UNIVERSIDAD NACIONAL DE INGENIERÍA

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

CONTROL DE UNA CORTADORA LINEAL PARA LA INDUSTRIA, UTILIZANDO EL SOFTWARE DE SIMULACION PROSYS

INFORME DE SUFICIENCIA

PARA OPTAR EL TÍTULO PROFESIONAL DE:

INGENIERO ELECTRONICO

PRESENTADO POR:

JORGE JUAN CHAPOÑAN CABRERA

PROMOCIÓN
2002 - I

LIMA – PERÚ
2007

"CONTROL DE UNA CORTADORA LINEAL PARA LA INDUSTRIA, UTILIZANDO EL SOFTWARE DE SIMULACION PROSYS"

Dedico este trabajo a:

Mis padres, inculcadores de perseverancia y sacrificio,

Mi Novia, por el apoyo incondicional entregado,

Mis hermanas por sus palabras de aliento.

SUMARIO

El presente trabajo presenta una alternativa de solución al Control de una Cortadora Lineal de 3 dimensiones diferentes, utilizando para ello la opción de simulación del software de programación Prosys v 2.04 de la Empresa Deltalogic. El programa ha sido elaborado utilizando el lenguaje Diagrama de Bloques de Función, utilizado por la mayoría de fabricantes de PLC's, dando con ello un fácil entendimiento al usuario.

En el Capítulo I se da a conocer la definición de un Controlador Lógico Programable (PLC), sus ventajas frente a la lógica convencional, su estructura básica y los criterios para poder seleccionar el más adecuado a nuestros requerimientos.

En el capitulo II se da a conocer los lenguajes de programación existentes sobre un PLC, su clasificación en lenguajes gráficos y textuales, los diferentes lenguajes usados por los fabricantes y la forma como se estructuran dentro de un programa.

En el capítulo III se da a conocer el modo de uso y las herramientas del software de simulación Prosys.

En el capitulo IV se desarrolla el planteamiento, la solución y la simulación del Programa para Controlar una Cortadora Lineal de 3 dimensiones diferentes.

En el capítulo V se desarrolla el Análisis Económico de la Implementación del Proyecto para Controlar una Cortadora Lineal de 3 dimensiones diferentes.

ÍNDICE

CAPÍTULO IV

CONTROL DE UNA CORTADORA DE TUBOS DE 3 DIMENSIONES DIFERENTES

CAPÍTULO V

ANALISIS ECONOMICO

PRÓLOGO

Desde sus orígenes el ser humano siempre se ha caracterizado por su capacidad creativa e innovadora capaz de lograr grandes desarrollos que lo han llevado a una aceleración marcada del proceso de cambio tecnológico con el que se cuenta hasta el día de hoy, es por esa razón que me gustaría retribuir a esa capacidad imaginativa con el desarrollo del presente trabajo.

El propósito del presente trabajo es proporcionar una alternativa de solución fácil y clara al problema de control de una cortadora lineal de tres dimensiones diferentes y proporcionales mediante el uso de un programa diseñado mediante un lenguaje de programación específico y que se ejecute sobre una plataforma de un Controlador Lógico Programable.

Se espera demostrar que el uso de programación en PLC es muy versátil para el campo industrial ofreciéndonos la oportunidad de solucionar problemas con la mayor facilidad y comodidad, el programa desarrollado busca ser una herramienta de apoyo y de fácil entendimiento al usuario.

El método de trabajo utilizado consiste en elaborar un Programa en lenguaje de Diagrama de Bloques de Función (FBD) utilizando para tal fin el software de programación en PLC Prosys ver. 2.04 de la Empresa Deltalogic y ejecutarlo mediante la herramienta de simulación del mencionado software.

El diseño para realizar el control de una cortadora lineal de diferentes dimensiones proporcionales tiene alcances en la industria donde se desea elaborar a gran escala cortes de tres diferentes dimensiones para aplicaciones específicas, cuenta con las medidas de

seguridad pertinentes en caso que se susciten problemas durante la ejecución del proceso, puede realizar hasta 32,767 cortes antes de que el proceso sea detenido; dentro de las limitaciones que se pueden mencionar se encuentra el caso de que si se desea elaborar cortes de longitud mayor a 3L es necesario aumentar las sentencias del programa según la longitud de corte que se desee realizar.

En el Capítulo I, se da a conocer la definición de un Controlador Lógico Programable, ventajas, estructura básica y criterios de selección. En el Capítulo II, se da a conocer los lenguajes de programación de los PLC y la forma como se estructuran dentro de un programa. En el Capítulo III, se da a conocer el modo de uso y herramientas del software de programación Prosys. En el Capítulo IV, se desarrolla la simulación del programa para controlar una cortadora lineal de tres dimensiones diferentes y proporcionales. En el Capítulo V, se da a conocer el estudio de análisis económico para la implementación del Proyecto en mención.

Finalmente me gustaría brindar un especial reconocimiento a las instituciones y personas que me proporcionaron las herramientas necesarias para manejar la Programación de los Controladores Lógicos Programables y de esta manera dar solución a problemas comunes de la vida cotidiana, como el que presento en el siguiente trabajo.

CAPÍTULO I
DEFINICION DE UN CONTROLADOR LOGICO PROGRAMABLE

1.1. DEFINICIÓN

Es un equipo Electrónico inteligente diseñado y basado en microprocesadores que consta de unidades o módulos encargados de cumplir funciones específicas como una Unidad Central de Procesamiento encargada de casi el 100% del control del sistema, módulos que permiten recibir información de todos los sensores y comandar todos los actuadores del sistema, siendo posible la factibilidad de agregarle módulos inteligentes para funciones de pre-procesamiento y comunicación.

1.2. EL PLC COMO ALTERNATIVA AL AUTOMATISMO

El PLC (Controlador Lógico Programable) es utilizado para automatizar sistemas Eléctricos, Electrónicos, Neumáticos e Hidráulicos de control discreto y análogo. Las funciones que pueden asumir estos equipos en el control se debe a la diversidad de operaciones a nivel discreto y análogo con que dispone para realizar los programas lógicos sin la necesidad de contar con Equipos adicionales.

Representa un bajo costo comparado con una serie de equipos que puedan hacer hasta cierto grado las mismas funciones, tales como: relés auxiliares, temporizadores, contadores, controladores, etc. Además de su capacidad para integrarse a otros equipos a través de redes industriales de comunicaciones, lo cual toma mayor aceptación en la industria por lo que significa la comunicación con otros equipos y por el costo adicional razonable.

1.3. VENTAJAS DEL PLC RESPECTO A LA LOGICA CONVENCIONAL

Son muchas las ventajas que justifican el empleo de PLC para automatizar sistemas, desde aplicaciones básicas hasta sistemas muy complejos no se requiere mucho análisis para decidir por la lógica programada en vez de la lógica cableada, entre ellas tenemos:

- Menor Costo: Especialmente en aplicaciones complejas prescinde del uso de dispositivos electromecánicos y electrónicos, tales como, relés auxiliares, temporizadores, algunos controladores, contadores, etc; ya que estos dispositivos simplemente son programados en el PLC sin realizar ninguna inversión adicional que seria mucho mayor al costo del PLC.

- Menor espacio: Un tablero de control que gobierna un sistema automático mediante PLC es mucho más compacto que si se controlara con dispositivos convencionales, esto debido a que el PLC esta en la capacidad de realizar todas las funciones de control, reduciendo los tableros de control la diferencia de espacio se hace muy notable.

- Confiabilidad: El PLC posee una probabilidad de fallo casi nula por razones constructivas salvo errores humanos que puedan surgir en algunas partes vulnerables como son los módulos de salida. Dado que sus componentes son de estado sólido con pocas partes mecánicas móviles esto hace que la confiabilidad se eleve en gran medida.

- Versatilidad: La versatilidad de estos equipos radica en que es posible realizar grandes modificaciones en el funcionamiento de un sistema automático con solo realizar un nuevo programa y mínimos cambios de cableado lo cual se traduce en un ahorro de tiempo realmente significativo.

- Poco mantenimiento: Debido a que cuentan con muy pocos componentes electromecánicos no requieren un mantenimiento periódico sino lo necesario para mantenerlo limpio y con sus terminales ajustados a los bornes y puesta a tierra.

- Fácil de Instalar: Debido a que el cableado de los dispositivos tanto de entrada como de salida se realiza de la manera más simple, además no requiere mucho cableado, su instalación es sencilla en comparación a la lógica convencional.

- Compatibilidad con dispositivos sensores actuadores: las normas establecen que los sistemas y equipos sean diseñados bajo un modelo abierto, de tal manera que puedan conectarse con cualquier otro equipo sin importar la marca ni procedencia.

- Integración en Redes Industriales: El crecimiento acelerado de las comunicaciones ha conllevado a que estos equipos tengan la capacidad de comunicarse vía una Red LAN entre ellos y otros equipos y de este modo trabajar en sistemas jerarquizados o distribuidos.

- Detección de Fallas: Son sencillas debido a la implementación de LED's indicadores de diagnóstico tales como: Estado de la CPU, Batería, terminales de E/S, etc. Además se puede recurrir a la memoria de errores ubicada en el CPU.

- Fácil Programación: Resulta fácil debido a que solo es necesario el manejo de conceptos básicos, contando además con diversos lenguajes de programación donde fácilmente el usuario se adapta a la representación que mejor se familiariza.

- Menor consumo de Energía: Como ya se conoce todo dispositivo electromecánico o electrónico requiere de un consumo de energía para su funcionamiento, lo cual es representativo cuando se tiene gran cantidad de ellos, es por ello que el consumo del PLC provee un ahorro sustancial en el tiempo.

- Lugar de Instalación: Por las características técnicas que presenta en cuanto a los requisitos para su instalación como nivel de temperatura, humedad, ruído, variaciones de tensión, distancias permisibles, etc. Es fácil encontrar un lugar en planta donde instalarlo aun en ambientes hostiles.

1.4. ESTRUCTURA BASICA DE UN PLC

Constituido por un conjunto de tarjetas o circuitos impresos sobre los cuales se colocan componentes electrónicos integrados. Cuando hablamos de un Controlador del tipo modular las diferentes tarjetas quedan alojadas en bastidores. Todas estas tarjetas se encuentran conectadas a través de elementos de bus, que son circuitos por donde fluye la información y generalmente se encuentran en la parte posterior. La estructura básica del hardware del controlador programable esta conformada por: Fuente de Alimentación, Unidad de Procesamiento Central (CPU), módulos o Interfaces de entrada / salida (E/S), módulos de memoria, Unidad de Programación y módulos inteligentes.

- **Fuente de Alimentación:** Suministra la energía eléctrica a la CPU y demás tarjetas según la configuración del PLC. Su ubicación es el primer lugar de izquierda a derecha en el bastidor central, su función principal es la de actuar como un rectificador a niveles que garanticen el funcionamiento del hardware del controlador programable. Ver Tabla N° 1.1.

Tabla N° 1.1 Utilidad de la Fuente de Alimentación

+5V	Para alimentar a todas las tarjetas
+5.2V	Para alimentar al programador
+24V	Para los canales de lazo de corriente 20mA

Es importante antes de seleccionar la potencia de la fuente, conocer la potencia de todas las tarjetas involucradas y preveer expansiones futuras.

- **Unidad de Procesamiento Central (C.P.U.):** Se puede considerar como el cerebro del controlador, diseñado a partir de microprocesadores y memorias. Su misión es leer los estados de las señales de las entradas, ejecutar el programa de control y gobernar las salidas, el procesamiento es permanente y a gran velocidad. Una función adicional es depositar, antes de la elaboración del programa, los estados de señal de todas las entradas en una memoria imagen del proceso de entradas y durante la ejecución del programa guardar los resultados de las combinaciones en otra memoria denominada imagen del proceso de salidas. El tiempo de lectura del programa esta en función del número y tipo de instrucciones, siendo del orden de los milisegundos.

La mayoría de fabricantes dan a conocer el "Scan time" (tiempo de exploración) del procesador en unidades de ms/KB, lo cual depende del tipo de instrucciones que tiene el programa siendo diferente escanear operaciones del tipo binarias que del tipo palabra.

- **Módulos o Interfases de Entrada y Salida:** Proporcionan el vínculo entre la CPU del controlador programable y los dispositivos de campo del sistema originándose el intercambio de información ya sea con la finalidad de adquisición de datos o el mando para el control de máquinas del proceso. Los módulos de entrada transforman la señal de entrada que se transmite hacia el controlador a niveles permitidos mediante el uso de un acoplador óptico aislando eléctricamente la sección lógica y protegiéndolo contra tensiones altas peligrosas y señales parásitas. Los módulos de salida por su parte permiten que la tensión llegue a los dispositivos de salida mediante el uso de un acoplador óptico y de un relé de impulso se asegura el aislamiento de los circuitos electrónicos del controlador y se transmite las órdenes hacia los captadores de mando (actuadores). Cuentan con módulos de entrada y salida discreta y análoga.

Módulo de entrada discreta: encargado de la captación de datos del sistema del tipo discreto, se conforman de una estructura de cuatro funciones operacionales, que son: Adquisición consistente en el cableado de la máquina desde el proceso hacia el módulo de entrada; Acondicionamiento de señal estableciendo los niveles de tensión de entrada a valores pertinentes; Señalización, formado por lámparas indicadoras LED que permiten un diagnóstico rápido; Aislamiento mediante el uso de opto-acopladores.

Módulo de salida discreta: Se usa como interfase entre la CPU del controlador programable y los actuadores en la que sólo se transmite 2 estados lógicos: activado y desactivado, pueden ser de tipo transistor, triac y relé. Contemplan las siguientes funciones operacionales: Terminación de alambrado desde máquina hacia los actuadores; Acondicionamiento de señal de lógico a conexión - desconexión; Aislamiento mediante opto-acopladores.

Módulo de entrada analógica: tiene como función digitalizar las señales analógicas para que puedan ser procesados por la CPU, estas señales analógicas que varían continuamente puede ser magnitudes de temperatura, presión, tensión, corriente, etc. Constituido por un

Conversor analógico / digital (ADC) y un multiplexor, la señal se obtiene primero por medio de un transductor y después por un canal del módulo analógico en el PLC, el multiplexor hace las veces de conmutador para seleccionar un canal al cual está conectado la señal análoga que se desea procesar para luego pasarlo al ADC y ser almacenada en una memoria denominada imagen del proceso de entrada.

Módulo de salida analógica: Usado para transmitir hacia los captadores análogos señales de tensión o corriente que varían continuamente, constituido por opto-acopladores, multiplexor, conversores digital / analógico (DAC), etc. Posee uno o más DAC dependiendo de la cantidad de canales de salida que disponga.

- **Módulos de Memoria:** Dispositivos electrónicos insertables en la CPU destinados a guardar información de manera provisional o permanente. Se tienen memorias volátiles (RAM) y no volátiles (EPROM y EEPROM) según requieran o no de energía para la conservación de la información. Siendo su capacidad desde 2Kb hasta 256Kb.

La memoria RAM se utiliza para almacenar el programa del usuario durante su elaboración y prueba donde es posible modificarlo constantemente perdiéndose su contenido si se desconecta el suministro de energía proporcionado por la fuente de alimentación (volátil) para esto se usa una batería de larga duración acoplada en la CPU con una duración de 2 a 5 años lo cual depende del tipo de CPU.

La memoria EPROM es insertable del tipo no volátil, usándose para almacenar programas definitivos. Para borrar la información se someten a rayos ultravioleta entre 15 a 45 min. La memoria EEPROM tiene las mismas características que la EPROM con la diferencia de que el borrado se realiza eléctricamente.

- **Unidad de Programación:** Los aparatos de programación son el medio de comunicación entre el hombre y la máquina, constituidos por un teclado y un dispositivo de visualización donde el teclado muestra todos los símbolos necesarios para la escritura del programa, mientras que el visualizador pone a vista todas las instrucciones programadas y registradas en memoria. Existen tres tipos de programadores: Handheld, los programadores de vídeo PC y el computador, siendo el

medio más completo de programación los dos últimos permitiendo utilizar los lenguajes de programación como Lista de instrucciones y método gráfico.

Los dispositivos de programación permiten modificar o borrar los programas de manera total o parcial, leer y borrar los programas contenidos en la RAM de la CPU o en la EPROM y EEPROM, simular la ejecución de las instrucciones del programa, detectar y visualizar las fallas originadas en los dispositivos de campo, visualizar los valores de los sensores y actuadores en tiempo real, transferencia de los programas de la memoria volátil hacia periféricos como cassette o impresora.

1.5. CRITERIOS PARA SELECCIONAR UN PLC

Son todas aquellas decisiones de selección que se basan en datos técnicos de hardware y software del PLC que en algunos casos son suficientes para cubrir una gran cantidad de aplicaciones del tipo general.

- **Fuente de Alimentación:**

Los valores detallados se encuentran en la Tabla N° 1.2.

Tabla N° 1.2 Datos de la Fuente de Alimentación

Tipo de corriente	AC/DC
Nivel de Tensión	Valor nominal: Vn Margen admisible: 0.85 a 1.2 Vn
Potencia admisible	Expresado en Watt
Frecuencia de la Red	Valor nominal: 50/60 Hz Margen admisible: ±5%
Capacidad de corriente	En Amperios
Condiciones ambientales	Temperatura °C Humedad % sin condensación
Indice de Protección	IP 20

Para determinar la potencia de la fuente requerida se debe considerar los consumos de las siguientes cargas CPU, módulos de E/S, módulos inteligentes, entre otros.

- **Unidad de Procesamiento Central (CPU):**

Los valores detallados se encuentran en la Tabla N° 1.3.

Tabla N° 1.3 Datos de la Unidad de Procesamiento Central

Capacidad de Memoria	Total: Kb Interna RAM: Kb o instrucciones Módulos de memorias: EPROM / EEPROM
Tiempo de Ejecución (Scan Time)	De cada operación binaria: us De cada operación tipo palabra: us De operaciones mixtas depende del fabricante
Tiempo de vigilancia de ciclo	Perro Guardián: ms
Cantidad de E/S discretas	A elegir
Cantidad de E/S análogas	A elegir
Cantidad de memorias internas	Total: Remanentes, No remanentes
Cantidad de Temporizadores	
Cantidad de Contadores	
Cantidad de entradas de alta frecuencia	
Cantidad de contadores de alta frecuencia	
Tipos de módulos inteligentes	
Otras funciones	Registrador de datos, Secuenciador, Operaciones digitales, Operaciones aritméticas, Comparadores, Saltos, etc.
Reloj calendario	
Algoritmo de regulación PID	
Canales de comunicación	
Posibilidad de integración a red	
Condiciones ambientales	Temperatura °C para montaje vertical / horizontal Humedad %

- **Entradas Discretas:**

Los valores detallados se encuentran en la Tabla N° 1.4.

Tabla N° 1.4 Datos de las Entradas Discretas

Cantidad de Entradas discretas	8 / modulo
Tipo de corriente	AC/DC
Nivel de Tensión Nominal	Volt
Intensidad de corriente	Miliamperes
Temperatura ambiente admisible	°C

- **Salidas Discretas:**

Los valores detallados se encuentran en la Tabla N° 1.5.

Tabla N° 1.5 Datos de las Salidas Discretas

Cantidad de salidas discretas	8 / modulo
Tipo de corriente	AC/DC (tipo transistor, relé o triac)
Nivel de Tensión	Valor nominal: V (24Vdc; 110/220Vac) Margen admisible
Capacidad admisible de	Corriente: mA Potencia: W(DC) / VA(AC)
Condiciones ambientales de temperatura	°C

- **Entradas y Salidas Analógicas:**

Los valores detallados se encuentran en la Tabla N° 1.6.

Tabla N° 1.6 Datos de las Entradas y Salidas Analógicas

Cantidad de entradas / salidas analógicas	
Tipo de señal	Corriente: 0-20mA, 4-20mA Tensión: 0-2V, 0-5V, 0-10V, ±10V, etc
Resistencia de entrada	MΩ
Resistencia de carga	Ω
Resolución	8, 12, 16 bits
Tiempo de escrutinio	ms / 50 Hz; ms / 60Hz
Corriente / Tensión de entrada admisible máxima	miliampere / volt
Corriente de Cortocircuito	Miliamperes

- **Módulos Inteligentes:**

  - Módulo de Temporizadores
  - Módulo de Contadores
  - Módulo de regulación PID
  - Módulo de posicionamiento
  - Controlador de motores paso a paso
  - Módulos de comunicación, etc.

- **Lenguaje de Programación:**

  - Lista de Instrucciones
  - Diagrama de Bloques de función
  - Diagrama escalera o diagrama de contactos

- **Sistema de Configuración:**

  - Configuración Compacto
  - Configuración Modular
  - Configuración Compacto – Modular

- **Soporte Técnico:**

  - Repuestos con la totalidad de las partes y accesorios de preferencia
  - Catálogos y manuales
  - Servicio Técnico de mantenimiento y programación
  - Asesoramiento inmediato

CAPÍTULO II
LENGUAJES DE PROGRAMACION SOBRE UN CONTROLADOR LOGICO PROGRAMABLE

2.1. CONCEPTOS GENERALES DE PROGRAMACION

Antes de iniciar con el proceso de programación es conveniente tener claro algunos conceptos preliminares respecto a la organización de los programas en la memoria del procesador. Además es importante reconocer las diferentes representaciones de los lenguajes de programación, así como su denominación en marcas de reconocido prestigio.

2.2. PROGRAMA, PROGRAMACION Y LENGUAJES DE PROGRAMACION

Desde el punto de vista del procesador un programa se define como un conjunto de instrucciones o proposiciones bien definidas en forma secuencial, cada instrucción es encargada de indicarle:

- La operación que realizará a continuación
- De dónde obtendrá los datos que necesita para realizarla
- Dónde guardará los resultados de la operación

Así desde el punto de vista del usuario un programa serían las especificaciones de un conjunto de operaciones que debe llevar a cabo el computador para lograr resolver una determinada tarea. Un programa se escribe utilizando un lenguaje de programación específico que permite simplificar la creación de programas debido a su fácil descripción de las instrucciones que ha de ejecutar el procesador, en algunos casos, agrupando varias

instrucciones y dando un sólo nombre al conjunto resultando de este modo fácil la comprensión y resolución de programas. Así reuniendo estos tres conceptos se puede decir: Un programa se escribe en un lenguaje de programación y a la actividad de expresar un algoritmo en forma de programa se le denomina programación.

2.3. CLASIFICACION DE LOS PROGRAMAS

Parte del programa es escrito por el usuario para ejecutar tareas que se desean automatizar, pero existen otros programas ya escritos que permiten procesar los programas del usuario. A continuación se definen estos tipos de programas.

2.3.1 Programas del Sistema

Existen lenguajes específicos (programas) que proporcionan servicios a los programas del usuario, es decir, realizan funciones internas operativas del controlador, estos programas incluyendo los traductores de lenguaje reciben la denominación colectiva de programas del sistema o software del sistema. Un ejemplo de esto es el Sistema Operativo cuyos servicios incluyen el manejo de los dispositivos de entrada y salida del PLC, el almacenamiento de la información durante largos períodos y la organización del procesamiento de los programas del usuario o aplicación, etc. Estos tipos de programas se encuentran almacenados en memoria EPROM dentro de la CPU por tanto no se pierden o alteran en caso de pérdida de alimentación del equipo. El usuario no tiene acceso a ellos.

2.3.2 Programas de Aplicación o del Usuario

Conjunto de instrucciones o proposiciones ingresadas por el usuario, con el fin de programar tareas de automatización específica y utiliza para ello el lenguaje de programación que mejor se adapte para ello o mejor se domine. Es importante indicar que se deben seleccionar aquellos fabricantes que ofrezcan todos los lenguajes de programación existentes en pro de poder realizar un buen trabajo.

En la fig. 2.1 se muestra la explicación esquemática de la Clasificación de los Programas.

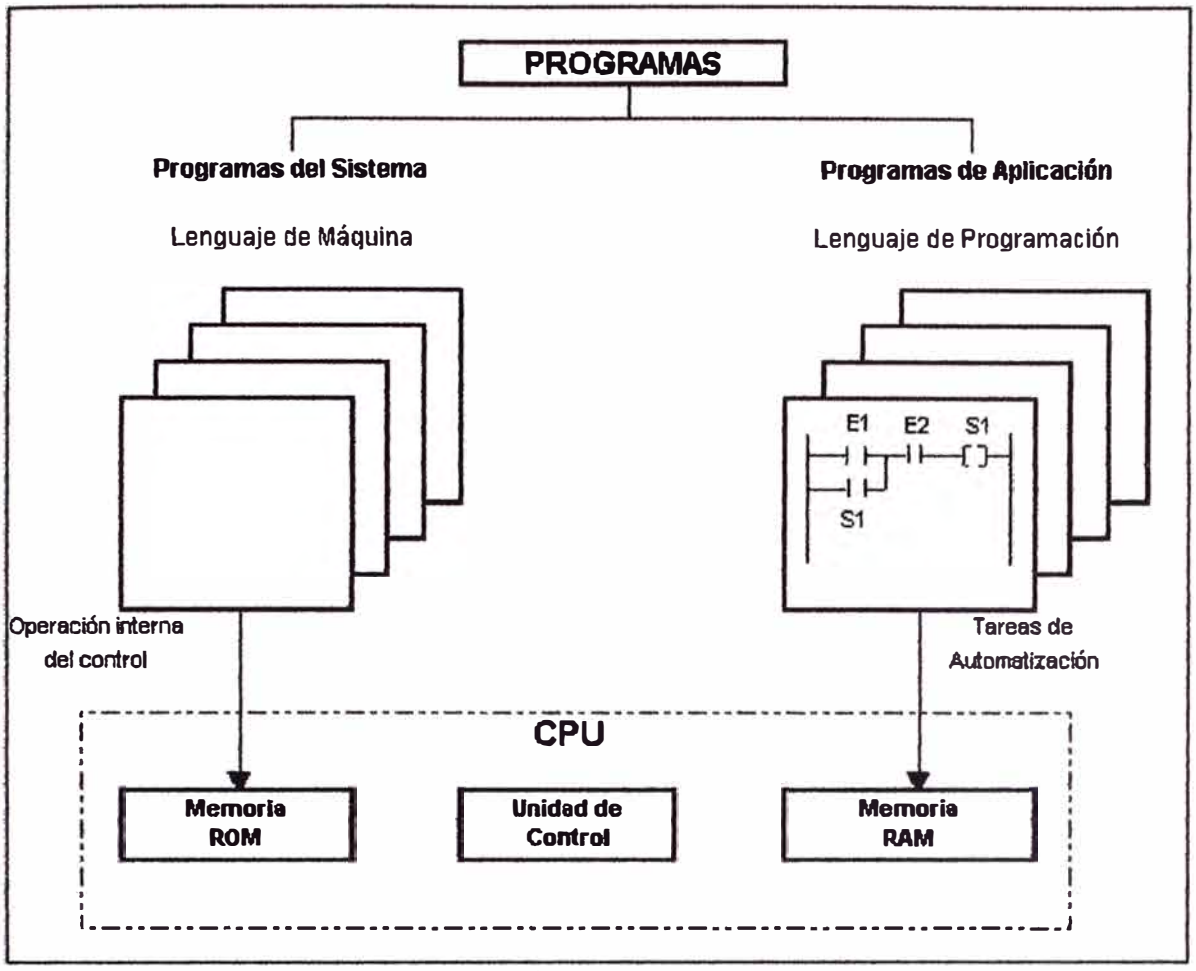

Fig. 2.1 Clasificación de los Programas

2.4. REPRESENTACION DE LOS LENGUAJES DE PROGRAMACION Y LA NORMA IEC 1131-3

En la actualidad cada fabricante diseña su propio lenguaje de programación, lo que significa, que existe una gran variedad comparable con la gran cantidad de PLC's que hay en el mercado. La forma que adopta el lenguaje de programación usado para realizar programas se denomina *Representación del lenguaje de programación.*

Hasta el momento existen tres tipos de representaciones estándar como las más difundidas a escala mundial que cada fabricante emplea para su programación y ellas son: Lista de Instrucciones, Diagrama de Bloques de Función y Diagrama de Contactos.

La gran diversidad de lenguajes de programación da lugar a que cada fabricante tenga su propia representación, originando cierta incomodidad al usuario que programa más de un

PLC. Así con el objetivo de uniformizar estas representaciones se ha establecido la norma internacional IEC 1131-3 encargada de estandarizar los lenguajes de programación dividiéndolos en dos tipos de lenguajes de programación: Lenguajes Gráficos y Lenguajes Textuales.

2.5 LENGUAJES GRAFICOS

Se denomina así a la representación basada en símbolos gráficos, de tal forma que según la disposición en que se encuentran cada uno de estos símbolos y en conformidad a su sintaxis que lo gobierna, expresa una lógica de mando y control. Así tenemos:

2.5.1 Carta de Funciones Secuenciales o Grafcet

Es una representación de análisis gráfico donde se establecen las funciones de un sistema secuencial, consistente en una secuencia de etapas y transiciones asociadas respectivamente con acciones y condiciones. Las etapas representan las acciones a realizar y las transiciones las condiciones que deben cumplirse para ir desarrollando acciones. La secuencia Etapa – Transición es un conjunto indisociable. Para mayor detalle ver fig. 2.2.

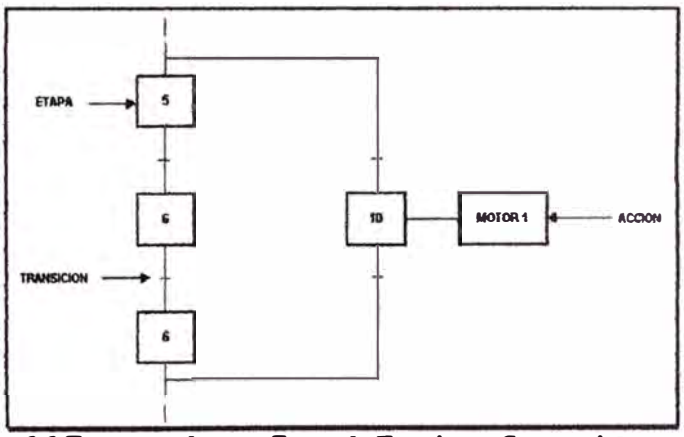

Fig. 2.2 Esquema de una Carta de Funciones Secuenciales

2.5.2 Plano de Funciones

Es una representación gráfica orientada a las puertas lógicas **AND, OR** y sus combinaciones. Las funciones individuales se representan con un símbolo, a su izquierda se ubican las entradas y a la derecha las salidas. Los símbolos usados son semejantes a los que se utilizan en los esquemas de bloques en electrónica digital. Ver fig. 2.3.

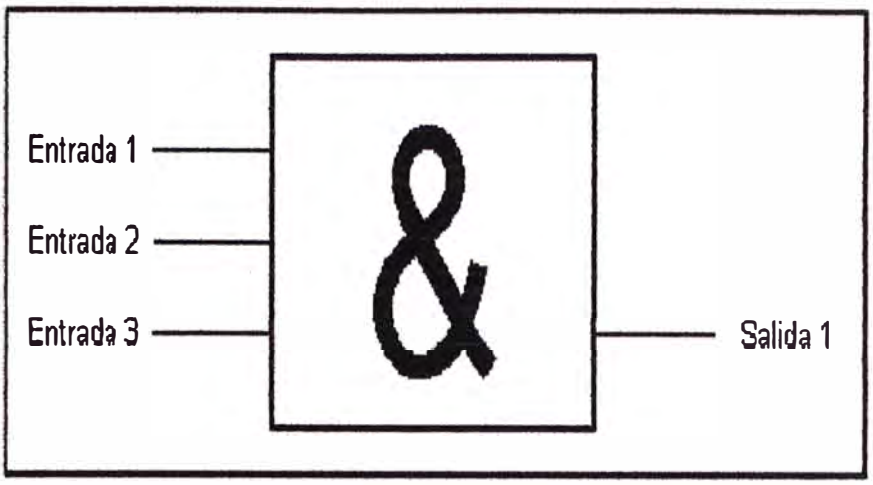

Fig. 2.3 Compuerta Lógica AND del Plano de Funciones

2.5.3 Diagrama de Contactos

Es la representación gráfica que tiene cierta analogía a los esquemas de contactos según la norma NEMA (USA).

Su estructura obedece a la semejanza que existe con los circuitos de control con lógica cableada, es decir, utiliza la misma representación de los contactos normalmente abiertos y normalmente cerrados, con la diferencia que su interpretación es totalmente diferente.

Además de los simples contactos que dispone, existen otros elementos que permiten realizar cálculos aritméticos, operaciones de comparación, implementar algoritmos de regulación, etc. Su gran difusión se debe por facilitar el trabajo a los usuarios. Ver fig. 2.4.

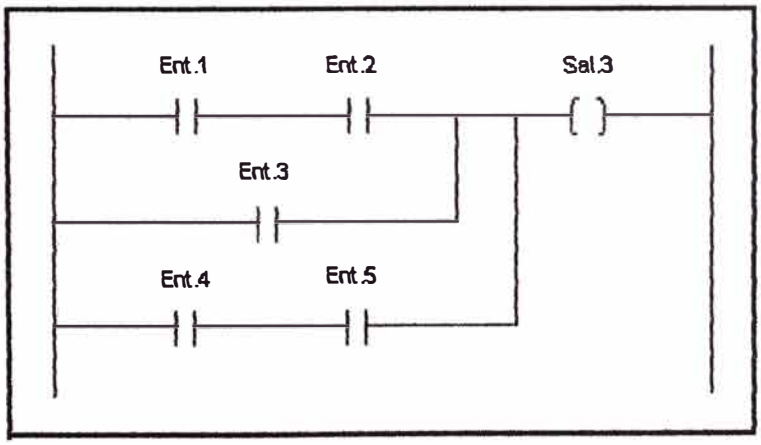

Fig. 2.4 Esquema del Diagrama de Contactos

2.6 LENGUAJES TEXTUALES

Este tipo de lenguaje se refiere básicamente al conjunto de instrucciones compuesto de letras, códigos y números de acuerdo a una sintaxis establecida. Se considera un lenguaje de menor nivel que los gráficos y por lo general se utilizan para programar pequeños PLC's cuyos programas no son muy complejos, o para programar instrucciones no programables en modo gráfico. Así existen dos lenguajes diferentes en nivel y tipo de aplicación, ellos son:

2.6.1 Lista de Instrucciones

Son instrucciones del tipo Booleanas, utilizando para su representación letras y números. Dado que se utilizan abreviaturas nemotécnicas, no se requiere gran memoria para tareas de automatización. La desventaja radica en la magnitud del trabajo que es necesario para su programación, especialmente si el programa consta de unos cientos de instrucciones. Para mayor detalle ver Tabla N° 2.1.

Tabla N° 2.1 Representación de un Programa en Lista de Instrucciones para diferentes marcas de PLC

Siemens (Simatic)	Telemecanique	General Electric
U E0.1	L I0.01	LD %I0001
U E0.2	A I0.02	AND %I0002
O E0.3	O I0.03	OR %I0003
= A3.1	= O3.01	OUT %Q0031

2.6.2 Texto Estructurado

Es un lenguaje del tipo booleano de alto nivel y estructurado, incluye las típicas sentencias de selección (IF-THEN-ELSE) y de interacción (FOR, WHILE y REPEAT), además de otras funciones específicas para aplicaciones de control. Su uso es ideal para aplicaciones en las que se requiere realizar cálculos matemáticos, comparaciones, emular protocolos, etc. Ver fig. 2.5.

```
LD      [%MW10>100]
ST      %Q0.3
LD      %M0
AND     [%MW20<%MW35]
ST      %Q0.2
LD      %I0.2
OR      [%MW30>=%MW40]
ST      %Q0.4
```

Fig. 2.5 Programa en texto estructurado para un PLC marca Telemecanique TSX-07

2.7 DENOMINACION DE LOS LENGUAJES DE PROGRAMACION DE DIFERENTES PLC's

Cada fabricante ha nombrado mediante siglas o palabras compuestas a su lenguaje de programación o software de programación que lo identifica del resto de PLC's. A continuación se presenta la Tabla N° 2.2 donde se indican estos nombres.

Tabla N° 2.2 Lenguajes de Programación de diferentes marcas de PLC

LENGUAJE	***GRAFICO***			***TEXTUAL***	
MARCA	Plano de Funciones	Plano de Contactos	Grafcet	Lista de Instrucciones	Texto Estructurado
Siemens (Simatic)	Step 5	Step 5, Step 7	Graph 5, S7-Graph	Step 5, Step 7	Step 7
Siemens (TI)	Tisoft (1)		Tisoft (2)	-	-
AEG (Modicón)	Modsoft		-	Modsoft	-
Klockner Moeller	-	Sucosoft S30	-	Sucosoft S30	
Telemecanique	-	PL7-2	PL7-2	PL7-1	PL7-0
Allen Bradley	-	APS (3)	-	-	-
General Electric	-	Logicmaster 90	-	-	Logicmaster 90

(1) RLL o Diagrama de contactos

(2) Machine Stage

(3) APS o Software de Programación Avanzada

2.8 ESTRUCTURA DEL PROGRAMA DE APLICACION

Los programas de aplicación se estructuran de acuerdo al modo como se procesan los programas (tareas), éstas pueden ser de dos tipos:

2.8.1 Programación Lineal

Se emplea para aplicaciones simples de automatización, su procesamiento es cíclico o secuencial y es suficiente programar las diferentes instrucciones en un solo bloque o sección de programación. Un procesamiento cíclico o secuencial consiste en la lectura, interpretación y ejecución de instrucción por instrucción, respetando el orden en que se han programado, salvo las instrucciones de salto. Para ejecutar las instrucciones se utilizan informaciones procedentes de la *Imagen de Proceso de Entradas* (IPE), memorias internas, memorias intermedias, así como, los datos actuales de los temporizadores y contadores. Los resultados se escriben en la *Imagen de Proceso de Salidas* (IPS). Ver fig. 2.6.

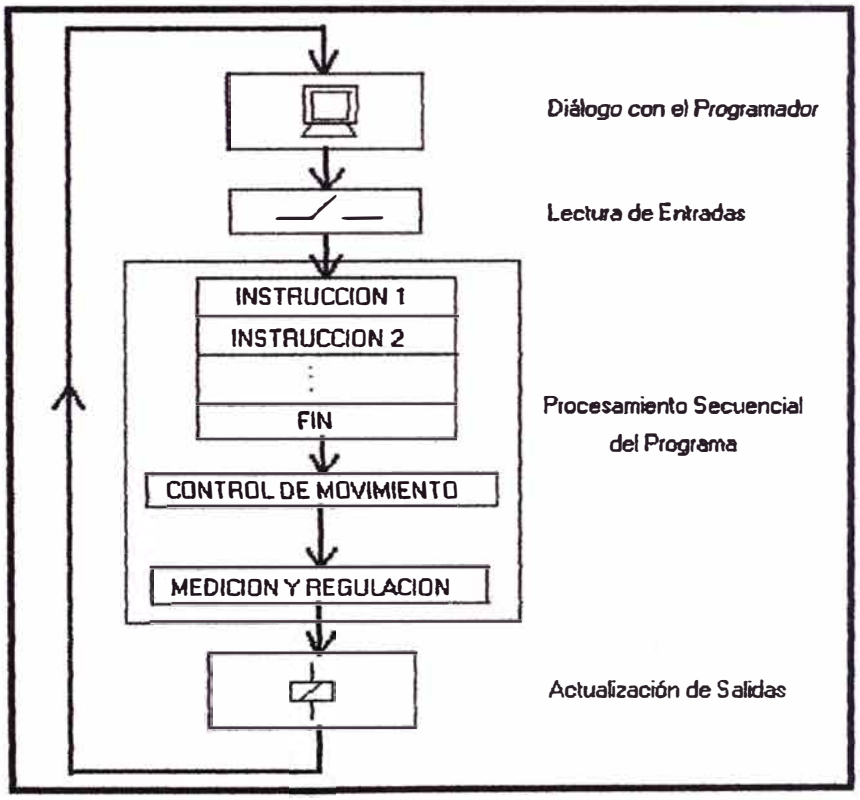

Fig. 2.6 Programación Lineal o Secuencial

Después de la ejecución del programa se ejecuta un ciclo de datos, esto significa el proceso durante el cual los datos de la IPS se transfieren a los módulos de salida, y simultáneamente, se transfieren a la IPE los datos actuales de los módulos de entrada. Con esta IPE actualizada, vuelve a realizarse la ejecución del programa, lo que significa repetir todo el proceso desde el inicio.

Los PLC's que realizan solamente este tipo de procesamiento, están diseñados con microprocesadores del tipo Intel 8086/8088 que se caracterizan por su limitada capacidad para ejecutar un solo programa a la vez. Estos tipos de PLC's son denominados también *PLCs secuenciales*, con capacidad además de ejecutar tareas de regulación, de comunicación, etc. Sin embargo, esta forma de procesamiento dificulta notablemente el trabajo cuando se tiene que procesar diferentes funciones a la vez, y en algunos casos es casi imposible estructurar los programas debido a las siguientes desventajas:

- Incremento del tiempo de barrido, que es proporcional a la complejidad del programa.
- En extensos programas es muy tedioso su diagnóstico, modificación y puesta a punto.
- Dificultad para la concepción del programa resultando muy complejo y difícil interpretarlo y actualizarlo.
- En muchos casos es indispensable el cumplimiento en tiempo real de funciones avanzadas tales como: medición analógica y regulación, servoposicionamiento, comunicación para el diálogo operador y control, funciones de monitoreo, etc.

Se considera Tiempo de barrido en los PLC como el tiempo que se emplea en leer la información de los módulos de entrada, ejecutar el programa de aplicación y finalmente actualizar el estado en los módulos de salida.

2.8.2 Programación Estructurada

Utilizado cuando se desea programar tareas de automatización muy complejas donde utilizar una programación lineal resulta demasiado laborioso, es conveniente en este caso dividir el problema en partes, de tal forma, que interpretándolo y resolviéndolo en forma parcial mediante bloques y al final unir este conjunto de programas en uno solo, resulta significativamente más fácil para el usuario.

A esta filosofía de programación se le conoce con el nombre de Programación Estructurada, que consiste en la división del Programa de Aplicación en bloques que se caracterizan por una independencia funcional, donde cada bloque del programa realiza una tarea específica claramente definida. La Programación Estructurada optimiza el tiempo de escaneo ya que no se ejecutan todos los bloques en cada ciclo de barrido, ejecutándose sólo los que están en actividad en el momento dado. Las ventajas que se obtienen de programar en forma estructurada son:

- La compresión, solución, simulación y pruebas es mucho más fácil cuando un problema muy complejo es tratado por partes.

- El diagnóstico de fallas y por ende su solución es también más fácil, dado que una vez identificado el bloque del programa donde se encuentra la falla, su corrección resulta más rápido que si se afrontara el programa global.

- Los programas parciales pueden ejecutarse independientemente por equipos de programadores, cada grupo elaborando bloques individuales; además se pueden usar reiteradamente durante el escaneo del programa, o formar parte de otro programa de aplicación.

- Se emplea mejor la capacidad de la memoria dado que pueden llamarse los bloques de programas las veces que se requiera sin que se tenga que programar repetidas veces.

- Optimización del tiempo de barrido.

Por otro lado, dependiendo del tipo de procesador que disponga el PLC la programación estructurada puede aprovecharse con menor o mayor eficiencia. Este es el caso, como se mencionó anteriormente de los PLC diseñados sobre la base de microprocesadores del tipo monotarea, donde la programación estructurada compuesta por una serie de bloques de programación, se ejecuta sobre la base del procesamiento secuencial o lineal de un bloque matriz, que viene a ser el núcleo de la estructura.

En la fig. 2.7 se puede observar un ejemplo de la programación estructurada cuya distancia medida por el número de bloques a los que "salta" se le conoce como *Profundidad de Encadenamiento o Anidado*. Con este tipo de microprocesador no se puede realizar en forma simultánea otras tareas como diálogo Hombre – Máquina, procesamiento analógico, etc.

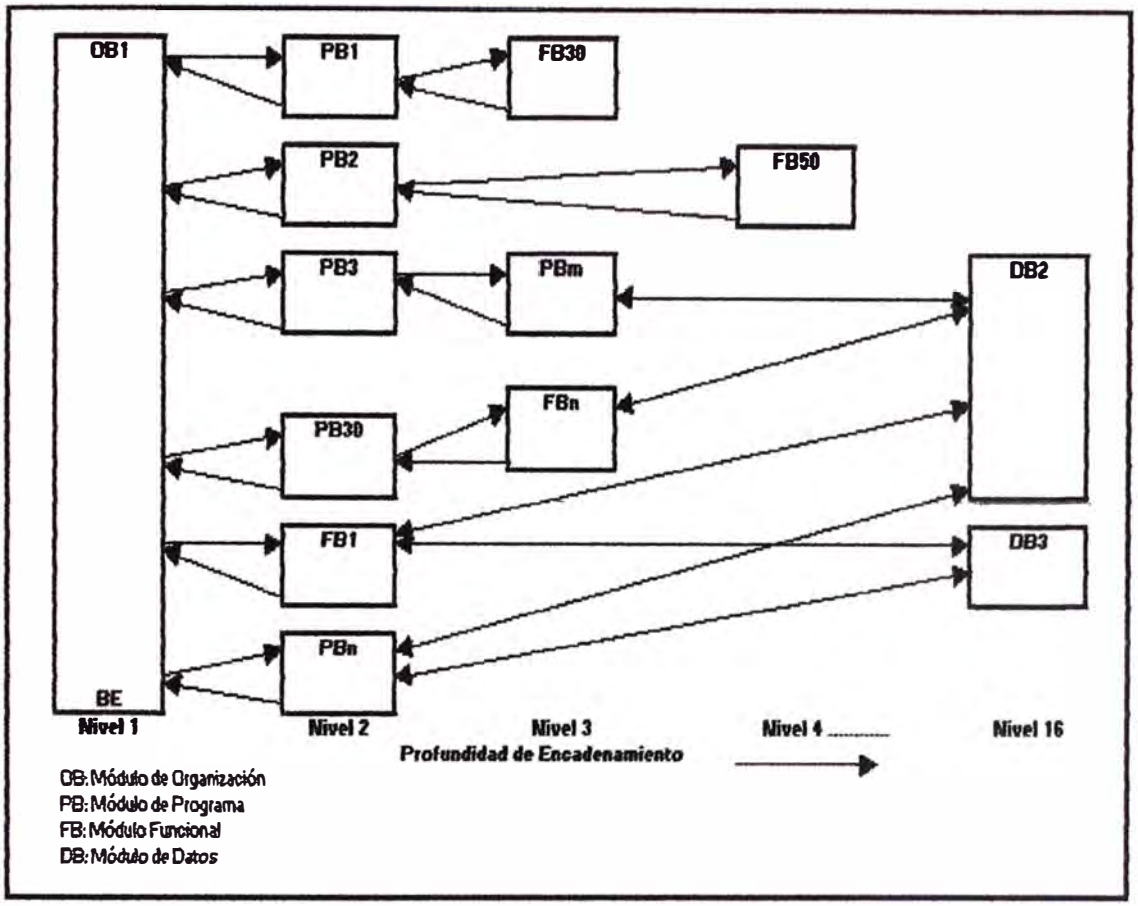

Fig. 2.7 Programación estructurada con procesador Monotarea (Diagrama de bloques según lenguaje de programación STEP5 – Simatic)

Sin embargo, en la actualidad se cuenta con procesadores de mayor velocidad de procesamiento, mayor memoria y características adicionales que le permiten a los PLC's ejecutar programas más rápidamente, estos son los llamados procesadores multifunción (286, 386, 486, etc.), con capacidad de ejecutar varios programas en forma simultánea tales como tareas de posicionamiento, medición analógica, tratamiento secuencial, diálogo, etc. Los PLC's multifunción desarrollados sobre la base de microprocesadores multitarea se caracterizan por su mayor velocidad para atender diferentes programas a la vez y en tiempo real, además por su mayor capacidad de memoria para ejecutar varios programas simultáneamente sin originar conflictos. En la estructura de la multitarea, donde el

conjunto de programas o tareas son totalmente independientes, un supervisor gobierna la ejecución de las diferentes tareas.

Así también, en estos procesadores la concepción del tratamiento secuencial es sobre la base de la división de bloques de programas, algo así como subrutinas, que básicamente es el concepto de la programación estructurada. En conclusión, la diferencia en el procesamiento de estos dos tipos de programas estructurados radica en que el primero funcionando con microprocesadores monotarea, ejecutan los diversos módulos o bloques de programación según un procesamiento secuencial, es decir, uno a continuación del otro, mientras que el procesador multifunción además del procesamiento secuencial, puede ejecutar el programa estructurado independientemente si se ejecutó el bloque anterior. Esto significa, que si en algún momento durante el proceso de barrido del programa en el sistema de control se origina una contingencia, puede ejecutarse una tarea de interrupción sin tener que esperar el barrido total del programa.

CAPÍTULO III
SOFTWARE DE SIMULACION PROSYS

3.1 BREVE DESCRIPCION

El software de simulación Prosys de ACON ha sido probado en el laboratorio de prueba de PLC de la TUV Nord y esta certificado para ser compatible con PLC Base de nivel IL (Certificado N° C-12 del 4 de Octubre de 1995) y de nivel ST (Certificado N° C-13 del 4 de Octubre de 1995).

La norma IEC 1131-3 es la norma internacional de lenguajes de programación para Controladores Lógicos Programables, siendo tomada como una norma nacional en la mayoría de países importantes como: Estados Unidos, Alemania, etc; por ejemplo, la correspondiente norma alemana es DIN EN 611313 la cual define un modelo de software y cinco lenguajes de programación: lista de instrucciones (tipo Assembler), lenguaje escalera, diagrama de bloques de funciones, texto estructurado (un alto nivel de lenguaje tipo Pascal), y carta de funciones secuenciales. Todos estos lenguajes son implementados en Prosys 1131 y cumplen con los requerimientos del estándar.

Para comenzar a trabajar dentro de la interfase del software de simulación Prosys es necesario saber por completo el correcto funcionamiento de las librerías que implican la correcta simulación de los programas que en ella se detallan.

3.2 MODO DE USO DEL SOFTWARE DE SIMULACION

Paso 1: Se procede a abrir el software de simulación Prosys y se crea un nuevo POU, según lo indica la fig. 3.1.

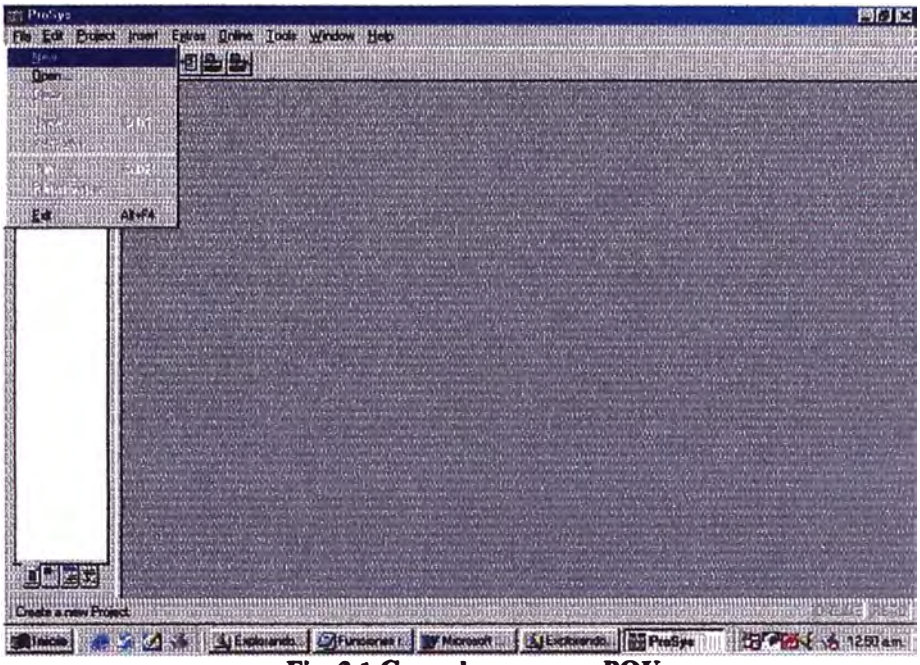

Fig. 3.1 Creando un nuevo POU

Paso 2: Se procede a adicionar objetos al proyecto POU creado anteriormente en donde se elaborarán la estructura de los programas. Ver fig. 3.2.

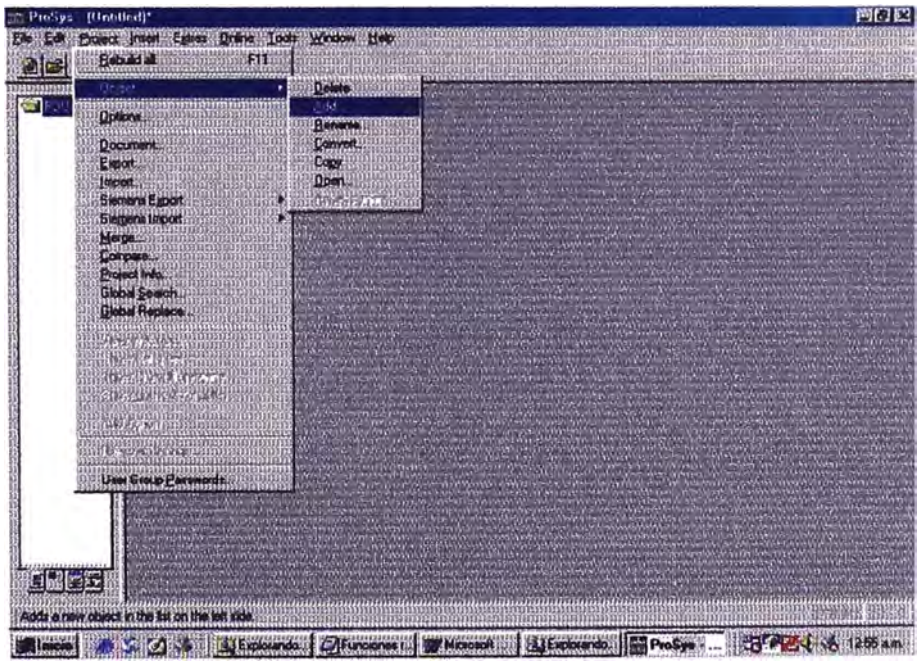

Fig. 3.2 Adicionando objetos al POU

Paso 3: Se procede a elegir el tipo de POU (Programa) y el lenguaje que va a manejar (Ladder o escalera) para interactuar con la interfaz de usuario. Ver fig. 3.3.

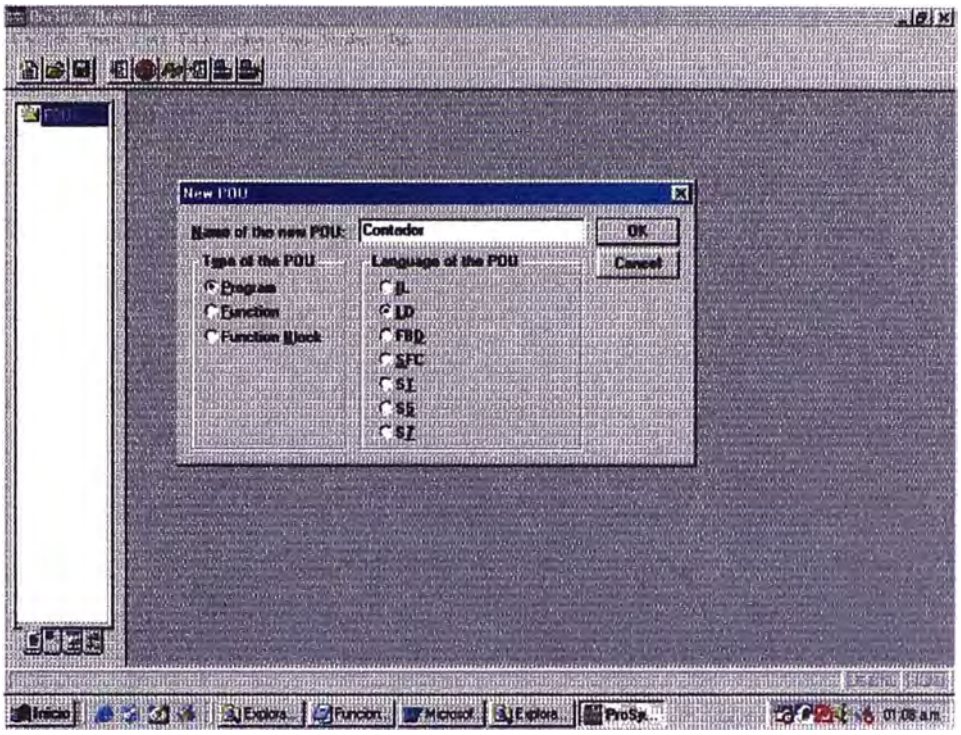

Fig. 3.3 Elección del Lenguaje de Programación

Paso 4: En el lenguaje "escalera" elegido anteriormente se procede a elaborar un programa de contactos estableciendo las direcciones y el tipo de cada variable creada. Ver fig. 3.4.

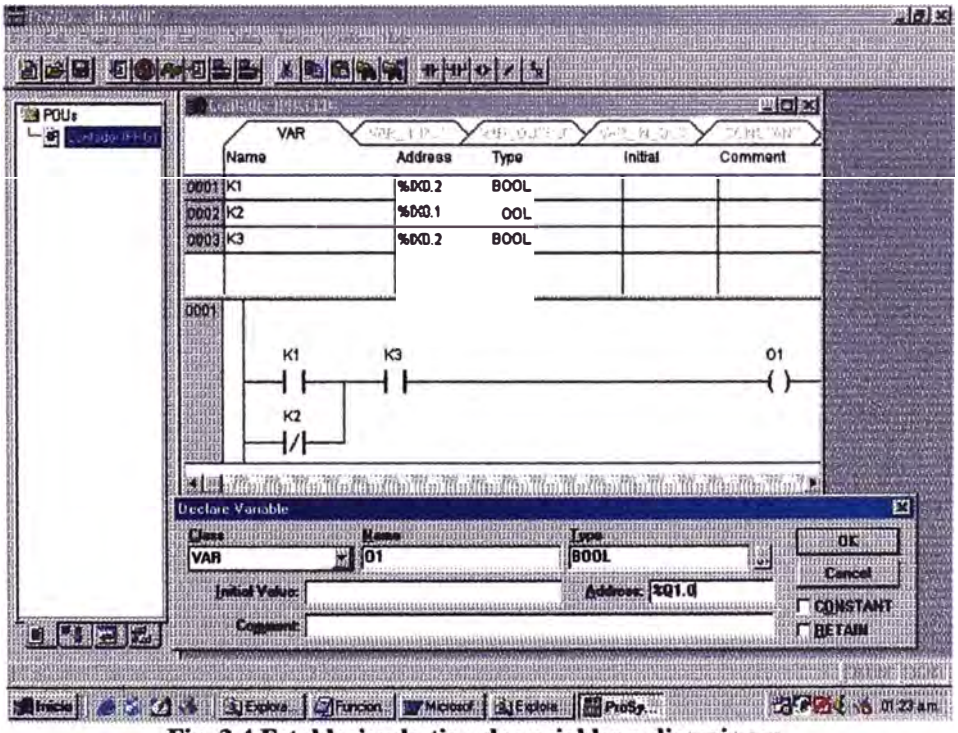

Fig. 3.4 Estableciendo tipo de variables y direcciones

Paso 5: En la pestaña "Resources" se procede a realizar la configuración del PLC indicando los bytes de entrada y salida requeridos según el ejemplo. Ver fig. 3.5.

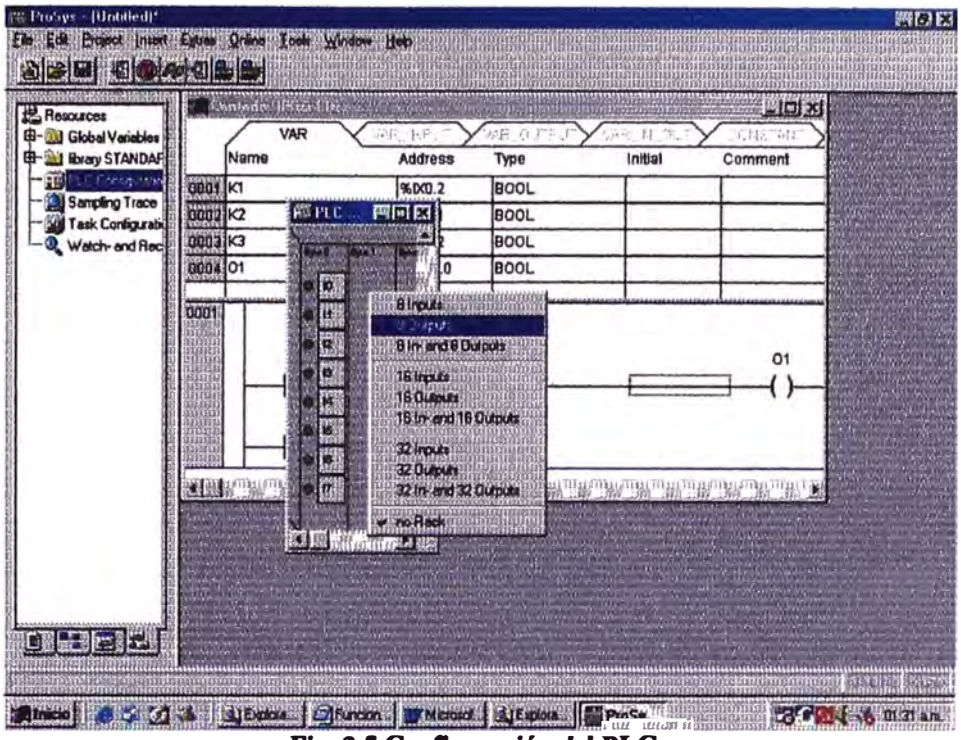

Fig. 3.5 Configuración del PLC

Paso 6: Se elige el modo de simulación para ingresar los valores de entrada por consola (en reemplazo del proceso) evitando generar errores al compilar el programa. Ver fig. 3.6.

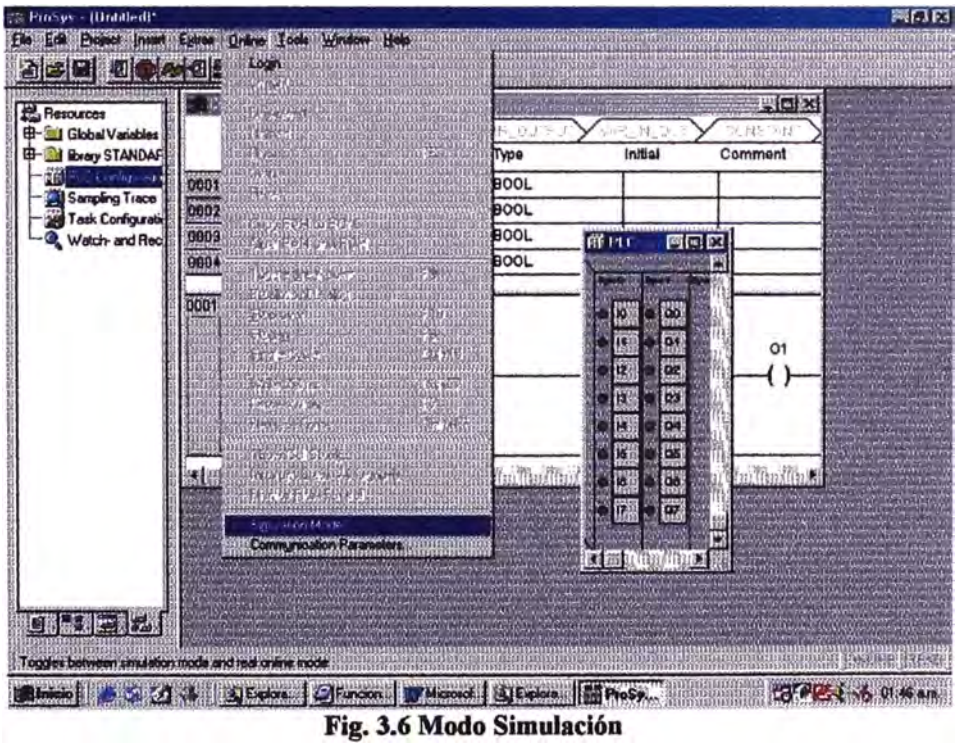

Fig. 3.6 Modo Simulación

Paso 7: Se eligen las opciones "Login", "Download" y "Run" para verificar que el programa no tenga errores y habilitar la memoria de las variables a manipular. Ver fig. 3.7.

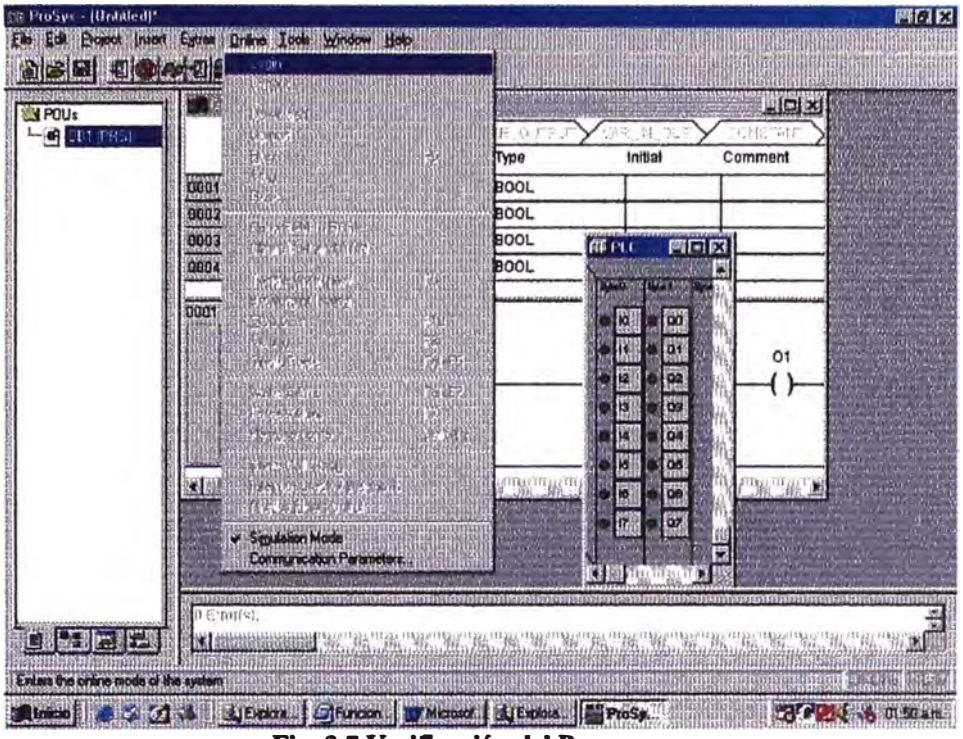

Fig. 3.7 Verificación del Programa

Paso 8: Se ejecuta el programa habilitando o deshabilitando las entradas binarias observando las respuestas esperadas de la lógica booleana. Ver fig. 3.8.

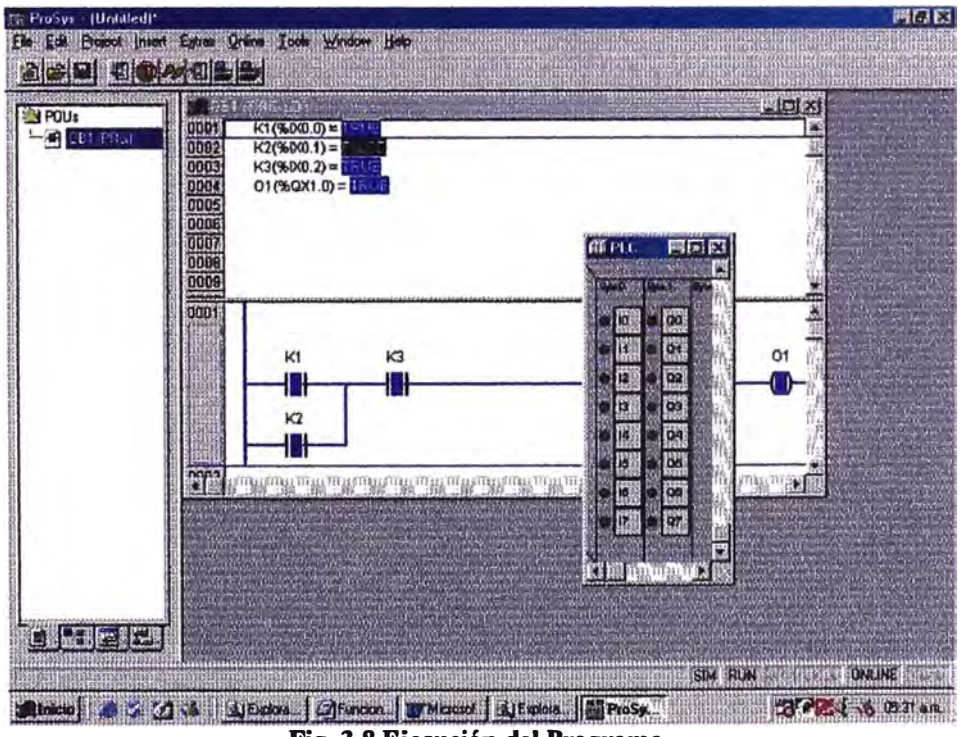

Fig. 3.8 Ejecución del Programa

3.3 HERRAMIENTAS DEL SOFTWARE DE SIMULACION

Dentro de las herramientas con las que cuenta el software de simulación podemos resaltar las siguientes:

3.3.1 Comando File

Dentro de ella tenemos las opciones básicas de todo programa de office como son: crear, abrir o cerrar un archivo; guardar los cambios y configurar impresiones. Ver fig. 3.9.

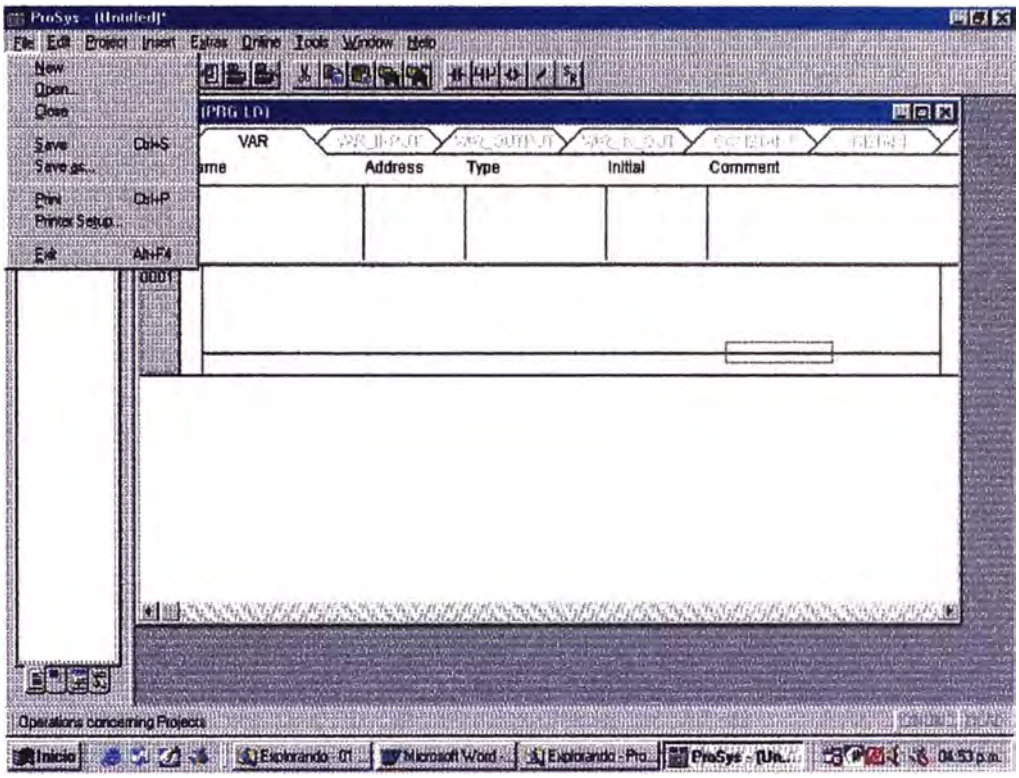

Fig. 3.9 Ejecutando el comando File

3.3.2 Comando Edit

Con las opciones Undo y Redo para deshacer o restablecer una opción elegida. Cortar, copiar, pegar y borrar para editar estructuras de un programa. Find para ubicar la dirección de las variables de forma inmediata. Para mayor detalle ver fig. 3.10.

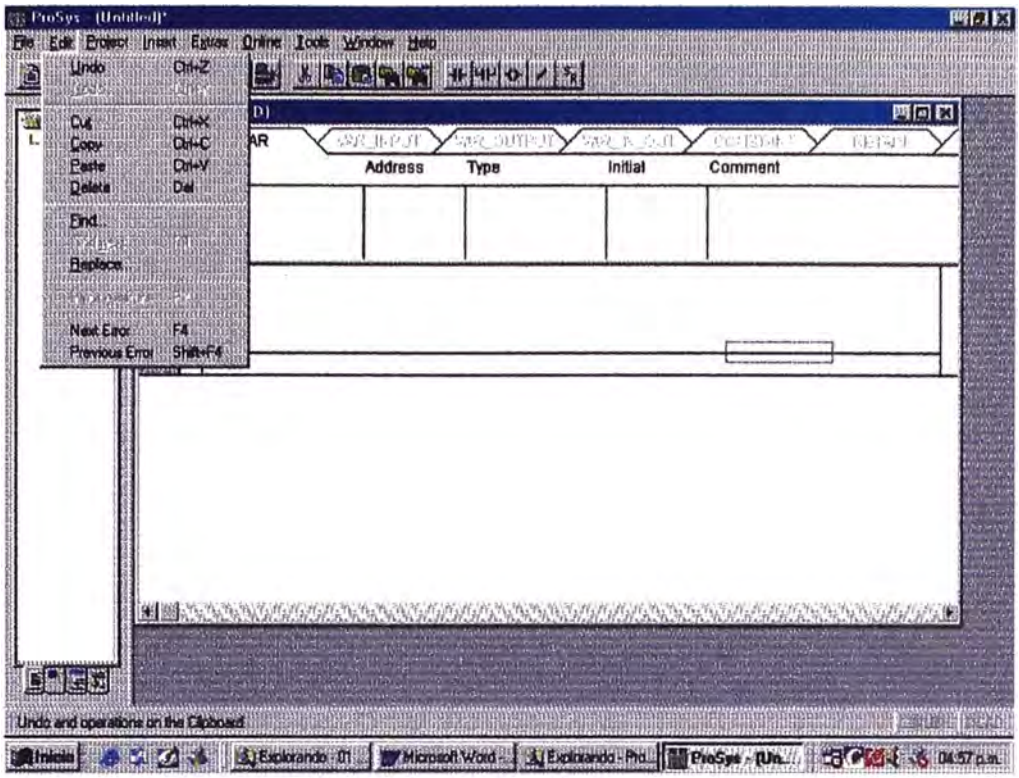

Fig. 3.10 Ejecutando el comando Edit

3.3.3 Comando Project

Contiene los siguientes sub-comandos (ver fig. 3.11):

- Rebuild, te permite revisar el estado de las constantes y variables globales, si es que no existen problemas en su direccionamiento.

- Object, te permite copiar, borrar, renombrar o adicionar nuevos objetos en el programa existente.

- Opciones, creación de copias de respaldo, información del usuario, tamaño de la fuente y color, ubicación de las librerías y los archivos de compilación, así como, opciones de seguridad.

- Export e Import, permite exportar e importar proyectos hacia o desde el ya existente.
- Merge, permite copiar proyectos enteros desde una ubicación determinada.
- Project Info, permite conocer información como título, autor, versión y descripción del proyecto.

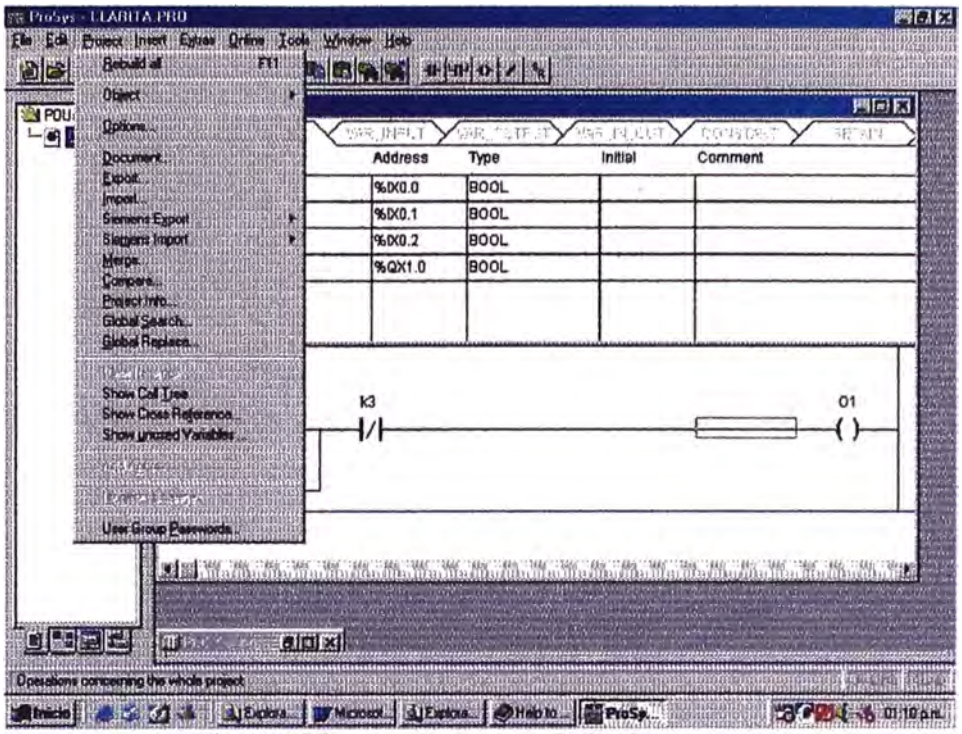

Fig. 3.11 Ejecutando el comando Project

3.3.4 Comando Insert

Contiene los siguientes sub-comandos (ver fig. 3.12):

- New declaration, permite ingresar una nueva variable para el programa indicando su tipo y dirección.

- Network Before y After, permite ingresar una nueva secuencia del programa anterior o posterior a la secuencia en la cual nos encontramos.

- Operator, coloca un operador lógico.
- Assign, coloca una variable dentro de un esquema de programa.
- Function, permite asignar funciones estándares y/o funciones definidas por el usuario en el programa.
- Function Block, permite asignar bloques de función (sub-programas estándares) o programas definidos por el usuario, en el programa.
- Comment, permite ingresar un comentario en cada secuencia del programa.

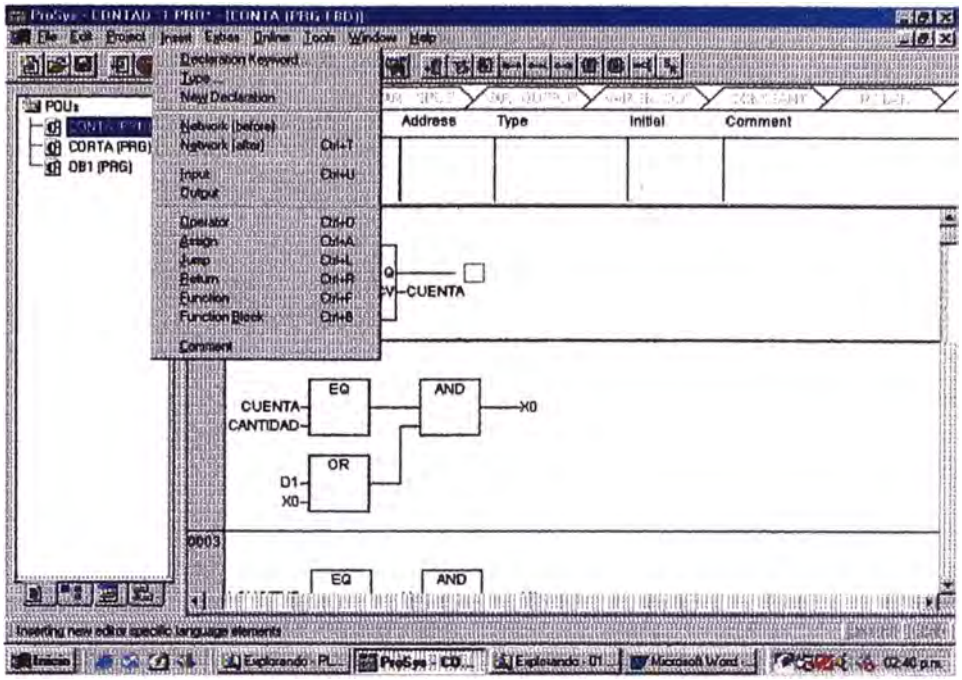

Fig. 3.12 Ejecutando el comando Insert

3.3.5 Comando Extras

Contiene los siguientes sub-comandos (ver fig. 3.13):

- Negate, permite negar las variables de un programa.
- Options, realiza configuraciones en el tamaño de los comentarios en una secuencia de programa.

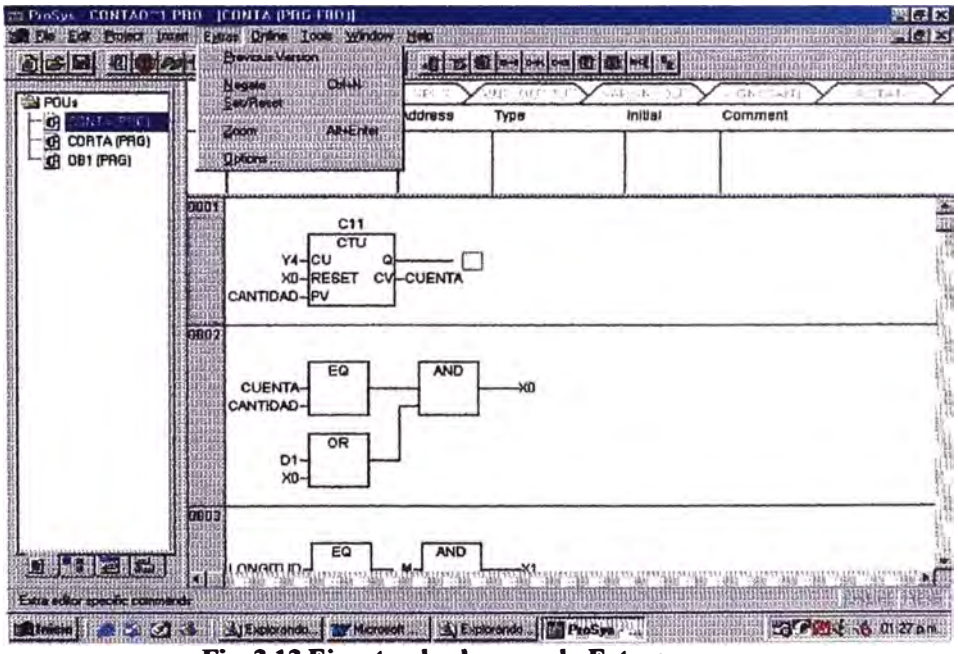

Fig. 3.13 Ejecutando el comando Extras

3.3.6 Comando Online

Contiene los siguientes sub-comandos (ver fig. 3.14):

- Simulation Mode, si la simulación es elegida, los servicios en línea no están actuando en el PLC pero si bajo la PC como sistema de programación.
- Login, este comando conecta el sistema al PLC o al software de PLC usado en el modo simulación y conmuta al modo "On line" (en línea). Si la compilación resulta en error entonces Prosys no puede conmutar al modo en línea.
- Download, descarga un proyecto compilado dentro del PLC.
- Upload, la data en el PLC puede ser salvado en un proyecto S5 o S7, el cual puede ser importado con el comando "Project Siemens Import".
- Run, activa el programa en el PLC. El estado actual de la simulación o controlador es mostrado en la ventana de Barra de estados.
- Logout, este comando permite conmutar al modo fuera de línea, la conexión al controlador es cerrada y se finaliza la simulación.
- Stop, detiene el PLC después del ciclo actual del PLC.
- Copy RAM to ROM, función soportada solo por algunos S7, copia el programa descargado en una memoria interna EPROM, si el PLC esta en STOP.
- Clear RAM and ROM, función soportada solo por algunos S7, borra el programa descargado en el PLC y en la memoria interna EPROM, si el PLC esta en STOP.
- Toggle Breakpoint, ajusta el punto de quiebre a la posición del cursor actual. Las posiciones donde el punto de quiebre puede ser ajustado depende del lenguaje del editor activo.
- Step over, un solo paso es realizado, se realiza una llamada de POUs en un solo paso.
- Step in, paso a la siguiente sentencia, aún cuando este se encuentre en un POU diferente.
- Single cycle, la ejecución es detenida después de la ejecución de un solo ciclo del controlador.
- Write values to PLC, para cambiar el valor de una variable realiza un doble click en la declaración de la variable, ingresar el nuevo valor y el cambio será inmediato.
- Force values to PLC, estos valores son escritos después de cada ciclo del PLC hasta que el comando "Release Force" sea ejecutado.

- Release Force, los valores forzados en el PLC son liberados. Todas las variables forzadas pueden ahora cambiar normalmente.
- Show Call Stack, este comando se ejecuta cuando la simulación esta detenida en un punto de quiebre. Aparece una ventana con el POU de inicio y el POU donde se encuentra la ejecución actual, para mostrar la secuencia en la que se encuentra el respectivo POU.
- Flow control, cada línea de cada red que es ejecutada durante el último ciclo del reloj es marcado.
- Communication Parameters, mediante el cual se configura el puerto por el que se descargará el programa hacia el PLC.

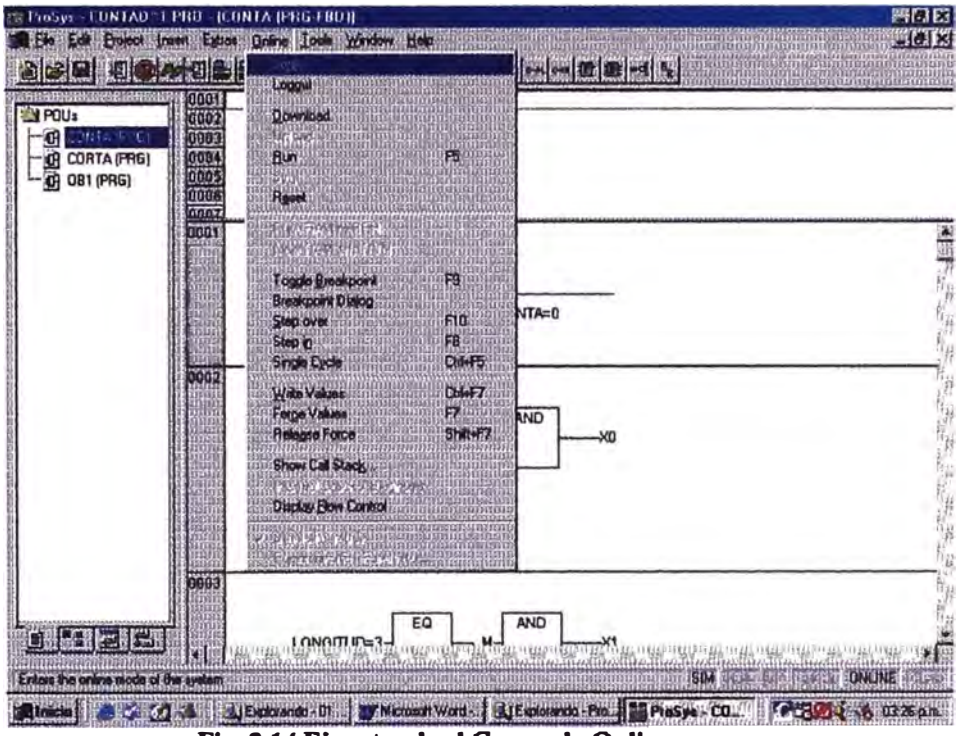

Fig. 3.14 Ejecutando el Comando Online

3.3.7 Comando Tools

Permite añadir herramientas complementarias al Software de Simulación Prosys, para mayor detalle ver fig, 3.15.

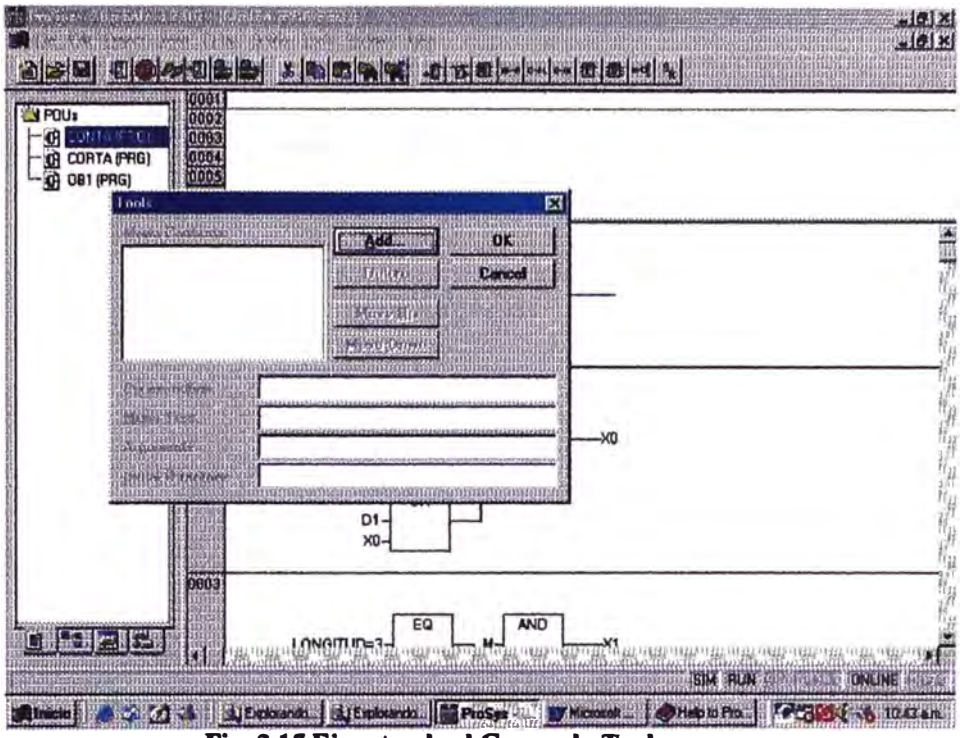

Fig. 3.15 Ejecutando el Comando Tools

CAPÍTULO IV
CONTROL DE UNA CORTADORA DE TUBOS DE 3 DIMENSIONES DIFERENTES

4.1. PLANTEAMIENTO: CONTROL DE UNA CORTADORA DE TUBOS

A continuación se dará a conocer el problema planteado motivo de la elaboración del presente informe:

"Se desea realizar el control de una cortadora de tubos para la Industria de 3 dimensiones diferentes, cuyos cortes deben estar relacionados por la siguiente proporción aritmética: L, 2L y 3L. La secuencia de estos cortes y la cantidad de cortes a realizar debe de ser elegida de forma aleatoria por el usuario, una vez obtenida la cantidad de cortes requerida se debe dar fin al proceso. Se debe incluir en el desarrollo del programa de control un pulsador de arranque, un pulsador de parada y un pulsador de parada de emergencia para la cortadora brindando de esta forma toda la seguridad posible al operador. Los cortes a realizar deben de ser rectos evitando inclinaciones defectuosas producto de la incorrecta colocación del tubo. En todo momento del desarrollo del control de la cortadora se debe tener conocimiento del cumplimiento de los estándares de seguridad requeridos".

4.2. SOLUCION : CONTROL DE UNA CORTADORA DE TUBOS

A continuación se dará a conocer la solución ofrecida para el problema planteado "Control de una cortadora de tubos de 3 dimensiones diferentes":

Se realizará un programa que sea capaz de controlar una cortadora colocada en forma vertical gobernada por un motor industrial, así como, el movimiento planar de un sistema formado por un retenedor de tubo y un subsistema formado por 1 retenedor de tubo y una plataforma que corre sobre un carril en el sentido que lo indique el programa (siendo su función la de obtener las dimensiones requeridas por el usuario) y perpendicular al movimiento de los retenedores cuya función es la de sujetar al tubo cuando se requiera para un corte recto y sin curvas, para lo cual además se utilizarán sensores de posición "Sb" que indicarán la presencia y posición correcta del tubo dentro de la canaleta.

El programa se desarrollará utilizando el lenguaje de Programación FBD (Diagrama de Bloques de Función) que forma parte del software de programación en PLC Siemens "Prosys" (versión 2.04) licenciado por la Empresa alemana Deltalogic.

El programa estará conformado por 2 sub-programas:

- Un programa denominado "Cortador" en el cual se controlará la secuencia de movimiento de cada uno de los retenedores, el carril y la cortadora, además de la marcha, parada, parada de emergencia y liberación del proceso de cortado.

- Un programa denominado "Contador" en el cual se controlará la longitud de cada corte y se llevará la cuenta de la cantidad de cortes realizados para poder detener el proceso cuando este llegue al valor preseleccionado, según requerimientos del operador encargado de la elaboración de los cortes de diferentes tamaños.

En la Fig. 4.1 del siguiente esquema se dará a conocer el método de solución ideado para controlar una cortadora de tubos de tres diferentes dimensiones, así como, la cantidad de cortes requerido inicialmente.

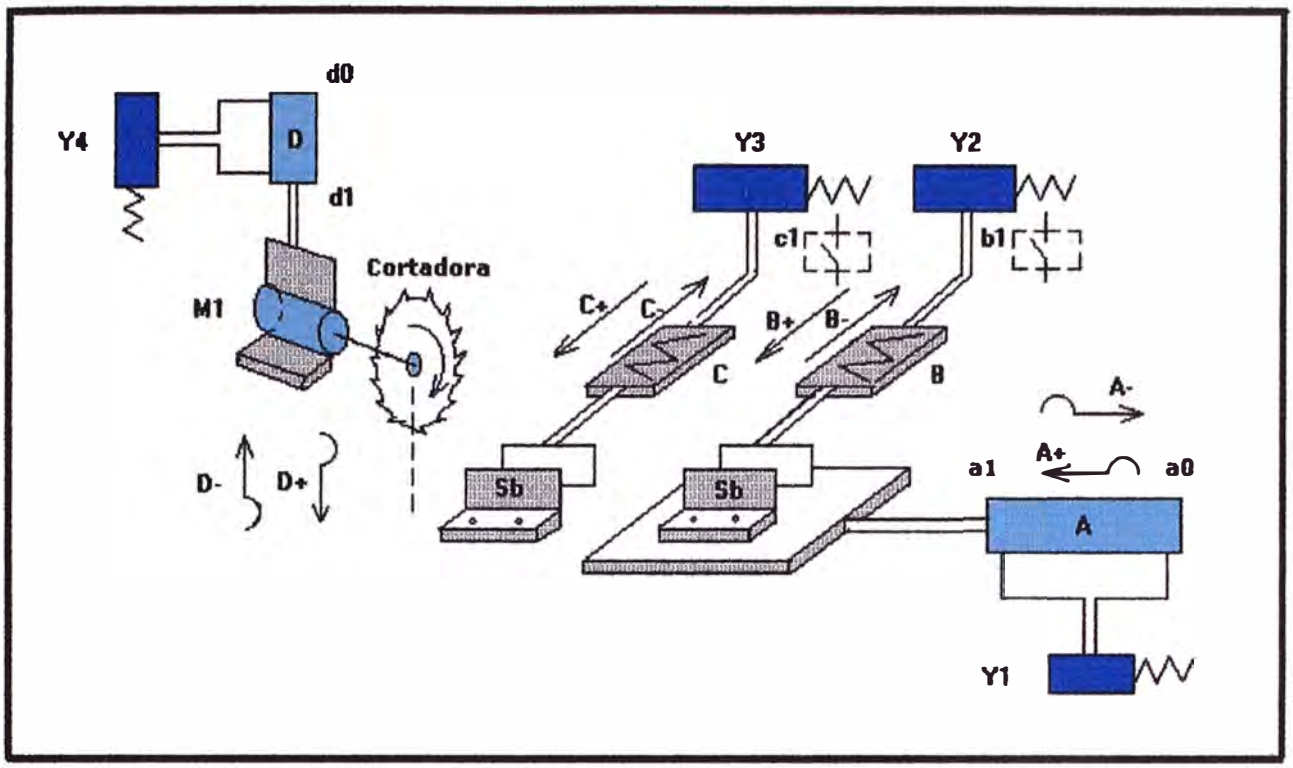

Fig. 4.1 Esquema del Proceso de Control de una Cortadora de Tubos

4.2.1 Secuencia para cortes de Tubo de Longitud L

De la Fig. 4.1 se obtiene la secuencia para obtener cortes de longitud L, ver Tabla N° 4.1.

Tabla N° 4.1 Secuencia para cortes de Tubo de longitud L

Condición Actual	Estado Siguiente
a1=1; a0=0; b1=0; c1=0; d0=1; d1=0	C+
a1=1; a0=0; b1=0; c1=1; d0=1; d1=0	A-
a1=0; a0=1; b1=0; c1=1; d0=1; d1=0	B+
a1=0: a0=1; b1=1; c1=1; d0=1; d1=0	C-
a1=0; a0=1; b1=1; c1=0; d0=1; d1=0	A+
a1=1; a0=0; b1=1; c1=0; d0=1; d1=0	C+
a1=1; a0=0; b1=1; c1=1; d0=1; d1=0	D+
a1=1; a0=0; b1=1; c1=1; d0=0; d1=1	D-
a1=1; a0=0; b1=1; c1=1; d0=1; d1=0	B-

4.2.2 Secuencia para cortes de Tubo de Longitud 2L

De la Fig. 4.1 se obtiene la secuencia para obtener cortes de longitud 2L, ver Tabla N° 4.2.

Tabla N° 4.2 Secuencia para cortes de Tubo de longitud 2L

Condición Actual	Estado Siguiente
a1=1; a0=0; b1=0; c1=0; d0=1; d1=0	C+
a1=1; a0=0; b1=0; c1=1; d0=1; d1=0	A-
a1=0; a0=1; b1=0; c1=1; d0=1; d1=0	B+
a1=0: a0=1; b1=1; c1=1; d0=1; d1=0	C-
a1=0; a0=1; b1=1; c1=0; d0=1; d1=0	A+
a1=1; a0=0; b1=1; c1=0; d0=1; d1=0	C+
a1=1; a0=0; b1=1; c1=1; d0=1; d1=0	B-
a1=1; a0=0; b1=0; c1=1; d0=1; d1=0	A-
a1=0; a0=1; b1=0; c1=1; d0=1; d1=0	B+
a1=0: a0=1; b1=1; c1=1; d0=1; d1=0	C-
a1=0; a0=1; b1=1; c1=0; d0=1; d1=0	A+
a1=1; a0=0; b1=1; c1=0; d0=1; d1=0	C+
a1=1; a0=0; b1=1; c1=1; d0=1; d1=0	D+
a1=1; a0=0; b1=1; c1=1; d0=0; d1=1	D-
a1=1; a0=0; b1=1; c1=1; d0=1; d1=0	B-

4.2.3 Secuencia para cortes de Tubo de Longitud 3L

De la Fig. 4.1 se obtiene la secuencia para obtener cortes de longitud 3L, ver Tabla N° 4.3.

Tabla N° 4.3 Secuencia para cortes de Tubo de longitud 3L

Condición Actual	Estado Siguiente
a1=1; a0=0; b1=0; c1=0; d0=1; d1=0	C+
a1=1; a0=0; b1=0; c1=1; d0=1; d1=0	A-
a1=0; a0=1; b1=0; c1=1; d0=1; d1=0	B+
a1=0: a0=1; b1=1; c1=1; d0=1; d1=0	C-

a1=0; a0=1; b1=1; c1=0; d0=1; d1=0	A+
a1=1; a0=0; b1=1; c1=0; d0=1; d1=0	C+
a1=1; a0=0; b1=1; c1=1; d0=1; d1=0	B-
a1=1; a0=0; b1=0; c1=1; d0=1; d1=0	A-
a1=0; a0=1; b1=0; c1=1; d0=1; d1=0	B+
a1=0: a0=1; b1=1; c1=1; d0=1; d1=0	C-
a1=0; a0=1; b1=1; c1=0; d0=1; d1=0	A+
a1=1; a0=0; b1=1; c1=0; d0=1; d1=0	C+
a1=1; a0=0; b1=1; c1=1; d0=1; d1=0	B-
a1=1; a0=0; b1=0; c1=1; d0=1; d1=0	A-
a1=0; a0=1; b1=0; c1=1; d0=1; d1=0	B+
a1=0: a0=1; b1=1; c1=1; d0=1; d1=0	C-
a1=0; a0=1; b1=1; c1=0; d0=1; d1=0	A+
a1=1; a0=0; b1=1; c1=0; d0=1; d1=0	C+
a1=1; a0=0; b1=1; c1=1; d0=1; d1=0	D+
a1=1; a0=0; b1=1; c1=1; d0=0; d1=1	D-
a1=1; a0=0; b1=1; c1=1; d0=1; d1=0	B-

4.3. PROGRAMA PARA CONTROLAR UNA CORTADORA DE TUBOS

Haciendo uso de la herramienta de programación en PLC Siemens "Prosys" se ha elaborado el programa en Lenguaje Diagrama de Bloques de Función para controlar una cortadora de tubos de diferentes dimensiones.

4.3.1 Programa Cortador

Sentencia N°01.- Se habilita el arranque de motor mediante el pulsador "MARCHA" teniendo en consideración de que la variable "X0" (encargada de terminar el proceso) se encuentra en "0" lógico al igual que la variable "PARO". La variable "EMERGENCIA" como condición inicial se encuentra en "1" lógico pues se trata de un pulsador del tipo rasante. Se produce un enclavamiento en la marcha debido a la variable "M". Ver fig. 4.2.

Sentencia N°02.- La variable "M" se encarga de cerrar el contacto de motor "K1".

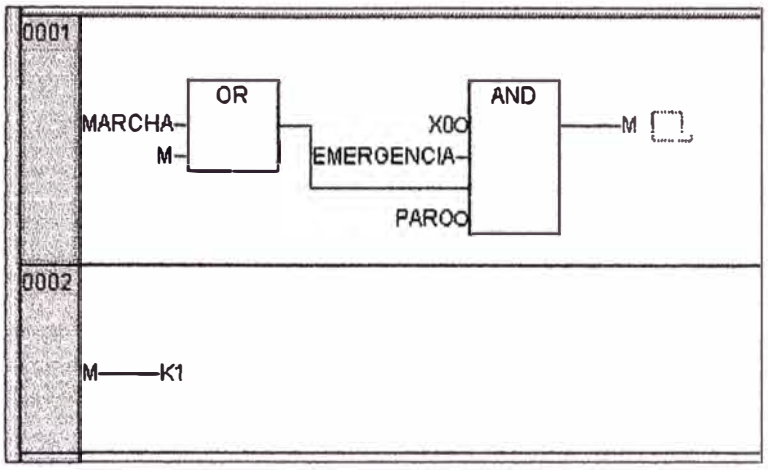

Fig. 4.2 Esquema Sentencia N°01 y N°02 del Cortador

Sentencia N°03.- Se setea la memoria "M1" mediante la variable "X0" o accionando el pulsador de emergencia, o cuando las condiciones de la figura habilitan las compuertas lógicas AND. Se resetea la memoria "M1" mediante la variable "LIBERA" o cuando las condiciones de la figura habiliten la compuerta AND. Ver fig. 4.3.

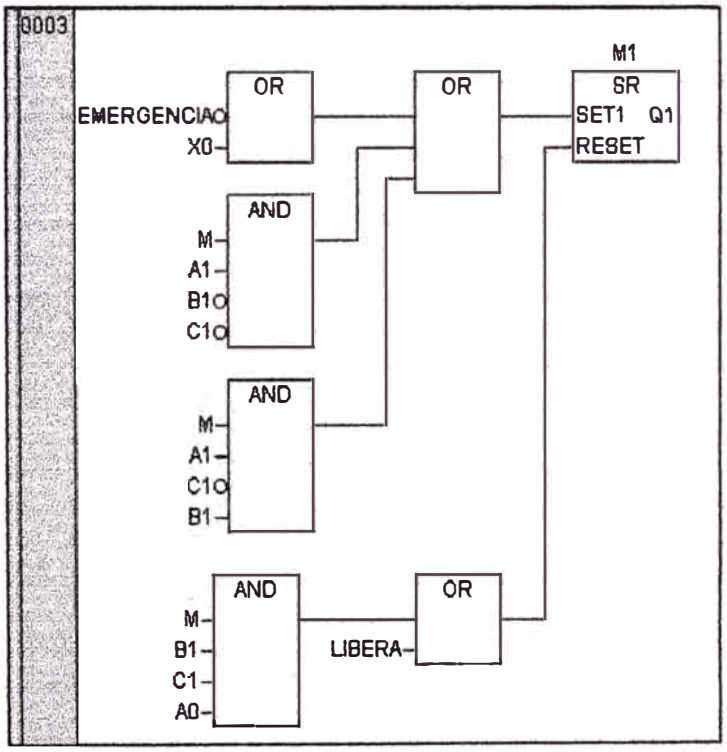

Fig. 4.3 Esquema Sentencia N°03 del Cortador

Sentencia N°04.- Por medio de la compuerta lógica AND se habilita o deshabilita la variable "Y3" dependiendo de la verdad o falsedad de la variable "Sb" y del seteo o reseteo de la memoria "M1". Ver fig. 4.4.

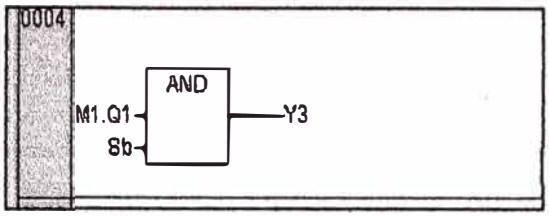

Fig. 4.4 Esquema Sentencia N°04 del Cortador

Sentencia N°05.- Se setea la memoria "M2" cuando las posiciones del proceso requerido en la figura habilitan la compuerta lógica AND. Se resetea la memoria "M2" mediante la variable "X0" o accionando el pulsador de emergencia, o si las condiciones de la figura para el proceso se cumplen habilitando la compuerta AND. Ver fig. 4.5.

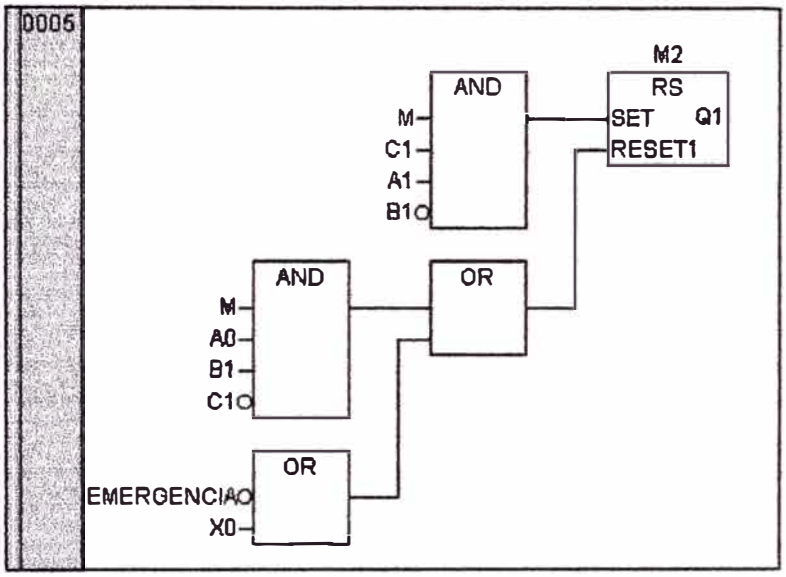

Fig. 4.5 Esquema Sentencia N°05 del Cortador

Sentencia N°06.- Por medio de la compuerta AND se habilita o deshabilita la variable "Y1" dependiendo de la variable "Sb" y del estado de la memoria "M2". Ver fig. 4.6.

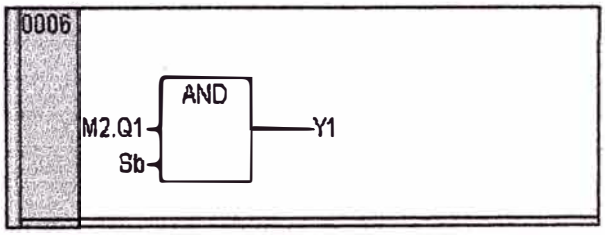

Fig. 4.6 Esquema Sentencia N°06 del Cortador

Sentencia N°07.- Se setea la memoria "M3" por medio de la variable "X0" o accionando el pulsador de emergencia, o si se cumplen las condiciones habilitando la compuerta AND. Se resetea la memoria "M3" por medio de las variables C12Q y X2, o por medio de C13Q y X3, o si las condiciones de posición habilitan AND. Ver fig. 4.7.

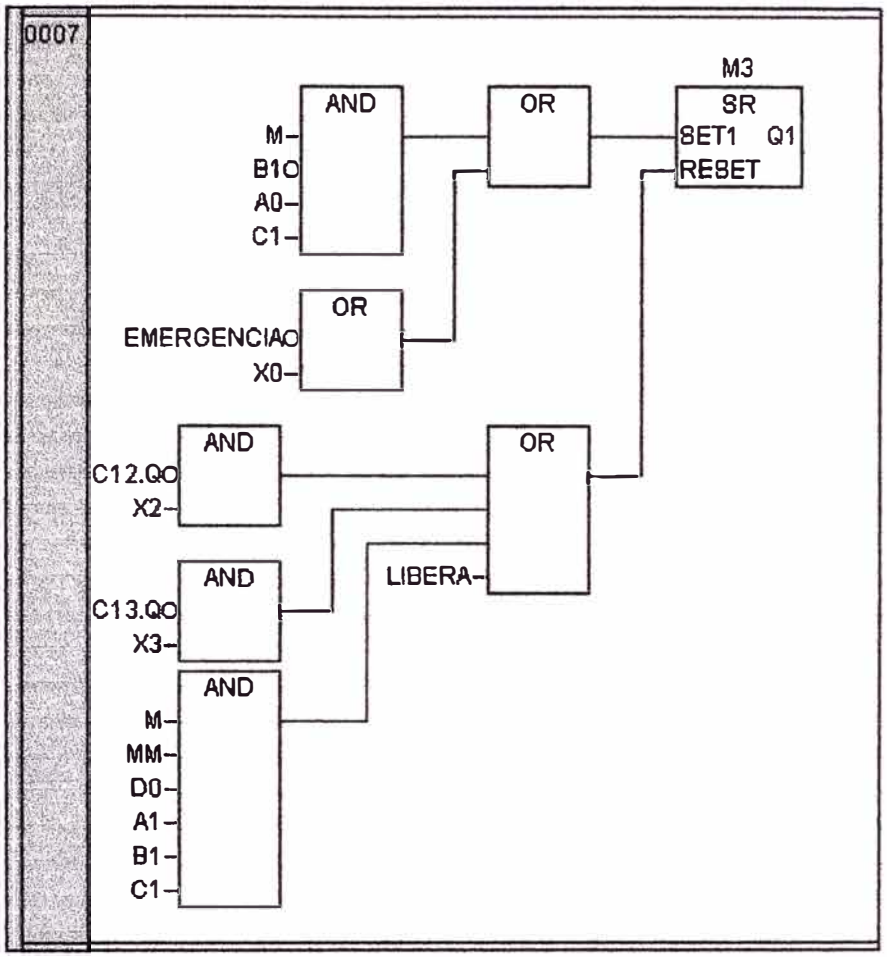

Fig. 4.7 Esquema Sentencia N°07 del Cortador

Sentencia N°08.- Por medio de la compuerta AND se habilita o deshabilita la variable "Y2" dependiendo de la variable "Sb" y del estado de la memoria "M3". Ver fig. 4.8.

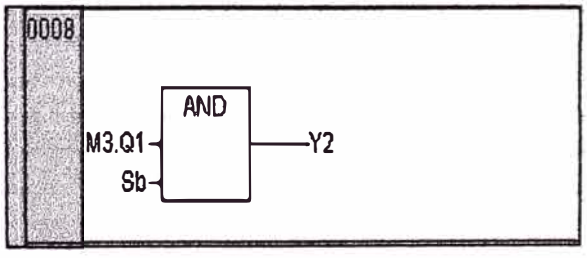

Fig. 4.8 Esquema Sentencia N°08 del Cortador

Sentencia N°09.- Se setea la memoria "M4" por medio de las variables X1 o C12Q o C13Q, y la falsedad de la variable "MM" (encargada de evitar el movimiento oscilatorio de la cortadora) y si las condiciones de posición habilitan la compuerta AND. Se resetea la memoria "M4" por medio de la variable X0 o accionando el pulsador de emergencia, o si las condiciones de posición habilitan la compuerta AND. Ver fig. 4.9.

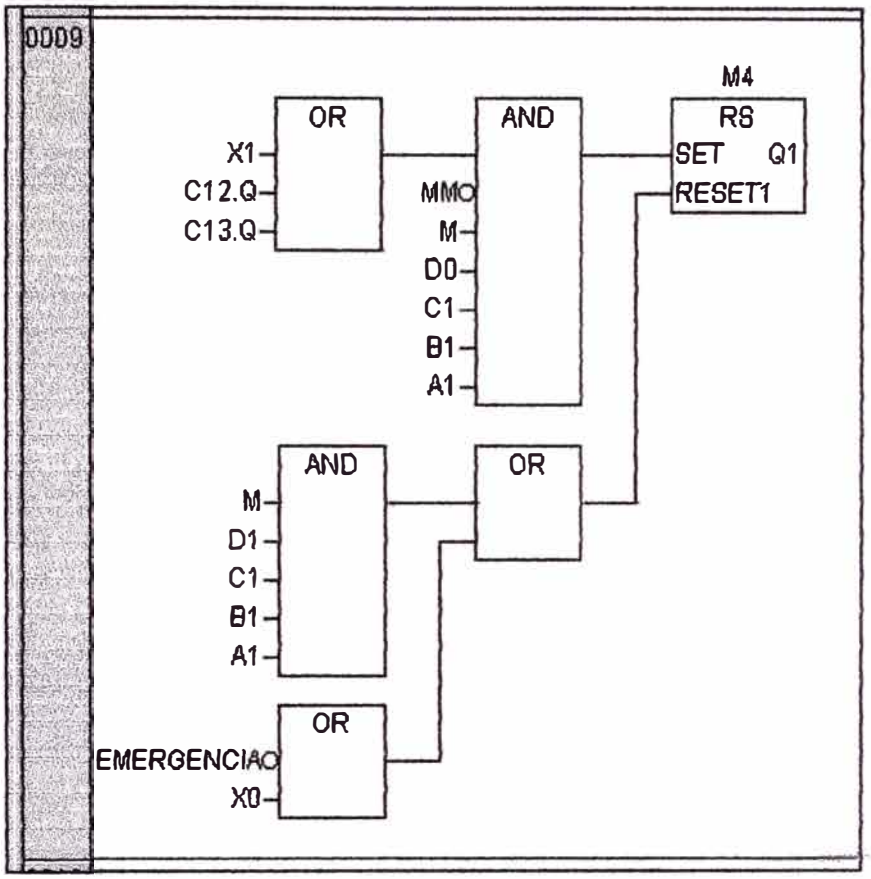

Fig. 4.9 Esquema Sentencia N°09 del Cortador

Sentencia N°10.- Por medio de la compuerta AND se habilita o deshabilita la variable "Y4" dependiendo de la variable "Sb" y del estado de la memoria "M4".

Sentencia N°11.- Por medio de la compuerta AND se habilita o deshabilita la variable "MM" dependiendo de las condiciones de posición de las variables indicadas, además se produce un enclavamiento con la variable D1.

Para mayor detalle de las sentencias N°10 y N°11 ver fig. 4.10.

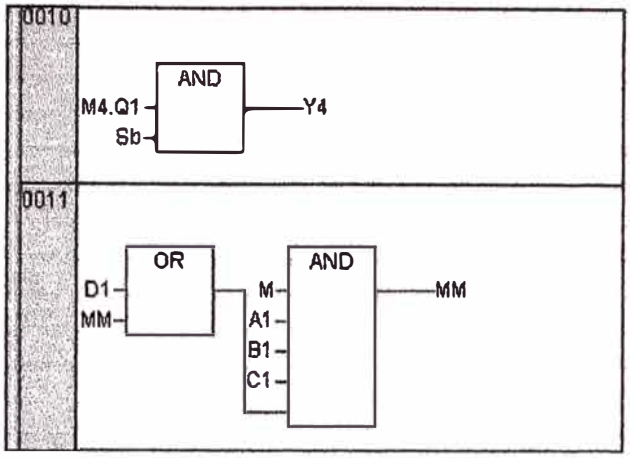

Fig. 4.10 Esquema Sentencia N°10 y N°11 del Cortador

4.3.2 Programa Contador

Sentencia N°01.- Se habilita un contador ascendente C11, en donde los pulsos de entrada están dados por la variable "Y4", el reset se realiza a través de la variable "X0" y el valor PV viene representado por la variable "CANTIDAD" que viene a ser la cantidad total de cortes que se van a realizar.

Sentencia N°02.- Se determina de que la variable "X0" sea habilitada por una función de igualdad entre las variables "CUENTA" y "CANTIDAD", y un enclavamiento con la variable D1. Ver fig. 4.11.

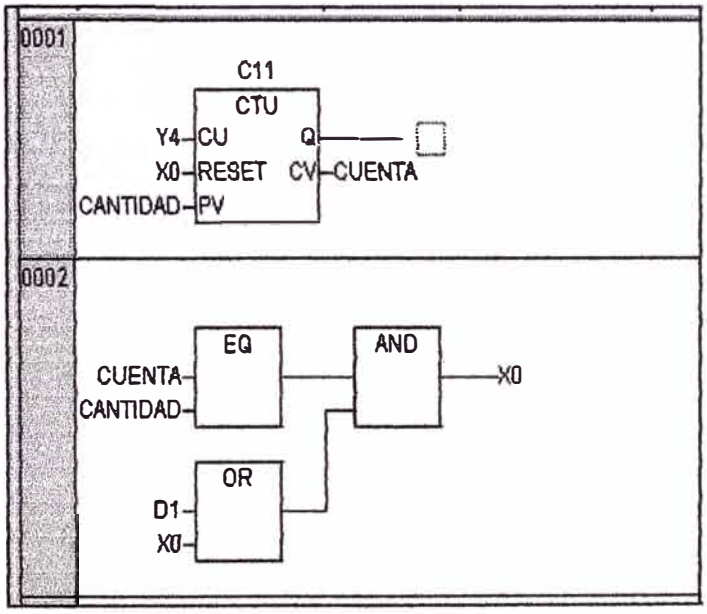

Fig. 4.11 Esquema Sentencia N°01 y N°02 del Contador

Sentencia N°03.- Por medio de la compuerta AND se habilita o deshabilita la variable "X1" dependiendo de las condiciones de posición de las variables indicadas y de que la variable "LONGITUD" (longitud del tubo deseado) tome el valor de 1. Ver fig. 4.12.

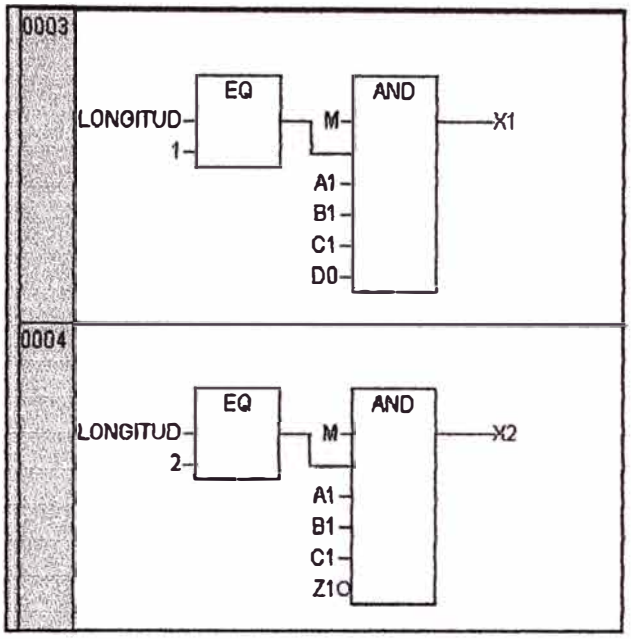

Fig. 4.12 Esquema Sentencia N°03 y N°04 del Contador

Sentencia N°04.- Por medio de la compuerta AND se habilita o deshabilita la variable "X2" dependiendo de las condiciones de posición de las variables indicadas y de que la variable "LONGITUD" (longitud del tubo deseado) tome el valor de 2.

Sentencia N°05.- Por medio de la compuerta AND se habilita o deshabilita la variable "Z1" (encargada de evitar que el contador C12 siga contando en el estado siguiente al corte, para lo cual se enclava con la variable D1) dependiendo de las condiciones de posición de las variables indicadas.

Sentencia N°06.- Utilizando un tren de pulsos con periodicidad de 1 seg, la variable "Z1" se encarga de habilitar el temporizador "T1" el cual a su vez sirve para resetear al contador ascendente "C12" cuando la variable "CUENTA1" tome el valor de la variable "PRESET1=2". Los pulsos de la entrada CU del contador dependen de X2.

Para mayor detalle de las sentencias N°05 y N°06 del Contador, ver fig. 4.13.

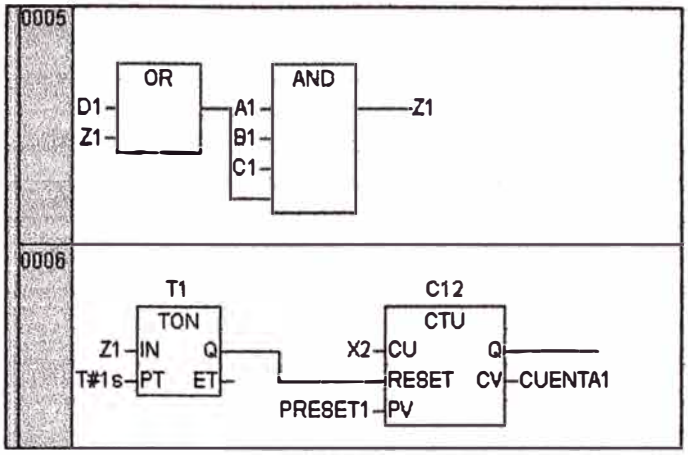

Fig. 4.13 Esquema Sentencia N°05 y N°06 del Contador

Sentencia N°07.- Por medio de la compuerta AND se habilita o deshabilita la variable "X3" dependiendo de las condiciones de posición de las variables indicadas y de que la variable "LONGITUD" (longitud del tubo deseado) tome el valor de 3.

Sentencia N°08.- Por medio de la compuerta AND se habilita o deshabilita la variable "Z2" (encargada de evitar que el contador C13 siga contando en el estado siguiente al corte, para lo cual se enclava con la variable D1) dependiendo de las condiciones de posición de las variables indicadas. Ver fig. 4.14.

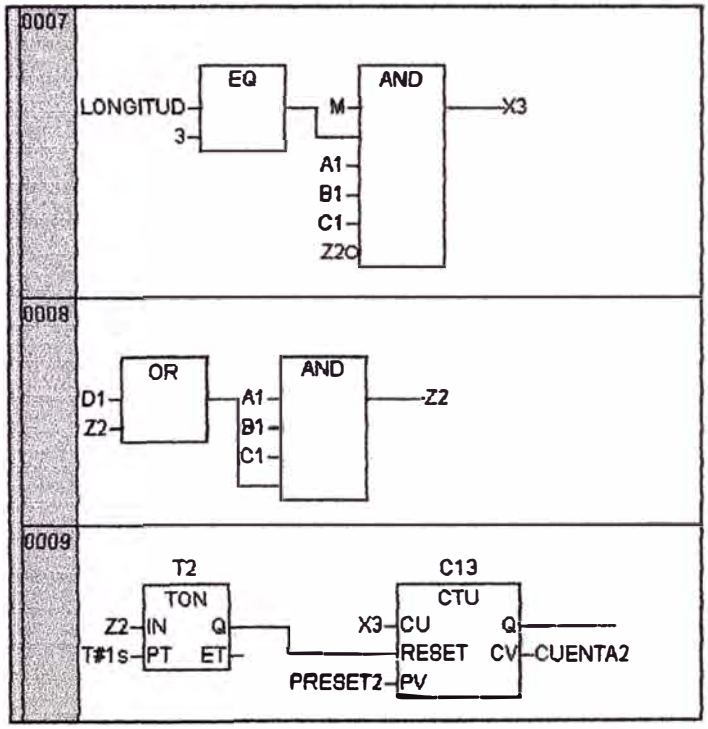

Fig. 4.14 Esquema Sentencia N°07, N°08 y N°09 del Contador

Sentencia N°09.- Utilizando un tren de pulsos con periodicidad de 1 seg, la variable "Z2" se encarga de habilitar el temporizador "T2" el cual a su vez sirve para resetear al contador ascendente "C13" cuando la variable "CUENTA2" tome el valor de la variable "PRESET2=3". Los pulsos de la entrada CU del contador dependen de la variable "X3".

Una vez conocidos los sub-programas "Cortador" y "Contador" se procedió a consolidarlos en un tercer sub-programa "OB1" para que las sentencias declaradas en cada sub-programa independiente sean ejecutadas en forma paralela si las condiciones de posición así lo establecen; y para que las variables declaradas en cada uno de los sub-programas puedan interactuar de manera eficiente entre ellas. Ver fig. 4.15.

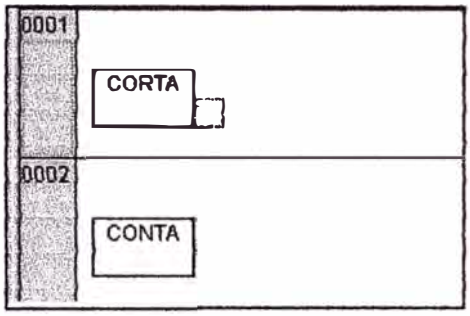

Fig. 4.15 Esquema Sub-programa OB1

4.4. SIMULACION EN MODO TEXTO DEL PROGRAMA "CORTADORA DE TUBOS DE DIFERENTES DIMENSIONES"

A continuación se procederá a realizar la simulación del programa desarrollado "Control de una cortadora de tubos de 3 dimensiones proporcionales" en lenguaje de máquina (Diagrama de Bloques de Función) utilizando para ello el Modo Simulación de la herramienta de programación Prosys.

4.4.1 Simulación en lenguaje de máquina para cortes de tubo de longitud "L"

Paso 1a.- Se acciona el pulsador de marcha para energizar la bobina del motor que controla a la cortadora dejando la variable "M" habilitada y enclavada, la variable "Emergencia" se encuentra habilitada desde el inicio, además, el sensor "Sb" detecta la posición del tubo y se habilita, así como, la variable "D0" que indica la posición inicial de la cortadora. Ver fig. 4.16.

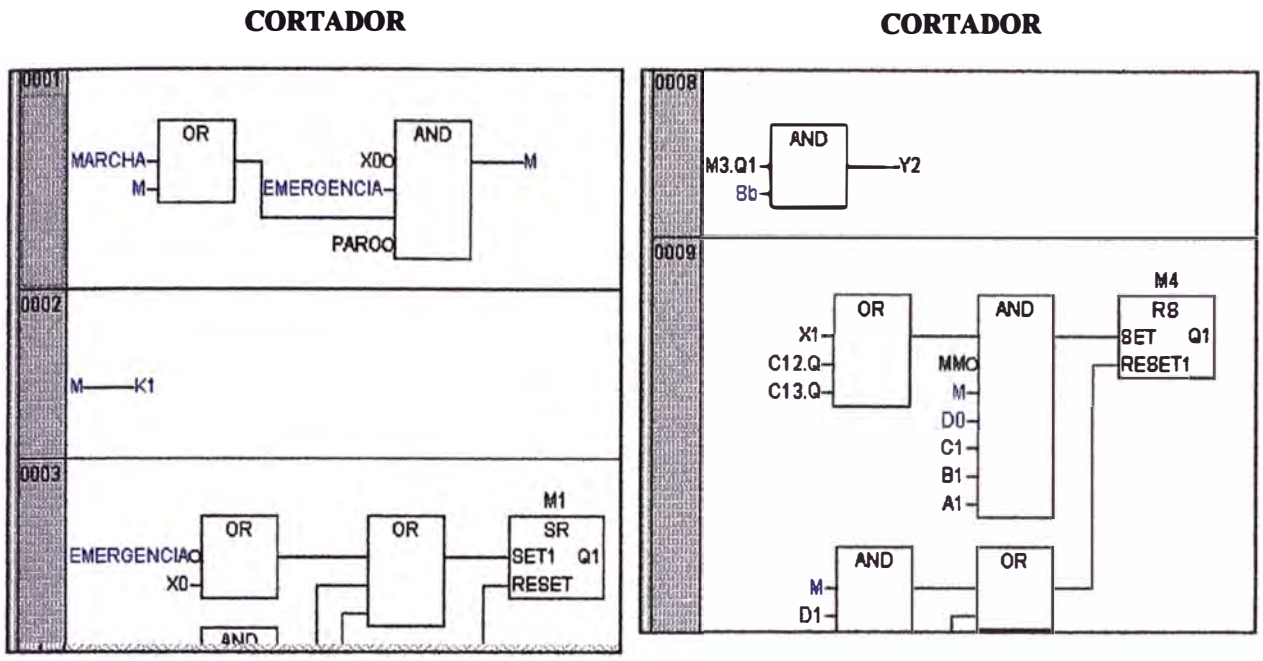

Fig. 4.16 Esquema de simulación en modo texto – Paso 1a

Paso 2a.- La posición del carril A (a1=1, a0=0) es detectada por el sensor de posición que setea la memoria M1 (Sb habilitado), enviando una orden de habilitación de la variable Y3 la cual produce un movimiento del retenedor C en la dirección C+. Ver fig. 4.17.

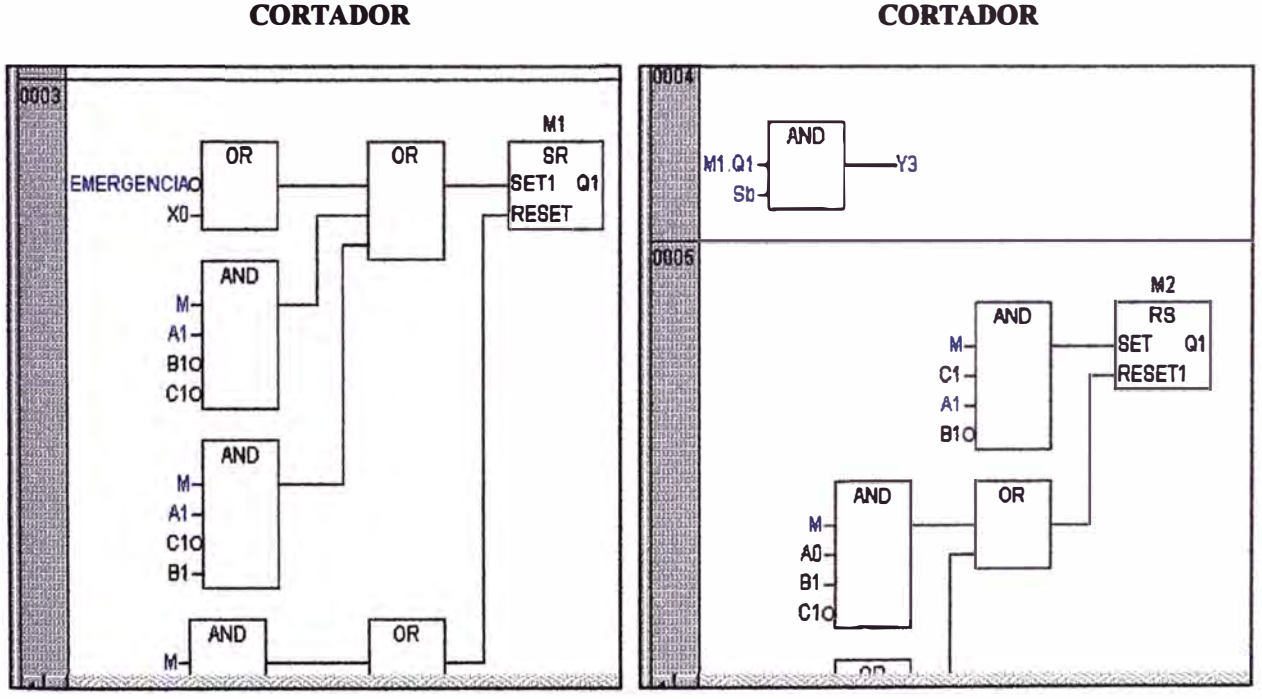

Fig. 4.17 Esquema de simulación en modo texto – Paso 2a

Paso 3a.- La posición del retenedor C (c1=1) es detectada por el sensor de posición que setea la memoria M2 (Sb habilitado), enviando una orden de habilitación de la variable Y1 la cual produce un movimiento del carril A en la dirección A-. Ver fig. 4.18.

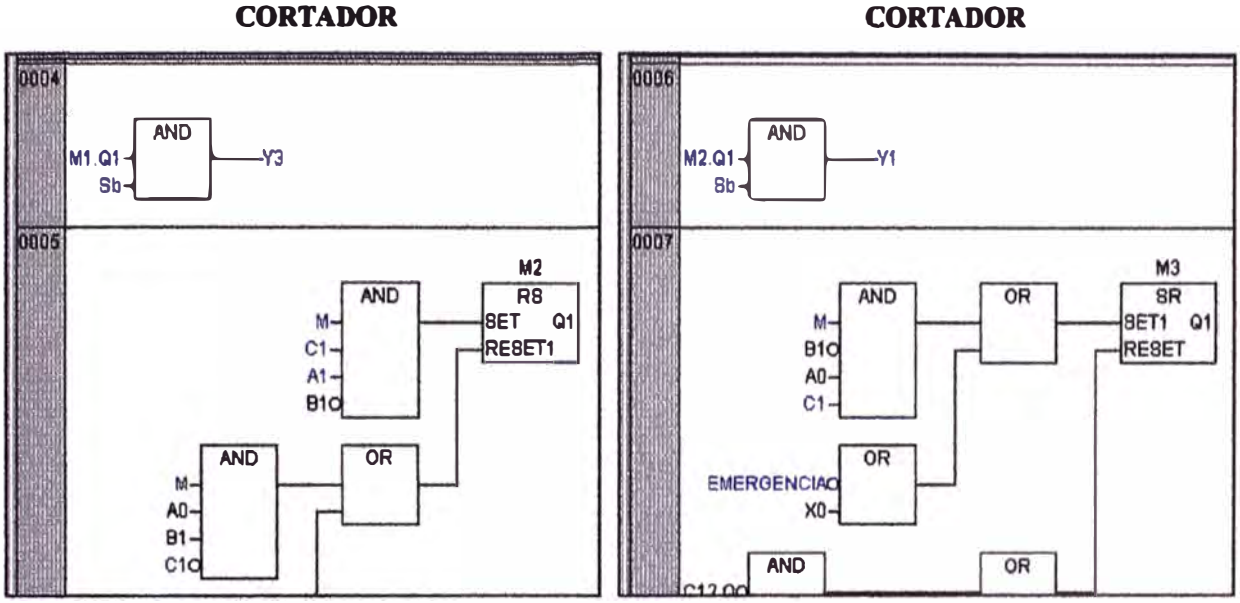

Fig. 4.18 Esquema de simulación en modo texto – Paso 3a

Paso 4a.- La posición del carril A (a1=0, a0=1) es detectada por el sensor de posición que setea la memoria M3 (Sb habilitado), enviando una orden de habilitación de la variable Y2 la cual produce un movimiento del retenedor B en la dirección B+. Ver fig. 4.19.

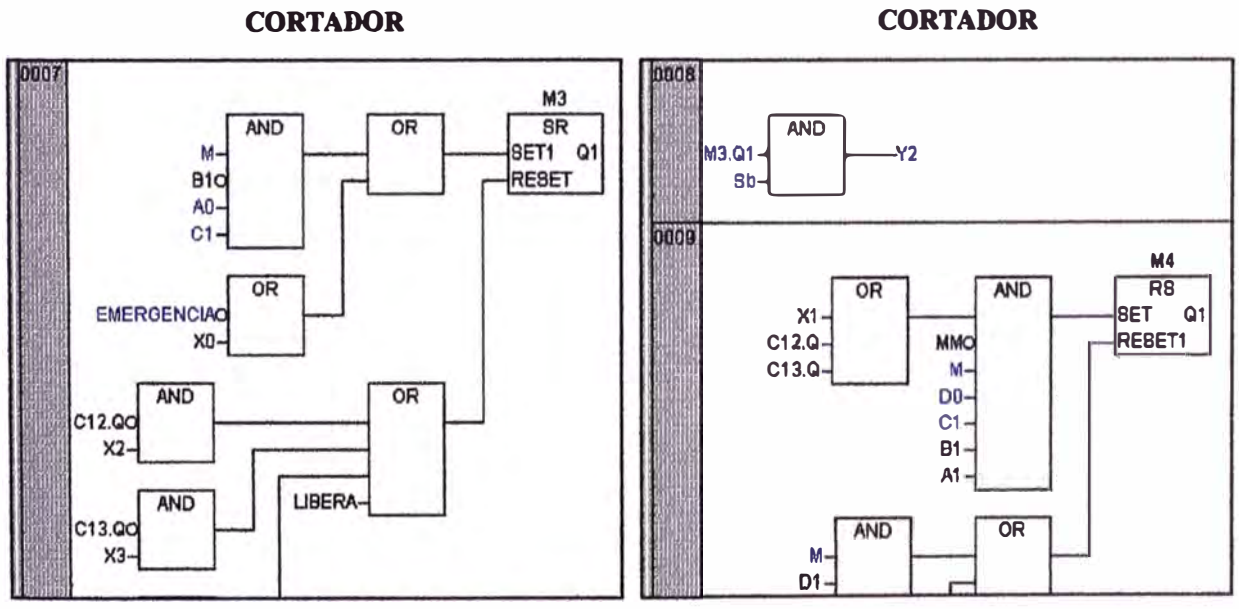

Fig. 4.19 Esquema de simulación en modo texto – Paso 4a

Paso 5a.- La posición del retenedor B (b1=1) es detectada por el sensor de posición que resetea la memoria M1, enviando una orden de deshabilitación de la variable Y3 la cual produce un movimiento del retenedor C en la dirección C-. Ver fig. 4.20.

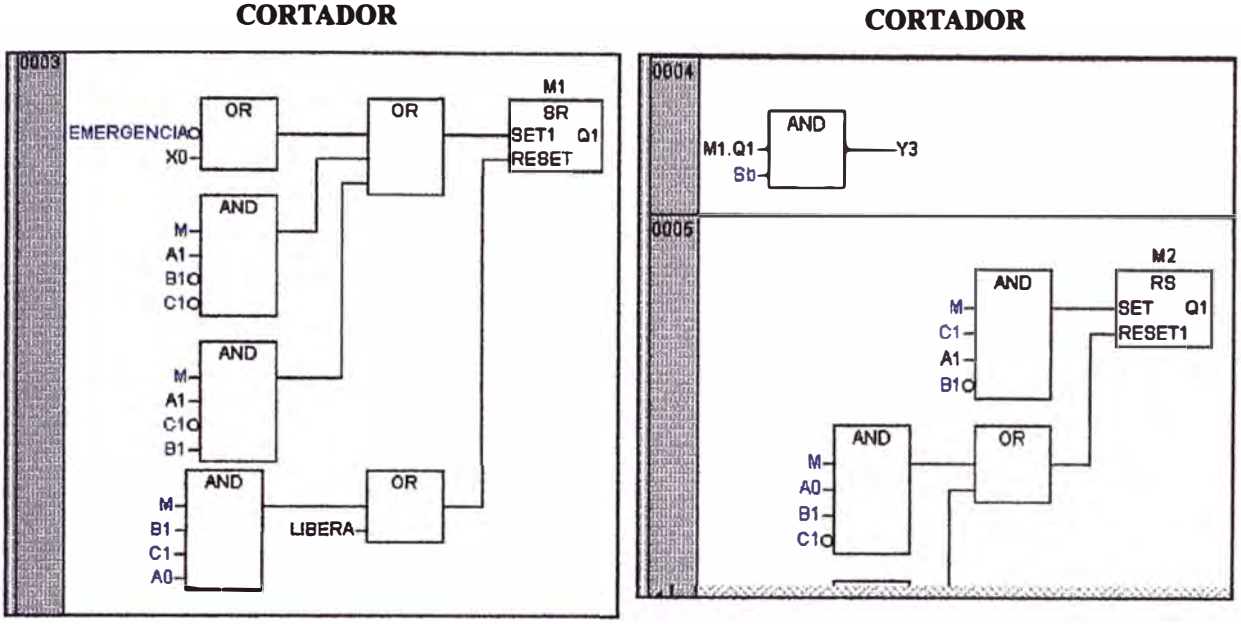

Fig. 4.20 Esquema de simulación en modo texto – Paso 5a

Paso 6a.- La posición del retenedor C (c1=0) es detectada por el sensor de posición que resetea la memoria M2, enviando una orden de deshabilitación de la variable Y1 la cual produce un movimiento del carril A en la dirección A+. Ver fig. 4.21.

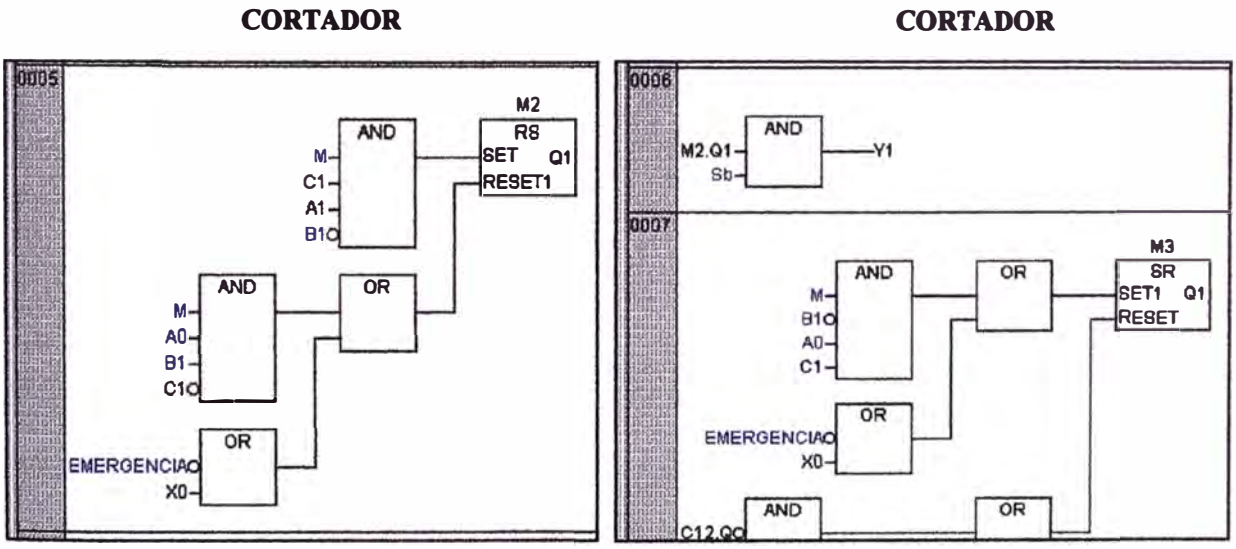

Fig. 4.21 Esquema de simulación en modo texto – Paso 6a

Paso 7a.- La posición del carril A (a1=1, a0=0) es detectada por el sensor de posición que setea la memoria M1 (Sb habilitado), enviando una orden de habilitación de la variable Y3 la cual produce un movimiento del retenedor C en la dirección C+. Ver fig. 4.22.

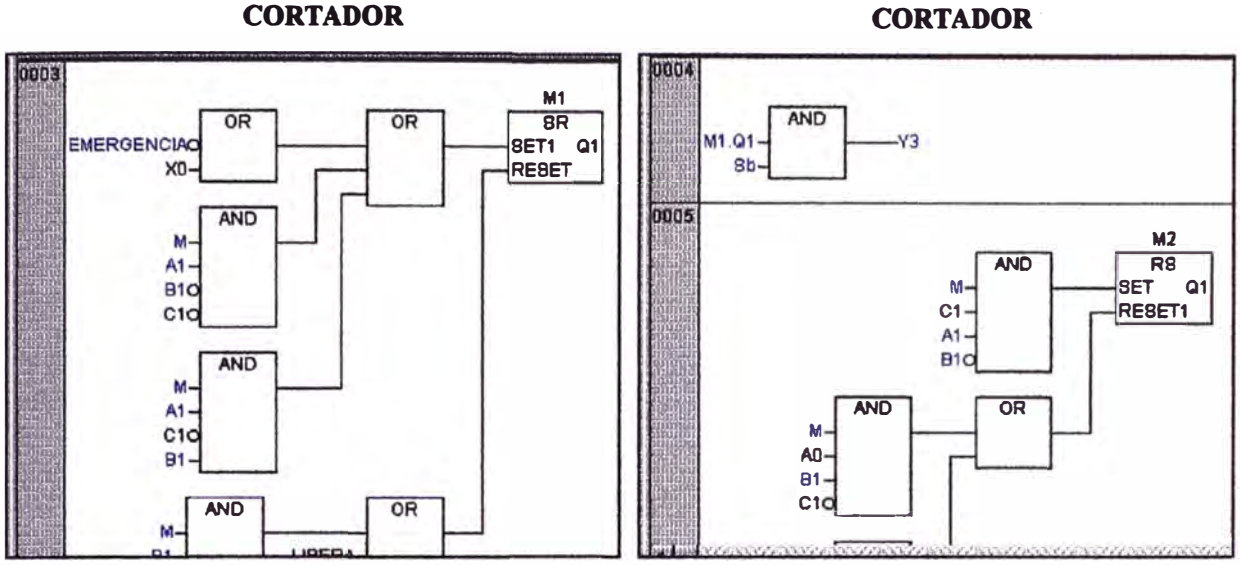

Fig. 4.22 Esquema de simulación en modo texto – Paso 7a

Paso 8a.- La posición de retenedor C (c1=1) es detectada por el sensor que habilita la variable X1, setea la memoria M4 (Sb habilitado), habilitando la variable Y4 la que mueve la cortadora en la dirección D+, e incrementa la variable "CUENTA" en 1. Ver fig. 4.23.

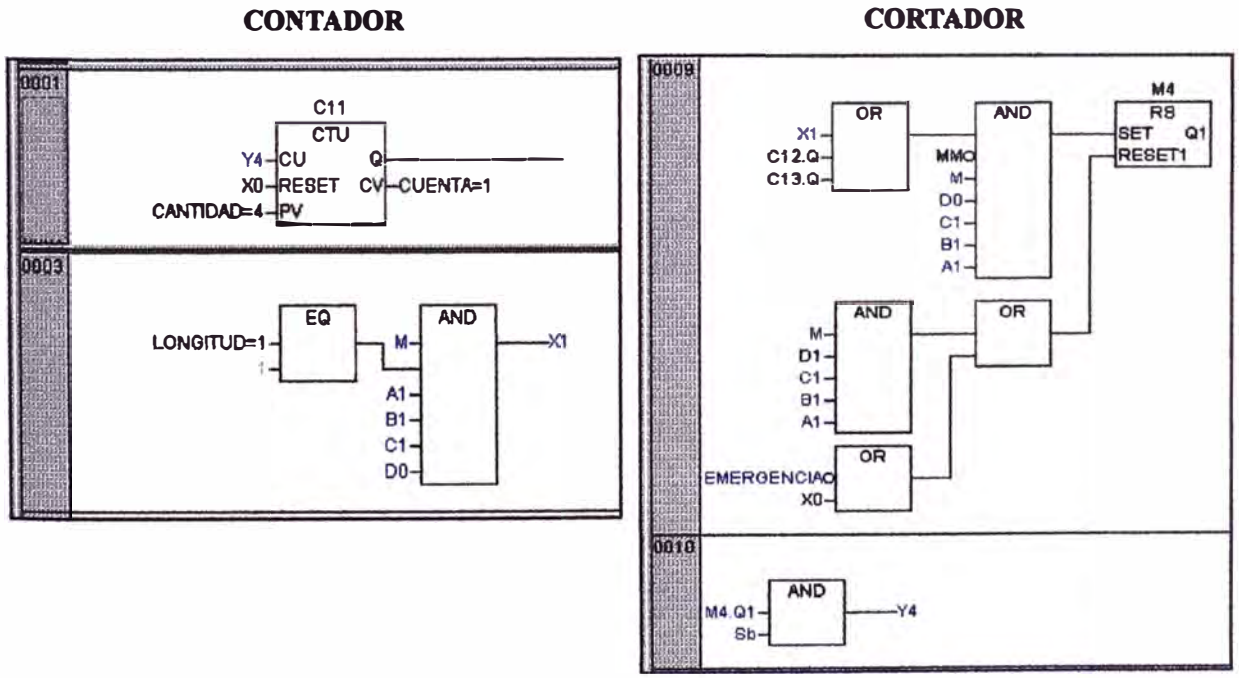

Fig. 4.23 Esquema de simulación en modo texto – Paso 8a

Paso 9a.- La posición de la cortadora (d0=0, d1=1) es detectada por el sensor que resetea la memoria M4, deshabilitando la variable Y4 y moviendo la cortadora en D-. Ver fig. 4.24.

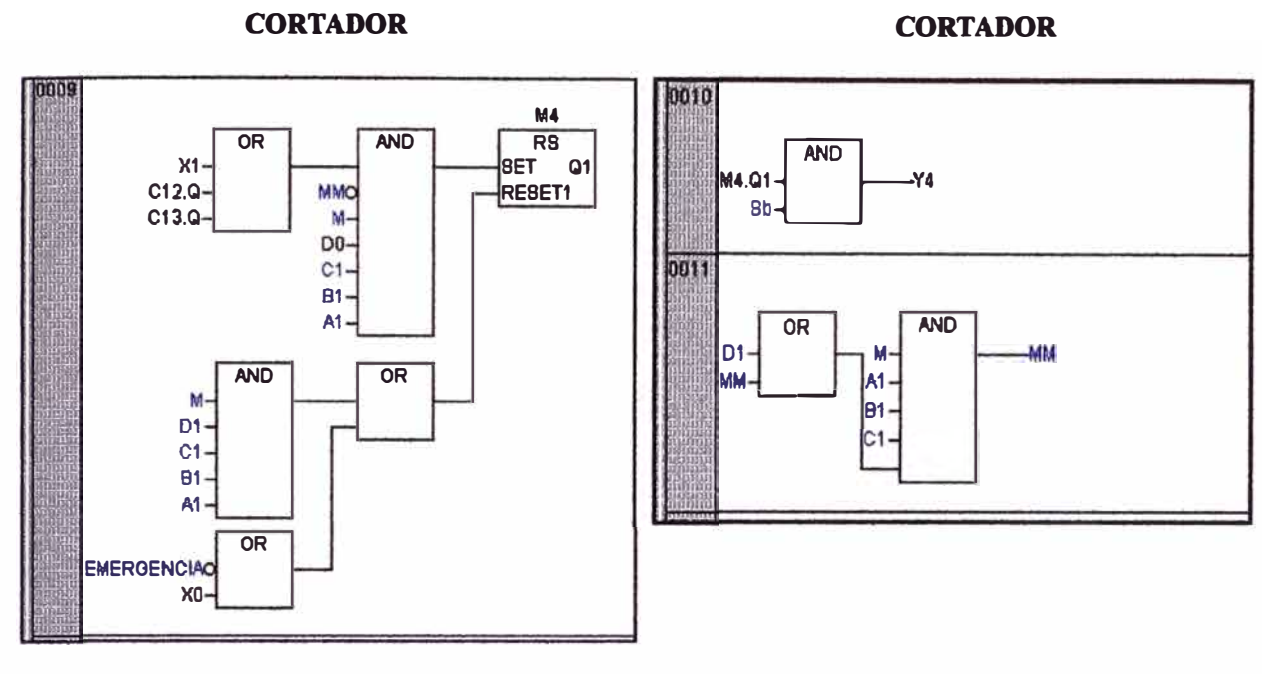

Fig. 4.24 Esquema de simulación en modo texto – Paso 9a

Nota:

Si el valor de la variable "CANTIDAD" asignada es igual al valor de la variable "CUENTA" y d1=1 entonces la variable "X0" se habilita y el proceso finaliza con la deshabilitación de la variable "M", la habilitación de las variables Y2 e Y3 y la deshabilitación de las variables Y1 e Y4 para seguridad del proceso y del operador. Caso contrario el proceso de cortado continúa en el "Paso 10a".

Paso 10a.- La posición de la cortadora (d0=1, d1=0) es detectada por el sensor de posición que resetea la memoria M3, deshabilitando la variable Y2 y moviendo el retenedor B en la dirección B-. La variable "MM" queda enclavada para evitar que la memoria M4 vuelva a ser seteada. El proceso se repite desde el "Paso 3a" hasta que el valor de la variable "CANTIDAD" sea igual al valor de la variable "CUENTA". Ver fig. 4.25.

CORTADOR **CORTADOR**

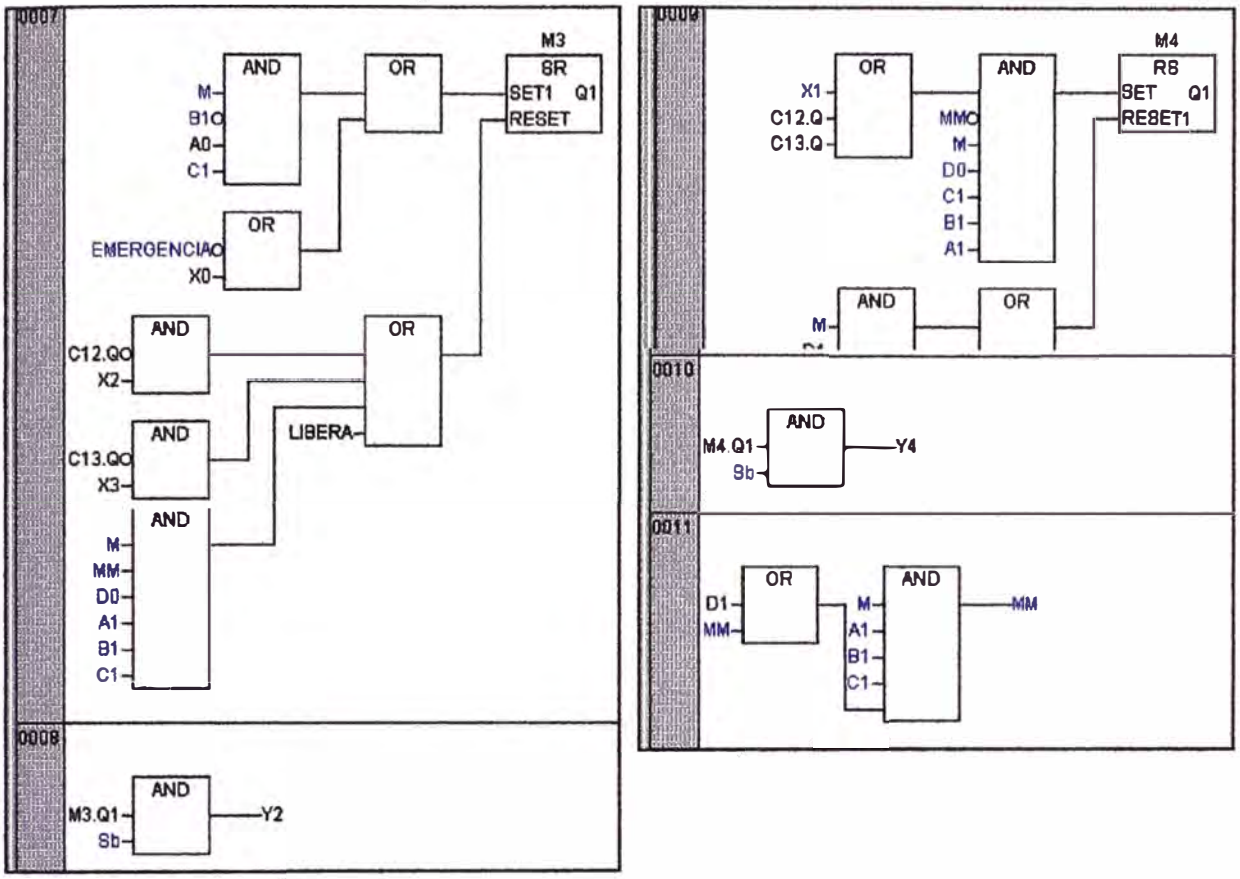

Fig. 4.25 Esquema de simulación en modo texto – Paso 10a

4.4.2 Simulación en lenguaje de máquina para cortes de tubo de longitud "2L"

Se repite el Procedimiento desde el "Paso 1a" hasta el "Paso 2a".

Paso 1b.- Desde el Paso 3a hasta el Paso 7a. La posición del retenedor C (c1=1) se detecta, habilita la variable X2, aumenta la cuenta del contador C12 en 1, así las variables C12.Q y X2 resetean la memoria M3 y mueve el retenedor B en sentido B-. Ver fig. 4.26.

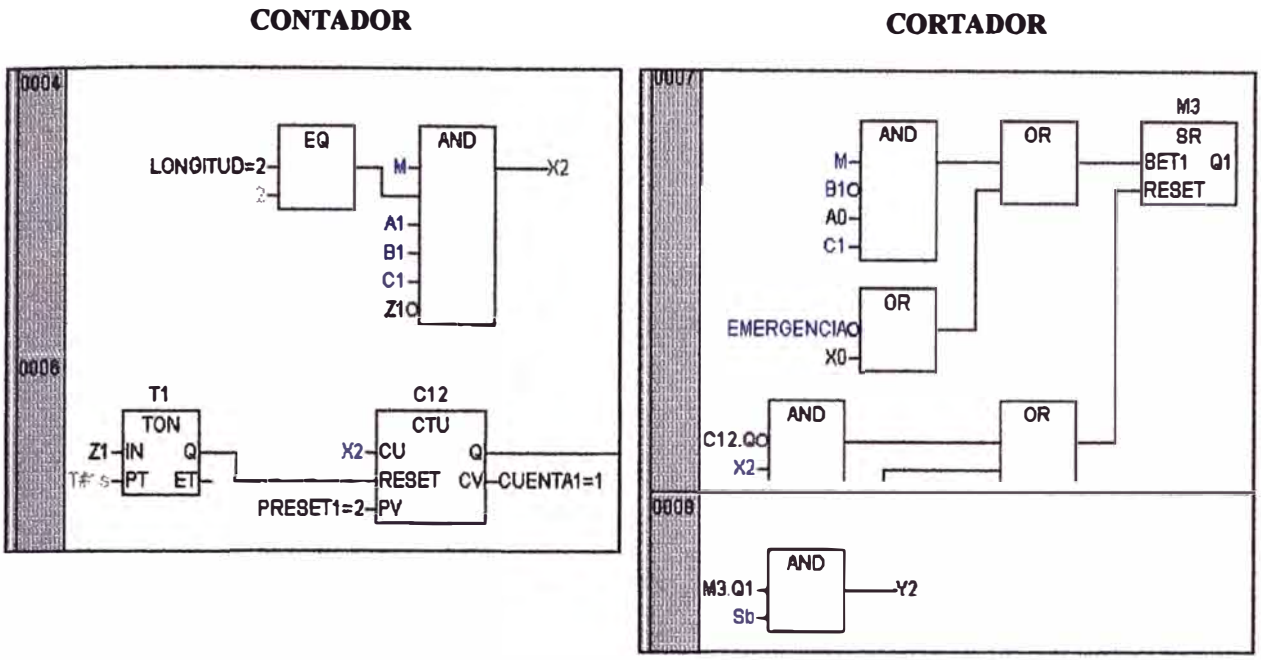

Fig. 4.26 Esquema de simulación en modo texto – Paso 1b

Paso 2b.- Se repite el procedimiento desde el "Paso 3a" hasta el "Paso 7a". La posición del retenedor C (c1=1) habilita la variable X2, aumentando la cuenta del contador C12 en 2, la variable C12.Q setea la memoria M4 y habilita la variable Y4. Ver fig. 4.27.

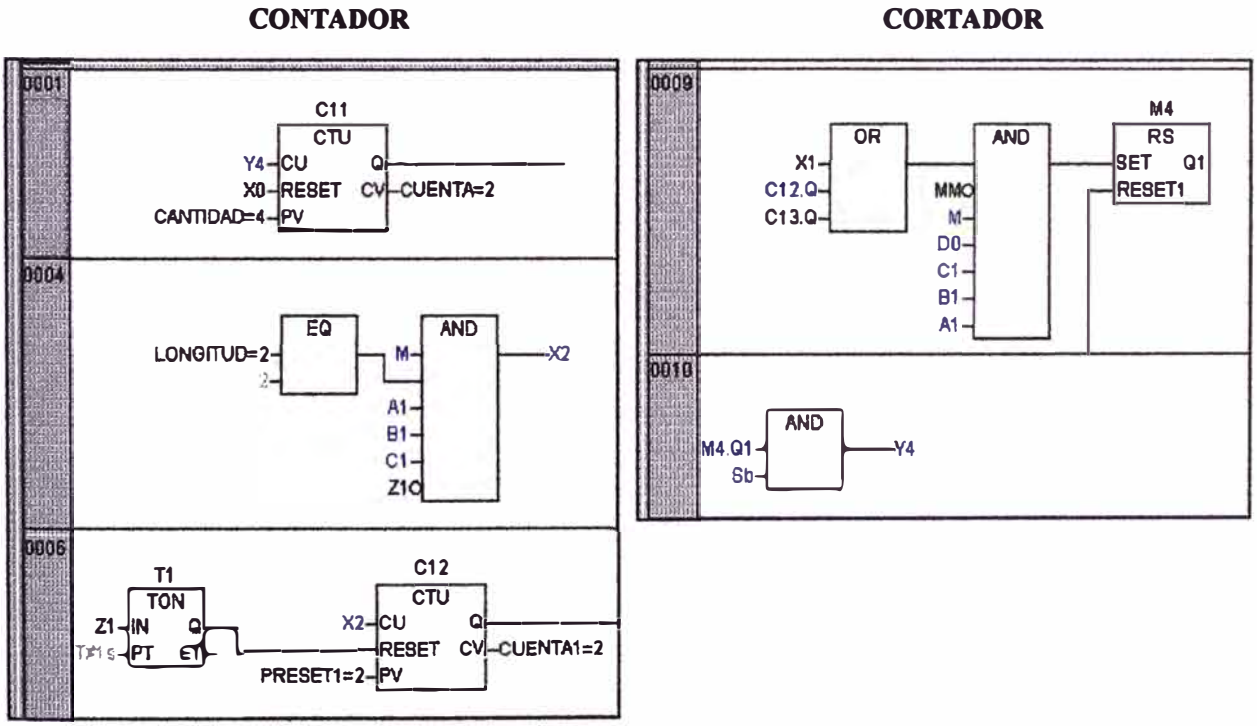

Fig. 4.27 Esquema de simulación en modo texto – Paso 2b

Paso 3b.- La posición de la cortadora (d0=0, d1=1) es detectada por el sensor que resetea memoria M4, deshabilita la variable X2, habilita la variable Z1 que activa el temporizador T1 que resetea al contador C12, y mueve la cortadora en dirección D-. Ver fig. 4.28.

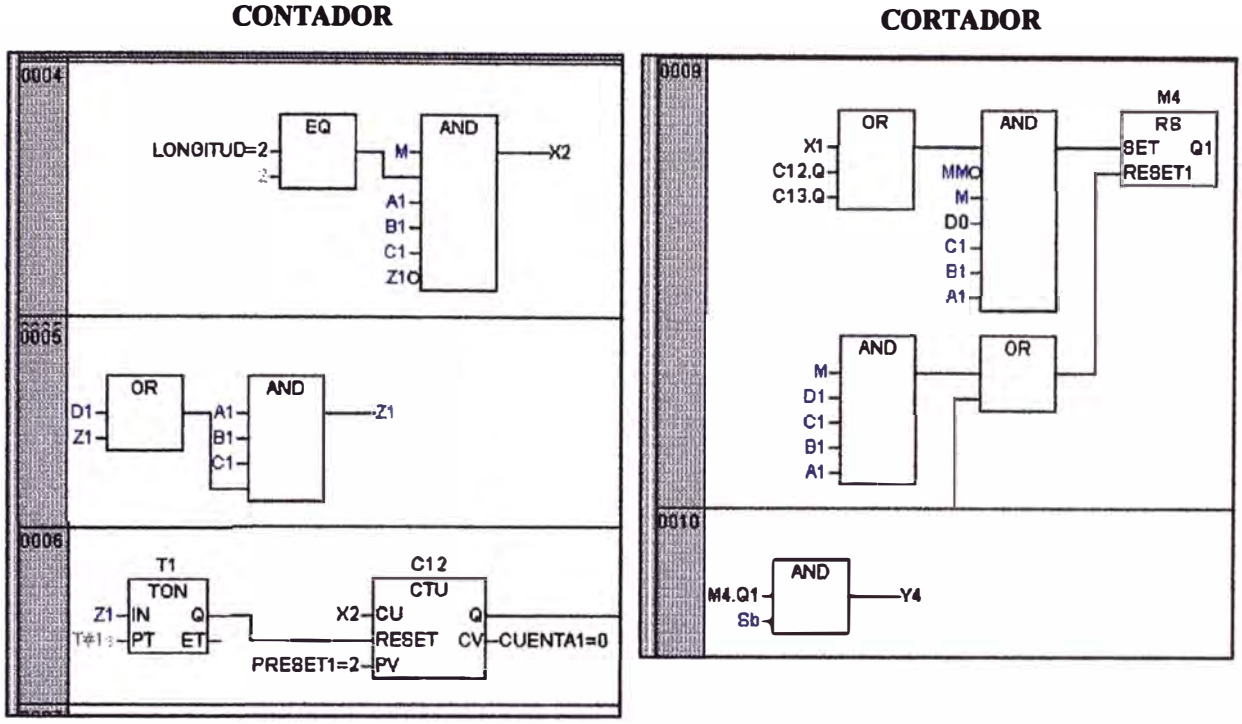

Fig. 4.28 Esquema de simulación en modo texto – Paso 3b

Nota:

Si el valor de la variable "CANTIDAD" asignada es igual al valor de la variable "CUENTA" y d1=1 entonces la variable "X0" se habilita y el proceso se detiene completamente con la deshabilitación de la variable "M", además de la habilitación de las variables "Y2" e "Y3" y la deshabilitación de las variables "Y1" e "Y4" por cuestiones de seguridad tanto para el proceso como para el operador. Caso contrario el proceso de cortado continúa en el "Paso 4b".

Paso 4b.- La posición de la cortadora (d0=1, d1=0) es detectada por el sensor de posición que resetea la memoria M3, deshabilitando la variable Y2 y moviendo el retenedor B en la dirección B-. La variable "MM" queda enclavada para evitar que la memoria M4 vuelva a ser seteada. El proceso se repite desde el "Paso 1b" hasta que el valor de la variable "CANTIDAD" sea igual al valor de la variable "CUENTA". Ver fig. 4.29.

CORTADOR **CORTADOR**

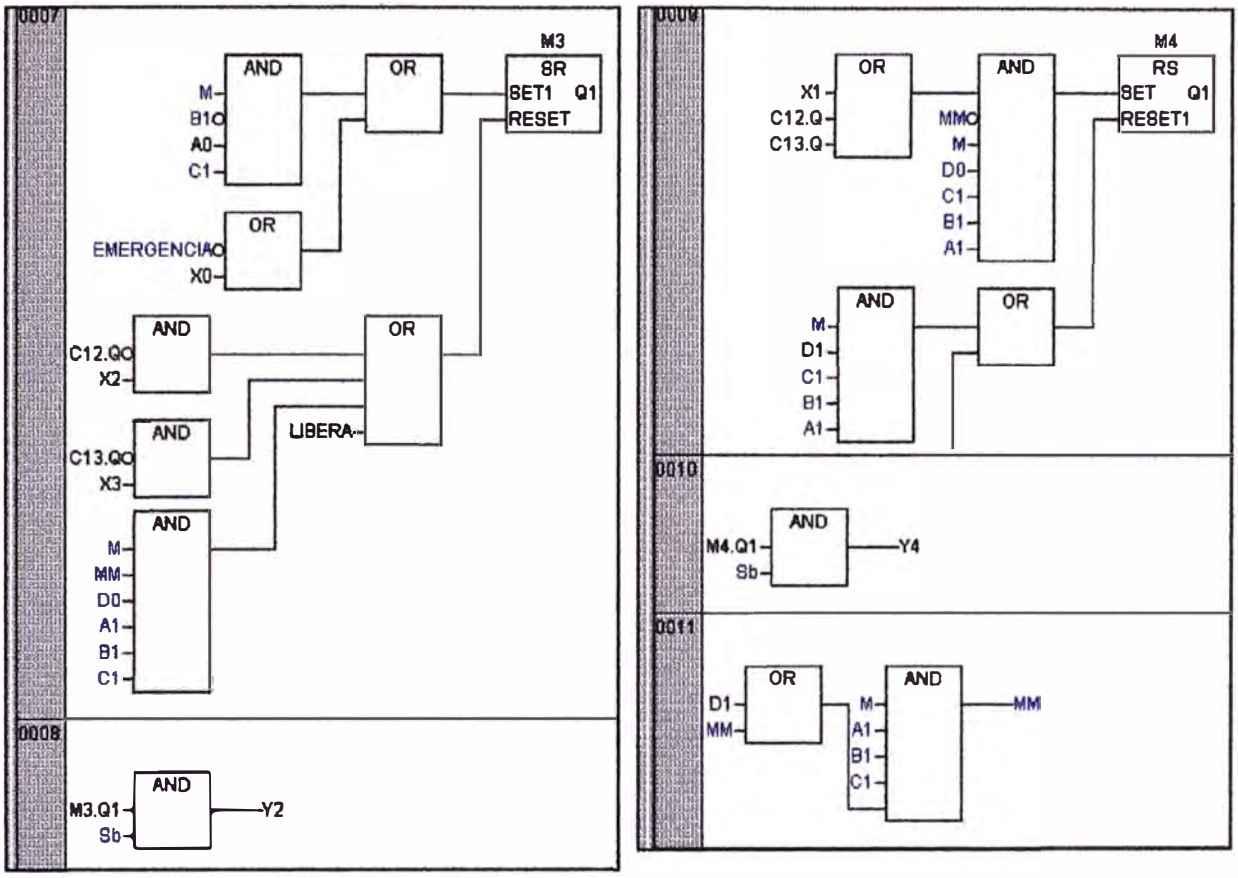

Fig. 4.29 Esquema de simulación en modo texto – Paso 4b

4.4.3 Simulación en lenguaje de máquina para cortes de tubo de longitud "3L"

Se repite el procedimiento desde el "Paso 1a" hasta el "Paso 2a".

Paso 1c.- Desde el Paso 3a hasta el Paso 7a. La posición del retenedor C (c1=1) se detecta, habilita la variable X3, aumenta la cuenta del contador C13 en 1, así las variables C13.Q y X3 resetean la memoria M3 y mueve el retenedor B en sentido B-. Ver fig. 4.30.

CONTADOR **CORTADOR**

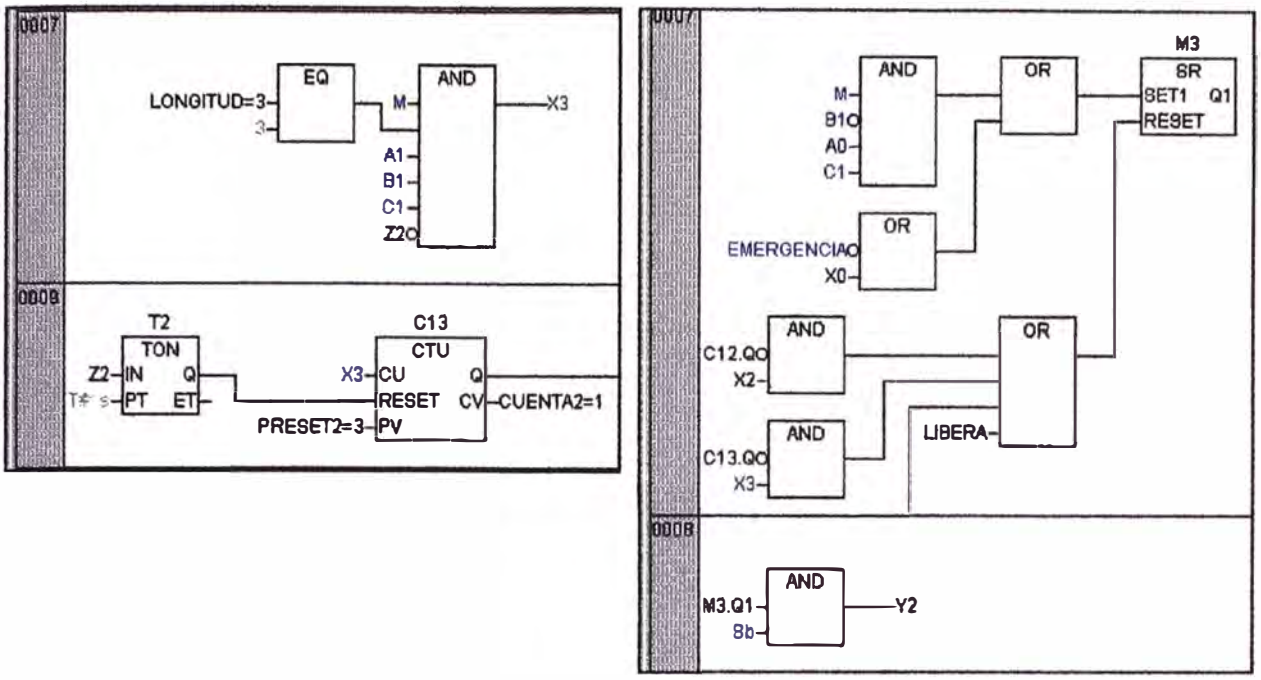

Fig. 4.30 Esquema de simulación en modo texto – Paso 1c

Paso 2c.- Desde el Paso 3a hasta el Paso 7a. La posición del retenedor C (c1=1) se detecta, habilita la variable X3, aumenta la cuenta del contador C13 en 2, así las variables C13.Q y X3 resetean la memoria M3 y mueve el retenedor B en sentido B-. Ver fig. 4.31.

CONTADOR **CORTADOR**

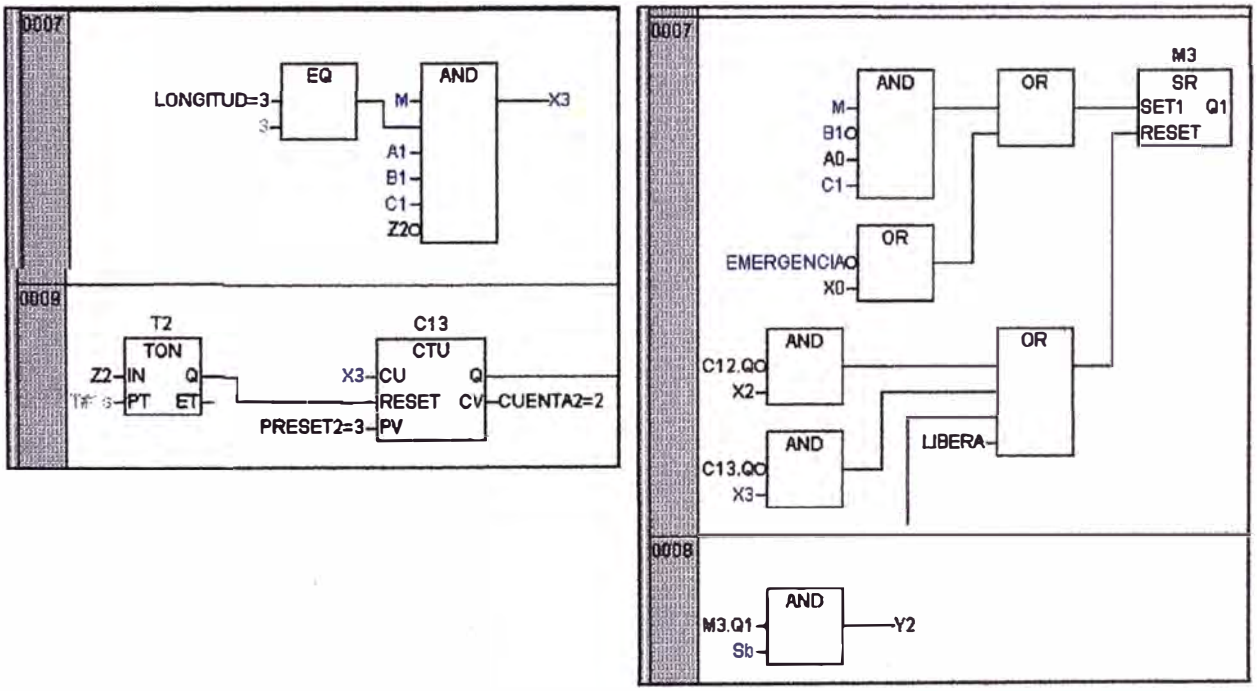

Fig. 4.31 Esquema de simulación en modo texto – Paso 2c

Paso 3c.- Se repite el procedimiento desde el "Paso 3a" hasta el "Paso 7a". La posición del retenedor C (c1=1) habilita la variable X3, aumentando la cuenta del contador C13 en 3, la variable C13.Q setea la memoria M4 y habilita la variable Y4. Ver fig. 4.32.

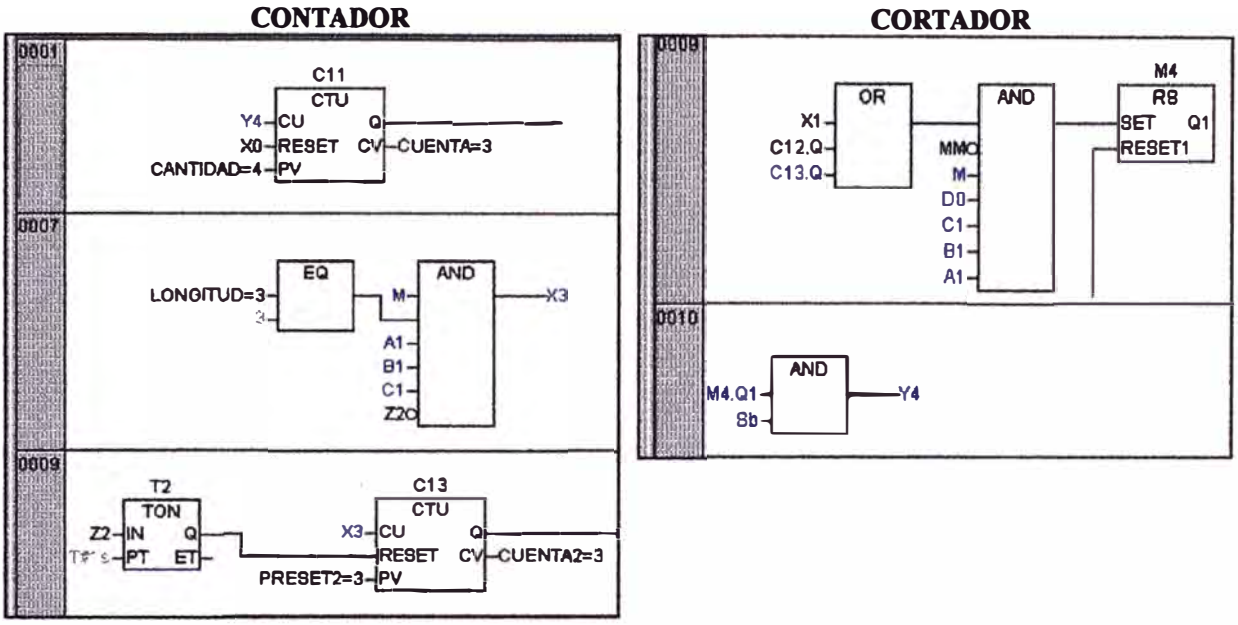

Fig. 4.32 Esquema de simulación en modo texto – Paso 3c

Paso 4c.- La posición de la cortadora (d0=0, d1=1) es detectada por el sensor que resetea memoria M4, deshabilita la variable X3, habilita la variable Z2 que activa el temporizador T2 que resetea al contador C13, y mueve la cortadora en dirección D-. Ver fig. 4.33.

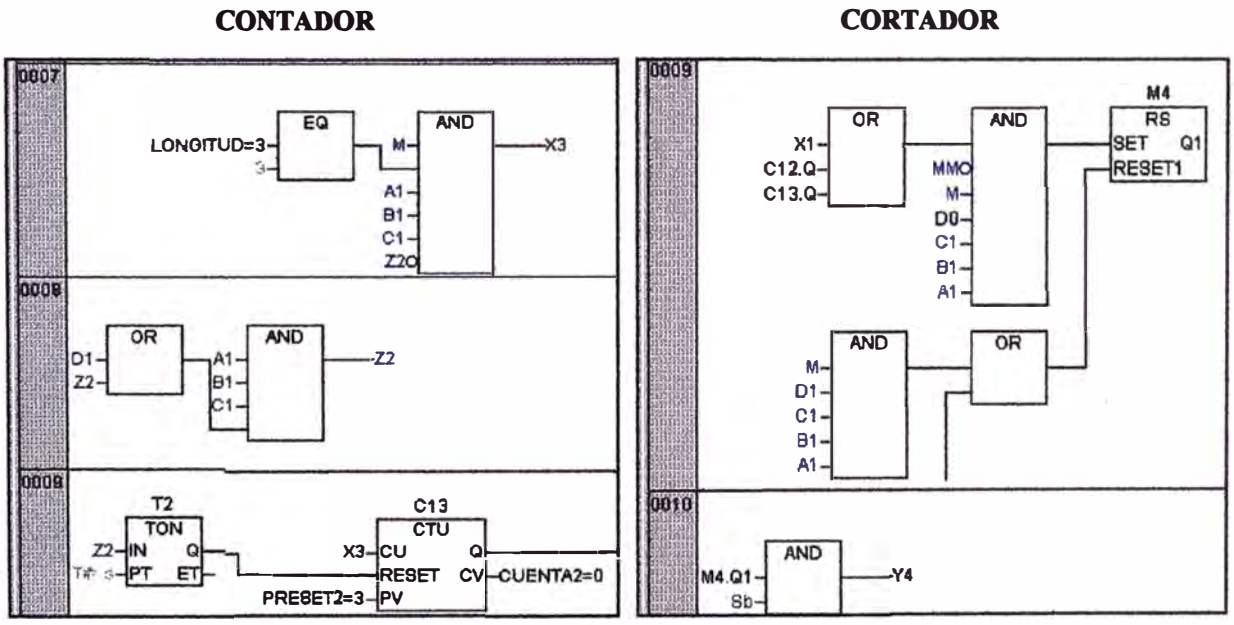

Fig. 4.33 Esquema de simulación en modo texto – Paso 4c

Paso 5c.- La posición de la cortadora (d0=1, d1=0) es detectada por el sensor de posición que resetea la memoria M3, deshabilitando la variable Y2 y moviendo el retenedor B en la dirección B-. La variable "MM" queda enclavada para evitar que la memoria M4 vuelva a ser seteada. El proceso se repite desde el "Paso 1c" hasta que el valor de la variable "CANTIDAD" sea igual al valor de la variable "CUENTA". Ver fig. 4.34.

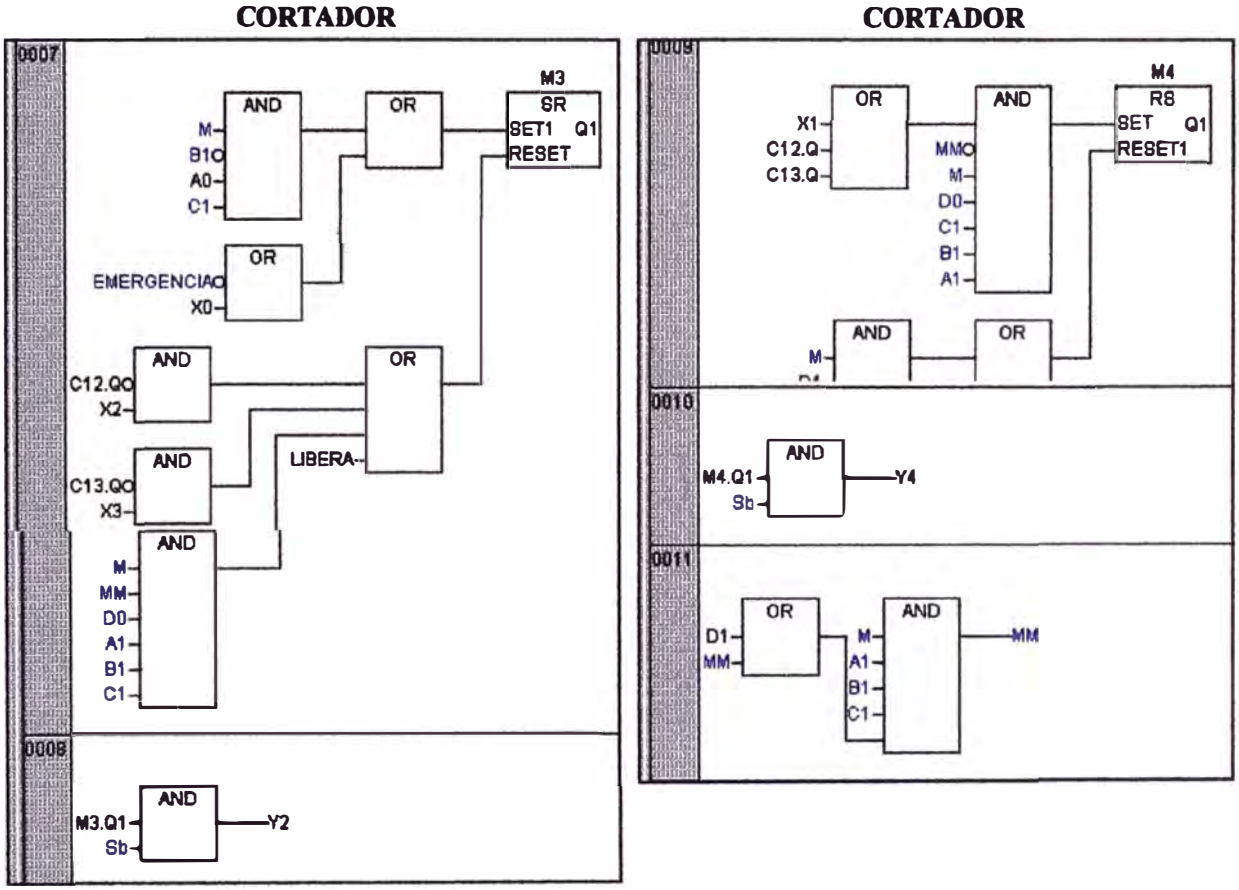

Fig. 4.34 Esquema de simulación en modo texto – Paso 5c

Finalmente cuando el valor de la variable "CANTIDAD" sea igual al valor de la variable "CUENTA" y se cumpla que a1=1; a0=0; b1=1; c1=1, d0=0; d1=1 la variable "X0" se habilita y se resetea el contador ascendente C11, el motor se detiene con la deshabilitación de la variable "M" y el proceso finaliza. Ver fig. 4.35.

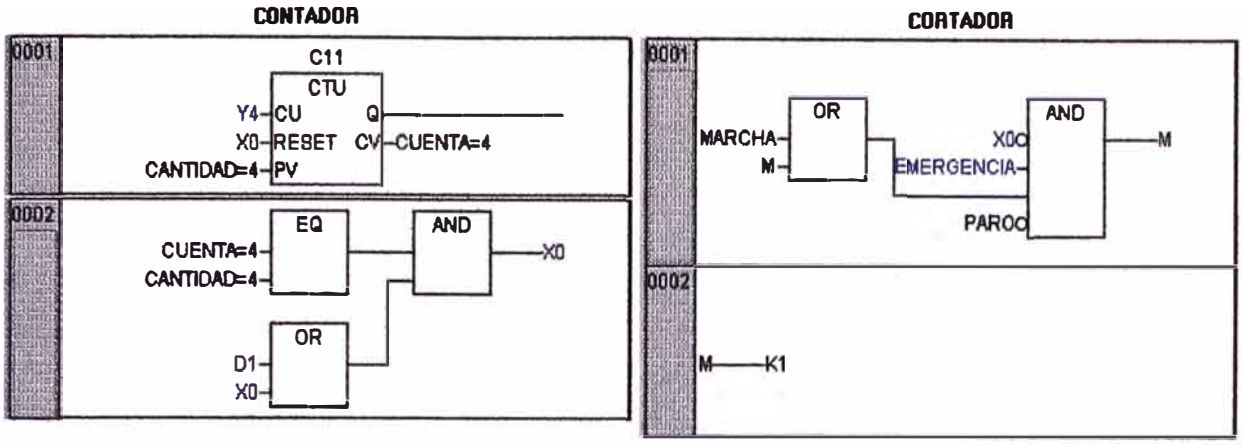

Fig. 4.35 Esquema de simulación en modo texto – Paso Final

4.4.4 Pulsadores de Seguridad "PARO", "EMERGENCIA" y "LIBERA"

- Paro: Variable cuyo valor inicial es un "0" lógico que sirve para colocar el proceso en "Stand by", el cual es reanudado nuevamente al presionar el pulsador de Marcha.

- Emergencia: Variable cuyo valor inicial es un "1" lógico (por cuestiones de fabricación) utilizada para detener el proceso ante un evento fortuito llevándolo a la posición del "Paso 7a", el cual es continuado al presionar el pulsador de marcha, perdiendo de esta manera una longitud L. Además las memorias "Set and Reset" (SR) han sido elegidas para darle prioridad a la señal del pulsador de Emergencia.

- Libera: Variable cuyo valor inicial es "0" lógico utilizado para evitar perder una longitud L luego de que se acciona de manera fortuita el pulsador de emergencia llevándolo a la posición del "Paso 2a".

4.5. SIMULACION USANDO EL ENTORNO GRAFICO DEL PROGRAMA "CORTADORA DE TUBOS DE DIFERENTES DIMENSIONES"

Paso 1.- Se escoge el valor de la variable "CANTIDAD", la variable "LONGITUD" se pone a 1 para cortes de longitud "L" y se presiona el pulsador de marcha, (A1, D0 y Sb habilitadas). Ver fig. 4.36.

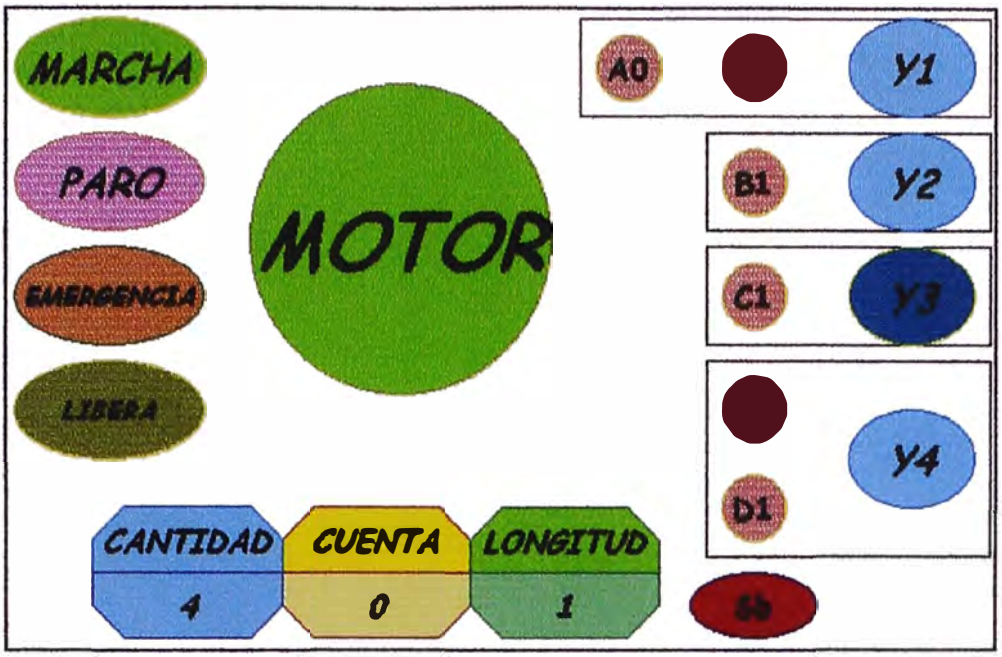

Fig. 4.36 Esquema de simulación en entorno gráfico – Paso 1

Paso 2.- Debido a la habilitación de la variable "Y3" entonces el valor de C1 cambia del estado C1=0 a C1=1. Ver fig. 4.37.

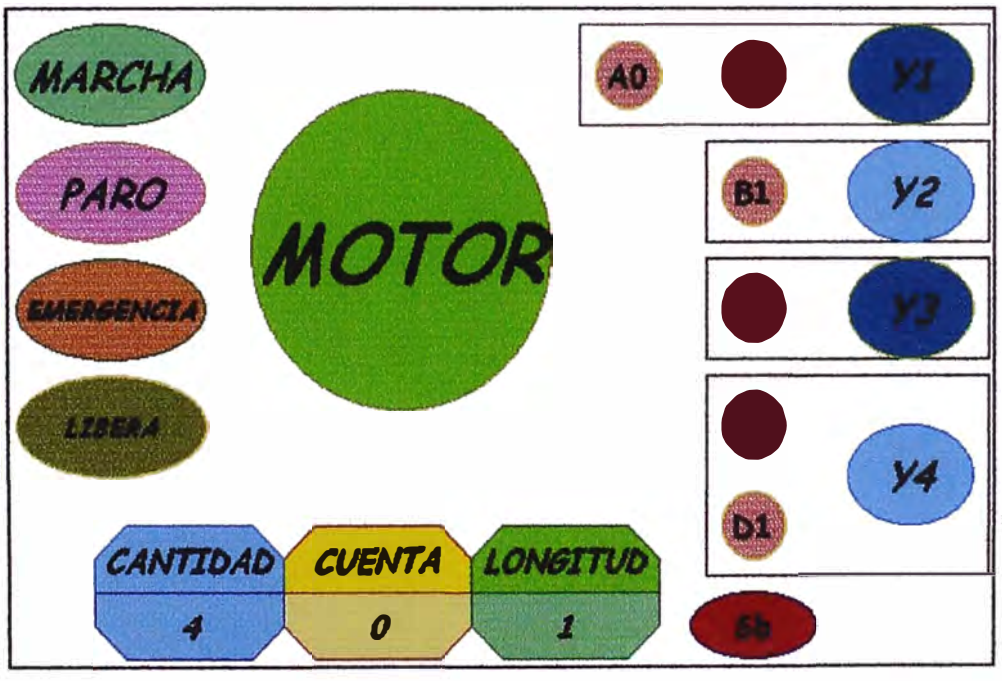

Fig. 4.37 Esquema de simulación en entorno gráfico – Paso 2

Paso 3.- Debido a la habilitación de la variable "Y1" entonces el valor de A1 cambia del estado A1=1 a A1=0 y el valor de A0 cambia del estado A0=0 a A0=1. Ver fig. 4.38.

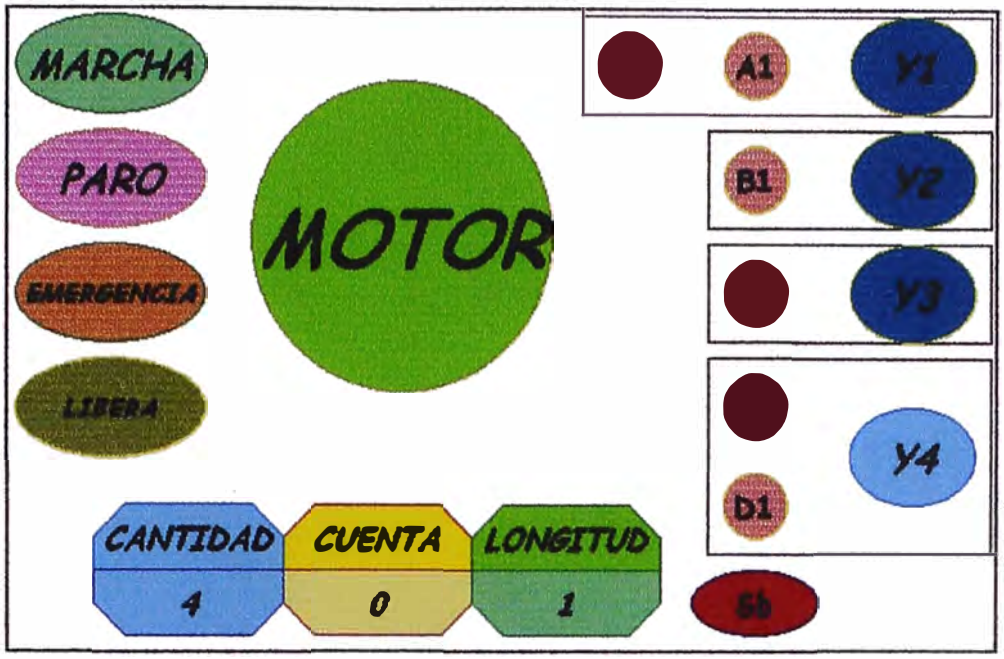

Fig. 4.38 Esquema de simulación en entorno gráfico – Paso 3

Paso 4.- Debido a la habilitación de la variable "Y2" entonces el valor de B1 cambia del estado B1=0 a B1=1. Ver fig. 4.39.

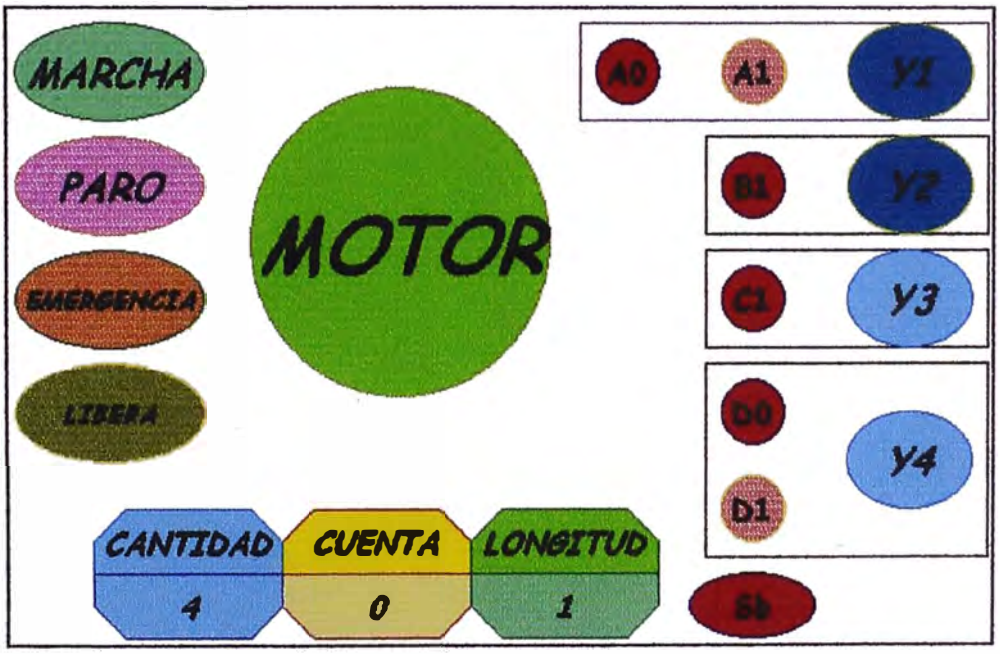

Fig. 4.39 Esquema de simulación en entorno gráfico – Paso 4

Paso 5.- Debido a la deshabilitación de la variable "Y3" entonces el valor de C1 cambia del estado C1=1 a C1=0. Ver fig. 4.40.

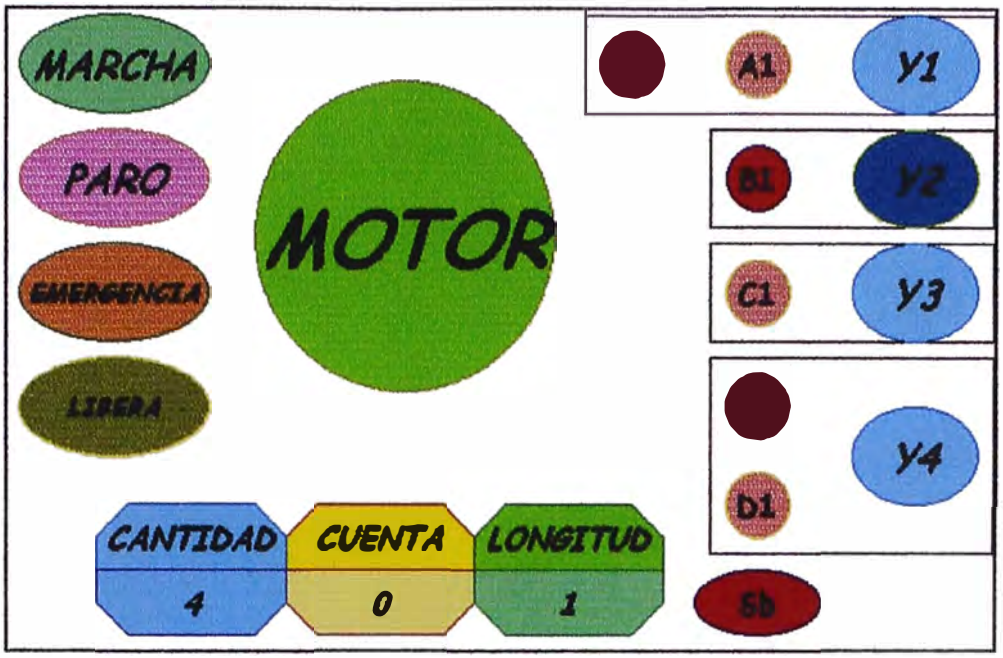

Fig. 4.40 Esquema de simulación en entorno gráfico – Paso 5

Paso 6.- Debido a la deshabilitación de la variable "Y1" entonces el valor de A0 cambia del estado A0=1 a A0=0 y el valor de A1 cambia del estado A1=0 a A1=1. Ver fig. 4.41.

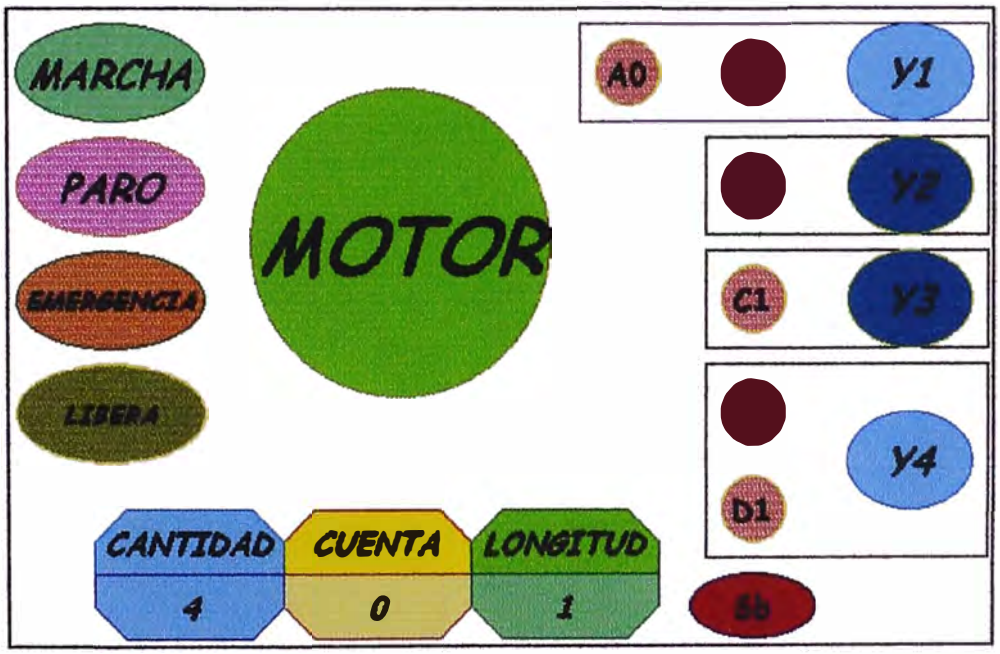

Fig. 4.41 Esquema de simulación en entorno gráfico – Paso 6

Paso 7.- Debido a la habilitación de la variable "Y3" entonces el valor de C1 cambia del estado C1=0 a C1=1. Ver fig. 4.42.

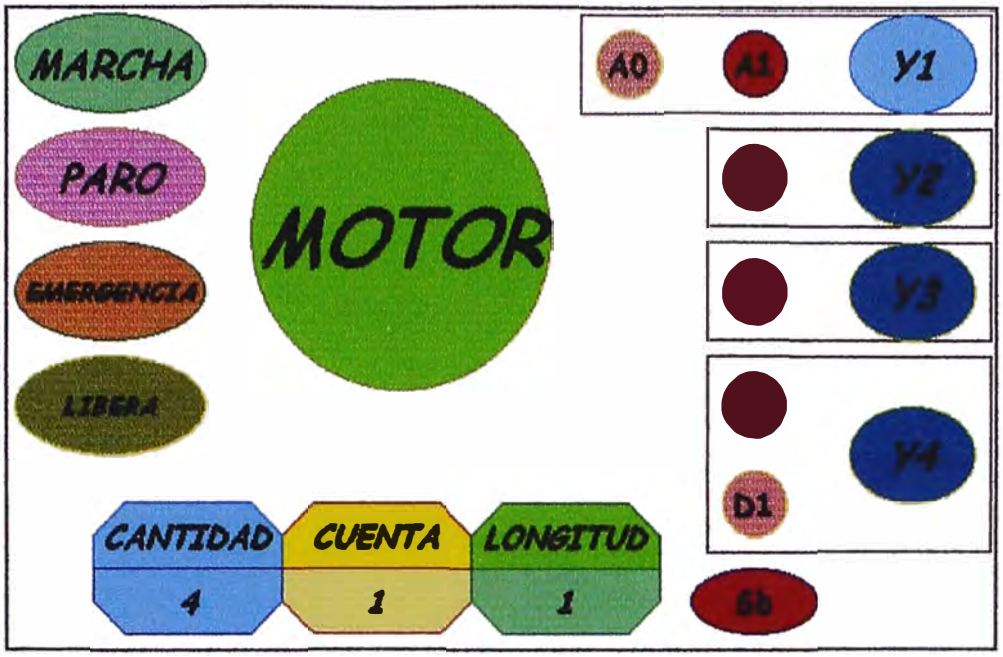

Fig. 4.42 Esquema de simulación en entorno gráfico – Paso 7

Paso 8.- La variable "CUENTA" toma el valor de 1. Debido a la habilitación de la variable Y4 el valor de D0=1 cambia a D0=0 y el valor de D1=0 cambia a D1=1. Ver fig. 4.43.

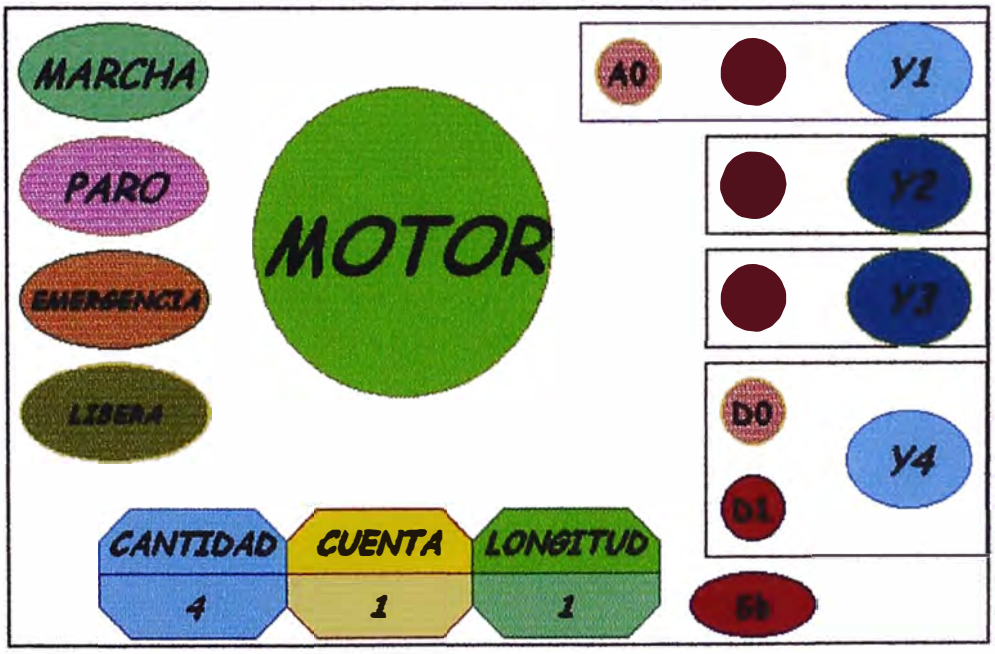

Fig. 4.43 Esquema de simulación en entorno gráfico – Paso 8

Paso 9.- Debido a la deshabilitación de la variable "Y4" entonces el valor de D1 cambia del estado D1=1 a D1=0 y el valor de D0 cambia del estado D0=0 a D0=1. Ver fig. 4.44.

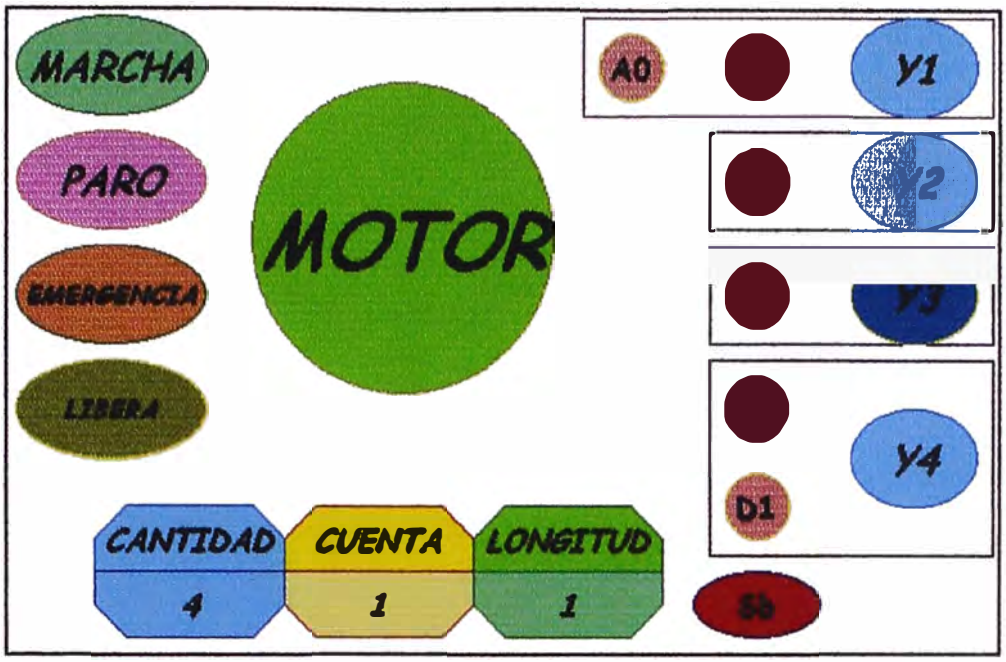

Fig. 4.44 Esquema de simulación en entorno gráfico – Paso 9

Paso 10.- Se cambia el valor de la variable "LONGITUD" a "2" para cortes "2L". La deshabilitación de la variable Y2 hace que B1=0 saltando del Paso 3 al Paso 6. Ver fig. 4.45.

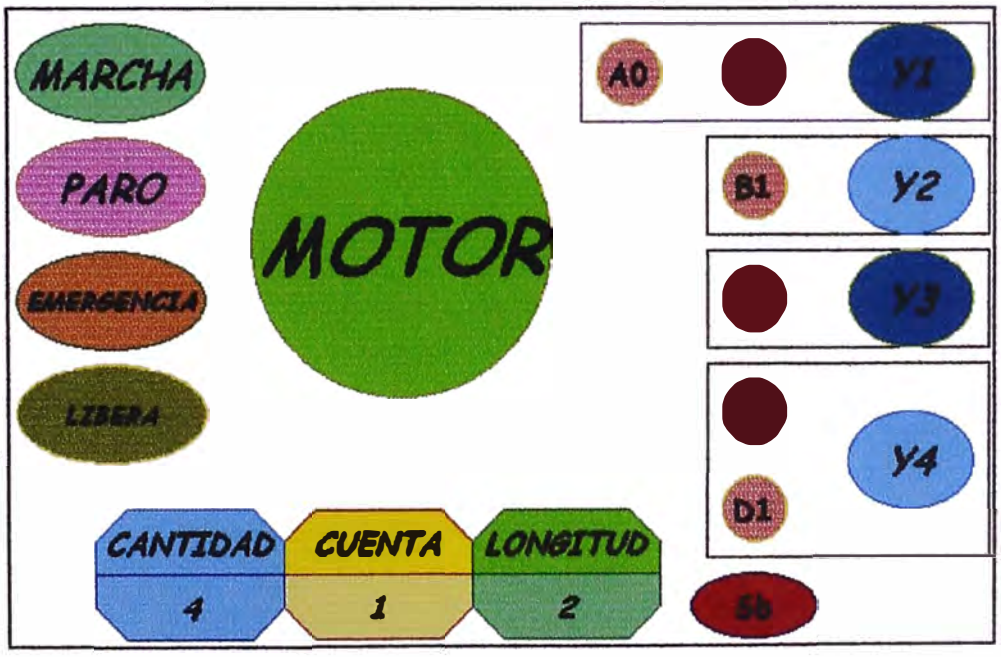

Fig. 4.45 Esquema de simulación en entorno gráfico – Paso 10

Paso 11.- Debido a la habilitación de la variable "Y3" entonces el valor de C1 cambia del estado C1=0 a C1=1. Saltando al Paso 10. Ver fig. 4.46.

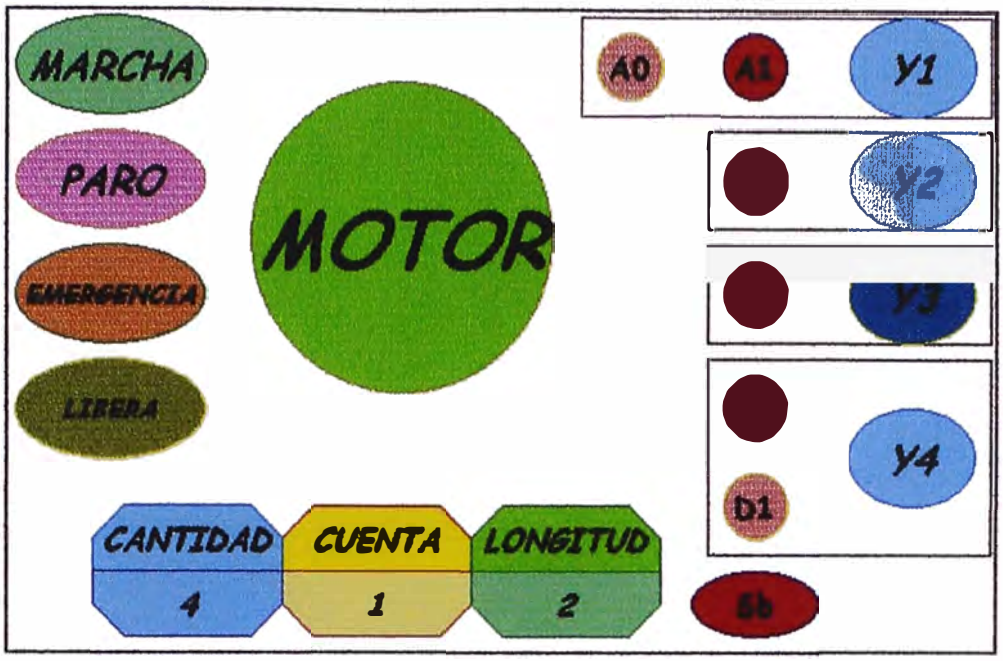

Fig. 4.46 Esquema de simulación en entorno gráfico – Paso 11

Paso 12.- Debido a la habilitación de la variable "Y3" entonces el valor de C1 cambia del estado C1=0 a C1=1. Ver fig. 4.47.

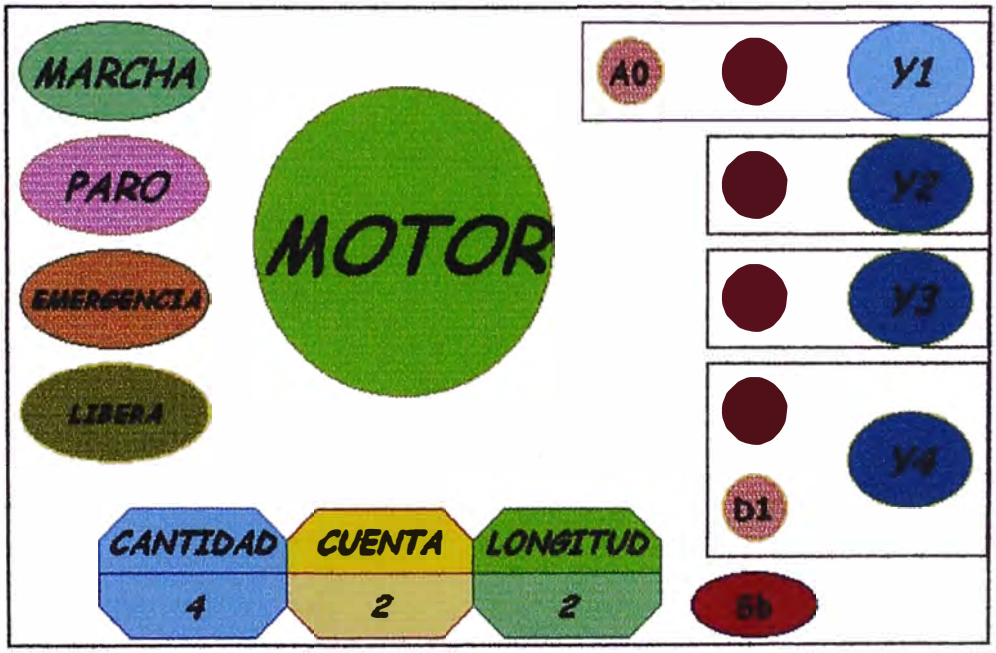

Fig. 4.47 Esquema de simulación en entorno gráfico – Paso 12

Paso 13.- La variable "CUENTA" toma el valor de 2. Debido a la habilitación de la variable Y4 el valor de D0=1 cambia a D0=0 y el valor de D1=0 cambia a D1=1. Ver fig. 4.48.

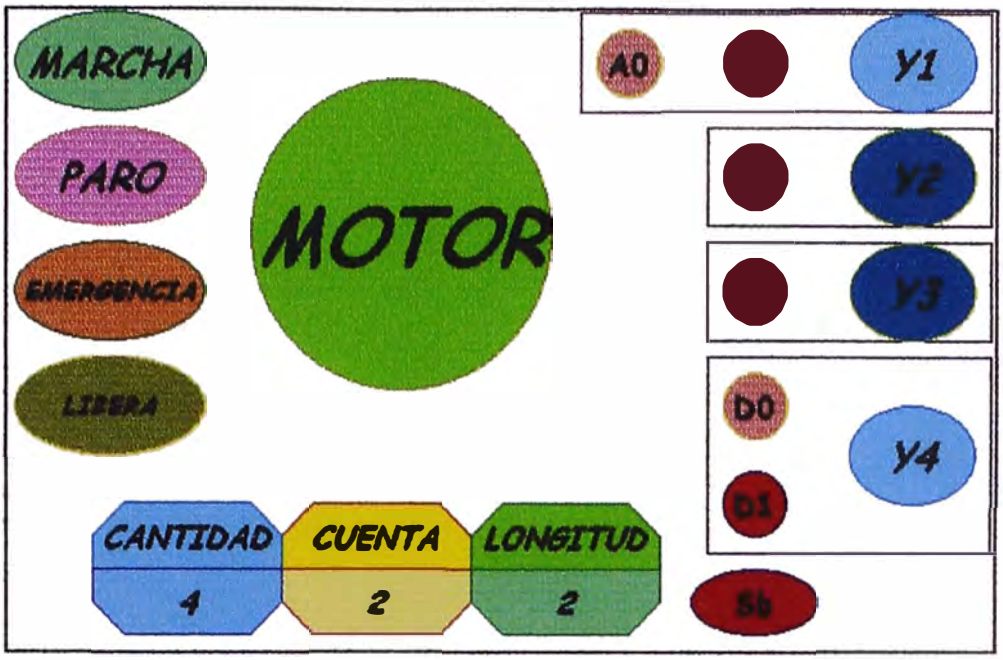

Fig. 4.48 Esquema de simulación en entorno gráfico – Paso 13

Paso 14.- Debido a la deshabilitación de la variable "Y4" entonces el valor de D1 cambia del estado D1=1 a D1=0 y el valor de D0 cambia del estado D0=0 a D0=1. Ver fig. 4.49.

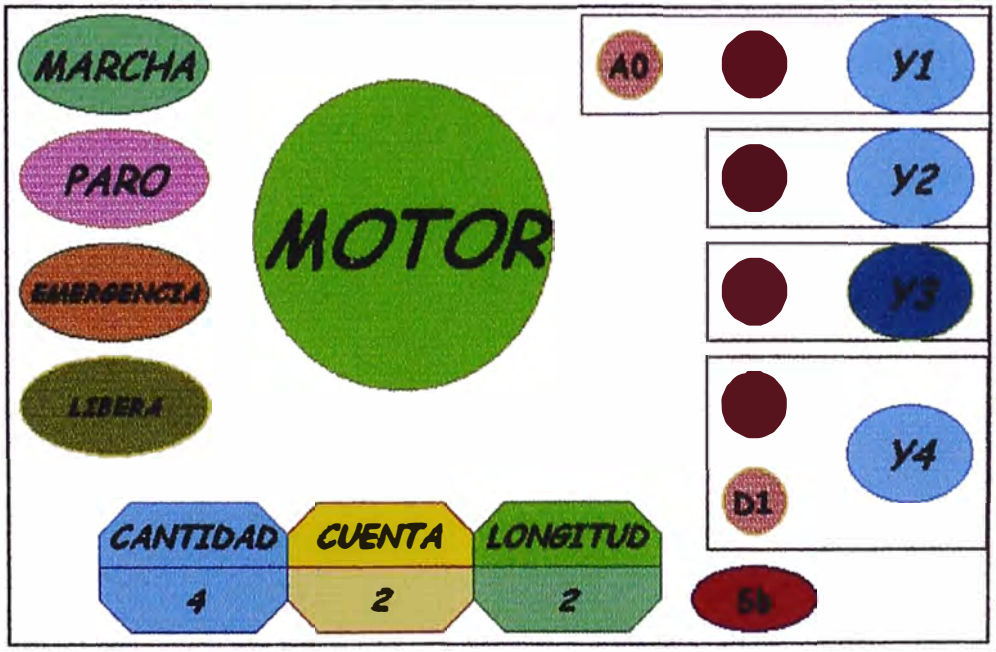

Fig. 4.49 Esquema de simulación en entorno gráfico – Paso 14

Paso 15.- Se cambia el valor de la variable "LONGITUD" a "3" para cortes "3L". La deshabilitación de la variable Y2 hace que B1=0 saltando del Paso 3 al Paso 6. Ver fig. 4.50.

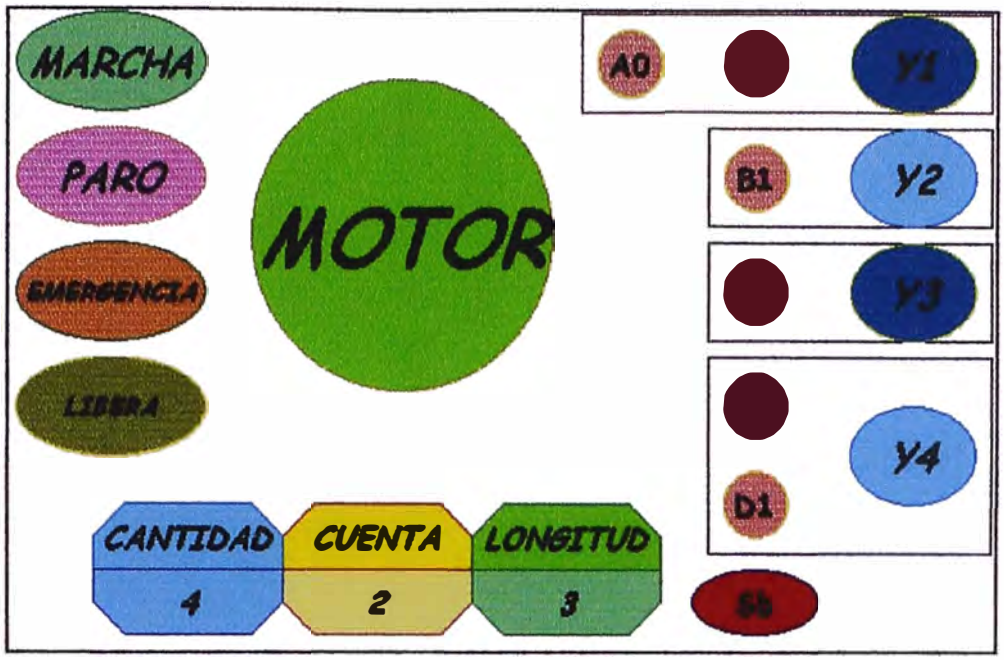

Fig. 4.50 Esquema de simulación en entorno gráfico – Paso 15

Paso 16.- Debido a la habilitación de la variable "Y3" entonces el valor de C1 cambia del estado C1=0 a C1=1. Saltando al Paso 15. Ver fig. 4.51.

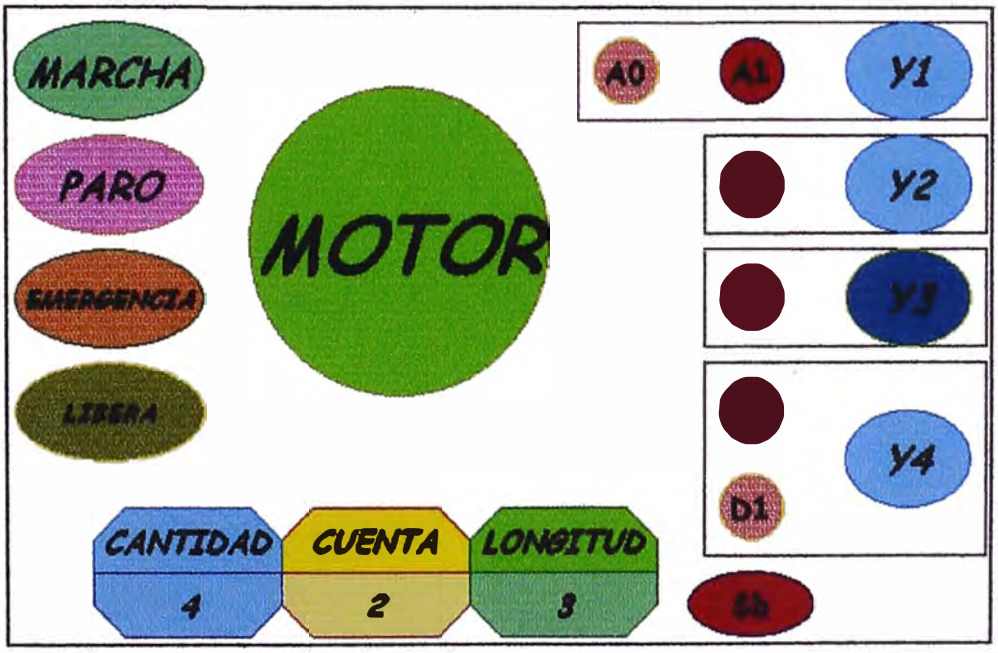

Fig. 4.51 Esquema de simulación en entorno gráfico – Paso 16

Paso 17.- Debido a la habilitación de la variable "Y3" entonces el valor de C1 cambia del estado C1=0 a C1=1. Saltando al Paso 15. Ver fig. 4.52.

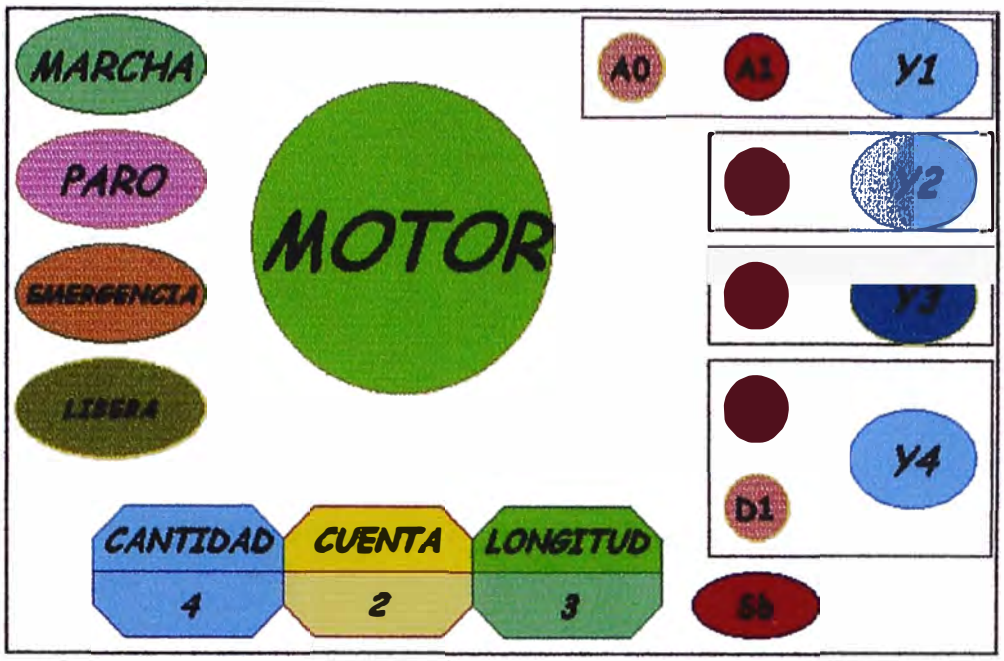

Fig. 4.52 Esquema de simulación en entorno gráfico – Paso 17

Paso 18.- Debido a la habilitación de la variable "Y3" entonces el valor de C1 cambia del estado C1=0 a C1=1. Ver fig. 4.53.

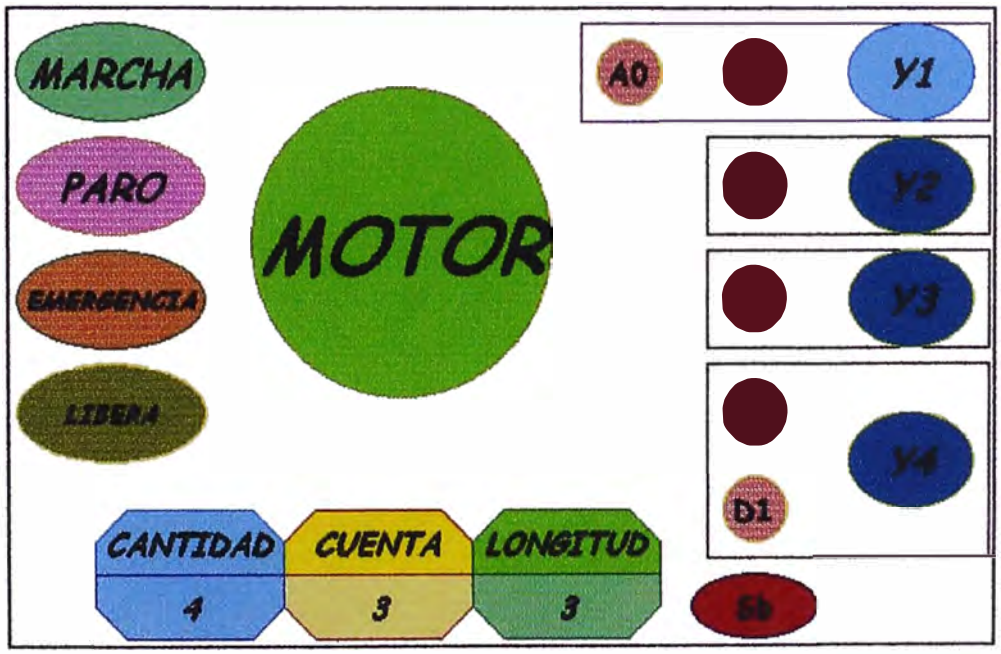

Fig. 4.53 Esquema de simulación en entorno gráfico – Paso 18

Paso 19.- La variable "CUENTA" toma el valor de 3. Debido a la habilitación de la variable Y4 el valor de D0=1 cambia a D0=0 y el valor de D1=0 cambia a D1=1. Ver fig. 4.54.

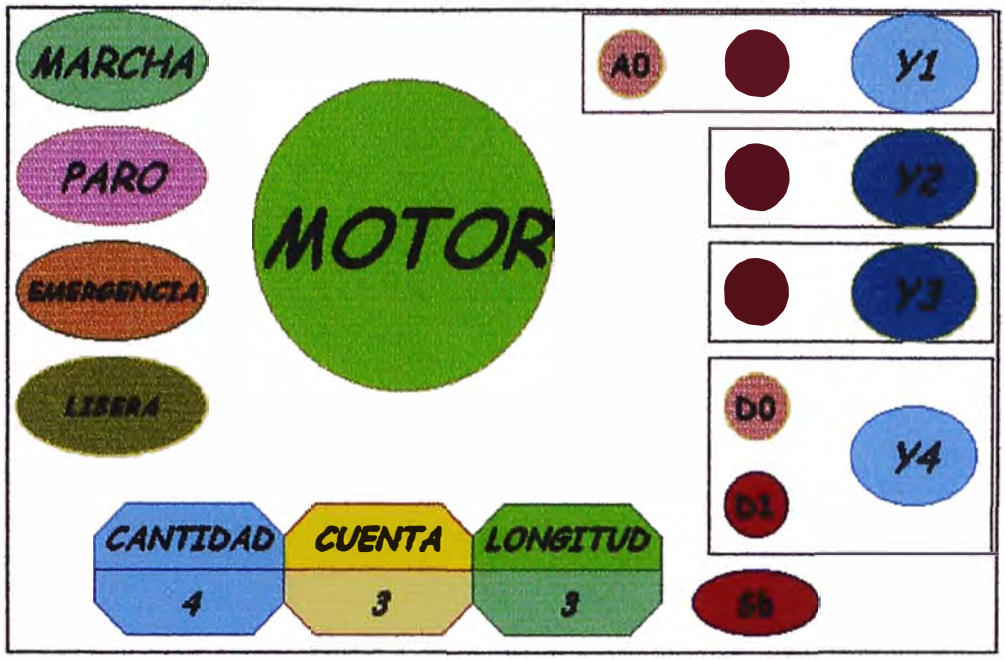

Fig. 4.54 Esquema de simulación en entorno gráfico – Paso 19

Paso 20.- Debido a la deshabilitación de la variable "Y4" entonces el valor de D1 cambia del estado D1=1 a D1=0 y el valor de D0 cambia del estado D0=0 a D0=1. Ver fig. 4.55.

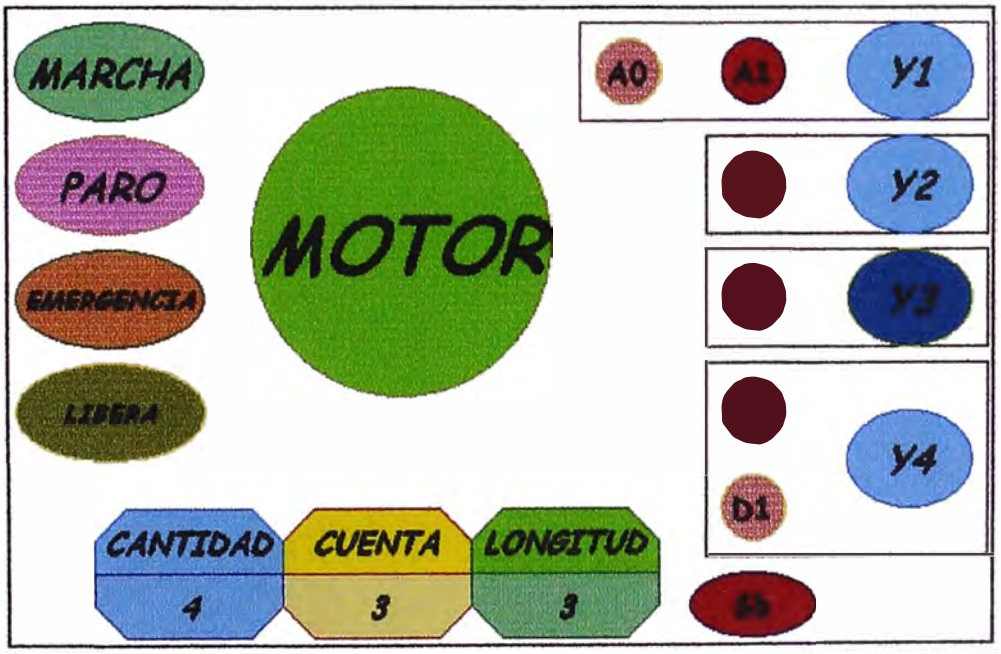

Fig. 4.55 Esquema de simulación en entorno gráfico – Paso 20

Paso 21.- Se cambia el valor de la variable "LONGITUD" a 1, 2 o 3 si se desea cortes L, 2L o 3L. La deshabilitación de la variable Y2 hace que B1=0 saltando del Paso 3 al Paso 6 y continuando según se requieran cortes de longitud L, 2L o 3L. Ver fig. 4.56.

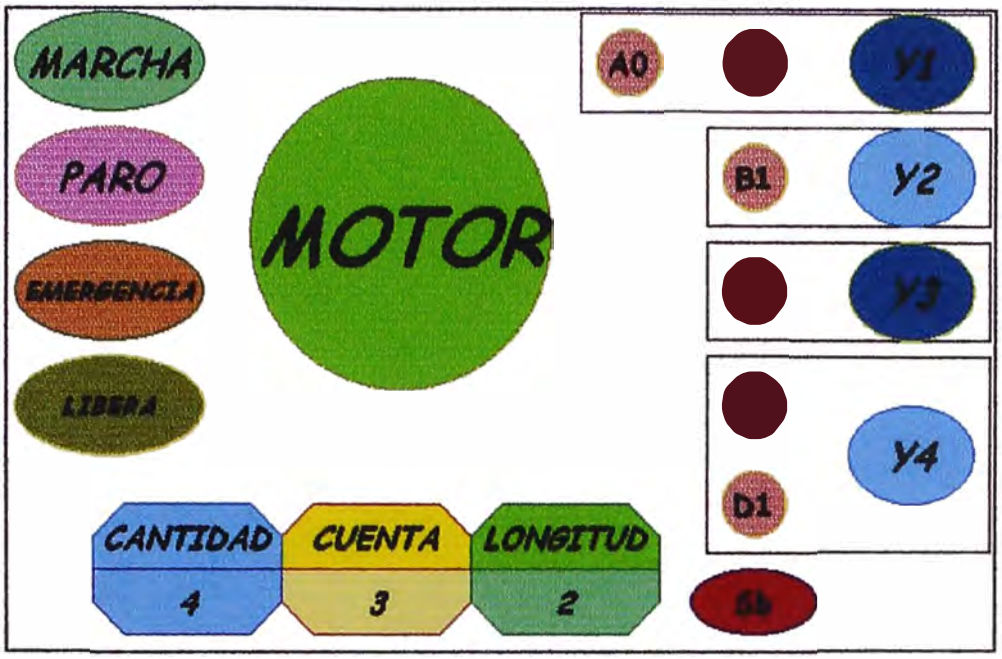

Fig. 4.56 Esquema de simulación en entorno gráfico – Paso 21

Caso A (Paso 1).- Puede darse el caso que durante la ejecución del proceso se presione el pulsador PARO entonces el proceso es detenido o colocado en "Stand by". Ver fig. 4.57.

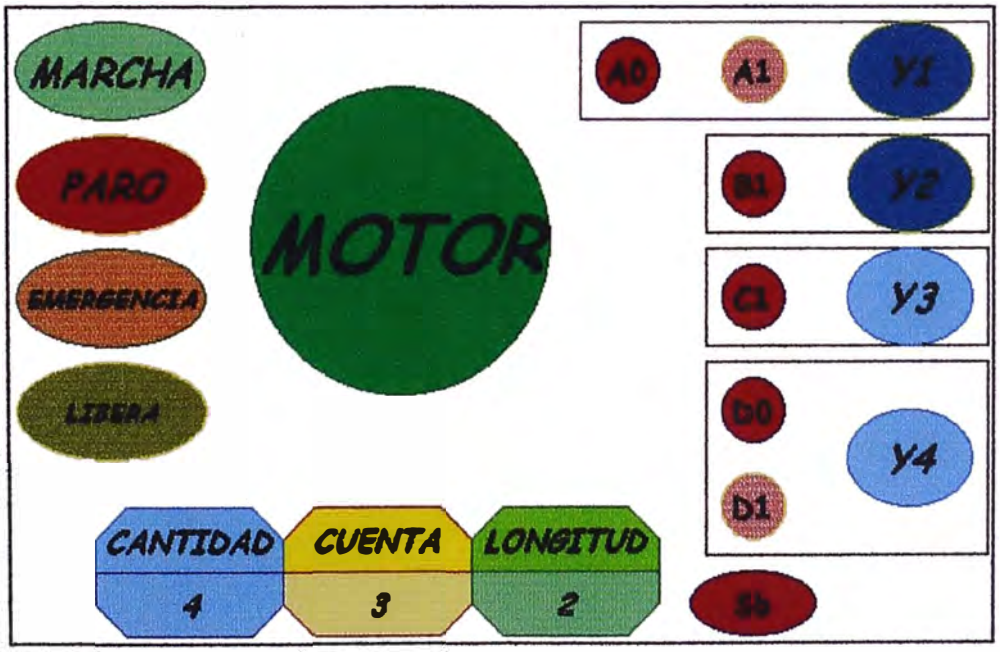

Fig. 4.57 Esquema de simulación en entorno gráfico – Caso A – Paso 1

Caso A (Paso 2).- Para continuar con el proceso de cortado en la posición interrumpida es necesario presionar el pulsador MARCHA que anula la orden enviada por el pulsador PARO. Ver fig. 4.58.

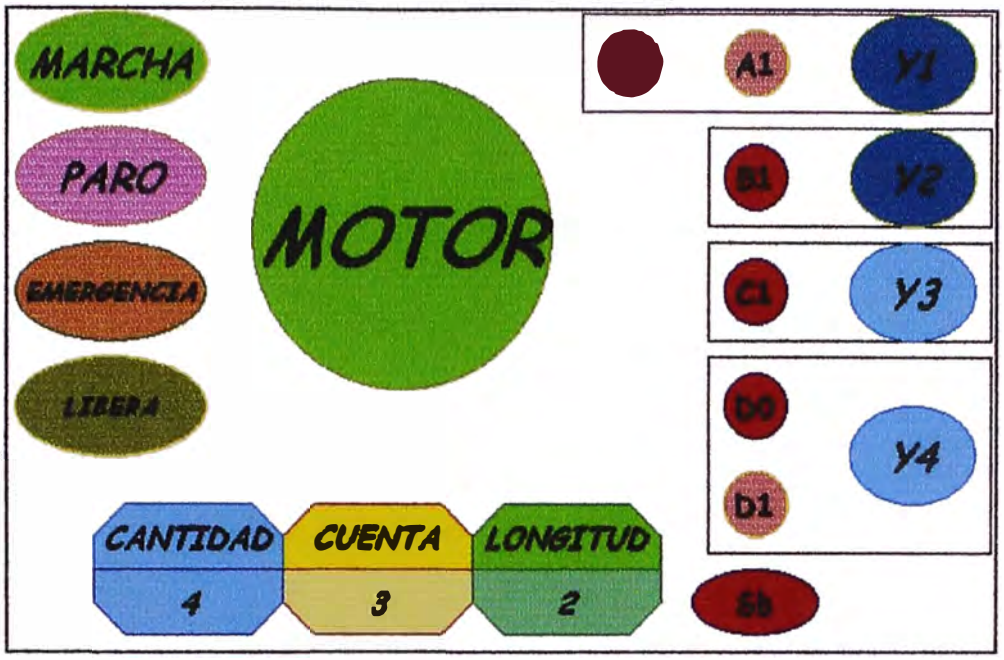

Fig. 4.58 Esquema de simulación en entorno gráfico – Caso A – Paso 2

Caso B (Paso 1).- Puede darse el caso que durante la ejecución del proceso se presione el pulsador EMERGENCIA, así las secuencias para el corte son anuladas. Ver fig 4.59.

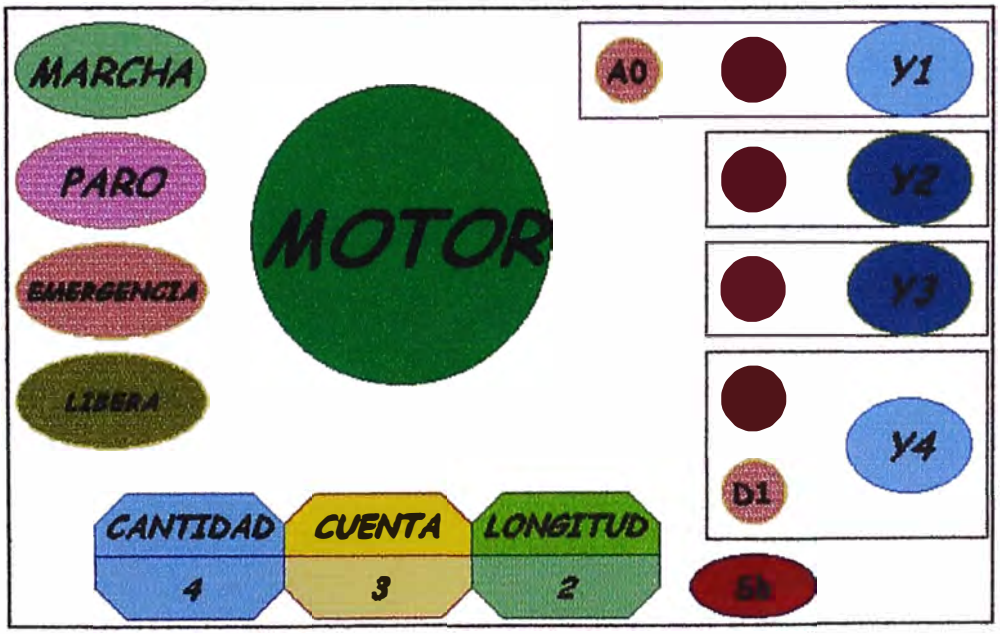

Fig. 4.59 Esquema de simulación en entorno gráfico – Caso B - Paso 1

Caso B (Paso 2).- Para continuar con el proceso de cortado desde la secuencia inicial del corte elegido es necesario presionar previamente el pulsador LIBERA. Ver fig. 4.60.

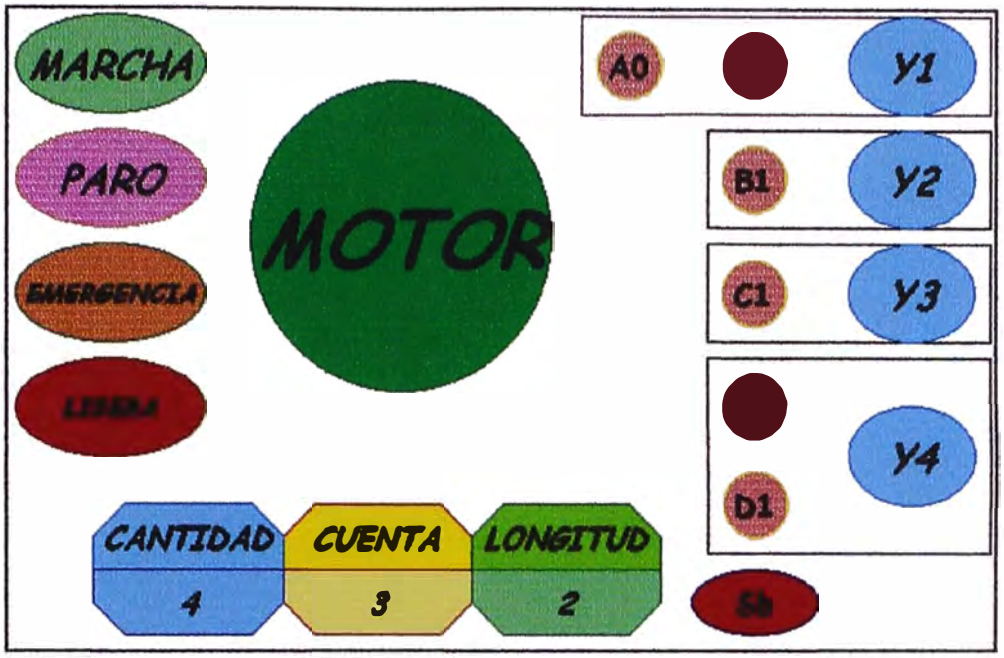

Fig. 4.60 Esquema de simulación en entorno gráfico – Caso B - Paso 2

Caso B (Paso 3).- Para iniciar el proceso de cortado desde la secuencia inicial es necesario presionar el pulsador MARCHA dado que las condiciones de posición ya fueron definidas. Ver fig. 4.61.

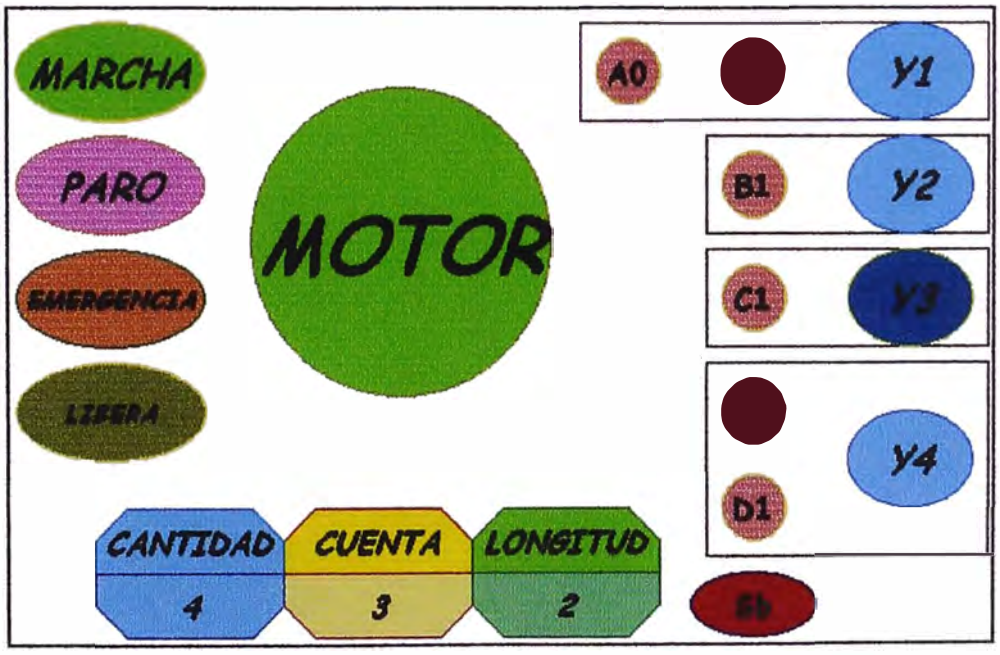

Fig. 4.61 Esquema de simulación en entorno gráfico – Caso B - Paso 3

CAPÍTULO V
ANALISIS ECONOMICO

5.1. FUNDAMENTO TEORICO

Se establece que un Proyecto es el proceso de búsqueda y hallazgo de una solución inteligente al planteamiento de un problema determinado, con la intención de resolver una de muchas necesidades humanas, es indispensable entender que tal acción debe tomarse con una base de decisión que justifique la aplicabilidad del proyecto, dado que la limitación de los recursos disponibles obliga a destinarlos conforme a su mejor aprovechamiento.

Tal aplicabilidad o viabilidad del proyecto obedece a estimar las ventajas y desventajas de asignar recursos a su realización, asegurando así la mayor productividad de los recursos. La evaluación de los proyectos desde el punto de vista económico se puede estructurar de la siguiente forma:

- **Análisis técnico,** el cual debe establecer la factibilidad técnica y operacional del proyecto.
- **Análisis económico,** el que determina la conveniencia económica o la rentabilidad del proyecto.

5.1.1 El Estudio Económico:

Un estudio económico corresponde a la valoración (expresada en términos económicos o sociales) de las diferencias existentes entre las alternativas disponibles, con el único fin de

comparar sus ventajas económicas. Si existen consideraciones técnicas involucradas dicha comparación es un estudio de *ingeniería económica.*

5.1.2 La Ingeniería Económica:

Se define como el conjunto de conceptos y técnicas cuantitativas de análisis, útiles para la evaluación y comparación económica de alternativas relativas a sistemas, productos, servicios, recursos, inversiones y equipos, para lograr decisiones entre las que se seleccionen la mejor opción de entre las que se tienen disponibles.

5.1.3 El Análisis Financiero:

Este análisis involucra el estudio de la disponibilidad, origen y uso que se dará a los recursos económicos necesarios para llevar a cabo un proyecto. Dicho estudio deberá considerar la fuente crediticia o financiera existente, los instrumentos financieros disponibles, los mecanismos de financiamiento, las condiciones de cada uno de ellos, que pueden ser muy diversos, y sobre todo los criterios establecidos para su otorgamiento por parte de las entidades financieras, sin olvidar los puntos exigidos para acceder a los mismos. La complejidad involucrada en la elaboración de este tipo de análisis ha dado lugar al desarrollo de la *ingeniería financiera.*

5.1.4 La Ingeniería Financiera:

Son el conjunto de principios, conceptos y técnicas cuantitativas de análisis, útiles para la evaluación, comparación económica y selección de alternativas, con relación a fuentes, instrumentos, mecanismos, criterios y condiciones para el otorgamiento y disposición de recursos económicos que financien proyectos, tanto de inversión como de desarrollo, en las condiciones más ventajosas para el financiador y/o para el financiado.

5.2. Evaluación de los Sistemas de Automatización del Proyecto

5.2.1 Sistema de Controlador Automático

El Controlador Lógico Programable (PLC) que se va a utilizar para el presente proyecto es de la marca Siemens, modelo Simatic S7 200, elegido por su facilidad de poder conectar las entradas y salidas digitales; y su fácil interpretación con la lógica computacional elaborada con el software de programación Prosys. Ver Fig. 5.1

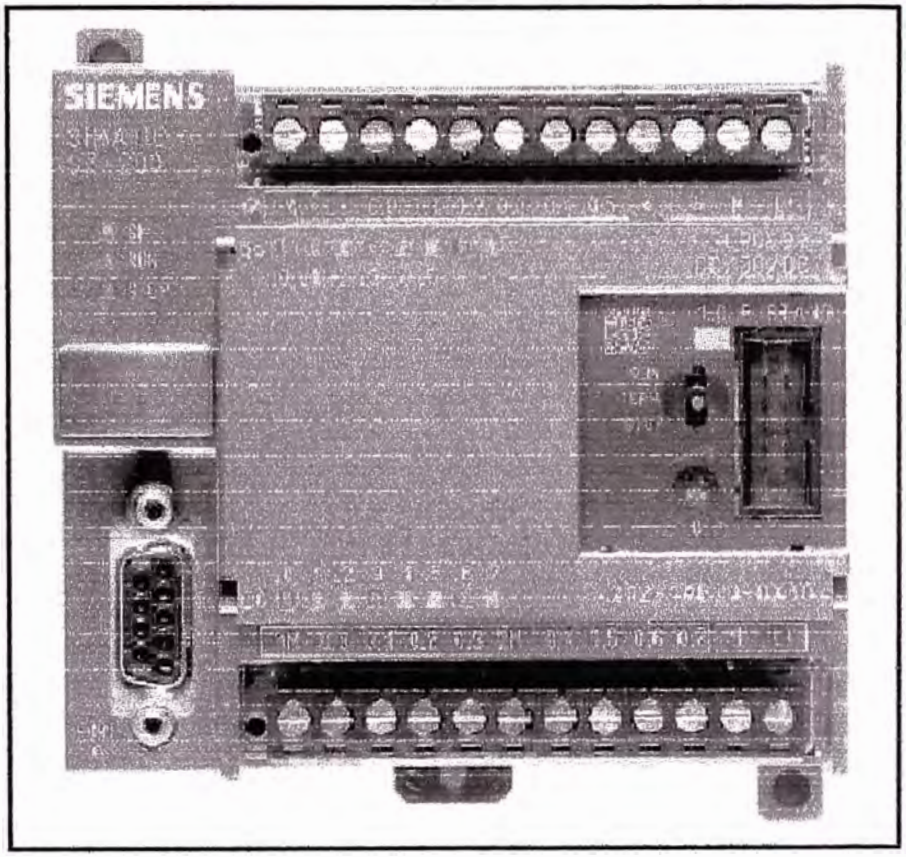

Fig. 5.1 PLC Siemens Simatic S7-200

Las características técnicas del tipo de PLC elegido para la presente aplicación las detallamos a continuación (Ver Tabla N° 5.1):

Tabla N° 5.1 Características técnicas del PLC

Marca	Siemens
Modelo	Modelo Simatic S7-200
Datos Técnicos de la CPU:	
Tipo de CPU	CPU 224 DC/DC/DC

Alimentación Nominal	24VCC
Cantidad de entradas digitales	14 x 24 VCC
Cantidad de salidas digitales	10 x 24 VCC
Dimensiones en mm	120.5 x 80 x 62
Peso	360 gramos
Disipación	7 watt
Corriente continua disponible	280 mA
Tamaño de la memoria EEPROM para el programa de usuario	12288 bytes
Tamaño de la memoria para datos de usuario	8192 bytes
Cantidad de Contadores	256
Cantidad de Temporizadores	256 (4 de 1ms; 16 de 10ms; 236 de 100ms)
Cantidad de marcas internas	256 (respaldado por pila) y 112 (en EEPROM)
Datos Técnicos de la Alimentación:	
Tensión de entrada	20.4 a 28.8 VCC
Intensidad de entrada	110 mA
Corriente de irrupción	12 A a 28.8 VCC
Tiempo retardo (desde pérdida de corriente)	10 ms a 24 VCC
Fusible (no reemplazable)	3 A, 250 V, de acción lenta
Tensión de sensores (potencia limitada)	L+ menos 5V
Intensidad límite	1.5 A pico, límite térmico no destructivo
Datos Técnicos de Entradas Digitales de la CPU:	
Tipo de datos	Sumidero de corriente tipo 1 IEC
Tensión nominal	24 VCC a 4 mA
Tensión máxima admisible	30 VCC
Sobretensión	35 VCC ; 0.5 s
Señal 1 lógica (mín)	15 VCC a 2.5 mA
Señal 0 lógica (máx)	5 VCC a 1 mA
Retardo de entrada	0.2 a 12.8 ms
Corriente de fuga admisible en sensor	1 mA (para sensor de proximidad)
Longitud de cable apantallado (máx)	500 m
Longitud de cable no apantallado (máx)	300 m
Datos Técnicos de Salidas Digitales de la CPU:	
Tipo de datos	Estado Sólido MOSFET
Tensión nominal	24 VCC

Rango de tensión	20.4 a 28.8 VCC
Sobreintensidad (máx)	8 A, 100 ms
Señal 1 lógica (mín)	20 VCC a intensidad máxima
Señal 0 lógica (máx)	0.1 VCC con 10KΩ de carga
Intensidad nominal por salida (máx)	0.75 A
Intensidad nominal por neutro (máx)	6 A
Corriente de fuga (máx)	10 uA
Carga de lámparas (máx)	5 W
Tensión de bloqueo inductiva	L+ menos 48 VCC, disipación de 1 W
Separación galvánica	500 VCA, 1 minuto
Retardo máx OFF a ON	2 us (Q0.0; Q0.1); 15 us (todas las demás)
Retardo máx ON a OFF	10 us (Q0.0; Q0.1); 130 us (todas las demás)
Longitud de cable apantallado (máx)	500 m
Longitud de cable no apantallado (máx)	150 m
Datos Técnicos de la Comunicación Integrada:	
Puertos de comunicación	1 puerto RS-485
Velocidades de transferencia PPI, DP/T	9.6; 19.2 y 187.5 kbps
Velocidades de transferencia Freeport	1.2 kbps a 115.2 kbps
Longitud máxima del cable por segmento	Con repetidor aislado: 1000m (187.5kbps) Sin repetidor aislado: 50m
Número máximo de estaciones	32 por segmento; 126 por red
Número máximo de maestros	32
Enlaces MPI	2 reservados; 1 para una PG; 1 para un OP

5.2.2 Fuente de Alimentación externa

Se realizará un cálculo de la cantidad de corriente que se debe suministrar al PLC S7-200 con la configuración deseada para determinar el tipo de fuente de alimentación externa requerida (ver Tabla N° 5.2), además el diagrama de cableado de las entradas y salidas del PLC tiene que ver con el consumo de corriente (ver Fig. 5.2).

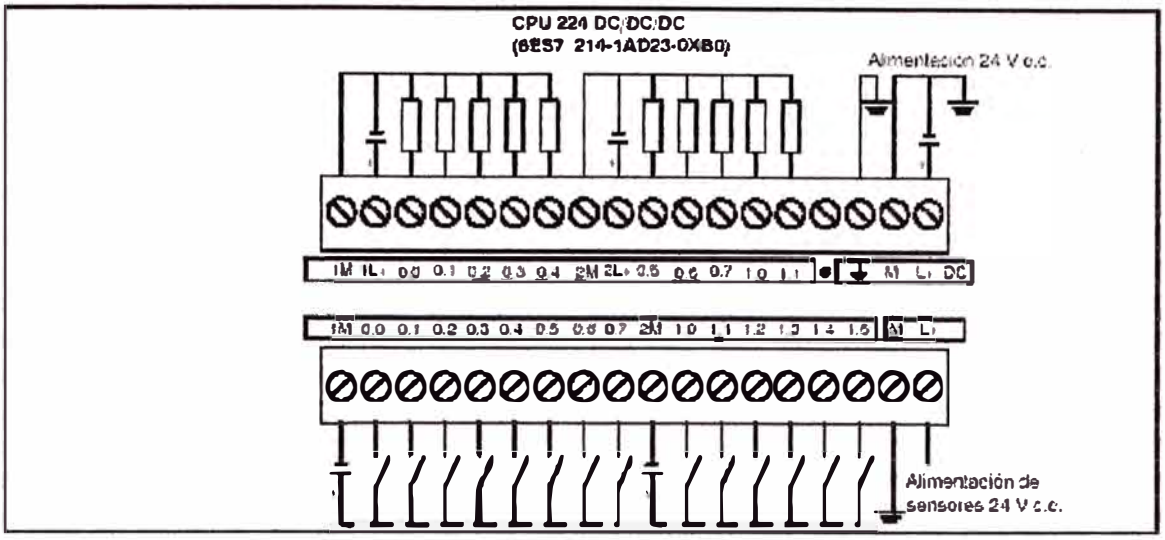

Fig. 5.2 Diagrama de Cableado para la CPU 224

Tabla N° 5.2 Cantidad de corriente a consumir

Consumo de la CPU	24 VCC
CPU 224 DC/DC/DC	**280** mA
Consumo del Sistema	24 VCC
CPU 224 (11 entradas utilizadas)	11 * 250 mA = **2750** mA
CPU 224 (5 salidas utilizadas)	5 * 50 mA = **250** mA

Por los valores obtenidos se determina que la carga total consume un valor de corriente de 3280 mA, es decir, casi 4A. La fuente de alimentación externa elegida para tal fin será la que detallaremos a continuación (ver Tabla N° 5.3):

Tabla N° 5.3 Fuente de Alimentación externa

Marca	Siemens
Modelo	Logo Power
Tensión nominal de entrada	100-240 VAC
Rango de frecuencia	47-63 Hz
Tensión nominal de salida	24 VDC
Corriente nominal de salida	4 A
Limitación estática de corriente	4.7 A
Corriente de salida máxima	10 A
Temperatura para funcionamiento	-20°C a +55°C
Grado de Protección	IP 20

5.2.3 Sistema de Sensores para medición de cambios del proceso

Los sensores que se van a utilizar para el presente proyecto son del tipo "sensores inductivos de proximidad", elegidos por su facilidad de proveer señales de posición y límites, lo cual los hace indispensables para este tipo de aplicación. Su principio de funcionamiento se basa en que ejecutan una conmutación electrónica cuando un objeto metálico atraviesa un campo electromagnético de alta frecuencia, originado por un oscilador electrónico dirigido hacia fuera del sensor, formando una región con una sensibilidad determinada denominada distancia de conmutación, cuando el cuerpo metálico se encuentra dentro de esta región, genera una distorsión provocando un cambio en el estado lógico del sensor. Las características técnicas del tipo de sensor elegido para la presente aplicación las detallamos a continuación (Ver Tabla N° 5.4 y Fig. 5.3):

Tabla N° 5.4 Características técnicas del sensor utilizado

Marca	IFM Electronics
Modelo	Efector 100
Voltaje	10.....36VDC
Corriente	250mA
Distancia de sensado	2mm
Tipo de sensor	Conmutador de 3 cables positivo, negativo, normalmente abierto o normalmente cerrado
Distinciones notorias	• Adaptabilidad a cualquier proceso requerido • Soluciones bajo costo debido a versatilidad • Detección sin contacto de todos los metales • Elevadas frecuencias de conmutación • Exactitud y durabilidad en los ambientes más agresivos

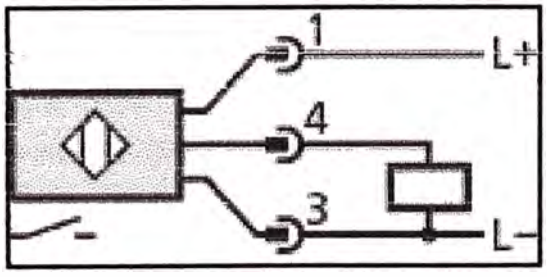

Fig. 5.3 Sensor inductivo de proximidad de 3 terminales

5.2.4 Sistema de Actuadores para variación del proceso

Los actuadores que se van a utilizar para el presente proyecto son del tipo "actuadores neumáticos con válvulas de solenoide operadas directamente", elegidos por su capacidad de conectar la neumática y la electrónica, por su uso en distribuidores neumáticos de menor caudal, por sus componentes pequeños con válvulas con solenoide con una gran relación de flujo y potencia para sus operaciones de ciclo cerrado. Su principio de funcionamiento se basa en que convierte una señal eléctrica procedente de un controlador en un movimiento de vástago-obturador (señal neumática) desde el cierre a la apertura o viceversa; denominado motor o servomotor se encarga de producir la fuerza necesaria para provocar un cambio en la apertura de la válvula.

Las características técnicas del tipo de actuador elegido para la presente aplicación las detallamos a continuación (Ver Tabla N° 5.5 y Fig. 5.4):

Tabla N° 5.5 Características técnicas del actuador utilizado

Marca	Pneumax
Modelo	Series 300 10mm – Tipo N3 8 1.1
Características neumáticas:	
Presión de Trabajo	De 0 a 7 Bar
Tamaño del orificio	0.7mm
Temperatura de Trabajo	De –5°C a 50°C
Caudal máximo a 6 Bar	14 Nl/min
Número de ciclos	2.7 ciclos / minuto
Tiempo de vida	50 millones de ciclos
Características Eléctricas:	
Voltaje	De 12 a 24 VDC
Potencia	1.3 Watt
Tolerancia de voltaje permitida	-5% a +10%
Tiempo de respuesta a energización	8.0 ms
Tiempo de respuesta a desenergización	10.0 ms
Grado de Protección	IP 40 – IP 65
Tipo de actuador	Dispositivo de 3/2 vías N.O. 24 VDC

Distinciones notorias	• Viabilidad y simplicidad de diseño • Usado donde no se requiere grandes fuerzas • Aplicable a válvulas con recorridos cortos • Económicos • Válvula de dimensiones mínimas • Válvula a alta velocidad de conmutación • Diseñado para ser montado en pequeños espacios •

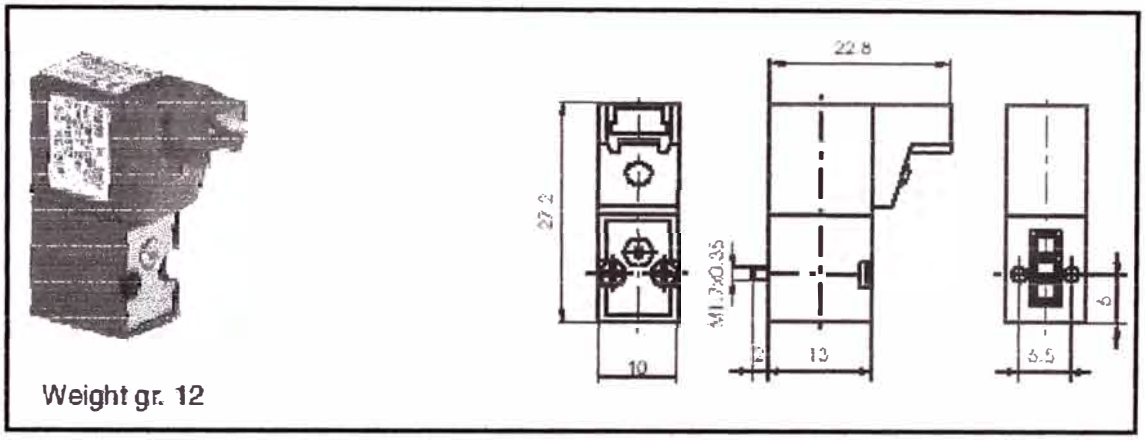

Fig. 5.4 Actuador de 3/2 vías N.O. conector a 90°

5.2.5 Sistema de optoacopladores

Los motivos fundamentales para la utilización de los optoacopladores se deben a la necesidad de adaptar la señal proveniente de los sensores y actuadores al nivel de la señal eléctrica del PLC; así como, el de proveerle aislamiento eléctrico. Ver Fig. 5.5

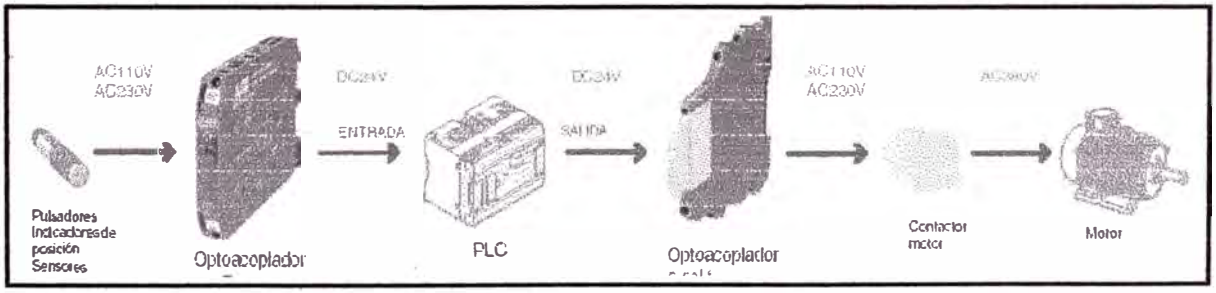

Fig. 5.5 Posicionamiento de los Optoacopladores en la lógica de control

Esta compuesto por un LED y un Fototransistor, sin existir conexión eléctrica entre el circuito de entrada y el de salida. Su principio de funcionamiento se basa en que la incidencia de la luz proveniente del LED (alimentado por una señal cuadrada) sobre la zona de la base del transistor (corriente de base) influye mucho en la corriente de colector (ver Fig. 5.6)

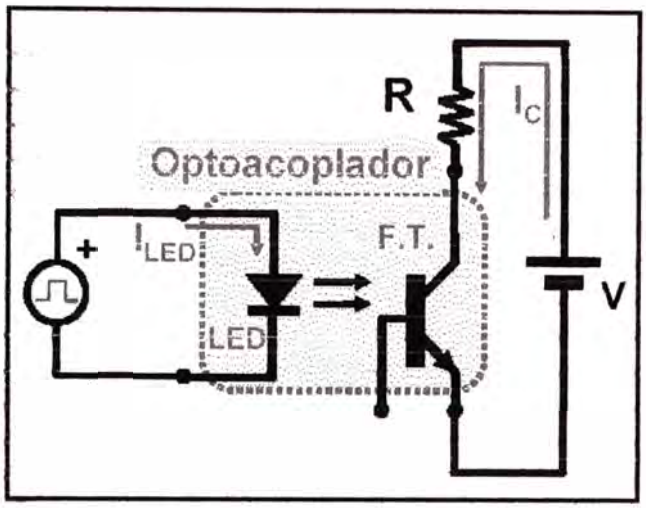

Fig. 5.6 Principio de funcionamiento de un Optoacoplador

Las características técnicas del tipo de optoacoplador elegido para la presente aplicación se muestran a continuación (ver Tabla N°5.6):

Tabla N° 5.6 Características técnicas del Optoacoplador

Marca	ABB
Modelo	Serie R500 – módulo de Relé enchufable
Características de la Bobina:	
Tensión	24 VDC (+20%, -15%)
Potencia	0. 3Watt
Intensidad Nominal	12 mA
Tensión de desenganche	2.4 Volt
Características del Contacto:	
Tipo	1 Contacto conmutado de 10mA a 6A / 250VA
Tensión de salida min /max	12 VAC / 250 VAC
Intensidad de corte min / max	10 mA / 6 A
Potencia de corte	AC min. /max: 0. 6/ 1500 VA (carga ohmica) DC min / max: 0. 6W / 140 W

Número de maniobras en carga	100000
Número de maniobras en vacío	10000000
Tiempo de cierre	5.0 ms
Tiempo de apertura	8.0 ms
Aislamiento bobina / contactos	4000 VRMS
Aislamiento contactos / contactos	1000 VRMS
Temperatura de almacenaje	-40°C a +80°C
Aislante	UL 94V0
Protección	IP20 NEMA 1

5.2.6 Motor para cortadora de Tubos

El motor eléctrico que sea capaz de poder vencer la inercia de la cortadora de metal, en forma de disco, encargada de realizar los cortes de los tubos en las dimensiones definidas por el programa del PLC, será elegido según los criterios siguientes:

- Se elige un motor de corriente alterna, debido a que no se va a regular la velocidad del eje, ni tampoco se va a utilizar un torque de arranque elevado debido a que la masa del cuerpo a girar es de magnitud moderada.

- Se desea que la excitación de estos motores sea realizada por una fuente de corriente alterna trifásica y no por una fuente de corriente continua, por ser más económico, por lo cual debe ser Asíncrono.

- Se desea que el motor sea barato, eficiente, compacto, de fácil construcción y mantenimiento para lo cual se considera un motor trifásico asíncrono tipo jaula de ardilla.

- Se desea que el motor tenga una velocidad máxima de 3600RPM.
- Se desea un Par de arranque capaz de poder mover la carga "cortadora de tubos", para lo cual la potencia requerida por la carga que se va accionar debe ser menor que la

potencia nominal del motor; el motor debe poseer una gran velocidad constante o casi constante.

Todos estos requerimientos se limitan a seleccionar el motor Asíncrono trifásico tipo Jaula de Ardilla que mejor se acomode a las condiciones del proyecto.

a) Calculo del Momento de Inercia de la carga a mover

Se sabe que el momento de inercia de cualquier cuerpo rígido (cortadora de tubos) esta determinada por la siguiente ecuación matemática:

$$I = \int (r)^2 \, dm = \int \rho * (r)^2 \, dA \ldots\ldots\ldots (5.1)$$

Debido a que la cortadora de tubos consiste de un disco afilado en su borde, el cual debido a su movimiento alrededor de su eje se convierte en una excelente cortadora dependiendo de la velocidad que se le imprima; pasaremos a calcular su Momento de Inercia

$$dA = 2 * \pi * r \, dr \ldots\ldots\ldots\ldots (5.2)$$

Reemplazando la ecuación (5.2) en la ecuación (5.1) tenemos:

$$I = \int (2\pi) * \rho * (r)^3 \, dr \ldots\ldots\ldots (5.3)$$

Resolviendo la integral de la ecuación (5.3) para valores de r: [0 – R] y reemplazando el valor de la densidad superficial ρ ($M / \pi R^2$), tenemos:

$$I = (M * R^2) / 2 \ldots\ldots\ldots\ldots (5.4)$$

Si la carga que deseamos girar (cortadora de tubos) tiene una masa de 1.0Kg y el radio de la misma es de 0.1m, reemplazando en la ecuación (5.4) tenemos que el momento de inercia para nuestras condiciones es: $I = 0.005$ Kg-m^2

b) Determinación del tiempo de arranque de la carga a mover

El tiempo de arranque debe ser el menor posible debido a que la corriente en esta etapa es superior al valor nominal, para evitar así el recalentamiento del motor y por ende la reducción de su tiempo de vida útil. De acuerdo a la norma de la IEE los tiempos de arranque máximos permitidos dependen de la potencia del motor elegido, según como se detalla la Tabla N° 5.7.

Tabla N° 5.7 Potencia vs Tiempo de Arranque

Motores de Potencia hasta (KW)	Máximo Tarranque (seg)
0.2	0.1
1	1.5
5	6
10	10
16	16
70	21
100	24

c) Determinación de la Potencia de carga requerida

Además, sabemos que el teorema del trabajo y la energía en el movimiento de rotación se expresa de la siguiente manera:

$$W = (1/2) * I * [w^2 - wo^2] \quad \text{..... (5.5)}$$

Finalmente, se concluye que la potencia de la carga (cortadora de tubos) se puede expresar en función de los siguientes parámetros:

$$Pcarga = W / (Tarranque) \quad \text{..... (5.6)}$$

Reemplazando la ecuación (5.5) en la ecuación (5.6) se tiene:

$$Pcarga = (1/2) * I * [w^2 - wo^2] / [Tarranque] \quad \text{..... (5.7)}$$

Si queremos que la carga gire a una velocidad angular de 3600 RPM, reemplazando en la ecuación (5.7) tenemos que la Potencia de la carga nominal para nuestras condiciones es: Pcarga = 90 watt.

Se determina la Potencia mecánica real afectando el valor de la Potencia de la carga anteriormente calculada por un factor de carga o de servicio, el cual depende del tipo de máquina movida y del tiempo de funcionamiento. Para nuestro caso se trata de una cortadora cuya carga se considera casi sin golpes y que será utilizada en el campo industrial, es decir, casi 24 horas por dia; así el factor de servicio es igual a 1.25 (según lo establece la norma NEMA).

$$Preal = 1.25 * Pcarga = 112.5 \text{ watt}$$

Usando el catálogo del fabricante de motores, se escoge aquel cuya potencia nominal sea igual o mayor que la potencia real calculada.

Finalmente se adquirirá un Motor Trifásico Asíncrono Tipo jaula de ardilla que cuente con las características de placa siguientes (ver Tabla N° 5.8):

Tabla N° 5.8 Datos de Placa de Motor para cortadora de tubos

Marca	Delcrosa
Potencia	0.2 KW
Voltaje	380 / 220 V Y / Δ
Corriente	0.52 / 0.90 A Y / Δ
Velocidad Angular	3600 RPM
Frecuencia	60 Hz
Factor de potencia	0.80

5.2.7 Sistema de Aire Comprimido

El motivo fundamental para la utilización de un sistema de aire comprimido es la necesidad de generar la neumática necesaria para poder realizar los movimientos

mecánicos del subsistema carril – retenedor, subsistema retenedor y subsistema de movimiento vertical de la cortadora. Ver Fig 5.7

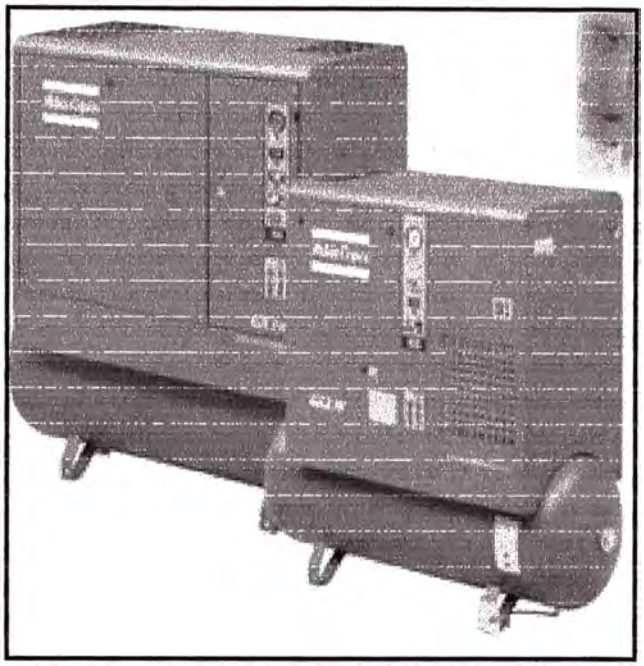

Fig. 5.7 Compresora Atlas Copco

El sistema de aire comprimido que será utilizado para poder realizar el movimiento de los diferentes actuadores presenta las siguientes características, ver Tabla N° 5.9

Tabla N° 5.9 Datos de Placa de Compresora para Sistema Neumático

Marca	Atlas Copco
Modelo	GX- 1
Presión de trabajo	2.0 Bar
Potencia del motor	2.2 kW
Nivel sonoro	31 dB (A)

Finalmente se dará a conocer una tabla resumen en la cual se detalla el presupuesto requerido de los diversos sistemas para poder afrontar con éxito la implementación de este proyecto de automatización. Ver Tabla N° 5.10

Tabla N° 5.10 Cuadro resumen de Presupuesto para el Proyecto

Item	Descripción	Cantidad	Precio (US$)
01	Computadora Personal con procesador Intel Pentium II, memoria RAM de 128MB, disco duro de 5GB	01	150.00
02	Microsistema PLC marca Siemens, modelo Simatic, CPU S7-200, con 14 entradas digitales y 10 salidas digitales	01	500.00
03	Fuente de Alimentación marca Siemens, modelo Logo Power	01	100.00
04	Software MicroWin 32 – Step 7	01	100.00
05	Cable MPI para comunicación PLC – PC	01	90.00
06	Sensores Inductivos de posición marca IFM electronics, modelo Efector 100	07	560.00
07	Actuadores neumáticos marca Neumax, modelo Serie 300	04	400.00
08	Optoacopladores marca ABB, modelo Serie R500	11	330.00
09	Sistema de Aire comprimido marca Atlas Copco, modelo GX-1	01	300.00
10	Motor Trifásico Asíncrono marca Delcrosa de 0.2 KW	01	100.00
11	Sub-sistema de Aluminio formado por carril y retenedor	01	50.00
12	Sub-sistema de Aluminio formado por un retenedor	01	30.00
13	Sub-sistema de Aluminio formado por cuchilla circular acoplada a un carril vertical con pedestal	01	50.00
14	Suministros varios: Interruptor principal, contactor de 24VDC, contactor de 220VAC, pulsadores tipo rasante, cables 12 AWG y 14 AWG, mangueras	01	200.00
		TOTAL:	***2,960.00***

CONCLUSIONES

1.- Durante todo momento de ejecución del programa de control del proceso, el motor que controla la cortadora se encuentra activado, tanto es así, que para el movimiento de una secuencia a otra del proceso es necesario y requisito imprescindible de que el motor se encuentre activado, esto es, debido a que en un proceso real si existieran problemas en el encendido del motor el proceso no continuaría hasta que este problema sea resuelto pues de lo contrario podrían aparecer errores en el proceso de cortado.

2.- La disposición del movimiento planar de los retenedores B y C y del carril A permite una correcta medición de la longitud de corte requerida durante el control del proceso mediante la secuencia de corte y un perfecto desplazamiento del tubo cuantas veces éste sea requerido.

3.- La variable "MM" me permite evitar de que la variable "Y4" se habilite constantemente con las condiciones de posición de retorno de la cortadora, debido al enclavamiento de esta variable con la variable de posición "D1", esta opción es muy importante si desea obtener cortes de tubo de longitudes diferentes.

4.- Cuando la variable "PARO" es activada, el proceso de cortado lineal se detiene o se congela sin que se ejecute la siguiente sentencia, esto es utilizado para revisar problemas de la mecánica del proceso o realizar los cambios necesarios en la instrumentación del proceso.

5.- Cuando la variable "EMERGENCIA" es activada, los comandos de seguridad del programa de control se efectúan con la habilitación de los retenedores B y C, el retorno del carril y de la cortadora a su posición inicial que permiten inmovilizar al elemento lineal que esta siendo cortado (tubo) al presionar la variable "LIBERA" las variables retornan a sus condiciones iniciales para reiniciar el proceso de cortado.

6.- Durante la elaboración del programa se utilizan memorias del tipo Set y Reset para habilitar las variables "Y1", "Y2", "Y3" e "Y4" debido a que la condición lógica se mantiene hasta que se produzca una instrucción contraria a la que se encuentra habilitada, esto permite manejar de mejor forma el proceso de cortado ante contínuos cambios en los valores de las variables de posición del proceso. Además estas memorias han sido configuradas teniendo en consideración las medidas de seguridad mencionadas en el paso anterior.

7.- Los contadores utilizados son reseteados por temporizadores a la conexión con un período de activación de 1 segundo y que son habilitados cuando la cuenta aumenta en uno debido a la habilitación de la variable "Y4".

8.- El lenguaje de programación utilizado para la elaboración del programa de control de la cortadora se representa en el Diagrama de Bloques de función (FBD) y es el más utilizado por los ingenieros electrónicos por su amplia similitud con la electrónica digital, para el caso de los ingenieros electricistas se les recomienda trabajar con el lenguaje escalera o "ladder" por su similitud con el diagrama de contactos bien conocido por ellos.

9.- Las variables del programa han sido definidas con más detalle en el Anexo C y dentro del programa han sido declaradas en el campo "variables globales" para que estas puedan interactuar de manera casi inmediata entre los sub-programas Cortador y Contador del presente trabajo. Algunas variables muestran un valor por defecto establecido según su aparición en el mercado o según lo requiera el programa en mención.

10.- La simulación en lenguaje de máquina me permite conocer el comportamiento lógico binario de los dispositivos electrónicos ante respuestas de los sensores de campo que son enviados durante la finalización de cada sentencia, es una forma de conocer de manera

directa el funcionamiento de control de la cortadora lineal de tres dimensiones diferentes y proporcionales para propósitos de análisis profundo y dirigido especialmente a programadores que deseen conocer a profundidad la lógica de control del proceso.

11.- La simulación en entorno gráfico es una interfase más amigable y cómoda para el operario encargado de controlar el proceso pues todo se reduce a una representación simbólica basada en el esquema del proceso de control de la cortadora lineal, en donde, se visualiza la simulación de pulsadores, sensores y señalizadores que permiten representar una copia fiel del proceso en cuestión.

12.- Se ha demostrado que cuando se selecciona un corte lineal ya sea de longitud L, 2L o 3L este se produce sin ningún problema y con todas las condiciones de seguridad impuestas por el programa de control, la secuencia es realizada en forma perfecta y no tiene lugar a equivocación certificando de esta forma que el programa de control es considerado de alta confiabilidad.

ANEXO A
PROGRAMACION DE UN PLC

A.1INTRODUCCION A LA PROGRAMACION

Antes de empezar la programación propiamente dicha, es necesario definir algunos conceptos que proporcionen al lector las bases suficientes para comprender de la manera mas clara la programación básica y avanzada, así por ejemplo, el lector deberá estar en condiciones de diferenciar una señal discreta de una análoga, representar las cantidades binarias, estructurar una instrucción de mando, tener presente las reglas básicas para las diferentes representaciones de los lenguajes de programación, etc.

A.2 TIPOS DE SEÑALES

Existen 2 tipos de señales bien definidas que un PLC puede procesar, estos son:

A.2.1 Señal Discreta

Este tipo de señal es conocido también con los siguientes nombres:

- Señal Binaria
- Señal Digital
- Señal Lógica
- Señal todo o nada (TON)

Se caracterizan porque solo pueden adoptar uno de dos posibles estados o niveles. A estos dos estados posibles se les asocia para efectos del procesamiento el **estado de señal 0** y el **estado de señal 1**. Así mismo, estos estados cuando se relacionan de acuerdo a su condición eléctrica se dice: no existe tensión y existe tensión, la magnitud de la tensión no interesa ya que dependerá del diseño del componente electrónico que pueda asumir esta tensión nominal. Para mayor detalle ver fig. A.1.

Como ejemplo, se puede citar aquellos dispositivos de campo de entrada y salida de donde provienen o se asigna una señal discreta con respecto a un PLC:

Entrada:

- Pulsador
- Interruptor de Posición
- Interruptor Fotoeléctrico, etc.

Salida:

- Contactor
- Lámpara Indicadora, etc.

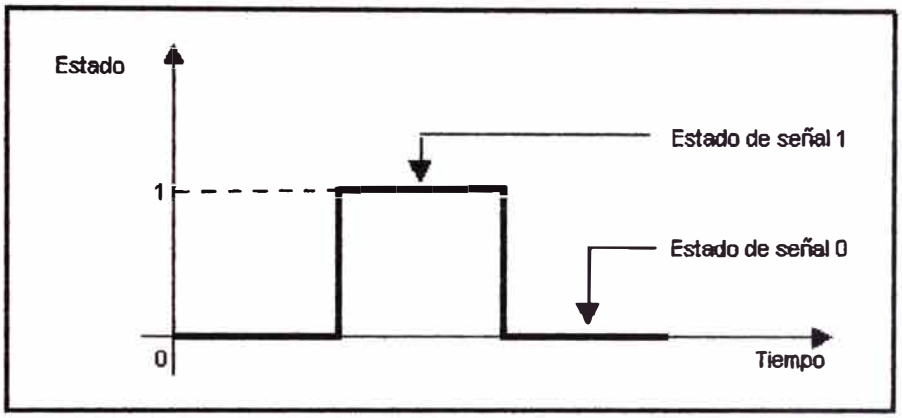

Fig. A.1 Esquema representativo de una señal Discreta

A.2.2 Señal Análoga

Se conoce como señal análoga, aquella cuyo valor varía con el tiempo y en forma continua, pudiendo asumir un número infinito de valores entre sus límites mínimos y máximos. Para mayor detalle ver fig. A.2.

A continuación se citan algunos parámetros físicos muy utilizados en los procesos industriales, tal que, en forma de señal análoga pueden ser controlados y medidos.

- Temperatura
- Velocidad

- Presión
- Flujo, nivel, etc.

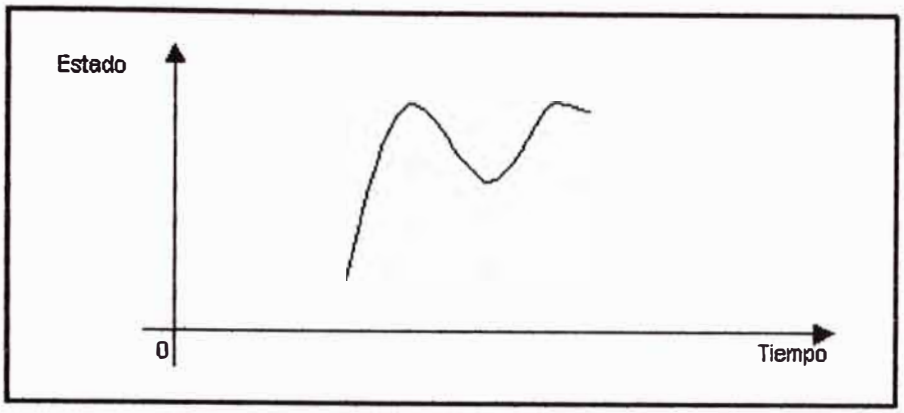

Fig. A.2 Esquema representativo de una señal Análoga

A.3 REPRESENTACION DE LAS CANTIDADES BINARIAS

Dado que el PLC recepciona la información proveniente del proceso ya sea en forma discreta o análoga, donde la información se almacena en forma de una agrupación binaria, es preciso por lo tanto disponer de un medio de representación que facilite su manejo y mejore la capacidad de procesamiento.

Para ello se emplean con frecuencia tres tipos de representación para la información, éstos son: bit, byte y palabra, en algunos casos se utiliza la doble palabra.

A.3.1 BIT

El Bit es la unidad elemental de información donde solo puede tomar dos valores un "**1**" o un "**0**", es decir, un bit es suficiente para representar una señal binaria.

A.3.2 BYTE

Es una unidad compuesta por una agrupación ordenada de 8 bits. Los bits se agrupan de derecha a izquierda tomando como numero de bit del 0 al 7. En un byte se puede representar el estado de hasta 8 señales binarias, puede usarse para almacenar un número cuya magnitud como máximo sería:

Número máximo de un Byte = 1111 1111 = 256 – 1 = 255 ..... (A.1)

A.3.3 PALABRA

Para obtener mayor capacidad de procesamiento se agrupan los bytes formando palabras. La palabra es una unidad mayor compuesta hasta de 16 bits = 2 Bytes. Los bits de una palabra se agrupan de derecha a izquierda tomando como número de bit del 0 al 15. En una palabra se pueden representar hasta 16 señales binarias, puede usarse para almacenar un número cuya magnitud como máximo sería:

Número máximo en una Palabra = 65536 – 1 = 65535 ..... (A.2)

A.4 DIRECCIONAMIENTO DE E/S

Cuando se elabora un programa de control, se van indicando las diferentes instrucciones de mando donde en cada instrucción se indica que operación se debe ejecutar, también figura la dirección exacta del módulo y canal o terminal de conexión de las señales de E/S involucradas en el proceso. El direccionamiento es de dos formas:

- Direccionamiento fijo
- Direccionamiento variable

A.4.1 Direccionamiento Fijo

Cuando la dirección de las señales de entrada y salida (E/S) queda determinada por la posición o puesto de conexión en que están ubicados los módulos de E/S respecto a la CPU, se dice que el direccionamiento es fijo. Además, un direccionamiento fijo puede ser del tipo Octal o Hexadecimal.

- ***Direccionamiento Fijo del tipo Octal:***

Queda determinado cuando a cada módulo de E/S se le agrupa los terminales por bytes, es decir, en grupos de 8 bits (del 0 al 7). En este caso, en la dirección se especificará el byte correspondiente al terminal seleccionado y que pertenece al puesto de enchufe según la posición que ocupa.

- ***Direccionamiento Fijo del tipo Hexadecimal:***

Este direccionamiento se diferencia del anterior en el agrupamiento de los terminales, siendo para este caso del tipo hexadecimal, es decir, en grupos de 16 bits (del 0 al F).

A.5 PROGRAMACION EN LISTA DE INSTRUCCIONES

Es una forma sencilla de programar aplicaciones de automatización sin necesidad de requerir conocimientos previos de alguna materia, debido a que los programas se basan en instrucciones del tipo booleano con simbología elemental y precisa. Las limitaciones que presenta esta forma de programar pueden ser:

- Cuando se tiene muchas instrucciones es difícil de entender rápidamente de lo que trata el programa.

- Un programa que consta de una gran cantidad de instrucciones es muy laborioso ingresarlas utilizando cualquier tipo de programador.

- Se emplea mayor tiempo en el diagnóstico y detección de fallas, etc.

No obstante una de las ventajas que presenta es que los programas diseñados para este propósito no son muy costosos (hand-held o portátiles) ni requieren softwares especiales como en el caso de las PC's. En esta parte se reconocerá la estructura de una instrucción de mando con ejemplos para algunas marcas de PLC's, y a continuación las operaciones binarias utilizando esta forma de representación.

A.6 ESTRUCTURA DE UNA INSTRUCCIÓN DE MANDO

Una instrucción de mando es la parte más pequeña de un programa y representa para el procesador una orden de trabajo. Para que la instrucción de mando cumpla su función es necesario especificar dos partes: la parte operacional y la parte del operando. Para una representación esquemática ver Tabla N° A.1.

Tabla N° A.1 Conformación de una Instrucción de Mando

INSTRUCCIÓN DE MANDO		
OPERACIÓN	*OPERANDO*	
	Tipo	*Dirección*

La parte operacional representa lo que hay que hacer, esto significa la operación a ejecutar. Por ejemplo ejecutar un(a):

- Combinación binaria "Y" (And)
- Combinación binaria "O" (Or)
- Combinación binaria "O-exclusiva" (XO)
- Operación de carga "L" (Load)
- Operación de transferencia "T" (Transference)
- Salto a una instrucción determinada "JMPi" (Jump)
- Asignación a un resultado "=", etc.

La parte del operando esta compuesto por el tipo de operando y su dirección. El operando responde a la pregunta con que se hace la operación. El tipo de operando puede ser un (a):

- Entrada
- Salida
- Memoria interna
- Dato
- Temporizador
- Contador, etc.

La dirección del operando se define según el tipo de direccionamiento que se emplee, fijo o variable y del número del terminal de los módulos de E/S.

A.7 EJEMPLO DE INSTRUCCIONES DE MANDO PARA DIFERENTES MARCAS DE PLC

Se tratará de detallar a continuación la estructura de la instrucción de mando para determinadas marcas de PLC's dando algunos ejemplos para una mejor comprensión.

A.7.1 Representación de una Instrucción de mando en PLC Siemens (Simatic S5)

Para conocer el significado de cada variable es necesario ver fig. A.3.

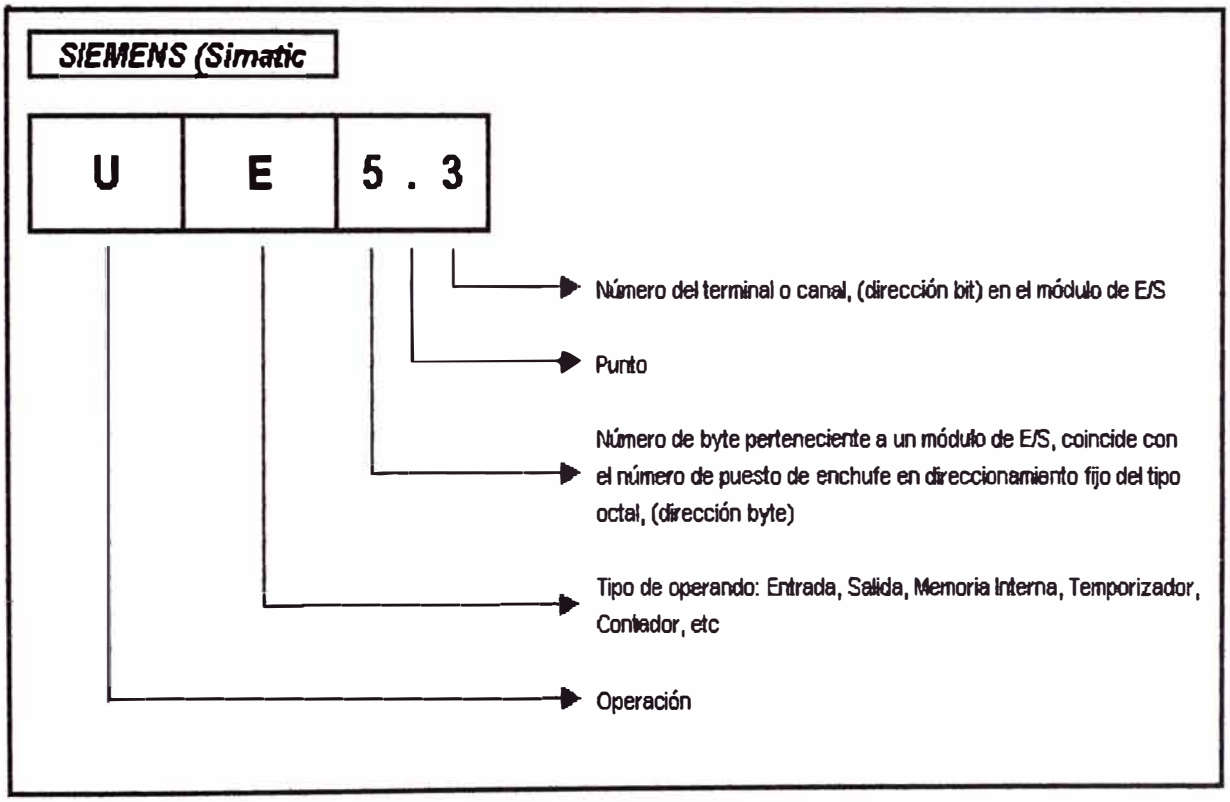

Fig. A.3 Estructura de una Instrucción de Mando en PLC Siemens

Ejemplo: En la Tabla N° A.2 se da a conocer algunos ejemplos de instrucciones.

Tabla N° A.2 Significado de Ejemplos de Instrucciones en PLC Siemens

INSTRUCCION		*SIGNIFICADO*
ALEMAN	*INGLES*	
U E 5.3	A I 5.3	*Lectura del estado de señal del canal 3, de un módulo de entrada digital de 8 canales, enchufado en el puesto 5.*
= A 10.6	= Q 10.6	*Salida del estado de señal por el canal 6, de un módulo de salida digital de 32 canales enchufado en el puesto 2. Dirección Byte 10.*
ON M 3.7	ON F 3.7	*Lectura del estado negado de la marca, con dirección byte 3 y dirección bit 7.*
L EB 7	L IB 7	*Lectura de los estados de señal de todos los canales, de un módulo digital de entrada de 8 canales enchufado en el puesto 7.*

A.7.2 Representación de una Instrucción de mando en PLC Telemecanique Compacto

Para conocer el significado de cada variable es necesario ver fig. A.4.

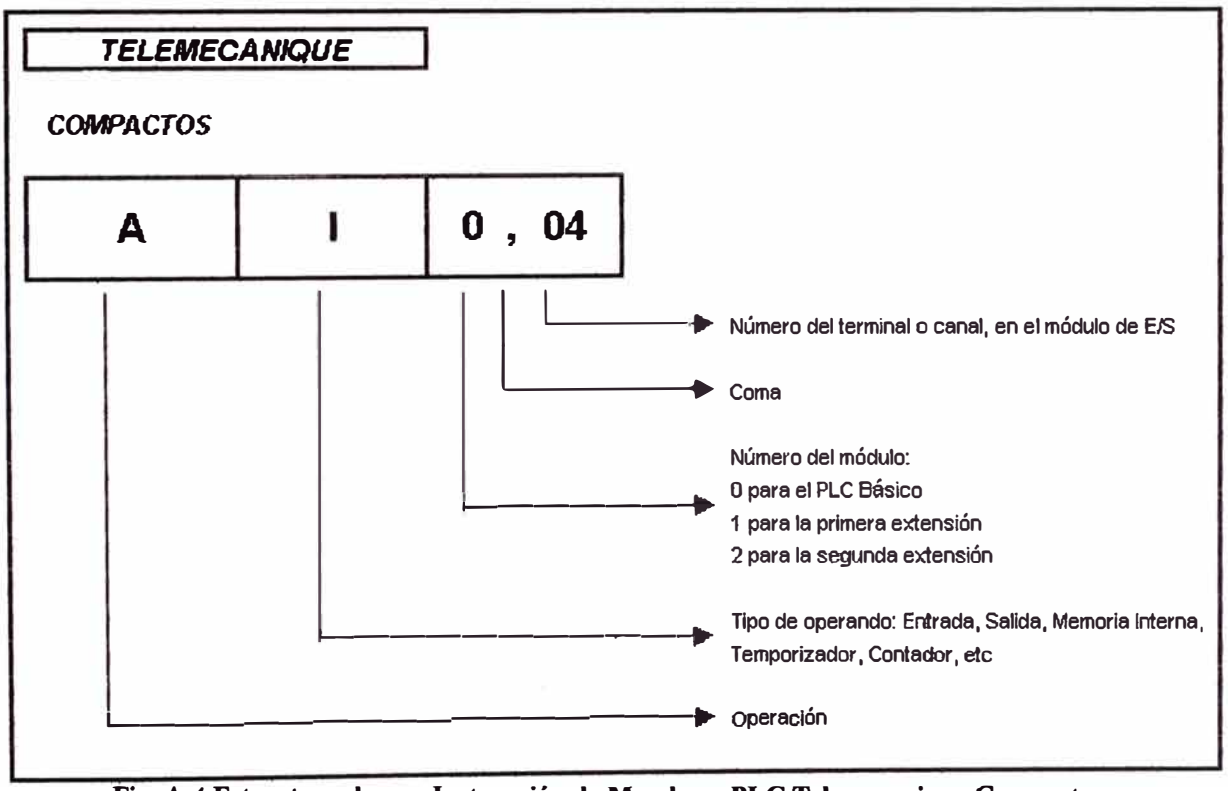

Fig. A.4 Estructura de una Instrucción de Mando en PLC Telemecanique Compacto

Ejemplo: En la Tabla N° A.3 se da a conocer algunos ejemplos de instrucciones.

Tabla N° A.3 Significado de Ejemplos de Instrucciones en PLC Telemecanique Compacto

INSTRUCCIÓN	*SIGNIFICADO*
A I 0,04	*Lectura del estado de señal del canal 4, del módulo 0 (módulo básico).*
= O 2,07	*Salida del estado de señal por el canal 7, del módulo 2 (módulo de segunda extensión).*
L T 5	*Lectura del Temporizador número 5.*

A.7.3 Representación de una Instrucción de mando en PLC Telemecanique Modular

Para conocer el significado de cada variable es necesario ver fig. A.5.

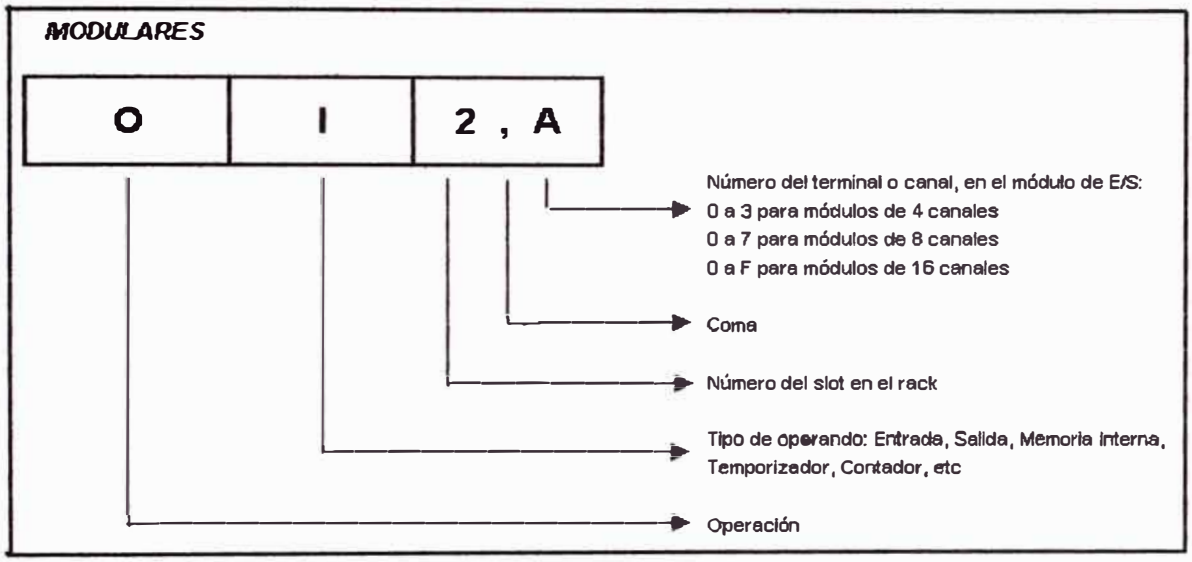

Fig. A.5 Estructura de una Instrucción de Mando en PLC Telemecanique Modular

Ejemplo: En la Tabla N° A.4 se da a conocer algunos ejemplos de instrucciones.

Tabla N° A.4 Significado de Ejemplos de Instrucciones en PLC Telemecanique Modular

INSTRUCCIÓN	*SIGNIFICADO*
O I 2,A	*Lectura del estado de señal del canal 10, de un módulo de entrada digital de 16 canales, enchufado en el puesto (slot) 2.*
= O 14,2	*Salida del estado de señal por el canal 2, de un módulo de salida digital de 32 canales, enchufado en el puesto (slot) 14.*
A C 8	*Lectura del Contador número 8.*

ANEXO B

UTILIZACION DE LAS INSTRUCCIONES EN UN PLC

B.1 INSTRUCCIONES BASICAS

Este capítulo contiene información acerca de las instrucciones generales y explica cómo funcionan en su programa de aplicación. Cada una de estas instrucciones básicas contiene información acerca de:

- Cómo aparecen los símbolos de instrucción
- Cómo usar la instrucción

En la Tabla N° B.1 se da a conocer las Instrucciones del Temporizador / Contador.

Tabla N° B.1 Instrucciones del Temporizador y Contador

Instrucción		Propósito
Mnemónico	**Nombre**	
TON	Temporizador a la conexión	*Cuenta los intervalos de la base de tiempo cuando la instrucción es verdadera.*
TOF	Temporizador a la desconexión	*Cuenta los intervalos de la base de tiempo cuando la instrucción es falsa.*
CTU	Conteo Progresivo	*Incrementa el valor acumulador a cada transición de falso a verdadero reteniendo el valor acumulado cuando la instrucción se hace falsa o cuando ocurre un ciclo de alimentación eléctrica.*
CTD	Conteo Regresivo	*Decrementa el valor acumulador a cada transición de falso a verdadero reteniendo el valor acumulado cuando la instrucción se hace falsa o cuando ocurre un ciclo de alimentación eléctrica.*

Estas instrucciones, cuando se usan en programas de escalera, representan circuitos de lógica cableada usados para el control de una máquina o equipo.

B.1.1 Descripción General de las Instrucciones del Temporizador

Cada dirección de temporizador se compone de un elemento de 3 palabras. Palabra 0 es la palabra de control, palabra 1 almacena el valor preseleccionado y palabra 2 almacena el valor acumulado.

Bits direccionables

EN = Bit 15 Habilitación

TT = Bit 14 Temporización del tempor.

DN = Bit 13 Efectuado

Palabras direccionables ACC = Valor acumulado

PRE = Valor preseleccionado

Valor acumulado (.ACC): Este es el tiempo transcurrido desde el último restablecimiento del temporizador. Cuando esta habilitado, el temporizador lo actualiza constantemente.

Valor preseleccionado (.PRE): Especifica el valor que el temporizador debe alcanzar antes de que el controlador establezca el bit efectuado. Cuando el valor acumulado sea igual o mayor al valor preseleccionado, el bit de efectuado estará establecido. Puede usar este bit para controlar un dispositivo de salida.

Los valores preseleccionados y acumulados para temporizadores tienen un rango desde 0 hasta +32,767. Si el valor preseleccionado o acumulador de temporizador es un número negativo, ocurre un error de tiempo de ejecución.

Base de Tiempo: Determina la duración de cada intervalo de base de tiempo. Para los procesadores SLC5/02 la base de tiempo ha sido establecida a 0.01 seg, para los procesadores SLC5/03, SLC5/04 y los controladores MicroLogix 1000, la base de tiempo es seleccionable como 0.01 o 1 seg.

Precisión del temporizador: Se refiere al tiempo transcurrido entre el momento en que una instrucción de temporizador está habilitada y el momento en que el intervalo temporizado se ha completado. La inexactitud causada por el scan del programa puede ser mayor que la base de tiempo del temporizador. También debe considerar el tiempo necesario para activar el dispositivo de salida.

La precisión de temporización es 0.01 a +0 segundos, con un scan de programa de hasta 2.5 segundos. El temporizador de 1 segundo mantiene la precisión con un scan de programa de hasta 1.5 segundos. Si sus programas pueden exceder 1.5 o 2.5 segundos, repita el renglón de instrucción del temporizador para que el renglón sea escaneado dentro de estos límites.

La temporización podría resultar inexacta si las instrucciones de salto (JMP), etiqueta (LBL), salto a subrutina (JSR) o subrutina (SBR) saltan el renglón que contiene una instrucción de temporizador mientras que el temporizador esté temporizando. Si la duración de salto es menor de 2.5 segundos, no se pierde ningún tiempo; si la duración de salto excede 2.5 segundos, ocurre un error de temporización no detectable. Cuando se usan subrutinas, es necesario que un temporizador esté ejecutando a un mínimo de cada 2.5 segundos para evitar un error de temporización.

Estructura del direccionamiento: Se establecen las direcciones de bits y palabras usando el formato **Tf:e.s/b**. Donde:

"T" Archivo de Temporizador.
"f" Número de archivo. Para los procesadores SLC 500, el número determinado es 4. Se puede usar un número entre 10 y 255 para almacenamiento adicional.
":" Delimitador de elemento
"e" Número de elemento, siendo elementos de 3 palabras. Para los procesadores SLC 500 el rango es 0-255. Para los controladores Micrologix 1000 el rango es de 0-39.
"." Elemento de palabras
"s" Sub-elemento
"/" Delimitador de bit
"b" Bit

B.1.1.a Temporizador a la conexión (TON)

Usar la instrucción TON para activar o desactivar una salida después de que el temporizador haya estado activado durante un intervalo de tiempo preseleccionado. La instrucción TON comienza a contar los intervalos de la base de tiempo cuando las condiciones de renglón se hacen verdaderas. Con tal que las condiciones de renglón permanezcan verdaderas, el temporizador ajusta su valor acumulado (ACC) durante cada evaluación hasta alcanzar el valor predeterminado (PRE). Cuando las condiciones de renglón se hacen falsas, el valor acumulado se reinicializa sin importar si el temporizador ha sobrepasado el límite de tiempo. Para mayor detalle ver Tabla N° B.2.

Tabla N° B.2 Bits del Temporizador TON

Este bit	Se establece cuando	Y permanece establecido hasta ocurrir uno de los
Bit de efectuado del Temporizador DN (bit 13)	El valor acumulado es igual o mayor que el valor preseleccionado	Las condiciones de renglón se hacen falsas
Bit de temporización del Temporizador TT (bit 14)	Las condiciones de renglón son verdaderas y el valor acumulado es menor que el valor preseleccionado	Las condiciones de renglón se hacen falsas o cuando el bit de efectuado esté establecido
Bit de habilitación del Temporizador EN (bit 15)	Las condiciones de renglón son verdaderas	Las condiciones de renglón se hacen falsas

Cuando el procesador cambia del modo de marcha REM o prueba REM al modo de programa REM o la alimentación eléctrica del usuario se pierde durante la temporización de la instrucción, pero no ha alcanzado su valor preseleccionado, ocurre lo siguiente:

- El bit de habilitación del temporizador (EN) permanece establecido
- El bit de temporización del temporizador (TT) permanece establecido
- El valor acumulado (ACC) permanece sin cambio

Puede ocurrir lo siguiente al regresar al modo de marcha REM o prueba REM (ver Tabla N° B.3):

Tabla N° B.3 Condición de verdad y falsedad del Temporizador TON

Condición	Resultado
Si el renglón es verdadero:	El bit EN permanece establecido El bit TT permanece establecido El valor ACC está puesto a cero
Si el renglón es falso:	El bit EN está restablecido El bit TT está restablecido El valor ACC está puesto a cero

B.1.1.b Temporizador a la desconexión (TOF)

Usar la instrucción TOF para activar o desactivar una salida después de que su renglón ha estado desactivado durante un intervalo de tiempo preseleccionado. La instrucción TOF comienza a contar los intervalos de la base de tiempo cuando el renglón efectúa una transición de verdadero a falso. Con tal que las condiciones permanezcan falsas, el

temporizador incrementa su valor acumulado (ACC) durante cada scan hasta alcanzar el valor preseleccionado (PRE). El valor acumulado se restablecerá cuando las condiciones de renglón se hagan verdaderas, sin importar si el tiempo en el temporizador se ha agotado. Ver Tabla N° B.4.

Tabla N° B.4 Bits del Temporizador TOF

Este bit	Se establece cuando	Y permanece establecido Hasta ocurrir uno de los siguientes eventos
Bit de efectuado del Temporizador DN (bit 13)	Las condiciones de renglón son verdaderas	Las condiciones de renglón se hacen falsas y el valor acumulado es mayor o igual que el valor preseleccionado
Bit de temporización del Temporizador TT (bit 14)	Las condiciones de renglón son falsas y el valor acumulado es menor que el valor preseleccionado	Las condiciones de renglón se hacen verdaderas o cuando el bit de efectuado se restablece
Bit de habilitación del Temporizador EN (bit 15)	Las condiciones de renglón son verdaderas	Las condiciones de renglón se hacen falsas

Cuando la operación del procesador cambia del modo de marcha REM o prueba REM al modo de programa REM o cuando se pierde la alimentación eléctrica del usuario durante la temporización de una instrucción de retardo con temporizador desactivado, pero no ha alcanzado su valor preseleccionado, ocurre lo siguiente:

- El bit de habilitación del temporizador (EN) permanece establecido
- El bit de temporización del temporizador (TT) permanece establecido
- El bit de efectuado del temporizador (DN) permanece establecido
- El valor acumulado (ACC) permanece sin cambio

B.1.1.c Temporizador Retentivo (RTO)

Use la instrucción RTO para activar o desactivar una salida después que el temporizador haya estado desactivado durante un intervalo de tiempo preseleccionado. La instrucción RTO es una instrucción retentiva que comienza a contar los intervalos de base de tiempo cuando las condiciones de renglón se hacen verdaderas (ver Tabla N° B.5). La instrucción RTO retiene su valor acumulado cuando ocurre cualquiera de los eventos siguientes:

- Las condiciones de renglón se hacen falsas
- Cambia la operación del procesador del modo de marcha REM o prueba REM al modo de programa REM
- Se corta la alimentación eléctrica del procesador (si cuenta con una batería auxiliar)
- Ocurre un fallo

Cuando regresa el procesador al modo de marcha REM o prueba REM y/o las condiciones de renglón se hacen verdaderas, la temporización continúa desde el valor acumulado retenido. Los temporizadores retentivos miden el período acumulativo durante el cual las condiciones de renglón son verdaderas mediante la retención de su valor acumulado.

Tabla N° B.5 Bits del Temporizador Retentivo RTO

Este bit	Se establece cuando	Y permanece establecido Hasta ocurrir uno de los siguientes eventos
Bit de habilitación del Temporizador EN (bit 15)	Las condiciones de renglón son verdaderas	Las condiciones de renglón se hacen falsas
Bit de temporización del Temporizador TT (bit 14)	Las condiciones de renglón son verdaderas y el valor acumulado es menor que el valor preseleccionado	Las condiciones de renglón se hacen falsas o cuando se establece el bit de efectuado
Bit de efectuado del Temporizador DN (bit 13)	El valor acumulado es igual o mayor que el valor preseleccionado	La instrucción RES apropiada se habilita

Cuando el procesador cambia del modo de marcha REM o prueba REM al modo de programa REM o fallo REM, o cuando se pierde la alimentación eléctrica del usuario durante la temporización del temporizador, pero todavía sin alcanzar el valor preseleccionado, ocurre lo siguiente:

- El bit de habilitación (EN) del temporizador permanece establecido
- El bit de temporización (TT) del temporizador permanece establecido
- El valor acumulado (ACC) permanece sin cambio

Puede ocurrir lo siguiente al regresar al modo de marcha REM o prueba REM o cuando se restaura la alimentación eléctrica (ver Tabla N° B.6):

Tabla N° B.6 Condición de verdad y falsedad del Temporizador Retentivo RTO

Condición	Resultado
Si el renglón es verdadero:	El bit EN permanece establecido El bit TT permanece establecido El valor ACC permanece sin cambio y vuelve a incrementar, continúa la cuenta
Si el renglón es falso:	El bit EN se restablece El bit TT se restablece El bit DN permanece en su último estado El valor ACC permanece en su último estado

B.1.2 Uso de los Contadores

B.1.2.a Elementos del Archivo de datos del contador

Cada dirección de contador se compone de un elemento de archivo de datos de 3 palabras. Palabra 0 es la palabra de control y contiene los bits de estado de la instrucción. Palabra 1 es el valor preseleccionado. Palabra 2 es el valor acumulado. Ver fig. B.1.

La palabra de control para las instrucciones del contador incluye seis bits de estado, según lo indicado a continuación:

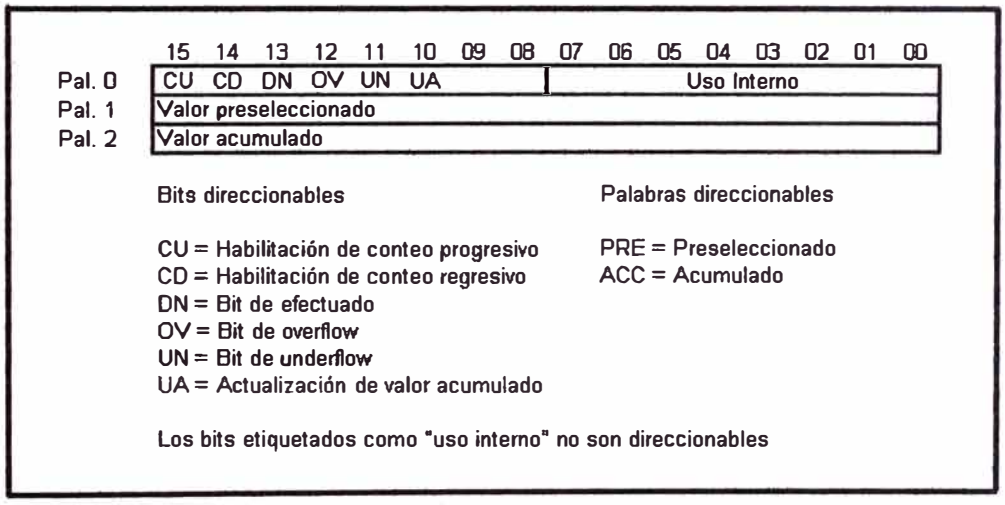

Fig. B.1 Estructura interna de datos del contador

Valor Acumulado (.ACC): Este es el número de transiciones de falso a verdadero que ha ocurrido desde el último restablecimiento del contador.

Valor Preseleccionado (.PRE): Especifica el valor que el contador debe alcanzar antes que el controlador establezca el bit de efectuado. Cuando el valor del acumulador se hace igual o mayor que el valor preseleccionado, se establece el bit de estado efectuado. Puede usar este bit para controlar un dispositivo de salida.

Los valores preseleccionados y acumulados para los contadores oscilan entre –32,768 hasta +32,767 y se almacenan como enteros con signos. Los valores negativos se almacenan en forma de complemento de 2. La fig. B.2 muestra cómo funciona un contador. El valor del contador debe permanecer dentro del rango de –32,768 hasta +32,767, si no se establece un bit de overflow (OV) o underflow (UN) de estado del contador, según sea el caso. Un contador se puede poner a cero usando la instrucción de restablecimiento (RES).

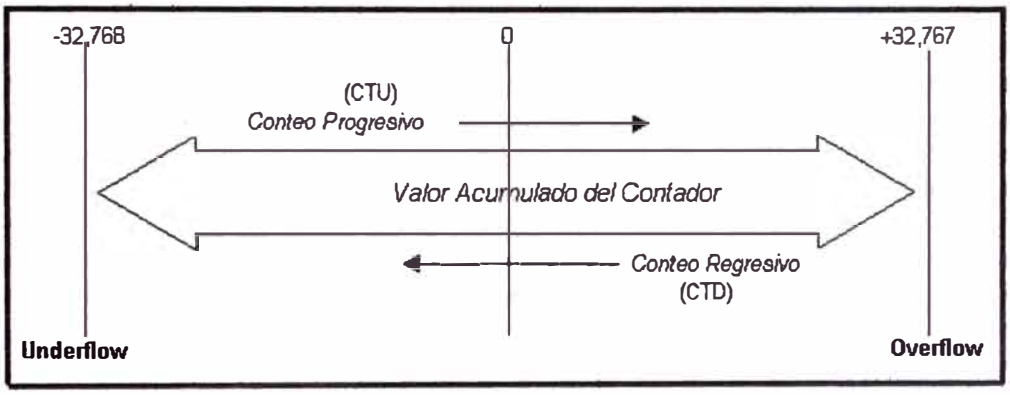

Fig. B.2 Explicación esquemática de funcionamiento de un Contador

B.1.2.b Conteo Progresivo (CTU)

El CTU es una instrucción que cuenta las transiciones de renglón de falso a verdadero. Las transiciones de renglón pueden ser provocadas por eventos ocurriendo en el programa (de la lógica interna o dispositivos de campo externos) tales como piezas que pasan por un detector o que activan un interruptor de límite. Cuando las condiciones de renglón para una instrucción CTU efectúan una transición de falso a verdadero, el valor acumulado se incrementa en uno, siempre que el renglón que contiene la instrucción CTU se evalúe entre estas transiciones. La capacidad del contador para detectar transiciones de falso a verdadero depende de la velocidad (frecuencia) de la señal de entrada.

La duración activada y desactivada de una señal de entrada no debe ser más rápida que el tiempo de scan (bajo un ciclo de trabajo del 50%). El valor acumulado se retiene cuando

las condiciones de renglón vuelven a hacerse falsas. El conteo acumulado se retiene hasta que sea puesto a cero por una instrucción de restablecimiento (RES) que tenga la misma dirección que el contador. Ver Tabla N° B.7.

Tabla N° B.7 Bits del Contador Progresivo CTU

Este bit	Se establece cuando	Y permanece establecido Hasta ocurrir uno de los siguientes eventos
Bit habilitación de conteo progresivo CU (bit 15)	Las condiciones de renglón son verdaderas	Las condiciones de renglón se hacen falsas o bien se habilita una instrucción RES con la misma dirección que la instrucción CTU
Bit de efectuado DN (bit 13)	El valor acumulado es igual o mayor que el valor preseleccionado	El valor acumulado se hace menor que el valor preseleccionado
Bit de overflow de conteo progresivo OV (bit 12)	El valor acumulado cambia a +32,768 (desde +32,767) y continúa contando desde ese punto	Se ejecuta una instrucción RES con la misma dirección que la instrucción CTU o bien el conteo se reduce a un valor menor o igual que +32,767 con una instrucción CTD

El valor acumulado se retiene después de que la instrucción CTU se hace falsa o la alimentación eléctrica se corta y luego se restaura al controlador. Además, el estado activado o desactivado de los bits de contador efectuado, overflow y underflow es retentivo. El valor acumulado y los bits de control se restablecen cuando se habilita la instrucción RES correcta. Los bits CU siempre se establecen antes de introducir los modos de marcha REM o prueba REM.

B.1.2.c Conteo Regresivo (CTD)

El CTD es una instrucción que cuenta las transiciones de renglón de falso a verdadero. Las transiciones de renglón pueden ser causadas por eventos que ocurren en el programa, tales como piezas pasando por un detector o accionando un final de carrera. Cuando las condiciones de renglón para una instrucción CTD han efectuado una transición de falso a

verdadero, el valor acumulado se disminuye en un conteo, siempre que el renglón que contiene la instrucción CTD se evalúe entre estas transiciones.

Los conteos acumulados se retienen cuando las condiciones de renglón se hacen falsas nuevamente. El conteo acumulado se retiene hasta que sea puesto a cero por una instrucción de restablecimiento (RES) que tiene la misma dirección que el contador restablecido. Ver Tabla N° B.8.

Tabla N° B.8 Bits del Contador Regresivo CTD

Este bit	Se establece cuando	Y permanece establecido Hasta ocurrir uno de los siguientes eventos
Bit habilitación de conteo regresivo CD (bit 14)	Las condiciones de renglón son verdaderas	Las condiciones de renglón se hacen falsas o bien se habilita una instrucción RES con la misma dirección que la instrucción CTD
Bit de efectuado DN (bit 13)	El valor acumulado es igual o mayor que el preseleccionado	El valor acumulado se hace menor que el preseleccionado
Bit de underflow de conteo regresivo UN (bit 11)	El valor acumulado cambia a -32,769 (desde -32,768) y continúa contando regresivamente desde ese punto	Una instrucción RES con la misma dirección que la instrucción CTD se ejecuta o bien el conteo es incrementado mayor o igual que –32,767 con una instrucción CTU

El valor acumulado se retiene después de que la instrucción CTD se hace falsa o cuando la alimentación eléctrica al controlador se corta y luego se restaura. Además, el estado activado o desactivado de los bits de contador efectuado, overflow y underflow es retentivo. El valor acumulado y los bits de control se restablecen cuando se habilita la instrucción RES correcta. Los bits CD siempre se establecen antes de introducir los modos de marcha REM o prueba REM.

B.1.2.d Contador de alta velocidad (HSC)

El contador de alta velocidad constituye una variación del contador CTU. La instrucción HSC se habilita cuando la lógica de renglón es verdadera y se inhabilita cuando la lógica de renglón es falsa. La instrucción HSC cuenta transiciones que ocurren en terminal de

entrada I:0/0. La instrucción HSC no cuenta las transiciones de renglón. Habilita o inhabilita el renglón HSC para habilitar o inhabilitar el conteo de transiciones que ocurren en la terminal de entrada I:0/0. Recomendamos colocar la instrucción HSC en un renglón incondicional. No coloque la instrucción XIC con la dirección I:0/0 en serie con la instrucción HSC ya que los conteos se perderán.

El HSC es un contador CTU especial para uso con los procesadores SLC fijos y SLC 5/01 de 24VCC. Los bits de estado y valores acumulados del HSC son no retentivos. Esta instrucción proporciona el conteo de alta velocidad para los controladores de E/S fijos con entrada de 24VCC. Se permite una sola instrucción HSC por cada controlador. Se recomienda un cable blindado para reducir el ruído a la entrada. Ver fig. B.3.

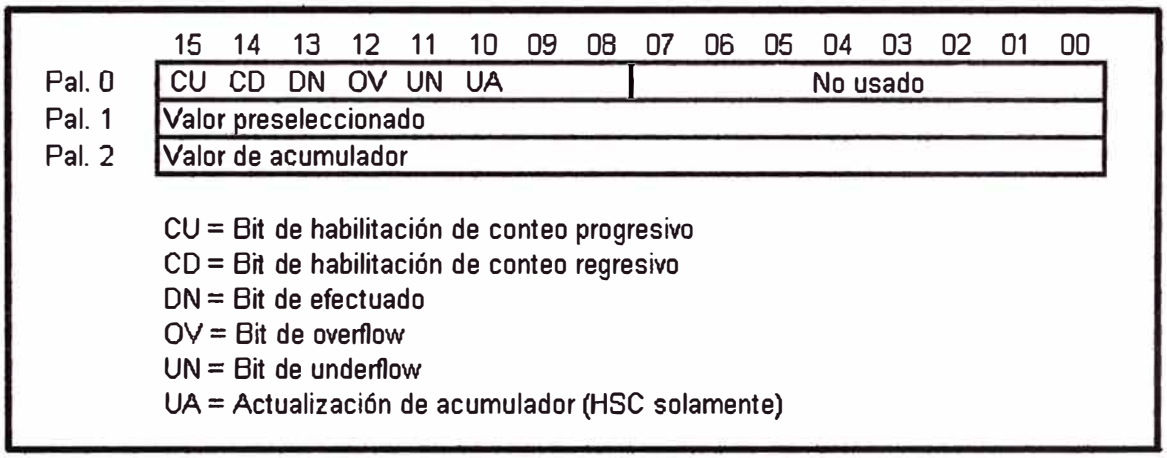

Fig. B.3 Estructura de la Data del Contador de Alta velocidad HSC

- La palabra 0 contiene los bits de estado siguientes de la instrucción HSC:

✓ El Bit 10 (UA) actualiza la palabra de acumulador del HSC para reflejar el estado inmediato del HSC cuando es verdadero.
✓ El Bit 12 (OV) indica la ocurrencia de un overflow de HSC.
✓ El Bit 13 (DN) indica si el valor preseleccionado de HSC ha sido alcanzado
✓ El Bit 15 (CU) muestra el estado de habilitación / inhabilitación de la instrucción HSC.

- La palabra 1 contiene el valor preseleccionado que se carga en el HSC cuando se ejecuta la instrucción RES, o cuando se establece el bit de efectuado o cuando se efectúa el encendido inicial.

- La palabra 2 contiene el valor del acumulador HSC. Esta palabra es actualizada cada vez que la instrucción HSC es evaluada y cuando el bit del acumulador de actualización es establecido usando una instrucción OTE. Este acumulador es de sólo lectura. Cualquier valor escrito en el acumulador resulta sobrescrito por el contador de alta velocidad durante la evaluación de instrucción, restablecimiento o introducción del modo de marcha de REM.

B.1.2.e Restablecimiento (RES)

Use una instrucción RES para restablecer un temporizador o contador. Cuando se habilita la instrucción RES, establece la instrucción de retardo del temporizador a la conexión (TON), temporizador retentivo (RTO), conteo progresivo (CTU) o conteo regresivo (CTD) con la misma dirección que la instrucción RES. Ver Tabla N° B.9.

Tabla N° B.9 Bits de la Instrucción RES

Usando una instrucción RES para un:	**El procesador restablece el:**
Temporizador (No use una instrucción RES con TOF)	Valor ACC a cero Bit DN Bit TT Bit EN
Contador	Valor ACC a cero Bit OV Bit Unt Bit DN Bit CU Bit CD
Control	Valor POS a cero Bit EN Bit Eut Bit DN Bit EM Bit ER Bit UL IN y FD van al último estado

Cuando restablece un contador, si la instrucción RES está habilitada y el renglón de contador está habilitado, se pone a cero el bit CU o CD. Si el valor preseleccionado del contador es negativo, la instrucción RES establece el valor acumulado a cero. Esto, a su vez, causa que el "bit de efectuado" sea establecida por una instrucción de conteo regresivo

o conteo progresivo. Ya que la instrucción RES establece el valor acumulado y los bits de efectuado, temporización y habilitados, no use la instrucción RES para restablecer una dirección de temporizador usada en una instrucción TOF. En caso contrario, puede ocurrir la operación inesperada de la máquina o lesiones al personal.

B.2 INSTRUCCIONES DE COMPARACION

A continuación hablaremos acerca de las instrucciones de comparación explicando cómo funcionan en el programa de aplicación (Ver Tabla N° B.10). Cada una de las instrucciones de comparación incluye información acerca de:

- Cómo debe de aparecer el símbolo de instrucción
- Cómo usar la instrucción

Tabla N° B.10 Instrucciones de comparación

Instrucción		Propósito
Mnemónico	**Nombre**	
EQU	Igual	Probar si dos valores son iguales.
NEQ	No Igual	Probar si un valor no es igual que un segundo valor.
LES	Menor que	Probar si un valor es menor que un segundo valor.
LEQ	Menor o igual que	Probar si un valor es menor o igual que un segundo valor.
GRT	Mayor que	Probar si un valor es mayor que otro.
GEQ	Mayor o igual que	Probar si un valor es mayor o igual que un segundo valor.
MEQ	Comparación Igualdad con Máscara	Probar porciones de dos valores para saber si son iguales. Compara datos de 16 bits de una dirección de fuente contra datos de 16 bits en una dirección de referencia mediante una máscara.
LIM	Prueba de límite	Probar si un valor se encuentra dentro del rango del límite de otros dos valores.

Las instrucciones de comparación se usan para probar parejas de valores para establecer condiciones de la continuidad lógica de un renglón. Como ejemplo, digamos que una instrucción LES se presenta con 2 valores. Si el primer valor es menor que el segundo, la instrucción de comparación es verdadero.

B.2.1 Igual (EQU)

La instrucción EQU se usa para probar si 2 valores son iguales. Si la fuente A y la fuente B son iguales, la instrucción es lógicamente verdadera. Si estos valores no son iguales, la instrucción es lógicamente falsa. La fuente A debe ser una dirección. La fuente B puede ser una constante de programa o una dirección. Los enteros negativos se almacenan de forma complementaria de dos.

B.2.2 No Igual (NEQ)

La instrucción NEQ se usa para probar si 2 valores no son iguales. Si la fuente A y la fuente B no son iguales, la instrucción es lógicamente verdadera. Si los dos valores son iguales, la instrucción es lógicamente falsa. La fuente A debe ser una dirección. La fuente B puede ser una constante de programa o una dirección. Los enteros negativos se almacenan de forma complementaria de dos.

B.2.3 Menor que (LES)

La instrucción LES se usa para probar si un valor (fuente A) es menor que otro (fuente B). Si la fuente A es menor que el valor de la fuente B, la instrucción es lógicamente verdadera. Si el valor de la fuente A es mayor o igual que el valor en la fuente B, la instrucción es lógicamente falsa.

La fuente A debe ser una dirección. La fuente B puede ser una constante de programa o una dirección. Los enteros negativos se almacenan de forma complementaria de dos.

B.2.4 Menor o Igual que (LEQ)

La instrucción LEQ se usa para probar si un valor (fuente A) es menor o igual que otro (fuente B). Si la fuente A es menor o igual que el valor de la fuente B, la instrucción es lógicamente verdadera. Si el valor en la fuente A es mayor que el valor en la fuente B, la instrucción es lógicamente falsa.

La fuente A debe ser una dirección. La fuente B puede ser una constante de programa o una dirección. Los enteros negativos se almacenan de forma complementaria de dos.

B.2.5 Mayor que (GRT)

La instrucción GRT se usa para probar si un valor (fuente A) es mayor que otro (fuente B). Si la fuente A es mayor que el valor en la fuente B, la instrucción es lógicamente verdadera. Si el valor en la fuente A es menor o igual que el valor en la fuente B, la instrucción es lógicamente falsa.

La fuente A debe ser una dirección. La fuente B puede ser una constante de programa o una dirección. Los enteros negativos se almacenan de forma complementaria de dos.

B.2.6 Mayor o igual que (GEQ)

La instrucción GEQ se usa para probar si un valor (fuente A) es mayor o igual que otro (fuente B). Si la fuente A es mayor o igual que el valor en la fuente B, la instrucción es lógicamente verdadera. Si el valor en la fuente A es menor que el valor en la fuente B, la instrucción es lógicamente falsa.

La fuente A debe ser una dirección. La fuente B puede ser una constante de programa o una dirección. Los enteros negativos se almacenan de forma complementaria de dos.

B.2.7 Comparación con máscara para igual (MEQ)

La instrucción MEQ se usa para comparar datos en una dirección de fuente contra datos en una dirección de comparación. El uso de esta instrucción permite que una palabra separada enmascare porciones de datos. Para lo cual se define los siguientes parámetros:

- **Fuente** es la dirección del valor que desea comparar.
- **Máscara** es la dirección de la máscara mediante la cual la instrucción mueve datos. La máscara puede ser un valor hexadecimal.
- **Comparación** es un valor de entero o la dirección de la referencia.

Si los 16 bits de datos en la dirección de fuente son iguales a los 16 bits de datos en la dirección de comparación (menos los bits con máscara), la instrucción es verdadera. La instrucción se hace falsa en el momento en que detecte una desigualdad. Los bits en la palabra de máscara enmascaran los datos al restablecerse; transmiten datos al establecerse.

B.2.8 Prueba de Límite (LIM)

La instrucción LIM se usa para probar los valores dentro o fuera de un rango especificado, según como usted haya establecido los límites.

Los valores de límite bajo, prueba y límite alto pueden ser direcciones de palabra o constantes restringidos a las combinaciones siguientes:

- Si el parámetro de prueba es una constante de programa, los parámetros de límite bajo y límite alto deben ser direcciones de palabra.

- Si el parámetro de prueba es una dirección de palabra, los parámetros de límite bajo y límite alto pueden ser una constante de programa o una dirección de palabra.

B.3 INSTRUCCIONES MATEMATICAS

A continuación se conocerán las instrucciones matemáticas y se explicarán cómo funcionan en su programa de lógica (Ver Tabla N° B.11). Cada una de las instrucciones matemáticas incluye información acerca de:

- Cómo aparece el símbolo de instrucción
- Cómo usar la instrucción

Tabla N° B.11 Instrucciones matemáticas

Instrucción		Propósito
Mnemónico	**Nombre**	
ADD	Añadir	Añade la fuente A a la fuente B y almacena el resultado en el destino.
SUB	Restar	Resta la fuente B de la fuente A y almacena el resultado en el destino.

MUL	Multiplicar	Multiplica la fuente A por la fuente B y almacena el resultado en el destino.
DIV	Dividir	Divide la fuente A por la fuente B y almacena el resultado en el destino y el registro matemático.
DDV	División doble	Divide el contenido del registro matemático por la fuente y almacena el resultado en el destino y el registro matemático.
CLR	Borrar	Pone todos los bits de una palabra a cero.
SQR	Raíz Cuadrada	Calcula la raíz cuadrada de la fuente y coloca el resultado de entero en el destino.
SCP	Escalar con parámetros	Produce un valor de salida escalado que tiene una relación lineal entre los valores de entrada y escalados.
SCL	Datos de escala	Multiplica la fuente por una tasa especificada, añade a un valor offset y almacena el resultado en el destino.
ABS	Absoluto	Calcula el valor absoluto de la fuente y coloca el resultado en el destino..
CPT	Calcular	Evalúa una expresión y almacena el resultado en el destino.
SWP	Cambiar	Cambia los bytes bajos y altos en un número especificado de palabras en un archivo de bit, entero, ASCII o cadena.
ASN	Arco Seno	Acepta el arco seno de un número y almacena el resultado (en radianes) en el destino.
ACS	Arco coseno	Acepta el arco coseno de un número y almacena el resultado (en radianes) en el destino.
ATN	Arco tangente	Acepta el arco tangente de un número y almacena el resultado (en radianes) en el destino.
COS	Coseno	Acepta el coseno de un número y almacena el resultado en el destino.
LN	Logaritmo natural	Acepta el logaritmo natural del valor en la fuente y lo almacena en el destino.
LOG	Logaritmo de base 10	Acepta el logaritmo de la base 10 del valor en la fuente y almacena el resultado en el destino.
SIN	Seno	Acepta el seno de un número y almacena el resultado en el destino.
TAN	Tangente	Acepta la tangente de un número y almacena el resultado en el destino.
XPY	X a la potencia de Y	Eleva un valor a la potencia y almacena el resultado en el destino.

La mayor parte de las instrucciones toman dos valores de entrada, realizan la función matemática y colocan el resultado en un lugar de memoria asignado.

Por ejemplo, las instrucciones ADD y SUB toman un par de valores de entrada, los añaden o los restan y colocan el resultado en el destino especificado. Si el resultado de la operación excede el valor permitido, un bit de overflow o underflow se establece.

La información general siguiente se aplica a las instrucciones matemáticas:

- **La Fuente** es la(s) dirección(es) del(los) valor(es) en que se realiza una operación matemática, lógica o de movimiento. Esto puede ser direcciones de palabra o constantes de programa. Una instrucción que tiene dos operandos de fuente no acepta constantes de programa en ambos operandos.

- **El destino** es la dirección de resultado de la operación. Los enteros con signo se almacenan de forma complementaria de dos y se aplican a los parámetros de fuente y destino.

B.3.1 Añadir (ADD)

La instrucción ADD se usa para añadir un valor (fuente A) a otro valor (fuente B) y coloque el resultado en el destino. Ver Tabla N° B.12.

Tabla N° B.12 Actualizaciones de bits de Instrucción ADD

Con este Bit:	El procesador:
Acarreo (C)	Se establece si el acarreo es generado; sino, se restablece (entero). Se pone a cero para el punto (coma) flotante.
Overflow (V)	Se establece si overflow es detectado en el destino; en caso contrario, se restablece. Durante overflow, el indicador de error menor también se establece. Para el punto (coma) flotante, el valor de overflow se coloca en el destino. Para un entero, el valor '-32,768' o '32,767' se coloca en el destino.
Cero (Z)	Se establece si el resultado es cero; en caso contrario, se restablece.
Signo (S)	Se establece si el resultado es negativo; en caso contrario, se restablece.

B.3.2 Restar (SUB)

La instrucción SUB se usa para restar un valor (fuente B) del otro (fuente A) y coloque el resultado en el destino. Ver Tabla N° B.13.

Tabla N° B.13 Actualizaciones de bits de Instrucción SUB

Con este Bit:	El procesador:
Acarreo (C)	Se establece si el acarreo es generado; en forma contrario, se restablece (entero). Se pone a cero para el punto (coma) flotante.
Overflow (V)	Se establece si es underflow; en caso contrario, se restablece. Durante underflow el indicador de error menor también se establece. Para el punto (coma) flotante, el valor de overflow se coloca en el destino. Para un entero, el valor '-32,768' o '32,767' se coloca en el destino.
Cero (Z)	Se establece si el resultado es cero; en caso contrario, se restablece.
Signo (S)	Se establece si el resultado es negativo; en caso contrario, se restablece.

B.3.3 Multiplicar (MUL)

La instrucción MUL se usa para multiplicar un valor (fuente A) por el otro (fuente B) y coloque el resultado en el destino. Ver Tabla N° B.14.

Tabla N° B.14 Actualizaciones de bits de Instrucción MUL

Con este Bit:	El procesador:
Acarreo (C)	Siempre se restablece.
Overflow (V)	Se establece si el overflow se detecta en el destino; en caso contrario, se restablece. Durante overflow, el indicador de error menor también se establece. El valor '-32,768' o '32,767' se coloca en el destino.
Cero (Z)	Se establece si el resultado es cero; en caso contrario, se restablece.
Signo (S)	Se establece si el resultado es negativo; en caso contrario, se restablece.

B.3.4 Dividir (DIV)

La instrucción DIV se usa para dividir un valor (fuente A) entre otro (fuente B). El cociente redondeado se coloca a su vez en el destino. El cociente no redondeado se almacena en la palabra más significativa del registro matemático. El resto se coloca en la palabra menos significativa del registro matemático. Ver Tabla N° B.15.

Tabla N° B.15 Actualizaciones de bits de Instrucción DIV

Con este Bit:	El procesador:
Acarreo (C)	Siempre se restablece.
Overflow (V)	Se establece si la división entre cero u overflow se detecta en el destino; en caso contrario, se restablece. Durante el overflow, el indicador de error menor también se establece. El valor '32,767' se coloca en el destino. Para los destinos de punto (coma) flotante, el resultado de overflow permanece en el destino.
Cero (Z)	Se establece si el resultado es cero; si no, se restablece; no definido si overflow esta establecido.
Signo (S)	Se establece si el resultado es negativo; si no, se restablece; no definido si overflow esta establecido.

B.3.5 División doble (DDV)

El contenido de 32 bits del registro matemático se divide entre el valor de fuente de 16 bits y el cociente redondeado se coloca en el destino. Si el residuo es 0.5 o mayor se redondea el destino. Típicamente esta instrucción sigue una instrucción MUL que crea un resultado de 32 bits. Ver Tabla N° B.16.

Tabla N° B.16 Actualizaciones de bits de Instrucción DDV

Con este Bit:	El procesador:
Acarreo (C)	Siempre se restablece.
Overflow (V)	Se establece si es división en cero o si el resultado es mayor de 32,767 o menor de –32,768; en caso contrario se restablece. Durante el overflow, también se establece el indicador de error menor. El valor 32,767 se coloca en el destino.

Cero (Z)	Se establece si el resultado es cero; en caso contrario, se restablece.
Signo (S)	Se establece si el resultado es negativo; en caso contrario, se restablece; no definido si el overflow esta establecido.

B.3.6 Borrar (CLR)

La instrucción CLR se usa para poner a cero el valor de destino de una palabra. Ver Tabla N° B.17.

Tabla N° B.17 Actualizaciones de bits de Instrucción CLR

Con este Bit:	El procesador:
Acarreo (C)	Siempre se restablece.
Overflow (V)	Siempre se restablece.
Cero (Z)	Siempre se restablece.
Signo (S)	Siempre se restablece.

B.3.7 Raíz Cuadrada (SQR)

Cuando esta instrucción es evaluada como verdadera, la raíz cuadrada del valor absoluto de la fuente es calculada y el resultado redondeado se coloca en el destino.

La instrucción calcula la raíz cuadrada de un numero negativo sin overflow ni fallos. En las aplicaciones donde el valor de fuente puede ser negativo, use una instrucción de comparación para evaluar el valor de fuente para determinar si el destino puede ser invalido. Ver Tabla N° B.18.

Tabla N° B.18 Actualizaciones de bits de Instrucción SQR

Con este Bit:	El procesador:
Acarreo (C)	Es reservado (entero). Para el punto (coma) flotante, siempre esta puesto a cero.
Overflow (V)	Siempre se restablece.
Cero (Z)	Se establece cuando el valor de destino es cero.
Signo (S)	Siempre se restablece.

B.3.8 Cómo escalar con parámetros (SCP)

La instrucción SCP se usa para producir un valor de salida escalado que tiene una relación lineal entre los valores de entrada y escalados. Esta instrucción tiene capacidad para valores de entero y punto (coma) flotante, ver Tabla N° B.19. Use la formula siguiente para convertir los datos de entrada analógicos en unidades de ingeniería:

$$y = mx + b \quad \text{..... (B.1)}$$

Donde:

y = salida escalada
m = pendiente (escala max. – escala min.) / (entrada max. – entrada min.)
x = valor de entrada
b = offset (intersección y) = escala min. – (estrada min. * inclinación)

La entrada mínima, la entrada máxima, escala mínima y escala máxima se usan para determinar los valores de inclinación y offset. El valor de entrada puede salir de los limites de entrada especificados sin requerir la puesta en orden. Por ejemplo, el valor de salida con escala no se encontrara necesariamente fijado entre los valores mínimos y máximos escalados.

Para poder programar la instrucción SCP se introduce los parámetros siguientes:

- El valor de entrada, puede ser una dirección de palabra o una dirección de elementos de datos de punto (flotante).
- Los valores mínimos y máximos de entrada, determinan el rango de datos que aparece en el parámetro de valor de entrada. El valor puede ser una dirección de palabra una constante de entero, elemento de datos de punto (coma) flotante o una constante de punto (coma) flotante.
- Los valores mínimos y máximos escalados, determinan el rango de datos que aparecen en el parámetro de salida con escala. El valor pude ser una dirección de palabra, una

constante de entero, elementos de datos de punto (coma) flotante o una constante de punto (coma) flotante.

- El valor de salida escalado, puede ser una dirección de palabra o una dirección de elementos de punto (coma) flotante.

Tabla N° B.19 Actualizaciones de bits de Instrucción SCP

Con este Bit:	El procesador:
Acarreo (C)	Siempre se restablece.
Overflow (V)	Se establece si el overflow es generado o si una entrada sin capacidad se detecta; si no, se restablece.
Cero (Z)	Se establece cuando el valor de destino es cero; si no, se restablece.
Signo (S)	Se establece cuando el valor de destino es negativo; si no, se restablece.

B.3.9 Escala de datos (SCL)

Cuando esta instrucción es verdadera, el valor en la dirección de fuente se multiplica por el valor del régimen. El resultado redondeado se añade al valor de offset y se coloca en el destino. A veces el término régimen significa pendiente. La función de régimen se limita al rango –3.2768 a 3.2767. Por ejemplo, -32768/10000 a +32767/10000. Ver Tabla N°B.20

El valor para los parámetros siguientes es entre –32,768 a 32,767. Para poder programar la instrucción SCL se introduce lo siguiente:

- La fuente, es una dirección de palabra
- El régimen, (o pendiente) es el valor positivo o negativo que usted introduce dividido entre 10,000. Puede usar una constante de programa o una dirección de palabra.
- El offset, puede ser una constante de programa o una dirección de palabra.

Tabla N° B.20 Actualizaciones de bits de Instrucción SCL

Con este Bit:	El procesador:
Acarreo (C)	Es reservado
Overflow (V)	Se establece si un overflow se detecta; en caso contrario, se restablece. La presencia de un overflow se verifica antes y después de la aplicación del valor de offset

Cero (Z)	Se establece cuando el valor de destino es cero
Signo (S)	Se establece si el valor de destino es negativo; en caso contrario, se restablece

B.3.10 Absoluto (ABS)

La instrucción ABS se usa para calcular el valor absoluto de la fuente y colocar el resultado en el destino (ver Tabla N° B.21). Esta instrucción tiene capacidad para los valores de entero y punto (coma) flotante. Para poder programar la instrucción ABS se introduce los parámetros siguientes:

- La fuente, puede ser una dirección de palabra, una constante de entero, elemento de datos de punto (coma) flotante o una constante de punto (coma) flotante.
- El destino, sólo puede ser una dirección de palabra o un elemento de datos de punto (coma) flotante.

Tabla N° B.21 Actualizaciones de bits de Instrucción ABS

Con este Bit:	**El procesador:**
Acarreo (C)	Siempre se restablece.
Overflow (V)	Siempre se restablece con un valor de punto (coma) flotante; se establece si la entrada es '-32,768' (valor de entero).
Cero (Z)	Se establece cuando el valor de destino es cero; en caso contrario, se restablece.
Signo (S)	Siempre se restablece.

B.3.11 Calcular (CPT)

La instrucción CPT efectúa operaciones de copiado, aritméticas, lógicas y conversión. La operación se define en la expresión y el resultado se escribe en el destino. El CPT usa funciones para operar en uno o más valores en la expresión para efectuar operaciones tales como:

- Convertir de un formato de número a otro
- Manejar los números
- Efectuar funciones trigonométricas

El tiempo de ejecución de una instrucción CPT es mayor que el de una sola operación aritmética y usa más palabras de instrucción. Ver Tabla N° B.22.

Para poder programar la instrucción CPT se introduce los parámetros siguientes:

- El destino, puede ser una dirección de palabra o la dirección de un elemento de datos de punto (coma) flotante.
- La expresión, es cero o más líneas, con hasta 28 caracteres por línea, hasta 255 caracteres.

Tabla N° B.22 Actualizaciones de bits de Instrucción CPT

Con este Bit:	El procesador:
Acarreo (C)	Se establece según el resultado de la última instrucción en la expresión
Overflow (V)	Se establece cuando un overflow ocurre durante la evaluación de la expresión.
Cero (Z)	Se establece según el resultado de la última instrucción en la expresión.
Signo (S)	Se establece según el resultado de la última instrucción en la expresión.

B.3.12 Intercambio (SWP)

La instrucción SWP se encarga de intercambiar los bytes bajos y altos de un número de palabras especificado en un archivo de bit, entero, ASCII o cadena.

Para poder programar la instrucción SWP se introduce los parámetros siguientes:

- La fuente, sólo puede ser una dirección de palabra indexada.
- La longitud, es una referencia al número de palabras que van a intercambiarse, pese al tipo de archivo. La dirección se limita a constantes de entero. Para los archivos de tipo bit, entero y ASCII, el rango de longitud es de 1 a 128. Para el archivo de tipo cadena, el rango de longitud es de 1 a 41. Esta instrucción se restringe a un solo elemento de cadena y no puede cruzar un límite de elemento de cadena.

B.3.13 Arco seno (ASN)

La instrucción ASN se usa para tomar el arco seno de un número (fuente en radianes) y almacenar el resultado (en radianes) en el destino. La fuente debe ser mayor o igual que –1 y menor o igual que 1. El valor resultante en el destino siempre es mayor o igual que $-\pi/2$ y menor o igual que $\pi/2$ ($\pi = 3.141592$). Ver Tabla N° B.23.

Tabla N° B.23 Actualizaciones de bits de Instrucción ASN

Con este Bit:	El procesador:
Acarreo (C)	Siempre se restablece.
Overflow (V)	Se establece si un overflow es generado o una entrada sin capacidad se detecta; sino, se restablece.
Cero (Z)	Se establece si el resultado es cero; en caso contrario, se restablece.
Signo (S)	Se establece si el resultado es negativo; en caso contrario, se restablece.

B.3.14 Arco coseno (ACS)

La instrucción ACS se usa para tomar el arco coseno de un número (fuente en radianes) y almacenar el resultado (en radianes) en el destino. La fuente debe ser mayor o igual que –1 y menor o igual que 1. El valor resultante en el destino siempre es mayor o igual que 0 y menor o igual que π ($\pi = 3.141592$). Ver Tabla N° B.24.

Tabla N° B.24 Actualizaciones de bits de Instrucción ACS

Con este Bit:	El procesador:
Acarreo (C)	Siempre se restablece
Overflow (V)	Se establece si un overflow es generado o una entrada sin capacidad se detecta; en caso contrario, se restablece
Cero (Z)	Se establece si el resultado es cero; en caso contrario, se restablece
Signo (S)	Siempre se restablece

B.3.15 Arco tangente (ATN)

La instrucción ATN se usa para tomar el arco tangente de un número (fuente) y almacenar el resultado (en radianes) en el destino. El valor resultante en el destino siempre es mayor o igual que -π/2 y menor o igual que π/2 (π = 3.141592). Ver Tabla N° B.25.

Tabla N° B.25 Actualizaciones de bits de Instrucción ATN

Con este Bit:	El procesador:
Acarreo (C)	Siempre se restablece.
Overflow (V)	Se establece si un overflow es generado o una entrada sin capacidad se detecta; en caso contrario, se restablece.
Cero (Z)	Se establece si el resultado es cero; en caso contrario, se restablece.
Signo (S)	Se establece si el resultado es negativo; en caso contrario, se restablece.

B.3.16 Coseno (COS)

La instrucción COS se usa para tomar el coseno de un número (fuente en radianes) y almacenar el resultado (en radianes) en el destino. La fuente debe ser mayor o igual que –205887.4 y menor o igual que 205887.4.

La óptima exactitud se obtiene cuando la fuente es mayor que -2π y menor que 2π (π = 3.141592). El valor resultante en el destino siempre es mayor o igual que –1 y menor o igual que 1. Ver Tabla N° B.26.

Tabla N° B.26 Actualizaciones de bits de Instrucción COS

Con este Bit:	El procesador:
Acarreo (C)	Siempre se restablece.
Overflow (V)	Se establece si un overflow es generado o una entrada sin capacidad se detecta; en caso contrario, se restablece.
Cero (Z)	Se establece si el resultado es cero; en caso contrario, se restablece.
Signo (S)	Siempre se restablece.

B.3.17 Logaritmo natural (LN)

La instrucción LN se usa para tomar logaritmo natural del valor en la fuente y almacenar el resultado en el destino. La fuente debe ser mayor que cero. El valor resultante en el destino siempre es mayor o igual que –87.33654 y menor o igual que 88.72284. Ver Tabla N° B.27

Tabla N° B.27 Actualizaciones de bits de Instrucción LN

Con este Bit:	El procesador:
Acarreo (C)	Siempre se restablece.
Overflow (V)	Se establece si un overflow es generado o una entrada sin capacidad se detecta; en caso contrario, se restablece.
Cero (Z)	Se establece si el resultado es cero; en caso contrario, se restablece.
Signo (S)	Siempre se restablece.

B.3.18 Logaritmo a la base 10 (LOG)

La instrucción LOG se usa para tomar el logaritmo de base 10 del valor en la fuente y almacenar el resultado en el destino. La fuente debe ser mayor que cero. El valor resultante en el destino siempre es mayor o igual que –37.92978 y menor o igual que 38.53184. Ver Tabla N° B.28.

Tabla N° B.28 Actualizaciones de bits de Instrucción LOG

Con este Bit:	El procesador:
Acarreo (C)	Siempre se restablece.
Overflow (V)	Se establece si un overflow es generado o una entrada sin capacidad se detecta; en caso contrario, se restablece.
Cero (Z)	Se establece si el resultado es cero; en caso contrario, se restablece.
Signo (S)	Siempre se restablece.

B.3.19 Seno (SIN)

La instrucción SIN se usa para tomar el seno de un número (fuente en radianes) y almacenar el resultado en el destino. La fuente debe ser mayor o igual que –205887.4 y menor o igual que 205887.4. La óptima exactitud se obtiene cuando la fuente es mayor que

-2π y menor que 2π (π = 3.141592). El valor resultante en el destino siempre es mayor o igual que –1 y menor o igual que 1. Ver Tabla N° B.29.

Tabla N° B.29 Actualizaciones de bits de Instrucción SIN

Con este Bit:	El procesador:
Acarreo (C)	Siempre se restablece.
Overflow (V)	Se establece si un overflow es generado o una entrada sin capacidad se detecta; en caso contrario, se restablece.
Cero (Z)	Se establece si el resultado es cero; en caso contrario, se restablece.
Signo (S)	Siempre se restablece.

B.3.20 Tangente (TAN)

La instrucción TAN se usa para tomar la tangente de un número (fuente en radianes) y almacenar el resultado en el destino. El valor de la fuente debe ser mayor o igual que –102943.7 y menor o igual que 102943.7. La óptima exactitud se obtiene cuando la fuente es mayor que -2π y menor que 2π (π = 3.141592). El valor resultante en el destino es un número real o infinito. Ver Tabla N° B.30.

Tabla N° B.30 Actualizaciones de bits de Instrucción TAN

Con este Bit:	El procesador:
Acarreo (C)	Siempre se restablece.
Overflow (V)	Se establece si un overflow es generado o una entrada sin capacidad se detecta; en caso contrario, se restablece.
Cero (Z)	Se establece si el resultado es cero; en caso contrario, se restablece.
Signo (S)	Siempre se restablece.

B.3.21 X a la potencia de Y (XPY)

La instrucción XPY se usa para elevar un valor (fuente A) a una potencia (fuente B) y almacenar el resultado en el destino. Si el valor en la fuente A es negativo, la exponente (fuente B) debe ser un número entero. Si no es un número entero, el bit de overflow se establece y el valor absoluto de la base se usa en el cálculo. Ver Tabla N° B.31.

La instrucción XPY usa el algoritmo siguiente:

$$XPY = 2 ** (Y * Log\ (X)) \quad \text{..... (B.2)}$$

Si cualquiera de las operaciones intermedias en este algoritmo provoca un overflow, se establece el bit de estado de overflow aritmético.

Tabla N° B.31 Actualizaciones de bits de Instrucción XPY

Con este Bit:	El procesador:
Acarreo (C)	Siempre se restablece.
Overflow (V)	Se establece si un overflow es generado o una entrada sin capacidad se detecta; en caso contrario, se restablece.
Cero (Z)	Se establece si el resultado es cero; en caso contrario, se restablece.
Signo (S)	Siempre se restablece.

B.4 INSTRUCCIONES DE MANEJO DE DATOS

A continuación se conocerán las instrucciones de manejo de datos y se explicarán cómo funcionan en su programa de aplicación. Cada una de las instrucciones incluye información acerca de:

- Cómo aparece el símbolo de instrucción
- Cómo usar la instrucción

Para mayor detalle ver Tabla N° B.32.

Tabla N° B.32 Instrucciones de manejo de datos

Instrucción		Propósito
Mnemónico	Nombre	
TOD	Convertir a BCD	Convierte el valor de fuente de entero en el formato BCD y lo almacena en el destino.
FRD	Convertir desde BCD	Convierte el valor de fuente BCD en un entero y lo almacena en el destino.
DEG	Convertir radian a grados	Convierte radianes (fuente) en grados y almacena el resultado en el destino.

RAD	Convertir grados a radianes	Convierte grados (fuente) en radianes y almacena el resultado en el destino.
DCD	Decodificar 4 a 1 de 16	Decodifica un valor de 4 bits (0-15) activando el bit correspondiente en el destino de 16 bits.
ENC	Codificar 1 de 16 a 4	Codifica una fuente de 16 bits a un valor de 4 bits. Busca la fuente desde el bit mínimo al bit máximo y busca el primer bit establecido. La posición de bit correspondiente se escribe en el destino como entero.
COP y FLL	Copiar archivo y llenar archivo	La instrucción COP copia datos del archivo de fuente al archivo de destino. La instrucción FLL carga un valor de fuente en cada posición al archivo de destino.
MOV	Mover	Mueve el valor de fuente al destino.
MVM	Mover con máscara	Mueve los datos de un lugar de fuente a una porción seleccionada del destino.
AND	Y	Realiza una operación Y por bit.
OR	O	Realiza una operación de O inclusiva por bit.
XOR	O exclusivo	Realiza una operación de O exclusiva por bit.
NOT	No	Realiza una operación NO.
NEG	Negar	Cambia el signo de la fuente y lo almacena en el destino.
FFL y FFU	Carga FIFO y descarga FIFO	La instrucción FFL carga una palabra en una pila FIFO en transiciones de falso a verdadero sucesivas. El FFU hace la función de descargar.
LFL y LFU	Carga LIFO y descarga LIFO	La instrucción LFL carga una palabra en una pila LIFO en transiciones de falso a verdadero sucesivas. El LFU hace la función de descargar.

B.4.1 Convertir en BCD (TOD)

La instrucción BCD se usa para convertir enteros de 16 bits en valores BCD. Si el valor de entero que introduce es negativo, el valor absoluto del número se usa para la conversión. Ver Tabla N° B.33.

Tabla N° B.33 Actualizaciones de bits de Instrucción BCD

Con este Bit:	**El procesador:**
Acarreo (C)	Siempre se restablece.
Overflow (V)	Se establece si el resultado BCD es mayor de 9999. El overflow resulta en un error menor
Cero (Z)	Se establece si el valor de destino es cero.
Signo (S)	Se establece si la palabra de fuente es negativa; en caso contrario, se restablece.

B.4.2 Convertir de BCD (FRD)

La instrucción FRD se usa para convertir los valores BCD en valores enteros. El parámetro de fuente puede ser una dirección de palabra en cualquier archivo de datos o un registro matemático. Ver Tabla N° B.34.

Tabla N° B.34 Actualizaciones de bits de Instrucción FRD

Con este Bit:	El procesador:
Acarreo (C)	Siempre se restablece.
Overflow (V)	Se establece si un valor que no sea BCD se contiene en la fuente o si el valor que va a convertir es mayor de 32,767; en caso contrario, se restablece. El overflow resulta en un error menor.
Cero (Z)	Se establece si el valor de destino es cero.
Signo (S)	Siempre se restablece.

Siempre se recomienda que se proporcione filtro de lógica de escalera para todos los dispositivos de entrada BCD antes de realizar la instrucción FRD. La mínima diferencia en el retardo del filtro de entrada de punto a punto puede provocar un overflow de la instrucción FRD debido a la conversión de un dígito que no sea BCD.

B.4.3 Radianes en grados (DEG)

La instrucción DEG se usa para convertir los radianes (fuente) en grados y almacenar el resultado en el destino (ver Tabla N° B.35), para lo cual se aplica la fórmula siguiente:

$$\text{Fuente} \times (180/\pi) \quad \text{..... (B.3)}$$

Donde $\pi = 3.141592$

Tabla N° B.35 Actualizaciones de bits de Instrucción DEG

Con este Bit:	El procesador:
Acarreo (C)	Siempre se restablece.
Overflow (V)	Se establece si un overflow es generado o si una entrada sin capacidad se detecta; en caso contrario, se restablece.
Cero (Z)	Se establece si el resultado es cero; en caso contrario, se restablece.

Signo (S)	Se establece si el resultado es negativo; en caso contrario, se restablece.

B.4.4 Grados en radianes (RAD)

La instrucción RAD se usa para convertir los grados (fuente) en radianes y almacenar el resultado en el destino (ver Tabla N° B.36), para lo cual se aplica la fórmula siguiente:

$$\text{Fuente} \times (\pi/180) \quad \text{..... (B.4)}$$

Donde $\pi = 3.141592$

Tabla N° B.36 Actualizaciones de bits de Instrucción RAD

Con este Bit:	**El procesador:**
Acarreo (C)	Siempre se restablece.
Overflow (V)	Se establece si un overflow es generado o si una entrada sin capacidad se detecta; en caso contrario, se restablece.
Cero (Z)	Se establece si el resultado es cero; en caso contrario, se restablece.
Signo (S)	Se establece si el resultado es negativo; en caso contrario, se restablece.

B.4.5 Decodificar 4 a 1 de 16 (DCD)

Una vez ejecutada, esta instrucción DCD establece un bit de la palabra de destino. El bit que se activa depende del valor de los cuatro primeros bits de la palabra de fuente. Ver fig. B.4.

	Fuente					Destino														
Bit	15-04	03	02	01	00	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
	x	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
	x	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
	x	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
	x	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
	x	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0
	x	0	1	0	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
	x	0	1	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
	x	0	1	1	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0
	x	1	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
	x	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
	x	1	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0
	x	1	0	1	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
	x	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
	x	1	1	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
	x	1	1	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	x	1	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig. B.4 Representación binaria de la Instrucción DCD

B.4.6 Codificar 1 de 16 a 4 (ENC)

Cuando el renglón es verdadero, esta instrucción de salida busca la fuente desde el bit mínimo hasta el bit máximo y busca el primer bit establecido (ver fig. B.5). La posición de bit correspondiente se escribe al destino como entero según se muestra en la tabla siguiente:

	Fuente																Destino				
Bit	15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00	15-04	03	02	01	00
	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	x	0	0	0	0
	x	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	x	0	0	0	1
	x	x	x	x	x	x	x	x	x	x	x	x	x	1	0	0	x	0	0	1	0
	x	x	x	x	x	x	x	x	x	x	x	x	1	0	0	0	x	0	0	1	1
	x	x	x	x	x	x	x	x	x	x	x	1	0	0	0	0	x	0	1	0	0
	x	x	x	x	x	x	x	x	x	x	1	0	0	0	0	0	x	0	1	0	1
	x	x	x	x	x	x	x	x	x	1	0	0	0	0	0	0	x	0	1	1	0
	x	x	x	x	x	x	x	x	1	0	0	0	0	0	0	0	x	0	1	1	1
	x	x	x	x	x	x	x	1	0	0	0	0	0	0	0	0	x	1	0	0	0
	x	x	x	x	x	x	1	0	0	0	0	0	0	0	0	0	x	1	0	0	1
	x	x	x	x	x	1	0	0	0	0	0	0	0	0	0	0	x	1	0	1	0
	x	x	x	x	1	0	0	0	0	0	0	0	0	0	0	0	x	1	0	1	1
	x	x	x	1	0	0	0	0	0	0	0	0	0	0	0	0	x	1	1	0	0
	x	x	1	0	0	0	0	0	0	0	0	0	0	0	0	0	x	1	1	0	1
	x	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	x	1	1	1	0
	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	x	1	1	1	1

Fig. B.5 Representación binaria de la Instrucción ENC

Los bits de estado aritmético se encuentran en la palabra 0, bits 0-3 en el archivo de estado del controlador. Después de la ejecución de una instrucción, los bits de estado aritmético en el archivo de estado se actualizan. Ver Tabla N° B.37.

Tabla N° B.37 Actualizaciones de bits de Instrucción ENC

Con este Bit:	El procesador:
Acarreo (C)	Siempre se restablece.
Overflow (V)	Se establece si más de un bit se establece; en caso contrario, se restablece.
Cero (Z)	Se establece si el valor de destino es cero.
Signo (S)	Siempre se restablece.

B.4.7 Copia de archivo (COP) y llenado de archivo (FLL)

El archivo de tipo destino determina el número de palabras que una instrucción transfiere. Por ejemplo, si el archivo de tipo destino es un contador y el archivo de tipo fuente es un entero, se transfieren tres palabras de entero por cada elemento en el archivo de tipo contador.

La instrucción COP copia bloques de datos de un lugar a otro. No usa bits de estado. Si usted necesita un bit de habilitación, programe una instrucción de salida (OTE) en paralelo usando un bit interno como la dirección de salida.

La instrucción FLL carga elementos de un archivo con una constante de programa o valor de una dirección de elemento. La instrucción llena las palabras de un archivo con un valor de fuente. No usa bits de estado. Si se necesita un bit de habilitación, programe una salida en paralelo que use una dirección de almacenamiento

B.4.8 Mover (MOV)

Esta instrucción de salida mueve el valor de fuente al lugar de destino. Siempre que el renglón permanezca verdadero, la instrucción mueve los datos durante cada escán. Si se desea mover una palabra de datos sin afectar los indicadores matemáticos, usar una instrucción de copiar (COP) con una longitud de 1 palabra en vez de la instrucción MOV. Ver Tabla N° B.38.

Tabla N° B.38 Actualizaciones de bits de Instrucción MOV

Con este Bit:	El procesador:
Acarreo (C)	Siempre se restablece.
Overflow (V)	Siempre se restablece.
Cero (Z)	Se establece si el resultado es cero; en caso contrario, se restablece.
Signo (S)	Se establece si el resultado es negativo (el bit más significante establecido); en caso contrario, se restablece.

B.4.9 Mover con máscara (MVM)

La instrucción MVM es una instrucción de palabra que mueve datos de un lugar de fuente a un destino y permite que porciones de los datos de destino estén enmascarados por una palabra separada. Siempre que el renglón permanezca verdadero, la instrucción mueve los datos durante cada escán. Ver Tabla N° B.39.

Si se desea mover una palabra de datos sin afectar los indicadores matemáticos, usar una instrucción de copiar (COP) con una longitud de 1 palabra en vez de la instrucción MOV.

Tabla N° B.39 Actualizaciones de bits de Instrucción MVM

Con este Bit:	El procesador:
Acarreo (C)	Siempre se restablece.
Overflow (V)	Siempre se restablece.
Cero (Z)	Se establece si el resultado es cero; en caso contrario, se restablece.
Signo (S)	Se establece si el resultado es negativo; en caso contrario, se restablece.

B.4.10 Y (AND)

El valor en la fuente A recibe la instrucción AND bit por bit con el valor en la fuente B y luego se almacena en el destino. Ver Tabla N° B.40 y B.41.

Las fuentes A y B pueden ser una dirección de palabra o una constante; sin embargo, ambas fuentes no pueden ser una constante. El destino debe ser una dirección de palabra.

Tabla N° B.40 Compuerta Lógica AND

Destino = A y B		
A	**B**	**Destino**
0	0	0
0	1	0
1	0	0
1	1	1

Tabla N° B.41 Actualizaciones de bits de Instrucción AND

Con este Bit:	**El procesador:**
Acarreo (C)	Siempre se restablece.
Overflow (V)	Siempre se restablece.
Cero (Z)	Se establece si el resultado es cero; en caso contrario, se restablece.
Signo (S)	Se establece si el bit más significativo esta establecido; en caso contrario, se restablece.

B.4.11 O (OR)

El valor en la fuente A recibe la instrucción OR bit por bit con el valor en la fuente B y luego se almacena en el destino. Las fuentes A y B pueden ser una dirección de palabra o una constante; sin embargo, ambas fuentes no pueden ser una constante. El destino debe ser una dirección de palabra. Ver Tabla N° B.42 y B.43.

Tabla N° B.42 Compuerta Lógica OR

Destino = A o B		
A	**B**	**Destino**
0	0	0
0	1	1
1	0	1
1	1	1

Tabla N° B.43 Actualizaciones de bits de Instrucción OR

Con este Bit:	**El procesador:**
Acarreo (C)	Siempre se restablece.
Overflow (V)	Siempre se restablece.
Cero (Z)	Se establece si el resultado es cero; en caso contrario, se restablece.

Signo (S)	Se establece si el resultado es negativo (el bit más significativo está establecido); en caso contrario, se restablece.

B.4.12 O exclusivo (XOR)

El valor en la fuente A recibe la instrucción de O exclusivo bit por bit con el valor en la fuente B y luego se almacena en el destino. Las fuentes A y B pueden ser una dirección de palabra o una constante; sin embargo, ambas fuentes no pueden ser una constante. El destino debe ser una dirección de palabra. Ver Tabla N° B.44 y B.45.

Tabla N° B.44 Compuerta Lógica XOR

Destino = A X o B		
A	**B**	**Destino**
0	0	0
0	1	1
1	0	1
1	1	0

Tabla N° B.45 Actualizaciones de bits de Instrucción XOR

Con este Bit:	**El procesador:**
Acarreo (C)	Siempre se restablece.
Overflow (V)	Siempre se restablece.
Cero (Z)	Se establece si el resultado es cero; en caso contrario, se restablece.
Signo (S)	Se establece si el resultado es negativo (el bit más significativo está establecido); en caso contrario, se restablece.

B.4.13 No (NOT)

El valor de fuente recibe la instrucción NOT bit por bit y luego se almacena en el destino (complemento de uno). La fuente y el destino deben ser direcciones de palabra. Para mayor detalle ver Tabla N° B.46 y B.47.

Tabla N° B.46 Compuerta Lógica NOT

Destino = NOT (A)	
A	**Destino**
0	1
1	0

Tabla N° B.47 Actualizaciones de bits de Instrucción NOT

Con este Bit:	**El procesador:**
Acarreo (C)	Siempre se restablece.
Overflow (V)	Siempre se restablece.
Cero (Z)	Se establece si el resultado es cero; en caso contrario, se restablece.
Signo (S)	Se establece si el resultado es negativo (el bit más significativo está establecido); en caso contrario, se restablece.

B.4.14 Negar (NEG)

La instrucción NEG se usa para cambiar el signo de la fuente y luego colóquelo en el destino. El destino contiene el complemento de dos de la fuente. Por ejemplo, si la fuente es 5, el destino sería –5. La fuente y el destino deben ser direcciones de palabra. Para mayor detalle ver Tabla N° B.48.

Tabla N° B.48 Actualizaciones de bits de Instrucción NEG

Con este Bit:	**El procesador:**
Acarreo (C)	Se pone a cero si es 0 u overflow; en caso contrario, se restablece.
Overflow (V)	Se establece si es un overflow; en caso contrario, se restablece. El overflow sólo ocurre si –32,768 es la fuente. Durante el overflow, el indicador de error menor también se establece. El valor 32,767 se coloca en el destino.
Cero (Z)	Se establece si el resultado es cero; en caso contrario, se restablece.
Signo (S)	Se establece si el resultado es negativo; en caso contrario, se restablece.

B.4.15 Carga FIFO (FFL) y descarga FIFO (FFU)

Las instrucciones FFL y FFU se usan conjuntamente. La instrucción FFL carga palabras en un archivo creado por el usuario que se llama una pila FIFO. La instrucción FFU descarga palabras de la pila FIFO en el mismo orden en que fueron cargadas.

La instrucción FFL/FFU carga/descarga un elemento a cada transición de falso a verdadero del renglón hasta que la pila sé llene/vacíe (34 elementos). Luego el procesador establece el bit de efectuado (DN) / vacío (EM).

B.4.16 Carga LIFO (LFL) y descarga LIFO (LFU)

Las instrucciones LFL y LFU se usan conjuntamente. La instrucción LFL carga palabras en un archivo creado por el usuario que se llama una pila LIFO. La instrucción LFU descarga palabras de la pila LIFO en el mismo orden en que fueron cargadas.

La instrucción LFL/LFU carga/descarga un elemento a cada transición de falso a verdadero del renglón hasta que la pila se llene/vacíe (34 elementos). Luego el procesador establece el bit de efectuado (DN) / vacío (EM).

B.5 INSTRUCCIÓN PROPORCIONAL INTEGRAL DERIVATIVA (PID)

Es una instrucción de salida que controla las características físicas tales como la temperatura, presión, nivel de líquido o régimen de caudal usando lazos de proceso. Esta instrucción PID normalmente controla un lazo cerrado usando entradas de un módulo de entrada analógico y proporcionando una salida a un módulo de salida analógico.

Para el control de temperatura se puede convertir la salida analógica a una salida activada / desactivada de tiempo proporcional para impulsar una unidad de calefacción o enfriamiento.

En la Fig. B.6 se puede observar el esquema de control de caudal / nivel de un fluído.

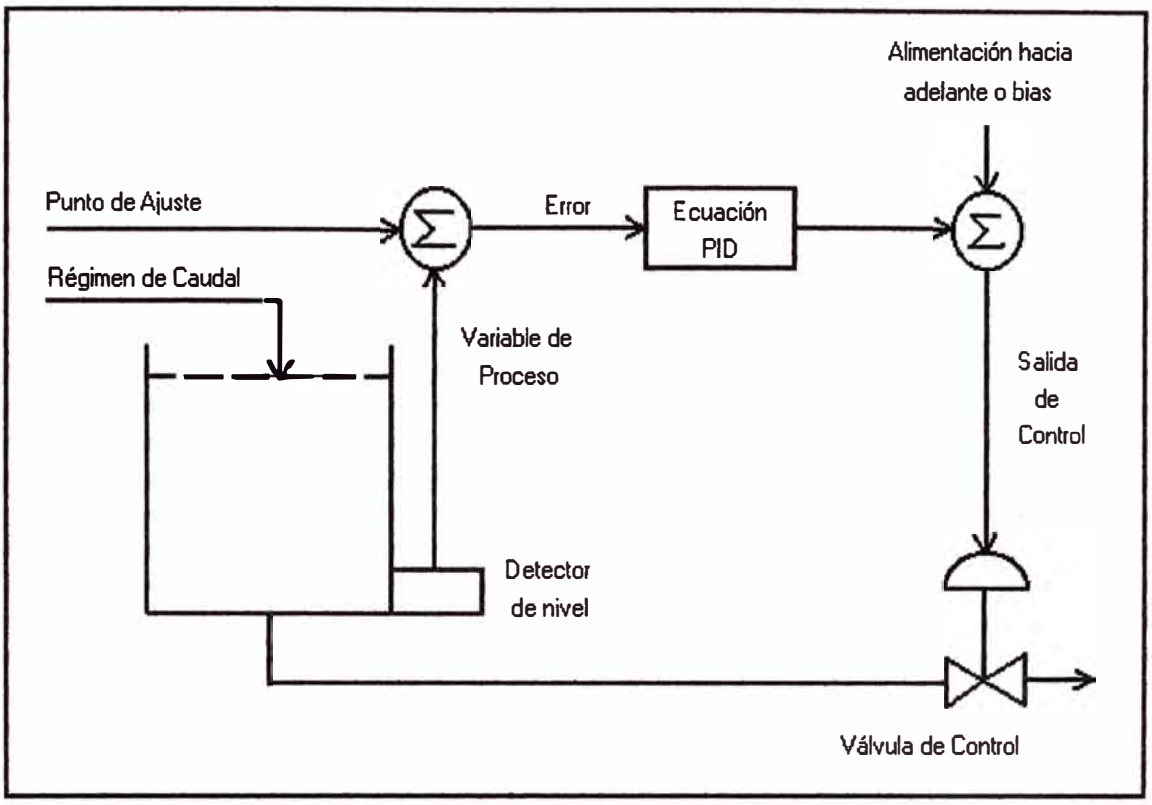

Fig. B.6 Esquema de control de caudal / nivel de un fluído

El control en lazo cerrado PID retiene una variable de proceso a un punto de ajuste deseado. La ecuación PID controla el proceso enviando una señal de salida a la válvula de control. Cuanto más grande sea el error entre el punto de ajuste y la entrada de variable del proceso (valor sensado) tanto más grande es la señal de salida y viceversa. Un valor adicional (alimentación hacia delante o bias) se puede añadir a la salida de control como offset. El resultado del cálculo PID (variable de control) impulsará la variable del proceso que controla hacia el punto de ajuste.

La instrucción PID usa el algoritmo siguiente:

$$\text{Salida} = Kc\,[(E) + (1/Ti) * \int(E)dt + Td * D(PV)/dt] + \text{bias} \quad \text{..... (B.5)}$$

Las constantes de ganancia estándar están determinadas por Kc, (1/Ti), Td.

Para conocer al detalle los parámetros del Controlador PID ver la Tabla N° B.49.

Tabla N° B.49 Parámetros del controlador PID

Término	Rango (bajo a alto)	Referencia
Ganancia de Controlador Kc	0.1 a 25.5 (adimensional) 0.01 a 327.67 (adimensional)	Proporcional
Término de restablecimiento 1/Ti	25.5 a 0.1 (minutos por repetición) 327.67 a 0.01 (minutos por repetición)	Integral
Término de régimen Td	0.01 a 2.55 (minutos) 0.01 a 327.67 (minutos)	Derivativa

El término régimen derivativo proporciona la uniformización por medio de un filtro de pasa bajo. La instrucción PID se coloca en un renglón sin lógica condicional, la salida permanece a su último valor cuando el renglón es falso, el término integral también se borra cuando el renglón es falso.

ANEXO C
VARIABLES DEL PROCESO

C.1 VARIABLES DEL PROCESO

Las variables globales que se darán a conocer a continuación, parte del Programa de Control de una Cortadora Lineal de diferentes dimensiones elaborada en lenguaje de Diagrama de Bloques de Función del software de programación Prosys para PLC, y su correcta determinación forman parte fundamental para la exitosa ejecución del programa realizado.

Por otra parte cabe señalar que estas variables son declaradas como globales para que puedan interactuar entre ellas en los sub-programas Cortador y Contador siendo actualizadas constantemente según el proceso lo requiera.

Se debe indicar la dirección asignada a cada una de estas variables que intervienen en el Proceso de Cortado de Tubos de diferentes dimensiones, el tipo de variable que se esta utilizando, en este caso en particular se utilizarán variables del tipo boolenaas (aquellas cuyos valores pueden ser "1" lógico o "0" lógico) para las variables de posición y activación de la cortadora; y del tipo byte o word (aquellas cuyos valores están representadas por un grupo de bits en código binario) para las variables CANTIDAD, CUENTA y LONGITUD. Ver Tabla N° C.1.

Tabla N° C.1 Variables Globales del Proceso "Cortadora Lineal de 3 dimensiones diferentes"

Nombre	Dirección	Tipo	Valor Inicial
Marcha	%I1.7	BOOL	
M	%M10.0	BOOL	
Paro	%I1.6	BOOL	
Emergencia	%I1.4	BOOL	TRUE
K1	%Q2.5	BOOL	
A0	%I0.2	BOOL	
A1	%I0.3	BOOL	
B1	%I0.5	BOOL	
C1	%I0.4	BOOL	
D0	%I0.1	BOOL	
D1	%I0.0	BOOL	
Sb	%I0.6	BOOL	TRUE
Y1	%Q2.1	BOOL	
Y2	%Q2.3	BOOL	
Y3	%Q2.2	BOOL	
Y4	%Q2.0	BOOL	
M1		SR	
M2		RS	
M3		SR	
M4		RS	
MM	%M10.2	BOOL	
LIBERA	%I1.5	BOOL	
C11		CTU	
CANTIDAD		INT	10
CUENTA		INT	
D11		R_TRIG	
C12		CTU	
PRESET1		INT	2
CUENTA1		INT	
C13		CTU	
PRESET2		INT	3
CUENTA2		INT	
LONGITUD		INT	3
X0	%M20.0	BOOL	
X1	%M20.1	BOOL	
X2	%M20.2	BOOL	
X3	%M20.3	BOOL	
Z1	%M20.4	BOOL	
Z2	%M20.5	BOOL	
T1		TON	
T2		TON	

BIBLIOGRAFIA

1. Ing. Raymundo Carranza Noriega, "Automatización, Tópicos de Instrumentación y Control", Pontificia Universidad Católica del Perú, 2005.

2. Ballina E., "Tendencias actuales del software en el campo de la Automatización", Perú, 1997.

3. Carranza R., "Instrumentación para Ingenieros de Procesos", Universidad Nacional del Callao - Perú, 1993.

4. Hernández P., "Hacia un Control Inteligente", Revista Ingeniería Química – Perú, 1994

5. ISA, "Basic Automatic Process Control", Instrumentation Video Series, 1993.

6. Ogata Katsuhiko, "Ingeniería de Control Moderno", Prentice Hall Hispanoamericana – México, 1985.

7. Arbildo A., "Aplicaciones de Control y Automatización usando PCs", Revista Electro - Electrónica PUCP – Perú, 1996.

8. Gugliandolo F., "Como seleccionar un motor eléctrico", Sección Electricidad y Electrónica Correo Central Perú, 1997

9. Deltalogic and Smart Software Solutions, "Prosys - Demo Version 2.04 for Simatic", Copyright 1995 – 1999.