UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DISEÑO Y CONSTRUCCIÓN DE UN CONTROLADOR LÓGICO PROGRAMABLE BASADO EN LA TARJETA MINICON_08GT

TESIS

Que para obtener el título de

INGENIERO ELÉCTRICO Y ELECTRÓNICO

P R E S E N T A N

César García Hernández

Nurivan Armando Lozano Zane

DIRECTOR DE TESIS

M.I. Antonio Salvá Calleja

Ciudad Universitaria, Cd. Mx., 2016

Dedicatoria.

A mis padres Armando y Leticia quienes me han dado todo lo que soy como persona y siempre me acompañan, sigo aprendiendo de ustedes.

A Ana, mi esposa, amiga y compañera, que siempre me ha apoyado en todos los buenos y malos momentos sin esperar nada a cambio.

A mi hermano Atzael, siempre presente en este trabajo.

Nurivan Armando Lozano Zane.

A la Universidad Nacional Autónoma de México y a la Facultad de Ingeniería por tantos años de aprendizaje dentro de sus aulas.

Agradecimientos.

Al maestro Antonio Salvá Calleja por su invaluable tiempo, orientación, seguimiento y asesoría como director de este trabajo.

Al jurado de esta tesis por sus observaciones y comentarios.

A la comunidad académica en general, con quien tuve la oportunidad de tomar clases.

A mi familia y amigos por el ánimo recibido.

A todos ellos, muchas gracias.

Nurivan Armando Lozano Zane.

Dedicatoria.

Dedico de manera especial a mi Padre, a mi Madre y a mi hermano que son personas que me han ofrecido su amor y cariño en todo momento, además, de su apoyo incondicional. ¡Los amo!

A mi novia Leny Palma Pérez Díaz, por su motivación y apoyo para terminar esta Licenciatura, por su amor y cariño. ¡Te amo Makito!

César García Hernández

Agradecimientos.

Le doy gracias a mis padres Socorro Hernández Carreño y Luis Francisco García Mandujano por apoyarme en todo momento y motivarme en los momentos difíciles durante todos estos años, por los valores que me ha inculcado, por haberme dado la oportunidad de tener una educación y por llenar mi vida de alegrías y amor cuando más lo he necesitado.

A mi hermano Luis Francisco García Hernández por ser parte muy importante en mi vida, por su apoyo, sus consejos y regaños en los buenos y malos momentos.

Le agradezco al Maestro Antonio Salvá Calleja por sus consejos, asesorías y por su paciencia para la terminación de este proyecto.

A la Universidad Nacional Autónoma de México y especialmente a la Facultad de Ingeniería por tantos años de aprendizaje.

A todos mis profesores por haber compartido conmigo su conocimiento y amistad.

A Armando por su perseverancia, paciencia y por haber sido un gran compañero de tesis.

A mis tíos, tías y primos por sus consejos, buenos deseos y por ser un ejemplo de desarrollo laboral.

A mis amigos por confiar y creer en mí, por compartir grandes momentos que nunca olvidaré.

A Aldo, Daniel, Arturo (Q. E. P. D.) por vivir grandes experiencias juntos.

César García Hernández

REFERENCIA DE SIGLAS

PLM3:	Controlador Lógico Modular 3
CLS:	Controlador Lógico y Secuencial
PLC:	Controlador Lógico Programable (Programmable Logic Controller)
MCU:	Microcontrolador
SIIL2:	Software de Interpretación de Instrucciones Lógicas
ML:	Módulos Lógicos
HT:	Hardware de la Tarjeta
ST:	Software de la Tarjeta
CC:	Computadora Central
BE:	Bloque de entradas
NFS:	Neutro de la Fuente de Sensores
BS:	Bloque de salidas
FA:	Fuente de Alimentación
VB:	Variables Booleanas
VBE:	Variables Booleanas de Entrada
VBI:	Variables Booleanas Intermediarias
VBS:	Variables Booleanas de Salida
CNA:	Contactos Normalmente Abiertos
SWMANPLM3:	Software Manejador del PLM3
CODM:	Cadena de caracteres del módulo
N:	Número asociado con un módulo
ENTm:	Designaciones asociadas con m variables de entrada
SALn:	Designaciones asociadas con n variables de salida
DAq:	Datos auxiliares
CADBI:	Cadena binaria
INPROG:	Inicio del subprograma principal
FINPP:	Fin del subprograma principal
INMODI:	Inicio del subprograma temporizado
FINMODI:	Fin del subprograma temporizado

RESUMEN

El presente trabajo fue realizado con el objetivo de mostrar el principio de construcción y funcionamiento de un dispositivo el cual es utilizado generalmente en el sector industrial, conocido como Controlador Lógico Programable (PLC por sus siglas en ingles) y no pretende ser una innovación para reemplazar a los PLC utilizados actualmente como son las marcas de Siemens o Allen Bradley, por mencionar algunas.

En el primer capítulo se describe el entorno del uso del PLC en el sector industrial en sus inicios, haciendo énfasis en la necesidad de contar con un dispositivo que facilite el control de diferentes aplicaciones. El segundo capítulo refiere a las características principales de la tarjeta de desarrollo empleada para la realización del PLM3, mientras que en el capítulo tres se expone sobre la estructura y sintaxis de programación que éste debe tener para un funcionamiento correcto del mismo. En el cuarto capítulo se describe a detalle como es la declaración de cada una de las funciones de control que realiza el PLM3 y con las cuales es posible realizar diversas aplicaciones. En el quinto capítulo se enuncia sobre el diseño y construcción del hardware adicional implementado a la tarjeta de desarrollo, siendo éste indispensable para que las aplicaciones por programar puedan suceder. En el sexto capítulo se expone sobre el funcionamiento de la interfaz que permitirá realizar la programación de la aplicación deseada. Para complementar este trabajo se realizó un ejemplo de aplicación el cual es descrito en el séptimo y último capítulo, mostrando así el desempeño y potencialidad que tiene el PLM3. Es importante señalar que este es un dispositivo sencillo ya que no cuenta con tarjetas para monitorización y/o control de señales analógicas o bien con un protocolo de comunicación para que pueda ser monitorizado de una PC remota.

MOTIVACIÓN

Esta tesis surge por el interés que tuvimos por conocer la versatilidad de proyectos y/o aplicaciones que puede tener la tarjeta de desarrollo MINICON_08GT la cual fue utilizada durante el ciclo escolar, y no solo conservar la idea de programación en lenguaje “ensamblador” o “C” de la misma. Es por esto que decidimos realizar una nueva versión del PLM utilizando la tarjeta mencionada, desarrollando el hardware adicional para su funcionamiento y realizando las actualizaciones necesarias al software de programación del primer prototipo.

ÍNDICE

CAPÍTULO 1 **1**

INTRODUCCIÓN **1**
1.1 ANTECEDENTES. 3

CAPÍTULO 2 **4**

CARACTERÍSTICAS FUNCIONALES DE LA TARJETA MINICON_08GT. **4**
2.1 HARDWARE DE LA TARJETA (HT). 5
2.2 SOFTWARE DE LA TARJETA (ST). 6
2.2.1 Software de base de la Tarjeta MINICON_08GT. 6
2.2.2 Software Manejador de la Tarjeta MINICON_08GT. 6

CAPÍTULO 3. **10**

ESTRUCTURA Y FUNCIONAMIENTO DEL PROGRAMADOR LÓGICO MODULAR PLM3. **10**
3.1 ESTRUCTURA DEL PROGRAMADOR LÓGICO MODULAR PLM3. 10
3.1.1 Computadora Central (CC). 10
11
3.1.2 Bloque de entradas (BE). 12
3.1.3 Bloque de salidas (BS). 12
3.1.4 Fuente de Alimentación (FA). 13
3.2 VARIABLES BOOLEANAS DEL PLM3. 13
3.2.1 Variables Booleanas de Entrada (VBE). 13
3.2.2 Variables Booleanas de Salida (VBS). 14
3.2.3 Variables booleanas Intermediarias (VBI). 14
3.3 CARACTERÍSTICAS GENERALES DE LOS MÓDULOS LÓGICOS. 15
3.3.1 Formas sintácticas asociadas con los Módulos Lógicos. 17
3.4 FORMATO DE UN PROGRAMA EN SIIL2 19
3.4.1 Forma de un programa fuente en SIIL2 21

CAPÍTULO 4 **25**

DESCRIPCIÓN DE LAS FUNCIONES DE CONTROL LÓGICO REALIZABLES POR EL PLM3. **25**
4.1 DESCRIPCIÓN DEL MÓDULO LÓGICO SEGUIDOR. 26
4.2 DESCRIPCIÓN DEL MÓDULO LÓGICO INVERSOR. 27
4.3 DESCRIPCIÓN DE LOS MÓDULOS LÓGICOS QUE REALIZAN COMPUERTAS DE DOS ENTRADAS. 29
4.4 DESCRIPCIÓN DE LOS MÓDULOS LÓGICOS QUE REALIZAN COMPUERTAS DE TRES ENTRADAS. 32
4.5 DESCRIPCIÓN DE LOS MÓDULOS LÓGICOS QUE REALIZAN COMPUERTAS DE CUATRO ENTRADAS. 35
4.6 DESCRIPCIÓN DE LOS MÓDULOS LÓGICOS QUE REALIZAN FLIP-FLOP R-S ASÍNCRONOS. 38

4.7 DESCRIPCIÓN DEL MÓDULO LÓGICO TEMPORIZADOR MONODISPARO (ONE SHOT) DISPARABLE POR NIVEL. 40
4.8 DESCRIPCIÓN DEL MÓDULO LÓGICO QUE REALIZA TEMPORIZADORES CON RETARDO A LA CONEXIÓN (ON-DELAY) O CON RETARDO A LA DESCONEXIÓN (OFF-DELAY). 44
4.9 DESCRIPCIÓN DEL MÓDULO LÓGICO QUE REALIZA UN TEMPORIZADOR ASTABLE. 48
4.10 DESCRIPCIÓN DEL MÓDULO LÓGICO TEMPORIZADOR MONODISPARO (ONE SHOT) DISPARABLE POR FLANCO. 51
4.11 DESCRIPCIÓN DEL MÓDULO LÓGICO QUE REALIZA UN CONTADOR DE EVENTOS 54
4.12 DESCRIPCIÓN DEL MÓDULO LÓGICO QUE REALIZA UN SECUENCIADOR DE ESTADOS 58
4.13 DESCRIPCIÓN DEL MÓDULO LÓGICO QUE REALIZA UNA COMPARACIÓN 64

CAPÍTULO 5. 67

DISEÑO DEL HARDWARE DE ENTRADAS, SALIDAS Y FUENTE DE ALIMENTACIÓN DEL PLM3 67
5.1 COMPUTADORA CENTRAL (CC). 67
5.2 DISEÑO ASOCIADO AL HARDWARE DEL BLOQUE DE ENTRADAS. 68
5.2.1 Circuitos de optoacoplamiento para sensores de entrada con fuente propia. 70
5.2.2 Circuitos de optoacoplamiento por contacto seco. 71
5.3 DISEÑO ASOCIADO AL HARDWARE DEL BLOQUE DE SALIDAS. 74
5.4 DISEÑO ASOCIADO A LA FUENTE DE ALIMENTACIÓN. 77
5.5 INTEGRACIÓN FÍSICA DE LAS TARJETAS DEL PLM3 79

CAPÍTULO 6 80

SOFTWARE MANEJADOR DEL PLM3 80
6.1 DESCRIPCIÓN GENERAL DEL SOFTWARE MANEJADOR DEL PLM3 80
6.2 DESCRIPCIÓN DE LA VENTANA DE EDICIÓN DEL SWMANPLM3 82
6.2.1 Botón *c-pfsil* (Compilación) 83
6.2.2 Botón *e-pfsil* (Compilación y ejecución inmediata) 84
6.2.3 Botón *b-feep* (Borrado de la memoria) 84
6.3 DESCRIPCIÓN DE LA OPCIÓN DEL MENU "ABR" DEL SWMANPLM3 85
6.4 REPORTE DE ERRORES 87
6.5 TESTIFICACIÓN DE DISPONIBILIDAD DE LA COMPUTADORA CENTRAL DEL PLM3 PARA RECIBIR COMANDOS DESDE EL SWMANPLM3 89
6.6 BORRADO DE LA MEMORIA NO VOLÁTIL 90
6.7 EJECUCIÓN DE UN PROGRAMA FUENTE CON EL SWMANPLM3 91
6.8 EJECUCIÓN AUTÓNOMA DE UN PROGRAMA EN EL PLM3 93

CAPÍTULO 7 97

EJEMPLO DE APLICACIÓN 97
7.1 DESCRIPCIÓN DEL PROCESO POR AUTOMATIZAR EMPLEANDO EL PLM3. 97
7.2 DIAGRAMA A BLOQUES DEL EJEMPLO DE APLICACIÓN. 100
7.3 VARIABLES ASOCIADAS AL EJEMPLO DE APLICACIÓN CON EL PLM3. 102
7.4 DESCRIPCIÓN DEL DIAGRAMA A BLOQUES. 102
7.5 PROGRAMA EN SIIL2 DEL EJEMPLO DE APLICACIÓN EN EL PLM3. 104

CONCLUSIONES **106**

CONTRIBUCIONES **107**

TRABAJO FUTURO **107**

BIBLIOGRAFÍA **108**

ÍNDICE DE FIGURAS **109**

ÍNDICE DE TABLAS **115**

CAPÍTULO 1

INTRODUCCIÓN

En la industria existen procesos o subprocesos para fines de control y/o monitorización de diferentes aplicaciones, que tienen asociados a ellos diferentes variables industriales tales como: temperatura, presión, nivel, humedad, peso, entre otros. Considerando cada una de las variables anteriores para un fin determinado, se implementan para estos subprocesos sendos sistemas de control de lazo cerrado analógicos o digitales; o bien, sistemas de control lógico y secuencial validados con un Controlador Lógico y Secuencial (CLS).

En la década de los sesenta, los sistemas de control lógico implementados para las diferentes aplicaciones basaban su funcionamiento en circuitos eléctricos con relevadores electromecánicos, sensores e interruptores que fueran normalmente conocidos como sistemas de lógica alambrada.

A principios de los setenta, el sector industrial busca cambiar los CLS realizados con lógica alambrada por algo más eficaz, con el que se pudieran reducir los tiempos muertos de operación (resultado de cuando se necesitaba realizar una modificación o bien para la detección de fallas), aumentar la velocidad de producción con una mayor calidad y que pudiesen controlar el número de variables necesarias como entradas y salidas de dicho sistema. Por lo anterior, puede pensarse en un sistema genérico para control lógico que contuviera un optoacoplador para cada entrada y sendos relevadores de baja potencia para cada salida, ligado todo esto con una computadora que ejecutara un programa que validara el control lógico para un proceso dado, desarrollando de esta manera el primer prototipo de Controlador Lógico Programable (PLC por sus siglas en inglés).

Con ayuda de los PLC, la automatización de procesos se desarrolló aún más y con esto comenzaron a surgir empresas dedicadas a producir y crear sus propios PLC junto con los elementos necesarios para poder implementarlos en los procesos o subprocesos e incluso algunas empresas comenzaron a desarrollar y a fabricar sus propias máquinas para lograr un fin específico.

Con base en el principio de funcionamiento ya mencionado, los PLC fueron actualizándose y hoy en día podemos encontrar desde un PLC básico con solo un par de entradas y salidas (analógicas o digitales) hasta un PLC con "N" entradas y "M" salidas (analógicas y digitales) que cuente con protocolo de comunicación, pantallas para monitorización y control (HMI) y un entorno de programación agradable y sencillo para el usuario que con ayuda de los manuales necesarios puede realizar aplicaciones dentro su empresa.

El objetivo de esta tesis es realizar un controlador lógico basado en el MCU MC9S08GT60 habilitado como dispositivo CHIPBAS8GT, que cuente con 16 entradas binarias, 8 salidas binarias y ochenta variables binarias intermediarias. Al dispositivo se ha denominado Programador Lógico Modular 3 (PLM3). Para fines del proyecto estamos empleando una tarjeta de desarrollo denominada MINICON_08GT que está basada en el MCU mencionado en este párrafo.

El PLM3 realiza bloques funcionales llamados módulos lógicos (ML) que realizan diversas funciones de Control Lógico. Los ML son implementados mediante líneas de código ejecutable por la tarjeta MINICON_08GT. La programación del PLM3 se realiza con ayuda de un lenguaje textual de programación propio de éste, denominado Software de Interpretación de Instrucciones Lógicas 2 (SIIL2), ejecutable en una PC de escritorio o portátil. Es importante mencionar que la tarjeta de desarrollo MINICON_08GT junto con el compilador propio del lenguaje de programación (SIIL2) fueron desarrollados en el Departamento de Control y Robótica de la Facultad de Ingeniería de la UNAM.

1.1 ANTECEDENTES.

En la Facultad de Ingeniería de la UNAM en el año de 1999, se desarrolló un controlador lógico programable denominado como PLM (Programador Lógico Modular), el cual basa su principio de funcionamiento en la tarjeta de desarrollo SIMMP-2 y utiliza el microcontrolador 68HC11F1 fabricado por la compañía Motorola. Éste cuenta con una tarjeta de 32 entradas optoacopladas agrupadas en cuatro grupos y con una tarjeta de 16 salidas, cada una de ellas cuenta con una interfaz a un relevador de baja potencia de contactos normalmente abiertos.

El PLM realiza bloques funcionales denominados como módulos lógicos que son usualmente requeridos en el control lógico de procesos; tales módulos manejarían entradas y salidas binarias y se implantan mediante tramos de código ejecutable realizados por la tarjeta SIMMP-2. La programación del PLM se realiza con el auxilio de un Software de Instrucciones Lógicas (SIIL1) el cual corre en una computadora tipo PC, de tal modo que el usuario final no se las tenga que ver con detalles técnicos de la arquitectura y funcionamiento del microcontrolador correspondiente, sino solo con la manera en que debe declarar los módulos lógicos que su aplicación requiera en un momento dado.

Adicionalmente, cuenta con una unidad desplegadora (LCD) que maneja dos renglones de 16 caracteres y un panel que contiene cinco postes que habilitan entradas binarias auxiliares y cuatro botones para comando local. También cuenta con un reloj de tiempo real, el cual puede servir simplemente como testigo de la hora de un evento o como base de tiempo para una función especial del dispositivo.

El PLM requiere para su funcionamiento de dos fuentes de voltaje, una de 12 volts y otra de 5 volts, la primera polariza únicamente a los relevadores del bloque de salidas y requiere de una capacidad de corriente de un Ampere, mientras que la segunda fuente es la encargada de polarizar la tarjeta principal y requiere una capacidad de 500 mA, para este primer prototipo las fuentes de alimentación se implantaron empleando una fuente comercial para laboratorio de electrónica.

CAPÍTULO 2

CARACTERÍSTICAS FUNCIONALES DE LA TARJETA MINICON_08GT.

En este capítulo se describen de forma general las herramientas de software y hardware que se emplearon como plataforma para el desarrollo del prototipo del PLM3.

Los componentes funcionales de la herramienta usada son:

1. Tarjeta MINICON_08GT basada en el MCU MC9S08GT60.
2. Software manejador denominado PUMMA_08+, ejecutable bajo WINDOWS 98/ XP/Vista/7/8.

La tarjeta está ligada mediante un enlace serie con la computadora donde se ejecuta el software manejador. A continuación se describen las funcionalidades básicas de la herramienta aquí mencionada:

- Es un hardware basado en el MCU MC9S08GT60 mediante el cual se pueden desarrollar aplicaciones basadas en microcontroladores.

- Puede operarse mediante un software propio, que al igual que la tarjeta, fue desarrollado en el Departamento de Control y Robótica de la Facultad de Ingeniería de la UNAM, este software manejador se denomina PUMMA_08+.

De esta forma, la tarjeta de desarrollo junto con el software manejador, forman en conjunto la base principal con dos componentes funcionales e indispensables para el desarrollo de aplicaciones utilizando este Microcontrolador. A estos componentes funcionales los denominaremos Hardware de la Tarjeta (HT) y Software de la Tarjeta (ST) respectivamente.

En la figura 2.1 se muestra el sistema en conjunto de los componentes funcionales antes mencionados.

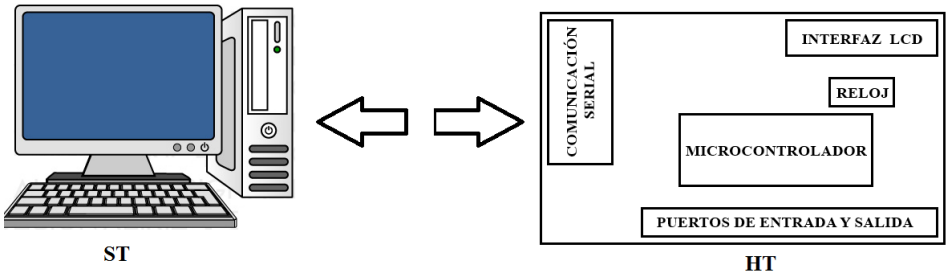


Figura 2.1 Componentes funcionales del ambiente integrado para la Tarjeta MINICON_08GT.

2.1 HARDWARE DE LA TARJETA (HT).

La tarjeta MINICON_08GT, cuenta a su vez con ciertas características funcionales con las cuales se logra obtener un mejor rendimiento de ésta.

Las características funcionales del Hardware de la Tarjeta (HT) son:

- Cuenta con un MCU principal perteneciente a la familia HCS08 de Freescale, siendo este el denominado MC9S08GT60.
- Contiene una interfaz de comunicación RS232.
- Incluye indicadores (Leds) testigo para el puerto A.
- Cuenta con postes (headers) auxiliares para diversos puntos de prueba.
- Tiene capacidad para ejecución autónoma de programas previamente escritos en la memoria no volátil del MCU.
- Contiene una interfaz de LCD para visualización de datos requeridos por el usuario.

En la figura 2.2 se muestra la imagen del Hardware de la Tarjeta.

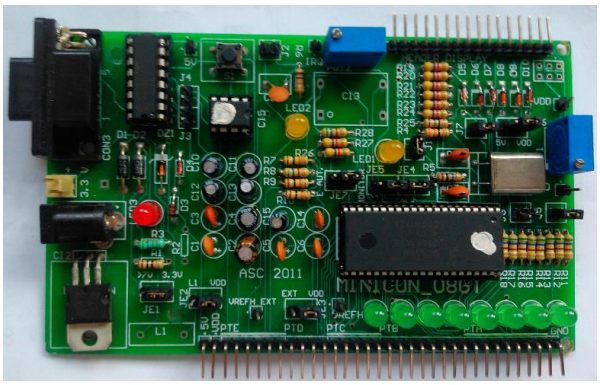


Figura 2.2 Hardware de la Tarjeta de desarrollo MINICON_08GT.

2.2 SOFTWARE DE LA TARJETA (ST).

2.2.1 Software de base de la Tarjeta MINICON_08GT.

La Tarjeta MINICON_08GT, cuenta con un software de base donde se fundamenta su funcionamiento principal, este firmware se denomina NBCP8 (Núcleo Básico de Comunicaciones con PUMMA_08+), se encuentra en una zona protegida y reservada de la memoria FLASH del Microcontrolador y permite que éste pueda ser manejado empleando el Software Manejador de la Tarjeta (PUMMA_08+).

2.2.2 Software Manejador de la Tarjeta MINICON_08GT.

Para complementar el funcionamiento de la Tarjeta MINICON_08GT, se cuenta con un software manejador propio de ésta, ejecutable en PC, siendo las características principales del denominado PUMMA_08+ las siguientes:

- Ejecutable en ambiente Windows (98/M/XP/Vista/7/8).
- Ejecución de programas previamente cargados en la tarjeta.
- Capacidad de escribir datos a la memoria del MCU.
- Capacidad de examinar datos de la memoria del MCU.
- Cuenta con una interfaz para que el usuario pueda escribir y almacenar programas previamente editados.
- Capacidad para programar la memoria del MCU destino.
- Capacidad para obtener el código máquina (archivo S19) ejecutable en el MCU del HT a partir del código fuente en ensamblador.

- Cuenta con un emulador de terminal con el cual se realiza la consola de interfaz con el usuario.

En la figura 2.3, se aprecia la ventana de inicio del Software manejador de la Tarjeta (ST) en donde se observan los datos de autor, entre otros.

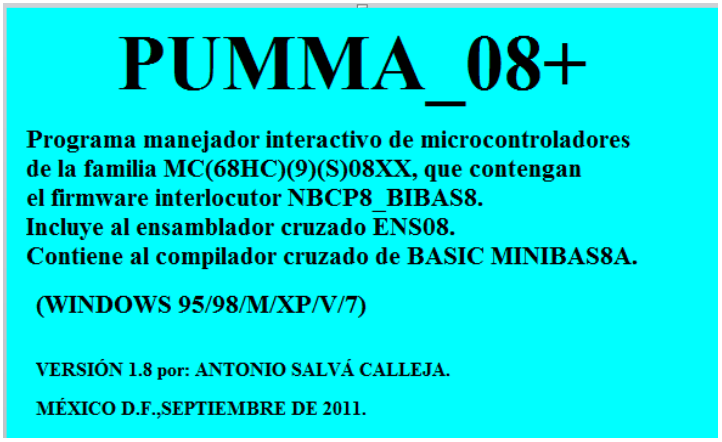


Figura 2.3 Ventana de inicio del Software de la Tarjeta MINICON_08GT.

Habiendo ejecutado el Software Manejador e ingresado los parámetros solicitados (ver manual muaida08ve2011), se desplegará la ventana principal de edición del PUMMA_08+, como se muestra en la figura 2.4, en esta ventana se muestran una serie de opciones, las cuales realizan las acciones necesarias para el funcionamiento adecuado del software, entre éstas se encuentran; nuevo, guardar, abrir, cortar, copiar, pegar.

Existen otras opciones que tienen gran importancia, ya que las funciones que realizan se encuentran vinculadas con el proceso de compilación y/o ensamble del código fuente. Además, se encuentra presente una opción mediante la cual, podemos borrar la memoria Flash de usuario.

Figura 2.4 Ventana principal del editor del PUMMA_08+.

A continuación se describen de manera general las opciones principales de la ventana de edición del software manejador PUMMA08+, siendo ésta la base principal en la cual se basa el desarrollo de los comandos y tareas a realizar por el PLM3 (Programador Lógico Modular 3), el cual se tratará más adelante.

- Botón **"e-ram"**, (ejecución inmediata en memoria RAM).

La función que realiza esta opción, es ensamblar y cargar el programa presente en el editor. El ensamble será exitoso cuando no aparezcan errores de sintaxis, inmediatamente después el programa se carga y ejecuta en el HT. Cuando se presenta algún error, el ensamblador detiene el proceso de ensamble y se manda un mensaje de error.

- Botón **"e-feep"**, (Ejecución inmediata en memoria FLASH).

La función que se realiza es ensamblar el programa presente en el editor, si no hay errores de sintaxis, posteriormente se carga en memoria FLASH y se ejecuta en el HT.

- Botón **"ens".**

Su función, es ensamblar de inmediato el programa que se encuentra presente en la ventana del editor. Si existe un error, éste se le reporta al usuario mediante un mensaje. Cuando se ha tenido éxito en el ensamble, se generan los siguientes archivos:

- nombre_de_archivo.s19
- nombre_de_archivo.lst

- Botón **“e-pfbas”**, (Compilación y ejecución inmediata).

Su función es compilar el programa presente en la ventana del editor. Para que la compilación sea exitosa, debe tener cero errores léxicos, sintácticos y semánticos. Posteriormente, se carga y ejecuta en el HT. Si existen errores, éstos se reportan al usuario. El programa fuente tiene que estar en lenguaje BASIC compatible con el compilador MINIBAS8A que se encuentra en el PUMMA_08+. Se debe de tener suficiente espacio de memoria FLASH en el HT para las direcciones de carga que correspondan al código objeto.

- Botón **“c-pfbas”**.

Su función es compilar el programa presente en la ventana del editor, para que la compilación sea exitosa, debe tener cero errores léxicos, sintácticos y semánticos. En caso de que se presenten errores, éstos se notifican al usuario mediante un mensaje. El programa fuente tiene que estar en lenguaje BASIC compatible con el compilador MINIBAS8 que se encuentra en el PUMMA_08+.

De esta manera, se concluye con la descripción básica de la herramienta utilizada para el desarrollo del PLM3. Cabe señalar que para el desarrollo e implementación del Software que se ejecutará en el MCU de la tarjeta para fines del PLM3, se usaron únicamente las facilidades de PUMMA_08+ para ensamble y prueba de programas escritos en ensamblador. No se usaron las facilidades para programar el MCU en BASIC.

En el siguiente capítulo se describe el desarrollo de la estructura y funcionamiento del Programador Lógico Modular 3 (PLM3), en donde se explicarán a detalle sus partes funcionales así como la forma de programación de éste.

CAPÍTULO 3.

ESTRUCTURA Y FUNCIONAMIENTO DEL PROGRAMADOR LÓGICO MODULAR PLM3.

En el desarrollo de este capítulo se describe de forma genérica la estructura y operación del Programador Lógico Modular versión 3 (PLM3), de igual forma se detalla la organización y nomenclatura de las variables booleanas con las cuales opera, y de esta forma, describir los módulos lógicos que realiza el dispositivo y el formato sintáctico de éstos, los cuales son declarados en un lenguaje propio del PLM3 (lenguaje SIIL2), mismo que es la interacción con el usuario final que se encuentre experimentando o bien trabajando con el PLM3.

3.1 ESTRUCTURA DEL PROGRAMADOR LÓGICO MODULAR PLM3.

Un esquema simplificado a bloques del PLM3 se puede observar en la figura 3.1, es importante señalar que el dispositivo se desarrolló para un conjunto de 16 entradas y 8 salidas booleanas las cuales operan junto con la Tarjeta MINICON_08GT (ver el capítulo 2 para mayor información de ésta), en el diagrama se aprecia que el PLM3 está integrado por los siguientes bloques funcionales:

- Computadora Central (Tarjeta MINICON_08GT).
- Bloque de entradas.
- Bloque de Salidas.
- Fuente de Alimentación.

Se describen a continuación los bloques funcionales del que está integrado el PLM3.

3.1.1 Computadora Central (CC).

En el capítulo 2 se menciona la descripción y generalidades de la Tarjeta MINICON_08GT, cuyo CPU es el microcontrolador MC9S08GT60 de Freescale. Al final de esta tesis, se anexará la información bibliográfica de ésta, en donde se describen a detalle sus características principales y generalidades.

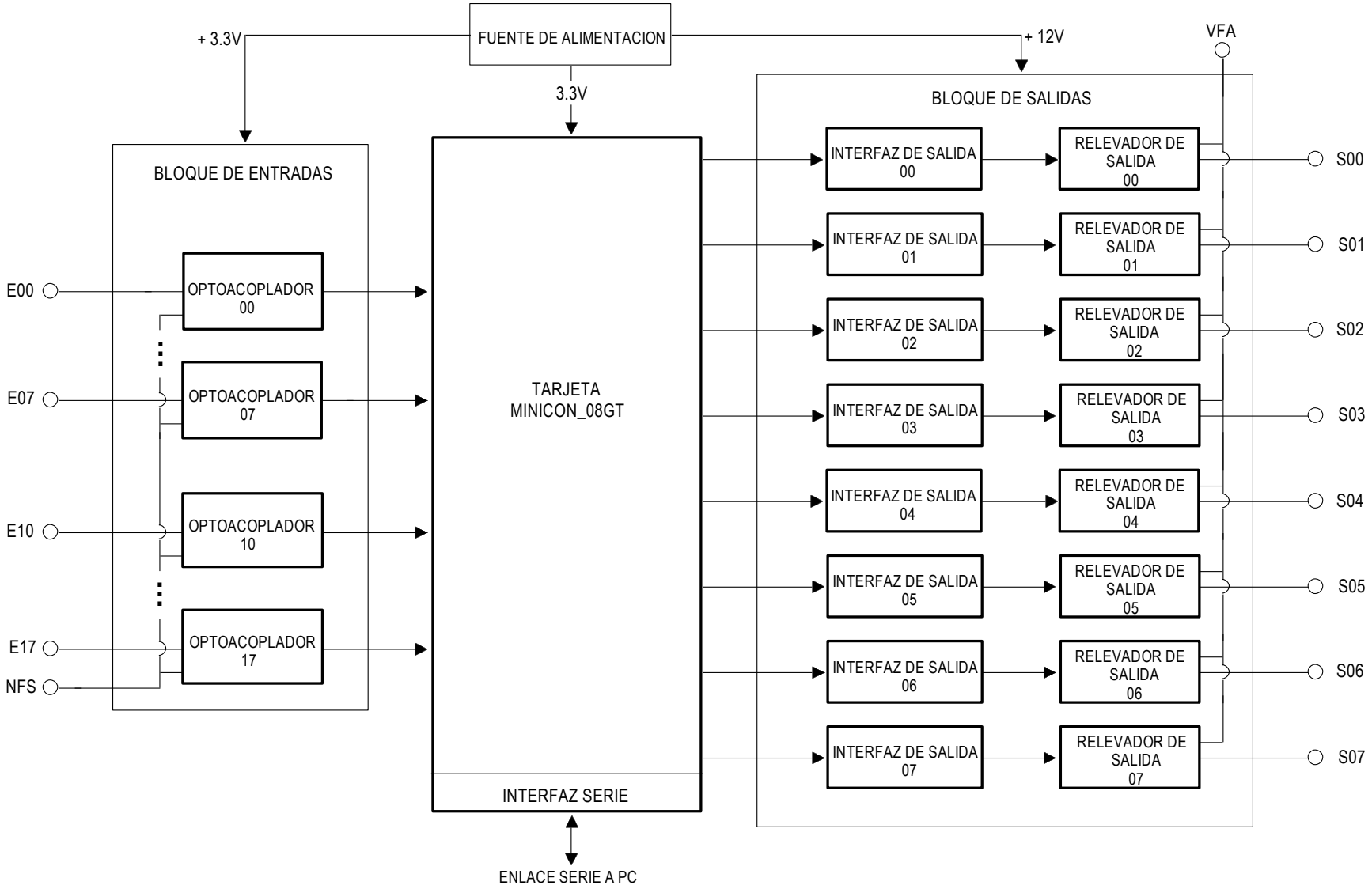


Fig. 3.1 Diagrama simplificado a bloques del PLM3.

3.1.2 Bloque de entradas (BE).

El bloque asociado a esta parte del PLM3 se conforma por 16 entradas optoacopladas y pueden operar de dos distintas formas lógicas para que sean reconocidas por la CC. La primera denominada contacto seco, cuando para cada entrada se presente continuidad entre ésta y la terminal denominada Neutro de la Fuente de Sensores (NFS). La otra forma valida el uno lógico y NFS, es decir; para cada uno de estos voltajes corresponden sendos niveles lógicos en el BE (cero y uno lógico respectivamente). Las dos formas de operar de las entradas son seleccionables por el usuario mediante el uso de un jumper (puente) permitiendo desarrollar su aplicación de manera flexible. La estructura y diseño del BE se tratará en el capítulo 4 del presente trabajo.

El bloque de entradas se compone por dos grupos, cada grupo está constituido por 8 entradas, esto debido a que la información en el microcontrolador de la CC se encuentra organizada en bytes.

Para declarar las entradas, se emplean tres caracteres, siendo el primero de ellos la letra “E” como mayúscula o minúscula, el segundo corresponde al número de grupo del bloque de entradas (cero o uno) y el tercer carácter hace referencia al número de bit de entrada que se esté utilizando (en un rango de cero a siete). A continuación se muestra un ejemplo de la declaración, utilizando el bit 4 del grupo 1 del BE en el PLM3, siendo ésta “E14”.

3.1.3 Bloque de salidas (BS).

Lo correspondiente a este bloque se conforma por 8 terminales las cuales están asociadas a un puerto de salida de la CC, cada una de éstas cuenta con una interfaz de baja potencia utilizando relevadores de contactos normalmente abiertos.

Las terminales del BS pueden utilizarse de forma independiente o bien pueden operar de una forma en común, esto depende de la decisión del usuario ya que si se desea operar de forma independiente el PLM3 se pueden utilizar 8 actuadores con diferente fuente de alimentación, en caso contrario se contaría con los mismos 8 actuadores pero con una fuente de alimentación en común. De igual forma, la estructura y diseño de este bloque se detallará en el capítulo 4 de este texto.

La forma de declaración es similar a la del bloque de entradas, se utilizan tres caracteres siendo el primero la letra “S” mayúscula o minúscula, el segundo corresponde al número de grupo el cual es cero (se reitera que sólo hay un grupo) y finalmente el tercer carácter representa el número de bit de salida, el cual se encuentra en un rango de cero a siete. Utilizaremos como ejemplo la salida tres, siendo “S03” la declaración correspondiente.

3.1.4 Fuente de Alimentación (FA).

Para que el PLM3 tenga un funcionamiento correcto junto con los bloques antes mencionados requiere de dos fuentes de alimentación, la primera de ellas es de 3.3 volts siendo ésta la encargada de alimentar a la CC (podemos encontrar las generalidades de la tarjeta en el capítulo 2 de esta tesis) y a su vez demanda una corriente máxima de 500 mA. La segunda fuente es de 5 volts y la corriente máxima que demanda es de un ampere, su función consiste en alimentar las bobinas internas de los relevadores. En el capítulo 5, se muestra el diseño y estructura de ambas fuentes.

3.2 VARIABLES BOOLEANAS DEL PLM3.

Las variables booleanas (VB) manejadas por el PLM3 están asociadas con los niveles lógicos de las entradas y salidas del dispositivo y con los propios asociados con variables denominadas intermediarias, que sirven de enlace entre módulos. De esta forma, las VB se clasifican como: Variables Booleanas de Entrada (VBE), Variables Booleanas de Salida (VBS) y Variables Booleanas Intermediarias. (VBI).

Las variables booleanas de entrada están organizadas en dos grupos de ocho elementos; las variables booleanas de salida están organizadas en un grupo de ocho elementos. A continuación se describe lo propio acerca de los tres tipos de variables booleanas que puede manejar el PLM3.

3.2.1 Variables Booleanas de Entrada (VBE).

Estas variables se encuentran optoacopladas para que la CC pueda recibir la información con un nivel lógico de voltaje correcto, y cada una ellas se encuentra asociada con sendas terminales del BE, al cual están conectados los sensores que se encuentran en el sistema de control de la aplicación a desarrollar, siendo éstos los encargados de enviar las señales de voltaje (0 ó 24 volts).

En la sección 3.1.2 se muestra la forma de declaración para este tipo de variables en el lenguaje propio del PLM3.

3.2.2 Variables Booleanas de Salida (VBS).

Las VBS son las encargadas de operar en conjunto con los actuadores del sistema de control. A cada una de ellas le corresponde una etapa de baja potencia utilizando relevadores de contactos normalmente abiertos (CNA) a los cuales se encuentran conectados los actuadores, los CNA cierran cuando la señal enviada por la CC es un uno lógico y abren cuando la señal es un cero lógico.

La forma de declaración para este tipo de variables booleanas se puede observar en la sección 3.1.3 del presente texto.

3.2.3 Variables booleanas Intermediarias (VBI).

Estas variables se utilizan y realizan internamente, es decir; sirven de enlace entre los módulos lógicos cuando esto se necesite. Dicho de otra forma, los valores que toman pueden ser entradas o salidas para otros módulos lógicos y de ahí la necesidad de contar con las VBI, ya que si se realiza por medio de VBE o bien por VBS, la aplicación puede quedar limitada en cuanto a sensores y actuadores. El PLM3 cuenta con 20 grupos de 8 variables cada uno y de esta forma se cuentan con 160 VBI y la declaración es similar a la de VBE o VBS, en este caso se utiliza la letra “I” y se respeta el orden de número de grupo y número de bit.

En la figura 3.2, se muestra como ejemplo un sistema lógico integrado por varios módulos lógicos en donde se aprecian las diversas variables booleanas implicadas.

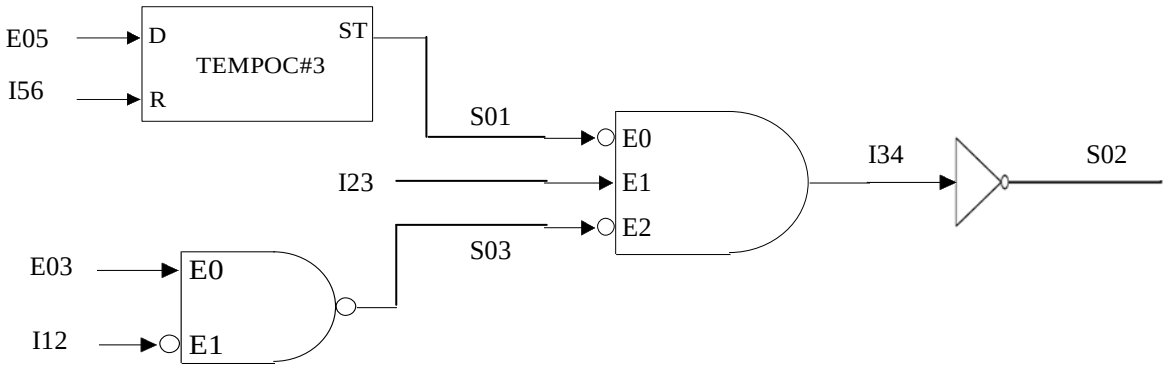


Figura 3.2 Sistema lógico realizable con el PLM3 donde se aprecia el uso de diversos tipos de variables booleanas.

3.3 CARACTERÍSTICAS GENERALES DE LOS MÓDULOS LÓGICOS.

A partir de los subtemas tratados anteriormente, se introducirá la descripción general de los Módulos Lógicos (ML) y la relación que tienen con las variables booleanas, los cuales, constituyen los bloques funcionales elementales para realización de aplicaciones de control lógico y pueden ser representados a nivel de "caja negra" como se muestra en la figura 3.3, donde se muestra un ML que presenta "m" entradas y "n" salidas; p y q varían de acuerdo con el tipo de función que un determinado ML realice, así por ejemplo; para una compuerta AND de tres entradas, los valores de p y n serian tres y uno respectivamente.

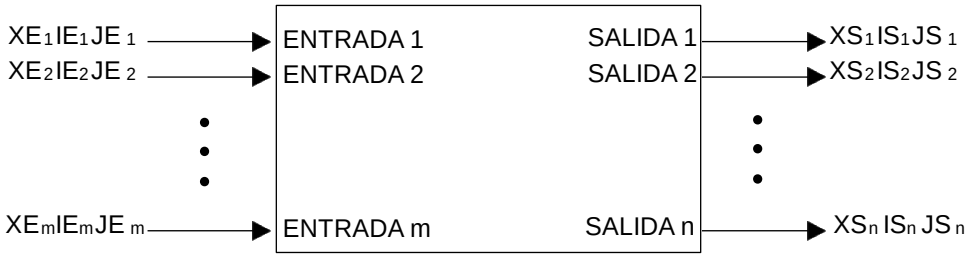


Figura 3.3 Representación de un módulo lógico de *m* entradas y *n* salidas.

En la figura 3.3, XE_k representaría a un carácter que podría ser cualquiera de las letras e, i o s mayúsculas o minúsculas dependiendo del tipo de variable (VBE, VBI o VBS) asociada con la entrada k-ésima del ML; IE_k y JE_k serían respectivamente los números asociados con el grupo y el número de bit correspondientes con la variable k-ésima de entrada; XS_k representaría a un carácter que podría ser cualquiera de las letras i o s mayúsculas o minúsculas dependiendo del tipo de

variable (VBI o VBS) asociada con la salida k-ésima del ML; IS_k y JS_k serían respectivamente los números asociados con el grupo y el número de bits correspondientes con la variable k-ésima de salida. Por ejemplo, una compuerta NAND de tres entradas con negación en una de ellas se muestra en la figura 3.4, las entradas son respectivamente las variables E01, I45 (entrada negada) y E13, la salida es la variable S02.

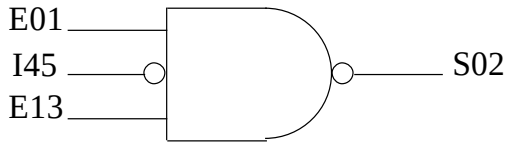


Figura 3.4 Representación de un Módulo Lógico que realiza una compuerta NAND de tres entradas con negación en una de ellas.

Para todos los ML que validan compuertas lógicas se tiene que los mismos responden al nivel que presenten sus entradas; sin embargo algunos de los ML que no son compuertas están diseñados de modo que responden a flancos que se presentan en una o varias de sus entradas.

En la figura 3.5 se muestra un ML que realiza un temporizador de tipo monodisparo (one-shot) que presentará en su salida (variable S14) un pulso verificado alto de una duración determinada, cada vez que en la entrada de disparo (variable E12) se manifieste un flanco de bajada, la otra entrada (variable E02) el ML responde al nivel, cuando el mismo es alto el temporizador está habilitado, en caso de que el nivel sea bajo se retorna la salida a su nivel no verificado no respondiendo el módulo a los disparos hasta que la entrada mencionada retorne a el nivel alto.

A nivel de los esquemas de bloques asociados con los ML la sensibilidad a flancos de una entrada es denotada mediante una flecha vertical cuyo sentido indica el tipo de flanco asociado, véase la figura 3.5.

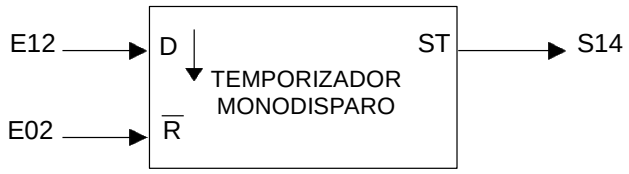


Figura 3.5. Representación en forma de bloque de un temporizador monodisparo (one-shot), la flecha hacia abajo indica que el disparo es por flanco de bajada.

En la figura 3.6, se observan los tres gráficos correspondientes al ML asociado a la figura 3.5. Cuando la VBE mande la señal de flanco de bajada, se activará el disparo mostrándose en la VBS correspondiente; todo lo anterior, mientras se cumpla la condición en la VBE E02.

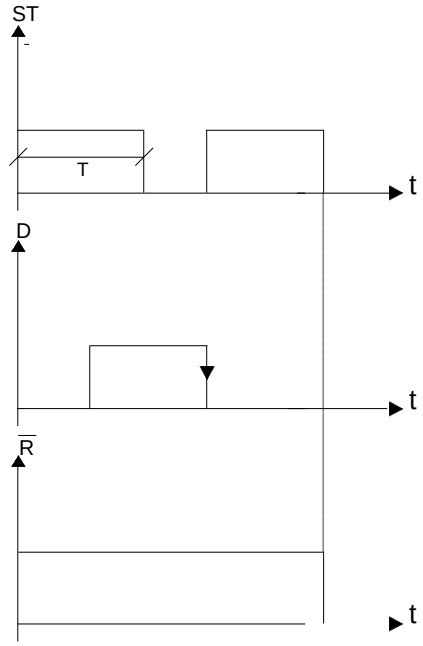


Figura 3.6. Diagrama de tiempos correspondiente a un temporizador monodisparo (one shot).

3.3.1 Formas sintácticas asociadas con los Módulos Lógicos.

Cada uno de los módulos lógicos requeridos en una determinada aplicación deben ser declarados secuencialmente en un archivo de texto para que el mismo sea procesado en una computadora anfitriona donde corre un software manejador del PLM3 (SWMANPLM3) que genera el código objeto que ha de ejecutar la CC del PLM3, finalmente, los ML requeridos han quedado realizados; además de las declaraciones asociadas con los módulos, en el archivo mencionado se requieren colocar otras instrucciones no relacionadas directamente con algún ML,

pero necesarias para la ejecución adecuada del programa que ha de ejecutarse en el PLM3, de esto se hablará más adelante. La funcionalidad del SWMANPLM3 se describe en el capítulo seis de esta tesis.

Al conjunto de instrucciones escritas en secuencia en un archivo de texto mencionadas anteriormente, se le denomina programa fuente en lenguaje SIIL2 asociado con la aplicación que ha de realizar el PLM3. A excepción de los módulos que requieren datos adicionales que ha de proporcionar el usuario, la forma sintáctica de las declaraciones asociadas con los mismos requiere de un solo renglón en el archivo de texto a procesar, esta forma se ilustra a continuación:

CODM#N ENT_1, ENT_2..., ENTm, SAL_1, SAL_2, ..., SALn, DA_1,... DAq,…, CADBI

Donde:

CODM Es una cadena de caracteres que simboliza la función efectuada por el módulo.

N Es el número asociado con el módulo, ya que todos los ML de un mismo tipo que use una aplicación, deben ser numerados.

ENT_1 a ENTm Son las designaciones asociadas con las m variables de entrada que el módulo requiera.

SAL_1 a SALn Son las designaciones asociadas con las n salidas que pudiera tener el módulo.

DA_1 a DAq Son datos auxiliares que pudieran ser requeridos por algunos módulos, estos podrían ser entre otros: el tiempo asociado con la duración de un pulso generado por un temporizador o bien el número de estados que ha de presentar un secuenciador, etc. Hay módulos que no requieren de estas especificaciones, tal es el caso de las compuertas lógicas.

Para los módulos que sí requieren de estos datos, "q" es un número que está comprendido entre cero y tres.

CADBI es una cadena formada por unos y ceros que especifica diversas características de funcionamiento como podrían ser; qué entradas a una compuerta van a tener negación implícita, a qué tipo de flanco responde una entrada de algún otro tipo de módulo, etc.

Como ejemplo de estructura sintáctica, a continuación se muestra la instrucción asociada con la declaración de la compuerta NAND de tres entradas mostrada en la figura 3.4.

NAND3#1 E01, I45, E13, S02, 101

En la instrucción anterior CODM es la palabra NAND3 que denota el hecho de que se trata de una compuerta NAND de tres entradas; por ejemplo, si se hubiera tratado de una compuerta NAND de cuatro entradas CODM hubiera sido NAND4.

Nótese además que la cadena binaria asociada (CADBI) consta de tres bits, ya que la compuerta es de tres entradas, se indica que la entrada I45 deberá tener una negación colocando un cero en la posición que corresponde a tal entrada, de este modo, si se requiriera que tuvieran negación las dos primeras entradas (E01 e I45) CADB sería 001.

3.4 FORMATO DE UN PROGRAMA EN SIIL2

Cuando se corre un programa en SIIL2 en la CC del PLM3, el código asociado con cada ML se ejecuta de forma cíclica siguiendo la siguiente secuencia.

1. Se copian en un buffer de entrada en RAM el estado que guardan los puertos asociados con las 16 entradas físicas VBE.

2. Se ejecuta uno a uno el código asociado con cada uno de los ML que el usuario haya declarado en el programa fuente correspondiente, las salidas que se van generando son colocadas en un buffer de salida en RAM.

3. Se copia el estado del BS en los puertos físicos asociados con las Variables Booleanas de Salida VBS.

4. Se regresa al paso uno.

Existen módulos que requieren que el periodo de repetición de la ejecución de su código asociado sea constante (10 ms), tal es el caso por ejemplo de los temporizadores, para hacer esto posible el código asociado es colocado en una rutina de servicio de interrupción que es invocada con una periodicidad de 10 ms, empleándose para ello las facilidades de temporización con que cuenta la CC del PLM3.

En consecuencia, el código asociado con un programa en SIIL2 está dividido en dos partes, una de ellas es la que se ejecuta de acuerdo con los cuatro pasos descritos anteriormente, a esta parte se le llama subprograma principal. La otra parte está constituida por el código cuya ejecución es temporizada y se denomina subprograma temporizado. En la figura 3.7 se ilustra esta idea.

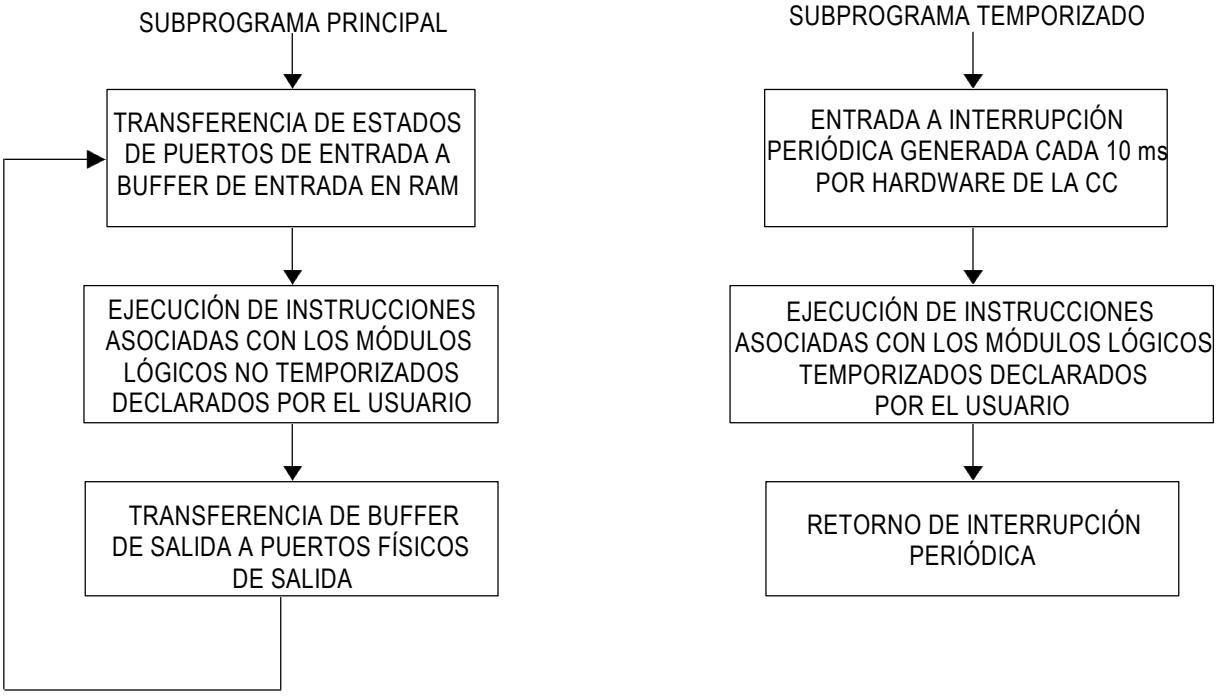


Figura 3.7. Ejecución en la CC del PLM3, de los subprogramas que integran un programa SIIL2.

Cabe señalar que para la versión actual (2015) del compilador de SIIL2, todos los módulos lógicos se declaran en el subprograma principal, el código requerido por los módulos temporizados es generado y colocado automáticamente en el subprograma temporizado; esto sin la intervención del usuario final del PLM3.

3.4.1 Forma de un programa fuente en SIIL2

Para poder asociar un programa fuente en SIIL2 con una aplicación, deben realizarse una serie de declaraciones, éstas pueden ser comandos o instrucciones; los comandos son indicaciones tales como inicio o fin de un subprograma principal, tipo de mapa de memoria empleado en la CC, inicio o fin de instrucciones asociadas con el subprograma temporizado. Las instrucciones son declaraciones asociadas con características que han de tener los módulos empleados por la aplicación y deben respetar la sintaxis descrita anteriormente. Como un panorama general y de forma más clara, un programa fuente en SIIL2 está integrado por la siguiente secuencia de declaraciones:

1. Comando que marca el inicio del subprograma principal, la palabra correspondiente a este comando es INPROG.

2. Instrucciones asociadas con los módulos que integran el sistema lógico a realizar con el PLM3.

3. Comando que marca el fin del subprograma principal, la palabra para este caso es FINPP.

4. Comando que marca el inicio del subprograma temporizado, la palabra asociada es INMODI.

5. Comando que indica el fin del subprograma temporizado, la sintaxis para este comando es FINMODI.

Los cinco componentes del programa fuente han de ser colocados respetando el orden anterior; al igual que en el caso de las instrucciones asociadas con los módulos.

Ejemplo 3.1

Con el fin de aclarar algunos conceptos antes mencionados, a continuación se muestra la forma de cómo se programaría el PLM3 para realizar una situación de control lógico.

En la figura 3.8 se muestra un esquema a bloques que emplea módulos propios del PLM3, para realizar un accionamiento de control lógico siguiendo la descripción antes mencionada, se

tienen a la entrada sensores que van hacia el PLM3 y a la salida de este se conectan actuadores que realizan dicha función de control.

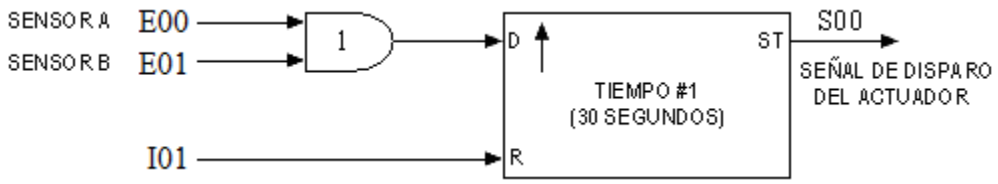


Figura 3.8 Esquema a bloques de una posible aplicación de control lógico, empleando el PLM3.

El temporizador requerido para este ejemplo es de tipo mono disparo (one shot) como el ilustrado en las figuras 3.5 y 3.6, para tener más claro este ejemplo, la sintaxis genérica asociada con un ML que realice un temporizador como el requerido se muestra a continuación:

TEMPOC #N DISPARO, HABILITACIÓN, SALIDA, DURACIÓN, ABC

Tabla 3.1 Declaraciones asociadas con un Módulo Lógico Temporizador.

TEMPOC	Cadena de caracteres que simboliza la función efectuada por un módulo lógico Temporizador Mono disparo (one shot).
N	Representa el número del temporizador.
DISPARO	Indica la variable booleana que dispara al temporizador.
HABILITACIÓN	Se refiere a la variable booleana asociada con la habilitación y restablecimiento del temporizador.
SALIDA	Es una variable booleana asociada con la salida del temporizador
DURACIÓN	Indica el tiempo que ha de especificarse de acuerdo con el formato 00:00:00:00 en horas, minutos, segundos y centésimas respectivamente.
A	Es un carácter que podrá ser cero si se desea que el disparo sea por flanco de bajada o uno si se desea que el disparo sea por flanco de subida.
B	Es un carácter que podrá ser cero si se desea que la habilitación sea por nivel alto y el restablecimiento sea por nivel bajo; en caso de que se requiera que la habilitación sea por nivel bajo y el restablecimiento por nivel alto, B deberá ser cero. Para que el temporizador responda a los disparos la habilitación del mismo deberá estar verificada, en caso de verificarse el restablecimiento mientras se verifica el pulso de salida, el mismo regresará a su nivel no verificado, véase la figura 3.4.
C	Es un carácter que podrá ser cero si se desea que el pulso de salida tenga verificación en bajo y uno en caso de que se desee que dicha verificación sea en alto.

Como se observa en la figura 3.6 se están empleando dos variables booleanas intermediarias, aclarándose aquí que el valor por defecto de estas variables es cero, siendo esta la causa de que el nivel requerido para la señal de habilitación sea bajo, apreciándose el empleo, como delimitadora, de la variable intermediaria I01.

El programa correspondiente en SIIL2 es:

```
‘Programa asociado con el ejemplo 3.1
    INPPROG   ‘Declaración de inicio de subprograma principal
‘Las siguientes líneas corresponden a instrucciones asociadas con los módulos que realizan
‘una compuerta lógica AND de dos entradas y un temporizador.
    AND2#1 E00, E01, I00, 111 ‘Compuerta and número 1
    TEMPOC#1 I00, I01, S00, 00:00:30:00, 101 ‘Temporizador
    FINPP   ‘Declaración de fin de subprograma principal
    INMODI   ‘Declaración de inicio de subprograma temporizado
    FINMODI   ‘Declaración de fin de subprograma temporizado
```

En los siguientes capítulos se describirá con más destalle el funcionamiento de los módulos aquí descritos, así como, la declaración de cada uno de ellos.

CAPÍTULO 4

DESCRIPCIÓN DE LAS FUNCIONES DE CONTROL LÓGICO REALIZABLES POR EL PLM3.

En el capítulo 4 se introduce a la descripción de los elementos de control lógico, siendo éstos los bloques funcionales que opera el PLM3. Los bloques generalmente utilizados en aplicaciones de control lógico pueden ser: compuertas lógicas, temporizadores, contadores de eventos y secuenciadores de estados. En el PLM3 denotaremos a estos bloques funcionales con el nombre de Módulos Lógicos (ML) los cuales son procesados y ejecutados con el software propio de éste (lenguaje SIIL2).

Se mencionan a continuación los módulos lógicos realizables por el PLM3:

- Seguidor lógico.
- Inversor lógico.
- Compuertas AND de dos a cuatro entradas.
- Compuertas OR de dos a cuatro entradas.
- Compuertas NAND de dos a cuatro entradas.
- Compuertas NOR de dos a cuatro entradas.
- Compuertas XOR de dos a cuatro entradas.
- Compuertas XOR negada de dos a cuatro entradas.
- Flip-Flops asíncronos.
- Temporizadores
- Contadores.
- Secuenciadores.
- Comparadores.

La descripción del funcionamiento y declaración sintáctica de los Módulos Lógicos previamente mencionados se puede observar en la secuencia de este capítulo.

4.1 DESCRIPCIÓN DEL MÓDULO LÓGICO SEGUIDOR.

El funcionamiento de este ML consiste en poner el nivel lógico que exista en la variable booleana declarada como entrada (VBE) en la variable booleana declarada como salida (VBS) al mismo, en la figura 4.1 se observa el diagrama genérico de este ML, la sintaxis de declaración de éste en el programa principal es la siguiente:

SEG#N XEIEJE,XSISJS;

En la tabla 4.1 se hace una breve descripción sobre la sintaxis de declaración asociada con el ML Seguidor.

Tabla 4.1 Descripción de la sintaxis de declaración asociada con el ML Seguidor.

SEG	Es una cadena de caracteres que simboliza la función efectuada por un ML Seguidor.
N	Indica el número que corresponde al seguidor, esto definido por el usuario.
XE	Puede ser la letra "e", "s" o "i" mayúscula o minúscula dependiendo esto de que la variable de entrada al seguidor sea una VBE, VBS o VBI respectivamente.
IE	Indica el número de grupo correspondiente a la VB declarada como entrada al seguidor.
JE	Indica el número de bit dentro del grupo IE, asociado a la variable de entrada al seguidor.
XS	Puede ser la letra "s" o "i" mayúscula o minúscula dependiendo esto de que la variable de salida al seguidor sea una VBS o VBI respectivamente.
IS	Indica el número de grupo correspondiente a la VB declarada como salida del seguidor.
JS	Indica el número de bit dentro del grupo IS, asociado a la variable de salida del seguidor.

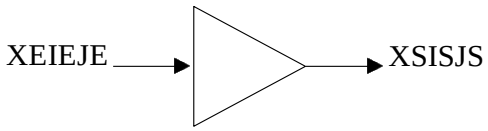


Figura 4.1 Representación genérica del ML seguidor.

A continuación se muestra un ejemplo de la declaración de un ML Seguidor en el programa principal SIIL2.

Ejemplo 4.1

El usuario del PLM3 desea realizar un seguidor lógico al cual se le asignará el número 6, se requiere que la entrada y salida de éste sean las variables booleanas E13 e I32 respectivamente, véase la figura 4.2. La sintaxis con la cual se le declara el ML en el programa principal es la siguiente:

SEG#6 E13, I32

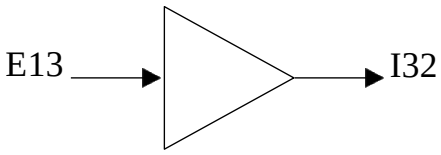


Figura 4.2 Bloque de un ML Seguidor cuya entrada es la E13 y salida I32.

4.2 DESCRIPCIÓN DEL MÓDULO LÓGICO INVERSOR.

El funcionamiento de este ML consiste en poner en la variable booleana declarada como salida (VBS) el nivel lógico opuesto al que se encuentre en la variable booleana que se declara como entrada (VBE) al mismo, en la figura 4.3 se ilustra en forma genérica este ML, la sintaxis de declaración de éste en el programa principal es la siguiente:

NOT#N XEIEJE,XSISJS

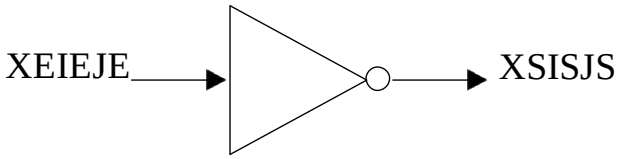


Figura 4.3 Representación genérica del ML Inversor.

En la tabla 4.2 se hace una breve descripción sobre la sintaxis de declaración asociada con el ML Inversor.

Tabla 4.2 Descripción de la sintaxis de declaración asociada con el ML Inversor.

NOT	Es una cadena de caracteres que simboliza la función efectuada por un ML Inversor.
N	Indica el número que corresponde al inversor, esto definido por el usuario.
XE	Puede ser la letra "e", "s" o "i" mayúscula o minúscula dependiendo esto de que la variable de entrada al inversor sea una VBE, VBS o VBI respectivamente.
IE	Indica el número de grupo correspondiente a la VB declarada como entrada al inversor.
JE	Indica el número de bit dentro del grupo IE, asociado a la variable de entrada al inversor
XS	Puede ser la letra "s" o "i" mayúscula o minúscula dependiendo esto de que la variable de salida al inversor sea una VBS o VBI respectivamente.
IS	Indica el número de grupo correspondiente a la VB declarada como salida del inversor.
JS	Indica el número de bit dentro del grupo IS, asociado a la variable de salida del inversor.

A continuación se muestra un ejemplo de la declaración de un ML de inversión, en el programa principal en SIIL2.

Ejemplo 4.2

El usuario del PLM3 desea realizar un inversor lógico al cual se le asignará el número 13, para esto, se requiere que la entrada y salida de éste sean las variables booleanas E06 y S02 respectivamente, véase la figura 4.4. La sintaxis con la cual se declara el ML en el programa principal siguiente:

```
NOT#13 E06, S02
```

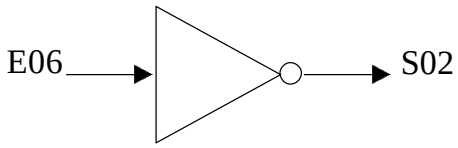


Figura 4.4 Ejemplo del ML Inversor con entrada E06 y salida S02.

4.3 DESCRIPCIÓN DE LOS MÓDULOS LÓGICOS QUE REALIZAN COMPUERTAS DE DOS ENTRADAS.

El PLM3 puede realizar seis tipos de compuertas lógicas de dos entradas, éstas son de tipo: AND, OR, NAND, OR NEGADA, OR EXCLUSIVA y OR EXCLUSIVA NEGADA, además, estos ML pueden diseñarse con preinversión en sus entradas. En la figura 4.5 se ilustra este ML, la sintaxis de declaración en el programa principal es la siguiente:

COMP#N X0I0J0,X1I1J1,XSISJS, AB

En la tabla 4.3 se hace una breve descripción sobre la sintaxis de declaración asociada con Módulos Lógicos que corresponden a Compuertas de dos entradas.

Tabla 4.3 Descripción de la sintaxis de declaración asociada con el ML de Compuertas de dos entradas.

COMP	Es una cadena de caracteres que simboliza la función efectuada por un ML correspondiente a compuertas de dos entradas que se deseen realizar, éstas pueden ser: AND2, OR2, NAND2, ORN2, EOR2, EORN2.
N	Indica el número que corresponde a la compuerta, esto definido por el usuario, la numeración es independiente para cada tipo de compuerta.
X0	Puede ser la letra "e", "s" o "i" mayúscula o minúscula dependiendo esto de que la variable de entrada E0 a la compuerta sea una VBE, VBS o VBI respectivamente.
I0	Indica el número de grupo correspondiente a la VB declarada como entrada E0 a la compuerta.
J0	Indica el número de bit dentro del grupo I0, asociado a la variable de entrada E0.
X1	Puede ser la letra "e", "s" o "i" mayúscula o minúscula dependiendo esto de que la variable de entrada E1 a la compuerta sea una VBE, VBS o VBI respectivamente.
I1	Indica el número de grupo correspondiente a la VB declarada como entrada E1 a la compuerta.
J1	Indica el número de bit dentro del grupo I1, asociado a la variable E1.
XS	Puede ser la letra "s" o "i" mayúscula o minúscula dependiendo esto de que la variable de salida "S" de la compuerta sea una VBS o VBI respectivamente.
IS	Indica el número de grupo correspondiente a la VB declarada como salida S de la compuerta.
JS	Indica el número de bit dentro del grupo IS, asociado a la variable S.
A	Corresponde a un dígito binario, que deberá ser cero, si se desea que la entrada "E1" tenga preinversión, en otro caso el digito deberá ser uno.
B	Corresponde a un dígito binario, que deberá ser cero, si se desea que la entrada "E0" tenga preinversión, en otro caso el dígito deberá ser uno.

Figura 4.5 Representación genérica de un ML de dos entradas realizable por el PLM3.

A continuación se muestra dos ejemplos de la declaración de un ML que realiza una compuerta de dos entradas.

Ejemplo 4.3

El usuario del PLM3 desea realizar una compuerta NAND de dos entradas, en donde las entradas E0, E1 y la salida S sean respectivamente las VB E03, I12 y S01, se requiere que la entrada E1 tenga preinversión y que el número de asignación a la compuerta sea 2, véase la figura 4.6. La sintaxis con la cual se declara el ML en el programa principal es la siguiente:

NAND2#2 E03, I12, S01, 01

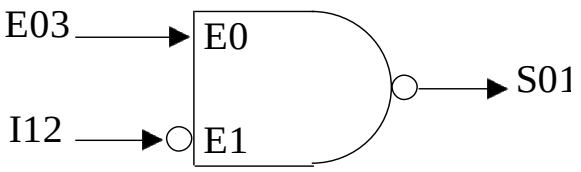


Figura 4.6 Ejemplo de compuerta NAND de dos entradas con preinversión en la entrada E1.

Ejemplo 4.4

El usuario del PLM3 desea realizar una compuerta NOR de dos entradas, en donde las entradas E0, E1 y la salida S sean respectivamente las VB E17, I26 e I03, respectivamente. El número de asignación a la compuerta será el 7, véase la figura 4.7. La sintaxis con la cual se declara el ML en el programa principal es la siguiente:

ORN2#7 E17, 126, I03, 11

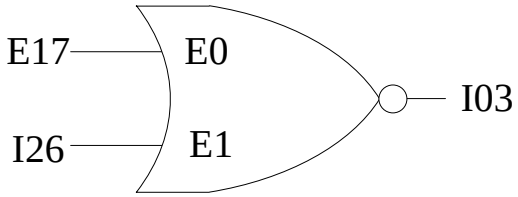


Figura 4.7 Ejemplo de compuerta NOR de dos entradas sin preinversión.

4.4 DESCRIPCIÓN DE LOS MÓDULOS LÓGICOS QUE REALIZAN COMPUERTAS DE TRES ENTRADAS.

El PLM3 puede realizar seis tipos de compuertas lógicas de tres entradas, estas son de tipo: AND, OR, NAND, OR NEGADA, OR EXCLUSIVA y OR EXCLUSIVA NEGADA, cabe mencionar que además se cuenta con la capacidad de preinversión en las entradas de éstas. En la figura 4.8 se ilustra este ML, la sintaxis de declaración de éste en el programa principal es la siguiente:

COMP#N X0I0J0,X1I1J1,X2I2J2,XSISJS,ABC

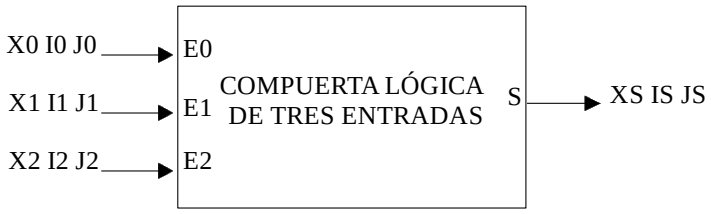


Figura 4.8 Representación genérica del ML de tres entradas realizable por el PLM3.

En la tabla 4.4 se hace una breve descripción sobre la sintaxis de declaración asociada con Módulos Lógicos que corresponden a compuertas de tres entradas.

Tabla 4.4 Descripción de la sintaxis de declaración asociada con el ML de Compuertas de tres entradas.

COMP	Es una cadena de caracteres que simboliza la función efectuada por un ML correspondiente a compuertas de tres entradas que se deseen realizar, éstas pueden ser: AND3, OR3, NAND3, ORN3, EOR3, EORN3.
N	Indica el número de compuerta, esto definido por el usuario, la numeración es independiente para cada tipo de compuerta.
X0	Hace referencia a una de las letras "e", "s" o "i", mayúscula o minúscula, esto va a depender de que la variable de entrada E0 a la compuerta sea una VBE, VBI o VBS respectivamente.
I0	Indica el número de grupo correspondiente a la VB declarada como entrada E0 a la compuerta.
J0	Indica el número de bit dentro del grupo I0, asociado a la variable de entrada E0.
X1	Hace referencia a una de las letras "e", "s" o "i", mayúscula o minúscula, esto va a depender de que la variable de entrada E1 a la compuerta sea una VBE, VBI o VBS respectivamente.
I1	Indica el número de grupo correspondiente a la VB declarada como entrada E1 a la compuerta.
J1	Indica el número de bit dentro del grupo I1, asociado a la variable de entrada E1.
X2	Hace referencia a una de las letras "e", "s" o "i", mayúscula o minúscula, esto va a depender de que la variable de entrada E2 a la compuerta sea una VBE, VBI o VBS respectivamente.
I2	Indica el número de grupo correspondiente a la VB declarada como entrada E2 a la compuerta.
J2	Indica el número de bit dentro del grupo I2, asociado a la variable de entrada E2.
XS	Hace referencia a una de las letras "e", "s" o "i", mayúscula o minúscula, esto va a depender de que la variable de salida S de la compuerta sea una VBI o VBS respectivamente.
IS	Indica el número de grupo correspondiente a la VB declarada como salida S de la compuerta.
JS	Indica el número de bit dentro del grupo IS, asociado a la variable de salida de la compuerta.
A	Corresponde a un dígito binario, que deberá ser cero si se desea que la entrada E2 tenga preinversión, en otro caso el dígito deberá ser uno.
B	Corresponde a un dígito binario, que deberá ser cero si se desea que la entrada E1 tenga preinversión, en otro caso el dígito deberá ser uno.
C	Corresponde a un dígito binario, que deberá ser cero si se desea que la entrada E0 tenga preinversión, en otro caso el dígito deberá ser uno.

A continuación se muestran dos ejemplos de Módulos Lógicos en SIIL2 de compuertas de tres entradas.

Ejemplo 4.5

Se pretende realizar con el PLM3 una compuerta AND de tres entradas, en donde se requiere que las entradas E0, E1, E2 y la salida S sean respectivamente las VB E10, I04, E17 y S05, sólo las entradas E0 y E2 tienen preinversión. El número de asignación será el 9, véase la figura 4.9. La sintaxis con la cual se declara el ML en el programa principal es la siguiente:

AND3#9 E10, I04, E17, S05, 010

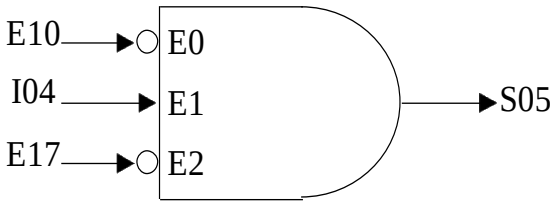


Figura 4.9 Compuerta AND de tres entradas con preinversión en las entradas E0 y E2.

Ejemplo4.6

Se pretende realizar con el PLM3 una compuerta EOR de tres entradas, se requiere que las entradas E0, E1, E2 y la salida S sean respectivamente las VB I01, I14, E02 y S0, ninguna de las entradas tienen preinversión. El número de asignación será el 2, véase la figura 4.10. La sintaxis con la cual se declara el ML en el programa principal es la siguiente:

EOR3#2 I01, I14, E02, S01, 111

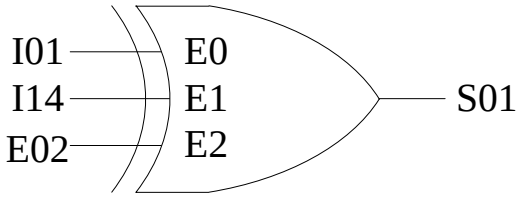


Figura 4.10 Compuerta EOR de tres entradas sin preinversión.

4.5 DESCRIPCIÓN DE LOS MÓDULOS LÓGICOS QUE REALIZAN COMPUERTAS DE CUATRO ENTRADAS.

El PLM3 puede realizar seis tipos de compuertas lógicas de cuatro entradas, estas son de tipo: AND, OR, NAND, OR NEGADA, OR EXCLUSIVA y OR EXCLUSIVA NEGADA, cabe mencionar que además se cuenta con la capacidad de preinversión en las entradas de éstas. En la figura 4.11 se ilustra este ML.

Figura 4.11 Representación genérica del ML de cuatro entradas realizable por el PLM3.

La sintaxis de declaración de este ML en el programa principal es la siguiente:

COMP#N X0I0J0, X1I1J1, X2I2J2, X3I3J3, XSISJS, ABCD

En la tabla 4.5 se hace una breve descripción sobre la sintaxis de declaración asociada con Módulos Lógicos que corresponden a Compuertas de cuatro entradas.

Tabla 4.5 Descripción de la sintaxis de declaración asociada con el ML Compuertas de cuatro entradas.

COMP	Es una cadena de caracteres que simboliza la función efectuada por un ML correspondiente a compuertas de cuatro entradas que se deseen realizar, éstas pueden ser: AND4, OR4, NAND4, ORN4, EOR4, EORN4.
N	Indica el número que corresponde a la compuerta, esto lo define el usuario, la numeración es independiente para cada tipo de compuerta.
X0	Hace referencia a una de las letras "e", "s" o "i", mayúscula o minúscula, esto va a depender de que la variable de entrada E0 a la compuerta sea una VBE, VBI o VBS respectivamente.
I0	Indica el número de grupo correspondiente a la VB declarada como entrada E0 a la compuerta.
J0	Indica el número de bit dentro del grupo I0, asociado a la variable de entrada E0.
X1	Hace referencia a una de las letras "e", "s" o "i", mayúscula o minúscula, esto va a depender de que la variable de entrada E1 a la compuerta sea una VBE, VBI o VBS respectivamente.
I1	Indica el número de grupo correspondiente a la VB declarada como entrada E1 a la compuerta.
J1	Indica el número de bit dentro del grupo I1, asociado a la variable de entrada E1.
X2	Hace referencia a una de las letras "e", "s" o "I" mayúscula o minúscula, esto va a depender de que la variable de entrada E2 a la compuerta sea una VBE, VBI o VBS respectivamente.
I2	Indica el número de grupo correspondiente a la VB declarada como entrada E2 a la compuerta.
J2	Indica el número de bit dentro del grupo I2, asociado a la variable de entrada E2.
X3	Hace referencia a una de las letras "e", "s" o "i", mayúscula o minúscula, esto va a depender de que la variable de entrada E3 a la compuerta sea una VBE, VBI o VBS respectivamente.
I3	Indica el número de grupo correspondiente a la VB declarada como entrada E3 a la compuerta.
J3	Indica el número de bit dentro del grupo I3, asociado a la variable de entrada E3.
XS	Hace referencia a una de las letras "e", "s" o "i", mayúscula o minúscula, esto va a depender de que la variable de salida S de la compuerta sea una VBI o VBS respectivamente.
IS	Indica el número de grupo correspondiente a la VB declarada como salida S de la compuerta.
JS	Indica el número de bit dentro del grupo IS, asociado a la variable de salida S.
A	Corresponde a un dígito binario, que deberá ser cero si se desea que la entrada E3 tenga preinversión, en otro caso el dígito deberá ser uno.
B	Corresponde a un dígito binario, que deberá ser cero si se desea que la entrada E2 tenga preinversión, en otro caso el dígito deberá ser uno.
C	Corresponde a un dígito binario, que deberá ser cero si se desea que la entrada E1 tenga preinversión, en otro caso el dígito deberá ser uno.
D	Corresponde a un dígito binario, que deberá ser cero si se desea que la entrada E0 tenga preinversión, en otro caso el dígito deberá ser uno.

A continuación se muestran dos ejemplos de la declaración de un ML que realiza una compuerta de cuatro entradas.

Ejemplo 4.7

Se desea realizar con el PLM3 una compuerta AND de cuatro entradas, la cual requiere que las entradas E0, E1, E2, E3 y la salida S sean respectivamente las VB E02, E06, I13, E16 y S04, el usuario asignará a esta compuerta el número 1 y sólo las entradas E1, E2 y E3 tienen preinversión, véase la figura 4.12. La sintaxis con la cual se declara el ML en el programa principal es la siguiente:

AND4#1 E02, E06, I13, E16, S04, 0001

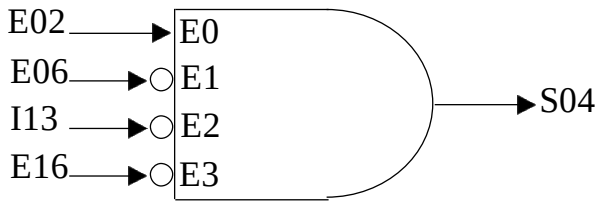


Figura 4.12 Compuerta AND de cuatro entradas con preinversión en las entradas E1, E2 y E3.

Ejemplo 4.8

Se desea realizar con el PLM3 una compuerta EORN de cuatro entradas, la cual requiere que las entradas E0, E1, E2, E3 y la salida S sean respectivamente las VB I34, I56, I21, E06 e I00, el usuario asignará a esta compuerta el número 4, véase la figura 4.13. La sintaxis con la cual se declara el ML en el programa principal es la siguiente:

EORN4#5 I34, I56, I21, E06, I00, 1111

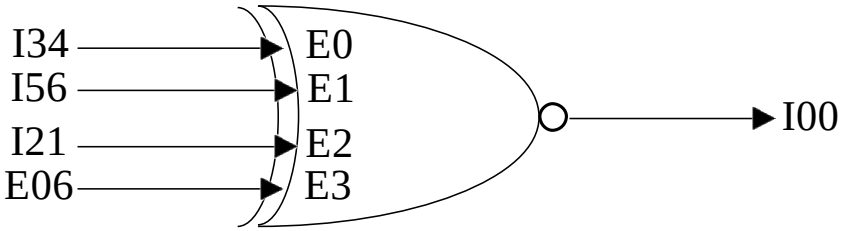


Figura 4.13 Compuerta EORN de cuatro entradas sin preinversión.

4.6 DESCRIPCIÓN DE LOS MÓDULOS LÓGICOS QUE REALIZAN FLIP-FLOP R-S ASÍNCRONOS.

Los elementos capaces de almacenar temporalmente datos provenientes de sus entradas, pasándolos a sus salidas y manteniéndolos son los *latches*, comúnmente conocidos como flip-flops. Estos presentan dos estados estables SET y RESET, se puede entender que dichos elementos son una forma básica de memoria. El PLM3 puede realizar módulos tipo *latch*, que en la nomenclatura del mismo se denominan como flip-flops asíncronos R-S (FFARS).

Este ML tiene la capacidad de preestablecer el nivel de verificación de sus entradas "S" y "R", además, es posible preestablecer también el nivel que éste tendrá a la salida cuando ambas entradas se verifican y el valor que se desea que tome la misma al iniciar el programa en SIIL2. En la figura 4.14 se muestra de forma general un FFARS.

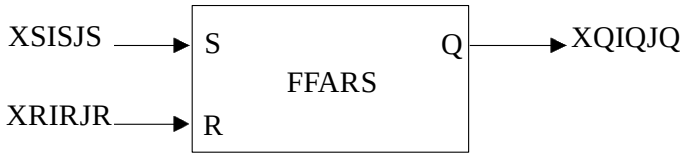


Figura 4.14 Representación genérica de un ML que realiza la función de un flip-flop Asíncrono R-S.

La sintaxis con la cual se declara el ML en el programa principal es la siguiente:

FFARS#N XSISJS, XRIRJR, XQIQJQ, ABC

En la tabla 4.6 se hace una breve descripción sobre la sintaxis de declaración asociada con el ML flip-flop R-S Asíncrono.

Tabla 4.6 Descripción de la sintaxis de declaración asociada con el ML flip-flop R-S asíncrono.

FFARS	Corresponde a una cadena de caracteres que simboliza la función efectuada por un Módulo Lógico flip-flop R-S asíncrono.
N	Corresponde al número de Flip-Flop definido por el usuario.
XS	Se le puede asignar la letra "e", "s" o "i", mayúscula o minúscula, dicha asignación dependerá de si la variable de entrada SET (S) al Flip-Flop es una VBE, VBS o VBI respectivamente.
IS	Indica el número de grupo correspondiente a la VB declarada como entrada S al Flip-Flop.
JS	Indica el número de bit dentro del grupo IS, asociado a la variable de entrada S.
XR	Se le puede asignar la letra "e", "s" o "i", mayúscula o minúscula, dicha asignación dependerá de si la variable de entrada RESET (R) al Flip-Flop es una VBE, VBS o VBI respectivamente.
IR	Indica el número de grupo correspondiente a la VB declarada como entrada R al Flip-Flop.
JR	Indica el número de bit dentro del grupo IR, asociado a la variable de entrada R.
XQ	Se le puede asignar la letra "s" o "i", mayúscula o minúscula, dicha asignación dependerá de si la variable de salida Q del Flip-Flop es una VBS o VBI respectivamente.
IQ	Indica el número de grupo correspondiente a la VB declarada como salida Q del Flip-Flop.
JQ	Indica el número de bit dentro del grupo IQ, asociado a la variable de salida Q.
A	Corresponde a un dígito binario, es cero si se desea que el nivel de verificación de la entrada S sea bajo, para el caso contrario el dígito será uno.
B	Corresponde a un dígito binario, es cero si se desea que el nivel de verificación de la entrada R sea bajo, para el caso contrario el dígito será uno.
C	Corresponde a un dígito binario, es cero si se desea que la salida Q se inicialice en cero lógico, para el caso contrario el dígito será uno.

A continuación se muestra un ejemplo de la declaración de un ML que realiza la función de un FFARS.

Ejemplo 4.9

Se desea realizar con el PLM3 un módulo tipo latch, para el cual se requiere que las entradas S, R y la salida S sean respectivamente las VB E01, E15 y S02, estableciendo las

siguientes condiciones; la entrada S se verifica en alto y la entrada R se verifica en bajo, el estado inicial de la salida Q será cero. A este latch se le asignará el número 2, véase la figura 4.15. La sintaxis con la cual se declara el ML en el programa principal es la siguiente:

FFARS#2 E01, E15, S02, 100

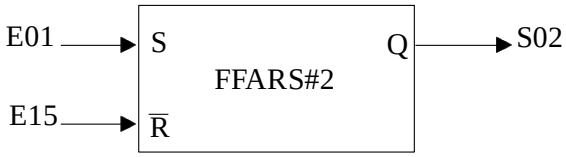


Figura 4.15 Ejemplo de un ML tipo latch realizable con el PLM3.

4.7 DESCRIPCIÓN DEL MÓDULO LÓGICO TEMPORIZADOR MONODISPARO (ONE SHOT) DISPARABLE POR NIVEL.

Los temporizadores son funciones de programación que permiten el control de acciones específicas en función del tiempo. Un temporizador (one shot) tiene la función de generar pulsos de duración fija predeterminada partiendo de un pulso externo de disparo. El PLM3 puede realizar temporizadores de tipo one shot disparables por nivel. En la figura 4.16 se muestra el bloque de este módulo.

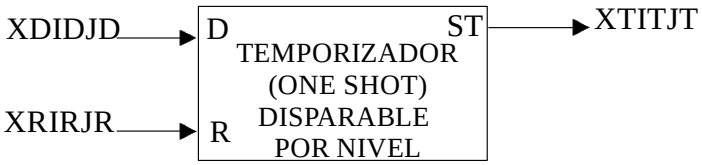


Figura 4.16 Representación genérica de un ML temporizador (one shot) disparable por nivel.

El intervalo de tiempo lo específica el usuario en la declaración sintáctica correspondiente, el tiempo mínimo de salida es de 10 ms y el máximo es de 46 horas con 36 minutos y 12.15 segundos, el disparo puede ser por nivel alto o nivel bajo, además, se tiene una entrada denominada "R" (RESET), que al verificarse coloca a este módulo en condición de espera de disparo, con su salida no verificada.

La entrada R responde al nivel, cuando esta entrada se verifica, la salida queda sin ser verificada y se restablece el contador de tiempo asociado.

En la figura 4.17 se observa el diagrama de tiempos asociado al Temporizador (one shot) disparable por nivel.

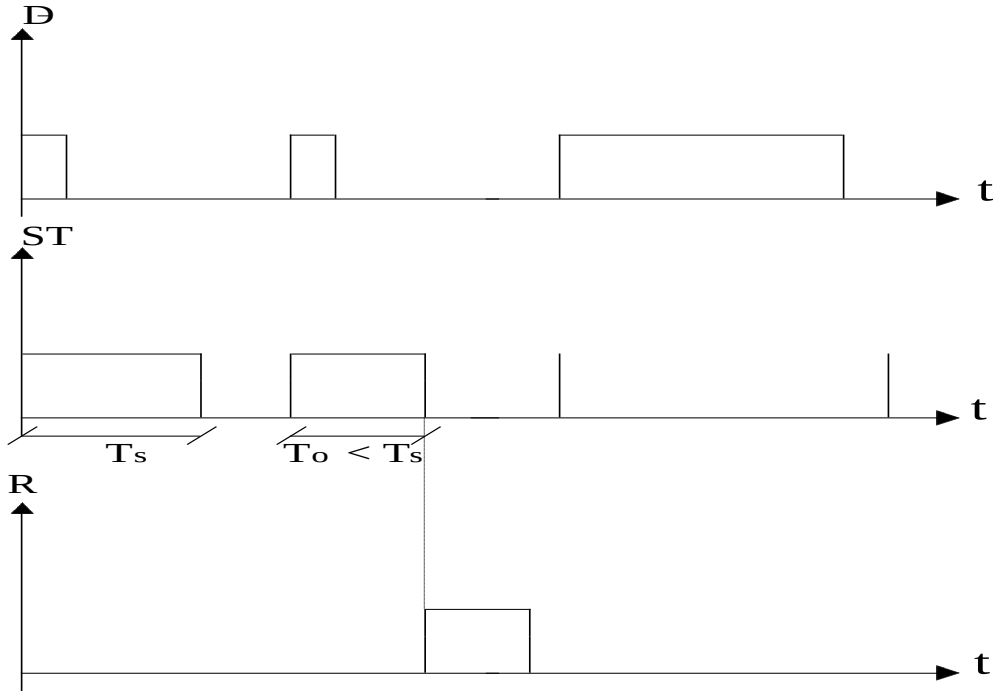


Figura 4.17 Diagrama de tiempos de un temporizador monodisparo redisparable.

La sintaxis con la cual se declara el ML temporizador monodisparo en el programa principal es la siguiente:

TEMPOC#N XDIDJD, XRIRJR, XTITJT, HH.MM.SS.CC, ABC

En la tabla 4.7 se hace una breve descripción sobre la sintaxis de declaración asociada con el ML Temporizador (one shot) disparable por nivel.

Tabla 4.7 Descripción de la sintaxis de declaración asociada con el ML Temporizador (one shot) disparable por nivel.

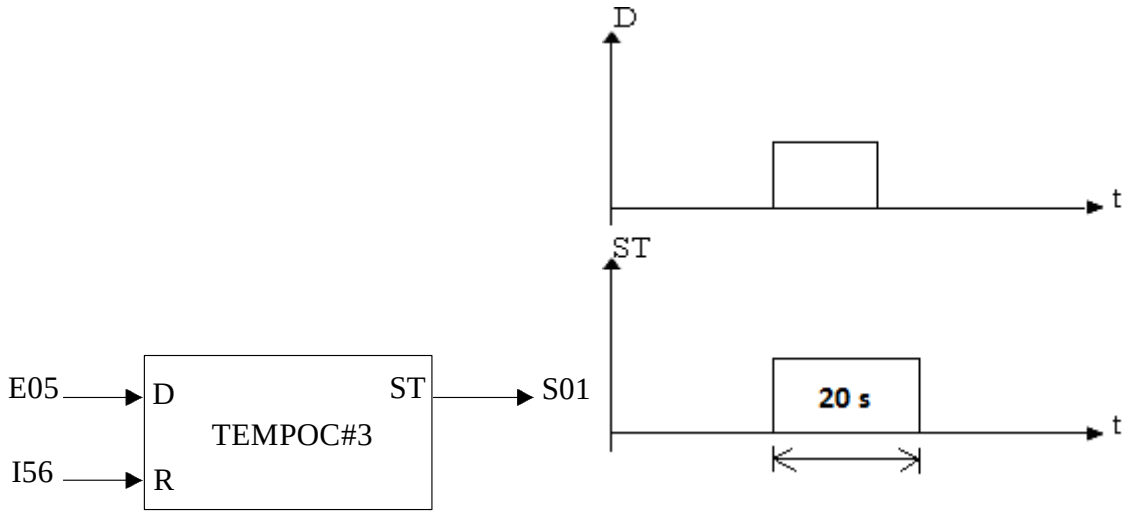
TEMPOC	Corresponde a una cadena de caracteres que simboliza la función efectuada por un Módulo Lógico Temporizador (one shot) disparable por nivel.
N	Corresponde al número de temporizador definido por el usuario.
XD	Se le puede asignar la letra "e", "s" o "i", mayúscula o minúscula, dicha asignación dependerá, si la variable de entrada D al temporizador es una VBE, VBS, VBI respectivamente.
ID	Indica el número de grupo correspondiente a la VB declarada como entrada D al temporizador.
JD	Indica el número de bit dentro del grupo ID, asociado a la variable de entrada D.
XR	Se le puede asignar la letra "e", "s" o "i", mayúscula o minúscula dicha asignación dependerá, si la variable de entrada de restablecimiento R al temporizador es una VBE, VBS, VBI respectivamente.
IR	Indica el número de grupo correspondiente a la VB declarada como entrada R al temporizador.
JR	Indica el número de bit dentro del grupo IR, asociado a la variable de entrada R.
XT	Se le puede asignar la letra "s" o "i", mayúscula o minúscula, dicha asignación dependerá, si la variable de salida ST del temporizador es una VBS, VBI respectivamente.
IT	Indica el número de grupo correspondiente a la VB declarada como salida ST del temporizador.
JT	Indica el número de bit dentro del grupo IT, asociado a la variable de salida ST.
HH	Especifican el número de horas para el tiempo Ts (Tiempo de salida).
MM	Especifican el número de minutos en Ts.
SS	Especifican los segundos en Ts.
CC	Especifican las centésimas de segundo en Ts.
A	Corresponde a un dígito binario, es cero si se desea que el temporizador sea disparado por nivel bajo en la entrada D, para el caso contrario, A deberá de ser uno.
B	Corresponde a un dígito binario, es cero si se desea que el temporizador sea restablecido por nivel bajo, para el caso contrario B tomará el valor de uno.
C	Corresponde a un dígito binario, es cero si se desea que el nivel de verificación de la salida ST sea bajo, para el caso contrario C tomará el valor de uno.

A continuación se muestra un ejemplo de la declaración de un ML temporizador monodisparo disparable por nivel.

Ejemplo 4.10

Se desea realizar utilizando el PLM3 un temporizador (one shot), para la entrada de disparo se le asignará la entrada física E05 y el restablecimiento será la I56, la salida será la S01. Se requiere que el nivel de salida sea verificado por nivel alto con una duración de 20 segundos, el disparo y el restablecimiento deberán de ser por nivel alto. A este temporizador se le asignará el número 3, véase las figuras 4.18. La sintaxis con la cual se declara el ML en el programa principal es la siguiente:

TEMPOC#3 E05, I56, S01, 00.00.20.00, 111



a b

Figura 4.18 Representación, a) del bloque del temporizador one shot descrito en el ejemplo 4.10, b) diagrama de tiempos del temporizador one shot del mismo ejemplo.

4.8 DESCRIPCIÓN DEL MÓDULO LÓGICO QUE REALIZA TEMPORIZADORES CON RETARDO A LA CONEXIÓN (ON-DELAY) O CON RETARDO A LA DESCONEXIÓN (OFF-DELAY).

El PLM3 ha sido diseñado para implementar temporizadores *on-delay* y *off-delay* empleando un solo módulo. En las figuras 4.19 y 4.20 se representa respectivamente el bloque correspondiente a este ML, así como, los diagramas de tiempo asociados al mismo.

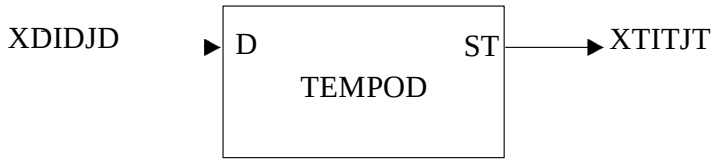
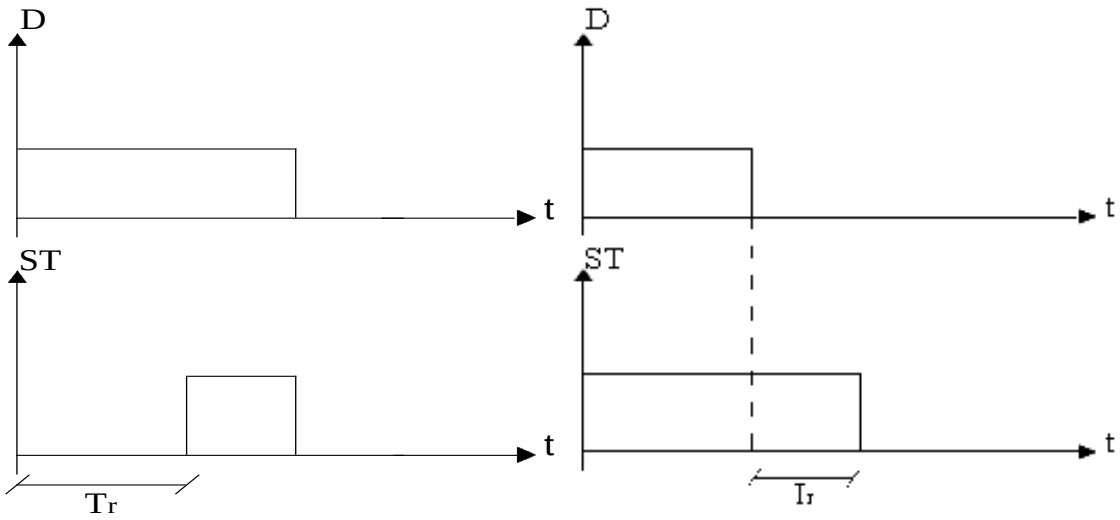


Figura 4.19 Representación general de un temporizador con retardo a la conexión (on-delay) o retardo a la desconexión (off-delay).



(a) on-delay (b) off-delay

Figura 4.20 Representación del diagrama de tiempos de un temporizador con capacidad de retardo a la conexión (a) y a la desconexión (b).

El intervalo de tiempo lo específica el usuario en la declaración sintáctica correspondiente, de esta forma, dicho tiempo puede estar comprendido entre 10 ms y 46 horas con 36 minutos y 12.15 segundos.

La sintaxis con la cual se declara el ML temporizador con retardo a la conexión o a la desconexión en el programa principal es la siguiente:

TEMPOD#N XDIDJD, XTITJT, HH.MM.SS.CC, A

En la tabla 4.8 se hace una breve descripción sobre la sintaxis de declaración asociada con el ML Temporizador on-delay y off-delay.

Tabla 4.8 Descripción de la sintaxis de declaración asociada con el ML Temporizador on-delay y off-delay

TEMPOD	Corresponde a una cadena de caracteres que simboliza la función efectuada por un Módulo Lógico Temporizador on-delay y off-delay.
N	Corresponde al número de temporizador definido por el usuario.
XD	Se le puede asignar la letra "e", "s" o "i", mayúscula o minúscula, dicha asignación dependerá de si la variable de entrada D al temporizador es una VBE, VBS, VBI respectivamente.
ID	Indica el número de grupo correspondiente a la VB declarada como entrada D al temporizador.
JD	Indica el número de bit dentro del grupo ID, asociado a la variable de entrada D.
XT	Se le puede asignar la letra "s" o "i", mayúscula o minúscula, dicha asignación dependerá de si la variable de salida ST del temporizador es una VBS, VBI respectivamente.
IT	Indica el número de grupo correspondiente a la VB declarada como salida ST del temporizador.
JT	Indica el número de bit dentro del grupo IT, asociado a la variable de salida ST.
HH	Especifican el número de horas para el tiempo Tr (Tiempo de retardo).
MM	Especifican el número de minutos en Tr.
SS	Especifican los segundos en Tr.
CC	Especifican las centésimas de segundo en Tr.
A	Corresponde a un dígito binario, es cero si se desea que el temporizador actúe con retardo a la desconexión (off-delay), para el caso contrario, el temporizador estará programado para que opere con retardo a la conexión (on-delay).

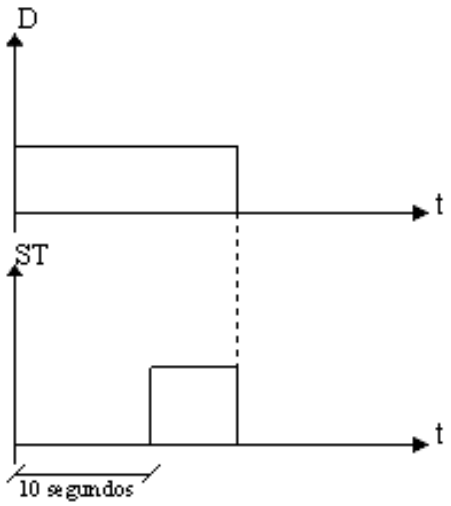
A continuación se muestra un ejemplo de la declaración de dos ML, uno con retardo a la conexión y el otro con retardo a la desconexión.

Ejemplo 4.11

Se debe diseñar utilizando el PLM3 dos temporizadores, uno con retardo a la conexión y el otro con retardo a la desconexión. Para el primero se tendrá un retardo a la conexión de 10 segundos y se le asignará el número 5. La entrada de disparo será la VB E02 y la salida deberá ser la VB I25. Para el segundo temporizador, se requiere que presente un retardo a la desconexión de 5 segundos. Se le asignará el número 15, la entrada de disparo será la VB I23 y la salida debe ser la VB S07, véase la figura 4.21. La sintaxis con la cual se declara cada uno de los ejemplos anteriores corresponde a la siguiente:

TEMPOD#5 E02,I25,00.00.10.00,1

TEMPOD#15 I23,S07,00.00.05.00,0



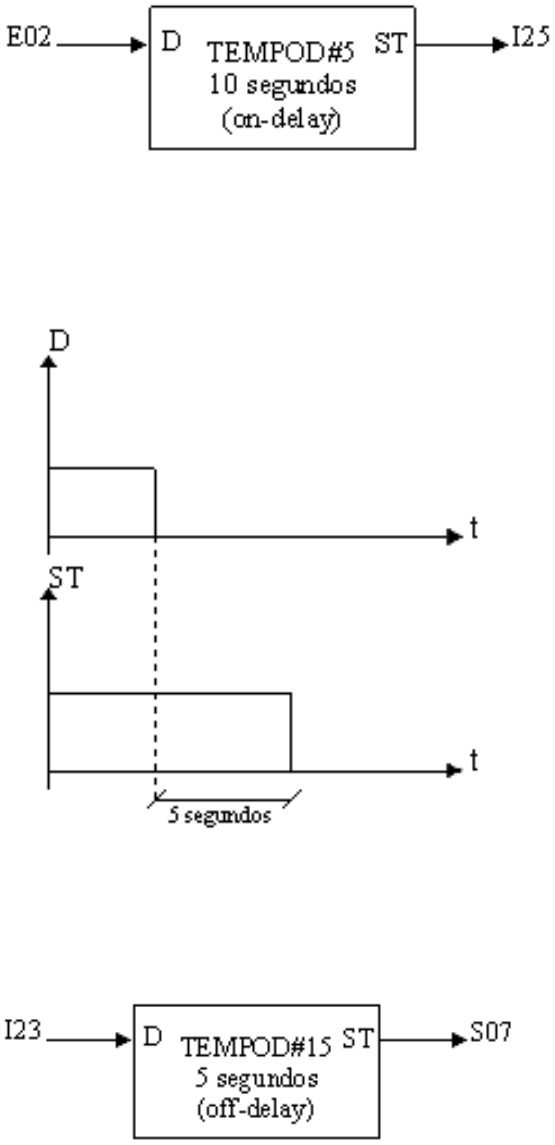


Figura 4.21 Representación a manera de bloques y diagrama de tiempos de los dos temporizadores del ejemplo 4.11.

4.9 DESCRIPCIÓN DEL MÓDULO LÓGICO QUE REALIZA UN TEMPORIZADOR ASTABLE.

El PLM3 cuenta con un ML temporizador capaz de generar una señal periódica binaria con un periodo y un determinado ciclo de trabajo que puede ser fijado por el usuario. A este tipo de temporizadores se les conoce normalmente con el nombre de astables. En la figura 4.22 se muestra el bloque asociado a este ML y en la figura 4.23 el diagrama de tiempo asociado.

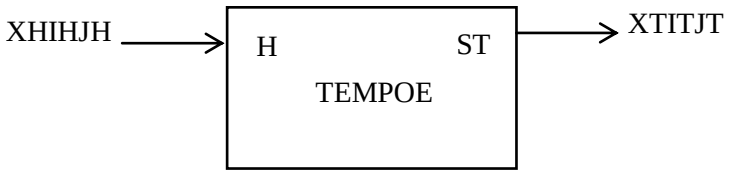


Figura 4.22 Representación genérica de un ML Temporizador astable.

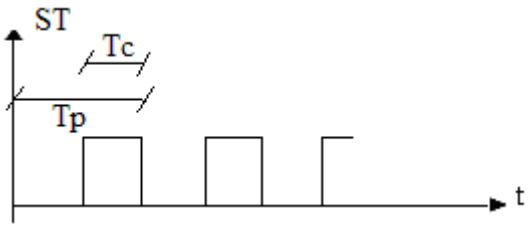


Figura 4.23 Representación del diagrama de tiempos de un temporizador astable

Los tiempos Tp (tiempo de periodo) y Tc (tiempo de ciclo de trabajo), son configurados por el usuario y pueden estar comprendidos entre 10 ms y 46 horas con 36 minutos y 12.15 segundos con la única condición de que Tc sea siempre menor a Tp. La sintaxis con la cual se declara el ML temporizador astable en el programa principal es la siguiente:

TEMPOE#N XHIHJH, XTITJT, HHp.MMp.SSp.CCp, HHc.MMc.SSc.CCc, A

En la tabla 4.9 se hace una breve descripción sobre la sintaxis de declaración asociada con el ML Temporizador Astable.

Tabla 4.9 Descripción de la sintaxis de declaración asociada con el ML Temporizador Astable.

TEMPOE	Corresponde a una cadena de caracteres que simboliza la función efectuada por un Módulo Lógico Temporizador Astable.
N	Corresponde al número de temporizador definido por el usuario.
XH	Se le puede asignar la letra "e", "s" o "i", mayúscula o minúscula, dicha asignación dependerá de si la variable de entrada H al temporizador es una VBE, VBS, VBI respectivamente.
IH	Indica el número de grupo correspondiente a la VB declarada como entrada H al temporizador.
JH	Indica el número de bit dentro del grupo IH, asociado a la variable de entrada H.
XT	Se le puede asignar la letra "s" o "i", mayúscula o minúscula, dicha asignación dependerá de si la variable de salida ST del temporizador es una VBS, VBI respectivamente.
IT	Indica el número de grupo correspondiente a la VB declarada como salida ST del temporizador.
JT	Indica el número de bit dentro del grupo IT, asociado a la variable de salida ST.
HHp	Especifican el número de horas para el tiempo Tp (Tiempo de periodo).
MMp	Especifican el número de minutos en Tp.
SSp	Especifican los segundos en Tp.
CCp	Especifican las centésimas de segundo en Tp.
HHc	Especifican el número de horas para el tiempo Tc (Tiempo de ciclo de trabajo).
MMc	Especifican el número de minutos en Tc.
SSc	Especifican los segundos en Tc.
CCc	Especifican las centésimas de segundo en Tc.
A	Corresponde a un dígito binario, es cero si se desea que la habilitación del temporizador sea por nivel bajo, en caso contario el digito deberá ser uno.

A continuación se muestra un ejemplo de la declaración de un ML temporizador astable y su sintaxis correspondiente.

Ejemplo 4.12

Se desea realizar con el PLM3 un temporizador astable, los datos son los siguientes: Tp y Tc corresponden a 3 segundos y 1.5 segundos respectivamente, la habilitación será por nivel bajo, se asignará a este temporizador el número 18 y las VB que ocupa el ML son la E13 para la entrada e I25 para la salida, véase la figura 4.24. La sintaxis con la cual se declara este temporizador es la siguiente:

TEMPOE#18 E13, I25, 00.00.03.00, 00.00.01.50, 0

Figura 4.24 Representación a manera de bloque del temporizador del ejemplo 4.12.

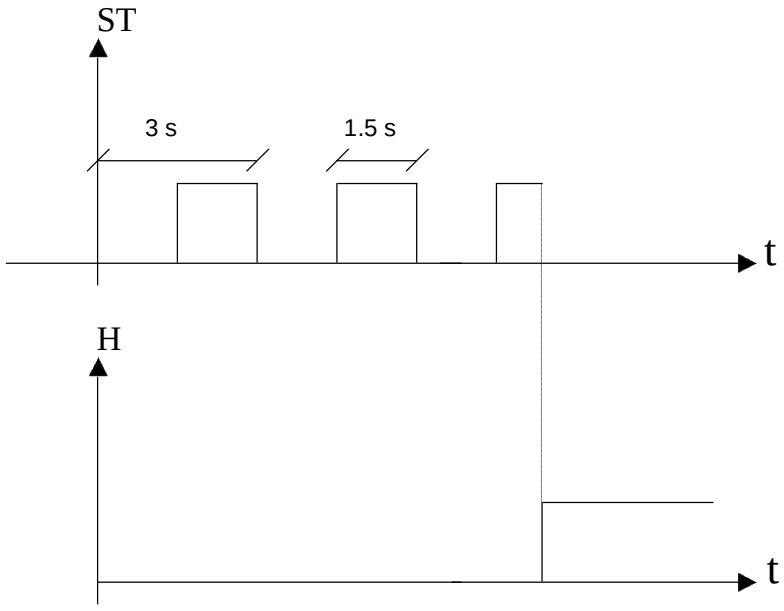


Figura 4.25 Diagrama de tiempos del temporizador del ejemplo 4.12.

4.10 DESCRIPCIÓN DEL MÓDULO LÓGICO TEMPORIZADOR MONODISPARO (ONE SHOT) DISPARABLE POR FLANCO.

Este ML es similar al del subtema 4.7, diferenciándose su comportamiento en la habilitación de cada uno de ellos. El PLM3 puede realizar temporizadores de tipo one shot disparables por flanco. En la figura 4.26 se muestra el bloque de este módulo lógico.

Figura 4.26 Representación genérica de un ML temporizador (one shot) disparable por flanco.

El intervalo de tiempo lo específica el usuario en la declaración sintáctica correspondiente, el tiempo mínimo de salida es de 10 ms y el máximo es de 10 minutos con 55 segundos y 36 centésimas. El disparo puede ser por flanco de subida o flanco de bajada, además, se tiene una entrada denominada "R" (RESET), que al verificarse coloca a este módulo en condición de espera de disparo, con su salida no verificada.

La entrada R responde al nivel, cuando esta entrada se verifica, la salida queda desverificada y se restablece el contador de tiempo asociado.

En la figura 4.27 se muestra el diagrama de tiempos asociado al ML Temporizador disparable por flanco.

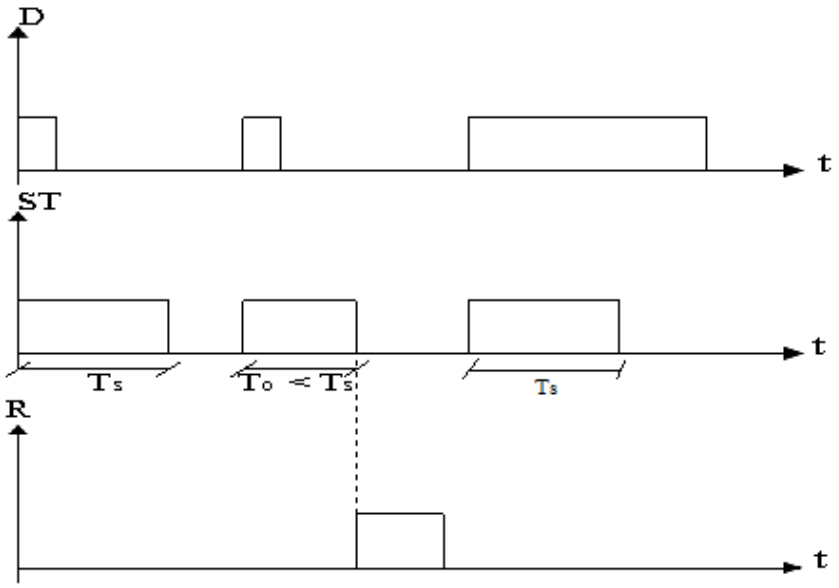


Figura 4.27 Diagrama de tiempos de un temporizador monodisparo redisparable.

La sintaxis con la cual se declara el ML temporizador monodisparo en el programa principal es la siguiente:

TEMPOF#N XDIDJD, XRIRJR, XTITJT, HH.MM.SS.CC, ABC

En la tabla 4.10 se hace una breve descripción sobre la sintaxis de declaración asociada con el ML Temporizador Monodisparo (one shot) disparable por flanco.

Tabla 4.10 Descripción de la sintaxis de declaración asociada con el ML Temporizador Monodisparo (one shot) disparable por flanco.

TEMPOF	Corresponde a una cadena de caracteres que simboliza la función efectuada por un Módulo Lógico Temporizador Monodisparo (one shot) disparable por flanco.
N	Corresponde al número de temporizador definido por el usuario.
XD	Se le puede asignar la letra "e", "s" o "i", mayúscula o minúscula, dicha asignación dependerá, si la variable de entrada D al temporizador es una VBE, VBS, VBI respectivamente.
ID	Indica el número de grupo correspondiente a la VB declarada como entrada D al temporizador.
JD	Indica el número de bit dentro del grupo ID, asociado a la variable de entrada D.
XR	Se le puede asignar la letra "e", "s" o "i", mayúscula o minúscula dicha asignación dependerá, si la variable de entrada de restablecimiento R al temporizador es una VBE, VBS, VBI respectivamente.
IR	Indica el número de grupo correspondiente a la VB declarada como entrada R al temporizador.
JR	Indica el número de bit dentro del grupo IR, asociado a la variable de entrada R.
XT	Se le puede asignar la letra "s" o "i", mayúscula o minúscula, dicha asignación dependerá, si la variable de salida ST del temporizador es una VBS, VBI respectivamente.
IT	Indica el número de grupo correspondiente a la VB declarada como salida ST del temporizador.
JT	Indica el número de bit dentro del grupo IT, asociado a la variable de salida ST.
HH	Especifican el número de horas para el tiempo Ts (Tiempo de salida).
MM	Especifican el número de minutos en Ts.
SS	Especifican los segundos en Ts.
CC	Especifican las centésimas de segundo en Ts.
A	Corresponde a un dígito binario, es cero si se desea que el temporizador sea verificado por flanco de bajada en la entrada de disparo D, para el caso contrario, A deberá de ser uno para que se verifique por flanco de subida.
B	Corresponde a un dígito binario, es cero si se desea que el temporizador sea restablecido por nivel bajo, para el caso contrario B tomará el valor de uno.
C	Corresponde a un dígito binario, es cero si se desea que el nivel de verificación de la salida ST sea bajo, para el caso contrario C tomará el valor de uno.

A continuación se muestra un ejemplo de la declaración de un ML temporizador disparable por flanco.

Ejemplo 4.13

Se desea realizar utilizando el PLM3 un temporizador (one shot) disparable por flanco, para la entrada de disparo se le asignará la entrada física E00 y el restablecimiento será la S02, la salida se presentará en la VB I23. El nivel de salida se debe verificar en bajo con una duración de 5 minutos con 3.5 segundos, el disparo deberá ser por flanco de subida, y el restablecimiento deberá ser por nivel alto. A este temporizador se le asignará el número 13, véase las figuras 4.28. La sintaxis con la cual se declara el ML en el programa principal es la siguiente:

TEMPOF#13 E00, S02, I23, 00.05.03.50, 110

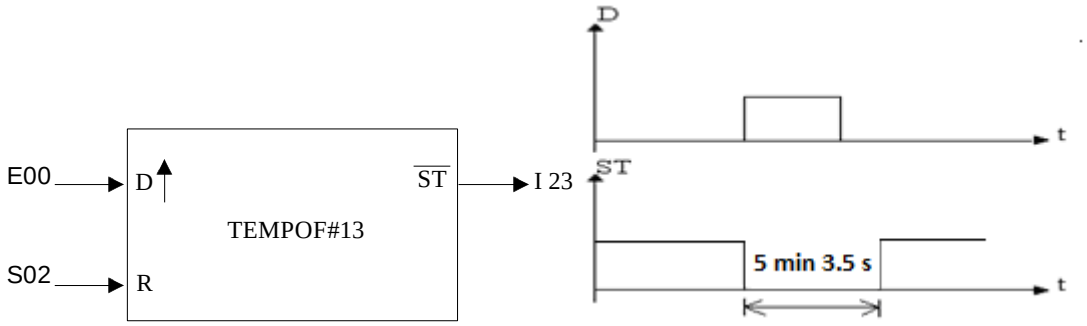


Figura 4.28 Representación en bloque y diagrama de tiempos del temporizador one shot del ejemplo 4.13.

4.11 DESCRIPCIÓN DEL MÓDULO LÓGICO QUE REALIZA UN CONTADOR DE EVENTOS

En los dispositivos programables existen elementos llamados contadores cuya función es realizar el conteo de eventos externos o internos. En aplicaciones de control que utilizan contadores se pueden desarrollar alguna de las siguientes funciones:

- Contar hasta alcanzar un valor preestablecido (cuenta completa) y hacer que un evento ocurra.
- Hacer que ocurra algún evento hasta que el conteo llegue al valor de la cuenta completa.

El PLM3 puede realizar módulos contadores de eventos ascendentes. Estos contadores son capaces de comparar los valores de cuenta acumulada (número actual de conteos que han sido registrados por un contador ascendente) con valores preestablecidos (número de cuentas al que debe contar un contador ascendente para obtener una señal de cuenta completa). Los valores de la cuenta pueden estar comprendidos dentro de un determinado intervalo y el usuario puede definir tanto la cuenta inicial (CUENTAI) como la cuenta final (CUENTAF).

Este ML presenta dos entradas: la primer entrada denominada "D" ha sido diseñada para que funcione por flancos de bajada o flancos de subida, los cambios a cada flanco hacen posible que se modifique la cuenta. La segunda entrada es de restablecimiento (RESET), cuando esta entrada es verificada hace que la cuenta retorne a su valor inicial. La salida de este módulo (TFC) se verifica cuando la cuenta ha llegado a su valor final. En la figura 4.29 se observa la representación de este módulo.

Este contador de eventos ha sido diseñado para que el usuario pueda predefinir los límites del intervalo de cuenta, el tipo de flanco que al ser detectado incrementa la cuenta, el nivel de verificación de la entrada de RESET, así como, el nivel de verificación de la salida testigo de cuenta final (TFC).

Figura 4.29 Representación genérica de un ML contador de eventos realizable en el PLM3.

La sintaxis con la cual se declara el ML temporizador monodisparo en el programa principal es la siguiente:

CONTEV#N XDIDJD, XRIRJR, XFIFJF, CUENTAI, CUENTAF, ABC

En la tabla 4.11 se hace una breve descripción sobre la sintaxis de declaración asociada con el ML Temporizador Contador de Eventos.

Tabla 4.11 Descripción de la sintaxis de declaración asociada con el ML Contador de Eventos.

CONTEV	Corresponde a una cadena de caracteres que simboliza la función efectuada por un Módulo Lógico Contador de Eventos.
N	Indica el número de contador de eventos definido por el usuario.
XD	Se le puede asignar la letra "e", "s" o "i", dicha asignación dependerá, si la variable de entrada D al contador es una VBE, VBS, VBI.
ID	Indica el número de grupo que corresponda a la VB declarada como entrada D al contador.
JD	Indica el número de bit dentro del grupo ID, asociado a la variable de entrada D.
XR	Se le puede asignar la letra "e", "s" o "i", dicha asignación dependerá, si la variable de entrada RESET (R) al contador es una VBE, VBS, VBI.
IR	Indica el número de grupo que corresponda a la VB declarada como entrada R al contador.
JR	Indica el número de bit dentro del grupo IR, asociado a la variable de entrada R.
XF	Se le puede asignar la letra "s" o "i", dicha asignación dependerá, si la variable de salida TFC del contador es una VBS, VBI.
IF	Indica el número de grupo que corresponda a la VB declarada como salida TFC del contador.
JF	Indica el número de bit dentro del grupo IF, asociado a la variable de salida del contador.
CUENTAI	Indica el valor de la cuenta de inicio, el cual puede estar comprendido entre 0 y 65535. Este valor debe ser siempre menor que el de la cuenta final debido a que es un contador de eventos ascendente.
CUENTAF	Indica el valor de la cuenta final, el cual está comprendido entre 0 y 65535.
A	Corresponde a un dígito binario, es cero si se desea que la cuenta incremente por flancos de bajada en la entrada de disparo D. Para el caso contrario, A se le asignará el valor de uno y la cuenta se incrementará cuando ocurran flancos de subida.
B	Corresponde a un dígito binario, es cero si se desea que el nivel de verificación de la entrada de RESET sea bajo, para el caso contrario, B tomará el valor de uno.
C	Corresponde a un dígito binario, es cero si se desea que el nivel de verificación de la salida (TFC) sea bajo, para el caso contrario C tomará el valor de uno y la salida se verificará en alto.

A continuación se muestra un ejemplo de la declaración de un ML contador de eventos.

Ejemplo 4.14

Mediante la utilización del PLM3, se desea realizar un módulo contador de eventos con un intervalo de cuenta entre cero y nueve, la entrada de disparo D será detectada por flancos de subida, requiriéndose que las entradas D y R sean respectivamente las VB I23, E05 y que la salida TFC sea la VB S07. La testificación de final de carrera del contador es por nivel bajo, el nivel de verificación de la entrada de RESET será por nivel alto y el usuario decide darle el número 50 a este contador, véanse las figura 4.30 y 4.31. La sintaxis con la cual se declara el ML en el programa principal es la siguiente:

CONTEV#50 I23, E05, S07, 0, 9, 110

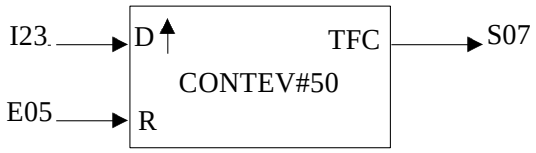


Figura 4.30 Representación en bloque de un módulo contador de eventos del ejemplo 4.14. La cuenta va ir en ascenso cada vez que se detecte un flanco de subida.

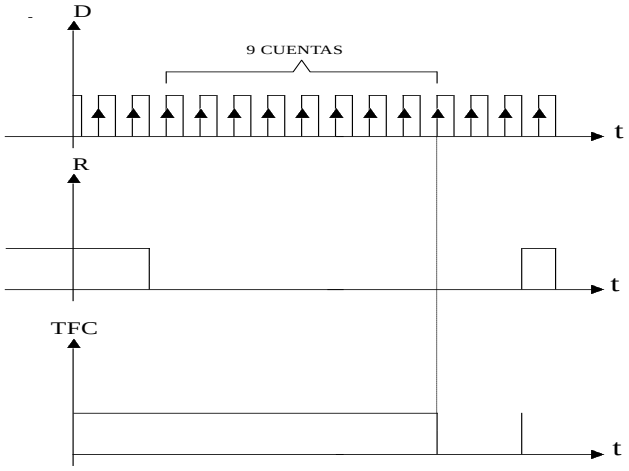


Figura 4.31 Diagrama de tiempos asociados al contador de eventos del ejemplo 4.14.

4.12 DESCRIPCIÓN DEL MÓDULO LÓGICO QUE REALIZA UN SECUENCIADOR DE ESTADOS

El PLM3 cuenta con un ML secuenciador de estados de un bit, el nivel asociado a cada uno de los bits de la secuencia se declara en renglones subsecuentes a la declaración propia de este ML. Si se deseara realizar un secuenciador estados donde la palabra básica sea de más de un bit, se deberá utilizar un ML secuenciador por cada uno de los bits que contenga la palabra binaria implicada.

Este ML presenta dos entradas; la primera entrada denominada D ha sido diseñada para que funcione por flancos de bajada o flancos de subida, los cambios a cada flanco hacen posible que se modifique el estado actual al estado siguiente. La segunda entrada es de restablecimiento (RESET), cuando esta entrada es verificada hace que el bit de estado tome su valor inicial. Cuenta también con dos salidas, la primera de ellas denominada como SB, indica el valor actual del bit de estado una vez que éste es habilitado y la segunda salida corresponde al testigo de fin de carrera denominada como FC. En la figura 4.32 se observa la representación de este módulo.

Este secuenciador ha sido diseñado para que el usuario pueda predefinir los valores binarios que tendrá el estado, el tipo de flanco que al ser detectado cambia el valor del estado conforme a la declaración, así como el nivel de verificación de la entrada de RESET.

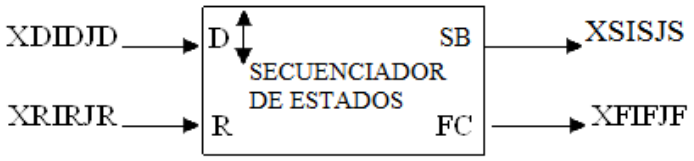


Figura 4.32 Representación genérica de un ML secuenciador de estados realizable en el PLM3.

La sintaxis con la cual se declara el ML secuenciador de estados en el programa principal es la siguiente:

SECBIT#N XDIDJD,XRIRJR,XSISJS,XTITJT,ABC
DATSECBIT#N F1,F2,F3,…,Fi,…,Fi+1,…,F1000
FINDATSECBIT#N

En la tabla 4.12 se hace una breve descripción sobre la sintaxis de declaración asociada con el ML Secuenciador de Estados.

Tabla 4.12 Descripción de la sintaxis de declaración asociada con el ML Secuenciador de Estados.

SECBIT	Corresponde a una cadena de caracteres que simboliza la función efectuada por un Módulo Lógico Secuenciador de Estados.
N	Indica el número de contador de eventos definido por el usuario.
XD	Se le puede asignar la letra "e", "s" o "i", dicha asignación dependerá, si la variable de entrada D al secuenciador es una VBE, VBS, VBI.
ID	Indica el número de grupo que corresponda a la VB declarada como entrada D al secuenciador.
JD	Indica el número de bit dentro del grupo ID, asociado a la variable de entrada D.
XR	Se le puede asignar la letra "e", "s" o "i", dicha asignación dependerá, si la variable de entrada RESET (R) al secuenciador es una VBE, VBS, VBI.
IR	Indica el número de grupo que corresponda a la VB declarada como entrada R al secuenciador.
JR	Indica el número de bit dentro del grupo IR, asociado a la variable de entrada R.
XS	Se le puede asignar la letra "s" o "i", dicha asignación dependerá, si la variable de salida SB del secuenciador es una VBS, VBI.
IS	Indica el número de grupo que corresponda a la VB declarada como salida TFC del secuenciador.
JS	Indica el número de bit dentro del grupo IS, asociado a la variable de salida del secuenciador.
XF	Se le puede asignar la letra "s" o "i", dicha asignación dependerá, si la variable de salida FC del secuenciador es una VBS, VBI.
IF	Indica el número de grupo que corresponda a la VB declarada como salida FC del secuenciador.
JF	Indica el número de bit dentro del grupo IF, asociado a la variable de salida del secuenciador.
A	Corresponde a un dígito binario, es cero si se desea que el secuenciador se habilite por flancos de bajada en la entrada de disparo D. Para el caso contrario, A se le asignará el valor de uno y el secuenciador se habilitará por flancos de subida.
B	Corresponde a un dígito binario, es cero si se desea que el nivel de verificación de la entrada de RESET sea bajo, para el caso contrario, B tomará el valor de uno.
C	Corresponde a un dígito binario, es cero si se desea que el nivel de verificación de la salida (FC) sea bajo, para el caso contrario C tomará el valor de uno.
Fi	Corresponde a los niveles binarios que tomará el bit de estado.
Fi+1	Corresponde al valor binario de fin de carrera de un estado, siendo este indispensable para que todos los estados deseados sean observados por el usuario, en caso contrario, el último estado no será visible. El valor de éste es irrelevante.

A continuación se muestran dos ejemplos de la declaración de un ML secuenciador de estados.

Cabe señalar que podrán usarse uno o varios renglones con la sentencia **DATSECBIT#N** al inicio para declarar los niveles lógicos de los bits de estado a generar.

Ejemplo 4.15

El usuario desea implementar con el PLM3 un secuenciador de estados de tres bits por tres estados, este deberá ser repetitivo, con un tiempo entre estados de medio segundo, y no debe ser habilitado por el usuario. La VB de disparo D habilitada por flanco de subida es en general I21, SB y FC del primer secuenciador son S00 e I22 respectivamente, SB y FC del segundo secuenciador son S01 e I23 respectivamente. Para el tercer secuenciador las variables SB y FC son S02 e 124 respectivamente, todas las variables FC se verifican en nivel alto. El RESET general para éstos se verificará en nivel bajo y lo ocupará la VB I25. El usuario decide asignar a estos secuenciadores los números 32, 33 y 34 respectivamente, el sistema lógico asociado se muestra figura 4.33; la lista de estados a secuenciar se muestra a continuación:

Estado 01 001

Estado 02 010

Estado 03 100

*Estado 04 111

* Es el estado que se debe poner para que los estados anteriores a éste sean observados en la aplicación, esto para cuando se realiza un secuenciador de estados repetitivo.

Las declaraciones en SIIL2 asociadas con el sistema lógico de estre ejemplo se muestran a continuación:

Para la generación de la señal periódica que generará la señal de disparo se ocupa un temporizador astable con periodo de medio segundo y CT de 4 %. La declaración asociada es:

TEMPOE#2 I06, I21, 00.00.00.50, 00.00.0.02, 1

Declaración de los secuenciadores:

SECBIT#32 I21, I25, S00, I23, 101

DATSECBIT#32 1,0,0,1

FINDATSECBIT#32

SECBIT#33 I21, I25, S01, I24, 101

DATSECBIT#33 0,1,0,1

FINDATSECBIT#33

SECBIT#34 I21, I25, S02, I24, 101

DATSECBIT#34 0, 0, 1, 1

FINDATSECBIT#34

El temporizador que genera la señal de Reset es disparado por nivel, con pulso de salida de 200 ms, su declaración es la siguiente:

TEMPOC#3 S02, I02, I25, 00.00.20.00, 100

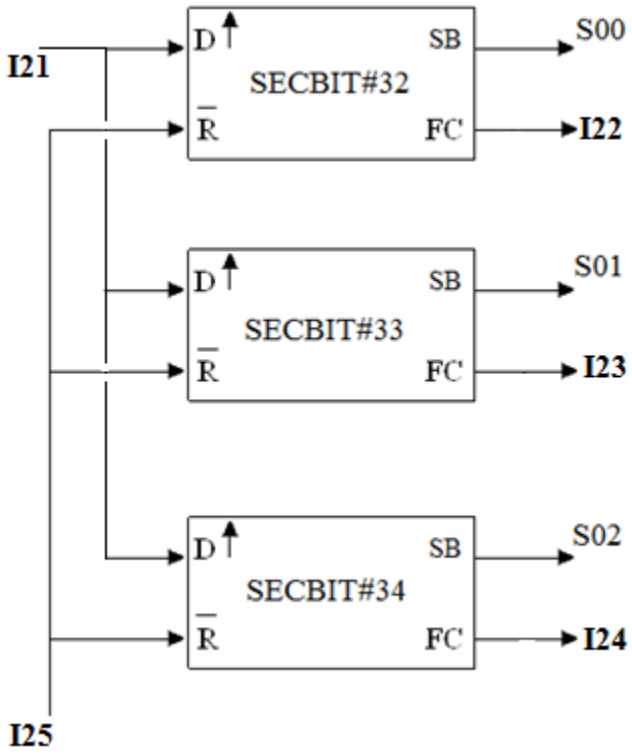


Figura 4.33 Representación en bloques de un ML secuenciador de tres bits por tres estados La habilitación comienza cada vez que se detecte un flanco de subida.

Ejemplo 4.16

El usuario desea implementar con el PLM3 un secuenciador de estados de tres bits por tres estados, se requiere que sea observado cada vez que es habilitado por el usuario. La VB de disparo D habilitada por flanco de subida es en general la entrada física E00, SB y FC del primer secuenciador son S00 y S03 respectivamente, SB y FC del segundo secuenciador son S01 y S04 respectivamente. Para el tercer secuenciador las variables SB y FC son S02 y S05 respectivamente, todas las variables FC se verifican en nivel alto. El RESET general para éstos se verificará en nivel bajo y lo ocupa la variable I22 cuando la variable FC del tercer secuenciador se verifica en alto. El usuario decide asignar a estos secuenciadores los números 32, 33 y 34 respectivamente, véase la figura 4.34; la lista de estados a secuenciar se muestra a continuación:

Estado 01 100

Estado 02 010

Estado 03 001

La sintaxis con la cual se declara el ML temporizador de tres estados en el programa principal es la siguiente:

```
SECBIT#32 E00, I22, S00, S03, 101

DATSECBIT#32 0, 0, 1

FINDATSECBIT#32

SECBIT#33 E00, I22, S01, S04, 101

DATSECBIT#33 0, 1, 0

FINDATSECBIT#33

SECBIT#34 E00, I22, S02, S05, 101

DATSECBIT#34 0, 0, 1, 1

FINDATSECBIT#34
```

Generación de reset mediante un ML de inversión:

```
NOT#2 S05, I22
```

En este caso no es necesario crear un estado de más, puesto que los secuenciadores, al verificarse el reset, quedan en espera de la habilitación del usuario y es posible observar los estados deseados.

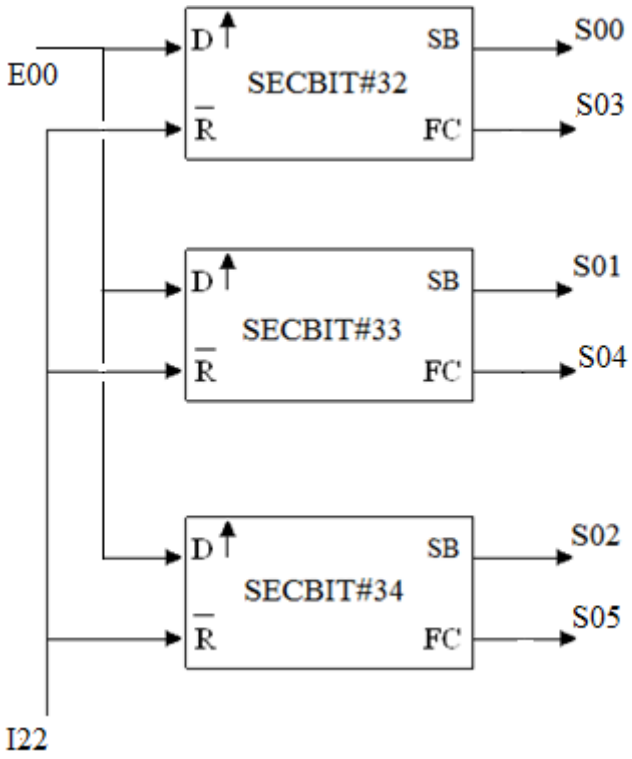


Figura 4.34 Representación en bloques de un ML secuenciador de tres bits por tres estados La habilitación comienza cada vez que se detecte un flanco de subida habilitada por el usuario.

4.13 DESCRIPCIÓN DEL MÓDULO LÓGICO QUE REALIZA UNA COMPARACIÓN

El PLM3 cuenta con un ML capaz de realizar una comparación de la cuenta de un contador de eventos con un número establecido por el usuario. Este comparador ha sido diseñado para que el usuario pueda predefinir el valor numérico con el cual, al verificarse la condición establecida, la salida del ML se verifique en nivel alto o bajo, también predefinido por el usuario. El ML de comparación no cuenta con entradas, y sólo se debe establecer una condición, véase la figura 4.35. La sintaxis con la cual se declara el ML de comparación en el programa principal es la siguiente:

VS_SI_CTACEV_XXX_NUM_#NCEV XSISJS,A

En la tabla 4.13 se hace una breve descripción sobre la sintaxis de declaración asociada con el ML Comparador.

Tabla 4.13 Descripción de la sintaxis de declaración asociada con el ML Comparador.

XXX	Corresponde a la condición que se desea establecer, y puede ser una de las seis siguientes: IGL IGUAL DIF DIFERENTE MAQ MAYOR QUE MEQ MENOR QUE MAI MAYOR O IGUAL MEI MENOR O IGUAL
NUM	Indica el número a comparar, establecido por el usuario.
NCEV	Indica el número correspondiente al contador de eventos con el cual se realizará la comparación.
XS	Se le puede asignar la letra "s" o "i", dicha asignación dependerá, si la variable de salida S del comparador es una VBS, VBI.
IS	Indica el número de grupo que corresponda a la VB declarada como salida s del comparador.
JS	Indica el número de bit dentro del grupo IF, asociado a la variable de salida del comparador.
A	Corresponde a un dígito binario, es cero si se desea que el nivel de verificación de la salida S sea bajo, para el caso contrario A tomará el valor de uno.

Figura 4.35 Representación genérica de un ML de comparación realizable en el PLM3.

A continuación se muestran dos ejemplos de la declaración de un ML de comparación.

Ejemplo 4.17

Se desea realizar con el PLM3 dos ML de comparación. Éstos deben cumplir con las condiciones que se establecen a continuación: al superarse el valor 20 con respecto a las cuentas del contador de eventos establecido por el usuario con el número 6, la salida S del comparador1 se verifica con nivel alto ocupado la VB física S01, mientras que, cuando se iguale el valor de 100

con las cuentas del contador de eventos establecido por el usuario con el número 18, la salida S del comparador2 se verifica con nivel bajo, estableciendo para ésta la VB I35, véase la figura 4.36. La sintaxis con la cual se declaran los ML del ejemplo anterior en el programa principal es la siguiente:

VS_SI_CTACEV_MAI_20_#6 S01,1

VS_SI_CTACEV_IGL_100_#18 I35,0

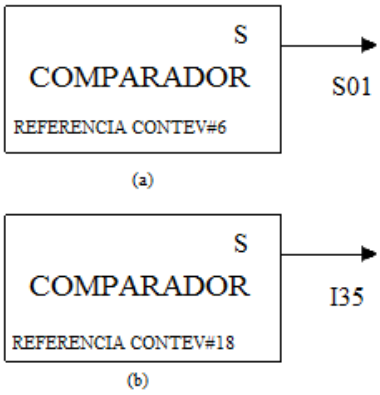


Figura 4.36 Representación en bloques de los comparadores del ejemplo 4.17.

CAPÍTULO 5.

DISEÑO DEL HARDWARE DE ENTRADAS, SALIDAS Y FUENTE DE ALIMENTACIÓN DEL PLM3

En el presente capítulo se describe el diseño de las tarjetas asociadas a los bloques de entradas, la computadora central (CC), las salidas y fuente de alimentación del PLM3. Son cuatro los bloques funcionales del dispositivo, se inicia con la computadora central, continuando con la circuitería contenida en el bloque de entradas correspondiente a la etapa de optoacoplamiento de las 16 entradas físicas, posteriormente se explica la etapa de potencia de las 8 salidas físicas correspondiente a la circuitería contenida en el bloque de salidas, finalizando con la fuente de alimentación del PLM3 (Véase la figura 3.1 del capítulo 3).

5.1 COMPUTADORA CENTRAL (CC).

La CC del PLM3 es la tarjeta MINICON_08GT (desarrollada en el Departamento de Control de la Facultad de Ingeniería de la UNAM), y está basada en el microcontrolador (MCU) MC9S08GT60 de Freescale (en el capítulo 2 se mencionan las generalidades de la tarjeta MINICON_08GT). En la figura 5.1 se puede apreciar el diagrama de pines del MCU que se utilizó para el diseño de esta tarjeta y en la figura 5.2 un diagrama a bloques del mismo. Al final de esta tesis hay un apéndice con información complementaria referente a los circuitos, conceptos y funcionamiento de la tarjeta de desarrollo.

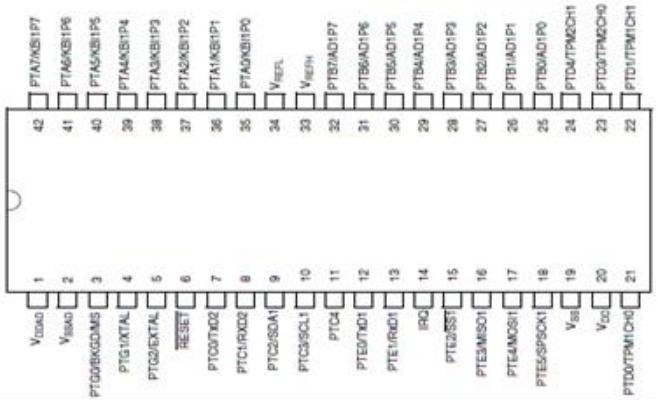


Figura 5.1 Microcontrolador MC9S9S08GT encapsulado DIP.

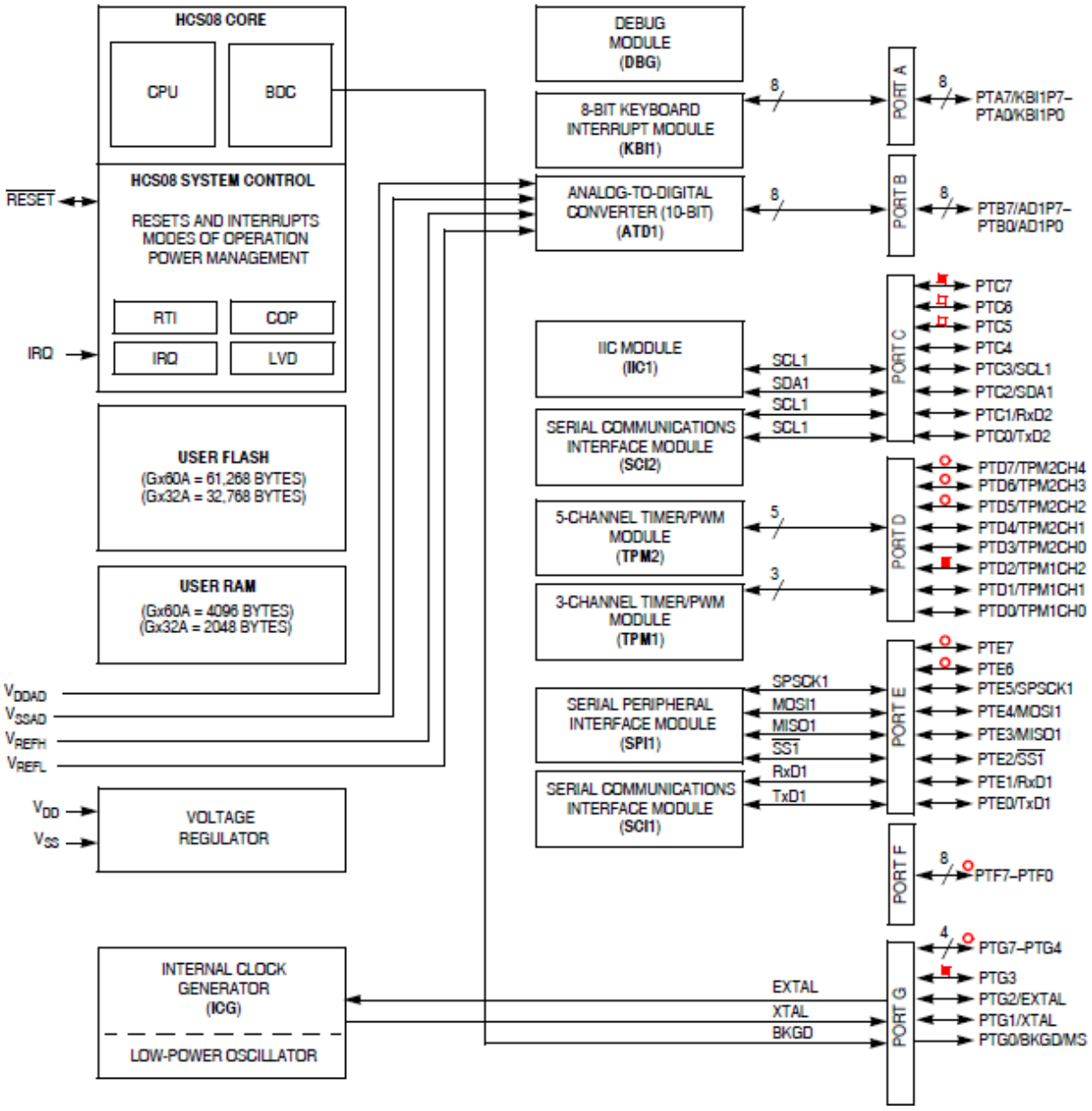


Figura 5.2 Diagrama de bloques del MCU MC9S9S08GT.

5.2 DISEÑO ASOCIADO AL HARDWARE DEL BLOQUE DE ENTRADAS.

El bloque de entradas del PLM3 está constituido por 16 entradas físicas, éstas reciben las señales provenientes de los sensores o de los interruptores que se encuentran en el sistema de control con el cual se desea trabajar. Generalmente en la industria, el nivel lógico de estas señales se encuentra en un rango de 0-24 volts el cual es totalmente diferente al nivel lógico con el cual opera la CC (0-3.3 volts), debido a lo anterior, se emplea para cada entrada un circuito de optoacoplamiento.

En la tabla 5.1 se muestran que bits de puerto del MCU corresponden a cada una de las entradas físicas del PLM3

Tabla 5.1 Líneas de puerto asociadas al hardware de entradas físicas del PLM3.

VB FÍSICA DEL PLM3	BIT DE PUERTO DEL MCU	DIRECCIÓN
E00	PTB0	ENTRADA
E01	PTB1	ENTRADA
E02	PTB2	ENTRADA
E03	PTB3	ENTRADA
E04	PTB4	ENTRADA
E05	PTB5	ENTRADA
E06	PTC0	ENTRADA
E07	PTC1	ENTRADA
E10	PTD0	ENTRADA
E11	PTD1	ENTRADA
E12	PTD3	ENTRADA
E13	PTD4	ENTRADA
E14	PTE2	ENTRADA
E15	PTE3	ENTRADA
E16	PTE4	ENTRADA
E17	PTE5	ENTRADA

En la siguiente figura se observa una descripción generalizada sobre la función que realiza el optoacoplador.

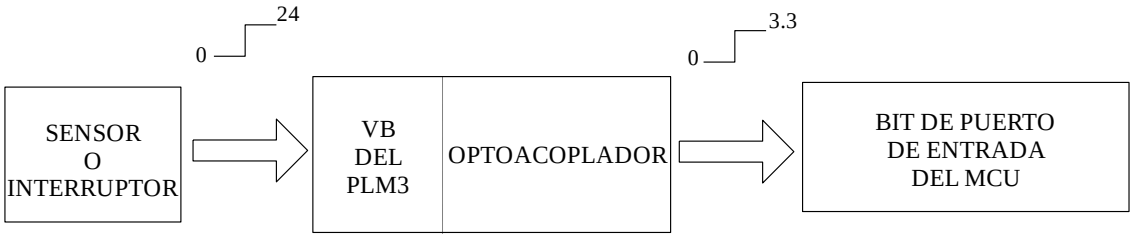


Figura 5.3 Representación genérica de la función del circuito optoacoplador empleado para una entrada física del PLM3.

Para fines de sus entradas físicas, el PLM3 permite configurar cada una de éstas, de modo que puedan aceptar sensores que contienen fuente propia; o bien, sensores que son simples

interruptores que se conectarían entre los bornes denotados genéricamente como EXX y NEUTRO que son las que se observan en la figura 5.4. Para configurar una u otra de las dos modalidades mencionadas existe para cada entrada un interruptor de 2 polos 2 tiros como se aprecia también en la figura 5.4.

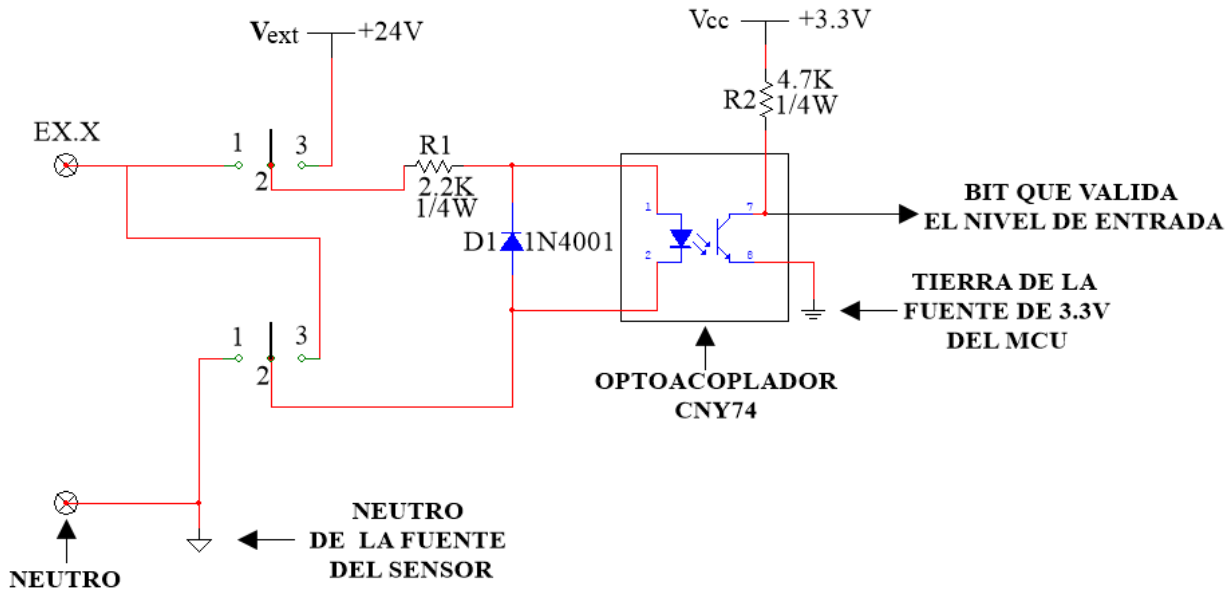


Figura 5.4 Circuito de entrada del PLM3 donde se observa que las uniones entre los bornes no se encuentran conectados (circuito abierto).

A continuación se explica el diseño del circuito de optoacoplamiento para señales provenientes de sensores, después se describirá el diseño del circuito de optoacoplamiento de señales provenientes de interruptores.

5.2.1 Circuitos de optoacoplamiento para sensores de entrada con fuente propia.

Cuando se va a conectar un sensor con fuente propia a una entrada física del PLM3 el interruptor mencionado tendrá que establecer el contacto físico entre las terminales 1-2 del circuito. En la figura 5.5 se ilustra este caso.

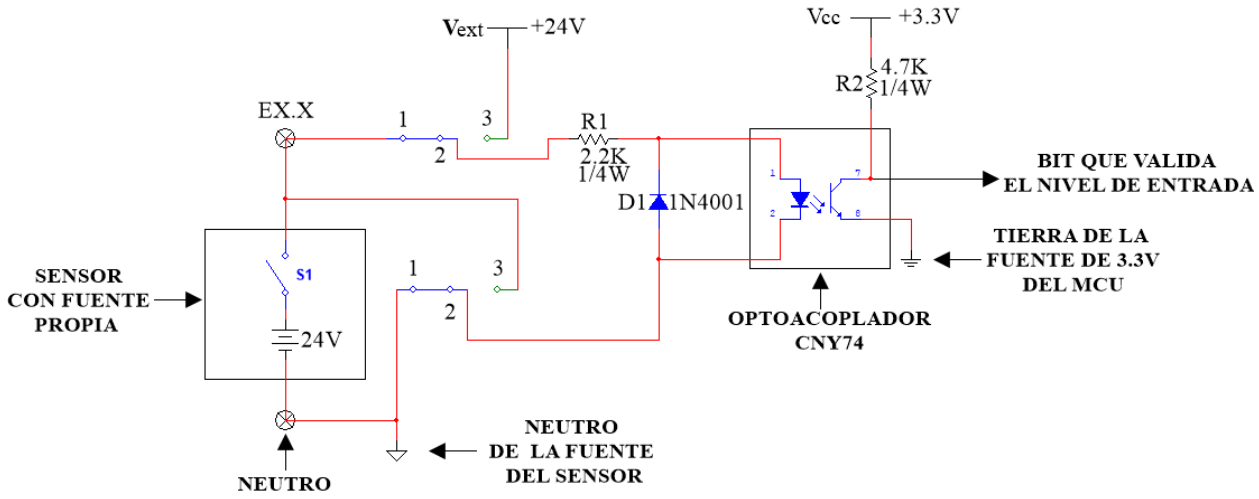


Figura 5.5 Circuito de optoacoplamiento utilizando un sensor a la entrada con fuente propia de alimentación.

Cuando se presenta un valor de 24 volts (1 lógico) en la entrada del optoacoplador, el transistor de éste trabaja en la región de saturación (contacto cerrado) y el voltaje a la salida es de cero volts, por lo tanto se tiene un nivel bajo a la entrada de la CC. Para el caso contrario, cuando se tiene un valor de cero volts en la entrada (0 lógico), el transistor trabaja en la región de corte (contacto abierto) y el voltaje a la salida es igual al voltaje de alimentación de la tarjeta MINICON_08GT que en este caso es de 3.3V, entonces se tendrá un nivel alto a la entrada de la CC. Como se puede observar, existe una inversión lógica en cada circuito de optoacoplamiento, es por eso que se realizó una inversión por software al momento de que el MCU lee cada uno de los puertos asociados con cada grupo de entradas.

A continuación se describe el diseño del circuito de optoacoplamiento de señales provenientes de interruptores.

5.2.2 Circuitos de optoacoplamiento por contacto seco.

Cuando se requiera de un contacto simple del interruptor propio del sensor entre las terminales EXX y NEUTRO propias de la entrada, el interruptor aquí mencionado deberá estar cerrado para las terminales 2-3 del circuito. A este tipo de conexionado frecuentemente se le denomina como contacto seco. En la figura 5.6 se muestra este caso.

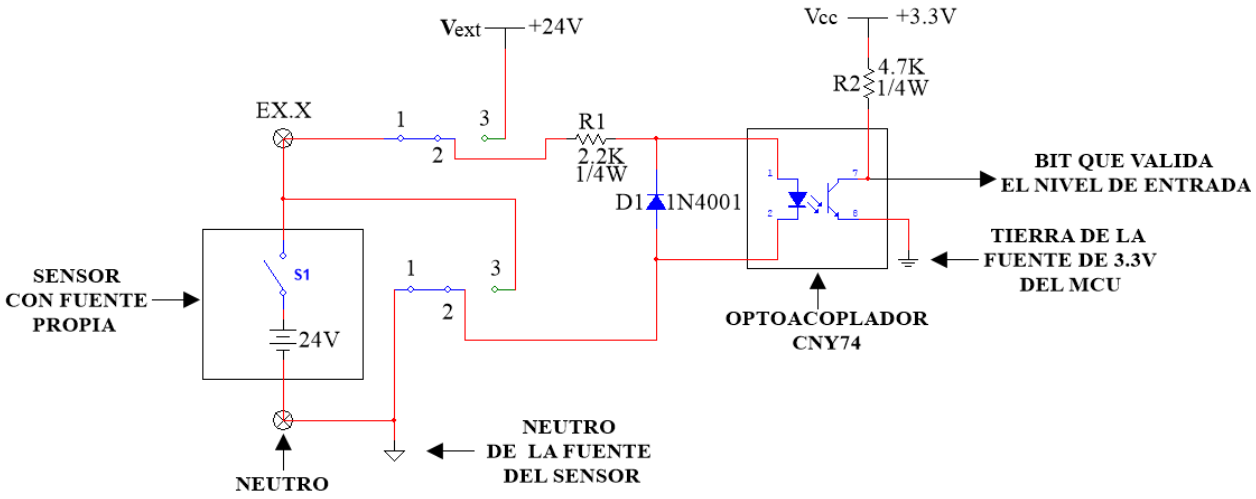


Figura 5.6 Circuito de optoacoplamiento utilizando un contacto seco.

Los optoacopladores se encuentran agrupados en una tarjeta que contiene cuatro circuitos integrados CNY74, cada uno de éstos, tiene a su vez cuatro circuitos como el que se muestra en la figura 5.4, de esta forma se tienen los 16 circuitos de optoacoplamiento empleados para las 16 entradas físicas. El diodo evita que el optoacoplador trabaje en inversa, es decir, es una protección para evitar que éste se dañe.

En la figura 5.7 se muestra el esquemático diseñado y utilizado en el hardware de entradas, realizado en un programa para diseño de circuitos esquemáticos y PCB's denominado Eagle.

En la figura 5.8 se muestra una foto de la tarjeta asociada al bloque de entradas del PLM3. Esta tarjeta se utilizó para validar el primer prototipo del PLM3.

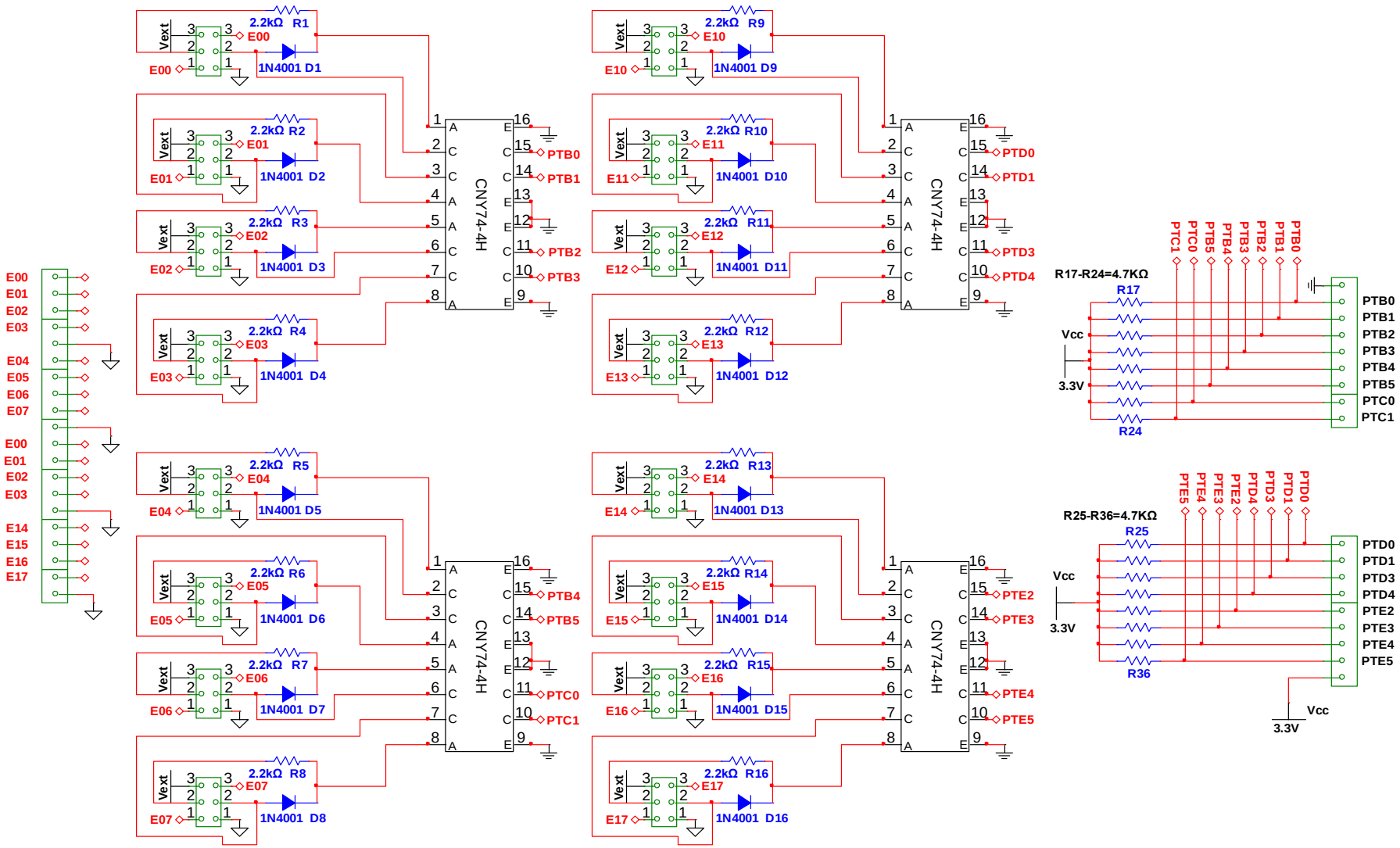


Figura 5.7 Diseño del esquemático de la tarjeta de entradas del PLM3

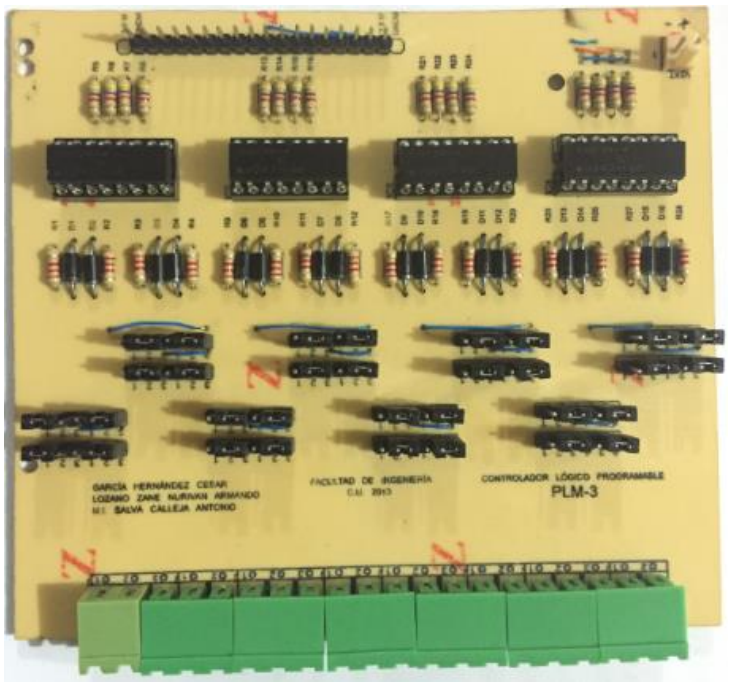


Figura 5.8 Foto de la tarjeta que muestra las 16 entradas del PLM3 (E00-E17).

5.3 DISEÑO ASOCIADO AL HARDWARE DEL BLOQUE DE SALIDAS.

El bloque de salidas del PLM3 está constituido por 8 salidas físicas, éstas reciben las señales provenientes de una interfaz comandada por la CC del PLM3. Se manifiestan mediante la apertura y cierre de los contactos de sendos relevadores de baja potencia. La interfaz es un circuito integrado constituido por ocho drivers, cada uno de ellos a su vez es una circuitería como se muestra en la figura 5.9, el diodo que se encuentra entre la salida (OUT) y el común (COM) es una protección por la bobina del relevador.

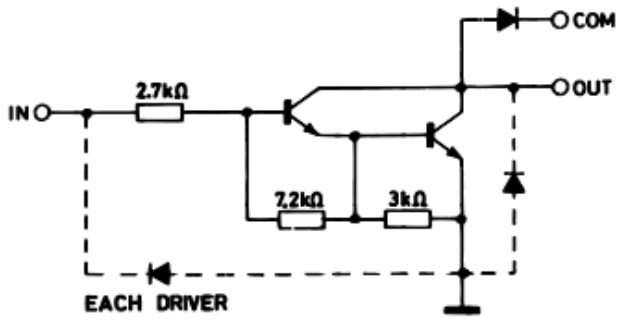


Figura 5.9 Circuitería de un driver de la interfaz para la tarjeta de salidas del PLM3.

En la tabla 5.2 se muestran qué bits de puerto del MCU están asociados con sendos VB de salida del PLM3.

Tabla 5.2 Líneas de puerto asociadas al hardware de salidas físicas del PLM3.

VB FÍSICA DEL PLM3	BIT DE PUERTO DEL MCU	DIRECCIÓN
S0.0	PTA0	SALIDA
S0.1	PTA1	SALIDA
S0.2	PTA2	SALIDA
S0.3	PTA3	SALIDA
S0.4	PTA4	SALIDA
S0.5	PTA5	SALIDA
S0.6	PTA6	SALIDA
S0.7	PTA7	SALIDA

Los relevadores que se emplearon requieren un voltaje de activación de doce volts para la bobina de éstos y la corriente que pueden soportar los contactos es de diez amperes. Se implementó en el diseño de esta tarjeta la opción de que cada salida fuera independiente, es decir, el común de los relevadores puede ser alimentado con el voltaje deseado por el usuario, de esta forma se puede operar hasta con ocho actuadores de diferente voltaje en su alimentación.

La interfaz y los 8 relevadores se encuentran agrupados en una tarjeta que contiene un circuito integrado ULN2803, dentro de éste, se encuentran los ocho drivers uno para cada relevador.

En la figura 5.10 se muestra el diseño esquemático utilizado para la realización del hardware de salidas. Para el desarrollo tanto de la tarjeta de entradas como la de salidas, se utilizó una herramienta de diseño y simulación de circuitos impresos denominada Eagle.

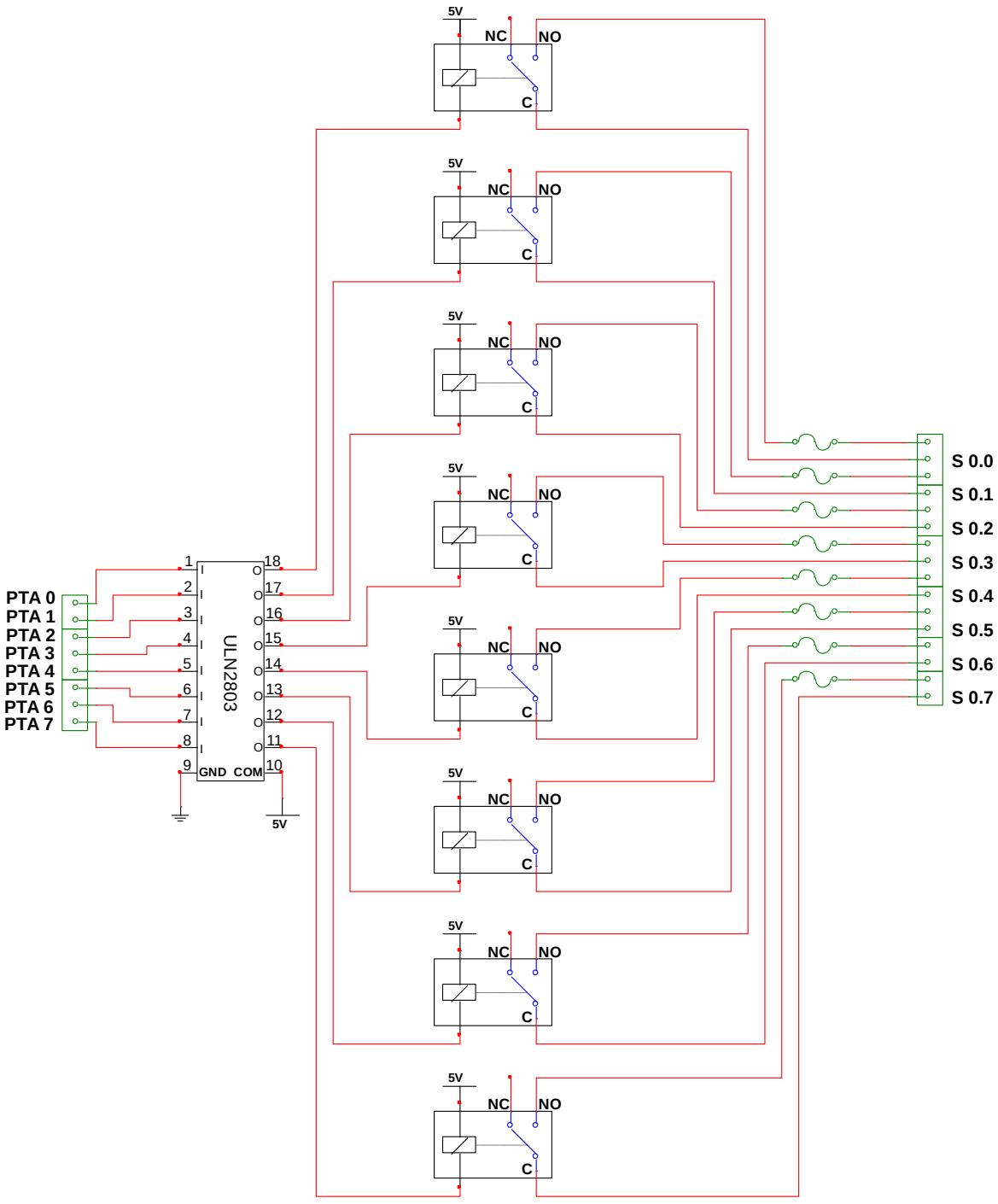


Figura 5.10 Diseño del esquemático de la tarjeta de salidas del PLM3.

En la figura 5.11 se muestra una foto de la tarjeta asociada al bloque de salidas del PLM3, aunque esta primera tarjeta se utilizó para validar el primer prototipo del PLM3.

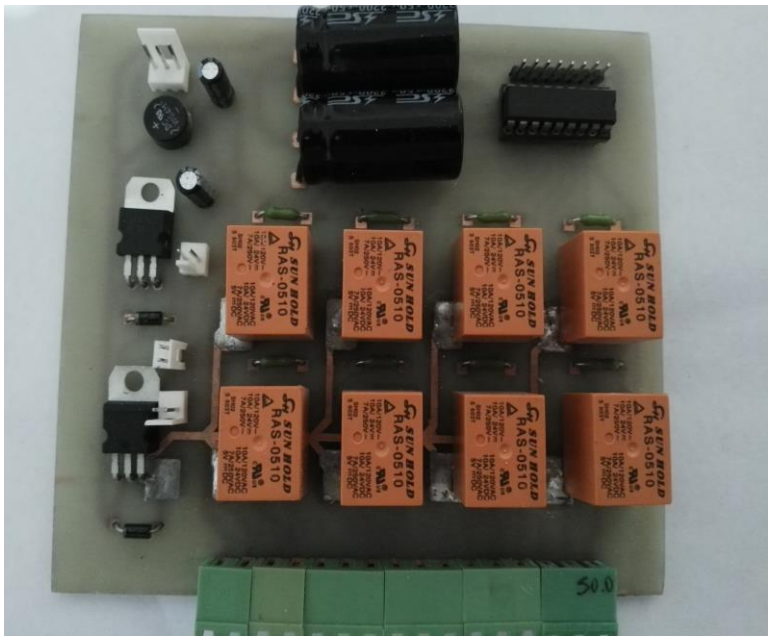


Figura 5.11 Foto de la tarjeta que muestra las 8 salidas del PLM3 (S00-S17).

5.4 DISEÑO ASOCIADO A LA FUENTE DE ALIMENTACIÓN.

La fuente de alimentación del PLM3 está diseñada e implementada para que el usuario no anexe instrumentación externa como soporte para el mismo. De esta forma, el usuario sólo conecta el PLM3 a la línea de 127 VCA y el dispositivo presenta los voltajes de 5 y 3.3 volts, en la tabla 5.3 se describe la funcionalidad de cada uno de estos voltajes.

Tabla 5.3 Descripción de las funciones de los voltajes de alimentación.

Voltaje	Función
5 V	Utilizada para alimentación de las bobinas de los relevadores y configuración de contacto seco.
3.3 V	Utilizada para alimentación de la CC.

El diseño de las fuentes de alimentación se basó en las etapas mostradas en la figura 5.12, cabe mencionar que para cada voltaje regulado se utilizó una filtración propia, es decir, se encuentran en cascada a partir de la rectificación, de esta forma la fuente de menor voltaje no depende de la fuente de un voltaje mayor.

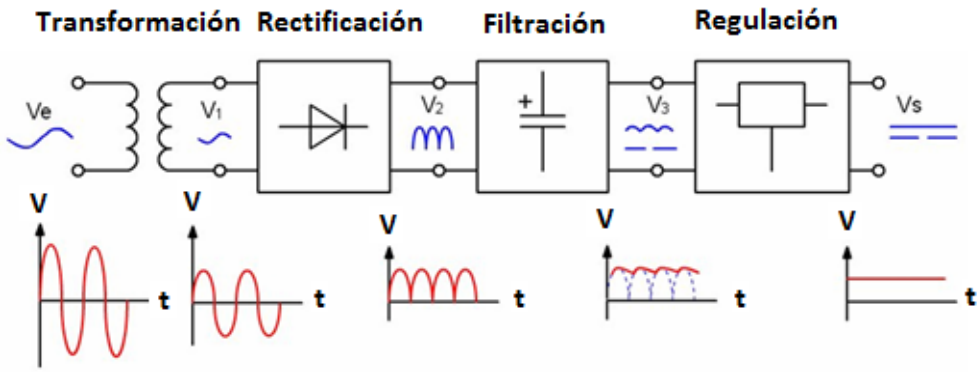


Figura 5.12 Diagrama a bloques de las etapas para generar un voltaje regulado.

Los voltajes mencionados anteriormente se encuentran físicamente visibles, es decir; existen sendas terminales físicas para cada uno de ellos y de esta forma el usuario puede utilizarlos conforme a la demanda de corriente que se especifica en la tabla 5.4, en caso contrario, el usuario se expone al daño del PLM3.

Tabla 5.4 Demanda de corriente de cada una de las fuentes de voltaje.

Voltaje	Demanda de corriente
5 V	1500 mA.
3.3 V	500 mA.

En la figura 5.13 se muestra una foto de la tarjeta asociada a la fuente de alimentación del PLM3, la cual fue implementada junto con la tarjeta de salidas. Aunque esta primera tarjeta se utilizó para validar el primer prototipo del PLM3.

Figura 5.13 Foto de la tarjeta que muestra las fuentes de alimentación del PLM3.

5.5 INTEGRACIÓN FÍSICA DE LAS TARJETAS DEL PLM3

En este primer prototipo, las tarjetas se colocaron de forma vertical y paralelamente, se empleó un gabinete con tapa al cual se le fijaron interiormente unas “L” muy pequeñas para que sujetaran las tarjetas y de esta forma el usuario pudiera disponer de ellas, por ejemplo; para cuando se requiera la configuración de la tarjeta de entradas.

En la figura 5.14 se muestra la fotografía del arreglo de tarjetas que realizan los bloques funcionales del PLM3 en conjunto con el gabinete empleado para este prototipo.

Figura 5.14 Foto del primer prototipo del PLM3.

Con el hardware del PLM3 terminado y con la explicación de la sintaxis de declaración de los diferentes módulos lógicos, se detallará en el siguiente capítulo el manejo del software manejador del PLM3 para su uso en las aplicaciones de control lógico deseadas.

CAPÍTULO 6

SOFTWARE MANEJADOR DEL PLM3

Este capítulo comienza con la presentación de un Software Manejador para desarrollo con el PLM3. Se describen las funcionalidades básicas para familiarizarse con este sistema, las cuales permitirán al usuario realizar programas para ser ejecutados en el PLM3. Se hace la descripción de algunos ejemplos para mostrar los errores de sintaxis que se pueden presentar en un programa fuente. El capítulo finaliza con ejemplos de programación en SIIL2, que validan diversos sistemas lógicos.

6.1 DESCRIPCIÓN GENERAL DEL SOFTWARE MANEJADOR DEL PLM3

El Software Manejador del PLM3 (SWMANPLM3) fue diseñado por un profesor de carrera del Departamento de Control y Robótica de la Facultad de Ingeniería de la UNAM. Mediante este software el usuario programador puede capturar y generar programas en SIIL2, los cuales son procesados por el compilador para finalmente generar el código objeto que se cargará y ejecutará en el procesador de la Computadora Central del PLM3. Para la realización de estos procesos, el PLM3 debe ser manejado a través de un enlace serie desde una PC.

Cuando se invoca el SWMANPLM3, se despliega un cuadro de diálogo como el que se muestra en la figura 6.1. El usuario debe especificar el tipo de procesador con el que se diseñó el PLM3. Cabe mencionar, que el MCU utilizado para desarrollo de la Computadora Central del PLM3 es el MC9S08GT60 de Freescale el cual viene predeterminado, de esta forma, no es necesario que el usuario especifique otro.

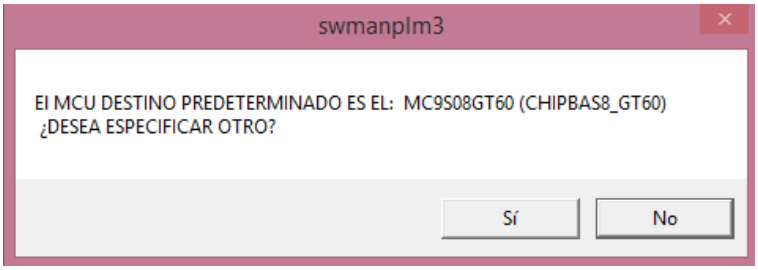


Figura 6.1 MCU de la Computadora Central del PLM3.

Una vez que el usuario ha definido el MCU de la Computadora Central del PLM3, inmediatamente después aparece otro cuadro de diálogo donde se indica el puerto serie de enlace preconfigurado del PLM3 como se muestra en la figura 6.2. Cuando no se tiene la certeza de qué puerto serie está disponible, el usuario debe abrir el Administrador de Dispositivos de Windows y buscar en la lista un puerto serie disponible para el enlace. Si se usa un puerto serie virtual, siempre que se ejecute el SWMANPLM3 el adaptador USB-SERIE debe estar conectado a la PC, de esta manera, se evita un mal funcionamiento del programa.

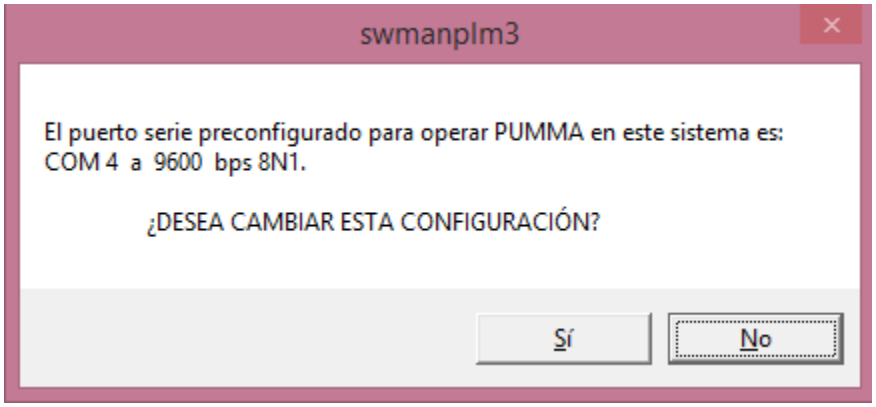


Figura 6.2 Configuración del puerto serie.

Cuando el puerto serie preconfigurado indicado en el diálogo de la figura 6.2 está disponible, el usuario deberá seleccionar "No". En otro caso se deberá oprimir el botón "Sí", lo que hará que aparezcan cuadros de diálogo para que el usuario indique el puerto que usará, quedando éste preconfigurado para subsecuentes ejecuciones.

Se debe definir 9600 bps como baudaje cuando el software así lo requiera, finalmente el SWMANPLM3 confirmará al usuario el baudaje y el número de puerto serie funcional utilizado para el enlace. El procedimiento descrito en los párrafos anteriores debe de realizarse correctamente, de lo contrario se tendrá un mal funcionamiento del SWMANPLM3.

Si el usuario realizó exitosamente los pasos anteriores, se mostrará la ventana de edición del SWMSNPLM3, misma que se describirá en la siguiente sección. Bajo estas condiciones, está todo listo para comenzar a trabajar con el Software Manejador del PLM3 y desarrollar aplicaciones de control lógico.

6.2 DESCRIPCIÓN DE LA VENTANA DE EDICIÓN DEL SWMANPLM3

A continuación se describen las funcionalidades básicas de la ventana de edición del SWMANPLM3

Para capturar un código en SIIL2 con el SWMANPLM3 se abre la ventana de edición como la que se ilustra en la figura 6.3. En esta ventana se pueden apreciar varios botones que realizan diversas acciones al ser oprimidos por el usuario. Para fines ilustrativos, en la figura 6.3 se observa el ejemplo de un programa previamente abierto.

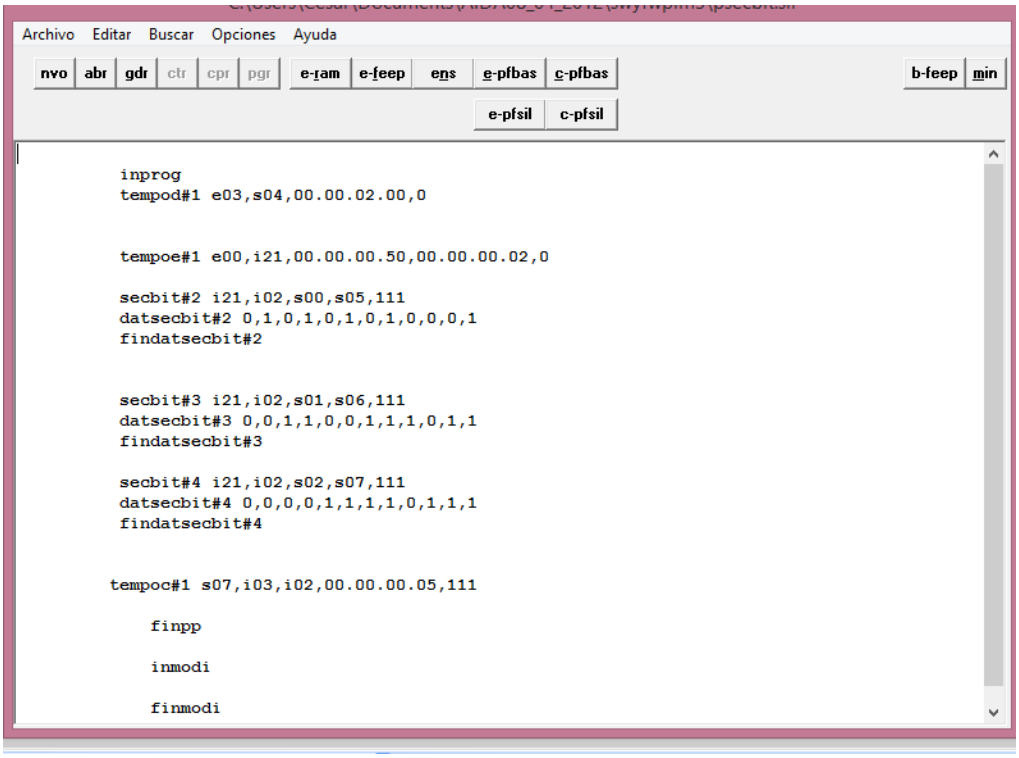


Figura 6.3 Ventana de edición del SWMANPLM3 que contiene el código de un programa fuente.

En la parte superior izquierda de la venta de edición (figura 6.3) se observan tres botones, los cuales están asociados con el funcionamiento del editor. Estos tres botones son:

- ***nvo*** (Crear un archivo nuevo)

- ***abr*** (Abrir un archivo existente)
- ***gdr*** (Guardar el archivo presente)

Además de los botones mencionados en el párrafo anterior existen otros ocho. Para fines del proceso de desarrollo de aplicaciones con el PLM3 se emplean sólo tres de los botones presentes en la ventana del editor. Dos de estos botones presentan las funciones asociadas con la compilación y/o compilación y ejecución de un código fuente, el tercer botón está relacionado con el borrado de la memoria no volátil de la Computadora Central del PLM3. A continuación se describen estos botones:

6.2.1 Botón *c-pfsil* (Compilación)

Cuando se oprime este botón (figura 6.4) se compila el programa presente en la ventana del editor. Para que la compilación sea exitosa no deben existir errores de sintaxis en el mismo programa, de lo contrario se mostrará una ventana donde se reportan una lista de errores que el usuario deberá corregir.

Después de que un programa ha sido compilado sin que haya errores, se generan diversos archivos de salida, uno de estos archivos tiene la extensión ".s19", y contiene el código de máquina ejecutable en el MCU de la Computadora Central del PLM3, asociado con el programa fuente en SIIL2 que se haya compilado.

Cabe señalar, que el usuario debe borrar la memoria no volátil del último programa cargado en el PLM3, de lo contrario el programa actual no podrá ser ejecutado.

Figura 6.4 Botón de la ventana del editor del SWMANPLM3 que compila un programa fuente en SIIL2.

6.2.2 Botón *e-pfsil* (Compilación y ejecución inmediata)

Cuando se oprime este botón (figura 6.5) se compila el programa presente en la ventana del editor; si no existen errores de sintaxis éste se carga y ejecuta de forma inmediata en el PLM3. En caso de que se presenten errores se visualiza una ventana con una lista reportando los errores que el usuario deberá corregir. Es importante señalar, que el usuario debe borrar la memoria no volátil del último programa ejecutado en el PLM3.

Figura 6.5 Botón de la ventana del editor del SWMANPLM3 que compila y ejecuta un programa fuente.

Ahora se describe la función que realiza la activación del botón situado en la parte superior derecha de la venta de edición:

6.2.3 Botón *b-feep* (Borrado de la memoria)

Cuando se oprime este botón (figura 6.6) se borra el último programa que se guardó en la memoria no volátil ejecutado en el PLM3. Cada vez que el usuario desee realizar un nuevo programa fuente, deberá oprimir el botón de borrado de memoria.

Figura 6.6 Botón de la ventana del editor del SWMANPLM3 que borra la memoria donde se guardó el último programa ejecutado.

A continuación se presenta las instrucciones para abrir un programa fuente con el SWMANPLM3:

6.3 DESCRIPCIÓN DE LA OPCIÓN DEL MENU "ABR" DEL SWMANPLM3

Los archivos que se pueden compilar y ejecutar a través del Software Manejador del PLM3 deben llevar la extensión .SIL.

Para abrir un archivo se selecciona la opción "abr" que se encuentra en la venta del editor del SWMANPLM3, posteriormente aparece una nueva ventana como la que se ilustra en la figura 6.7 en la cual, el usuario debe definir el nombre de la carpeta asociada con el archivo que desee abrir.

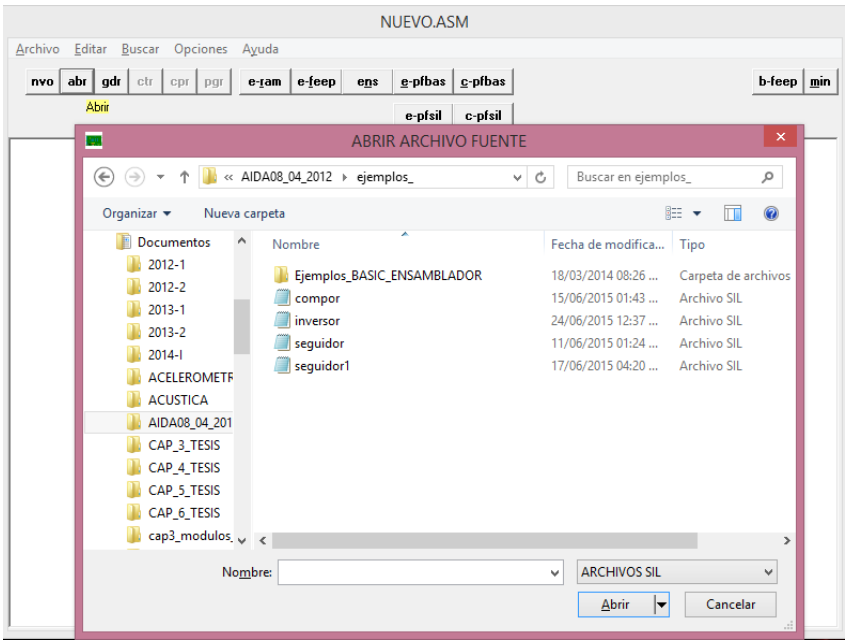


Figura 6.7 Ventana para abrir archivo fuente con extensión .SIL.

Una vez que el usuario haya oprimido el botón "Abrir", se desplegará la ventana del editor donde se observa el código del programa correspondiente al archivo fuente que se seleccionó. Esta descripción se ilustra en la figura 6.8.

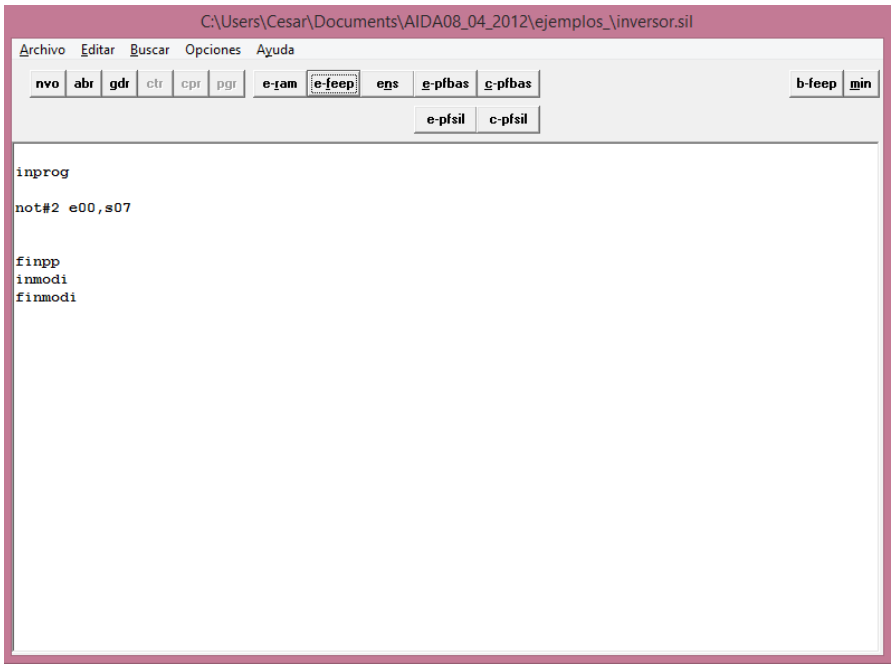


Figura 6.8 Programa fuente en lenguaje SIIL2.

Para fines ilustrativos, se abrió el archivo "inversor.SIL", el cual se trata de un módulo inversor. Las características que se muestran en el código del programa fuente correspondientes a este archivo son las siguientes:

- INPROG ‘Indica la declaración de inicio de subprograma principal.
- NOT#2 E00, S07 ‘Esta línea corresponde a la instrucción asociada con el módulo lógico requerido, que para fines prácticos de esta sección se utilizó un módulo inversor.
- FINPP ‘Es la declaración de fin del subprograma principal.
- INMODI ‘Es la declaración de inicio del subprograma temporizado.
- FINMODI ‘Declaración de inicio de subprograma temporizado.

Cabe señalar que para delimitar comentarios en un programa fuente en SIIL2 se usa el carácter apóstrofe (‘), de forma tal que todo lo que esté a la derecha de éste no es procesado por el compilador. Los comentarios finalizan con el cambio de renglón implicado en un momento dado.

Los detalles de la sintaxis asociada con los módulos lógicos de este ejemplo o bien el funcionamiento de los mismos, puede consultarse en el capítulo cuatro de esta tesis.

6.4 REPORTE DE ERRORES

En esta sección se muestra un programa que contiene algunos errores de sintaxis, con la finalidad de ilustrar como el SWMANPLM3 muestra éstos al usuario.

El ejemplo aquí utilizado corresponde a un módulo lógico de una compuerta AND de dos entradas como la mostrada en la figura 6.9.

Figura 6.9 Módulo lógico de una compuerta AND de dos entradas.

En la figura 6.10 se muestra el archivo and2.SIL, que corresponde a una compuerta AND descrita en el párrafo anterior, el cual se puede compilar y ejecutar con el SWMANPLM3.

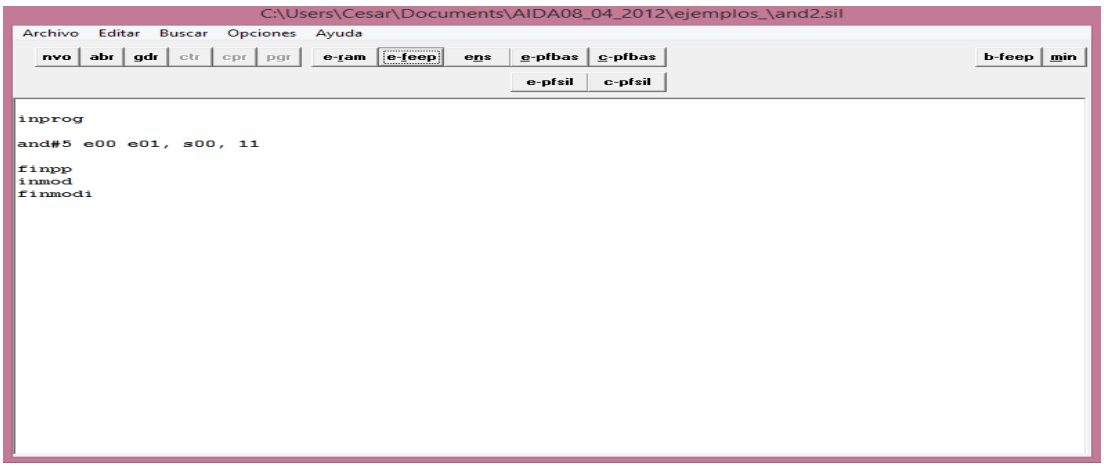


Figura 6.10 Programa fuente con errores de sintaxis.

Durante el proceso de análisis de un programa fuente, el compilador puede detectar errores en la declaración de uno o más módulos lógicos. Cuando esta situación ocurre, el sistema presenta

al usuario una ventana en la cual se indican los errores de sintaxis que se encontraron dentro del programa fuente, dicha ventana se ilustra en la figura 6.11.

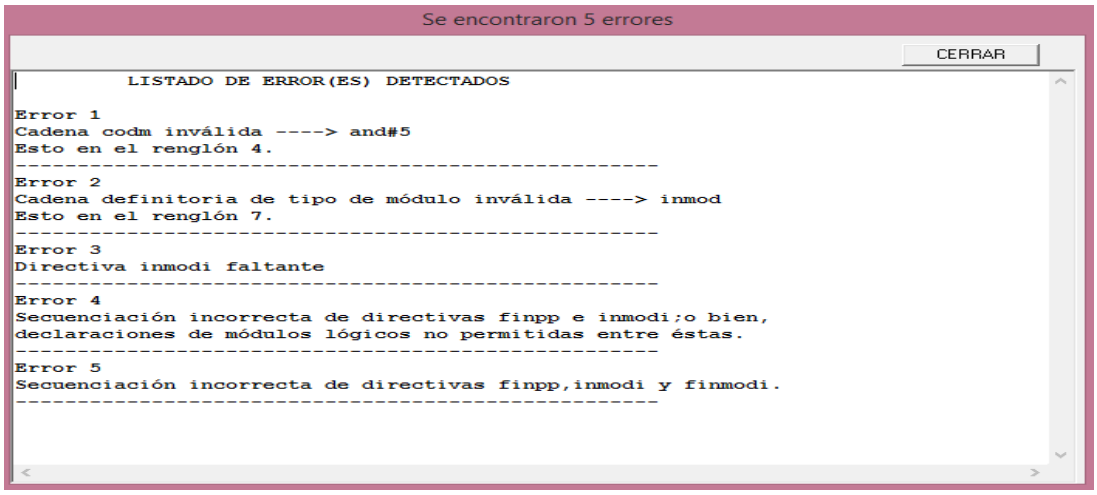


Figura 6.11 Lista de errores de un programa fuente compilado con el SWMSNPLM3.

Como se observa, esta ventana contiene una lista con los errores encontrados por el Software Manejador del PLM3, donde se muestra a detalle:

- El número de error
- El tipo de error.
- El número de renglón donde se encontró el error.
- La descripción del error.

En la parte superior derecha de la venta de errores, se ubica la opción CERRAR, el usuario debe oprimir este botón para regresar a la venta del editor y corregir los errores encontrados.

Después de que el usuario ha hecho los cambios necesarios en el código, se debe guardar de nuevo el programa para volver a compilarse. Una vez que se ha revisado el código fuente y el compilador no encontró más errores aparece un cuadro de diálogo como el de la figura 6.12, donde se le informa al usuario que la compilación ha sido exitosa. Cuando el usuario oprime el botón "Aceptar", el programa compilado está listo para ser cargado y ejecutado en el PLM3.

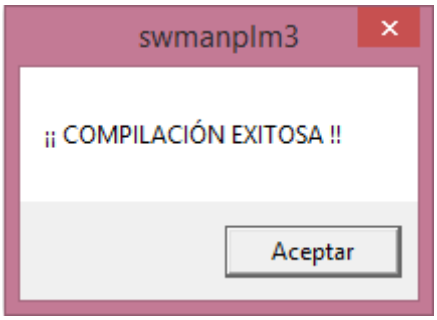


Figura 6.12 Cuadro de diálogo mostrado al finalizar la compilación de un programa fuente sin errores.

6.5 TESTIFICACIÓN DE DISPONIBILIDAD DE LA COMPUTADORA CENTRAL DEL PLM3 PARA RECIBIR COMANDOS DESDE EL SWMANPLM3

Para que el PLM3 pueda procesar los comandos que son enviados desde el SWMANPLM3, se requiere que la Computadora Central esté ejecutando el software receptor de comandos, el cual es parte del firmware de base del MCU presente en la tarjeta MINICON08_GT. Esto es testificado con el continuo encendido y apagado de un led presente en la tarjeta; esto se hace a una cadencia visible para el usuario. A este led lo denominamos "Led PUMMA".

Mientras el "Led PUMMA esté parpadeando podemos ejecutar comandos desde el SWMANPLM3 en el PLM3 tales como; entre otros:

- Borrar la memoria no volátil.
- Cargar y ejecutar un programa en SIIL2.

Cabe señalar que cuando se está ejecutando un programa en SIIL2 en la Computadora Central del PLM3, fundamentalmente es como un lazo que se ejecuta repetitivamente, esta acción detiene la ejecución del receptor de comandos testificándose esto con el hecho de que el "Led PUMMA" se encuentre apagado, lo que indica al usuario que no puede efectuarse acción alguna sobre el PLM3 desde el SWMANPLM3.

Para regresar al receptor de comandos y poder así realizar acciones sobre el PLM3 desde el SWMANPLM3, bastará con oprimir y soltar el botón de Reset del dispositivo. Si el usuario no oprime el botón de Reset, se mostrará un cuadro como el que se ilustra en la figura 6.13, indicando que no se puede realizar acción alguna sobre el PLM3.

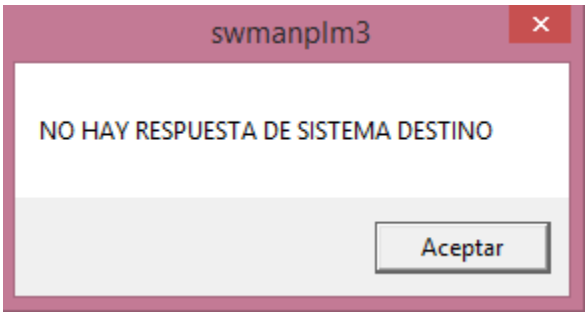


Figura 6.13 Cuadro de diálogo indicando que no se pueden realizar acciones sobre el PLM3 desde el SWMANPLM3.

Para solucionar este error, el usuario debe oprimir "Aceptar" y posteriormente oprimir el botón Reset del dispositivo.

6.6 BORRADO DE LA MEMORIA NO VOLÁTIL

Como ya se ha mencionado en párrafos anteriores, cada vez que se realice una compilación y ejecución de un programa con el SWMANPLM3, previo a ésta, el usuario debe oprimir el botón ***b-feep*** para borrar la memoria del último programa cargado en el PLM3. En caso de que no se haya realizado esta acción, el programa presente en la ventana de edición no se cargará en la Computadora Central del PLM3 y se mostrarán cuadros de diálogo como el de la figura 6.14, indicando que la memoria del último programa cargado en el PLM3 no ha sido borrada.

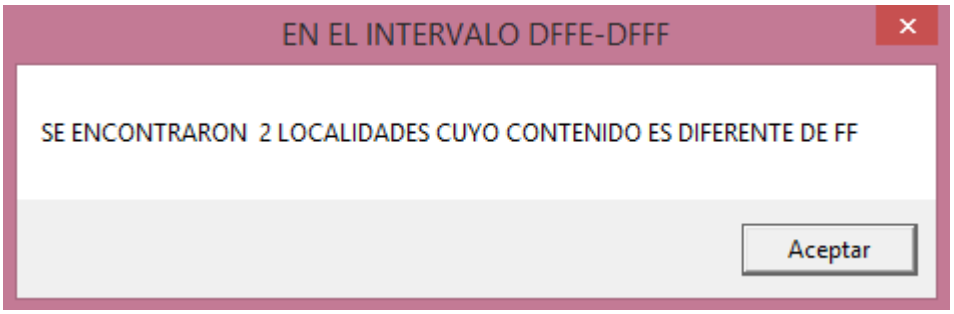


Figura 6.14 Error cuando no se ha borrado la memoria del último programa cargado en el PLM3.

Para dar solución a este tipo de error, el usuario debe oprimir Aceptar e inmediatamente después, oprimir el botón ***b-feep*** y finalmente volver a compilar y ejecutar el nuevo programa.

6.7 EJECUCIÓN DE UN PROGRAMA FUENTE CON EL SWMANPLM3

Después de que un programa fuente ha sido compilado exitosamente con el Software Manejador del PLM3, el usuario debe oprimir el botón correspondiente para ejecutar dicho programa. Cabe señalar, que el botón aquí descrito puede realizar la función de compilar y ejecutar de forma inmediata el programa en proceso como ya se ha descrito en la sección 6.2.2 de este capítulo.

Una vez que se ha oprimido el botón ***e-pfsil*** (compilar y ejecutar), el programa se ha ejecutado exitosamente quedando la ventana de edición desplegada en pantalla como se observa en la figura 6.15 con el código del programa ejecutado y cargado en el PLM3.

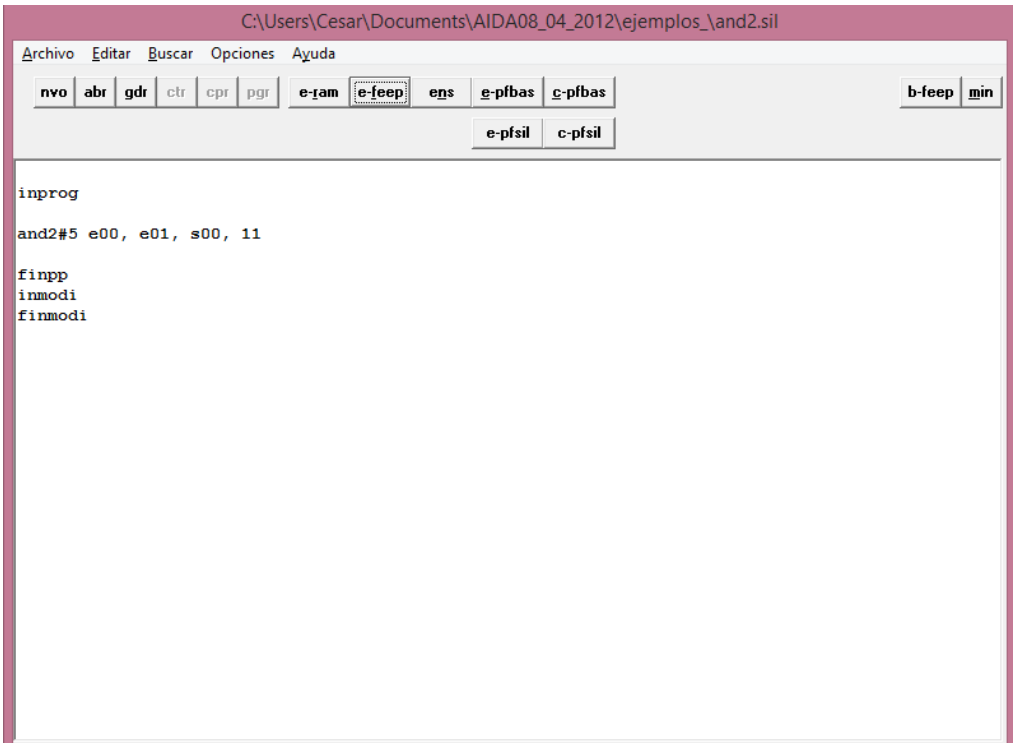


Figura 6.15 Ejemplo de un programa en SIIL2 correspondiente a una compuerta AND de dos entradas sin errores de sintaxis.

Para que el usuario tenga de una forma mucho más clara el orden para cargar y ejecutar correctamente un nuevo programa en el PLM3, se muestran a continuación una serie de pasos a seguir:

1. Capturar el código de un archivo nuevo en la ventana de edición del SWMANPLM3.
2. Guardar dicho código con extensión .SIL.
3. Oprimir el botón de reset de la Computadora Central del PLM3.
4. Oprimir el botón ***b-feep*** para borrado de la memoria del último programa ejecutado. en el PLM3.
5. Oprimir el botón ***c-pfsil*** para compilación del nuevo programa creado.
6. Oprimir el botón ***e-pfsil*** para ejecución del programa compilado exitosamente.
7. Fin.

6.8 EJECUCIÓN AUTÓNOMA DE UN PROGRAMA EN EL PLM3

El PLM3 tiene la capacidad de ejecutar en modo autónomo el último programa que se haya cargado. Para activar este modo de funcionamiento, el usuario debe accionar un interruptor de 1 polo 2 tiros presente en el panel frontal del PLM3, finalmente, el interruptor debe quedar puesto en la opción denominada "AUTÓNOMO".

Cuando se tiene activado el modo AUTÓNOMO, cada vez que el usuario oprima el botón de Reset del PLM3, el último programa cargado en éste, volverá a ejecutarse. Esta funcionalidad está basada en hardware propio para ejecución autónoma, presente en la tarjeta MINICON_08GT.

Si el usuario quiere cargar un nuevo programa en el PLM3, deberá accionar el interruptor en la posición "MONITOR" y conectar el puerto serie para finalmente volver a realizar los siete pasos descritos anteriormente.

Para finalizar este capítulo se presentan dos ejemplos de programas realizados con el SWMANPLM3; esto se hace con la finalidad de que el usuario vaya conociendo la declaración de los módulos lógicos, se familiarice con el entorno de programación y sea capaz de desarrollar aplicaciones para dar soluciones acordes a problemas de control lógico.

Ejemplo 6.1

Se desea realizar con el SWMANPLM3 un programa que accione dos Contadores de Eventos (CONTEV) por medio de una Compuerta OR de dos entradas, donde el usuario tendrá el control de ésta. Los incrementos de cada contador se realizarán por flancos de subida. Cuando se realicen 5 cuentas la entrada S de un flip-flop R-S Asíncrono (FFARS) será activada por nivel alto. La salida Q inicial de este flip-flop estará en nivel bajo, además, la entrada R de éste también deberá ser activada por nivel bajo. Cada vez que el usuario realice 5 cuentas, la salida Q del flip-flop cambiará de estado. El diagrama del ejemplo 6.1 se ilustra en la figura 6.16.

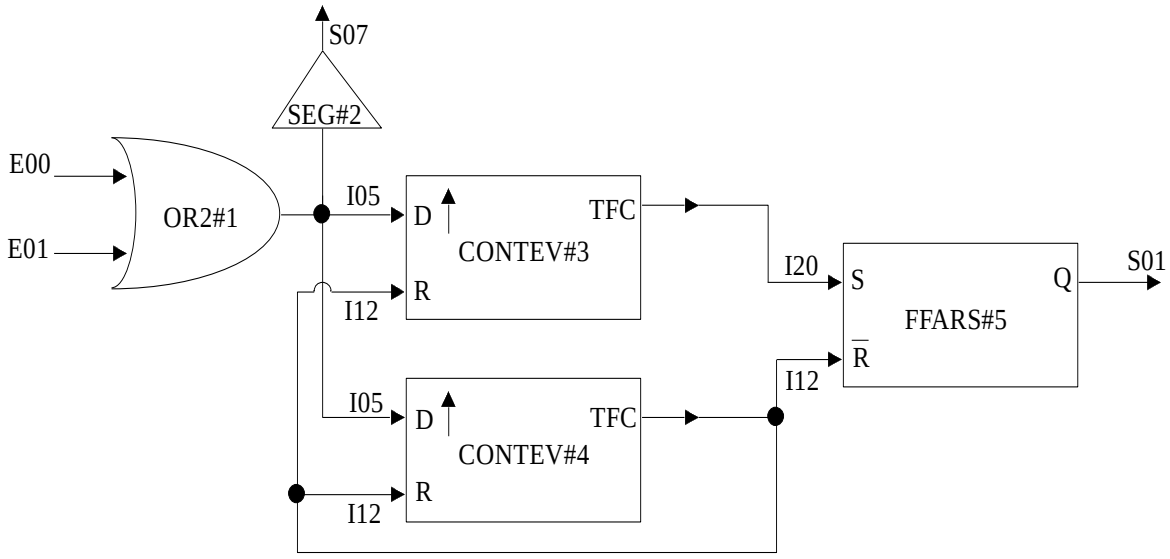


Figura 6.16 Diagrama de módulos lógicos interconectados correspondiente al ejemplo 6.1.

A continuación se muestran las entradas y salidas asignadas para cada módulo de este ejemplo:

Las entradas de la compuerta OR son las VB E00 y E01 y su salida la I05, a este módulo se le asignó el número 2. Las entradas D y R de los contadores son las VB I05, I12 respectivamente. La salida testigo de final de carrera (TFC) del contador 3 es la VB I20 y la del contador 4 la I12. Finalmente, para el flip-flop se tienen asignadas la entradas S y R las VB I20, I12 respectivamente y la salida Q es la S01, este módulo tiene asignado el número 5. Se utilizó una señal testigo para observar el número de cuentas que el usuario va realizando.

En la figura 6.17 se puede observar la ventana de edición del SWMANPLM3 que contiene el código capturado correspondiente el ejemplo 6.1.

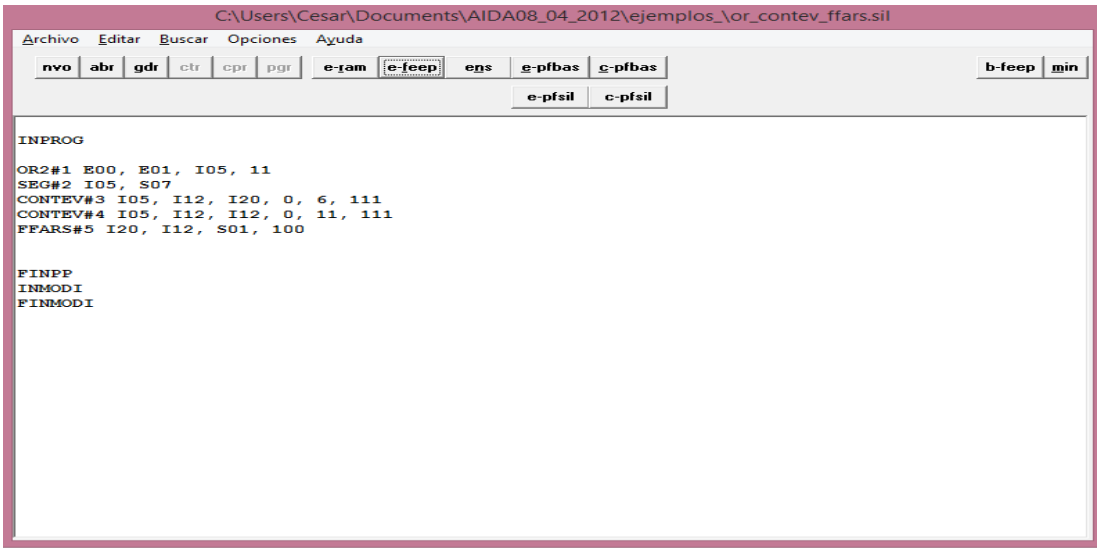


Figura 6.17 Código fuente en SIIL2 para el sistema lógico del ejemplo 6.1, capturado en la ventana de edición del SWMSNPLM3.

Ejemplo 6.2

Se requiere activar un Temporizador Monodisparo (one-shot) disparable por flancos de subida, este mismo será accionado por un Contador de Eventos activado también por flancos de subida. Cuando el contador detecte 5 cuentas, mandará la señal de disparo al temporizador mismo que quedará activado durante dos segundos, inmediatamente después se apagará mandando una señal de reset para volver a empezar una nueva cuenta. Para activar el contador de eventos, se ha utilizado un Temporizador Astable habilitado por nivel alto, con pulsos de un segundo y un ciclo de trabajo de 50%. El programa está condicionado para que el usuario active el temporizador.

Para la señal de habilitación "H" del Temporizador Astable se ha asignado la VB E00, la salida de éste, misma que activará el contador de eventos tiene asignada la VB I00. El testigo de final de carrera del contador de eventos es la VB I01, que es la señal que activa al temporizador monodisparo. La señal de salida del temporizador monodisparo es la VB I02, esta misma es la que activa la entrada de Reset del contador de eventos. En la figura 6.18 se observa el diagrama correspondiente al ejemplo descrito. Se han utilizado módulos lógicos de tipo Seguidor como salidas testigo para observar de forma clara el funcionamiento de este ejemplo.

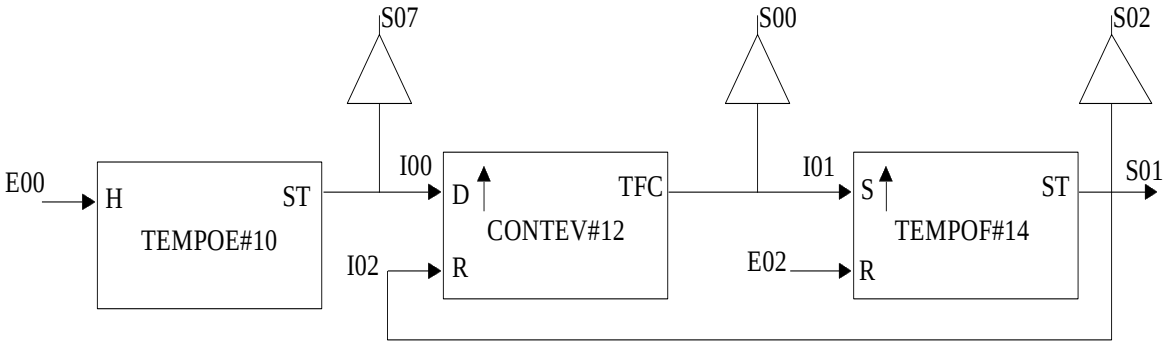


Figura 6.18 Diagrama del ejemplo 6.2.

Después de capturar el programa en la ventana de edición del SWMNSPLM3 y compilarlo sin errores, se ilustra en la figura 6.19 el código ejecutado.

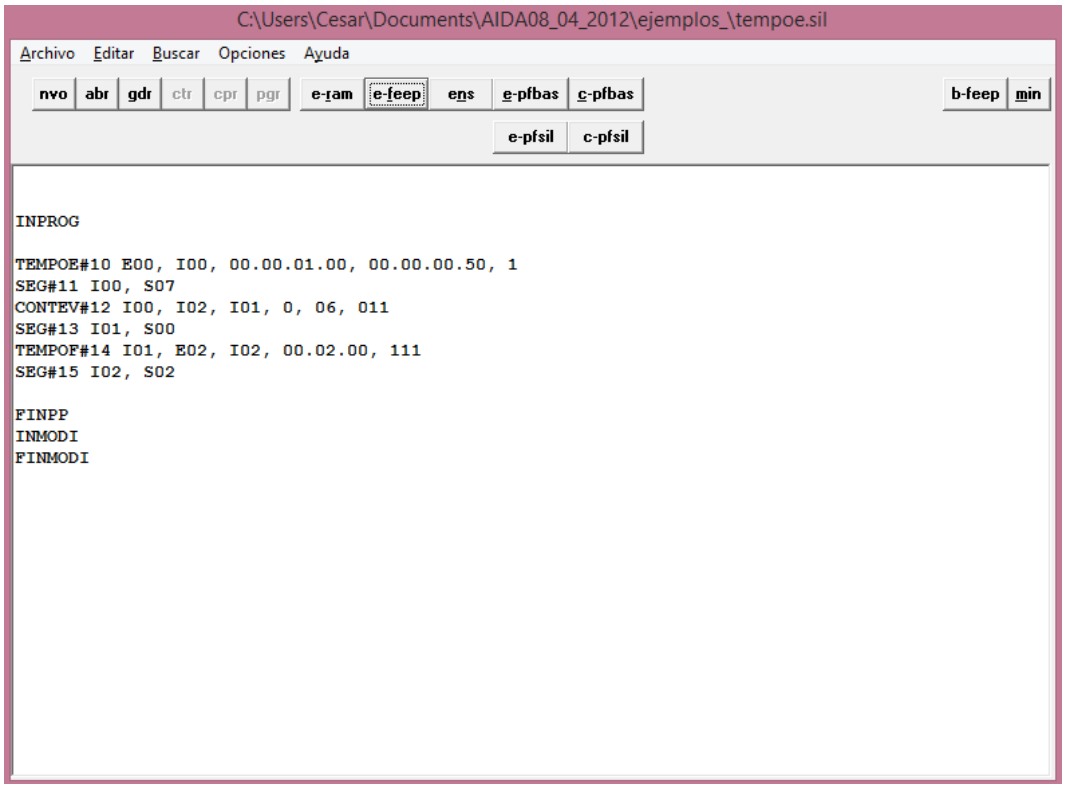


Figura 6.19 Código correspondiente al ejemplo 6.2.

CAPÍTULO 7

EJEMPLO DE APLICACIÓN

En el presente capítulo, se muestra un ejemplo de aplicación utilizando el PLM3. Hemos visto en el desarrollo de esta tesis, que el objetivo principal del dispositivo es ejecutar la automatización de la aplicación deseada por el usuario, mostrando así, el potencial que se puede conseguir en diferentes procesos o subprocesos de control lógico.

Se comenzará por describir a detalle el proceso por controlar, posteriormente se mostrará un esquema en donde se podrán observar los sensores y actuadores involucrados en este ejemplo, además, se señalarán también las variables del PLM3 asociadas en cada caso. De esta forma se procederá con la descripción de los módulos lógicos empleados para la automatización y finalmente se presentará el programa en SIIL2 que debe ejecutar el PLM3 para la realización de la aplicación.

7.1 DESCRIPCIÓN DEL PROCESO POR AUTOMATIZAR EMPLEANDO EL PLM3.

En una determinada planta industrial dedicada a la fabricación de cierto tipo de bebidas, se tiene un proceso general en donde es necesario el arranque de un motor trifásico para que la máquina siempre esté en funcionamiento. Para fines de visualización en el ejemplo, se plantea alcanzar la velocidad nominal del motor en 20 segundos. El operador desea que el arranque del mismo sea por la configuración de tensión reducida. En la figura 7.1, se muestra el conexionado correspondiente.

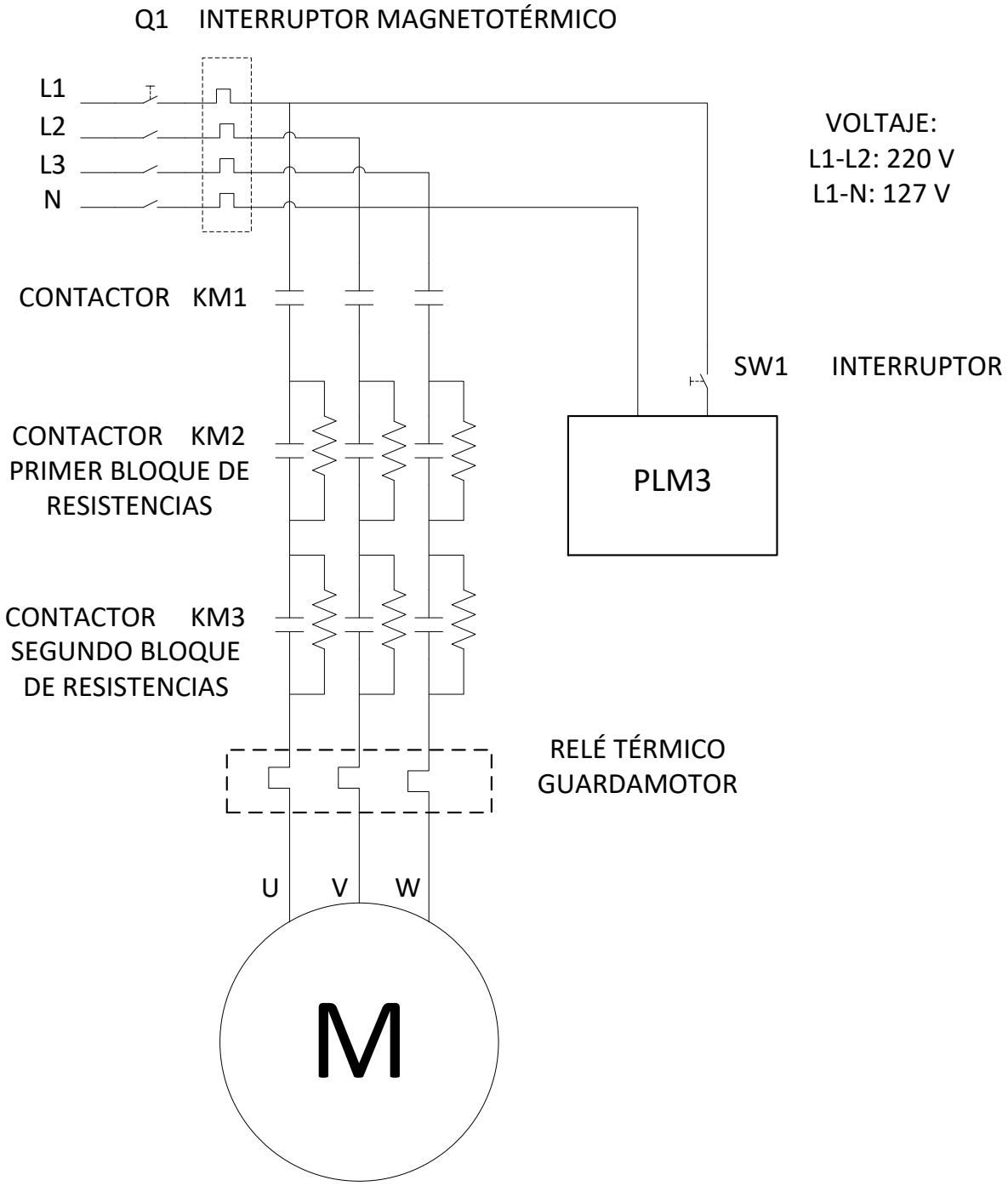


Figura 7.1 Conexión de la configuración de tensión reducida para arranque de un motor trifásico.

En la tabla 7.1 se muestra la configuración deseada para cada paso de activación de los contactores denominados como KM1, KM2 y KM3.

Tabla 7.1 Configuración básica de operación Tensión Reducida.

PASO	PRINCIPAL KM1	BLOQUE 1 RESISTENCIAS KM2	BLOQUE 2 RESISTENCIAS KM3
A	Cerrado	abierto	abierto
B	Cerrado	cerrado	abierto
C	Cerrado	cerrado	cerrado

El sistema debe contar con un botón de inicio normalmente abierto y con un botón de paro de emergencia normalmente cerrado, además de un relevador térmico F1 normalmente cerrado el cual abrirá al detectarse una sobrecarga en el motor.

Para la secuencia de arranque del motor, se requiere lo siguiente.

1. El botón de paro de emergencia debe estar enclavado.
2. Se oprime el botón de inicio.
3. Dos segundos después, se cierra el contactor KM1.

   En este punto, los dos bloques de resistencias presentan una caída de tensión.
4. Diez segundos después, se cierra el contactor KM2.

   En este punto, un bloque de resistencias se pone en circuito corto.
5. Diez segundos después, se cierra el contactor KM3.

   En este punto, el segundo bloque de resistencias es puesto en circuito corto.
6. Cuando el botón de paro de emergencia es deshabilitado, un indicador de color rojo realiza una intermitencia de un segundo hasta su habilitación nuevamente.
7. Cuando el contacto del relevador térmico se abre, un indicador de color amarillo realiza una intermitencia de dos segundos hasta su restablecimiento.

Habiendo transcurrido la secuencia anterior, el motor opera con la tensión nominal. Es importante señalar que las condiciones bajo las cuales el motor se encuentra en funcionamiento deben cumplirse, si alguna de ellas no se realiza, el relevador térmico F1 detecta una sobrecarga y ocasionará que los contactores KM1, KM2 y KM3 se abran desconectando las fases del motor del suministro correspondiente, o bien cuando el botón de paro de emergencia se ha deshabilitado.

7.2 DIAGRAMA A BLOQUES DEL EJEMPLO DE APLICACIÓN.

En la figura 7.2, se muestra un posible diagrama de bloques asociado con los módulos lógicos que integran al sistema lógico requerido, apreciándose ahí los módulos lógicos propios del PLM3, que se utilizaron para la realización de este ejemplo de aplicación, denominado “Arrancador Suave por Tensión Reducida”.

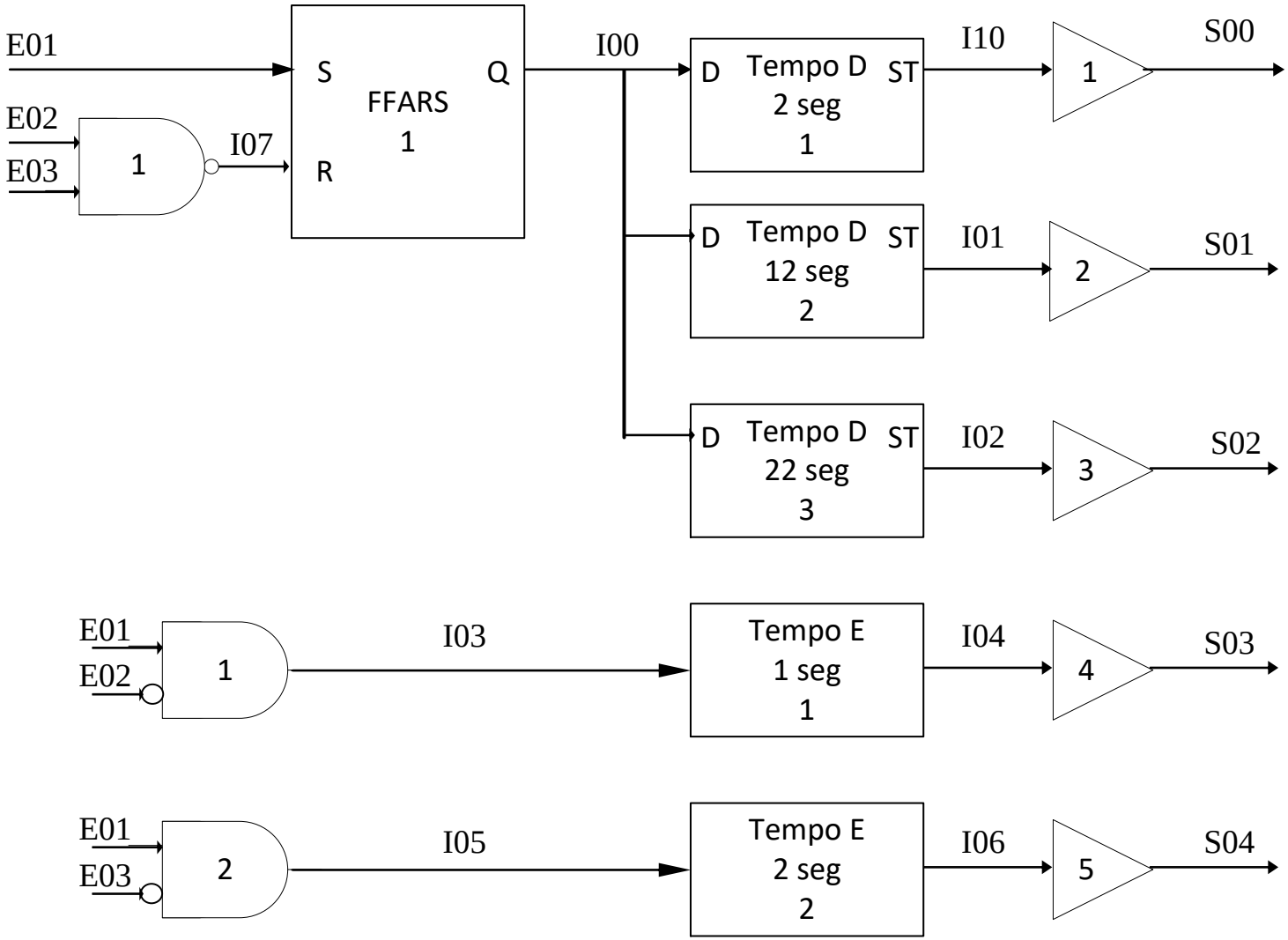


Figura 7.2 Diagrama de bloques asociado al ejemplo de aplicación del arrancador suave por tensión reducida.

7.3 VARIABLES ASOCIADAS AL EJEMPLO DE APLICACIÓN CON EL PLM3.

A continuación se muestran las variables relacionadas con los sensores y actuadores a utilizar en el PLM3.

E01 Contacto normalmente abierto del botón de inicio.

E02 Contacto normalmente cerrado del botón de paro de emergencia.

E03 Contacto normalmente cerrado del relevador térmico.

I00 Variable intermediaria auxiliar salida FFARS#1.

I01 Variable intermediaria auxiliar salida Tempod#2.

I02 Variable intermediaria auxiliar salida Tempod#3.

I03 Variable intermediaria auxiliar salida And#1.

I04 Variable intermediaria auxiliar salida Tempoe#1.

I05 Variable intermediaria auxiliar salida And#2.

I06 Variable intermediaria auxiliar salida Tempoe#2.

I07 Variable intermediaria auxiliar Nand#1.

I10 Variable intermediaria auxiliar salida Tempod #1

S00 Salida para contactor KM1.

S01 Salida para contactor KM2.

S02 Salida para contactor KM3.

S03 Salida de indicación el botón paro de emergencia fue deshabilitado.

S04 Salida de indicación el contacto del relevador térmico se abrió (sobrecarga).

7.4 DESCRIPCIÓN DEL DIAGRAMA A BLOQUES.

El sistema lógico de la figura 7.2 está integrado por los siguientes ML realizables con el PLM3.

- Compuerta NAND de dos entradas, éstas son las VBE E02 y E03. La salida correspondiente es la VBI I07. A este ML se le asigna el número 1 de su tipo.

- Flip-Flop asíncrono RS con nivel inicial bajo a su salida SQ. La entrada S (set) corresponde a las VBE E01 y la entrada R (reset) está asignada a la VBI I07, mientras que la salida de éste es la VBI I00. A este ML se le asigna el número 1 de su tipo.
- Temporizador con retardo a la conexión de dos segundos, su entrada es la VBI I00 y su salida corresponde a la VBI I10. A este ML se le asigna el número 1 de su tipo.
- Seguidor lógico, su entrada corresponde a la VBI I10 y su salida es la VBS S00. Se le asigna el número 1 de su tipo.
- Temporizador con retardo a la conexión de 12 segundos, su entrada es la VBI I00 y su salida corresponde a la VBI I01. A este ML se le asigna el número 2 de su tipo.
- Seguidor lógico, su entrada corresponde a la VBI I01 y su salida es la VBS S01. Se le asigna el número 2 de su tipo.
- Temporizador con retardo a la conexión de 22 segundos, su entrada es la VBI I00 y su salida corresponde a la VBI I02. A este ML se le asigna el número 2 de su tipo.
- Seguidor lógico, su entrada corresponde a la VBI I02 y su salida es la VBS S02. Se le asigna el número 3 de su tipo.
- Compuerta AND de dos entradas, éstas son las VBE E01 y E02, la salida corresponde a la VBI I03. A este ML se le asigna el número 3 de su tipo.
- Temporizador astable, la entrada corresponde a las VBI I03 y su salida es la VBI I04, el tiempo de periodo y de ciclo de trabajo corresponden a 2 y a 1 segundos respectivamente. Se le asigna el número 1 de su tipo.
- Seguidor lógico, su entrada corresponde a la VBI I04 y su salida es la VBS S03. Se le asigna el número 4 de su tipo.
- Compuerta AND de dos entradas, éstas son las VBE E01 y E03, la salida corresponde a la VBI I05. A este ML se le asigna el número 4 de su tipo.
- Temporizador astable, la entrada corresponde a las VBI I05 y su salida es la VBI I06, el tiempo de periodo y de ciclo de trabajo corresponden a 4 y 2 segundos respectivamente. Se le asigna el número 2 de su tipo.
- Seguidor lógico, su entrada corresponde a la VBI I06 y su salida es la VBS S04. Se le asigna el número 5 de su tipo.

Finalmente, a continuación se muestra el programa fuente en SIIL2 para el ejemplo de aplicación mencionado

7.5 PROGRAMA EN SIIL2 DEL EJEMPLO DE APLICACIÓN EN EL PLM3.

El programa realizado en el swmplm3 para el ejemplo del Arrancador Suave por Tensión Reducida, es el siguiente:

```
'Ejemplo de aplicación con el PLM3'
'Arrancador suave por Tensión Reducida'

inprog                        'sentencia delimitadora de inicio del SPP

nand2#1 e02,e03,i07,11              'compuerta nand (1)
ffars#1 e01,i07,i00,111             'FlipFlop RS asincrono (1)
tempod#1 i00,i10,00.00.02.00,1      'Tempod1 de 2 segundos (1)
seg#1 i10,s00                       'seguidor (1)
tempod#2 i00,i01,00.00.12.00,1      'tempod de 12 segundos (2)
seg#2 i01,s01                       'seguidor (2)
tempod#3 i00,i02,00.00.20.00,1      'tempod de 22 segundos (3)
seg#3 i02,s02                       'seguidor (3)

''' Casos de paro'''

and2#1 e01,e02,i03,01                           'compuerta and (1)
tempoe#1 i03,i04,00.00.02.00,00.00.01.00,1      'tempoe (1)
seg#4 i04,s03                                   'seguidor (4)
and2#2 e01,e03,i05,01                           'compuerta and (2)
tempoe#2 i05,i06,00.00.04.00,00.00.02.00,1      'tempoe (2)
seg#5 i06,s04                                   'seguidor 5

finpp                           'sentencia delimitadora del fin del SPP

inmodi                              'inicio del subprograma temporizado
finmodi                             'fin del subprograma temporizado
```

En la figura 7.3, se muestra el conexionado al PLM3 relacionado con los sensores y actuadores implicados en el ejemplo de aplicación realizado, mostrando así la potencialidad del PLM3 para realizar las temporizaciones necesarias por el arrancador.

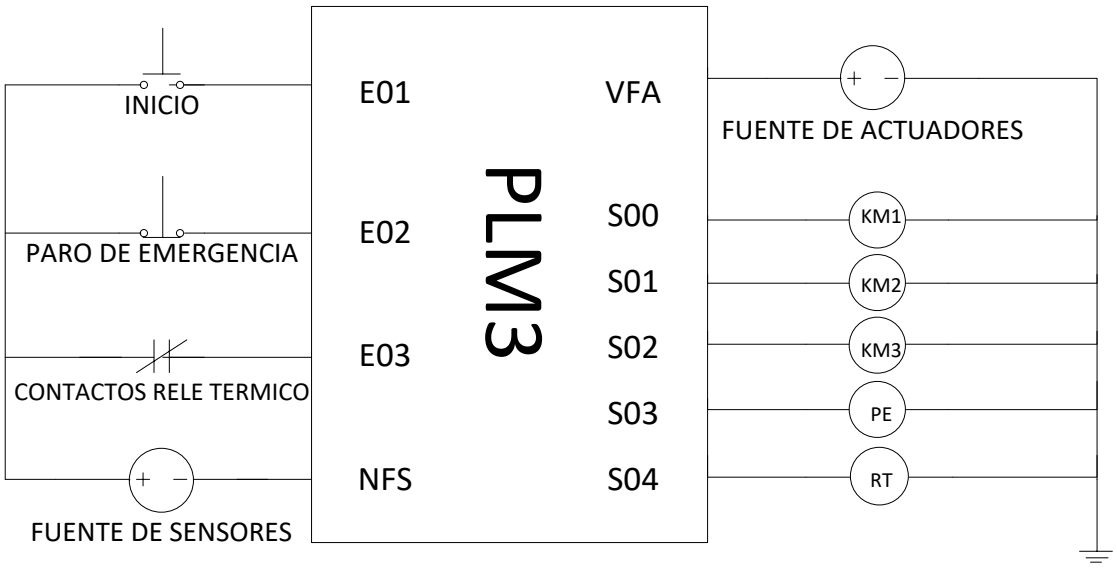


Figura 7.3 Conexionado al PLM3 de los sensores y actuadores utilizados en el ejemplo de aplicación.

CONCLUSIONES

El objetivo del presente trabajo cubre las necesidades mencionadas al principio del mismo, dando como resultado un nuevo prototipo basando su funcionamiento en una nueva plataforma la cual fue desarrollada en el Departamento de Control y Robótica de la Facultad de Ingeniería denominada hasta ahora Tarjeta MINICON_08GT.

El diseño y construcción del Programador Lógico Modular PLM3 descrito en el transcurso de esta tesis, ha sido desarrollado para incrementar nuestros conocimientos y habilidades dentro de los procesos de automatización, buscando de esta forma, nuevas áreas de interés y oportunidad en donde podamos aplicar la Ingeniería en beneficio de las nuevas tecnologías que sean capaces de orientar y dar solución tanto al sector académico como al sector industrial.

Para los fines mencionados, se desarrollaron los bloques de Hardware tanto de entrada como de salida que integran el PLM3 utilizando componentes electrónicos (Circuitos integrados). Con base en el diseño, se logró obtener una eficiente interfaz entre la Tarjeta MINICON_08GT y éstos, además, analizando cada circuito funcional se aplicaron reglas de diseño en el PCB de cada bloque finalizando con un prototipo compacto.

En cuanto al software, se atendió a cada solicitud que se presentase para realizar la actualización necesaria en el primer prototipo (lenguaje SILL1 desarrollado en el Departamento de Control y Robótica de la Facultad de Ingeniería), en donde con la amplia experiencia del tutor de este trabajo se hizo más eficiente la sintaxis para la descripción de cada módulo lógico convirtiéndose en una poderosa herramienta, ya que nos permite encontrar no solo una, sino varias soluciones a problemas de control lógico.

El Software Manejador del PM3 simplifica la programación del dispositivo, facilitando la implementación de diversos códigos ofreciendo al usuario versatilidad y robustez al momento de desarrollar sus proyectos, para ofrecer un nuevo método de programación sin dejar atrás los métodos de programación ya existentes.

En cuanto a mejoras, este trabajo pudo realizarse con componentes electrónicos de montaje superficial en su totalidad, con la finalidad de lograr un prototipo aún más compacto y amigable

para el usuario, o bien, se pudo anexar una tarjeta de entradas analógicas para monitorear y controlar los subprocesos de los sensores y actuadores que así lo requieran. Por otro lado, es indispensable mencionar que el PLM3 pudo haberse realizado con otra plataforma de desarrollo, sin embargo, con la tarjeta MINICON08_GT surgió el proyecto desde la impartición de la materia.

Finalmente, las áreas de conocimiento que adquirimos a lo largo de estos años en nuestra institución (Facultad de Ingeniería de la UNAM) se complementaron y ampliaron al conocer el diseño y principio de funcionamiento de un instrumento de control el cual es fundamental en el 80% del sector industrial.

CONTRIBUCIONES

Recordando el objetivo principal de este trabajo el cual fue realizar un controlador lógico programable utilizando la tarjeta de desarrollo MINICON_08GT, las contribuciones de esta tesis son:

- Se han desarrollado tarjetas de entradas, salidas y fuente de alimentación, haciendo que el prototipo sea compacto y de fácil uso para el usuario.
- Se acondicionaron los puertos del MCU en lenguaje “ensamblador” para poder tener los respectivos grupos de entradas y salidas físicos en el PLM3.

TRABAJO FUTURO

La primera línea de continuación para este trabajo es el desarrollo de una interfaz gráfica de programación donde el usuario pueda identificar visualmente los módulos lógicos tratados en este escrito, como actualmente se interpreta en un diagrama de escalera. Aunando a éste, la posibilidad de anexar una tarjeta que pueda monitorizar y controlar diversas señales analógicas utilizadas en la industria.

BIBLIOGRAFÍA

Salvá Calleja A. Programador Lógico Modular, México D. F., Tesis de Maestría, Facultad de Ingeniería UNAM, 1999.

Salvá Calleja A., Altamirano Yépez L. Dispositivos Chipbas8, microcontroladores HC08 programables en lenguaje BASIC.

Salvá Calleja A., Guzmán Tinajero F., Altamirano Yépez L. PLM08, Controlador Lógico basado en el microcontrolador 68HC908GP32.

Salvá Calleja A. Guía Básica de SIIL2, Departamento de Control y Robótica, Facultad de Ingeniería UNAM, Mayo 2015.

BOYLESTAD Robert L. Electrónica: Teoría de Circuitos y Dispositivos Electrónicos. 10a. edición. México: Prentice Hall, 2009.

Freescale, MC9S08GB60A Microncontrollers. Technical Data.

Vishay Semiconductors CNY74-4H Technical Data.

Texas Instruments ULN2803 Technical Data.

ÍNDICE DE FIGURAS

Figura 2.1 Componentes funcionales del ambiente integrado para la Tarjeta. MINICON_08GT. 5

Figura 2.2 Hardware de la Tarjeta de desarrollo MINICON_08GT. 6

Figura 2.3 Ventana de inicio del Software de la Tarjeta MINICON_08GT. 7

Figura 2.4 Ventana principal del editor del PUMMA_08+. 8

Figura 3.1 Diagrama simplificado a bloques del PLM3. 11

Figura 3.2 Sistema lógico realizable con el PLM3 donde se aprecia el uso de diversos tipos de variables booleanas. 15

Figura 3.3 Representación de un módulo lógico de *m* entradas y *n* salidas. 15

Figura 3.4 Representación de un Módulo Lógico que realiza una compuerta NAND de tres entradas con negación en una de ellas. 16

Figura 3.5 Representación en forma de bloque de un temporizador monodisparo (one-shot), la flecha hacia abajo indica que el disparo es por flanco de bajada. 17

Figura 3.6 Diagrama de tiempos correspondiente a un temporizador monodisparo (one shot). 17

Figura 3.7 Ejecución en la CC del PLM3, de los subprogramas que integran un programa SIIL2. 20

Figura 3.8 Esquema a bloques de una posible aplicación de control lógico, empleando el PLM3. 22

Figura 4.1 Representación genérica del ML seguidor. 26

Figura 4.2 Bloque de un ML Seguidor cuya entrada es la E13 y salida I32. 27

Figura 4.3 Representación genérica del ML Inversor. 27

Figura 4.4 Ejemplo del ML Inversor con entrada E06 y salida S02. 28

Figura 4.5 Representación genérica de un ML de dos entradas realizable por el PLM3. 30

Figura 4.6 Ejemplo de compuerta NAND de dos entradas con preinversión en la entrada E1. 31

Figura 4.7 Ejemplo de compuerta NOR de dos entradas sin preinversión. 31

Figura 4.8 Representación genérica del ML de tres entradas realizable por el PLM3. 32

Figura 4.9 Compuerta AND de tres entradas con preinversión en las entradas E0 y E2. 34

Figura 4.10 Compuerta EOR de tres entradas sin preinversión. 34

Figura 4.11 Representación genérica del ML de cuatro entradas realizable por el PLM3. 35

Figura 4.12 Compuerta AND de cuatro entradas con preinversión en las entradas E1, E2 y E3. 37

Figura 4.13 Compuerta EORN de cuatro entradas sin preinversión. 38

Figura 4.14 Representación genérica de un ML que realiza la función de un Flip-Flop Asíncrono R-S. 38

Figura 4.15 Ejemplo de un ML tipo latch realizable con el PLM3. 40

Figura 4.16 Representación genérica de un ML temporizador (one shot) disparable por nivel. 40

Figura 4.17 Diagrama de tiempos de un temporizador monodisparo redisparable. 41

Figura 4.18 Representación, a) del bloque del temporizador one shot descrito en el ejemplo 4.10, b) diagrama de tiempos del temporizador one shot del mismo ejemplo. 43

Figura 4.19 Representación general de un temporizador con retardo a la conexión (on-delay) o retardo a la desconexión (off-delay). 44

Figura 4.20 Representación del diagrama de tiempos de un temporizador con capacidad de retardo a la conexión (a) y a la desconexión (b). 44

Figura 4.21 Representación a manera de bloques y diagrama de tiempos de los dos temporizadores del ejemplo 4.11. 47

Figura 4.22 Representación genérica de un ML Temporizador astable. 48

Figura 4.23 Representación del diagrama de tiempos de un temporizador astable. 48

Figura 4.24 Representación a manera de bloque del temporizador del ejemplo 4.12. 50

Figura 4.25 Diagrama de tiempos del temporizador del ejemplo 4.12. 50

Figura 4.26 Representación genérica de un ML temporizador (one shot) disparable por flanco. 51

Figura 4.27 Diagrama de tiempos de un temporizador monodisparo redisparable. 52

Figura 4.28 Representación en bloque y diagrama de tiempos del temporizador one shot del ejemplo 4.13. 54

Figura 4.29 Representación genérica de un ML contador de eventos realizable en el PLM3. 55

Figura 4.30 Representación en bloque de un módulo contador de eventos del ejemplo 4.14. La cuenta va ir en ascenso cada vez que se detecte un flanco de subida. 57

Figura 4.31 Diagrama de tiempos asociados al contador de eventos del ejemplo 4.14. 57

Figura 4.32 Representación genérica de un ML secuenciador de estados realizable en el PLM3. 58

Figura 4.33 Representación en bloques de un ML secuenciador de tres bits por tres estados La habilitación comienza cada vez que se detecte un flanco de subida. 62

Figura 4.34 Representación en bloques de un ML secuenciador de tres bits por tres estados La habilitación comienza cada vez que se detecte un flanco de subida habilitada por el usuario. 64

Figura 4.35 Representación genérica de un ML de comparación realizable en el PLM3. 65

Figura 4.36 Representación en bloques de los comparadores del ejemplo 4.17. 66

Figura 5.1 Microcontrolador MC9S9S08GT encapsulado DIP. 67

Figura 5.2 Diagrama de bloques del MCU MC9S9S08GT. 68

Figura 5.3 Representación genérica de la función del circuito optoacoplador empleado para una entrada física del PLM3. 69

Figura 5.4 Circuito de entrada del PLM3 donde se observa que las uniones entre los bornes no se encuentran conectados (circuito abierto). 70

Figura 5.5 Circuito de optoacoplamiento utilizando un sensor a la entrada con fuente propia de alimentación. 71

Figura 5.6 Circuito de optoacoplamiento utilizando un contacto seco. 72

Figura 5.7 Diseño del esquemático de la tarjeta de entradas del PLM3. 73

Figura 5.8 Foto de la tarjeta que muestra las 16 entradas del PLM3 (E00-E17). 74

Figura 5.9 Circuitería de un driver de la interfaz para la tarjeta de salidas del PLM3. 74

Figura 5.10 Diseño del esquemático de la tarjeta de salidas del PLM3. 76

Figura 5.11 Foto de la tarjeta que muestra las 8 salidas del PLM3 (S00-S17). 77

Figura 5.12 Diagrama a bloques de las etapas para generar un voltaje regulado. 78

Figura 5.13 Foto de la tarjeta que muestra las fuentes de alimentación del PLM3. 78

Figura 5.14 Foto del primer prototipo del PLM3. 79

Figura 6.1 MCU de la Computadora Central del PLM3. 80

Figura 6.2 Configuración del puerto serie. 81

Figura 6.3 Ventana de edición del SWMANPLM3 que contiene el código de un programa fuente. 82

Figura 6.4 Botón de la ventana del editor del SWMANPLM3 que compila un programa fuente en SIIL2. 83

Figura 6.5 Botón de la ventana del editor del SWMANPLM3 que compila y ejecuta un programa fuente. 84

Figura 6.6 Botón de la ventana del editor del SWMANPLM3 que borra la memoria donde se guardó el último programa ejecutado. 84

Figura 6.7 Ventana para abrir archivo fuente con extensión .SIL. 85

Figura 6.8 Programa fuente en lenguaje SIIL2. 86

Figura 6.9 Módulo lógico de una compuerta AND de dos entradas. 87

Figura 6.10 Programa fuente con errores de sintaxis. 87

Figura 6.11 Lista de errores de un programa fuente compilado con el SWMSNPLM3. 88

Figura 6.12 Dialogo mostrado al finalizar la compilación de un programa fuente sin errores. 89

Figura 6.13 Cuadro de dialogo indicando que no se pueden realizar acciones sobre el PLM3 desde el SWMANPLM3. 90

Figura 6.14 Error cuando no se ha borrado la memoria del último programa cargado en el PLM3. 91

Figura 6.15 Ejemplo de un programa en SIIL2 correspondiente a una compuerta AND de dos entradas sin errores de sintaxis. 92

Figura 6.16 Diagrama de módulos lógicos interconectados correspondiente al ejemplo 6.1. 94

Figura 6.17 Código fuente en SIIL2 para el sistema lógico del ejemplo 6.1, capturado en la ventana de edición del SWMSNPLM3. 95

Figura 6.18 Diagrama del ejemplo 6.2. 96

Figura 6.19 Código correspondiente al ejemplo 6.2. 96

Figura 7.1 Conexión configuración tensión reducida para arranque de un motor trifásico. 98

Figura 7.2 Diagrama de bloques asociado al ejemplo de aplicación del arrancador suave por tensión reducida. 101

Figura 7.3 Conexionado al PLM3 de los sensores y actuadores utilizados en el ejemplo de aplicación. 105

ÍNDICE DE TABLAS

Tabla 3.1	Declaraciones asociadas con un Módulo Lógico Temporizador.	23
Tabla 4.1	Descripción de la sintaxis de declaración asociada con el ML Seguidor.	26
Tabla 4.2	Descripción de la sintaxis de declaración asociada con el ML Inversor.	28
Tabla 4.3	Descripción de la sintaxis de declaración asociada con el ML de Compuertas de dos entradas.	30
Tabla 4.4	Descripción de la sintaxis de declaración asociada con el ML de Compuertas de tres entradas.	33
Tabla 4.5	Descripción de la sintaxis de declaración asociada con el ML Compuertas de cuatro entradas.	36
Tabla 4.6	Descripción de la sintaxis de declaración asociada con el ML Flip-Flop R-S Asíncrono.	39
Tabla 4.7	Descripción de la sintaxis de declaración asociada con el ML Temporizador (one shot) disparable por nivel.	42
Tabla 4.8	Descripción de la sintaxis de declaración asociada con el ML Temporizador on-delay y off-delay.	45
Tabla 4.9	Descripción de la sintaxis de declaración asociada con el ML Temporizador Astable.	49
Tabla 4.10	Descripción de la sintaxis de declaración asociada con el ML Temporizador Monodisparo (one shot) disparable por flanco.	53
Tabla 4.11	Descripción de la sintaxis de declaración asociada con el ML Contador de Eventos.	56

Tabla 4.12 Descripción de la sintaxis de declaración asociada con el ML Secuenciador de Estados. 59

Tabla 4.13 Descripción de la sintaxis de declaración asociada con el ML Comparador. 65

Tabla 5.1 Líneas de puerto asociadas al hardware de entradas físicas del PLM3. 69

Tabla 5.2 Líneas de puerto asociadas al hardware de salidas físicas del PLM3. 75

Tabla 5.3 Descripción de las funciones de los voltajes de alimentación. 77

Tabla 5.4 Demanda de corriente de cada una de las fuentes de voltaje. 78

Tabla 7.1 Configuración básica de operación Tensión Reducida. 99