90

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

FACULTAD DE INGENIERÍA

DISEÑO DE UN AMBIENTE VISUAL PARA DESARROLLO CON EL PROGRAMADOR LÓGICO MODULAR

T E S I S

Que para obtener el título de :

I N G E N I E R O E N C O M P U T A C I Ó N

P r e s e n t a :

J O R G E R A O S G A R C Í A

ASESOR:
M. I. ANTONIO SALVÁ CALLEJA

MÉXICO, D. F. **JULIO 2002.**

PAGINACIÓN DISCONTINUA

DEDICATORIA

A DIOS:
Por haberme dado la vida y la oportunidad de estar aquí.

A la vida
Por enseñarme que puedo transformarla.

A la Universidad
Por darme las herramientas para hacerlo.

A mis Padres:
Alcira García Pétriz y Rubén Raos Robles.
Por haberme guiado sabiamente hasta la culminación de esta meta , y por dejarme saber que siempre cuento con ellos.

A mis Hermanos:
Rubén y Alejandro
Por el cariño, la compañía, el constante apoyo y por compartir todos los sueños que hemos madurado juntos, deseándoles de todo corazón el mejor futuro.

A mi Abuelita:
La Sra.. Isabel Pétriz Muñoz, en forma muy especial.
Por todo su cariño e invaluable apoyo durante todos estos años.

A todos mis Amigos:
Por.el apoyo y los buenos momentos.

A aquellos Familiares que me ofrecieron apoyo y siempre estuvieron cerca de mi.

A mis Profesores y Sinodales
En especial al M.I. Antonio Salvá Calleja, como reconocimiento a su valioso papel académico y por todo su apoyo a lo largo de la elaboración de este trabajo de tesis.

A la Facultad de Ingeniería y a su Personal Académico
Por brindarnos a muchos la oportunidad de crecer y así poder buscar un mejor futro para nuestro País.

A todos aquellos que contribuyeron para que esto sea posible ahora.

ÍNDICE

INTRODUCCIÓN i

CAPITULO I FUNDAMENTOS DE L0S PLC's

1.1 GENERALIDADES DE LOS PLC's 1
1.2 LA EVOLUCIÓN AL PLC ACTUAL 1
1.3 DEFINICIÓN DE UN CONTROLADOR LÓGICO PROGRAMABLE 4
1.4 DESCRIPCIÓN DEL SISTEMA DEL PLC 5
1.4.1 ESTRUCTURA EXTERNA 6
1.4.2 EL SISTEMA GLOBAL DE UN PLC 7
1.5 NIVEL DE CONOCIMIENTO REQUERIDO PARA LA PROGRAMACIÓN DE UN PLC 8
1.6 CAMPOS DE APLICACIÓN 9
1.7 VENTAJAS Y DESVENTAJAS DEL PLC 10
1.7.1 VENTAJAS 10
1.7.2 DESVENTAJAS 13

CAPITULO II PROCEDIMIENTOS GENERALES DE PROGRAMACIÓN DE UN PLC

2.1 EL EQUIPO DE PROGRAMACIÓN 14
2.2 FORMATOS DE PROGRAMACIÓN 16
2.3 REGISTROS Y ACUMULADORES 16
2.4 TEMPORIZADORES Y CONTADORES 17
2.5 CONSTANTES Y VARIABLES INTERMEDIARIAS 17
2.6 TIPOS DE MÓDULOS 18

2.7 FALLAS OPERACIONALES DEL PLC 19
2.8 ESTRUCTURA DE UN PROGRAMA 19
2.9 LENGUAJES DE PROGRAMACIÓN 20
2.9.1 GRÁFICO SECUENCIAL DE FUNCIONES (SFC o GRAFCET) 21
2.9.2 LISTA DE INSTRUCCIONES (IL o AWL) 22
2.9.3 TEXTO ESTRUCTURADO 23
2.9.4 DIAGRAMA DE CONTACTOS 23
2.9.5 DIAGRAMA O PLANO DE FUNCIONES 24

CAPITULO III ESTRUCTURA Y FUNCIONAMIENTO BÁSICO DEL PROGRAMADOR LÓGICO MODULAR

3.1 ESTRUCTURA BÁSICA DEL PROGRAMADOR LÓGICO MODULAR 25
3.1.1 COMPUTADORA CENTRAL (CC) 28
3.1.2 BLOQUE DE ENTRADAS (BE) 29
3.1.3 BLOQUE DE SALIDAS (BS) 29
3.1.4 BLOQUE DE COMANDO LOCAL Y DESPLIEGUE (BCLD) 30
3.1.5 FUENTE DE ALIMENTACIÓN (FA) 31
3.2 VARIABLES BINARIAS EN EL PLM 31
3.2.1 VARIABLES BINARIAS DE ENTRADA (VBE) 32
3.2.2 VARIABLES BINARIAS DE SALIDA (VBS) 32
3.2.3 VARIABLES BINARIAS INTERMEDIARIAS (VBI) 32
3.3 DESCRIPCIÓN GENERAL DE LOS MÓDULOS LÓGICOS 33
3.4 FORMAS SINTÁCTICAS ASOCIADAS CON LOS MÓDULOS LÓGICOS 35
3.5 FORMATO DE UN PROGRAMA EN SIIL1. 37
3.5.1 CARACTERÍSTICAS GENERALES DE LA EJECUCIÓN DE UN PROGRAMA EL SIIL1 EN LA CC DEL PLM 37
3.5.2 FORMA DE UN PROGRAMA FUENTE EN SIIL1 38

3.6 METODOLOGÍA A SEGUIR PARA ESTRUCTURAR UNA APLICACION DE CONTROL LÓGICO EMPLEANDO EL PLM 40
3.7 ESTRUCTURA DE UN PROGRAMA EN SIIL1 42
3.8 FLUJO DE EJECUCION DEL SUBPROGRAMA PRINCIPAL 42
3.9 FLUJO DE EJECUCIÓN DEL SUBPROGRAMA TEMPORIZADO 44
3.10 RESUMEN 45
3.11 ELECCIÓN DEL LENGUAJE DE PROGRAMACIÓN 47

CAPITULO IV EL LENGUAJE DE PROGRAMACIÓN VISUAL BASIC

4.1 ANTECEDENTES HISTÓRICOS 48
4.2 CARACTERÍSTICAS GENERALES DE VISUAL BASIC 50
4.3 CÓMO FUNCIONA WINDOWS 50
4.4 EVENTOS DE PROGRAMACIÓN 51
4.4.1 OBJETOS Y EVENTOS 52
4.5 MÉTODOS Y PROPIEDADES 53
4.5.1 PROPIEDADES 53
4.5.2 MÉTODOS 55
4.5.3 VENTAJAS DE USAR PROPIEDADES Y MÉTODOS 55
4.6 LAS FORMAS EN VISUAL BASIC 56
4.7 CONTROLES 56

CAPITULO V DESARROLLO DEL SOFTWARE

5.1 ELEMENTOS BÁSICOS 58
5.1.1 MÓDULOS AUXILIARES 60
5.2 GENERALIDADES SOBRE EL SOFTWARE 60

5.3 DESARROLLO DEL SOFTWARE 61
5.3.1 NOMENCLATURA PARA DECLARACIÓN DE ENTRADA(S) Y SALIDA(S) 61
5.3.2 MODULOS FUNCIONALES 62
5.3.2.1 Módulos AND2, AND3 y AND4 62
5.3.2.2 Módulos OR2, OR3 y OR4 65
5.3.2.3 Módulos NAND2, NAND3 y NAND4 68
5.3.2.4 Módulos NOR2, NOR3 y NOR4 70
5.3.2.5 Módulos INV y SEG 73
5.3.2.6 Módulo FFARS 75
5.3.2.7 Módulos TEMPO A, TEMPO C, TEMPO D y TEMPO E 77
5.3.3 GRAFICACIÓN DE LOS MÓDULOS 80
5.3.4 CADENAS BINARIAS 81
5.3.5 CADENA DE IDENTIFICACIÓN DE MÓDULO 83
5.3.6 ESCRITURA Y LECTURA DE ARCHIVOS 87
5.3.6.1 Archivos Gráficos 87
5.3.6.2 Archivo SILL 90

CAPITULO VI EJEMPLOS DE APLICACIÓN

6.1 CÓMO REALIZAR UN DECODIFICADOR 2 A 4 91
6.2 UN EJEMPLO DE APLICACIÓN INDUSTRIAL 100

CONCLUSIONES 104

ANEXO 1

BIBLIOGRAFÍA

INTRODUCCIÓN

El proceso de globalización que se vive actualmente involucra muchos factores de desarrollo tanto políticos, sociales y económicos, todos tendientes a apuntalar una dinámica de desarrollo que permita optimizar procesos en todos los ámbitos que influyen en el crecimiento de un país.

Partiendo de esta premisa y hablando del rubro económico, es claro que todo cambio que pretenda incidir en el crecimiento de un país, debe llevar a cabo una transformación que le permita ser más eficiente. Y es aquí, donde las industrias y sus procesos de producción juegan un papel fundamental y es precisamente en este sentido, en el que la búsqueda de la optimización de los mismos es fuente de creación de nuevas tecnologías que involucran a la vez la utilización de otras tecnologías de punta.

Ante la necesidad de incrementar y mejorar la producción de una industria surge como una posibilidad la iniciativa de automatización del proceso de producción, el cual normalmente se lleva a cabo con un PLC (Pogramable Logic Controller), lo cual resulta ser la utilización de una computadora de uso particular programada para controlar el proceso de producción.

En un principio se podría pensar en un circuito electrónico que sólo fuera capaz de controlar determinado subproceso de producción, o bien, en una computadora que fuera diseñada única y exclusivamente para controlar dicho subproceso y que no fuera capaz de controlar otro, la cual prescindiría de programación alguna o de algún otro tipo de control y bastaría con llevar a cabo la conexión correspondiente de manera correcta.

Si intentáramos avanzar en esto, sería posible pensar que el siguiente paso fuera poder variar los tiempos de ejecución entre cada uno de los procesos internos en los cuales se subdivide el proceso general y que a la vez se pudiera establecer un contador que nos permitiera repetir un subproceso determinado número de veces mientras el siguiente subproceso espera a que las repeticiones de éste terminen para luego entrar en ejecución, de esa manera lo único que tendría que programarse son los tiempos de accionamiento entre cada uno de los subprocesos y el número de repeticiones en algunos de ellos. Bajo este esquema de consideraciones no sería factible pensar que se controle un proceso de esa naturaleza sin hacer uso de otro controlador entre subprocesos y difícilmente podría este mismo dispositivo controlar otro proceso, salvo la remota posibilidad de que determinado proceso de producción tuviera los mismos subprocesos, o con la omisión de algunos de los ya contemplados, para lo cual sería necesario ver la posibilidad de desactivar o desconectar algunos de los circuitos que controlan ciertos subprocesos, lo cual ya ofrece complicaciones adicionales que además no sabemos si serían salvables.

Lo anterior nos lleva a pensar en una computadora de uso particular con la capacidad de controlar procesos industriales y que no sea tan limitada como la

que se acaba de describir, es decir que nos ofrezca algún grado de adaptabilidad, o sea un rango de acciones y opciones más basto, que permitiría modificar o ampliar el proceso sin la necesidad de cambiar de computadora que lo controle o hacer cambios internos en ella, lo cual parece lo más adecuado pero es entonces cuando un factor más entra en cuestión.

El hecho de poder modificar el proceso de producción implica cuando menos, modificar la lógica de control de algún subproceso y hacer esto resultaría simple si pensamos que el dispositivo de control que se está utilizando es como una computadora de escritorio que se puede programar, de otra manera, la siguiente opción más sencilla es tener que llevar a cabo algunas conexiones internas en nuestro dispositivo de control. Aunque si pensamos en la programación como en una PC, nos tendríamos que enfrentar a la necesidad de dominar el lenguaje de programación, para entonces poder programar el dispositivo, tal como se hace con un PLC. Y es de esta manera como surge la necesidad de programar una computadora de uso particular cuyo lenguaje de programación no es común y por ende existen pocas personas especializadas capaces de hacerlo.

En un principio se podría pensar en recibir servicio de la compañía que fabricó el PLC o bien, enviar a un empleado a un curso de capacitación donde aprenda a programarlo, pero las dos opciones nos ofrecen ciertas desventajas.

La primera resulta poco costeable para la empresa y la segunda, implica una inversión inicial que sin duda sería costosa por el hecho de no tener otra instancia que pueda proporcionar dicha capacitación, pero a largo plazo resultaría una opción más rentable para la empresa y hasta ahora, la mejor, solo que ofrece la desventaja de depender de un solo empleado para estas cuestiones lo cual derivaría finalmente en un aumento de importancia del empleado para la empresa y posiblemente en ciertos vicios en la relación Dueño-Empleado y en determinado momento hasta en un incremento de sueldo para el mismo. Pero eso es aún rescatable, quizá la manera de evitarlo, sea la capacitación de más de un empleado, que implica seguramente una inversión más fuerte por parte de la empresa, lo cual sabemos que no es lo más rentable.

Aquí debemos recordar que en cierta forma habríamos solucionado lo relacionado con la programación pero todavía tenemos un PLC limitado a llevar a cabo tareas que de acuerdo a su estructura le sea posible realizar. De esto, surgió pensar en un PLC que tenga la ventaja de trabajar con bloques funcionales y que por medio de ellos fuera capaz de llevar a cabo los procesos de manera virtual.

Dicho dispositivo, fue diseñado en la Facultad de Ingeniería, lo cual aportó la ventaja de tener un PLC que nos diera una gama más amplia de posibilidades y en concreto versatilidad. En este dispositivo en particular, se llaman Módulos a los bloques funcionales del mismo, llegando de esta manera a lo que conocemos como el Programador Lógico Modular y que en esta tesis denominaremos simplemente como PLM, que además nos aporta el beneficio de contar con su

propio lenguaje de programación, que es mucho más sencillo que programar directamente el microprocesador que se esté utilizando en el PLC.

Después de tener el PLM, que nos da mayores posibilidades de operación y que más allá de solo no ser un PLC común, hace patente un concepto importante en cuanto a estos dispositivos, nos encontramos con que en principio manejamos una vez más la programación para el PLM con un código propio y nos enfrentamos otra vez a un problema anteriormente tratado. Nuevamente tenemos que programar y la forma de hacerlo no es algo comercial. Todo esto nos lleva a entender la necesidad de avanzar en este aspecto y establecer una forma de programación mas sencilla que permita que la capacitación de los usuarios no sea tan costosa y que facilite esta labor con el PLM y como consecuencia le dé mayores posibilidades de operación a quien lo utilice.

No debemos olvidar que estamos hablando de un dispositivo particular que no es del dominio público y que lo que se busca con él es ofrecer una nueva alternativa con algunas innovaciones que lo hagan más funcional y que proporcione ventajas sobre lo existente en el mercado común. Aún cuando ya se ha elaborado un código y una estructura de programación, todavía no llegamos a un punto en el cual se utilicen todos los recursos a nuestro alcance que nos permitan simplificar al máximo la utilización de este dispositivo y por tanto se ha pensado en proponer y elaborar una alternativa que nos permita hacer mas sencilla dicha tarea.

En primer término es fácil pensar que el proponer una nueva forma de programación en la cual se utiliza un lenguaje sencillo, tan sencillo que una persona relacionada con estos dispositivos no tuviera problema en familiarizarse con él, daría una solución, pero esto finalmente termina por reducir la programación a lo que siempre ha sido y nos enfrentamos a problemas inherentes a los que todo programador se enfrenta en cualquier lenguaje, como son aprender las instrucciones, tener que cuidar la sintaxis, buscar los errores en caso de equivocarse, etc. Repitiendo de esta manera ciertos esquemas que se tienen desde hace décadas y por consiguiente tendríamos algunas de las carencias que siempre se han tenido.

¿Hacia donde vamos?

La propuesta es crear un "lenguaje universalmente entendible"(dando por hecho que el usuario tiene conocimientos de componentes electrónicos y que conoce el PLM), una forma de programación que nos evite todos esos esquemas y problemas que hemos mencionado, familiarizándonos lo más pronto posible con la programación sin tener que hacer uso de extensos manuales, beneficiándonos con las facilidades que el ambiente gráfico nos brinda, proporcionándonos un medio de trabajo más ameno y de fácil comprensión, así como la utilización de menor tiempo para su aprendizaje y por supuesto reducir el tiempo de elaboración de las tareas que se tengan que realizar mediante programación.

En concreto, crear un ambiente visual de programación para uso con el PLM (que nos permita reducir de manera significativa el tiempo empleado en esta tarea) que genere el código fuente para interactuar de manera directa con el PLM y que además nos permita hacer modificaciones en el archivo ya guardado como un programa, generando un nuevo código fuente.

CAPITULO I

FUNDAMENTOS DE LOS PLC's

Tomando en cuenta que en esta tesis se trabaja con un PLC prototipo, en primera instancia se considera importante definir qué es un Controlador Lógico Programable (PLC), así como tratar la evolución de la lógica de relevadores y sistemas de computadora que dieron paso a éste, para después dar una breve explicación de lo que es el sistema del PLC y algunas ventajas y desventajas de su uso sobre de otros sistemas de control.

1.1 GENERALIDADES DE LOS PLC's

El término PLC (de sus siglas en inglés) de amplia difusión en el medio significa, Controlador Lógico Programable. Originalmente se denominaban PC's (Programmable Controllers), pero con la llegada de las IBM PC's, para evitar confusión, se emplearon definitivamente las siglas PLC.

En Europa, el mismo concepto es llamado Autómata Programable.

Esta familia de aparatos se distingue de otros controladores automáticos, en que puede ser programado para controlar cualquier tipo de máquina, a diferencia de otros muchos que, solamente pueden controlar un tipo específico de aparato. Un programador o Control de Flama de una caldera, es un ejemplo de estos últimos.
Además de poder ser programados, se insiste en el término "Control Automático", que corresponde solamente a los aparatos que comparan ciertas señales provenientes de la máquina controlada de acuerdo con algunas reglas programadas con anterioridad para emitir señales de control para mantener la operación estable de dicha máquina.
Las instrucciones almacenadas en memoria permiten modificaciones así como su monitoreo externo.

1.2 LA EVOLUCIÓN AL PLC ACTUAL

Hasta no hace mucho tiempo el control de procesos industriales se hacía de forma cableada por medio de contactores y relevadores. Al operario que se encontraba a cargo de este tipo de instalaciones, se le exigía tener altos conocimientos técnicos para poder realizarlas y posteriormente mantenerlas.

Además, cualquier variación en el proceso suponía modificar físicamente gran parte de las conexiones de los montajes, siendo necesario para ello un gran esfuerzo técnico y un mayor desembolso económico.

Los primeros sistemas de PLC evolucionaron de las computadoras convencionales en los últimos años de los sesenta y principios de los años setenta. Los PLC's se introdujeron por primera vez en la industria en 1960 aproximadamente. La razón principal de tal hecho fue la necesidad de eliminar el gran costo que se producía al reemplazar el complejo sistema de control basado en relevadores y contactores. Bedford Associates propuso algo denominado Controlador Digital Modular (MODICON, MOdular DIgital CONtroler) a un gran fabricante de coches. Otras compañías propusieron a la vez esquemas basados en computadoras, uno de los cuales estaba basado en el PDP-8. El MODICON 084 resultó ser el primer PLC del mundo en ser producido comercialmente.

El problema de los relevadores era que cuando los requerimientos de producción cambiaban también lo hacía el sistema de control. Esto comenzó a resultar bastante caro cuando los cambios fueron frecuentes. Dado que los relevadores son dispositivos mecánicos y poseen una vida limitada se requería una estricta manutención planificada. Por otra parte, a veces se debían realizar conexiones entre cientos o miles de relevadores, lo que implicaba un enorme esfuerzo de diseño y mantenimiento.

Los "nuevos controladores" debían ser fácilmente programables por ingenieros de planta o personal de mantenimiento. El tiempo de vida debía ser largo y los cambios en el programa tenían que realizarse de forma sencilla. Finalmente se imponía que trabajaran sin problemas en entornos industriales adversos.

Estos primeros PLC's fueron principalmente instalados en las plantas automotrices. Tradicionalmente, las auto-plantas tenían que ser cerradas por más de un mes para cambiar de modelo. Los primeros PLC's fueron usados durante mucho tiempo con otras nuevas técnicas de automatización para acortar el tiempo de cambio. Uno de los procedimientos de cambio de mayor consumo de tiempo había sido la nueva instalación eléctrica o la revisión de los relevadores y paneles de control. El procedimiento de reprogramación del PLC mediante el teclado reemplazó el realambrado de un tablero lleno de cables, relevadores, cronómetros, y otros componentes. Los nuevos PLC's ayudaron a reducir el tiempo de reprogramado a cuestión de unos días.

Pero había un problema mayor con estos procedimientos de programación para las primeras computadoras/PLC a principios de los años setentas. Los programas eran complicados y requerían un programador muy especializado que hiciera los cambios. A través de los últimos años de los setentas, fueron hechas mejoras en los programas de los PLC's para hacerlos algo más amistoso al

usuario; en 1978, la introducción del microprocesador aumentó el poder de la computadora para todos los tipos de sistemas de automatización y disminuyó el costo de los dispositivos de computo. Por consiguiente, la robótica, los dispositivos de automatización, y computadoras de todos los tipos, incluso el PLC, tuvieron muchas mejoras. Los programas de PLC se volvieron más entendibles a más personas. Y los PLC's también fueron más económicos.

A mediados de los setenta las tecnologías dominantes de los PLC's eran máquinas de estado secuenciales y CPU basadas en desplazamiento de bit. Los microprocesadores convencionales daban la potencia necesaria para resolver de forma rápida y completa la lógica de los pequeños PLC's. Por cada modelo de microprocesador había un modelo de PLC basado en el mismo. En los años ochenta se produjo un intento de estandarización de las comunicaciones con el protocolo MAP (Manufacturing Automation Protocol) de General Motor's. También fue un tiempo en el que se redujeron las dimensiones del PLC y se pasó a programar con programación simbólica a través de ordenadores personales en vez de las clásicas terminales de programación.

En los años 8Os, con más poder de la computadora disponible por dólar, el PLC entró en un crecimiento exponencial en su uso.
Algunas compañías de computo y electrónica grandes y algunas divisiones corporativas diversas de la electrónica encontraron que el PLC se habia vuelto su más grande producto. El mercado de los PLC's creció de un volumen de $80 millones en 1978, a $1 billón de dólares por año en 1990 y aún sigue creciendo. Incluso la industria de máquinas y herramientas dónde se usaban las computadoras de control numérico (CNCS), se están usando PLC's. También se usan extensivamente PLC's en la creación de energía y sistemas de control de seguridad. Otros usos no tradicionales de PLC's, como en el hogar y en el equipo médico, aumentarán en los próximos años.

La década de los 90 ha mostrado una gradual reducción en el número de nuevos protocolos, y en la modernización de las capas físicas de los protocolos más populares que sobrevivieron a los 80. El último estándar (IEC 1131-3) intenta unificar el sistema de programación de todos los PLC en un único estándar internacional. Ahora existen PLC's que pueden ser programados en diagramas de bloques, lista de instrucciones y texto estructurado al mismo tiempo.

En la actualidad, no se puede entender un proceso complejo de alto nivel, desarrollado por técnicas cableadas. El ordenador y los PLC's programables han intervenido de forma considerable para que este tipo de instalaciones se haya visto sustituidas por otras controladas de forma programada.

1.3 DEFINICIÓN DE UN CONTROLADOR LÓGICO PROGRAMABLE

Se entiende por controlador lógico programable (PLC), o autómata programable, a toda máquina electrónica diseñada para controlar en tiempo real y en medio industrial procesos secuenciales. Esta definición se está quedando un poco desfasada, ya que han aparecido los micro-plc's, destinados a pequeñas necesidades y al alcance de cualquier persona.

La definición más apropiada es: Sistema Industrial de Control Automático que trabaja bajo una secuencia almacenada en memoria, de instrucciones lógicas. Es un sistema porque contiene todo lo necesario para operar, e industrial por tener todos los registros necesarios para operar en los ambientes hostiles encontrados en la industria.

Pero de cualquier forma y en términos más amplios, un Controlador Lógico Programable es una computadora electrónica de uso fácil, de aplicación industrial desarrollada en torno a un microprocesador. De acuerdo a un programa preestablecido. Es capaz de supervisar y/o controlar diversos tipos de máquinas o de procesos en el ámbito industrial.

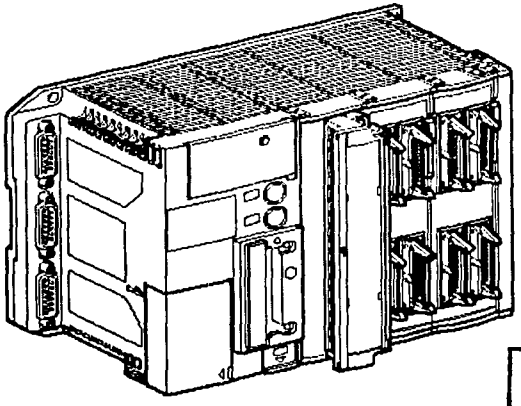


Figura 1.1 Esquema de un PLC, el TSX17-10

El sistema recibe como información de entrada los estados de sensores y/o transductores (de presión, temperatura, velocidad, posición, etc.) que se encuentran convenientemente distribuidos a lo largo del proceso; a partir de esta información, y de acuerdo al programa en ejecución, se generan señales de salida que permiten activar ciertos actuadores como pueden ser: motores, electroválvulas, contadores, relevadores, sistemas neumáticos, oleodinámicos, etc.

Considerado técnicamente, un PLC es como un instrumento de informática industrial, como tal, puede programarse, controlarse, y operarse por una persona inexperta en la operación de computadora. El Controlador Lógico Programable dibuja esencialmente las líneas y dispositivos de diagramas de la escalera. El dibujo resultante en la computadora toma en mucho el lugar del cableado externo requerido para el control de un proceso. De una manera más simple, se puede decir que el Controlador Lógico Programable operará cualquier sistema que tenga dispositivos de salida que enciendan y apaguen. También puede operar cualquier sistema con salidas variables. Está diseñado para realizar todas las funciones de automatización, como lo son, operación, supervisión y regulación.

El PLC puede usarse en muchas aplicaciones industriales, desde aquellas que requieren un simple control ON/OFF, hasta los más exigentes y complejos requerimientos en sistemas secuenciales, de entradas y salidas analógicas, en manipulación de datos y regulación de variables físicas que controlan un proceso.

1.4 DESCRIPCIÓN DEL SISTEMA DEL PLC

Para conocer un PLC es necesario describir los componentes y módulos[1] que lo constituyen.
Un sistema de PLC simple se aloja en uno, o posiblemente dos módulos cada uno de los cuales incluirían múltiples funciones. Un PLC más complejo, controlando un proceso grande, puede tener desde tres a cinco o más módulos separados interconectados que contengan los subsistemas del PLC. Así mismo es importante conocer, al menos en términos generales las interconexiones eléctricas de las varias partes de PLC.

La mayoría de las conexiones eléctricas del PLC hace fácilmente con simples cables entre las unidades. Sin embargo, conectar los módulos de entrada y salida al mundo exterior puede resultar más complicado.
Es importante también, saber que existen computadoras personales y sistemas funcionales de disco disponibles, para llevar a cabo la programación de PLC. Los sistemas de PLC operan en diferentes rangos que la computadora comercial. La proporción normalmente llamada, la proporción del "baudio", depende de qué

[1] No se deben confundir los módulos de un PLC con los módulos o bloques funcionales del PLM.

partes del sistema de PLC se están comunicando. Un PLC puede operar a un baudage de 9600 con el CPU, a 1200 con un grabador de cinta, y a 2400 al trabajar con una impresora.

1.4.1 ESTRUCTURA EXTERNA

Todos los PLC's programables, poseen una de las siguientes estructuras:

- Compacta: en un solo bloque están todos lo elementos.
- Modular:
  - Estructura americana: separa las E/S del resto del PLC.
  - Estructura europea: cada módulo es una función (fuente de alimentación, CPU, E/S, etc.).

Exteriormente se encontrarán cajas que contienen una de estas estructuras, las cuales poseen indicadores y conectores en función del modelo y fabricante. La figura 1.2 muestra la estructura externa de un PLC.

Figura 1.2 Estructura externa de un PLC

1.4.2 EL SISTEMA GLOBAL DE UN PLC

La figura 1.3 muestra en forma de bloques, las cinco unidades más importantes de un sistema de PLC y cómo se interconectan. Cada una de las cinco partes del controlador será descrito después en detalle:

- **La Unidad Central de Proceso (CPU).** El "cerebro" del sistema.

- **Módulo de Programación (Programmer/Monitor o PM).** El programador es el teclado en que las instrucciones del programa se teclean por el usuario. El Monitor es una pantalla como de televisión en la cual se despliega la información que teclea el usuario.

- **Los Módulos de Entrada/Salida (E/S).** El módulo de la entrada tiene terminales en las que el usuario ingresa desde fuera del proceso las señales eléctricas. El módulo del Salida tiene otro conjunto de terminales que envían las señales de acción al proceso. Puede agregarse un sistema electrónico remoto para conectar módulos E/S a lugares remotos si es necesario. El proceso de operación actual bajo el control del PLC puede estar a miles de metros del CPU y sus módulos de E/S.

- **La Impresora**. Un dispositivo en que el programa en la Unidad Central de Proceso puede ser impreso. Adicionalmente, la información de operación puede imprimirse.

- **El Programa Recorder/Player**. Algunos sistemas de PLC viejos usan los dispositivos de cinta; otros usan los sistemas del disco flexible, dispositivos que externamente pueden grabar los programas en el CPU. Los PLC's más nuevos usan los discos duros por programar y grabar. Los programas grabados que se guardan pueden ser reintroducidos después en el CPU si el programa original está perdido o desarrolla un error.

Para las operaciones grandes, otra posible opción es la conexión del CPU a una computadora Maestra. La computadora Maestra puede usarse en una fábrica grande o en sistemas de procesos para coordinar muchos sistemas de PLC individuales. Los buses eléctricos interconectados, son algo llamado carreteras de información.

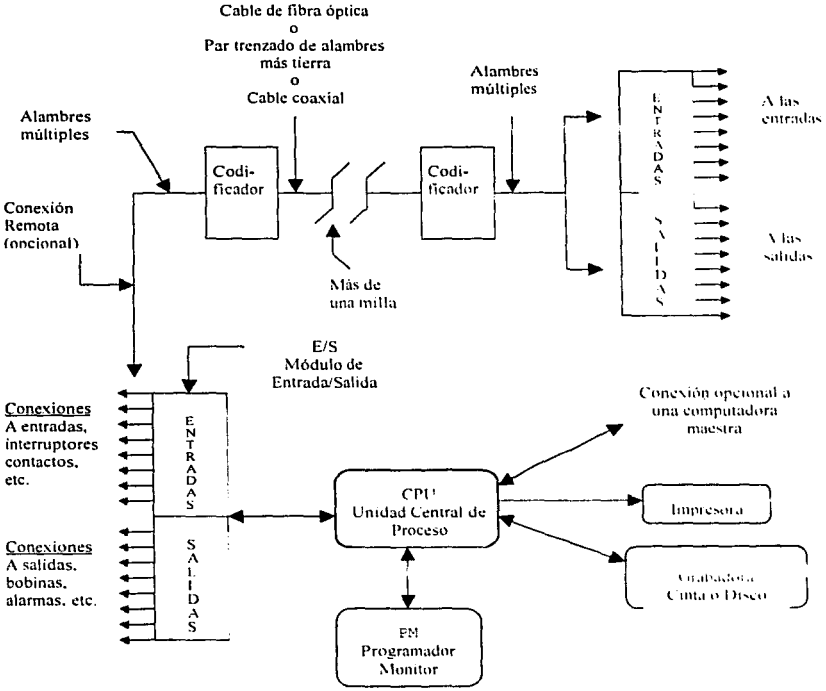


Figura 1.3 Las cinco unidades más importantes de un sistema de PLC y su interconexión

1.5 NIVEL DE CONOCIMIENTO REQUERIDO PARA LA PROGRAMACIÓN DE UN PLC

La implementación de un sistema PLC requiere de hardware y software. El hardware (parte física del sistema), lo constituyen los diferentes circuitos electrónicos de control, procesamiento e interfase con el usuario y el proceso. El software (programas), es el conjunto de instrucciones y de datos que determinan el funcionamiento del sistema.

Una persona conocedora de los sistemas de lógica de relevador puede dominar la mayoría de las funciones de un PLC en pocas horas. Estas funciones podrían incluir bobinas, contactos, cronómetros. y contadores. Lo mismo es para una persona con un respaldo de lógica digital. Para las personas poco familiarizadas con diagramas de escalera o principios digitales, sin embargo, el proceso de aprendizaje toma más tiempo.

Una persona que conoce la lógica de relevador puede dominar las funciones avanzadas del PLC en pocos días con la instrucción apropiada. Las escuelas de la compañía y los manuales de operación son muy útiles en el aprendizaje de estas funciones avanzadas. A fin de aprender las funciones avanzadas podrían incluir controlador secuenciador/tambor, uso de registro de bits, y funciones del movimiento.

1.6 CAMPOS DE APLICACIÓN

Un PLC suele emplearse en procesos industriales que tengan una o varias de las siguientes necesidades:

- Espacio reducido.
- Procesos de producción periódicamente cambiantes.
- Procesos secuenciales.
- Maquinaria de procesos variables.
- Instalaciones de procesos complejos y amplios.
- Chequeo de programación centralizada de las partes del proceso.

Aplicaciones generales:

- Maniobra de máquinas.
- Maniobra de instalaciones.
- Señalización y control.

Tal y como se indicó anteriormente, esto se refiere a los PLC's, dejando de lado los pequeños PLC's para uso más personal (que se pueden emplear, incluso, para automatizar procesos en el hogar, como la puerta de una cochera o las luces de la casa).

1.7 VENTAJAS Y DESVENTAJAS DEL PLC

1.7.1 VENTAJAS

Las siguientes son algunas de las mayores ventajas de usar a un Controlador Programable:

Previsto para operar en tiempo real. El tiempo de ejecución de una instrucción está en el orden de los microsegundos, lo que garantiza una recolección casi instantánea de las señales de entrada y un procesamiento en tiempo real de las variables de control.

Flexibilidad. En el pasado, cada máquina de la producción electrónicamente controlada requería su propio controlador; 15 máquinas podrían requerir 15 controladores diferentes. Ahora, es posible usar simplemente un modelo de un PLC para ejecutar cualquiera de las 15 máquinas. Además, probablemente se necesitarían menos de 15 controladores, porque un PLC puede ejecutar muchas máquinas fácilmente. Cada una de las 15 máquinas bajo el control del PLC tendría su propio programa distinto. Además los PLC's se adaptan fácilmente en una amplia gama de aplicaciones, debido a que su operación depende directamente del programa que se diseñe con el fin de controlar las operaciones que conforman un proceso.

Diseñado para trabajar en un ambiente industrial. La tecnología electrónica utilizada por los fabricantes de PLC's garantiza su operación dentro de un ambiente industrial, el cual se garantiza por condiciones especiales de trabajo: ruido eléctrico, alta/baja temperatura, vibraciones mecánicas, etc.

Implementando Cambios y Corrigiendo los Errores. Con un tablero alambrado del tipo relevador, cualquier alteración del programa requiere tiempo para el realambrado de tableros y dispositivos. Cuando un cambio es hecho en el circuito diseño de la secuencia, el programa de PLC puede cambiarse desde un teclado en cuestión de minutos. No se requiere ningún realambrado para un sistema de PLC-controlado. También, si un error de la programación tiene que ser corregido o si por algún motivo un proceso de control es ampliado y las tareas de control se ven incrementadas, un cambio puede teclearse rápidamente y el PLC puede ser reprogramado y reconfigurado con la finalidad de ajustarse a los nuevos requerimientos.

Programable por personas no necesariamente informáticas pero sí con conocimientos de automatización. La programación de un PLC no requiere el conocimiento de un lenguaje sofisticado, son capaces de funcionar en el medio industrial con versiones que utilizan el lenguaje de los electricistas: contactos "normalmente cerrados" y "normalmente abiertos". Incluyen funciones internas que facilitan la introducción de un programa, detectan, corrigen errores y supervisan la puesta en marcha del sistema.

Grandes Cantidades de Contactos. El PLC tiene un número grande de contactos por cada bobina disponible en su programación. Suponiendo un caso en el que un tablero alambrado de relevador tiene cuatro contactos y están en uso cuando un cambio de diseño requiere tres contactos más. Esto significaría que se debe tomar tiempo para instalar un nuevo relevador o un bloque de contacto de relevador. Usando un PLC, sin embargo, sólo requerirían que tres contactos más sean tecleados. Los tres contactos estarían automáticamente disponibles en el PLC. De hecho, pueden usarse cien contactos de un relevador si la memoria disponible de la computadora es suficiente.

El más bajo Costo. El crecimiento de la tecnología hace posible compactar más funciones en paquetes más pequeños y menos caros. Desde los años noventa es posible comprar un PLC con numerosos relevadores, cronómetros, contadores, un secuenciador, y otras funciones por pocos cientos de dólares.

Comunicación en red. Pueden interconectarse a través de toda la planta, con un conjunto variado de dispositivos de entrada/salida como: controles locales en base a microprocesadores, microcomputadores, PLC's compatibles, periféricos, etc., permitiendo configurar un sistema distribuido de supervisión y control.

Piloto de Corrido. Un circuito de PLC programado puede pre-correrse y evaluarse en la oficina o laboratorio. El programa puede teclearse, probarse, observarse, y modificarse si es necesario, ahorrando el valioso tiempo de la fábrica. En el contraste, los sistemas convencionales de relevador han sido mejor probados en la planta de la fábrica, lo es un factor que consume mucho tiempo.

Observación visual. El funcionamiento de un circuito de PLC puede verse directamente durante el funcionamiento en una pantalla de CRT. El funcionamiento o dis-funcionamiento de un circuito puede observarse cuando ocurre. Las rutas lógicas se iluminan en la pantalla conforme se van energizando. La solución de problemas puede hacerse más rápidamente durante la observación visual.
En los sistemas de PLC avanzados, un mensaje del operador puede programarse para cada posible funcionamiento defectuoso. La descripción del funcionamiento defectuoso aparece en la pantalla cuando el funcionamiento defectuoso es descubierto por la lógica del PLC (por ejemplo, "MOTOR # 715 SOBRECARGADO"). Los sistemas de PLC avanzados también pueden tener descripciones de la función de cada componente del circuito.

Velocidad de Funcionamiento. Los relevadores pueden tomar una cantidad inaceptable de tiempo para actuar La velocidad operacional para el programa de PLC es muy rápida. La velocidad para la lógica de operación del PLC es determinada por el tiempo de exploración que es cuestión de milisegundos.

Método de Programación de Escalera. La programación de los PLC's puede lograrse en el modo de escalera por un electricista o técnico. Alternativamente, un

programador de PLC que trabaja en control lógico o binario de sistemas también pueden realizar fácilmente la programación de un PLC.

Fiabilidad. Los dispositivos en estado sólidos son más fiables, en general, que los relevadores mecánicos o eléctricos y cronómetros. El PLC está hecho de componentes electrónicos en estado sólido con rangos de fiabilidad muy altos.

Simplicidad de los Componentes del Sistema de Mando. Un PLC es un dispositivo con una fecha de entrega. Cuando el PLC llega, todos los contadores, relevadores, y otros componentes llegan también. Por otro lado, al diseñar un tablero de relevador, se podrían tener 20 distintos relevadores y cronómetros de 12 proveedores diferentes. Dejando las cosas más claras, esto involucra varias fechas de la entrega y disponibilidades. Con un PLC se tiene un producto y tiene tiempo para la entrega. En un sistema de relevador, la falta de un componente significaría un retardo en el inicio del sistema de control hasta que ese componente llegue. Con el PLC, un relevador más está siempre disponible si se tiene un PLC con bastante poder de computo extra.

Documentación. Una copia impresa del circuito de control que realiza virtualmente el PLC está disponible en minutos, si es requerido. El PLC imprime a menudo el circuito de control en funcionamiento en un momento dado, en cambio las impresiones de archivo para tableros de relevador no se guardan propiamente actualizados.

Seguridad. Un cambio en el programa del PLC no puede hacerse a menos que el PLC sea apropiadamente, desbloqueado y programado. Los tableros de relevador tienden a sufrir cambios indocumentados, las personas que hacen los cambios a horas poco usuales no siempre graban las alteraciones del tablero hechas cuando el área de la oficina se cierra con llave durante la noche.

Facilidad de Cambios Reprogramando. Desde que el PLC puede reprogramarse rápidamente, pueden lograrse el procesamiento de producción mixta. Por ejemplo, si la parte B baja la línea de ensamble está detenida mientras la parte A todavía está procesándose, un programa, para la parte los B puede ser reprogramado en la producción de maquinaria en segundos.

Entre otras ventajas están las siguientes:

- Mínimo espacio de ocupación.
- Menor costo de mano de obra.
- Mantenimiento económico.
- Posibilidad de gobernar varias máquinas con el mismo PLC.
- Menor tiempo de puesta en funcionamiento.
- Si el PLC queda pequeño para el proceso industrial puede seguir siendo de utilidad en otras máquinas o sistemas de producción.

1.7.2 DESVENTAJAS

Las siguientes son algunas de las desventajas, o quizás las precauciones para, uso de los PLC's:

La más nueva Tecnología. Es difícil de cambiar al personal de las ideas de escaleras y relevadores a los conceptos de computo del PLC.

Aplicaciones fijas del Programa. Algunas aplicaciones son aplicaciones de una sola función. No se debe pagar por usar un PLC que incluye las capacidades de la programación múltiple si no se necesita. Un ejemplo está en el uso del tambor controlador/secuenciadores. Algunos fabricantes de equipo todavía usan un tambor mecánico con las clavijas a ventaja del costo global. Su sucesión operacional raramente o nunca es cambiada, así que la reprogramación con el PLC no sería necesaria.

Consideraciones del medioambiente. Ciertos ambientes del proceso, como el calor alto y vibración, interfieren con los dispositivos electrónicos, lo cual limita su uso.

Funcionamiento "seguro contra fallas". En los sistemas de relevador, el botón de relevador desconecta el circuito eléctricamente; si la energía falla, el sistema se detiene. Además, el sistema del relevador no reinicia automáticamente cuando el poder se restaura. Esto, claro, puede ser programado en el PLC; sin embargo, en algunos programas de PLC, a veces se tiene que aplicar un voltaje de la entrada para causar que un dispositivo se detenga. Estos sistemas no están "seguros contra fallas." Se tendrían que agregar relevadores de seguridad a un sistema de PLC.

Funcionamiento fijo del circuito. Si el circuito en el funcionamiento nunca es alterado, un sistema de control fijo como un tambor mecánico, por ejemplo, podría ser menos costoso que un PLC. El PLC es muy eficaz cuando los cambios en el funcionamiento son periódicos.

CAPITULO II

PROCEDIMIENTOS GENERALES DE PROGRAMACIÓN DE UN PLC

Después de definir que es un PLC, examinar el hardware del sistema, y establecer los requerimientos mínimos para poder aprender su programación, es posible comenzar a tratar algunos de los procedimientos de programación más importantes. Para este fin, es necesario conocer primero el equipo de programación: módulo de programación (programmer/monitors o PMs) y software con el que cuenta el PLC, para después tratar los formatos de programación y entonces proceder a la construcción apropiada de los diagramas de escalera del PLC.

Finalmente, se verá cómo el PLC examina errores operacionales y cómo le comunica al usuario lo que es incorrecto a través de mensajes de error y de los paneles de LED's.

2.1 EL EQUIPO DE PROGRAMACIÓN

El equipo de programación permite que se escriba, corrija, y supervise un programa, así como realizar varios procedimientos de diagnóstico. En la mayoría de los casos el dispositivo de programación, el programmer/monitor (PMs) se debe conectar con la CPU mientras que se escriben los programas. Sin embargo, otros PMs, permiten ser programados fuera de línea y después descargar el programa a la CPU del PLC. Los programas se escriben generalmente en lógica de escalera, aunque otros lenguajes de programación alternativos están disponibles.

Existen tres tipos de PMs de uso común, también llamados cargadores de programas. Uno de ellos es la unidad, del tamaño de la palma de la mano (hand-held) con los teclados numéricos de doble función y una pantalla de cristal líquido (LCD) o de diodo electroluminoso (LED). En un nivel de uso más fácil están los teclados del mismo tamaño, acompañados por una pantalla grande del LCD o una pantalla del tubo de rayos catódicos (CRT). Una tercera opción de programación existe con el software que permite que los programas sean desarrollados en PC's compatibles con IBM.

Los modelos disponibles de unidades de programación se han incrementado en años recientes. Con la función de la tecla de mayúsculas, que trabaja como una segunda llave de funcionamiento en calculadoras, con lo cual se

consigue tener un teclado numérico relativamente completo. Los símbolos del dispositivo, los indicadores de la función, las llaves numéricas, los botones de la edición de programas y de la entrada, y las llaves del movimiento del cursor quedan justo en la yema del dedo. Un teclado numérico típico es codificado en color en base de la función y utiliza las llaves de la membrana que proporcionan una regeneración de audio.

La pantalla para las unidades hand-held también se ha ampliado y se ha mejorado. El LCD para estos dispositivos es capaz de indicar ocho peldaños de un diagrama de la escalera al mismo tiempo, conteniendo en cada peldaño hasta nueve elementos (contactos) y una función de bobina. Además, mensajes escritos, completamente en alfanuméricos, aparecen en una línea de mensaje.

Cuando la unidad se pone en el modo del monitor, la operación de dispositivos se puede observar, no solamente en la línea de mensaje, sino también en el diagrama de la escalera. Por ejemplo, cuando se corre un programa para un contador de tiempo, la línea de mensaje contará el tiempo de manera descendente, mientras que los varios elementos aparecen sombreados al pasar por ellos la energía. Los PM's de tamaño normal, cuentan con un teclado completo y un monitor grande.

El teclado contiene generalmente todos los símbolos del ASCII (teclado típico de la computadora) más un anfitrión de las funciones llaves dedicadas a la programación del PLC. No se requiere ninguna tecla de mayúsculas para invocar una segunda función, como es el caso con las unidades hand-held más pequeñas.

Debido a su tamaño más grande, la pantalla del monitor puede presentar una cantidad de información considerable al mismo tiempo. Además del uso de programadores hand-held o de tamaño normal, se encuentra disponible el software de gran alcance para programar PLC's que corre en máquinas compatibles con IBM. Una vez que la programación actual está completa, el programa se descarga al PLC.

Todos esos programas de software del PLC ahora forman parte de un menú con el cual se les puede manejar. A través del programa, es posible puede cambiar de menú a menú incorporando el número de la selección adecuado. Cuando se requiere la información adicional, el programa solicita una entrada de información por parte del usuario. Tal entrada puede ser hecha presionando la llave apropiada o en algunos programas, usando un ratón.

2.2 FORMATOS DE PROGRAMACIÓN

En cuanto a los procesos de programación, los distintos fabricantes tienen distintos formatos. Tal vez una buena forma para dominar este campo sea utilizar un formato general como los de las compañías que tienen una parte importante del mercado del PLC actualmente. La experiencia ha demostrado que cuando una persona aprende programar un tipo de PLC, él o ella puede dominar fácilmente otros sistemas de PLC, aunque los formatos tienen algunas diferencias. Algunos de los factores que varían entre los formatos son nomenclatura, la numeración de los esquemas, y aspecto de la pantalla. Las descripciones de la nomenclatura se pueden cubrir realizando diferentes ejemplos.

Otra variación del formato está en los formatos de enumeración para los contactos, las salidas, y los registros. Estos formatos incluyen letras, números, o una combinación de ambos. Los manuales de operación individuales del PLC explican los varios sistemas de designación de funciones y registros.

En general la programación de un PLC puede llevarse a cabo de acuerdo a como lo indica su manual de operación, no obstante, sabemos que existen 4 formas significativas para programar un PLC (aunque como se verá más adelante no son las únicas):

- Lenguaje a contactos: LD ó KOP.
- Lenguaje por Lista de Instrucciones: IL ó AWL.
- Gráfico de Orden Etapa Transición (GRAFCET) .
- Plano de Funciones (FBD) .

Sin embargo, antes de programar un PLC es conveniente revisar de manera rápida la lógica interna del mismo.

2.3 REGISTROS Y ACUMULADORES

Todas las operaciones que se hagan con las entradas y las salidas se deben efectuar en algún sitio. Para ello, se definen:

- **Registro de estado (VKE)**: Su tamaño es de 1 bit. Aquí es donde se efectuaren las instrucciones combinacionales, la carga de entradas y la asignación de salidas a nivel de bit.

- **Acumuladores (AKKU1 y AKKU2)**: Sus tamaños son de 16 bits cada uno. Cada vez que cargue un dato en los acumuladores se seguirá la siguiente secuencia:

Contenido de AKKU2 ===> Se pierde el contenido
Contenido de AKKU1 ===> AKKU2
DATO ===> AKKU1

A su vez, cuando se realiza una operación entre AKKU's (como suma o resta) el resultado se introducirá en el AKKU1, perdiéndose el valor antes allí contenido.

2.4 TEMPORIZADORES Y CONTADORES

Varían en función de marcas y modelos, pero los más usados suelen incorporar 32 temporizadores: T0 ... T31 y 32 contadores: Z0 ... Z31
De los 32 contadores, 8 no se borran al desconectar el PLC (son remanentes), dichos contadores son Z0 a Z7. Para consultar el estado de cada uno de ellos es posible usarlos como si fueran entradas (mediante operaciones combinacionales) o introduciendo su valor en los AKKU.

2.5 CONSTANTES Y VARIABLES INTERMEDIARIAS

Las variables intermediarias son manejadas internamente y no tienen entradas o salidas físicas asociadas, su función consiste en servir como enlace entre funciones lógicas.

Normalmente los PLC's presentan un número determinado de entradas y salidas los cuales se agrupan en múltiplos de 8, por ejemplo una configuración típica es de 16 entradas y 8 salidas. A la hora de cargar datos en acumuladores, temporizadores, registros, etc. existen varias posibilidades en la forma de introducir el dato:

- KB: 8 bits (0 a 255 en decimal).
- KC: 8 bits (2 caracteres alfanuméricos).
- KF: 16 bits (nº en coma fija, +32768 a -32768).
- KH: 16 bits (nº hexadecimal, 0000 a FFFF).
- KM: 16 bits (binario natural).
- KY: 16 bits (2 bytes, 0 a 255 en decimal cada uno).

- KT: 16 bits (valor de preselección de temporizadores, 0.0 a 999.3 en decimal).
- KZ: 16 bits (valor de preselección de contadores, 0 a 999 en decimal).

2.6 TIPOS DE MÓDULOS

Generalmente, existen cuatro tipos de módulos en cualquier PLC:

- **Módulos de organización (OB)**: Son los que gestionan el programa de usuario. Numerados OB1, OB3, OB21 y OB22. Destacar el OB1, que es el módulo del programa principal, el OB3, que es el que contiene el programa controlado por alarma, y el OB13, que es el módulo para programas controlados por tiempo. El OB22 es empleado por el sistema operativo.
- **Módulos de programa (PB)**: Son los que incluyen el programa de usuario dividido, normalmente, según aspectos funcionales o tecnológicos. PB0 ... PB63
- **Módulos funcionales (FB)**: Son módulos de programa especiales. Aquí se introducen las partes de programa que aparecen con frecuencia o poseen gran complejidad. Poseen un juego de instrucciones ampliado. FB0 ... FB63
- **Módulos de datos (DB)**: En ellos se almacenan datos para la ejecución del programa, como valores reales, textos; etc. Adoptan los valores: DB0 ... DB63

Los módulos DB1 y DB2 se emplean para definir las condiciones internas del PLC, por lo que no deben emplearse.

La mayor ventaja que aportan es la facilidad para variar el proceso que controlan, ya que basta con cambiar el programa introducido en el PLC (en la mayoría de los casos). Otra ventaja es que el PLC también permite saber el estado del proceso, incluyendo la adquisición de datos para un posterior estudio.

2.7 FALLAS OPERACIONALES DEL PLC

Cada PLC tiene códigos de error para identificar la programación incorrecta y fallas de operación. Los códigos aparecen en el monitor, generalmente en forma del código en sistemas pequeños o en lenguaje de uso sencillo en sistemas más grandes, cuando algo es incorrecto. Por ejemplo, en un sistema pequeño, se exhibe el código de error usando uno o dos dígitos hexadecimales. Cada dígito del código de error indica un conjunto diverso de condiciones que requieren atención. Para ilustrar, un código de error suponga que la lectura es "24." Consecuentemente, dos condiciones requieren la atención: "error de la revisión de la suma del programa, del primer grafico y " reemplazo de la memoria "del segundo gráfico. Un código de error "80" indica que existe un "error de programación" (8 es el dígito de la exhibición del código de error a la izquierda; 0, el dígito de la exhibición del código de error a la derecha). El usuario se referiría a sus manuales operacionales para una explicación de cómo localizar el problema.

En muchos sistemas, en el caso de que se pierda la conexión del sistema, o conexiones pobres, el usuario recibe un mensaje como "error de comunicación." Para otros problemas diversos aparecerán mensajes en la pantalla, generalmente en la parte de abajo. La mayoría del PLC's tiene otras ayudas de diagnóstico en forma de LED en el panel delantero del controlador. Un fabricante, por ejemplo, utiliza cinco LED's para indicar la varias condiciones de error, generando un código hexadecimal y un LED testigo de encendido y apagado.

2.8 ESTRUCTURA DE UN PROGRAMA

Se tienen dos opciones para escribir el programa:

- **Lineal**: Se emplea un único módulo de programa (OB1). Este módulo se procesa cíclicamente, es decir, tras la última instrucción se volverá a ejecutar la primera. Si la tarea a controlar es simple esta es la mejor forma.

- **Estructurada**: Para el caso de tareas complejas es más conveniente dividir el programa en módulos. Mediante esta forma se logra un programa más claro y se adquiere la posibilidad de poder llamar a un módulo desde distintas partes del programa (lo que evita repetir código).

En la programación estructurada se comienza y termina en el módulo OB1, desde el cual se puede saltar y retornar a los módulos que se deseen (la figura 2.1 muestra un esquema a bloques de esto). Por supuesto se podrá saltar desde un módulo a otro (anidado), siempre que no se superen los 16 niveles de salto que permite como máximo el PLC. Otras limitaciones son:

- El salto de un módulo a otro debe ser siempre hacia adelante (ej. Se podrá saltar de PB1 a PB2, pero no a la inversa).

- No se pueden dar dos saltos a un mismo módulo desde el módulo actual. (ej. No se podrá saltar dos veces a PB3 desde PB2, pero si puede saltarse a PB3 desde distintos módulos).
- Tanto en la programación lineal como en la estructurada los módulos acabarán mediante la instrucción BE.
- La memoria del PLC S5-90U está limitada a 2K bytes. Cada instrucción ocupa generalmente 2 bytes, por lo que es posible disponer 1000 líneas de programa aproximadamente.

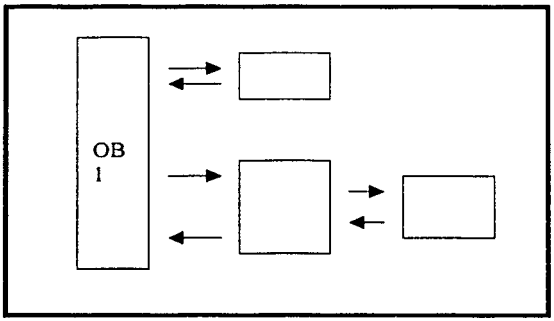


Figura 2.1 Esquema a bloques de la programación estructurada.

2.9 LENGUAJES DE PROGRAMACIÓN

Cuando surgieron los PLC's, lo hicieron con la necesidad de sustituir a los enormes cuadros de maniobra construidos con contactores y relevadores. Por lo tanto, la comunicación hombre-maquina debería ser similar a la utilizada hasta ese momento. El lenguaje usado, debería ser interpretado, con facilidad, por los mismos técnicos electricistas que anteriormente estaban en contacto con la instalación. Estos lenguajes han evolucionado, en los últimos tiempos, de tal forma que algunos de ellos ya no tienen nada que ver con el típico plano eléctrico a relevadores.

La inevitable dispersión en los lenguajes de programación ha llevado a requerir la estandarización para la programación de PLC's. La cual se llevó a cabo en Agosto de 1992, con el IEC 1131-3. Los lenguajes gráficos y textuales definidos

en el estándar son una fuente base para entornos de programación potentes en PLC's.

Con la idea de hacer el estándar adecuado para un gran abanico de aplicaciones, han sido definidos en total cinco lenguajes:

- Gráfico secuencial de funciones (grafcet).
- Lista de instrucciones (LDI o AWL).
- Texto estructurado.
- Diagrama de flujo.
- Diagrama de contactos.

2.9.1 GRÁFICO SECUENCIAL DE FUNCIONES (SFC o GRAFCET)

Es el llamado Gráfico de Orden Etapa Transición. Ha sido especialmente diseñado para resolver problemas de automatismos secuenciales. Las acciones son asociadas a las etapas y las condiciones a cumplir a las transiciones. Este lenguaje resulta enormemente sencillo de interpretar por operarios sin conocimientos de automatismos eléctricos.

Muchos de los PLC's que existen en el mercado permiten la programación en GRAFCET, tanto en modo gráfico o por lista de instrucciones. También es posible utilizarlo para resolver problemas de automatización de forma teórica y posteriormente convertirlo a plano de contactos, la figura 2.2 muestra un ejemplo gráfico.

El gráfico secuencial de funciones (Grafcet) es un lenguaje gráfico que proporciona una representación en forma de diagrama de las secuencias del programa. Soporta selecciones alternativas de secuencia y secuencias paralelas. Los elementos básicos son pasos y transiciones. Los pasos consisten de piezas de programa que son inhibidas hasta que una condición especificada por las transiciones es conocida. Como consecuencia de que las aplicaciones industriales funcionan en forma de pasos.

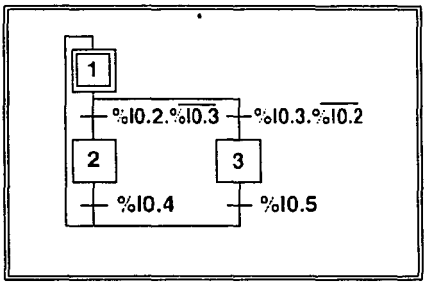


Figura 2.2 Ejemplo gráfico de programación en GRAFCET.

2.9.2 LISTA DE INSTRUCCIONES (IL o AWL)

En los PLC's de gama baja, es el único modo de programación. Consiste en elaborar una lista de instrucciones o nemónicos que se asocian a los símbolos y su combinación en un circuito eléctrico a contactos, como se muestra en la figura 2.3. Este tipo de lenguaje es, en algunos casos, la forma más rápida de programación e incluso la más potente.

La lista de instrucciones es un lenguaje de bajo nivel, similar al lenguaje ensamblador. Con IL solo una operación es permitida por línea (ej. almacenar un valor en un registro). Este lenguaje es adecuado para pequeñas aplicaciones y para optimizar partes de una aplicación.

```
000   LD    %I0.1   Bp. inicio ciclo
      AND   %I0.0   Op. presencia vehiculo
      AND   %M3     Bit autorizacion reloj calendario
      AND   %I0.5   Fc. alto rodillo
      AND   %I0.4   Fc. detrás pórtico
005   S     %M0     Memo inicio ciclo
      LD    %M2
      AND   %I0.5
      OR    %I0.2   Bp. parada ciclo
      R     %M0
010   LD    %M0
      ST    %Q0.0   Piloto ciclo
```

Figura 2.3 Ejemplo de código en Lista de Instrucciones.

2.9.3 TEXTO ESTRUCTURADO

El texto estructurado (structured text o ST) es un lenguaje de alto nivel estructurado por bloques que posee una sintaxis parecida al PASCAL. El ST puede ser empleado para realizar rápidamente sentencias complejas que manejen variables con un amplio rango de diferentes tipos de datos, incluyendo valores analógicos y digitales. También se especifica tipos de datos para el manejo de horas, fechas y temporizaciones, algo importante en procesos industriales. El lenguaje posee soporte para bucles iterantes como REPEAT UNTIL, ejecuciones condicionales empleando sentencias IF-THEN-ELSE y funciones como SQRT() y SIN().

2.9.4 DIAGRAMA DE CONTACTOS

También llamado Diagrama de Escalera, es el que más similitudes tiene con el utilizado por un electricista al elaborar cuadros de automatismos.

Muchos PLC's incluyen módulos especiales de software para poder programar gráficamente de esta forma. El diagrama de contactos (ladder diagram LD) es un lenguaje que utiliza un juego de símbolos de programación, un ejemplo de esta forma de programación se muestra en la figura siguiente.

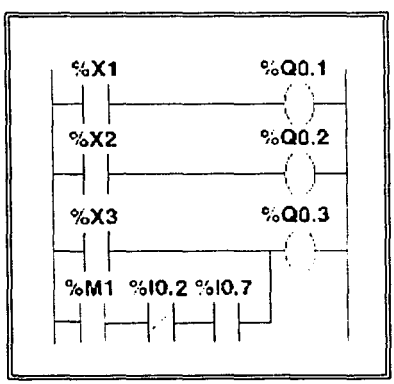


Figura 2.4 Ejemplo de programación para un PLC con Diagrama de Escalera.

2.9.5 DIAGRAMA O PLANO DE FUNCIONES

El diagrama de funciones (function block diagram o FBD) es un lenguaje gráfico que permite programar elementos que aparecen como bloques para ser cableados entre sí de forma análoga al esquema de un circuito. FBD es adecuado para muchas aplicaciones que involucren el flujo de información o datos entre componentes de control.

El plano de funciones lógicas, resulta especialmente cómodo de utilizar, a técnicos habituados a trabajar con circuitos de puertas lógicas, ya que la simbología usada en ambos es equivalente.

Los bloques de funciones son bloques que ejecutan algoritmos como reguladores PID. Hay controles empleando parámetros externos, mientras que los algoritmos internos permanecen ocultos empleando Programación Orientada a Objetos.

La siguiente figura presenta un ejemplo de programación por este método.

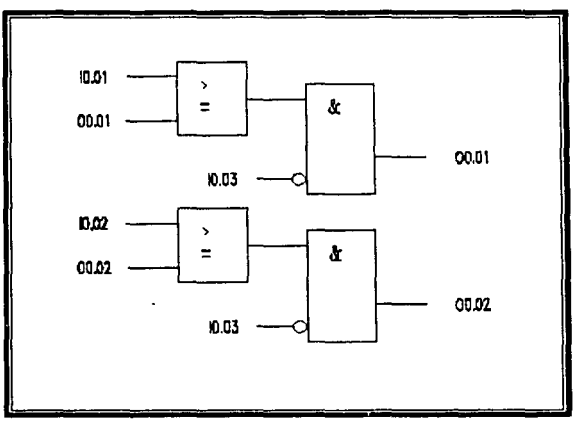


Figura 2.5 Programación mediante diagrama de bloques de funciones.

CAPITULO III

ESTRUCTURA Y FUNCIONAMIENTO BÁSICO DEL PROGRAMADOR LÓGICO MODULAR

Como ya hemos visto anteriormente, cada PLC cuenta con estructura, lenguaje de programación y lógica de funcionamiento propios, lo cual depende del fabricante y del modelo que se trate.

De lo anterior, es lógico pensar que el PLC prototipo diseñado en la Facultad de Ingeniería, que llamamos Programador Lógico Modular (PLM), cuente también con su propia estructura, lenguaje de programación y lógica de funcionamiento, por lo tanto es necesario definir de manera precisa los detalles más importantes de este dispositivo para después poder pasar al programa que se plantea como parte de este trabajo de tesis.

En este capítulo se describe de manera general la estructura a bloques del Programador Lógico Modular (PLM); la organización y nomenclatura asociada con las variables binarias que maneja, las características a nivel de "caja negra", de los módulos y lógicos que puede realizar el dispositivo y el formato sintáctico de las instrucciones para declararlas en SIIL1 (Software de Interpretación, de Instrucciones Lógicas), lenguaje propio del PLM para utilización por parte del usuario final.

Se presenta una descripción del formato que un programa en SIIL1 debe tener, de manera que el mismo pueda ser procesado en una PC para obtener el código objeto, listo para ser cargado y ejecutado en el PLM. El capítulo concluye con la metodología a seguir para la realización de una aplicación de control lógico empleando el PLM.

3.1 ESTRUCTURA BÁSICA DEL PROGRAMADOR LÓGICO MODULAR

El PLM, es un dispositivo orientado a la realización de diversos bloques funcionales típicos de aplicaciones de control lógico, como podrían ser: compuertas lógicas, temporizadores, contadores de eventos y secuenciadores de estados. En la nomenclatura del PLM se le llama módulo lógico a un bloque funcional, realizado virtualmente por software ejecutable en el microcontrolador 68HC11 F1, que gobierna el funcionamiento del dispositivo.

Los módulos lógicos que el PLM puede realizar son:

a) Compuertas AND de dos, tres y cuatro entradas.

b) Compuertas OR de dos, tres y cuatro entradas.

e) Compuertas NAND de dos, tres, y cuatro entradas.

d) Compuertas NOR de dos, tres y cuatro entradas.

e) Compuertas XOR de dos, tres y cuatro entradas.

f) Compuertas XOR negada de dos, tres y cuatro entradas.

g) Inversores y seguidores lógicos.

h) Cinco tipos diferentes de temporizador.

i) Dos tipos de contadores de eventos.

j) Secuenciadores de estado de uno a ocho bits.

k) Flip-Flops asíncronos (Latch RS).

Cabe señalar que en la terminología del PLM las compuertas XOR y XOR negada se denominan respectivamente como EOR y EORN; además que existe la posibilidad de negación para todas las compuertas en cualquiera de las entradas, lo cual contribuye a disminuir el número de módulos requeridos en una aplicación determinada.

El PLM puede operar de dos modos denominados autónomo y esclavo. Al operar de manera autónoma el PLM puede realizar un sistema de control lógico, ejecutando el código correspondiente que se encontrará residente como firmware en una EPROM contenida en el mismo, esta idea se ilustra en la figura 3.1; cuando el PLM opera en modo esclavo el mismo se encontrará ligado vía serie con una computadora de tipo PC donde puede ejecutarse software que permitirá probar y depurar los programas que requiera el PLM para realizar una determinada aplicación de control lógico, véase la figura 3.2.

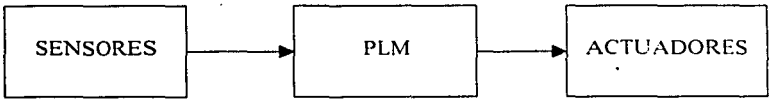


Figura 3.1 PLM operando en forma autónoma

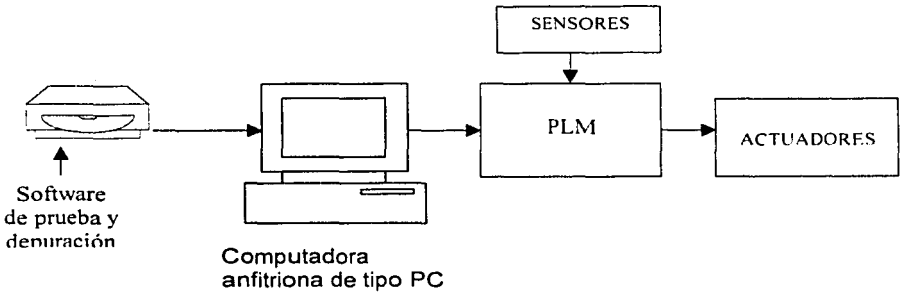


Figura 3.2 PLM operando en modo esclavo

El PLM cuenta con 32 entradas y 16 salidas binarias y está conformado por los siguientes cinco bloques funcionales

1) Computadora Central (CC)

2)Bloque de Entradas (BE)

3) Bloque de Salidas (BS)

4) Bloque de Comando Local y Despliegue (BCLD)

5) Fuente de Alimentación (FA)

A continuación se describe, en lo general, el funcionamiento de cada uno de los bloques del PLM.

3.1.1 COMPUTADORA CENTRAL (CC)

La computadora central del PLM está realizada por la computadora monotablilla (CMT) SIMMP-2, cuya CPU es el microcontrolador 68HC11 F1 fabricado por la compañía Motorola, la CMT SIMMP-2 puede operar en cualquiera de los cuatro modos en los que puede funcionar el 68HC11 y cuenta con facilidades que permiten que la misma opere de manera autónoma o bien controlada vía serie por una computadora anfitriona de tipo PC. La CMT SIMMP-2 fue desarrollada en la Facultad de Ingeniería y tiene las siguientes características principales:

1) Capacidad para operar en cualquiera de los cuatro modos asociados con el 68HC11.

2) Firmware interlocutor que permite enlazarla vía serie a una computadora PC, donde se ejecuta un manejador hexadecimal (programa pumma.exe) mediante el cual se puede entre otras cosas hacer lo siguiente:

   a. Cargar desde la PC, programas en lenguaje de máquina del microcontrolador para su ejecución en el mismo.

   b. Lectura desde la PC, de la memoria contenida en la tarjeta.

   c. Ejecución en la tarjeta de programas originalmente escritos en lenguaje C o ensamblador, lográndose esto mediante la importación del archivo S19 correspondiente que haya sido generado por el software de ensamble o compilación respectivo. Todo lo anterior, gracias a su compatibilidad con herramientas de software asociadas con el 68HC11.

   d. Configurar diversos mapas de memoria al operar el modo expandido.

   e. Programarse EPROM's usando el propio manejador hexadecimal y hardware contenido en la tarjeta, gracias al programador integrado de memorias EPROM.

La CMT SIMMP-2 como computadora central del PLM opera en el modo expandido del 68HC11, contándose en este caso con seis puertos, cuatro de entrada y dos de salida, con los que se realizan a nivel de la CC las entradas y salidas con que cuenta el PLM.

3.1.2 BLOQUE DE ENTRADAS (BE)

Esta parte está conformada por 32 entradas optoacopladas, el PLM reconoce un nivel de uno lógico, para una determinada entrada, cuando nominalmente se presente un voltaje de 24 volts medido entre la terminal correspondiente y el punto NFS (neutro de la fuente de sensores), en otro caso el nivel tomado será cero lógico.

Las 32 entradas del PLM están agrupadas en cuatro grupos de ocho entradas cada uno, esto se debe a que la información en el microcontrolador empleado está organizada en bytes. Las entradas son denotadas empleando tres caracteres, el primero puede ser la letra "e" mayúscula o minúscula, el segundo es un número comprendido en un rango de cero a 3 que indica el grupo, y finalmente el tercer caracter puede ser un número comprendido entre cero y siete que indica el bit de entrada correspondiente; así por ejemplo, la entrada correspondiente al bit 3 del grupo 2 puede ser indicada como "E23"; para cada grupo de entradas corresponde un puerto físico con una determinada dirección en el mapa de puertos de la CC.

3.1.3 BLOQUE DE SALIDAS (BS)

Este bloque está realizado por dos puertos de salida de la CC, de modo que para cada uno de sus bits se cuenta con una interfaz a un relevador de baja potencia de contactos normalmente abiertos. Todas las terminales comunes de contactos de los 16 relevadores están conectadas al punto VFA (vivo de la fuente de actuadores) en tanto que para cada relevador el otro contacto está conectado con su correspondiente terminal de salida asociada.

Las salidas se denotan con tres caracteres, el primero es la letra "s" mayúscula o minúscula, el segundo es un número que puede ser cero o uno indicando esto el grupo al que pertenece la salida en cuestión, finalmente el tercer caracter es un número comprendido entre cero y siete que denota el número de bit de salida correspondiente; así por ejemplo, la salida 4 del grupo uno puede indicarse como "S14"

Al verificarse el nivel de uno lógico para una determinada salida habrá continuidad eléctrica entre las terminales VFA y la propia correspondiente con la salida, en otro caso no habrá continuidad eléctrica, la máxima corriente permisible, para disparo del actuador correspondiente, es 2 [A].

3.1.4 BLOQUE DE COMANDO LOCAL Y DESPLIEGUE (BCLD)

Desde el punto de vista del usuario final este bloque está constituido por tres componentes, uno de ellos es la Unidad Desplegadora (UD) que maneja dos renglones de 16 caracteres, otro es un panel que contiene cinco postes que habilitan sendas entradas binarias auxiliares, cuatro botones para comando local y dos pares de postes donde se podrían colocar puentes (jumpers) que configurarían la manera en que el PLM respondería a una reinicialización del programa del usuario.

El tercer componente del BCLD es un reloj de tiempo real, que puede servir simplemente como testigo de la hora o como base de tiempo para una función especial del dispositivo que permite generar disparos a otros módulos lógicos para horarios predeterminados por el usuario en el programa en SIIL1, hecho de acuerdo con las necesidades de una determinada aplicación. Para la implantación de la unidad desplegadora se usa el desplegado alfanumérico inteligente AND491 fabricado por la corporación electrónica PURDY; en lo que toca al reloj de tiempo real se emplea el chip MM58274N fabricado por NATIONAL, que es una componente pensada para interconectarse con un microcontrolador o microprocesador.

Los botones y postes para entradas auxiliares o colocación de puentes, están validados por dos puertos de entrada (denotados como PAUXA y PAUXB) adicionales a los que forman parte de la arquitectura de la CMT SIMMP-2. Los cuatro botones están denotados como BAXA, BAXB, BAXC y BAXD; los tres primeros son usados para poner el reloj de tiempo real a una hora determinada, el último es usado como auxiliar en una de las instrucciones que manejan la UD; los postes que presentan las cinco entradas auxiliares están denotados como EA1, EA2, EA3, EA4 y EA5; los puentes sirven para configurar tipos de respuesta al restablecimiento y se denominan Ja y Jb. En la tabla 1.1 se resumen, en lo general, las funciones de los botones de comando local, los puentes y las entradas binarias auxiliares del BCLD.

Instancia de Bloque de comando local despliegue (BCLD)	Uso en el PLM desde el punto de vista del usuario final
Botones BAXA, BAXB y BAXC	Ajuste y puesta a tiempo del reloj de tiempo real (RTR)
Botón BAXD	Este botón se emplea para desplegar secuencialmente mensajes priorizados en la UD
Entradas auxiliares EA 1 a EAS	Reservadas para funciones futuras que

	pudieran requerir botones o puentes
Puente Ja	Con Ja no colocado, al reinicializar el programa del usuario el reloj de tiempo real se pone en ceros (00:00:00), en otro caso e RTR conserva la hora al reinicializar el programa del usuario
Puente Jb	Con Jb no colocado, al reinicializar el programa del usuario, se ponen en cero todas las variables binarias que use la aplicación, en otro caso las variables conservan el valor tenían antes de la reinicialización

Tabla 3.1 Resumen de funciones asociadas con instancias del BCLD del PLM.

3.1.5 FUENTE DE ALIMENTACIÓN (FA)

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 1 Ampere, el requerimiento de corriente de la segunda fuente mencionada es de 500 mA.

3.2 VARIABLES BINARIAS EN EL PLM

Las entradas y salidas de los módulos lógicos que pueden ser realizados con el PLM son variables binarias que son clasificadas como: variables binarias de entrada (VBE), variables binarias de salida (VBS) y variables binarias intermediarias (VBI).

Dado que la información en el microcontrolador 68HC11 está organizada en bytes, las variables mencionadas están aglutinadas en conjuntos (grupos) de ocho variables de un mismo tipo; esto es, hay cuatro grupos de variables binarias de entrada, dos grupos de variables binarias de salida y 21 grupos de variables binarias intermediarias. A continuación se describen conceptos asociados con cada uno de los tipos de variables binarias del dispositivo.

3.2.1 VARIABLES BINARIAS DE ENTRADA (VBE)

Este tipo de variables están asociadas con sendas terminales de entrada siendo cada una de ellas optoacoplada a la CC, cada terminal de entrada puede recibir una sería lógica de voltaje (0-24 volts) proveniente de algún sensor, que sea parte del sistema de control lógico, que se requiera implantar en un momento dado.

El PLM está pensado para manejar 32 VBE's y como se ha mencionado anteriormente cada VBE se especifica con tres caracteres, el primero es una letra "e" mayúscula o minúscula, el segundo es un número del cero al tres que denota el grupo al que pertenece la VBE y el tercero es un número del cero al siete que define el bit asociado del puerto de entrada relacionado con el grupo de entradas de que se trate; así por ejemplo, la quinta variable del grupo de entradas dos se podría denotar como E25.

3.2.2 VARIABLES BINARIAS DE SALIDA (VBS)

Existen para el PLM 16 variables binarias de salida, cada una de ellas está asociada con un relevador de baja potencia cuyos contactos se cierran al presentar la variable de salida correspondiente el nivel de uno lógico, al ser cero lógico el valor en cuestión de tales contactos permanecen abiertos; las VBS están aglutinadas en dos grupos de ocho salidas cada uno; de esta forma, se emplean tres caracteres para denotar a una VBS, el primero es la letra "s" mayúscula o minúscula, el segundo es un número que puede ser cero o uno que denota el número de grupo de salida y el tercero es un dígito del cero al siete que define el bit asociado con el puerto de salida físico de la CC relacionado; así por ejemplo, la salida dos del grupo de salidas cero se podría definir como S02.

3.2.3 VARIABLES BINARIAS INTERMEDIARIAS (VBI)

Este tipo de variables son manejadas internamente y no tienen entradas o salidas físicas asociadas, su función consiste en servir de enlace entre módulos lógicos cuando esto sea necesario; por ejemplo, supóngase que se tiene una situación de control lógico que requiere de varias compuertas lógicas, puede suceder que las salidas de algunas de ellas sean variables requeridas como entrada de otras, el emplear variables físicas de salida para habilitar esta circunstancia no sería conveniente dado que su número es limitado de ahí la necesidad de contar con las VBi; desde luego que una variable binaria, que sea entrada de un módulo y salida de otro, puede ser una salida física si esto es necesario, lo que obviamente no es permitido es el hecho de que una variable sea simultáneamente salida de más de un módulo.

Las VBI están aglutinadas en 21 grupos de ocho variables cada uno, de esta forma se puede contar dentro del PLM con 168 variables de este tipo; la

notación empleada para designarlas emplea cuatro caracteres, el primero es la letra "I" mayúscula o minúscula, el segundo y tercero representan a un número comprendido entre cero y veinte que denota el número de grupo al que pertenece y el cuarto es un dígito del cero al siete que define el número de VBI dentro del grupo, así por ejemplo, la VBI cuatro del grupo doce de VBI's se denominaría I124.

3.3 DESCRIPCIÓN GENERAL DE LOS MÓDULOS LÓGICOS

Los módulos lógicos (ML) que puede realizar el PLM constituyen los bloques funcionales elementales para la 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; m y n varían de acuerdo con el tipo de función que un determinado ML realice; así por ejemplo, para una compuerta AND de tres entradas m y n serían tres y uno respectivamente; en cambio, un secuenciador de estados con palabras de cuatro bits requerirá tres entradas y cinco salidas.

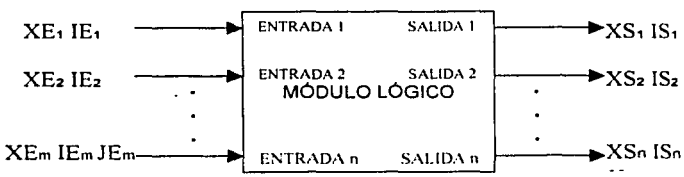


Figura 3.3 Representación de bloque de un módulo lógico de m entradas y n salidas.

En la figura 3.3 XEk representaría a un caracter que podría ser cualquiera de las letras E, I o S mayúsculas o minúsculas dependiendo esto del tipo de variable (VBE, VBI o VBS) asociada con la entrada k-esima del ML; IEk y JEk serían respectivamente los números asociados con el grupo y el número de bit correspondientes con la variable k-esima de entrada; XSk representaría a un caracter que podría ser cualquiera de las letras I o S mayúsculas o minúsculas dependiendo esto del tipo de variable (VBI o VBS) asociada con la salida k-esima del ML; ISk y JSk serían respectivamente los números asociados con el grupo y el número de bit correspondientes con la variable k-esima 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

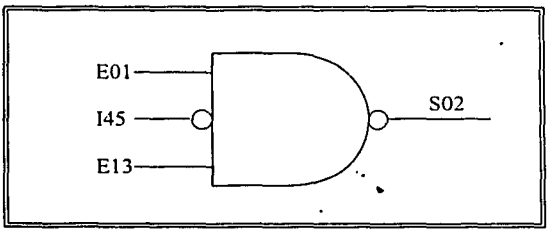


Figura 3.4 Representación de un Módulo Lógico que realiza una compuerta NAND de tres entradas con negación en una de ellas, nótese que las entradas pueden ser de grupos diferentes.

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 presenten en una o varias de sus entradas en la figura 3.5 se muestra un ML que se encuentra en este caso, se trata de 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, en lo que toca a 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 al 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.

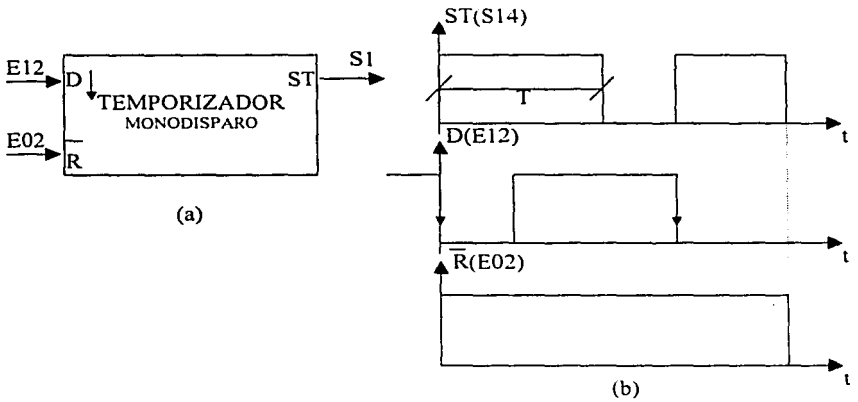


Figura 3.5 a) Representación en forma de bloque de un temporizador Monodisparo (one-shot), nótese la flecha vertical indicando que el disparo es por flanco de bajada.

b) Diagrama de tiempos correspondiente al temporizador mostrado.

3.4 FORMAS SINTÁCTICAS ASOCIADAS CON LOS MÓDULOS LÓGICOS

Cada uno de los módulos lógicos, requeridos en una determinada aplicación, debe haber sido declarados secuencialmente en un archivo de texto, para que el mismo sea procesado por una computadora anfitriona empleando un programa denominado SIIL1 (Software de Interpretación, de Instrucciones Lógicas), que genera el código objeto que ha de ejecutar la CC del PLM de modo que los ML requeridos queden realizados. Además de las declaraciones asociadas con los módulos, en el archivo mencionado se requieren colocar otras instrucciones, no relacionadas directamente con algún Módulo Lógico, pero necesarias para la ejecución adecuada del programa que ha de ejecutarse en el PLM. Al conjunto de instrucciones, mencionadas anteriormente, escritas en secuencia en un archivo de texto, se le denomina *programa fuente de lenguaje SIIL1* asociado con la aplicación que ha de realizar el PLM. 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 si 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: que entradas a una compuerta van a tener negación implícita, a que tipo de flanco responde una entrada de algún otro tipo de módulo, etc. Cabe señalar que el primer caracter de la instrucción nunca deberá estar en la primera columna y que al final de la misma siempre ha de colocarse el caracter ";". 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.

Notese además de que la cadena binaria asociada (CADBI) consta de tres bits, ya que la compuerta es de tres entradas, indicándose que la entrada I45

deberá tener negación implícita colocando un cero en la posición que corresponde a tal entrada; así, si se requiriera que tuvieran negación implícita las dos primeras entradas (EO1 e I45) CADBI sería 001. Se aprecia también en la declaración anterior que se le ha asignado el número uno a la compuesta NAND en cuestión.

3.5 FORMATO DE UN PROGRAMA EN SIIL1

3.5.1 CARACTERÍSTICAS GENERALES DE LA EJECUCIÓN DE UN PROGRAMA EL SIIL1 EN LA CC DEL PLM

Al correr un programa en SIIL1 en la CC del PLM, el código asociado con cada ML es ejecutado cíclicamente siguiendo la siguiente secuencia.

1. Se copian en un buffer de entrada (BE) en RAM el estado que guardan los cuatro puertos asociados con las 32 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, tomándose del BE las entradas que cada módulo requiera. Las salidas que se fueran generando son colocadas en un buffer de salida (BS) en RAM; si el ML emplea una o varias VBI como entradas los valores asociados con las mismas son tomados de un buffer intermediario (BI) en RAM. En caso de que en el ML haya una o varias salidas de tipo VBI los valores correspondientes son escritos en el BI.

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

4. Se regresa al paso uno.

De lo anterior se aprecia que el código asociado con cada módulo es ejecutado repetitivamente, variando el intervalo de repetición de acuerdo con el número módulos que contenga el programa; esto es, a mayor número de módulos crece el periodo de repetición.

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 facilidades de temporización con que cuenta la CC del PLM.

En consecuencia, el código asociado con un programa en SIIL1 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 3.6 se ilustra esta idea. Cabe señalar aquí que todo programa en SIIL1 debe tener un subprograma principal; sin embargo, puede haber programas que no contengan un subprograma temporizado.

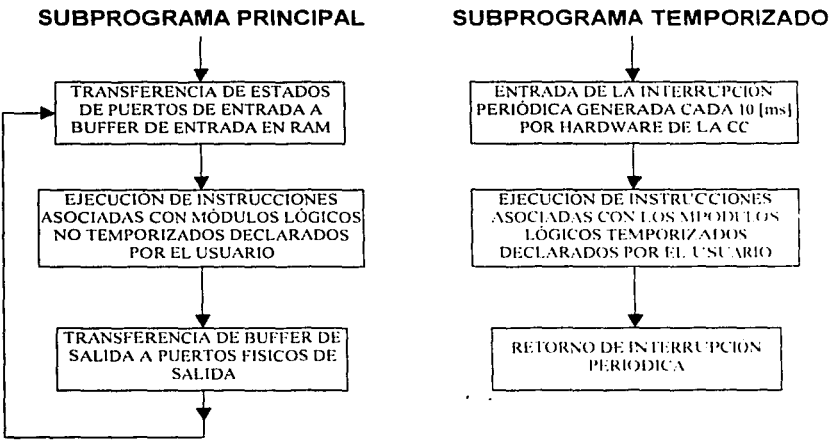


Figura 3.6 Ejecución en la CC del PLM, de los dos subprogramas que Integran un programa en SIIL1.

3.5.2 FORMA DE UN PROGRAMA FUENTE EN SIIL1

El programa fuente en SIIL1 asociado con una aplicación está constituido por declaraciones, que pueden ser comandos o instrucciones. Los comandos son indicaciones, tales como inicio o fin del 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 en párrafos anteriores.

En lo general un programa fuente en SIIL1 está integrado por la siguiente secuencia de declaraciones:

1. Comando que indica el tipo de configuración de funcionamiento, la sintaxis asociada es CONFIGN, donde N es un número entero que puede ser uno, dos, o tres; de esta manera, existen a la fecha de elaboración de este trabajo de tesis, tres posibles configuraciones de funcionamiento para el PLM. En la tabla 1.2 se resumen las características principales de funcionamiento asociadas con cada configuración.

2. Comando que marca el inicio del subprograma principal: la sintaxis correspondiente en este caso es INPROG.

3. Instrucciones asociadas con los módulos que se desea integren al subprograma principal.

4. Comando que marca el fin del subprograma principal: la sintaxis en este caso es FINPP.

5. Comando que marca el inicio del subprograma temporizado: la sintaxis asociada es INMODI.

6. Instrucciones asociadas con los módulos que han de integrar al subprograma temporizado.

7. Comando que indica el fin del subprograma temporizado, la sintaxis en este caso es FINMODI.

Configuración	*Entradas*	*Salidas*	*Tamaño máximo del programa [kb]*
1	32	16	7.5
*2**	8	8	7.5
3	32	16	24

Tabla 1.2 Resumen de características de funcionamiento del PLM asociadas con las diferentes configuraciones de funcionamiento.

* Esta configuración se empleó para la prueba de los módulos con lógica nivel TTL.

Los siete 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. Los comandos deberán ser concluidos con el caracter ";". Todo texto colocado a la derecha del caracter ";" no es tomado en cuenta por el programa que genera el código objeto, de esta manera pueden adicionarse comentarios al programa fuente.

3.6 METODOLOGÍA A SEGUIR PARA ESTRUCTURAR UNA APLICACION DE CONTROL LÓGICO EMPLEANDO EL PLM

Toda aplicación de control lógico puede integrarse empleando tres conjuntos de elementos funcionales denominados como:

1. Elementos sensores, los cuales son dispositivos que presentan en su salida un nivel lógico que testifica un determinado evento como podrían ser por ejemplo el paso de un producto por una banda transportadora, el fin del recorrido de un embolo, el que un operador oprima un botón, etc. Los sensores pueden estar constituidos desde por simples interruptores hasta por bloques que basan su funcionamiento en componentes electrónico s. Frecuentemente en la industria los niveles lógicos empleados son cero y 24 volts.

2. Elementos lógicos, los cuales son dispositivos que realizan funciones binarias cuyas entradas son las variables presentadas por los sensores, siendo sus salidas variables lógicas que comandan a elementos actuadores, de modo que físicamente se realicen los eventos que la aplicación requiera.

3. Elementos actuadores, que son dispositivos que actúan directamente sobre el proceso y son comandados por las salidas que generan los elementos lógicos mencionados en el párrafo anterior, ejemplos de actuadores podrían ser resistencias eléctricas que suministren calor, motores eléctricos que muevan elementos mecánicos de diversa índole, etc.

El papel del PLM en la realización de un sistema de control lógico es el de implantar los elementos lógicos que la aplicación requiera, empleando para ello a los módulos lógicos que él mismo puede realizar virtualmente.

A continuación se describe el proceso a seguir para que en el PLM tomen forma el conjunto de módulos lógicos necesarios en un control lógico, donde se supone que el PLM debe operar en modo esclavo y debe estar convenientemente enlazado con una computadora anfitriona de tipo PC entendiendo que el desarrollo completo debe contemplar lo relacionado con los sensores y actuadores sin embargo, aquí se habla únicamente acerca de lo concerniente al PLM. En síntesis los pasos a seguir son los siguientes:

1. Definir los módulos lógicos que la aplicación requiera, especificando para cada uno las variables de entrada y salida empleadas respetando el hecho de que una variable no puede ser salida de más de un módulo de lo anterior escribir el programa fuente asociado siguiendo los lineamientos dados en el punto 3.5, empleando para ello a un editor de texto convencional que se ejecute en la computadora anfitriona.

2. Guardar el texto generado en el paso anterior en un archivo con un nombre escogido por el usuario y con la extensión SIL.

3. Ejecutar en la computadora anfitriona el Software de Interpretación de Instrucciones Lógicas (programa SIIL1.EXE) tomando como archivo de entrada el generado en el paso anterior. En caso de haber errores de sintaxis en las declaraciones mencionadas en el paso uno se mostrarán en pantalla los mismos. En caso que no haya errores se indicará esto, generándose además un archivo binario con el mismo nombre dado por el usuario en el paso dos y con la extensión BLM; este último archivo contendrá el código ejecutable por el PLM correspondiente a la aplicación que se esté desarrollando.

4. Si se detectaron errores en el paso anterior corregirlos en el editor de texto y regresar al paso dos. Si no hubo errores de sintaxis proceder al paso cinco.

5. Transferir para ejecución, a la memoria RAM del PLM, el código contenido en el archivo BLM generado en el paso tres, esto puede hacerse empleando el manejador hexadecimal PUMMA propio de la tarjeta SIMMP-2.

6. En caso de que el programa no opere correctamente en el PLM hacer los cambios necesarios, a nivel de los módulos lógicos empleados por la aplicación, y regresar al paso uno. Si el programa opera correctamente proceder al siguiente paso.

7. Desenergizar el PLM, colocar una EPROM borrada en la base correspondiente, así como los puentes J4 y J5 en la CC (tarjeta SIMMP-2). Energizar el PLM, oprimir y soltar el botón de restablecimiento del mismo.

8. Programar la EPROM con el código objeto contenido en el archivo BLM generado en el paso tres. Para efectuar esto se puede emplear el manejador hexadecimal PUMMA, propio de la tarjeta SIMMP-2.

9. Quitar los puentes J4 y J5 y validar el modo autónomo de operación, esto último se hace colocando el puente J11 en la CC (CMT SIMMP-2).

10. Oprimir y soltar el botón de restablecimiento del PLM, al hacer esto deberá ejecutarse en forma autónoma el programa del usuario, de esta manera el sistema de control lógico diseñado funcionará siendo realizado por el PLM.

3.7 ESTRUCTURA DE UN PROGRAMA EN SIIL1

Un programa en SIIL1, está dividido en dos subprogramas denominados *subprograma principal* y *subprograma temporizado*, el primero podrá contener tanto el código correspondiente a módulos que realizan compuertas lógicas y Flip-Flops como el correspondiente a módulos auxiliares, además de código de inicialización, de lectura y escritura de buffers y de salto a la posición del inicio del ciclo de barrido (scan); en lo que toca al subprograma temporizado, éste es invocado por interrupción cada 10 [ms], empleándose para ello al canal OC2 del temporizador interno del microcontrolador, siendo el código asociado el correspondiente a los módulos que realizan temporizadores, secuenciadores, contadores de eventos y módulos auxiliares, además de haber código propio del manejo de la interrupción OC2 y de actualización de buffers testigo de estado anterior.

Los módulos auxiliares mencionados en el párrafo anterior, están asociados con acciones relacionadas con el manejo de la Unidad Desplegadora (UD) y el reloj de tiempo real, algunos deberán ser colocados en el subprograma principal y otros deberán declararse en el subprograma temporizado.

3.8 FLUJO DE EJECUCION DEL SUBPROGRAMA PRINCIPAL

El código del subprograma principal está dividido en dos partes denominadas respectivamente como INPLM3 y LPP (Lazo Programa Principal), en la primera se inicializan aspectos tales como la programación de puertos físicos, puesta a cero de buffers, carga de información relacionada con enrutamiento de interrupciones, carga de registros asociados con el canal OC2 del temporizador interno y colocación de testigos de primer paso por el LPP y subprograma temporizado, estos testigos son empleados en tiempo de ejecución, para inicializar condiciones lógicas en módulos que así lo requieran; por ejemplo, si el pulso de salida de un temporizador es verificado en alto, la salida correspondiente al mismo deberá ser inicializada con un nivel bajo y esta acción ha de ser efectuada la primera vez que se ejecute el código correspondiente a tal temporizador en la Computadora Central del PLM. Este proceso se describe de manera gráfica en la figura 3.7.

En lo que toca al código correspondiente al LPP a continuación se describen las acciones llevadas a cabo por el mismo:

1. Se copian en un buffer de entrada (BE) en RAM el estado que guardan los cuatro puertos asociados con las 32 entradas físicas VBE. El tramo de código asociado con esta acción se denomina LECBUF3.

2. Se ejecuta uno a uno el código asociado con cada uno de los ML que el usuario haya declarado en el subprograma principal, tomándose del BE las

entradas que cada módulo requiera, las salidas que se fueran generando son colocadas en un buffer de salida (BS) en RAM; si el ML emplea una o varias VBI como entradas, los valores asociados con las mismas son tomados de un buffer intermediario (BI) en RAM, en caso de que haya en el ML una o varias salidas de tipo VBI los valores correspondientes son escritos en el BI. El código correspondiente a esta parte, estará constituido por la unión de dos tramos de código correspondientes a cada uno de los ML que integren el subprograma principal, colocados en el orden en que los mismos hayan sido declarados.

3. Se copia el estado del BS en los puertos físicos asociados con las VBS. El tramo de código correspondiente a esta acción se denomina ESCBUF1.
4. Se regresa al paso uno. El tramo de código correspondiente a esta parte se denomina SALTO.

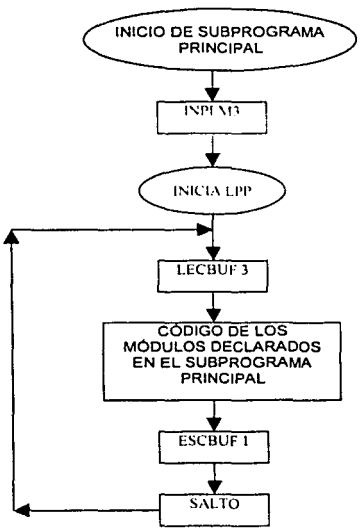


Figura 3.7 Flujo de ejecución del Subprograma Principal.

3.9 FLUJO DE EJECUCIÓN DEL SUBPROGRAMA TEMPORIZADO

El código correspondiente al subprograma temporizado, es ejecutado cada 10 [ms] y bajo la perspectiva del microcontrolador 68HC11, constituye la rutina de servicio de interrupción OC2 y está integrado por cinco tramos de código que son:

1. Entrada a rutina de servicio de interrupción OC2, este TC actualiza umbral de comparación del canal OC2 del temporizador interno del microcontrolador y pone a cero la bandera correspondiente (OC2F), el CEN asociado con este TC se denomina ENRUS.
2. Código asociado con los ML que hayan sido declarados en el subprograma temporizado, que deberá corresponder con módulos cuyo código requiere ser ejecutado cada 10 ms (v.g. temporizadores) y/o contienen entradas sensibles a flancos de subida o bajada.
3. Código que copia buffer de valor presente de todos los grupos de variables binarias en buffer testigo de valor anterior correspondiente. Este tramo de código, en su versión normalizada, se denomina ACT.
4. Código que copia a RAM el estado de los dos puertos de entrada auxiliares (PAUXA y PAUXB), este tramo de código, en su versión normalizada, se denomina ACTBOT.
5. Código de retorno de interrupción, denominado RETRUS.

En la figura 3.8 se muestra a bloques el flujo de ejecución del subprograma temporizado.

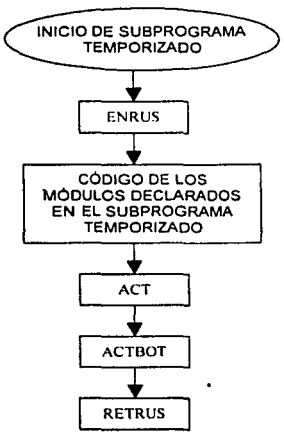


Figura 3.8 Flujo de ejecución del Subprograma Principal.

3.10 RESUMEN

Es claro que al programar el PLM se tienen dos tipos distintos de Módulos Lógicos, los cuales se declaran ya sea en el *Subprograma Principal* o en el *Subprograma Temporizado*, y para dejar una idea clara de cuáles son aquellos que se declaran en uno u otro, a continuación se enlistan por separado de acuerdo a la parte del programa en la que se tengan que declarar.

Módulos de subprograma principal

1. Seguidor lógico
2. Inversor lógico
3. Compuertas lógicas de 3 y 4 entradas:
   - AND
   - OR
   - NAND
   - NOR
   - OR EXCLUSIVA
   - OR EXCLUSIVA NEGADA
4. Flip-Flop Asíncrono RS (FFARS)

Módulos de Subprograma Temporizado

1. Compuertas lógicas de 2 entradas:
   - AND
   - OR
   - NAND
   - NOR
   - OR EXCLUSIVA
   - OR EXCLUSIVA NEGADA
2. Contador de eventos.
3. Secuenciador de estados.
4. Temporizador monodisparo.
5. Temporizador monodisparo 2° tipo.
6. Temporizador con retardo a la activación.

7. Temporizador astable.
8. Temporizador multipulso.
9. Temporizador generador de pulsos con el reloj de tiempo real (6 clases).

Módulos Auxiliares

1. Módulo con capacidad de generar texto estático y/o dinámico en la Unidad Desplegadora (UD).
2. Módulo con capacidad para generar mensajes de alarma.
3. Módulo observador del estado de contadores de eventos.
4. Módulo manejador del reloj de tiempo real.
5. Módulo DESP, que copia a la UD el contenido de su buffer asociado.
6. Módulo MANDESP, que permite observar para verificación, el texto asociado con módulos de tipo mensajero.

Hasta este momento se ha definido que es un PLC en términos generales partiendo de su historia, su estructura interna, su funcionamiento básico y sus aplicaciones más comunes, así como los formatos, equipos y procedimientos de programación, al mismo tiempo que se han tratado los lenguajes de programación más comunes en estos dispositivos, lo cual nos permitió pasar a la explicación de lo que es el PLM como un PLC prototipo desarrollado en la Facultad de Ingeniería, sobre el cual también se ha expuesto de manera general su estructura, funcionamiento y algunas de sus particularidades. Por lo tanto, ahora que se tienen estos elementos, es posible abordar la otra parte de este trabajo de tesis, la cual consta de la construcción de un ambiente visual que nos permita la programación del PLM mediante la conexión de los módulos anteriormente mencionados, es decir, la elaboración de un software que comprenda el desarrollo de dichos módulos funcionales en un *Diagrama o Plano de funciones*, de tal manera que el usuario sólo tenga que utilizar figuras de un menú ya disponible y que mediante una conexión pueda establecer una secuencia lógica que se desarrollará de manera virtual dentro del PLM y que en términos prácticos, esta metodología junto con el PLM se puedan utilizar para el control de procesos industriales y es esa la parte a la que se dedicará el siguiente capitulo de este trabajo. No sin antes pasar por la elección del lenguaje de programación que se utilizará para desarrollar dicho software.

3.11 ELECCIÓN DEL LENGUAJE DE PROGRAMACIÓN

La elección del lenguaje de programación se hizo de acuerdo a las necesidades que se tenían para realizar este trabajo de tesis. Como se señaló en la definición del problema, se requería de compatibilidad en la mayor parte de sistemas de computo utilizados y facilidades para la creación de programas en ambiente visual y en plataforma Windows.

Ante estas necesidades y después de hacer una evaluación de las bondades de los lenguajes que teníamos disponibles y de los inconvenientes que algunos de ellos presentaban, se optó por trabajar en Visual Basic, por ser un lenguaje de programación diseñado para desarrollo de aplicaciones en el entorno gráfico Windows, y que por la misma razón, se da un muy buen aprovechamiento de las posibilidades de Windows en cuanto a intercambio de información, de sus librerías, de sus drivers y controladores.

Por otra parte, Visual Basic es una herramienta de diseño de aplicaciones para Windows, en la que estas se desarrollan en una gran parte a partir del diseño de una interfaz gráfica, además Visual Basic es un lenguaje estructurado, lo que permite crear programas modularmente, mediante subrutinas y módulos, los cuales resultan competitivos con los creados en otros lenguajes de alto nivel, por lo cual no se perdía ninguno de los beneficios que nos ofrecían los otros lenguajes y se obtenían otros.

CAPITULO IV

EL LENGUAJE DE PROGRAMACIÓN VISUAL BASIC

4.1 ANTECEDENTES HISTÓRICOS

El lenguaje de programación BASIC (Beginner's All purpose Symbolic Instruction Code) nació en el año 1964 como una herramienta destinada a principiantes, buscando una forma sencilla de realizar programas, empleando un lenguaje casi igual al usado en la vida ordinaria (en inglés), y con instrucciones muy sencillas y escasas. Teniendo en cuenta el año de su nacimiento, este lenguaje cubría casi todas las necesidades para la ejecución de programas. Téngase en cuenta que las máquinas existentes en aquella época estaban estrenando los transistores como elementos de conmutación.

La evolución del BASIC por los años 70 fue escasa, dado el auge que tomaron en aquella época lenguajes de alto nivel como el FORTRAN y el COBOL. En 1978 se definió una norma para unificar los Basic's existentes creándose la normativa BASIC STANDARD.

Con la aparición de las primeras computadoras personales, dedicadas comercialmente al usuario particular, allá por la primera mitad de los años ochenta, el BASIC resurgió como lenguaje de programación pensado para principiantes, y muchas de estas pequeñas computadoras personales lo usaban como único sistema operativo (Sinclair, Spectrum, Amstrad)

Con la popularización de la PC, salieron varias versiones del BASIC que funcionaban en este tipo de computadoras (Versiones BASICA, GW BASIC), pero todas estas versiones del BASIC no hicieron otra cosa que terminar de dispersar este lenguaje. Los programadores profesionales no llegaron a utilizarlo, ante las desventajas de este lenguaje respecto a otras herramientas de programación (PASCAL, C, CLIPPER). El BASIC con estas versiones para PC llegó incluso a perder crédito entre los profesionales de la informática.
Las razones para ello eran obvias:

- No era un lenguaje estructurado.
- No existían herramientas de compilación fiables.
- No disponía de herramientas de intercambio de información.
- No tenía librerías.

Tal fue ese abandono por parte de los usuarios, que la aparición del Quick BASIC de Microsoft, una versión ya potente del BASIC, que corregía casi todos los defectos de las versiones anteriores pasó prácticamente inadvertida, a no ser porque las últimas versiones del sistema operativo MS-DOS incluían una versión de Quick BASIC algo recortada (Q-Basic) como un producto mas dentro de la amplia gama de ficheros ejecutables que acompañan al sistema operativo, y aprovecha el editor de textos del mismo (Cada vez que se llama al EDIT estamos corriendo el editor del Q-Basic).

Esta versión del popular BASIC ya es un lenguaje estructurado, lo que permite crear programas modularmente, mediante subrutinas y módulos, capaz de crear programas ya competitivos con otros lenguajes de alto nivel. Sin embargo llegaba tarde, pues los entornos MS-DOS estaban ya superados por el entorno gráfico Windows.

Sin embargo algo había en el BASIC que tentaba a superarse: su gran sencillez de manejo. Si a esto se le añade el entorno gráfico Windows, el aprovechamiento al máximo de las posibilidades de Windows en cuanto a intercambio de información, de sus librerías, de sus drivers y controladores, manejo de bases de datos, etc., el producto resultante puede ser algo que satisfaga todas las necesidades de programación en el entorno Windows. La suma de todas estas cosas es VISUAL BASIC. Esta herramienta conserva del BASIC de los años 80 únicamente su nombre y su sencillez, y tras su lanzamiento al mercado, la aceptación a nivel profesional hizo borrar por fin el "mal nombre" asociado a la palabra BASIC.

Desde su salida al mercado, cada versión supera y mejora la anterior. Dados los buenos resultados a nivel profesional de este producto, y el apoyo prestado por el fabricante para la formación de programadores, Visual Basic se ha convertido en la primera herramienta de desarrollo de aplicaciones en entorno Windows.

Es obligado decir sin embargo, que sigue siendo BASIC. No se pueden comparar sus prestaciones con otros lenguajes cuando deseamos llegar al fondo de la máquina y controlar uno a uno sus registros. No es ese el fin perseguido con Visual Basic y si es necesario llegar a esas precisiones será necesario utilizar otro lenguaje que permita bajar el nivel de programación. (Visual-C), o realizar librerías (DLLs) que lo hagan. En la mayor parte de las aplicaciones, las herramientas aportadas por Visual Basic son mas que suficiente para lograr un programa fácil de realizar y de altas prestaciones.

4.2 CARACTERÍSTICAS GENERALES DE VISUAL BASIC

Visual Basic es una herramienta de diseño de aplicaciones para Windows, en la que estas se desarrollan en una gran parte a partir del diseño de una interfaz gráfica. En una aplicación Visual Basic, el programa está formado por una parte de código puro, y otras partes asociadas a los objetos que forman la interfaz gráfica.

Es por tanto un termino medio entre la programación tradicional, formada por una sucesión lineal de código estructurado, y la programación orientada a objetos. Combina ambas tendencias.

La creación de un programa bajo Visual Basic lleva los siguientes pasos:

- **Creación de una interfaz de usuario.** Esta interfaz será la principal vía de comunicación hombre máquina, tanto para salida de datos como para entrada. Será necesario partir de una ventana - Formulario - a la que le iremos añadiendo los controles necesarios.

- **Definición de las propiedades de los controles - Objetos - que hayamos colocado en ese formulario**. Estas propiedades determinarán la forma estática de los controles, es decir, como son los controles y para qué sirven.

- **Generación del código asociado a los eventos que ocurran a estos objetos.** A la respuesta a estos eventos (click, doble click, una tecla pulsada, etc.) le llamamos Procedimiento, y deberá generarse de acuerdo a las necesidades del programa.

- **Generación del código del programa.** Un programa puede hacerse solamente con la programación de los distintos procedimientos que acompañan a cada objeto. Sin embargo, Visual Basic ofrece la posibilidad de establecer un código de programa separado de estos eventos. Este código puede introducirse en unos bloques llamados Módulos, en otros bloques llamados Funciones, y otros llamados Procedimientos. Estos Procedimientos no responden a un evento acaecido a un objeto, sino que responden a un evento producido durante la ejecución del programa.

4.3 CÓMO FUNCIONA WINDOWS

Todos vivimos en un mundo de eventos. Raras veces nos detenemos a pensar cómo trabaja el reloj despertador, cómo el refrigerador mantiene las cosas frías o cómo la gasolina se convierte en energía. Nada es más natural que la

relación entre los objetos y los eventos que se generan cuando se interactúa con ellos. En Ese sentido un programa escrito para una *Interfaz Gráfica de Usuario (GUI [Graphical User interface])* como Windows, debería ser igualmente familiar y natural de usar.

En Windows, casi todo lo que se ve en la pantalla son ventanas. Estas ventanas tienen propiedades únicas que las distinguen entre sí, como pueden ser la clase o tipo, el tamaño, color, posición, etc.

Visual Basic hace una clara distinción de unas ventanas de otras. Por un lado están las ventanas principales que en Visual Basic son los *formularios*, y por otro están los *controles*, que son el resto y que se incluyen dentro de los formularios.

Windows asigna a cada ventana un identificador único y continuamente estará rastreándolas en busca de signos de actividad, o lo que es lo mismo, *eventos*. Los eventos ocurren cuando el usuario realiza alguna acción (ej. pulsar el ratón), porque estaban programados o debidos a la actividad de otras ventanas.

Cada vez que ocurre un evento, el sistema operativo recibe un mensaje indicando lo que ha ocurrido, lo procesa y lo manda a todas las ventanas las cuales actuarán en consecuencia si necesitan hacer algo para ese evento. Las ventanas pueden tener asociadas acciones a muchos tipos de mensajes distintos.

Por ejemplo, cuando se hace click en un botón, Windows recibe el mensaje de que el ratón ha sido pulsado y en que posición de la pantalla, entonces se encarga de procesar la información para descubrir cual ha sido la ventana sobre la cual se hizo el click y envía un mensaje indicándoselo a todas las demás, incluida la que se quiere accionar. Ahora las ventanas pueden actuar en consecuencia con el mensaje que les ha sido entregado. En este caso le importará especialmente la que tiene que dibujar la imagen de botón pulsado y ejecutar el código que tenga asociado a este evento.

4.4 EVENTOS DE PROGRAMACIÓN

Una aplicación típica de Windows presenta una o más pantallas llenas de objetos con los cuales interactuará el usuario para determinar el flujo del programa. En Visual Basic existen herramientas para el trazado de botones oprimibles, cuadros de lista, cuadros combinados desplegabas, casillas de verificación, botones de opción (también llamados botones de radio), barras de desplazamiento, etc. Cada uno de estos objetos muestra su propio comportamiento predefinido. Cuando se hace click en él, el botón oprimible "se

hunde y vuelve a salir", y cuando el usuario teclea, un cuadro de texto recibe y despliega cada tecleo.

Sin embargo, para producir un programa complejo se debe aumentar o sobrescribir este comportamiento limitado. Esto se puede hacer escribiendo el código personalizado y conectándolo con el evento significativo del objeto. Por ejemplo, un evento de click en un botón de un comando podría contener el comando individual End en Visual Basic. Cuando se hace click en el botón, se dispara el código asociado con su evento Click, y, en este caso, el programa termina.

Los objetos y eventos de la programación están íntimamente relacionados, como sucede con los objetos y eventos en la vida real. Los eventos tienen lugar como resultado de la acción del usuario o del código del programa, o pueden ser activados por el sistema. La mayoría de los objetos responden a un número de eventos generados por el usuario, incluyendo un click de ratón o doble click, la opresión de una tecla o al arrastrar y soltar. Además, los objetos pueden reconocer eventos del sistema, incluyendo los eventos de temporizador y de carga, activando el primero a intervalos especificados y el segundo cuando un objeto, como una forma, es cargado por primera vez en la memoria.

4.4.1 OBJETOS Y EVENTOS

La mayoría de las aplicaciones Windows, así como otros programas, emplean una Interfaz Gráfica de Usuario, consistente de una o más pantallas llenas de objetos, menús, botones, listas desplegables, cuadros de edición, todos inactivos hasta que el usuario provoca un evento en la forma de una opresión de tecla, un click del ratón, un toque de dedo o un comando de voz. Una vez que el evento ocurre, el usuario espera que cada objeto se comporte de una manera confiable (y predecible), una selección de menú abre un cuadro de diálogo, una serie de teclas oprimidas son capturados y desplegados en un control de texto y un click de ratón sobre un botón OK cierra el cuadro de diálogo para que el procesamiento pueda continuar.

Lo anterior hace que la programación orientada a objetos y manejada por eventos sea perfecta para el desarrollo de las aplicaciones Windows. Visual Basic proporciona un ambiente de desarrollo donde el trabajo con tales objetos y eventos llega a ser un proceso directo y bien estructurado.

4.5 MÉTODOS Y PROPIEDADES

4.5.1 PROPIEDADES

Los objetos del mundo real están cargados de propiedades. Por ejemplo una prenda de vestir es de un color, tamaño y peso determinados, y está hecha de un material especifico con su propia textura particular. De igual manera, un objeto de programación también se define por sus propios conjuntos de propiedades. La figura 4.1 muestra el cuadro de propiedades de Visual Basic.

Hablando estrictamente, *una propiedad* es un atributo nominal de un objeto de programación. Las propiedades definen las características del objeto, como el tamaño, el color, la localización en la pantalla, o algunas veces, la manera en la cual se comporta el objeto, por ejemplo, si un cuadro de texto acepta o no lineas múltiples o si un elemento del menú está actualmente activado.

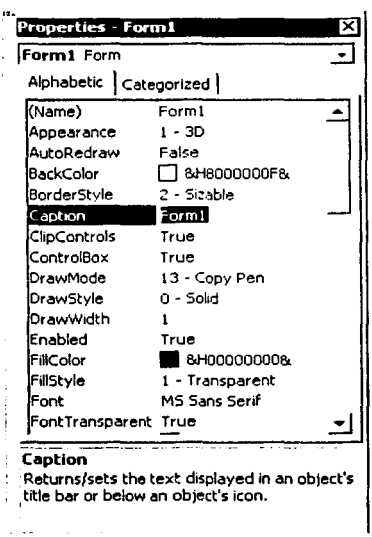


Figura 4.1 Ventana de propiedades en Visual Basic

A continuación se muestra una tabla que enlista algunas de las propiedades más importantes y su utilidad.

Propiedad	Utilidad
(Name)	Es el nombre que utilizarás en el código para referirte a un control. Microsoft recomienda que se antepongan tres letras identificativas del tipo de control antes del nombre. Por ejemplo: • los botones serán "cmdBoton1" o "cmdAceptar" (cmd es el diminutivo de CommandButton) • los recuadros de texto "txtNombre" (txt) • las listas "lstUsuarios" (lst)
BackColor	Color de fondo del botón. Modificando esta propiedad se puede dar color a los programas.
Caption	Es el texto que aparece en el botón
Enabled	Los controles pueden estar activados o desactivados. Si están desactivados el usuario no puede interactuar con ellos.
Font	Tipo de letra del texto que aparecerá en el control
Height	Altura del control
Left	Distancia del control a la parte izquierda del formulario que lo contiene
TabStop	Si está a "true" (verdadero) permite que el usuario seleccione el control pulsando tab
Top	Distancia del control a la parte superior del formulario que lo contiene
Visible	Si está a "true" el control aparecerá en el formulario en tiempo de ejecución, en otro caso no se podrá ver en ese momento.
Width	Ancho del control

Tabla 4.1 Algunas de las propiedades más importantes de Visual Basic.

Todas estas propiedades se pueden modificar tanto en tiempo de diseño como en tiempo de ejecución (mediante código).

4.5.2 MÉTODOS

Toda descripción completa de un objeto debe incluir no solo sus características físicas sino también una descripción de lo que hace. Por ejemplo los métodos de un animal pueden ser moverse, dormir, jugar, comer y respirar.
De manera similar, en la programación orientada a objetos un *método es* un procedimiento conectado o integrado, un bloque de código que puede llamarse para impartir alguna acción a un objeto particular. A diferencia de otros procedimientos o funciones del lenguaje Visual Basic, los métodos requieren un objeto que les dé un contexto.

En pocas palabras, mientras las propiedades tienden a describir un objeto, los métodos permiten que el objeto haga algo. Las propiedades son datos, los métodos son código. En la gramática de la programación orientada a objetos, los objetos son sustantivos, las propiedades son adjetivos y los métodos son verbos.

4.5.3 VENTAJAS DE USAR PROPIEDADES Y MÉTODOS

Por medio de la manipulación de objetos, la puesta de propiedades y la llamada de métodos en el tiempo de ejecución, es posible controlar partes separadas de un programa sin interferir con el resto de la aplicación. Lo que es más, las técnicas objeto-propiedad-método hacen que el desarrollo de las aplicaciones sea más fácil, proporcionando herramientas que se adaptan más de cerca a la manera en que las personas piensan acerca del mundo.

4.6 LAS FORMAS EN VISUAL BASIC

Las formas o diálogos son las ventanas principales de toda aplicación de Visual Basic. Pueden ser de distintas formas y tamaños pero generalmente son rectangulares, con una barra superior donde está escrito el título a la izquierda y tres botones a la derecha, uno para minimizarla, otro para expandirla y otra para cerrarla.

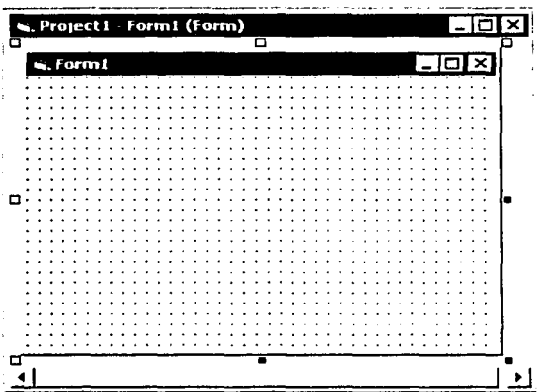


Figura 4.2 Forma de Visual Basic lista para la edición

Cada control o cada ventana tiene ciertas propiedades que indican cómo debe representarse gráficamente y algunos patrones de comportamiento cuando esté en ejecución. Éstas propiedades, aparecen en una cuadrícula que suele estar situada en la parte inferior derecha del entorno de trabajo, en la primera columna aparece el nombre y en la segunda el valor que toma.

4.7 CONTROLES

Aunque con las formas se pueden hacer algunas cosas, lo que realmente nos permite crear aplicaciones de forma rápida con Visual Basic son los controles. Los controles son diferentes elementos que se pueden incluir en la forma y que tienen una funcionalidad específica dependiendo del tipo que sean.

Visual Basic incluye en todos los proyectos los controles más usados por las aplicaciones de Windows como pueden ser: los botones, los cuadros de texto, cuadros de selección, etc. Pueden ser encontrados en la barra de herramientas que aparece a la izquierda del entorno de desarrollo.

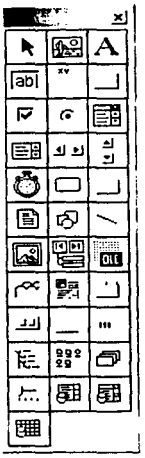


Figura 4.3 Barra de Herramientas con controles de Visual Basic

CAPITULO V

DESARROLLO DEL SOFTWARE

5.1 ELEMENTOS BÁSICOS

Una vez que la elección de lenguaje de programación fue llevada a cabo, se conocieron los métodos de programación y la estructura interna del PLM, así como la programación en SIIL1, se aboca el trabajo al desarrollo del software, el cual comprendería la programación de un ambiente visual, que nos debe permitir programar el Programador Lógico Modular (PLM) mediante el lenguaje de Diagrama o Plano de Funciones, visto anteriormente en el capitulo II, de donde es posible recordar que el diagrama de funciones es un lenguaje gráfico que permite programar elementos que aparecen como bloques para ser cableados entre si de forma análoga al esquema de un circuito.

Para el caso particular del PLM, es necesario recordar también que cuenta con Módulos Funcionales que de acuerdo al SIIL1 (Software de Interpretación, de Instrucciones Lógicas), lenguaje propio del PLM para utilización por parte del usuario final, se deben declarar ya sea en el subprograma principal o en el subprograma temporizado.

Recordemos que un programa en SIIL1, está dividido en dos subprogramas denominados *subprograma principal* y *subprograma temporizado*, y que en la parte del subprograma principal se deben declarar tanto el código correspondiente a módulos que realizan compuertas lógicas y Flip-Flops como el correspondiente a módulos auxiliares, además de código de inicialización, de lectura y escritura de buffers y de salto a la posición del inicio del ciclo de barrido.

En lo que toca al subprograma temporizado, se debe recordar que en esta parte se declara el código asociado a los módulos que realizan temporizadores, secuenciadores, contadores de eventos y módulos auxiliares, además de haber código propio del manejo de la interrupción OC2 y de actualización de buffers testigo de estado anterior.

Después de lo anterior, únicamente hace falta recordar cuales son lo módulos que se deben declarar en cada una de las partes, lo cual se muestra en la siguiente tabla.

MÓDULOS DE SUBPROGRAMA PRINCIPAL	MÓDULOS DE SUBPROGRAMA TEMPORIZADO
1. Seguidor lógico 2. Inversor lógico 3. Compuertas lógicas de 3 y 4 entradas: • AND • OR • NAND • NOR • OR EXCLUSIVA • OR EXCLUSIVA NEGADA 4. Flip-Flop Asíncrono RS (FFARS)	1. Compuertas lógicas de 2 entradas: • AND • OR • NAND • NOR • OR EXCLUSIVA • OR EXCLUSIVA NEGADA 2. Contador de eventos. 3. Secuenciador de estados. 4. Temporizador monodisparo. 5. Temporizador monodisparo 2° tipo. 6. Temporizador con retardo a la activación. 7. Temporizador astable. 8. Temporizador multipulso. 9. Temporizador generador de pulsos con el reloj de tiempo real (6 clases).

Tabla 6.1 Módulos del PLM que se declaran en el Subprograma Principal y en el Subprograma Temporizado.

Se debe recordar que el PLM también cuenta con módulos auxiliares, los cuales se enlistan a continuación:

5.1.1 MÓDULOS AUXILIARES

1. Módulo mensajero con capacidad de generar texto estático y/o dinámico en la Unidad Desplegadora (UD).
2. Módulo con capacidad para generar mensajes de alarma.
3. Módulo observador del estado de contadores de eventos.
4. Módulo manejador del reloj de tiempo real.
5. Módulo DESP, que copia a la UD el contenido de su buffer asociado.
6. Módulo MANDESP, que permite observar para verificación, el texto asociado con módulos de tipo mensajero.

5.2 GENERALIDADES SOBRE EL SOFTWARE

En este capitulo se intenta dar una breve explicación de lo que es el software que se realizó y de su desarrollo sin entrar a detalle de la programación en su estructura, pero sí dejando con los ejemplos una idea clara de lo que se hizo ya que el explicar todas las funciones, la utilización de las variables y otros temas de la programación extendería este capitulo de manera tal que sería poco digeribile y no se expondría claramente lo que es el software que se ha creado, sin embargo, se exponen algunas partes que se consideran esenciales en la ejecución del programa, con el fin de que se entienda el funcionamiento del mismo durante la ejecución.

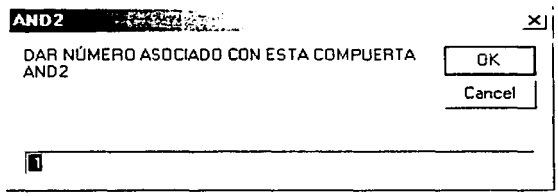
Dentro del desarrollo del software se debe tomar en cuenta que el propósito fue elaborar un programa que permita programar el PLM mediante el lenguaje conocido como *Diagrama o Plano de Funciones*. Para tal efecto se tendría que desarrollar un programa que nos permitiera visualizar en pantalla los símbolos de las diferentes funciones o módulos lógicos con los que es capaz de trabajar el PLM.

Lo anterior nos lleva a tener que trabajar módulos lógicos con un botón de comando de Visual Basic (VB) para cada uno, en el cual se especificarían las características necesarias para cada uno, como son variables de entrada y salida además de contar con otra parte del programa en la forma de trabajo que nos permitiera formar de manera gráfica cada uno de los módulos para poder visualizarlo en ella misma con el evento de mousedown y que además permitiera asignar la numeración que identifique a la(s) entrada(s) correspondiente(s) y la salida respectiva para cada módulo, las cuales serían introducidas por el usuario a petición del programa por medio de un *InputBox*, teniendo como variables locales para cada módulo aquellas que sirvan para poderle asignar la(s) entrada(s) y las salida(s). Dichas variables locales se declaran en el programa como variables del tipo *string*.

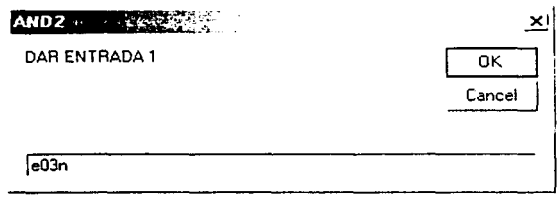
5.3 DESARROLLO DEL SOFTWARE

5.3.1 NOMENCLATURA PARA DECLARACIÓN DE ENTRADA(S) Y SALIDA(S)

Cuando el usuario selecciona con un click el módulo que desea utilizar, el programa comienza solicitándole algunos datos con el fin de poder definir completamente dicho módulo, por ello es importante que el usuario sepa que existe una nomenclatura específica para poder asignar la(s) entrada(s) y salida(s) y que antes de ingresar cualquiera de éstas el programa le pide mediante un *InputBox* como el que se muestra en la figura 5.1 que ingrese en un el número asociado con el tipo de módulo funcional que se está utilizando, que permite controlar los de módulos de un mismo tipo que se están utilizando simultaneamente. Cabe señalar que el programa tiene un contador que pone de facto un número, aunque el usuario lo puede cambiar.



(a)



(b)

Figura 5.1 Ventanas para el caso particular de un módulo AND2.

a. Ventana que se despliega para que el usuario introduzca el número de módulo con el que desea trabajar.
b. Ventana en la que el usuario puede ingresar la notación que identifique una entrada.

Para el caso de las entradas, estas pueden empezar con una letra "e" o una letra "i" mayúscula o minúscula seguida por dos dígitos que le permitirán dar un número de identificación a cada entrada de una función y para el caso de la salida, puede empezar con una letra "i" o con una letra "s" mayúscula o minúscula seguida por dos dígitos que también permiten dar un número de identificación a la salida. En el caso que la entrada que el usuario proporcione no sea válida, el programa simplemente le volverá a pedir que introduzca el valor requerido anteriormente.

Así mismo, a cada tipo de modulo que se realizó, se le asignó un número, el cual serviría más tarde para identificarlo y poder saber de que tipo de módulo se trata, ya que con el mismo número se identificaría en un *case* que sirve para volver a dibujar los módulos (la utilidad de esto se explicará más adelante) y esto se hace teniendo por separado los módulos según el subprograma que le corresponda de acuerdo a la tabla que se mencionó anteriormente.

Todos estos detalles se explican a continuación para cada módulo:

5.3.2 MODULOS FUNCIONALES

5.3.2.1 Módulos AND2, AND3 y AND4

Estos módulos se refieren a la compuerta lógica AND de dos, tres y cuatro entradas, cada uno se maneja con un botón de comando que tiene un código asociado que dependiendo del caso de que se trate cambia principalmente en el número de variables y por consiguiente en el número de bloques de instrucciones para el manejo de las mismas. El cambio es únicamente en el número de entradas ya que todas tienen una sola salida.

A continuación se muestra el código asociado al botón de comando del Módulo AND4:

```
Private Sub command3_Click()
exp.defselmod 3

Dim ent3 As String
Dim ent2 As String
Dim ent1 As String
Dim ent0 As String
Dim sal0 As String
cadbi = 0

nuand4 = nuand4 + 1
strnuand4 = Str(nuand4)
snuand4 = InputBox("DAR NÚMERO ASOCIADO CON ESTA COMPUERTA AND4", "AND4", strnuand4)
```

```
nuand4 = Val(snuand4)
exp. defnumod nuand4

capte3:
ent3 = InputBox("DAR ENTRADA 3 ", "AND4")
ent3 = UCase(ent3)
If varval(ent3) = False Then
GoTo capte3
End If

If Right(ent3, 1) <> "N" Then
cadbi = cadbi + 8
End If
exp. defent ent3, 3
capte2:
ent2 = InputBox("DAR ENTRADA 2 ", "AND4")
ent2 = UCase(ent2)
If varval(ent2) = False Then
GoTo capte2
End If

If Right(ent2, 1) <> "N" Then
cadbi = cadbi + 4
End If
exp. defent ent2, 2

captel:
entl = InputBox("DAR ENTRADA 1 ", "AND4")
entl = UCase(entl)
If varval(entl) = False Then
GoTo captel
End If

If Right(entl, 1) <> "N" Then
cadbi = cadbi + 2
End If
exp. defent entl, 1

capte0:
ent0 = InputBox("DAR ENTRADA 0 ", "AND4")
ent0 = UCase(ent0)
If varval(ent0) = False Then
GoTo capte0
End If

If Right(ent0, 1) <> "N" Then
cadbi = cadbi + 1
End If
exp. defent ent0, 0

captsO:
salO = InputBox("DAR SALIDA ", "AND4")
salO = UCase(salO)
```

```
If varval(sal0) = False Then
GoTo capts0
End If

If Right(sal0, 1) = "N" Then
GoTo capts0
End If
If Left(sal0, 1) = "E" Then
GoTo capts0
End If

exp.defsal sal0, 0
exp.defcadbin cadbi

End Sub
```

Como se estableció previamente, en el caso de los módulos AND2 y AND3, el código asociado a los botones de comando correspondientes tiene algunas variaciones con respecto al mostrado anteriormente, dichas variaciones son principalmente en los bloques de código para el manejo de variables, ya que en el caso de la AND2 y AND3, son menos las entradas que se tienen.

El código anterior muestra la asignación del número de tipo de módulo, el cual en este caso es el "3" (exp.defselmod 3) así como también se asignan las variables, de igual manera se muestran las instrucciones para solicitar y asignar el número asociado de compuerta de tipo AND de cuatro entradas que se quiera manejar y los bloques de instrucciones para las cuatro entradas y la salida.

Al final de este código se encuentra una instrucción con notación de punto, que sirve para el manejo de la cadena binaria. Más adelante se explicará esto con mayor detalle.

En la figura 5.2 se muestran ejemplos de las compuertas lógicas AND de dos, tres y cuatro entradas, donde se puede ver la entrada E05 de la compuerta AND de tres entradas previamente negada, la cual tuvo que ser declarada como E05N.

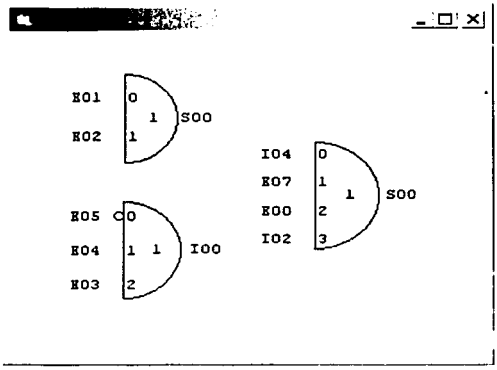


Figura 5.2 Ejemplos gráficos de las compuertas lógicas AND de dos, tres y cuatro entradas.

5.3.2.2 Módulos OR2, OR3 y OR4

Los módulos OR2, OR3 y OR4 se refieren a la compuerta lógica OR de dos, tres y cuatro entradas, las cuales al igual que las anteriores, se manejan para cada caso con un botón de comando que tiene un código asociado que cambia dependiendo del caso de que se trate, principalmente en el número de variables de entrada y por ende en el número de bloques de instrucciones para el manejo de éstas. El cambio es nuevamente en el número de entradas ya que todas tienen una sola salida.

A continuación se muestra el código asociado al botón de comando del Módulo OR4:

```
Private Sub Command6_Click()
exp.defselmod 6

Dim ent3 As String
Dim ent2 As String
Dim ent1 As String
Dim ent0 As String
Dim sal0 As String
cadbi = 0
nuor4 = nuor4 + 1
strnuor4 = Str(nuor4)
snuor4 = InputBox("DAR NÚMERO ASOCIADO CON ESTA COMPUERTA OR4", "OR4", strnuor4)
nuor4 = Val(snuor4)
exp.defnumod nuor4
```

```
capte3:
ent3 = InputBox("DAR ENTRADA 3 ", "OR4")
ent3 = UCase(ent3)
If varval(ent3) = False Then
GoTo capte3
End If

If Right(ent3, 1) <> "N" Then
cadbi = cadbi + 8
End If
exp.defent ent3, 3
capte2:
ent2 = InputBox("DAR ENTRADA 2 ", "OR4")
ent2 = UCase(ent2)
If varval(ent2) = False Then
GoTo capte2
End If

If Right(ent2, 1) <> "N" Then
cadbi = cadbi + 4
End If
exp.defent ent2, 2

captel:
entl = InputBox("DAR ENTRADA 1 ", "OR4")
entl = UCase(entl)
If varval(entl) = False Then
GoTo captel
End If

If Right(entl, 1) <> "N" Then
cadbi = cadbi + 2
End If
exp.defent entl, 1

capteO:
entO = InputBox("DAR ENTRADA O ", "OR4")
entO = UCase(entO)
If varval(entO) = False Then
GoTo capteO
End If
If Right(entO, 1) <> "N" Then
cadbi = cadbi + 1
End If
exp.defent entO, O

captsO:
salO = InputBox("DAR SALIDA ", "OR4")
salO = UCase(salO)
If varval(salO) = False Then
GoTo captsO
End If
If Right(salO, 1) = "N" Then
```

```
GoTo captsO
End If
If Left(salO, 1) = "E" Then
GoTo captsO
End If

exp.defsal salO, 0
exp.defcadbin cadbi
End Sub
```

Como en el caso anterior, para los módulos OR2 y OR3, el código asociado a los botones de comando correspondientes tiene algunas variaciones con respecto al de la OR4, mostrado anteriormente, dichas variaciones son principalmente en los bloques de código para el manejo de variables, ya que en el caso de la OR2 y OR3, tienen únicamente dos y tres entradas respectivamente.

El código anterior muestra la asignación del número de tipo de módulo. el cual en este caso es el "6" (exp.defselmod 6) así como también se asignan las variables y sus tipos y de igual manera se muestran las instrucciones para el manejo del número asociado de compuerta OR de cuatro entradas que se quiera usar y los bloques de instrucciones para manejar las cuatro entradas y la salida. En la figura 5.3 se muestran ejemplos de las compuertas lógicas OR de dos, tres y cuatro entradas, con algunas de ellas previamente negadas.

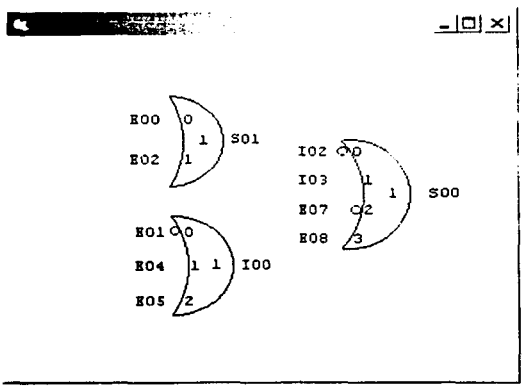


Figura 5.3 Ejemplos de compuertas lógicas OR de dos, tres cuatro entradas, con algunas de ellas previamente negadas.

5.3.2.3 Módulos NAND2, NAND3 y NAND4

En el caso de los módulos NAND2, NAND3 y NAND4, estos se refieren a la compuerta lógica NAND de dos, tres y cuatro entradas, los cuales también se manejan con un botón de comando para cada uno, que tiene un código asociado que varía dependiendo del caso de que se trate, principalmente en el número de variables de entrada y al mismo tiempo en el número de bloques de instrucciones para su manejo. Como es de notarse, el cambio ocurre en el número de entradas ya que todas siguen teniendo una sola salida.

A continuación se muestra el código asociado al botón de comando del Módulo NAND3:

```
Private Sub Command8_Click()
exp.defselmod 8
Dim ent2 As String
Dim ent1 As String
Dim ent0 As String
Dim sal0 As String
cadbi = 0

nunand3 = nunand3 + 1
strnunand3 = Str(nunand3)
snunand3 = InputBox("DAR NÚMERO ASOCIADO CON ESTA COMPUERTA NAND3", "NAND3", strnunand3)
nunand3 = Val(snunand3)
exp.defnumod nunand3

capte2:
ent2 = InputBox("DAR ENTRADA 2 ", "NAND3")
ent2 = UCase(ent2)
If varval(ent2) = False Then
GoTo capte2
End If

If Right(ent2, 1) <> "N" Then
cadbi = cadbi + 4
End If
exp.defent ent2, 2

captel:
ent1 = InputBox("DAR ENTRADA 1 ", "NAND3")
ent1 = UCase(ent1)
If varval(ent1) = False Then
GoTo captel
End If

If Right(ent1, 1) <> "N" Then
cadbi = cadbi + 2
End If
exp.defent ent1, 1
capte0:
```

```
ent0 = InputBox("DAR ENTRADA 0 ", "NAND3")
ent0 = UCase(ent0)
If varval(ent0) = False Then
GoTo capte0
End If

If Right(ent0, 1) <> "N" Then
cadbi = cadbi + 1
End If
exp.defent ent0, 0

capts0:
sal0 = InputBox("DAR SALIDA ", "NAND3")
sal0 = UCase(sal0)
If varval(sal0) = False Then
GoTo capts0
End If

If Right(sal0, 1) = "N" Then
GoTo capts0
End If
If Left(sal0, 1) = "E" Then
GoTo capts0
End If

exp.defsal sal0, 0
exp.defcadbin cadbi
End Sub
```

En el código anterior es posible observar la línea de código para la asignación del número de tipo de módulo, el cual en este caso es el "8" (`exp.defselmod 8`) así como también se asignan las variables y tipos de las mismas. También se muestran las instrucciones para el manejo del número asociado de compuerta NAND de cuatro entradas que se quiera utilizar y los bloques de instrucciones para las cuatro entradas y la salida.

En este caso, existen variaciones del código mostrado con respecto al código asociado de los módulos NAND2 y NAND4, dichas variaciones son de manera esencial en los bloques de código para el manejo de variables, puesto que las compuertas NAND2 y NAND4 tienen menos y más entradas respectivamente. Por ejemplo, para el caso de la NAND4, sería necesario agregar el siguiente bloque de instrucciones además de modificar variables entre otros.

```
capte3:
ent3 = InputBox("DAR ENTRADA 3 ", "NAND4")
ent3 = UCase(ent3)
If varval(ent3) = False Then
GoTo capte3
```

También en este caso se tiene la instrucción con notación de punto para el manejo de la cadena binaria.

En la figura 5.4 se muestran ejemplos de las compuertas lógicas NAND de dos, tres y cuatro entradas, donde se puede observar la compuerta NAND de tres entradas con dos de ellas previamente negadas.

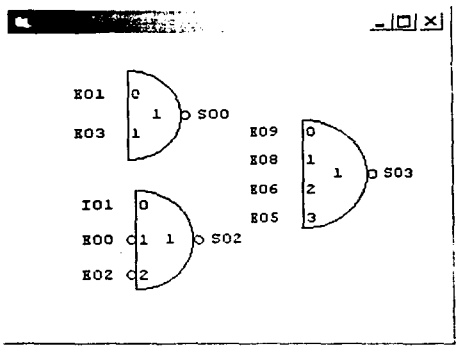


Figura 5.4 Compuertas lógicas NAND de dos, tres y cuatro entradas, con algunas de entradas previamente negadas.

5.3.2.4 Módulos NOR2, NOR3 y NOR4

Estos módulos se refieren a la compuerta lógica NOR de dos, tres y cuatro entradas, los cuales también se manejan con un botón de comando que tiene un código asociado con variaciones entre cada uno. El cambio es únicamente en el número de entradas ya que todas tienen una sola salida.

Como ejemplo se muestra el código asociado al botón de comando del Módulo AND4:

```
Private Sub Command12_Click()
exp.defselmod 12

Dim ent3 As String
Dim ent2 As String
```

```
Dim ent1 As String
Dim ent0 As String
Dim sal0 As String
cadbi = 0

nunor4 = nunor4 + 1
strnunor4 = Str(nunor4)
snunor4 = InputBox("DAR NÚMERO ASOCIADO CON ESTA COMPUERTA NOR4", "NOR4", strnunor4)
nunor4 = Val(snunor4)
exp.defnumod nunor4


capte3:
ent3 = InputBox("DAR ENTRADA 3 ", "NOR4")
ent3 = UCase(ent3)
If varval(ent3) = False Then
GoTo capte3
End If

If Right(ent3, 1) <> "N" Then
cadbi = cadbi + 8
End If
exp.defent ent3, 3

capte2:
ent2 = InputBox("DAR ENTRADA 2 ", "NOR4")
ent2 = UCase(ent2)
If varval(ent2) = False Then
GoTo capte2
End If

If Right(ent2, 1) <> "N" Then
cadbi = cadbi + 4
End If
exp.defent ent2, 2

captel:
ent1 = InputBox("DAR ENTRADA 1 ", "NOR4")
ent1 = UCase(ent1)
If varval(ent1) = False Then
GoTo captel
End If

If Right(ent1, 1) <> "N" Then
cadbi = cadbi + 2
End If
exp.defent ent1, 1

capte0:
ent0 = InputBox("DAR ENTRADA 0 ", "NOR4")
ent0 = UCase(ent0)
If varval(ent0) = False Then
GoTo capte0
```

```
End If

If Right(ent0, 1) <> "N" Then
cadbi = cadbi + 1
End If
exp.defent ent0, 0

capts0:
sal0 = InputBox("DAR SALIDA ", "NOR4")
sal0 = UCase(sal0)
If varval(sal0) = False Then
GoTo capts0
End If
If Right(sal0, 1) = "N" Then
GoTo capts0
End If
If Left(sal0, 1) = "E" Then
GoTo capts0
End If
exp.defsal sal0, 0
exp.defcadbin cadbi

End Sub
```

Como en el caso anterior, para los módulos NOR2 y NOR3, el código asociado a los botones de comando correspondientes tiene algunas variaciones con respecto al de la OR4, mostrado anteriormente, principalmente en los bloques de código para el manejo de variables, ya que en el caso de la OR2 y OR3, tienen únicamente dos y tres entradas respectivamente.

En el código anterior se muestra la asignación del número de tipo de módulo, el cual en este caso es el "12" (`exp.defselmod 12`) de igual manera se asignan las variables, se muestran las instrucciones para solicitar e introducir el número asociado de compuerta de tipo NOR de cuatro entradas que se quiera manejar y los bloques de instrucciones para las cuatro entradas y la salida.

En la figura 5.5 se muestran ejemplos de las compuertas lógicas NOR de dos, tres y cuatro entradas.

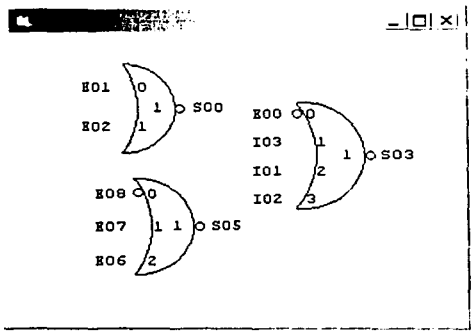


Figura 5.5 Ejemplos de las compuertas lógicas NOR de dos, tres y cuatro entradas.

5.3.2.5 Módulos INV y SEG

Estos módulos se refieren a un inversor (INV) y a un seguidor (SEG), éstos se manejan con un botón de comando al igual que todas las compuertas anteriormente mencionadas y también tienen un código asociado.

Como ejemplo se muestra a continuación el código asociado al botón de comando del Módulo INV:

```
Private Sub Command13_Click()
exp.defselmod 13

Dim ent0 As String
Dim sal0 As String
cadbi = 0

nuinv = nuinv + 1
strnuinv = Str(nuinv)
snuinv = InputBox("DAR NÚMERO ASOCIADO CON ESTE INVERSOR", "INV", strnuinv)
nuinv = Val(snuinv)
exp.defnumod nuinv

capte0:
ent0 = InputBox("DAR ENTRADA ", "INV")
ent0 = UCase(ent0)
If varval(ent0) = False Then
GoTo capte0
End If
```

```
If Right(ent0, 1) <> "N" Then
cadbi = cadbi + 1
End If
exp.defent ent0, 0

capts0:
sal0 = InputBox("DAR SALIDA ", "INV")
sal0 = UCase(sal0)
If varval(sal0) = False Then
GoTo capts0
End If
If Right(sal0, 1) = "N" Then
GoTo capts0
End If
If Left(sal0, 1) = "E" Then
GoTo capts0
End If

exp.defsal sal0, 0
exp.defcadbin cadbi
End Sub
```

En el código anterior se muestra la asignación del número de tipo de módulo, el cual en este caso es el "13" (exp.defselmod 13) también se muestra la asignación de variables, las instrucciones para solicitar e introducir el número asociado de inversor que se quiera manejar y los bloques de instrucciones para la entrada y la salida.

El código asociado a los botones de comando correspondientes a estos módulos siguen la misma estructura que la de las compuertas lógicas, aunque obviamente es diferente. Cabe resaltar que son los módulos más sencillos, por lo cual el número de variables utilizadas es menor. En la figura 5.6 se muestran ejemplos de los dibujos de un Inversor y un Seguidor.

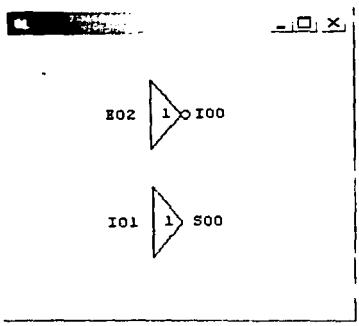


Figura 5.6 Dibujos de un Inversor y un Seguidor.

5.3.2.6 Módulo FFARS

Este módulo es referido a un Flip-Flop R-S asíncrono, que se maneja con un botón de comando que también tiene un código asociado, parecido a varios anteriormente vistos.

Se muestra a continuación el código asociado al botón de comando del Módulo FFARS:

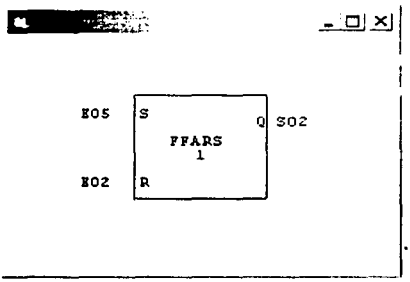
```
Private Sub Command20_Click()
exp.defselmod 20

Dim ent1 As String
Dim ent0 As String
Dim sal0 As String
cadbi = 0

nuffars = nuffars + 1
strnuffars = Str(nuffars)
snuffars = InputBox("DAR NÚMERO ASOCIADO CON ESTE FLIP-FLOP ASINCRONO RS", "ffars",
strnuffars)
nuffars = Val(snuffars)
exp.defnumod nuffars

captel:
ent1 = InputBox("DAR ENTRADA R ", "ffars")
ent1 = UCase(ent1)
If varval(ent1) = False Then
GoTo captel
End If

If Right(ent1, 1) <> "N" Then
cadbi = cadbi + 2
End If
exp.defent ent1, 1

capte0:
ent0 = InputBox("DAR ENTRADA S ", "ffars")
ent0 = UCase(ent0)
If varval(ent0) = False Then
GoTo capte0
End If

If Right(ent0, 1) <> "N" Then
cadbi = cadbi + 1
End If
exp.defent ent0, 0

capts0:
sal0 = InputBox("DAR SALIDA Q", "ffars")
```

```
sal0 = UCase(sal0)
If varval(sal0) = False Then
GoTo capts0
End If

If Right(sal0, 1) = "N" Then
GoTo capts0
End If
If Left(sal0, 1) = "E" Then
GoTo capts0
End If

exp.defsal sal0, 0
exp.defcadbin cadbi
End Sub
```

Para el código asociado al módulo FFARS, se muestra la asignación del número de tipo de módulo, el cual en este caso es el "20" (exp.defsal [illegible] "20"), también se asignan las variables, se muestran las instrucciones para solicitar e introducir el número asociado de Flip-Flop R-S asíncrono que se maneje y los bloques de instrucciones para el manejo de las variables de entrada y de salida.

También en este caso se tiene la instrucción con notación de punto para el manejo de la cadena binaria.

El dibujo de un Flip-Flop R-S asíncrono, se muestra a continuación.



5.3.2.7 Módulos TEMPO A, TEMPO C, TEMPO D y TEMPO E

Los módulos TEMPO A, TEMPO C, TEMPO D y TEMPO E son temporizadores de diferentes tipos que contiene el PLM como parte de sus módulos funcionales, los cuales se enlistan en la siguiente tabla.

TEMPORIZADOR	TIPO
TEMPO A	Temporizador monodisparo (one shot) de tipo uno.
TEMPO C	Temporizador monodisparo (one shot) de tipo dos.
TEMPO D	Temporizador con retardo a la activación (on-delay) o con retardo a la desactivación (off- delay).
TEMPO E	Temporizador astable.

Tabla 5.1 Definición de cada uno de los temporizadores.

Todos estos temporizadores también se manejan con un botón de comando que tiene un código asociado diferente para cada uno.

Como ejemplo se muestra el código asociado al botón de comando del Módulo TEMPO A:

```
Private Sub Command15_Click()
exp.defselmod 15

Dim ent2 As String
Dim ent1 As String
Dim ent0 As String
Dim sal0 As String
cadbi = 0
nutempoA = nutempoA + 1
strnutempoA = Str(nutempoA)
snutempoA = InputBox("DAR NÚMERO ASOCIADO CON ESTE TEMPORIZADOR", "tempoA", strnutempoA)
nutempoA = Val(snutempoA)
exp.defnumod nutempoA
capte2:
ent2 = InputBox("DAR ENTRADA H ", "tempoA")
ent2 = UCase(ent2)
```

```
If varval(ent2) = False Then
GoTo capte2
End If

If Right(ent2, 1) <> "N" Then
cadbi = cadbi + 4
End If
exp.defent ent2, 2

captel:
entl = InputBox("DAR ENTRADA R ", "tempoA")
entl = UCase(entl)
If varval(entl) = False Then
GoTo captel
End If

If Right(entl, 1) <> "N" Then
cadbi = cadbi + 2
End If
exp.defent entl, 1

capte0:
ent0 = InputBox("DAR ENTRADA D ", "tempoA")
ent0 = UCase(ent0)
If varval(ent0) = False Then
GoTo capte0
End If

If Right(ent0, 1) <> "N" Then
cadbi = cadbi + 1
End If
exp.defent ent0, 0

capts0:
sal0 = InputBox("DAR SALIDA T", "tempoA")
sal0 = UCase(sal0)
If varval(sal0) = False Then
GoTo capts0
End If

If Right(sal0, 1) = "N" Then
GoTo capts0
End If
If Left(sal0, 1) = "E" Then
GoTo capts0
End If
exp.defsal sal0, 0
exp.defcadbin cadbi

End Sub
```

Para los módulos TEMPO A, TEMPO C, TEMPO D y TEMPO E, el código asociado a los botones de comando correspondientes tiene algunas variaciones con respecto al mostrado anteriormente, dichas variaciones son principalmente en los bloques de código para el manejo de variables de entrada, dado que todos tienen una sola variable de salida.

El número de tipo de módulo, asignado para el TEMPO A es el "15" (exp. defselmod 15) el código anterior muestra como se asignan las variables, de igual manera se muestran el bloque de instrucciones para solicitar y asignar el número asociado a este tipo de módulo y los bloques de instrucciones para las entradas y la salida.

Los dibujos de los temporizadores antes mencionados se muestran a continuación.

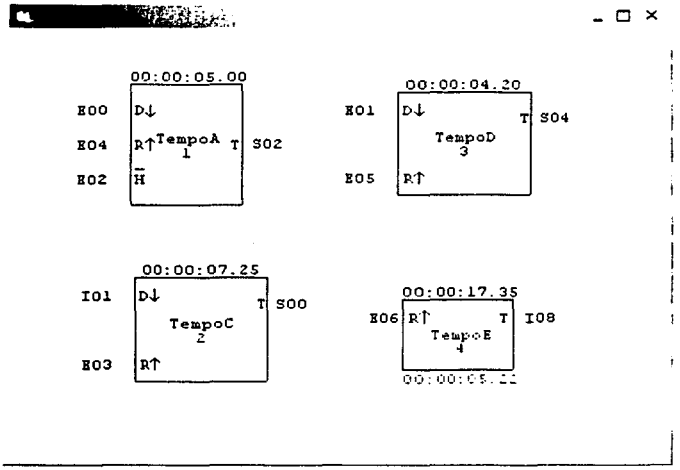


Figura 5.7 Temporizadores A, C, D y E.

5.3.3 GRAFICACIÓN DE LOS MÓDULOS

Dentro del desarrollo del software se tuvo que contemplar como una parte la que se refiere al dibujo de todas las compuertas, temporizadores y todos los módulos mostrados anteriormente, lo cual ocupó parte importante del tiempo invertido en este proyecto de tesis del cual se muestra algunas figuras en los ejemplos anteriores con el fin de que el lector se pueda familiarizar con ellos.

El código resultante de estos dibujos no se muestra en esta parte por ser bastante extenso, bastará con mencionar de manera general como se elaboraron dichas figuras y un ejemplo que se dará a continuación.

Las figuras se realizaron básicamente con los métodos *Line* y *Circle*, haciendo previamente el cálculo del tamaño que se deseaba además de que se tenía que imprimir en pantalla junto con la figura, los números y letras que diferenciarían las entradas y salidas, así como el nombre y el número asociado al tipo de módulo que se utilizara.

En cualquiera de las compuertas y también en los temporizadores y otros módulos más es posible tener entradas negadas o activadas con flancos de bajada, por lo tanto fue necesario introducir la posibilidad de dibujar una compuerta con todas o alguna de las entradas previamente negada o algún otro módulo funcional con entrada con flanco de bajada, todo esto según el usuario lo desee, para lo cual el usuario lo único que tiene que hacer es colocar una letra "n" mayúscula o minúscula al final de la nomenclatura con la que desee identificar esa entrada.

Como un ejemplo se muestra a continuación el código para dibujar una compuerta AND de dos entradas.

```
Public Sub diband2(xx, yy, e0, e1, s, cadbin, nn)
xi(numodgen) = xx
xd(numodgen) = xx + 525
yi(numodgen) = yy - 525
ys(numodgen) = yy + 525

Circle (xx, yy), 525, , 4.7, 1.57
Line (xx, yy - 525)-(xx, yy + 525)
CurrentX = xx + 130
CurrentY = yy - 120
Print nn

CurrentX = xx - 540
CurrentY = yy - 360
Print e0

CurrentX = xx - 540
CurrentY = yy + 105
Print e1

CurrentX = xx + 555
CurrentY = yy - 120
```

```
Print s

CurrentX = xx - 80
CurrentY = yy - 360
Print 0

CurrentX = xx - 80
CurrentY = yy + 105
Print 1
Select Case cadbin
Case 0
Circle (xx - 45, yy - 245), 45
Circle (xx - 45, yy + 210), 45
Case 1
Circle (xx - 45, yy + 210), 45
Case 2
Circle (xx - 45, yy - 245), 45
End Select

End Sub
```

A continuación se muestra una compuerta AND de dos entradas con una de ellas negada.

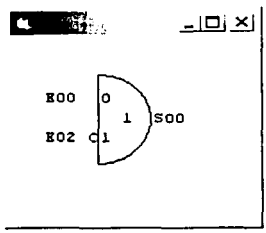


Figura 5.8 AND de dos entradas con una entrada previamente negada.

5.3.4 CADENAS BINARIAS

En los ejemplos anteriores se mencionó la cadena binaria, la cual se tiene como una cadena de caracteres que nos permite identificar en procesos posteriores del programa, cuales de las entradas de los distintos módulos funcionales fueron designadas por el usuario como entradas previamente negadas y cuales no, o en su caso cuales entradas de los contadores o temporizadores se activan mediante flanco de subida y cuales mediante flanco de bajada, así como también es posible identificar si el usuario preasignó alguna salida de un temporizador con un nivel de verificación bajo o no.

Esta cadena está compuesta por tantos dígitos binarios como entradas tenga el Módulo Funcional, en el caso de las compuertas lógicas, esto quiere decir que para un módulo de tres entradas, tendremos una cadena binaria de tres dígitos. Por ejemplo hablando de una compuerta AND3, cuya primera y última entrada fueran declaradas como previamente negadas por el usuario, se tendría la siguiente cadena binaria:

Cadena binaria = 010

Continuando con este ejemplo, en el caso de que el usuario no negara ninguna entrada previamente, esta cadena será compuestas como sigue:

Cadena binaria = 111

Es conveniente indicar que para el manejo de esta cadena se tuvo que hacer un procedimiento de conversión a binario ya que el usuario introduce números decimales, por lo tanto se debe leer de derecha a izquierda empezando por el bit menos significativo hasta el más significativo, que para efectos del programa aquí presentado no tiene mas significado que simplemente saber que el bit más significativo es el correspondiente al que se introduce primero y el menos significativo es el que corresponde al último que se introdujo.

Por ejemplo, en una cadena 1101 correspondiente a una compuerta NAND de cuatro entradas, leyendo de derecha a izquierda es posible saber que el primer "1" corresponde a la Entrada 0 y que el "0" corresponde a la Entrada 1 así como los otros dos bits en "1" corresponden a las entradas 2 y 3 respectivamente.

La figura desplegada para el ejemplo anterior sería la siguiente:

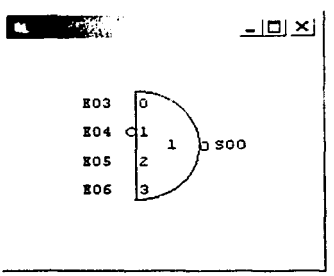


Figura 5.9 AND de cuatro entradas con la entrada E04 previamente negada.

En el caso de algunos temporizadores como los de tipo A, C y E, se tienen cadenas binarias de 4, 3 y 2 dígitos respectivamente los cuales cuentan con un bit extra para denotar el tipo de verificación de las salidas de cada uno.

Para el Flip-Flop asíncrono se tiene una cadena binaria de 4 digitos en donde de izquierda a derecha, el penúltimo dígito habrá de ser uno si se desea que la variable booleana de entrada SET tenga prioridad, en otro caso debe ser cero. Si la entrada SET tiene prioridad implica que si ambas entradas SET y RESET se verifican simultáneamente, la salida Q será uno lógico y para el otro caso será cero lógico.

Estas cadenas binarias forman parte de una cadena general de identificación que el programa genera para cada módulo que el usuario utiliza, y de hecho toma los últimos lugares en esta cadena.

5.3.5 CADENA DE IDENTIFICACIÓN DE MÓDULO

Con el fin de poder identificar y manejar en el programa los distintos módulos, se crearon cadenas de caracteres que nos permitieran concentrar toda la información necesaria para poder llevar a cabo mediante ellas la manipulación de la información por partes según fuera necesario y también la de los módulos en su totalidad.

Cada uno de los módulos tiene su propia cadena de identificación y está compuesta de 6 partes principales, las cuales se describen a continuación por orden de impresión:

1. **Número General de Módulo**.- Este es un número secuencial el cual se va incrementando cada vez que el usuario coloca un módulo en la pantalla, éste nos sirve para poder identificar a cada módulo de acuerdo al orden en el que se colocaron y sin importar que tipo de módulo sea.

2. **Cadena de Identificación de Tipo**.- Esta cadena está predefinida para cada módulo y es fija, se imprime en la cadena completa después del Número General de Módulo y está compuesta por dos partes:

   - Número de tipo de módulo.
   - Nombre del módulo.

   El número de tipo de módulo se utiliza en el programa para identificar el tipo de módulo con el que se esta trabajando ya que se requiere para poder llevar a cabo la ejecución de algunas partes del programa mediante un CASE que nos permite discriminar dependiendo del módulo con el que se esté trabajando, un ejemplo claro es el procedimiento de re-dibujo, donde en principio es necesario saber que tipo de módulo se va a dibujar después de haberse borrado.

3. **Número Secuencial de Módulo**.- Nos permite llevar un conteo de la cantidad de módulos de un mismo tipo que se están utilizando, el cual marca la diferencia primaria entre cada uno de los módulos de un mismo tipo que se estén utilizando.

4. **Entradas**.- En esta parte es donde se imprimen las distintas entradas que el usuario haya declarado. El número de entradas varía de un módulo a otro, por lo cual en esta parte también se imprimirán tantas entradas como tenga el módulo del que se trate y sólo se imprime la letra y los dos dígitos que se le hayan asignado y en el caso de que alguna entrada haya sido definida como previamente negada, esta información se guarda en la cadena binaria del propio módulo.

5. **Salida**.- Después de las entradas se imprime la salida donde nuevamente se imprimen únicamente la letra y los dos dígitos con que se haya declarado. En el caso anteriormente mencionado de los temporizadores que asignan alguna característica adicional a la salida, la letra "n" no se imprime ya que esta información se guarda en la cadena binaria.

6. **Cadena Binaria**.-Esta es la última parte de la cadena de identificación de módulo. Ésta es la parte donde se imprimen los dígitos binarios que conservan la información explicada con anterioridad.

En el código del programa la cadena de identificación para cada módulo que se utiliza, se forma en el evento MouseDown de Visual Basic; como ejemplo se muestra a continuación el código correspondiente al módulo AND de 2 entradas:

```
Case 1
'AND2
numodgen = numodres + numodgen + 1
numodspp = numodspp + 1
nnn = cad_n3d(numod)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If

ngen = cad_n3d(numod)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If

If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If

If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
```

```
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If

diband2 X, y, e0, e1, ss, cadbin, numod
ngen = cad_n3d(numodgen)
bin = gen_strbin(cadbin, 2)

cadefspp = cadefspp & ngen & "001AND2" & nnn & e0 & e1 & ss & bin & Chr(13) & Chr(10)
cadspp = ngen & "001AND2" & nnn & e0 & e1 & ss & bin & Chr(13) & Chr(10)
escodsilpp = escodsilpp & "              " & "AND2" & "#" & nnn & " " & vauxent(0) & "," &
vauxent(1) & "," & vauxsal(0) & "," & bin & ";" & Chr(13) & Chr(10)

xp(numodgen) = X
yp(numodgen) = y
```

Si se observa el código de los otros módulos para la formación de las respectivas cadenas de identificación, es posible notar que se definen dos tipos diferentes de cadenas, un tipo para los módulos del subprograma principal "*cadefspp*" y otro tipo para los módulos correspondientes al subprograma temporizado"*cadefspt*".

De esta manera, la cadena de identificación que se tiene para cada uno de los módulos se conforma como sigue:

Módulo	Cadena de Identificación
AND2	ngen 001AND2 nnn e0 e1 ss bin
AND3	ngen 002AND3 nnn e0 e1 e2 ss bin
AND4	ngen 003AND4 nnn e0 e1 e2 e3 ss bin
OR2	ngen 004OR2 nnn e0 e1 ss bin
OR3	ngen 005OR3 nnn e0 e1 e2 ss bin
OR4	ngen 006OR4 nnn e0 e1 e2 e3 ss bin
NAND2	ngen 007NAND2 nnn e0 e1 ss bin
NAND3	ngen 008NAND3 nnn e0 e1 e2 ss bin
NAND4	ngen 009NAND4 nnn e0 e1 e2 e3 ss bin
NOR2	ngen 010NOR2 nnn e0 e1 ss bin
NOR3	ngen 011NOR3 nnn e0 e1 e2 ss bin
NOR4	ngen 012NOR4 nnn e0 e1 e2 e3 ss bin

INVERSOR	ngen 013INV nnn e0 ss bin
SEGUIDOR	ngen 014SEG nnn e0 ss bin
FLIP-FLOP	ngen 015FFARS nnn e0 e1 ss bin
TEMPO A	Tt ngen 016TEMA nnn e0 e1 e2 ss bin
TEMPO C	Tt ngen 017TEMC nnn e0 e1 ss bin
TEMPO D	Tt ngen 018TEMD nnn e0 e1 ss bin
TEMPO E	Tt tt1 ngen 019TEME nnn e0 ss bin

Donde

ngen = Número general de módulo
nnn = Número secuencial de módulo
eX = Entradas
ss = Salidas
bin = Cadena binaria
Tt y tt1 = Tiempo

Una vez que el usuario tiene en pantalla uno o más módulos, y se lleva a cabo la ejecución de alguna aplicación ajena al programa, es posible que al regresar a este los módulos hayan desaparecido de la pantalla debido a que no se utiliza una función de refresco ya que consumiría más recursos de la computadora, pero en su lugar y como una mejor opción dado que es un procedimiento más sencillo se elaboró un procedimiento que permite volver a mostrar en pantalla los módulos anteriormente colocados en ella. Para lo cual solamente es necesario presionar el o los botones identificados como *Mostrar Módulos SPP y Mostrar Módulos SPT* según sea el caso.

5.3.6 ESCRITURA Y LECTURA DE ARCHIVOS

5.3.6.1 Archivos Gráficos

Después de que el usuario ha declarado algunos o todos los módulos deseados, es posible guardar esto en un archivo, esto se logra haciendo click en el botón ***Guardar***. Inmediatamente después aparecerá la ventana de guardado desplegando de facto el nombre de *ProVisPLM*, si se quiere guardar el archivo con ese nombre, es posible hacerlo pero si se desea cambiar por otro, bastará borrar el que aparece y escribir el que se elija.

El archivo quedará guardado con la extensión GRF en la ubicación que el usuario le haya asignado. Este archivo al igual que cualquier otro quedará disponible para cuando el usuario lo desee abrir.

Un ejemplo pequeño de archivo es el siguiente:

```
3

3

025050265500100lAND2001E01 E02 S00 11

016350157500200lAND2002190 E09 S99 11

013950340500300lAND2003E04 E03 S02 11

051000258000:00:06.07                004016TEMA001E13 E19 E06 S09 0100

077100328500:00:08.37 00:00:06.00 005019TEME002I06 S15 11
```

Como se muestra, este archivo se guarda en modo texto y es posible leerlo desde cualquier editor, éste guarda en la primera línea el numero de líneas que ocupan los módulos del subprograma principal que estén guardados y en la siguiente línea el número de módulos del subprograma temporizado que estén guardados más uno, debido a la línea vacía que separa ambas partes, estos datos sirven para saber el número de líneas de cada subprograma que se tienen que leer.

En seguida, se escriben las líneas que nos van a permitir identificar la ubicación de los módulos que se guardaron, el tipo y sus características. En concreto, los 10 primeros números son las coordenadas en X y Y que denotan su ubicación. En el caso de los temporizadores, estos 10 números van seguidos de

los tiempos asignados a cada uno. La última parte es la que se refiere a la cadena de identificación anteriormente vista.

En el caso particular de este ejemplo, se está utilizando un temporizador E que requiere de la asignación de dos tiempos, los cuales se muestran en la figura, por ello es que en el texto del temporizador A aparece un espacio vacío, el cual se asigna de esa manera para facilitar la lectura al momento de recuperar los datos del archivo que se está leyendo.

Para la apertura y lectura de archivos, se debe hacer click en el botón ***Abrir*** e inmediatamente aparecerá una ventana para que pueda abrir el archivo deseado como se hace normalmente.

Al abrir un archivo se actualizan todos los datos del programa y si el usuario lo requiere se pueden agregar módulos a los ya existentes en el archivo leído y es posible volver a guardar la nueva edición. Este proceso de actualización de datos es parte importante en este trabajo de tesis ya que requirió no solo de la actualización de los distintos datos que cada módulo contiene, sino también de la actualización del contador general interno que permitiera identificar de manera exacta la ubicación de cada módulo para continuar agregando módulos.

Como ejemplo se muestra el código de actualización de una compuerta NAND de 3 entradas:

```
numodgen = numodgen + 1
miy = yp2(numodres)
mix = xp2(numodres)
nnc = Val(Mid(sppresc, 12, 3))

e00 = Mid(sppresc, 15, 4)
vauxent(0) = e00
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If

e11 = Mid(sppresc, 19, 4)
vauxent(1) = e11
If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If

e22 = Mid(sppresc, 23, 4)
vauxent(2) = e22
If Len(vauxent(2)) = 3 Then
e2 = vauxent(2) & " "
Else
e2 = vauxent(2)
```

```
End If

s = Mid(sppresc, 27, 4)
sss = s
vauxsal(0) = sss
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If

cdbi = Mid(sppresc, 31, 3)
bin = cdbi
convierte CX
dibnand3 mix, miy, e0, e1, e2, s, valcdbi, nnc .

nnn = cad_n3d(nnc)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If

ngen = cad_n3d(nngen)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If

cadefspp = cadefspp & ngen & "008NAND3" & nnn & e0 & e1 & e2 & ss & bin & Chr(13) &
Chr(10)
cadspp = cadspp & ngen & "008NAND3" & nnn & e0 & e1 & e2 & ss & bin & Chr(13) & Chr(10)
escodsilpp = escodsilpp & "            " & "NAND3" & "#" & nnn & " " & vauxent(0) & "," &
vauxent(1) & "," & vauxent(2) & "," & vauxsal(0) & "," & bin & ";" & Chr(13) & Chr(10)
yp1(numodgen) = yp2(numodres)
xp1(numodgen) = xp2(numodres)
```

5.3.6.2 Archivo SIL

En este caso, el programa puede generar código SIIL1 de los módulos que se estén utilizando y guardarlo en un archivo listo para ejecutarse en el compilador y ponerlo a funcionar con el PLM, para ello es necesario hacer click en el botón Generar SIIL y se generará un archivo en la unidad A llamado ArchSIL con extensión SIL, lo cual es uno de los objetivos principales de este trabajo de tesis.

Un ejemplo de código SIL es el siguiente:

```
INPROG;

        OR3#001  E03, E01, E02, S01, 111;

        AND2#001  E04, E05, S02, 11;

        NOT#001  E08, S06, 1;

        FFARS#001  E02, E02, S05, 1111;

        NOR4#001  E12, E10, E09, E07, S10, 1111;

    FINPP;

    INMODI;

        TEMPOD#001  E15, E14, S08, 00:00:00.25, 11;

        TEMPOE#002  E16, S15, 00:00:16.11, 00:00:03.25, 11;

        TEMPOA#003  E24, E21, E20, S09, 00:00:35.62, 1111;

    FINMODI;
```

Como ya se dijo anteriormente los archivos gráficos pueden ser guardados y luego abiertos, por lo cual se aclara que el archivo SIL puede ser generado en cualquier momento ya sea cuando se haya acabado de abrir algún archivo o cuando se le hayan agregado algunos módulos más o de hecho cuando se esté trabajando en un archivo nuevo.

CAPITULO VI

EJEMPLOS DE APLICACIÓN

6.1 CÓMO REALIZAR UN DECODIFICADOR 2 A 4

Antes de empezar a explicar el ejemplo, hay que aclarar que aquí se plantea una forma particular de realizar un decodificador 2 a 4, dado que existen otras posibilidades. El caso que se toma como ejemplo es un decodificador que puede ser armado inversores y compuertas NAND de 3 entradas únicamente.

Este ejemplo comprende un decodificador 2 a 4 líneas con una entrada de habilitación, esto es, todas las entradas son iguales a 1 si la entrada de habilitación es igual a 1 sin importar los valores en las otras dos entradas. Cuando la entrada de habilitación es 0 el circuito opera como un decodificador con salidas complementadas. A continuación se muestra la tabla de verdad para este caso.

Entradas			Salidas			
E	A	B	D_0	D_1	D_2	D_3
1	X	X	1	1	1	1
0	0	0	0	1	1	1
0	0	1	1	0	1	1
0	1	0	1	1	0	1
0	1	1	1	1	1	0

Tabla 6.1 Tabla de verdad de un decodificador 2 a 4 líneas con una entrada de habilitación.

Las X bajo A y B son condiciones despreocupadas. La operación normal del decodificador ocurre sólo con E=0, y las salidas se seleccionan cuando están en el estado 0.

El diagrama del decodificador propuesto se muestra en la siguiente figura.

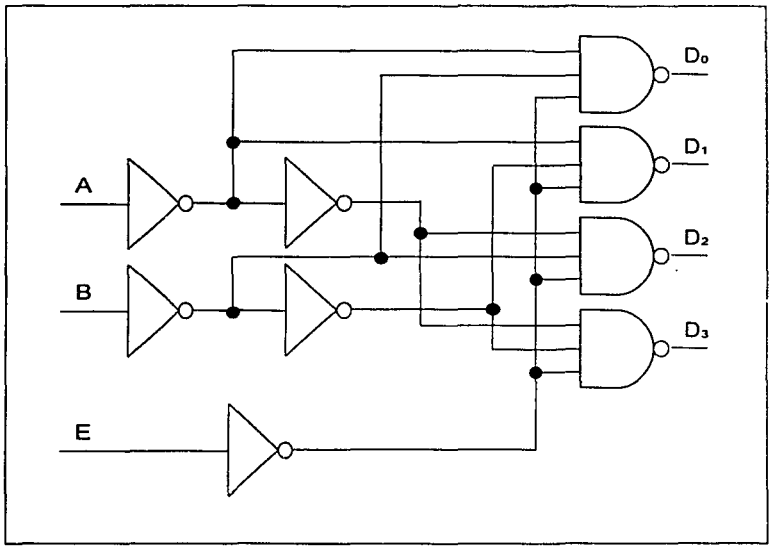


Figura 6.1 Diagrama de un decodificador 2 a 4 líneas con una entrada de habilitación.

Para poder programara el diagrama anterior en el PLM con el software desarrollado, es necesario asignarle un nombre a las entradas salidas y variables intermediarias que sea acorde a la notación utilizada en los programas del PLM, tal como se indicó en el punto 5.3.1. Para entonces poder trabajar directamente con el software en ambiente visual y poder designar las entradas y salidas correctas, porque de otra forma, cuando el programa genere el código fuente para pasarlo al compilador, éste detectaría algunos errores y por ende sería imposible hacer que el PLM funcione con ese programa, y por lo tanto se tendría que corregir.

Para hacerlo de una manera más explicita, se vuelve a hacer uso del diagrama mostrado anteriormente, donde ahora se le asignaran las nomenclaturas correspondientes.

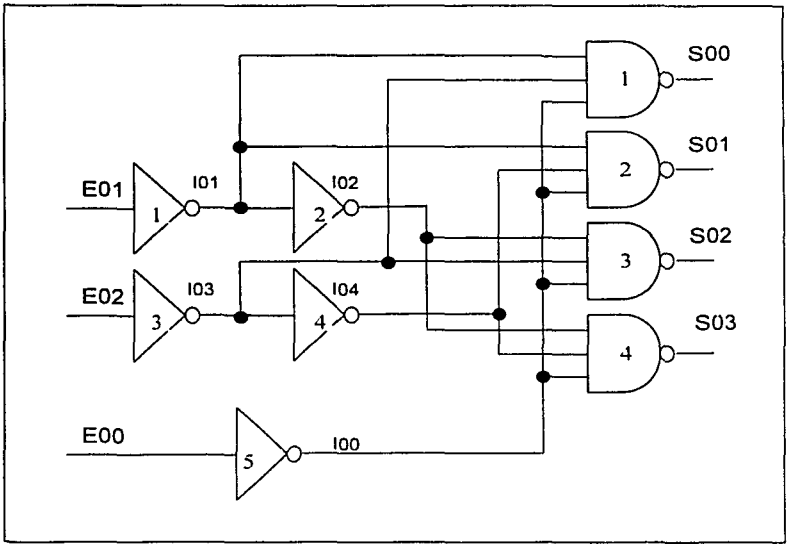


Figura 6.2 Representación del diagrama de un decodificador 2 a 4 líneas con una entrada de habilitación, con la nomenclatura definida para su uso en SIIL1.

Después de esto el usuario deberá escoger entre los módulo disponibles en el menú que se tiene en el programa el que desee utilizar primero, ya que el orden en el cual se introduzcan los módulos no cambia la operatividad del decodificador que se quiere procesar, basta con que el usuario haga click sobre el módulo deseado para seleccionarlo, e inmediatamente después es necesario que se introduzca en la ventana que se despliega el número de módulo con el que se esté trabajando, o bien se puede presionar "Intro" ya que de facto aparece un número secuencial, se debe notar que los módulos lógicos del diagrama anterior, también se les asignó un número para poder identificarlos.

Suponiendo que el usuario eligiera una compuerta NAND 3, la ventana de entrada de datos que se desplegaría es como la que se muestra en la siguiente figura.

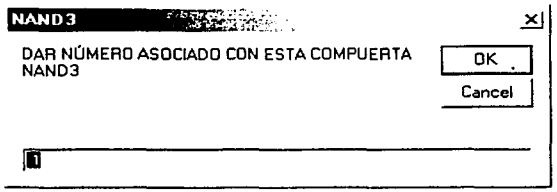
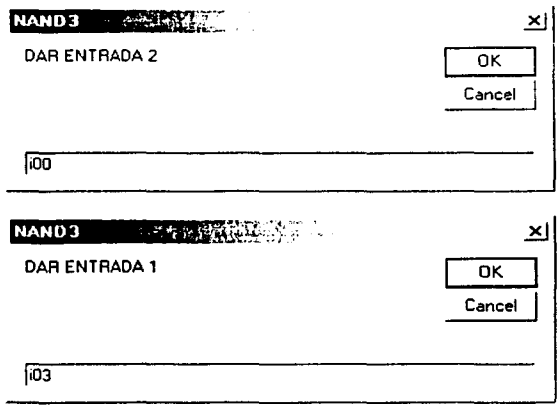


Figura 6.3 Ventana para ingresar el número del tipo de compuerta que se está utilizando.

En seguida de que el usuario asigna el número, es posible comenzar a introducir las entradas y salidas de cada módulo, las cuales se introducen en otras ventanas que aparecen de forma consecutiva. Continuando con el mismo ejemplo de la NAND 3, las siguientes ventanas en aparecer serían como las que se muestran en la figura 6.4. Es importante notar que el programa solicita primero la entrada número 2, esto quiere decir que si se leen las entradas de una compuerta de abajo hacia arriba, ésta se refiere a la primera, o sea la entrada que se encuentra ubicada en la parte de abajo, para después hacer lo mismo con la de en medio y finalmente con la de arriba, y para terminar se introduce la salida. continuación:



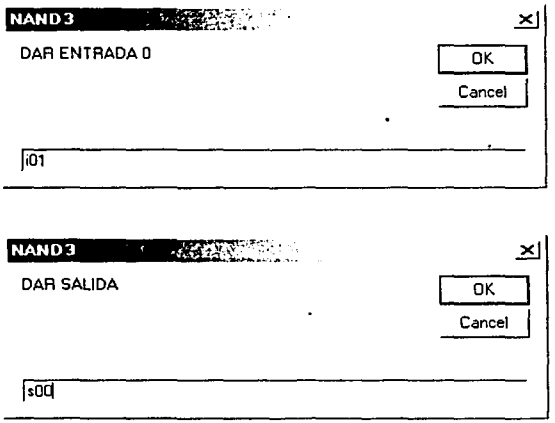


Figura 6.4 Ventanas en las cuales se ingresan los datos que se pueden observar mientras se trabaja con una compuerta NAND de 3 entradas.

Posteriormente bastará con hacer click en cualquier lugar de la pantalla para que la figura aparezca en donde el usuario apunte con el mouse, quedando como lo muestra la figura 6.5. Se sugiere que el usuario acomode los módulos de tal manera que el esquema permita visualizar el orden de las entradas y las salidas, para su mejor entendimiento, aunque esto no es indispensable.

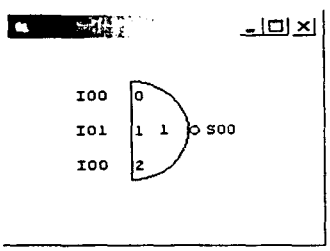
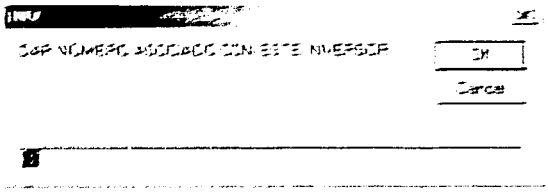


Figura 6.5 Dibujo desplegado de la compuerta NAND 3 que se trabajó como ejemplo.

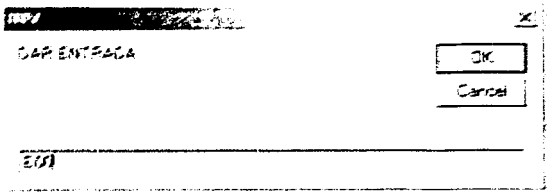
El procedimiento anterior se tendrá que repetir de forma semejante para las otras compuertas NAND de 3 entradas.

Posteriormente un proceso similar ocurrirá para los inversores, donde para el caso del inversor que se numeró como 5 el proceso será como sigue:

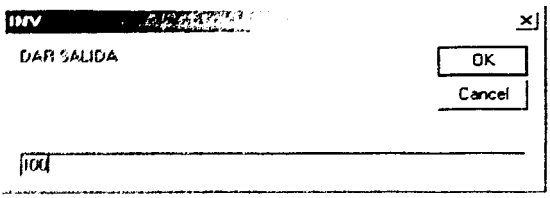
- Hacer click sobre el botón "INV" para seleccionarlo con lo que se desplegará la siguiente ventana:



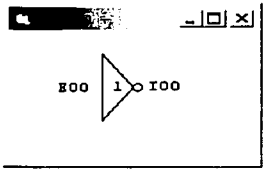
- Introducir el número de inversor y presionar "intro" para que se despliegue la ventana siguiente



- Proporcionar la nomenclatura de identificación de la entrada y en seguida aparecerá la siguiente ventana:

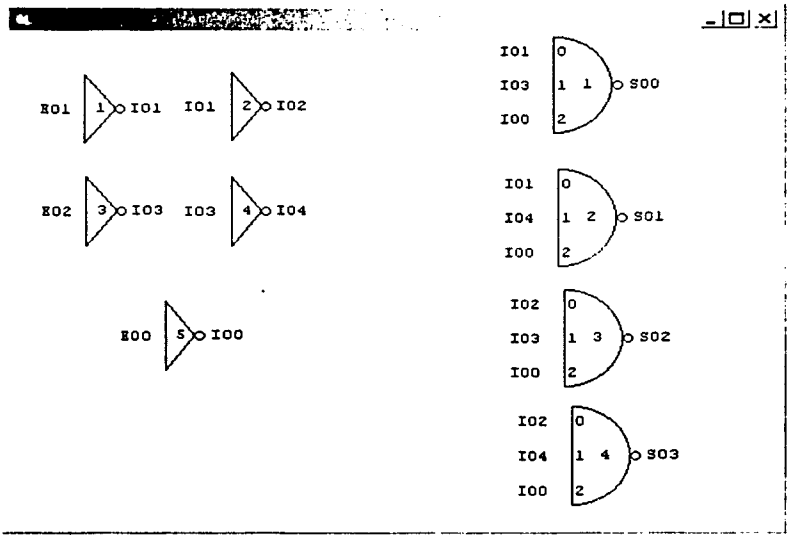


- Después de haber introducido "i00", presionar "intro" y posicionar el mouse donde se quiera que aparezca el dibujo en la pantalla, de donde el dibujo que aparecería sería el como el que se muestra:



- Repetir el mismo procedimiento para cada elemento.

Al concluir dicho proceso, se obtiene la siguiente pantalla:



Posteriormente, es posible generar el código SIIL1 haciendo click en el botón ***Generar SIIL1***, donde se genera un archivo en la unidad A llamado ArchSIL con extensión SIL, obteniendo el código siguiente:

```
INPROG;

        NOT#1 E01 , I01 , 1;

        NOT#2 I01 , I02 , 1;

        NOT#3 E02 , I03 , 1;

        NOT#4 I03 , I04 , 1;

        NOT#5 E00 , I00 , 1;

        NAND3#1 I01 , I03 , I00 , S00 , 111;

        NAND3#2 I01 , I04 , I00 , S01 , 111;

        NAND3#3 I02 , I03 , I00 , S02 , 111;

        NAND3#4 I02 , I04 , I00 , S03 , 111;

    FINPP;

    INMODI;


    FINMODI;
```

De igual forma, es posible guardar el archivo gráfico para tenerlo disponible cuando se desee. Para este fin, se debe hacer click en el botón ***Guardar***. Inmediatamente después aparecerá la ventana de guardado lista para que se le asigne el nombre deseado al archivo. De lo anterior se obtiene el código que se muestra en el siguiente cuadro:

```
 9

 1

007500091500l013INV001E01 I01 1

021900088500201 3INV002I01 I02 1

007650213000301 3INV003E02 I03 1

021900213000401 3INV004I03 I04 1

015450360000501 3INV005E00 I00 1

053850063000600 8NAND3001I01 I03 I00 S00 111

054300220500700 8NAND3002I01 I04 I00 S01 111

054900363000800 8NAND3003I02 I03 I00 S02 111

055650501000900 8NAND3004I02 I04 I00 S03 111
```

Este es un ejemplo sencillo basado en operadores lógicos que se enfoca principalmente al ámbito académico, pero el PLM es capaz de llevar a cabo operaciones más complejas con las cuales es posible llevar el control de un proceso real en la industria.

6.2 UN EJEMPLO DE APLICACIÓN INDUSTRIAL

A continuación se presenta un ejemplo completo de la generación de un programa para una función determinada, la cual se plantea a continuación.

Se propone realizar con el PLM la lógica requerida por un arrancador de voltaje reducido basado en autotransformador con transición de cierre de circuito abierto para un motor de inducción trifásico; en la industria existen arrancadores de este tipo implantados con lógica alambrada.

En la figura 6.6 se ilustra el esquema del conexionado al motor de los elementos actuadores de potencia (contactores), se supone que la secuencia de arranque requiere que los contactores 1A y 2A se cierren al oprimirse el botón de arranque "A" y permanezcan así por tres segundos para abrirse una vez que haya transcurrido este tiempo, después de esto debe haber un tiempo de espera de medio segundo para cerrar el contactor M (marcha); como es usual en este tipo de sistemas se cuenta con un sensor de sobrecarga (OL) que abre sus contactos asociados al detectarse esta condición, además de que se debe contar con un botón de paro (P) de modo que al oprimirse el mismo, el motor sea desconectado del suministro trifásico que lo alimenta, se supone que el tap seleccionado para los autotransformadores es el adecuado para lograr el par de arranque requerido por la carga mecánica que mueve el motor.

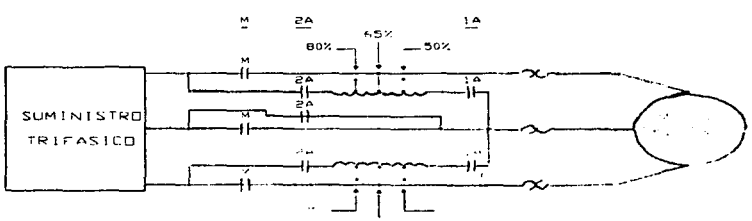


Figura 6.6 Conexionado de arrancador de voltaje reducido basado en autotransformador cuya secuencia de arranque se ha de implantar con el PLM.

Una forma de realizar la secuencia de arranque con módulos lógicos realizables por el PLM se ilustra en la figura 6.7, en ella se aprecia el empleo de cinco módulos los cuales son: una compuerta AND de dos entradas, un Flip-Flop RS con prioridad al RESET, un temporizador monodisparo (one-shot) con salida verificada en alto y duración de tres segundos con disparo por flanco de subida, un temporizador con retardo de activación (ON DELAY) con retardo de 3.5 segundos

y un seguidor lógico; en la misma figura se muestran las variables booleanas empleadas y con que elementos físicos están relacionadas.

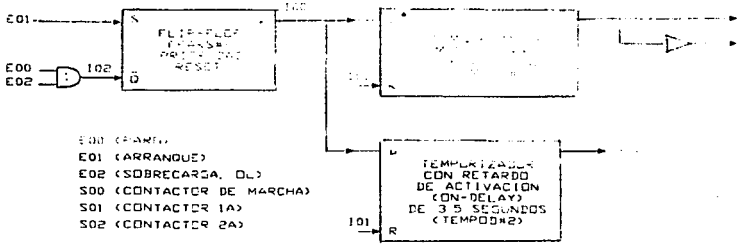
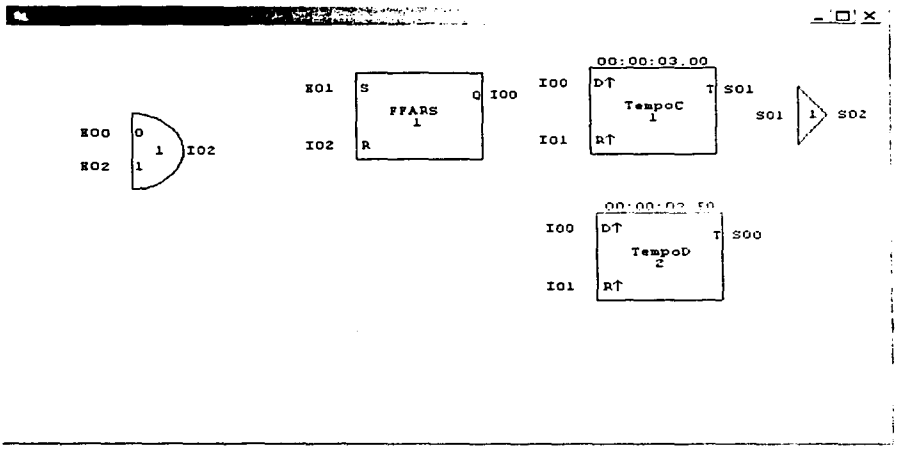


Figura 6.7 Implantación de la secuencia de arranque requerido, empleando módulos realizables por el PLM.

El software desarrollado en este trabajo de tesis al igual que el lenguaje SIIL1 permite especificar entre otras cosas para un mismo tipo de módulo, los niveles de verificación de las entradas y salidas que el mismo pudiera contener; de esta manera, los niveles de verificación para las entradas "S" y "R" del Flip-Flop se especificaron como alto y bajo respectivamente, que es lo adecuado para este caso, el nivel de verificación de las entradas de RESET para los dos temporizadores se definió como alto, el nivel de verificación para las salidas de los mismos se definió como alto, se debe notar el empleo de la Variable Booleana Intermediaria (VBI) 101 como señal de RESET para ambos temporizadores, cabe señalar aquí que siempre que se inicia la ejecución de un programa en el PLM, los buffers asociados con las variables booleanas son inicializados con ceros, de esta manera, una VBI que no sea empleada como salida por algún módulo tendrá siempre un nivel de cero lógico

De esta manera, el desarrollo en el software de ambiente visual para la aplicación antes mencionada queda como se muestra en la figura siguiente:



De donde es posible generara el código SIIL1 haciendo click en el botón ***Generar SIIL1*** obteniendo un archivo en la unidad A llamado ArchSIL con extensión SIL, obteniendo el código siguiente:

```
INPROG;

    FFARS#1 E01 ,I02 ,I00 ,1000;

    AND2#1 E00 ,E02 ,I02 ,11;

    SEG#1 E01 ,S02 ,1;

FINPP;

INMODI;

    TEMPOC#1 I00 ,I01 ,S01 ,00:00:03.00 , I01;

    TEMPOD#2 I00 ,I01 ,S00 ,00:00:03.50 ,10;

FINMODI;
```

Igualmente es posible guardar el archivo gráfico y tenerlo disponible cuando se desee. Para ello basta con hacer click en el botón ***Guardar***. Inmediatamente después aparece la ventana de guardado para que se le asigne el nombre deseado. De lo anterior se obtiene el código que se muestra en la siguiente figura.

```
3

3

0280501350003015FFARS001E01 I02 I00 1000

0067501725004001AND2001E00 E02 I02 11

0808501785005014SEG001E01 S02 1

055650130500:00:03.00                001017TEMC001I00 I01 S01 111

0562503720 00:00:03.50                002018TEMD002I00 I01 S00 11
```

Después de estos ejemplos se espera que el lector tenga una idea clara de lo que es el funcionamiento del Ambiente Visual para la programación del PLM, que es el objetivo de esta tesis, con el cual se busca facilitar la parte de la programación esperando reducir significativamente la dificultad que ello representa.

CONCLUSIONES

El desarrollo de este trabajo de tesis ha permitido conocer distintos aspectos sobre los PLC's, los cuales van desde los más complejos y utilizados ampliamente en la industria hasta los PLC prototipos como con el que se trabajó para la elaboración de este trabajo.

El PLM como un PLC prototipo es un dispositivo muy importante, ya que es posible realizar trabajos con él, sin tener que sacrificar nada en el desempeño, ya que cuenta con todas las posibilidades de funcionamiento con las que cuenta un PLC común. Para este caso particular, lo anterior ha permitido crear un software de programación que en los inicios era solamente una idea sin forma, ya que aún cuando se tenía la idea de lo que se quería hacer, no se conocía completamente el nivel de complejidad al que se tendría que enfrentar quien desarrollara dicho software.

Sin embargo, se tenía una clara idea del objetivo, aunque se desconocieran algunos aspectos que se tendrían que enfrentar, lo cual causó en un principio, el deseo de desarrollar un software con todas las herramientas del ambiente visual con las que cuentan programas como Windows y sus aplicaciones más comunes, al mismo tiempo que esto se conjuntaba con las características que permitieran la elaboración de circuitos de programación en el ambiente visual como lo hacen algunos programas comerciales para el diseño de circuitería como el Tango y otros, pero que además contara con la característica de tener la capacidad de traducir el circuito diseñado en ambiente visual a un programa en modo texto que quedara ya estructurado en el lenguaje propio del PLM, es decir, elaborar un circuito en ambiente visual y traducirlo a SIIL1.

Lo anterior resultó bastante complicado, ya que el desarrollo de herramientas como Windows, toman muchas horas hombre frente a una computadora al igual que las aplicaciones para el desarrollo de circuitos en ambiente visual, las cuales funcionan a veces en ambiente MS-DOS.

Al comenzar con la elaboración de este proyecto, se desconocían algunos temas relacionados con la programación de los PLC's, las cuales se fueron encontrando con el paso de los temas incluidos en este trabajo, en los cuales se tuvo que realizar investigación. Lo más importante en cuanto a la realización del software fue la existencia de un organismo a nivel mundial para la estandarización de la programación de PLC's, que define las formas de programación y como se deben llevar a cabo. Por su parte, el PLM como un PLC prototipo, cuenta hoy día con un tipo de programación del más alto nivel en cuanto al PLC's se refiere.

Cabe destacar que la programación mediante Ambiente Visual con la que hoy cuenta el PLM, no obedece totalmente a los estándares anteriormente mencionados, ya que no se trata de un dispositivo comercial y que además se

buscó hacerlo más esquemático ya que se presentan las figuras como son conocidas normalmente y no como forma de bloques en su totalidad, tal cual lo indica el instrumento de estandarización, ya que además, no se pudo tener acceso a la información completa de dicha estandarización por falta de licencia para disponer de ella.

Al desarrollar este software hubieron cosas que se tuvieron que hacer a un lado debido a la inversión en tiempo que ello significaba, consiguiendo desarrollar lo que se presenta en esta Tesis, lo cual era en términos globales lo que se pretendía realizar. Este programa ahorrará tiempo de programación del PLM aún a los usuarios más experimentados en él.

Hoy día el PLM puede ser programado de la manera más sencilla y con el mejor ambiente ya que se consiguen los principales objetivos que se plantearon al principio, que es crear un medio de programación fácil y ameno en el cual se pueda hacer uso del PLM con la única condición de tener conocimiento de los dispositivos electrónicos que el PLM lleva a cabo de manera virtual.

Por otra parte, la presente Tesis representa la primera aportación en esta forma de programación para el Programador Lógico Modular, la cual sin duda podrá ser mejorada y ampliada, al mismo tiempo que se podrán crear otras aplicaciones para su programación, pero para las metas fijadas al principio de este trabajo, es posible decir que las expectativas fueron satisfactoriamente cubiertas.

ANEXO 1

FORMA B

```
Dim nuand2, nuand3, nuand4 As Integer
Dim nuor2, nuor3, nuor4 As Integer
Dim nunand2, nunand3, nunand4 As Integer
Dim nunor2, nunor3, nunor4 As Integer
Dim nuinv, nuseg, nuffars, nucontev, nusecuest As Integer
Dim numtemgen As Integer
Public Function varvale(X As String) As Boolean
'Retorna false si el string x no es una entrada o salida válido'en
otro caso retorna true
Dim strpp As String
For i = 0 To 9
For j = 0 To 9
strpp = "E" & Right(Str(i), Len(Str(i)) - 1) & Right(Str(j),
Len(Str(j)) - 1)
If X = strpp Or X = strpp & "N" Then
varvale = True
Exit Function
End If
Next j
Next i
For i = 0 To 9
For j = 0 To 9
strpp = "I" & Right(Str(i), Len(Str(i)) - 1) & Right(Str(j),
Len(Str(j)) - 1)
If X = strpp Or X = strpp & "N" Then
varvale = True
Exit Function
End If
Next j
Next i
varvale = False
End Function
```

```
Public Function varvals(X As String) As Boolean
'Retorna false si el string x no es una entrada o salida válido
'en otro caso retorna true
Dim strpp As String
For i = 0 To 9
For j = 0 To 9
strpp = "S" & Right(Str(i), Len(Str(i)) - 1) & Right(Str(j),
Len(Str(j)) - 1)
If X = strpp Or X = strpp & "N" Then
varvals = True
Exit Function
End If
Next j
Next i
For i = 0 To 9
For j = 0 To 9
strpp = "I" & Right(Str(i), Len(Str(i)) - 1) & Right(Str(j),
Len(Str(j)) - 1)
If X = strpp Or X = strpp & "N" Then
varvals = True
Exit Function
End If
Next j
Next i
varvals = False
End Function
```

```
Private Sub Command1_Click()
exp.defselmod 1
Dim ent1 As String
Dim ent0 As String
Dim sal0 As String
cadbi = 0
nuand2 = nuand2 + 1
strnuand2 = Str(nuand2)
snuand2 = InputBox("DAR NÚMERO ASOCIADO CON ESTA
COMPUERTA AND2", "AND2", strnuand2)
nuand2 = Val(snuand2)
exp.defnumod nuand2
capte1:
ent1 = InputBox("DAR ENTRADA 1 ", "AND2")
ent1 = UCase(ent1)
If varvale(ent1) = False Then
GoTo capte1
End If
If Right(ent1, 1) <> "N" Then
cadbi = cadbi + 2
End If
exp.defent ent1, 1
capte0:
ent0 = InputBox("DAR ENTRADA 0 ", "AND2")
ent0 = UCase(ent0)
If varvale(ent0) = False Then
GoTo capte0
End If
If Right(ent0, 1) <> "N" Then
cadbi = cadbi + 1
End If
exp.defent ent0, 0
capts0:
sal0 = InputBox("DAR SALIDA ", "AND2")
sal0 = UCase(sal0)
If varvals(sal0) = False Then
GoTo capts0
End If
If Right(sal0, 1) = "N" Then
GoTo capts0
End If
If Left(sal0, 1) = "E" Then
GoTo capts0
End If
exp.defsal sal0, 0
exp.defcadbin cadbi
End Sub
```

```
Private Sub Command2_Click()
exp.defselmod 2
Dim ent2 As String
Dim ent1 As String
Dim ent0 As String
Dim sal0 As String
cadbi = 0
nuand3 = nuand3 + 1
strnuand3 = Str(nuand3)
snuand3 = InputBox("DAR NÚMERO ASOCIADO CON ESTA
COMPUERTA AND3", "AND3", strnuand3)
nuand3 = Val(snuand3)
exp.defnumod nuand3
capte2:
ent2 = InputBox("DAR ENTRADA 2 ", "AND3")
ent2 = UCase(ent2)
If varvale(ent2) = False Then
GoTo capte2
End If
If Right(ent2, 1) <> "N" Then
cadbi = cadbi + 4
End If
exp.defent ent2, 2
capte1:
ent1 = InputBox("DAR ENTRADA 1 ", "AND3")
ent1 = UCase(ent1)
If varvale(ent1) = False Then
GoTo capte1
End If
If Right(ent1, 1) <> "N" Then
cadbi = cadbi + 2
End If
exp.defent ent1, 1
capte0:
ent0 = InputBox("DAR ENTRADA 0 ", "AND3")
ent0 = UCase(ent0)
```

```
If varvale(ent0) = False Then
GoTo capte0
End If
If Right(ent0, 1) <> "N" Then
cadbi = cadbi + 1
End If
exp.defent ent0, 0
capts0:
sal0 = InputBox("DAR SALIDA ", "AND3")
sal0 = UCase(sal0)
If varvals(sal0) = False Then
GoTo capts0
End If
If Right(sal0, 1) = "N" Then
GoTo capts0
End If
If Left(sal0, 1) = "E" Then
GoTo capts0
End If
exp.defsal sal0, 0
exp.defcadbin cadbi
End Sub
```

```
Private Sub command3_Click()
exp.defselmod 3
Dim ent3 As String
Dim ent2 As String
Dim ent1 As String
Dim ent0 As String
Dim sal0 As String
cadbi = 0
nuand4 = nuand4 + 1
strnuand4 = Str(nuand4)
snuand4 = InputBox("DAR NUMERO ASOCIADO CON ESTA
COMPUERTA AND4", "AND4", strnuand4)
nuand4 = Val(snuand4)
exp.defnumod nuand4
capte3:
ent3 = InputBox("DAR ENTRADA 3 ", "AND4")
ent3 = UCase(ent3)
If varvale(ent3) = False Then
GoTo capte3
End If
If Right(ent3, 1) <> "N" Then
cadbi = cadbi + 8
End If
exp.defent ent3, 3
capte2:
ent2 = InputBox("DAR ENTRADA 2 ", "AND4")
ent2 = UCase(ent2)
If varvale(ent2) = False Then
GoTo capte2
End If
If Right(ent2, 1) <> "N" Then
cadbi = cadbi + 4
End If
exp.defent ent2, 2
capte1:
ent1 = InputBox("DAR ENTRADA 1 ", "AND4")
ent1 = UCase(ent1)
If varvale(ent1) = False Then
GoTo capte1
End If
If Right(ent1, 1) <> "N" Then
cadbi = cadbi + 2
End If
exp.defent ent1, 1
capte0:
ent0 = InputBox("DAR ENTRADA 0 ", "AND4")
ent0 = UCase(ent0)
If varvale(ent0) = False Then
GoTo capte0
End If
If Right(ent0, 1) <> "N" Then
cadbi = cadbi + 1
End If
exp.defent ent0, 0
capts0:
sal0 = InputBox("DAR SALIDA ", "AND4")
sal0 = UCase(sal0)
If varvals(sal0) = False Then
GoTo capts0
End If
If Right(sal0, 1) = "N" Then
GoTo capts0
End If
If Left(sal0, 1) = "E" Then
GoTo capts0
End If
exp.defsal sal0, 0
exp.defcadbin cadbi
End Sub
```

```
Private Sub Command4_Click()
exp.defselmod 4
Dim ent1 As String
Dim ent0 As String
Dim sal0 As String
cadbi = 0
nuor2 = nuor2 + 1
strnuor2 = Str(nuor2)
snuor2 = InputBox("DAR NUMERO ASOCIADO CON ESTA
COMPUERTA OR2", "OR2", strnuor2)
nuor2 = Val(snuor2)
exp.defnumod nuor2
capte1:
ent1 = InputBox("DAR ENTRADA 1 ", "OR2")
ent1 = UCase(ent1)
If varvale(ent1) = False Then
GoTo capte1
End If
If Right(ent1, 1) <> "N" Then
cadbi = cadbi + 2
End If
exp.defent ent1, 1
capte0:
ent0 = InputBox("DAR ENTRADA 0 ", "OR2")
ent0 = UCase(ent0)
If varvale(ent0) = False Then
GoTo capte0
End If
If Right(ent0, 1) <> "N" Then
cadbi = cadbi + 1
End If
exp.defent ent0, 0
capts0:
sal0 = InputBox("DAR SALIDA ", "OR2")
sal0 = UCase(sal0)
If varvals(sal0) = False Then
GoTo capts0
End If
If Right(sal0, 1) = "N" Then
GoTo capts0
End If
If Left(sal0, 1) = "E" Then
GoTo capts0
End If
exp.defsal sal0, 0
exp.defcadbin cadbi
End Sub
```

```
Private Sub Command5_Click()
exp.defselmod 5
Dim ent2 As String
Dim ent1 As String
```

```
Dim ent0 As String
Dim sal0 As String
cadbi = 0
nuor3 = nuor3 + 1
strnuor3 = Str(nuor3)
snuor3 = InputBox("DAR NÚMERO ASOCIADO CON ESTA
COMPUERTA OR3", "OR3", strnuor3)
nuor3 = Val(snuor3)
exp.defnumod nuor3
capte2:
ent2 = InputBox("DAR ENTRADA 2 ", "OR3")
ent2 = UCase(ent2)
If varvale(ent2) = False Then
GoTo capte2
End If
If Right(ent2, 1) <> "N" Then
cadbi = cadbi + 4
End If
exp.defent ent2, 2
capte1:
ent1 = InputBox("DAR ENTRADA 1 ", "OR3")
ent1 = UCase(ent1)
If varvale(ent1) = False Then
GoTo capte1
End If
If Right(ent1, 1) <> "N" Then
cadbi = cadbi + 2
End If
exp.defent ent1, 1
capte0:
ent0 = InputBox("DAR ENTRADA 0 ", "OR3")
ent0 = UCase(ent0)
If varvale(ent0) = False Then
GoTo capte0
End If
If Right(ent0, 1) <> "N" Then
cadbi = cadbi + 1
End If
exp.defent ent0, 0
capts0:
sal0 = InputBox("DAR SALIDA ", "OR3")
sal0 = UCase(sal0)
If varvals(sal0) = False Then
GoTo capts0
End If
If Right(sal0, 1) = "N" Then
GoTo capts0
End If
If Left(sal0, 1) = "E" Then
GoTo capts0
End If
exp.defsal sal0, 0
exp.defcadbin cadbi
End Sub
```

```
Private Sub Command6_Click()
exp.defselmod 6
Dim ent3 As String
Dim ent2 As String
Dim ent1 As String
Dim ent0 As String
Dim sal0 As String
cadbi = 0
nuor4 = nuor4 + 1
strnuor4 = Str(nuor4)
snuor4 = InputBox("DAR NÚMERO ASOCIADO CON ESTA
COMPUERTA OR4", "OR4", strnuor4)
nuor4 = Val(snuor4)
exp.defnumod nuor4
capte3:
ent3 = InputBox("DAR ENTRADA 3 ", "OR4")
ent3 = UCase(ent3)
If varvale(ent3) = False Then
GoTo capte3
End If
If Right(ent3, 1) <> "N" Then
cadbi = cadbi + 8
End If
exp.defent ent3, 3
capte2:
ent2 = InputBox("DAR ENTRADA 2 ", "OR4")
ent2 = UCase(ent2)
If varvale(ent2) = False Then
GoTo capte2
End If
If Right(ent2, 1) <> "N" Then
cadbi = cadbi + 4
End If
exp.defent ent2, 2
capte1:
ent1 = InputBox("DAR ENTRADA 1 ", "OR4")
ent1 = UCase(ent1)
If varvale(ent1) = False Then
GoTo capte1
End If
If Right(ent1, 1) <> "N" Then
cadbi = cadbi + 2
End If
exp.defent ent1, 1
capte0:
ent0 = InputBox("DAR ENTRADA 0 ", "OR4")
ent0 = UCase(ent0)
If varvale(ent0) = False Then
GoTo capte0
End If
If Right(ent0, 1) <> "N" Then
cadbi = cadbi + 1
End If
exp.defent ent0, 0
capts0:
sal0 = InputBox("DAR SALIDA ", "OR4")
sal0 = UCase(sal0)
If varvals(sal0) = False Then
GoTo capts0
End If
If Right(sal0, 1) = "N" Then
GoTo capts0
End If
If Left(sal0, 1) = "E" Then
GoTo capts0
End If
exp.defsal sal0, 0
exp.defcadbin cadbi
End Sub
```

```
Private Sub Command7_Click()
exp.defselmod 7
Dim ent1 As String
Dim ent0 As String
Dim sal0 As String
cadbi = 0
nunand2 = nunand2 + 1
strnunand2 = Str(nunand2)
snunand2 = InputBox("DAR NÚMERO ASOCIADO CON ESTA
COMPUERTA NAND2", "NAND2", strnunand2)
nunand2 = Val(snunand2)
exp.defnumod nunand2
capte1:
ent1 = InputBox("DAR ENTRADA 1 ", "NAND2")
ent1 = UCase(ent1)
If varvale(ent1) = False Then
GoTo capte1
End If
```

```
If Right(ent1, 1) <> "N" Then
cadbi = cadbi + 2
End If
exp.defent ent1, 1
capte0:
ent0 = InputBox("DAR ENTRADA 0 ", "NAND2")
ent0 = UCase(ent0)
If varvale(ent0) = False Then
GoTo capte0
End If
If Right(ent0, 1) <> "N" Then
cadbi = cadbi + 1
End If
exp.defent ent0, 0
capts0:
sal0 = InputBox("DAR SALIDA ", "NAND2")
sal0 = UCase(sal0)
If varvals(sal0) = False Then
GoTo capts0
End If
If Right(sal0, 1) = "N" Then
GoTo capts0
End If
If Left(sal0, 1) = "E" Then
GoTo capts0
End If
exp.defsal sal0, 0
exp.defcadbin cadbi
End Sub
```

```
Private Sub Command8_Click()
exp.defselmod 8
Dim ent2 As String
Dim ent1 As String
Dim ent0 As String
Dim sal0 As String
cadbi = 0
nunand3 = nunand3 + 1
strnunand3 = Str(nunand3)
snunand3 = InputBox("DAR NUMERO ASOCIADO CON ESTA
COMPUERTA NAND3", "NAND3", strnunand3)
nunand3 = Val(snunand3)
exp.defnumod nunand3
capte2:
ent2 = InputBox("DAR ENTRADA 2 ", "NAND3")
ent2 = UCase(ent2)
If varvale(ent2) = False Then
GoTo capte2
End If
If Right(ent2, 1) <> "N" Then
cadbi = cadbi + 4
End If
exp.defent ent2, 2
capte1:
ent1 = InputBox("DAR ENTRADA 1 ", "NAND3")
ent1 = UCase(ent1)
If varvale(ent1) = False Then
GoTo capte1
End If
If Right(ent1, 1) <> "N" Then
cadbi = cadbi + 2
End If
exp.defent ent1, 1
capte0:
ent0 = InputBox("DAR ENTRADA 0 ", "NAND3")
ent0 = UCase(ent0)
If varvale(ent0) = False Then
GoTo capte0
End If
If Right(ent0, 1) <> "N" Then
cadbi = cadbi + 1
End If
exp.defent ent0, 0
capts0:
sal0 = InputBox("DAR SALIDA ", "NAND3")
sal0 = UCase(sal0)
If varvals(sal0) = False Then
GoTo capts0
End If
If Right(sal0, 1) = "N" Then
GoTo capts0
End If
If Left(sal0, 1) = "E" Then
GoTo capts0
End If
exp.defsal sal0, 0
exp.defcadbin cadbi
```

```
Private Sub Command9_Click()
exp.defselmod 9
Dim ent3 As String
Dim ent2 As String
Dim ent1 As String
Dim ent0 As String
Dim sal0 As String
cadbi = 0
nunand4 = nunand4 + 1
strnunand4 = Str(nunand4)
snunand4 = InputBox("DAR NUMERO ASOCIADO CON ESTA
COMPUERTA NAND4", "NAND4", strnunand4)
nunand4 = Val(snunand4)
exp.defnumod nunand4
capte3:
ent3 = InputBox("DAR ENTRADA 3 ", "NAND4")
ent3 = UCase(ent3)
If varvale(ent3) = False Then
GoTo capte3
End If
If Right(ent3, 1) <> "N" Then
cadbi = cadbi + 8
End If
exp.defent ent3, 3
capte2:
ent2 = InputBox("DAR ENTRADA 2 ", "NAND4")
ent2 = UCase(ent2)
If varvale(ent2) = False Then
GoTo capte2
End If
If Right(ent2, 1) <> "N" Then
cadbi = cadbi + 4
End If
exp.defent ent2, 2
capte1:
ent1 = InputBox("DAR ENTRADA 1 ", "NAND4")
ent1 = UCase(ent1)
If varvale(ent1) = False Then
GoTo capte1
End If
If Right(ent1, 1) <> "N" Then
cadbi = cadbi + 2
End If
exp.defent ent1, 1
capte0:
ent0 = InputBox("DAR ENTRADA 0 ", "NAND4")
ent0 = UCase(ent0)
If varvale(ent0) = False Then
GoTo capte0
End If
If Right(ent0, 1) <> "N" Then
cadbi = cadbi + 1
End If
exp.defent ent0, 0
capts0:
sal0 = InputBox("DAR SALIDA ", "NAND4")
```

```
sal0 = UCase(sal0)
If varvals(sal0) = False Then
GoTo capts0
End If
If Right(sal0, 1) = "N" Then
GoTo capts0
End If
If Left(sal0, 1) = "E" Then
GoTo capts0
End If
exp.defsal sal0, 0
exp.defcadbin cadbi
End Sub
```

```
Private Sub Command10_Click()
exp.defselmod 10
Dim ent1 As String
Dim ent0 As String
Dim sal0 As String
cadbi = 0
nunor2 = nunor2 + 1
strnunor2 = Str(nunor2)
snunor2 = InputBox("DAR NÚMERO ASOCIADO CON ESTA
COMPUERTA NOR2", "NOR2", strnunor2)
nunor2 = Val(snunor2)
exp.defnumod nunor2
capte1:
ent1 = InputBox("DAR ENTRADA 1 ", "NOR2")
ent1 = UCase(ent1)
If varvale(ent1) = False Then
GoTo capte1
End If
If Right(ent1, 1) <> "N" Then
cadbi = cadbi + 2
End If
exp.defent ent1, 1
capte0:
ent0 = InputBox("DAR ENTRADA 0 ", "NOR2")
ent0 = UCase(ent0)
If varvale(ent0) = False Then
GoTo capte0
End If
If Right(ent0, 1) <> "N" Then
cadbi = cadbi + 1
End If
exp.defent ent0, 0
capts0:
sal0 = InputBox("DAR SALIDA ", "NOR2")
sal0 = UCase(sal0)
If varvals(sal0) = False Then
GoTo capts0
End If
If Right(sal0, 1) = "N" Then
GoTo capts0
End If
If Left(sal0, 1) = "E" Then
GoTo capts0
End If
exp.defsal sal0, 0
exp.defcadbin cadbi
End Sub
```

```
Private Sub Command11_Click()
exp.defselmod 11
Dim ent2 As String
Dim ent1 As String
Dim ent0 As String
Dim sal0 As String
cadbi = 0
nunor3 = nunor3 + 1
strnunor3 = Str(nunor3)
snunor3 = InputBox("DAR NÚMERO ASOCIADO CON ESTA
COMPUERTA NOR3", "NOR3", strnunor3)
nunor3 = Val(snunor3)
exp.defnumod nunor3
capte2:
ent2 = InputBox("DAR ENTRADA 2 ", "NOR3")
ent2 = UCase(ent2)
If varvale(ent2) = False Then
GoTo capte2
End If
If Right(ent2, 1) <> "N" Then
cadbi = cadbi + 4
End If
exp.defent ent2, 2
capte1:
ent1 = InputBox("DAR ENTRADA 1 ", "NOR3")
ent1 = UCase(ent1)
If varvale(ent1) = False Then
GoTo capte1
End If
If Right(ent1, 1) <> "N" Then
cadbi = cadbi + 2
End If
exp.defent ent1, 1
capte0:
ent0 = InputBox("DAR ENTRADA 0 ", "NOR3")
ent0 = UCase(ent0)
If varvale(ent0) = False Then
GoTo capte0
End If
If Right(ent0, 1) <> "N" Then
cadbi = cadbi + 1
End If
exp.defent ent0, 0
capts0:
sal0 = InputBox("DAR SALIDA ", "NOR3")
sal0 = UCase(sal0)
If varvals(sal0) = False Then
GoTo capts0
End If
If Right(sal0, 1) = "N" Then
GoTo capts0
End If
If Left(sal0, 1) = "E" Then
GoTo capts0
End If
exp.defsal sal0, 0
exp.defcadbin cadbi
End Sub
```

```
Private Sub Command12_Click()
exp.defselmod 12
Dim ent3 As String
Dim ent2 As String
Dim ent1 As String
Dim ent0 As String
Dim sal0 As String
cadbi = 0
nunor4 = nunor4 + 1
strnunor4 = Str(nunor4)
snunor4 = InputBox("DAR NÚMERO ASOCIADO CON ESTA
COMPUERTA NOR4", "NOR4", strnunor4)
nunor4 = Val(snunor4)
exp.defnumod nunor4
capte3:
ent3 = InputBox("DAR ENTRADA 3 ", "NOR4")
ent3 = UCase(ent3)
If varvale(ent3) = False Then
GoTo capte3
End If
If Right(ent3, 1) <> "N" Then
cadbi = cadbi + 8
End If
exp.defent ent3, 3
```

```
capte2:
ent2 = InputBox("DAR ENTRADA 2 ", "NOR4")
ent2 = UCase(ent2)
If varvale(ent2) = False Then
GoTo capte2
End If
If Right(ent2, 1) <> "N" Then
cadbi = cadbi + 4
End If
exp.defent ent2, 2
capte1:
ent1 = InputBox("DAR ENTRADA 1 ", "NOR4")
ent1 = UCase(ent1)
If varvale(ent1) = False Then
GoTo capte1
End If
If Right(ent1, 1) <> "N" Then
cadbi = cadbi + 2
End If
exp.defent ent1, 1
capte0:
ent0 = InputBox("DAR ENTRADA 0 ", "NOR4")
ent0 = UCase(ent0)
If varvale(ent0) = False Then
GoTo capte0
End If
If Right(ent0, 1) <> "N" Then
cadbi = cadbi + 1
End If
exp.defent ent0, 0
capts0:
sal0 = InputBox("DAR SALIDA ", "NOR4")
sal0 = UCase(sal0)
If varvals(sal0) = False Then
GoTo capts0
End If
If Right(sal0, 1) = "N" Then
GoTo capts0
End If
If Left(sal0, 1) = "E" Then
GoTo capts0
End If
exp.defsal sal0, 0
exp.defcadbin cadbi
End Sub
```

```
Private Sub Command13_Click()
exp.defselmod 13
Dim ent0 As String
Dim sal0 As String
cadbi = 0
nuinv = nuinv + 1
strnuinv = Str(nuinv)
snuinv = InputBox("DAR NÚMERO ASOCIADO CON ESTE
INVERSOR", "INV", strnuinv)
nuinv = Val(snuinv)
exp.defnumod nuinv
capte0:
ent0 = InputBox("DAR ENTRADA ", "INV")
ent0 = UCase(ent0)
If varvale(ent0) = False Then
GoTo capte0
End If
If Right(ent0, 1) <> "N" Then
cadbi = cadbi + 1
End If
exp.defent ent0, 0
capts0:
sal0 = InputBox("DAR SALIDA ", "INV")
sal0 = UCase(sal0)
If varvals(sal0) = False Then
GoTo capts0
End If
If Right(sal0, 1) = "N" Then
GoTo capts0
End If
If Left(sal0, 1) = "E" Then
GoTo capts0
End If
exp.defsal sal0, 0
exp.defcadbin cadbi
End Sub
```

```
Private Sub Command14_Click()
exp.defselmod 14
Dim ent0 As String
Dim sal0 As String
cadbi = 0
nuseg = nuseg + 1
strnuseg = Str(nuseg)
snuseg = InputBox("DAR NÚMERO ASOCIADO CON ESTE
SEGUIDOR", "SEG", strnuseg)
nuseg = Val(snuseg)
exp.defnumod nuseg
capte0:
ent0 = InputBox("DAR ENTRADA ", "SEG")
ent0 = UCase(ent0)
If varvale(ent0) = False Then
GoTo capte0
End If
If Right(ent0, 1) <> "N" Then
cadbi = cadbi + 1
End If
exp.defent ent0, 0
capts0:
sal0 = InputBox("DAR SALIDA ", "SEG")
sal0 = UCase(sal0)
If varvals(sal0) = False Then
GoTo capts0
End If
If Right(sal0, 1) = "N" Then
GoTo capts0
End If
If Left(sal0, 1) = "E" Then
GoTo capts0
End If
exp.defsal sal0, 0
exp.defcadbin cadbi
End Sub
```

```
Private Sub Command20_Click()
exp.defselmod 15
Dim ent3 As String
Dim ent2 As String
Dim ent1 As String
Dim ent0 As String
Dim sal0 As String
cadbi = 0
cadbit = 0
nuffars = nuffars + 1
strnuffars = Str(nuffars)
snuffars = InputBox("DAR NÚMERO ASOCIADO CON ESTE
FLIP-FLOP ASINCRONO RS", "ffars", strnuffars)
nuffars = Val(snuffars)
exp.defnumod nuffars
capte3:
ent3 = InputBox("DAR ENTRADA R ", "ffars")
ent3 = UCase(ent3)
If varvale(ent3) = False Then
GoTo capte3
End If
If Right(ent3,,1) <> "N" Then
cadbi = cadbi + 2
```

```
cadbit = cadbit + 8
End If
exp.defent ent3, 3
capte2:
ent2 = InputBox("DAR ENTRADA S ", "ffars")
ent2 = UCase(ent2)
If varvale(ent2) = False Then
GoTo capte2
End If
If Right(ent2, 1) <> "N" Then
cadbi = cadbi + 1
cadbit = cadbit + 4
End If
exp.defent ent2, 2
capte1:
ent1 = InputBox("SI DESEA PRIORIDAD PARA LA
ENTRADA SET TECLEE '1', DE OTRA FORMA TECLEE '0'",
"ffars")
'ent1 = UCase(ent1)
If ent1 <> "1" And ent1 <> "0" Then
GoTo capte1
End If
If Val(ent1) = 1 Then
cadbit = cadbit + 2
End If
exp.defent ent1, 1
capte0:
ent0 = InputBox("SI DESEA QUE LA SALIDA Q SE
INICIALICE EN CERO LÓGICO TECLEE '0', DE OTRA
FORMA TECLEE '1'", "ffars")
ent0 = UCase(ent0)
If ent0 <> "1" And ent0 <> "0" Then
GoTo capte0
End If
If Val(ent0) = 1 Then
cadbit = cadbit + 1
End If
exp.defent ent0, 0
capts0:
sal0 = InputBox("DAR SALIDA Q", "ffars")
sal0 = UCase(sal0)
If varvals(sal0) = False Then
GoTo capts0
End If
If Right(sal0, 1) = "N" Then
GoTo capts0
End If
If Left(sal0, 1) = "E" Then
GoTo capts0
End If
exp.defsal sal0, 0
exp.defcadbin cadbi
exp.defcadbit cadbit
End Sub
```

```
Private Sub Command15_Click()
exp.defselmod 16
Dim ent2 As String
Dim ent1 As String
Dim ent0 As String
Dim sal0 As String
Dim tim0 As String
cadbi = 0
cadbit = 0
numtemgen = numtemgen + 1
strnumtemgen = Str(numtemgen)
snumtemgen = InputBox("DAR NÚMERO ASOCIADO CON
ESTE TEMPORIZADOR", "tempoA", strnumtemgen)
numtemgen = Val(snumtemgen)
exp.defnumod numtemgen
capte2:
ent2 = InputBox("DAR ENTRADA H ", "tempoA")
ent2 = UCase(ent2)
If varvale(ent2) = False Then
GoTo capte2
End If
If Right(ent2, 1) <> "N" Then
cadbi = cadbi + 4
cadbit = cadbit + 8
End If
exp.defent ent2, 2
capte1:
ent1 = InputBox("DAR ENTRADA R ", "tempoA")
ent1 = UCase(ent1)
If varvale(ent1) = False Then
GoTo capte1
End If
If Right(ent1, 1) <> "N" Then
cadbi = cadbi + 2
cadbit = cadbit + 4
End If
exp.defent ent1, 1
capte0:
ent0 = InputBox("DAR ENTRADA D ", "tempoA")
ent0 = UCase(ent0)
If varvale(ent0) = False Then
GoTo capte0
End If
If Right(ent0, 1) <> "N" Then
cadbi = cadbi + 1
cadbit = cadbit + 2
End If
exp.defent ent0, 0
capts0:
sal0 = InputBox("DAR SALIDA T", "tempoA")
sal0 = UCase(sal0)
If varvals(sal0) = False Then
GoTo capts0
End If
If Right(sal0, 1) <> "N" Then
cadbit = cadbit + 1
End If
If Left(sal0, 1) = "E" Then
GoTo capts0
End If
exp.defsal sal0, 0
capti0:
tim0 = InputBox("DAR EL EL TIEMPO t PARA ESTE
TEMPORIZADOR, HH:MM:SS.CS", "tempoA")
exp.deftim tim0, 0
exp.defcadbin cadbi
exp.defcadbit cadbit
End Sub
```

```
Private Sub Command16_Click()
exp.defselmod 17
Dim ent1 As String
Dim ent0 As String
Dim sal0 As String
Dim tim0 As String
cadbi = 0
cadbit = 0
numtemgen = numtemgen + 1
strnumtemgen = Str(numtemgen)
snumtemgen = InputBox("DAR NÚMERO ASOCIADO CON
ESTE TEMPORIZADOR", "tempoC", strnumtemgen)
numtemgen = Val(snumtemgen)
exp.defnumod numtemgen
capte1:
ent1 = InputBox("DAR ENTRADA R ", "tempoC")
ent1 = UCase(ent1)
If varvale(ent1) = False Then
GoTo capte1
End If
```

```
If Right(ent1, 1) <> "N" Then
cadbi = cadbi + 2
cadbit = cadbit + 4
End If
exp.defent ent1, 1
capte0:
ent0 = InputBox("DAR ENTRADA D ", "tempoC")
ent0 = UCase(ent0)
If varvale(ent0) = False Then
GoTo capte0
End If
If Right(ent0, 1) <> "N" Then
cadbi = cadbi + 1
cadbit = cadbit + 2
End If
exp.defent ent0, 0
capts0:
sal0 = InputBox("DAR SALIDA T", "tempoC")
sal0 = UCase(sal0)
If varvals(sal0) = False Then
GoTo capts0
End If
If Right(sal0, 1) <> "N" Then
cadbit = cadbit + 1
End If
If Left(sal0, 1) = "E" Then
GoTo capts0
End If
exp.defsal sal0, 0
captt0:
tim0 = InputBox("DAR EL EL TIEMPO t PARA ESTE
TEMPORIZADOR, HH:MM:SS.CS", "tempoC")
exp.deftim tim0, 0
exp.defcadbin cadbi
exp.defcadbit cadbit
End Sub
```

```
Private Sub Command17_Click()
exp.defselmod 18
Dim ent1 As String
Dim ent0 As String
Dim sal0 As String
cadbi = 0
numtemgen = numtemgen + 1
strnumtemgen = Str(numtemgen)
snumtemgen = InputBox("DAR NÚMERO ASOCIADO CON
ESTE TEMPORIZADOR", "tempoD", strnumtemgen)
numtemgen = Val(snumtemgen)
exp.defnumod numtemgen
capte1:
ent1 = InputBox("DAR ENTRADA R ", "tempoD")
ent1 = UCase(ent1)
If varvale(ent1) = False Then
GoTo capte1
End If
If Right(ent1, 1) <> "N" Then
cadbi = cadbi + 2
End If
exp.defent ent1, 1
capte0:
ent0 = InputBox("DAR ENTRADA D ", "tempoD")
ent0 = UCase(ent0)
If varvale(ent0) = False Then
GoTo capte0
End If
If Right(ent0, 1) <> "N" Then
cadbi = cadbi + 1
End If
exp.defent ent0, 0
capts0:
sal0 = InputBox("DAR SALIDA T", "tempoD")
sal0 = UCase(sal0)
If varvals(sal0) = False Then
GoTo capts0
End If
If Right(sal0, 1) = "N" Then
GoTo capts0
End If
If Left(sal0, 1) = "E" Then
GoTo capts0
End If
exp.defsal sal0, 0
captt0:
tim0 = InputBox("DAR EL EL TIEMPO t PARA ESTE
TEMPORIZADOR, HH:MM:SS.CS", "tempoD")
exp.deftim tim0, 0
exp.defcadbin cadbi
End Sub
```

```
Private Sub Command18_Click()
exp.defselmod 19
Dim ent0 As String
Dim sal0 As String
cadbi = 0
cadbit = 0
numtemgen = numtemgen + 1
strnumtemgen = Str(numtemgen)
snumtemgen = InputBox("DAR NÚMERO ASOCIADO CON
ESTE TEMPORIZADOR", "tempoE", strnumtemgen)
numtemgen = Val(snumtemgen)
exp.defnumod numtemgen
capte0:
ent0 = InputBox("DAR ENTRADA D ", "tempoE")
ent0 = UCase(ent0)
If varvale(ent0) = False Then
GoTo capte0
End If
If Right(ent0, 1) <> "N" Then
cadbi = cadbi + 1
cadbit = cadbit + 2
End If
exp.defent ent0, 0
capts0:
sal0 = InputBox("DAR SALIDA T", "tempoE")
sal0 = UCase(sal0)
If varvals(sal0) = False Then
GoTo capts0
End If
If Right(sal0, 1) <> "N" Then
cadbit = cadbit + 1
End If
If Left(sal0, 1) = "E" Then
GoTo capts0
End If
exp.defsal sal0, 0
captt1:
tim1 = InputBox("DAR EL EL TIEMPO tm > tc PARA ESTE
TEMPORIZADOR, HH:MM:SS.CS", "tempoE")
exp.deftim tim1, 1
captt0:
tim0 = InputBox("DAR EL EL TIEMPO tc < tm PARA ESTE
TEMPORIZADOR, HH:MM:SS.CS", "tempoE")
exp.deftim tim0, 0
exp.defcadbin cadbi
exp.defcadbit cadbit
End Sub
```

FORMA A

```
Dim numod, numodspp, numodspt, numodres As Integer
Dim numodgen As Integer
Dim numodrespt As Integer
Dim selmod, cadbin, cadbit, fil As Integer
Dim vauxent(5), vauxsal(12), vauxtim(11) As String
Dim cadspp, cadspt, cadefspp, cadefsppr, cadefspt, escodsilpp,
escodsilpt As String
Dim xi(99999) As String
Dim xd(99999) As String
Dim yi(99999) As String
Dim ys(99999) As String
Dim xp(99999) As String
Dim xp1(99999) As String
Dim xp2(99999) As Single
Dim xp3(99999) As String
Dim yp(99999) As String
Dim yp1(99999) As String
Dim yp2(99999) As Single
Dim yp3(99999) As String
Dim sppresc As String
Dim cdbi, valcdbi, linspp, linspt, linlecspp, linlecspt As Integer
Dim cacorspp, cadecspp, cadecsppre, cacorspt, cadecspt,
cadecsptre As String
```

```
Public Sub ActModspp(X)
posini = 1
Do
yss = InStr(posini, X, Chr(13) & Chr(10), 1)
If yss > 0 Then
sppresc = Mid(X, posini, yss - posini)
nngen = Val(Mid(sppresc, 1, 3)) 'obtiene de "cadefspp" el numero
general de módulo
nnmod = Val(Mid(sppresc, 4, 3))
'CASE IDENTIFICADOR
Select Case nnmod
Case 1
'and2
numodgen = numodgen + 1
miy = yp2(numodres)
mix = xp2(numodres)
nnc = Val(Mid(sppresc, 11, 3))
e00 = Mid(sppresc, 14, 4)
vauxent(0) = e00
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
e11 = Mid(sppresc, 18, 4)
vauxent(1) = e11
If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
s = Mid(sppresc, 22, 4)
sss = s
vauxsal(0) = sss
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
cdbi = Mid(sppresc, 26, 2)
bin = cdbi
convierte CX
diband2 mix, miy, e0, e1, s, valcdbi, nnc
nnn = cad_n3d(nnc)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
ngen = cad_n3d(nngen)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
cadefspp = cadefspp & ngen & "001AND2" & nnn & e0 & e1 &
ss & bin & Chr(13) & Chr(10)
cadspp = cadspp & ngen & "001AND2" & nnn & e0 & e1 & ss &
bin & Chr(13) & Chr(10)
escodsilpp = escodsilpp & "          " & "AND2" & "#" & nnn & "
" & vauxent(0) & "," & vauxent(1) & "," & vauxsal(0) & "," &
bin & "," & Chr(13) & Chr(10)
xp1(numodgen) = xp2(numodres)
xp1(numodgen) = xp2(numodres)
Case 2
'and3
numodgen = numodgen + 1
miy = yp2(numodres)
mix = xp2(numodres)
nnc = Val(Mid(sppresc, 11, 3))
e00 = Mid(sppresc, 14, 4)
vauxent(0) = e00
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
e11 = Mid(sppresc, 18, 4)
vauxent(1) = e11
If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
e22 = Mid(sppresc, 22, 4)
vauxent(2) = e22
If Len(vauxent(2)) = 3 Then
e2 = vauxent(2) & " "
Else
e2 = vauxent(2)
End If
s = Mid(sppresc, 26, 4)
sss = s
vauxsal(0) = sss
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
cdbi = Mid(sppresc, 30, 3)
bin = cdbi
convierte CX
diband3 mix, miy, e0, e1, e2, s, valcdbi, nnc
nnn = cad_n3d(nnc)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
ngen = cad_n3d(nngen)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If

cadefspp = cadefspp & ngen & "002AND3" & nnn & e0 & e1 &
e2 & ss & bin & Chr(13) & Chr(10)
cadspp = cadspp & ngen & "002AND3" & nnn & e0 & e1 & e2 &
ss & bin & Chr(13) & Chr(10)
escodsilpp = escodsilpp & "          " & "AND3" & "#" & nnn & "
" & vauxent(0) & "," & vauxent(1) & "," & vauxent(2) & "," &
vauxsal(0) & "," & bin & "," & Chr(13) & Chr(10)
```

```
yp1(numodgen) = yp2(numodres)
xp1(numodgen) = xp2(numodres)
Case 3
'and4
numodgen = numodgen + 1
miy = yp2(numodres)
mix = xp2(numodres)
nnc = Val(Mid(sppresc, 11, 3))
e00 = Mid(sppresc, 14, 4)
vauxent(0) = e00
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
e11 = Mid(sppresc, 18, 4)
vauxent(1) = e11
If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
e22 = Mid(sppresc, 22, 4)
vauxent(2) = e22
If Len(vauxent(2)) = 3 Then
e2 = vauxent(2) & " "
Else
e2 = vauxent(2)
End If
e33 = Mid(sppresc, 26, 4)
vauxent(3) = e33
If Len(vauxent(3)) = 3 Then
e3 = vauxent(3) & " "
Else
e3 = vauxent(3)
End If
s = Mid(sppresc, 30, 4)
sss = s
vauxsal(0) = sss
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
cdbi = Mid(sppresc, 34, 4)
bin = cdbi
convierte CX
diband4 mix, miy, e0, e1, e2, e3, s, valcdbi, nnc
nnn = cad_n3d(nnc)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
ngen = cad_n3d(nngen)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
cadefspp = cadefspp & ngen & "003AND4" & nnn & e0 & e1 &
e2 & e3 & ss & bin & Chr(13) & Chr(10)
cadspp = cadspp & ngen & "003AND4" & nnn & e0 & e1 & e2 &
e3 & ss & bin & Chr(13) & Chr(10)
escodsilpp = escodsilpp & "          " & "AND4" & "#" & nnn & "
" & vauxent(0) & "," & vauxent(1) & "," & vauxent(2) & "," &
vauxent(3) & "," & vauxsal(0) & "," & bin & ";" & Chr(13) &
Chr(10)
yp1(numodgen) = yp2(numodres)
xp1(numodgen) = xp2(numodres)
Case 4
'or2
numodgen = numodgen + 1
miy = yp2(numodres)
mix = xp2(numodres)
nnc = Val(Mid(sppresc, 10, 3))
e00 = Mid(sppresc, 13, 4)
vauxent(0) = e00
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
e11 = Mid(sppresc, 17, 4)
vauxent(1) = e11
If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
s = Mid(sppresc, 21, 4)
sss = s
vauxsal(0) = sss
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
cdbi = Mid(sppresc, 25, 2)
bin = cdbi
convierte CX
dibor2 mix, miy, e0, e1, s, valcdbi, nnc
nnn = cad_n3d(nnc)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
ngen = cad_n3d(nngen)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
cadefspp = cadefspp & ngen & "004OR2" & nnn & e0 & e1 & ss
& bin & Chr(13) & Chr(10)
cadspp = cadspp & ngen & "004OR2" & nnn & e0 & e1 & ss &
bin & Chr(13) & Chr(10)
escodsilpp = escodsilpp & "          " & "OR2" & "#" & nnn & " "
& vauxent(0) & "," & vauxent(1) & "," & vauxsal(0) & "," & bin
& ";" & Chr(13) & Chr(10)
yp1(numodgen) = yp2(numodres)
xp1(numodgen) = xp2(numodres)
Case 5
'or3
numodgen = numodgen + 1
miy = yp2(numodres)
mix = xp2(numodres)
nnc = Val(Mid(sppresc, 10, 3))
e00 = Mid(sppresc, 13, 4)
vauxent(0) = e00
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
e11 = Mid(sppresc, 17, 4)
vauxent(1) = e11
If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
e22 = Mid(sppresc, 21, 4)
vauxent(2) = e22
If Len(vauxent(2)) = 3 Then
e2 = vauxent(2) & " "
Else
```

```
e2 = vauxent(2)
End If
s = Mid(sppresc, 25, 4)
sss = s
vauxsal(0) = sss
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
cdbi = Mid(sppresc, 29, 3)
bin = cdbi
convierte CX
nnc = Val(Mid(sppresc, 10, 3))
dibor3 mix, miy, e0, e1, e2, s, valcdbi, nnc
nnn = cad_n3d(nnc)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
ngen = cad_n3d(nngen)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
cadefspp = cadefspp & ngen & "005OR3" & nnn & e0 & e1 & e2
& ss & bin & Chr(13) & Chr(10)
cadspp = cadspp & ngen & "005OR3" & nnn & e0 & e1 & e2 &
ss & bin & Chr(13) & Chr(10)
escodsilpp = escodsilpp & "          " & "OR3" & "#" & nnn & " "
& vauxent(0) & "," & vauxent(1) & "," & vauxent(2) & "," &
vauxsal(0) & "," & bin & ";" & Chr(13) & Chr(10)
yp1(numodgen) = yp2(numodres)
xp1(numodgen) = xp2(numodres)
Case 6
'or4
numodgen = numodgen + 1
miy = yp2(numodres)
mix = xp2(numodres)
nnc = Val(Mid(sppresc, 10, 3))
e00 = Mid(sppresc, 13, 4)
vauxent(0) = e00
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
e11 = Mid(sppresc, 17, 4)
vauxent(1) = e11
If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
e22 = Mid(sppresc, 21, 4)
vauxent(2) = e22
If Len(vauxent(2)) = 3 Then
e2 = vauxent(2) & " "
Else
e2 = vauxent(2)
End If
e33 = Mid(sppresc, 25, 4)
vauxent(3) = e33
If Len(vauxent(3)) = 3 Then
e3 = vauxent(3) & " "
Else
e3 = vauxent(3)
End If
s = Mid(sppresc, 29, 4)
sss = s
vauxsal(0) = sss
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
cdbi = Mid(sppresc, 33, 4)
bin = cdbi
convierte CX
dibor4 mix, miy, e0, e1, e2, e3, s, valcdbi, nnc
nnn = cad_n3d(nnc)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
ngen = cad_n3d(nngen)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
cadefspp = cadefspp & ngen & "006OR4" & nnn & e0 & e1 & e2
& e3 & ss & bin & Chr(13) & Chr(10)
cadspp = cadspp & ngen & "006OR4" & nnn & e0 & e1 & e2 &
e3 & ss & bin & Chr(13) & Chr(10)
escodsilpp = escodsilpp & "          " & "OR4" & "#" & nnn & " "
& vauxent(0) & "," & vauxent(1) & "," & vauxent(2) & "," &
vauxent(3) & "," & vauxsal(0) & "," & bin & ";" & Chr(13) &
Chr(10)
yp1(numodgen) = yp2(numodres)
xp1(numodgen) = xp2(numodres)
Case 7
'nand2
numodgen = numodgen + 1
miy = yp2(numodres)
mix = xp2(numodres)
nnc = Val(Mid(sppresc, 12, 3))
e00 = Mid(sppresc, 15, 4)
vauxent(0) = e00
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
e11 = Mid(sppresc, 19, 4)
vauxent(1) = e11
If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
s = Mid(sppresc, 23, 4)
sss = s
vauxsal(0) = sss
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
cdbi = Mid(sppresc, 27, 2)
bin = cdbi
convierte CX
dibnand2 mix, miy, e0, e1, s, valcdbi, nnc
nnn = cad_n3d(nnc)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
ngen = cad_n3d(nngen)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
cadefspp = cadefspp & ngen & "007NAND2" & nnn & e0 & e1 &
ss & bin & Chr(13) & Chr(10)
```

```
cadspp = cadspp & ngen & "007NAND2" & nnn & e0 & e1 & ss
& bin & Chr(13) & Chr(10)
escodsilpp = escodsilpp & "          " & "NAND2" & "#" & nnn &
" " & vauxent(0) & "," & vauxent(1) & "," & vauxsal(0) & "," &
bin & ";" & Chr(13) & Chr(10)
yp1(numodgen) = yp2(numodres)
xp1(numodgen) = xp2(numodres)
Case 8
'nand3
numodgen = numodgen + 1
miy = yp2(numodres)
mix = xp2(numodres)
nnc = Val(Mid(sppresc, 12, 3))
e00 = Mid(sppresc, 15, 4)
vauxent(0) = e00
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
e11 = Mid(sppresc, 19, 4)
vauxent(1) = e11
If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
e22 = Mid(sppresc, 23, 4)
vauxent(2) = e22
If Len(vauxent(2)) = 3 Then
e2 = vauxent(2) & " "
Else
e2 = vauxent(2)
End If
s = Mid(sppresc, 27, 4)
sss = s
vauxsal(0) = sss
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
cdbi = Mid(sppresc, 31, 3)
bin = cdbi
convierte CX
dibnand3 mix, miy, e0, e1, e2, s, valcdbi, nnc
nnn = cad_n3d(nnc)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
ngen = cad_n3d(nngen)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
cadefspp = cadefspp & ngen & "008NAND3" & nnn & e0 & e1 &
e2 & ss & bin & Chr(13) & Chr(10)
cadspp = cadspp & ngen & "008NAND3" & nnn & e0 & e1 & e2
& ss & bin & Chr(13) & Chr(10)
escodsilpp = escodsilpp & "          " & "NAND3" & "#" & nnn &
" " & vauxent(0) & "," & vauxent(1) & "," & vauxent(2) & "," &
vauxsal(0) & "," & bin & ";" & Chr(13) & Chr(10)
yp1(numodgen) = yp2(numodres)
xp1(numodgen) = xp2(numodres)
Case 9
'nand4
numodgen = numodgen + 1
miy = yp2(numodres)
mix = xp2(numodres)
nnc = Val(Mid(sppresc, 12, 3))
e00 = Mid(sppresc, 15, 4)
vauxent(0) = e00
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
e11 = Mid(sppresc, 19, 4)
vauxent(1) = e11
If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
e22 = Mid(sppresc, 23, 4)
vauxent(2) = e22
If Len(vauxent(2)) = 3 Then
e2 = vauxent(2) & " "
Else
e2 = vauxent(2)
End If
e33 = Mid(sppresc, 27, 4)
vauxent(3) = e33
If Len(vauxent(3)) = 3 Then
e3 = vauxent(3) & " "
Else
e3 = vauxent(3)
End If
s = Mid(sppresc, 31, 4)
sss = s
vauxsal(0) = sss
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
cdbi = Mid(sppresc, 35, 4)
bin = cdbi
convierte CX
dibnand4 mix, miy, e0, e1, e2, e3, s, valcdbi, nnc
nnn = cad_n3d(nnc)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
ngen = cad_n3d(nngen)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
cadefspp = cadefspp & ngen & "009NAND4" & nnn & e0 & e1 &
e2 & e3 & ss & bin & Chr(13) & Chr(10)
cadspp = cadspp & ngen & "009NAND4" & nnn & e0 & e1 & e2
& e3 & ss & bin & Chr(13) & Chr(10)
escodsilpp = escodsilpp & "          " & "NAND4" & "#" & nnn &
" " & vauxent(0) & "," & vauxent(1) & "," & vauxent(2) &
vauxent(3) & "," & vauxsal(0) & "," & bin & ";" & Chr(13) &
Chr(10)
yp1(numodgen) = yp2(numodres)
xp1(numodgen) = xp2(numodres)
Case 10
'nor2
numodgen = numodgen + 1
miy = yp2(numodres)
mix = xp2(numodres)
nnc = Val(Mid(sppresc, 11, 3))
e00 = Mid(sppresc, 14, 4)
vauxent(0) = e00
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
```

```
e11 = Mid(sppresc, 18, 4)
vauxent(1) = e11
If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
s = Mid(sppresc, 22, 4)
sss = s
vauxsal(0) = sss
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
cdbi = Mid(sppresc, 26, 2)
bin = cdbi
convierte CX
dibnor2 mix, miy, e0, e1, s, valcdbi, nnc
nnn = cad_n3d(nnc)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
ngen = cad_n3d(nngen)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
cadefspp = cadefspp & ngen & "010NOR2" & nnn & e0 & e1 &
ss & bin & Chr(13) & Chr(10)
cadspp = cadspp & ngen & "010NOR2" & nnn & e0 & e1 & ss &
bin & Chr(13) & Chr(10)
escodsilpp = escodsilpp & "          " & "NOR2" & "#" & nnn & "
" & vauxent(0) & "," & vauxent(1) & "," & vauxsal(0) & "," &
bin & "," & Chr(13) & Chr(10)
yp1(numodgen) = yp2(numodres)
xp1(numodgen) = xp2(numodres)
Case 11
'nor3
numodgen = numodgen + 1
miy = yp2(numodres)
mix = xp2(numodres)
nnc = Val(Mid(sppresc, 11, 3))
e00 = Mid(sppresc, 14, 4)
vauxent(0) = e00
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
e11 = Mid(sppresc, 18, 4)
vauxent(1) = e11
If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
e22 = Mid(sppresc, 22, 4)
vauxent(2) = e22
If Len(vauxent(2)) = 3 Then
e2 = vauxent(2) & " "
Else
e2 = vauxent(2)
End If
s = Mid(sppresc, 26, 4)
sss = s
vauxsal(0) = sss
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
cdbi = Mid(sppresc, 30, 3)
bin = cdbi
convierte CX
dibnor3 mix, miy, e0, e1, e2, s, valcdbi, nnc
nnn = cad_n3d(nnc)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
ngen = cad_n3d(nngen)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
cadefspp = cadefspp & ngen & "011NOR3" & nnn & e0 & e1 &
e2 & ss & bin & Chr(13) & Chr(10)
cadspp = cadspp & ngen & "011NOR3" & nnn & e0 & e1 & e2 &
ss & bin & Chr(13) & Chr(10)
escodsilpp = escodsilpp & "          " & "NOR3" & "#" & nnn & "
" & vauxent(0) & "," & vauxent(1) & "," & vauxent(2) & "," &
vauxsal(0) & "," & bin & "," & Chr(13) & Chr(10)
yp1(numodgen) = yp2(numodres)
xp1(numodgen) = xp2(numodres)
Case 12
'nor4
numodgen = numodgen + 1
miy = yp2(numodres)
mix = xp2(numodres)
nnc = Val(Mid(sppresc, 11, 3))
e00 = Mid(sppresc, 14, 4)
vauxent(0) = e00
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
e11 = Mid(sppresc, 18, 4)
vauxent(1) = e11
If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
e22 = Mid(sppresc, 22, 4)
vauxent(2) = e22
If Len(vauxent(2)) = 3 Then
e2 = vauxent(2) & " "
Else
e2 = vauxent(2)
End If
e33 = Mid(sppresc, 26, 4)
vauxent(3) = e33
If Len(vauxent(3)) = 3 Then
e3 = vauxent(3) & " "
Else
e3 = vauxent(3)
End If
s = Mid(sppresc, 30, 4)
sss = s
vauxsal(0) = sss
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
cdbi = Mid(sppresc, 34, 2)
bin = cdbi
convierte CX
dibnor4 mix, miy, e0, e1, e2, e3, s, valcdbi, nnc
nnn = cad_n3d(nnc)
If nnn = "--" Then
```

```
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
ngen = cad_n3d(nngen)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
cadefspp = cadefspp & ngen & "012NOR4" & nnn & e0 & e1 &
e2 & e3 & ss & bin & Chr(13) & Chr(10)
cadspp = cadspp & ngen & "012NOR4" & nnn & e0 & e1 & e2 &
e3 & ss & bin & Chr(13) & Chr(10)
escodsilpp = escodsilpp & "          " & "NOR4" & "#" & nnn & "
" & vauxent(0) & "," & vauxent(1) & "," & vauxent(2) & "," &
vauxent(3) & "," & vauxsal(0) & "," & bin & ";" & Chr(13) &
Chr(10)
yp1(numodgen) = yp2(numodres)
xp1(numodgen) = xp2(numodres)
Case 13
'inversor
numodgen = numodgen - 1
miy = yp2(numodres)
mix = xp2(numodres)
nnc = Val(Mid(sppresc, 11, 3))
e00 = Mid(sppresc, 13, 4)
vauxent(0) = e00
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
s = Mid(sppresc, 17, 4)
sss = s
vauxsal(0) = sss
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
cdbi = Mid(sppresc, 21, 1)
bin = cdbi
convierte CX
dibinv mix, miy, e0, s, valcdbi, nnc
nnn = cad_n3d(nnc)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
ngen = cad_n3d(nngen)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
cadefspp = cadefspp & ngen & "013INV" & nnn & e0 & ss & bin
& Chr(13) & Chr(10)
cadspp = cadspp & ngen & "013INV" & nnn & e0 & ss & bin &
Chr(13) & Chr(10)
escodsilpp = escodsilpp & "          " & "NOT" & "#" & nnn & " "
& vauxent(0) & "," & vauxsal(0) & "," & bin & ";" & Chr(13) &
Chr(10)
yp1(numodgen) = yp2(numodres)
xp1(numodgen) = xp2(numodres)
Case 14
'seguidor
numodgen = numodgen + 1
miy = yp2(numodres)
mix = xp2(numodres)
nnc = Val(Mid(sppresc, 11, 3))
e00 = Mid(sppresc, 13, 4)
vauxent(0) = e00
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
s = Mid(sppresc, 17, 4)
sss = s
vauxsal(0) = sss
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
cdbi = Mid(sppresc, 21, 1)
bin = cdbi
convierte CX
dibseg mix, miy, e0, s, valcdbi, nnc
nnn = cad_n3d(nnc)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
ngen = cad_n3d(nngen)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
cadefspp = cadefspp & ngen & "014SEG" & nnn & e0 & ss & bin
& Chr(13) & Chr(10)
cadspp = cadspp & ngen & "014SEG" & nnn & e0 & ss & bin &
Chr(13) & Chr(10)
escodsilpp = escodsilpp & "          " & "SEG" & "#" & nnn & " "
& vauxent(0) & "," & vauxent(1) & "," & vauxent(2) & "," &
Chr(10)
yp1(numodgen) = yp2(numodres)
xp1(numodgen) = xp2(numodres)
Case 15
'FFARS
numodgen = numodgen + 1
miy = yp2(numodres)
mix = xp2(numodres)
nnc = Val(Mid(sppresc, 12, 3))
e22 = Mid(sppresc, 15, 4)
vauxent(2) = e22
If Len(vauxent(2)) = 3 Then
e2 = vauxent(2) & " "
Else
e2 = vauxent(2)
End If
e33 = Mid(sppresc, 19, 4)
vauxent(3) = e33
If Len(vauxent(3)) = 3 Then
e3 = vauxent(3) & " "
Else
e3 = vauxent(3)
End If
s = Mid(sppresc, 23, 4)
sss = s
vauxsal(0) = sss
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
cdbit = Mid(sppresc, 27, 4)
bin = cdbit

cdbi = Mid(sppresc, 27, 2)
convierte CX
dibFFars mix, miy, e2, e3, s, valcdbi, nnc
nnn = cad_n3d(nnc)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
```

```
End If
ngen = cad_n3d(nngen)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
cadefspp = cadefspp & ngen & "015FFARS" & nnn & e2 & e3 &
ss & bin & Chr(13) & Chr(10)
cadspp = cadspp & ngen & "015FFARS" & nnn & e2 & e3 & ss
& bin & Chr(13) & Chr(10)
escedsilpp = escedsilpp & "          " & "FFARS" & "#" & nnn & "
" & vauxent(2) & "." & vauxent(3) & "." & vauxsal(0) & "." &
bin & "." & Chr(13) & Chr(10)
yp1(numedgen) = yp2(numedres)
xp1(numedgen) = xp2(numedres)
End Select
'FIN IDENTIFICACIÓN Y DIBUJO
posini = yss + 2
Else
Exit Sub
End If
Loop Until yss = 0
'numedres = 0
End Sub
```

```
Public Function cad_n3d(X) As String
'Retorna un string de longitud 3, que representa a
'un número positivo x comprendido entre cero y 999.
'En caso de que el número esté fuera de ese intervalo
'retorna "--".
If X < 0 Or X > 999 Then
cad_n3d = "--"
Exit Function
End If
xa = Right(Str(X), Len(Str(X)) - 1)
If Len(xa) = 1 Then
cad_n3d = "0" & "0" & xa
Exit Function
End If
If Len(xa) = 2 Then
cad_n3d = "0" & xa
Exit Function
End If
cad_n3d = xa
End Function
```

```
Public Function cad_n5d(X) As String
'Retorna un string de longitud 5, que representa a
'un número positivo x comprendido entre cero y 999.
'En caso de que el número esté fuera de ese intervalo
'retorna "--".
If X < 0 Or X > 11985 Then
cad_n5d = "--"
Exit Function
End If
xa = Right(Str(X), Len(Str(X)) - 1)
If Len(xa) = 1 Then
cad_n5d = "0" & "0" & "0" & "0" & xa
Exit Function
End If
If Len(xa) = 2 Then
cad_n5d = "0" & "0" & "0" & xa
Exit Function
End If
If Len(xa) = 3 Then
cad_n5d = "0" & "0" & xa
Exit Function
End If
If Len(xa) = 4 Then
cad_n5d = "0" & xa
Exit Function
End If
cad_n5d = xa
End Function
```

```
Public Sub convierte(CX)
valedbi = 0
For i = Len(edbi) To 1 Step -1
mibit = Val(Mid(edbi, i, 1))
valedbi = valedbi + mibit * 2 ^ (Len(edbi) - i)
Next i
End Sub
```

```
Public Sub defcadbin(X)
cadbin = X
End Sub
```

```
Public Sub defcadbit(X1)
cadbit = X1
End Sub
```

```
Public Sub defent(X, y)
'Define la variable vauxent(y) como x
If Right(X, 1) = "N" Then
X = Left(X, Len(X) - 1)
End If
vauxent(y) = X
End Sub
```

```
Public Sub defnumod(X)
'Define número asociado con modulo a colocar.
numed = X
End Sub
```

```
Public Sub defsal(X, y)
'Define la variable vauxsal(y) como x
If Right(X, 1) = "N" Then
X = Left(X, Len(X) - 1)
End If
vauxsal(y) = X
End Sub
```

```
Public Sub defselmod(xs As Integer)
selmod = xs
End Sub
```

```
Public Sub deftint(XT, y)
vauxtint(y) = XT
End Sub
```

```
Public Sub cibasd2(xx, yy, e0, e1, s, cadbit, nn)
xx(numedgen) = xx
xs(numedgen) = xx - 525
yi(numedgen) = yy - 525
ys(numedgen) = yy - 525
Circle (xx, yy), 525, , 4.7, 1.57
Line (xx, yy - 525)-(xx, yy + 525)
CurrentX = xx - 130
CurrentY = yy - 120
Print nn
CurrentX = xx - 540
CurrentY = yy - 360
Print e0
CurrentX = xx - 540
CurrentY = yy + 105
Print e1
CurrentX = xx + 555
CurrentY = yy - 120
Print s

CurrentX = xx - 80
CurrentY = yy - 360
Print 0
CurrentX = xx - 80
CurrentY = yy + 105
Print 1
Select Case cadbit
Case 0
Circle (xx - 45, yy - 245), 45
Circle (xx - 45, yy + 210), 45
Case 1
```

```
Circle (xx - 45, yy + 210), 45
Case 2
Circle (xx - 45, yy - 245), 45
End Select
End Sub
```

```
Public Sub diband3(xx, yy, e0, e1, e2, s, cadbin, nn)
xi(numodgen) = xx
xd(numodgen) = xx + 575
yi(numodgen) = yy - 575
ys(numodgen) = yy + 575
Circle (xx, yy), 575, , 4.7, 1.57
Line (xx, yy - 575)-(xx, yy + 575)
CurrentX = xx + 180
CurrentY = yy - 120
Print nn
CurrentX = xx - 540
CurrentY = yy - 510
Print e0
CurrentX = xx - 540
CurrentY = yy - 100
Print e1
CurrentX = xx - 540
CurrentY = yy + 300
Print e2
CurrentX = xx + 655
CurrentY = yy - 120
Print s
CurrentX = xx - 80
CurrentY = yy - 510
Print 0
CurrentX = xx - 80
CurrentY = yy - 100
Print 1
CurrentX = xx - 80
CurrentY = yy + 300
Print 2
Select Case cadbin
Case 0
Circle (xx - 45, yy + 400), 45
Circle (xx - 45, yy), 45
Circle (xx - 45, yy - 400), 45
Case 1
Circle (xx - 45, yy), 45
Circle (xx - 45, yy + 400), 45
Case 2
Circle (xx - 45, yy - 400), 45
Circle (xx - 45, yy + 400), 45
Case 3
Circle (xx - 45, yy + 400), 45
Case 4
Circle (xx - 45, yy - 400), 45
Circle (xx - 45, yy), 45
Case 5
Circle (xx - 45, yy), 45
Case 6
Circle (xx - 45, yy - 400), 45
End Select
End Sub
```

```
Public Sub diband4(xx, yy, e0, e1, e2, e3, s, cadbin, nn)
xi(numodgen) = xx
xd(numodgen) = xx + 625
yi(numodgen) = yy - 625
ys(numodgen) = yy + 625
Circle (xx, yy), 625, , 4.7, 1.57
Line (xx, yy - 625)-(xx, yy + 625)
CurrentX = xx + 200
CurrentY = yy - 120
Print nn
CurrentX = xx - 540
CurrentY = yy - 600
Print e0
CurrentX = xx - 540
CurrentY = yy - 275
Print e1
CurrentX = xx - 540
CurrentY = yy + 75
Print e2
CurrentX = xx - 540
CurrentY = yy + 400
Print e3
CurrentX = xx + 700
CurrentY = yy - 120
Print s
CurrentX = xx - 80
CurrentY = yy - 600
Print 0
CurrentX = xx - 80
CurrentY = yy - 275
Print 1
CurrentX = xx - 80
CurrentY = yy + 75
Print 2
CurrentX = xx - 80
CurrentY = yy + 400
Print 3
Select Case cadbin
Case 0
Circle (xx - 45, yy - 490), 45
Circle (xx - 45, yy - 165), 45
Circle (xx - 45, yy + 175), 45
Circle (xx - 45, yy + 510), 45
Case 1
Circle (xx - 45, yy - 165), 45
Circle (xx - 45, yy + 175), 45
Circle (xx - 45, yy + 510), 45
Case 2
Circle (xx - 45, yy - 490), 45
Circle (xx - 45, yy + 175), 45
Circle (xx - 45, yy + 510), 45
Case 3
Circle (xx - 45, yy + 175), 45
Circle (xx - 45, yy + 510), 45
Case 4
Circle (xx - 45, yy - 490), 45
Circle (xx - 45, yy - 165), 45
Circle (xx - 45, yy + 510), 45
Case 5
Circle (xx - 45, yy - 165), 45
Circle (xx - 45, yy + 510), 45
Case 6
Circle (xx - 45, yy - 490), 45
Circle (xx - 45, yy + 510), 45
Case 7
Circle (xx - 45, yy + 510), 45
Case 8
Circle (xx - 45, yy - 490), 45
Circle (xx - 45, yy - 165), 45
Circle (xx - 45, yy + 175), 45
Case 9
Circle (xx - 45, yy - 165), 45
Circle (xx - 45, yy + 175), 45
Case 10
Circle (xx - 45, yy - 490), 45
Circle (xx - 45, yy + 175), 45
Case 11
Circle (xx - 45, yy + 175), 45
Case 12
Circle (xx - 45, yy - 490), 45
Circle (xx - 45, yy - 165), 45
Case 13
Circle (xx - 45, yy - 165), 45
Case 14
Circle (xx - 45, yy - 490), 45
```

```
End Select
End Sub
```

```
Public Sub dibcontev(xx, yy, e0, e1, e2, s, cadbin, nn)
xi(numodgen) = xx
xd(numodgen) = xx + 575
yi(numodgen) = yy - 575
ys(numodgen) = yy + 575
Line (xx, yy - 700)-(xx, yy + 700)
Line (xx + 1100, yy - 700)-(xx + 1100, yy + 700)
Line (xx, yy + 700)-(xx + 1100, yy + 700)
Line (xx, yy - 700)-(xx + 1100, yy - 700)
CurrentX = xx + 400
CurrentY = yy - 120
Print nn
CurrentX = xx + 250
CurrentY = yy - 280
Print "ContEv"
CurrentX = xx - 540
CurrentY = yy - 510
Print e0
CurrentX = xx - 540
CurrentY = yy - 100
Print e1
CurrentX = xx - 540
CurrentY = yy + 300
Print e2
CurrentX = xx + 1200
CurrentY = yy - 120
Print s
CurrentX = xx + 50
CurrentY = yy - 510
Print "D"
CurrentX = xx + 50
CurrentY = yy - 100
Print "C"
CurrentX = xx + 50
CurrentY = yy + 300
Print "R"
CurrentX = xx + 880
CurrentY = yy - 100
Print "TF"
Select Case cadbin
Case 0
'fleabD
Line (xx + 200, yy - 330)-(xx + 200, yy - 480)
Line (xx + 200, yy - 330)-(xx + 140, yy - 390)
Line (xx + 200, yy - 330)-(xx + 250, yy - 390)
'R
Line (xx + 50, yy + 290)-(xx + 150, yy + 290)
'C
Line (xx + 50, yy - 110)-(xx + 150, yy - 110)
Case 1
'fleartD
Line (xx + 200, yy - 330)-(xx + 200, yy - 470)
Line (xx + 200, yy - 470)-(xx + 140, yy - 410)
Line (xx + 200, yy - 470)-(xx + 250, yy - 410)
'R
Line (xx + 50, yy + 290)-(xx + 150, yy + 290)
'C
Line (xx + 50, yy - 110)-(xx + 150, yy - 110)
Case 2
'fleabD
Line (xx + 200, yy - 330)-(xx + 200, yy - 480)
Line (xx + 200, yy - 330)-(xx + 140, yy - 390)
Line (xx + 200, yy - 330)-(xx + 250, yy - 390)
'R
Line (xx + 50, yy + 290)-(xx + 150, yy + 290)
Case 3
'fleartD
Line (xx + 200, yy - 330)-(xx + 200, yy - 470)
Line (xx + 200, yy - 470)-(xx + 140, yy - 410)
Line (xx + 200, yy - 470)-(xx + 250, yy - 410)
'R
Line (xx + 50, yy + 290)-(xx + 150, yy + 290)
Case 4
'fleabD
Line (xx + 200, yy - 330)-(xx + 200, yy - 480)
Line (xx + 200, yy - 330)-(xx + 140, yy - 390)
Line (xx + 200, yy - 330)-(xx + 250, yy - 390)
'C
Line (xx + 50, yy - 110)-(xx + 150, yy - 110)
Case 5
'fleartD
Line (xx + 200, yy - 330)-(xx + 200, yy - 470)
Line (xx + 200, yy - 470)-(xx + 140, yy - 410)
Line (xx + 200, yy - 470)-(xx + 250, yy - 410)
'C
Line (xx + 50, yy - 110)-(xx + 150, yy - 110)
Case 6
'fleabD
Line (xx + 200, yy - 330)-(xx + 200, yy - 480)
Line (xx + 200, yy - 330)-(xx + 140, yy - 390)
Line (xx + 200, yy - 330)-(xx + 250, yy - 390)
Case 7
'fleartD
Line (xx + 200, yy - 330)-(xx + 200, yy - 470)
Line (xx + 200, yy - 470)-(xx + 140, yy - 410)
Line (xx + 200, yy - 470)-(xx + 250, yy - 410)
End Select
End Sub
```

```
Public Sub dibFFars(xx, yy, e2, e3, s, cadbin, nn)
xi(numodgen) = xx
xd(numodgen) = xx + 575
yi(numodgen) = yy - 575
ys(numodgen) = yy + 575
Line (xx, yy - 600)-(xx, yy + 600)
Line (xx + 1300, yy - 600)-(xx + 1300, yy + 600)
Line (xx, yy + 600)-(xx + 1300, yy + 600)
Line (xx, yy - 600)-(xx + 1300, yy - 600)
CurrentX = xx - 500
CurrentY = yy - 20
Print nn
CurrentX = xx + 350
CurrentY = yy - 150
Print "FFARS"
CurrentX = xx - 540
CurrentY = yy - 500
Print e2
CurrentX = xx - 540
CurrentY = yy + 300
Print e3
CurrentX = xx + 1400
CurrentY = yy - 400
Print s
CurrentX = xx + 50
CurrentY = yy - 500
Print "S"
CurrentX = xx + 50
CurrentY = yy + 300
Print "R"

CurrentX = xx + 1200
CurrentY = yy - 400
Print "Q"
Select Case cadbin
Case 0
Line (xx + 50, yy - 485)-(xx + 150, yy - 485)
Line (xx + 50, yy + 300)-(xx + 150, yy + 300)
Case 1
Line (xx + 50, yy + 300)-(xx + 150, yy + 300)
Case 2
Line (xx + 50, yy - 485)-(xx + 150, yy - 485)
```

```
End Select
End Sub
```

```
Public Sub dibinv(xx, yy, e0, s, cadbin, nn)
xi(numodgen) = xx - 400
xd(numodgen) = xx - 300
yi(numodgen) = yy - 400
ys(numodgen) = yy + 445
Line (xx, yy - 400)-(xx, yy + 400)
'Line (xx - 400, yy)-(xx, yy)
Line (xx, yy - 400)-(xx + 300, yy)
Line (xx, yy + 400)-(xx + 300, yy)
Circle (xx + 345, yy), 45
CurrentX = xx
CurrentY = yy - 120
Print nn
CurrentX = xx - 450
CurrentY = yy - 100
Print e0
CurrentX = xx + 455
CurrentY = yy - 120
Print s
Select Case cadbin
Case 0
'Circle (xx - 45, yy - 245), 45
'Circle (xx - 45, yy + 210), 45
Case 1
'Circle (xx - 45, yy + 210), 45
Case 2
'Circle (xx - 45, yy - 245), 45
End Select
End Sub
```

```
Public Sub dibnand2(xx, yy, e0, e1, s, cadbin, nn)
xi(numodgen) = xx
xd(numodgen) = xx + 525
yi(numodgen) = yy - 525
ys(numodgen) = yy + 525
Circle (xx + 570, yy), 40, , 0, 0
Circle (xx, yy), 525, , 4.7, 1.57
Line (xx, yy - 525)-(xx, yy + 525)
CurrentX = xx + 130
CurrentY = yy - 120
Print nn
CurrentX = xx - 540
CurrentY = yy - 360
Print e0
CurrentX = xx - 540
CurrentY = yy + 105
Print e1
CurrentX = xx + 700
CurrentY = yy - 120
Print s
CurrentX = xx - 80
CurrentY = yy - 360
Print 0
CurrentX = xx - 80
CurrentY = yy + 105
Print 1
Select Case cadbin
Case 0
Circle (xx - 45, yy - 245), 45
Circle (xx - 45, yy + 210), 45
Case 1
Circle (xx - 45, yy + 210), 45
Case 2
Circle (xx - 45, yy - 245), 45
End Select
End Sub
```

```
Public Sub dibnand3(xx, yy, e0, e1, e2, s, cadbin, nn)
xi(numodgen) = xx
xd(numodgen) = xx + 575
yi(numodgen) = yy - 575
ys(numodgen) = yy + 575
Circle (xx + 630, yy), 40, , 0, 0
Circle (xx, yy), 575, , 4.7, 1.57
Line (xx, yy - 575)-(xx, yy + 575)
CurrentX = xx + 180
CurrentY = yy - 120
Print nn
CurrentX = xx - 540
CurrentY = yy - 510
Print e0
CurrentX = xx - 540
CurrentY = yy - 100
Print e1
CurrentX = xx - 540
CurrentY = yy + 300
Print e2
CurrentX = xx + 755
CurrentY = yy - 120
Print s
CurrentX = xx - 80
CurrentY = yy - 510
Print 0
CurrentX = xx - 80
CurrentY = yy - 100
Print 1
CurrentX = xx - 80
CurrentY = yy + 300
Print 2
Select Case cadbin
Case 0
Circle (xx - 45, yy + 400), 45
Circle (xx - 45, yy), 45
Circle (xx - 45, yy - 400), 45
Case 1
Circle (xx - 45, yy), 45
Circle (xx - 45, yy + 400), 45
Case 2
Circle (xx - 45, yy - 400), 45
Circle (xx - 45, yy + 400), 45
Case 3
Circle (xx - 45, yy + 400), 45
Case 4
Circle (xx - 45, yy - 400), 45
Circle (xx - 45, yy), 45
Case 5
Circle (xx - 45, yy), 45
Case 6
Circle (xx - 45, yy - 400), 45
End Select
End Sub
```

```
Public Sub dibnand4(xx, yy, e0, e1, e2, e3, s, cadbin, nn)
xi(numodgen) = xx
xd(numodgen) = xx + 625
yi(numodgen) = yy - 625
ys(numodgen) = yy + 625
Circle (xx + 690, yy), 40, , 0, 0
Circle (xx, yy), 625, , 4.7, 1.57
Line (xx, yy - 625)-(xx, yy + 625)
CurrentX = xx + 200
CurrentY = yy - 120
Print nn
CurrentX = xx - 540
CurrentY = yy - 600
Print e0
CurrentX = xx - 540
CurrentY = yy - 275
Print e1
CurrentX = xx - 540
CurrentY = yy + 75
Print e2
CurrentX = xx - 540
```

```
CurrentY = yy + 400
Print e3
CurrentX = xx + 800
CurrentY = yy - 120
Print s
CurrentX = xx - 80
CurrentY = yy - 600
Print 0
CurrentX = xx - 80
CurrentY = yy - 275
Print 1
CurrentX = xx - 80
CurrentY = yy + 75
Print 2
CurrentX = xx - 80
CurrentY = yy + 400
Print 3
Select Case cadbin
Case 0
Circle (xx - 45, yy - 490), 45
Circle (xx - 45, yy - 165), 45
Circle (xx - 45, yy + 175), 45
Circle (xx - 45, yy + 510), 45
Case 1
Circle (xx - 45, yy - 165), 45
Circle (xx - 45, yy + 175), 45
Circle (xx - 45, yy + 510), 45
Case 2
Circle (xx - 45, yy - 490), 45
Circle (xx - 45, yy + 175), 45
Circle (xx - 45, yy + 510), 45
Case 3
Circle (xx - 45, yy + 175), 45
Circle (xx - 45, yy + 510), 45
Case 4
Circle (xx - 45, yy - 490), 45
Circle (xx - 45, yy - 165), 45
Circle (xx - 45, yy + 510), 45
Case 5
Circle (xx - 45, yy - 165), 45
Circle (xx - 45, yy + 510), 45
Case 6
Circle (xx - 45, yy - 490), 45
Circle (xx - 45, yy + 510), 45
Case 7
Circle (xx - 45, yy + 510), 45
Case 8
Circle (xx - 45, yy - 490), 45
Circle (xx - 45, yy - 165), 45
Circle (xx - 45, yy + 175), 45
Case 9
Circle (xx - 45, yy - 165), 45
Circle (xx - 45, yy + 175), 45
Case 10
Circle (xx - 45, yy - 490), 45
Circle (xx - 45, yy + 175), 45
Case 11
Circle (xx - 45, yy + 175), 45
Case 12
Circle (xx - 45, yy - 490), 45
Circle (xx - 45, yy - 165), 45
Case 13
Circle (xx - 45, yy - 165), 45
Case 14
Circle (xx - 45, yy - 490), 45
End Select
End Sub
```

```
Public Sub dibnor2(xx, yy, e0, e1, s, cadbin, nn)
xi(numodgen) = xx
xd(numodgen) = xx + 525
yi(numodgen) = yy - 525
ys(numodgen) = yy + 525
Circle (xx + 570, yy), 40, , 0, 0
Circle (xx, yy), 525, , 4.7, 1.57
Circle (xx - 900, yy), 1050, , 5.7595, 0.5235
CurrentX = xx + 200
CurrentY = yy - 120
Print nn
CurrentX = xx - 400
CurrentY = yy - 360
Print e0
CurrentX = xx - 400
CurrentY = yy + 110
Print e1
CurrentX = xx + 700
CurrentY = yy - 120
Print s
CurrentX = xx + 50
CurrentY = yy - 360
Print 0
CurrentX = xx + 50
CurrentY = yy + 110
Print 1
Select Case cadbin
Case 0
Circle (xx + 60, yy - 250), 45
Circle (xx + 70, yy + 215), 45
Case 1
Circle (xx + 70, yy + 215), 45
Case 2
Circle (xx + 60, yy - 250), 45
End Select
End Sub
```

```
Public Sub dibnor3(xx, yy, e0, e1, e2, s, cadbin, nn)
xi(numodgen) = xx
xd(numodgen) = xx + 575
yi(numodgen) = yy - 575
ys(numodgen) = yy + 575
Circle (xx + 630, yy), 40, , 0, 0
Circle (xx, yy), 575, , 4.7, 1.65
Circle (xx - 900, yy), 1050, , 5.7119, 0.5711
CurrentX = xx + 250
CurrentY = yy - 120
Print nn
CurrentX = xx - 300
CurrentY = yy - 500
Print e0
CurrentX = xx - 400
CurrentY = yy - 100
Print e1
CurrentX = xx - 400
CurrentY = yy + 300
Print e2
CurrentX = xx + 755
CurrentY = yy - 120
Print s
CurrentX = xx
CurrentY = yy - 500
Print 0
CurrentX = xx + 60
CurrentY = yy - 100
Print 1

CurrentX = xx
CurrentY = yy + 300
Print 2
Select Case cadbin
Case 0
Circle (xx + 10, yy + 400), 45
Circle (xx + 90, yy), 45
Circle (xx + 10, yy - 400), 45
Case 1
```

```
Circle (xx - 90, yy), 45
Circle (xx + 10, yy + 400), 45
Case 2
Circle (xx + 10, yy - 400), 45
Circle (xx + 10, yy + 400), 45
Case 3
Circle (xx + 10, yy + 400), 45
Case 4
Circle (xx + 10, yy - 400), 45
Circle (xx + 90, yy), 45
Case 5
Circle (xx + 90, yy), 45
Case 6
Circle (xx + 10, yy - 400), 45
End Select
End Sub
```

```
Public Sub dibnor4(xx, yy, e0, e1, e2, e3, s, cadbin, nn)
xi(numodgen) = xx
xd(numodgen) = xx + 625
yi(numodgen) = yy - 625
ys(numodgen) = yy + 625
Circle (xx + 690, yy), 40, , 0, 0
Circle (xx, yy), 625, , 4.65, 1.65
Circle (xx - 900, yy), 1050, , 5.65, 0.64
CurrentX = xx - 300
CurrentY = yy - 120
Print nn
CurrentX = xx - 500
CurrentY = yy - 600
Print e0
CurrentX = xx - 500
CurrentY = yy - 265
Print e1
CurrentX = xx - 500
CurrentY = yy + 75
Print e2
CurrentX = xx - 500
CurrentY = yy + 400
Print e3
CurrentX = xx + 800
CurrentY = yy - 120
Print s
CurrentX = xx - 60
CurrentY = yy - 600
Print 0
CurrentX = xx + 50
CurrentY = yy - 265
Print 1
CurrentX = xx - 50
CurrentY = yy + 75
Print 2
CurrentX = xx - 60
CurrentY = yy + 400
Print 3
Select Case cadbin
Case 0
Circle (xx - 45, yy - 490), 45
Circle (xx + 80, yy - 165), 45
Circle (xx + 80, yy + 175), 45
Circle (xx - 45, yy + 510), 45
Case 1
Circle (xx + 80, yy - 165), 45
Circle (xx + 80, yy + 175), 45
Circle (xx - 45, yy + 510), 45
Case 2
Circle (xx - 45, yy - 490), 45
Circle (xx + 80, yy + 175), 45
Circle (xx - 45, yy + 510), 45
Case 3
Circle (xx + 80, yy + 175), 45
Circle (xx - 45, yy + 510), 45
Case 4
Circle (xx - 45, yy - 490), 45
Circle (xx + 80, yy - 165), 45
Circle (xx - 45, yy + 510), 45
Case 5
Circle (xx + 80, yy - 165), 45
Circle (xx - 45, yy + 510), 45
Case 6
Circle (xx - 45, yy - 490), 45
Circle (xx - 45, yy + 510), 45
Case 7
Circle (xx - 45, yy + 510), 45
Case 8
Circle (xx - 45, yy - 490), 45
Circle (xx + 80, yy - 165), 45
Circle (xx + 80, yy + 175), 45
Case 9
Circle (xx + 80, yy - 165), 45
Circle (xx + 80, yy + 175), 45
Case 10
Circle (xx - 45, yy - 490), 45
Circle (xx + 80, yy + 175), 45
Case 11
Circle (xx + 80, yy + 175), 45
Case 12
Circle (xx - 45, yy - 490), 45
Circle (xx + 80, yy - 165), 45
Case 13
Circle (xx + 80, yy - 165), 45
Case 14
Circle (xx - 45, yy - 490), 45
End Select
End Sub
```

```
Public Sub dibor2(xx, yy, e0, e1, s, cadbin, nn)
xi(numodgen) = xx
xd(numodgen) = xx + 525
yi(numodgen) = yy - 525
ys(numodgen) = yy + 525
Circle (xx, yy), 525, , 4.7, 1.57
Circle (xx - 900, yy), 1050, , 5.7595, 0.5235
CurrentX = xx + 200
CurrentY = yy - 120
Print nn
CurrentX = xx - 400
CurrentY = yy - 360
Print e0
CurrentX = xx - 400
CurrentY = yy - 110
Print e1
CurrentX = xx - 600
CurrentY = yy - 140
Print s
CurrentX = xx - 50
CurrentY = yy - 360
Print 0
CurrentX = xx - 50
CurrentY = yy - 110
Print 1
Select Case cadbin
Case 0
Circle (xx + 60, yy - 250), 45
Circle (xx - 70, yy + 215), 45
Case 1
Circle (xx + 70, yy + 215), 45
Case 2
Circle (xx + 60, yy - 250), 45
End Select
End Sub
```

```
Public Sub dibor3(xx, yy, e0, e1, e2, s, cadbin, nn)
xi(numodgen) = xx
xd(numodgen) = xx + 575
```

```
yi(numodgen) = yy - 575
ys(numodgen) = yy + 575
Circle (xx, yy), 575, , 4.7, 1.65
Circle (xx - 900, yy), 1050, , 5.7119, 0.5711
CurrentX = xx + 250
CurrentY = yy - 120
Print nn
CurrentX = xx - 400
CurrentY = yy - 500
Print e0
CurrentX = xx - 400
CurrentY = yy - 100
Print e1
CurrentX = xx - 400
CurrentY = yy + 300
Print e2
CurrentX = xx + 655
CurrentY = yy - 120
Print s
CurrentX = xx
CurrentY = yy - 500
Print 0
CurrentX = xx + 60
CurrentY = yy - 100
Print 1
CurrentX = xx
CurrentY = yy + 300
Print 2
Select Case cadbin
Case 0
Circle (xx + 10, yy + 400), 45
Circle (xx + 90, yy), 45
Circle (xx + 10, yy - 400), 45
Case 1
Circle (xx + 90, yy), 45
Circle (xx + 10, yy + 400), 45
Case 2
Circle (xx + 10, yy + 400), 45
Circle (xx + 10, yy + 400), 45
Case 3
Circle (xx + 10, yy + 400), 45
Case 4
Circle (xx + 10, yy + 400), 45
Circle (xx + 90, yy), 45
Case 5
Circle (xx + 90, yy), 45
Case 6
Circle (xx + 10, yy + 400), 45
End Select
End Sub
```

```
Public Sub dibor4(xx, yy, e0, e1, e2, e3, s, cadbin, nn)
xi(numodgen) = xx
xd(numodgen) = xx + 625
yi(numodgen) = yy - 625
ys(numodgen) = yy + 625
Circle (xx, yy), 625, , 4.65, 1.65
Circle (xx - 900, yy), 1050, , 5.65, 0.64
CurrentX = xx + 300
CurrentY = yy - 120
Print nn
CurrentX = xx - 500
CurrentY = yy - 600
Print e0
CurrentX = xx - 500
CurrentY = yy - 265
Print e1
CurrentX = xx - 500
CurrentY = yy + 75
Print e2
CurrentX = xx - 500
CurrentY = yy + 400
Print e3
CurrentX = xx + 800
CurrentY = yy - 120
Print s
CurrentX = xx - 60
CurrentY = yy - 600
Print 0
CurrentX = xx + 50
CurrentY = yy - 265
Print 1
CurrentX = xx + 50
CurrentY = yy + 75
Print 2
CurrentX = xx - 60
CurrentY = yy + 400
Print 3
Select Case cadbin
Case 0
Circle (xx - 45, yy - 490), 45
Circle (xx + 80, yy - 165), 45
Circle (xx + 80, yy + 175), 45
Circle (xx - 45, yy + 510), 45
Case 1
Circle (xx + 80, yy - 165), 45
Circle (xx + 80, yy + 175), 45
Circle (xx - 45, yy + 510), 45
Case 2
Circle (xx - 45, yy - 490), 45
Circle (xx + 80, yy + 175), 45
Circle (xx - 45, yy + 510), 45
Case 3
Circle (xx + 80, yy + 175), 45
Circle (xx - 45, yy + 510), 45
Case 4
Circle (xx - 45, yy - 490), 45
Circle (xx + 80, yy - 165), 45
Circle (xx - 45, yy + 510), 45
Case 5
Circle (xx + 80, yy - 165), 45
Circle (xx - 45, yy + 510), 45
Case 6
Circle (xx - 45, yy - 490), 45
Circle (xx - 45, yy + 510), 45
Case 7
Circle (xx - 45, yy + 510), 45
Case 8
Circle (xx - 45, yy - 490), 45
Circle (xx + 80, yy - 165), 45
Circle (xx + 80, yy + 175), 45
Case 9
Circle (xx + 80, yy - 165), 45
Circle (xx + 80, yy + 175), 45
Case 10
Circle (xx - 45, yy - 490), 45
Circle (xx + 80, yy + 175), 45
Case 11
Circle (xx + 80, yy + 175), 45
Case 12
Circle (xx - 45, yy - 490), 45
Circle (xx + 80, yy - 165), 45
Case 13
Circle (xx + 80, yy - 165), 45
Case 14
Circle (xx - 45, yy - 490), 45
End Select
End Sub
```

```
Public Sub dibor8(xx, yy, e0, e1, e2, e3, s, cadbin, nn)
xi(numodgen) = xx - 800
xd(numodgen) = xx + 800
yi(numodgen) = yy - 800
ys(numodgen) = yy + 800
```

```
Line (xx, yy - 400)-(xx, yy + 400)
Line (xx, yy - 400)-(xx + 300, yy + 10)
Line (xx, yy + 400)-(xx + 300, yy - 10)
CurrentX = xx
CurrentY = yy - 120
Print nn
CurrentX = xx - 450
CurrentY = yy - 100
Print e0
CurrentX = xx + 400
CurrentY = yy - 120
Print s
Select Case cadbin
End Sub
```

```
Public Sub dibtempoa(xx, yy, e0, e1, e2, s, t, cadbin, nn)
xi(numodgen) = xx
xd(numodgen) = xx + 575
yi(numodgen) = yy - 575
ys(numodgen) = yy + 575
Line (xx, yy - 700)-(xx, yy + 700)
Line (xx + 1100, yy - 700)-(xx + 1100, yy + 700)
Line (xx, yy + 700)-(xx + 1100, yy + 700)
Line (xx, yy - 700)-(xx + 1100, yy - 700)
CurrentX = xx - 400
CurrentY = yy - 20
Print nn
CurrentX = xx + 250
CurrentY = yy - 180
Print "TempoA"
CurrentX = xx - 540
CurrentY = yy - 510
Print e0
CurrentX = xx - 540
CurrentY = yy - 100
Print e1
CurrentX = xx - 540
CurrentY = yy + 300
Print e2
CurrentX = xx + 1200
CurrentY = yy - 120
Print s
CurrentX = xx + 50
CurrentY = yy - 510
Print "D"
CurrentX = xx + 30
CurrentY = yy - 100
Print "R"
CurrentX = xx - 50
CurrentY = yy + 300
Print "H"
CurrentX = xx + 980
CurrentY = yy - 100
Print "T"
CurrentX = xx + 0
CurrentY = yy - 900
Print t
Select Case cadbin
Case 0
'fleabD
Line (xx + 200, yy - 330)-(xx + 200, yy - 480)
Line (xx + 200, yy - 330)-(xx + 140, yy - 390)
Line (xx + 200, yy - 330)-(xx + 250, yy - 390)
'fleabR
Line (xx + 200, yy + 80)-(xx + 200, yy - 70)
Line (xx + 200, yy + 70)-(xx + 140, yy + 10)
Line (xx + 200, yy + 70)-(xx + 250, yy + 10)
'H
Line (xx + 50, yy + 290)-(xx + 150, yy + 290)
Case 1
'flearrD
Line (xx + 200, yy - 330)-(xx + 200, yy - 470)
Line (xx + 200, yy - 470)-(xx + 140, yy - 410)
Line (xx + 200, yy - 470)-(xx + 250, yy - 410)
'fleabR
Line (xx + 200, yy + 80)-(xx + 200, yy - 70)
Line (xx + 200, yy + 70)-(xx + 140, yy + 10)
Line (xx + 200, yy + 70)-(xx + 250, yy + 10)
'H
Line (xx + 50, yy + 290)-(xx + 150, yy + 290)
Case 2
'fleabD
Line (xx + 200, yy - 330)-(xx + 200, yy - 480)
Line (xx + 200, yy - 330)-(xx + 140, yy - 390)
Line (xx + 200, yy - 330)-(xx + 250, yy - 390)
'flearrR
Line (xx + 200, yy - 70)-(xx + 200, yy + 80)
Line (xx + 200, yy - 70)-(xx + 140, yy + 10)
Line (xx + 200, yy - 70)-(xx + 250, yy - 20)
'H
Line (xx + 50, yy + 290)-(xx + 150, yy + 290)
Case 3
'flearrD
Line (xx + 200, yy - 330)-(xx + 200, yy - 470)
Line (xx + 200, yy - 470)-(xx + 140, yy - 410)
Line (xx + 200, yy - 470)-(xx + 250, yy - 410)
'flearrR
Line (xx + 200, yy + 70)-(xx + 200, yy + 80)
Line (xx + 200, yy + 70)-(xx + 140, yy + 10)
Line (xx + 200, yy + 70)-(xx + 250, yy - 20)
'H
Line (xx + 50, yy + 290)-(xx + 150, yy + 290)
Case 4
'fleabD
Line (xx + 200, yy - 330)-(xx + 200, yy - 480)
Line (xx + 200, yy - 330)-(xx + 140, yy - 390)
Line (xx + 200, yy - 330)-(xx + 250, yy - 390)
'fleabR
Line (xx + 200, yy + 80)-(xx + 200, yy - 70)
Line (xx + 200, yy + 70)-(xx + 140, yy + 10)
Line (xx + 200, yy + 70)-(xx + 250, yy + 10)
Case 5
'flearrD
Line (xx + 200, yy - 330)-(xx + 200, yy - 470)
Line (xx + 200, yy - 470)-(xx + 140, yy - 410)
Line (xx + 200, yy - 470)-(xx + 250, yy - 410)
'fleabR
Line (xx + 200, yy + 80)-(xx + 200, yy - 70)
Line (xx + 200, yy + 70)-(xx + 140, yy + 10)
Line (xx + 200, yy + 70)-(xx + 250, yy + 10)
Case 6
'fleabD
Line (xx + 200, yy - 330)-(xx + 200, yy - 480)
Line (xx + 200, yy - 330)-(xx + 140, yy - 390)
Line (xx + 200, yy - 330)-(xx + 250, yy - 390)

'flearrR
Line (xx + 200, yy - 70)-(xx + 200, yy + 80)
Line (xx + 200, yy - 70)-(xx + 140, yy + 10)
Line (xx + 200, yy - 70)-(xx + 250, yy - 20)
Case 7
'flearrD
Line (xx + 200, yy - 330)-(xx + 200, yy - 470)
Line (xx + 200, yy - 470)-(xx + 140, yy - 410)
Line (xx + 200, yy - 470)-(xx + 250, yy - 410)
'flearrR
Line (xx + 200, yy - 70)-(xx + 200, yy + 80)
Line (xx + 200, yy - 70)-(xx + 140, yy + 10)
Line (xx + 200, yy - 70)-(xx + 250, yy - 20)
End Select
End Sub
```

```
Public Sub dibtempoc(xx, yy, e0, e1, s, t, cadbin, nn)
xi(numodgen) = xx
```

```
xd(numodgen) = xx + 575
yi(numodgen) = yy - 575
ys(numodgen) = yy + 575
Line (xx, yy - 600)-(xx, yy + 600)
Line (xx + 1300, yy - 600)-(xx + 1300, yy + 600)
Line (xx, yy + 600)-(xx + 1300, yy + 600)
Line (xx, yy - 600)-(xx + 1300, yy - 600)
CurrentX = xx + 500
CurrentY = yy - 20
Print nn
CurrentX = xx + 350
CurrentY = yy - 180
Print "TempoC"
CurrentX = xx - 540
CurrentY = yy - 500
Print e0
CurrentX = xx - 540
CurrentY = yy + 300
Print e1
CurrentX = xx + 1400
CurrentY = yy - 400
Print s
CurrentX = xx + 50
CurrentY = yy - 500
Print "D"
CurrentX = xx + 50
CurrentY = yy + 300
Print "R"
CurrentX = xx + 1200
CurrentY = yy - 400
Print "T"
CurrentX = xx + 70
CurrentY = yy - 800
Print t
Select Case cadbin
Case 0
'fleabD
Line (xx + 200, yy - 330)-(xx + 200, yy - 480)
Line (xx + 200, yy - 330)-(xx + 140, yy - 390)
Line (xx + 200, yy - 330)-(xx + 250, yy - 390)
'fleabR
Line (xx + 200, yy + 330)-(xx + 200, yy + 470)
Line (xx + 200, yy + 470)-(xx + 140, yy + 410)
Line (xx + 200, yy + 470)-(xx + 250, yy + 410)
Case 1
'flearrD
Line (xx + 200, yy - 330)-(xx + 200, yy - 470)
Line (xx + 200, yy - 470)-(xx + 140, yy - 410)
Line (xx + 200, yy - 470)-(xx + 250, yy - 410)
'fleabR
Line (xx + 200, yy + 330)-(xx + 200, yy + 470)
Line (xx + 200, yy + 470)-(xx + 140, yy + 410)
Line (xx + 200, yy + 470)-(xx + 250, yy + 410)
Case 2
'fleabD
Line (xx + 200, yy + 330)-(xx + 200, yy - 480)
Line (xx + 200, yy - 330)-(xx + 140, yy - 390)
Line (xx + 200, yy - 330)-(xx + 250, yy - 390)
'flearrR
Line (xx + 200, yy + 330)-(xx + 200, yy + 475)
Line (xx + 200, yy + 330)-(xx + 140, yy + 390)
Line (xx + 200, yy + 330)-(xx + 250, yy + 390)
Case 3
'flearrD
Line (xx + 200, yy - 330)-(xx + 200, yy - 470)
Line (xx + 200, yy - 470)-(xx + 140, yy - 410)
Line (xx + 200, yy - 470)-(xx + 250, yy - 410)
'flearrR
Line (xx + 200, yy + 330)-(xx + 200, yy + 475)
Line (xx + 200, yy + 330)-(xx + 140, yy + 390)
Line (xx + 200, yy + 330)-(xx + 250, yy + 390)
End Select
End Sub
Public Sub dibtempod(xx, yy, e0, e1, s, t, cadbin, nn)
xi(numodgen) = xx
xd(numodgen) = xx + 575
yi(numodgen) = yy - 575
ys(numodgen) = yy + 575
Line (xx, yy - 600)-(xx, yy + 600)
Line (xx + 1300, yy - 600)-(xx + 1300, yy + 600)
Line (xx, yy + 600)-(xx + 1300, yy + 600)
Line (xx, yy - 600)-(xx + 1300, yy - 600)
CurrentX = xx + 500
CurrentY = yy - 20
Print nn
CurrentX = xx + 350
CurrentY = yy - 180
Print "TempoD"
CurrentX = xx - 540
CurrentY = yy - 500
Print e0
CurrentX = xx - 540
CurrentY = yy + 300
Print e1
CurrentX = xx + 1400
CurrentY = yy - 400
Print s
CurrentX = xx + 50
CurrentY = yy - 500
Print "D"
CurrentX = xx + 50
CurrentY = yy + 300
Print "R"
CurrentX = xx + 1200
CurrentY = yy - 400
Print "T"
CurrentX = xx + 70
CurrentY = yy - 800
Print t
Select Case cadbin
Case 0
'fleabD
Line (xx + 200, yy - 330)-(xx + 200, yy - 480)
Line (xx + 200, yy - 330)-(xx + 140, yy - 390)
Line (xx + 200, yy - 330)-(xx + 250, yy - 390)
'fleabR
Line (xx + 200, yy + 330)-(xx + 200, yy + 470)
Line (xx + 200, yy + 470)-(xx + 140, yy + 410)
Line (xx + 200, yy + 470)-(xx + 250, yy + 410)
Case 1
'flearrD
Line (xx + 200, yy - 330)-(xx + 200, yy - 470)
Line (xx + 200, yy - 470)-(xx + 140, yy - 410)
Line (xx + 200, yy - 470)-(xx + 250, yy - 410)

'fleabR
Line (xx + 200, yy + 330)-(xx + 200, yy + 470)
Line (xx + 200, yy + 470)-(xx + 140, yy + 410)
Line (xx + 200, yy + 470)-(xx + 250, yy + 410)
Case 2
'fleabD
Line (xx + 200, yy - 330)-(xx + 200, yy - 480)
Line (xx + 200, yy - 330)-(xx + 140, yy - 390)
Line (xx + 200, yy - 330)-(xx + 250, yy - 390)
'flearrR
Line (xx + 200, yy + 330)-(xx + 200, yy + 475)
Line (xx + 200, yy + 330)-(xx + 140, yy + 390)
Line (xx + 200, yy + 330)-(xx + 250, yy + 390)
Case 3
'flearrD
Line (xx + 200, yy - 330)-(xx + 200, yy - 470)
Line (xx + 200, yy - 470)-(xx + 140, yy - 410)
Line (xx + 200, yy - 470)-(xx + 250, yy - 410)
```

```
'fleatrR
Line (xx + 200, yy + 330)-(xx + 200, yy + 475)
Line (xx + 200, yy + 330)-(xx + 140, yy + 390)
Line (xx + 200, yy + 330)-(xx + 250, yy + 390)
End Select
End Sub
```

```
Public Sub dibtempoe(xx, yy, e0, s, t, te, cadbin, nn)
xi(numodgen) = xx
xd(numodgen) = xx + 575
yi(numodgen) = yy - 575
ys(numodgen) = yy + 575
Line (xx, yy - 400)-(xx, yy + 400)
Line (xx + 1100, yy - 400)-(xx + 1100, yy + 400)
Line (xx, yy + 400)-(xx + 1100, yy + 400)
Line (xx, yy - 400)-(xx + 1100, yy - 400)
CurrentX = xx + 400
CurrentY = yy + 50
Print nn
CurrentX = xx + 250
CurrentY = yy - 100
Print "TempoE"
CurrentX = xx - 340
CurrentY = yy - 300
Print e0
CurrentX = xx + 1200
CurrentY = yy - 300
Print s
CurrentX = xx + 50
CurrentY = yy - 300
Print "R"
CurrentX = xx + 950
CurrentY = yy - 300
Print "T"
CurrentX = xx + 0
CurrentY = yy - 600
Print t
CurrentX = xx + 0
CurrentY = yy + 400
Print te
Select Case cadbin
Case 0
'fleabD
Line (xx + 200, yy - 140)-(xx + 200, yy - 290)
Line (xx + 200, yy - 140)-(xx + 140, yy - 200)
Line (xx + 200, yy - 140)-(xx + 250, yy - 200)
Case 1
'fleatrD
Line (xx + 200, yy - 140)-(xx + 200, yy - 290)
Line (xx + 200, yy - 290)-(xx + 140, yy - 230)
Line (xx + 200, yy - 290)-(xx + 250, yy - 230)
End Select
End Sub
```

```
Public Function gen_strbin(X, z) As String
'Retorna una cadena que representa el equivalente binario
'de un número entero decimal x.
'z representa el número de dígitos binarios deseado
cdd = ""
If X < 2 Then
  If X = 1 Then
  cdd = "1"
  Else
  cdd = "0"
  End If
  GoTo ajuste
End If
Do
y = X / 2
rdo = (y - Int(y)) * 2
If rdo = 0 Then
cdd = "0" & cdd
Else
cdd = "1" & cdd
End If
X = Int(y)
Loop Until y < 2
If y = 0 Then
cdd = "0" & cdd
Else
cdd = "1" & cdd
End If
ajuste:
If Len(cdd) < z Then
  For i = 1 To z - Len(cdd)
  cdd = "0" & cdd
  Next i
End If
gen_strbin = cdd
End Function
```

```
Public Function nomarchab(tit, denar) As String
'Retorna nombre de archivo a abrir dado por el usuario
'tit es un string que representa el título en la barra
'desado,si el usuario oprimió el botón cancel,
'retorna el string "--".
'denar,denota un string que especifica los filtros
'y denotaciones deseados:
'por ejemplo:
'denar="ARCHIVOS ASMI*.asmlARCHIVOS
LSTI*.lstlARCHIVOS s19I*.s19|ARCHIVOS
DESI*.deslARCHIVOS S1XI*.s1xlARCHIVOS
LSXI*.lsxlArchivos BASI*.basl"
'SUPONE QUE UN CONTROL COMMDLG DE NOMBRE
'commondialog1 está en la forma que contenga
'esta fución.
CommonDialog1.DialogTitle = tit
CommonDialog1.Filter = denar
CommonDialog1.Filename = ""
'La siguiente línea habilita aviso al usuario.
'si es que el archivo seleccionado no existe en el subdirectorio
'(carpeta) especificado.
CommonDialog1.Flags = cdlOFNFileMustExist
Rem ************
Rem Las siguientes dos líneas necesarias para detectar opresión de
tecla cancel
CommonDialog1.CancelError = True
On Error Resume Next
Rem ************
CommonDialog1.ShowOpen
Rem código detecter de opresión del botón "cancel"
If Err = cdlCancel Then
nomarchab = "--"
'Err Clear
Exit Function
End If

nomarchab = CommonDialog1.filename
End Function
```

```
Public Function nomarchgu(tit, denar, archgu)
'Retorna nombre de archivo a guardar dado por el usuario
'tit es un string que representa el título en la barra
'desado,si el usuario oprimió el botón cancel,
'retorna el string "--".
'denar,denota un string que especifica los filtros
'y denotaciones deseados:
'por ejemplo:
'denar="ARCHIVOS ASMI*.asmlARCHIVOS
LSTI*.lstlARCHIVOS s19I*.s19|ARCHIVOS
DESI*.deslARCHIVOS S1XI*.s1xlARCHIVOS
LSXI*.lsxlArchivos BASI*.basl"
'archgu es el nombre del archivo a guardar.
'SUPONE QUE UN CONTROL COMMDLG DE NOMBRE
'commondialog1 está en la forma que contenga
'esta fución.
```

```
CommonDialog1.Flags = cdlOFNOverwritePrompt
CommonDialog1.DialogTitle = tit
CommonDialog1.Filter = denar
CommonDialog1.filename = archgu
':::::::::::::
Rem Las siguientes dos lineas necesarias para detectar opresión de
tecla cancel
CommonDialog1.CancelError = True
On Error Resume Next
Rem ************
CommonDialog1.ShowSave
Rem código detector de opresión del botón "cancel"
If Err = cdlCancel Then
nomarchgu = "--"
'Err.Clear
Exit Function
End If
nomarchgu = CommonDialog1.filename
End Function
```

```
Public Sub RedibModspp(X)
escodsilpp = ""
posini = 1
Do
yss = InStr(posini, X, Chr(13) & Chr(10), 1)
If yss > 0 Then
sppresc = Mid(X, posini, yss - posini)
nngen = Val(Mid(sppresc, 1, 3)) 'obtiene de "cadefspp" el numero
general de módulo
nnmod = Val(Mid(sppresc, 4, 3))
'CASE IDENTIFICADOR
Select Case nnmod
Case 1
'and2
miy = yp(nngen) + yp1(numodgen)
mix = xp(nngen) + xp1(numodgen)
e0 = Mid(sppresc, 14, 4)
e1 = Mid(sppresc, 18, 4)
s = Mid(sppresc, 22, 4)
cdbi = Mid(sppresc, 26, 2)
convierte CX
nnc = Val(Mid(sppresc, 11, 3))
diband2 mix, miy, e0, e1, s, valcdbi, nnc
escodsilpp = escodsilpp & "          " & "AND2" & "#" & nnc & "
" & e0 & "," & e1 & "," & s & "," & cdbi & ";" & Chr(13) &
Chr(10)
Case 2
'and3
miy = yp(nngen) + yp1(numodgen) + yp2(numodres)
mix = xp(nngen) + xp1(numodgen) + xp2(numodres)
e0 = Mid(sppresc, 14, 4)
e1 = Mid(sppresc, 18, 4)
e2 = Mid(sppresc, 22, 4)
s = Mid(sppresc, 26, 4)
cdbi = Mid(sppresc, 30, 3)
convierte CX
nnc = Val(Mid(sppresc, 11, 3))
diband3 mix, miy, e0, e1, e2, s, valcdbi, nnc
escodsilpp = escodsilpp & "          " & "AND3" & "#" & nnc & "
" & e0 & "," & e1 & "," & e2 & "," & s & "," & cdbi & ";" &
Chr(13) & Chr(10)
Case 3
'and4
miy = yp(nngen) + yp1(numodgen) + yp2(numodres)
mix = xp(nngen) + xp1(numodgen) + xp2(numodres)
e0 = Mid(sppresc, 14, 4)
e1 = Mid(sppresc, 18, 4)
e2 = Mid(sppresc, 22, 4)
e3 = Mid(sppresc, 26, 4)
s = Mid(sppresc, 30, 4)
cdbi = Mid(sppresc, 34, 4)
convierte CX
nnc = Val(Mid(sppresc, 11, 3))
diband4 mix, miy, e0, e1, e2, e3, s, valcdbi, nnc
escodsilpp = escodsilpp & "          " & "AND4" & "#" & nnc & "
" & e0 & "," & e1 & "," & e2 & "," & e3 & "," & s & "," & cdbi
& ";" & Chr(13) & Chr(10)
Case 4
'or2
miy = yp(nngen) + yp1(numodgen) + yp2(numodres)
mix = xp(nngen) + xp1(numodgen) + xp2(numodres)
e0 = Mid(sppresc, 13, 4)
e1 = Mid(sppresc, 17, 4)
s = Mid(sppresc, 21, 4)
cdbi = Mid(sppresc, 25, 2)
convierte CX
nnc = Val(Mid(sppresc, 10, 3))
dibor2 mix, miy, e0, e1, s, valcdbi, nnc
escodsilpp = escodsilpp & "          " & "OR2" & "#" & nnc & " "
& e0 & "," & e1 & "," & s & "," & cdbi & ";" & Chr(13) &
Chr(10)
Case 5
'or3
miy = yp(nngen) + yp1(numodgen) + yp2(numodres)
mix = xp(nngen) + xp1(numodgen) + xp2(numodres)
e0 = Mid(sppresc, 13, 4)
e1 = Mid(sppresc, 17, 4)
e2 = Mid(sppresc, 21, 4)
s = Mid(sppresc, 25, 4)
cdbi = Mid(sppresc, 29, 3)
convierte CX

nnc = Val(Mid(sppresc, 10, 3))
dibor3 mix, miy, e0, e1, e2, s, valcdbi, nnc
escodsilpp = escodsilpp & "          " & "OR3" & "#" & nnc & " "
& e0 & "," & e1 & "," & e2 & "," & s & "," & cdbi & ";" &
Chr(13) & Chr(10)
Case 6
'or4
miy = yp(nngen) + yp1(numodgen) + yp2(numodres)
mix = xp(nngen) + xp1(numodgen) + xp2(numodres)
e0 = Mid(sppresc, 13, 4)
e1 = Mid(sppresc, 17, 4)
e2 = Mid(sppresc, 21, 4)
e3 = Mid(sppresc, 25, 4)
s = Mid(sppresc, 29, 4)
cdbi = Mid(sppresc, 33, 4)
convierte CX
nnc = Val(Mid(sppresc, 10, 3))
dibor4 mix, miy, e0, e1, e2, e3, s, valcdbi, nnc
escodsilpp = escodsilpp & "          " & "OR4" & "#" & nnc & " "
& e0 & "," & e1 & "," & e2 & "," & e3 & "," & s & "," & cdbi &
";" & Chr(13) & Chr(10)
Case 7
'nand2
miy = yp(nngen) + yp1(numodgen) + yp2(numodres)
mix = xp(nngen) + xp1(numodgen) + xp2(numodres)
e0 = Mid(sppresc, 15, 4)
e1 = Mid(sppresc, 19, 4)
s = Mid(sppresc, 23, 4)
cdbi = Mid(sppresc, 27, 2)
convierte CX
nnc = Val(Mid(sppresc, 12, 3))
dibnand2 mix, miy, e0, e1, s, valcdbi, nnc
escodsilpp = escodsilpp & "          " & "NAND2" & "#" & nnc &
" " & e0 & "," & e1 & "," & s & "," & cdbi & ";" & Chr(13) &
Chr(10)
Case 8
'nand3
miy = yp(nngen) + yp1(numodgen) + yp2(numodres)
mix = xp(nngen) + xp1(numodgen) + xp2(numodres)
e0 = Mid(sppresc, 15, 4)
e1 = Mid(sppresc, 19, 4)
e2 = Mid(sppresc, 23, 4)
```

```
s = Mid(sppresc, 27, 4)
cdbi = Mid(sppresc, 31, 3)
convierte CX
nnc = Val(Mid(sppresc, 12, 3))
dibnand3 mix, miy, e0, e1, e2, s, valcdbi, nnc
escodsilpp = escodsilpp & "          " & "NAND3" & "#" & nnc &
" " & e0 & "," & e1 & "," & e2 & "," & s & "," & cdbi & ";" &
Chr(13) & Chr(10)
Case 9
'nand4
miy = yp(nngen) - yp1(numodgen) - yp2(numodres)
mix = xp(nngen) - xp1(numodgen) - xp2(numodres)
e0 = Mid(sppresc, 15, 4)
e1 = Mid(sppresc, 19, 4)
e2 = Mid(sppresc, 23, 4)
e3 = Mid(sppresc, 27, 4)
s = Mid(sppresc, 31, 4)
cdbi = Mid(sppresc, 35, 4)
convierte CX
nnc = Val(Mid(sppresc, 12, 3))
dibnand4 mix, miy, e0, e1, e2, e3, s, valcdbi, nnc
escodsilpp = escodsilpp & "          " & "NAND4" & "#" & nnc &
" " & e0 & "," & e1 & "," & e2 & "," & e3 & "," & s & "," & cdbi
& ";" & Chr(13) & Chr(10)
Case 10
'nor2
miy = yp(nngen) - yp1(numodgen) - yp2(numodres)
mix = xp(nngen) - xp1(numodgen) - xp2(numodres)
e0 = Mid(sppresc, 14, 4)
e1 = Mid(sppresc, 18, 4)
s = Mid(sppresc, 22, 4)
cdbi = Mid(sppresc, 26, 2)
convierte CX
nnc = Val(Mid(sppresc, 11, 3))
dibnor2 mix, miy, e0, e1, s, valcdbi, nnc
escodsilpp = escodsilpp & "          " & "NOR2" & "#" & nnc & "
" & e0 & "," & e1 & "," & s & "," & cdbi & ";" & Chr(13) &
Chr(10)
Case 11
'nor3
miy = yp(nngen) - yp1(numodgen) - yp2(numodres)
mix = xp(nngen) - xp1(numodgen) - xp2(numodres)
e0 = Mid(sppresc, 14, 4)
e1 = Mid(sppresc, 18, 4)
e2 = Mid(sppresc, 22, 4)
s = Mid(sppresc, 26, 4)
cdbi = Mid(sppresc, 30, 3)
convierte CX
nnc = Val(Mid(sppresc, 11, 3))
dibnor3 mix, miy, e0, e1, e2, s, valcdbi, nnc
escodsilpp = escodsilpp & "          " & "NOR3" &"#" & nnc & "
" & e0 & "," & e1 & "," & e2 & "," & s & "," & cdbi & ";" &
Chr(13) & Chr(10)
Case 12
'nor4
miy = yp(nngen) - yp1(numodgen) - yp2(numodres)
mix = xp(nngen) - xp1(numodgen) - xp2(numodres)
e0 = Mid(sppresc, 14, 4)
e1 = Mid(sppresc, 18, 4)
e2 = Mid(sppresc, 22, 4)
e3 = Mid(sppresc, 26, 4)
s = Mid(sppresc, 30, 4)
cdbi = Mid(sppresc, 34, 4)
convierte CX
nnc = Val(Mid(sppresc, 11, 3))
dibnor4 mix, miy, e0, e1, e2, e3, s, valcdbi, nnc
escodsilpp = escodsilpp & "          " & "NOR4" & "#" & nnc & "
" & e0 & "," & e1 & "," & e2 & "," & e3 & "," & s & "," & cdbi
& ";" & Chr(13) & Chr(10)
Case 13
'inversor
miy = yp(nngen) + yp1(numodgen) + yp2(numodres)
mix = xp(nngen) + xp1(numodgen) + xp2(numodres)
e0 = Mid(sppresc, 13, 4)
s = Mid(sppresc, 17, 4)
cdbi = Mid(sppresc, 21, 1)
convierte CX
nnc = Val(Mid(sppresc, 11, 3))
dibinv mix, miy, e0, s, valcdbi, nnc
escodsilpp = escodsilpp & "          " & "NOT" & "#" & nnc & " "
& e0 & "," & s & "," & cdbi & ";" & Chr(13) & Chr(10)
Case 14
'seguidor
miy = yp(nngen) - yp1(numodgen) - yp2(numodres)
mix = xp(nngen) + xp1(numodgen) + xp2(numodres)
e0 = Mid(sppresc, 13, 4)
s = Mid(sppresc, 17, 4)
cdbi = Mid(sppresc, 21, 1)
convierte CX
nnc = Val(Mid(sppresc, 11, 3))
dibseg mix, miy, e0, s, valcdbi, nnc
escodsilpp = escodsilpp & "          " & "SEG" & "#" & nnc & " "
& e0 & "," & s & "," & cdbi & ";" & Chr(13) & Chr(10)
Case 15
'FFARS
miy = yp(nngen) + yp1(numodgen) + yp2(numodres)
mix = xp(nngen) + xp1(numodgen) + xp2(numodres)
e2 = Mid(sppresc, 15, 4)
e3 = Mid(sppresc, 19, 4)
s = Mid(sppresc, 23, 4)
cdbi = Mid(sppresc, 27, 2)
convierte CX
nnc = Val(Mid(sppresc, 12, 3))
dibFFars mix, miy, e2, e3, s, valcdbi, nnc
escodsilpp = escodsilpp & "          " & "FFARS" & "#" & nnc & "
" & e2 & "," & e3 & "," & s & "," & cdbi & ";" & Chr(13) &
Chr(10)
End Select
'FIN IDENTIFICACIÓN Y DIBUJO
posini = yss + 2
Else
Exit Sub
End If
Loop Until yss = 0
End Sub
```

```
Public Sub RedibModspt(X)
escodsilpt = ""
posini = 1
Do
yss = InStr(posini, X, Chr(13) & Chr(10), 1)
If yss > 0 Then
sptresc = Mid(X, posini, yss - posini)
nngen = Val(Mid(sptresc, 25, 3))
nnmod = Val(Mid(sptresc, 28, 3))
'CASE IDENTIFICADOR
Select Case nnmod
Case 16
'TEMPOA
miy = yp(nngen) - yp1(numodgen)
mix = xp(nngen) - xp1(numodgen)
tiem = Mid(sptresc, 1, 11)
e0 = Mid(sptresc, 38, 4)
e1 = Mid(sptresc, 42, 4)
e2 = Mid(sptresc, 46, 4)
s = Mid(sptresc, 50, 4)
cdbi = Mid(sptresc, 54, 3)
convierte CX
nnc = Val(Mid(sptresc, 35, 3))
dibtempoa mix, miy, e0, e1, e2, s, tiem, valcdbi, nnc
escodsilpt = escodsilpt & "          " & "TEMPOA" & "#" & nnc &
" " & e0 & "," & e1 & "," & e2 & "," & s & "," & tiem & "," &
cdbi & "," & Chr(13) & Chr(10)
Case 17
```

```
'TEMPO C
miy = yp(nngen) + yp1(numodgen)
mix = xp(nngen) + xp1(numodgen)
tiem = Mid(sptresc, 1, 11)
e0 = Mid(sptresc, 38, 4)
e1 = Mid(sptresc, 42, 4)
s = Mid(sptresc, 46, 4)
cdbi = Mid(sptresc, 49, 3)
convierte CX
nnc = Val(Mid(sptresc, 35, 3))
dibtempoc mix, miy, e0, e1, s, tiem, valcdbi, nnc
escodsilpt = escodsilpt & "          " & "TEMPOC" & "#" & nnc &
" " & e0 & "," & e1 & "," & s & "," & tiem & "," & cdbi & ";" &
Chr(13) & Chr(10)
Case 18
'TEMPO D
miy = yp(nngen) + yp1(numodgen)
mix = xp(nngen) + xp1(numodgen)
tiem = Mid(sptresc, 1, 11)
e0 = Mid(sptresc, 38, 4)
e1 = Mid(sptresc, 42, 4)
s = Mid(sptresc, 46, 4)
cdbi = Mid(sptresc, 50, 2)
convierte CX
nnc = Val(Mid(sptresc, 35, 3))
dibtempod mix, miy, e0, e1, s, tiem, valcdbi, nnc
escodsilpt = escodsilpt & "          " & "TEMPOD" & "#" & nnc &
" " & e0 & "," & e1 & "," & s & "," & tiem & "," & cdbi & ";" &
Chr(13) & Chr(10)
Case 19
'TEMPO E
miy = yp(nngen) + yp1(numodgen)
mix = xp(nngen) + xp1(numodgen)
tiem = Mid(sptresc, 1, 11)
tiem1 = Mid(sptresc, 13, 11)
e0 = Mid(sptresc, 38, 4)
s = Mid(sptresc, 42, 4)
cdbi = Mid(sptresc, 46, 1)
convierte CX
nnc = Val(Mid(sptresc, 35, 3))
dibtempoe mix, miy, e0, s, tiem, tiem1, valcdbi, nnc
escodsilpt = escodsilpt & "          " & "TEMPOE" & "#" & nnc &
" " & e0 & "," & s & "," & tiem1 & "," & tiem & "," & cdbi & ";"
& Chr(13) & Chr(10)
Case 20
'Contador de Eventos
miy = yp(nngen) + yp1(numodgen)
mix = xp(nngen) + xp1(numodgen)
e0 = Mid(sptresc, 16, 4)
e1 = Mid(sptresc, 20, 4)
e2 = Mid(sptresc, 24, 4)
s = Mid(sptresc, 28, 4)
cdbi = Mid(sptresc, 32, 3)
convierte CX
nnc = Val(Mid(sptresc, 13, 3))
dibcontev mix, miy, e0, e1, e2, s, valcdbi, nnc
End Select
'FIN IDENTIFICACIÓN Y DIBUJO
posini = yss + 2
Else
Exit Sub
End If
Loop Until yss = 0
End Sub
```

```
Public Sub Form_Click()
If selmod > 0 And selmod <= 15 Then
ord = cad_n5d(xp(numodgen))
absc = cad_n5d(yp(numodgen))
coord = ord & absc
cacorspp = coord & cadspp
cadecspp = cadecspp & cacorspp
End If
If selmod > 15 Then
ord = cad_n5d(xp(numodgen))
absc = cad_n5d(yp(numodgen))
coord1 = ord & absc
cacorspt = coord1 & cadspt
cadecspt = cadecspt & cacorspt
End If
selmod = 0
End Sub
```

```
Private Sub Form_MouseDown(Button As Integer, Shift As
Integer, X As Single, y As Single)
'case seleccionador de módulo
Select Case selmod
Case 1
'AND2
numodgen = numodgen + 1
numodspp = numodspp + 1
nnn = cad_n3d(numod)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
ngen = cad_n3d(numod)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
diband2 X, y, e0, e1, ss, cadbin, numod
ngen = cad_n3d(numodgen)
bin = gen_strbin(cadbin, 2)
cadecspp = cadecspp & ngen & "001AND2" & nnn & e0 & e1 &
ss & bin & Chr(13) & Chr(10)
cadspp = ngen & "001AND2" & nnn & e0 & e1 & ss & bin &
Chr(13) & Chr(10)
escodsilpp = escodsilpp & "          " & "AND2" & "#" & nnn & "
" & vauxent(0) & "," & vauxent(1) & "," & vauxsal(0) & "," &
bin & ";" & Chr(13) & Chr(10)
xp(numodgen) = X
yp(numodgen) = y
Case 2
'AND3
numodgen = numodgen + 1
numodspp = numodspp + 1
nnn = cad_n3d(numod)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
ngen = cad_n3d(numod)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
```

```
Else
e0 = vauxent(0)
End If
If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
If Len(vauxent(2)) = 3 Then
e2 = vauxent(2) & " "
Else
e2 = vauxent(2)
End If
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
diband3 X, y, e0, e1, e2, ss, cadbin, numod
ngen = cad_n3d(numodgen)
bin = gen_strbin(cadbin, 3)
cadefspp = cadefspp & ngen & "002AND3" & nnn & e0 & e1 &
e2 & ss & bin & Chr(13) & Chr(10)
cadspp = ngen & "002AND3" & nnn & e0 & e1 & e2 & ss & bin
& Chr(13) & Chr(10)
escodsilpp = escodsilpp & "          " & "AND3" & "#" & nnn & "
" & vauxent(0) & "," & vauxent(1) & "," & vauxent(2) & "," &
vauxsal(0) & "," & bin & "," & Chr(13) & Chr(10)
xp(numodgen) = X
yp(numodgen) = y
Case 3
'AND4
numodgen = numodgen + 1
numodspp = numodspp + 1
nnn = cad_n3d(numod)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
ngen = cad_n3d(numod)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If

If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
If Len(vauxent(2)) = 3 Then
e2 = vauxent(2) & " "
Else
e2 = vauxent(2)
End If

If Len(vauxent(3)) = 3 Then
e3 = vauxent(3) & " "
Else
e3 = vauxent(3)
End If

If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If

diband4 X, y, e0, e1, e2, e3, ss, cadbin, numod
ngen = cad_n3d(numodgen)
bin = gen_strbin(cadbin, 4)

cadefspp = cadefspp & ngen & "003AND4" & nnn & e0 & e1 &
e2 & e3 & ss & bin & Chr(13) & Chr(10)
cadspp = ngen & "003AND4" & nnn & e0 & e1 & e2 & e3 & ss
& bin & Chr(13) & Chr(10)
escodsilpp = escodsilpp & "          " & "AND4" & "#" & nnn & "
" & vauxent(0) & "," & vauxent(1) & "," & vauxent(2) & "," &
vauxent(3) & "," & vauxsal(0) & "," & bin & "," & Chr(13) &
Chr(10)
xp(numodgen) = X
yp(numodgen) = y
Case 4
'OR2
numodgen = numodgen + 1
numodspp = numodspp + 1
nnn = cad_n3d(numod)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
ngen = cad_n3d(numod)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
dibor2 X, y, e0, e1, ss, cadbin, numod
ngen = cad_n3d(numodgen)
bin = gen_strbin(cadbin, 2)
cadefspp = cadefspp & ngen & "004OR2" & nnn & e0 & e1 & ss
& bin & Chr(13) & Chr(10)
cadspp = ngen & "004OR2" & nnn & e0 & e1 & ss & bin &
Chr(13) & Chr(10)
escodsilpp = escodsilpp & "          " & "OR2" & "#" & nnn & " "
& vauxent(0) & "," & vauxent(1) & "," & vauxsal(0) & "," & bin
& "," & Chr(13) & Chr(10)

xp(numodgen) = X
yp(numodgen) = y


Case 5
'OR3
numodgen = numodgen + 1
numodspp = numodspp + 1
nnn = cad_n3d(numod)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If

ngen = cad_n3d(numod)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
```

```
Exit Sub
End If

If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If

If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If

If Len(vauxent(2)) = 3 Then
e2 = vauxent(2) & " "
Else
e2 = vauxent(2)
End If
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
dibor3 X, y, e0, e1, e2, ss, cadbin, numod
ngen = cad_n3d(numodgen)
bin = gen_strbin(cadbin, 3)
cadefspp = cadefspp & ngen & "005OR3" & nnn & e0 & e1 & e2
& ss & bin & Chr(13) & Chr(10)
cadspp = ngen & "005OR3" & nnn & e0 & e1 & e2 & ss & bin &
Chr(13) & Chr(10)
escodsilpp = escodsilpp & "          " & "OR3" & "#" & nnn & " "
& vauxent(0) & "," & vauxent(1) & "," & vauxent(2) & "," &
vauxsal(0) & "," & bin & ";" & Chr(13) & Chr(10)
xp(numodgen) = X
yp(numodgen) = y
Case 6
'OR4
numodgen = numodgen + 1
numodspp = numodspp + 1
nnn = cad_n3d(numod)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
ngen = cad_n3d(numod)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
If Len(vauxent(2)) = 3 Then
e2 = vauxent(2) & " "
Else
e2 = vauxent(2)
End If
If Len(vauxent(3)) = 3 Then
e3 = vauxent(3) & " "
Else
e3 = vauxent(3)
End If
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
dibor4 X, y, e0, e1, e2, e3, ss, cadbin, numod
ngen = cad_n3d(numodgen)
bin = gen_strbin(cadbin, 4)
cadefspp = cadefspp & ngen & "006OR4" & nnn & e0 & e1 & e2
& e3 & ss & bin & Chr(13) & Chr(10)
cadspp = ngen & "006OR4" & nnn & e0 & e1 & e2 & e3 & ss &
bin & Chr(13) & Chr(10)
escodsilpp = escodsilpp & "          " & "OR4" & "#" & nnn & " "
& vauxent(0) & "," & vauxent(1) & "," & vauxent(2) & "," &
vauxent(3) & "," & vauxsal(0) & "," & bin & ";" & Chr(13) &
Chr(10)
xp(numodgen) = X
yp(numodgen) = y
Case 7
'NAND2
numodgen = numodgen + 1
numodspp = numodspp + 1
nnn = cad_n3d(numod)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
ngen = cad_n3d(numod)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
dibnand2 X, y, e0, e1, ss, cadbin, numod
ngen = cad_n3d(numodgen)
bin = gen_strbin(cadbin, 2)
cadefspp = cadefspp & ngen & "007NAND2" & nnn & e0 & e1 &
ss & bin & Chr(13) & Chr(10)
cadspp = ngen & "007NAND2" & nnn & e0 & e1 & ss & bin &
Chr(13) & Chr(10)
escodsilpp = escodsilpp & "          " & "NAND2" & "#" & nnn &
" " & vauxent(0) & "," & vauxent(1) & "," & vauxsal(0) & "," &
bin & ";" & Chr(13) & Chr(10)
xp(numodgen) = X
yp(numodgen) = y
Case 8
'NAND3
numodgen = numodgen + 1
numodspp = numodspp + 1
nnn = cad_n3d(numod)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
ngen = cad_n3d(numod)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
```

```
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
If Len(vauxent(2)) = 3 Then
e2 = vauxent(2) & " "
Else
e2 = vauxent(2)
End If
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
dibnand3 X, y, e0, e1, e2, ss, cadbin, numod
ngen = cad_n3d(numodgen)
bin = gen_strbin(cadbin, 3)
cadefspp = cadefspp & ngen & "008NAND3" & nnn & e0 & e1 &
e2 & ss & bin & Chr(13) & Chr(10)
cadspp = ngen & "008NAND3" & nnn & e0 & e1 & e2 & ss &
bin & Chr(13) & Chr(10)
escodsilpp = escodsilpp & "          " & "NAND3" & "#" & nnn &
" " & vauxent(0) & "," & vauxent(1) & "," & vauxent(2) & "," &
vauxsal(0) & "," & bin & "," & Chr(13) & Chr(10)
xp(numodgen) = X
yp(numodgen) = y
Case 9
'NAND4
numodgen = numodgen + 1
numodspp = numodspp + 1
nnn = cad_n3d(numod)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
ngen = cad_n3d(numod)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If

If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
If Len(vauxent(2)) = 3 Then
e2 = vauxent(2) & " "
Else
e2 = vauxent(2)
End If
If Len(vauxent(3)) = 3 Then
e3 = vauxent(3) & " "
Else
e3 = vauxent(3)
End If
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
dibnand4 X, y, e0, e1, e2, e3, ss, cadbin, numod
ngen = cad_n3d(numodgen)
bin = gen_strbin(cadbin, 4)
cadefspp = cadefspp & ngen & "009NAND4" & nnn & e0 & e1 &
e2 & e3 & ss & bin & Chr(13) & Chr(10)
cadspp = ngen & "009NAND4" & nnn & e0 & e1 & e2 & e3 & ss
& bin & Chr(13) & Chr(10)
escodsilpp = escodsilpp & "          " & "NAND4" & "#" & nnn &
" " & vauxent(0) & "," & vauxent(1) & "," & vauxent(2) & "," &
vauxent(3) & "," & vauxsal(0) & "," & bin & "," & Chr(13) &
Chr(10)
xp(numodgen) = X
yp(numodgen) = y
Case 10
'NOR2
numodgen = numodgen + 1
numodspp = numodspp + 1
nnn = cad_n3d(numod)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
ngen = cad_n3d(numod)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
dibnor2 X, y, e0, e1, ss, cadbin, numod
ngen = cad_n3d(numodgen)
bin = gen_strbin(cadbin, 2)
cadefspp = cadefspp & ngen & "010NOR2" & nnn & e0 & e1 &
ss & bin & Chr(13) & Chr(10)
cadspp = ngen & "010NOR2" & nnn & e0 & e1 & ss & bin &
Chr(13) & Chr(10)
escodsilpp = escodsilpp & "          " & "NOR2" & "#" & nnn & "
" & vauxent(0) & "," & vauxent(1) & "," & vauxsal(0) & "," &
bin & "," & Chr(13) & Chr(10)
xp(numodgen) = X
yp(numodgen) = y
Case 11
'NOR3
numodgen = numodgen + 1
numodspp = numodspp + 1
nnn = cad_n3d(numod)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
ngen = cad_n3d(numod)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
```

```
End If
If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
If Len(vauxent(2)) = 3 Then
e2 = vauxent(2) & " "
Else
e2 = vauxent(2)
End If
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
dibnor3 X, y, e0, e1, e2, ss, cadbin, numod
ngen = cad_n3d(numodgen)
bin = gen_strbin(cadbin, 3)
cadefspp = cadefspp & ngen & "011NOR3" & nnn & e0 & e1 &
e2 & ss & bin & Chr(13) & Chr(10)
cadspp = ngen & "011NOR3" & nnn & e0 & e1 & e2 & ss & bin
& Chr(13) & Chr(10)
escodsilpp = escodsilpp & "          " & "NOR3" & "#" & nnn & "
" & vauxent(0) & "," & vauxent(1) & "," & vauxent(2) & "," &
vauxsal(0) & "," & bin & ";" & Chr(13) & Chr(10)
xp(numodgen) = X
yp(numodgen) = y
Case 12
'NOR4
numodgen = numodgen + 1
numodspp = numodspp + 1
nnn = cad_n3d(numod)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
ngen = cad_n3d(numod)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
If Len(vauxent(2)) = 3 Then
e2 = vauxent(2) & " "
Else
e2 = vauxent(2)
End If
If Len(vauxent(3)) = 3 Then
e3 = vauxent(3) & " "
Else
e3 = vauxent(3)
End If
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
dibnor4 X, y, e0, e1, e2, e3, ss, cadbin, numod
ngen = cad_n3d(numodgen)
bin = gen_strbin(cadbin, 4)
cadefspp = cadefspp & ngen & "012NOR4" & nnn & e0 & e1 &
e2 & e3 & ss & bin & Chr(13) & Chr(10)
cadspp = ngen & "012NOR4" & nnn & e0 & e1 & e2 & e3 & ss
& bin & Chr(13) & Chr(10)
escodsilpp = escodsilpp & "          " & "NOR4" & "#" & nnn & "
" & vauxent(0) & "," & vauxent(1) & "," & vauxent(2) & "," &
vauxent(3) & "," & vauxsal(0) & "," & bin & ";" & Chr(13) &
Chr(10)
xp(numodgen) = X
yp(numodgen) = y
Case 13
'INVERSOR
numodgen = numodgen + 1
numodspp = numodspp + 1
nnn = cad_n3d(numod)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
ngen = cad_n3d(numod)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
dibinv X, y, e0, ss, cadbin, numod
ngen = cad_n3d(numodgen)
bin = gen_strbin(cadbin, 1)
cadefspp = cadefspp & ngen & "013INV" & nnn & e0 & ss & bin
& Chr(13) & Chr(10)
cadspp = ngen & "013INV" & nnn & e0 & ss & bin & Chr(13) &
Chr(10)
escodsilpp = escodsilpp & "          " & "NOT" & "#" & nnn & " "
& vauxent(0) & "," & vauxsal(0) & "," & bin & ";" & Chr(13) &
Chr(10)
xp(numodgen) = X
yp(numodgen) = y
Case 14
'SEGUIDOR
numodgen = numodgen + 1
numodspp = numodspp + 1
nnn = cad_n3d(numod)
If nnn = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
ngen = cad_n3d(numod)
If ngen = "--" Then
MsgBox "El número asignado a esta compuerta es invalido"
Exit Sub
End If
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
dibseg X, y, e0, ss, cadbin, numod
ngen = cad_n3d(numodgen)
bin = gen_strbin(cadbin, 1)
```

```
cadefspp = cadefspp & ngen & "014SEG" & nnn & e0 & ss & bin
& Chr(13) & Chr(10)
cadspp = ngen & "014SEG" & nnn & e0 & ss & bin & Chr(13) &
Chr(10)
escodsilpp = escodsilpp & "          " & "SEG" & "#" & nnn & " "
& vauxent(0) & "," & vauxsal(0) & "," & bin & ";" & Chr(13) &
Chr(10)
xp(numodgen) = X
yp(numodgen) = y
Case 15
'FFARS
numodgen = numodgen + 1
numodspp = numodspp + 1
nnn = cad_n3d(numod)
If nnn = "--" Then
MsgBox "El número asignado a este Flip-Flop es invalido"
Exit Sub
End If
ngen = cad_n3d(numod)
If ngen = "--" Then
MsgBox "El número asignado a este Flip-Flop es invalido"
Exit Sub
End I
If Len(vauxent(2)) = 3 Then
e2 = vauxent(2) & " "
Else
e2 = vauxent(2)
End If
If Len(vauxent(3)) = 3 Then
e3 = vauxent(3) & " "
Else
e3 = vauxent(3)
End If
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
dibFFars X, y, e2, e3, ss, cadbin, numod
ngen = cad_n3d(numodgen)
bin = gen_strbin(cadbit, 4)
cadefspp = cadefspp & ngen & "015FFARS" & nnn & e2 & e3 &
ss & bin & Chr(13) & Chr(10)
cadspp = ngen & "015FFARS" & nnn & e2 & e3 & ss & bin &
Chr(13) & Chr(10)
escodsilpp = escodsilpp & "          " & "FFARS" & "#" & nnn & "
" & vauxent(2) & "," & vauxent(3) & "," & vauxsal(0) & "," &
bin & ";" & Chr(13) & Chr(10)
xp(numodgen) = X
yp(numodgen) = y
Case 16
'TempoA
numodgen = numodgen + 1
numodspt = numodspt + 1
nnn = cad_n3d(numod)
If nnn = "--" Then
MsgBox "El número asignado a este temporizador es invalido"
Exit Sub
End If
ngen = cad_n3d(numod)
If ngen = "--" Then
MsgBox "El número asignado a este temporizador es invalido"
Exit Sub
End If
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
If Len(vauxent(2)) = 3 Then
e2 = vauxent(2) & " "
Else
e2 = vauxent(2)
End If
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
If Len(vauxtim(0)) = 11 Then
tt = vauxtim(0) & " "
Else
tt = vauxtim(0)
End If
dibtempoa X, y, e0, e1, e2, ss, tt, cadbin, numod
ngen = cad_n3d(numodgen)
bin = gen_strbin(cadbit, 4)
cadefspt = cadefspt & tt & "          " & ngen & "016TEMA" &
nnn & e0 & e1 & e2 & ss & bin & Chr(13) & Chr(10)
cadspt = tt & "          " & ngen & "016TEMA" & nnn & e0 & e1
& e2 & ss & bin & Chr(13) & Chr(10)
escodsilpt = escodsilpt & "          " & "TEMPOA" & "#" & nnn &
" " & vauxent(0) & "," & vauxent(1) & "," & vauxent(2) & "," &
vauxsal(0) & "," & vauxtim(0) & "," & bin & ";" & Chr(13) &
Chr(10)
xp(numodgen) = X
yp(numodgen) = y
Case 17
'TempoC
numodgen = numodgen + 1
numodspt = numodspt + 1
nnn = cad_n3d(numod)
If nnn = "--" Then
MsgBox "El número asignado a este temporizador es invalido"
Exit Sub
End If
ngen = cad_n3d(numod)
If ngen = "--" Then
MsgBox "El número asignado a este temporizador es invalido"
Exit Sub
End If
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
If Len(vauxtim(0)) = 11 Then
tt = vauxtim(0) & " "
Else
tt = vauxtim(0)
End If
dibtempoc X, y, e0, e1, ss, tt, cadbin, numod
ngen = cad_n3d(numodgen)
bin = gen_strbin(cadbit, 3)
cadefspt = cadefspt & tt & "          " & ngen & "017TEMC" &
nnn & e0 & e1 & ss & bin & Chr(13) & Chr(10)
cadspt = tt & "          " & ngen & "017TEMC" & nnn & e0 & e1
& ss & bin & Chr(13) & Chr(10)
```

```
escodsilpt = escodsilpt & "          " & "TEMPOC" & "#" & nnn &
" " & vauxent(0) & "," & vauxent(1) & "," & vauxsal(0) & "," &
vauxtim(0) & "," & bin & ";" & Chr(13) & Chr(10)
xp(numodgen) = X
yp(numodgen) = y
Case 18
'TempoD
numodgen = numodgen + 1
numodspt = numodspt + 1
nnn = cad_n3d(numod)
If nnn = "--" Then
MsgBox "El número asignado a este temporizador es invalido"
Exit Sub
End If
ngen = cad_n3d(numod)
If ngen = "--" Then
MsgBox "El número asignado a este temporizador es invalido"
Exit Sub
End If
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
If Len(vauxtim(0)) = 11 Then
tt = vauxtim(0) & " "
Else
tt = vauxtim(0)
End If
dibtempod X, y, e0, e1, ss, tt, cadbin, numod
ngen = cad_n3d(numodgen)
bin = gen_strbin(cadbin, 2)
cadefspt = cadefspt & tt & "          " & ngen & "018TEMD" &
nnn & e0 & e1 & ss & bin & Chr(13) & Chr(10)
cadspt = tt & "          " & ngen & "018TEMD" & nnn & e0 & e1
& ss & bin & Chr(13) & Chr(10)
escodsilpt = escodsilpt & "          " & "TEMPOD" & "#" & nnn &
" " & vauxent(0) & "," & vauxent(1) & "," & vauxsal(0) & "," &
vauxtim(0) & "," & bin & ";" & Chr(13) & Chr(10)
xp(numodgen) = X
yp(numodgen) = y
Case 19
'TempoE
numodgen = numodgen + 1
numodspt = numodspt + 1
nnn = cad_n3d(numod)
If nnn = "--" Then
MsgBox "El número asignado a este temporizador es invalido"
Exit Sub
End If
ngen = cad_n3d(numod)
If ngen = "--" Then
MsgBox "El número asignado a este temporizador es invalido"
Exit Sub
End If
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
If Len(vauxtim(0)) = 11 Then
tt = vauxtim(0) & " "
Else
tt = vauxtim(0)
End If
If Len(vauxtim(1)) = 11 Then
tt1 = vauxtim(1) & " "
Else
tt1 = vauxtim(1)
End If
dibtempoe X, y, e0, ss, tt1, tt, cadbin, numod
ngen = cad_n3d(numodgen)
bin = gen_strbin(cadbin, 2)
cadefspt = cadefspt & tt1 & tt & ngen & "019TEME" & nnn & e0
& ss & bin & Chr(13) & Chr(10)
cadspt = tt1 & tt & ngen & "019TEME" & nnn & e0 & ss & bin
& Chr(13) & Chr(10)
escodsilpt = escodsilpt & "          " & "TEMPOE" & "#" & nnn &
" " & vauxent(0) & "," & vauxsal(0) & "," & vauxtim(1) & "," &
vauxtim(0) & "," & bin & ";" & Chr(13) & Chr(10)
xp(numodgen) = X
yp(numodgen) = y
Case 20
'Contador de Eventos
numodgen = numodgen + 1
numodspt = numodspt + 1
nnn = cad_n3d(numod)
If nnn = "--" Then
MsgBox "El número asignado a este Contador de Eventos es
invalido"
Exit Sub
End If
ngen = cad_n3d(numod)
If ngen = "--" Then
MsgBox "El número asignado a este Contador de Eventos es
invalido"
Exit Sub
End If
If Len(vauxent(0)) = 3 Then
e0 = vauxent(0) & " "
Else
e0 = vauxent(0)
End If
If Len(vauxent(1)) = 3 Then
e1 = vauxent(1) & " "
Else
e1 = vauxent(1)
End If
If Len(vauxent(2)) = 3 Then
e2 = vauxent(2) & " "
Else
e2 = vauxent(2)
End If
If Len(vauxsal(0)) = 3 Then
ss = vauxsal(0) & " "
Else
ss = vauxsal(0)
End If
dibcontev X, y, e0, e1, e2, ss, cadbin, numod
ngen = cad_n3d(numodgen)
cadefspt = cadefspt & ngen & "020ContEv" & nnn & e0 & e1 &
e2 & ss & gen_strbin(cadbin, 3) & Chr(13) & Chr(10)
cadspt = ngen & "020ContEv" & nnn & e0 & e1 & e2 & ss &
gen_strbin(cadbin, 3) & Chr(13) & Chr(10)
xp(numodgen) = X
yp(numodgen) = y
End Select
End Sub
```

```
Private Sub command3_Click()
numodres = 0
RedibModspp cadefspp
selmod = 0
End Sub
```

```
Private Sub Command4_Click()
numodres = 0
RedibModspt cadefspt
selmod = 0
End Sub
```

```
Private Sub Command5_Click()
numsal = FreeFile
Open "a:listsill.sil" For Output As #numsal
Print #numsal, "      INPROG;"
Print #numsal, escodsilpp
Print #numsal, "      FINPP;"
Print #numsal, "      INMODI;"
Print #numsal, escodsilpt
Print #numsal, "      FINMODI;"
Close #numsal
End Sub
```

```
Private Sub Command6_Click()
linspp = numodspp + linlecspp
linspt = numodspt + linlecspt - 1
If linlecspp = linspp And linlecspt = linspt Then
MsgBox "No se ha introducido Módulo y/o El Archivo ya Existe"
Exit Sub
End If
den = "ARCHIVOS GRFI*.grfi"
archt = nomarchgu("GUARDAR", den, "prueba7")
MsgBox archt
nn = FreeFile
Open archt For Output As #nn
Print #nn, linspp
Print #nn, linspt
Print #nn, cadecspp
Print #nn, cadecspt
Close #nn
End Sub
```

```
Private Sub Command7_Click()
Dim cadefsppresc As String
Dim cadefsptresc As String
cadefspp = ""
cadefspt = ""
den = "ARCHIVOS GRFI*.grfi"
nom = nomarchab("ABRIDOR", den)
nn = FreeFile
Open nom For Input As #nn
Line Input #nn, numlinspp
Line Input #nn, numlinspt
For spp = 1 To numlinspp
numodrespt = numodrespt + 1
numodres = numodres + 1
Line Input #nn, cadefsppresc
xp2(numodres) = Mid(cadefsppresc, 1, 5)
yp2(numodres) = Mid(cadefsppresc, 6, 5)
cadefsppre = Mid(cadefsppresc, 11)
cadefsppre = cadefsppre & Chr(13) & Chr(10)
ord = cad_n5d(xp2(numodres))
absc = cad_n5d(yp2(numodres))
coord = ord & absc
cacorsppre = coord & cadefsppre
cadecsppre = cadecsppre & cacorsppre
ActModspp cadefsppre
Next
linlecspp = numlinspp
cadecspp = cadecsppre
For spt = 1 To numlinspt
numodres = numodres + 1
numodrespt = numodrespt + 1
Line Input #nn, cadefsptresc
xp3(numodrespt) = Mid(cadefsptresc, 1, 5)
yp3(numodrespt) = Mid(cadefsptresc, 6, 5)
cadefsptre = Mid(cadefsptresc, 11)
cadefsptre = cadefsptre & Chr(13) & Chr(10)
ord = xp3(numodrespt)
absc = yp3(numodrespt)
coord1 = ord & absc
cacorsptre = coord1 & cadefsptre
cadecsptre = cadecsptre & cacorsptre
ActModspt cadefsptre
Next
linlecspt = numlinspt
cadecspt = cadecsptre
Close #nn
End Sub
```

BIBLIOGRAFÍA

DESIGNIN THE USER INTERFACE
Strategies for Effective Human Computer Interaction
Ben Shneiderman
2ª Ed. 1993.
573 pag.

A SYSTEMS APROACH TO PROGRAMABLE CONTROLLERS
Fred Swainston
1ª Ed. 1992.
394 pag.
Delmar Publishers Inc.

PROGRAMABLE LOGIC CONTROLLERS
John W. Webb & Ronald A Reis
4ª Ed. 1999.
443 pag.
Prentice may

INGENIERÍA DEL SOFTWARE. Un enfoque práctico
Roger S. Pressman
3ª ed. 1993.
822 pag.
Mc Graw-Hill, Inc.

SOFTWARE ENGINEERING
Ian Sommerville
4ª ed. 1992.
649 pag.
Addisson-Wesley.

DISEÑO DIGITAL
M. Morris Mano
1ª ed. 1987.
477 pag.
Prentice-Hall Inc.

TESIS DE MAESTRIA "PROGRAMADOR LÓGICO MODULAR"
Antonio Salvá Calleja
Posgrado Facultad de Ingeniería, UNAM Febrero 1999.

SITIOS EN INTERNET:

- www.fim.utp.ac.pa/Revista/vol1/plc.html
- http://www.femz.es/cursos/Automatas/tema04/tema04.htm
- http://www.automatas.org/redes/grafcet.htm
- http://www.automatas.org/software.htm
- http://carrerinternet.org/~lcc/plc/menu1.htm
- http://fiet.delasalle.edu.mx/rruelas/misec616/sei09a.html
- http://www.plaquetodo.com/plaquetodo/articulos/automatas/plctut.htm
- http://www.telecable.es/personales/jrubi/index2.htm
- http://www.ciberteca.net/cgi-bin/visitarenlace.asp?id=491