 2.2).
4. Se envía el comando Fin de programación (End Programming) de la Figura 2.10.
5. Se incrementa la siguiente dirección (Figura 2.8).
6. Se repite esta secuencia alternativa como requerimiento para escribir el programa y la memoria de configuración.

Es importante tener en cuenta que la dirección y contador de programa se restablece (Reset) a 0000h cuando el pin MCLR del dispositivo llega a VIL o cuando se vuelva a entrar al modo de programación.

El programa y configuración de memoria puede leerse o puede verificarse usando el comando Leer datos (Read Data) de la Figura 2.11 e Incremento de dirección (Increment Adress) de la Figura 2.8.

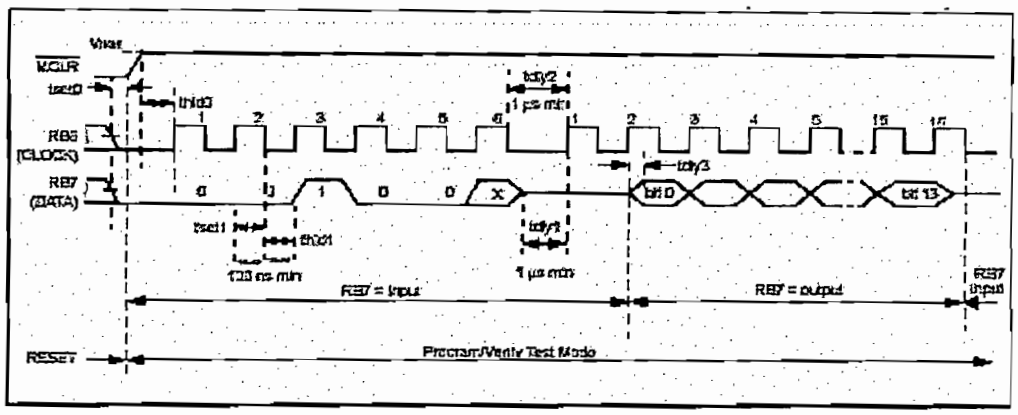


Figura 2.11. Comando Leer un dato en la memoria de programa.

Como se puede ver en las figuras de tiempo de los comandos, el pin RB6 es utilizado para ingresar los pulsos de Reloj, y el pin RB7 es usado para ingresar los comandos, y para ingresar o sacar datos durante la operación serial. Al ingresar un comando, el reloj en el pin RB6 es ciclado seis veces. Cada bit de un comando es almacenado en el flanco de bajada del reloj, con el bit Menos Significativo del comando entrando primero. El dato en RB7 requiere un tiempo mínimo de almacenamiento tset1 y un tiempo mínimo de retención (thold1) (mirar las especificaciones de la Tabla 2.2), con respecto al flanco de bajada del reloj.

Los comandos asociados con los datos (leer y cargar) se especifican para tener un retardo mínimo de 1 μs entre el comando y el dato. Después de este retardo el pin al que llega la señal de reloj es ciclado 16 veces; en el primer ciclo se emite un 0L como bit de Arranque (Star) y el último ciclo se emite un bit de parada también 0L. Los datos se transfieren empezando con el bit menos significativo (LSB). Durante una operación de lectura, el bit LSB puede ser almacenado durante el flanco de bajada del segundo ciclo. Se especifica un mínimo retardo de 1 μs (tdly2) entre los comandos consecutivos. Todos los comandos y las palabras de datos son transmitidos empezando por el bit LSB.

Los datos son transmitidos durante el flanco de subida, y almacenados mientras el flanco de bajada del reloj. Para permitir que el PIC entienda los comandos se requiere un tiempo por lo menos de 1 μs entre un comando y una palabra de datos, u otro comando. El funcionamiento de los comandos disponibles se encuentra más adelante y se listan en la Tabla 2.1.

Command	Mapping (MSB ... LSB)	Data	Voltage Range
Load Configuration	0 0 0 0 0	0,data (14), 0	2.2V - 5.5V
Load Data for Program Memory	0 0 0 1 0	0, data (14), 0	2.2V - 5.5V
Read Data from Program Memory	0 0 1 0 0	0, data (14), 0	2.2V - 5.5V
Increment Address	0 0 1 1 0		2.2V - 5.5V
Begin Erase/Programming Cycle	0 1 0 0 0	4 ms typical, internally timed	2.2V - 5.5V
Begin Programming Only Cycle	1 1 0 0 0	1 ms typical, externally timed	4.5V - 5.5V
Bulk Erase Program Memory	0 1 0 0 1	4 ms typical, internally timed	4.5V - 5.5V
Bulk Erase Data Memory	0 1 0 1 1	4 ms typical, internally timed	4.5V - 5.5V
Chip Erase	1 1 1 1 1	4 ms typical, internally timed	4.5V - 5.5V
Load Data for Data Memory	0 0 0 1 1	0, data (14), 0	2.2V - 5.5V
Read Data from Data Memory	0 0 1 0 1	0, data (14), 0	2.2V - 5.5V
End Programming	1 0 1 1 1		

Tabla 2.1. Listado de Comandos para Programar al PIC16F87X.

La Tabla 2.2 resume las características eléctricas que el microcontrolador PIC, necesita para su correcto funcionamiento.

AC/DC CHARACTERISTICS POWER SUPPLY PINS	**Standard Operating Procedure (unless otherwise stated)** Operating temperature 0 ≤ TA ≤ +70°C Operating Voltage 2.0V ≤ VDD ≤ 5.5V					
Characteristics	**Sym**	**Min**	**Typ**	**Max**	**Units**	**Conditions/Comments**
General						

VDD level for Begin Erase/Program operations and EECON write of program memory	VDD	2.0		5.5	V	
VDD level for Begin Erase/Program operations and EECON write of data memory	VDD	2.0		5.5	V	
VDD level for Bulk Erase/Write, Chip Erase, and Begin Program operations, of program and data memory	VDD	4.5		5.5	V	
Begin Programming Only cycle time	tprog1	1			ms	Externally Timed
Begin Erase/Programming	tprog2		4	8	ms	Internally Timed
Chip Erase cycle time	tprog3		4	8	ms	Internally Timed
High voltage on MCLR and RA4/T0CKI for Test mode entry	VIHH	VDD + 3.5		13.5	V	
MCLR rise time (VSS to VHH) for Test mode entry	tVHHR			1.0	µs	
(RB6, RB7) input high level	VIH1	0.8VDD			V	Schmitt Trigger input
(RB6, RB7) input low level	VIL1	0.2VDD			V	Schmitt Trigger input
RB<7:4> setup time before MCLR↑ (Test mode selection pattern setup time)	tset0	100			ns	
RB<7:4> hold time after MCLR↑ (Test mode selection pattern setup time)	thld0	5			µs	
Serial Program/Verify						
Data in setup time before clock↓	tset1	100			ns	
Data in hold time after clock↓	thld1	100			ns	
Data input not driven to next clock input (delay required between	tdly1	1.0			µs	2.0V ≤ VDD < 4.5V

command/data or command/command)		100			ns	4.5V ≤ VDD ≤ 5.5V
Delay between clock↓ to clock↑ of next command or data	tdly2	1.0			µs	2.0V ≤ VDD < 4.5V
		100			ns	4.5V ≤ VDD ≤ 5.5V
Clock↑ to data out valid (during read data)	tdly3	80			ns	

Tabla 2.2. Requerimientos de tiempo para el modo programación verificación.

Los comandos que se explican a continuación, sirven para programar al PIC y, saber como se ejecutan es importante para un mejor entendimiento del Capítulo 4, del diseño del Software.

Cargar Datos para la Memoria de Programa

Después de recibir este comando el chip carga una palabra (14 bits constituyen una palabra de datos), a ser programada en la memoria de programa cuando se transcurren 16 ciclos de los mostrados en la Figura 2.7.

Cargar datos en la memoria de datos

Después de recibir este comando, el chip puede cargar 14 bits de una palabra de datos cuando se aplican 16 ciclos. Sin embargo la memoria de datos EEPROM es solamente de 8 bits de ancho, y así sólo los 8 primeros bits de datos después del STAR, pueden ser programados en la memoria de datos. Vale indicar que es necesario aplicar los 16 ciclos para que la circuitería interna funcione bien. La memoria de datos contiene 256 Bytes. Si el PIC tiene activado el código de protección al leer los datos, estos se leen como ceros.

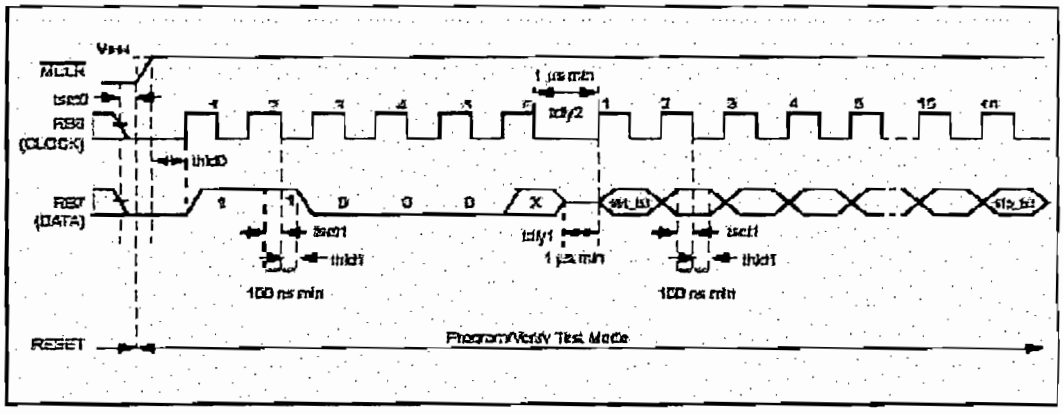


Figura 2.12. Comando Cargar un dato en la memoria de datos.

Leer datos desde la Memoria de Programa

Después de recibir este comando, el chip puede transmitir datos de la memoria de programa (usuario o configuración), empezando en el segundo flanco de subida del reloj. El pin RB7 puede ir al modo de salida en el segundo flanco de subida del reloj y regresar al modo de entrada (alta impedancia) después del 16 avo flanco de subida. Un diagrama de tiempo de este comando se muestra en la Figura 2.11.

Leer datos desde la Memoria de datos

Después de recibir este comando, el PIC transmite los bits de los datos fuera de su memoria de datos (EEPROM), empezando con el segundo flanco de subida del reloj. El pin RB7 va al modo de salida en el segundo flanco de subida y se revierte al modo de entrada (alta impedancia) después del 16avo flanco de subida. Como previamente se dijo, la memoria de datos es de 8 bits de ancho, y por consiguiente, sólo los primeros 8 bits de salida son datos reales. Un diagrama de tiempo de este comando se muestra en la Figura 2.13.

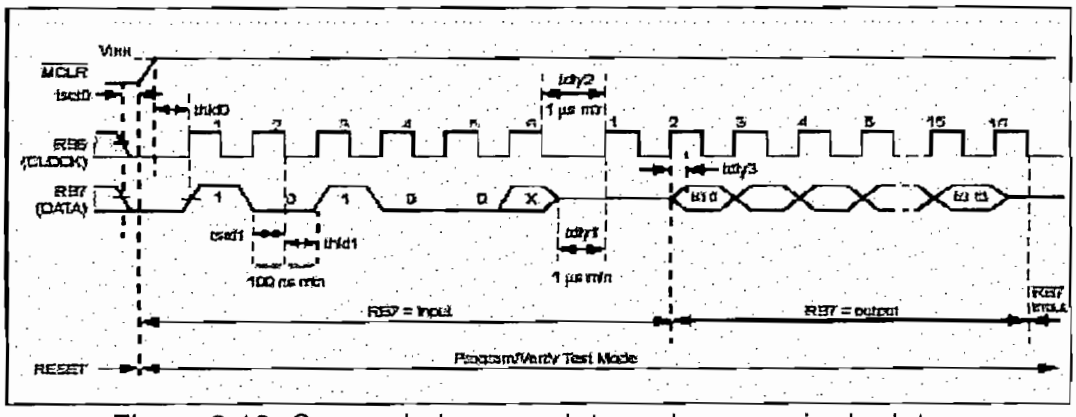


Figura 2.13. Comando Leer un dato en la memoria de datos.

Incremento de la Dirección

El contador de programa se incrementa cuando se recibe este comando, en la Figura 2.8 se muestra el diagrama de tiempo de este comando

Fin de programación

Después de recibir este comando, el chip detiene la programación de la memoria (la memoria de prueba o la memoria de usuario) que estaba siendo programando en ese instante.

Nota: Este comando también cambia todos los datos escritos en el latch a 1L para evitar problemas con solamente la descarga de una palabra antes de escribirla. En la Figura 2.10 se observa el diagrama de tiempos de este comando.

2.2.2 MODO BORRAR/PROGRAMAR

Las ocho localidades deben cargarse antes de cada comando Empezar a Borrar/Programar (Begin Erase/Programming). Después de esto, el comando es recibido y descifrado, ocho palabras de la memoria de programa se borrarán y se programarán con los valores cargados antes. La dirección de PC descifrará que 8 palabras son programadas. Los tres bits mas bajos del PC se ignoran, así si el PC apunta a la dirección 003h, se escriben las ocho localidades desde 000h a 007h. Un mecanismo interno temporizado ejecuta un borrado antes de escribir. El usuario debe permitir una combinación de tiempo para borrar y programar, como

está especificado en las características eléctricas de la Tabla 2.2, para poder completar la programación. Aquí no se necesita el comando Fin de programación (End Programming).

1. Si la dirección está apuntando a la memoria de usuario, sólo dicha memoria será afectada.
2. Si la dirección está apuntando a la memoria de prueba que esta implementada físicamente (2000h -201Fh), dicha memoria será escrita. La palabra de configuración no será escrita a menos que la dirección este apuntando específicamente a 2007h.

Nota:

a) Los bits del código de protección no pueden ser borrados con este comando.

b) Todas las operaciones Empezar a Borrar/Programar (Begin Erase/Programming) tienen lugar o suceden encima del rango de VDD.

Un diagrama de tiempo de este comando es presentado en la Figura 2.14:

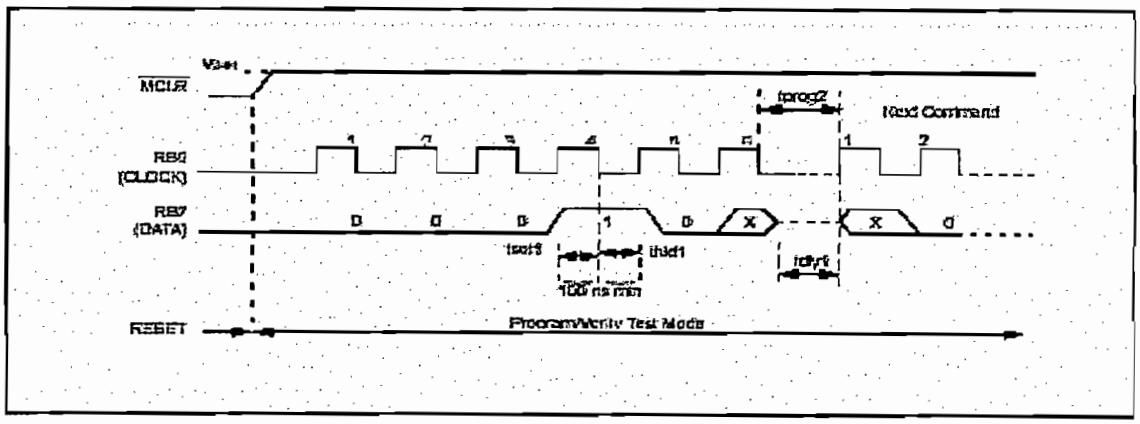


Figura 2.14. Comando Empezar a borrar/programar.

2.2.3 MODO SOLAMENTE PROGRAMAR

Las operaciones relacionadas con este Modo tienen lugar en el rango de 4.5V a 5.5V de VDD.

Este comando es similar al comando Empezar el ciclo Borrar / programar, excepto que en este caso no es necesario borrar previamente la localidad a ser programada y el temporizador interno no se usa. Después de que llega este comando se puede programar la memoria de programa y la memoria de datos. El usuario debe esperar o dar un tiempo para la programación como se especifica en las características eléctricas de la Tabla 2.2, para una programación completa. Aquí se requiere el comando Fin de Programación (End Programming), ya que no se utiliza el temporizador interno para detener la programación.

1. Si se está apuntando a la memoria de usuario, sólo esta será afectada.
2. Si se está apuntando a la memoria de prueba (que está físicamente implementada de la 2000h-201Fh), esta será escrita. La palabra de configuración no se escribirá a menos que la dirección esté específicamente apuntando a 2007h. Un diagrama de tiempo para este comando es indicado en la Figura 2.9.

2.2.4 BORRADO MASIVO DEL MICROCONTROLADOR

Cuando se ejecuta el comando Chip Erase, se borra la memoria de programa, los datos de la EEPROM, y los bits del código de protección. Todo en el chip, memorias FLASH y EEPROM son borradas, sin tener en cuenta la dirección que tuvo el PC (contador de programa).

Si el PC apunta a la memoria de usuario, la fila de prueba (2000h hasta 201Fh) no es borrada con el comando Chip Erase, excepto la palabra de configuración (en 2007h). Si se desea borrar la fila de prueba totalmente la dirección del PC debe apuntar a la memoria de configuración. Cuando el PC apunta a 2000h – 201Fh, la palabra de configuración, la memoria de prueba y la memoria de programa de usuario pueden ser borradas con el comando Chip Erase. Esto permite al usuario borrar todo el programa y el contenido de la configuración, incluyendo los bits del código de protección, sin comprometer los bits de las localidades ID (2000h a 2004h), o cualquier código de paso en las filas de prueba.

Un diagrama de tiempo de este comando es mostrado en la Figura 2.15.

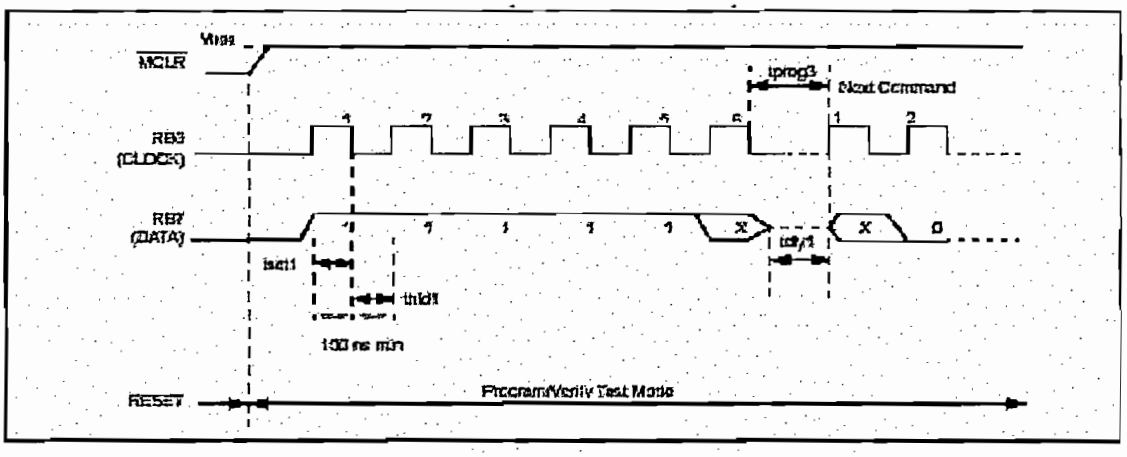


Figura 2.15. Comando de borrado completo y total del chip.

Nota: El comando Chip Erase toma lugar en un rango de voltaje para VDD entre 4.5V a 5.5V.

Todos estos comandos se generan a través del puerto paralelo de la computadora, como la mayoría de programadores de PICs. Como ya se explicó anteriormente estos son los comandos que se usarán en el software que se desarrolló para el presente proyecto y que se explicará detalladamente en el Capítulo 4.

CAPÍTULO 3

DISEÑO DEL HARDWARE DEL PLC

3.1 INTRODUCCIÓN

Este capítulo está dedicado al diseño de los circuitos necesarios para el funcionamiento del PLC desarrollado en el presente proyecto, cuya constitución es la que se muestra en la Figura 3.1:

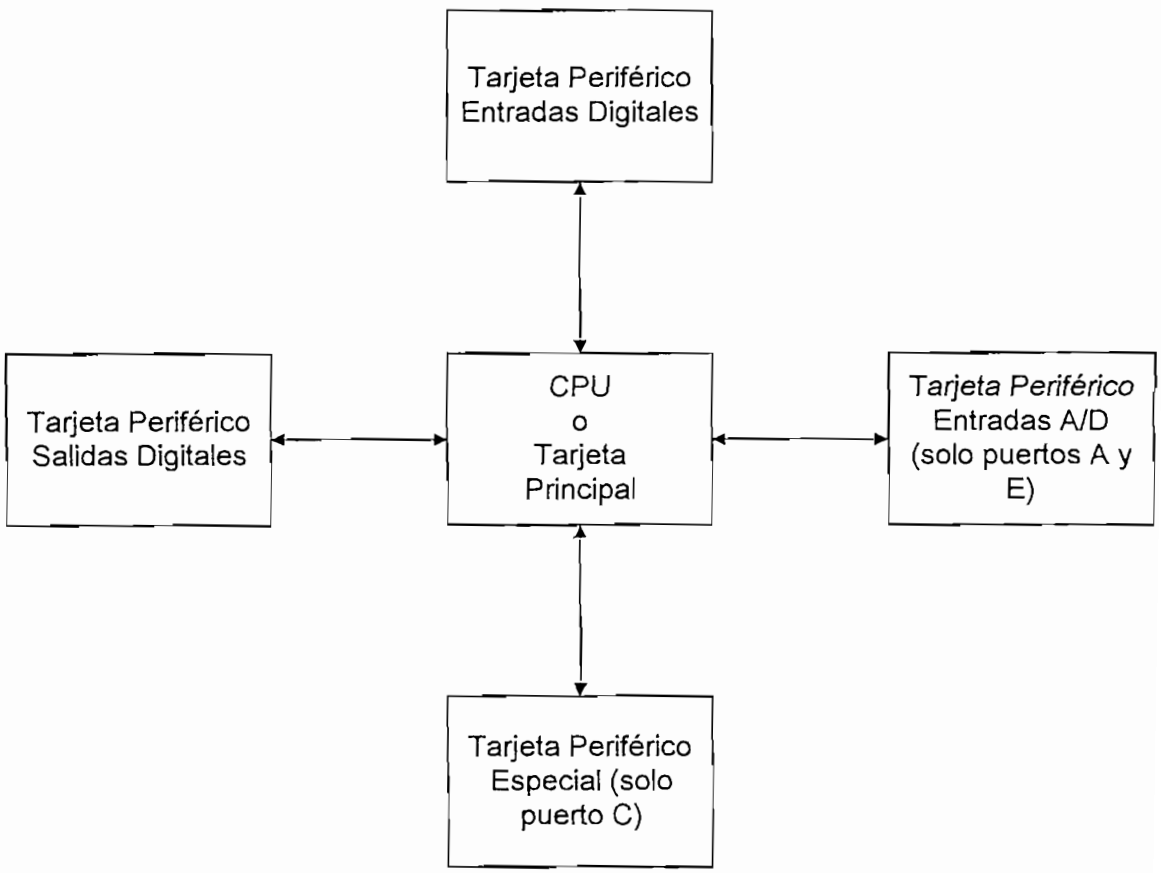


Figura 3.1. Disposición modular del PLC.

3.2 DISEÑO DEL CIRCUITO DE LA UNIDAD CENTRAL DE PROCESO (CPU)

El diseño de este circuito está dividido en dos partes: diseño del circuito que permite al microcontrolador funcionar en modo "RUN" (modo de funcionamiento normal) y el circuito para el modo de programación. Para conmutar entre estos dos modos se necesita un switch ya sea electrónico o mecánico. La solución que se escogió fue mediante un switch mecánico como el que se puede ver en la Figura 3.2, esta opción es la que actualmente usan los PLCs de la Industria.

Para el modo "RUN" el microcontrolador necesitará tener, según especificaciones eléctricas: polarización de 5Vdc en los pines V_{DD} (pin 11 y pin 32), 0V en los pines V_{SS} (pin 12 y pin 31) y conexión a un circuito típico de reseteo conformado por una red RC en el pin MCLR (pin 1).

El diseño del circuito de reset que forma parte de la Figura 3.2 se lo hizo de la siguiente manera: el valor del capacitor C1 se asumió de 1μF, mientras el de la resistencia R1 debía cumplir con un valor menor a 40 kΩ; estos valores son recomendados por el fabricante. Del rango para la resistencia R1 se tomó un valor de 10kΩ. La resistencia R2 en la Figura 3.2 es para limitar la corriente que entra al microcontrolador, MICROCHIP aconseja poner una resistencia comprendida entre un valor mayor o igual a 1kΩ, del cual se escogió el mínimo; esto es, 1kΩ para garantizar que el voltaje caiga a un valor que el microcontrolador reconozca como cero lógico.

En la Figura 3.2 también se puede ver la conexión del cristal oscilador de 4MHz en los pines OSC1 y OSC2 (pin 13 y 14 respectivamente). Estos pines se desacoplan a tierra mediante 2 capacitores de 15pF como lo recomienda el fabricante del microcontrolador.

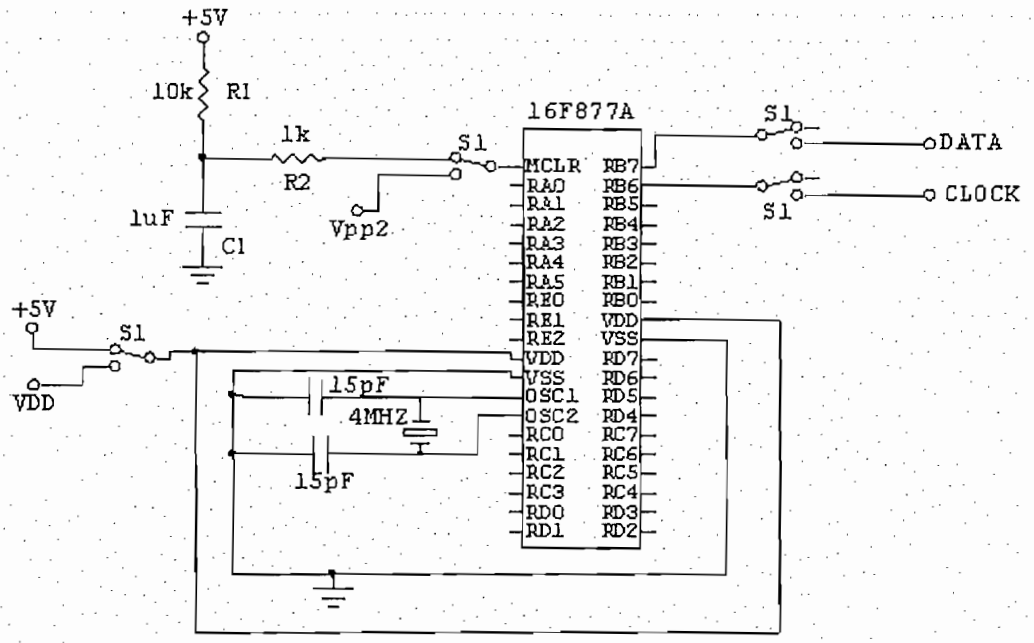


Figura 3.2. *Circuito de modo "RUN", los interruptores sirven para cambiar al modo de programación.*

En el modo de programación se necesita conmutar con el switch S1 los pines que se indican a continuación, hacia los circuitos que se explica mas adelante:

- El pin MCLR/VPP (pin 1) al circuito VPP2, voltaje de programación >13Vdc.
- Los pines VDD (pines 32 y 11) al circuito VDD, voltaje de polarización +5Vdc.
- El pin RB7 (pin 40) al circuito DATA, señales para programación.
- El pin RB6 (pin 39) al circuito CLOCK, señal de reloj.

La memoria del microcontrolador es de tipo flash por lo tanto se diseñó un circuito que permite grabar y borrar eléctricamente dicha memoria. En Internet se encuentran diversos circuitos para programar microcontroladores PIC, de los cuales se escogió el hardware del programador ICPROG (software) para puerto paralelo, ya que se trata de un circuito sencillo cuyo funcionamiento ha sido comprobado.

En la Figura 3.3 se presenta el circuito programador antes mencionado.

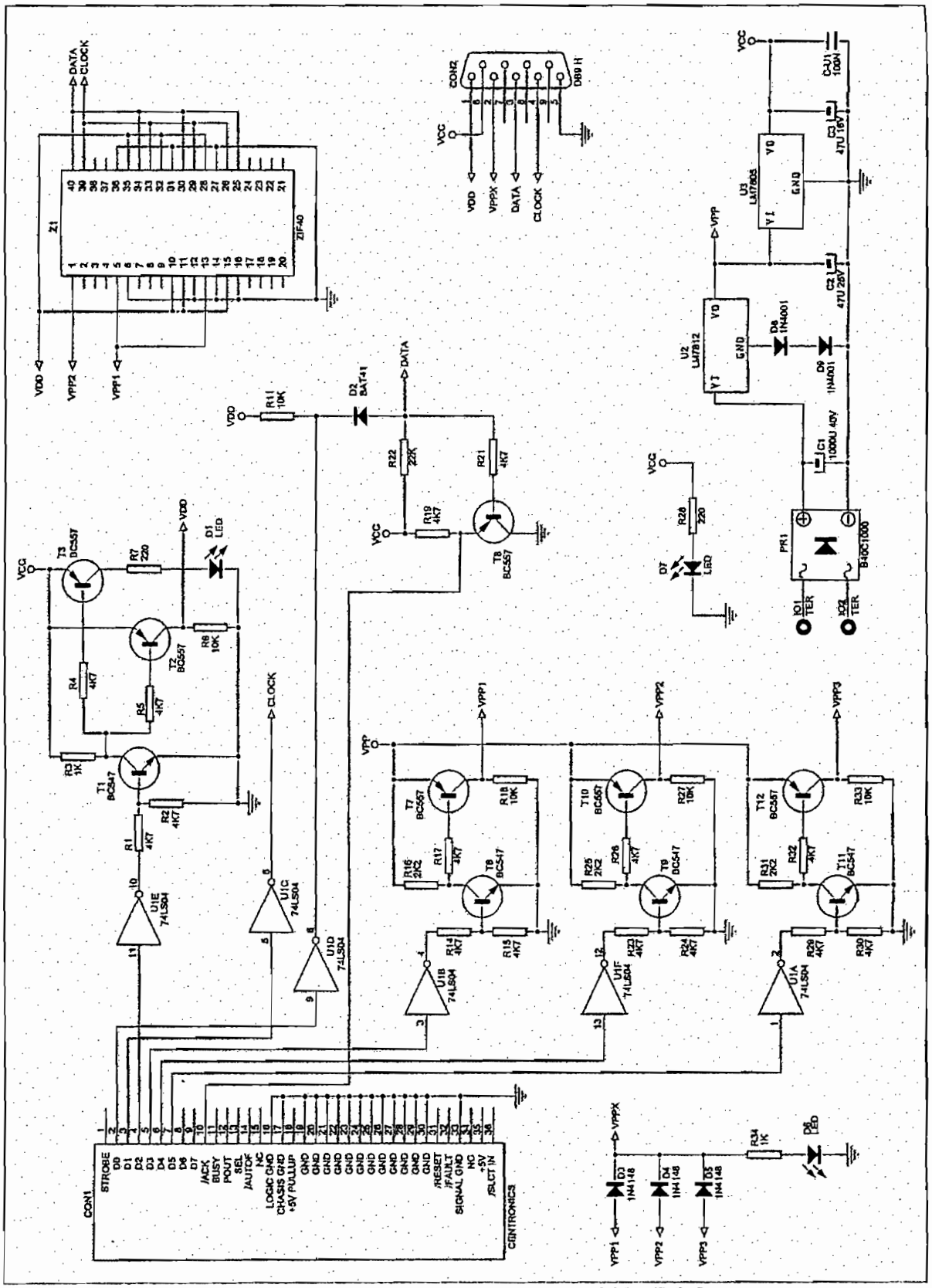


Figura 3.3. Circuito Programador del ICPROG.

Este circuito sirve para programar a los diferentes tipos de PICs que existen, desde los de 8 pines hasta los de 40 pines, pero como se va a programar

únicamente al PIC16F877A, las partes que se necesitan de este circuito son aquellas que proporcionan el control de: VDD, VPP2, DATA y CLOCK.

El circuito que controla el voltaje VDD se muestra en la Figura 3.4. Este circuito conecta el pin D2 (pin 4) del bus de datos a los pines VDD del PIC, mediante una compuerta inversora NOT 74LS04. La función de esta compuerta es proteger y aislar al puerto paralelo. Se puede ver también que hay un LED, que facilita al usuario saber si este voltaje está o no presente.

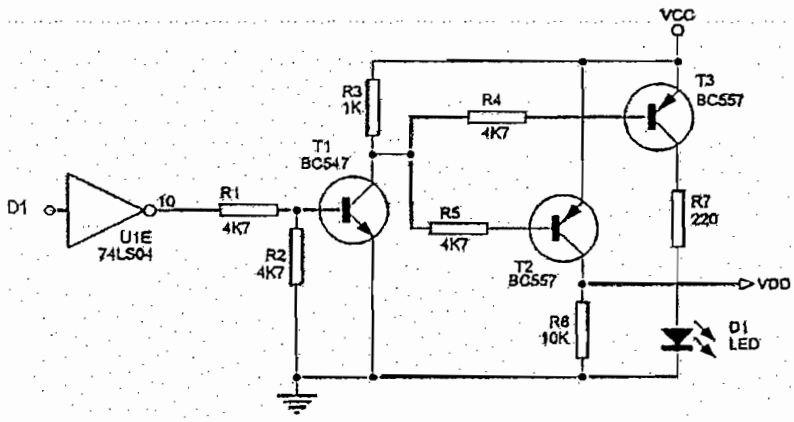


Figura 3.4. Circuito que controla el voltaje VDD al microcontrolador.

El circuito que controla el voltaje de programación VPP2, que es con el cual se programa al microcontrolador y tiene que ser mayor a 13V, como ya se explicó anteriormente. Se muestra en la Figura 3.5, este circuito conecta el pin D4 (pin 6) del bus de datos del puerto paralelo al pin MCLR del PIC, mediante una compuerta inversora NOT 74LS04. En este caso también hay un diodo LED. El voltaje VPP que se observa en la Figura 3.5 es el voltaje de polarización de +12Vdc.

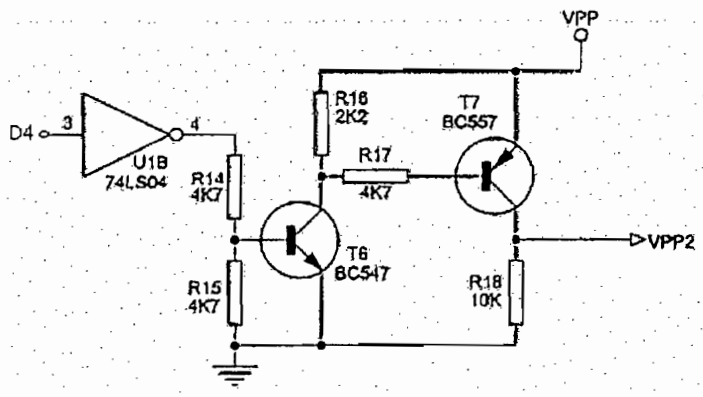


Figura 3.5. Circuito que controla el voltaje de programación VPP2.

Según los requerimientos eléctricos para programar al PIC (Tabla 2.2), VPP2 debe estar comprendido entre 13 y 14 voltios. Para este propósito se utilizó un regulador LM7812 que tiene conectado dos diodos en serie al pin GND (pin 3) del regulador y sirve para obtener 1.2V, que sumados al voltaje real del regulador, que es aproximadamente 11.9V, se consigue un voltaje de 13.1Vdc. A este regulador se conectan también dos condensadores C1 y C2, uno a la entrada y el otro a la salida, respectivamente. Estos sirven para filtrar la señal de DC. Para alimentar al regulador se tomó 15Vdc de la fuente switching disponible. Como se puede ver esta es una buena solución para obtener el voltaje que requiere el PIC, ya que dentro de los estándares no existe un regulador de 13 ó 14V como integrado. En la Figura 3.6 se indica el circuito diseñado.

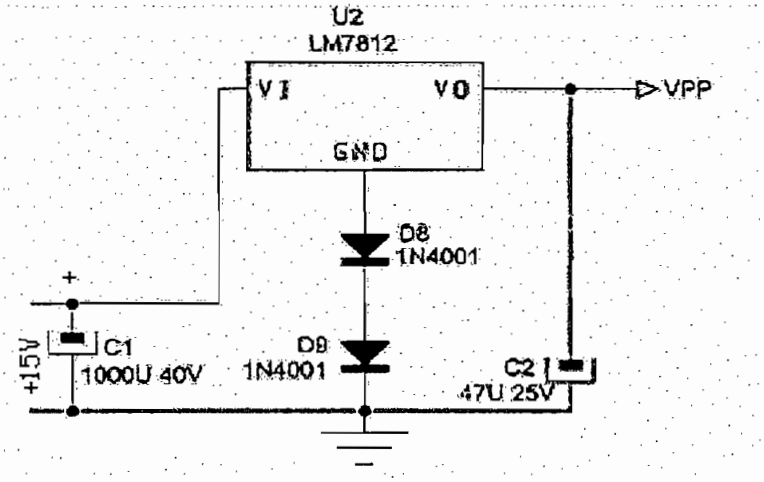


Figura 3.6. Circuito que provee el voltaje necesario para programar al PIC, en este caso 13.1V.

La transmisión de comandos y datos hacia el PIC es de tipo half-duplex, lo que quiere decir que por un mismo pin se transmite y se recibe la información. Sin embargo, como es de conocimiento, las líneas del puerto de datos del puerto paralelo son unidireccionales. Por ello se requiere un circuito que pueda conectar la línea de DATA del PIC, hacia dos pines del puerto paralelo, el uno de salida cuando se esté programando al PIC y el otro de entrada cuando se desee leer la información del microcontrolador. Este circuito se aprecia en la Figura 3.7, en el que se puede ver que para la transmisión de datos al PIC desde la computadora se escogió el pin D0 (pin 2) del bus de datos del puerto paralelo, y para la recepción o lectura de datos del PIC se eligió el pin ASK (pin 10), que es una de

las cuatro líneas de entrada del puerto conocidas como líneas de estado. Es importante destacar el siguiente punto, cuando el PIC esté transmitiendo datos hacia el computador el D0 del puerto paralelo tiene que estar en 0L para que la salida de la compuerta esté en 1L y de esta manera el diodo D2(BAT 41) esté polarizado inversamente y no represente un corto circuito para la línea de DATA del PIC.

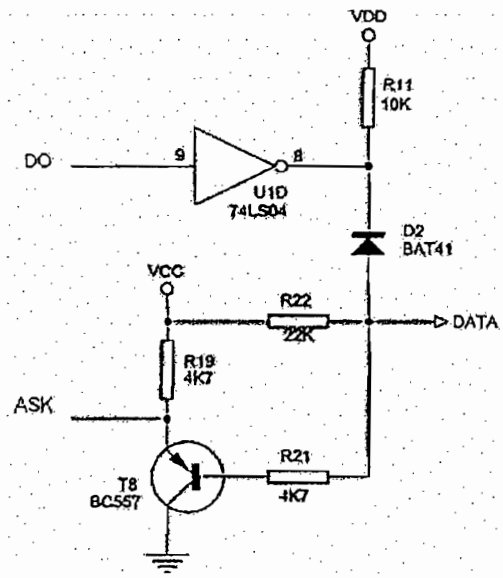


Figura 3.7. Circuito controlador de la línea de DATA.

Para transmitir la señal de sincronización de reloj se utiliza únicamente una compuerta 7404, como se puede ver en la Figura 3.8.

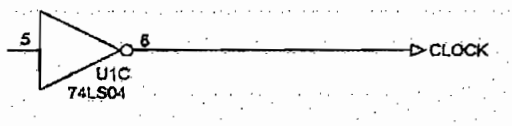


Figura 3.8. Circuito de la señal de reloj que se obtiene del pin D1 (pin 3), del bus de datos del puerto paralelo.

Nota: Debido a que se utiliza un circuito integrado TTL de compuertas NOT, la información que llega al PIC, por software, es invertida; así mismo, al arrancar el software que programa al PLC, las líneas que se utilizan para el circuito programador se ponen automáticamente en 1L para que los circuitos que controlan VPP, VDD, Data y CLOCK queden deshabilitados

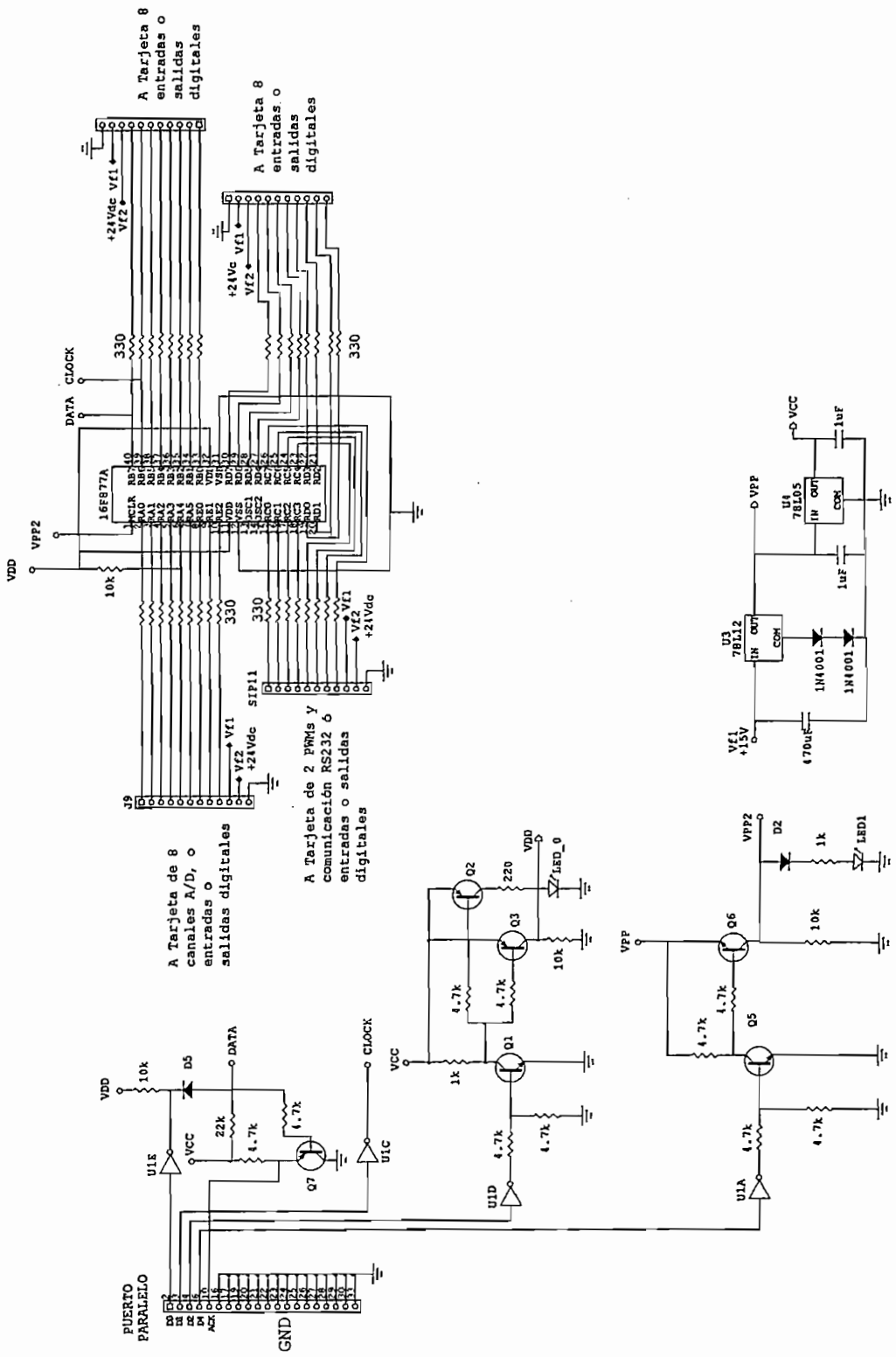


Figura 3.9. Circuito total del programador para el PIC16F877A

NOTA: Todos los puertos del PIC tienen conectados resistencias de 330 ohmios para proteger a estos en caso de sobre corriente.

3.3 DISEÑO DE LOS MÓDULOS DE ENTRADA DIGITALES Y ANALÓGICOS

Como ya se explicó en el diagrama de la Figura 3.1, el PLC diseñado en el presente proyecto está sujeto a una estructura modular, que facilita sobre todo el uso de tarjetas de entradas o salidas digitales, en cualquiera de los 4 puertos. A continuación se explica detalladamente su diseño.

3.3.1 DISEÑO DE LAS ENTRADAS DIGITALES

Las entradas digitales son señales de voltaje de 0V para 0L y de 24V para 1L, se escogió este voltaje por ser estándar en la mayoría de los PLCs. Pero como las entradas digitales que soporta el PIC son de 0V para 0L y 5V para 1L, fue necesario acondicionar las señales de entrada para que se ajuste al estándar. Para solucionar este problema se tuvieron las siguientes opciones:
Como primera opción se probó un circuito divisor de tensión. En este circuito se eligió las resistencias de 20k y 4.7k que son para crear una caída de tensión de 5V con un consumo bajo de corriente. Para aislar la señal de 5V se utilizó un seguidor de tensión constituido por un amplificador operacional y a la salida de dicho amplificador, para protección de sobre tensiones, se puso un limitador de tensión conformado por una resistencia de 70 ohms y un diodo zener de 5.1V tal como se muestra en la Figura 3.10.

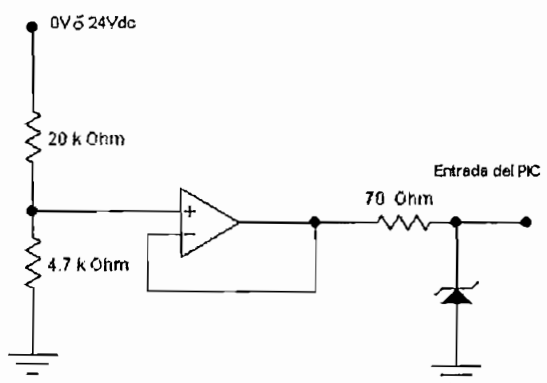


Figura 3.10. Circuito acondicionador para las entradas de digitales, con seguidor de tensión.

Sin embargo este circuito presenta la siguiente desventaja: La corriente que fluye por dichas resistencias es pequeña para la fuente de 24V, pero si se ingresa un voltaje mucho mayor (por error), es posible que el amplificador operacional haga cortocircuito y ese alto voltaje pase al diodo zener, que aunque tendrá un cierto rango de protección, con un valor excesivo no protegerá al terminal de entrada del PIC. Por ello se decidió que esta no es la solución más óptima para el acondicionamiento de la señal de entrada de 24V.

Por otra parte, todos los fabricantes de PLC tienen en sus entradas digitales aislamientos de tipo óptico, es por esto que se decidió utilizar este tipo de acoplamiento. El opto acoplador que se eligió es el ECG3045 ya que es el que mejores características eléctricas posee y es económico; una de sus características es que su salida es un transistor Darlington, como se muestra en la Figura 3.11.

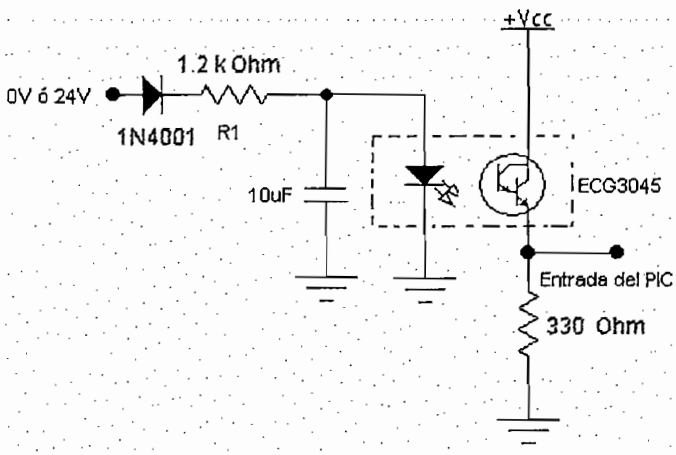


Figura 3.11. Circuito acondicionador para las entradas digitales con aislamiento óptico.

Para su diseño se emplearon los criterios siguientes:
Como se conoce, el voltaje que se necesita para activar a un led es de 1.7Vdc (ANEXO B), con un consumo de corriente de 20 mA, entonces por medio de la Ecuación 3.1, se puede calcular la resistencia R1 que limite la corriente que atraviesa el diodo.

$$R = \frac{V_{in} - V_{led}}{I_{led}} = \frac{24V - 1.7V}{20mA} = 1.12k\Omega \approx 1.2k\Omega \qquad \text{E.C. 3.1}$$

La ganancia que tiene el foto transistor, según las características eléctricas del ECG3045 (ANEXO B), es aproximadamente de 1; es decir, si en saturación la corriente de conducción es de 20mA, cumple la siguiente ecuación.

$$R = \frac{V_{cc}}{I_{colector}} = \frac{5V}{20mA} = 250\Omega \approx 330\Omega \qquad \text{E.C. 3.2}$$

Como se puede ver en la Figura 3.11 el diodo rectificador 1N4007 sirve para evitar polarizaciones inversas y el capacitor de 10uF es para eliminar el rebote que pueden introducir interruptores de tipo mecánico en los módulos de entrada.

El condesador se calcula de la siguiente manera: Según el fabricante cuando se utilice un circuito RC a la entrada de cualquier pin del microcontrolador, la constante τ=RC debe ser aproximadamente de 10ms **(2)**.

$$RC = 10ms = 1.2k\Omega \cdot C \rightarrow C = 8.333\mu F \approx 10\ \mu F \qquad \text{E.C. 3.3}$$

3.3.2 DISEÑO DE LAS ENTRADAS ANALÓGICAS

Tomando como base las características de PLCs comerciales, las entradas analógicas pueden soportar valores de 0 a 10V, por este motivo estas señales necesitan ser acondicionadas ya que de acuerdo a las características eléctricas del microcontrolador este solo admite voltajes de 0 a 5V en sus entradas analógicas. El uso de esta tarjeta, solo puede ser en el puerto A y E, que es el que posee los 8 canales analógicos.

El primer circuito que se diseñó se muestra en la Figura 3.12.

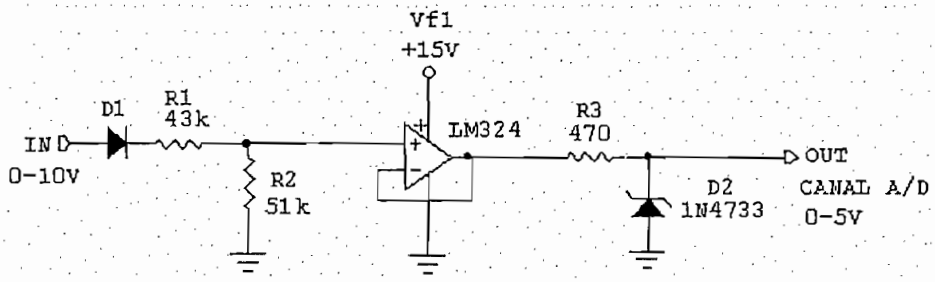


Figura 3.12. Circuito acondicionador para la entrada analógica de 0-10V.

A este circuito divisor de tensión le sigue un amplificador operacional en configuración de seguidor de tensión para acoplar las impedancias. Además esta protegido contra polarizaciones inversas con un diodo rectificador. Si no se utilizara el diodo rectificador las dos resistencias del divisor de tensión deberían ser iguales (para conseguir 5V cuando se ingrese 10V) pero por la caída de tensión de 0.7V en la juntura semiconductora del diodo, estas resistencias ya no son iguales. Es por eso que el divisor de tensión, formado por las resistencias R1 y R2 (Figura 3.12), se han calculado de tal forma que se pueda corregir este error. Si se ingresa 10V el divisor de tensión debe proveer 5V, por lo tanto se tiene que cumplir la siguiente relación:

$$\frac{R_2}{R_1+R_2}=\frac{5V}{9.3V} \qquad \text{E.C. 3.4}$$

Sí R2 es igual a 51k, entonces R1 es 43.86k la cual se le aproxima a un valor estándar de 43k.

La función del amplificador operacional es realizar un debido acople de impedancias y de alguna manera protegerlo contra sobre tensiones; además, se ha añadido también un diodo zéner de 5.1V y una resistencia (R3) para que proporcione el valor que se calcula en la Ecuación 3.5.

$$R_3=\frac{V_{sat}-V_z}{I_z}=\frac{[15-5.1]V}{20mA}=495\Omega \quad \text{Estandarizando} \quad 470\Omega \qquad \text{E.C. 3.5}$$

Donde:

- V_{sat}, es el voltaje de saturación del amplificador operacional y es igual a su voltaje de polarización (15V).
- V_z, es el voltaje del diodo zéner 5.1V
- I_z, es la corriente nominal de funcionamiento del zéner.

Sin embargo, existe un punto que no se ha considerado en el circuito acondicionador de la Figura 3.12, y es que cuando el voltaje de entrada sea menor al voltaje de conducción del diodo D1; es decir, menor que 0.7V, la lectura que va hacia el conversor A/D del microcontrolador tendrá un error pues en este rango siempre tomará el valor como 0V. Por esto, la solución que se muestra en la Figura 3.13, es retirar al diodo rectificador y tener como única protección contra voltajes negativos al amplificador operacional que, al ingresar estos valores, dará como resultado una saturación a cero voltios. Este es el caso del amplificador operacional LM324 que utiliza una sola fuente positiva.

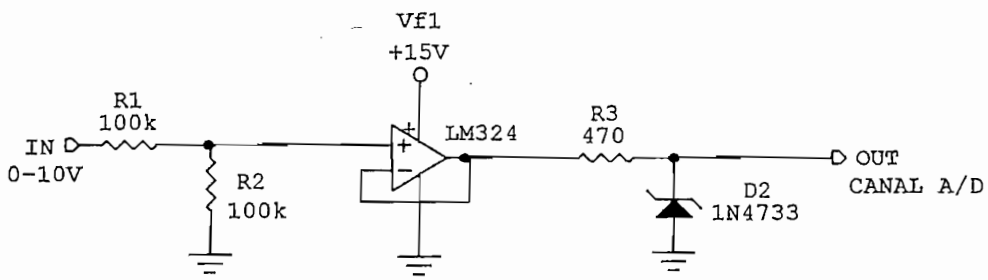


Figura 3.13. Segundo circuito acondicionador de la señal de entrada analógica.

Como se puede ver en el circuito de la Figura 3.13 anterior, ya no es necesaria ninguna compensación de error, por esto se decidió colocar dos resistencias iguales y de un valor alto (R1=R2=100k) para hacer un divisor de tensión con 5V de voltaje de salida. El resto del circuito permanece igual que el de la Figura 3.12. Aunque de esta forma el circuito tiene una protección adecuada, puede suceder que el voltaje de entrada supere el voltaje máximo de 30V. En este caso, el amplificador operacional, puede deteriorarse.

3.4 DISEÑO DE LOS MÓDULOS DE SALIDAS DIGITALES

Como ya se mencionó en el Capítulo 2, se utilizan únicamente salidas de tipo relé normalmente abierto. Este tipo de salida es versátil ya que podrá conectar a la mayoría de cargas y a voltajes comprendidos desde 0 hasta 250VAC, ó 0 hasta 30VDC. Estos módulos pueden ser utilizados en cualquiera de los puertos del microcontrolador.

El único circuito diseñado para este tipo de salidas digitales es el que se muestra en la Figura 3.14.

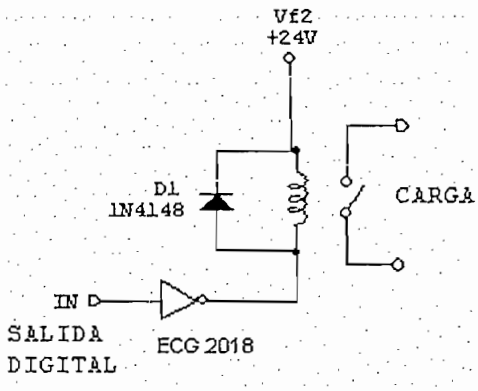


Figura 3.14. Circuito de salida digital, tipo relé.

El diodo freewheeling D1 sirve para que se libere la energía adecuadamente de la bobina. Para amplificar la corriente que el microcontrolador entrega al relé se utiliza el driver inversor ECG2018, que es como una compuerta inversora de tipo colector abierto y que puede manejar como máximo una corriente de 600mA (para mayor información favor referirse al ANEXO C). Su elección se debió a la facilidad que presta al tener ocho compuertas en un solo encapsulado. El relé se escogió después de una búsqueda entre algunos que, aunque cumplían las características que se necesitaban, tenían un tamaño muy grande para ubicar nueve en una sola tarjeta; sin embargo, favorablemente se disponía de unos que se encontró en tarjetas de un PLC Siemens dañado. Las características que según la placa estos tienen son:

Marca: TAKAMISAWA Modelo NY24W-K
Voltaje de la bobina del Relé: 24Vdc.
Tipo de contacto: Normalmente Abierto.
Carga Inductiva: 1/8 HP, 125VAC/250VAC
Carga Resistiva: 5A, 30VDC, 250VAC (carga solamente resistiva).

3.5 DISEÑO DEL MÓDULO ESPECIAL

Este módulo reúne en una sola tarjeta la comunicación serial RS-232 tipo half duplex y las dos salidas PWM. Esta se seleccionó así principalmente por fines económicos. Esta tarjeta puede ser utilizada únicamente en el puerto C.

3.5.1 DISEÑO DEL CIRCUITO DE COMUNICACION SERIAL RS-232

El circuito para la comunicación entre el PLC y una computadora necesita un intermediario, debido a que los niveles del PIC son TTL, este es el MAX 232.

En la Figura 3.15 siguiente se observa como está conectado el MAX 232 al microcontrolador y posteriormente al DB9.

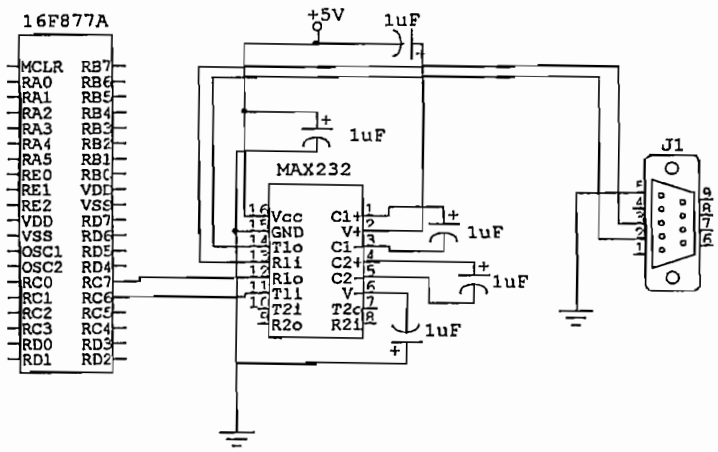


Figura 3.15. Circuito de conexión al puerto serial del PC.

3.5.2 DISEÑO DE LAS SALIDAS PWM

Las salidas PWM serán de 0-24Vdc, el primer circuito que se pensó para amplificar la señal que se obtiene del PIC que es de 0-5V, es un circuito con transistor NPN en emisor común como se presenta en la Figura 3.16.

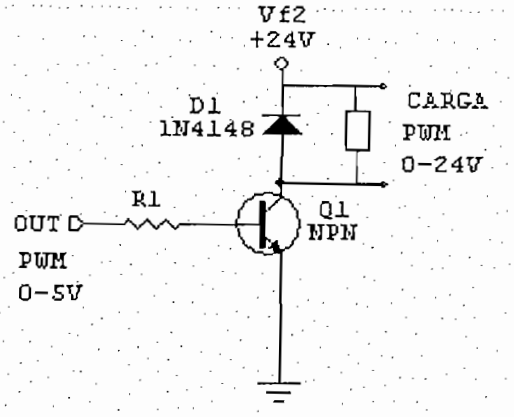


Figura 3.16. Circuito acondicionador para el PWM, con transistor NPN.

Este circuito presenta un inconveniente ya que lo que se controla es la tierra de la carga que se va a manejar y lo ideal sería que la tierra de la carga esté siempre conectada a la tierra del PLC, por lo cual la solución a este problema será el circuito de la Figura 3.17.

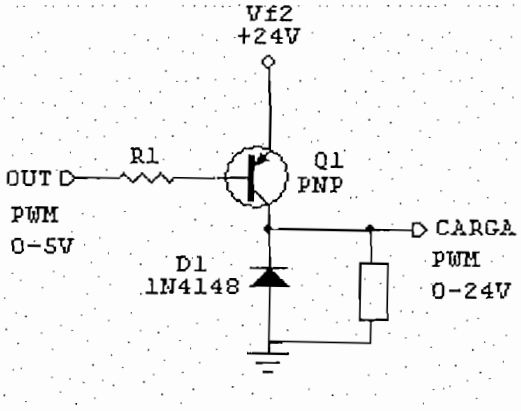


Figura 3.17. Segundo circuito acondicionador de la señal PWM, con transistor PNP.

Este circuito también tiene un problema, ya que debido a que el transistor de potencia es de tipo PNP (para mayor información de este elemento, por favor

referirse al ANEXO D), para activarlo se necesita 0L en la base y para apagarlo 1L; es decir, la señal del PWM saldría invertida en la carga. Esto podría solucionarse por software, complementando el valor que se envía al PWM y que es una palabra de 0-255. Para evitar esto se utilizó un circuito inversor en la base de un transistor NPN en configuración emisor común, como el que observa en la Figura 3.18.

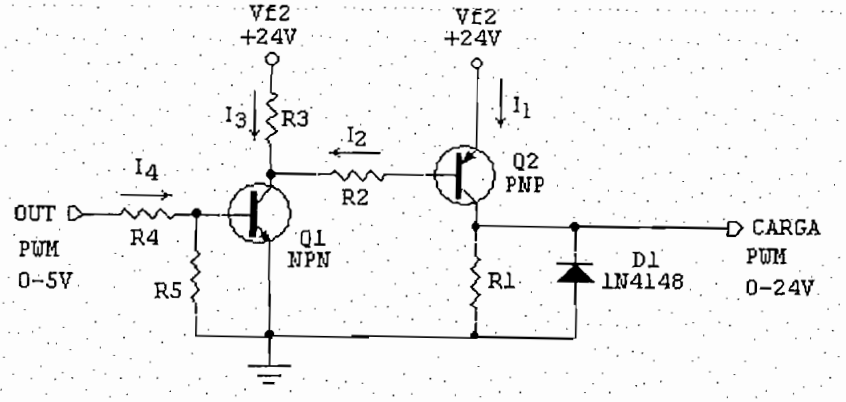


Figura 3.18. Tercer circuito acondicionador para el PWM.

Los cálculos para este circuito son los siguientes: Como el PWM está diseñado para entregar una corriente máxima de salida de 1A, que es la corriente de colector en el transistor Q2 y, por lo tanto, la corriente de base de dicho transistor que pasa por la resistencia R2 se calcula según la Ecuación 3.6:

$$I_2 = \frac{I_{carga}}{\beta_{Q2}} = \frac{1A}{1000} = 1mA \qquad \text{E.C. 3.6.}$$

La corriente que pasa por la resistencia R3 es I3. Cuando el transistor Q1 está conduciendo la corriente de colector es igual a la suma de la corriente I2+I3, pero cuando no existe carga la corriente I2 es despreciable, por lo que la corriente de colector del transistor Q1 es I3 y ésta debe tener por lo menos un valor mínimo para que el transistor Q1 esté activado; de aquí que se asume que debe ser de 1mA (ANEXO D). Cuando exista carga (siendo ésta máxima), la corriente de colector en el transistor Q1 es:

$$Icolector_{Q1} = I2 + I3 = 1mA + 1mA = 2mA \qquad \text{E.C. 3.7.}$$

Por lo que la corriente de base del transistor Q1 es:

$$I4 = \frac{Icolector_{Q1}}{\beta_{Q1}} = \frac{2mA}{150} = 13{,}33\mu 3 \qquad \text{E.C. 3.8.}$$

Con las corrientes calculadas se pueden hallar los valores de las resistencias, según las siguientes ecuaciones:

$$R_4 = \frac{V_{PWM} - V_{BEQ1}}{I_4} = \frac{5V - 0.7V}{13.33\mu 3} = 322{,}5k\Omega \qquad \text{E.C. 3.9.}$$

Se aproximó a un valor estandarizado de 330kΩ

$$R_3 = \frac{V_{EE}}{I_3} = \frac{24V}{1mA} = 24k\Omega \qquad \text{E.C. 3.10.}$$

$$R_2 = \frac{V_{EE} - V_{BEQ2}}{I_2} = \frac{24V - 1.4V}{1mA} = 22.6k\Omega \qquad \text{E.C. 3.11.}$$

A un valor estandarizado de 22kΩ

La resistencia R1 de 10K representa la carga mínima que está conectada en el colector del transistor de potencia Q2 y D1 es un diodo de libre circulación de corriente en caso de que se utilice cargas de tipo inductivas.

3.6 DISEÑO DE LA TARJETA DE VISUALIZACIÓN DE ENTRADAS O SALIDAS

Para saber el estado lógico de los terminales de entrada o salida se utilizaron diodos emisores de luz (LEDs). Estos pueden conectarse directamente al Microcontrolador, ya que los pines de los puertos pueden manejar corrientes alrededor de 50mA; sin embargo, para no sobrecargar al PIC y aislarlo se decidió utilizar seguidores de tensión para el puerto A y puerto E, ya que estos tendrán la posibilidad de conectarse al módulo de entradas analógicas. Para los demás puertos se utilizó únicamente un buffer inversor TTL, como el que se puede ver en la Figura 3.19.

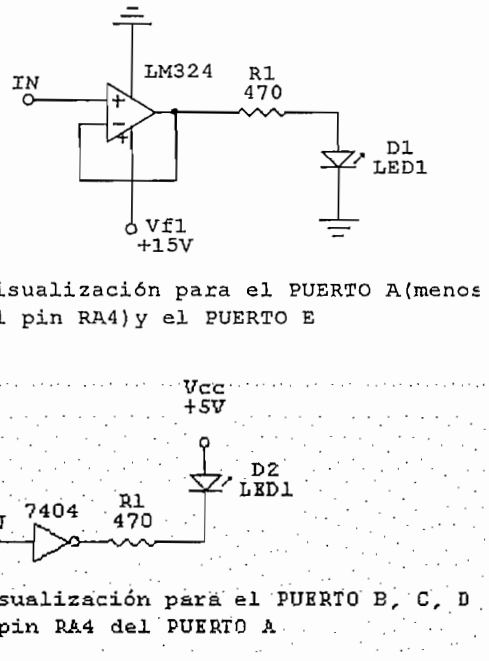


Figura 3.19. Circuitos para visualización del estado de las entradas o salidas.

El presente capítulo recoge el diseño de todo el hardware necesario, para el funcionamiento del PLC del proyecto de esta tesis. En resumen se tiene los siguientes circuitos:

- Circuito de la unidad central de proceso (CPU).
- Circuito de Entradas digitales.
- Circuito de Entradas analógicas.
- Circuito de Salidas digitales.
- Módulo especial: Comunicación serial RS-232 y salidas PWM.
- Circuito de Visualización de entradas o salidas.

Como ya se mencionó en el Capítulo 1, el PLC que se estructuró en esta Tesis sigue la tendencia modular, pero respetando la estructura del Microcontrolador, es decir que el usuario por ejemplo, deberá asignar todos los pines de un puerto como entradas o salidas, ya que en el hardware no se dispone de combinaciones de entradas y salidas en una misma tarjeta. Adicionado a esto, esta el hecho del uso de las Tarjetas Especiales: Salidas Analógicas y Módulo Especial, las cuales solo pueden usarse en los puertos A y C, respectivamente. Tomando en cuenta

estos aspectos, se aumentó al listado de Tarjetas, una adicional de entradas y otra de salidas con lo cual se puede manejar los puertos A y C, con estas y no necesariamente con las especiales. En síntesis son 8 Tarjetas, de las cuales 2 son fijas y las 6 restantes sirven para dar al PLC la tendencia modular de la que se hablado. En el ANEXO E se encuentra el diseño de los circuitos impresos de todas estas Tarjetas, mientras que en el ANEXO F están las fotografías del PLC y de las tarjetas que lo conforman.

CAPÍTULO 4

DISEÑO DEL SOFTWARE DE SOPORTE

4.1 PRÓLOGO

En este capítulo se explica el diseño del software para el manejo del PLC desarrollado en este proyecto, con este software el usuario podrá configurar y elaborar programas para aplicaciones industriales de distinta índole.

El lenguaje de programación FBD, que es el que se ha desarrollado para el presente proyecto, es un lenguaje de alto nivel; es decir, es un programa que tiene varias herramientas para que el usuario elabore más fácilmente un programa. Como ya se ha mencionado anteriormente, el lenguaje con el se programa a los microcontroladores PIC (elemento principal del CPU del PLC de este proyecto) es assembler, que es complejo pues se requiere combinar algunas instrucciones para desarrollar funciones básicas. Por otro lado se necesita configurar a varios de sus registros y en general tener un buen conocimiento de su arquitectura y funcionamiento, para conseguir que este se comporte como un PLC.

Para explicar de mejor manera lo antes dicho, en la Figura 4.1 se visualiza un ejemplo gráfico de estos conceptos.

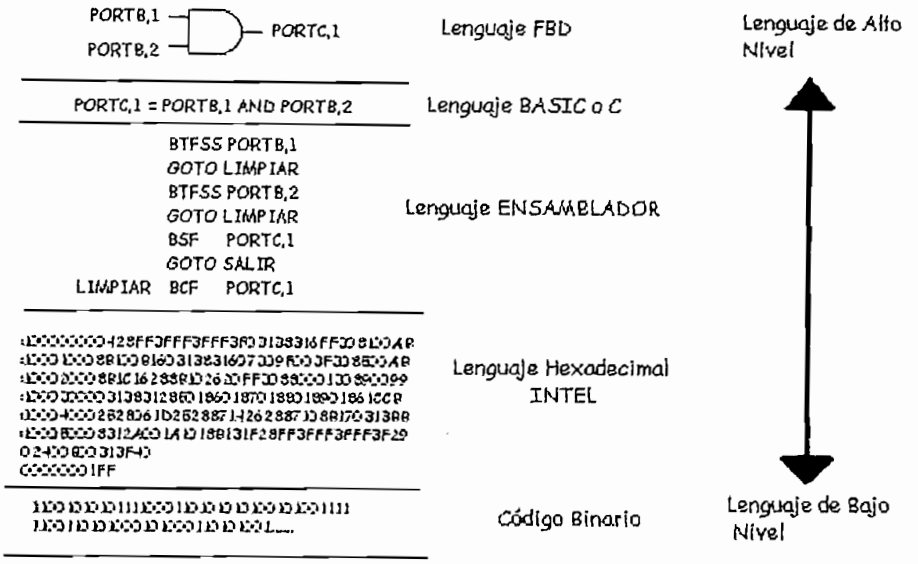


Figura 4.1. Niveles de lenguajes de programación.

El software que aquí se reporta fue concebido para que proporcione una interfaz gráfica que posibilite el desarrollo de un programa mediante bloques de funciones (FBD); un simulador, que es una gran herramienta para poder predecir el comportamiento del hardware y depurar los posibles errores; y una interfaz para descargar el programa realizado desde la PC hacia el PLC y que además tiene la opción de generar el código hexadecimal necesario para programar al PIC.

La explicación de este capítulo es extensa por lo cual se hace necesario aclarar, la lógica que seguirá el mismo. En primer lugar se tratará la parte que el usuario ve y maneja; en segundo lugar se describirá aquella que es invisible a este y que en realidad constituye el corazón del PLC. Entender esta última parte será un tanto difícil, en particular sin no se posee un conocimiento por lo menos básico de Visual Basic. Sin embargo, se procurará explicar acerca del desarrollo del software de la manera mas sencilla y concisa, esperando que se entienda como se realizaron todas las subrutinas que se expondrán mas adelante.

Partiendo de lo indicado, el software desarrollado posee dos partes:

- Interfaz Gráfica de usuario.
- Algoritmos para el microcontrolador PIC.

4.2 INTERFAZ GRÁFICA DE USUARIO

Como ya se dijo, el programa base para el diseño del software es Microsoft Visual Basic, que es una herramienta para el desarrollo de aplicaciones gráficas bajo Windows, y que se basa en lenguaje de programación Basic. La interfaz gráfica para elaborar los proyectos y que se muestra en la Figura 4.2 es un ejemplo de esto.

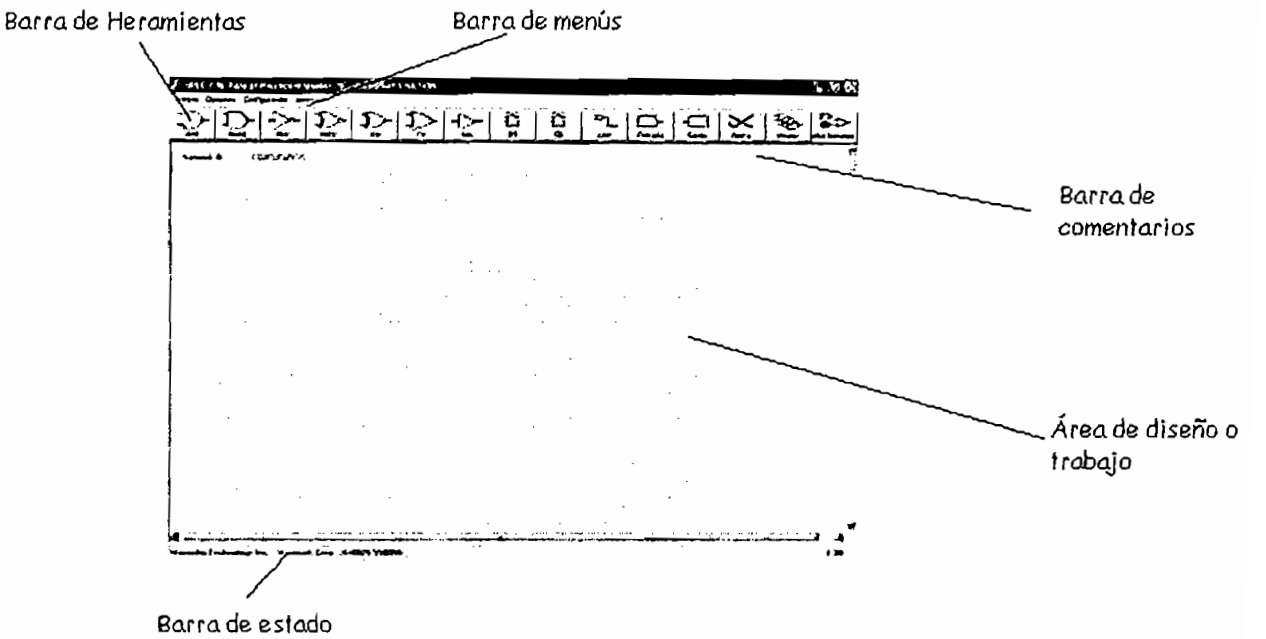


Figura 4.2. Pantalla de desarrollo de aplicaciones FBD.

La interfaz de usuario que se puede ver en la Figura 4.2 anterior es solo una de las tres interfaces que conforman todo el software, las dos restantes son la interfaz de simulación y la descarga, que se explicarán mas adelante.

Explicar como se desarrollo la interfaz gráfica sería extremadamente largo e innecesario, ya que la parte relevante del proyecto es el hecho de cómo se pasa del lenguaje FBD a la simulación y a la programación del Microcontrolador PIC. Por este motivo únicamente se explican las herramientas que posee cada una de las interfaces.

4.2.1 INTERFAZ GRÁFICA PARA ELABORACIÓN DE PROYECTOS

En esta HMI se encuentran las herramientas (Figura 4.2), cuya función es permitir la elaboración de proyectos de automatización de cualquier índole. Esta área de diseño o trabajo, está dividida en 16 partes (de 0 a 15) denominadas 'NetWork', donde se podrá ubicar las funciones a utilizar. La idea de las NetWorks, se la tomó del software para programar PLCs de la marca SIEMENS (Programa MicroWin). Esta interfaz la conforman:

- Una barra de estado.
- Una barra de comentarios.
- Una barra de herramientas.
- Una barra de menús.

4.2.1.1 Barra de Estado

Como se puede ver en la Figura 4.2, la barra de estado se encuentra ubicada en la parte inferior de esta ventana. En esta barra se puede visualizar, las coordenadas del puntero o mouse en la zona o área de trabajo, la acción que se va realizar, indicaciones generales y la hora del sistema.

4.2.1.2 Barra de Comentarios

Permite al usuario poner un comentario de texto, para cada NetWork. Esta opción es útil para hacer una programación mejor estructurada.

4.2.1.3 Barra de Herramientas

Está compuesta por todas las funciones y comandos que posee el PLC. En la Figura 4.3, se observa la primera parte de la barra de herramientas que está compuesta por las siguientes funciones:

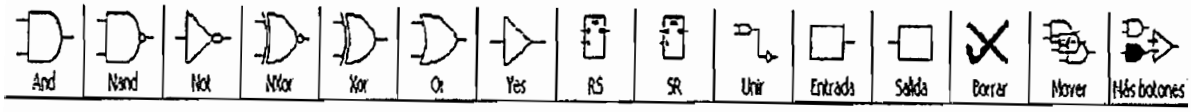
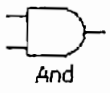


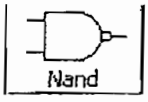
Figura 4.3 Barra de herramientas.

Botón AND



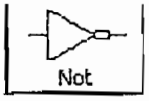
Esta función permite realizar un AND lógico entre dos bits de los puertos del microcontrolador o entre dos bits de algún registro de la memoria RAM.

Botón NAND



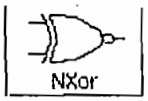
Esta función permite realizar un NAND lógico entre dos bits de los puertos del microcontrolador o entre dos bits de algún registro de la memoria RAM

Botón NOT.



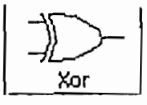
Esta función lee el estado lógico de un bit de un puerto del microcontrolador o un bit de algún registro de la memoria RAM, e invierte su estado para colocarlo en otro bit de un puerto o registro.

Botón XOR negado



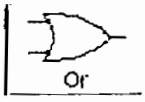
Esta función permite realizar un XOR negado lógico entre dos bits de los puertos del microcontrolador o entre dos bits de algún registro de la memoria RAM.

Botón XOR.



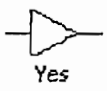
Esta función permite realizar un XOR lógico entre dos bits de los puertos del microcontrolador o entre dos bits de algún registro de la memoria RAM.

Botón OR.



Esta función permite realizar un OR lógico entre dos bits de los puertos del microcontrolador o entre dos bits de algún registro de la memoria RAM.

Botón YES



Esta función lee el estado lógico de un bit de un puerto del microcontrolador o un bit de algún registro de la memoria RAM, y lo coloca en otro bit de un puerto o registro.

Botón RS

Esta función es de tipo memoria RS, es decir que tiene un Terminal de entrada S (set) y otro R (reset).

Botón SR

Esta función es de tipo memoria SR, es decir que tiene un Terminal de entrada S (set) y otro R (reset).

Botón Unir.

Cuando se escoge este comando, el usuario tiene la posibilidad de unir los terminales de salida y/o entrada de las funciones como se indica en la Figura 4.4.

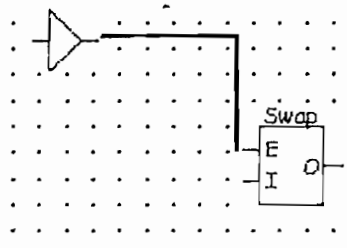


Figura 4.4. Ejemplo del comando "unir".

Las uniones entre funciones no pueden darse en todas circunstancias, esto quiere decir que el usuario no puede unir salidas entre salidas, entradas entre entradas, combinar funciones booleanas entre funciones a nivel de Bytes, etc. Más adelante se explicará detalladamente este tema.

Botón bit de entrada

Esta función sirve para configurar como entrada cualquier bit del microcontrolador o cualquier bit de los registros de la memoria RAM.

Botón bit de salida.

Esta función sirve para configurar como salida cualquier bit del microcontrolador o cualquier bit de los registros de la memoria RAM.

Cuando se efectúa el evento clic en cualquiera de los dos botones anteriores, se despliega un cuadro de diálogo, para escoger un bit de cualquiera de los puertos del PIC, como se puede ver en la Figura 4.5 siguiente:

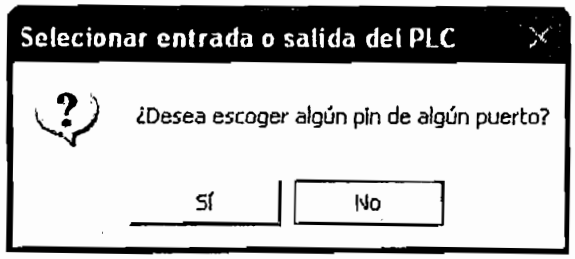


Figura 4.5. Caja de diálogo para confirmar que se va utilizar un bit de un puerto.

En caso de una respuesta afirmativa, se despliega un cuadro de diálogo con el que se puede escoger que bit se desea utilizar como entrada o salida. Al dar un clic en la viñeta de la caja que dice Pines se despliega una lista con todos los bits de todos los puertos que posee el PIC.

Figura 4.6. Caja de diálogo para escoger un bit de un puerto.

Cuando se presiona el botón de cerrar  se cancela la acción de graficar y cargar la función 'bit de entrada' o 'bit de salida' y se indica al usuario dicha cancelación con un cuadro de diálogo.

Al presionar el botón "No" del cuadro de dialogo indicado en la Figura 4.5, el usuario tendrá la opción de escoger un bit de un registro de la memoria RAM, de la ventana que se despliega e indica en la Figura 4.7. Nótese que se debe seleccionar también el banco del registro a utilizar, se puede escoger de entre los cuatro bancos que conforman la memoria RAM y que se explicó en la Sección 2.1.4.

Los registros para el 'Banco 0' van desde la posición 32 decimal hasta 127 decimal, pero el usuario solo tiene acceso desde la 34 decimal hasta 125 decimal.

Los registros para el 'Banco 1' van desde la posición 160 decimal hasta la 255 decimal, pero el usuario solamente dispone desde la 160 hasta la 254.

Los registros para el 'Banco 2' van desde la posición 272 decimal hasta 383 decimal, pero el usuario solamente dispone desde la 272 hasta la posición 366.

Los registros para el 'Banco 3' van desde la posición 400 decimal hasta la posición 511 decimal, pero el usuario dispone desde la posición 400 hasta la 494.

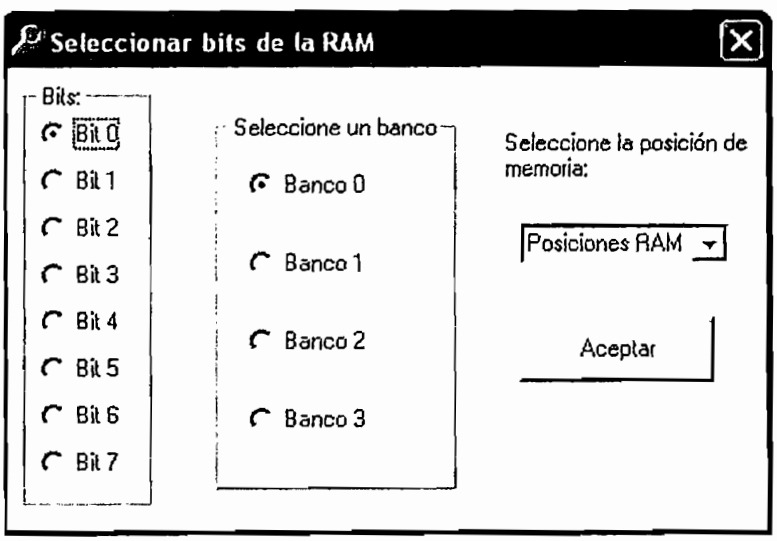


Figura 4.7. Cuadro de diálogo para escoger una posición de la RAM.

Existe un motivo para reservar algunas posiciones de la memoria RAM; esto es que sirven como registros auxiliares dentro de las operaciones que permiten

obtener un PLC con el microcontrolador PIC. Cuando en esta ventana no se presiona el botón aceptar y se presiona el botón cerrar, la acción se interrumpe y se informa al usuario mediante un cuadro de diálogo que indica que no se ha escogido ningún bit de la memoria RAM.

Una vez que se haya escogido un bit como entrada o salida de un puerto o registro de la memoria RAM, la imagen tendrá su respectiva identificación como se puede ver en la zona de trabajo mostrada en la Figura 4.8.

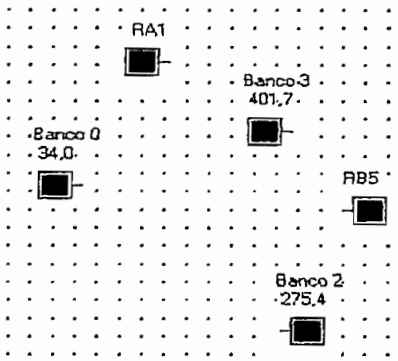


Figura 4.8. Asignación de nombres tanto para salidas y como entradas.

Nota: Cuando se asigna a un bit de un puerto como entrada, ya no se puede seleccionar el mismo bit del mismo puerto como salida, y viceversa, una acción incorrecta como esta se informa al usuario mediante un cuadro de diálogo. Sin embargo, los bits de los registros de la memoria RAM si pueden funcionar como entrada y salida al mismo tiempo.

Los eventos que se desencadenan en la barra de herramientas del formulario de diseño y que servirán para manipular las diferentes funciones o figuras son:

Botón borrar.

Con este botón el usuario puede borrar una función, o una línea conector, simplemente después de haber escogido este comando se da un clic sobre la función o línea y se pregunta al usuario si realmente desea eliminar la función o línea, como se indica en la Figura 4.9.

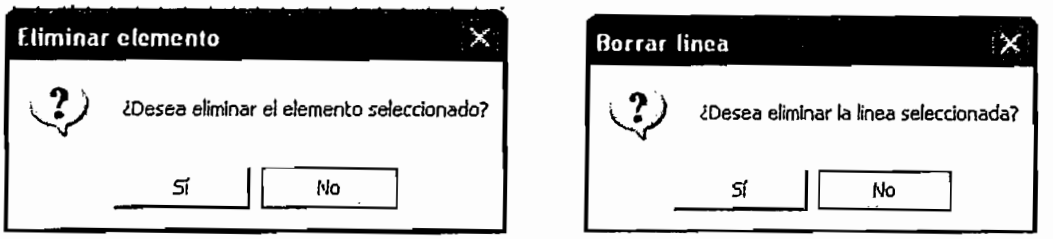


Figura 4.9. Cuadros de diálogo para borrar elementos y líneas.

Botón Mover

Con este comando se puede mover una función a otro lugar, siempre y cuando no tenga ninguna línea conector conectada a sus terminales. El motivo por el cual no se puede hacer esto es que es difícil desarrollar un algoritmo que permita mover las líneas que ya están asociadas a la función. Una vez elegido este comando y cumplida la condición anterior, se mueve la función a una nueva posición dando un clic con el botón izquierdo sobre el objeto y arrastrando el puntero del ratón hasta la nueva posición, para finalmente soltar el botón izquierdo del ratón cuando se a llegado a la posición deseada. Algo muy importante que se debe tener en cuenta es que una función se puede mover solamente, dentro del NetWork en que se encuentra.

Botón 'Mas botones'

Este botón oculta a todos los botones anteriormente explicados, y carga nuevos botones para nuevas funciones. La barra de herramientas queda como se indica en la Figura 4.10.

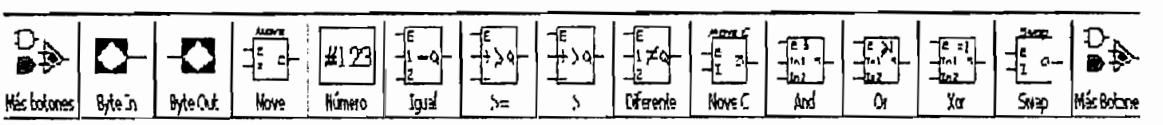
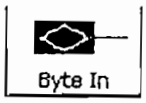


Figura 4.10. Segunda parte de la barra de herramientas.

El primer botón a la izquierda vuelve a colocar en la barra de herramientas en los botones ya explicados, mientras que los nuevos botones indicados en la Figura anterior vuelven a ser invisibles. Los botones que conforman la segunda parte de la barra de herramientas, son los siguientes:

Botón Byte In



Con esta función se puede seleccionar todo un puerto o registro de la memoria RAM para ser leído por las funciones a nivel de Bytes. Cuando se escoge esta función y se da un clic con el botón izquierdo sobre la zona de trabajo, se despliega un cuadro de diálogo como se muestra en la Figura 4.11.

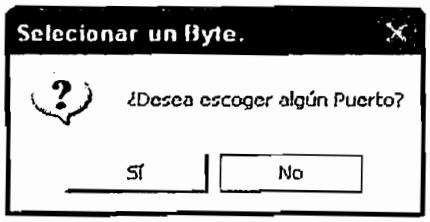


Figura 4.11. Cuadro de diálogo para escoger o no un puerto del microcontrolador.

En caso de presionar 'Sí', se despliega una ventana como la que se puede ver en la Figura 4.12. En esta ventana, el usuario puede escoger que puerto desea configurar y que bits del mismo desea que sean entradas. En caso de presionar 'Cancelar' en esta venta, se presenta el cuadro de información en que se resalta que no se ha escogido ningún puerto del Microcontrolador.

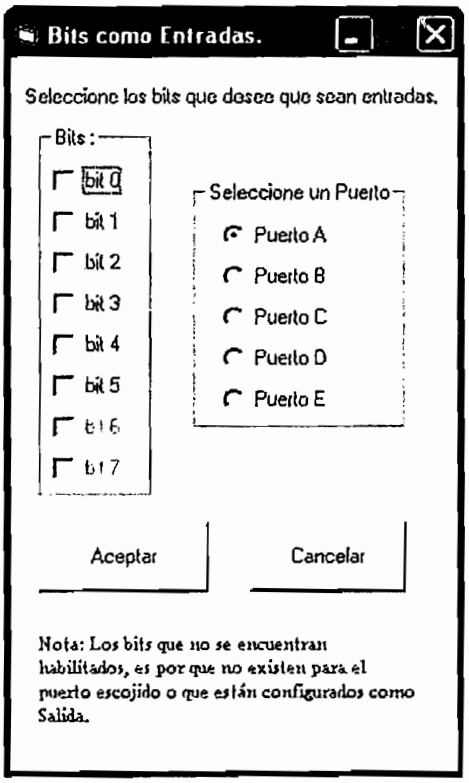


Figura 4.12. Cuadro de diálogo para escoger un puerto del PIC.

En caso de que se haya presionado 'No' en la ventana de la Figura 4.11, se asume que el usuario desea escoger un registro de la memoria RAM o un registro de la memoria EEPROM. La caja de diálogo que se despliega es la que se observa en la Figura 4.13. En esta ventana se escoge los bytes de la memoria RAM y EEPROM, (los cuales no necesitan ser configurados como entradas o salidas se los puede leer o escribir simultáneamente). En esta ventana el usuario deberá seleccionar: un registro de la RAM del banco que desee o uno de los 256 registros de la memoria EEPROM (donde se almacenan datos, que no se quieren perder al apagar el equipo).

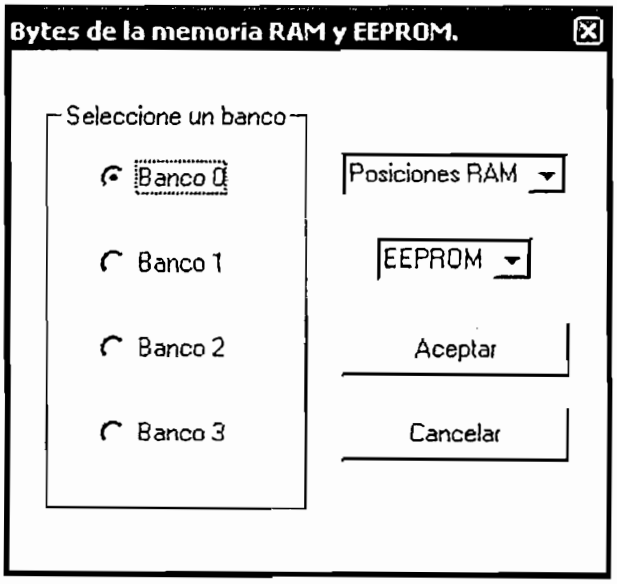


Figura 4.13. Cuadro de diálogo para escoger el registro de la RAM o de la EEPROM.

Una vez escogido el registro o puerto, este tendrá su debida identificación como la que se muestra en la Figura 4.14. A los puertos se los nombra directamente, mientras que los registros de la memoria RAM poseen el número de posición al cual pertenecen de acuerdo con el banco seleccionado; y los registros de la memoria EEPROM poseen la posición del registro, precedidos de las iniciales 'EE'.

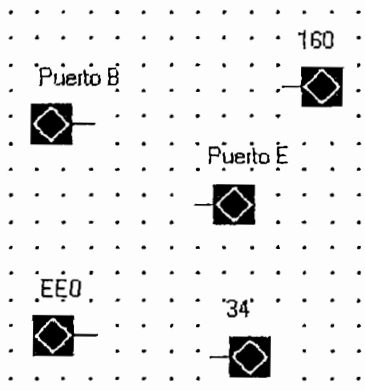
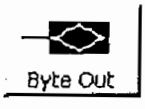


Figura 4.14. Asignación como entradas o salidas para puertos o bytes del PIC.

Botón Byte Out



Este botón es similar al botón 'Byte In'. Sirve para poder escribir directamente en un puerto un byte (palabra de 8 bits) o en un registro de la memoria RAM o PROM.

Previamente a la explicación de los siguientes botones es necesario aclarar algunos conceptos que permitirán tener un mejor entendimiento de las funciones que estos realizan.

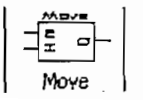
'E': es el Terminal del habilitador, en este Terminal se necesita tener el valor lógico '1', para que la función realice la operación a la que está destinada. En este se puede conectar cualquier bit de un puerto que esté configurado como entrada, un bit de un registro de la RAM o también el resultado de cualquier operación booleana.

'Byte': es un puerto, un registro cualquiera de la memoria RAM que el usuario tiene acceso o un número de 0 – 255. En esta explicación se excluye a cualquier registro de la Memoria EEPROM.

'Q': es salida a un bit de un byte

'S': es salida a un byte.

Botón 'Move'



Esta función permite transferir el contenido de un byte a otro. Posee un 'Enable', para habilitar la transferencia de información de un byte que se

encuentra conectado en el Terminal 'In' hacia un byte conectado en el Terminal 'S'.

En la Figura 4.15 se presenta un ejemplo del uso de esta función. El Terminal RA0 configurado como entrada controla el Enable de la función, la información se va a transferir del Puerto B al registro 35 de la memoria RAM (el registro 35 decimal se encuentra en el banco cero).

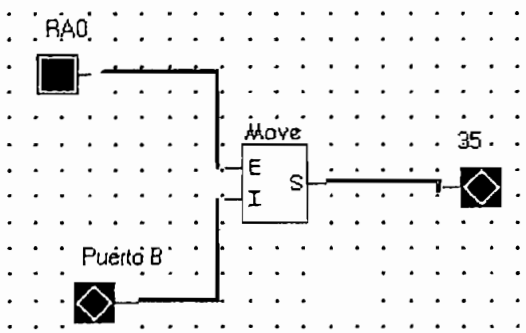
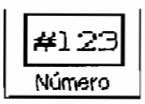


Figura 4.15. Ejemplo de conexión para la función MOVE de un puerto a una posición de la RAM.

Botón 'Número'



Esta función se puede utilizar conjuntamente con la función 'Move' descrita anteriormente y con funciones que se explicarán más adelante. En la Figura siguiente se muestra un ejemplo con la función 'Move', en la que se trasfiere el número 7 hacia el puerto C cuando RA0 esté en 1 lógico; es decir que el Puerto tendrá 00000111.

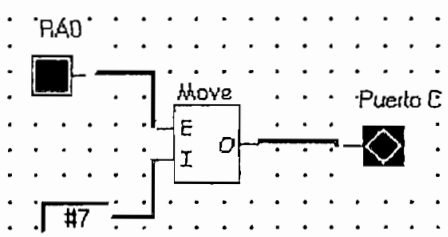


Figura 4.16. Ejemplo de conexión para un Número, con el uso de la función Move.

Cuando se selecciona la función 'Número' aparece una caja de diálogo que pide ingresar un número entre 0 a 255, como ser puede ver en la Figura 4.17.

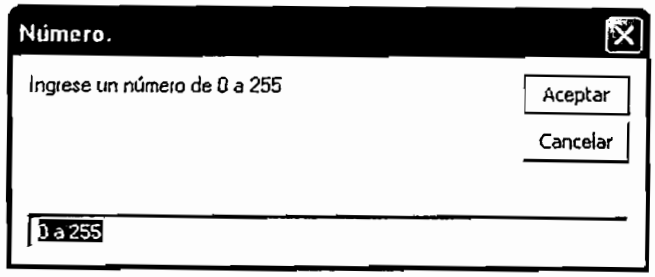
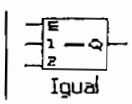


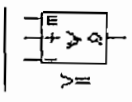
Figura 4.17. Cuadro de diálogo para escoger un número.

Botón 'Igual'



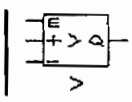
Esta función permite comparar si dos bytes son iguales. La función tiene un habilitador E, las entradas a bytes '1' y '2' y la salida 'Q'. Lo que hace esta función es comprobar que los bytes comparados sean exactamente iguales bit a bit. Si ese es el caso el Terminal de salida 'Q' se pone a 1 lógico, caso contrario, se pondrá en 0 lógico.

Botón 'Mayor igual'



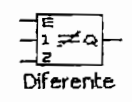
Esta función compara cual de dos bytes es mayor o igual. Tiene un habilitador 'E', dos entradas a Bytes '+' y '-' y una salida 'Q'. Lo que la función hace es comprobar que el Byte conectado en el Terminal de entrada '+' sea mayor o igual que el byte conectado en el Terminal de entrada '-', el Terminal de salida 'Q' se pone a 1 lógico en caso de que esta condición sea cierta; caso contrario será 0 lógico.

Botón 'Mayor'



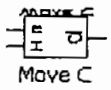
Esta función permite comprobar que el Byte conectado en el Terminal de entrada '+' sea mayor que el byte conectado en el Terminal de entrada '-'. El Terminal de salida 'Q' se pone a 1 lógico en caso de que esta condición sea cierta; caso contrario, se pondrá en 0 lógico. También posee un Terminal de entrada 'E' que es el enable o habilitador de la función.

Botón 'Diferente'



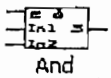
Esta operación es de similares características que la función 'Igual' ya que compara dos Bytes bit a bit, en caso de que sean diferentes, aunque sea en un solo bit, la salida Q da 1 lógico.

Botón 'Move C'



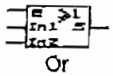
Esta operación se parece a la función 'Move', pero se diferencia en que se mueve el valor de un Byte a otro, pero complementado.

Botón 'AND'



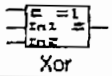
Esta operación permite realizar un AND lógico bit a bit entre dos Bytes. El resultado se lo envía a otro Byte en el Terminal de salida 'S'. Esta función tiene un habilitador en el Terminal 'E' y dos entradas 'In1' e 'In2'.

Botón 'OR'



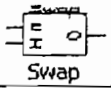
Esta función permite realizar un OR lógico bit a bit entre dos Bytes. Esta función tiene un habilitador en el Terminal 'E'. El resultado de la operación entre las dos entradas 'In1' e 'In2' se lo envía a otro Byte en el Terminal de salida 'S'.

Botón 'XOR'



Esta función permite realizar un XOR lógico bit a bit entre dos Bytes, la operación entre sus dos entradas son 'In1' e 'In2' se realiza si el bit conectado en el habilitador es 1L. El resultado es enviado a otro Byte conectado en el Terminal de salida 'S'.

Botón 'Swap'



Esta función se parece a la operación 'Move', con la diferencia de que durante la operación el contenido del Byte conectado en la entrada 'I' intercambia su parte alta con su parte baja. El resultado es enviado al byte conectado en el Terminal 'S' de salida.

Botón 'Mas Botones'

Este comando es similar al botón 'Más botones' anterior, oculta los botones anteriormente explicados y visualiza más botones, como se indica en la Figura 4.18.

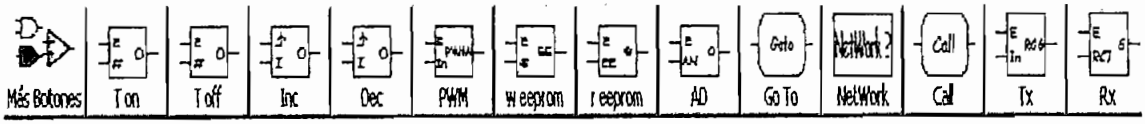
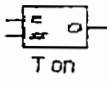


Figura 4.18. Tercera parte de la Barra de herramientas que se despliega con el botón Más Botones.

El primer botón a la izquierda vuelve a colocar en la barra de herramientas los botones de la segunda parte ya explicados, mientras que los nuevos indicados en la Figura 4.18 los vuelve invisibles.

Botón Ton.



Esta función permite colocar un temporizado tipo ON (con la opción de utilizar un número máximo de 8 temporizadores de este tipo). En esta función cuando el Terminal de entrada 'E' está en 1 lógico el tiempo empieza a correr internamente en un temporizador del simulador, este es un timer interno de la computadora. Cuando el tiempo preestablecido en el Terminal de entrada '#' se completa, el bit conectado en el Terminal 'O' de salida pasa a 1 lógico. Si por alguna razón el Terminal 'E' pasa a 0 lógico antes de haber alcanzado el tiempo preestablecido, el contador interno se detiene pero no se resetea; y cuando 'E' vuelve a 1 lógico se renueva el conteo hasta llegar al tiempo establecido. Para volver a poner en 0 lógico el Terminal de salida, el Terminal 'E' debe también volver a 0 lógico. El Terminal de entrada '#' solamente admite la 'Función Número', no se puede conectar aquí ningún registro de la RAM o puerto. Cuando en el Terminal '#' se encuentra conectado la 'Función Número' con valor 1, el tiempo de este temporizador, de 16bits y que está configurado con un prescaler de 2, es de aproximadamente 0.131070s tanto para el temporizador Ton, como para el Toff; mientras que, cuando la 'Función Número' está con el valor máximo de 255, se tiene un tiempo aproximado de 33.6s, el cálculo de estos valores se explicará con mas detalle en el diagrama de flujo de los temporizadores. El hecho de que este temporizador (y el de tipo OFF) no sea tan flexible, es por que el microcontrolador PIC utilizado solamente posee 3 temporizadores internos, de los cuales el T0 se lo utiliza conjuntamente para el 'perro guardián' (watch dog timer) y el T2 se utiliza en las funciones PWM, quedando únicamente el Timer 1 que es

de 16 bits, por lo cual fue algo difícil tratar de conseguir más temporizadores con un solo timer. Por este motivo es que el manejo de 8 temporizadores ON y 8 temporizadores OFF en realidad es solamente un artificio que se tuvo que hacer para que el PLC desarrollado en este proyecto tenga estos recursos tan importantes como son los temporizadores. En la Figura 4.19 se presenta el diagrama de tiempo del comportamiento del temporizador tipo ON.

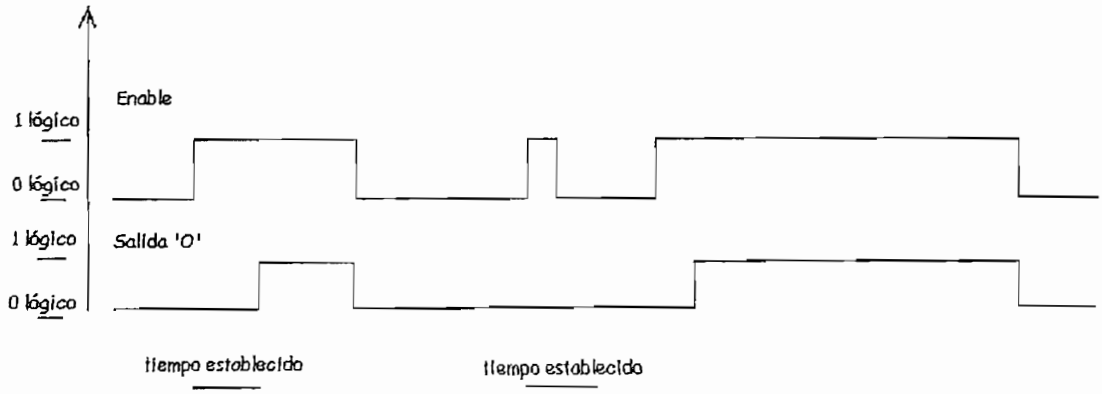
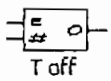


Figura 4.19. Diagrama de tiempo para el temporizador tipo ON

Botón Toff.



Este botón permite ingresar un temporizador tipo OFF. El número máximo de estos temporizadores es 8, y las propiedades que posee son similares al temporizador tipo ON (entradas, duración de tiempos y salidas). El comportamiento de este temporizador es el que se indica el diagrama de la Figura 4.20.

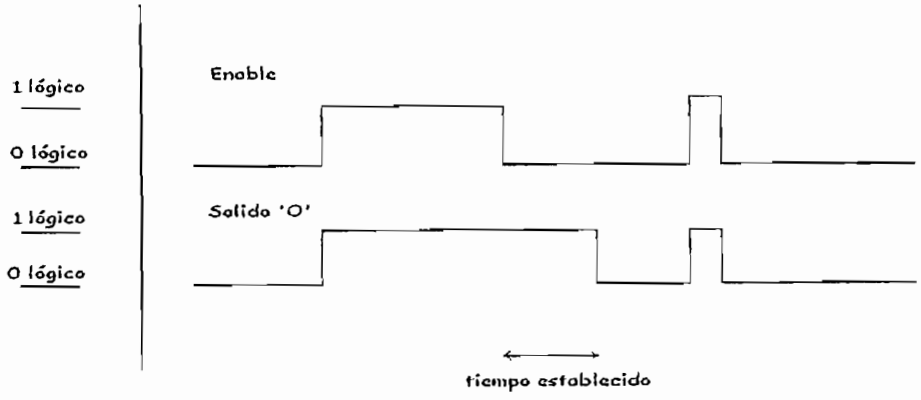
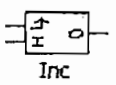


Figura 4.20. Comportamiento del Temporizador Toff.

Botón 'Inc'



Este botón permite ingresar una función de tipo 'Incremento'. Su funcionamiento es algo similar al la función 'Move' y se diferencia en que su habilitador (representado con una flecha apuntando hacia arriba) se activa con un flanco de subida (transición de 0 lógico a 1 lógico). Cuando sucede esta acción, se recoge el valor del byte en el Terminal 'I', y lo transfiere al byte conectado el Terminal de salida 'O', pero incrementado 1. La razón de que esta función se habilite mediante un flanco positivo, y no por estado lógico, es porque de esta manera se puede tener control del incremento de la variable. Para entender mejor esto, se tiene el ejemplo de la Figura 4.21.

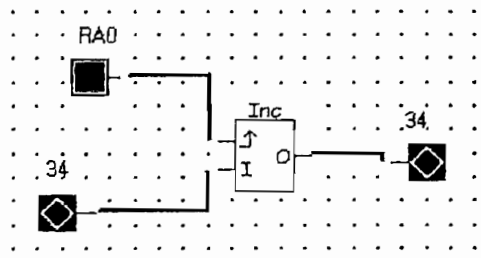
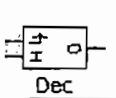


Figura 4.21. Ejemplo de conexión para la función Inc.

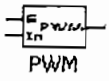
En este ejemplo se desea controlar el incremento del registro de la memoria RAM que se encuentra en la posición 34 (34 está en el banco 0), mediante el cambio del bit RA0. Si la operación de la función 'Inc' se debiera al estado lógico de bit RA0, la variable se incrementaría a si misma sin ningún control y la tasa de crecimiento del valor de este registro dependería del tamaño del programa y de la velocidad del microcontrolador. En cambio, con el control por flanco, el incremento del registro es cada vez que RA0 cambia de estado, garantizando al usuario una función mucho más eficiente.

Botón 'Dec'



Esta función es similar a la anterior. Se diferencia en que transfiere el valor del byte conectado al Terminal 'I' al byte conectado al Terminal 'O' pero disminuido en 1. Tiene el mismo funcionamiento que el 'Botón Inc'; es decir, el cambio de estado en el Terminal que tiene una flecha apuntando hacia arriba representa el decremento por cambio de 0L a 1L.

Botón 'PWM'



Esta función permite graficar hasta 2 funciones PWM que es característica propia del microcontrolador PIC utilizado. La representación de estas funciones se representa en la Figura 4.22.

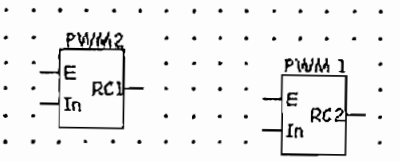
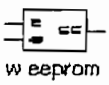


Figura 4.22. Representación gráfica de las Funciones PWM

La señal PWM1 y PWM2 solamente pueden conectarse en el Terminal RC2 y RC1 respectivamente, los cuales deberán ser configurados como terminales de salida.

El periodo de las dos señales PWM se explicará más adelante en una de las herramientas de la 'Barra de Menús'.

Botón 'w eeprom'



Esta función permite escribir el contenido de un byte en un registro de la memoria EEPROM del microcontrolador. Cuando su Terminal habilitador 'E' tiene el estado lógico 1, recoge el contenido del byte conectado en el Terminal de entrada '@' y lo escribe en la posición de la memoria EEPROM que se encuentra en el Terminal de salida 'EE'. En la Figura 4.23 se muestra un ejemplo de escritura de una posición de la memoria EEPROM. Como se puede ver, el contenido del puerto B se transfiere a la posición 254 de la memoria EEPROM, el control de dicha escritura se realiza con el Terminal RC7 del puerto C.

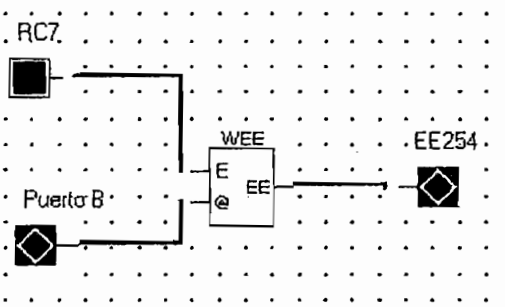
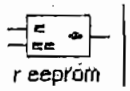


Figura 4.23. Ejemplo de la transferencia de un dato que no se desea perder del puerto B con el condicionante de un bit del puerto C.

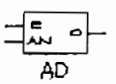
La memoria EEPROM tiene un número máximo aproximado de un millón de veces de escritura/borrado, según Microchip®; es decir, si en el ejemplo de la Figura 4.23, el Terminal habilitador 'E' se mantiene constantemente en 1 lógico, la posición de la memoria EEPROM 254 estaría constantemente escribiéndose en ella y, por consiguiente, se acortaría la vida útil de dicha posición de memoria. Para evitar este inconveniente esta función opera de la siguiente manera: Primeramente, cuando el Terminal 'E' está en 1 lógico la función lee el contenido de la posición conectada en el Terminal de salida 'EE' y lo compara con el contenido del byte en el Terminal '@'. Si estos valores son iguales no se escribe el contenido de la posición indicada en el Terminal 'EE', protegiéndose así la vida útil de las posiciones de la memoria EEPROM. El tiempo de escritura que tiene la memoria EEPROM es de aproximadamente 4 ms.

Botón 'r eeprom'



Esta función permite leer un registro de la memoria EEPROM y almacenar su contenido en un byte. Aquí no es necesario tener ninguna precaución con la memoria EEPROM.

Botón 'AD'



Esta función permite leer el contenido de uno de los canales del conversor AD (Puerto A o Puerto E). Previamente se debe configurar al conversor AD y el número de canales a utilizar, esto se explicará más adelante en el funcionamiento de la 'Barra de Menús'.

Botón 'Go To'

Este comando permite saltar hacia otro NetWork dentro del programa principal. Se utiliza conjuntamente con una 'etiqueta de subrutina' que se explicará más adelante. Es importante primeramente, explicar como debe estructurarse un programa para que el PLC desarrolle una aplicación.

Como es de conocimiento general, los programas comprenden dos partes básicas: El programa principal y las subrutinas (opcional). El programa principal siempre residirá en el NetWork 0 hasta aquel que el usuario defina (máximo el

NetWork 15). Los NetWorks restantes se podrán utilizar para subrutinas que se pueden llamar desde el programa principal o desde eventos especiales (en el siguiente orden: la interrupción externa en el Terminal RB0 primero y luego la interrupción por recepción en la comunicación serial RS 232). En la Figura 4.24 se muestra la estructura básica de un programa.

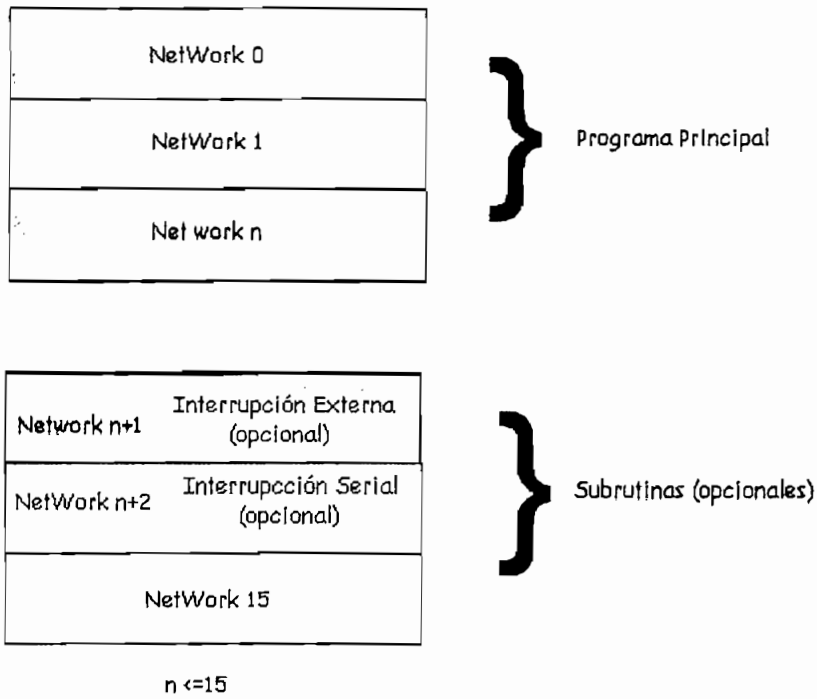


Figura 4.24. Estructura para diferenciar la capacidad del programa principal de las subrutinas

En la Figura 4.25 se presenta un ejemplo del comando 'Go To', en este se realiza un salto a NetWork 5, si y solo si, RE1 esta en 1 lógico. Lógicamente dentro del programa principal deberá estar el NetWork 5. La configuración de la estructura del programa principal, se explicará con mas detalle en el funcionamiento de la 'Barra de Menús'.

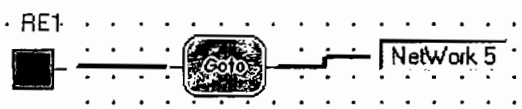
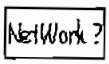


Figura 4.25. Conexión para la función Go To.

El programa está diseñado para analizar y evitar situaciones contradictorias, advirtiendo al usuario que se ha cometido un error cuando hay un salto al mismo NetWork donde está la función 'Go to', o cuando haya un salto a un NetWork que no se encuentre dentro del programa principal que ha definido el usuario.

Botón 'NetWork'



Esta etiqueta 'NetWork' que se explicó anteriormente se utiliza con el comando 'Go To' y con el comando 'Call' que se explicará más adelante. Cuando el usuario escoge este comando se muestra la siguiente ventana de diálogo (Figura 4.26) que le permite asignar a que NetWork pertenecerá la etiqueta.

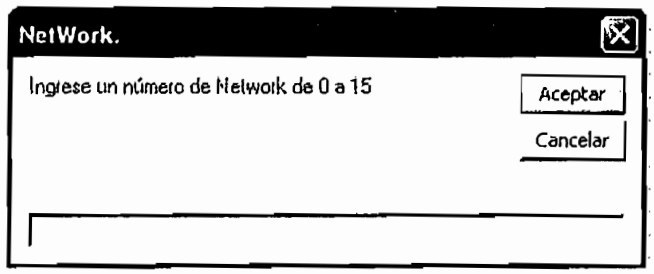
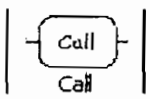


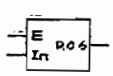
Figura 4.26. Cuadro de diálogo para determinar a que NetWork se necesita ir.

Botón 'Call'



Este comando es muy similar al comando 'Go To', y permite llamar a una NetWork utilizada como subrutina. Una vez que se ejecutan todas las funciones en el NetWork subrutina, el flujo del programa vuelve al lugar desde donde se llamó a la subrutina. Se puede llamar dentro de una subrutina a otra, con un número máximo de 8 niveles de profundidad (Pila del microcontrolador).

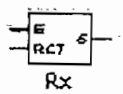
Botón 'Tx'



Esta función permite transmitir el contenido de un byte a través del puerto serial del microcontrolador, cuando su habilitador 'E' recibe un flanco positivo (de 0 lógico a 1 lógico) evitando de esta manera que se transmita un valor al puerto serial, durante todo el tiempo que 'E' permanece en 1 lógico, solamente el Terminal RC6 del Puerto C, podrá ser utilizado como salida Tx. La

configuración de la comunicación serial se explicará más adelante en el funcionamiento de la 'Barra de Menús'.

Botón 'Rx'



Esta función permite leer el contenido del buffer para la recepción en la comunicación serial RS232 cuando su Terminal habilitador 'E' esté en 1 lógico. El contenido del buffer en RC7 es transmitido a un Puerto o a un registro de la memoria RAM. Esta función se deberá utilizar en el NetWork subrutina, asociado con la interrupción por recepción en la comunicación serial.

4.2.1.4 Barra de Menús

La barra de menús, como en todos los programas, es una herramienta indispensable para el manejo de los recursos que posee dicho programa. En la Figura 4.27 se muestra las opciones que el software desarrollado en este proyecto presenta.

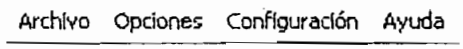


Figura 4.27. Barra de Menús.

Categoría Archivo

Al elegir la opción Archivo se despliega la lista indicada en la Figura 4.28.

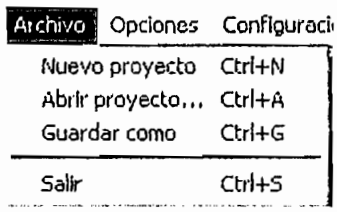


Figura 4.28. Despliegue de la herramienta Archivo en la Barra de Menús.

'Nuevo Archivo' (ctrl.+N)

Si se da un clic en esta opción se pregunta al usuario si desea guardar el proyecto actual.

'Abrir Proyecto' (ctrl.+A)

Si el usuario escoge esta opción puede abrir un proyecto previamente realizado en el programa para el PLC. Los proyectos tienen la extensión .tad y despliegan

una caja de diálogo típica de los programas basados en Windows como se indica en la Figura 4.29.

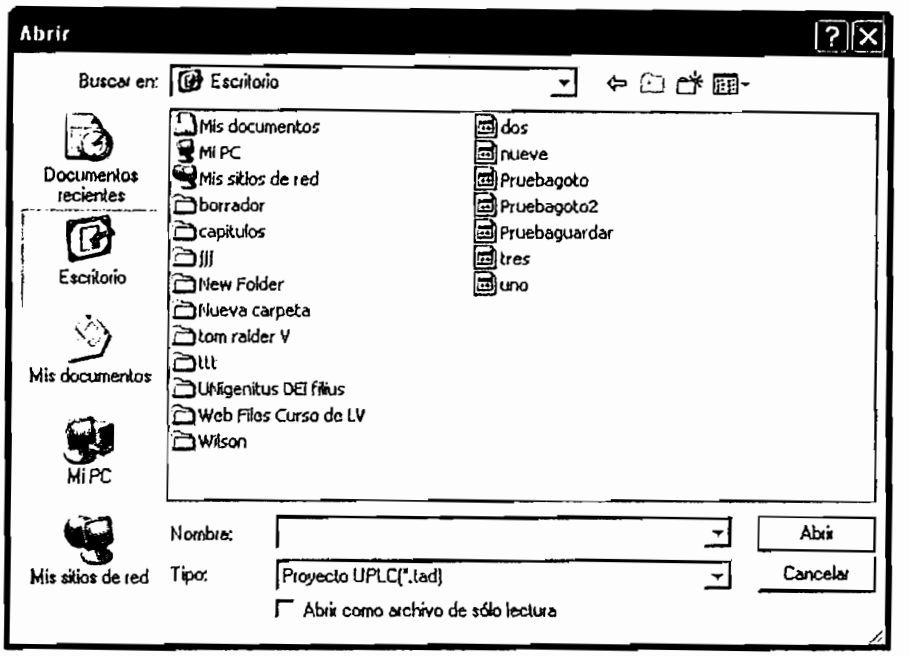


Figura 4.29. Cuadro de diálogo para guardar un proyecto.

'Guardar Proyecto' (ctrl.+G)

Si el usuario decide seleccionar esta opción se despliega una caja de diálogo similar a la anterior, como se puede ver en la Figura 4.30.

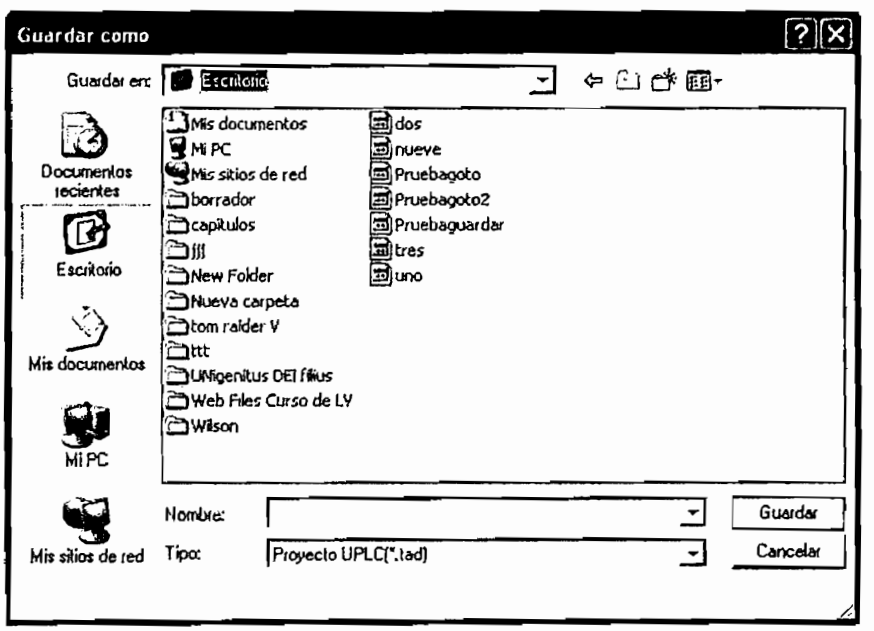


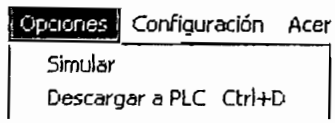
Figura 4.30. Cuadro de diálogo para guardar un Proyecto.

'Salir' (ctrl.+S)

Con esta opción se termina la aplicación.

Categoría Opciones

En la ficha 'Opciones' se despliegan los comandos de la lista mostrada en la Figura 4.31.



4.31. Herramienta Opciones.

'Simular'

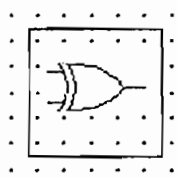
Con esta herramienta el usuario puede simular el comportamiento del PLC, esto se explicará más adelante.

'Descargar a PLC' (ctrl.+D)

Al seleccionar el usuario esta opción puede descargar al PLC el proyecto realizado.

Tanto en la opción 'Simular' y 'Descargar a PLC', el programa realiza un análisis del proyecto para determinar posibles errores que serán reportados al usuario, para que los corrija.

A continuación se muestra un ejemplo de uno de los posibles errores que puede cometer un usuario al realizar un proyecto. Cuando una función no tiene conectado alguno de sus terminales, el programa advierte en que NetWork se encuentra dicha función y además la marca con un cuadro de color rojo, como se puede ver en la Figura 4.32. Durante este proceso la simulación o descarga al PLC se interrumpen.



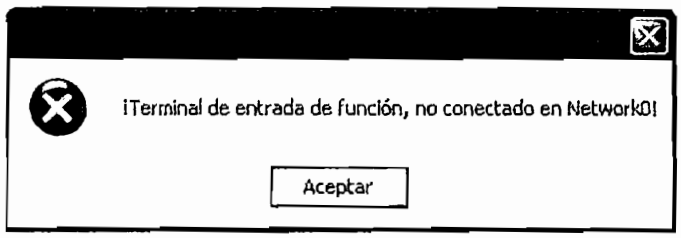


Figura 4.32. Cuadro de advertencia de error en un NetWork de un proyecto y señalización de la función en conflicto.

En los dos comandos de la herramienta Opciones se pregunta al usuario hasta donde desea estructurar el programa principal, como se puede ver en la Figura 4.33.

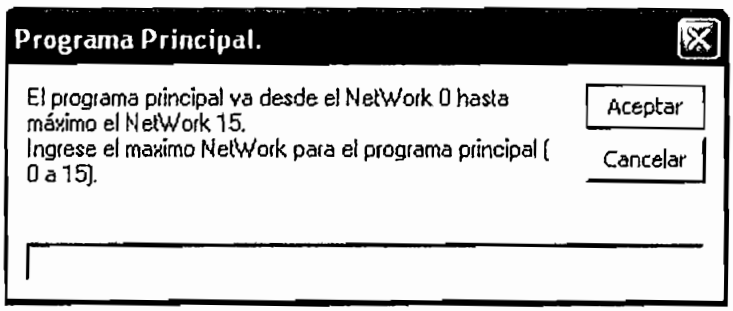


Figura 4.33. Caja de diálogo para definir el alcance que tendrá el programa principal.

Una vez indicado el tamaño del programa principal, se despliega un cuadro de resumen de la configuración de las funciones especiales que posee el PLC como: la comunicación serial RS232, interrupción externa, AD, etc., (véase la Figura 4.34). Estas características y configuraciones se pueden alterar únicamente en la opción 'Configuración' de la barra de herramientas que se explicará más adelante.

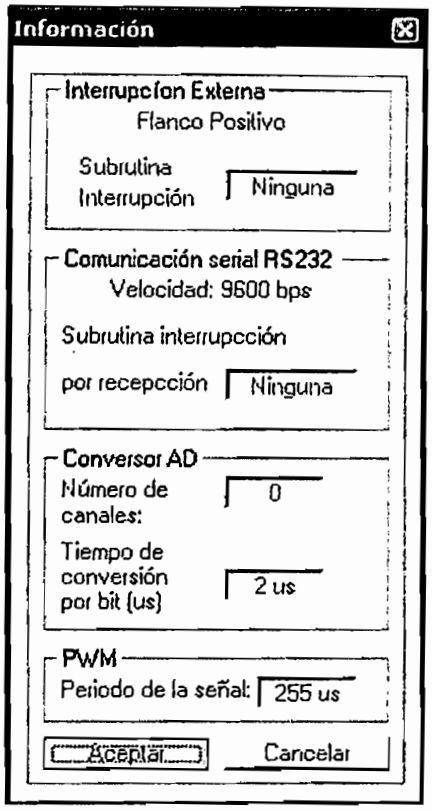


Figura 4.34. Cuadro de Resumen de las funciones especiales del PLC, previo a la simulación y descarga.

Categoría Configuración

En esta opción se despliega la siguiente lista de comandos.

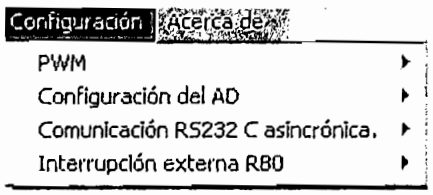


Figura 4.35. Lista de comando de la herramienta configuración.

'PWM'

En esta opción existe un sub lista con las opciones para configurar el período que se necesita que tenga el PWM, las opciones para configurar el periodo son 255μs, 1.02ms y 4.08ms. El valor que se encuentra señalado por defecto es el primero como se puede ver en la Figura 4.36.

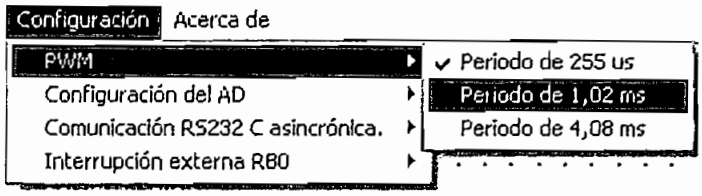


Figura 4.36. Opciones para configurar el período del PWM.

'Configuración del AD'

En esta opción se tiene dos posibilidades: 'Tiempo de conversión de cada bit' y 'Número de Canales. Como se puede ver en la Figura 4.37, en la primera opción se refiere a cuanto se demora el AD en convertir digitalmente la muestra de voltaje análogo conectado a uno de sus canales. El tiempo de conversión por bit es de 2 a 6 μs con un tiempo de conversión promedio de 4μs. El cálculo de este tiempo se lo explica de la siguiente manera: como ya se dijo anteriormente el conversor AD del microcontrolador es de 10 bits (existe 2 registros internos de 8 bits para almacenar este resultado), pero el programa del PLC que se diseñó en este proyecto utiliza solamente los 8 bits mas significativos del conversor, por lo que el AD se comporta en realidad como uno de 8 bits. Para señales que cambian muy rápido en el tiempo se recomienda utilizar el tiempo de conversión más corto, en cambio para señales lentas (como por ejemplo la temperatura), se recomienda utilizar el tiempo de conversión más largo. Cuando el microcontrolador entra en modo Sleep, la única forma que el conversor AD siga operando es con el modo RC. Esta es la tendencia que se sigue en el presente proyecto.

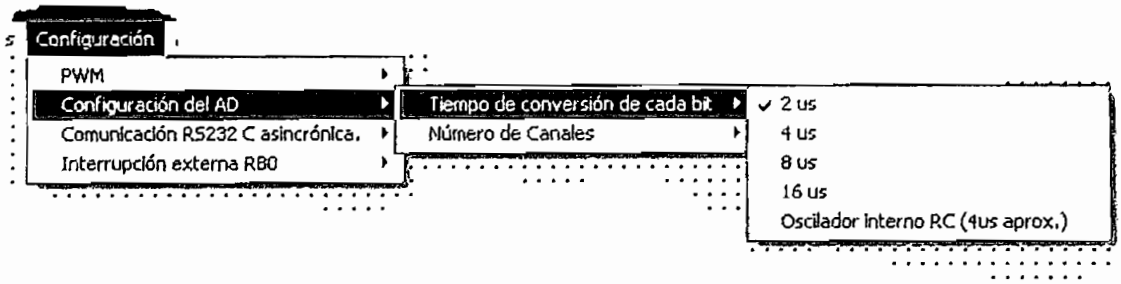


Figura 4.37. Primera opción de configuración del AD.

La segunda opción, 'Número de Canales', que es la que se observa en la Figura 4.38, permite seleccionar el número de canales para el conversor AD. En el

diseño del hardware se optó por construir una tarjeta con 8 canales AD; sin embargo y como este software permite la creación del código hexadecimal con el cual cualquier hardware de PICs. Se deja al usuario la opción de elegir y se indica cuales son las variantes que existen.

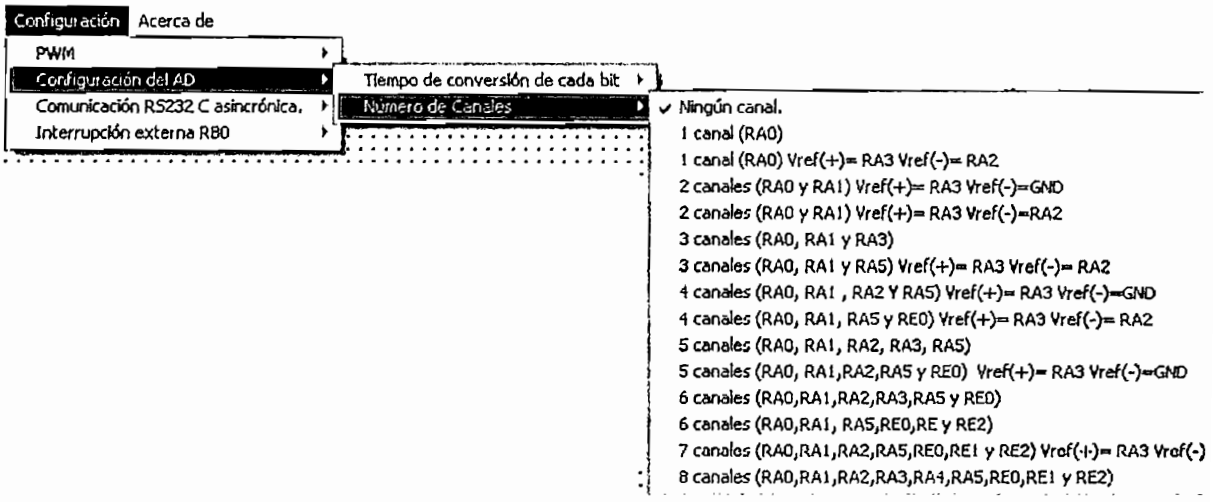


Figura 4.38. Segunda opción de configuración del AD.

'Comunicación RS 232 C asincrónica'

En esta opción, el usuario puede configurar el puerto serial del microcontrolador con dos opciones. En la primera opción se configura la velocidad de transmisión o de recepción de datos, como se puede ver en la Figura 4.39. El valor que se halla seleccionado por defecto es de 9600 baudios.

La comunicación serial es de tipo *full duplex* asincrónica, la línea de recepción Rx está en el Terminal RC7, mientras que la línea de transmisión Tx está en el Terminal RC6. No existe un protocolo definido en este PLC, por lo que el usuario es libre de realizar uno en caso lo desee, al comunicarse con otro dispositivo. Como es de conocimiento la comunicación es de 8 bits y puede ser sin paridad y con 1 bit de parada.

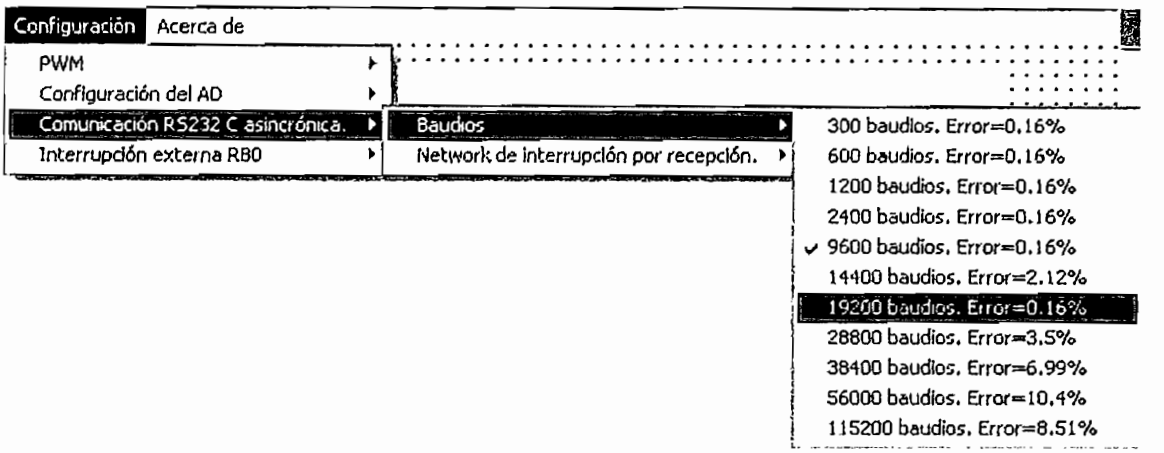


Figura 4.39. Despliegue de la opción Baudios para la configuración de la Comunicación Serial.

La opción 'NetWork de interrupción por recepción' (Figura 4.40) permite al usuario asociar la interrupción que se produce cuando llega un dato al puerto serial del microcontrolador con un NetWork que será utilizado como subrutina, además que es aquí donde se debe colocar la función 'Rx'.

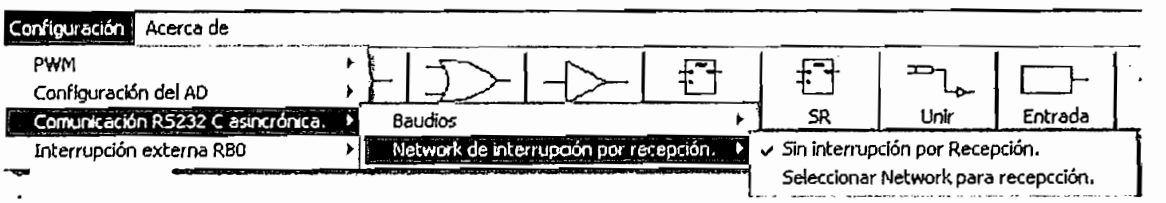


Figura 4.40. Opción para escoger si se desea o no tener interrupción por Recepción.

'Interrupción externa RB0'

Esta es la última opción de la lista desplegable y se conforma por dos comandos que se observan en la Figura 4.41. La primera opción, "Flanco activo de la interrupción externa" permite determinar por que causa se producirá la interrupción externa; es decir, si será por flanco positivo o flanco negativo en el Terminal RB0. La segunda opción 'Seleccionar NetWork para interrupción externa' permite en cambio asociar a esta interrupción un NetWork subrutina, igual que en el caso de la interrupción serial.

Nota: En la prioridad por interrupción, este software antepone la interrupción externa, a la interrupción por comunicación serial. Por lo cual en el flujo del programa y si se usan las dos, el orden deberá ser el antes mencionado.

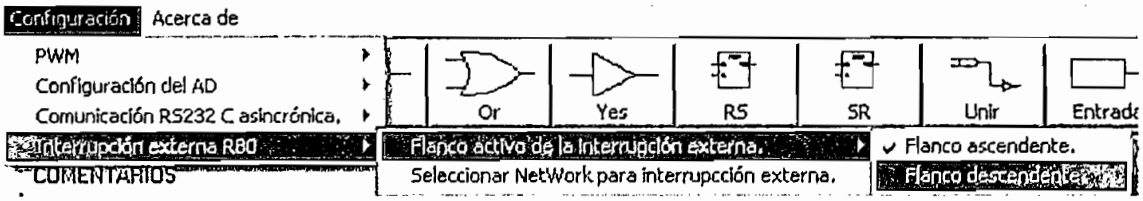


Figura 4.41. Configuración de la interrupción externa en RB0.

Categoría Ayuda

En esta opción se despliega una lista con las siguientes opciones:

'Acerca de PLC877A' y *'Contenido'*. Al dar clic en la primera opción se abre una ventana con un anuncio general de los derechos de autor, mientras que al presionar en Contenido se abre la ayuda interactiva de un archivo *.hlp, llamado FBD.

4.2.2 INTERFAZ GRAFICA DEL SIMULADOR

Cuando ya se tiene un proyecto realizado en el formulario o zona de trabajo, desde de la barra de menús en *'Opciones'* se puede invocar al simulador con *'Simular'*. Al dar un clic en esta opción suceden importantes eventos como:

- Verificar que todas las funciones no tengan ningún Terminal sin conectar.
- Preguntar al usuario hasta que NetWork va a ser el programa principal.
- Comprobar que las funciones GoTo (si es que se usan) llamen a NetWorks que estén dentro del programa principal, caso contrario se notifica al usuario y se detiene el proceso de simulación.
- Verificar que las funciones Call (si es que se usan) llamen a NetWorks que estén fuera del programa principal, caso contrario se notifica al usuario y se detiene el proceso de simulación.

Una vez cumplidas estas condiciones, se despliega el formulario correspondiente al simulador, como el que se muestra en la Figura 4.42.

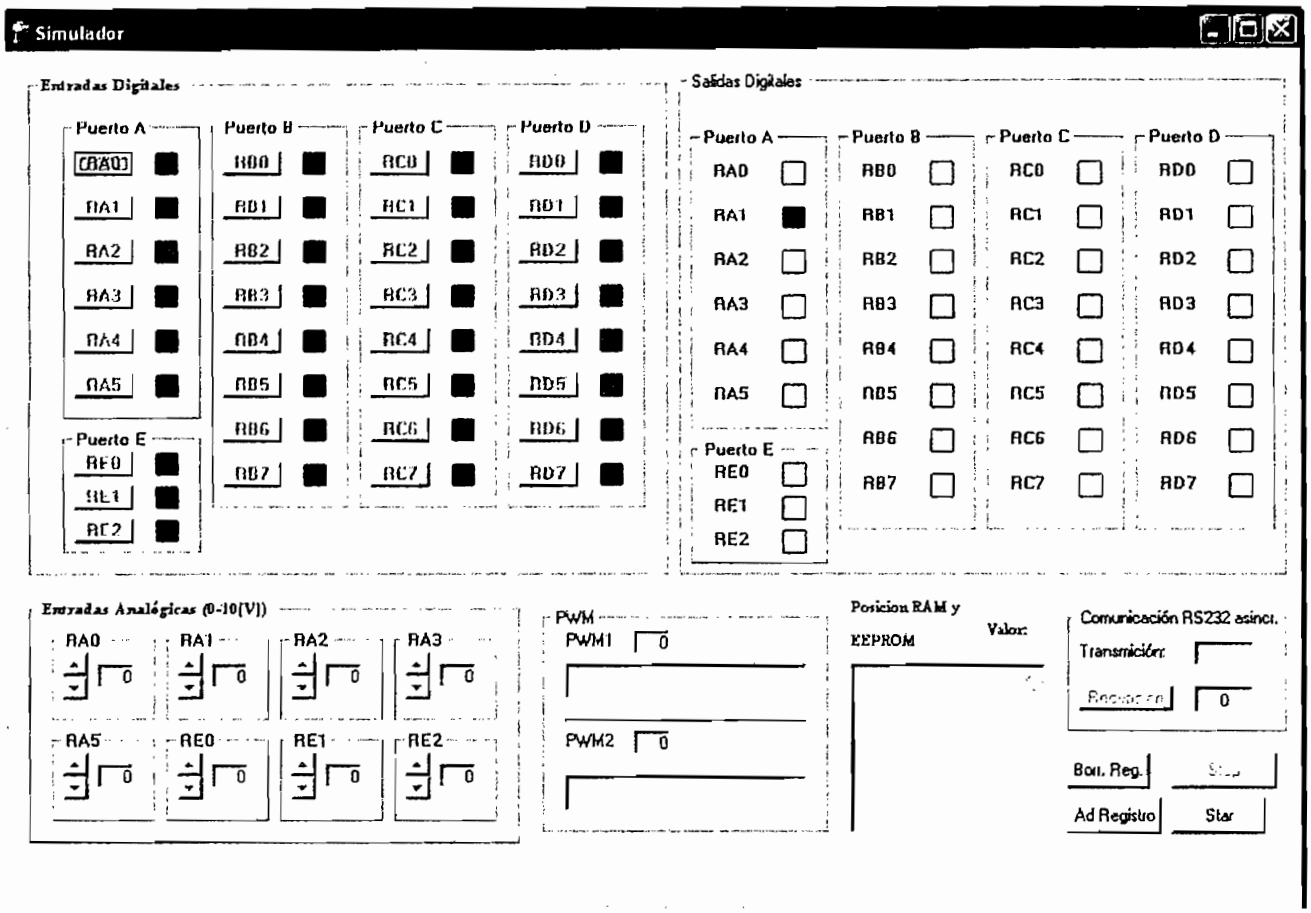


Figura 4.42. Interfaz gráfica del simulador.

Aquí es necesario explicar cual es la función que cumplen todas las herramientas de las cual dispone la interfaz gráfica llamada 'Simulador'.

Entradas Digitales

En la parte superior izquierda de este formulario se encuentran los pulsadores que simulan las entradas digitales. Están presentes todos los bits de los Puertos del PLC ya que todos pueden ser configurados como entradas digitales, pero únicamente están habilitados los pulsadores que son escogidos como tales en el programa o área de trabajo.

Salidas Digitales

En la parte superior derecha se hallan los visualizadores (semejantes a diodos luminiscentes LED), que representan las salidas digitales. En este caso también se hallan presentes todos bits de los puertos del PLC y, de igual manera, solamente están habilitados aquellos que han sido configurados como salida.

Entradas Analógicas

En la parte inferior izquierda, se encuentran todas las entradas analógicas correspondientes a los bits RA0, RA1, RA2, RA3 y RA5 del puerto A y los bits RE0, RE1 y RE2 del puerto E. Su representación es mediante pulsadores de incremento o decremento, que simulan la variación de voltaje mediante la variación de 0 a 255 decimales.

Salidas PWM

En la parte inferior central se halla la visualización de las dos salidas PWM. Cada una se halla representada por un display y una barra de estado que indican el valor de salida del PWM, que puede ser de 0 a 255, este valor representa el ancho del pulso.

Estado de los registros y Comunicación serial

Finalmente, en la parte inferior derecha se encuentra una ventana donde el usuario puede ver los registros de la memoria RAM y/o de la memoria EEPROM, que ha decidido añadir. Para conseguir esto se debe dar un clic en el botón *'Añadir'*. A continuación aparecerá una interfaz como la que se muestra en la Figura 4.43 y que sirve para seleccionar el registro que se desea ver.

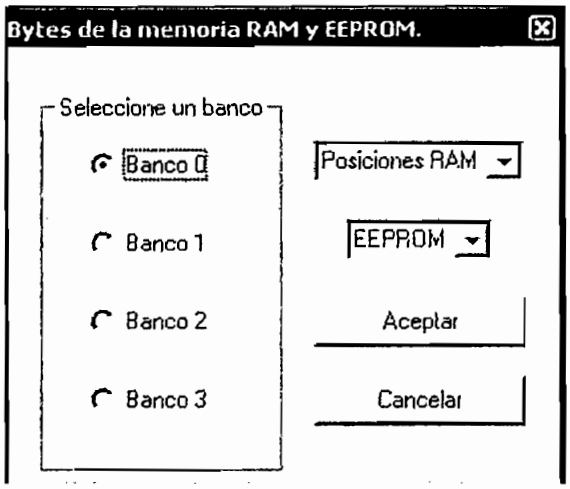


Figura 4.43. Cuadro de opción para añadir los registros que se desea ver durante la simulación.

Luego aparecerá en la parte correspondiente del simulador el registro añadido, como se puede ver en el ejemplo de la Figura 4.44. En este ejemplo se visualizan dos registros uno de la memoria RAM y otro de la memoria EEPROM.

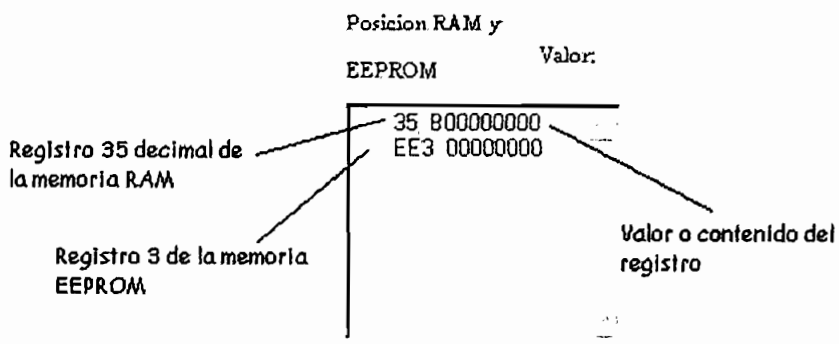


Figura 4.44. Ejemplo de la visualización de los registros de la RAM y EEPROM.

Para una simulación más dinámica se implementó la opción de alterar el valor de dichos registros. Para esto el usuario puede dar doble clic sobre el valor de la posición que desea alterar. En la Figura 4.44 por ejemplo, dando doble clic sobre la posición 35 (RAM) y/o sobre la posición EE3 (EEPROM), se despliega una ventana que pide que se ingrese un valor (de 0 a 255 decimal) como se muestra en la Figura 4.45.

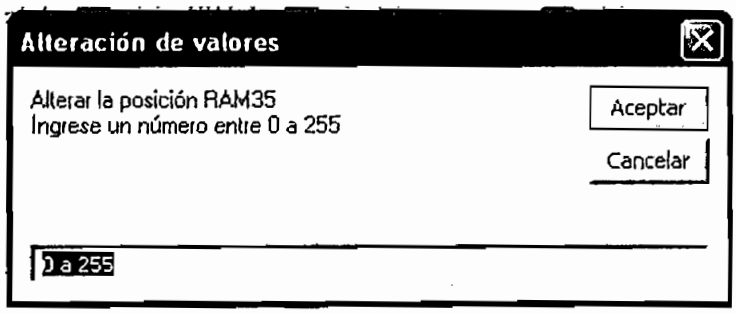


Figura 4.45. Cuadro de diálogo para alterar registros.

En esta misma zona se encuentra el simulador para la comunicación asincrónica RS232C de 8 bits sin paridad, tanto para Transmisión, como para la Recepción. En la Figura 4.46 se tiene la imagen de esta parte.

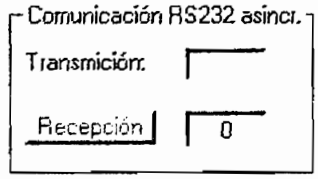


Figura 4.46. Parte del simulador para la comunicación serial.

Cuando el habilitador de una función 'TX' pasa de cero lógico a uno lógico, en la caja de texto, a la izquierda de la etiqueta 'Transmisión' del simulador de comunicación, se visualiza por un instante el valor que se va a transmitir. Mientras que cuando existe en la zona de trabajo al menos una función 'RX', el botón 'Recepción' del simulador de comunicación se habilita, para permitir que se presione dicho botón, con lo cual se puede simular, que ha llegado un byte al puerto serial.

4.2.3 INTERFAZ GRAFICA PARA DESCARGAR AL PLC

Como ya se explicó anteriormente, para poder generar la señal de reloj y la señal de datos para programar al microcontrolador, se utiliza el puerto paralelo del computador. Visual Basic no puede directamente trabajar con dicho puerto, sino que recurre a una librería DLL desarrollada en C++, que se puede conseguir fácilmente desde Internet.

Para invocar al programador desde el formulario de trabajo, se debe ingresar en la barra de menús en la parte de 'Opciones' y luego 'Descargar a PLC'. De igual manera que cuando se invocaba al simulador, se realizan los 4 eventos que ya se mencionó anteriormente, y que se deben cumplir antes de ir a la descarga de un proyecto.

Una vez cumplidos esas condiciones, se despliega el formulario del programador que se muestra en la Figura 4.47

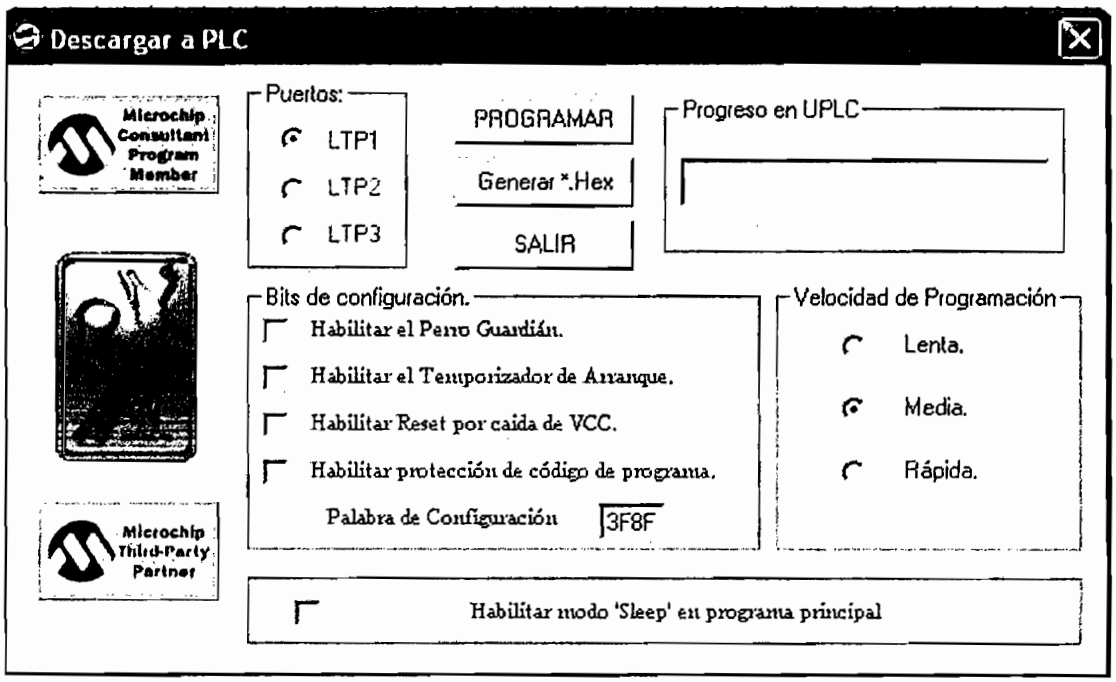


Figura 4.47. Interfaz para descargar al PLC.

Esta ventana está conformada por las siguientes opciones:

Puertos

En la parte superior de la ventana de la Figura 4.47 anterior, el usuario puede escoger cual puerto paralelo de la computadora desea usar, ya sea el LPT1, LPT2 o LPT3.

Bits de Configuración

En esta zona el usuario tiene las siguientes posibilidades:

1. Habilitar el Perro Guardián. Esta opción activa a un temporizador interno del microcontrolador que es independiente del cristal oscilador. Cuando dicho temporizador se desborda produce un reset del PIC, por lo que es útil para evitar que el microcontrolador se quede en un estado de indecisión. Para evitar un reset indeseado, el usuario debería limpiar a dicho temporizador constantemente, pero en este proyecto se halla implementada esta limpieza, y además es transparente para el usuario. Se la realiza entre cada NetWork del programa del PLC.

2. Habilitar el temporizador de arranque. Esta opción evita que en el momento que se aplica 5V de polarización al PIC este opere normalmente, sino que entre en un estado de 'stan by' por un tiempo de 72ms aproximadamente, en el cual la fuente de alimentación se puede estabilizar; además de evitar cualquier interferencia por arranque de un equipo, que pueda afectar al PIC.

3. Habilitar reset por caída de VCC. Esta opción permite que se produzca un reset cuando el voltaje de la fuente de alimentación falla.

4. Habilitar código de protección de programa. Permite proteger el código de programa ubicado en la memoria flash, de esta manera se evita que otras personas puedan copiar el código hexadecimal almacenado dentro del PIC.

En la 'Palabra de configuración' que se encuentra ubicada en la parte inferior de esta sección, se escribe automáticamente un valor Hexadecimal, correspondiente a la configuración de los bits antes mencionados y de otros que están configurados por defecto. Como la programación por alto voltaje que se mencionó durante el Capítulo 2.

En la parte inferior de la configuración de bits se encuentra la habilitación del modo 'Sleep' para el programa principal. El modo 'Sleep' permite que el procesador del microcontrolador entre en un estado de bajo consumo de energía

en el cual no realiza ninguna operación y los puertos del PIC se quedan en el estado lógico anterior previo a ingresar a este modo. El procesador saldrá de este modo cuando se produzca cualquier tipo de interrupción. Esta función es válida en aquellos proyectos donde el programa principal no realiza ninguna operación y el PLC solo actúa en los eventos por las interrupciones.

Programar

Esta opción permite al usuario descargar el programa al PLC. La barra de estado que se encuentra a la derecha (Progreso en UPLC) da una idea de cuanto tiempo falta para terminar la programación. Cuando se ha terminado la descarga del programa al PLC, se pregunta al usuario si desea la verificación del programa almacenado en la memoria FLASH, con un cuadro de diálogo como se indica en la Figura 4.48.

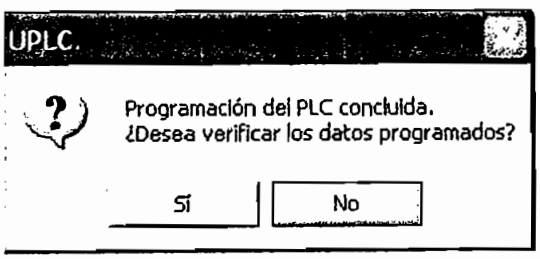


Figura 4.48. Cuadro de diálogo para verificación de la programación.

En caso de error se informa al usuario de que la programación ha fallado, con un cuadro de advertencia.

Generar *.HEX

Esta herramienta permite crear un archivo de tipo hexadecimal desarrollado por INTEL para la programación de memorias EEPROM. Con esta opción el usuario puede crear este código y programar al PIC con cualquier hardware programador que desee. Al dar clic en este botón se despliega la ventana como la que se indica en la Figura 4.49.

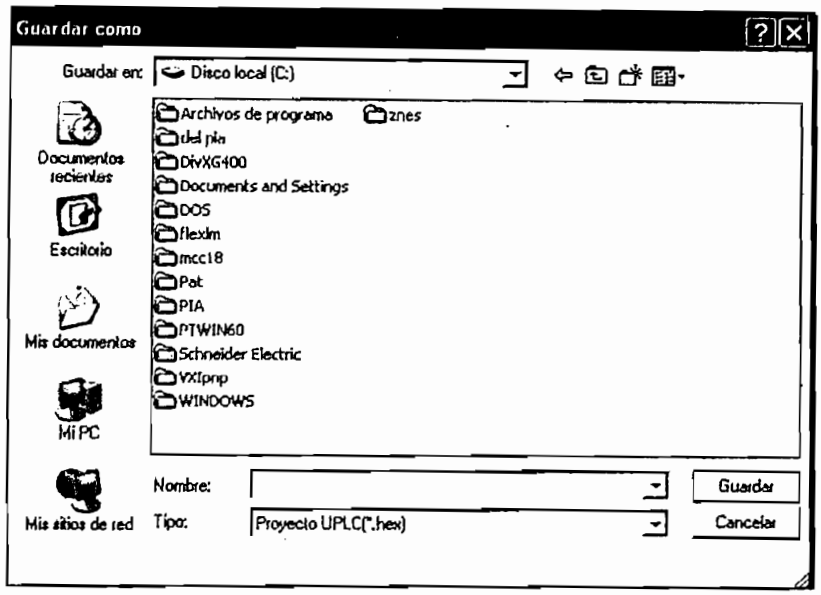


Figura 4.49. Ventana para guardar el código hexadecimal.

El usuario podrá guardar en cualquier parte el código *.HEX generado.

Velocidad de Programación

En la parte inferior derecha está la opción que permite al usuario, programar al microcontrolador a distintas velocidades. Esta velocidad depende de la velocidad del computador, pero mientras más rápido se programe al PIC, es más probable que dicha programación falle, es por eso que existen tres opciones para poder calibrar dicha rapidez, que más bien estará de acuerdo con la computadora que se esté trabajando.

4.3 DESARROLLO DE LOS ALGORITMOS PARA EL MICROCONTROLADOR PIC

El diseño del software que es transparente al usuario está conformado por las siguientes partes:

- Diseño de la interfaz gráfica para la programación del PLC.
- Diseño del simulador para el PLC.

- Diseño del programador del microcontrolador y generación del código hexadecimal.
- Diseño del firmware para el microcontrolador.

4.3.1 DISEÑO DE LA INTERFAZ GRÁFICA PARA LA PROGRAMACIÓN DEL PLC

La parte más complicada de este proyecto en realidad fue el desarrollo de la Interfaz gráfica para elaborar programas o proyectos, ya que esto significó realizar acciones como: cargar las funciones al lugar de trabajo, enlazar a estas mediante líneas conectores y cambiar su posición, entre otras opciones más. En todos estos eventos se destaca como principal herramienta el manejo del Mouse o ratón del computador, ya que este permite realizar las acciones antes mencionadas.

Debido a que el desarrollo de la interfaz gráfica es la parte más compleja de este proyecto, representar todos los algoritmos de forma detallada es muy extenso. Por lo cual se decidió detallar de la manera más simple, los más importantes y que se cree serán de más utilidad a los lectores de esta Tesis.

Los eventos que controla Visual Basic con el Mouse (Ratón) son los siguientes:

- MouseDown. Ocurre cuando el usuario pulsa cualquier botón del Ratón.
- MouseUp. Ocurre cuando el usuario suelta cualquier botón del Ratón.
- MouseMove. Ocurre cada vez que el usuario mueve el puntero del Ratón a una nueva posición.

Para que exista un mejor entendimiento de algunos eventos que se producen en Visual Basic, es necesaria una explicación mediante diagramas de flujo, como los que se muestran a continuación.

Las funciones (AND, OR, PWM, etc.) que son usadas en la barra de herramientas de la Interfaz gráfica del Área de trabajo, son controles de tipo IMAGE que

contienen la imagen de cada función. Están ocultas en el formulario de trabajo ya que tienen la propiedad Visible en False. A partir de estas se crearán nuevas funciones puesto que estos controles conforman una matriz, que se ira incrementando cuando se importen más funciones al formulario de trabajo, o se irá decrementando cuando se borren funciones de la zona de diseño.

Cuando se selecciona una función u operación de la barra de herramientas, existe una variable denominada 'función' que se carga con un número para identificar que opción se ha escogido, al dar clic sobre la zona de diseño se hace visible la función elegida, en la Figura 4.50 se muestra es el diagrama de flujo de esta acción.

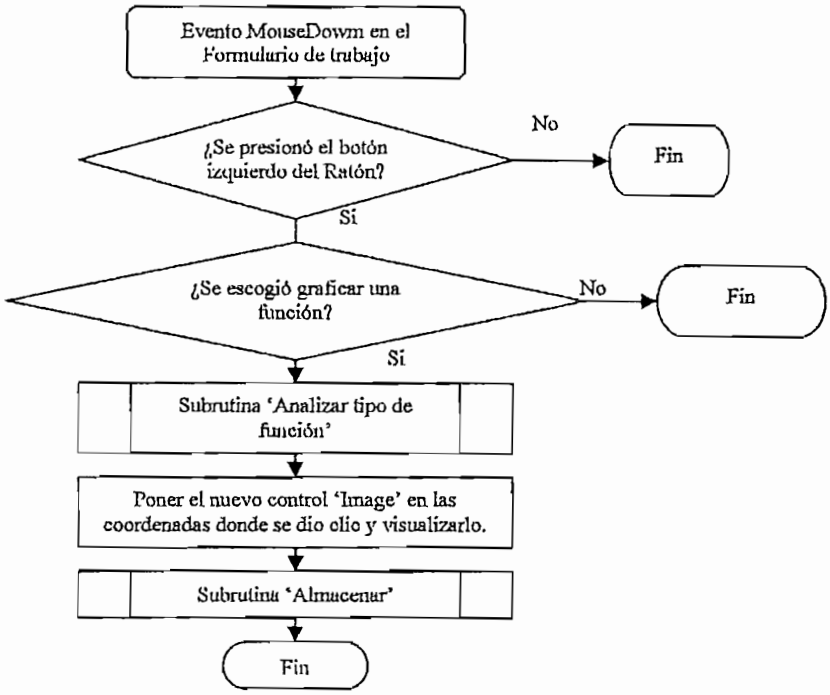


Figura 4.50. Diagrama de flujo para cargar una función en el formulario de trabajo.

La estructura del algoritmo de la Figura anterior se describe a continuación en lenguaje estructurado:

Analizar tipo de función

Si la función escogida es un bit de entrada o de salida

Preguntar al usuario si desea un bit de un Puerto o un bit de la memoria RAM.

Permitir que el usuario seleccione el bit (De 0 a 7) y el registro deseado.

En caso de no seleccionar algún bit, o seleccionar un bit de puerto como entrada cuando este ya se está utilizando como salida, advertir al usuario y terminar la tarea

Si la función escogida es un byte de entrada o de salida

Preguntar al usuario si desea un Puerto o un byte de la memoria RAM o EEPROM.

Si se escogió un puerto

Permitir al usuario configurar que bits desea como entrada o salida

Si se escogió RAM o EEPROM.

Permitir al usuario que seleccione el registro que desee

Caso contrario

Advertir al usuario que se cancelo la tarea.

Si la función escogida es 'Numero'

Permitir que el usuario ingrese un número de 0 a 255 o cancelar la tarea.

En caso de ingresar un parámetro no valido por el usuario, éste puede ingresar el número otra vez o cancelar la tarea

Si la función escogida es 'Ton', Toff', 'Inc' o 'Dec'

Si el número de cualquiera de estas funciones es mayor que 8, se advierte al usuario que ya no se puede agregar más funciones de la que se escogió y se cancela la tarea.

Si la función escogida es 'PWM'

Si el número de funciones PWM que están en el formulario de trabajo es igual a 2, se advierte al usuario que ya no es posible agregar más funciones de este tipo y se cancela la tarea.

Fin Tarea

Poner el control 'Image' donde se dio clic y visualizarlo.

Dependiendo de la función seleccionada, la caja de imagen 'Image' se carga con la figura de la función escogida.

Se asigna a las coordenadas X e Y de la caja de imagen 'Image', aquellas donde se dio clic.

Se pone la propiedad 'Visible' de la caja de imagen 'Image' en verdadero.

Fin Tarea.

Almacenar

Se almacena en una matriz las siguientes características:

Tipo de función.

Coordenadas X e Y de la caja de imagen 'Image'

Coordenadas X e Y de los terminales de entrada y salida

Fin Tarea

Cuando se selecciona los comandos 'Borrar' o 'Mover' de la barra de herramientas, estos se ejecutan cuando se da un clic sobre las funciones que se encuentran en el formulario de trabajo. El algoritmo para ejecutar estas acciones se representa en la Figura 4.51.

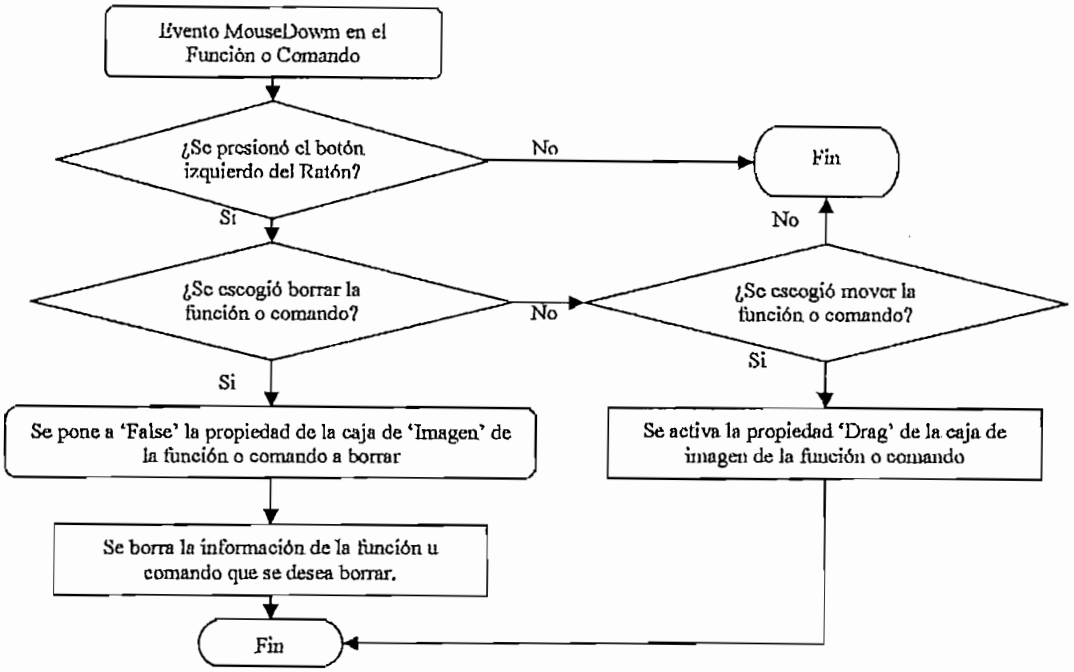


Figura 4.51. Diagrama de Flujo de los eventos borrar y mover.

Cuando se desea mover una función o comando, la propiedad 'Drag' de la caja de imagen debe estar activada. Mientras que, si se mueve el puntero del Mouse con uno de sus botones presionados se produce un evento llamado 'DragDrop', el algoritmo que produce el evento mover la caja de imagen de una función o comando, es el que se indica en la Figura 4.52.

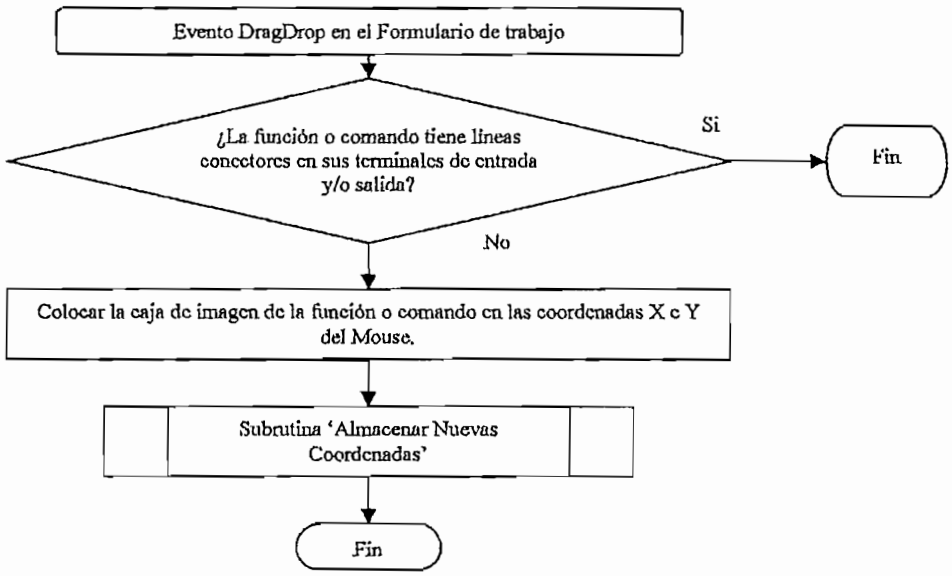


Figura 4.52. Diagrama de Flujo para el evento mover una función.

A continuación se explican los algoritmos correspondientes, a eventos de la barra de menús. El algoritmo para crear un nuevo proyecto desde la barra de menús se describe en la Figura 4.53.

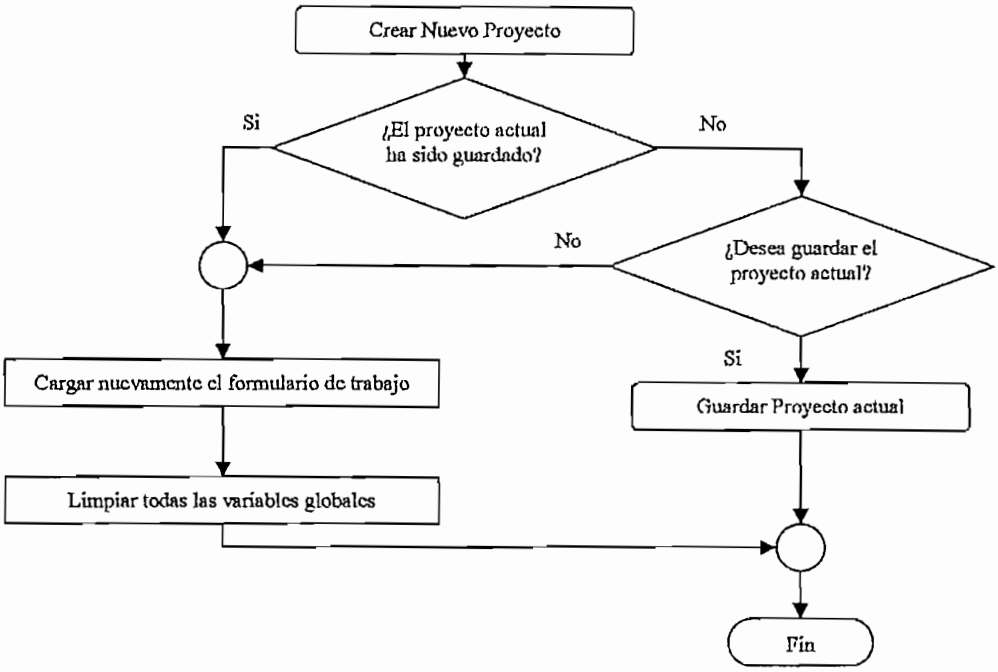


Figura 4.53. Diagrama de Flujo del algoritmo para crear un nuevo proyecto.

La estructura del algoritmo de la Figura 4.53 anterior se describe a continuación en lenguaje estructurado:

Cargar nuevamente el formulario de trabajo

El formulario actual de trabajo se lo descarga de la memoria RAM del computador

Se carga un nuevo formulario de trabajo, al realizar esto, por defecto, las variables utilizadas aquí se inicializan a sus respectivos valores iniciales

Fin Tarea

Limpiar variables globales.

Se limpian las variables globales que están a nivel del formulario contenedor MDI y el módulo .BAS

Fin Tarea.

El algoritmo para guardar un proyecto desde la barra de menús se representa en la Figura 4.54.

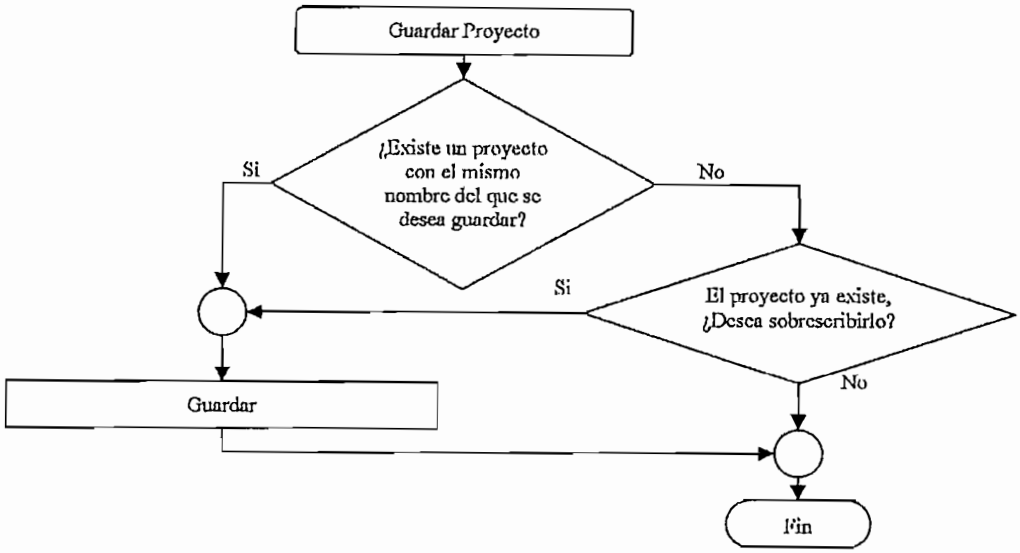


Figura 4.54. Diagrama de Flujo para guardar un proyecto.

La estructura del algoritmo de la Figura 4.54 se describe a continuación en lenguaje estructurado:

Guardar

Si existe un archivo con el mismo nombre que se desea guardar

Borrar el archivo

Crear un nuevo archivo

Se escribe el archivo con las variables que contienen información del proyecto

Fin Tarea

El algoritmo para abrir un proyecto existente se describe en la Figura 4.55.

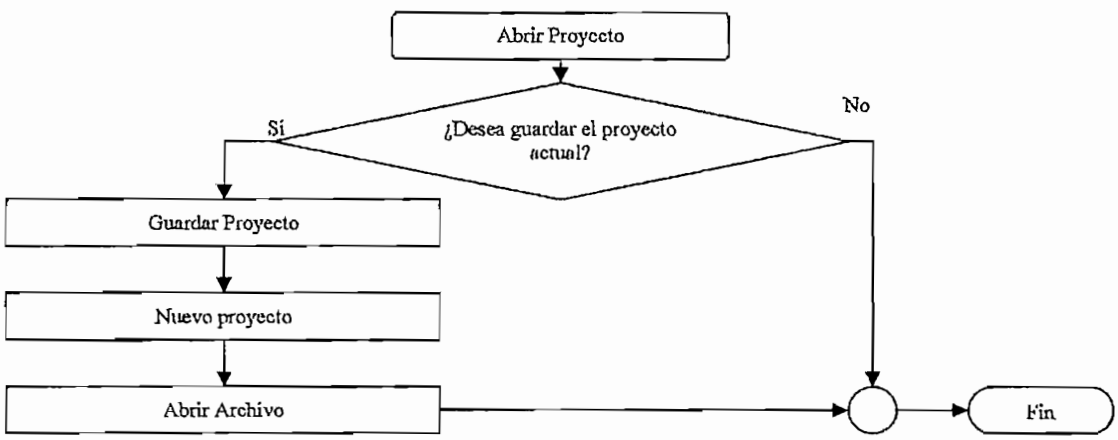


Figura 4.55. Diagrama de Flujo del algoritmo para abrir un proyecto.

La estructura del algoritmo de la Figura 4.55 anterior se describe a continuación en lenguaje estructurado:

Guardar Proyecto

Se llama a la rutina Guardar Proyecto

Fin Tarea

Nuevo Proyecto

Se llama a la rutina Nuevo Proyecto

Fin Tarea

Abrir Proyecto

Se lee el archivo, para cargar a las variables correspondientes.

En función de las variables se carga y visualiza las cajas de imágenes de las funciones y comandos FBD.

Fin Tarea

4.3.2 DISEÑO DEL SIMULADOR PARA EL PLC

En esta parte, se hace necesario dividir la explicación en dos partes:

1. Desarrollo de los algoritmos de las funciones que posee el PLC para la Simulación
   - Funciones tipo booleanas.
   - Funciones a nivel de Bytes.

2. Desarrollo del algoritmo para el Simulador

4.3.2.1 Funciones Tipo Booleanas

Son aquellas funciones que permiten trabajar a nivel de bits, pueden ser las funciones AND, OR, XOR entre otras. Su implementación es necesaria pero como el microcontrolador utilizado, no posee todas las operaciones a nivel de bits, por lo cual fue necesario hacer el código, en base a las operaciones que se nombrarán a continuación:

- bcf REG, b que pone a 0L el bit 'b' de un registro de la memoria RAM
- bsf REG, b que pone a 1L el bit 'b' de un registro de la memoria RAM
- btfsc REG, b salta a la siguiente instrucción si el bit 'b' del su registro REG está en 0 L.
- btfss REG, b salta a la siguiente instrucción si el bit 'b' del su registro REG está en 1 L.

El código en lenguaje ensamblador para la Función AND de dos entradas (Tabla 4.1) y el diagrama de flujo (Figura 4.56) es el siguiente:

Donde:

RA,1= Entrada 1

RA,2= Entrada 2

RB,1= Salida

Posición en la memoria de programa	Código Ensamblador	Descripción
n	Btfss RA,1	Si RA1 está en 1 L, entonces salto a n+2
n+1	Goto n+6	Salto a n+6
n+2	Btfss RA,2	Si RA2 está en 1 L, entonces salto a n+4
n+3	Goto n+6	Salto a n+6
n+4	Bsf RB,1	Pongo a 1 l RB1
n+5	Goto n+7	Salto a n+7
n+6	Bcf RB,1	Pongo a 0L RB1
n+7	Siguiente inst.	

Tabla 4.1. Código en lenguaje ensamblador para la Función AND, entre dos bits cualquier.

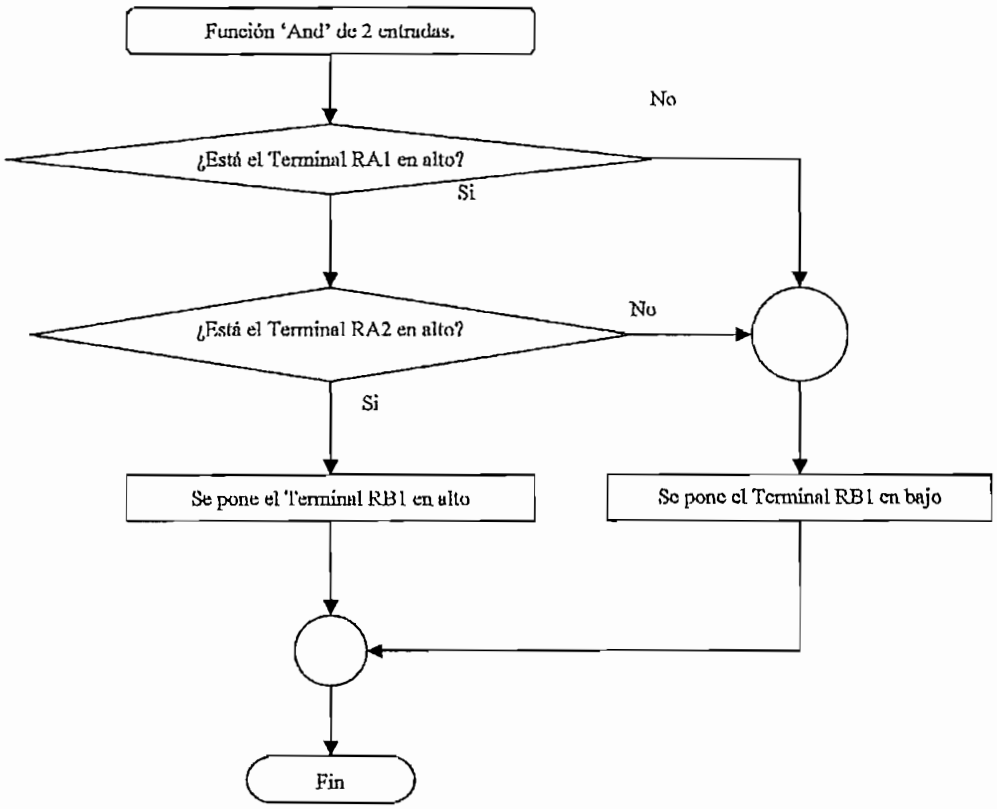


Figura 4.56. Diagrama de flujo básico de la Función AND.

El diagrama de flujo de la Figura 4.56 anterior, permite realizar un AND lógico, cuyas entradas y salida se encuentran en el mismo banco de la memoria RAM. En este caso, posiblemente se necesita introducir las instrucciones para el cambio de banco la primera vez, antes de escribir el código de la función.

Sin embargo, no todas las operaciones se hacen entre bits que estén en registros del mismo 'Banco', esto implica agregar instrucciones que permitan al puntero de la memoria RAM del microcontrolador, estar en el banco apropiado antes de cualquier operación sobre un Byte (leer o escribir en un byte o bits que lo conforman). El algoritmo lo que hace es revisar que si todos los bits que integran una función estén en el mismo Banco, y si al menos un registro conectado a la función está en un banco diferente, introduce varias veces las instrucciones de cambio de banco, con lo que el programa aumenta su tamaño. Pero su funcionamiento es el correcto.

En la Figura 4.57 se indica un ejemplo de lo dicho anteriormente, en la derecha está la función AND en la que un Terminal de entrada esta en el bit 7 del puerto C, el otro esta en el bit 2 del registro 34 (banco 0) y la salida se conecta en el bit 4 del registro 40 (banco 0), en este caso todos los terminales de la función AND están en el banco 0 (todos los puertos del microcontrolador están en el banco 0).

A la izquierda de la Figura 4.57 se tiene una función AND, con un Terminal de entrada conectado en el bit 3 del registro 162 (banco 1), y los demás terminales conectados a bits de registros del banco 0. Como es de suponerse en este caso el código de la función AND es mucho más largo que el código de la función AND de la izquierda, ya que en este caso el algoritmo introducirá constantemente el código de cambio de banco, dentro del código de la función AND.

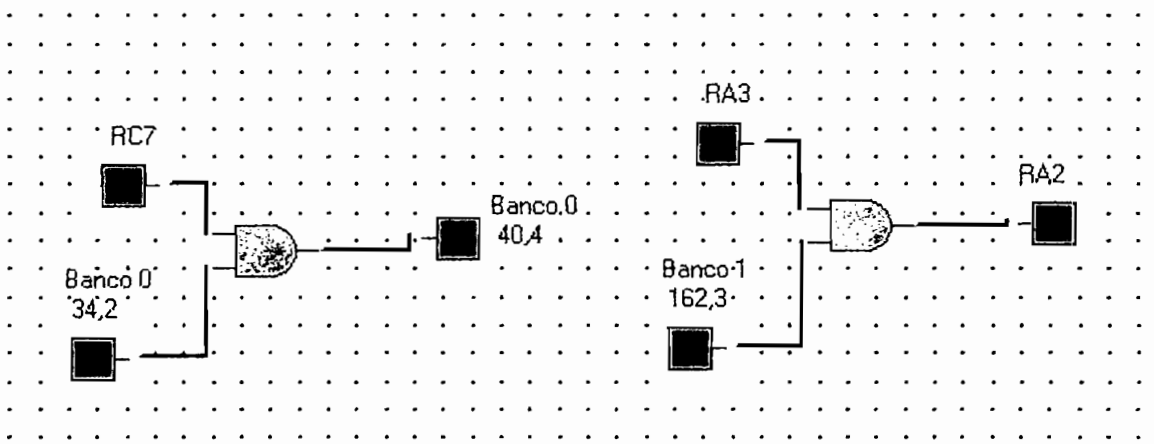
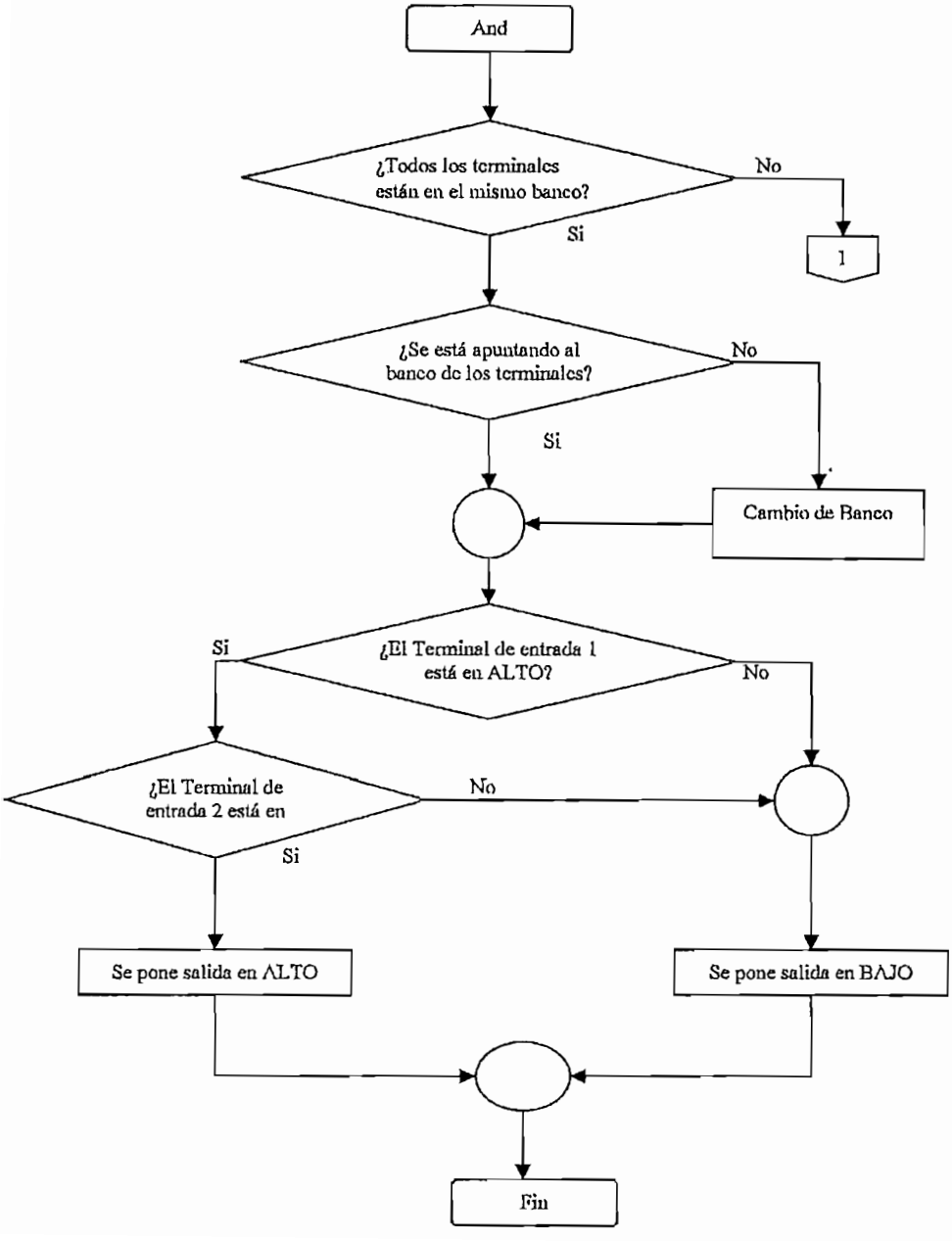


Figura 4.57. Ejemplo de conexión para la Función AND.

'Función AND'

En la Figura 4.58 se representa los dos diagramas de flujo, para los posibles casos de funciones AND.


And
¿Todos los terminales están en el mismo banco?
No
Si
1
¿Se está apuntando al banco de los terminales?
No
Si
Cambio de Banco
Si
¿El Terminal de entrada 1 está en ALTO?
No
¿El Terminal de entrada 2 está en
No
Si
Se pone salida en ALTO
Se pone salida en BAJO
Fin

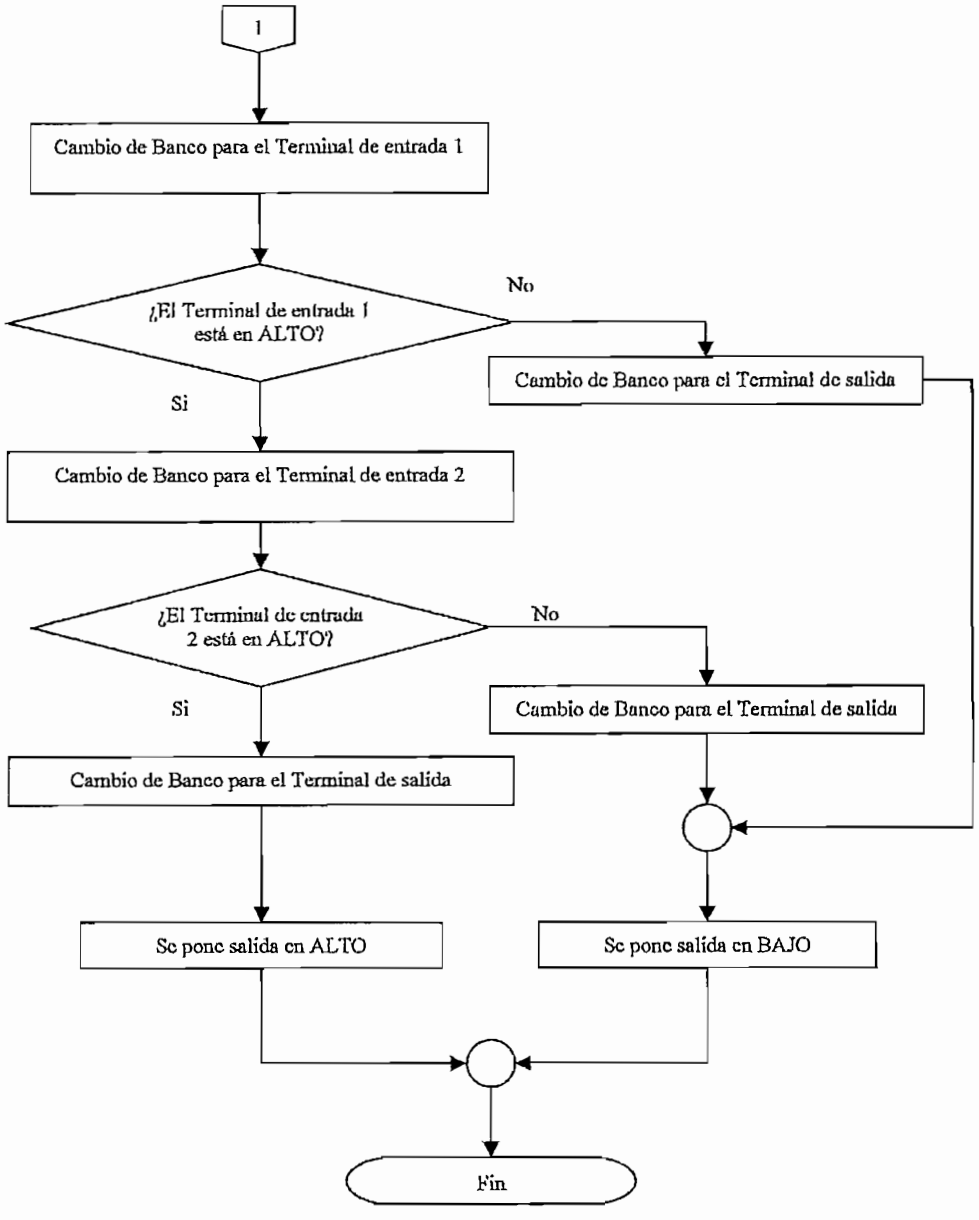


Figura 4.58. Diagrama de Flujo de la Función AND.

Como se puede apreciar en la Figura 4.58 anterior, cuando los terminales de una función están conectados a registros que se encuentran en diferentes bancos, el código escrito para el PIC es mucho más largo. Lo cual constituye la principal desventaja de un lenguaje de alto nivel para programar un microcontrolador.

El algoritmo anteriormente presentado es similar para todas las funciones Booleanas (AND, NAND, NOT, YES, OR, XOR, RS, etc.), por lo cual no se los representa.

Indicaciones generales del uso de las funciones Booleanas

Las funciones Booleanas necesitan en ocasiones 'bits auxiliares', para almacenar el resultado de sus operaciones, en el ejemplo de la Figura 4.59 se muestra por que son necesarios dichos bits. El resultado de las compuertas NAND y OR necesita por ejemplo, almacenarse en bits auxiliares antes de ejecutar la función RS.

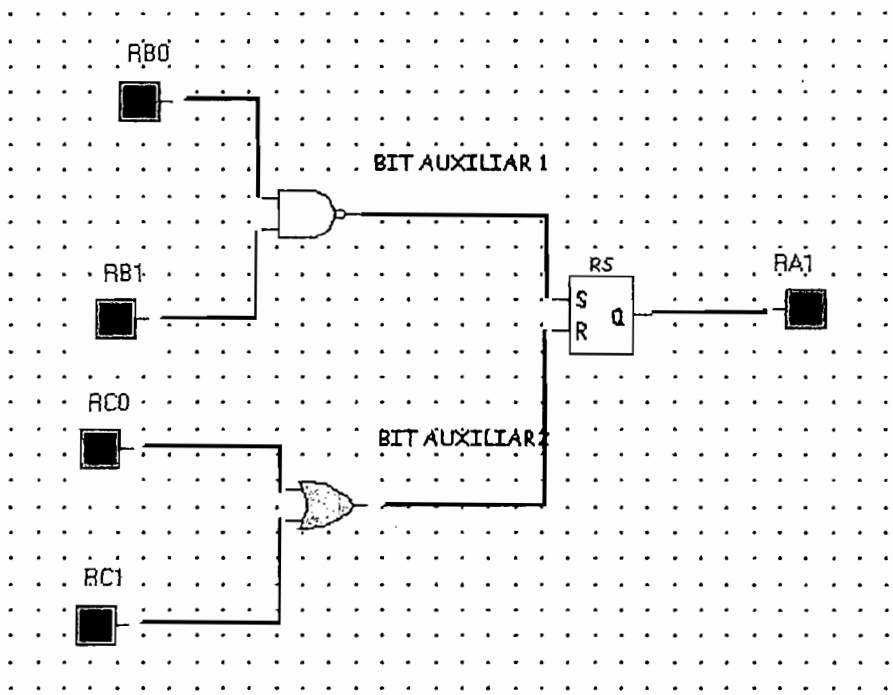


Figura 4.59. Ejemplo de conexión en el que se necesita un bit auxiliar.

Estos bits auxiliares son los registros 32 y 33 decimal (20 y 21H) del banco 0 de la memoria RAM, por lo tanto están reservados para este propósito y no están disponibles para el usuario. Al cambiar de NetWork el programa siempre limpia dichos registros auxiliares para ser utilizados nuevamente, es decir que por cada NetWork se pueden poner un máximo de 16 líneas conectores internas, si se sobrepasa este número el usuario deberá pasar a utilizar otro NetWork.

Otra parte fundamental para las funciones booleanas es el código que generan para la simulación, descarga al microcontrolador y creación del archivo hexadecimal, que no puede ser generado en cualquier orden. En ejemplo de la

Figura 4.59 anterior se debe generar primero, el código de las compuertas NAND o OR (cualquiera de las dos puede ser la primera), sus resultados se almacenarán en los bits auxiliares y finalmente se generará la función RS cuyo resultado se almacenará en el Terminal RA1. El algoritmo que genera el código para Simulación/descarga/archivo .HEX debe leer en un orden correcto las funciones booleanas. Se parte al contrario de lo determinado anteriormente, es decir de atrás hacia delante. Primero se busca todas las funciones booleanas cuyas salidas estén conectadas a un bit de un puerto o a un bit de registro de la memoria RAM, pero no los registros auxiliares y se almacena esta información en un archivo o matriz estática. Luego se procede a analizar los valores del archivo para leer la información sobre los terminales de entrada de las funciones contenidas en la matriz estática y se procede a buscar a las funciones cuyos terminales de salida están conectadas a las entradas de las funciones del archivo anteriormente explicado.

Una vez encontradas estas nuevas funciones se las agrega a la matriz estática. A partir de estas nuevas funciones añadidas al registro, se procede de igual manera a buscar otras funciones y todo esto llega a un fin cuando se descubre que los terminales de entrada de las funciones son bits de cualquier puerto o bits de cualquier registro de la memoria RAM (pero no bits auxiliares), de esta manera se consigue un orden invertido de las funciones booleanas.

Para generar el código simulación/descarga/archivo .HEX, se debe leer la matríz archivo desde su último valor hasta el primero, en la Figura 4.60 se representa el diagrama de flujo del algoritmo de búsqueda de funciones booleanas.

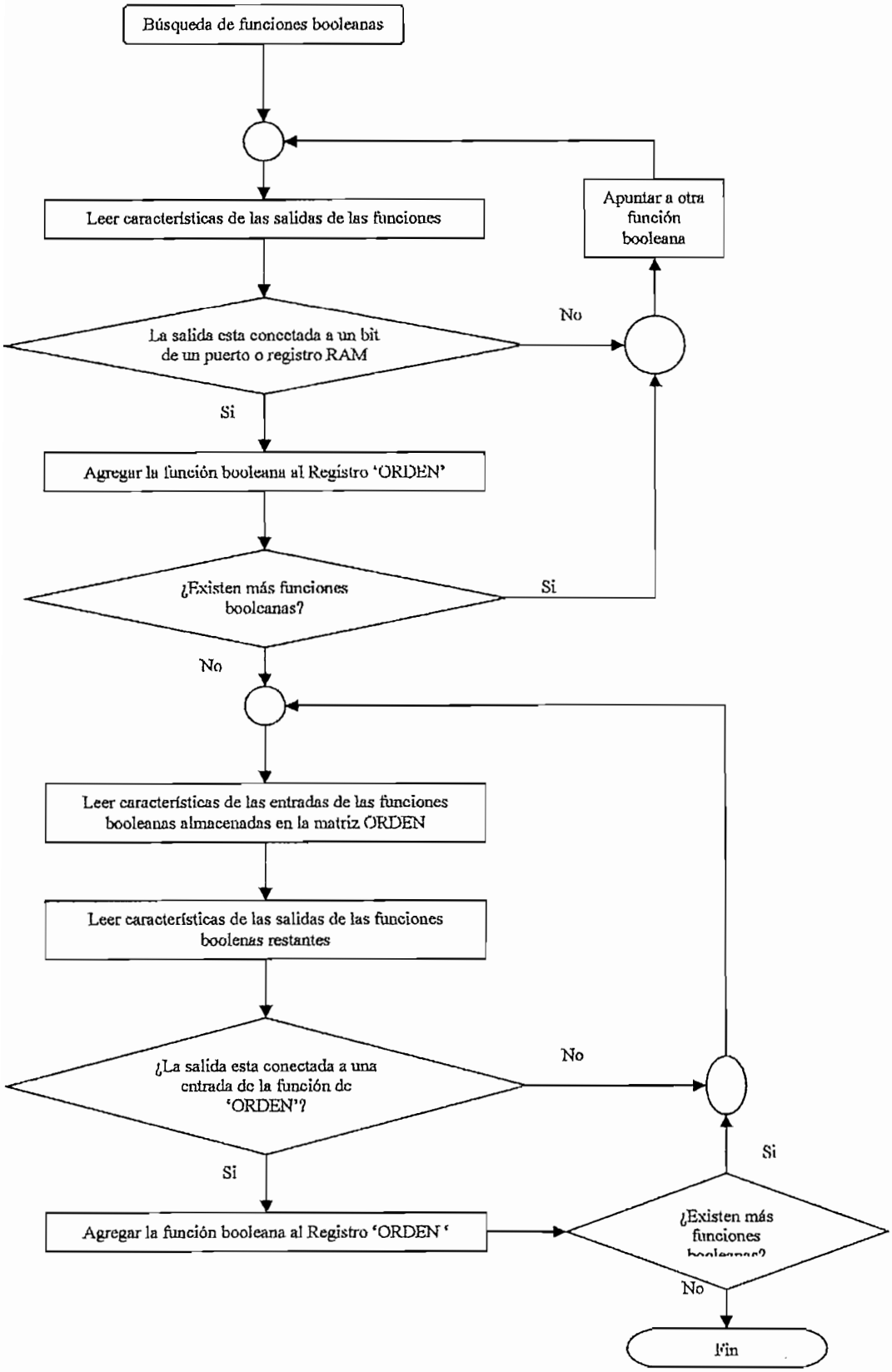


Figura 4.60. Diagrama de Flujo para búsqueda de funciones booleanas.

4.3.2.2 Funciones a Nivel de Bytes

Son aquellas funciones que permiten operar y manejar directamente todos los bits de un registro de la RAM o puerto del microcontrolador. Las funciones que posee el microcontrolador y que se utilizarán para modelar las funciones FBD son:

Operaciones que manejan registros:

- ADDWF REG, d Suma el acumulador W con un registro REG y el resultado lo envía a W si d=0 ó a REG si d=1.
- ANDWF REG, d Realiza un AND bit a bit con el acumulador W y el registro REG, el resultado lo envía a W si d=0 ó a REG si d=1.
- CLRF REG Pone a 0 todos los bits del registro REG.
- CLRW Pone a 0 todos los bits del registro acumulador W.
- COMF REG, d Complementa al registro REG, el resultado lo envía a W si d=0 ó a REG si d=1.
- DECF REG, d Decrementa en 1 el registro REG, el resultado lo envía a W si d=0 ó a REG si d=1.
- INCF REG, d Incrementa en 1 el registro REG, el resultado lo envía a W si d=0 ó a REG si d=1.
- IORWF REG, d Realiza un OR bit a bit con el acumulador W y el registro REG, el resultado lo envía a W si d=0 ó a REG si d=1.
- MOVF REG, d Mueve el registro REG a W si d=0 o a REG si d=1
- MOVWF REG Mueve W al registro REG
- SUBWF REG, d Resta W al Registro REG, el resultado lo envía a W si d=0 ó a REG si d=1.
- SWAPF REG, d Intercambia la parte alta de bits del registro REG con la parte baja, el resultado lo envía a W si d=0 ó a REG si d=1.
- XORWF REG, d Realiza un XOR bit a bit con el acumulador W y el registro REG, el resultado lo envía a W si d=0 ó a REG si d=1.

Operaciones que manejan números:

- ADDLW Num Suma un número Num con el contenido de W, 0 <= Num <=255.
- ANLD Num Realiza un AND lógico bit a bit con un número Num y el contenido de W, 0 <= Num <=255.
- IORLW Num Realiza un OR lógico bit a bit con un número Num y el contenido de W, 0 <= Num <=255.
- MOVLW Num Mueve un número Num al acumulador W, 0 <= Num <=255.
- SUBLW Num Resta W de un número Num, 0 <= Num <=255.
- XORLW Num Realiza un XOR lógico bit a bit con un número Num y el contenido de W, 0 <= Num <=255.

Para una explicación mas estructurada, se decidió clasificar las funciones a nivel de Bytes en 9 grupos de funciones, cuya lógica es similar (diagrama de flujo semejante), por lo cual se representará únicamente el algoritmo de una función de cada grupo. Los grupos son los siguientes:

1. Funciones de Transferencia (mueven un byte a otro): Move, MoveC, Swap y PWM.
2. Funciones de comparación: Mayor, Mayor o igual, Igual y Diferente.
3. Funciones de Operación Binaria para bytes: And, Or y Xor.
4. Funciones de Temporización: TON y TOFF.
5. Funciones de Alteración: 'Incremento' y 'Decremento'.
6. Funciones para el manejo de la memoria de datos EEPROM: WEEPROM y RWEEPROM.
7. Funciones de Salto y subrutina: 'Go To' y 'Call'.
8. Funciones para recepción y transmisión serial asincrónica RS232C.
9. Función de conversión analógica, 'AD'.

Funciones de Transferencia

Para el primer grupo se representará el diagrama de flujo de la función 'Move'. En el ejemplo de la Figura 4.61 se tiene el ejemplo de la función 'MOVE' que permite copiar todo el contenido de un registro y enviarlo a otro (como ya se explicó anteriormente). Para este caso se copia el valor del Puerto B y se lo envía al Puerto C. Esta función solamente se ejecuta si el habilitador 'E' donde está conectado RA0 está en 1L.

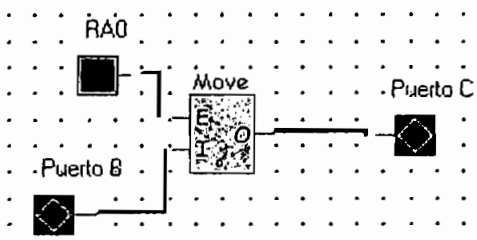


Figura 4.61. Conexión para la Función Move.

Posición en la M. de programa	Código Ensamblador	Explicación
n	Btfss RA,0	Si RA0 está en 1 L, entonces salto a n+2
n+1	Goto n+4	Salto a n+4
n+2	Movf PORTB , W	Copio el contenido de PORTB y lo envío a W
n+3	Movwf PORTC	El contenido de W lo envío a PORTC
n+4	Next Instrucción	

Tabla 4.2. Código Ensamblador de la Función Move.

Como se puede ver en la Tabla anterior, las funciones que manejan Bytes, se podrían subdividir en algunos tipos, que dependen de los bancos donde se encuentran los bytes, que se conectan a los terminales de entrada y salida de cada función, analizando este punto se concluye que los tipos de funciones son 4:

- El primer tipo es cuando todos sus terminales son registros y están todos en el mismo banco.
- El segundo tipo corresponde, cuando por lo menos uno de los registros que están conectado a los terminales de entrada o de salida, está en un banco diferente del resto de los registros unidos a la función.
- El tercero, es cuando todos los registros están en el mismo banco, pero por lo menos un Terminal de entrada es un número de 0 a 255.
- Finalmente el cuarto tipo corresponde cuando por lo menos un registro que se encuentra conectado a la función está en un banco diferente al resto de los registros y tiene en uno de los terminales de entrada un número de 0 a 255.

Los dos últimos tipos explicados anteriormente, demuestran que las funciones a nivel de bytes pueden operar no solamente con registros o puertos, sino con números. Por ejemplo en la Figura 4.62 se mueve el número 85 decimal (10101010 en binario) al Puerto B cuando RE2 está en 1 lógico.

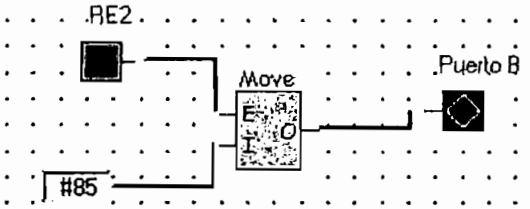
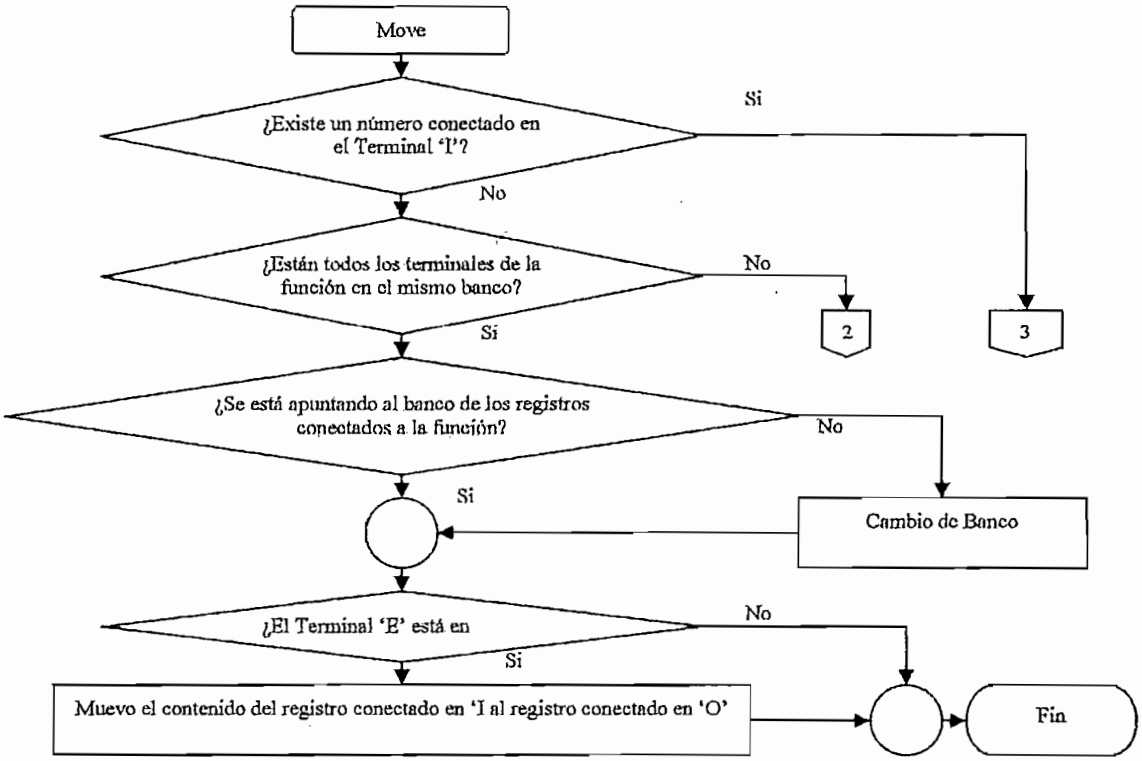
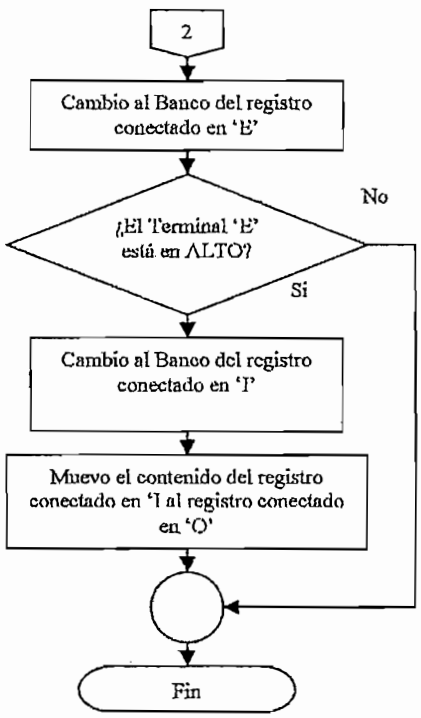


Figura 4.62. Ejemplo de conexión para una Función Move.

En la Figura 4.63 se representa el diagrama de flujo tanto para la simulación, descarga al PIC o generación del código hexadecimal, de la función Move.


Move
¿Existe un número conectado en el Terminal 'I'?
Si
No
¿Están todos los terminales de la función en el mismo banco?
No
Si
2
3
¿Se está apuntando al banco de los registros conectados a la función?
No
Si
Cambio de Banco
¿El Terminal 'E' está en
No
Si
Muevo el contenido del registro conectado en 'I al registro conectado en 'O'
Fin


2
Cambio al Banco del registro conectado en 'E'
¿El Terminal 'E' está en ALTO?
No
Si
Cambio al Banco del registro conectado en 'I'
Muevo el contenido del registro conectado en 'I al registro conectado en 'O'
Fin

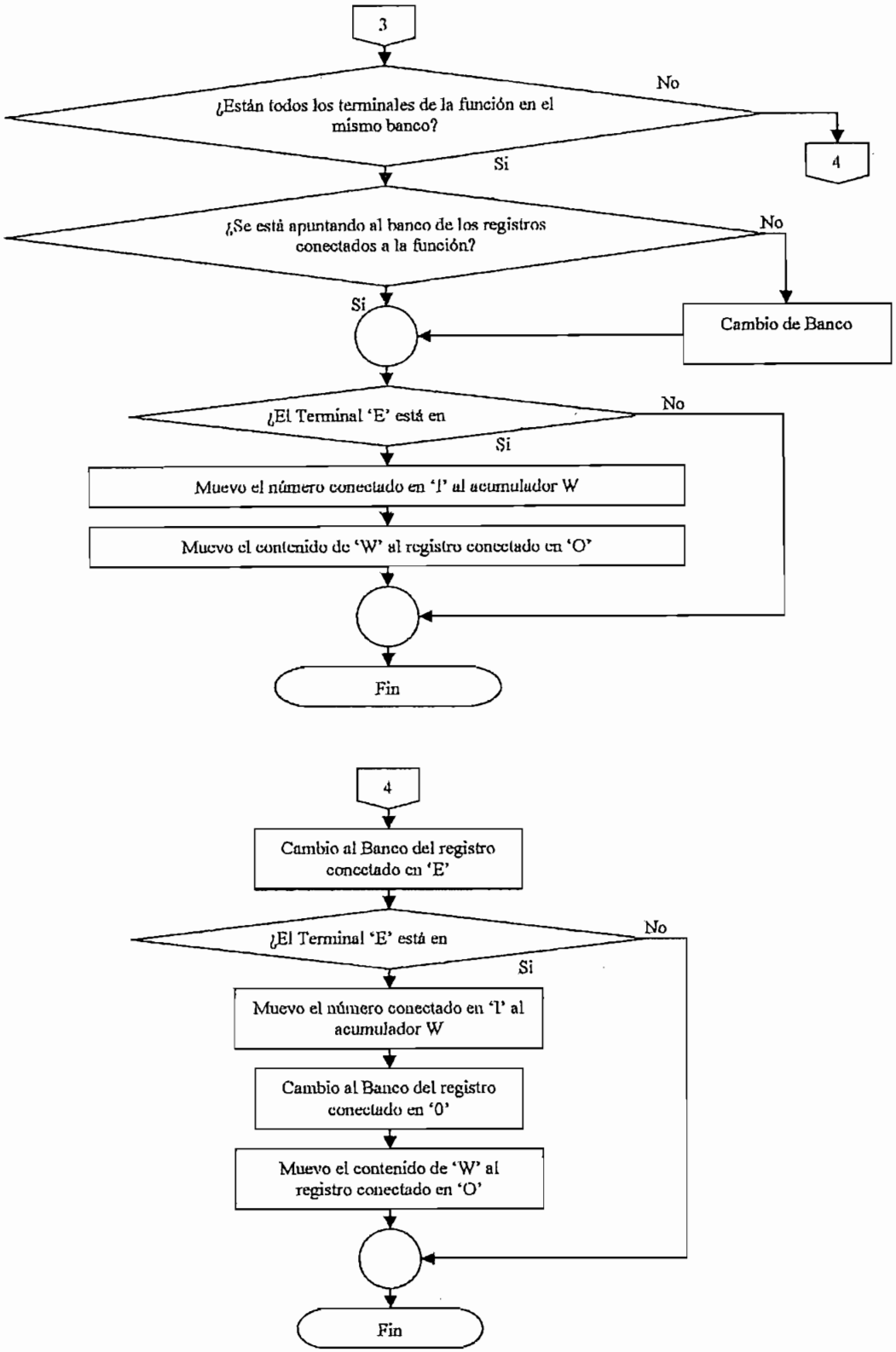
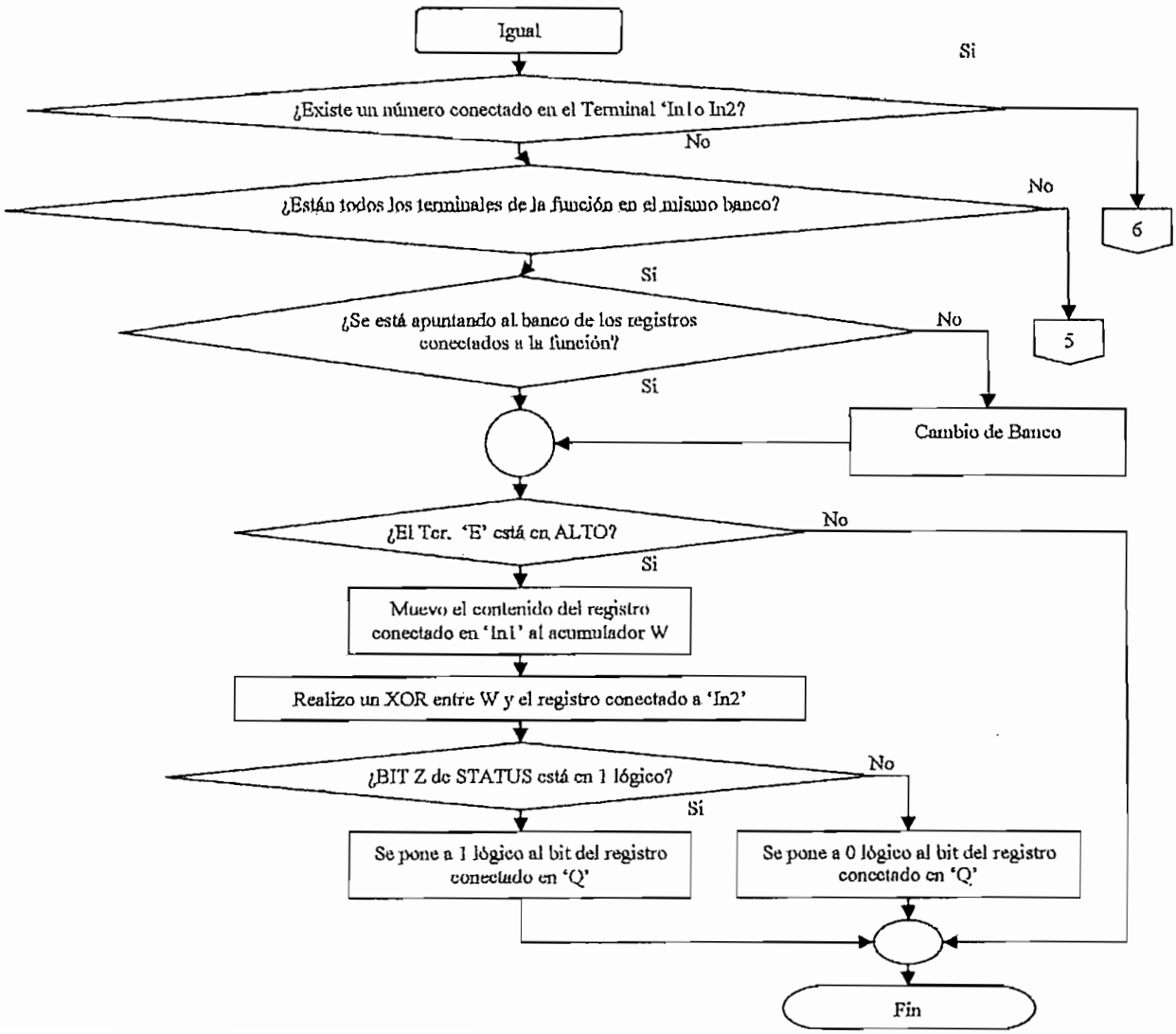
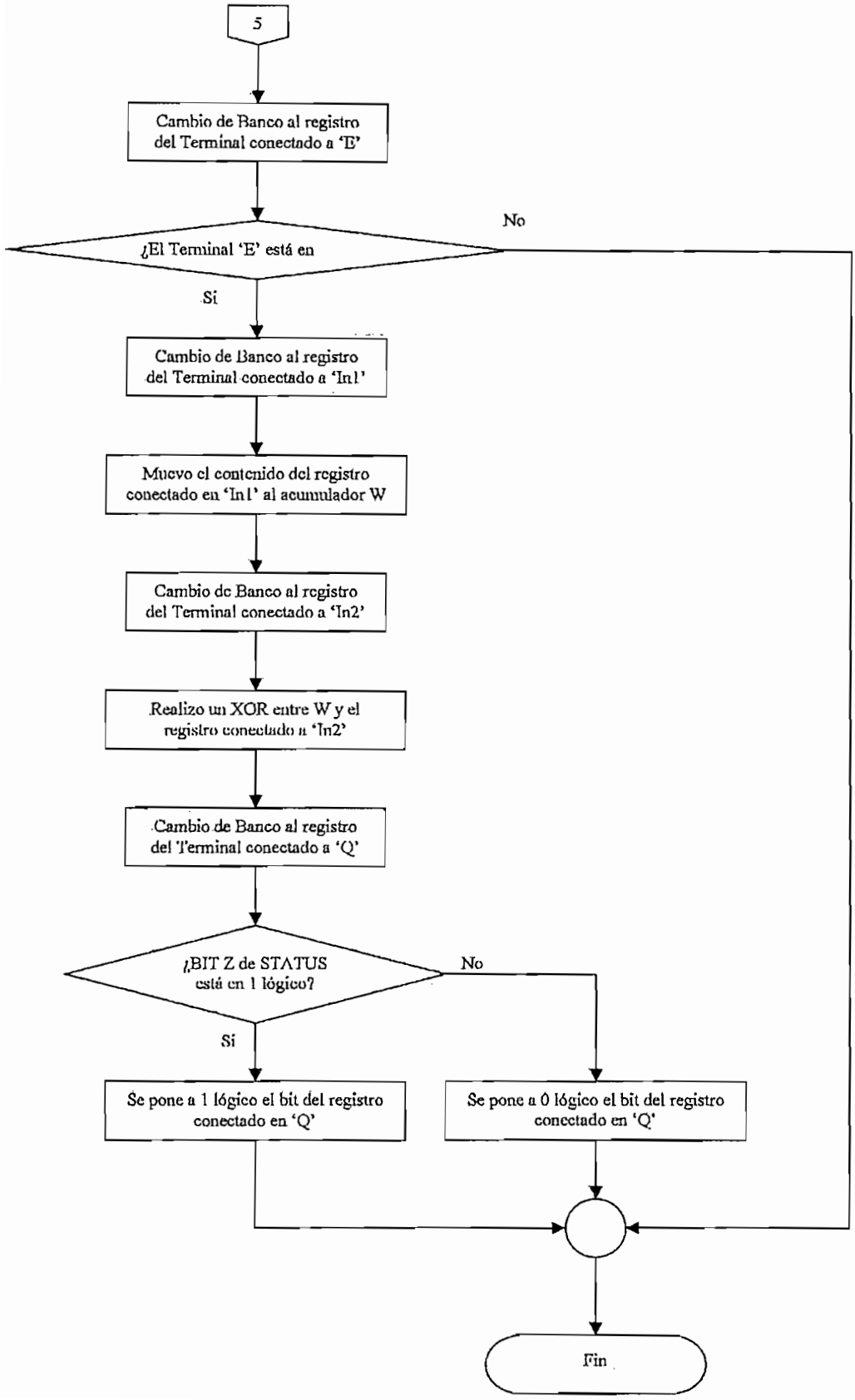


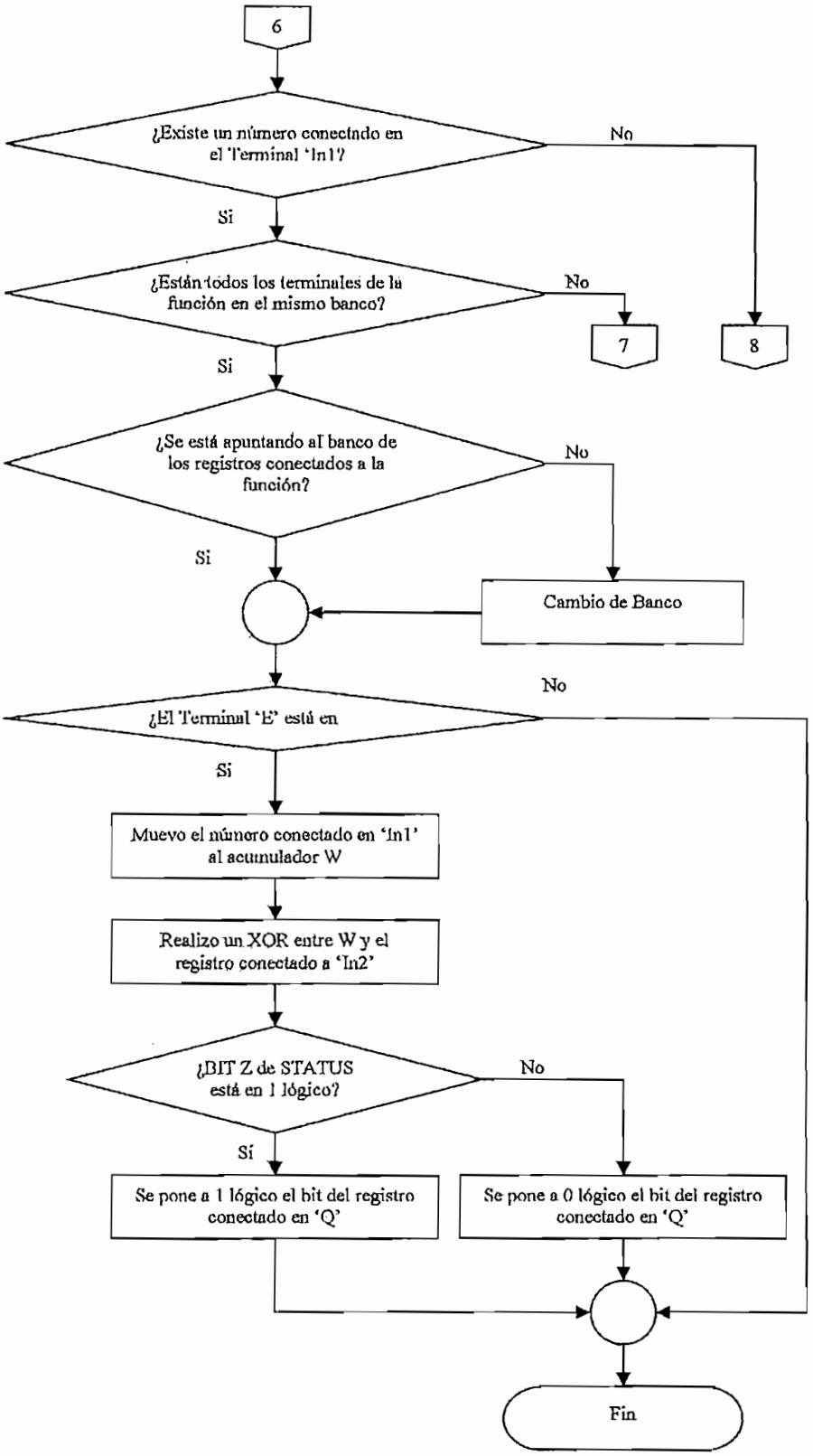
Figura 4.63. Diagrama de Flujo de la Función Move.

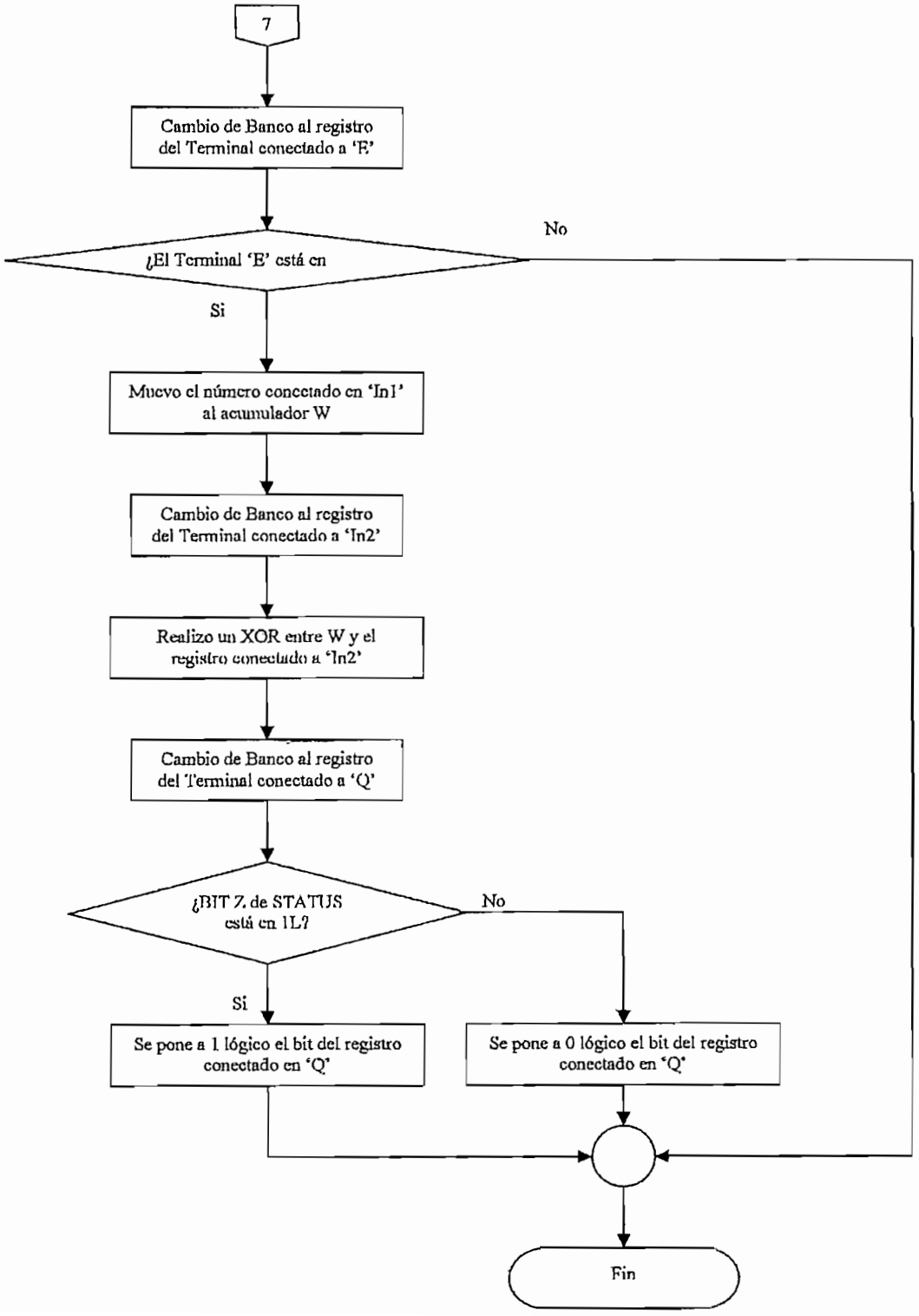
Funciones de comparación

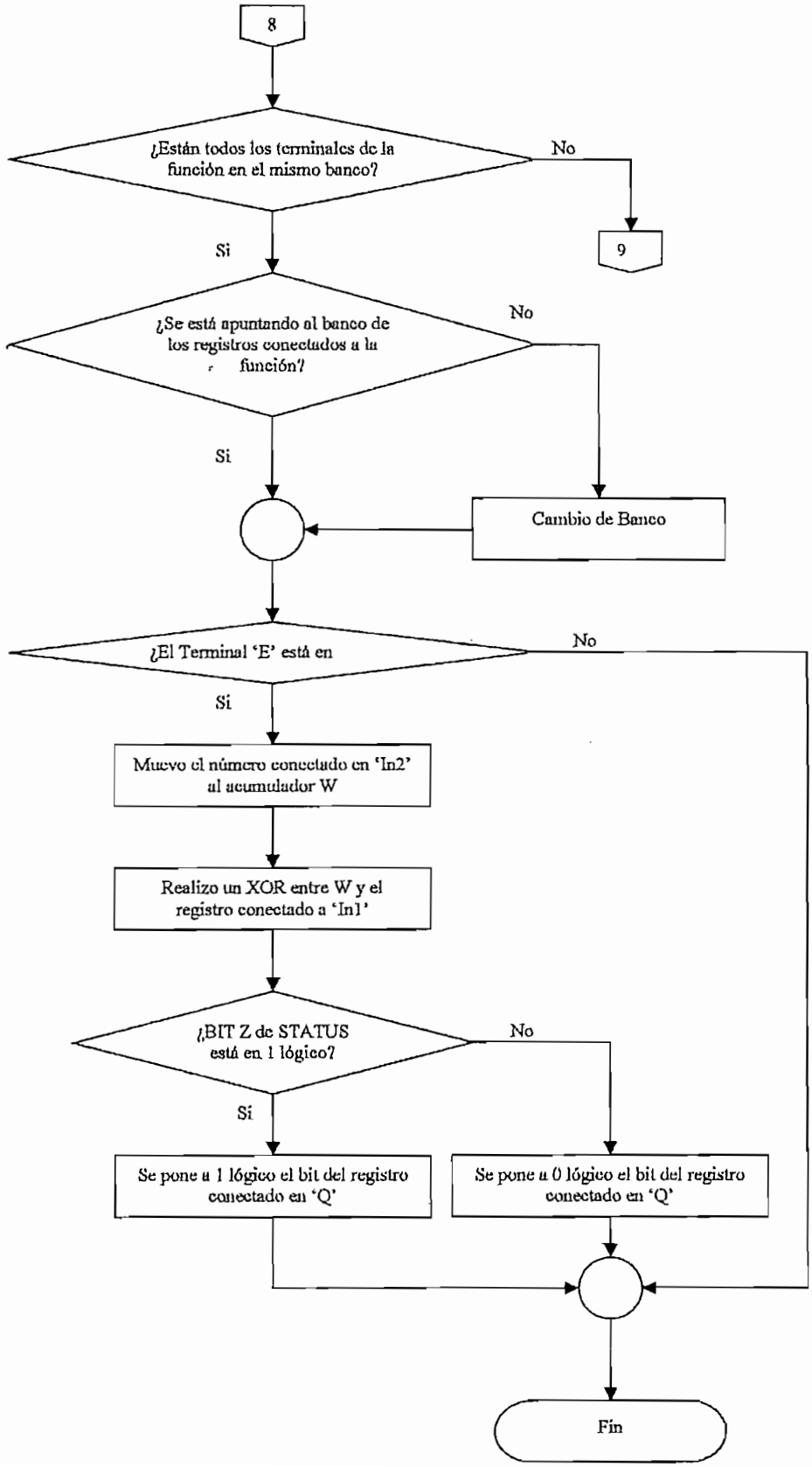
Para el segundo grupo se representará el diagrama de flujo de la función 'Igual', como se explicó anteriormente esta Función permite comparar dos bytes, cuando estos son iguales, el bit del registro conectado al Terminal 'Q' es puesto en 1L. En la Figura 4.64 se representa el diagrama de flujo tanto para la simulación, como para la descarga al PIC o la generación del código hexadecimal. Hay que indicar que dentro de las instrucciones del PIC utilizado, no existe una función de comparación entre dos bytes, lo que se realiza en realidad es un XOR bit a bit entre los dos bytes, en esta operación si los bits son iguales el resultado es '0', esto dentro del PIC representa un 1L en el bit del registro de estado (STATUS, Z), de esta forma es como se realizó la función IGUAL y las demás del grupo.




5
Cambio de Banco al registro del Terminal conectado a 'E'
¿El Terminal 'E' está en
No
Si
Cambio de Banco al registro del Terminal conectado a 'In1'
Muevo el contenido del registro conectado en 'In1' al acumulador W
Cambio de Banco al registro del Terminal conectado a 'In2'
Realizo un XOR entre W y el registro conectado a 'In2'
Cambio de Banco al registro del Terminal conectado a 'Q'
¿BIT Z de STATUS está en 1 lógico?
No
Si
Se pone a 1 lógico el bit del registro conectado en 'Q'
Se pone a 0 lógico el bit del registro conectado en 'Q'
Fin


6
¿Existe un número conectado en el Terminal 'In1'?
No
Si
¿Están todos los terminales de la función en el mismo banco?
No
7
8
Si
¿Se está apuntando al banco de los registros conectados a la función?
No
Si
Cambio de Banco
No
¿El Terminal 'E' está en
Si
Muevo el número conectado en 'In1' al acumulador W
Realizo un XOR entre W y el registro conectado a 'In2'
¿BIT Z de STATUS está en 1 lógico?
No
Si
Se pone a 1 lógico el bit del registro conectado en 'Q'
Se pone a 0 lógico el bit del registro conectado en 'Q'
Fin


7
Cambio de Banco al registro del Terminal conectado a 'E'
¿El Terminal 'E' está en
No
Si
Muevo el número conectado en 'In1' al acumulador W
Cambio de Banco al registro del Terminal conectado a 'In2'
Realizo un XOR entre W y el registro conectado a 'In2'
Cambio de Banco al registro del Terminal conectado a 'Q'
¿BIT Z de STATUS está en 1L?
No
Si
Se pone a 1 lógico el bit del registro conectado en 'Q'
Se pone a 0 lógico el bit del registro conectado en 'Q'
Fin


8
¿Están todos los terminales de la función en el mismo banco?
No
9
Si
¿Se está apuntando al banco de los registros conectados a la función?
No
Si
Cambio de Banco
¿El Terminal 'E' está en
No
Si
Muevo el número conectado en 'In2' al acumulador W
Realizo un XOR entre W y el registro conectado a 'In1'
¿BIT Z de STATUS está en 1 lógico?
No
Si
Se pone a 1 lógico el bit del registro conectado en 'Q'
Se pone a 0 lógico el bit del registro conectado en 'Q'
Fin

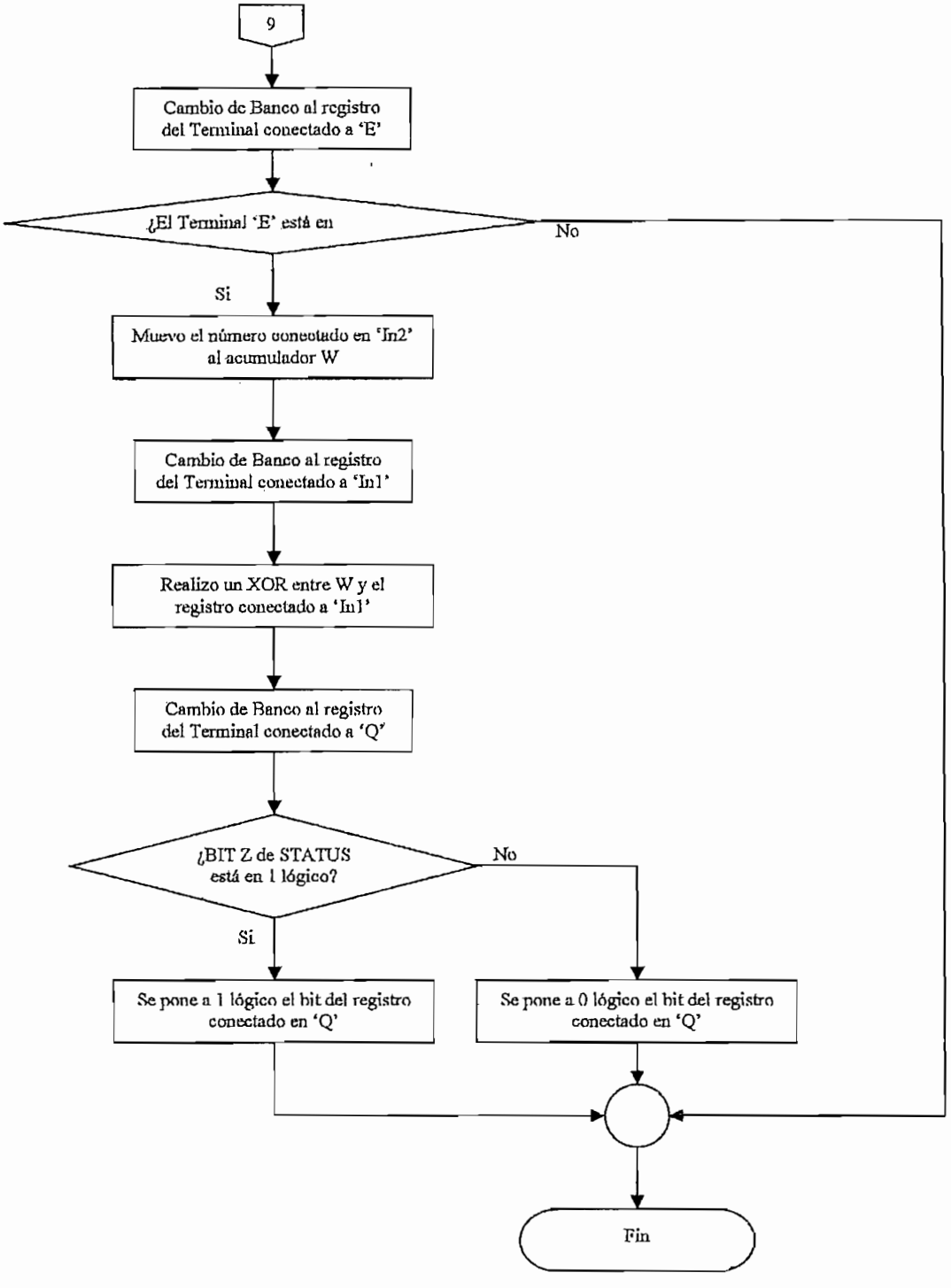
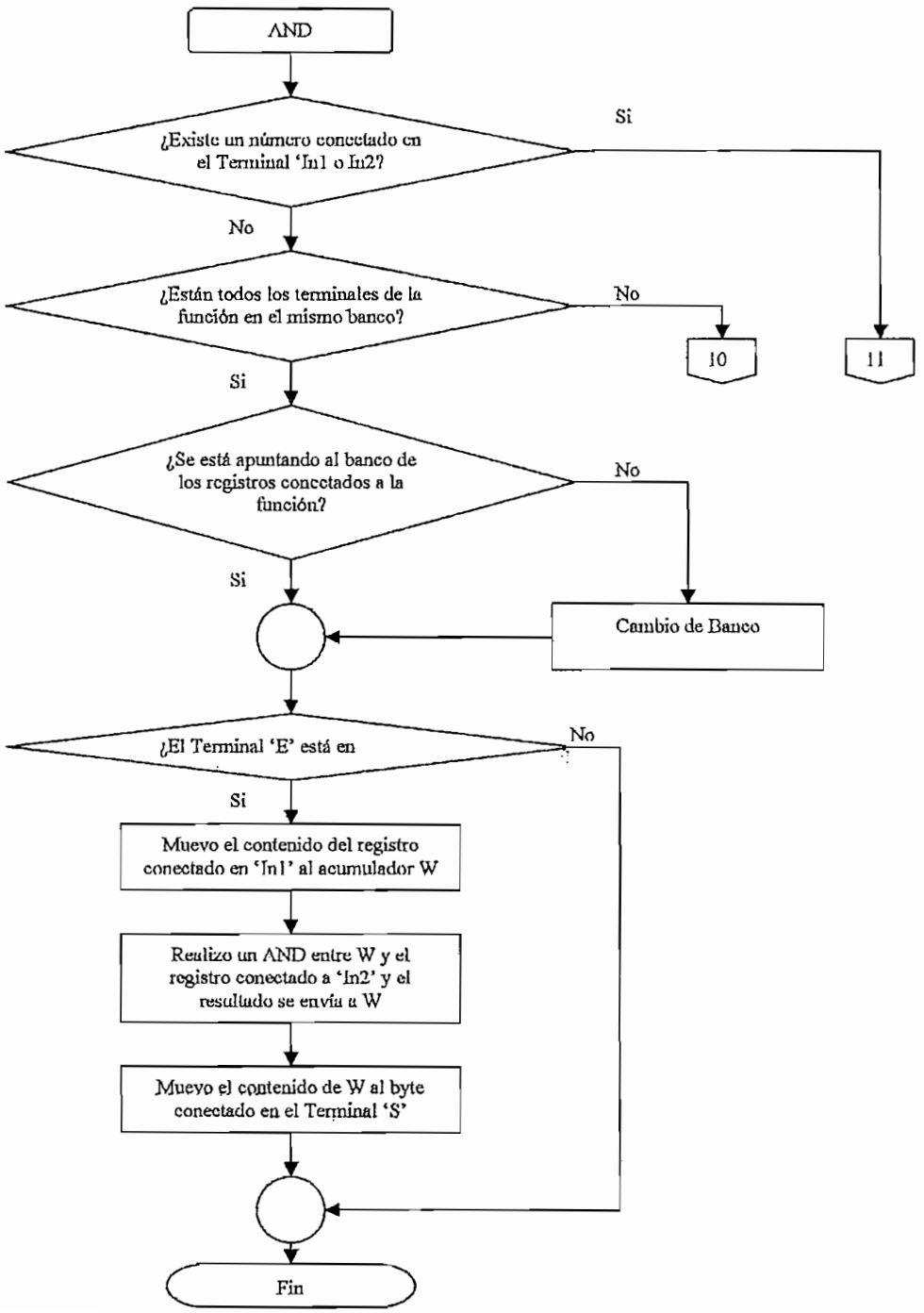


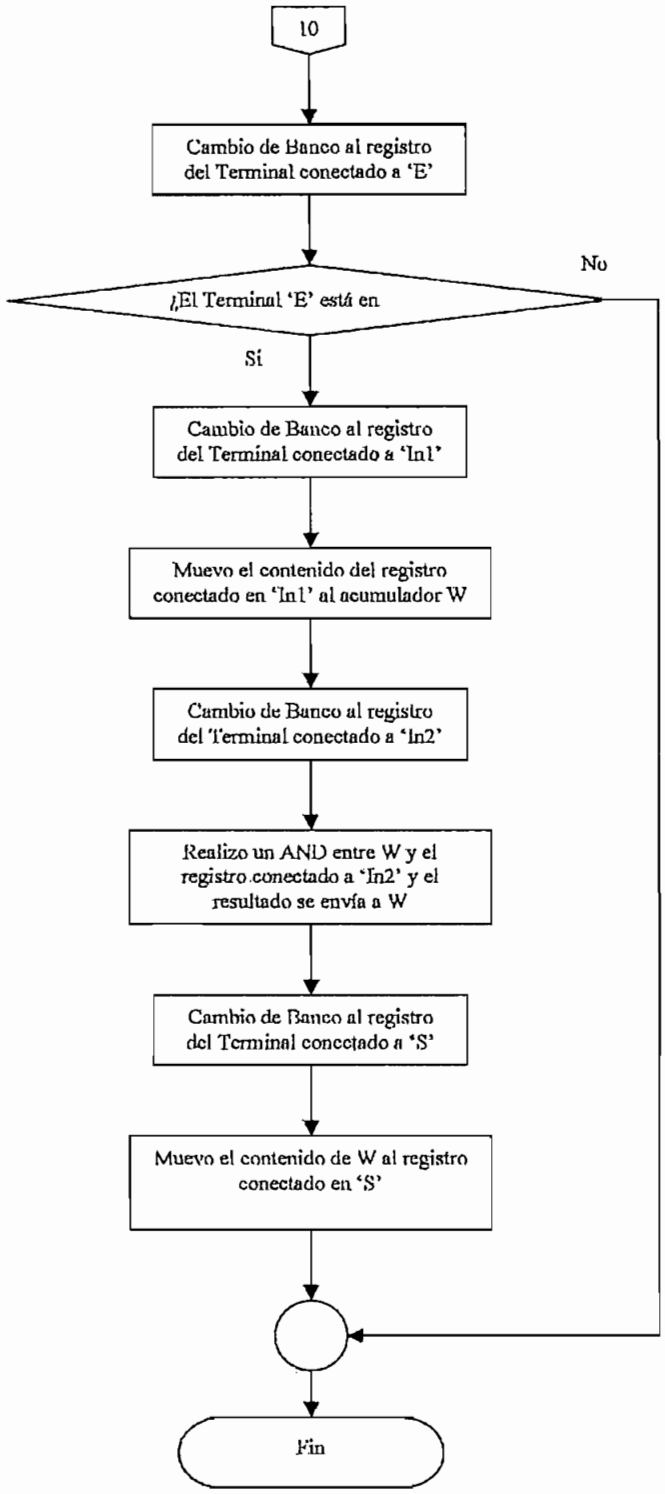
Figura 4.64. Diagrama de Flujo de la Función Igual.

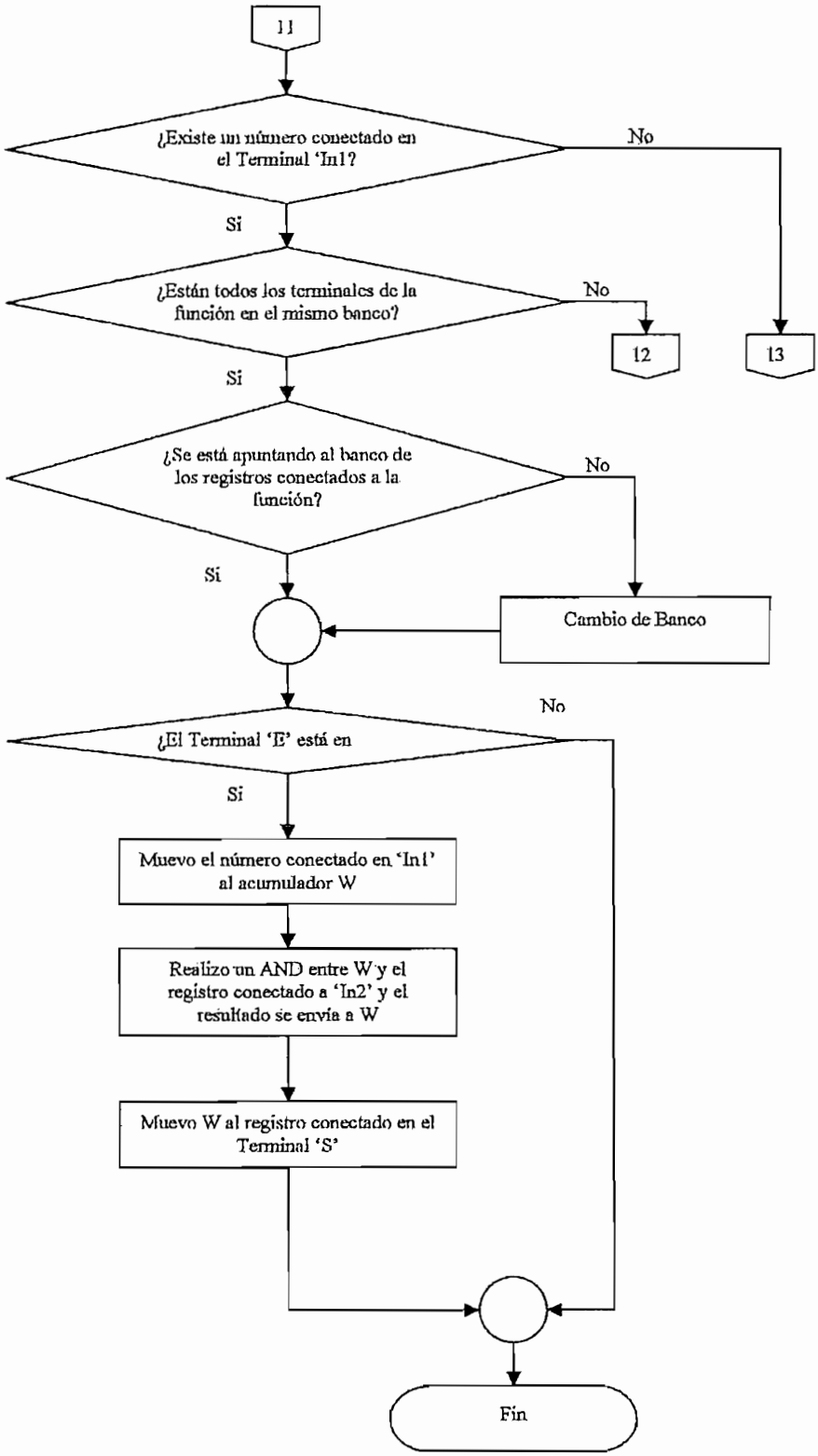
Dentro de las instrucciones que posee el microcontrolador no existen instrucciones para comprobar si un byte es mayor que otro, por lo que para la función 'Mayor igual' y 'Mayor' se realiza una resta entre los dos bytes a comparar, luego se comprueba el estado del bit carry del registro Status.

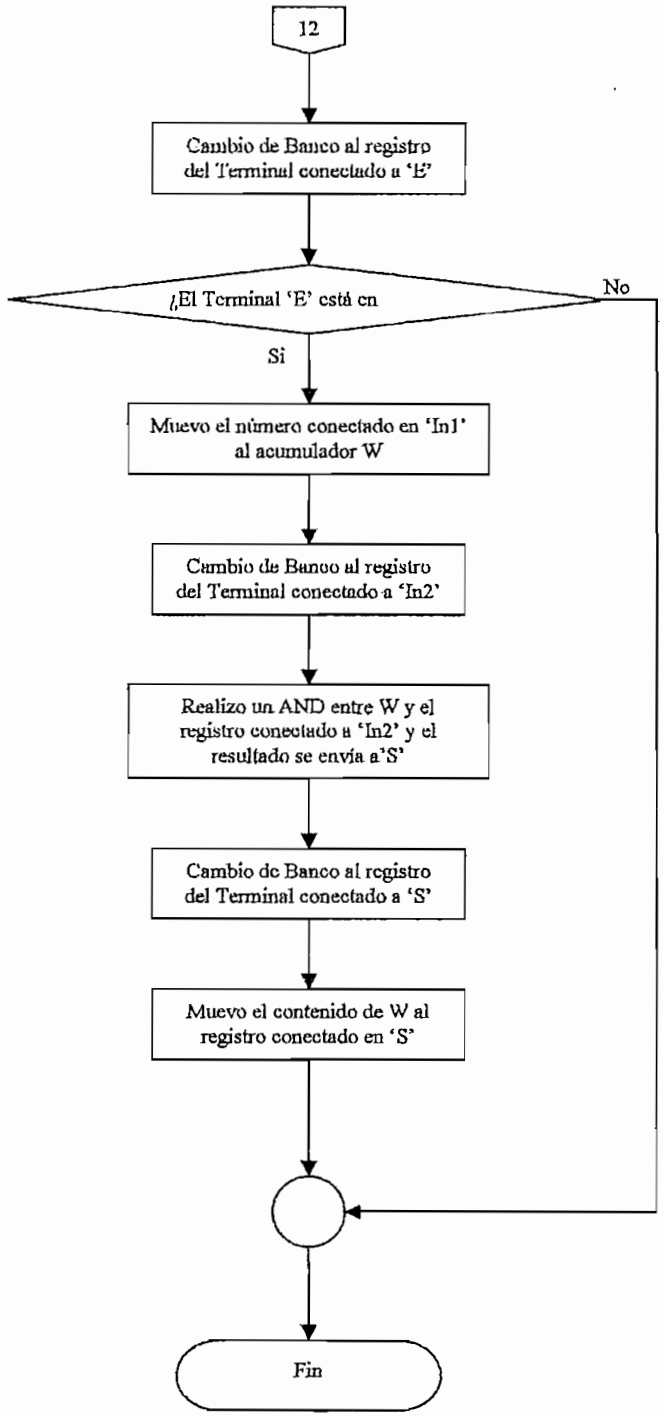
Funciones de Operación Binaria para bytes

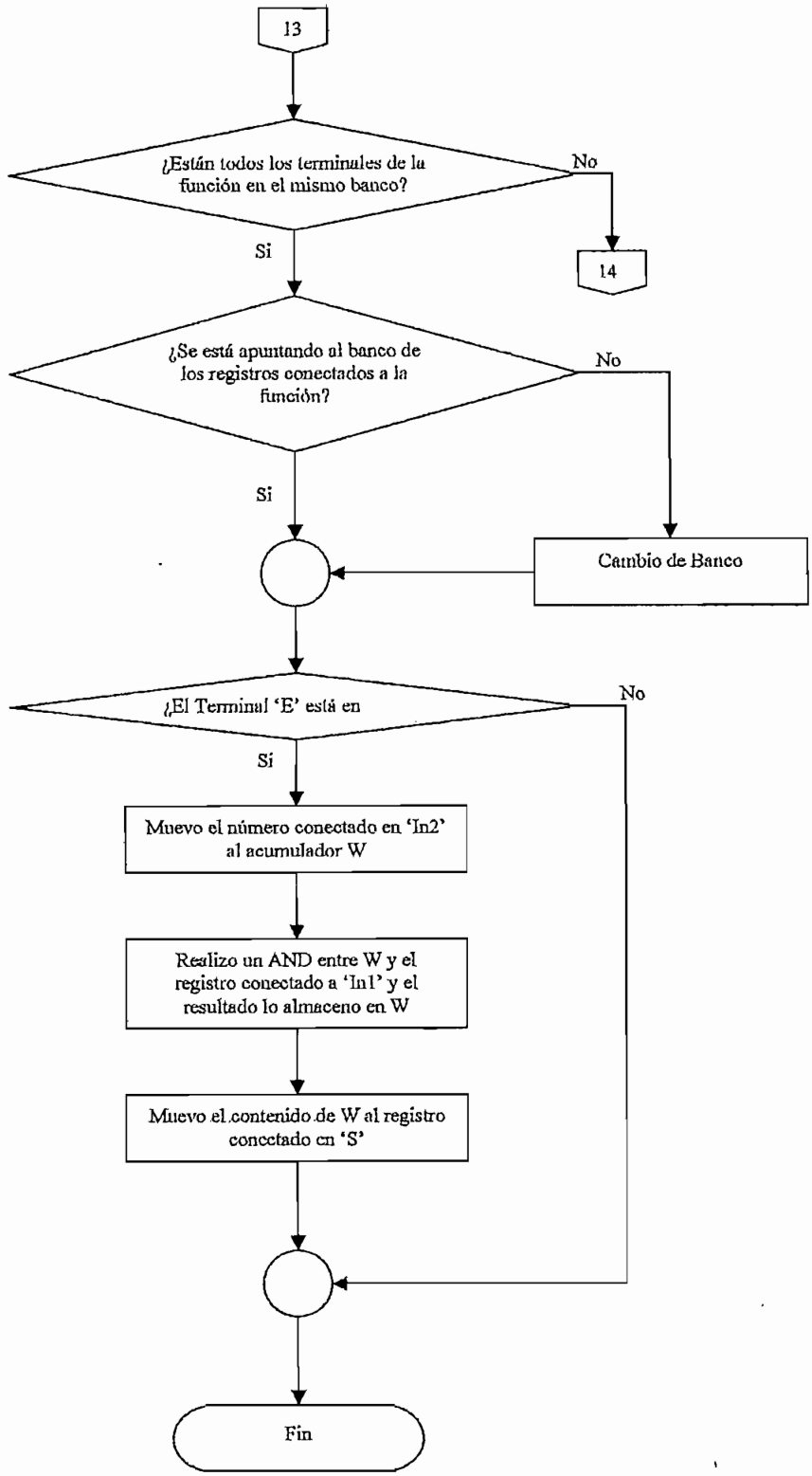
Para el tercer grupo se representará el diagrama de flujo de la función 'And', esta permite realizar un AND lógico bit a bit entre dos bytes y el resultado enviar a otro byte. En la Figura 4.65 se representa el diagrama de flujo tanto para la simulación, descarga al PIC o generación del código hexadecimal.




10
Cambio de Banco al registro del Terminal conectado a 'E'
¿El Terminal 'E' está en
No
Si
Cambio de Banco al registro del Terminal conectado a 'In1'
Muevo el contenido del registro conectado en 'In1' al acumulador W
Cambio de Banco al registro del Terminal conectado a 'In2'
Realizo un AND entre W y el registro conectado a 'In2' y el resultado se envía a W
Cambio de Banco al registro del Terminal conectado a 'S'
Muevo el contenido de W al registro conectado en 'S'
Fin


I1
¿Existe un número conectado en el Terminal 'In1'?
No
Si
¿Están todos los terminales de la función en el mismo banco?
No
I2
I3
Si
¿Se está apuntando al banco de los registros conectados a la función?
No
Si
Cambio de Banco
No
¿El Terminal 'E' está en
Si
Muevo el número conectado en 'In1' al acumulador W
Realizo un AND entre W y el registro conectado a 'In2' y el resultado se envía a W
Muevo W al registro conectado en el Terminal 'S'
Fin


12
Cambio de Banco al registro del Terminal conectado a 'E'
¿El Terminal 'E' está en
No
Si
Muevo el número conectado en 'In1' al acumulador W
Cambio de Banco al registro del Terminal conectado a 'In2'
Realizo un AND entre W y el registro conectado a 'In2' y el resultado se envía a 'S'
Cambio de Banco al registro del Terminal conectado a 'S'
Muevo el contenido de W al registro conectado en 'S'
Fin


13
¿Están todos los terminales de la función en el mismo banco?
No
Si
14
¿Se está apuntando al banco de los registros conectados a la función?
No
Si
Cambio de Banco
¿El Terminal 'E' está en
No
Si
Muevo el número conectado en 'In2' al acumulador W
Realizo un AND entre W y el registro conectado a 'In1' y el resultado lo almaceno en W
Muevo el contenido de W al registro conectado en 'S'
Fin

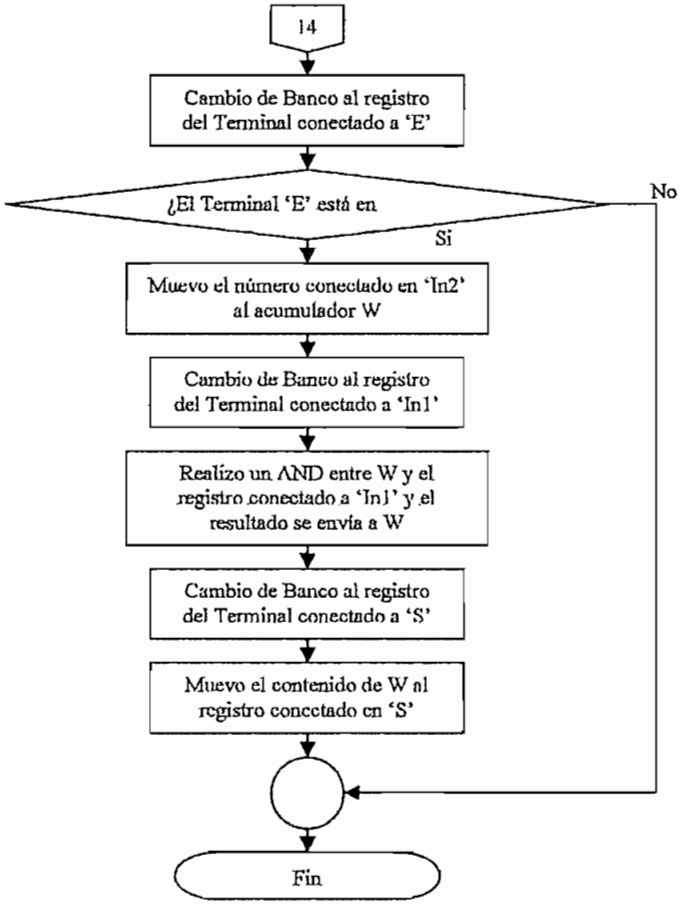


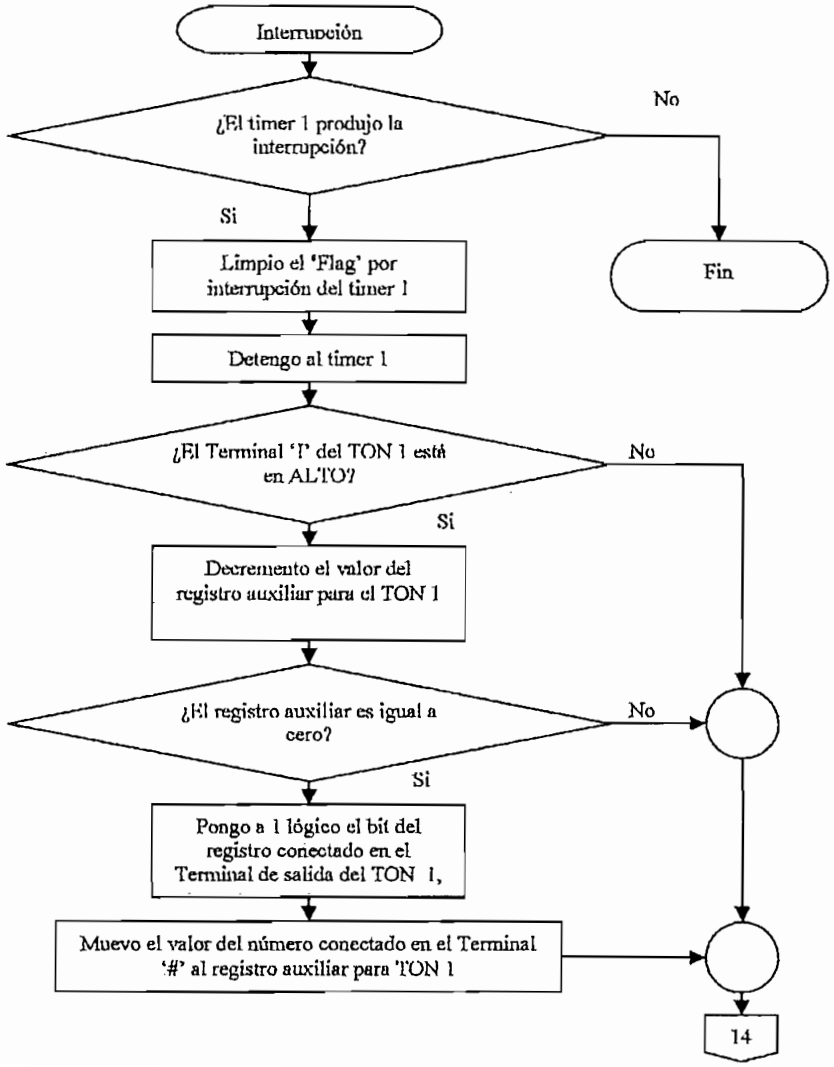
Figura 4.65. Diagrama de Flujo de la Función AND entre dos bytes.

Funciones de Temporización

Para el cuarto grupo de Funciones se tiene los 'Temporizadores ON y OFF'. Los temporizadores internos que posee el microcontrolador utilizado en este proyecto son 3, de estos uno funciona conjuntamente con el perro guardián (Watch dog timer) del sistema y el otro es parte funcional de los módulos internos PWM. El único temporizador disponible es el Timer 1, y a partir de este se consiguió crear varios temporizadores tipo ON y tipo OFF, esto se consigue con ayuda de 2 registros auxiliares de la memoria RAM. Los registros que se utilizaron van desde de posición 480 decimal hasta la posición 487 decimal del banco 3 para los temporizadores TON y desde la posición 488 hasta la 495 decimal del mismo banco para los temporizadores tipo OFF. Como se puede ver, únicamente pueden existir 8 temporizadores tipo ON y 8 tipo OFF, los 16 registros auxiliares no podrán ser utilizados por el usuario.

El timer 1 es un temporizador de 16 bits con un ciclo de máquina de 1 μs, que se desborda produciendo una interrupción en el microcontrolador, cada 65536μs ó 65,536ms por 2, ya que se configuró a este con un prescaler de 2. Los registros auxiliares, que ya se explicaron anteriormente, permiten aumentar este tiempo hasta 255 veces. Es decir que el valor que se carga en el Terminal '#' de los temporizadores, es el valor que va hacia los registros auxiliares, los cuales se decrementan cada vez que el timer 1 se desborda, cuando este registro llega a cero el temporizador (ON u OFF) actúa sobre su salida 'Q'.

A continuación se observa en la Figura 4.66, el diagrama de flujo para los temporizadores tipo ON cuando se produce la interrupción en el microcontrolador o en la simulación.



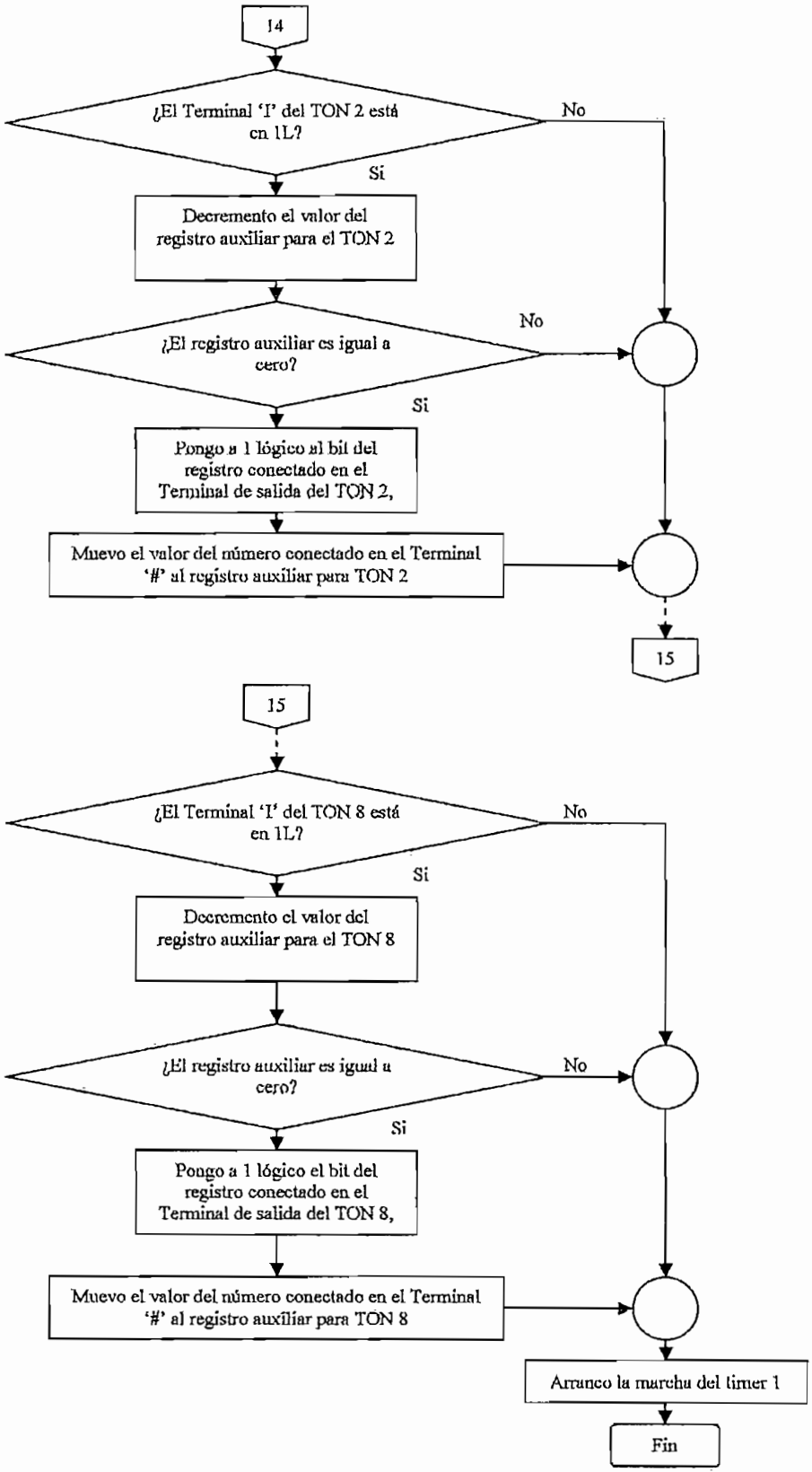
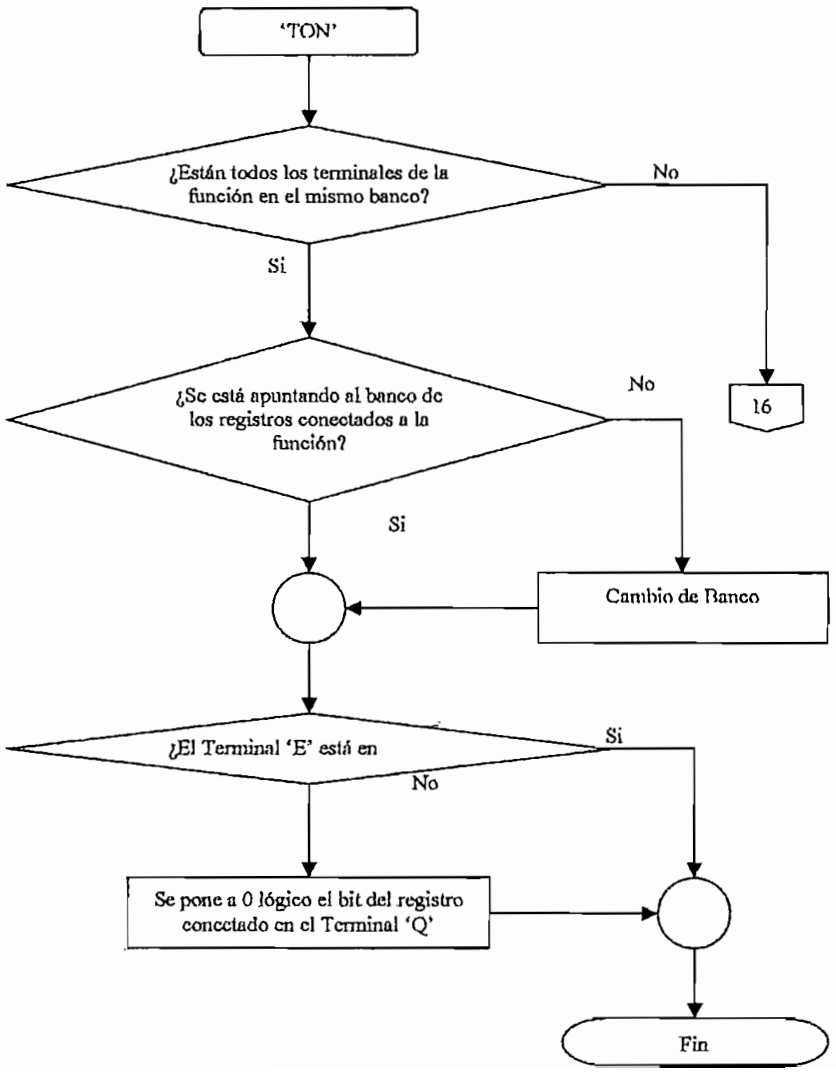


Figura 4.66. Diagrama de Flujo de la Interrupción que se produce con el desborde del Timer TON.

Nota: El diagrama de flujo anterior es para el caso en que se utiliza los 8 temporizadores tipo ON. Si dicho número es menor el diagrama de flujo avanza solamente hasta el número de temporizador usado.

El temporizador tipo ON dentro del funcionamiento normal del programa del PLC, lee constantemente el Terminal de entrada 'E' y en caso de que esté en 0L pone a 0L al bit del registro conectado en el Terminal de salida 'Q'. El diagrama de flujo de la Figura 4.67 muestra el diagrama de flujo de un temporizador ON.



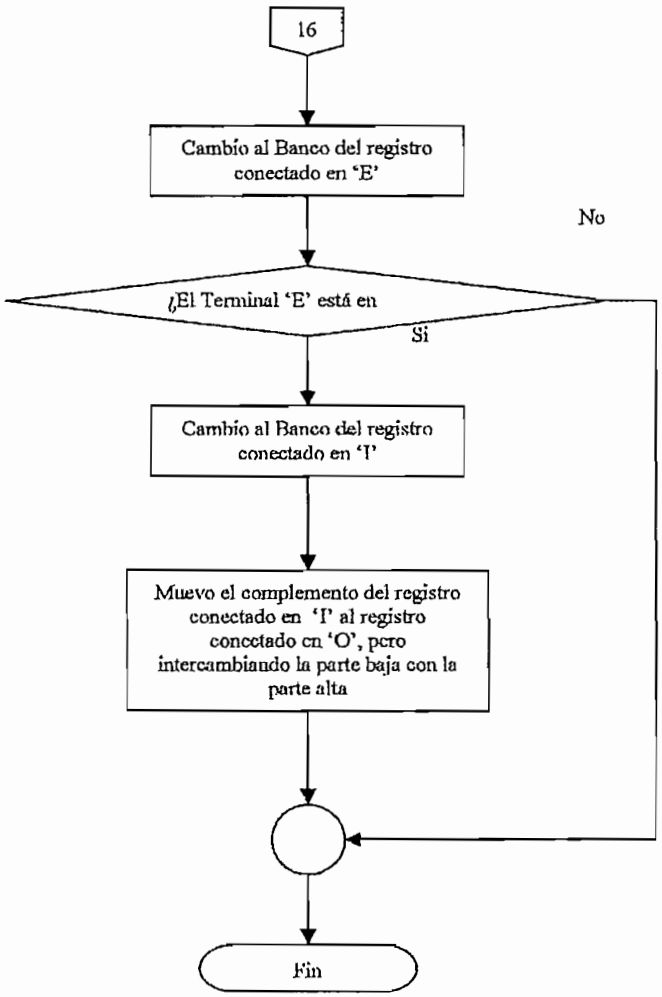
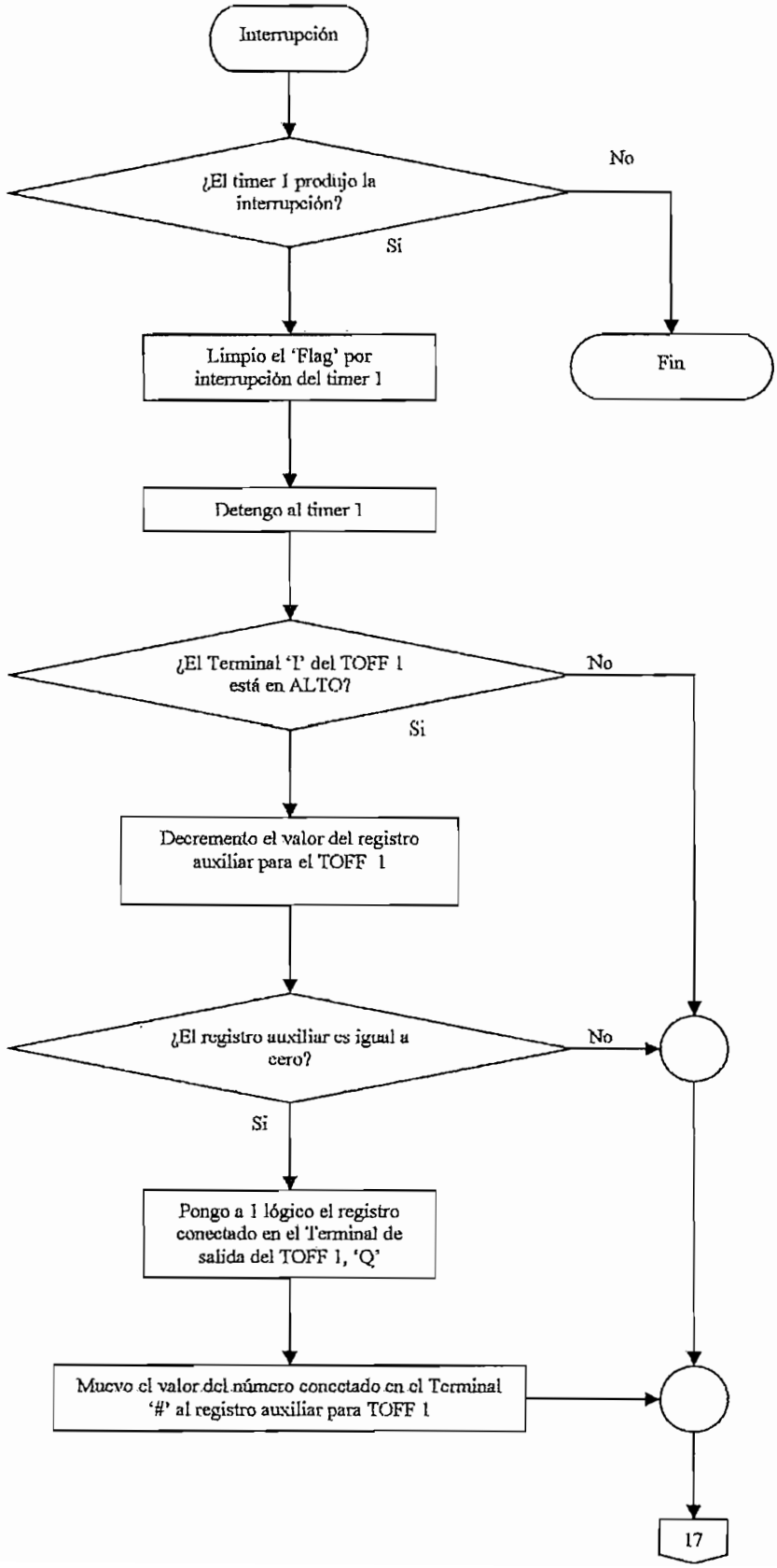


Figura 4.67. Diagrama de Flujo del Temporizador ON.

Los temporizadores tipo OFF tienen un comportamiento similar a los tipo ON, con la diferencia de que estos empiezan la cuenta del tiempo, desde que el Terminal de entrada 'E' esté en 0L.

En la Figura 4.68, se muestra el diagrama de flujo de la interrupción que se produce con el desborde de un Temporizador tipo OFF.


Interrupción
¿El timer 1 produjo la interrupción?
No
Si
Limpio el 'Flag' por interrupción del timer 1
Fin
Detengo al timer 1
¿El Terminal 'I' del TOFF 1 está en ALTO?
No
Si
Decremento el valor del registro auxiliar para el TOFF 1
¿El registro auxiliar es igual a cero?
No
Si
Pongo a 1 lógico el registro conectado en el Terminal de salida del TOFF 1, 'Q'
Muevo el valor del número conectado en el Terminal '#' al registro auxiliar para TOFF 1
17

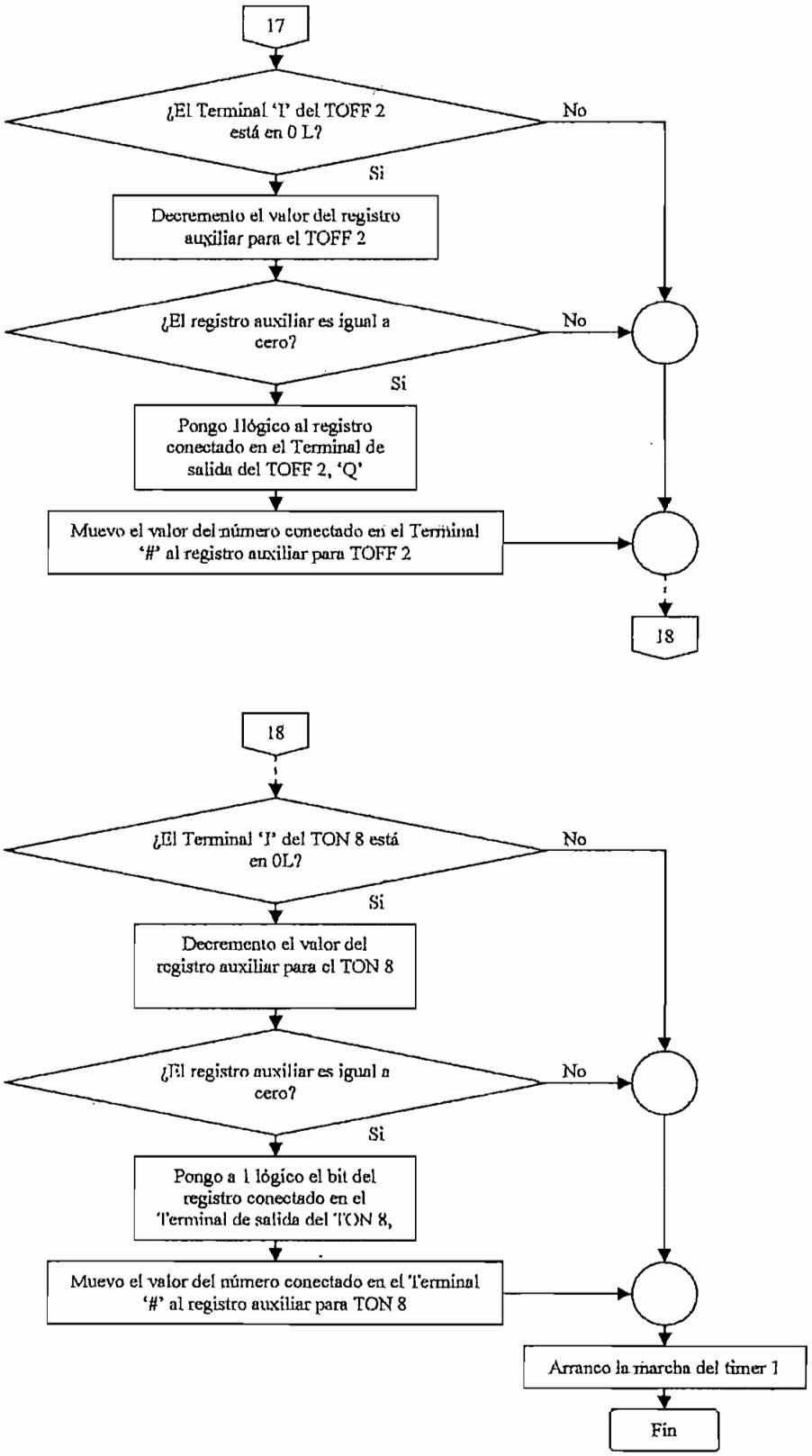
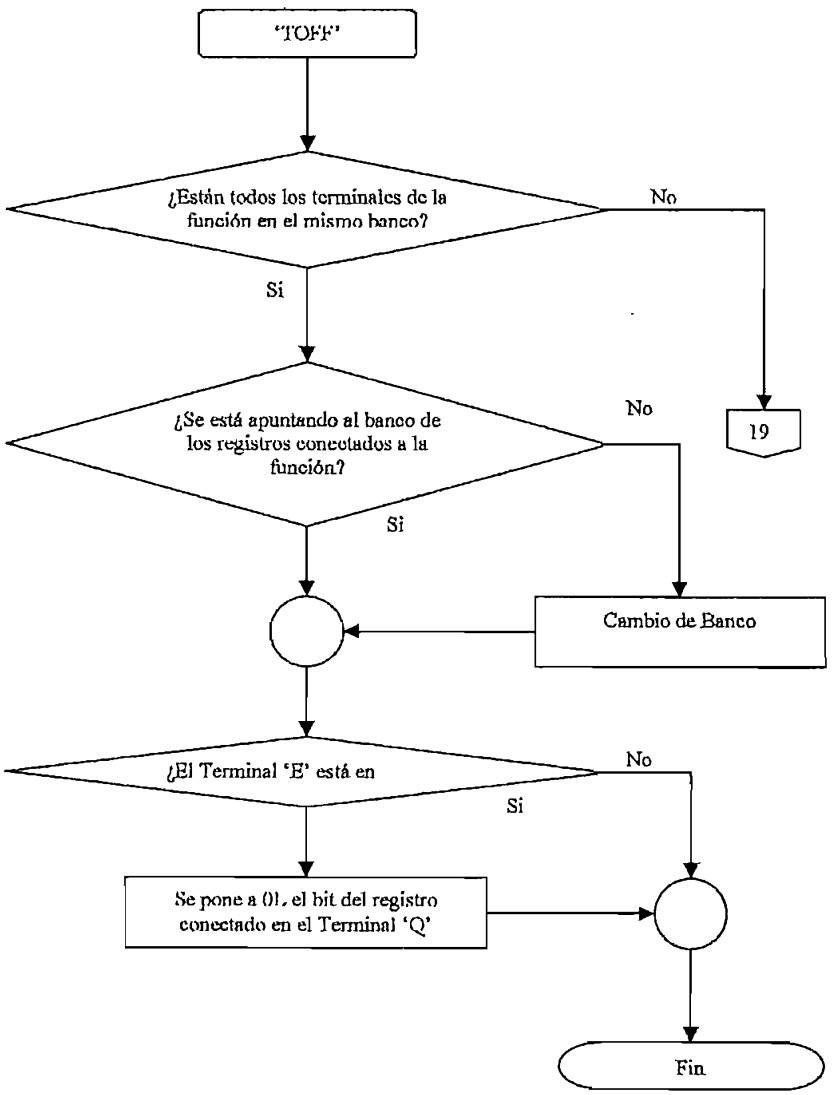


Figura 4.68. Diagrama de Flujo de la Interrupción que se produce con el desborde del Timer TOFF.

De igual manera que el diagrama de flujo de la interrupción del temporizador tipo ON, si se utilizan los 8 temporizadores TOFF disponible, el diagrama de flujo se extiende demasiado comparado con uno de menor número de timers.

El diagrama de flujo para este temporizador dentro del programa principal del PLC se muestra en la Figura 4.69.



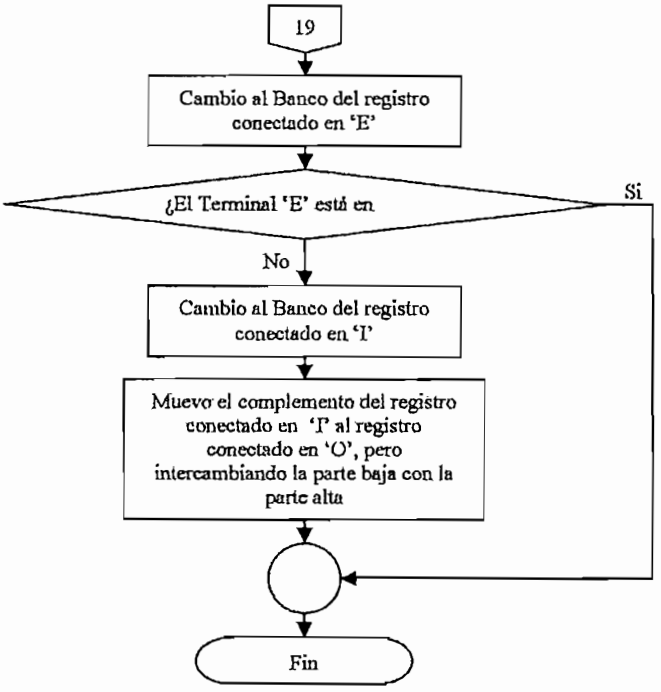
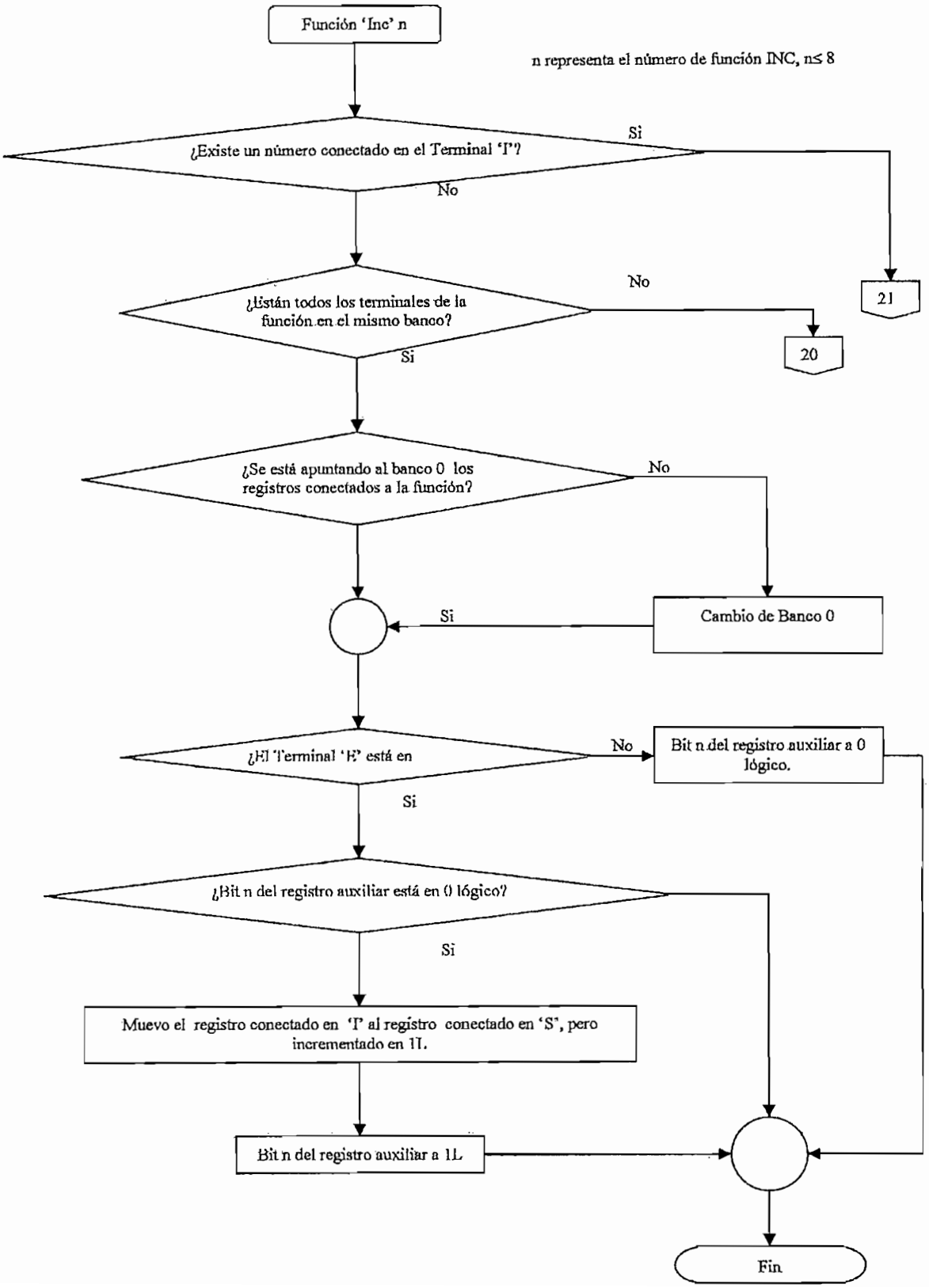
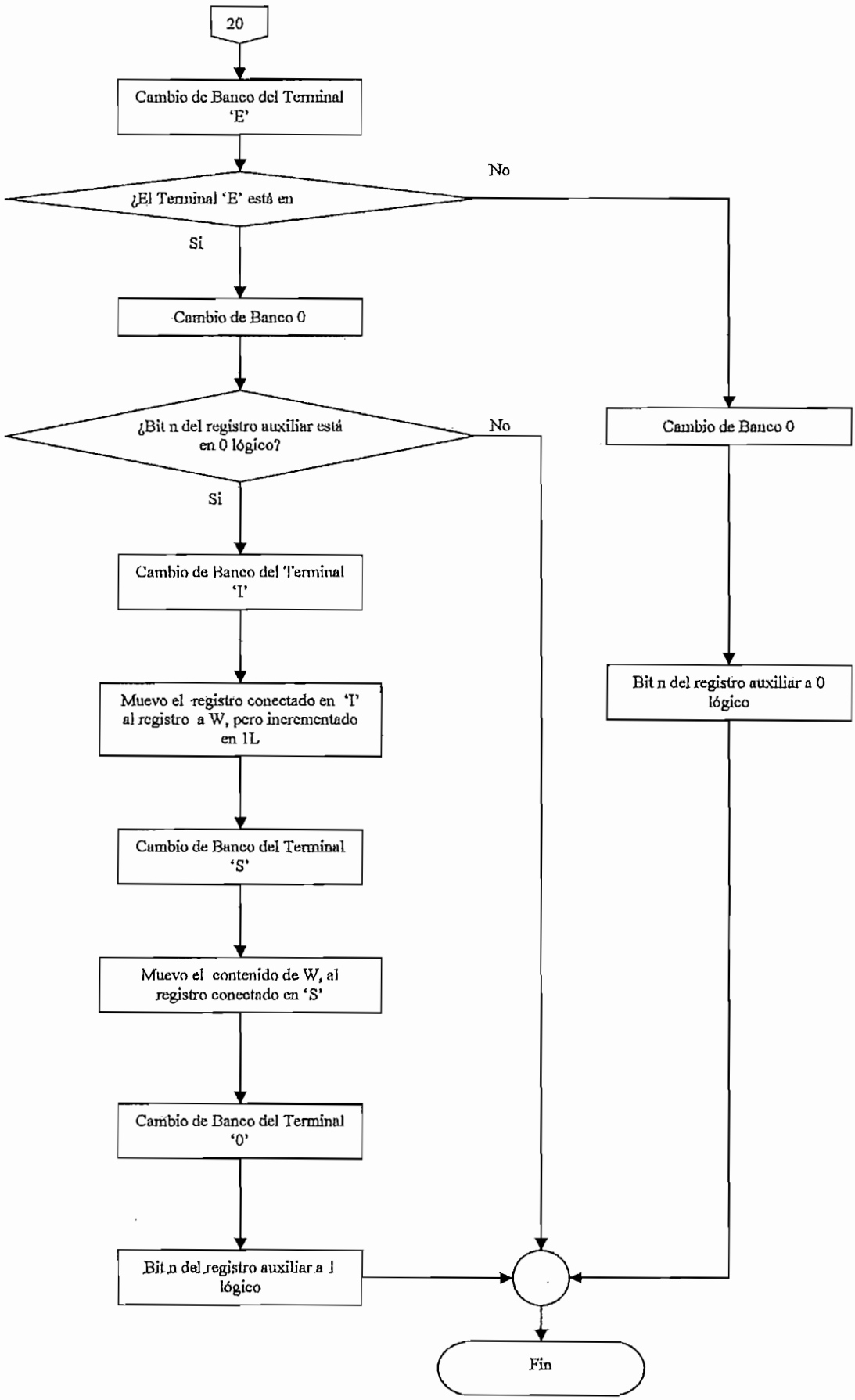


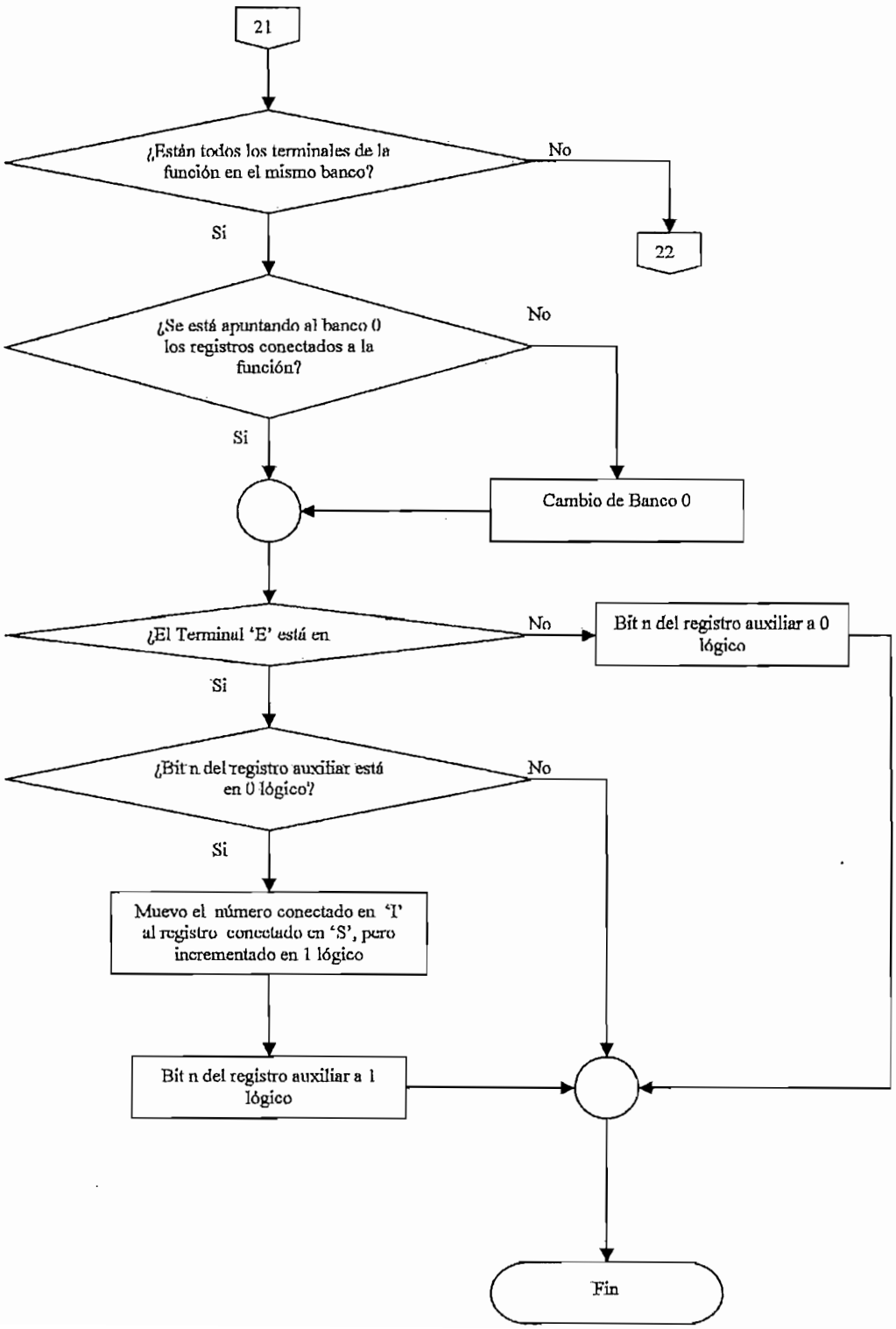
Figura 4.69. Diagrama de Flujo del Temporizador OFF.

Funciones de Alteración

En el quinto grupo de funciones cuyas características son similares, se encuentran las funciones 'Incremento' y 'Decremento'. Estas permiten tomar el contenido de un byte y pasarlo a otro, pero incrementado o decrementado en 1, respectivamente. Como ya se explicó anteriormente su funcionamiento no es por estado, sino por flanco positivo en el Terminal '↑'. Para poder detectar el flanco de subida o de bajada de un bit, se necesita la ayuda, de un bit un registro auxiliar que permita reconocer esta acción. Es por esto que se asignó la posición 7Eh de la memoria RAM en el banco 0, para auxiliar de las funciones 'Incremento' y la posición 7Fh del mismo banco como auxiliar de las funciones 'Decremento'; es decir, que solamente pueden existir un máximo de 8 funciones 'Inc' y 8 funciones 'Dec'. El diagrama de flujo para la función incremento se representa a continuación en la Figura 4.70.


Función 'Inc' n
n representa el número de función INC, n≤ 8
¿Existe un número conectado en el Terminal 'I'?
Si
No
21
¿Están todos los terminales de la función en el mismo banco?
No
Si
20
¿Se está apuntando al banco 0 los registros conectados a la función?
No
Si
Cambio de Banco 0
¿El Terminal 'E' está en
No
Bit n del registro auxiliar a 0 lógico.
Si
¿Bit n del registro auxiliar está en 0 lógico?
Si
Muevo el registro conectado en 'I' al registro conectado en 'S', pero incrementado en 1L.
Bit n del registro auxiliar a 1L
Fin


20
Cambio de Banco del Terminal 'E'
¿El Terminal 'E' está en
No
Si
Cambio de Banco 0
¿Bit n del registro auxiliar está en 0 lógico?
No
Si
Cambio de Banco del Terminal 'I'
Muevo el registro conectado en 'I' al registro a W, pero incrementado en 1L
Cambio de Banco del Terminal 'S'
Muevo el contenido de W, al registro conectado en 'S'
Cambio de Banco del Terminal '0'
Bit n del registro auxiliar a 1 lógico
Cambio de Banco 0
Bit n del registro auxiliar a 0 lógico
Fin


21
¿Están todos los terminales de la función en el mismo banco?
No
Si
22
¿Se está apuntando al banco 0 los registros conectados a la función?
No
Si
Cambio de Banco 0
¿El Terminal 'E' está en
No
Bit n del registro auxiliar a 0 lógico
Si
¿Bit n del registro auxiliar está en 0 lógico?
No
Si
Muevo el número conectado en 'I' al registro conectado en 'S', pero incrementado en 1 lógico
Bit n del registro auxiliar a 1 lógico
Fin

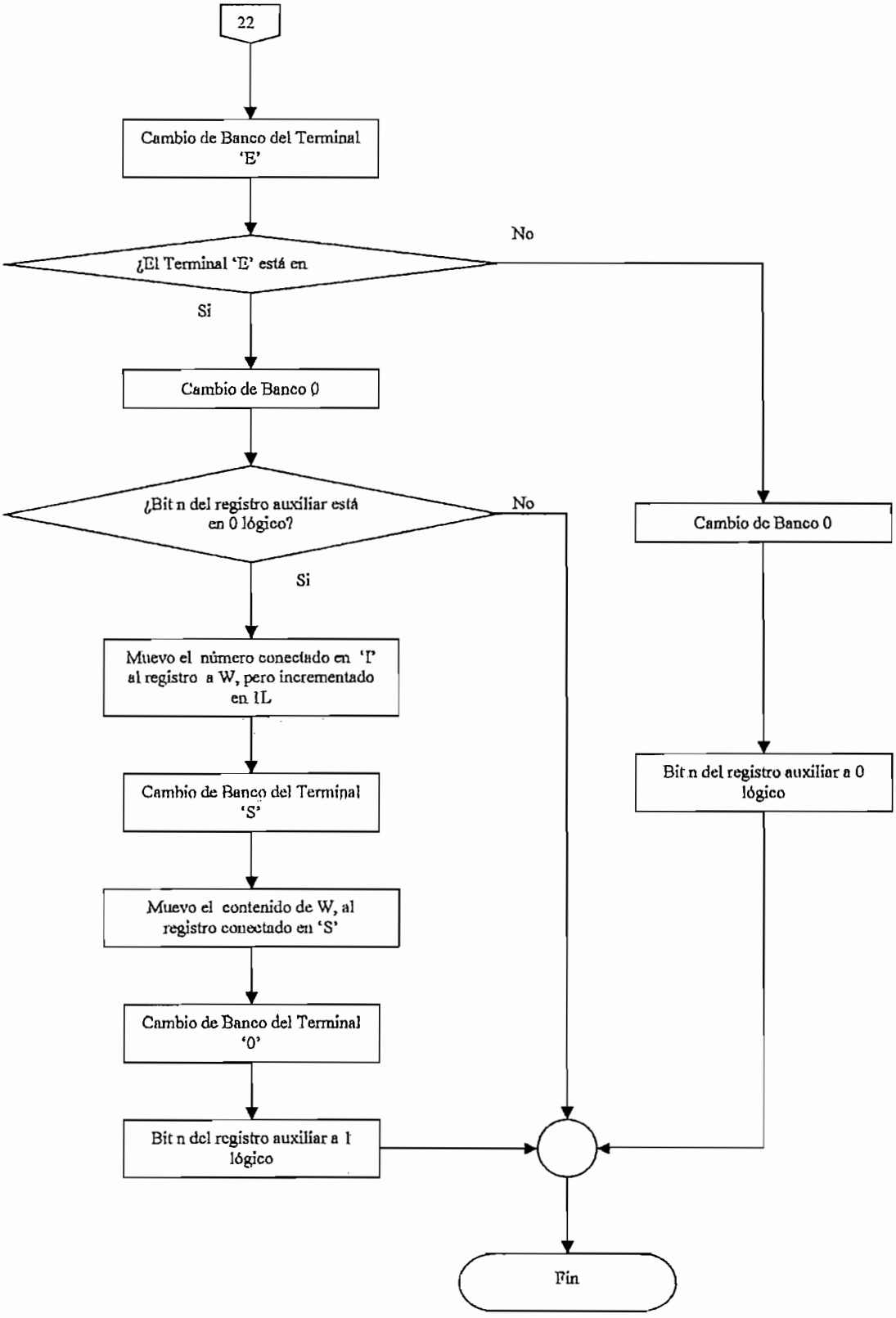
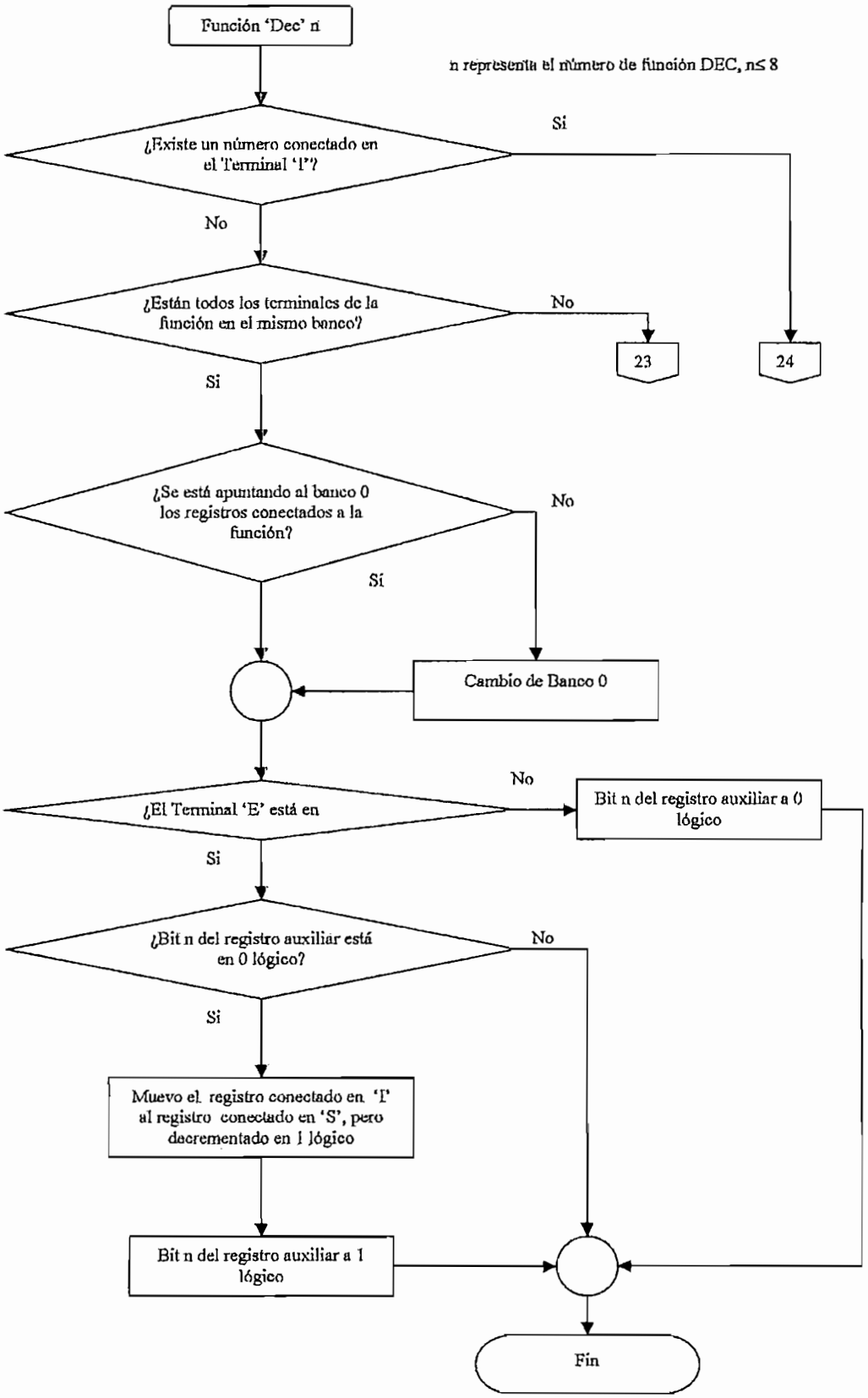
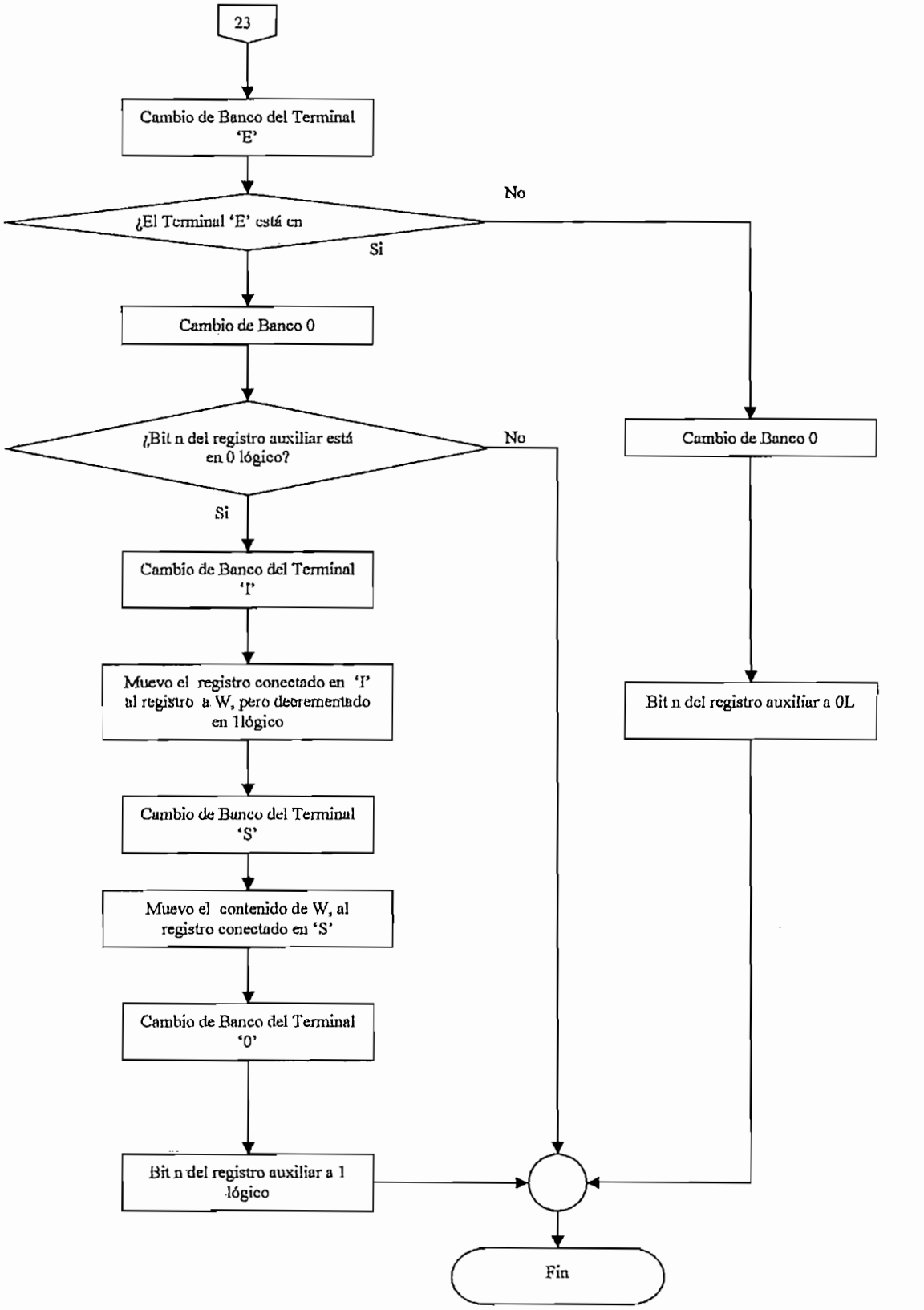
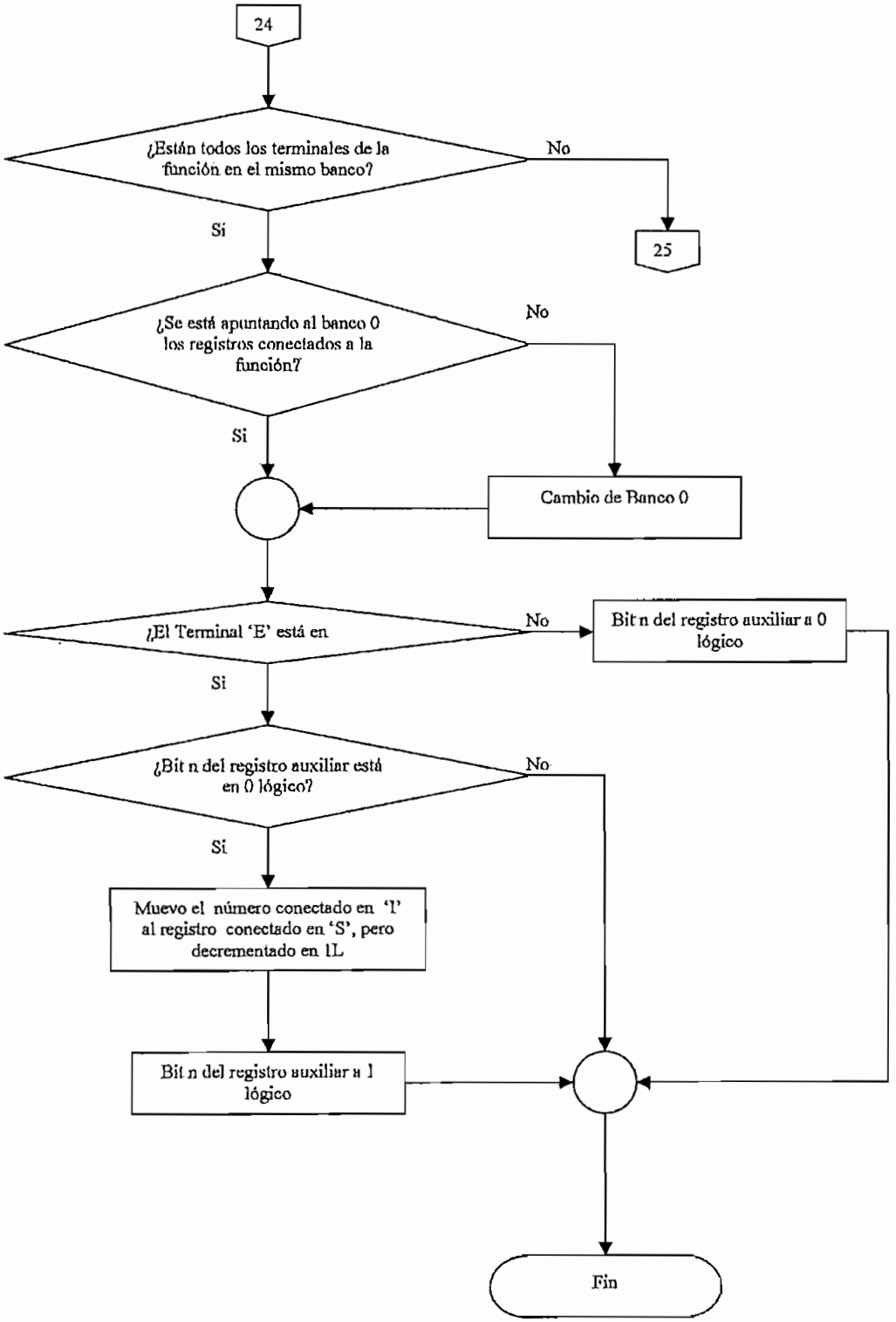


Figura 4.70. Diagrama de Flujo de la Función Incremento.

El diagrama de flujo para la Función decremento se representa en la Figura 4.71.




23
Cambio de Banco del Terminal 'E'
¿El Terminal 'E' está en
No
Si
Cambio de Banco 0
¿Bit n del registro auxiliar está en 0 lógico?
No
Si
Cambio de Banco del Terminal 'I'
Muevo el registro conectado en 'I' al registro a W, pero decrementado en 1lógico
Cambio de Banco del Terminal 'S'
Muevo el contenido de W, al registro conectado en 'S'
Cambio de Banco del Terminal '0'
Bit n del registro auxiliar a 1 lógico
Cambio de Banco 0
Bit n del registro auxiliar a 0L
Fin


24
¿Están todos los terminales de la función en el mismo banco?
No
Si
25
¿Se está apuntando al banco 0 los registros conectados a la función?
No
Si
Cambio de Banco 0
¿El Terminal 'E' está en
No
Bit n del registro auxiliar a 0 lógico
Si
¿Bit n del registro auxiliar está en 0 lógico?
No
Si
Muevo el número conectado en 'I' al registro conectado en 'S', pero decrementado en 1L
Bit n del registro auxiliar a 1 lógico
Fin

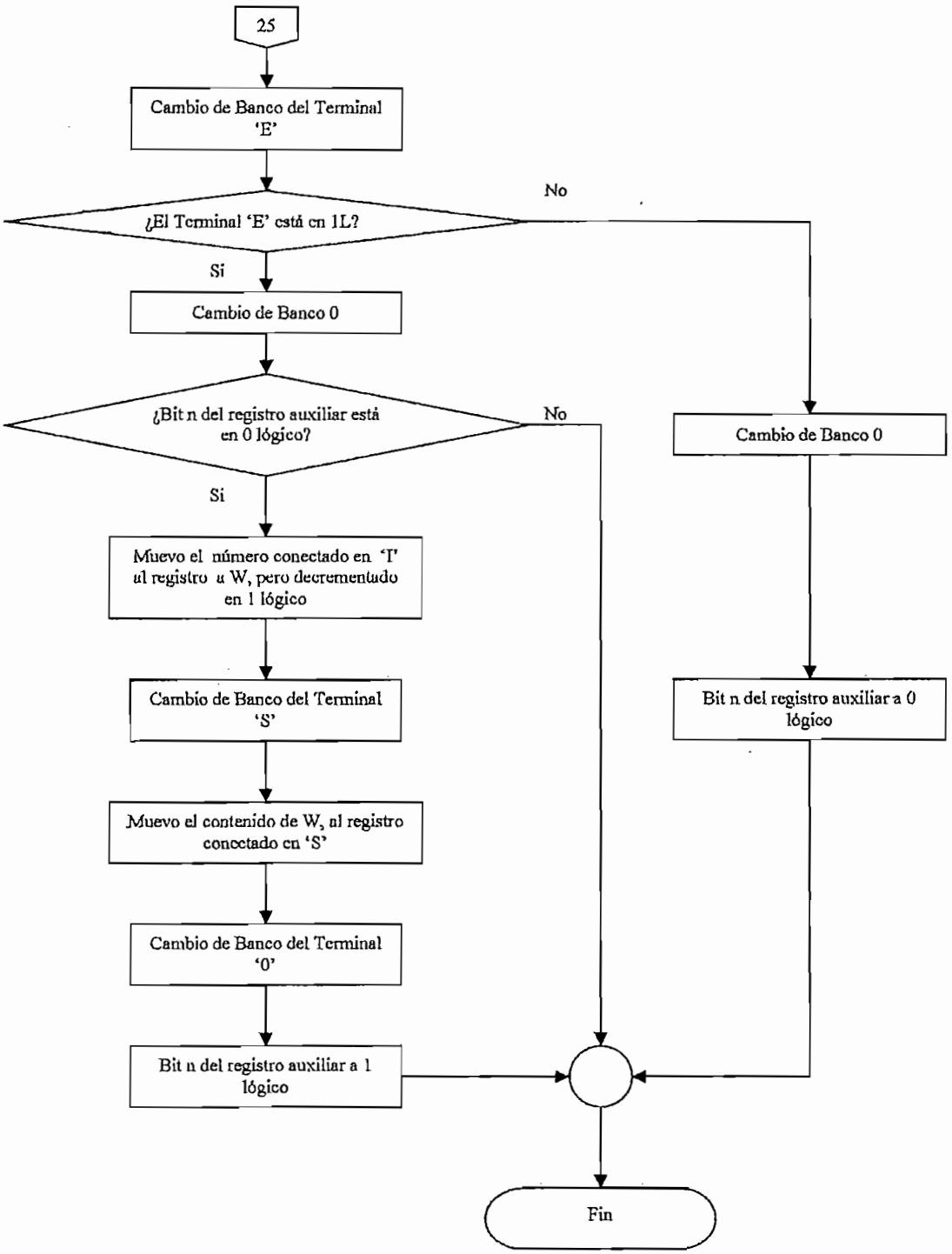
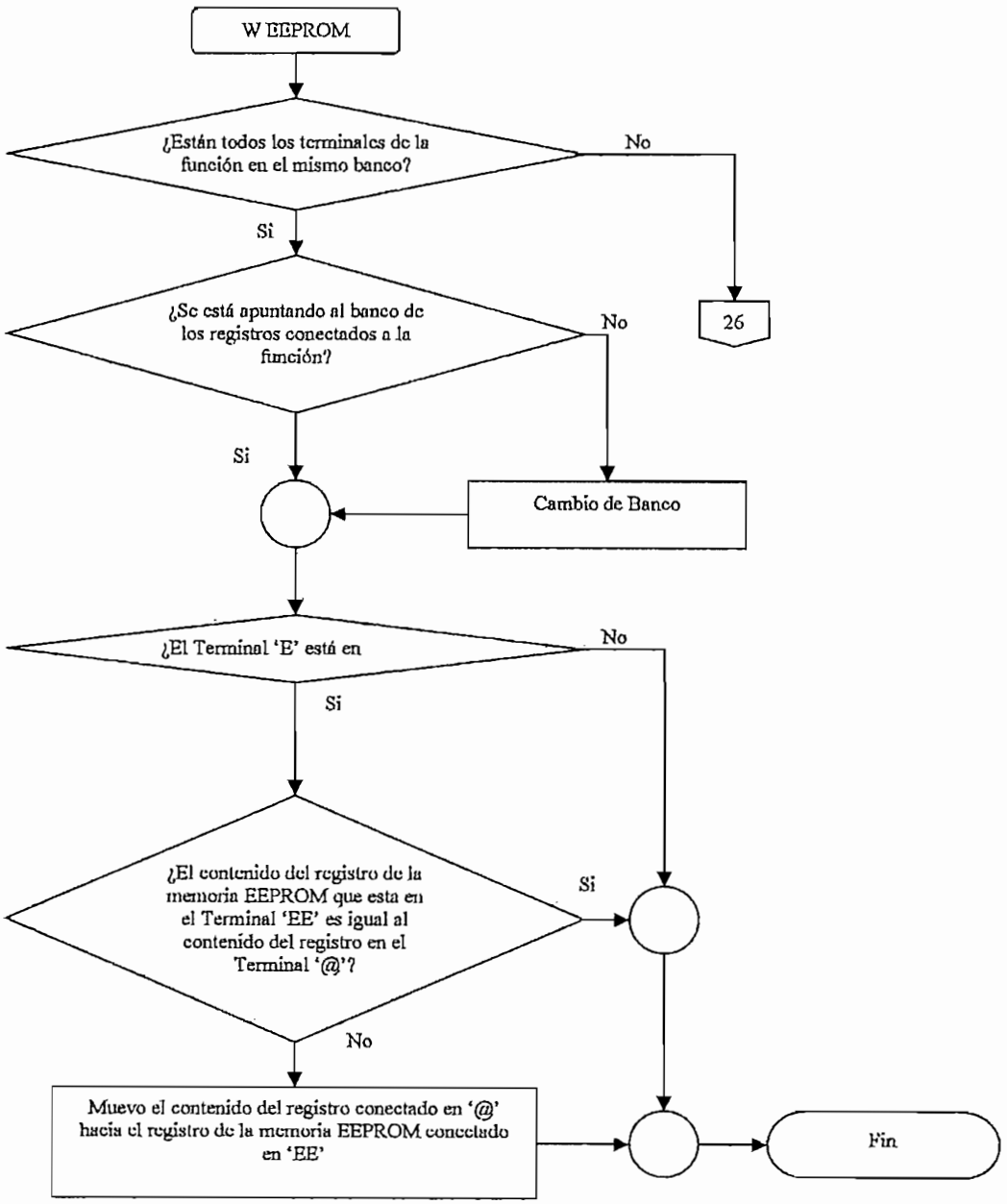


Figura 4.71. Diagrama de Flujo de la Función Decremento.

Funciones para el manejo de la memoria de datos EEPROM

Función 'W EEPROM', escribe un dato en una posición de la memoria de datos, como ya se indicó anteriormente. En el algoritmo para esta función, primeramente se lee el contenido del registro conectado en el Terminal de entrada '@' y luego se lo compara con el contenido de la posición de la memoria EEPROM donde se desea escribir (Terminal de salida 'EE'). Si los valores son diferentes, se escribe el valor del Terminal '@', en la posición indicada, siempre y cuando el Terminal 'E' está en 1L, el diagrama de flujo de esta función se muestra en la Figura 4.72.



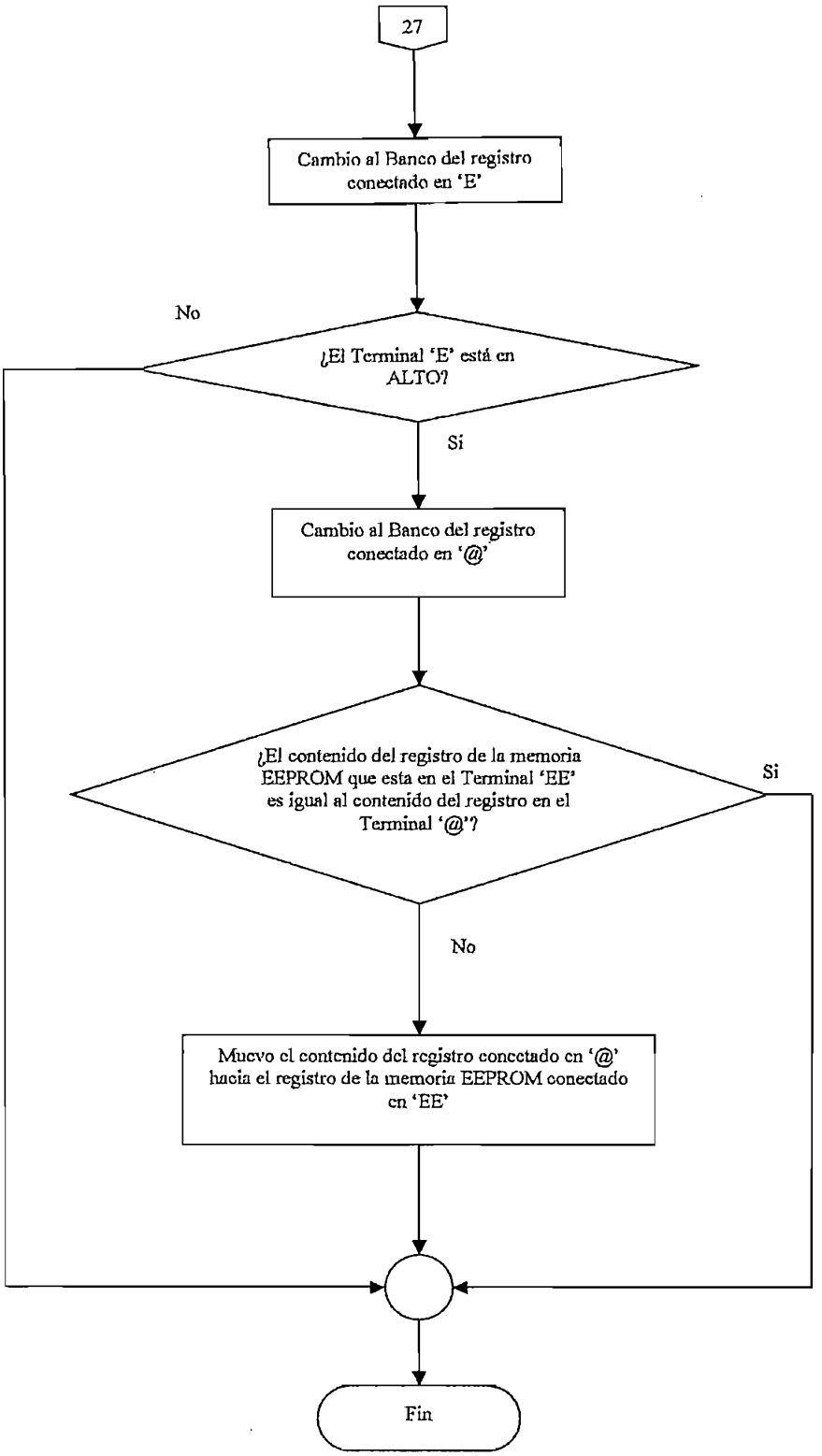
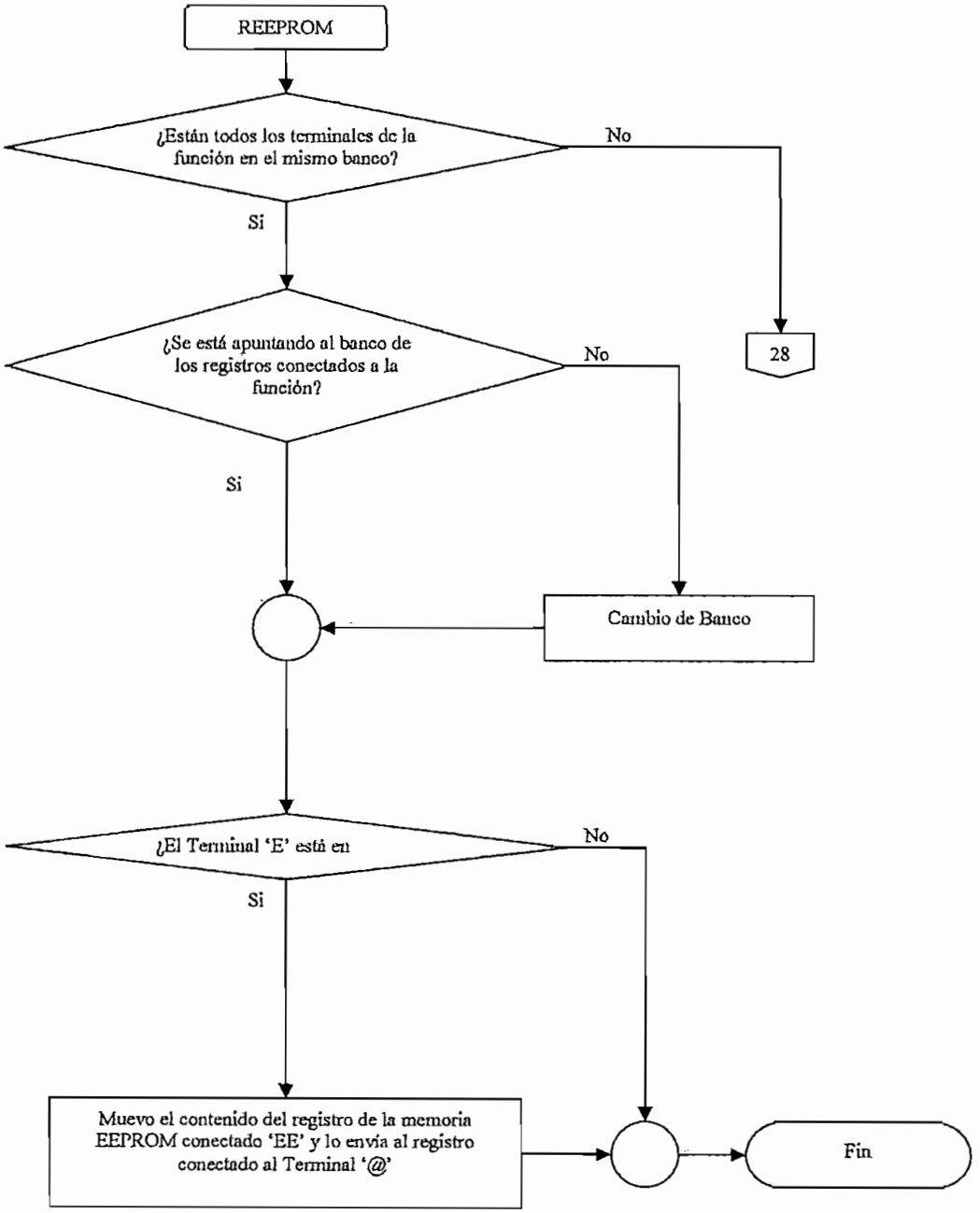


Figura 4.72. Diagrama de Flujo de la Función WEEPROM.

Función 'R EEPROM', esta función lee el contenido del registro de la memoria EEPROM conectado en el Terminal de entrada 'EE' y lo transfiere al registro conectado en el Terminal de salida '@', cuando el habilitador 'E' está en 1L. El diagrama de flujo de esta función se muestra en la Figura 4.73.



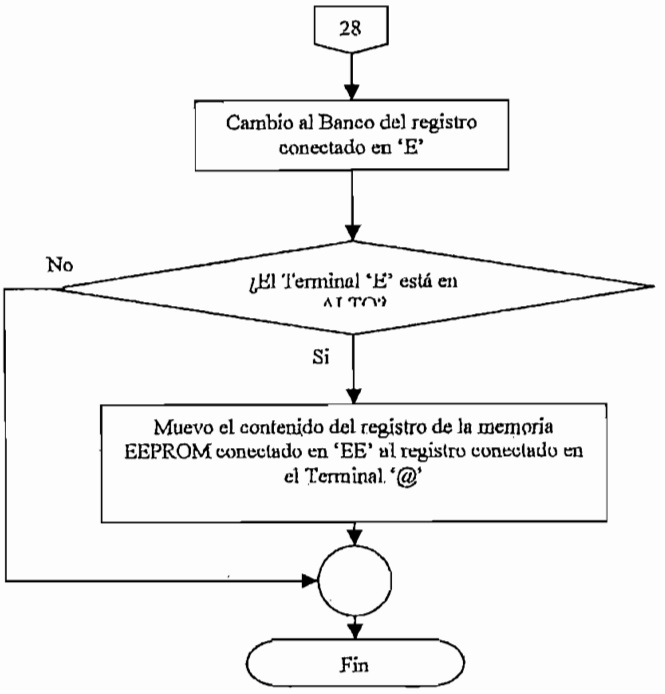


Figura 4.73. Diagrama de Flujo de la Función REEPROM.

Funciones de Salto y subrutina

Función 'Go To'

Como ya se explicó anteriormente, el programa principal del PLC puede estar formado por varios NetWorks. El programa de la simulación que está dentro del microcontrolador se va ejecutando desde el NetWork 0 hasta el NetWork, que el usuario haya definido como fin del programa principal de ahí se retorna nuevamente al NetWork 0. El comando 'Go To' permite alterar el flujo del programa principal, consiguiendo un salto condicionado hacia otro NetWork. Si el habilitador 'E' está en 1L, se realiza el salto hacia el NetWork que se indica en la etiqueta que está conectada al Terminal de salida. Se debe indicar que la memoria de programa del microcontrolador utilizado en el presente proyecto, esta divida en 4 zonas llamadas 'páginas' (de forma similar a la memoria RAM). Cuando se va a realizar un salto que sobrepasa el tamaño de la página actual, se debe realizar previamente un cambio de página. Esta función tiene esta propiedad e internamente en el programa de la computadora se controla esta posible situación. El diagrama de flujo para este comando se indica en la Figura 4.74.

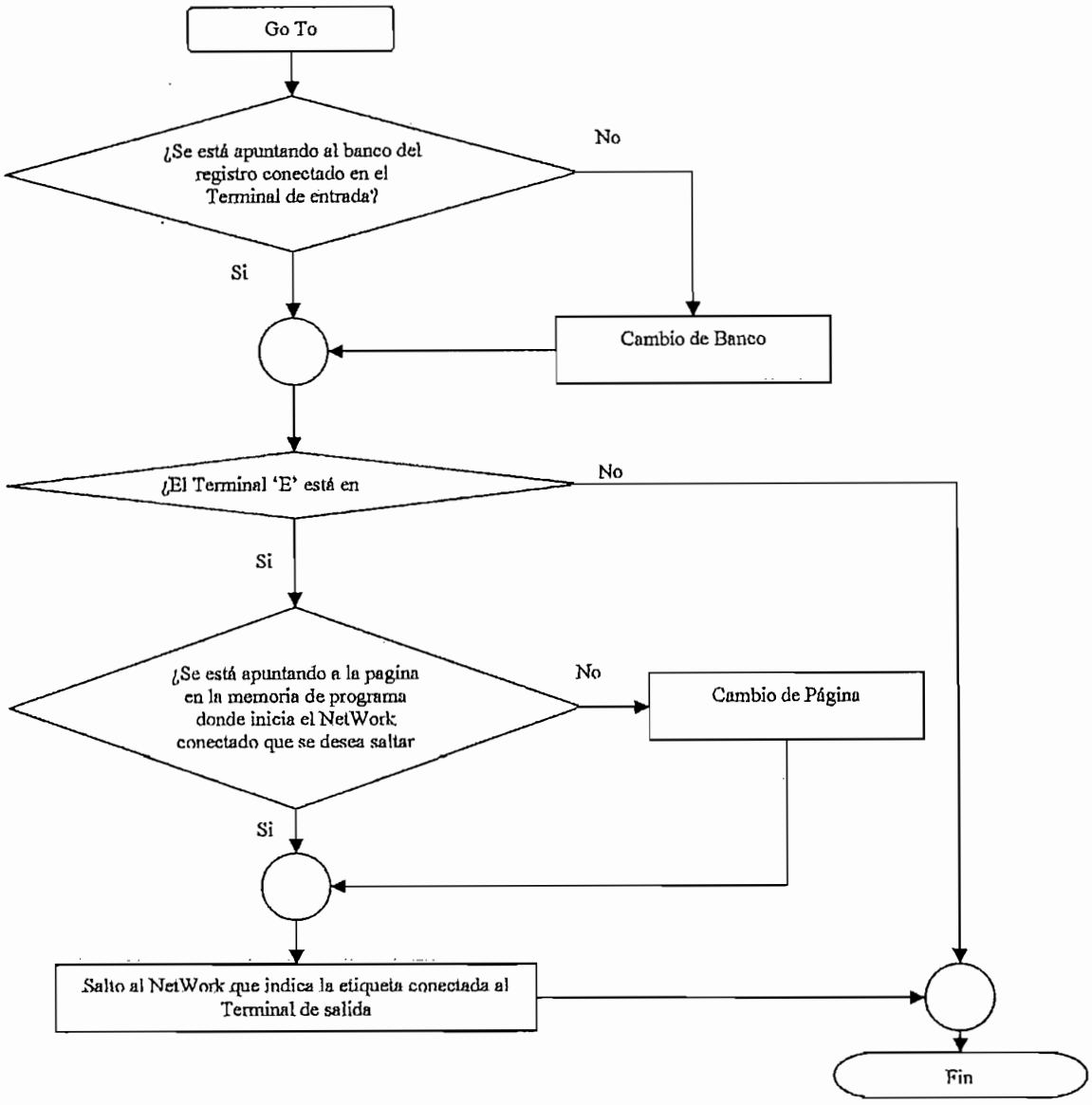


Figura 4.74. Diagrama de Flujo de la Función de salto GO TO.

Comando 'Call'

Los NetWorks que no se utilizan dentro del programa principal pueden ser utilizados para subrutinas. Para llamar a estás se recurre al comando 'Call' cuyo comportamiento es similar al comando 'Go To'; con la diferencia, que el flujo del programa principal regresa al punto donde se llamo a la subrutina. En este caso, también es necesario revisar si es amerita o no, un cambio de página antes de llamar a la subrutina, y previamente antes de retornar. El diagrama de flujo para este comando se indica en la Figura 4.75.

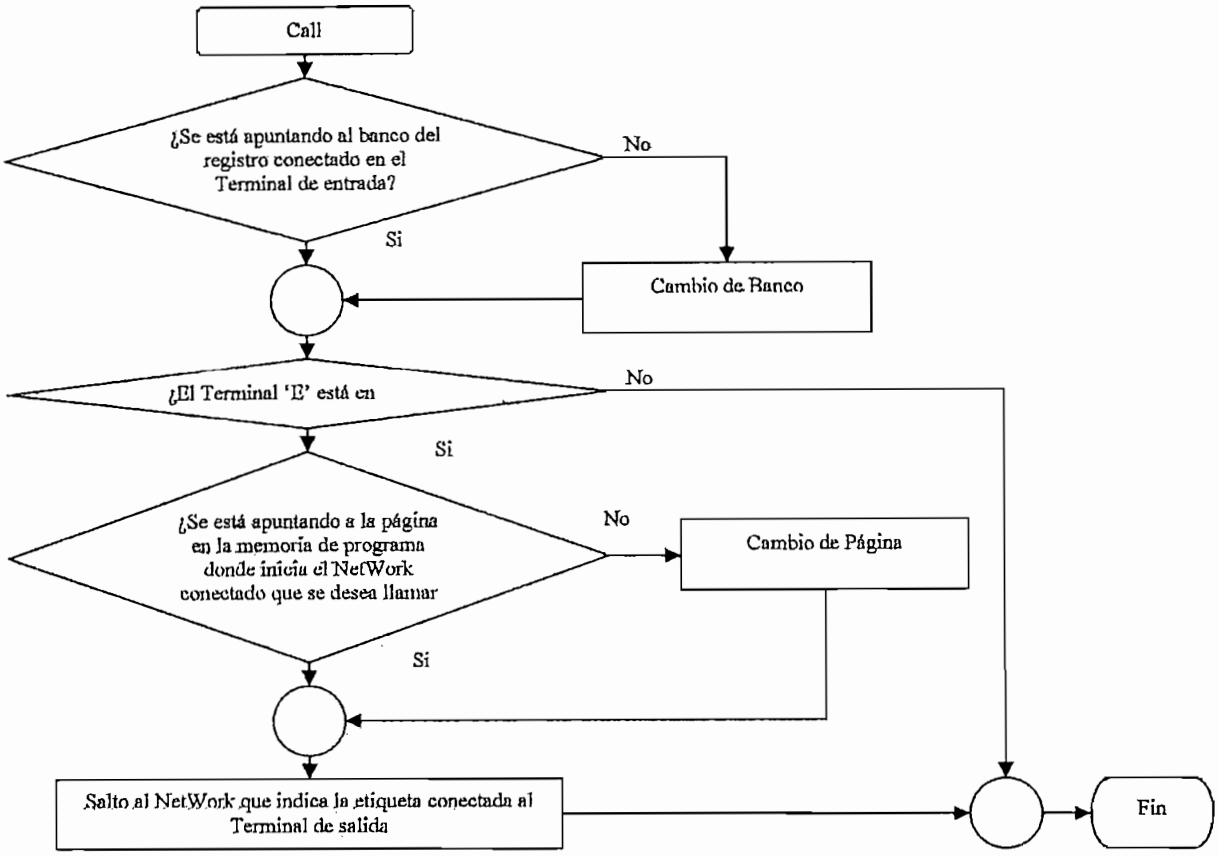


Figura 4.75. Diagrama de Flujo de la Función de salto CALL.

Funciones para recepción y transmisión serial asincrónica RS232C

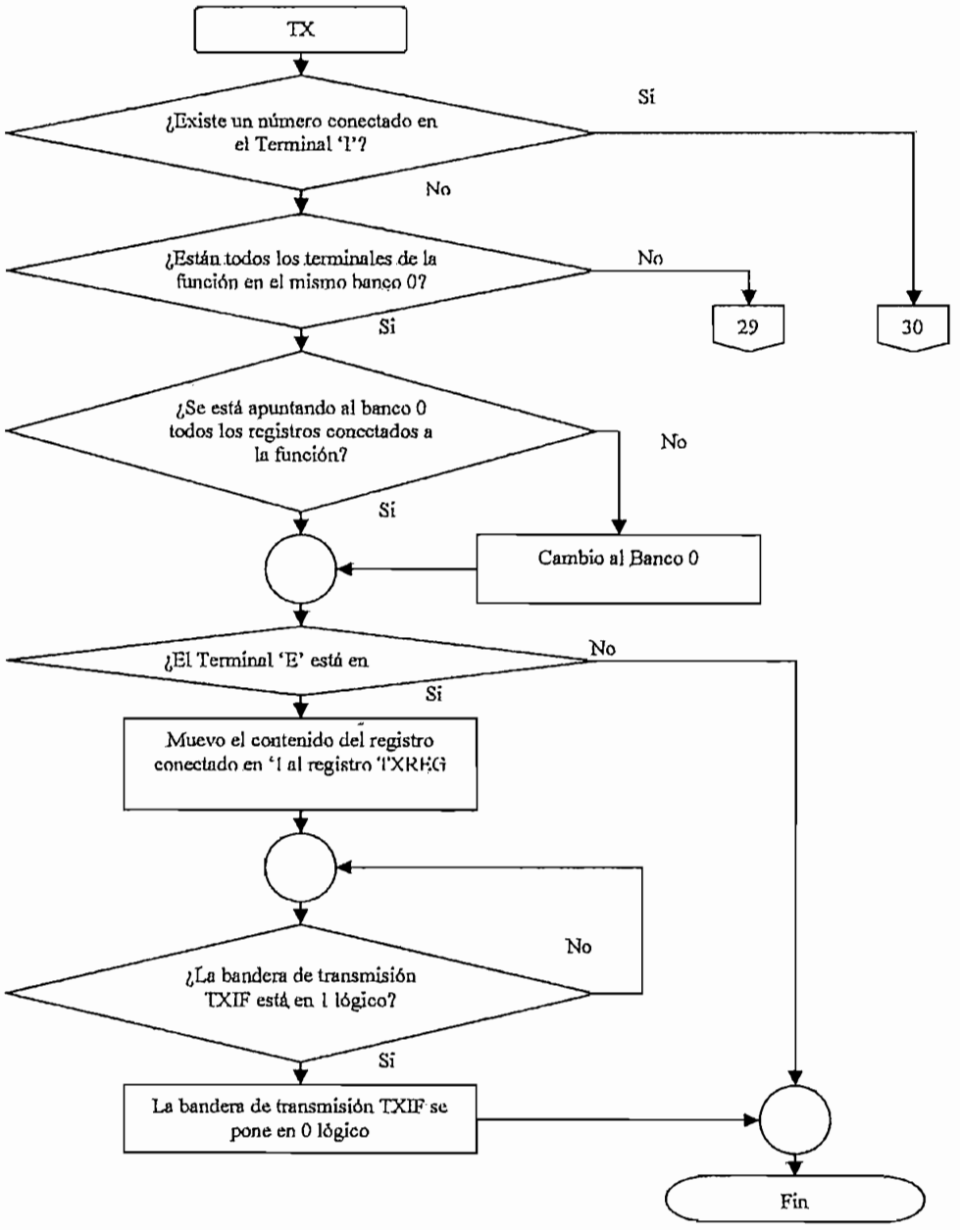
El microcontrolador PIC, tiene internamente un módulo USART (Universal Synchronous Asynchronous Receiver Transmitter) que es una interfaz de comunicación serial. Este módulo configura la comunicación en modo full duplex y realiza las siguientes funciones de manera automática:

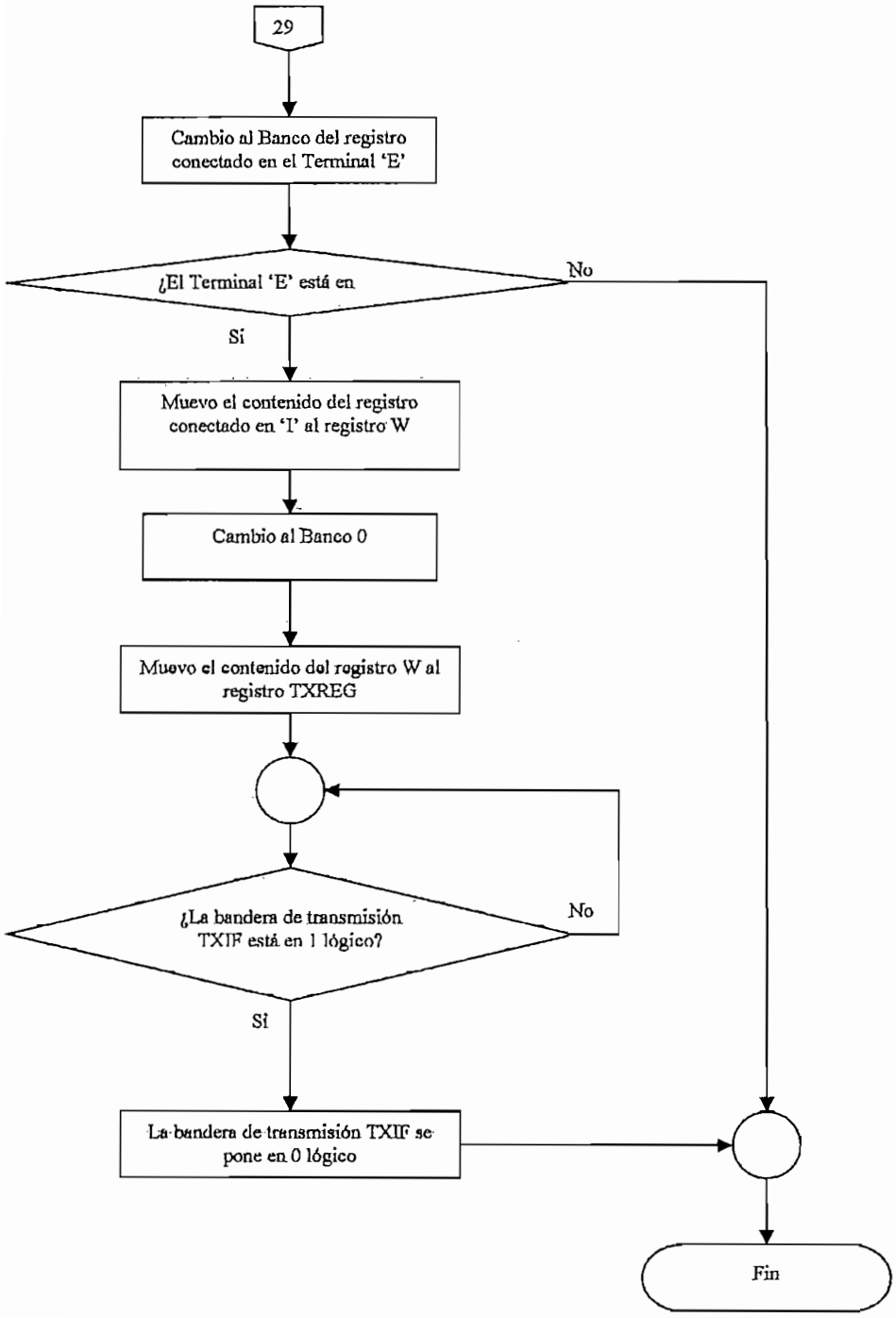
- Circuito de muestreo
- Generación de Baudios
- Transmisor y receptor asincrónico.

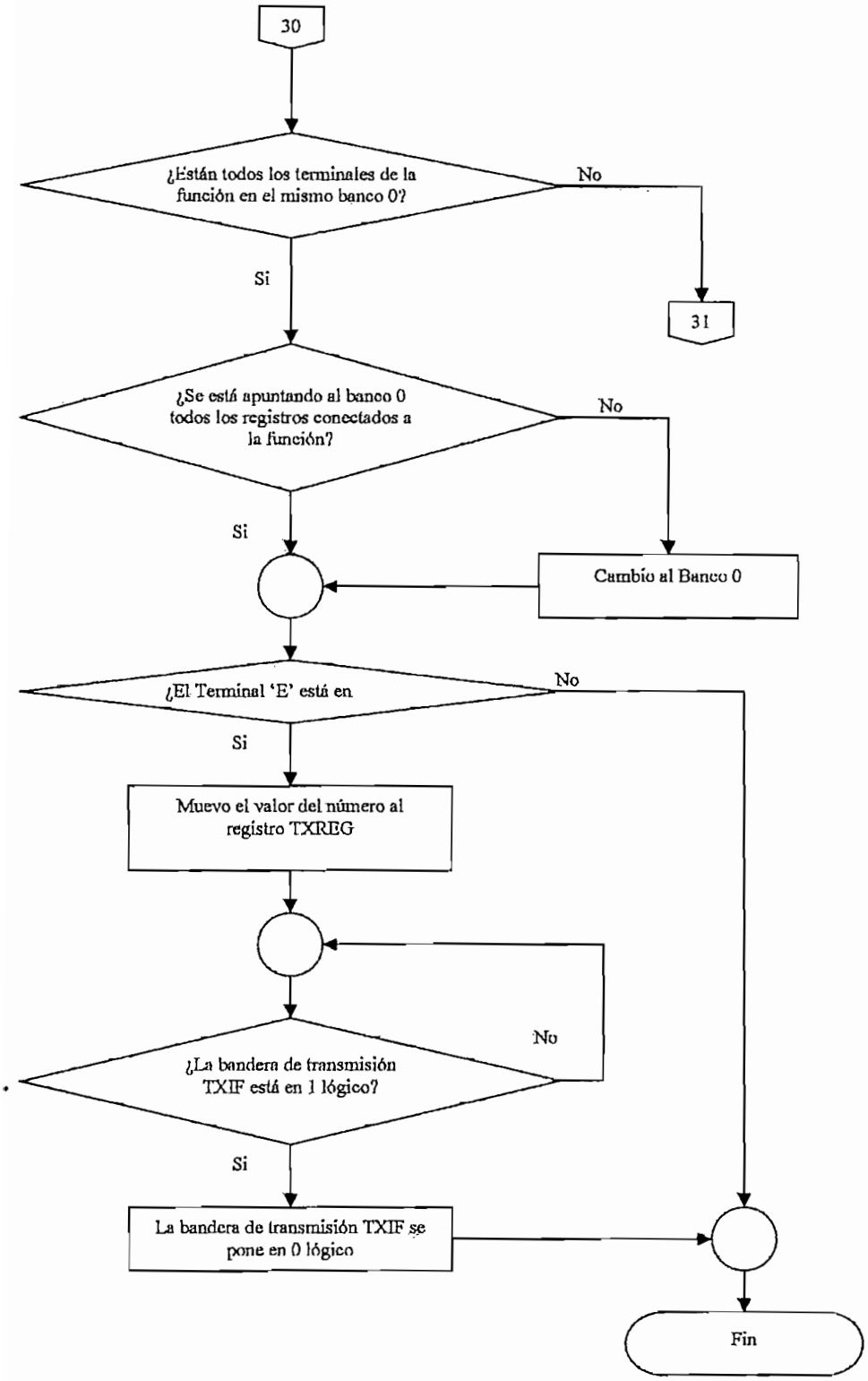
El circuito de muestreo revisa que por el pin RC7 llegue la información (bytes). La generación de baudios permite adaptar la velocidad de transmisión y recepción del PIC a las velocidades de comunicación normalizadas RS232 C de 8 bits, sin

bit de paridad. El transmisor asincrónico permite enviar un byte al pin de salida RC6 (de manera casi transparente para el usuario), primeramente se deposita el valor que se desea transmitir a un registro de la memoria RAM llamado TXREG, luego se espera que el flag o bandera TXIF se ponga en 1L, lo que representa el término de la transmisión. El receptor asincrónico en cambio, genera una interrupción y levanta la bandera RCIF cuando llega un byte al registro RCREG.

La función 'Tx' permite trasmitir un byte serialmente por el Terminal RC6 del PIC. El diagrama de flujo de la función 'TX' se representa en la Figura 4.76.




29
Cambio al Banco del registro conectado en el Terminal 'E'
¿El Terminal 'E' está en
No
Si
Muevo el contenido del registro conectado en 'I' al registro W
Cambio al Banco 0
Muevo el contenido del registro W al registro TXREG
¿La bandera de transmisión TXIF está en 1 lógico?
No
Si
La bandera de transmisión TXIF se pone en 0 lógico
Fin


30
¿Están todos los terminales de la función en el mismo banco 0?
No
Si
31
¿Se está apuntando al banco 0 todos los registros conectados a la función?
No
Si
Cambio al Banco 0
¿El Terminal 'E' está en
No
Si
Muevo el valor del número al registro TXREG
No
¿La bandera de transmisión TXIF está en 1 lógico?
Si
La bandera de transmisión TXIF se pone en 0 lógico
Fin

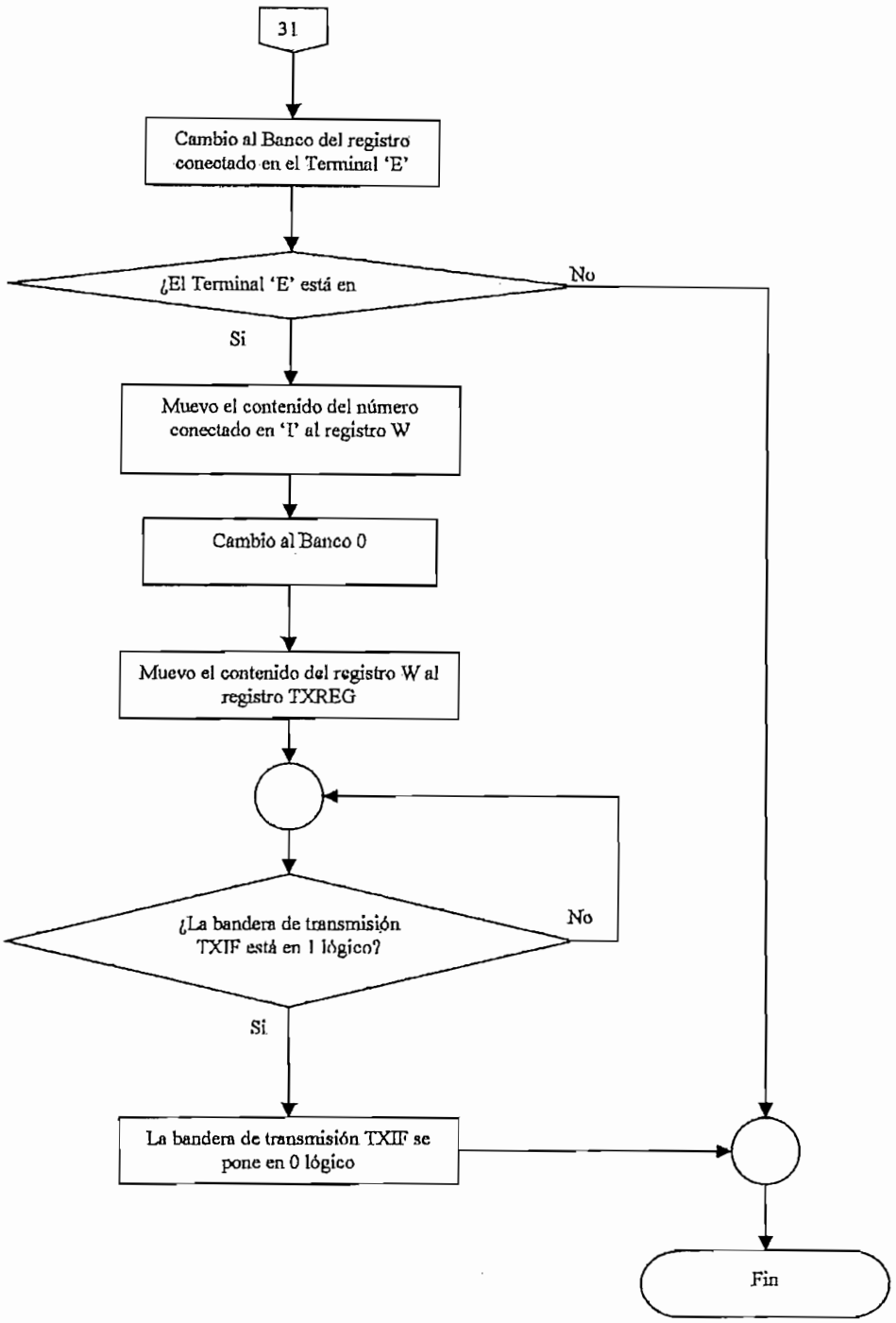


Figura 4.76. Diagrama de Flujo de la Función Tx.

Cuando se desea recibir un byte por medio del puerto serial del PLC, se debe asociar a un NetWork una subrutina de interrupción, por recepción serial. En esta subrutina se debe utilizar la función 'RX'

El diagrama de flujo de la función 'RX' se representa en la Figura 4.77.

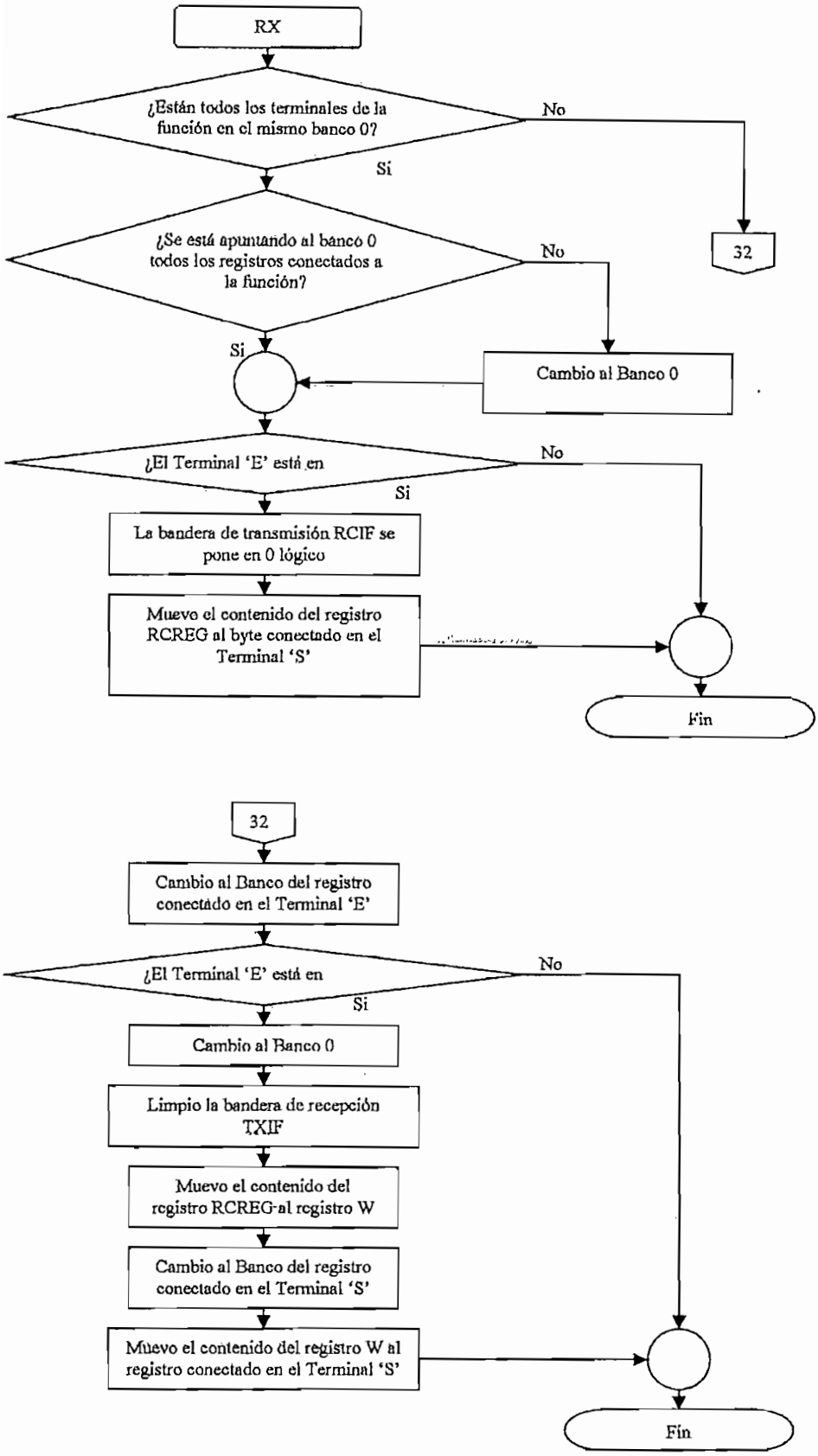
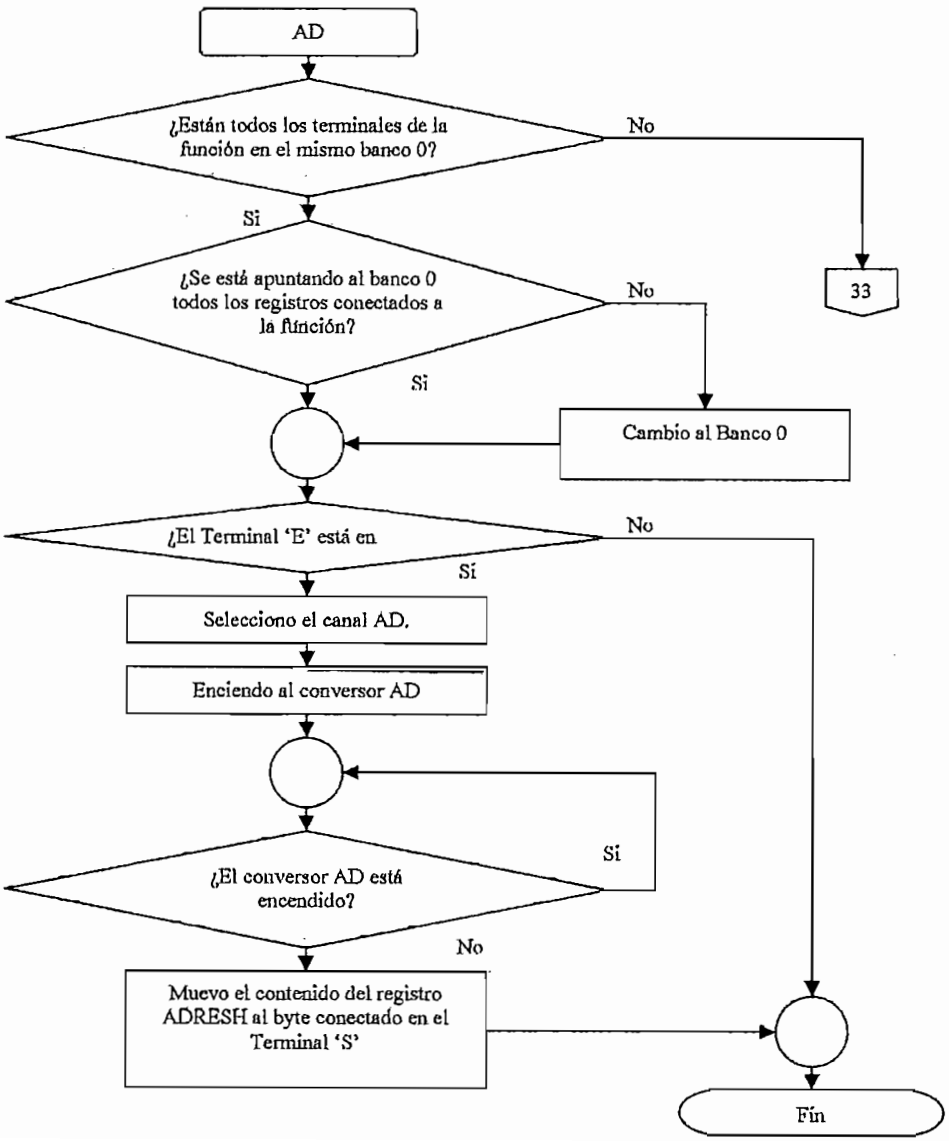


Figura 4.77. Diagrama de Flujo de la Función RX

Función de conversión analógica

Correspondiente al noveno grupo de funciones a nivel de bytes, esta la Función A/D. El conversor AD interno que posee el microcontrolador PIC, tiene la posibilidad de utilizar hasta 8 canales analógicos (Puerto A y Puerto E). El registro que controla a dicho canal es el ADCON0, con el bit 2 de este registro se puede encender al conversor AD y cuando la conversión termina, automáticamente este bit se pone a cero lógico. La selección del canal AD se consigue con los bits 5, 4 y 3, el resultado de la conversión se almacena en el registro ADRESH.

El diagrama de flujo para esta función se representa en la Figura 4.78.



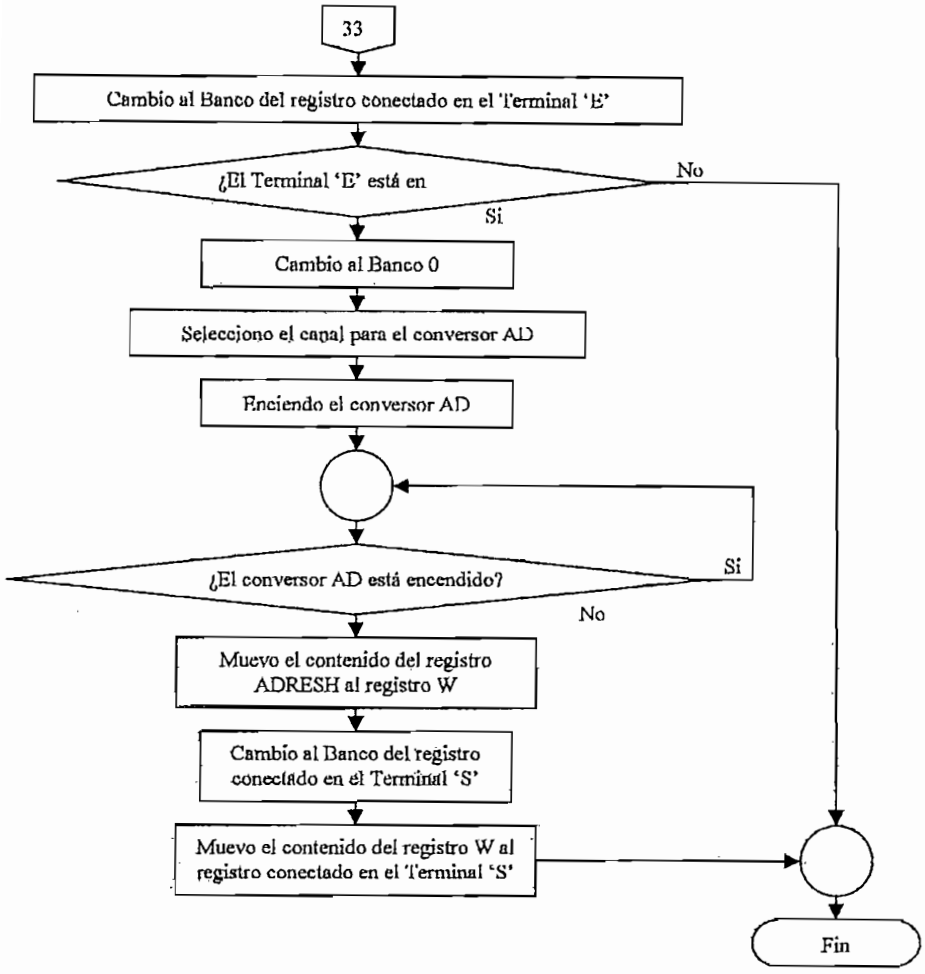


Figura 4.78. Diagrama de flujo de la Función 'A/D'.

Los algoritmos de las diferentes funciones del PLC y que se explicaron anteriormente, son utilizados para la simulación, programación del microcontrolador o generación del código *.hex. Sin embargo, estos no son todos los algoritmos que este software dispone, pero tal vez sean los de mayor importancia para los lectores de esta Tesis.

4.3.2.3 Desarrollo del algoritmo para el Simulador

Para poder simular el software del PLC, se necesitó emular la arquitectura del microcontrolador PIC, con las características que posee el PLC de este proyecto. En el programa del simulador existen matrices dinámicas para almacenar la información que contienen: los puertos, la memoria RAM y la memoria EEPROM;

así como, los registros que se utilizan en el conversor AD, las salidas PWM y la comunicación serial RS232.

Para poder actualizar constantemente la información de las matrices anteriormente mencionadas. Se utiliza un temporizador interno para que lea constantemente las entradas, ejecute las operaciones del PLC, y actualice las salidas cada 10 ms; es decir, que la simulación no es en tiempo real. Los algoritmos que se explicaron anteriormente (de las funciones booleanas y las funciones a nivel de bytes) son subrutinas que tanto el simulador como el programador del microcontrolador PIC utilizan para su funcionamiento. Este temporizador se lo enciende al presionar el pulsador '*Star*' y se lo detiene con el pulsador '*Stop*'. El algoritmo que se efectúa cuando el temporizador se desborda, se representa en la Figura 4.79.

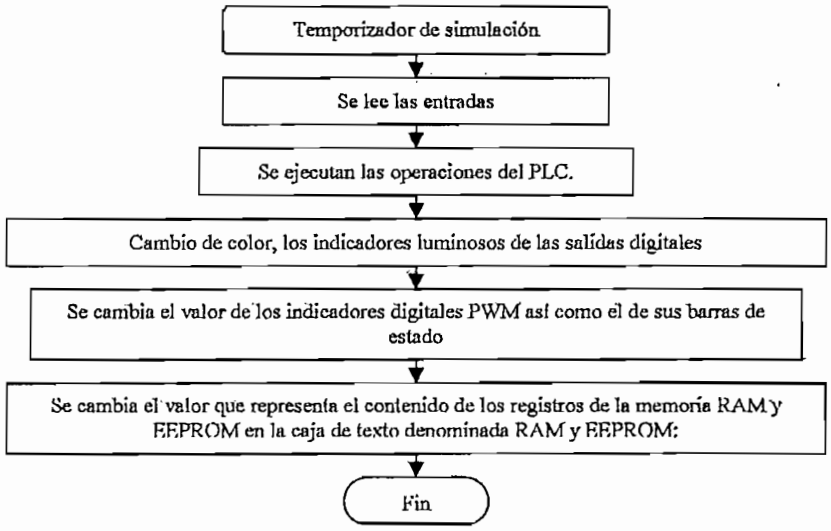


Figura 4.79. Algoritmo del desborde del Temporizador que revisa el estado de las salidas.

La estructura del algoritmo de la Figura 4.79 anterior se describe a continuación en lenguaje estructurado:

Se lee las entradas

Desde el NetWork 0 hasta el NetWork que el usuario haya definido como programa principal, se van leyendo las funciones en orden.

Si la función no es un comando Goto o Call entonces

Se lee la información de cada Terminal de entrada de las funciones para conocer donde están conectadas dichas entradas.

Si la función es un comando 'Goto' entonces

Se altera el orden de lectura los NetWorks

Se lee la información de cada Terminal de entrada de las funciones para conocer donde están conectadas dichas entradas.

Si la función es un comando 'Call'

El orden de lectura de los NetWorks se dirige hacia aquellos que no están en el programa principal.

Se incrementa en uno la variable denominada 'control de Pila'

Se verifica que la variable 'Control de Pila' no exceda de 8, ya que este es el máximo número de anidamientos posibles con subrutinas.

Se lee la información de cada Terminal de entrada de las funciones para conocer donde están conectadas dichas entradas.

Al final del NetWork rutina, se vuelve a restaurar el normal flujo de lectura de los NetWorks y se decrementa en uno la variable 'Control de Pila'

Fin Tarea

Ejecutar operaciones del PLC.

Dependiendo del tipo de función se llama a la subrutina que ejecutara la operación indicada.

El resultado de la operación realizada se almacena en varios registros temporales denominados 'Resultado'

Fin Tarea.

Cambiar de color de los indicadores luminosos de las salidas digitales.

Se lee todas las salidas de las funciones que sean cualquier bit de cualquier puerto del PLC para saber donde están y a que registros pertenecen.

Con ayuda de los registros 'Resultado' y la información de donde están las salidas digitales se procede a cambiar el color de los indicadores luminosos

Fin Tarea.

Cambiar Visualizadores PWM.

Se lee todas las salidas de las funciones PWM, si es que existen.

Con ayuda de los registros 'Resultado', y la información anterior, se procede a cambiar los indicadores digitales de ancho de pulso PWM y el ancho de las barrar de estado que también indican el ancho de pulso PWM.

Fin Tarea.

Cambiar el valor contenido en la caja de texto 'RAM y EEPROM'

Si el usuario ha decidido visualizar registros de la RAM y/o EEPROM entonces.

En función de los resultados obtenidos se cambia el valor de los registros indicados en la caja de texto 'RAM y EEPROM'

Fin Tarea

4.3.3 DISEÑO DEL PROGRAMADOR DEL MICROCONTROLADOR Y GENERACIÓN DEL CÓDIGO HEXADECIMAL

Los diagramas de flujo de las diferentes opciones que se muestran en esta interfaz gráfica, es decir, Programar y Generar código *.hex. Se presentan a continuación:

Opción 'Programar'

Internamente en el programa existe una matriz dinámica llamada MemoriaFlash donde se carga todo lo que se va a descargar al PIC. Esta matriz contiene la información del firmware, que se verá más adelante. Esta matriz se construye en función de los algoritmos desarrollados en la Sección 4.3.2 de este capítulo. El algoritmo para esta opción, se indica en la Figura 4.80.

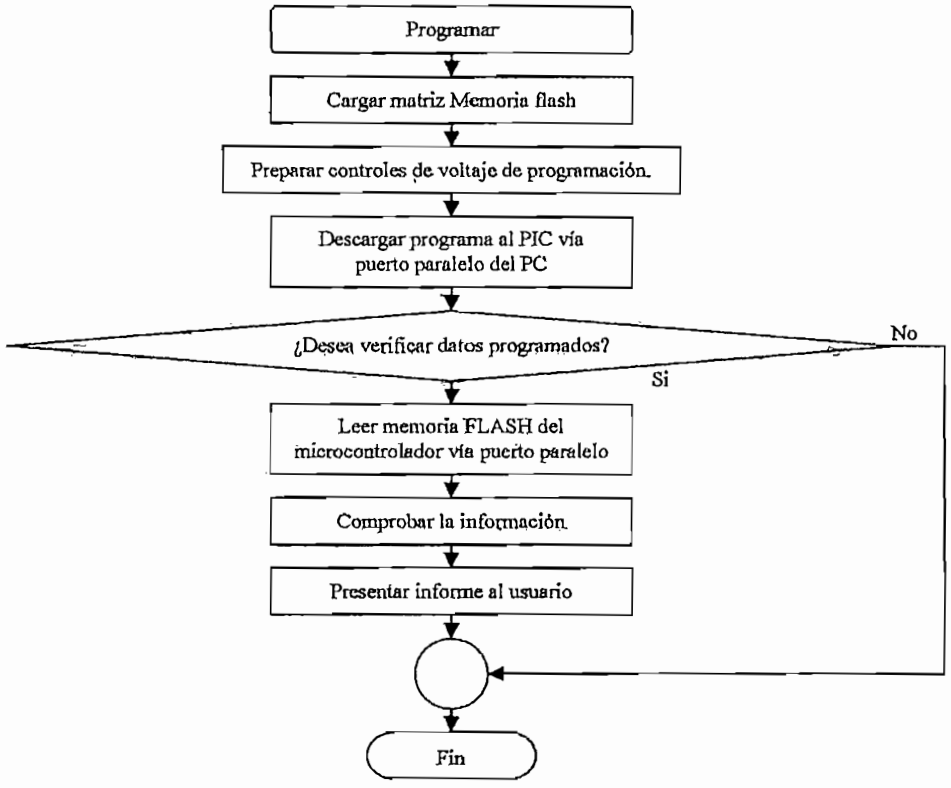


Figura 4.80. Algoritmo de programación.

La estructura del algoritmo de la Figura 4.80 anterior se describe a continuación en lenguaje estructurado:

Cargar Matriz Flash

Esto se explicará más detalladamente en el desarrollo del Firmware.

Fin Tarea

Preparación de niveles de voltaje.

A través del puerto paralelo, se aplican al microcontrolador las señales controladas de voltaje para la programación

Fin Tarea.

Descargar programa al PIC vía puerto paralelo del PC

Se desarrollan los algoritmos descritos en el Capitulo 2 para generar los comandos para programar la memoria Flash del PIC

Fin Tarea

Leer memoria FLASH del microcontrolador vía puerto paralelo

Se desarrollan los algoritmos descritos en el Capitulo 2 para generar los comandos para leer la memoria Flash del PIC

Fin Tarea

Comprobar la información

Se comprueba la información leída desde el microcontrolador PIC con la matriz MemoriaFlash para conocer si son iguales.

Fin Tarea

Presentar informe al usuario

Si la información leída desde el PIC es igual a la información de la matriz MemoriaFlash se presenta un mensaje al usuario que la programación fue correcta.

Caso contrario se informa al usuario que la programación fallo y se indica en que posición de la memoria flash del PIC no coincide con la matriz MemoriaFlash.

Fin Tarea

Opción 'Generar código *.hex'

Similar a la opción anterior, en este caso, también se carga con la información del firmware a la matriz Memoria flash. Dicha información permitirá crear el código hexadecimal, este formato está organizado por una sucesión de registros que van escritos en una línea, seguidos por los códigos ASCII de retorno de carro y avance de línea. Dentro de un registro, el dato binario es representado por dígitos ASCII hexadecimales, dos dígitos por byte binario. Por ejemplo el valor de 200 es representado por C8 en el archivo.

El detalle de estos registros se describe en la Tabla 4.3 que se encuentra a continuación.

Tipo de formato	Número de caracteres	Descripción
Cabecera	1	Representado siempre por ":"
Bytes a grabar	2	Indica el número de datos a ser grabados que contiene el registro
Dirección de inicio	4	Indica la dirección de la localidad de memoria donde se va ubicar el primer dato del registro. Para el caso de Fin de registro el valor es "0000"
Identificador de tipo de línea	2	Puede indicar dos posibilidades: "00": Indica que es una línea intermedia. "01": indica línea final
Datos	Variable	Representa los datos a ser grabados. La longitud o el número de datos es indicada por los "Bytes a grabar"
Checksum	2	Representa el complemento de dos de la suma de los Bytes precedentes

Tabla 4.3. Tipos de registros donde se guardan los datos hexadecimales.

A continuación se indica un ejemplo del formato Intel:

: 02400E00313F40

":" Que es la cabecera.

"02" Son los Bytes a grabar.

"400E" Es la dirección desde donde se va a grabar los bytes (en este caso son 2).

"00" Es el identificador de línea intermedia

"31" Es el primer byte

"3F" Es el segundo byte

"40" Es el checksum, que es el complemento de dos de la suma 02+40+0E+00+31+3F

Una vez presionado el botón 'Generar código *.hex' se despliega la ventana donde el usuario podrá guardar su proyecto en código hexadecimal, el algoritmo de esta opción se muestra a continuación en la Figura 4.49.

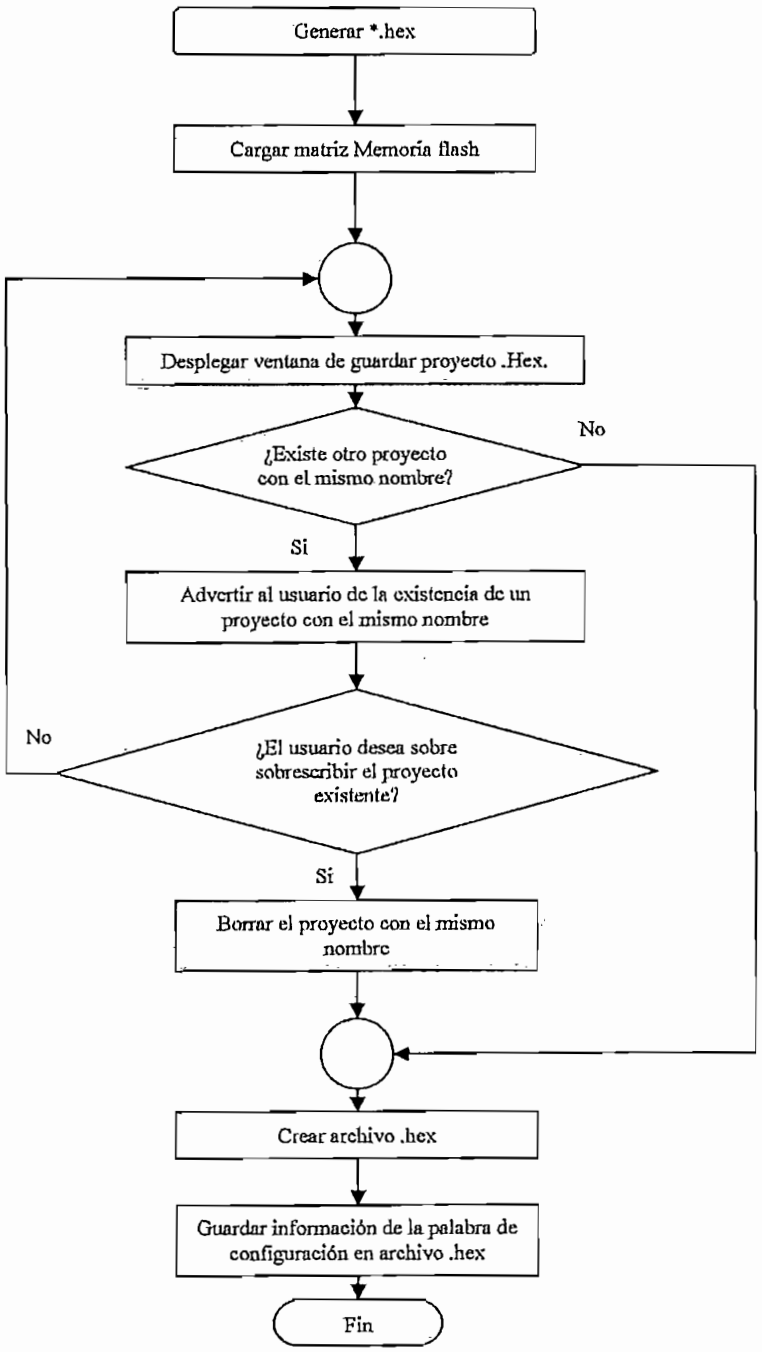


Figura 4.81. Diagrama de flujo para generar el código .hex.

La estructura del algoritmo de la Figura 4.81 anterior se describe a continuación en lenguaje estructurado:

Cargar Matriz Flash

Esto se explicará más detalladamente en el desarrollo del Firmware.

Fin Tarea

*Desplegar ventana de guardar proyecto *.hex*

Se presenta al usuario una ventana donde puede guardar en cualquier localidad de su PC el proyecto en código *.hex

Se espera que el usuario guarde el archivo con un nombre

Fin Tarea.

Advertir al usuario de la existencia de un proyecto con el mismo nombre

Si el usuario desea guardar el proyecto actual con el nombre de otro proyecto existente, se lo advierte y se le pregunta si desea sobrescribir el archivo .hex existente

Fin Tarea

Borrar el proyecto con el mismo nombre

Se procede a eliminar el archivo *.hex cuyo nombre es igual al del archivo que se desea guardar.

Fin Tarea

*Crear archivo *.hex*

Se crea un archivo hexadecimal con el nombre deseado por el usuario en blanco.

En base a la variable MatrizFlash se procede a llenar el archivo hexadecimal creado

Fin Tarea

*Guardar información de la palabra de configuración en archivo *.hex*

En función de los bits configuración previamente seleccionados por el usuario, se genera la información necesaria para la palabra de configuración.

La información de la palabra de configuración se añade al archivo *.hex.

Fin Tarea

4.3.4 DESARROLLO DEL FIRMWARE EN EL MICROCONTROLADOR

El firmware es el software que se encuentra dentro de la memoria de programa del microcontrolador PIC. Dicho software tiene las instrucciones para que el microcontrolador se comporte como un PLC, y de acuerdo al programa desarrollado y simulado en las HMIs explicadas anterior. Esta información se almacena previamente en un matriz denominada MemoriaFlash que representa a la memoria de programa del PIC16F877A.

Para escribir la información del software del PLC en la Matriz MemoriaFlash, primeramente se debe almacenar en esta, la información de configuración del microcontrolador. La configuración del microcontrolador implica cargar los valores adecuados para las distintas operaciones y funciones en los registros internos del PIC, para que este se comporte como PLC y de acuerdo con el programa elaborado en la interfaz gráfica.

En el programa de interfaz de usuario y que se maneja con Bloques de Funciones (FBD), el orden en que escriben las instrucciones en caso de ser utilizadas es:

- Si existiera por lo menos una función temporizador (ON u OFF), se debe escribir la subrutina de interrupción por dicho timer.

- Si el usuario ha asignado a un NetWork la subrutina de la interrupción externa, se debe escribir su configuración.

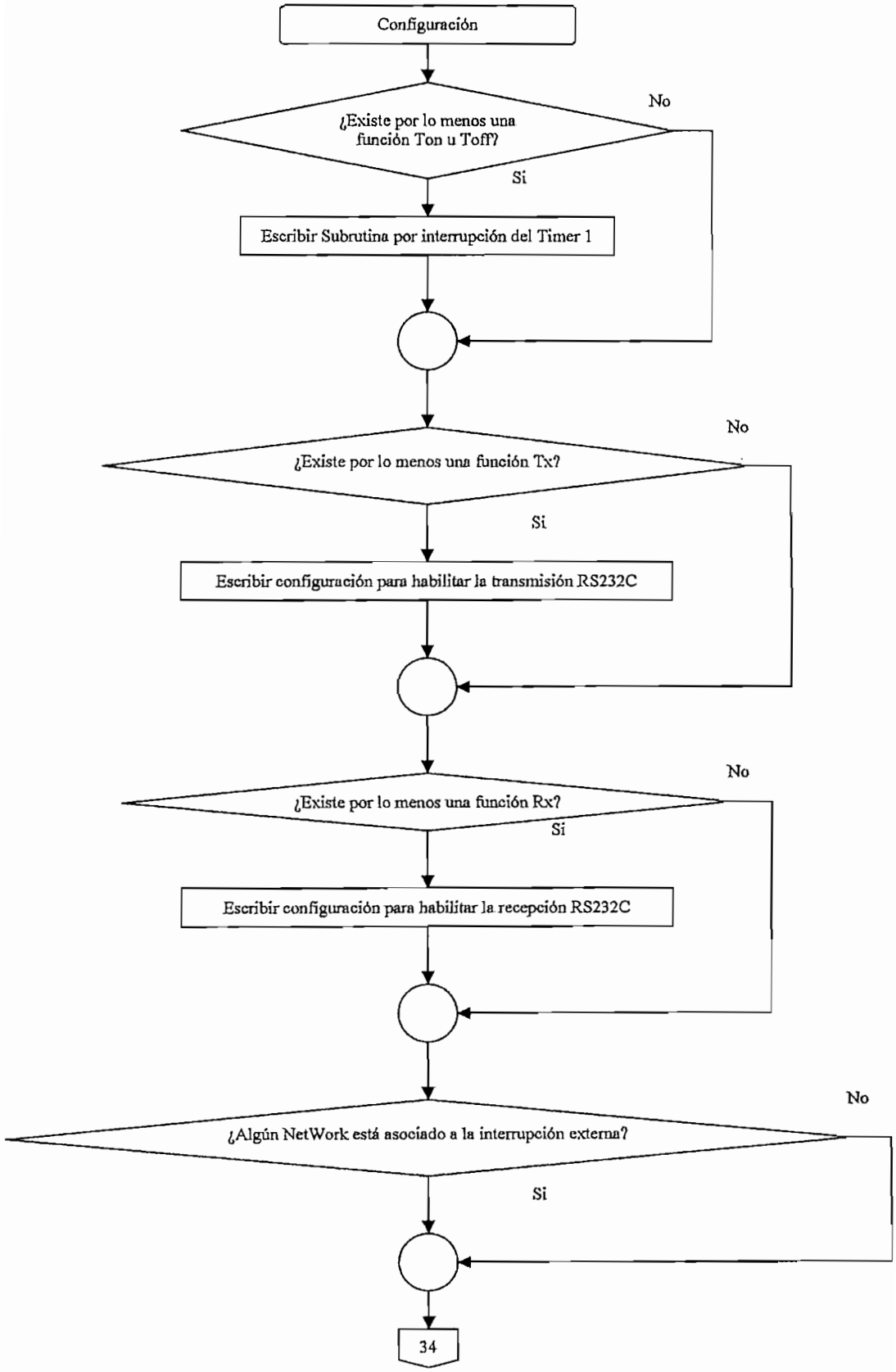
- Si existiera una función RX se debe escribir la subrutina de interrupción por recepción serial RS232C de 8bits.

- Si existiera una función TX o RX se debe escribir la configuración de la comunicación serial RS232C de 8 bits.

- Si existiera una función PWM1 y/o PWM2 se debe escribir la configuración para habilitar al módulo interno del PIC PWM.

- En caso de existir por lo menos una función AD, se debe escribir la configuración para habilitar al conversor AD dentro del PIC.

A continuación se presenta en la Figura 4.82, el diagrama de flujo de la configuración de los registros del microcontrolador PIC.


Configuración
¿Existe por lo menos una función Ton u Toff?
No
Si
Escribir Subrutina por interrupción del Timer 1
¿Existe por lo menos una función Tx?
No
Si
Escribir configuración para habilitar la transmisión RS232C
¿Existe por lo menos una función Rx?
No
Si
Escribir configuración para habilitar la recepción RS232C
¿Algún NetWork está asociado a la interrupción externa?
No
Si
34

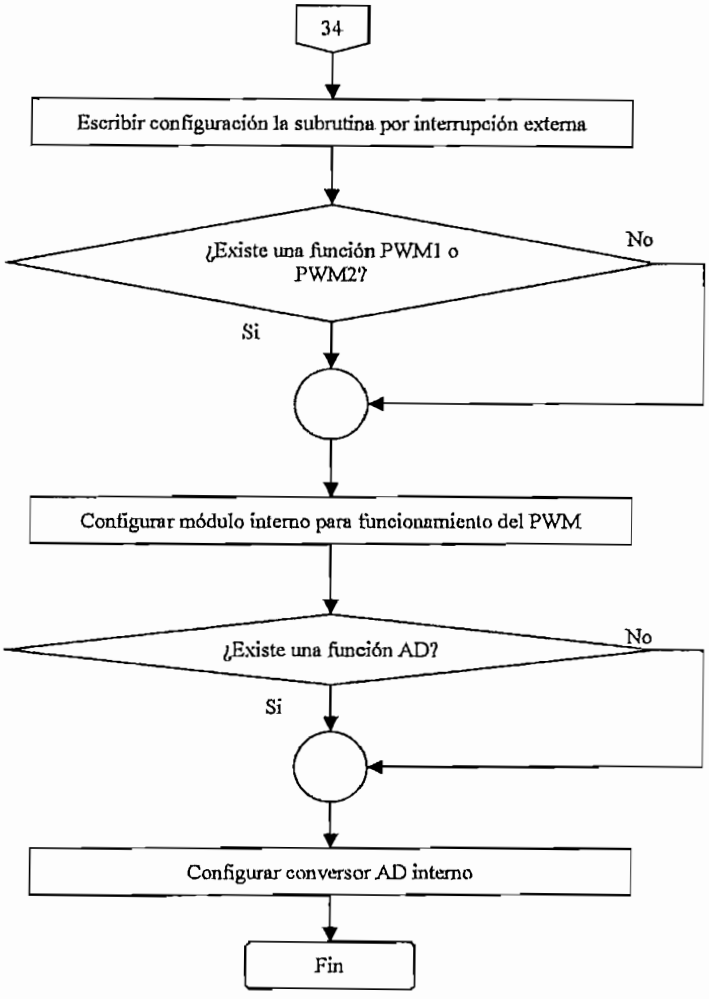


Figura 4.82. Diagrama de Flujo de la Configuración de los registros del PIC.

Escribir subrutina por interrupción del Timer 1.

Se verifica que la interrupción se ha producido por el timer 1.

Se limpia el bit bandera que indica que la interrupción se ha producido por el timer 1.

Se detiene al timer 1

Se escribe el código desarrollado en la Figura 4.66. Tanto para los temporizadores On y OFF.

Se prende el timer 1

Fin Tarea.

Escribir configuración para habilitar la transmisión RS232C

Se calibra los registros para que el módulo interno RS232 funcione a la velocidad en baudios escogida por el usuario.

Se calibra los registros para que funcione la transmisión serial en modo asincrónico de 8 bits sin paridad.

Fin Tarea

Escribir configuración para habilitar la recepción RS232C

Si los registros que configuran la velocidad en baudios aún no están calibrados, se los gradúa para que funcionen a la velocidad escogida por el usuario

Se calibra los registros necesarios para que la recepción funcione en el mismo modo de la transmisión.

Se habilita la interrupción por recepción serial RS232C

Fin Tarea

Escribir configuración la subrutina por interrupción externa

Se limpia la bandera por interrupción externa en RB0

Se llama al NetWork que el usuario a asociado con esta interrupción.

Fin Tarea

Configurar módulo interno para funcionamiento del PWM

Se configura los registros necesarios y al Timer 2 para el funcionamiento del módulo PWM.

Se prende al timer 2

Fin Tarea

Configurar conversor AD interno

Se configura el conversor AD interno de 10 bits de tal manera que solamente se utilice los 8 bits más significativos.

Se configura los canales de entrada al conversor como entradas analógicas (Puerto A y/o E)

Fin Tarea

Una vez cumplida con las condiciones anteriores se empezará a buscar en orden de NetWork cada una de las funciones, en base a los algoritmos desarrollados en la Sección 4.3.2. Luego se procederá a escribir el código en la matriz denominada MatrizFlash y entre cada NetWork, se escribirá el código para borrar los registros auxiliares de las funciones a nivel de bits.

Se debe configurar los bits de los puertos como entradas y salidas de acuerdo, como el usuario ha dispuesto en su proyecto. Todos los demás terminales de los puertos siempre serán configurados como entradas para protección de los mismos. Además en esta parte se procederá a cargar a los registros auxiliares de las funciones temporizador Ton y Toff explicados anteriormente y que servirán para producir los tiempos previamente establecidos en dichas funciones en el Terminal de entrada '#'. También se realizara la configuración del timer1. En la Figura 4.83 se representa el diagrama de flujo de esta configuración denominada 'Configuración 2'.

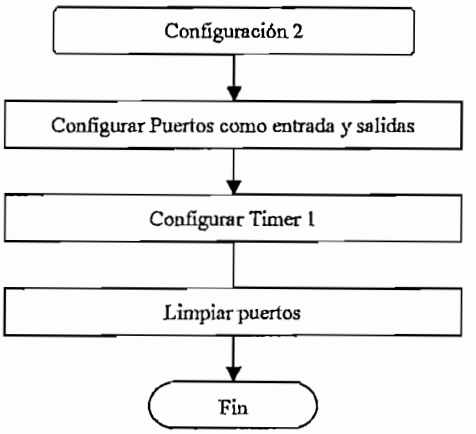


Figura 4.83. Diagrama de Flujo de la Configuración 2 del PIC.

Configurar puertos como entradas y salidas.

Dependiendo que bits de los puertos estén configurados como entradas o salidas, se procederá a configurar los registros que controlan dicha disposición.

Los bits de los puertos que no se utilizan se configurán como entradas.

Fin Tarea.

Configurar Timer 1.

Si existe por lo menos una función Ton o Toff entonces

Se procede a borrar la bandera por interrupción del timer 1.

Se calibra al timer 1 para que se desborde cada 65536 μs.

Se arranca el timer 1 y se procede a habilitar la interrupción por desborde de dicho timer.

Fin Tarea

Limpiar puertos.

Se procede a limpiar todos los puertos desde el A hasta el E

Fin Tarea

Una vez realizada esta segunda configuración, se procede finalmente a escribir el código que hará que el PIC se comporte como un PLC. Para cumplir esto se empezará a leer cada función FBD en orden de NetWork. Entre cada NetWork se procederá a limpiar a los registros auxiliares de las funciones booleanas y en caso de que el usuario haya configurado al timer perro guardián se procederá a limpiarlo para evitar el reset del PIC. Si el siguiente NetWork no está dentro del programa principal, se procederá a añadir el código de retorno de subrutina, el diagrama de flujo de este procedimiento se indica en la Figura 4.84.

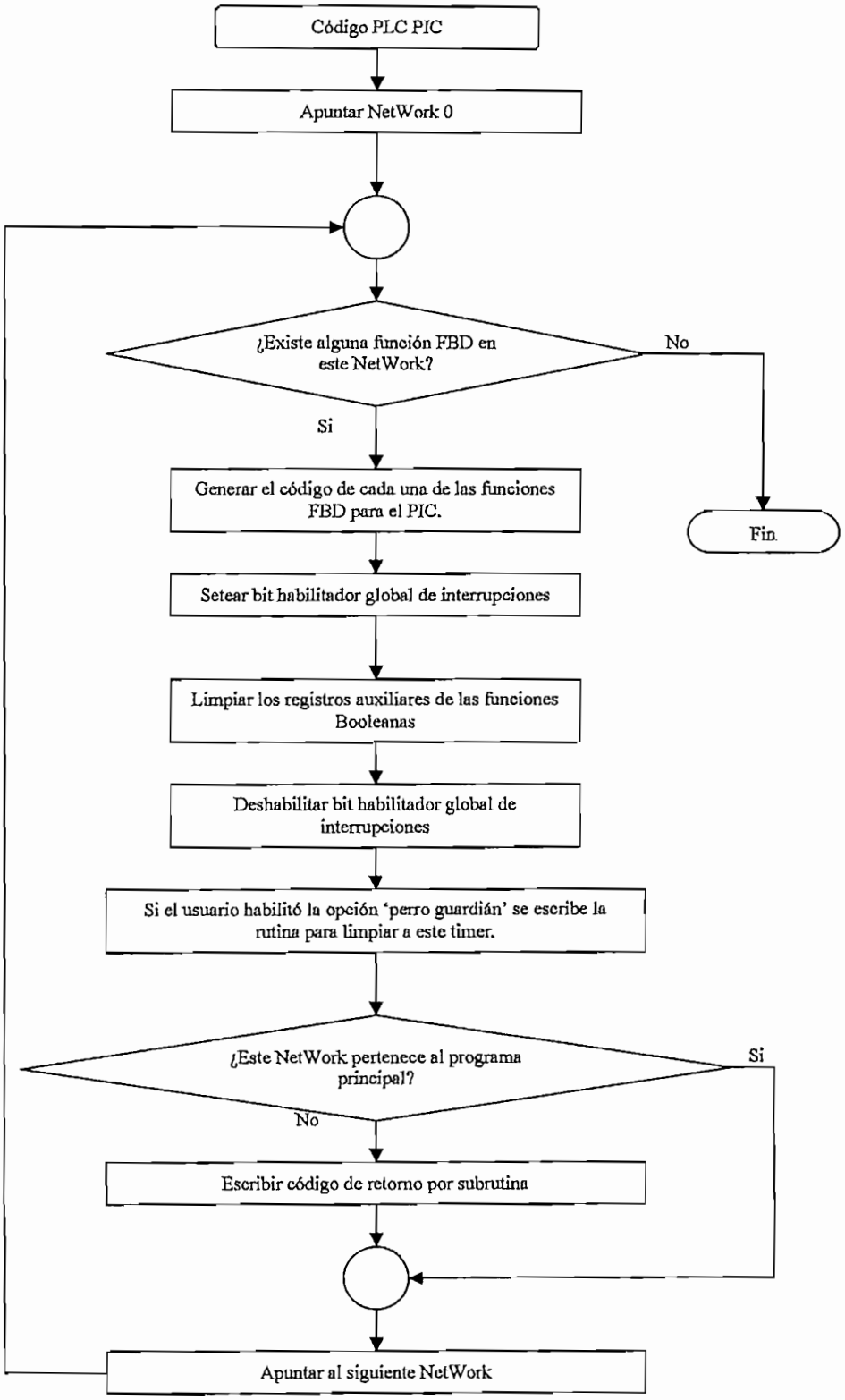


Figura 4.84. Diagrama de Flujo del Código que permite al PIC comportarse como PLC.

Generar Código para cada una de las funciones FBD.

Se lee función FBD para su código en la matriz denomina MemoriaFlash, en función de los algoritmos generados en SECCIÓN 4.3.2.

Si el código generado para cada función FBD alcanza dentro del espacio de Página de memoria de programa entonces:

Se escribe código en matriz MemoriaFlash.

Caso contrario

Se llena el espacio sobrante que queda en la página actual de memoria de programa con instrucciones NOP.

Se escribe el código necesario para cambio de página de memoria de programa.

Se escribe código en matriz MemoriaFlash (que ahora está en otra página de memoria).

Fin Tarea.

Setear bit habilitador de interrupciones.

Si pone a 1 lógico el BIT GIE del registro Status, que permite se genere cualquier interrupción en el PIC:

Fin Tarea

Limpiar registros auxiliares.

Se escribe el código para limpiar los registros 20h y 21h de la memoria RAM en el banco 0 que son los registros auxiliares de las funciones booleanas.

Fin Tarea

Una vez generado el código que se desea que contenga el microcontrolador PIC (dicho código está en la matriz MemoriaFlash), depende si el usuario desea transferirlo por medio del puerto paralelo o generar el código *.hex para programar al PIC con otros medios.

En conclusión no todas son ventajas al usar un lenguaje de alto nivel, ya que el tamaño del código hexadecimal dentro del microcontrolador suele ser una proporción más grande que el código de un programa desarrollado en lenguaje assembler, con lo cual el tiempo de ejecución del algoritmo de control se prolonga, aunque esto está compensado pues la memoria de programa del microcontrolador es de un tamaño suficiente para el desarrollo de aplicaciones. Las funciones que se desarrollaron en este software talvez no sean todas las necesarias para la elaboración de proyectos con el PLC. Sin embargo, se cree que estas son suficientes para simular el comportamiento del PIC como PLC y el programa puede ser mejorado en una versión superior a esta.

CAPÍTULO 5

PRUEBAS Y RESULTADOS

5.1 PRUEBAS

Con el fin de demostrar que el PLC construido en el presente proyecto tiene un funcionamiento correcto, fue necesario concebir ejemplos de posibles proyectos de automatización con los cuales probar al mismo. Este capítulo se orienta en este sentido, y trata desde los ejemplos más básicos como el mando memorizado hasta circuitos de control más complejos.

Cada uno de setos circuitos de prueba fueron programados y probados, siguiendo el orden que se muestra a continuación:

- Elaboración y Simulación de los ejemplos.
- Generación del código hexadecimal y descarga de dicho código al PLC a través de un programador de PICs.
- Descarga al PLC desde la interfase de programación del software desarrollado en este proyecto.
- Pruebas con el hardware.

CIRCUITO No. 1: Mando Memorizado. Figura 5.1

Este circuito consiste en accionar un motor M1 desde un pulsador P1 y apagar dicho motor desde otro pulsador P2.

Variables	Tipo de Variable	Bit en el PLC
P1	IN	RD0

P2	IN	RD1
M1	OUT	RB0

Tabla 5.1. Resumen de las variables utilizadas.

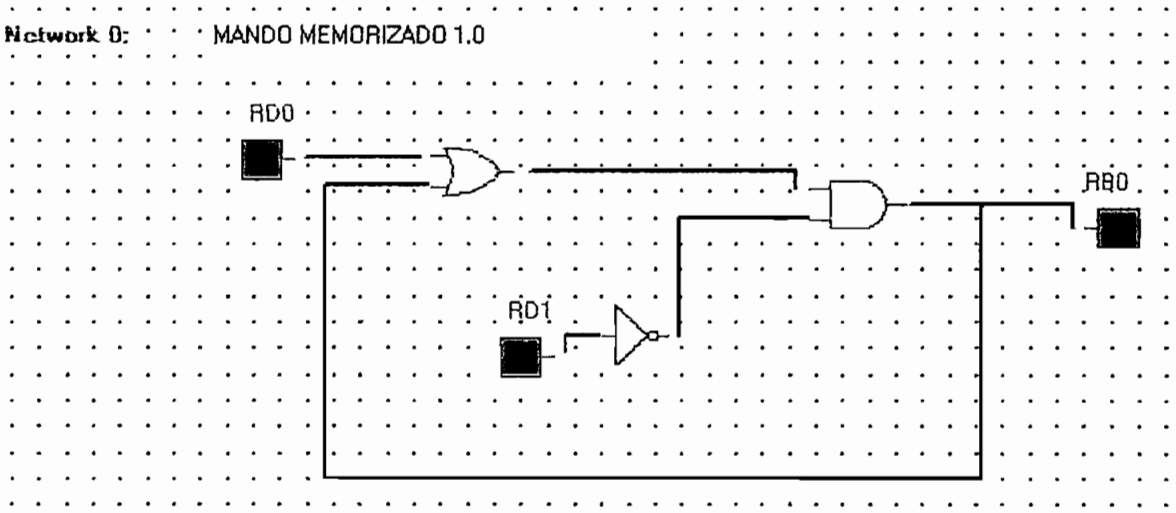


Figura 5.1 Mando Memorizado

CIRCUITO No. 2: Secuencia de encendido correcto de salidas H1, H2 y H3 pulsando en el siguiente orden los pulsadores C1, C2 y C3. Apagado H3, H2 y H1 según C4, C5 y C6. Figura 5.2.

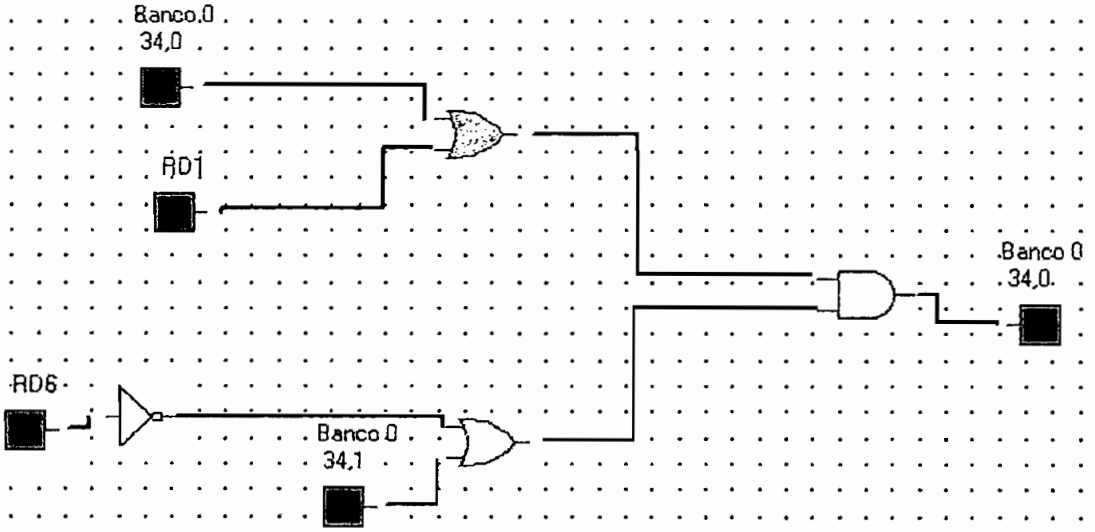
En este circuito se debe presionar correctamente la secuencia para el encendido, caso contrario no se prenden H1, H2 y H3. Una vez prendidos también hay una secuencia para el apagado que es C4, C5 y C6.

Variables	Tipo de Variable	Bit en el PLC
C1	IN	RD1
C2	IN	RD2
C3	IN	RD3
C4	IN	RD4
C5	IN	RD5
C6	IN	RD6

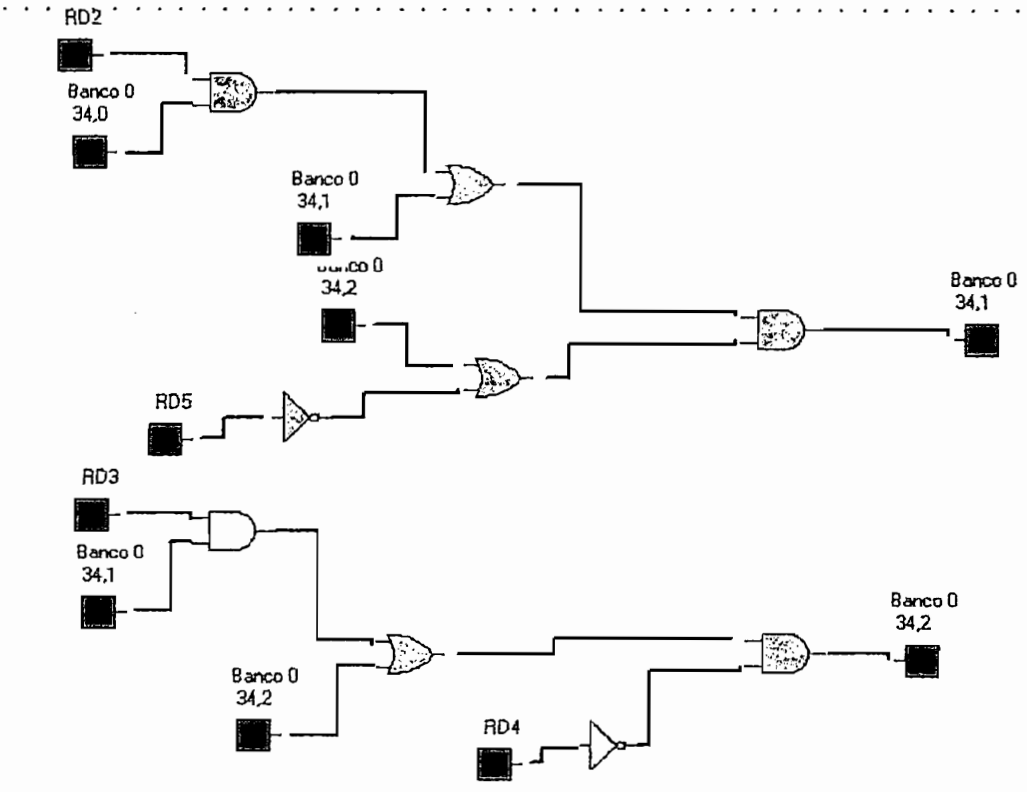
H3	OUT	RB1
H2	OUT	RB2
H1	OUT	RB3

Tabla 5.2. Tipos de variables e indicación de su uso en el PLC.

Network 0



Network 1



Network 2

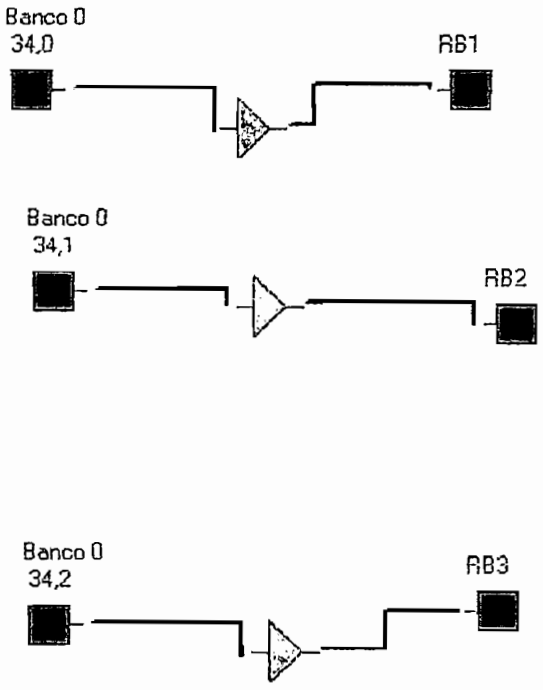


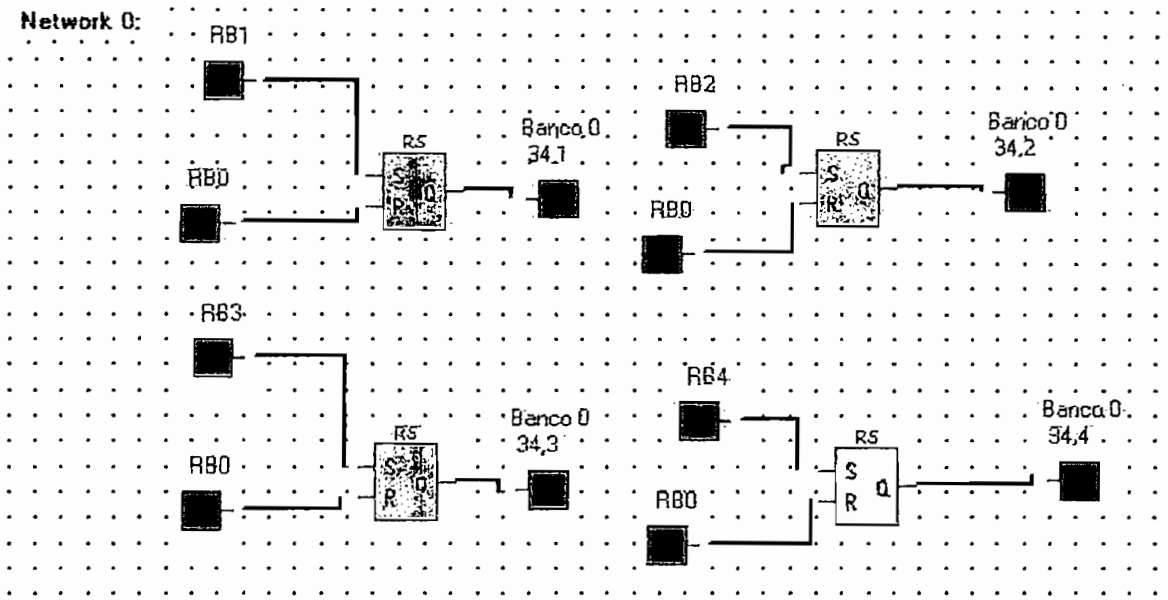
Figura 5.2. Ejemplo de una secuencia de encendido.

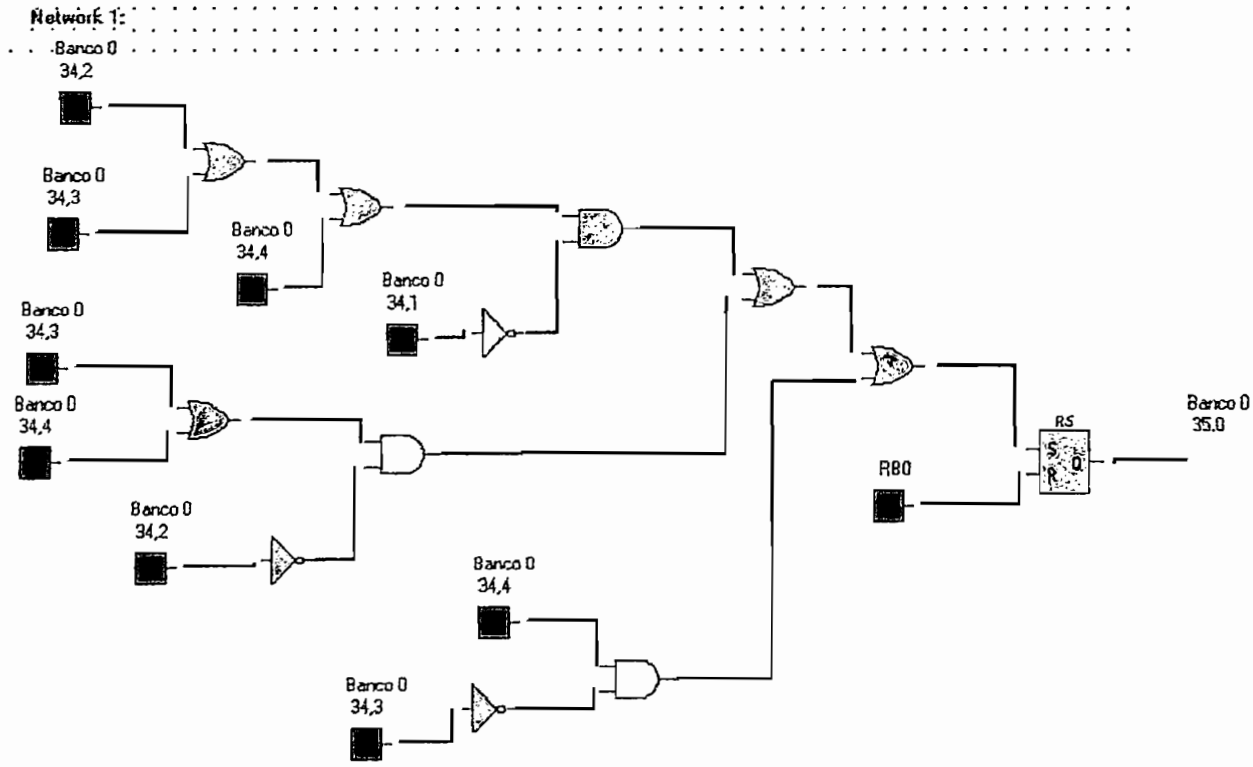
CIRCUITO No. 3: Advertencia de secuencia de encendido incorrecto. Figura 5.3.

El siguiente circuito es para controlar una maquinaria accionando en el orden correcto 3 pulsadores, en caso de una secuencia incorrecta se encenderá una alarma. Para apagar la maquinaria y la alarma, existe un pulsante de reset.

Variables	Tipo de Variable	Bit en el PLC
C1	IN	RB1
C2	IN	RB2
C3	IN	RB3
Reset	IN	RB0
M	Out	RD1
Alarma	Out	RD0

Tabla 5.3. Resumen de variables utilizadas.


Network 0:
RB1
RB0
RS
Banco 0
34,1
RB2
RB0
RS
Banco 0
34,2
RB3
RB0
RS
Banco 0
34,3
RB4
RB0
RS
Banco 0
34,4


Network 1:
Banco 0
34,2
Banco 0
34,3
Banco 0
34,4
Banco 0
34,1
Banco 0
34,3
Banco 0
34,4
Banco 0
34,2
Banco 0
34,4
Banco 0
34,3
RS
RB0
Banco 0
35.0

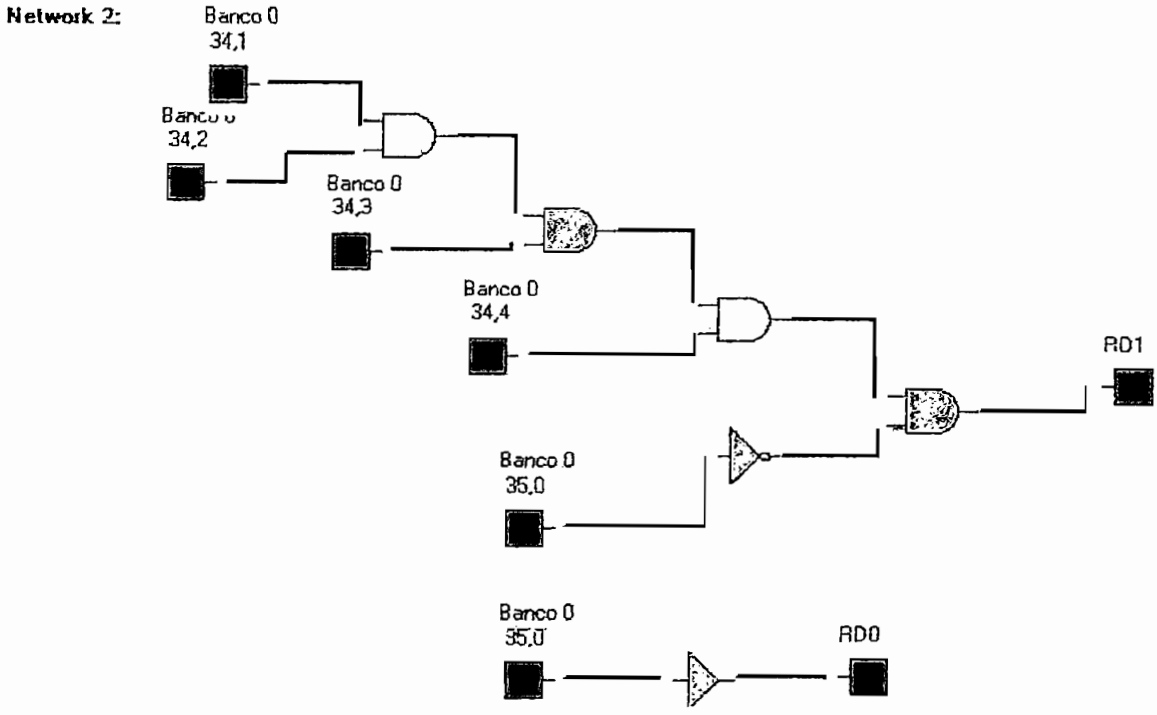


Figura 5.3. Circuito con alarma de advertencia en encendido incorrecto.

CIRCUITO No. 4; Comprobación del Timer TON. Figura 5.4.

En este circuito se habilita uno de los ocho timers Ton, con el bit 0 del puerto A, mientras que el RB0, es la salida del timer.

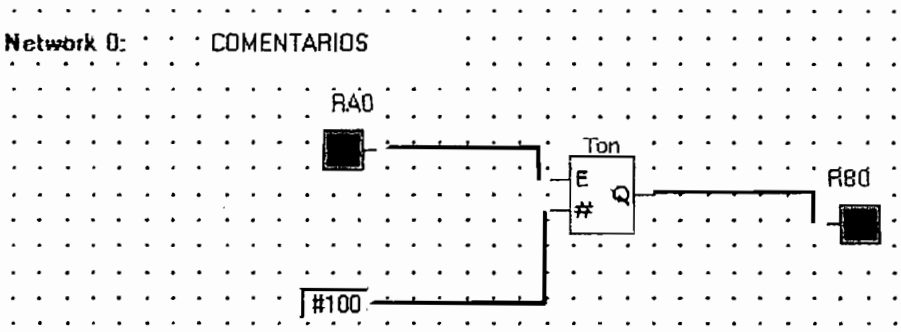


Figura 5.4. Funcionamiento de un temporizador Ton.

CIRCUITO No.5: Timer Toff. Figura 5.5.

En este circuito se habilita uno de los ocho timers Toff, con el bit 0 del puerto D, mientras que el RB0, es la salida del timer.

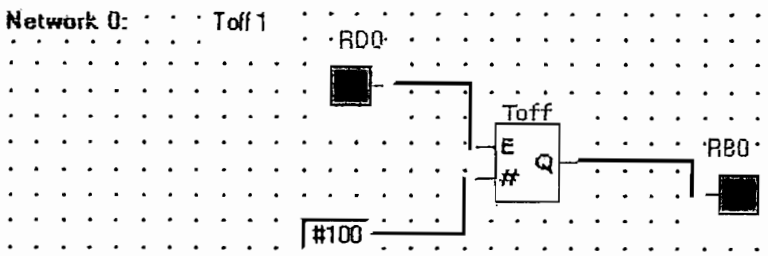


Figura 5.5. Funcionamiento de un temporizador Toff.

CIRCUITO No. 6: Oscilador 1. Figura 5.6.

En esta parte se construye un circuito de temporización cíclico, empleando dos temporizadores Ton.

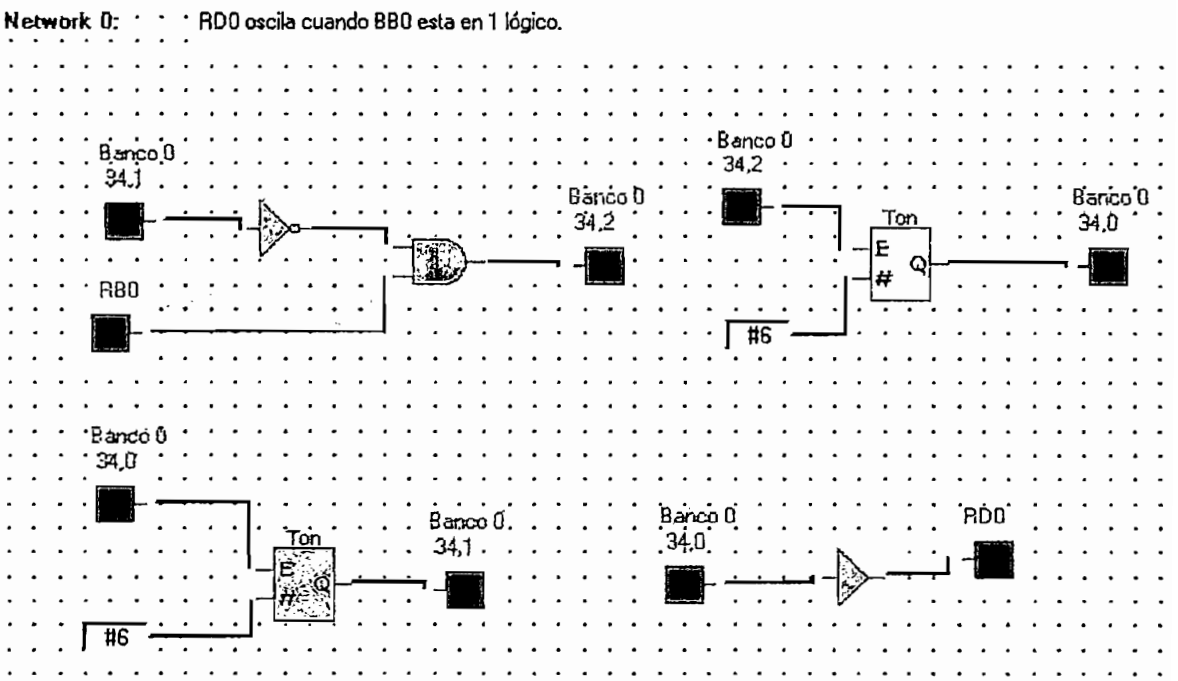


Figura 5.6. Ejemplo 1 de un oscilador.

CIRCUITO No. 7: Oscilador 2. Figura 5.7.

En esta parte se construye un circuito de temporización cíclico similar al anterior, empleando dos temporizadores un Ton y otro Toff.

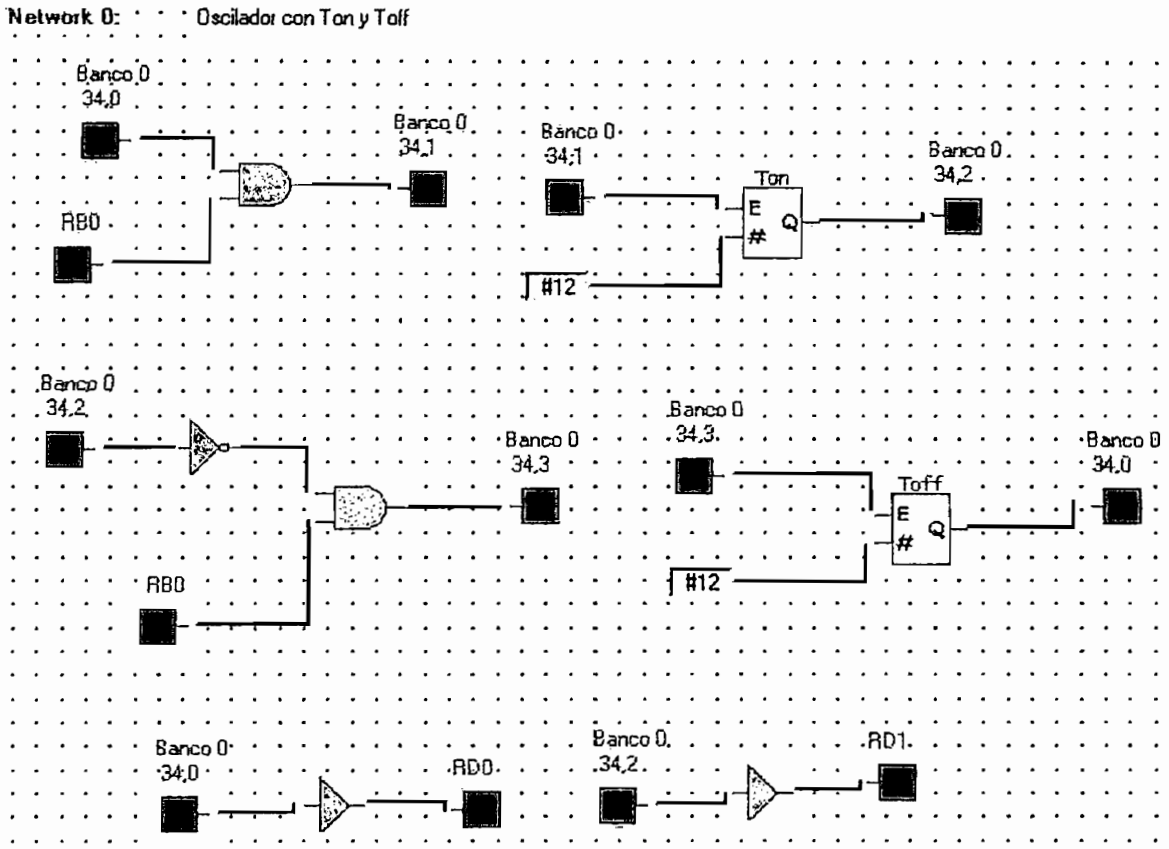


Figura 5.7. Ejemplo 2 de un oscilador.

CIRCUITO No. 8: Incremento y decremento del PWM. Figura 5.8.

Una característica importante con la que cuenta este PLC es las dos salidas PWM. Para este ejemplo se controla la velocidad de un motor mediante uno de los PWMs. Para incrementar el ancho del pulso de la señal PWM se utiliza el bit RB0, mientras que para decrementar este se usa el bit RB1, con esto se consigue variar la velocidad de un motor DC; sin embargo, al decrementar el ancho del pulso una vez después de cero, el PWM invierte su valor al máximo. Para lo cual se necesita hacer un circuito que evite dicho desbordamiento tanto superior e inferior.

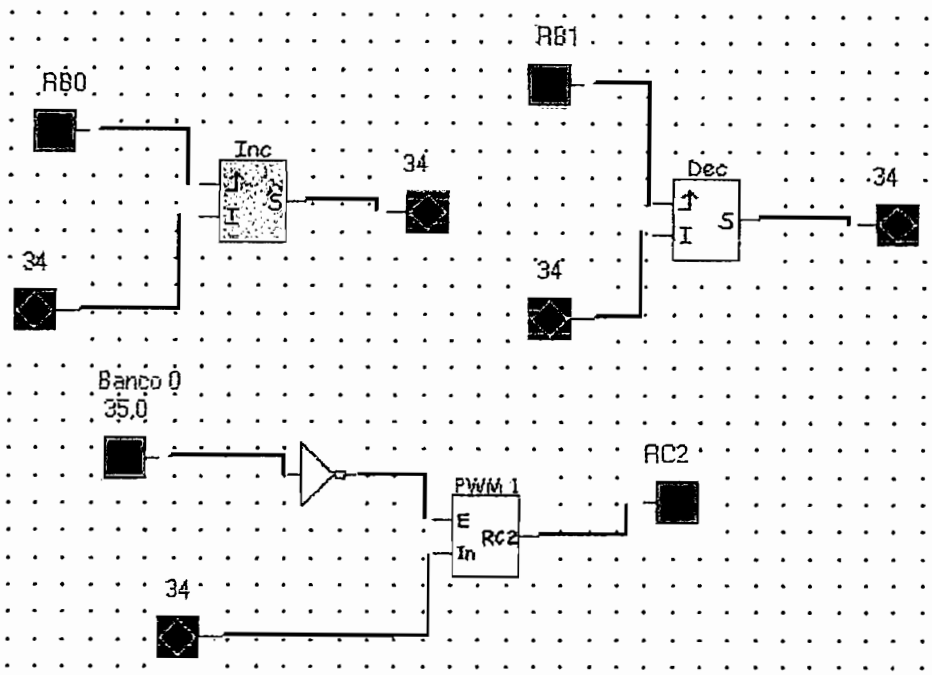


Figura 5.8. Circuito de incremento y decremento del PWM.

CIRCUITO No. 9: Incremento y decremento del PWM, controlado. Figura 5.9.

En este circuito se usa el RA2 para habilitar el PWM (como paro de emergencia) mientras que para incrementar el ancho del pulso de la señal PWM se utiliza el bit RA1, y para decrementar este se usa el bit RA0, con esto se consigue variar la velocidad de un motor DC.

La ventaja de este circuito respecto del anterior es el control de los límites inferior y superior del PWM, que evitan que cuando el ancho de pulso este en lo mínimo y se decida decrementar una vez más, este pase al valor máximo; o viceversa, cuando el ancho de pulso esté en lo máximo, pase a lo mínimo de manera brusca.

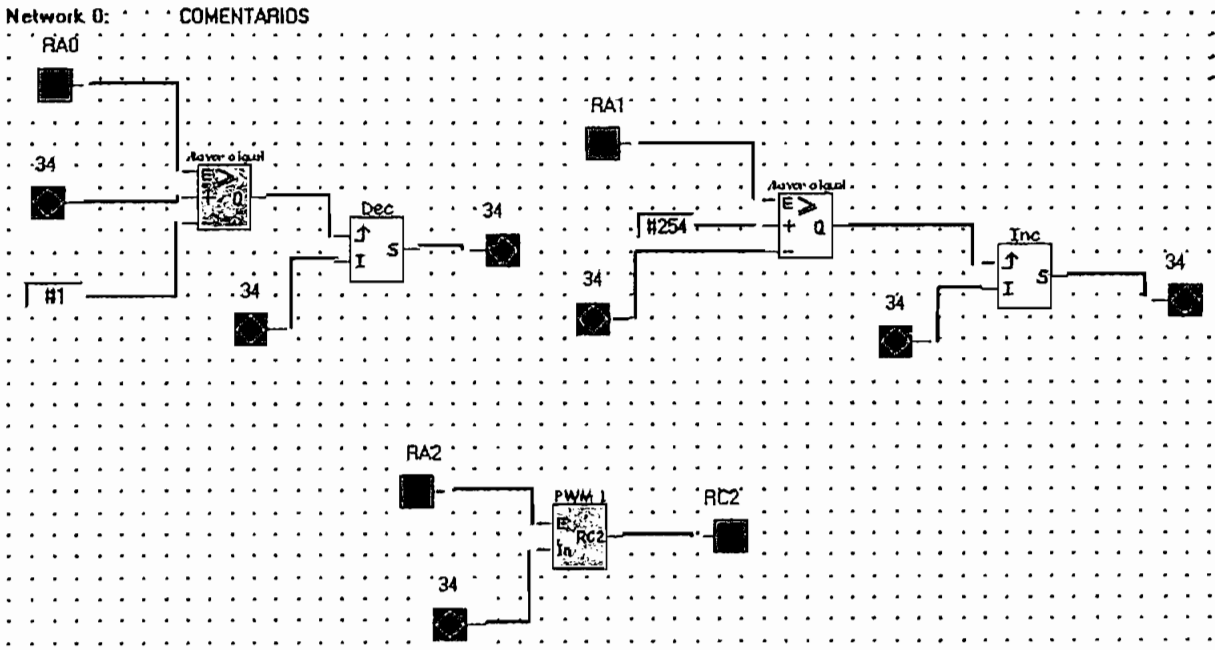


Figura 5.9. Incremento y decremento controlado del PWM.

CIRCUITO No. 10: Conversor AD. Figura 5.10.

En este circuito se usa el RB0 para habilitar al conversor AD, el canal que lee la señal analógica es el canal 0 correspondiente al terminal RA0, está señal es digitalizada a 8 bits y enviada al puerto D.

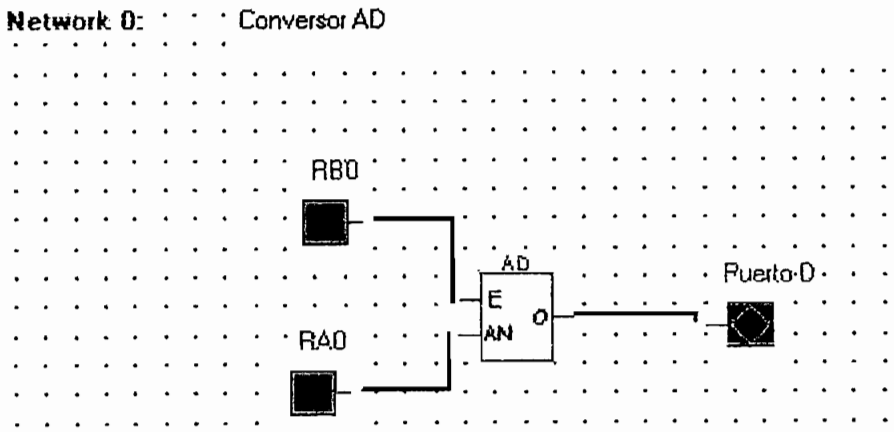


Figura 5.10. Comprobación del conversor AD.

CIRCUITO No. 11: Control de Histéresis para un Motor. Figura 5.11.

El siguiente circuito tiene como objetivo mantener la velocidad de un motor entre dos niveles. Cuando la velocidad del motor está por debajo del nivel mínimo, se incrementa el ancho de pulso de una salida PWM para aumentar la velocidad. Si la velocidad sobrepasa el valor máximo se decrementa el ancho de pulso de la salida PWM para disminuir la velocidad. Y si la velocidad del motor esta dentro del nivel mínimo y máximo, el PWM no es alterado. En la Figura 5.11 se representa un esquema simple de cómo el PLC controla al motor y sensa su velocidad.

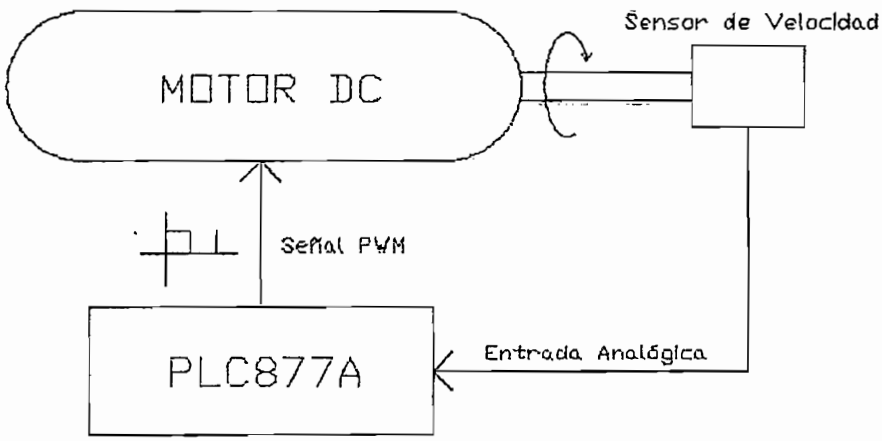


Figura 5.11. Diagrama de Control para un Motor DC.

El programa de control para el PLC debe tener una salida PWM y un entrada AD. Como las funciones de incremento y decremento solo funcionan con un flanco de subida en su terminal de habilitación 'E', entonces se necesita un temporizador cíclico para que constantemente habilite o deshabilite esta función y se pueda incrementar o decrementar el ancho de pulso de la señal PWM. Para conocer si la señal analógica está entre los límites superior o inferior se utiliza la función de comparación. Se tiene además un pulsador en el terminal RB0 para que el sistema funcione. Calibrando a los temporizadores TON que conforman el temporizador cíclico, se ajusta la velocidad de incremento o decremento de la salida PWM.

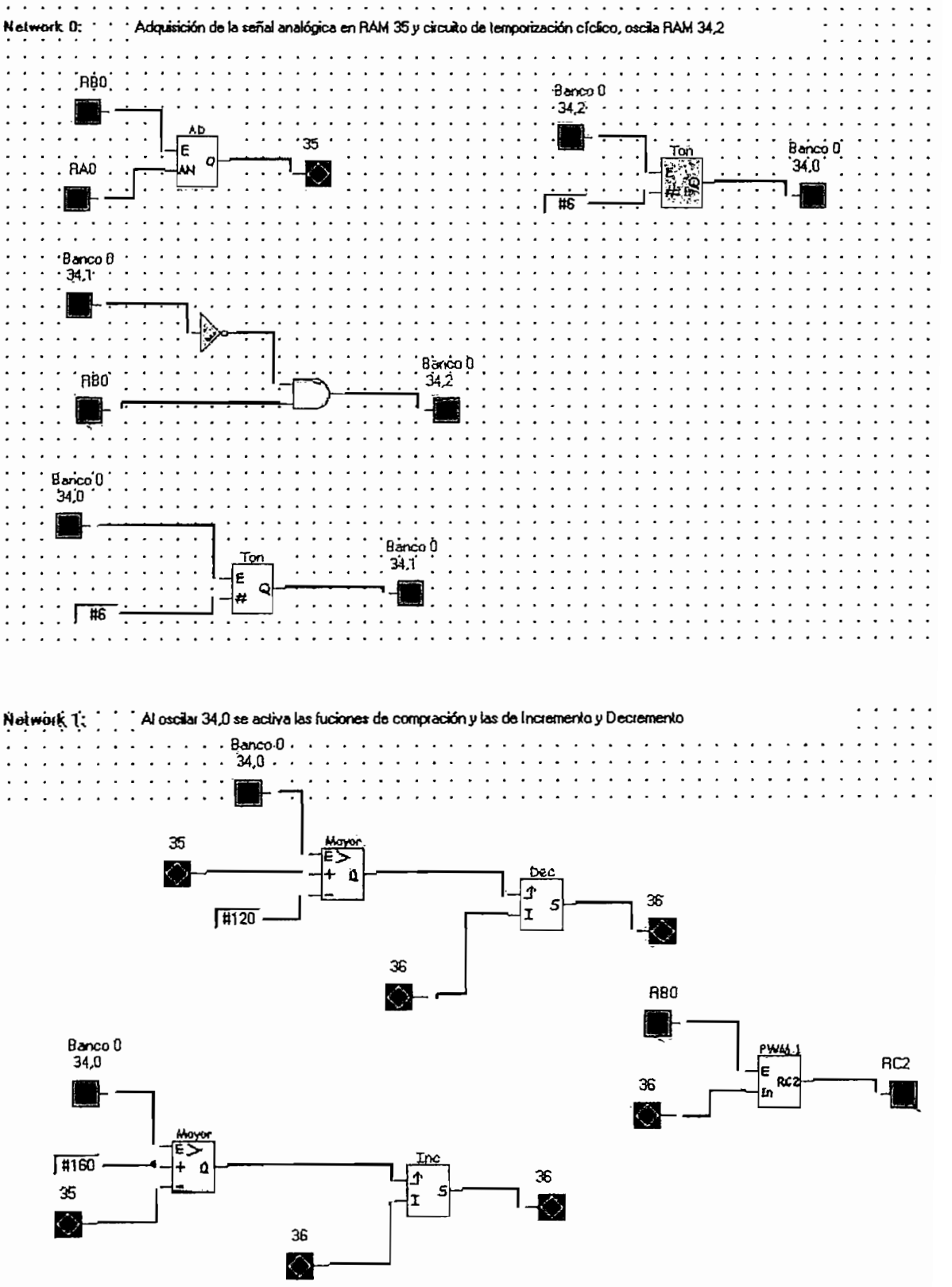


Figura 5.12. Control de Histéresis para un Motor DC.

CIRCUITO No. 12: Comunicación serial. Figura 5.12.

El siguiente programa permite recibir un número de 0 a 255 por el puerto serial RS232 y transmitirlo nuevamente. El programa principal no realiza ninguna actividad eficaz. Pero cuando se produce la interrupción por recepción y el terminal de entrada RB0 está en alto, se almacena lo que contiene el buffer de recepción en una posición de la memoria RAM y el contenido de dicha posición se lo carga en el buffer de transmisión.

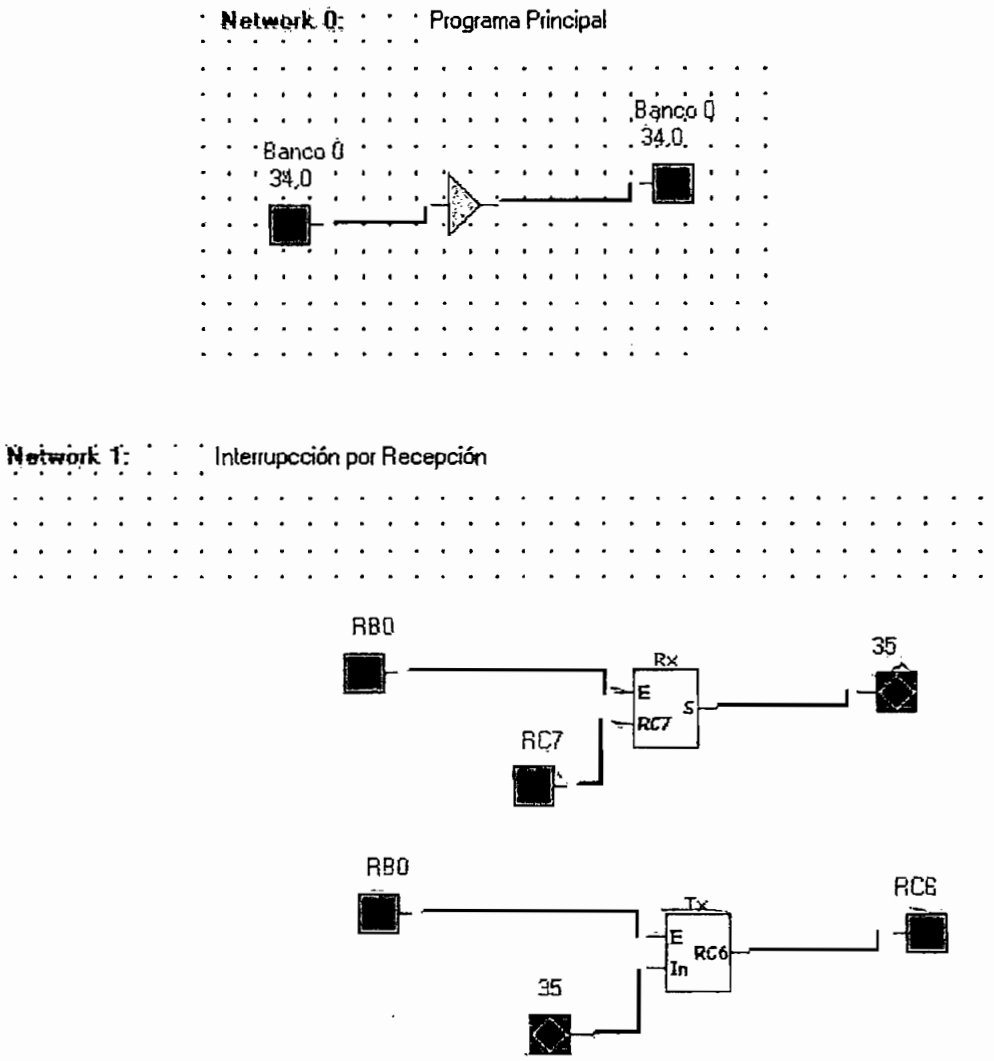


Figura 5.13. Comunicación serial RS232 con rebote.

CIRCUITO No. 13: Manejo de la señal PWM desde un PC a través del Puerto Serial RS232. Figura 5.13.

El siguiente programa permite enviar un número de 0 a 255 por el puerto serial, este valor corresponde al ancho de pulso de la señal PWM. El objetivo de este ejemplo es el control de la velocidad de un motor desde la computadora. La función PWM, está habilitada siempre mediante el bit 0 de la posición 34 de la RAM.

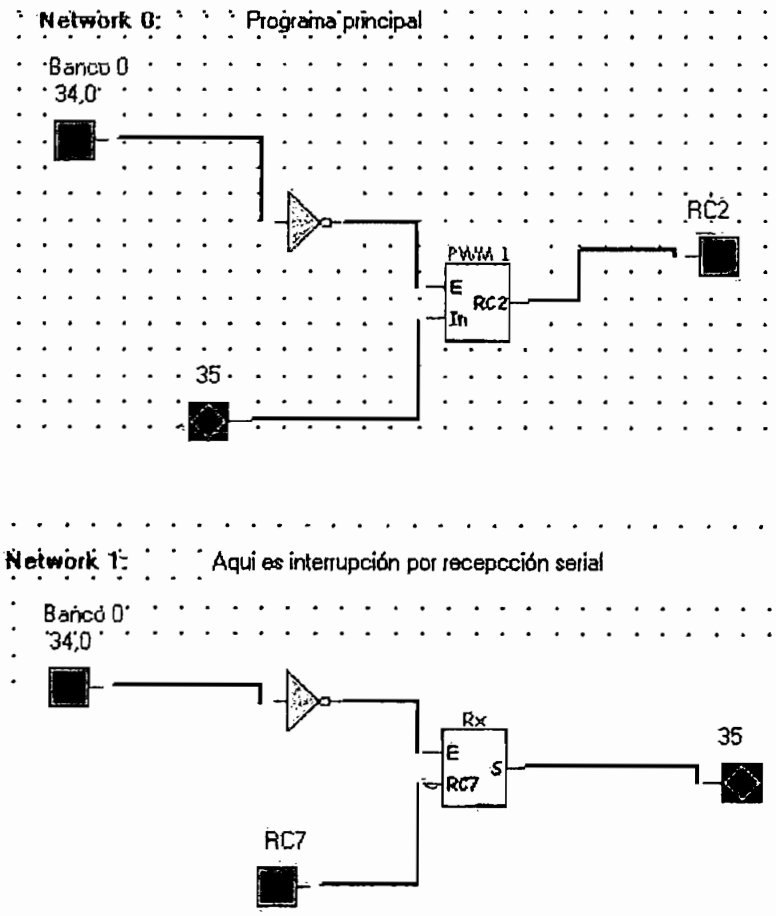


Figura 5.14. Control del PWM desde el puerto serial.

CIRCUITO No. 14: Almacenar información en la memoria EEPROM de datos y lectura de la misma. Figura 5.14.

Cuando el terminal RA0 esté en alto, el contenido del puerto B pasa directamente a almacenarse en el byte 1 de la memoria EEPROM. Mientras que cuando el

terminal RA1 esté en alto el contenido del byte 1 de la memoria EEPROM pasa al puerto D. La memoria EEPROM no es volátil, por lo que al apagar el PLC y después encenderlo, se recupera la información que poseía el byte 1 de la memoria EEPROM.

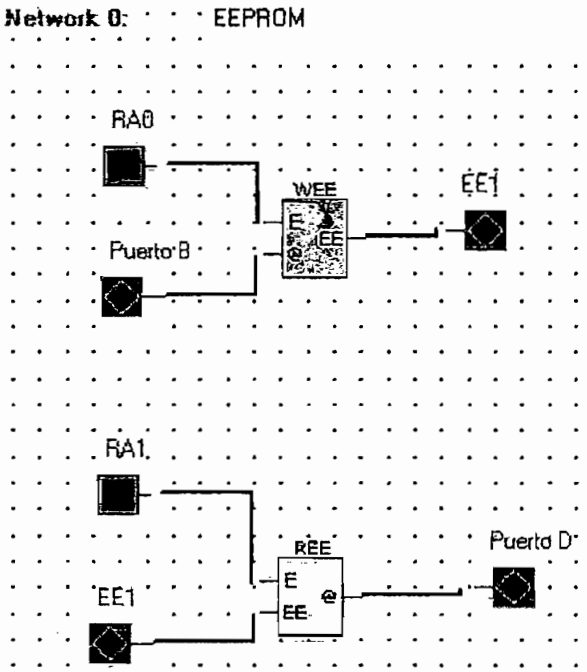
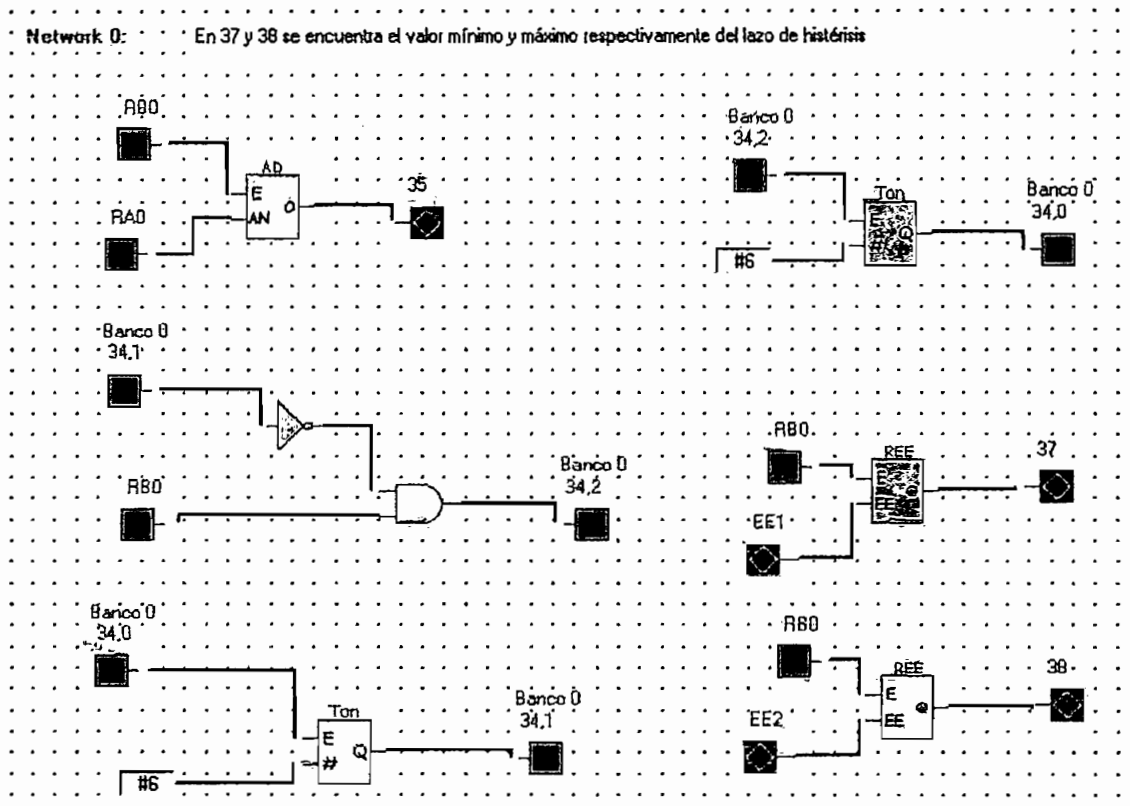
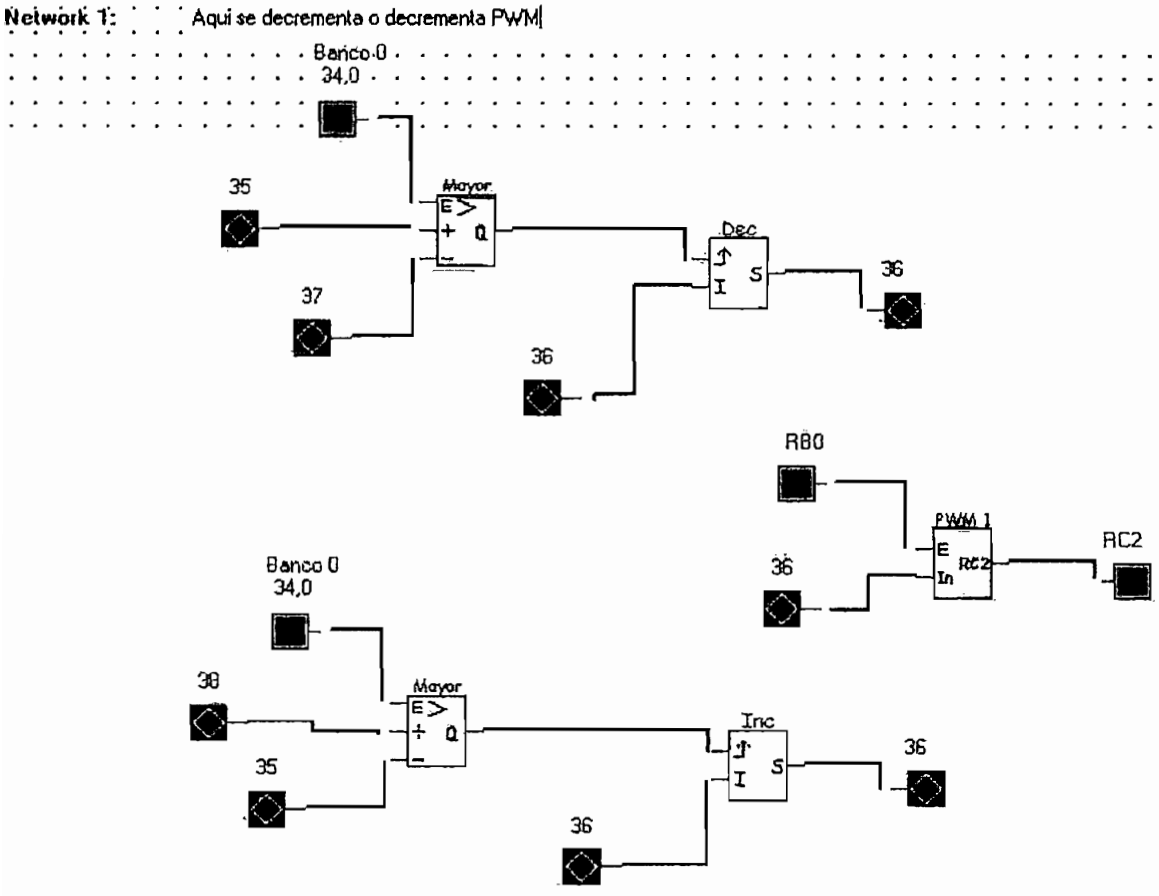


Figura 5.15. Manejo de la memoria EEPROM.

CIRCUITO No. 15: Control de Histéresis desde el computador para un Motor, con calibración de los valores mínimo y máximo, que se encuentran almacenados en la memoria EEPROM. Figura 5.15.

Este programa es muy similar al del CIRCUITO 10, con la diferencia que los valores mínimo y máximo están almacenados en dos posiciones de la memoria EEPROM. Dichos valores se pueden alterar desde el computador por comunicación serial RS232. Para probar este programa se debe enviar desde el computador dos bytes consecutivamente (valor máximo y valor mínimo del control por histéresis) con estos se altera las posiciones 1 y 2 de la EEPROM. El programa principal está en el NetWork 0 y 1; y la subrutina por recepción serial está en el NetWork 2.


Network 0:
En 37 y 38 se encuentra el valor mínimo y máximo respectivamente del lazo de histérisis
RB0
RA0
AD
E
AN
0
35
Banco 0
34,2
Ton
E
#
Q
#6
Banco 0
34,0
Banco 0
34,1
RB0
Banco 0
34,2
RB0
REE
E
EE
37
EE1
Banco 0
34,0
Ton
E
#
Q
#6
Banco 0
34,1
RB0
REE
E
EE
38
EE2


Network 1:
Aquí se decrementa o decrementa PWM
Banco 0
34,0
35
Mayor
E >
+ Q
-
37
Dec
S
I
36
36
RB0
PWM 1
E
In
RC2
RC2
36
Banco 0
34,0
38
Mayor
E >
+ Q
-
35
Inc
S
I
36
36

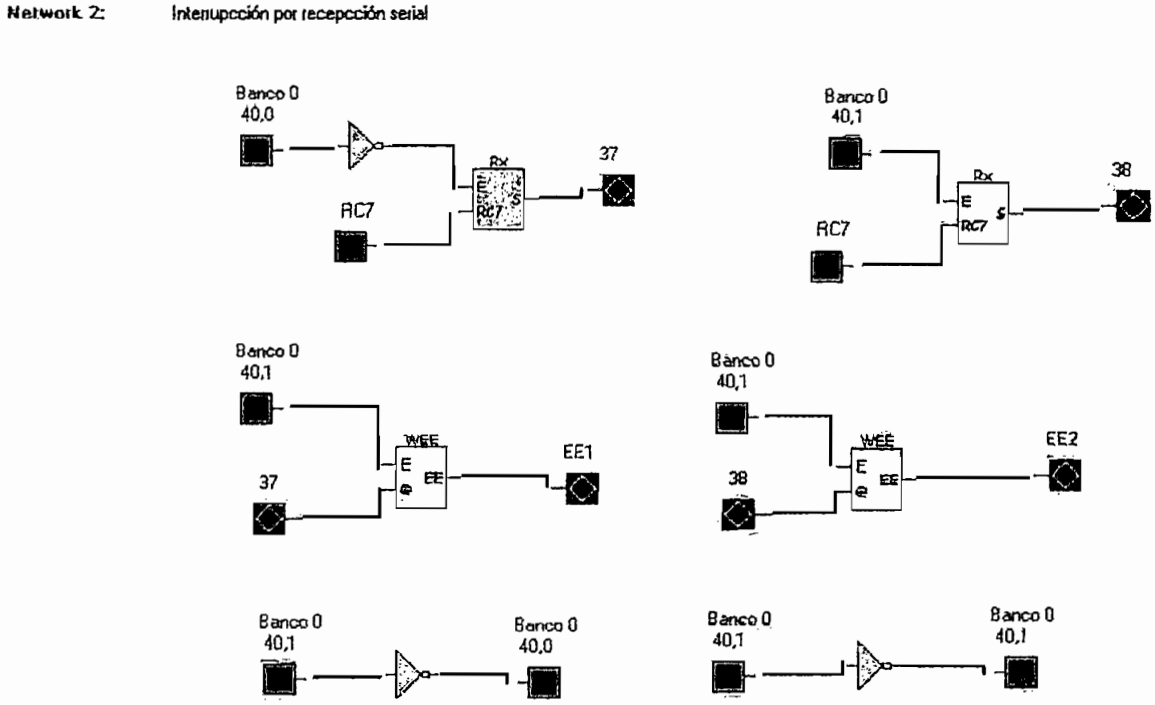
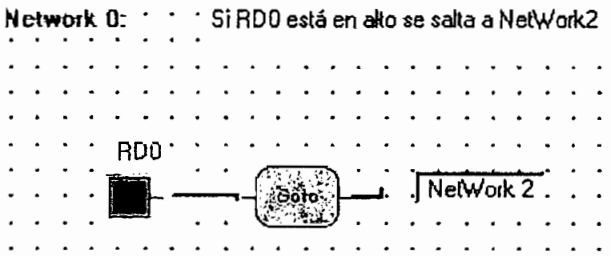


Figura 5.16. Ejemplo de control para un motor DC, mediante el puerto serial y con calibración mediante la EEPROM.

CIRCUITO No. 16: Utilización del comando 'GoTo'. Figura 5.16.

En este programa se controla el flujo del programa por medio del terminal RD0, si este bit está en alto se ejecuta el NetWork 0 y el NetWork 2, caso contrario se ejecuta el NetWork 0 y el NetWork 1, pero si RD1 esta en alto el programa salta al NetWork 0, sin ejecutar el NetWork 2.



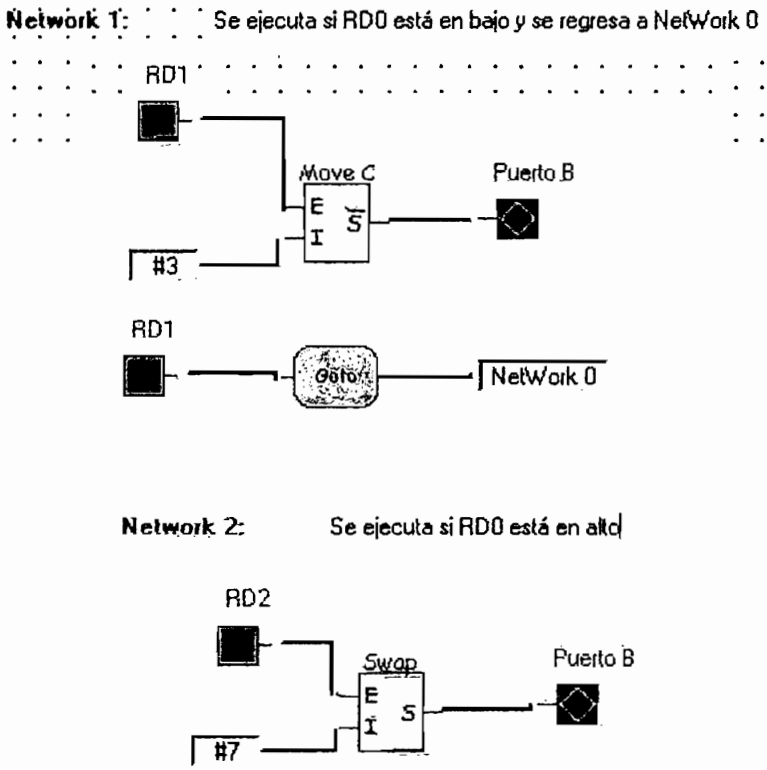
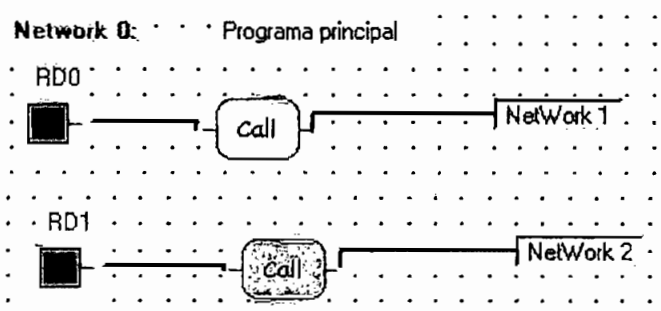


Figura 5.17. Ejemplo de la Función Goto.

CIRCUITO No. 17: Utilización del comando 'Call'. Figura 5.17.

En este programa se llama a otros NetWorks como subrutinas, el NetWork 0 es el programa principal mientras que el NetWork 1 y el NetWork 2 son subrutinas. Para llamar a la subrutina 1 (NetWork 1), RD0 debe estar en alto y para llamar a la subrutina 2 (NetWork 2), RD1 debe estar en alto. Hay que indicar que cuando se ejecuta una subrutina, el flujo del programa, al final de ésta regresa al punto desde donde se la llamo.



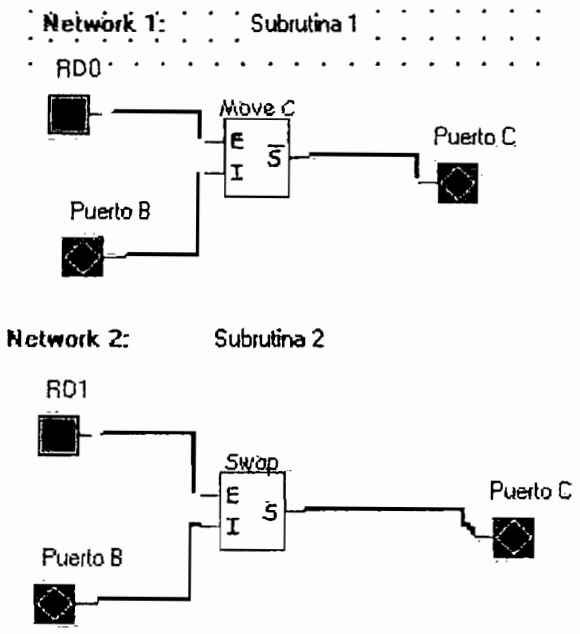


Figura 5.18. Utilización de la Función Call.

Todos estos circuitos fueron programados en la HMI del sistema. Luego fueron bajados al PLC, donde se los corrió con resultados totalmente correctos. De hecho, sería interesante que estos programas se los emplee en prácticas de laboratorio para que los estudiantes verifiquen la validez de los mismos así como del PLC como tal.

5.2 RESULTADOS

Los resultados que arrojan las pruebas se centran solamente alrededor de los estados lógicos de las salidas digitales que se usan en el programa que el usuario haya elaborado. Mientras que las salidas PWM y las entradas analógicas son características propias del microcontrolador utilizado; por lo mismo, su comportamiento se rige por la configuración que el usuario haya elegido. Sin embargo, los únicos resultados que se pueden analizar y comparar son los de los Timers Ton y Toff puesto que estos no posee el microcontrolador sino que se añadieron como funciones del PLC de este proyecto.

Para la función Ton y función Toff se probaron varios valores de tiempo, siendo la base mínima de tiempo para estos timers de 131,1 ms. Como ya se explicó anteriormente el número que se conecta en el terminal '#' es el valor que debe multiplicarse por la base, los resultados que se obtuvieron para diferentes tiempos están tabulados en la Tabla 5.1.

Número en el terminal '#'	Tiempo ideal generado [s]	Tiempo real generado [s]	|Error %|
10	1.31	1.36	3.82
20	2.62	2.72	3.81
30	3.93	4.0	1.78
40	5.24	5.34	1.90
100	13.11	13.06	0.38
200	26.21	26.40	0.72
255	33.42	33.25	0.51

Tabla 5.1. Tabulación de los resultados de los Timers.

Como se puede ver en los resultados de la Tabla 5.1 anterior, los porcentajes de errores obtenidos (que por cierto están expresados como valor absoluto), son aceptables ya que van desde un rango de 3.82 a 0.51. Estas pruebas que se hicieron son importantes para el usuario, ya que con estos resultados podrá predecir el comportamiento de su proyecto, sobretodo si el conteo de tiempo que necesita está en el rango de los milisegundos.

CAPÍTULO 6

COMPARACIÓN DE COSTOS

El presente capítulo tiene como objetivo cuantificar el valor económico del PLC diseñado y construido en este proyecto, además de comparar este costo respecto de los PLCs comerciales, cuyas características y funciones sean similares al diseñado.

Para una valoración más adecuada la comparación de costos se dividirá en dos partes: El costo del hardware y el costo del software.

6.1 COSTO DEL HARDWARE

El análisis económico en su primera parte, pretende obtener el costo de elaboración de las tarjetas, el costo de todos los elementos utilizados según el esquema presentado en la Figura 3.1 del Capítulo 3 y el costo de la estructura que contendrá a las tarjetas del PLC diseñado en el presente proyecto.

El esquema a seguir es el siguiente:

- CPU o Tarjeta Principal.
- Tarjeta de entradas digitales
- Tarjeta de salidas digitales
- Tarjetas de entradas analógicas.
- Tarjeta módulo especial
- Más la Tarjeta de Visualización, cuyo objetivo es permitir al usuario ver el estado de los bits de los puertos del PIC.

En las tablas que se encuentran a continuación, se tiene el costo de cada una de estas tarjetas, en estas se enumeran los elementos y la cantidad utilizada, su costo unitario (C. Unitario) y el costo total (C. Total).

TARJETA PRINCIPAL (CPU)

Elemento	Cantidad	C. Unitario	C. Total
Capacitor Cerámico	5	0,12	0,60
Capacitor Polar 100uF	1	0,12	0,12
Capacitor Polar 10uF	1	0,06	0,06
Capacitor Polar 1uF	1	0,06	0,06
Capacitor Polar 220uF	1	0,12	0,12
Capacitor Polar 470uF	1	0,12	0,12
Conector RS232 Hembra	1	0,40	0,40
Cristal de 4MHz.	1	0,60	0,60
Diodo de Señal 1N4148	3	0,06	0,18
Diodo Led	2	0,10	0,20
Diodo Led azul	1	0,40	0,40
Diodo Rectificador 1N4007	1	0,05	0,05
Diodo Zener 24V	1	0,12	0,12
Espadines dobles 5x2	4	0,20	0,80
Espadines dobles 8x2	4	0,45	1,80
Fusible Pequeño	2	0,15	0,30
Microcontrolador PIC16F877A	1	14,00	14,00
Mini Pulsador	1	0,05	0,05
Portafusible pequeño	2	0,35	0,70
Regulador Integrado 7805	1	0,40	0,40
Regulador Integrado 7812	1	0,60	0,60
Resistencia de 1/4W	55	0,02	1,10
Switch (4 en 1, 2 posiciones)	1	0,80	0,80
Transistores 2N3904	2	0,08	0,16
Transistores 2N3906	4	0,05	0,20
Zócalo 7P	4	0,07	0,28
Zócalo Maquinado 20P	1	0,80	0,80
		TOTAL:	25,02

Tabla 6.1. Resumen de costos de la Tarjeta Principal.

Tarjeta de entradas digitales

Elemento	Cantidad	C. Unitario	C. Total
Borneras 2X1	5	0,30	1,50
Capacitor Polar 1uF	1	0,06	0,06
Capacitor Polar 22uF	9	0,06	0,54
Diodo Led	1	0,10	0,10
Diodo Rectificador 1N4007	9	0,05	0,45
Espadines dobles 8x2	4	0,45	1,80
Opto acopladores ECG 3045	9	1,79	16,11
Regulador Integrado 7805	1	0,40	0,40
Resistencia de 1/4W	19	0,02	0,38
Zócalo 3P	9	0,05	0,45
		TOTAL:	21,79

Tabla 6.2. Resumen de costos de la Tarjeta de Entradas digitales.

Tarjeta de salidas digitales

Elemento	Cantidad	C. Unitario	C. Total
Borneras 2 terminales	5	0,30	1,50
Capacitor Polar 22uF	1	0,06	0,06
Diodo de Señal 1N4148	1	0,06	0,06
Diodo Led	1	0,10	0,10
Driver ECG 2008	1	1,34	1,34
Espadines dobles 8x2	1	0,45	0,45
Fusible Pequeño	1	0,15	0,15
Relé de 24V	9	1,70	15,30
Resistencia de 1/2W	3	0,05	0,15
Transistor 2N3904	1	0,08	0,08
Zócalo 9P	1	0,10	0,10
		TOTAL:	19,29

Tabla 6.3. Resumen de costos de la Tarjeta de Salidas digitales.

Tarjetas de entradas analógicas.

Elemento	Cantidad	C. Unitario	C. Total
Borneras 2 terminales	5	0,30	1,50
Capacitor Cerámico	1	0,12	0,12
Capacitor Polar 100uF	1	0,12	0,12

Diodo Led	1	0,10	0,10
Diodo Zener 5V	8	0,15	1,20
Espadines dobles 8x2	1	0,45	0,45
Integrado LM324	2	0,50	1,00
Resistencia de 1/4W	34	0,02	0,68
Zócalo 7P	2	0,07	0,14
		TOTAL:	5,31

Tabla 6.4. Resumen de costos de la Tarjeta de Entradas Analógicas.

Tarjeta módulo especial.

Elemento	Cantidad	C. Unitario	C. Total
Bornera 3 Terminales	1	0,35	0,35
Capacitor Cerámico	1	0,12	0,12
Capacitor Polar 1uF	5	0,06	0,30
Capacitor Polar 1uF	1	0,06	0,06
Capacitor Polar 22uF	1	0,06	0,06
Conector DB9 Superficial	1	0,90	0,90
Diodo Led	1	0,10	0,10
Espadines dobles 8x2	1	0,45	0,45
MAX232	1	2,80	2,80
Regulador Integrado 7805	1	0,40	0,40
Resistencia de 1/4W	13	0,02	0,26
Transistor 2N3906	2	0,05	0,10
Transistor Darlington	2	0,40	0,80
Zócalo 8P	1	0,09	0,09
		TOTAL:	6,79

Tabla 6.5. Resumen de costos de la Tarjeta módulo especial.

Tarjeta de Visualización.

Elemento	Cantidad	C. Unitario	C. Total
Capacitor Cerámico	3	0,12	0,36
Capacitor Polar 10uF	1	0,06	0,06
Capacitor Polar 33uF	1	0,06	0,06
Diodo Led rojo de 3mm	33	0,06	1,98
Espadines dobles 8x2	4	0,45	1,80
Integrado 74LS04	4	0,40	1,60

Integrado LM324	2	0,50	1,00
Regulador Integrado 7805	1	0,40	0,40
Resistencia de 1/4W	41	0,02	0,82
Transistor 2N3904	3	0,08	0,24
Zócalo 7P	6	0,07	0,42
		TOTAL:	8,74

Tabla 6.6. Resumen de costos de la Tarjeta de Visualización.

TARJETAS	C. UNITARIO	C. TOTAL
CPU o Tarjeta Principal.	25,02	25,02
Tarjeta de entradas digitales x 2	21,79	43,58
Tarjeta de salidas digitales x 2	19,29	38,58
Tarjetas de entradas analógicas.	5,31	5,31
Tarjeta módulo especial	6,79	6,79
Tarjeta de Visualización	8,74	8,74
	TOTAL:	128,02

Tabla 6.7. Suma total de los costos de las tarjetas.

En la parte del hardware también se debe considerar la suma de los adicionales como:

Elemento	Cantidad	Unidades	C. Unitario	C. Total
Serigrafía	2			20,00
Alambre Tipo bus	4	metros	1,40	5,60
Alambre UTP CAT5	1,5	metros	0,26	0,39
Caja de acrílico 35x20x18 cm.	1		30,00	30,00
Circuitos Impresos	8			90,00
Conector DB25 + carcaza	1		1,20	1,20
Conector DB9 + carcaza	1		0,85	0,85
Conectores para espadines	8		0,90	7,20
Fuente Switching	1		50,00	50,00
Riel de aluminio	1,5	metros	1,30	1,95
			TOTAL:	207,19

Tabla 6.8. Resumen de costos de los adicionales.

El cálculo total del Hardware es la suma del costo de las tarjetas, más los adicionales, dando un total de $ 335,21.

6.2 COSTO DEL SOFTWARE

Para el cálculo del software es necesario considerar los siguientes puntos:
La elaboración de la HMI realizada en Visual Basic y cuyas funciones permiten diseñar, simular y descargar un proyecto de automatización hacia el PLC construido, se puede considerar como un trabajo a nivel Ingeniería. Por lo mismo, el costo de Hora Hombre (H.H.) es el que se especifica en la Tabla 6.9.

Es necesario también considerar el costo de la elaboración de los circuitos impresos, que se hizo en el Programa PCB. Para esta labor se consideró un costo de 4 dólares la Hora Hombre.

En la Tabla 6.9 se cuantifica también el costo de una Hora Equipo (H.E.), este costo es el que tiene una computadora por hora, que es de 0.10 dólares. Este valor se obtuvo de la siguiente manera: El costo de una computadora de medianas características actualmente es de 600 dólares, con un tiempo de funcionamiento normal de 5 años y considerando que por año se trabaja un total de 10 meses, 20 días cada mes y durante 5 horas por día, este cálculo por lo tanto da un costo de:

$$\left|\frac{\$\,600}{5\ años}\right|\left|\frac{1\ año}{10\ meses}\right|\left|\frac{1\ mes}{20\ días}\right|\left|\frac{1\ día}{5\ horas}\right| = \$\,0.10\,/\,hora$$

Recursos Utilizados	H.H.	Costo H.H.	Costo Total
Visual Basic	480	8	3840,00
PCB	15	4,00	60,00
	H. E.	Costo H.E.	Costo Total
Computadora	495	0,10	49,50
		TOTAL:	3949,50

Tabla 6.9. Costo del Software.

Sin embargo y como ningún análisis no está completo sin una comparación con un producto que tenga las mismas características, se hace necesario obtener el costo total que el PLC del presente proyecto, podría tener en el mercado. Para este propósito las consideraciones y cálculos son los siguientes:

- El costo total que se obtiene de la suma del costo del software más el costo del hardware es de 4284.71 dólares, sin incluir porcentajes adicionales.
- Se debe sumar un porcentaje de utilidad adecuada al hardware, pero no en el software ya que este se consideró como un trabajo de Ingeniería donde se incluyó hasta el desgaste de la computadora.
- Se debe agregar un porcentaje de Gastos de Ventas al costo total. Estos gastos son todos los pagos en se incurre, para efectuar la venta del producto. En estos están los pagos a vendedores, publicidad, arriendo del local de venta, entre otros.

No. Unid.	Costo Hardware	Costo Software	Hardware + Util. 20%	Costo Total + G.V. 15%	V.M.U.
1	335,21	3949,50	402,25	5004,51	5004,51
2	335,21	3949,50	402,25	5467,10	2733,55
3	335,21	3949,50	402,25	5929,69	1976,56
4	335,21	3949,50	402,25	6392,28	1598,07
5	335,21	3949,50	402,25	6854,87	1370,97
6	335,21	3949,50	402,25	7317,46	1219,58
7	335,21	3949,50	402,25	7780,05	1111,44
8	335,21	3949,50	402,25	8242,64	1030,33
9	335,21	3949,50	402,25	8705,23	967,25
10	335,21	3949,50	402,25	9167,82	916,78
20	335,21	3949,50	402,25	13793,72	689,69
30	335,21	3949,50	402,25	18419,62	613,99
40	335,21	3949,50	402,25	23045,52	576,14
50	335,21	3949,50	402,25	27671,42	553,43
60	335,21	3949,50	402,25	32297,31	538,29
70	335,21	3949,50	402,25	36923,21	527,47
80	335,21	3949,50	402,25	41549,11	519,36
90	335,21	3949,50	402,25	46175,01	513,06
100	335,21	3949,50	402,25	50800,91	508,01

Tabla 6.10. Valor Unitario del PLC en el Mercado.

Como se puede ver en la Tabla 6.10, se calculó que debía hacerse una proyección de venta de 100 unidades, para que este producto pueda compararse con otros ya existentes en el mercado. La fórmula general que abarca estos cálculos es la siguiente:

$$V.M.U. = \frac{\{[(H \times No.Unidades) + U] + S\} + G.V.}{No.\,Unidades} \qquad E.C.\ 6.1$$

Donde:

V.M.U.= Valor de Mercado Unitario.

H= Costo del Hardware.

U= Porcentaje de Utilidad, que en este caso es del 20%.

S= Costo del Software.

G.V.= Gastos de Venta, que es del 15%.

6.3 COMPARACIÓN CON EL COSTO DE OTROS PLCs

La primera comparación se hace con dos PLCs cuyo precio y características se encontró en Internet. Estos PLCs que se muestran en las Tablas 6.11 y 6.12 reúnen algunas de las características del PLC desarrollado en esta Tesis.

	MicroLogix 1000 Controller
Entradas	12
Salidas	8, 2.5 Amp, Relay
Memoria	735 líneas de programa
Velocidad	500 instrucciones en 1.56 ms.
Timers	40
Fuente de Poder	16 Vdc
Puerto Serial	1
Programación	Ladder
PRECIO	120

Tabla 6.11. Características del Micrologix 1000

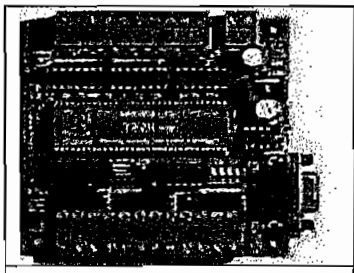
	**H-SERIES** **Original and classic PLCs**	
	T28H RELAY	**T40H RELAY**
Entradas	16	24 optoacopladas
Salidas	12, 1 Amp, Relay	15, 1 Amp, Relay
Memoria	400 líneas de programa	400 líneas de programa
Velocidad	12 µs/instrucción	12 µs/instrucción
Timers	20	20
Fuente de Poder	12 Vdc	12 Vdc
Puerto Serial	1	1
Programación	Ladder	Ladder
PRECIO	229	269

Tabla 6.12. Ejemplo del Costo de un PLC de los fabricantes LatinTech.

Como se puede ver en las tablas anteriores, estos costos son inferiores al del PLC diseñado y construido en esta Tesis que asciende a $508; sin embargo, esta comparación no es tan justa ya que los componentes que se adquirieron para el PLC construido aquí son a precio unitario; mientras que los fabricantes de PLCs comerciales compran al mayor los materiales, y poseen una capacidad instalada para el desarrollo de nuevos equipos tecnológicos.

La segunda comparación que se hizo con los costos de PLCs a nivel local. El precio de los PLCs consultados se encuentra en la Tabla 6.13.

Marca de PLC	Entradas/ Salidas	Costo en dólares
Siemens	16	600
Allen Bradley	14	480
Telemecanique	10	250

Tabla 6.13. Costo de PLCs comercializados a nivel local.

En la Tabla 6.13 anterior se tiene un aproximado de los precios de los PLCs que actualmente se manejan en la industria local. Estos precios son referenciales ya que dependen del proveedor y de la cantidad que se compre. Se puede ver que estos precios están cercanos a los que se obtuvo con el PLC de este proyecto.

Por último, y como desventaja de los PLCs comerciales, está el costo que tienen los módulos de ampliación que, aunque facilitan el uso y manejo de estos, encarecen su precio y que en ocasiones es tan alto, que iguala el precio del PLC original. A continuación se indican diferentes módulos de ampliación comerciales que se encontró en Internet. (Tabla 6.14.)

	• AS-B875-111 Reg: $349.00USD AEG ANALOG INPUT MODULE. Módulo de entradas analógicas AEG
	• AS-P120-000 NEW Reg: $79.00USD POWER SUPPLY MODULE. This item is "new open box". Módulo de Fuente de Alimentación
	• 140DDO15310 TSX QUANTUM Reg:$139.00USD GROPUE SCNEIDER, DC OUT 5V TTL 4X8 SINK. Salidas a transistor de colector abierto
	• 1785-ME64 Reg: $349.00USD ALLEN BRADLEY, MEMORY MODULE CARTRIDGE PLC-5/40 5/60 EEPROM. Módulo de Memoria EEPROM

Tabla 6.14. Costo de módulos de ampliación de diferentes marcas

Como ya se ha mencionado anteriormente, el PLC desarrollado en este proyecto tiene la posibilidad de modificar sus entradas o salidas solamente con cambiar dichas tarjetas, sin mucho costo y esta es una ventaja respecto a los PLCs comerciales. La competencia en el mercado actual es alta; sin embargo, se logró el objetivo general de esta Tesis, siendo esta la primera vez que se consigue que un microcontrolador PIC funcione como un PLC.

CAPÍTULO 7

CONCLUSIONES Y RECOMENDACIONES

Después de haber realizado las pruebas con el hardware y software desarrollado, y de los resultados obtenidos, es posible extraer las conclusiones siguientes:

7.1 CONCLUSIONES

- El programa desarrollado para el manejo del PLC del presente proyecto, resultó ser una gran herramienta para programar aplicaciones en lenguaje FBD, lo cual demostró que la selección de este lenguaje fue la correcta.

- Con este software también se puede generar el código hexadecimal del microcontrolador PIC, lo que posibilitaría programarlo mediante cualquier otra herramienta de programación de PICs y no solamente con la que se incluye en este proyecto.

- El programa desarrollado permite programar al microcontrolador PIC de una forma diferente y más fácil que en lenguaje ensamblador. Esto evita que el usuario deba configurar todos los registros necesarios para utilizar los módulos internos que posee el PIC (A/D, PWM, comunicación serial, etc.)

- A pesar de que se tienen varias funciones incorporadas en el PLC aquí programado, puede ser que durante su utilización el usuario requiera otras. Esta posible necesidad no es un limitante ya que se puede ampliar el número de funciones trabajando en una nueva versión del software.

- La estructura de acrílico que contiene al PLC no tiene el acabado que se esperaba. Se puede mejorarla pero pagando un costo muy alto, el cual puede reducirse tan solo si se llegan a producir en gran escala.

- En todo programa se necesita una ayuda interactiva, y esta se desarrolló mediante el programa VisualHelp. Esta ayuda es invocada desde el software del PLC y resultó ser de mucha utilidad durante el uso del PLC.

- La ventaja que tienen los microcontroladores PIC, que pueden ser programados en el mismo circuito de aplicación que controlan sin ser extraídos de la tarjeta, resultó ser de mucha importancia, corroborando que su elección fue apropiada.

- El microcontrolador PIC utilizado tienen únicamente 3 timers, de los cuales solo uno se utilizó para simular los 16 temporizadores que posee un PLC típico (8 Ton y 8 Toff), con lo cual se cumple con los requisitos de un PLC básico con un mínimo error de conteo de tiempo en dichos timers.

- Como el PLC desarrollado posee una estructura modular, se podrían elaborar tarjetas combinadas de entradas y salidas según las necesidades lo requieran. Al momento el PLC tiene varias como 8 entradas digitales, 8 salidas digitales, 8 entradas analógicas y 2 salida PWM, más comunicación serial RS232, características que son suficientes para que el PLC pueda ser empleado en un buen número de aplicaciones.

- El PLC de este proyecto no posee funciones para operaciones matemáticas ni tampoco controladores PID debido, principalmente, a que el tiempo de realización del proyecto se consumió mayormente en el desarrollo de la interfaz gráfica y la simulación. Sin embargo, cabe recalcar que estas características no se encuentran dentro del alcance del proyecto de titulación.

- En general la herramienta que este software representa en la elaboración de proyectos, es la demostración de que se puede desarrollar programas no solo como Proyectos de Titulación, sino también para aplicaciones que la industria actualmente necesita y seguramente a un menor costo.

7.2 RECOMENDACIONES

- En la interfaz gráfica se puede ver que hay una pequeña separación entre las líneas conectoras y los terminales de I/O de las funciones. Esto se debe a que el programa fue desarrollado en Visual Basic 6.0, el cual no permite desarrollo de aplicaciones gráficas de buena resolución. Por lo mismo, si bien la programación en Basic permitió optimizar tiempo en el desarrollo del simulador y del programador, se recomienda que para la interfaz gráfica de futuros proyectos, con similares características, se emplee Visual C++ o Visual C#.

- Para nuevas versiones de este PLC se recomienda utilizar los PICs 18FXXX, que poseen varias funciones incorporadas dentro de sus instrucciones o también DSPICs 30FXX.

- Si se desea agregar módulos de ampliación al PLC como reloj en tiempo real, conversores A/D, memoria EEPROM, entre otros, se podría utilizar la comunicación I2C, que este microcontrolador dispone. En este proyecto no se utilizó esta característica basados en que el resto de características propias del microcontrolador empleado son suficientes para el PLC desarrollado.

- Se recomienda que la utilización de las tarjetas electrónicas que dispone el PLC sean manejas adecuadamente y con precaución, para evitar su deterioro o daño permanente.

- Este proyecto demostró que es posible emplear un PIC como el procesador base de un PLC. De los resultados obtenidos se recomienda se continúen con esfuerzos para desarrollar un PLC, esta vez con características estéticas que posibiliten su comercialización.

REFERENCIAS BIBLIOGRÁFICAS

(1) CEBALLOS, Francisco. Curso de programación de Visual Basic 6. Primera Edición. Editora AlfaOmega. México. 2003.

(2) ANGULO, José; ANGULO, Ignacio. Microcontroladores PIC, Diseño Práctico de Aplicaciones Tomo I. Primera Edición. Editora McGrawHill. España. 2000.

(3) ANGULO, José; ROMERO, Susana. Microcontroladores PIC, Diseño Práctico de Aplicaciones Tomo II. Primera Edición. Editora McGrawHill. España. 2000.

(4) CARROBLES, Marcial; RODRÍGUEZ, Felix. Manual de Mecánica Industrial, Autómatas y Robótica III. Editora Cultural. España. 2000.

(5) DEITEL, Harvey; DEITEL, Paúl. Visual Basic 6. Cuarta Edición. Editora Prentice Hall. México. 2003.

(6) LASCANO, Carlos; VALLEJO, Fabián. Tesis de Diseño y Construcción de un Controlador Lógico Programable. Ecuador. 2001.

(7) MICROCHIP TECHNOLOGY INC. FLASH Memory Programming Specification PIC16F87XA. USA. 2002

(8) MICROCHIP TECHNOLOGY INC. Data Sheet PIC16F87XA, 28/40/44-Pin Enhanced Flash Microcontrollers 8-Bit. USA. 2003.

DIRECCIONES ELECTRÓNICAS

(9) MICROCHIP TECHNOLOGY INC. http://www.microchip.com

(10) Fujitsu Components America. http://www.fujitsu.com

(11) http://www.pablin.com.ar/computer/programa/vb/iodll.htm

(12) http://www.latin-tech.com

(13) http://www.ad.siemens.de/s7-200

(14) http://www.automationpartswarehouse.com/

(15) http://www.cadllp.com/track.php

(16) http://www.automationpartswarehouse.com/home.html

ANEXOS

ANEXO A

ESPECIFICACIONES DE LA PROGRAMACIÓN DE LA MEMORIA FLASH DE LOS PIC16F87XA

PIC16F87XA

FLASH Memory Programming Specification

This document includes programming specifications for the following devices:

- PIC16F873A
- PIC16F874A
- PIC16F876A
- PIC16F877A

1.0 PROGRAMMING THE PIC16F87XA

The PIC16F87XA is programmed using a serial method. The Serial mode will allow the PIC16F87XA to be programmed while in the user's system. This allows for increased design flexibility. This programming specification applies to PIC16F87XA devices in all packages.

1.1 Programming Algorithm Requirements

The programming algorithm used depends on the operating voltage (VDD) of the PIC16F87XA device, or whether internal or external timing is desired.

Algorithm #	VDD Range	Timing
1	2.0V ≤ VDD < 5.5V	Internal; 4 ms/op
2	4.5V ≤ VDD ≤ 5.5V	External; 1 ms/op

Both algorithms can be used with the two available programming entry methods. The first method follows the normal Microchip Programming mode entry of holding pins RB6 and RB7 low, while raising $\overline{MCLR}$ pin from VIL to VIHH (13V ± 0.5V). The second method, called Low Voltage ICSP™ or LVP for short, applies VDD to $\overline{MCLR}$ and uses the I/O pin RB3 to enter Programming mode. When RB3 is driven to VDD from ground, the PIC16F87XA device enters Programming mode.

1.2 Programming Mode

The Programming mode for the PIC16F87XA allows programming of user program memory, data memory, special locations used for ID, and the configuration word.

Pin Diagrams

PDIP, SOIC

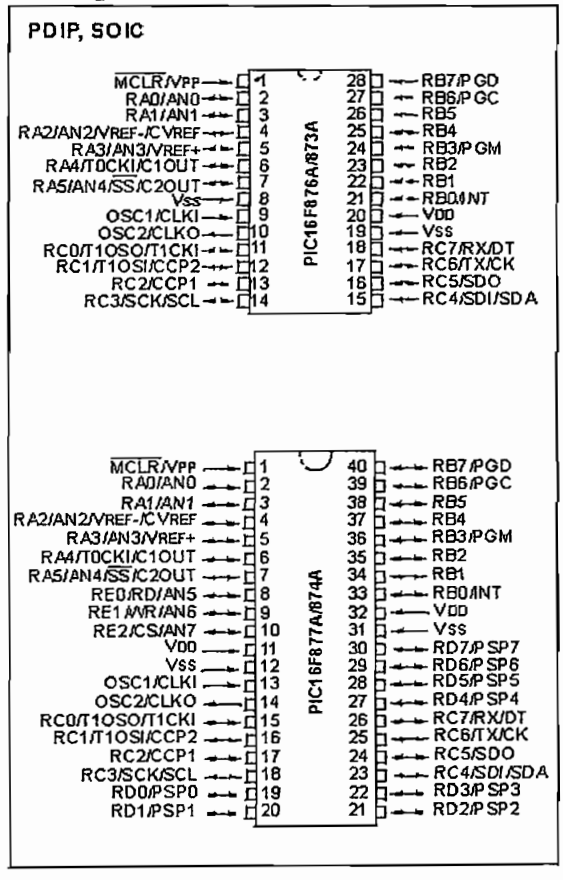


TABLE 1-1: PIN DESCRIPTIONS (DURING PROGRAMMING): PIC16F87XA

Pin Name	During Programming		
	Function	Pin Type	Pin Description
RB3	PGM	I	Low voltage ICSP programming input if LVP configuration bit equals '1'
RB6	CLOCK	I	Clock input
RB7	DATA	I/O	Data input/output
$\overline{MCLR}$	VTEST MODE	P*	Program Mode Select
VDD	VDD	P	Power Supply
VSS	VSS	P	Ground

Legend: I = Input, O = Output, P = Power

* To activate the Programming mode, high voltage needs to be applied to the $\overline{MCLR}$ input. Since $\overline{MCLR}$ is used for a level source, this means that $\overline{MCLR}$ does not draw any significant current.

2.0 PROGRAM MODE ENTRY

2.1 User Program Memory Map

The user memory space extends from 0000h to 1FFFh (8 K words). In Programming mode, the program memory space extends from 0000h to 3FFFh, with the first half (0000h - 1FFFh) being user program memory and the second half (2000h - 3FFFh) being configuration memory. The PC will increment from 0000h to 1FFFh and wrap around to 0000h. From 2000h, the PC will increment up to 3FFFh and wrap around to 2000h (not to 0000h). Once in configuration memory, the highest bit of the PC stays a '1', thus always pointing to the configuration memory. The only way to point to user program memory is to reset the part and re-enter Program/Verify mode, as described in Section 2.4.

In the configuration memory space, 2000h - 200Fh are physically implemented. However, only locations 2000h through 2007h are available. Other locations are reserved. Locations beyond 200Fh will physically access user memory (see Figure 2-1).

2.2 Data EEPROM Memory

The EEPROM data memory space is a separate block of high endurance memory that the user accesses, using a special sequence of instructions. The amount of data EEPROM memory depends on the device and is shown below in number of bytes.

Device	# of Bytes
PIC16F873A	128
PIC16F874A	128
PIC16F876A	256
PIC16F877A	256

The contents of data EEPROM memory have the capability to be embedded into the HEX file.

The programmer should be able to read data EEPROM information from a HEX file and conversely (as an option), write data EEPROM contents to a HEX file, along with program memory information and configuration bit information.

The 256 data memory locations are logically mapped starting at address 2100h. The format for data memory storage is one data byte per address location, LSB aligned.

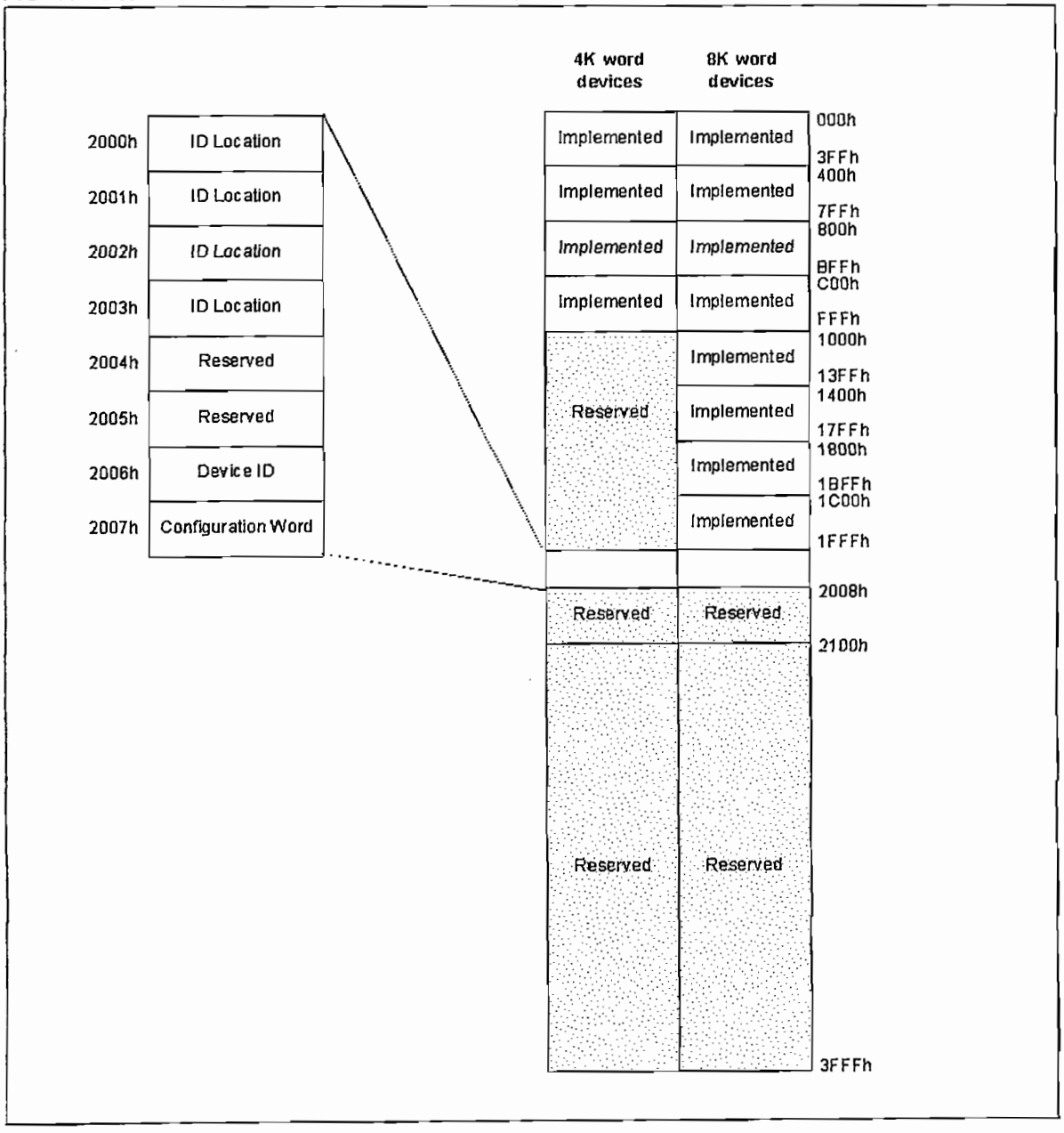
2.3 ID Locations

A user may store identification information (ID) in four ID locations. The ID locations are mapped in addresses 2000h - 2003h. It is recommended that the user use only the four Least Significant bits of each ID location. In some devices, the ID locations read out in an unscrambled fashion after code protection is enabled.

For these devices, it is recommended that ID location is written as "11 1111 1000 bbbb", where 'bbbb' is ID information.

In other devices, the ID locations read out normally, even after code protection. To understand how the devices behave, refer to Table 5-1.

FIGURE 2-1: PIC16F87XA PROGRAM MEMORY MAPPING



2.4 Program/Verify Mode

The Program/Verify mode is entered by holding pins RB6 and RB7 low, while raising $\overline{MCLR}$ pin from V_{IL} to V_{IHH} (high voltage). In this mode, the state of the RB3 pin does not effect programming. Low Voltage ICSP Programming mode is entered by raising RB3 from V_{IL} to V_{DD}, and then applying V_{DD} to $\overline{MCLR}$. Once in this mode, the user program memory and the configuration memory can be accessed and programmed in serial fashion. The mode of operation is serial, and the memory accessed is the user program memory. RB6 and RB7 are Schmitt Trigger inputs in this mode.

Note:	The OSC must not have 72 osc clocks while the device $\overline{MCLR}$ is between V_{IL} and V_{IHH}.

The sequence that enters the device into the Programming/Verify mode places all other logic into the RESET state (the $\overline{MCLR}$ pin was initially at V_{IL}). This means all I/O are in the RESET state (high impedance inputs).

A device RESET will clear the PC and set the address to '0'. The 'Increment Address' command will increment the PC. The 'Load Configuration' command will set the PC to 2000h. The available commands are shown in Table 2-1.

The normal sequence for programming eight program memory words at a time is as follows:

1. Load a word at the current program memory address using the 'Load Data' command.
2. Issue an 'Increment Address' command.
3. Load a word at the current program memory address using the 'Load Data' command.
4. Repeat Step 2 and Step 3 six times.
5. Issue a 'Begin Programming' command to begin programming.
6. Wait tprog (about 1 ms).
7. Issue an 'End Programming' command.
8. Increment to the next address.
9. Repeat this sequence as required to write program and configuration memory.

The alternative sequence for programming one program memory word at a time is as follows:

1. Set a word for the current memory location using the 'Load Data' command.
2. Issue a 'Begin Programming Only' command to begin programming.
3. Wait tprog.
4. Issue an 'End Programming' command.
5. Increment to the next address.
6. Repeat this alternative sequence as required to write program and configuration memory.

The address and program counter are reset to 0000h by resetting the device (taking $\overline{MCLR}$ below V_{IL}) and re-entering Programming mode. Program and configuration memory may then be read or verified using the 'Read Data' and 'Increment Address' commands.

2.4.1 LOW VOLTAGE ICSP PROGRAMMING MODE

Low Voltage ICSP Programming mode allows a PIC16F87XA device to be programmed using V_{DD} only. However, when this mode is enabled by a configuration bit (LVP), the PIC16F87XA device dedicates RB3 to control entry/exit into Programming mode.

When LVP bit is set to '1', the low voltage ICSP programming entry is enabled. Since the LVP configuration bit allows low voltage ICSP programming entry in its erased state, an erased device will have the LVP bit enabled at the factory. While LVP is '1', RB3 is dedicated to low voltage ICSP programming. Bring RB3 and then, $\overline{MCLR}$ to V_{DD} to enter Programming mode. All other specifications for high voltage ICSP apply.

To disable Low Voltage ICSP mode, the LVP bit must be programmed to '0'. This must be done while entered with the High Voltage Entry mode (LVP bit = '1'). RB3 is now a general purpose I/O pin.

2.4.2 SERIAL PROGRAM/VERIFY OPERATION

The RB6 pin is used as a clock input pin, and the RB7 pin is used to enter command bits, and to input or output data during serial operation. To input a command, the clock pin (RB6) is cycled six times. Each command bit is latched on the falling edge of the clock, with the Least Significant bit (LSb) of the command being input first. The data on RB7 is required to have a minimum setup (tset1) and hold (thold1) time (see AC/DC specifications), with respect to the falling edge of the clock. Commands with associated data (read and load) are specified to have a minimum delay (tdly1) of 1 µs between the command and the data. After this delay, the clock pin is cycled 16 times, with the first cycle being a START bit (0) and the last cycle being a STOP bit (0). Data is transferred LSb first.

During a read operation, the LSb will be transmitted onto RB7 on the rising edge of the second cycle, and during a load operation, the LSb will be latched on the falling edge of the second cycle. A minimum 1 µs delay (tdly2) is specified between consecutive commands.

All commands and data words are transmitted LSb first. The data is transmitted on the rising edge, and latched on the falling edge of the clock. To allow decoding of commands and reversal of data pin configuration, a time separation of at least 1 µs (tdly1) is required between a command and a data word, or another command.

The available commands are described in the following paragraphs and listed in Table 2-1.

2.4.2.1 Load Configuration

After receiving this command, the program counter (PC) will be set to 2000h. By then applying 16 cycles to the clock pin, the chip will load 14 bits in a "data word," as described above, to be programmed into the configuration memory. A description of the memory mapping schemes of the program memory for normal operation and configuration mode operation is shown in Figure 2-1. After the configuration memory is entered, the only way to get back to the user program memory is to exit the Program/Verify Test mode by taking $\overline{MCLR}$ low (V_{IL}).

2.4.2.2 Load Data for Program Memory

After receiving this command, the chip will load one word (with 14 bits as a "data word") to be programmed into user program memory when 16 cycles are applied. A timing diagram for this command is shown in Figure 6-1.

2.4.2.3 Load Data for Data Memory

After receiving this command, the chip will load in a 14-bit "data word" when 16 cycles are applied. However, the data memory is only 8-bits wide, and thus, only the first 8 bits of data after the START bit will be programmed into the data memory. It is still necessary to cycle the clock the full 16 cycles in order to allow the internal circuitry to reset properly. The data memory contains up to 256 bytes. If the device is code protected, the data is read as all zeros. A timing diagram for this command is shown in Figure 6-2.

2.4.2.4 Read Data from Program Memory

After receiving this command, the chip will transmit data bits out of the program memory (user or configuration) currently accessed, starting with the second rising edge of the clock input. The RB7 pin will go into Output mode on the second rising clock edge, and it will revert back to Input mode (hi-impedance) after the 16th rising edge. A timing diagram of this command is shown in Figure 6-3.

2.4.2.5 Read Data from Data Memory

After receiving this command, the chip will transmit data bits out of the data memory, starting with the second rising edge of the clock input. The RB7 pin will go into Output mode on the second rising edge, and it will revert back to Input mode (hi-impedance) after the 16th rising edge. As previously stated, the data memory is 8-bits wide, and therefore, only the first 8 bits that are output are actual data. A timing diagram for this command is shown in Figure 6-4.

2.4.2.6 Increment Address

The PC is incremented when this command is received. A timing diagram of this command is shown in Figure 6-5.

2.4.2.7 Begin Erase/Program Cycle

Eight locations must be loaded before every 'Begin Erase/Programming' command. After this command is received and decoded, eight words of program memory will be erased and programmed with the values contained in the program data latches. The PC address will decode which eight words are programmed. The lower three bits of the PC are ignored, so if the PC points to address 003h, then all eight locations from 000h to 007h are written.

An internal timing mechanism executes an erase before write. The user must allow the combined time for erase and programming, as specified in the electrical specs, for programming to complete. No 'End Programming' command is required.

1. If the address is pointing to user memory, the user memory alone will be affected.
2. If the address is pointing to the physically implemented test memory (2000h - 201Fh), test memory will be written. The configuration word will not be written unless the address is specifically pointing to 2007h.

This command can be used to perform programming over the entire V_{DD} range of the device.

Note 1:	The code protect bits cannot be erased with this command.
2:	All Begin Erase/Programming operations can take place over the entire V_{DD} range.

A timing diagram for this command is shown in Figure 6-6.

2.4.2.8 Begin Programming Only

Note:	Begin Programming Only operations must take place at the 4.5V to 5.5V V_{DD} range.

This command is similar to the 'Erase/Programming Cycle' command, except that a word erase is not done, and the internal timer is not used. Programming of program and data memory will begin after this command is received and decoded. The user must allow the time for programming, as specified in the electrical specs, for programming to complete. An 'End Programming' command is required.

The internal timer is not used for this command, so the 'End Programming' command must be used to stop programming.

1. If the address is pointing to user memory, the user memory alone will be affected.
2. If the address is pointing to the physically implemented test memory (2000h - 201Fh), the test memory will be written. The configuration word will not be written unless the address is specifically pointing to 2007h.

A timing diagram for this command is shown in Figure 6-7.

2.4.2.9 End Programming

After receiving this command, the chip stops programming the memory (test program memory or user program memory) that it was programming at the time.

Note:	This command will also set the write data shift latches to all '1's to avoid issues with downloading only one word before the write.

TABLE 2-1: COMMAND MAPPING FOR PIC16F87XA

Command	Mapping (MSB ... LSB)					Data	Voltage Range
Load Configuration	0	0	0	0	0	0, data (14), 0	2.2V - 5.5V
Load Data for Program Memory	0	0	0	1	0	0, data (14), 0	2.2V - 5.5V
Read Data from Program Memory	0	0	1	0	0	0, data (14), 0	2.2V - 5.5V
Increment Address	0	0	1	1	0		2.2V - 5.5V
Begin Erase/Programming Cycle	0	1	0	0	0	4 ms typical, internally timed	2.2V - 5.5V
Begin Programming Only Cycle	1	1	0	0	0	1 ms typical, externally timed	4.5V - 5.5V
Bulk Erase Program Memory	0	1	0	0	1	4 ms typical, internally timed	4.5V - 5.5V
Bulk Erase Data Memory	0	1	0	1	1	4 ms typical, internally timed	4.5V - 5.5V
Chip Erase	1	1	1	1	1	4 ms typical, internally timed	4.5V - 5.5V
Load Data for Data Memory	0	0	0	1	1	0, data (14), 0	2.2V - 5.5V
Read Data from Data Memory	0	0	1	0	1	0, data (14), 0	2.2V - 5.5V
End Programming	1	0	1	1	1		

2.5 Erasing Program and Data Memory

Depending on the state of the code protection bits, program and data memory will be erased using different methods. The first two commands are used when both program and data memories are not code protected. The third command is used when either memory is code protected, or if you want to also erase the fuse locations, including the code protect bits. A device programmer should determine the state of the code protection bits and then apply the proper command to erase the desired memory.

2.5.1 ERASING NON-CODE PROTECTED PROGRAM AND DATA MEMORY

When both program and data memories are not code protected, they must be individually erased using the following commands. The only way that both memories are erased using a single command is if code protection is enabled for one of the memories. These commands do not erase the configuration word or ID locations.

2.5.1.1 Bulk Erase Program Memory

When this command is performed, and is followed by a 'Begin Erase/Programming' command, the entire program memory will be erased.

If the address is pointing to user memory, only the user memory will be erased.

If the address is pointing to the test program memory (2000h - 201Fh), then both the user memory and the test memory will be erased. The configuration word will not be erased, even if the address is pointing to location 2007h.

Previously, a load data with 0FFh command was recommended before any Bulk Erase. On these devices, this will not be required.

The Bulk Erase command is disabled when the CP bit is programmed to '0' enabling code protect.

A timing diagram for this command is shown in Figure 6-8.

2.5.1.2 Bulk Erase Data Memory

When this command is performed, and is followed by a 'Begin Erase/Programming' command, the entire data memory will be erased.

The Bulk Erase Data command is disabled when the CPD bit is programmed to '0' enabling protected data memory. A timing diagram for this command is shown in Figure 6-9.

Note:	All Bulk Erase operations must take place at the 4.5V to 5.5V VDD range.

2.5.1.3 Chip Erase

This command, when performed, will erase the program memory, EE data memory, and all of the fuse locations, including the code protection bits. All on-chip FLASH and EEPROM memory is erased, regardless of the address contained in the PC.

When a Chip Erase command is issued and the PC points to (0000h - 1FFFh), the configuration word and the user program memory will be erased, but not the test row (see Section 2.5.2.1). Chip Erase can also be used to erase code protected memory, as described in Section 2.5.2.

This command will also erase the code protect and code protect data fuses if they are programmed. This is the only command that allows a user to erase the code protect fuses.

The Chip Erase is internally self-timed to ensure that all program and data memory is erased before the code protect bits are erased. A timing diagram for this command is shown in Figure 6-10.

Note:	The Chip Erase operation must take place at the 4.5V to 5.5V VDD range.

2.5.2 ERASING CODE PROTECTED MEMORY

For the PIC16F87XA devices, once code protection is enabled, all protected program and data memory locations read all '0's and further programming is disabled. The ID locations and configuration word read out unscrambled and can be reprogrammed normally. The only command to erase a code protected PIC16F87XA device is the Chip Erase. This erases program memory, data memory, configuration bits and ID locations. **Since all data within the program and data memory will be erased when this command is executed, the security of the data or code is not compromised.**

2.5.2.1 Chip Erase

This command, when performed, will erase the program memory, data EEPROM, and all of the fuse locations, including the code protection bits, code protect fuses, and code protect data fuses. All on-chip FLASH and EEPROM memory is erased, regardless of the address contained in the PC.

If the PC points to user memory, the test row (2000h through 201Fh) is not erased with a Chip Erase command, except for the configuration word (at 2007h). If the test row is to be completely erased, the address in the PC must point to configuration memory.

When the PC points to 2000h - 201Fh, the configuration word, test program memory, and the user program memory will all be erased with a Chip Erase command. This allows the user to erase all program and configuration content, including the code protect bits, without compromising the user ID bits (2000h through 2004h), or any pass codes stored in the test row.

The Chip Erase is internally self-timed to ensure that all program and data memory is erased before the code protect bits are erased.

A timing diagram for this command is shown in Figure 6-10.

Note:	*The Chip Erase operation must take place at the 4.5V to 5.5V VDD range.*

FIGURE 2-2: ALGORITHM 1 FLOW CHART – PROGRAM MEMORY (2.0V ≤ VDD < 5.5V)

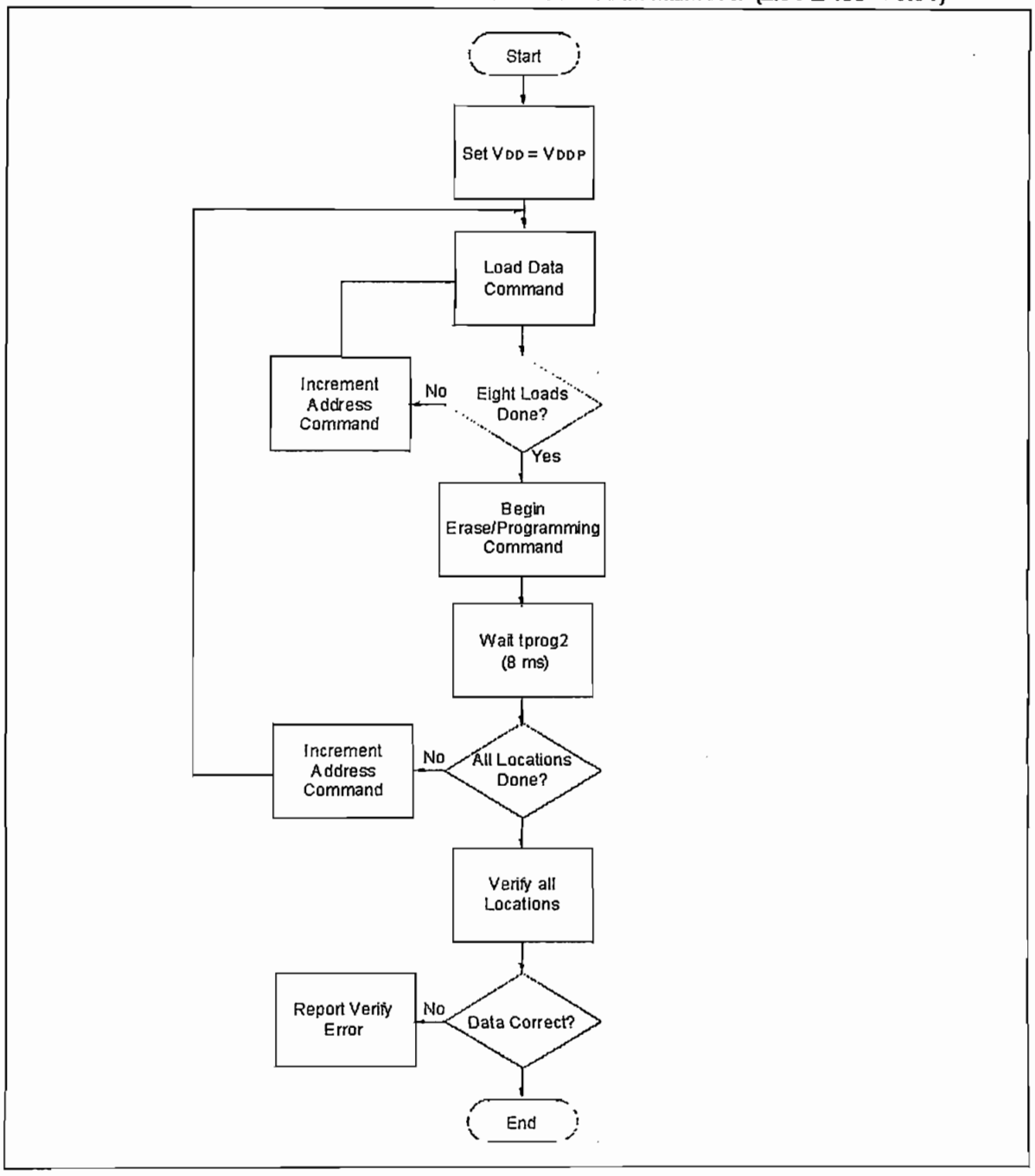


FIGURE 2-3: ALGORITHM 2 FLOW CHART – PROGRAM MEMORY (4.5V ≤ VDD ≤ 5.5V)

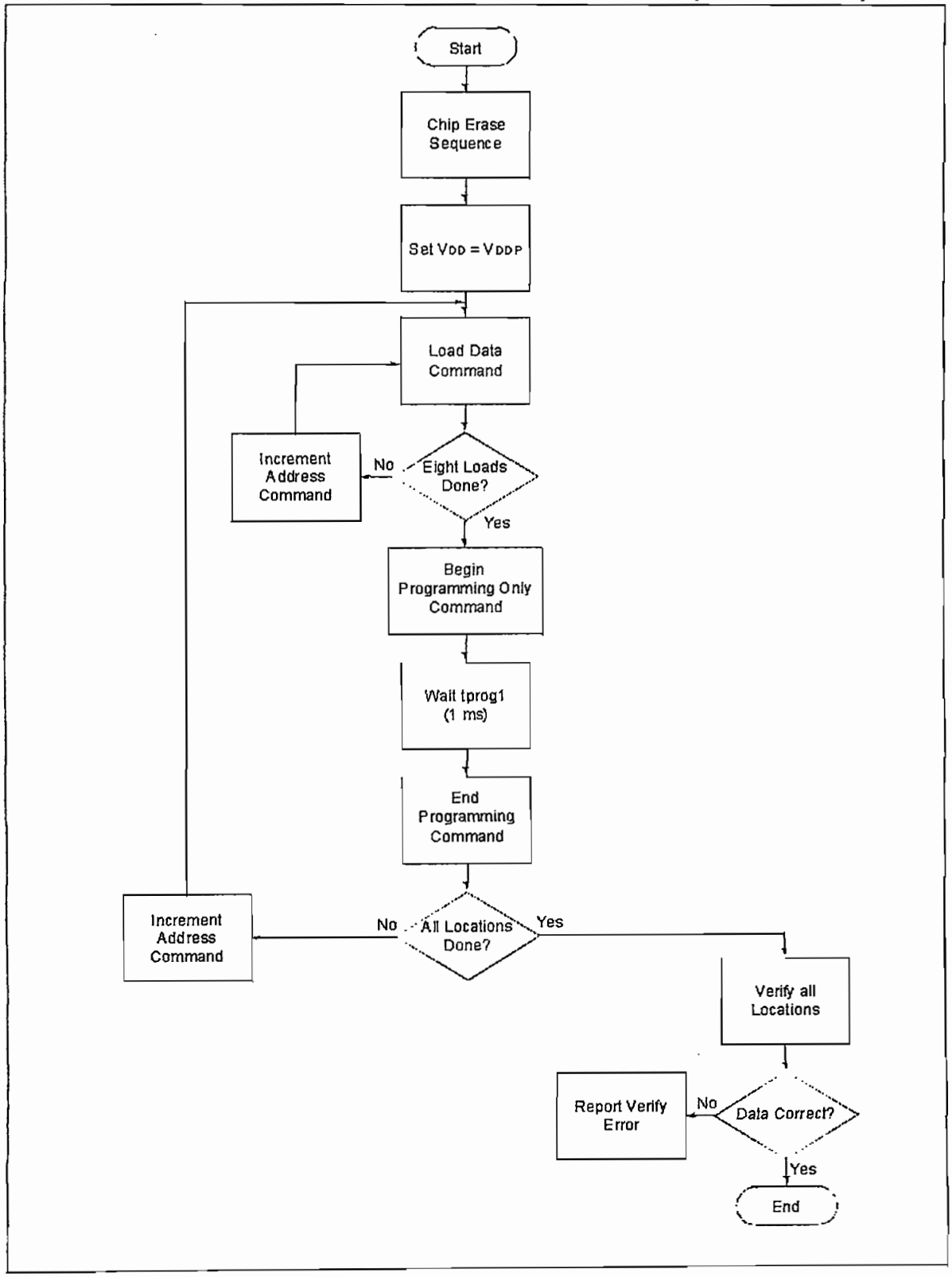


FIGURE 2-4: FLOW CHART – PIC16F87XA CONFIGURATION MEMORY (2.0V ≤ VDD < 5.5V)

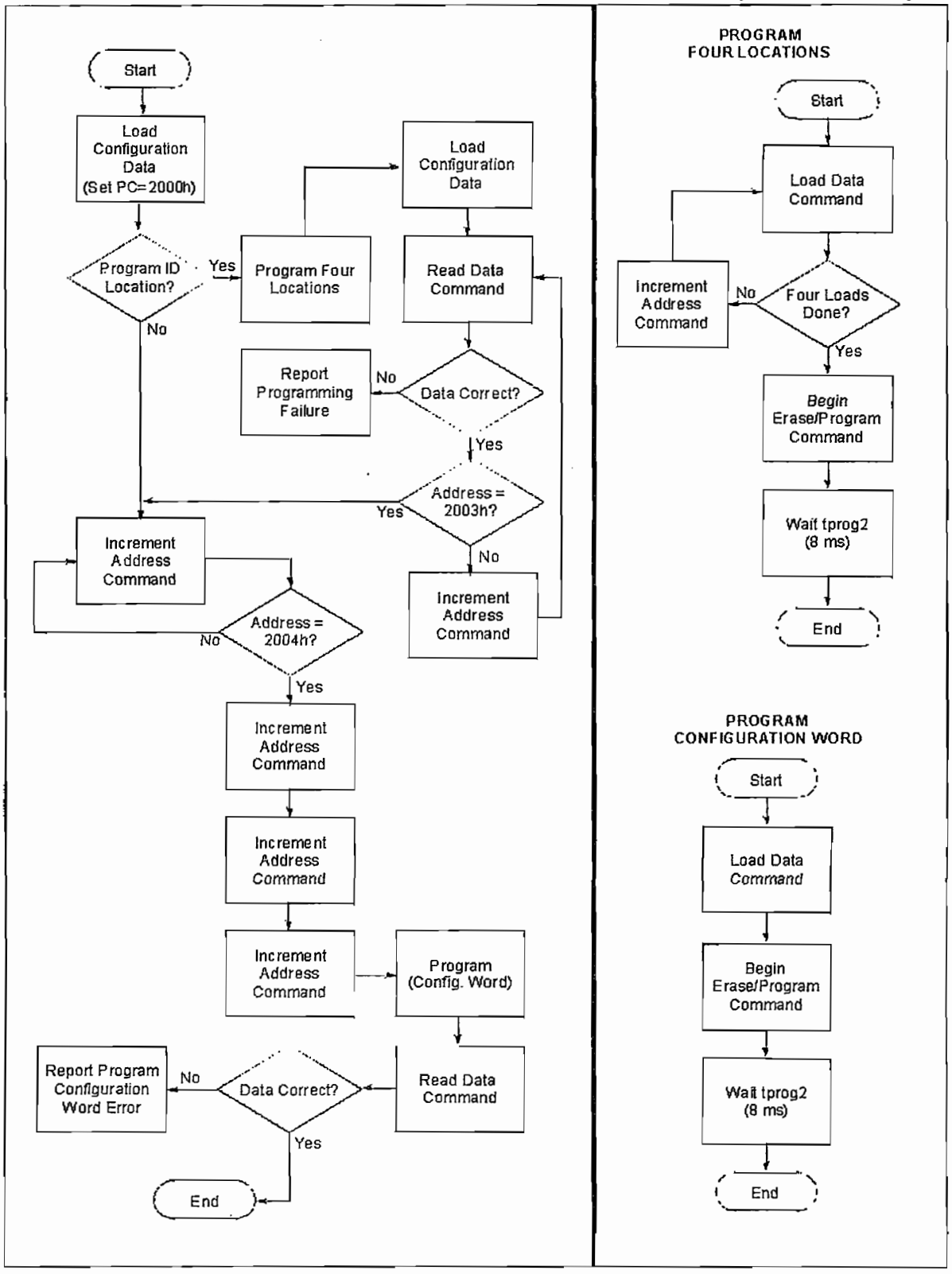
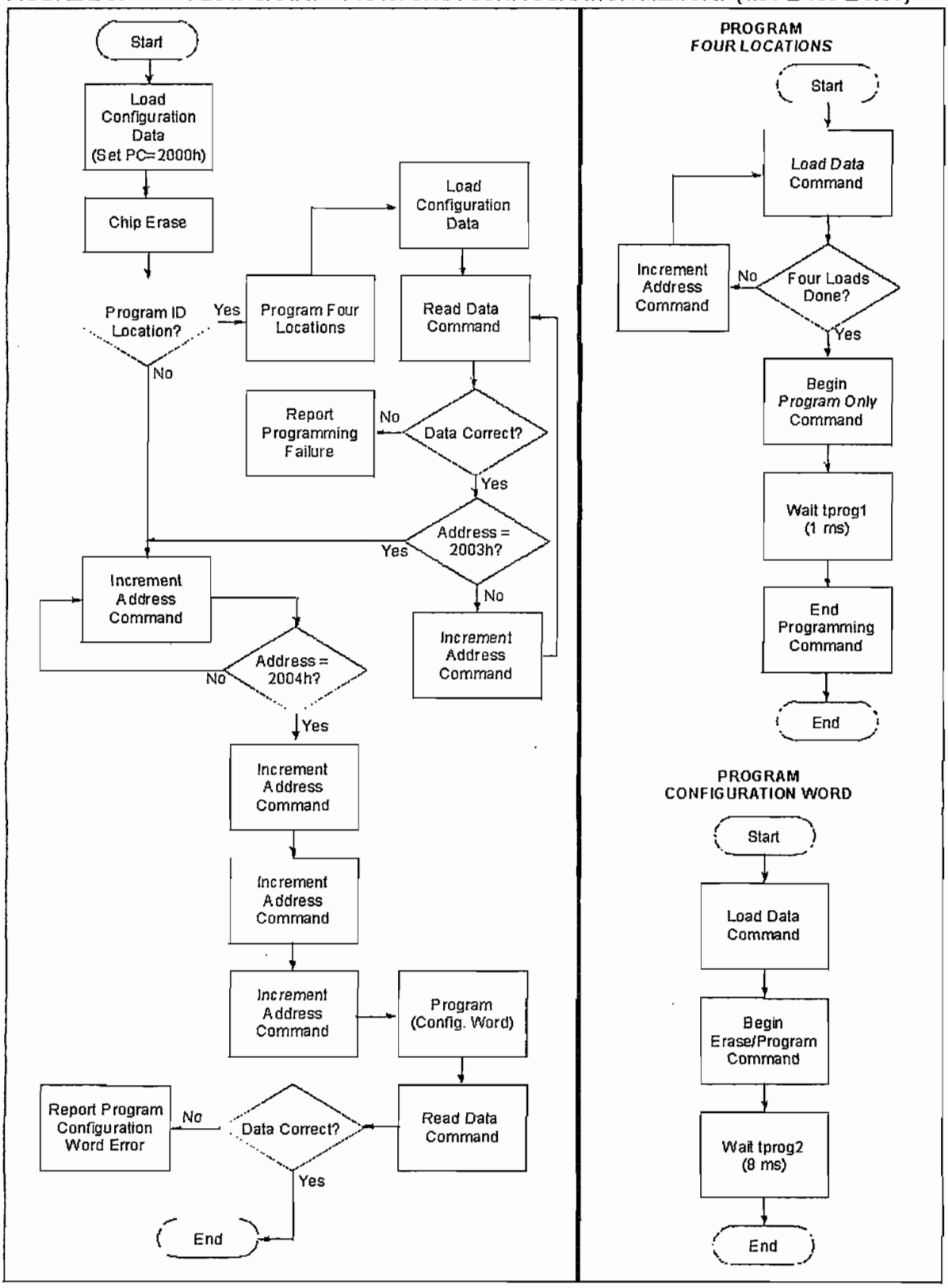


FIGURE 2-5: FLOW CHART – PIC16F87XA CONFIGURATION MEMORY (4.5V ≤ VDD ≤ 5.5V)



3.0 CONFIGURATION WORD

The PIC16F87XA has several configuration bits. These bits can be set (reads '0'), or left unchanged (reads '1'), to select various device configurations.

3.1 Device ID Word

The device ID word for the PIC16F87XA is located at 2006h.

TABLE 3-1: DEVICE ID VALUE

Device	Device ID Value	
	Dev	Rev
PIC16F873A	00 1110 0100	xxxx
PIC16F874A	00 1110 0110	xxxx
PIC16F876A	00 1110 0000	xxxx
PIC16F877A	00 1110 0010	xxxx

REGISTER 3-1: CONFIGURATION WORD REGISTER

R/P-1	U-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	R/P-1	U-1	U-1	R/P-1	R/P-1	R/P-1	R/P-1
CP	—	DEBUG	WRT1	WRT0	CPD	LVP	BOREN	—	—	PWRTEN	WDTEN	FOSC1	FOSC0
bit 13													bit 0

bit 13 **CP:** FLASH Program Memory Code Protection bit
(PIC16F877A/876A):
1 = Code protection off
0 = 0000h to 1FFFh code protected
(PIC16F874A/873A):
1 = Code protection off
0 = 0000h to 0FFFh code protected
1000h to 1FFFh wraps to 0000h to 0FFFh

bit 12 **Unimplemented:** Read as '1'

bit 11 **DEBUG:** Background Debugger Mode bit
1 = Background debugger functions not enabled
0 = Background debugger functional

bit 10-9 **WRT<1:0>:** FLASH Program Memory Write Enable bits
(PIC16F877A/876A):
11 = Write protection off
10 = 0000h to 00FFh write protected, 0100h to 1FFFh may be modified by EECON control
01 = 0000h to 07FFh write protected, 0800h to 1FFFh may be modified by EECON control
00 = 0000h to 0FFFh write protected, 1000h to 1FFFh may be modified by EECON control
(PIC16F874A/873A):
11 = Write protection off
10 = 0000h to 00FFh write protected, 0100h to 0FFFh may be modified by EECON control
01 = 0000h to 03FFh write protected, 0400h to 0FFFh may be modified by EECON control
00 = 0000h to 07FFh write protected, 0800h to 1FFFh may be modified by EECON control

bit 8 **CPD:** Data EE Memory Code Protection bit
1 = Code protection off
0 = Data EE memory code protected

bit 7 **LVP:** Low Voltage Programming Enable bit
1 = RB3/PGM pin has PGM function, low voltage programming enabled
0 = RB3 is digital I/O, HV on $\overline{MCLR}$ must be used for programming

bit 6 **BOREN:** Brown-out Reset Enable bit
1 = BOR enabled
0 = BOR disabled

bit 5-4 **Unimplemented:** Read as '1'

bit 3 **$\overline{PWRTEN}$:** Power-up Timer Enable bit
1 = PWRT disabled
0 = PWRT enabled

bit 2 **WDTEN:** Watchdog Timer Enable bit
1 = WDT enabled
0 = WDT disabled

bit 1-0 **FOSC<1:0>:** Oscillator Selection bits
11 = RC oscillator
10 = HS oscillator
01 = XT oscillator
00 = LP oscillator

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '1'	
-n = Default value	1 = Bit is erased	0 = Bit is programmed	x = Bit is unknown

4.0 EMBEDDING CONFIGURATION WORD AND ID INFORMATION IN HEX FILE

To allow portability of code, the programmer is required to read the configuration word and ID locations from the HEX file when loading the HEX file. If configuration word information was not present in the HEX file, then a simple warning message may be issued. Similarly, while saving a HEX file, configuration word and ID information must be included. An option to not include this information may be provided.

Specifically for the PIC16F87XA, the EEPROM data memory should also be embedded in the HEX file (see Section 2.2).

Microchip Technology Inc. feels strongly that this feature is important for the benefit of the end customer.

5.0 CHECKSUM COMPUTATION

Checksum is calculated by reading the contents of the PIC16F87XA memory locations and adding up the opcodes up to the maximum user addressable location, e.g., 0x1FF for the PIC16F87XA. Any carry bits exceeding 16-bits are neglected. Finally, the configuration word (appropriately masked) is added to the checksum. Checksum computation for each member of the PIC16F87XA devices is shown in Table 5-1.

The checksum is calculated by summing the following:

- The contents of all program memory locations
- The configuration word, appropriately masked
- Masked ID locations (when applicable)

The Least Significant 16 bits of this sum are the checksum.

The following table describes how to calculate the checksum for each device. Note that the checksum calculation differs depending on the code protect setting. Since the program memory locations read out differently depending on the code protect setting, the table describes how to manipulate the actual program memory values to simulate the values that would be read from a protected device. When calculating a checksum by reading a device, the entire program memory can simply be read and summed. The configuration word and ID locations can always be read.

Note that some older devices have an additional value added in the checksum. This is to maintain compatibility with older device programmer checksums.

TABLE 5-1: CHECKSUM COMPUTATION

Device	Code Protect	Checksum*	Blank Value	25E6h at 0 and max address
PIC16F873A	OFF	SUM[0000:0FFF] + (CFGW & 2FCF)	1FCF	EB9D
	ON	(CFGW & 2FCF) + SUM_ID	4F9E	1B6C
PIC16F874A	OFF	SUM[0000:0FFF] + (CFGW & 2FCF)	1FCF	EB9D
	ON	(CFGW & 2FCF) + SUM_ID	4F9E	1B6C
PIC16F876A	OFF	SUM[0000:1FFF] + (CFGW & 2FCF)	0FCF	DB9D
	ON	(CFGW & 2FCF) + SUM_ID	1F9E	EB6C
PIC16F877A	OFF	SUM[0000:1FFF] + (CFGW & 2FCF)	0FCF	DB9D
	ON	(CFGW & 2FCF) + SUM_ID	1F9E	EB6C

Legend: CFGW = Configuration Word
SUM[a:b] = [Sum of locations a to b inclusive]
SUM_ID = ID locations masked by 0Fh then made into a 16-bit value with ID0 as the most significant nibble. For example, ID0 = 01h, ID1 = 02h, ID3 = 03h, ID4 = 04h, then SUM_ID = 1234h
*Checksum = *[Sum of all the individual expressions] MODULO [FFFFh]*
+ = Addition
& = Bitwise AND

6.0 PROGRAM/VERIFY MODE ELECTRICAL CHARACTERISTICS

TABLE 6-1: TIMING REQUIREMENTS FOR PROGRAM/VERIFY MODE

AC/DC CHARACTERISTICS POWER SUPPLY PINS	Standard Operating Procedure (unless otherwise stated) Operating temperature 0 ≤ TA ≤ +70°C Operating Voltage 2.0V ≤ VDD ≤ 5.5V					
Characteristics	**Sym**	**Min**	**Typ**	**Max**	**Units**	**Conditions/Comments**
General						
VDD level for Begin Erase/Program operations and EECON write of program memory	VDD	2.0		5.5	V	
VDD level for Begin Erase/Program operations and EECON write of data memory	VDD	2.0		5.5	V	
VDD level for Bulk Erase/Write, Chip Erase, and Begin Program operations, of program and data memory	VDD	4.5		5.5	V	
Begin Programming Only cycle time	tprog1	1			ms	Externally Timed
Begin Erase/Programming	tprog2		4	8	ms	Internally Timed
Chip Erase cycle time	tprog3		4	8	ms	Internally Timed
High voltage on $\overline{MCLR}$ and RA4/T0CKI for Test mode entry	VIHH	VDD + 3.5		13.5	V	
$\overline{MCLR}$ rise time (Vss to VHH) for Test mode entry	tVHHR			1.0	µs	
(RB6, RB7) input high level	VIH1	0.8 VDD			V	Schmitt Trigger input
(RB6, RB7) input low level	VIL1	0.2 VDD			V	Schmitt Trigger input
RB<7:4> setup time before $\overline{MCLR}$↑ (Test mode selection pattern setup time)	tset0	100			ns	
RB<7:4> hold time after $\overline{MCLR}$↑ (Test mode selection pattern setup time)	thld0	5			µs	
Serial Program/Verify						
Data in setup time before clock↓	tset1	100			ns	
Data in hold time after clock↓	thld1	100			ns	
Data input not driven to next clock input (delay required between command/data or command/command)	tdly1	1.0			µs	2.0V ≤ VDD < 4.5V
		100			ns	4.5V ≤ VDD ≤ 5.5V
Delay between clock↓ to clock↑ of next command or data	tdly2	1.0			µs	2.0V ≤ VDD < 4.5V
		100			ns	4.5V ≤ VDD ≤ 5.5V
Clock↑ to data out valid (during read data)	tdly3	80			ns	

FIGURE 6-1: LOAD DATA FOR USER PROGRAM MEMORY COMMAND (PROGRAM/VERIFY)

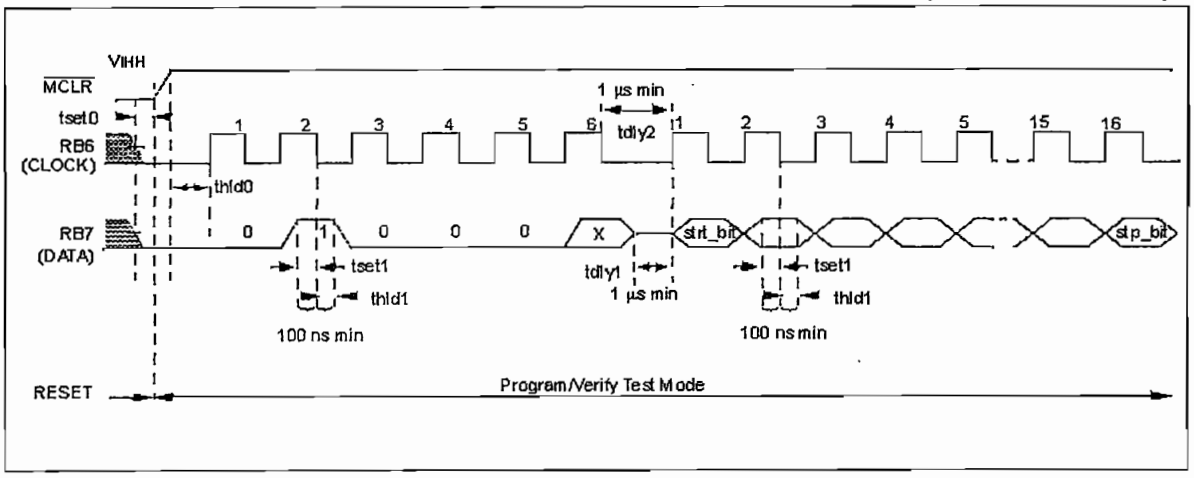


FIGURE 6-2: LOAD DATA FOR USER DATA MEMORY COMMAND (PROGRAM/VERIFY)

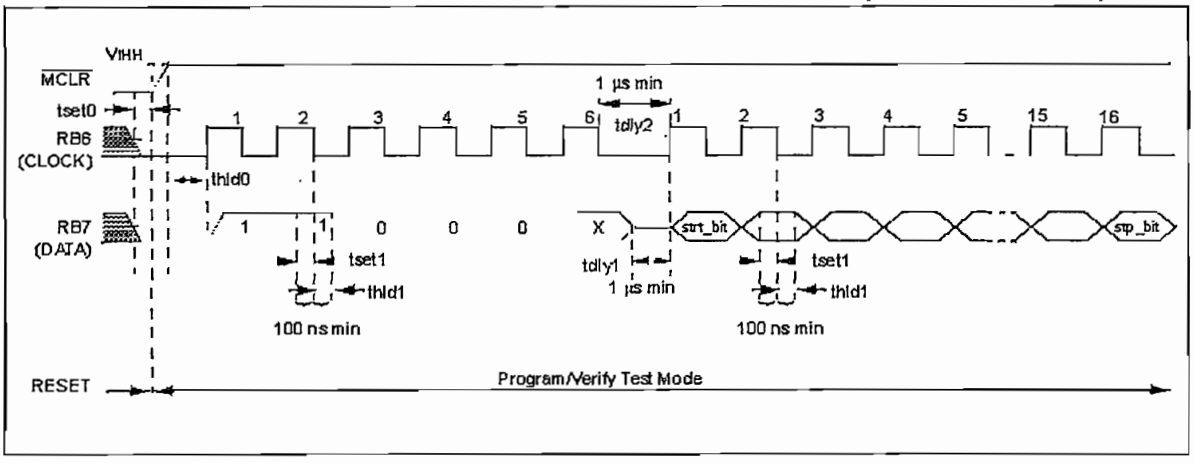


FIGURE 6-3: READ DATA FROM PROGRAM MEMORY COMMAND (PROGRAM/VERIFY)

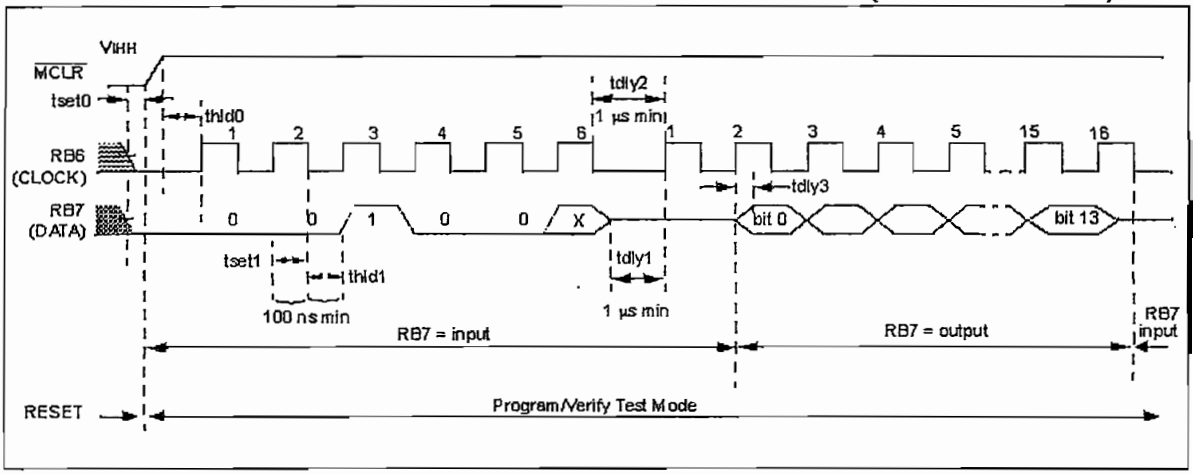


FIGURE 6-4: READ DATA FROM DATA MEMORY COMMAND (PROGRAM/VERIFY)

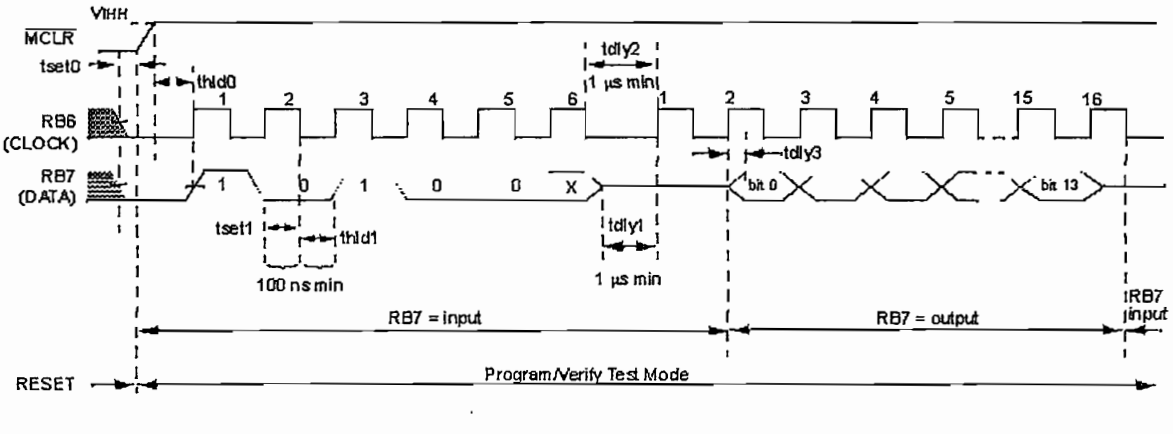


FIGURE 6-5: INCREMENT ADDRESS COMMAND (PROGRAM/VERIFY)

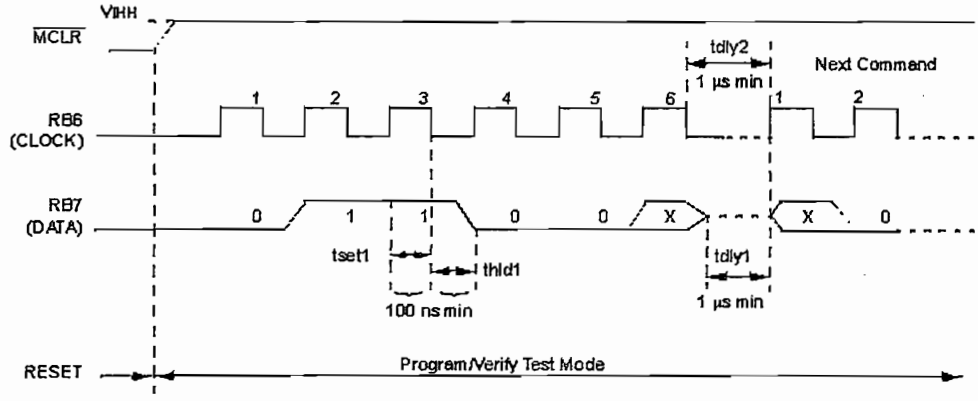


FIGURE 6-6: BEGIN ERASE/PROGRAMING COMMAND (PROGRAM/VERIFY)

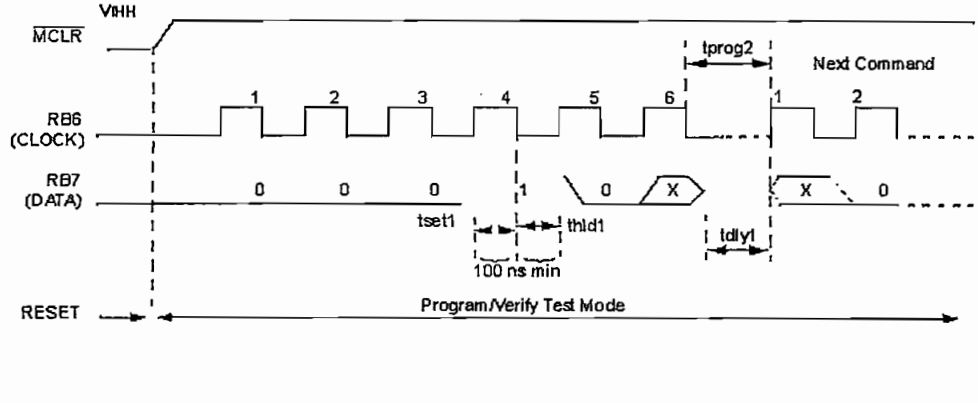


FIGURE 6-7: BEGIN PROGRAMING ONLY COMMAND (PROGRAM/VERIFY)

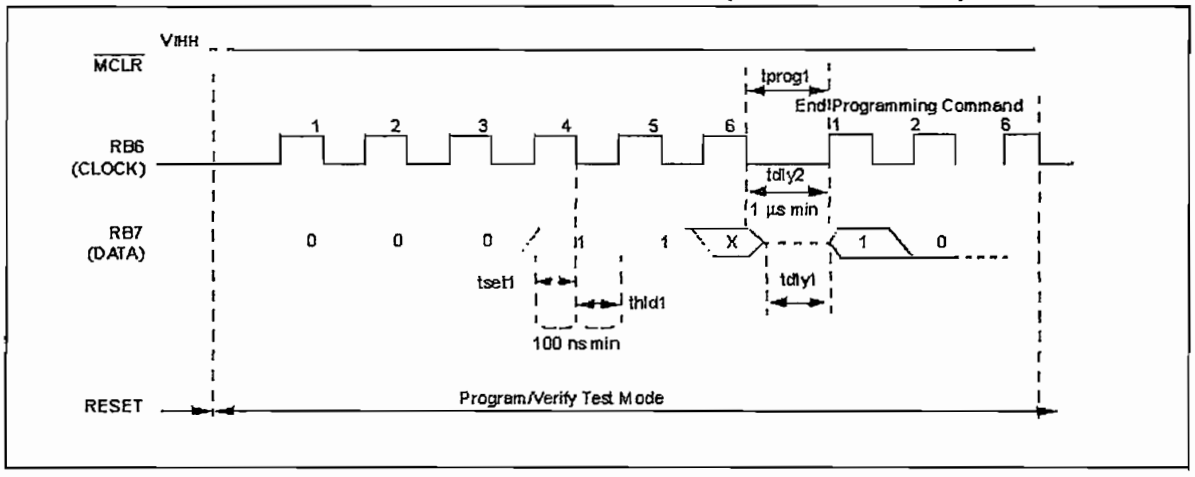


FIGURE 6-8: BULK ERASE PROGRAM MEMORY COMMAND (PROGRAM/VERIFY)

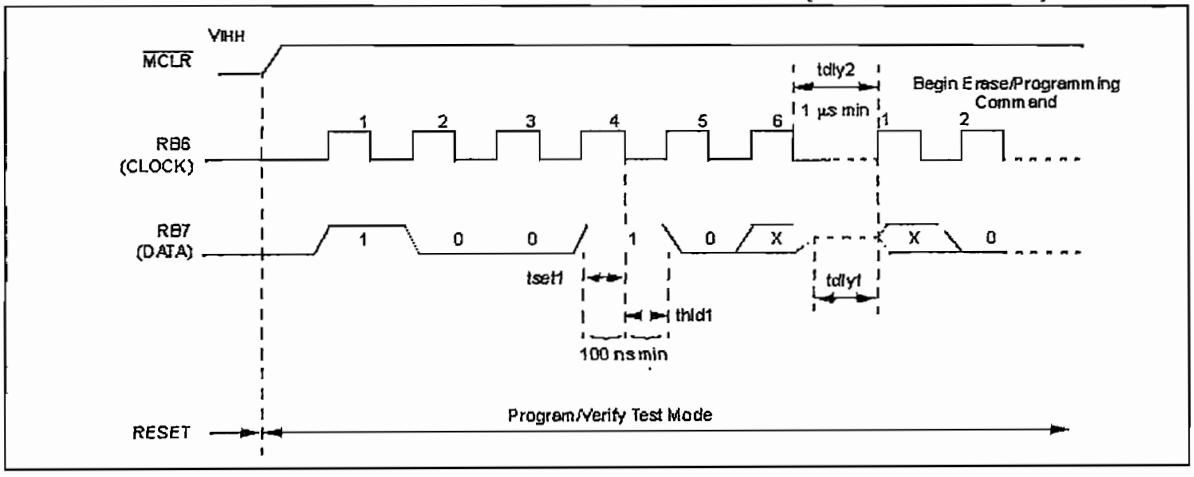


FIGURE 6-9: BULK ERASE DATA MEMORY COMMAND (PROGRAM/VERIFY)

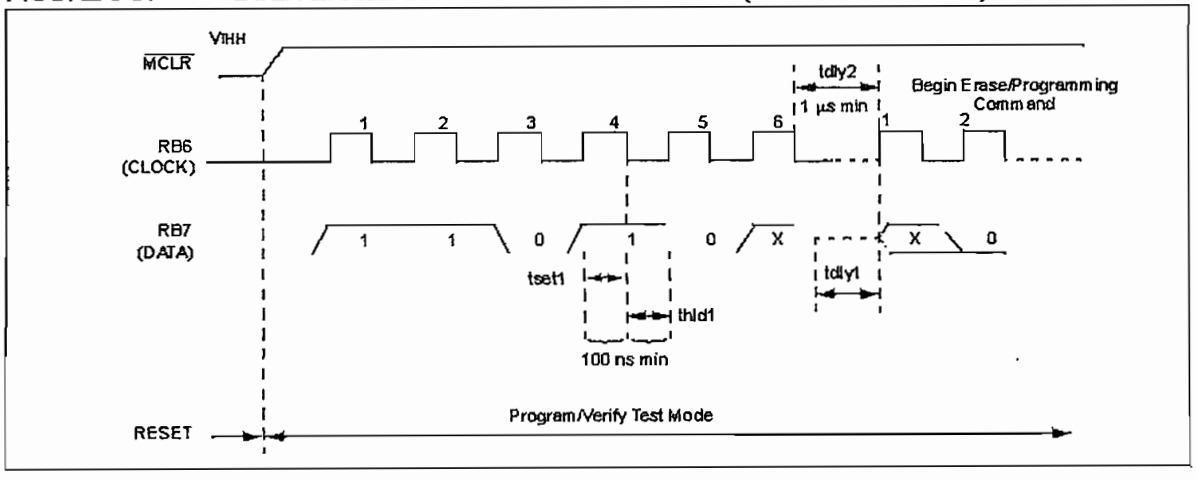
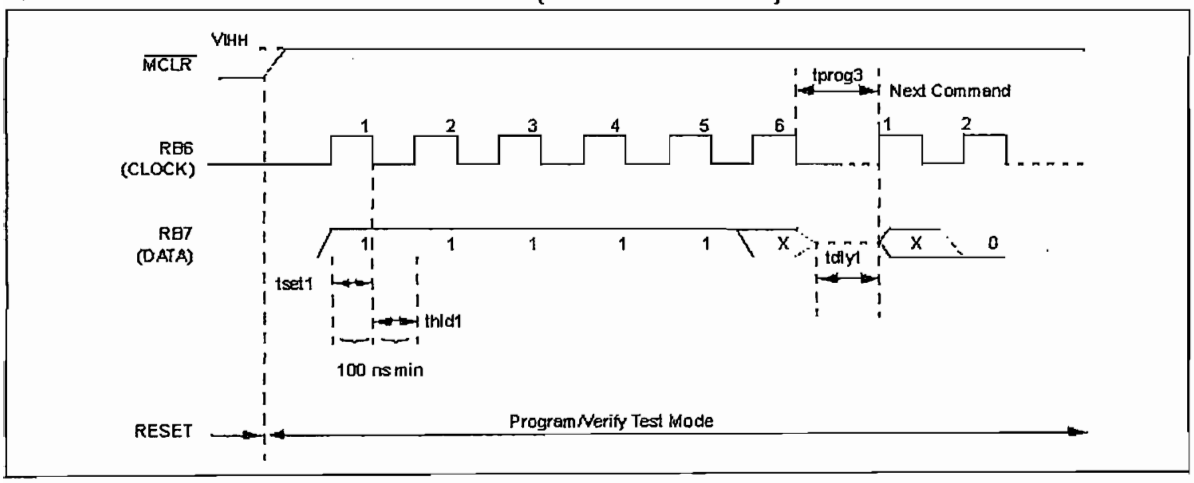


FIGURE 6-10: CHIP ERASE COMMAND (PROGRAM/VERIFY)



ANEXO B

ESPECIFICACIONES DEL OPTO AISLADOR NTE3045

NTE
LECTRONICS, INC.
4 FARRAND STREET
LOOMFIELD, NJ 07003
973) 748-5089

NTE3045
Optoisolator
Silicon NPN Darlington Phototransistor Output

Description:

The NTE3045 is a gallium arsenide LED optically coupled to a Silicon Photo Darlington transistor in a 6–Lead DIP type package designed for applications requiring electrical isolation, high breakdown voltage, and high current transfer ratios. Characterized for use as telephone relay drivers but provides excellent performance in interfacing and coupling systems, phase and feedback controls, solid state relays, and general purpose switching circuits.

Features:

- High Sensitivity to Low Input Drive Current
- High Collector–Emitter Breakdown Voltage: $V_{(BR)CEO}$ = 80V (Min)
- High Input–Output Isolation Guaranteed: V_{ISO} = 7500V (Peak)

Absolute Maximum Ratings: (T_A = +25°C, unless otherwise specified)

Input LED

Reverse Voltage, V_R 3V
Continuous Forward Current, I_F 60mA
LED Power Dissipation (T_A = +25°C with Negligible Power in Output Detector), P_D 120mW
Derate Above 25°C . . . . . 1.41mW/°C

Output Detector

Collector–Emitter Voltage, V_{CEO} 80V
Emitter–Collector Voltage, V_{ECO} 5V
Detector Power Dissipation (T_A = +25°C with Negligible Power in Input LED), P_D 150mW
Derate Above 25°C . . . . . 1.76mW/°C

Total Device

Isolation Surge Voltage (Peak AC Voltage, 60Hz, 1sec Duration, Note 1), V_{ISO} 7500V
Total Device Power Dissipation (T_A = +25°C), P_D 250mW
Derate Above 25°C . . . . . 2.94mW/°C
Operating Temperature Range, T_A . . . . . –55° to +100°C
Storage Temperature Range, T_{stg} . . . . . –55° to +150°C
Lead Temperature (During Soldering for 10sec, 1/16" from Case), T_L . . . . . +260°C

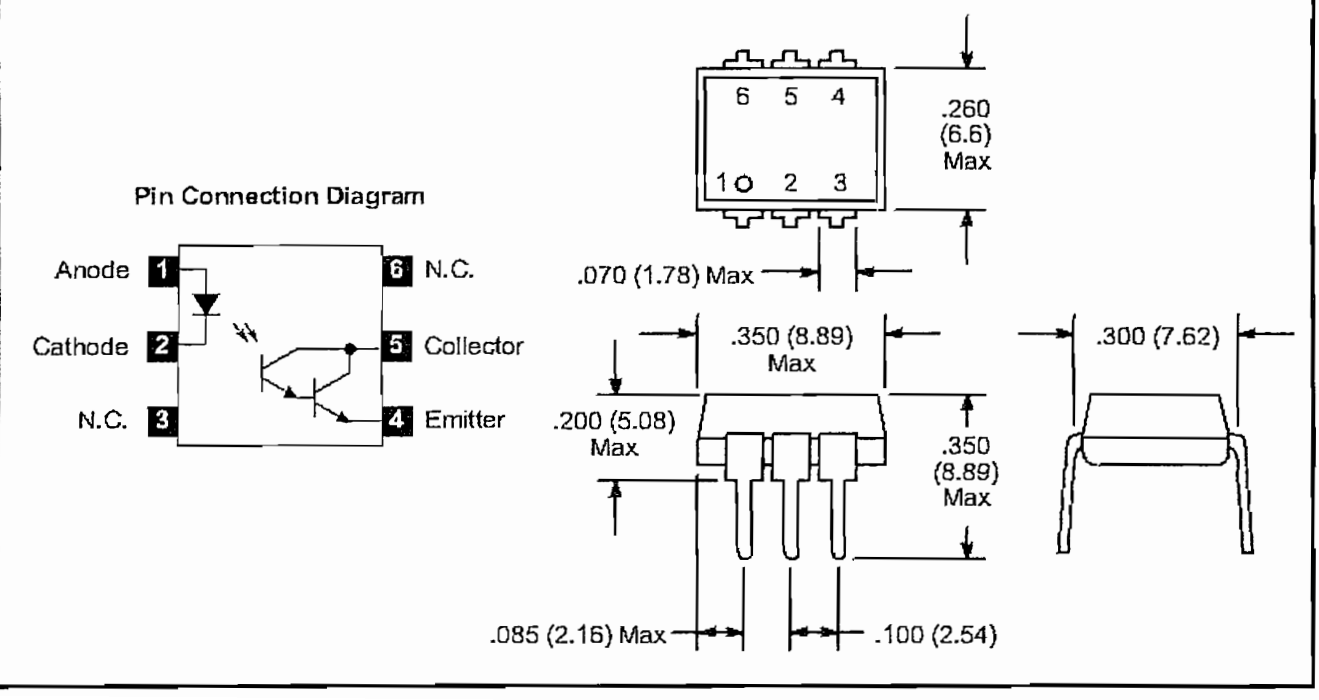
Note 1. Isolation surge voltage is an internal device dielectric breakdown rating. For this test, Pin1 and Pin2 are common, and Pin4, Pin5, and Pin6 are common.

Electrical Characteristics: (T_A = +25°C unless otherwise specified)

Parameter	Symbol	Test Conditions	Min	Typ	Max	Unit
Input LED						
Reverse Leakage Current	I_R	V_R = 3V	–	0.05	10	µA
Forward Voltage	V_F	I_F = 10mA	–	1.15	2.0	V
Capacitance	C	V_R = 0, f = 1MHz	–	18	–	pF
Photodarlington (T_A = +25°C, I_F = 0 unless otherwise specified)						
Collector–Emitter Dark Current	I_{CEO}	V_{CE} = 60V	–	–	1	µA
Collector–Emitter Breakdown Voltage	$V_{(BR)CEO}$	I_C = 1mA	80	–	–	V
Emitter–Collector Breakdown Voltage	$V_{(BR)ECO}$	I_E = 100µA	5	–	–	V
Coupled (T_A = +25°C unless otherwise specified)						
Collector Output Current	I_C	V_{CE} = 1.5V, I_F = 10mA	50	–	–	mA
Isolation Surge Voltage	V_{ISO}	60Hz Peak AC, 5sec, Note 2, Note 3	7500	–	–	V
Isolation Resistance	R_{ISO}	V = 500V, Note 2	–	10^{11}	–	Ω
Isolation Capacitance	C_{ISO}	v = 0, f = 1MHz, Note 2	–	0.2	–	pF
Switching						
Tum–On Time	t_{on}	V_{CC} = 10V, R_L = 100Ω, I_F = 5mA	–	3.5	–	µs
Tum–Off Time	t_{off}		–	95	–	µs
Rise Time	t_r		–	1	–	µs
Fall Time	t_f		–	2	–	µs

Note 2. For this test LED Pin1 and Pin2 are common and Phototransistor Pin4 and Pin5 are common.

Note 3. Isolation Surge Voltage, V_{ISO}, is an internal device dielectric breakdown rating.



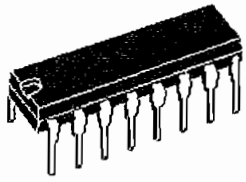
ANEXO C

ESPECIFICACIONES DEL DRIVER AMPLIFICADOR DE CORRIENTE ULN2001A

ULN2001A-ULN2002A
ULN2003A-ULN2004A

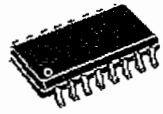
SEVEN DARLINGTON ARRAYS

- SEVEN DARLINGTONS PER PACKAGE
- OUTPUT CURRENT 500mA PER DRIVER (600mA PEAK)
- OUTPUT VOLTAGE 50V
- INTEGRATED SUPPRESSION DIODES FOR INDUCTIVE LOADS
- OUTPUTS CAN BE PARALLELED FOR HIGHER CURRENT
- TTL/CMOS/PMOS/DTL COMPATIBLE INPUTS
- INPUTS PINNED OPPOSITE OUTPUTS TO SIMPLIFY LAYOUT



DIP16

ORDERING NUMBERS: ULN2001A/2A/3A/4A



SO16

ORDERING NUMBERS: ULN2001D/2D/3D/4D

DESCRIPTION

The ULN2001A, ULN2002A, ULN2003 and ULN2004A are high voltage, high current darlington arrays each containing seven open collector darlington pairs with common emitters. Each channel rated at 500mA and can withstand peak currents of 600mA. Suppression diodes are included for inductive load driving and the inputs are pinned opposite the outputs to simplify board layout.

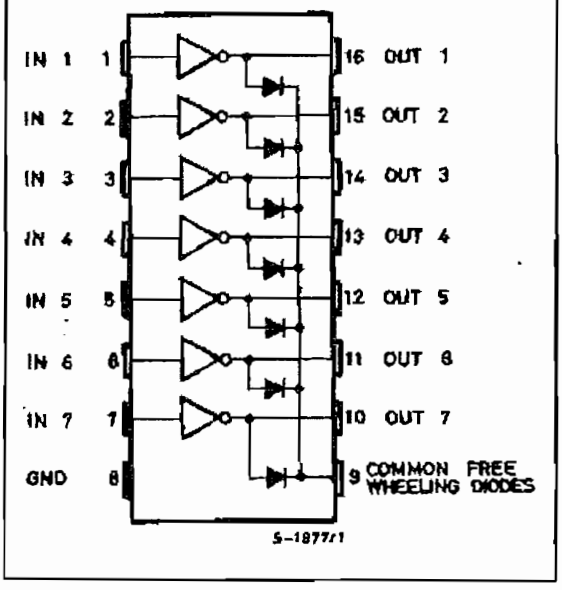
The four versions interface to all common logic families :

ULN2001A	General Purpose, DTL, TTL, PMOS, CMOS
ULN2002A	14-25V PMOS
ULN2003A	5V TTL, CMOS
ULN2004A	6–15V CMOS, PMOS

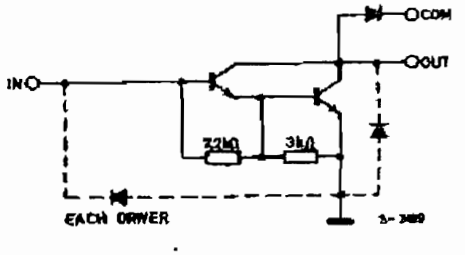
These versatile devices are useful for driving a wide range of loads including solenoids, relays DC motors, LED displays filament lamps, thermal printheads and high power buffers.

The ULN2001A/2002A/2003A and 2004A are supplied in 16 pin plastic DIP packages with a copper leadframe to reduce thermal resistance. They are available also in small outline package (SO-16) as ULN2001D/2002D/2003D/2004D.

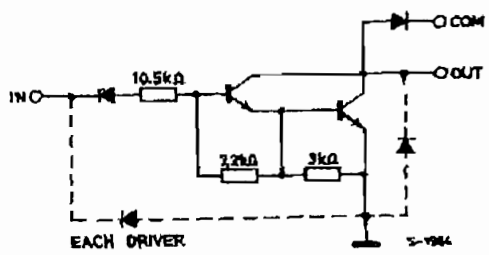
PIN CONNECTION



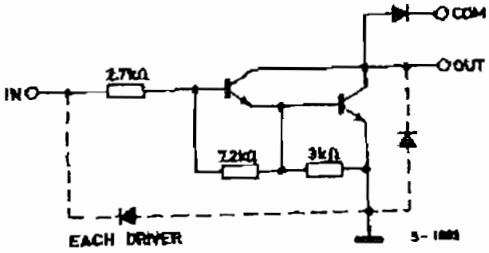
SCHEMATIC DIAGRAM



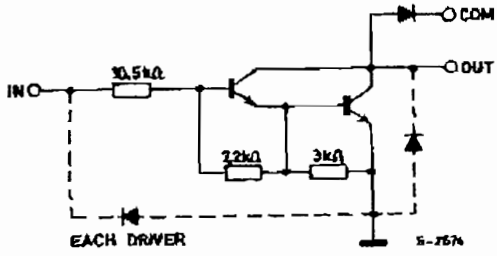
Series ULN-2001A
(each driver)



Series ULN-2002A
(each driver)



Series ULN-2003A
(each driver)



Series ULN-2004A
(each driver)

ABSOLUTE MAXIMUM RATINGS

Symbol	Parameter	Value	Unit
V_o	Output Voltage	50	V
V_{in}	Input Voltage (for ULN2002A/D - 2003A/D - 2004A/D)	30	V
I_c	Continuous Collector Current	500	mA
I_b	Continuous Base Current	25	mA
T_{amb}	Operating Ambient Temperature Range	– 20 to 85	°C
T_{stg}	Storage Temperature Range	– 55 to 150	°C
T_j	Junction Temperature	150	°C

THERMAL DATA

Symbol	Parameter		DIP16	SO16	Unit
$R_{th\ j\text{-}amb}$	Thermal Resistance Junction-ambient	Max.	70	120	°C/W

ELECTRICAL CHARACTERISTICS (T_{amb} = 25°C unless otherwise specified)

Symbol	Parameter	Test Conditions	Min.	Typ.	Max.	Unit	Fig.
I_{CEX}	Output Leakage Current	V_{CE} = 50V			50	µA	1a
		T_{amb} = 70°C, V_{CE} = 50V			100	µA	1a
		T_{amb} = 70°C for ULN2002A V_{CE} = 50V, V_i = 6V			500	µA	1b
		for ULN2004A V_{CE} = 50V, V_i = 1V			500	µA	1b
$V_{CE(sat)}$	Collector-emitter Saturation Voltage	I_C = 100mA, I_B = 250µA		0.9	1.1	V	2
		I_C = 200 mA, I_B = 350µA		1.1	1.3	V	2
		I_C = 350mA, I_B = 500µA		1.3	1.6	V	2
$I_{i(on)}$	Input Current	for ULN2002A, V_i = 17V		0.82	1.25	mA	3
		for ULN2003A, V_i = 3.85V		0.93	1.35	mA	3
		for ULN2004A, V_i = 5V		0.35	0.5	mA	3
		V_i = 12V		1	1.45	mA	3
$I_{i(off)}$	Input Current	T_{amb} = 70°C, I_C = 500µA	50	65		µA	4
$V_{i(on)}$	Input Voltage	V_{CE} = 2V for ULN2002A				V	5
		I_C = 300mA			13		
		for ULN2003A					
		I_C = 200mA			2.4		
		I_C = 250mA			2.7		
		I_C = 300mA			3		
		for ULN2004A					
		I_C = 125mA			5		
		I_C = 200mA			6		
		I_C = 275mA			7		
		I_C = 350mA			8		
h_{FE}	DC Forward Current Gain	for ULN2001A V_{CE} = 2V, I_C = 350mA	1000				2
C_i	Input Capacitance			15	25	pF	
t_{PLH}	Turn-on Delay Time	0.5 V_i to 0.5 V_o		0.25	1	µs	
t_{PHL}	Turn-off Delay Time	0.5 V_i to 0.5 V_o		0.25	1	µs	
I_R	Clamp Diode Leakage Current	V_R = 50V			50	µA	6
		T_{amb} = 70°C, V_R = 50V			100	µA	6
V_F	Clamp Diode Forward Voltage	I_F = 350mA		1.7	2	V	7

TEST CIRCUITS

Figure 1a.

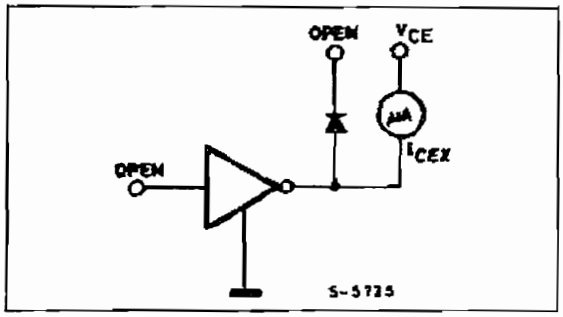


Figure 1b.

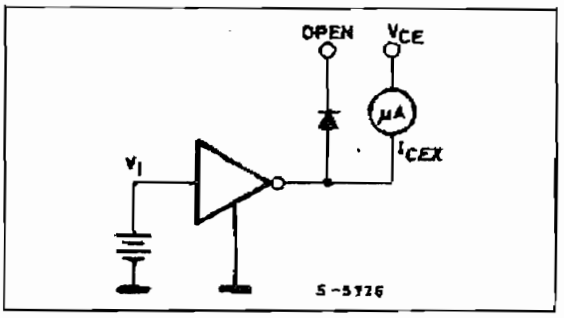


Figure 2.

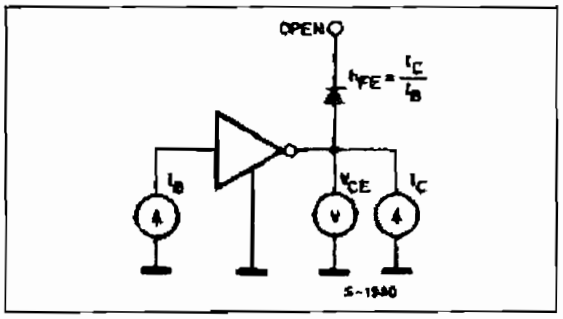


Figure 3.

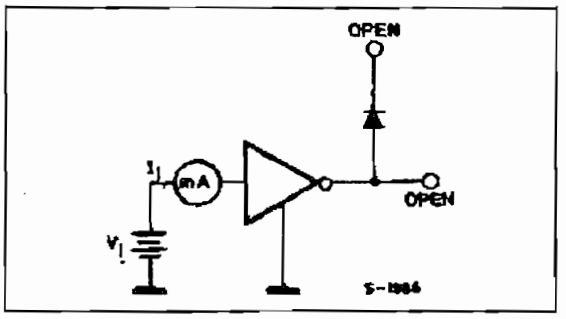


Figure 4.

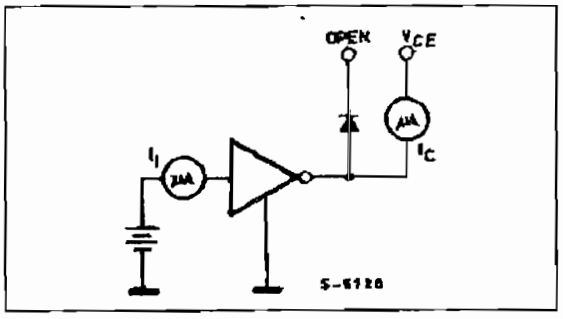


Figure 5.

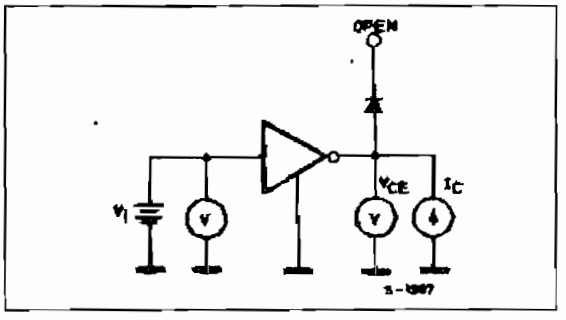


Figure 6.

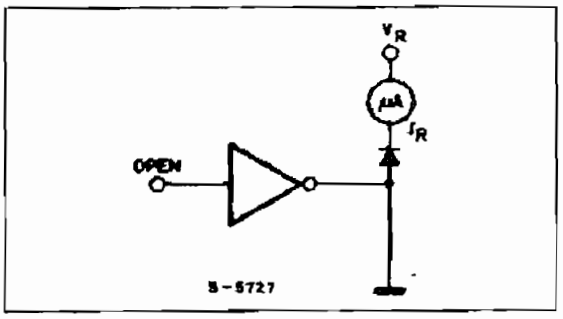


Figure 7.

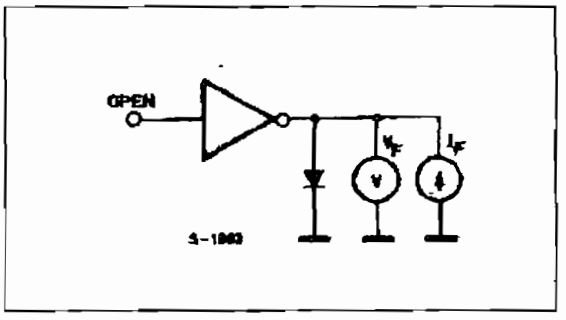


Figure 8: Collector Current versus Input Current

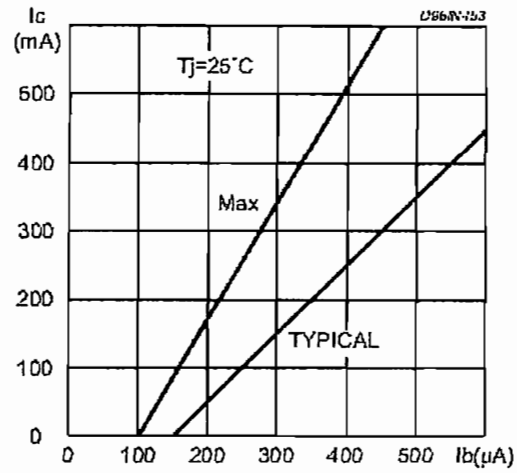


Figure 9: Collector Current versus Saturation Voltage

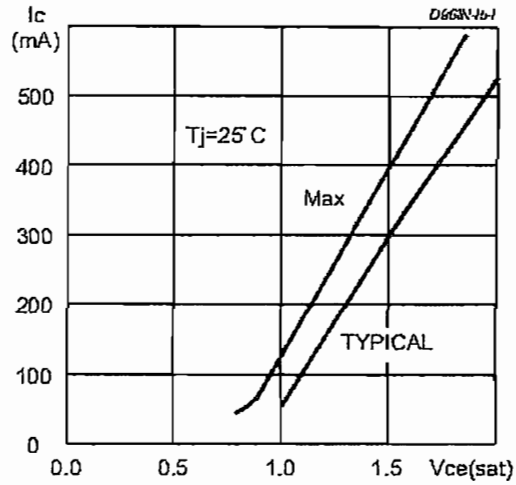


Figure 10: Peak Collector Current versus Duty Cycle

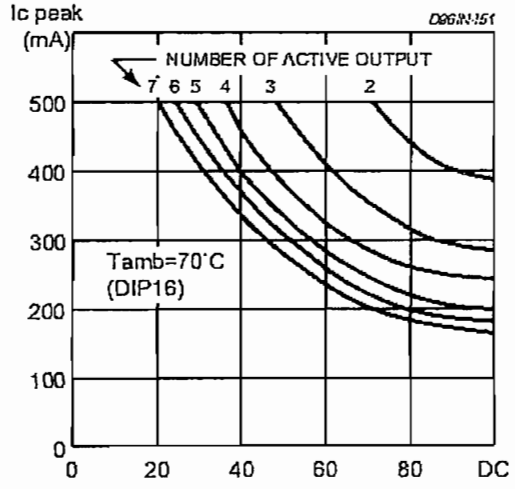
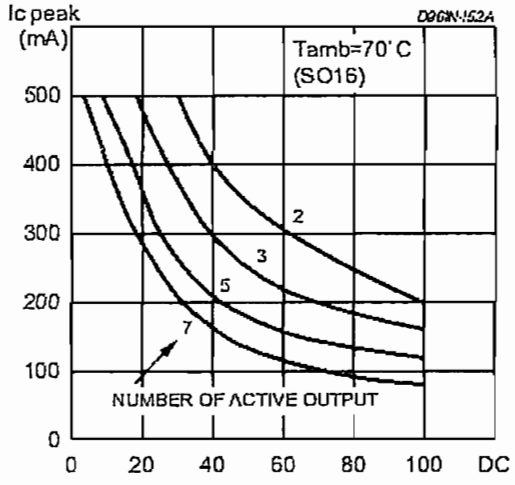
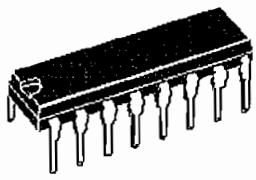


Figure 11: Peak Collector Current versus Duty Cycle

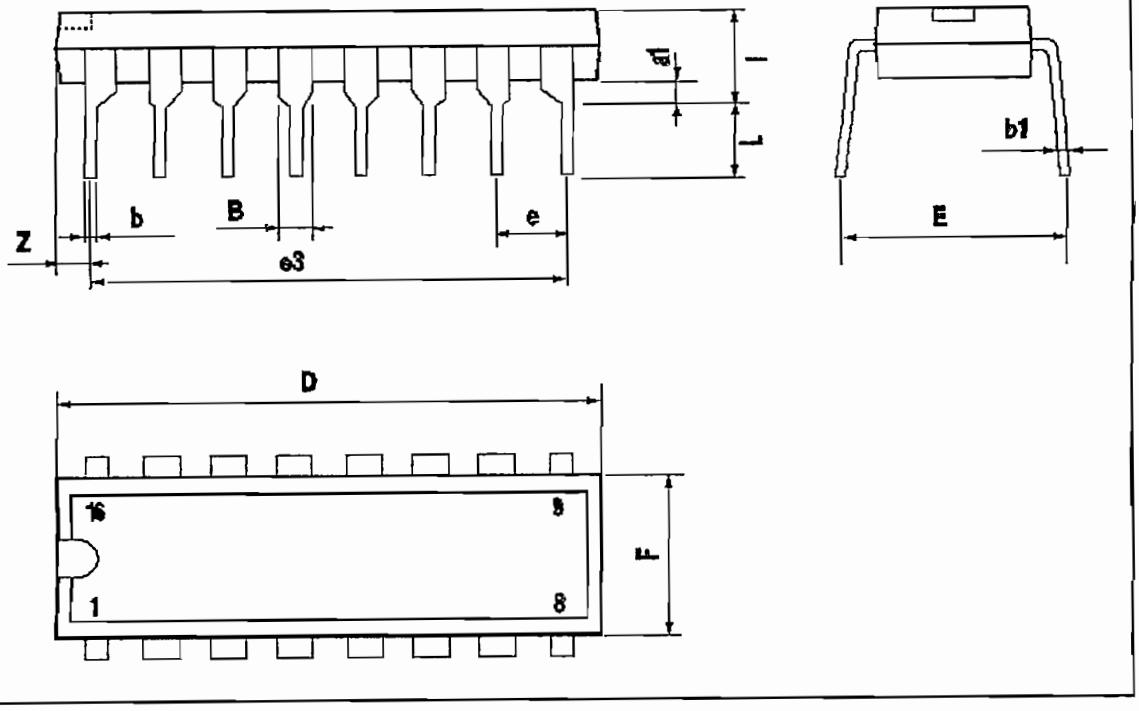


DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
a1	0.51			0.020		
B	0.77		1.65	0.030		0.065
b		0.5			0.020	
b1		0.25			0.010	
D			20			0.787
E		8.5			0.335	
e		2.54			0.100	
e3		17.78			0.700	
F			7.1			0.280
I			5.1			0.201
L		3.3			0.130	
Z			1.27			0.050

OUTLINE AND MECHANICAL DATA

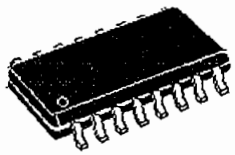


DIP16



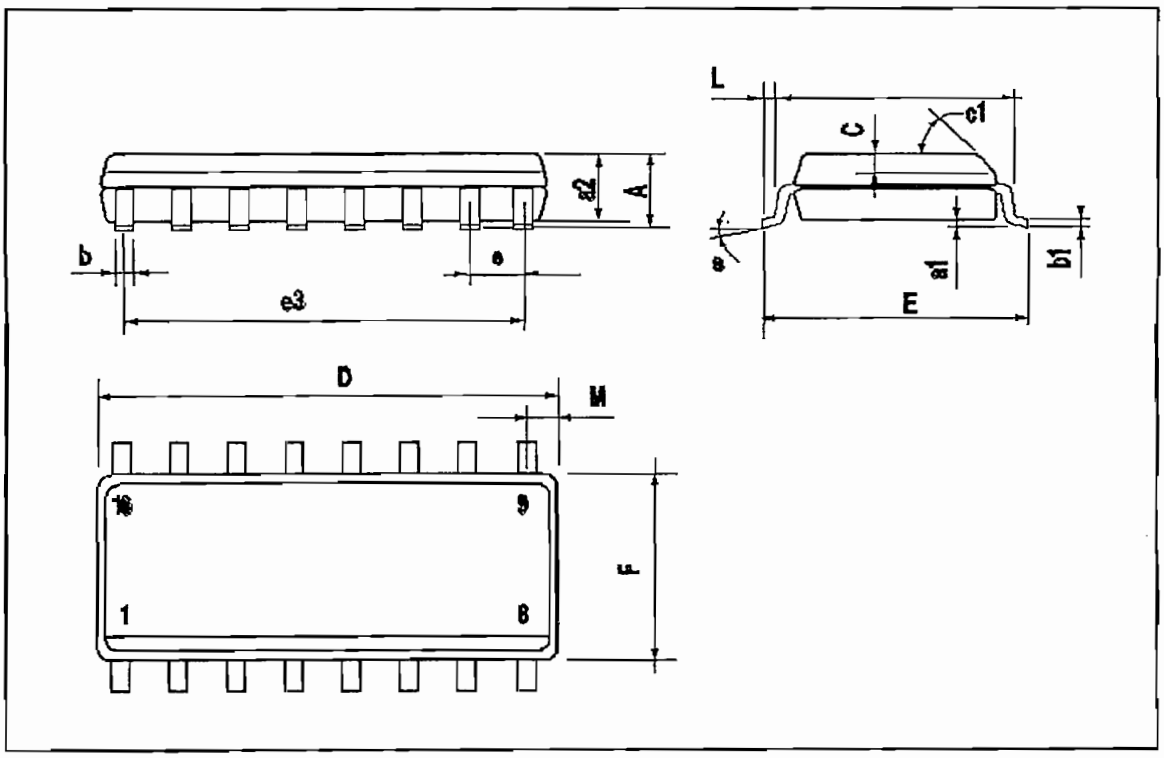
DIM.	mm			inch		
	MIN.	TYP.	MAX.	MIN.	TYP.	MAX.
A			1.75			0.069
a1	0.1		0.25	0.004		0.009
a2			1.6			0.063
b	0.35		0.46	0.014		0.018
b1	0.19		0.25	0.007		0.010
C		0.5			0.020	
c1	45° (typ.)					
D (1)	9.8		10	0.386		0.394
E	5.8		6.2	0.228		0.244
e		1.27			0.050	
e3		8.89			0.350	
F (1)	3.8		4	0.150		0.157
G	4.6		5.3	0.181		0.209
L	0.4		1.27	0.016		0.050
M			0.62			0.024
S	8°(max.)					

OUTLINE AND MECHANICAL DATA



SO16 Narrow

(1) D and F do not include mold flash or protrusions. Mold flash or potrusions shall not exceed 0.15mm (.006inch).



ANEXO D

ESPECIFICACIONES DEL TRANSISTOR AMPLIFICADOR DE POTENCIA B676

NTE
ELECTRONICS, INC.
44 FARRAND STREET
BLOOMFIELD, NJ 07003
(973) 748-5089

NTE261 (NPN) & NTE262 (PNP)
Silicon Complementary Transistors
Darlington Power Amplifier

Description:

The NTE261 (NPN) and NTE262 (PNP) are complementary silicon Darlington power transistors in a TO220 type package designed for general purpose amplifier and low–speed switching applications.

Features:

- High DC Current Gain: h_{FE} = 2500 Typ @ I_C = 4A
- Collector–Emitter Sustaining Voltage: $V_{CEO(sus)}$ = 100V Min @ 100mA
- Low Collector–Emitter Saturation Voltage:
  $V_{CE(sat)}$ = 2V Max @ I_C = 3A
  = 4V Max @ I_C = 5A
- Monolithic Construction with Built–In Base–Emitter Shunt Resistor

Absolute Maximum Ratings:

Collector–Emitter Voltage, V_{CEO} 100V
Collector–Base Voltage, V_{CB} 100V
Emitter–Base Voltage, V_{EB} 5V
Collector Current, I_C
  Continuous 5A
  Peak 8A
Base Current, I_B 120mA
Total Power Dissipation (T_C = +25°C), P_D 65W
  Derate Above 25°C . . . . . 0.52W/°C
Total Power Dissipation (T_A = +25°C), P_D 2W
  Derate Above 25°C . . . . . 0.016W/°C
Unclamped Inductive Load Energy (Note 1), E 50mJ
Operating Junction Temperature range, T_J . . . . . –65° to +150°C
Storage Temperature Range, T_{stg} . . . . . –65° to +150°C
Thermal Resistance, Junction–to–Case, R_{thJC} . . . . . 1.92°C/W
Thermal Resistance, Junction–to–Ambient, R_{thJA} . . . . . 62.5°C/W

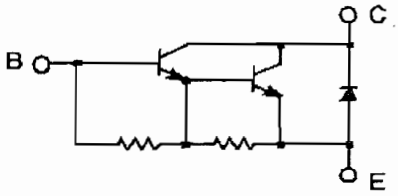
Note 1. I_C = 1A, L = 100mH, P.R.F. = 10Hz, V_{CC} = 20V, R_{BE} = 100Ω.

Electrical Characteristics: (T_C = +25°C unless otherwise specified)

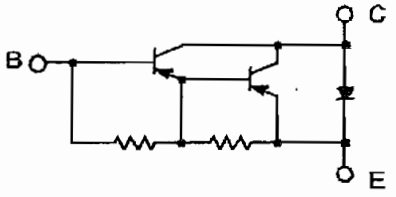
Parameter	Symbol	Test Conditions	Min	Typ	Max	Unit
OFF Characteristics						
Collector–Emitter Sustaining Voltage	$V_{CEO(sus)}$	I_C = 100mA, I_B = 0, Note 2	100	–	–	V
Collector Cutoff Current	I_{CEO}	V_{CE} = 50V, I_B = 0	–	–	0.5	mA
	I_{CBO}	V_{CB} = 100V, I_E = 0	–	–	0.2	mA
Emitter Cutoff Current	I_{EBO}	V_{BE} = 5V, I_C = 0	–	–	2.0	mA
ON Characteristics (Note 2)						
DC Current Gain	h_{FE}	I_C = 0.5A, V_{CE} = 3V	1000	–	–	
		I_C = 3A, V_{CE} = 3V	1000	–	–	
Collector–Emitter Saturation Voltage	$V_{CE(sat)}$	I_C = 3A, I_B = 12mA	–	–	2.0	V
		I_C = 5A, I_B = 20mA	–	–	4.0	V
Base–Emitter ON Voltage	$V_{BE(on)}$	I_C = 3A, V_{CE} = 3V	–	–	2.5	V
Dynamic Characteristics						
Small–Signal Current Gain	$\|h_{fe}\|$	I_C = 3A, V_{CE} = 4V, f = 1MHz	4.0	–	–	
Output Capacitance NTE261	C_{ob}	V_{CB} = 10V, I_E = 0, f = 0.1MHz	–	–	300	pF
NTE262			–	–	200	pF

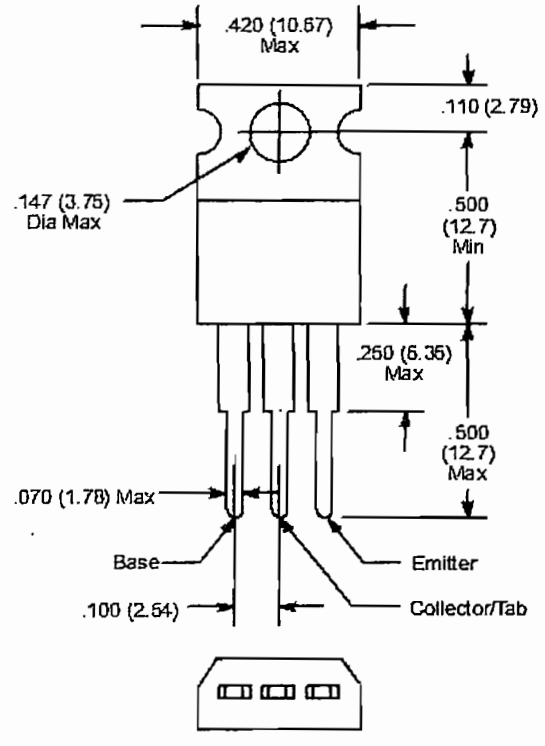
Note 2. Pulse Test: Pulse Width ≤ 300μs, Duty Cycle ≤ 2%.

NTE261



NTE262





ANEXO E

DISEÑO DE LAS TARJETAS DEL PLC

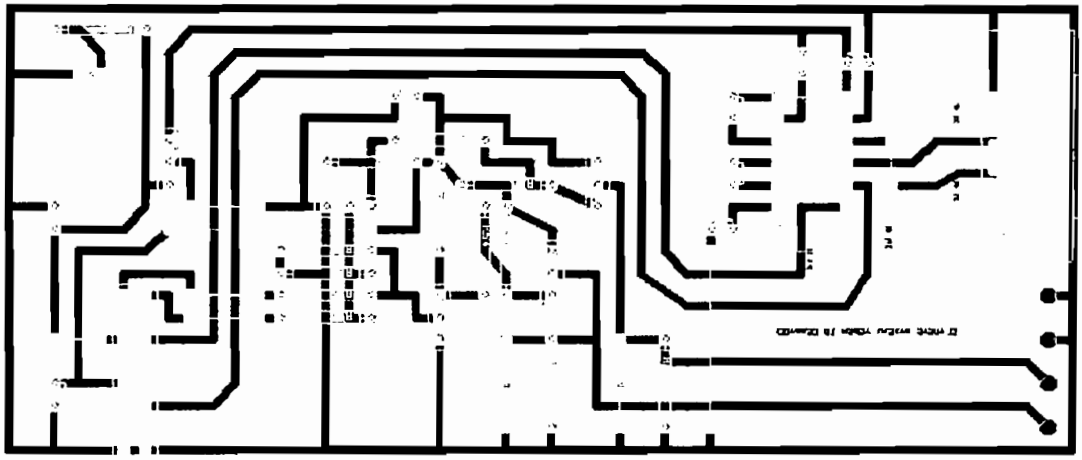


Figura E.8. Tarjeta especial con dos salidas PWM y comunicación serial RS232.

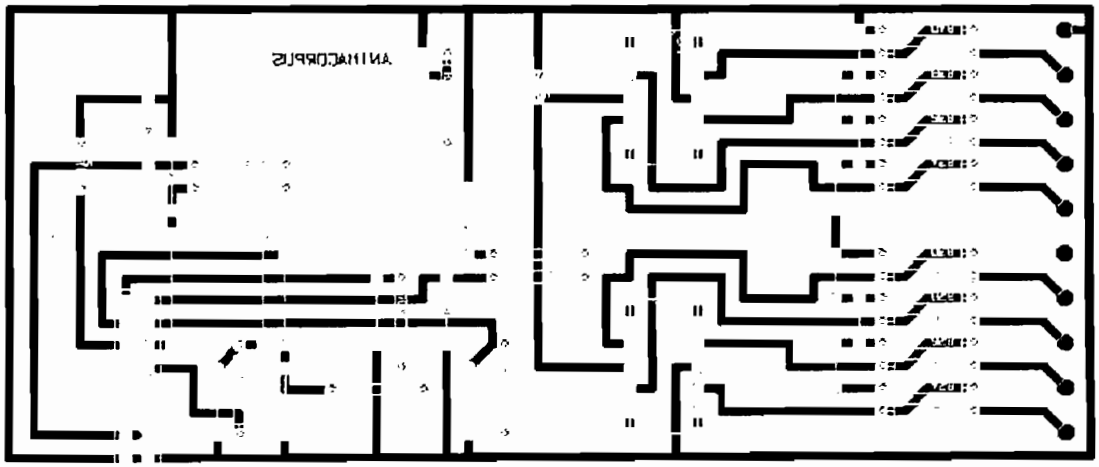


Figura E.9. Tarjeta de entradas analógicas, lado de componentes.

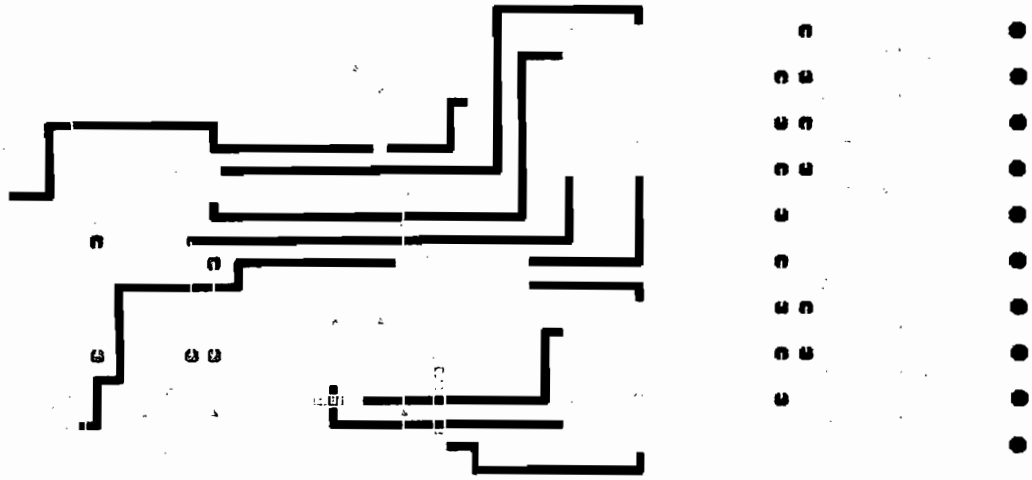


Figura E.10. Tarjeta de entradas analógicas, lado de soldadura.

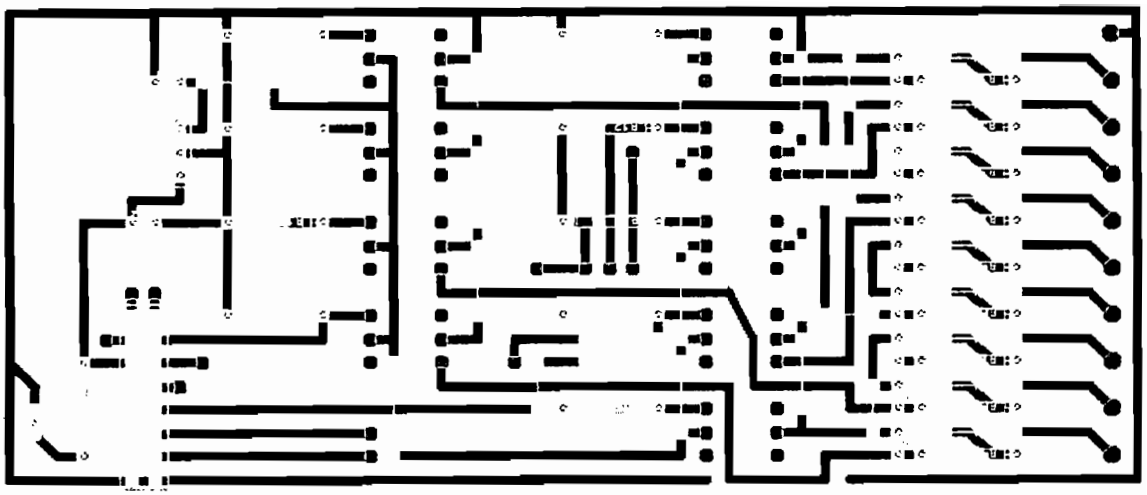


Figura E.5. Tarjeta de entradas digitales, lado de componentes.

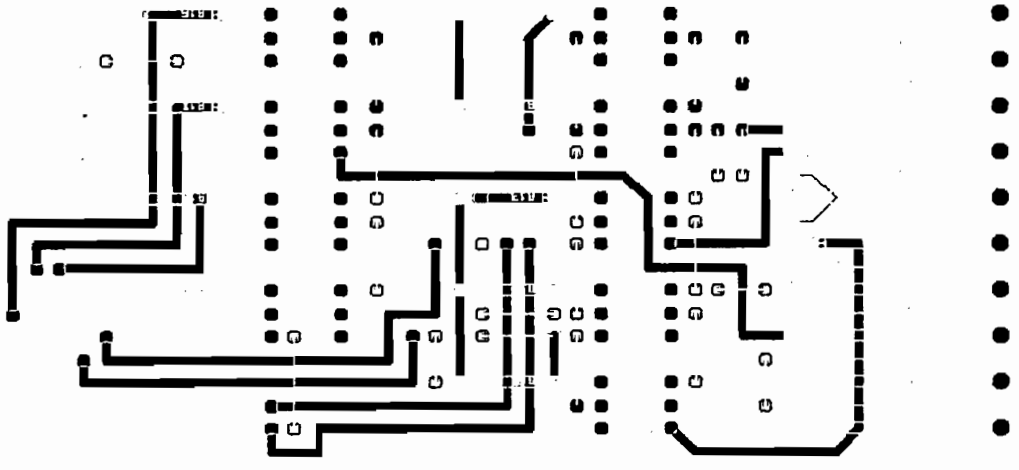


Figura E.6. Tarjeta de entradas digitales, lado de soldadura

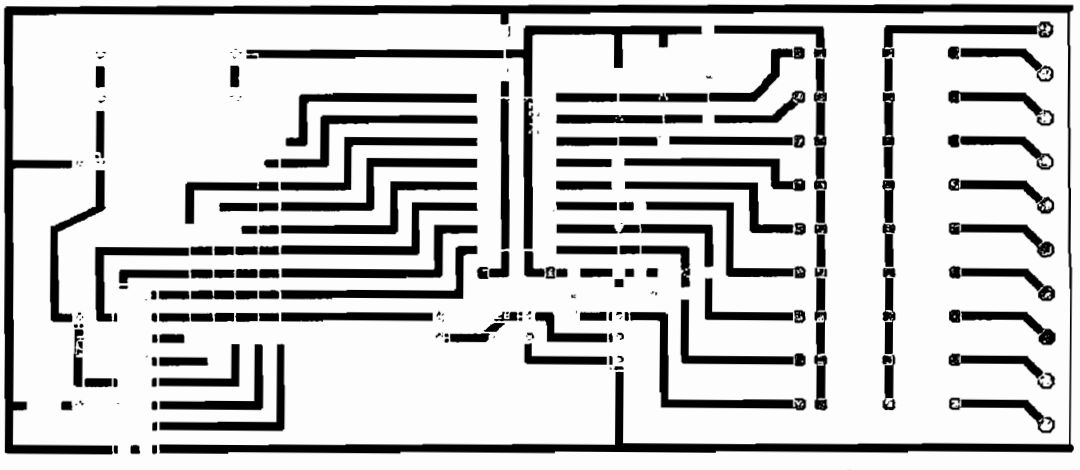


Figura E.7. Tarjeta de salidas digitales a relé.

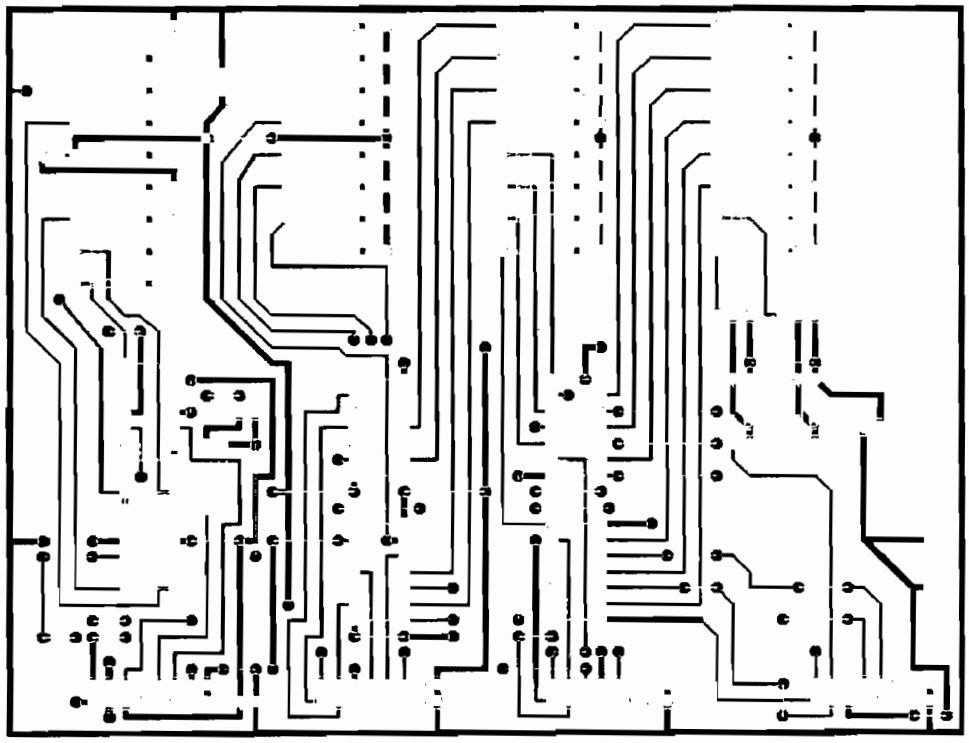


Figura E.3. Tarjeta de visualización, lado de componentes.

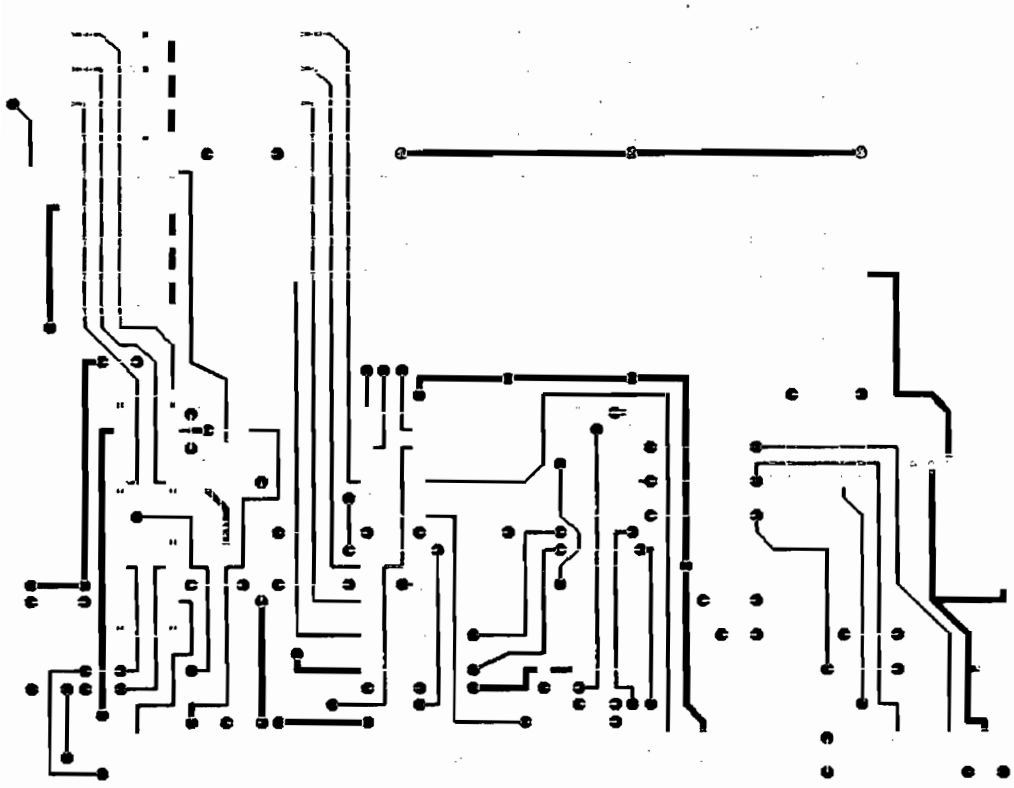


Figura E.4. Tarjeta de visualización, lado de soldadura.

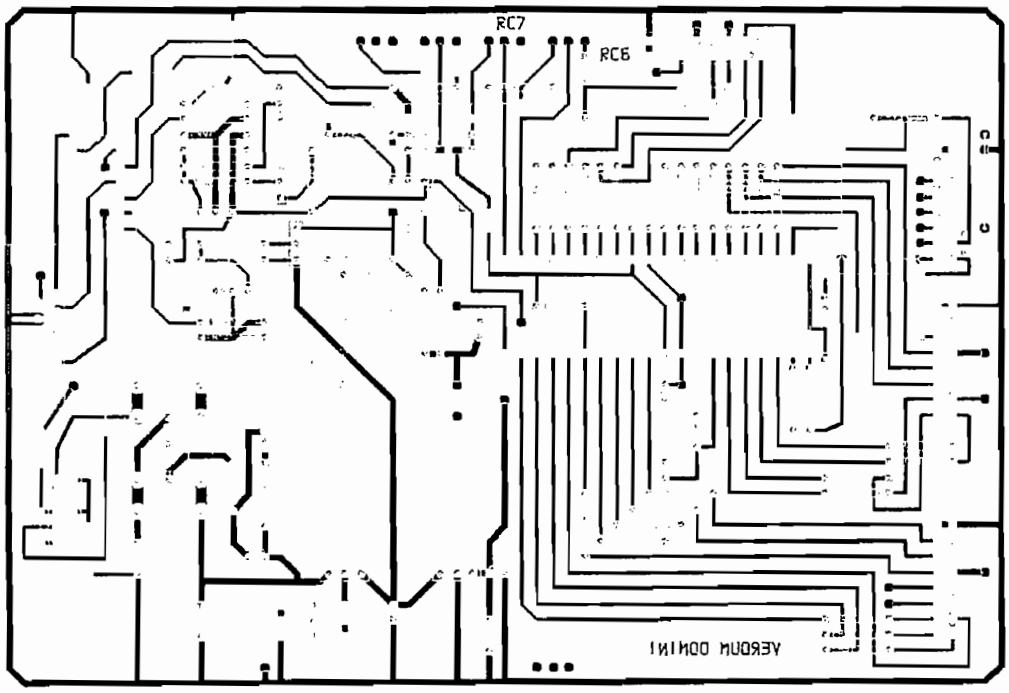


Figura E.1. Tarjeta CPU, lado de componentes

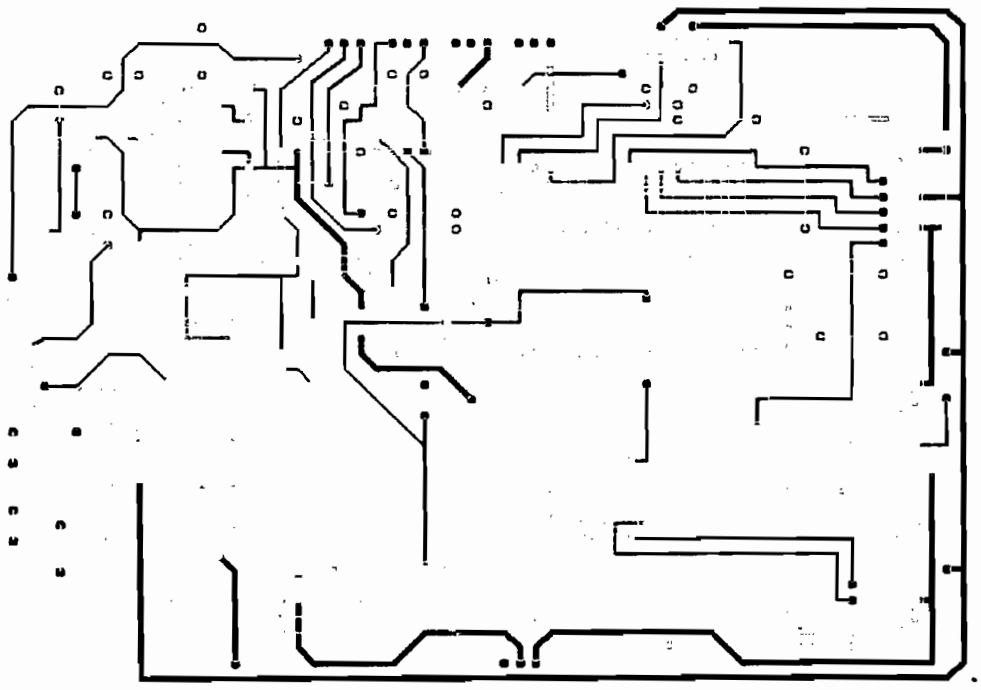


Figura E.2. Tarjeta CPU, lado de soladura.

ANEXO F

FOTOGRAFÍAS DEL PLC DESARROLLADO Y DE LAS TARJETAS QUE LO CONFORMAN

Fotografías del PLC desarrollado y de las tarjetas que lo conforman

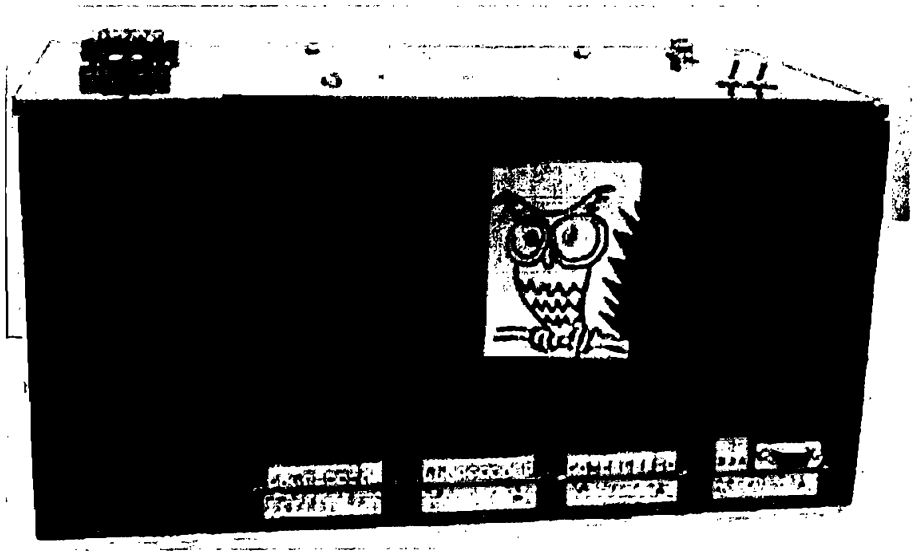


Figura F.1. Vista frontal del PLC.

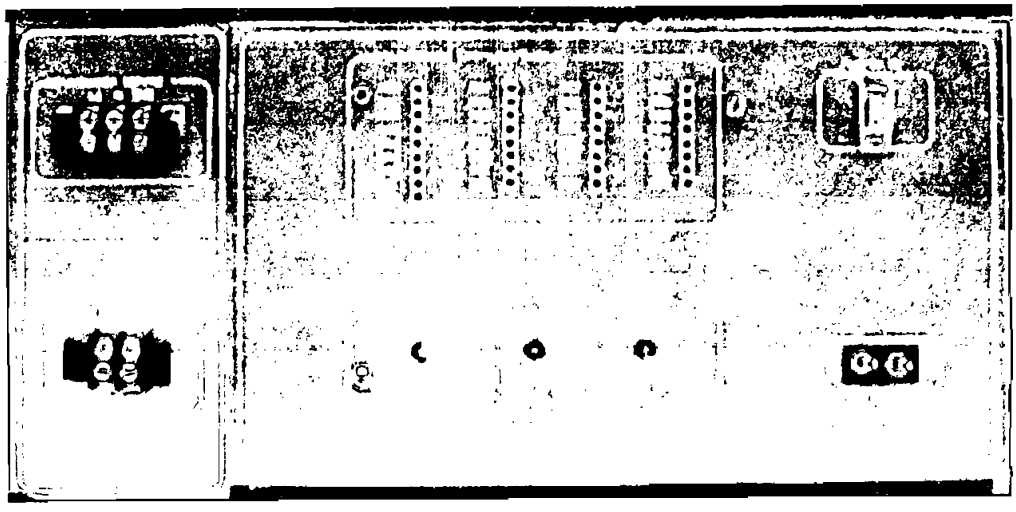


Figura F.2. Vista superior del PLC

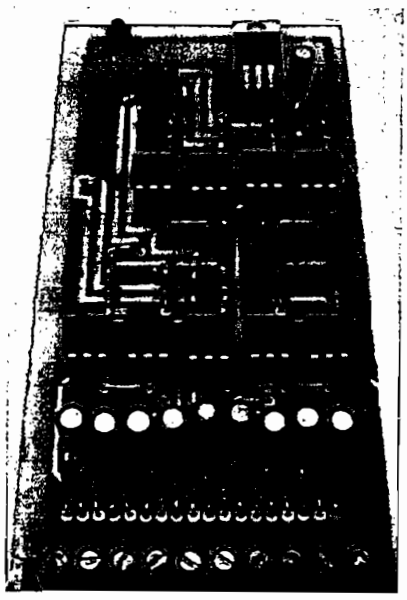


Figura F.3. Tarjeta de entradas digitales

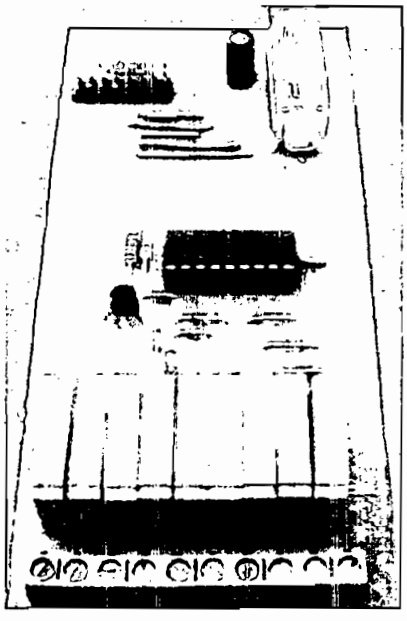


Figura F.4. Tarjeta de salidas digitales a relé

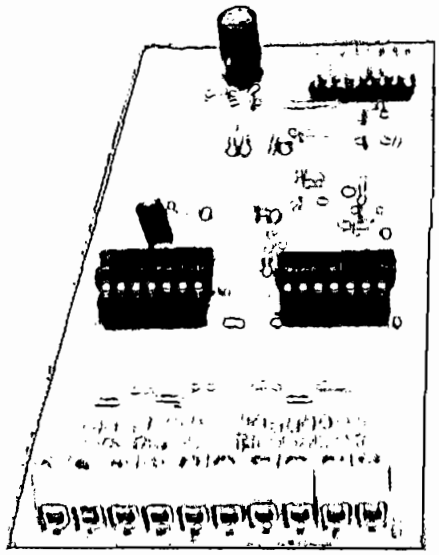


Figura F.5. Tarjeta de entradas analógicas (0 -10V)

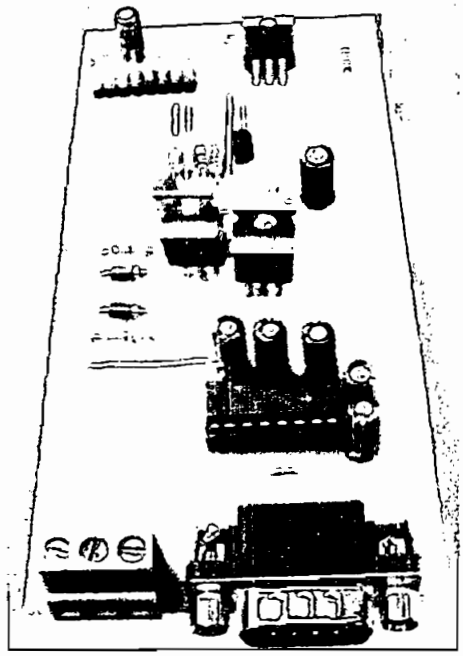


Figura F.6. Tarjeta Especial con dos salidas PWM y un puerto de comunicación serial RS232.