



**UNIVERSIDAD NACIONAL AUTÓNOMA DE
MÉXICO**

FACULTAD DE ESTUDIOS SUPERIORES CUAUTITLÁN

**FUNCIONAMIENTO DE UN TRABLERO DE CONTROL
CON UN PLC ALLEN BRADLEY**

TESIS Y EXAMEN PROFESIONAL

**QUE PARA OBTENER EL TÍTULO DE:
INGENIERO MECÁNICO ELECTRICISTA**

**PRESENTA:
FRANCISCO JAVIER HERNANDEZ GARCIA**

ASESOR: M EN I BENJAMÍN CONTRERAS SANTACRUZ

CUAUTITLÁN IZCALLI, ESTADO DE MÉXICO 2015



Universidad Nacional
Autónoma de México



UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

Agradecimientos

*A mis padres pues ni con la riquezas mas grandes del mundo
podré pagarles.*

A FESC por darme valiosos conocimientos.

*A mis maestros, a todos mis profesores por ayudarme a
cumplir mi meta.*

*A mis inolvidables amigas y amigos de la carrera, los cuales
emprendimos caminos distintos para jamás volver a vernos
pero siempre recordarnos: Carmen, Edgar Alfredo,
Francisco, Fidel, Zenaido, Marco, Nery, Farias, ...*

*Un agradecimiento a las empresas de automatización en
México ABSA y Allen Bradley por haber facilitado el material
de esta tesis.*

TITULO

FUNCIONAMIENTO DE UN TABLERO DE CONTROL CON UN PLC ALLEN

BRADLEY

INDICE

RESUMEN.....	1
MARCO TEÓRICO.....	1
OBJETIVOS.....	2
HIPÓTESIS.....	2
METODOLOGÍA.....	3
RESULTADOS.....	3
ANÁLISIS DE RESULTADOS Y DISCUSIÓN.....	4
PRÓLOGO.....	5
INTRODUCCIÓN.....	6

CAPÍTULO 1 GENERALIDADES

1.1 BREVE HISTORIA DE LOS PLC.....	7
1.2 RELÉ LÓGICO PROGRAMABLE PLR.....	11
1.3 LA INGENIERÍA AUTOMÁTICA.....	13
1.4 COMPONENTES DE CONTROLADORES DE LÓGICA PROGRAMABLE.....	23

CAPÍTULO 2 PROGRAMACIÓN

2.1 DIAGRAMAS DISCRETOS.....	27
2.2 FUNCIONES DE LÓGICA.....	34
2.3 SISTEMAS NUMÉRICOS.....	36
2.4 LENGUAJE LADDER.....	39
2.5 PROGRAMACIÓN DE CONTACTOS.....	54
2.6 PROGRAMACIÓN DE FUNCIONES DE LÓGICA.....	56

CAPÍTULO 3 TABLERO DEL PLC

3.1 PLC.....	65
3.2 DELEGAR PUNTOS DE ENTRADA, SALIDA Y RELEVOS INTERNOS.....	66
3.3 DOCUMENTACIÓN DE PROGRAMAS.....	67
3.4 ENTRADAS Y SALIDAS DISCRETAS.....	68
3.5 ENTRADAS Y SALIDAS ANÁLOGAS.....	72
3.6 MÓDULOS DE ALAMBRADO PARA ENTRADAS Y SALIDAS.....	76
3.7 MULTIPLEXING.....	79
3.8 ARRANQUE Y PARO.....	80
3.9 DIAGRAMA DE ESCALERA.....	84

CAPÍTULO 4 PROGRAMACIÓN BÁSICA.

4.1 TIMERS.....	91
4.2 CONTADORES.....	98
4.3 CONFIGURACIÓN DE DRIVERS.....	104
4.4 PROGRAMA RS LOGIC 500.....	108
4.5 DIRECCIONAMIENTO EN RS LOGIC 500.....	112
4.6 SÍMBOLOS DE RS LOGIC 500.....	118
4.7 SECUENCIADOR.....	121

CAPÍTULO 5 FUNCIONES AVANZADAS

5.1 SISTEMAS DE COMUNICACIÓN	130
5.2 FUNCIONES DE COMPARACIÓN.....	131
5.3 FUNCIONES MATEMÁTICAS.....	137
5.4 FUNCIONES AVANZADAS DE LÓGICA.....	140
5.5 FUNCIONES AVANZADAS DE PROGRAMACIÓN.....	144
5.6 BOBINAS DE TIPO RETENTIVO.....	151
5.7 JUMPO TO LABEL.....	153

CAPÍTULO 6 PROGRAMACIÓN AVANZADA

6.1 SUBRUTINAS.....	156
6.2 FUNCIONES MOVE Y COPY.....	158
6.3 FUNCIÓN FILL.....	160
6.4 OPERACIONES MATEMÁTICAS.....	165
6.5 DIRECCIONAMIENTO INDIRECTO.....	168
6.6 FUNCIÓN BITSHIFT.....	171
6.7 COMO PROGRAMAR.....	176
6.8 MÁSCARAS.....	180
CONCLUSIONES.....	184

RESUMEN

El control lógico programable contiene instrucciones para diagramas de escalera en lógica combinacional y también para lógica secuencial, en el capítulo 1 se da una breve historia de los PLC, la ingeniería automática y de los componentes externos asociados al PLC, que pueden simular cualquier circuito de control eléctrico.

El capítulo 2 es representación de simbología de PLC, lenguaje de escalera, programación de contactos y programación de lógica. El capítulo 3 trata acerca de variables discretas y variables analógicas y su tratamiento por el PLC, del módulo de conexiones y alambrado o partes que integran un tablero de control por PLC. Los demás capítulos hacen uso de dos elementos básicos de programación del control por PLC que son las subrutinas y los saltos de peldaño en el diagrama de escalera, estas dos funciones son controladas mediante su respectiva función de PLC que lo habilitan para tomar decisiones de donde saltar peldaños, cuantos y en que punto retornar.

El capítulo 3 divide el contenido en entradas y salidas discretas y análogas y el diagrama de escalera, el capítulo 4 de diagramas de timers y contadores y como configurar y que es un driver del PLC. El capítulo 5 introducción a una programación avanzada con algunos ejemplos de funciones retentivas, matemáticas y de lógica. El capítulo 6 es programación avanzada de subrutinas y de ciertas funciones especiales y de máscara de bits.

MARCO TEÓRICO

Es poner a prueba la funcionalidad del PLC para realizar cualquier lógica de diagrama de escalera que no sobrepase de ciertas características. Trabaja con entradas discretas o binarias y también con entradas análogas que son convertidas a valores digitales mediante un módulo de entrada que tenga tantos pines de entrada como bits en la palabra deseada. Y también que el PLC puede en todos los casos reemplazar al diagrama de control de escalera y el control convencional. Se prueba que el PLC es mejor alternativa que la lógica de control convencional cableada, cuando se necesita establecer una lógica de control pues

el PLC puede realizar si es programado saltos de rutina que en control convencional no se tiene, el PLC contiene mas capacidades que la lógica de control convencional además de todos los controles que contiene el control convencional.

También puede simular ciertos componentes como relevadores y contadores, lo que lo hace no necesarios como componentes externos, estos se encuentran contenidos en le PLC internos como parte de la programación y se puede seleccionar como operan o darles los parámetros necesarios; calibrarlos o ajustarlos. De este modo el circuito de escalera al realizarlo con el PLC contiene elementos virtuales. El único límite de la programación es la cantidad de contadores y de módulos de tarjetas de entrada para convertir de analógico a digital.

Empezando por el diagrama de escalera de paro y arranque de una carga que es un modelo retentivo con auto alimentación para mantenerse en cierto estado de activación, y un contacto de interrupción para desactivar el circuito simularlo mediante un relevador interno.

OBJETIVOS

El objetivo de esta tesis es mostrar el funcionamiento de un simulador haciendo uso del PLC SLC 500 aplicando conceptos y problemas reales de la industria. A técnicos e ingenieros de esta área de control, razón por cual tratamos de dar un enfoque generalizado sobre las partes preponderantes de este equipo con la consecuente descripción del funcionamiento; así como también dando exposiciones de programas reales de aplicación de control, para tratar de ambientar de una manera concreta la utilización de estos equipos.

HIPÓTESIS

Es en que el control convencional con componentes discretos puede ser sustituido en su totalidad por el control por PLC en funciones combinacionales, también la hipótesis es que el PLC tiene mas capacidad que el control convencional, ya que cuenta con mas elementos de programación que elementos físicos de control convencional, secuencial y combinación

de ambas usando programación de funciones básicas y avanzadas. La hipótesis es que todas las industrias ya utilizan lógica de control programable en lugar de cómo se hacía anteriormente con lógica de control convencional.

METODOLOGÍA

Previa elaboración del índice de la tesis presentar los principios, características y funciones del controlador lógico programable para posteriormente probarlos en forma práctica en el tablero creado para tal fin. La información o materiales de esta tesis en parte proviene o fue obtenida de las empresas de automatización ABSA, cursos y manuales de Allen Bradley, como referencias de fuentes de información en vídeos de entrenamiento para programar PLC. El contenido está agrupado en 4 partes: lógica combinacional, lógica secuencial, saltos de línea de peldaños y subrutinas de programación. Cada diagrama de escalera fue probado en un vídeo de entrenamiento y su funcionamiento descrito.

La metodología a seguir es mostrar un diagrama de escalera de cada ejemplo y luego programar el PLC para que lo haga igual que el diagrama de escalera. Toda programación en PLC puede ser representada mediante diagrama de escalera.

RESULTADOS

Todos los casos que se proponen tienen para realizarse el control por PLC y se pueden simular componentes de relevadores, contadores. Los resultados de esta investigación es que en México actualmente todas las industrias automatizadas utilizan lógica de control programable por PLC, que sustituye el control convencional que tenían.

Los componentes combinacionales como interruptores son contactos normalmente abiertos y otros contactos normalmente cerrados. En un tablero de control se muestra como diferentes diagramas de escalera y funciones controlan una salida seleccionada.

ANÁLISIS DE RESULTADOS Y DISCUSIÓN

Los resultados se separan en 4 grupos que son la simulación de funciones básicas, funciones avanzadas, y la programación de funciones básicas y programación de funciones avanzadas. Todos ellos les corresponde el diagrama de escalera que es básico en el control eléctrico. Mediante ese diagrama de escalera de cada caso se expresa la relación que tienen los componentes del circuito.

Los resultados obtenidos en esta tesis es un PLC tiene sus ventajas en que puede realizar cualquier circuito que la lógica convencional proponga, ya que se comparte el diagrama de escalera para control convencional y control con PLC. Una ventaja que tiene el PLC sobre el control convencional es que puede realizar subrutinas de programación y también puede saltar peldaños del diagrama de escalera con programación de saltos. Estas dos alternativas hacen que el PLC ofrezca más capacidad de programación que la que se tiene por construir con control convencional. Cualquier función lógica que no exceda de ciertas variables de entrada puede ser programada mediante PLC.

PROLOGO.

La poca difusión del conocimiento sobre sistemas de control con el cual operan equipos y maquinaria construidos con tecnología de vanguardia, utilizados en el sector industrial; originan uno de los problemas más comunes a los que se tienen que enfrentar los Ingenieros incorporados a la industria de la transformación. Tal es el caso de los Controladores Lógicos Programables del inglés “Programmable Logic Controller” (PLC), que desde su aparición en México han tenido muy poca difusión sobre su funcionamiento y aplicación. Aunque actualmente son los pilares de la automatización en grandes y crecientes empresas; no ha existido el interés debido por fomentar la enseñanza sobre estos, dentro de las Universidades.

Por lo que es la inquietud de querer aportar un poco de lo mucho que éste campo del control requiere. La elaboración de la tesis aquí expuesta basa su contenido en la descripción del funcionamiento, entorno y programación de los PLC's, para comprender de una manera generalizada el modo operativo de estos equipos, puesto que las instrucciones utilizadas en la creación de programas de control por PLC son casi similares o iguales entre los controladores de distintas marcas, razón por la cual la información contenida puede servir como base en la comprensión y el diseño de cualquier programa de Máquina, sin importar el tipo, capacidad y marca del controlador con el que trabajar.

Para facilitar la comprensión del comportamiento de las instrucciones lógicas utilizadas en los programas de PLC's, la descripción de su funcionamiento se da en detalle y con ejemplos ilustrativos de programas reales con y su respectivo esquema en controles industriales. También a lo largo del desarrollo se hace uso de los símbolos, diagramas, etiquetas (Indicadores de texto) y esquemas de sistemas eléctricos-electrónicos utilizados en la industria, que están ligados a los sistemas de control por PLC, esto con el fin de familiarizarse con éste tipo de información.

INTRODUCCIÓN

Con la actual política de globalización industrial se hace imperiosa la necesidad de las empresas de ser más productivas, una pieza clave en esta productividad es la implantación de la automatización en los procesos industriales, la actual demanda de productos de mejor calidad a un costo más competitivo nos obliga a tener procesos más confiables con menor variabilidad en los productos.

Es por ello que actualmente en todas las empresas se cuenta con controladores lógicos programables por lo que se busca con este trabajo de tesis adentrar a los alumnos de ingeniería industrial, mecánica, eléctrica en el conocimiento de los controladores lógicos programables dando un panorama general de principios de operación, características y programación de controladores lógicos programables.

Estos controladores lógicos programables PLC además de mejorar la productividad por el mínimo tiempo invertido en la detección de fallas, proporcionan un ambiente mas seguro de trabajo en los procesos ya que se evitan riesgos existentes en sistemas de control convencional. A su vez la mayoría de los equipos PLC están dispuestos de una manera que haya una rápida adaptación a nuevos procesos y también la expansión de los ya presentes sin tener que rediseñar todo el sistema de control.

En la actualidad la información de esta tecnología de primera necesidad PLC es escasa y un curso de conocimientos generales de PLC es muy costoso, por lo que los alumnos estamos distantes de tener contacto con estos equipos antes de ingresar a la industria, por ello elaboramos este material esperando sea de utilidad para todos los compañeros que estén interesados en tener un panorama general de estos equipos, la información o material de esta tesis en parte proviene o fue conseguida en las empresas de automatización absa, cursos y manuales de Allen Bradley como referencias de fuentes de información.

La decisión de tomar como ejemplo el PLC Allen Bradley se debe que son los primeros en la historia en patentar PLC, surgieron PLC's de distintas marcas después de la primera marca Allen Bradley.

1 GENERALIDADES

Un controlador lógico programable sirve para automatizar dispositivos electromecánicos tales como control de maquinaria de fábrica en líneas de montaje o atracciones mecánicas, más conocido por sus siglas en inglés PLC (Programmable Logic Controller), es un computador o pequeño ordenador empleado en automatización industrial y producción. Es un dispositivo electrónico utilizado para controlar de forma autónoma diferentes procesos en las plantas productivas.

Los PLCs son empleados en toda una gama de industrias y maquinaria. A diferencia de las computadoras de propósito general el PLC esta construido para controlar mediante grupos de señales de entrada señales de salida, es resistente a trabajar en rangos de temperatura de gran variación, ofrecen inmunidad al ruido eléctrico, tienen resistencia a las vibraciones e impacto.

La programación para control de funcionamiento de máquinas comunmente respaldada con baterías y realizada una copia de seguridad o en memorias no volátiles. Un PLC es un sistema de tiempo real duro donde las señales de salida deben ser producidas en respuesta a condiciones de señales de entrada dentro del tiempo limitado en nanosegundos, que de lo contrario no producirá el resultado deseado.

1.1 BREVE HISTORIA DE LOS PLC's.

En la actualidad existen una gran variedad de PLC's en el mercado, el desarrollo inició en 1968 en General Motors industria automotriz, debido a sus necesidades de crecimiento empiezan a demandar sistemas de control que permitan mejorar la eficiencia y reducir los costos de ampliación o cambios a las líneas de producción.

Hacer sistemas de control mas potentes y confiables que permitieran reducir los costos de producción en las líneas de ensamble. Fue así como a finales de 1968 General Motors estableció los lineamientos que deberían cumplir los nuevos sistemas de control.

En 1970 Inicia el diseño y desarrollo de lo que después en 1974 se patentaría como PLC por la empresa Allen Bradley marcándose el inicio del desarrollo de diversos modelos de controladores automáticos que no vieron la luz hasta 1977 en que Allen Bradley lanzó al mercado un PLC 2 con un procesador Intel 8080, el desarrollo y lanzamiento de otros modelos de PLC's continuó por varios años hasta que en 1986 Allen Bradley anuncia su último controlador lógico programable el PLC 5. Tras la aparición del PLC 5 se inició el diseño y desarrollo acelerado de nuevos modelos de PLC, los cuales han evolucionado tanto en capacidad como en desempeño y seguridad, ofreciendo así mejores y mas rápidas aplicaciones para la industria.

El autómatas es la primera máquina con lenguaje, es decir, un calculador lógico cuyo juego de instrucciones se orienta hacia los sistemas de evolución secuencial. En la disciplina perteneciente a la informática, se describen tres tipos de autómatas que reconocen tipos diferentes de lenguajes: los autómatas a pila, máquinas de Turing, los autómatas finitos. No obstante, aunque estos ordenadores resolvían los inconvenientes de un Sistema cableado o la llamada lógica cableada, presentaban nuevos problemas. La aparición de los ordenadores a mediados de los 50's inauguró el campo de lógica programada para el control de procesos industriales:

- Mala adaptación al entorno industrial.
- Coste elevado de los equipos.
- Necesidad de personal informático para la realización de los programas.
- Necesidad de personal especializado para el mantenimiento.

R. E. Moreley, el padre del autómata programable considerado por muchos, trabajando independientemente de las especificaciones de la General Motors desarrolló un equipo que respondía a las necesidades de dicha multinacional. Hacia la primera mitad de los años 70 los autómatas programables incorporan tecnología de los microcontroladores, aumentando de este modo sus prestaciones:

- Posibilidad de utilizar redes de comunicaciones.
- Mejoras en los lenguajes de programación.
- Realización de operaciones aritméticas.
- Comunicación con los ordenadores.
- Incremento de la capacidad de memoria.
- Posibilidad de entradas y salidas analógicas.

La década de los años 80 se caracteriza por incorporación de microprocesadores, con esto consiguiendo:

- Alta velocidad de respuesta.
- Reducción de las dimensiones.
- Mayor seguridad de funcionamiento.
- Gran capacidad de almacenamiento de datos.
- Lenguajes de programación más potentes: contactos, bloques funcionales, GRAFCET (GRAFica de Control de Etapa de Transición).

En la actualidad existen autómatas que permiten automatizar a todos los niveles, desde pequeños sistemas mediante autómatas compactos, hasta sistemas sumamente complejos mediante la utilización de grandes redes de autómatas.

En los finales de los años de 1960 la industria se empezó a interesar en construir nuevas tecnologías electrónicas, para controlar de modo muy efectivo para reemplazar los anteriores sistemas de control basados en circuitos electromecánicos de relevadores, interruptores y otros dispositivos externos utilizados para control de sistemas de lógica combinacional. Estos problemas se solucionarían con la aparición del autómata programable o PLC (Controlador Lógico Programable; en inglés *Programmable Logic Controller*).



Fig 1.1 UR2 de 9 ranuras, de izquierda a derecha: fuente de alimentación PS407 4A, CPU 416-3, módulo de interfaz IM 460-0 y procesador de comunicaciones CP 443-1.

El resultado de la colaboración fue un equipo programado, cuyo empleo no tardó en extenderse a otras industrias, denominado PDP-14. En un principio, por lo que los problemas que requerían la manipulación de magnitudes analógicas se dejaron para los tradicionales sistemas de control distribuido, los autómatas programables sólo trabajaban con control discreta (Si o No). preocupada por los elevados costos de los sistemas de control a base de relés, de lógica cableada. A mediados de los años 60, *General Motors*, comenzó a trabajar con *Digital* que evitara los inconvenientes de la lógica programada en el desarrollo de un sistema de control.

En 1968 GM Hydramatic (la división de transmisión automática de General Motors) incorporó una serie de propuestas para reemplazar mediante componentes electrónicos a los sistemas cableados de relevadores. Modicon (MODular DIGital CONtroller) fue la propuesta ganadora vino de Bedford Associates. fue el primer PLC 084 diseño de Bedford Associates nº 84. Bedford Associates estableció una reciente empresa cuya actividad es desarrollo, fabricación, venta y mantenimiento del primer y nuevo PLC.

Uno de los técnicos que trabajaron en ese proyecto de desarrollar el primer PLC fue Dick Morley, considerado como el padre del PLC. La primera marca Modicon fue vendida en 1977 a Gould Electronics, posteriormente adquirida por la compañía alemana AEG y luego por la francesa Schneider Electric, el actual titular o dueño, además Siemens está a la vanguardia de dispositivos de última tecnología en PLC a la par de Allen Bradley.

Actualmente los PLC pueden ser programados mediante diferentes opciones, desde la lógica de relevadores electromecánicos a los lenguajes de programación especialmente códigos derivados del BASIC y C, lógica de estado, lenguajes de programación de alto nivel basados en diagramas de estado y especialmente adaptados para PLCs. En sus inicios el PLC está diseñado para sustituir sistemas electromecánicos de relevadores lógicos. Los PLC son programables mediante el lenguaje de escalera, se parece mucho a la lógica de los diagramas convencionales de relevadores. Esto fue elegido para aminorar la demanda de formación de los técnicos existentes, otros autómatas emplean una base de datos de listas de instrucciones programables.

1.2 RELÉ LÓGICO PROGRAMABLE PLR

Estos pequeños dispositivos PLR's se hacen frecuentemente en un tamaño físico y forma común por varios fabricantes, y con la marca de los fabricantes más grandes de PLC's para completar su gama baja de producto final.

En derivación al PLC surgieron unos controles tipo dispositivos pequeños llamados PLR's (relés lógicos programables), muy conocidos por otros nombres parecidos, se han vuelto populares y utilizados. Estos presentan una gran semejanza a los PLC, su uso es muy evidente en la industria ligera, donde no se requieren muchos puntos de E/S (es decir, unas pocas señales que llegan desde el mundo real y algunas que salen) participan en el proceso, y el bajo costo es deseado a comparación de un PLC que no sería del todo aprovechado, por ejemplo donde solo vamos a controlar de modo independiente un Displays de dígitos de 7 segmentos y pequeñas pantallas numéricas LED, un torniquete, una puerta entre otras aplicaciones muy aisladas.

Los PLR's comunmente contienen entre 8 y 12 entradas digitales, 4 y 8 salidas discretas, y hasta 2 entradas analógicas. El tamaño es en promedio alrededor de 4 pulgadas, 3 pulgadas y 3 pulgadas en forma de prisma rectangular. La mayoría de estos dispositivos incluyen una pantalla LCD de tamaño pequeño para la visualización simplificada lógica de escalera (sólo una porción muy pequeña del programa está visible en un momento dado) y el estado de los puntos de E/S. Normalmente estas pantallas están acompañados por una botonera basculante de 4 posiciones más cuatro pulsadores más separados, y se usan para navegar y editar la lógica.

La mayoría tienen un pequeño conector para la conexión a través de RS-232 o RS-485 a un ordenador personal para que los programadores pueden utilizar simples aplicaciones de Windows para la programación en lugar de verse obligados a utilizar la pantalla LCD y el conjunto de pequeños pulsadores para este fin. A diferencia de los PLCs regulares que son generalmente modulares y ampliables en gran medida, los PLRs son por lo general no modulares o expansibles, pero su precio puede ser dos órdenes de magnitud menos de un PLC y todavía ofrecen un diseño robusto y de ejecución determinista de la lógica.

1.3 LA INGENIERÍA AUTOMÁTICA

En todos los PLC's se pueden identificar 3 elementos principales de hardware, tarjetas de entrada, tarjetas de salida, CPU ó unidad procesadora, las tarjetas de entrada tiene como objetivo convertir las señales de campo a un formato lógico de ceros y unos que el procesador del PLC pueda entender. Por ejemplo la temperatura de un tanque se mide utilizando un termopar el cual varía el voltaje dependiendo de la temperatura censada, dicho voltaje se conecta a una tarjeta de entradas al PLC por medio de sus terminales, la tarjeta convierte la señal de voltaje a un valor lógico, de tal manera que el controlador pueda utilizar la temperatura dentro de la secuencia ya programada.

El procesador o CPU tiene por objetivo implementar las instrucciones almacenadas en memoria del PLC, y de tomar decisiones en función de las variables de campo conectadas a la tarjeta de entrada. Las tarjetas de salida son utilizadas para convertir la señal lógica de ceros y unos resultante de la evaluación del procesador a señales de campo que permitan activar válvulas, bombas, motores, etc. Como se relacionan entre sí la tarjeta de entrada, el CPU y la tarjeta de salida.

Dentro de las ventajas que estos equipos poseen se encuentra que, gracias a ellos, es posible ahorrar tiempo en la elaboración de proyectos, pudiendo realizar modificaciones sin costos adicionales. Por otra parte, son de tamaño reducido y mantenimiento de bajo costo, además permiten ahorrar dinero en mano de obra y la posibilidad de controlar más de una máquina con el mismo equipo. Sin embargo, y como sucede en todos los casos, los controladores lógicos programables, o PLC's, presentan ciertas desventajas como es la necesidad de contar con técnicos cualificados y adiestrados específicamente para ocuparse de su buen funcionamiento.

Hoy en día, los PLC no sólo controlan la lógica de funcionamiento de máquinas, plantas y procesos industriales, sino que también pueden realizar operaciones aritméticas, manejar señales analógicas para realizar estrategias de control, tales como controladores PID (Proporcional Integral y Derivativo).

Las ocupaciones básicas y esenciales del PLC ha cambiado con el tiempo para incluir el control del relé secuencial, control de movimiento, control de procesos, Sistemas de Control Distribuido y comunicación por red. La potencia de manipulación, almacenamiento, velocidad de procesamiento y de comunicación de algunos PLCs modernos son aproximadamente equivalentes a computadoras o pequeños ordenadores. Un enlace-PLC programado combinado con dispositivos de E/S remoto, permite utilizar un ordenador de sobremesa de uso general para suplantar algunos PLC en algunas aplicaciones.

En cuanto al sentido práctico de estos controladores de ordenadores de sobremesa basados en lógica, es esencial tener en cuenta que no se han establecido globalmente en la industria pesada o ciertas industrias debido a que los ordenadores de sobremesa ejecutan sistemas operativos inestables a comparación de los PLCs, y porque el hardware del ordenador de escritorio no es comunmente diseñado para resistir los mismos niveles de tolerancia a la temperatura, humedad, vibraciones, y la longevidad como los procesadores utilizados en los PLC.

Además de las deficiencias de los dispositivos de lógica basada en ordenadores de escritorio; los sistemas operativos tales como Windows no presentan ejecución de lógica determinista, con el resultado de que la lógica no siempre puede responder a los cambios en el estado de la lógica o de los estado de entrada con la consistencia extrema en el tiempo como se espera de los PLCs. Sin embargo, este tipo de aplicaciones de escritorio lógicas encuentran uso en situaciones menos críticas, como la automatización de laboratorio y su uso en instalaciones pequeñas en las que la aplicación es menos exigente y crítica, ya que por lo general son mucho menos costosos que los PLCs.

Para ello recurramos al ejemplo mas cercano para nosotros, que pasaría si vamos caminando y se atraviesa un objeto adelante, el ojo envía la información al cerebro de que hay algo adelante, esta información es transmitida al cerebro, el cual empieza a desarrollar o realizar una serie de toma de decisiones, para finalmente el cerebro envía estos datos a las piernas, ocasionando con ello que los pies detengan el movimiento.

De igual manera funciona un PLC, la tarjeta del sensor es la que obtiene los datos de entrada, el CPU ejecuta el programa y el CPU después envía los datos a la tarjeta salida ya calculados y esto activa las válvulas ó motores. Al utilizar PLC's en las plantas productivas podemos obtener varios beneficios uno de ellos es

- A. Mayor velocidad al desarrollar proyectos. Es ventaja para quienes hacen integración.
- B. Menor coste de crecimiento y de mantenimiento, al ser reutilizables y en caso de necesitar crecimiento anexamos tarjeta adicional.
- C. Reducción de mermas y problemas que por errores humanos que pueden estar ocasionando pérdidas.
- D. Estandarización de calidad, este otro punto muy importante ya que no dependen de un operador que pueda hacer las cosas de un modo diferente cada día. Se trata de un sistema calibrado para trabajar siempre de la misma manera.
- E. Incremento de la productividad tanto en personas que desarrollan proyectos como en usuarios que utilizan los proyectos de automatización.
- F. Así un PLC es un dispositivo electrónico capaz de ejecutar acciones de forma autónoma en función de las señales de campo como sensores, switches, con el objeto de mejorar la productividad y calidad de los sistemas de producción.

Como sabemos con anterioridad los PLC's vinieron a sustituir a sistemas electromecánicos de control, no es de extrañar que la manera de programar los PLC's sea utilizando una estructura muy similar a los diagramas unifilares de los sistemas eléctricos, estos elementos de programación se les conoce como diagrama de escalera, el cual estará basado en el actual hardware de evaluar contactos, bobinas, que es muy similar a como vienen en el diagrama unifilar.

Sobre las implicaciones de utilizar un PLC en una planta productiva, para lo cual tenemos un tanque de mezclado, dos válvulas y dos tanques de líquido, una banda transportadora y un envase. Que ocurriría si se necesitara trabajar manualmente este sistema, mediante un operador que estuviera trabajando y llenando constantemente los envases, al cabo de unas cuantas horas tal vez 8 que sea la jornada de trabajo es probable que tuviéramos problemas que merman la calidad del producto o bajaría el rendimiento. Por eso ahora la gran mayoría de plantas productivas cuentan ya con sistemas de PLC autónomos que permitan mejorar la productividad sin arriesgar a los operarios estandarizando la calidad.

La **ingeniería automática**, conocida también como **ingeniería de control**, es el uso de elementos sistemáticos como control numérico NC controladores lógicos programables PLC y otros sistemas de control industrial) relacionados con otras aplicaciones de la tecnología de la información como son tecnologías de ayuda por computador CAD, CAM, CAx, para el control industrial de maquinaria y procesos, reduciendo la necesidad de intervención humana. En el ámbito de la industrialización, la automatización esta un paso por delante de la mecanización. Mientras que la mecanización provee operadores humanos con maquinaria para ayudar a exigencias musculares de trabajo, la automatización reduce considerablemente la necesidad para exigencias humanas sensoriales y mentales, los procesos y los sistemas también pueden ser automatizados.

La Ingeniería de Control se preocupó desde sus orígenes de la automatización y del control automático de sistemas complejos, sin intervención humana directa. Campos como el Control de procesos, Control de sistemas electromecánicos, Supervisión y ajuste de controladores y otros donde se aplican teorías y técnicas entre las que podemos destacar: Control Óptimo, Control Predictivo, Control Robusto, Control no lineal, y Control de sistemas entre otros. Todo ello con trabajos y aplicaciones muy diversas: investigación básica, investigación aplicada, militares, industriales, comerciales, etc, las cuales han hecho de la Ingeniería de Control una materia científica y tecnológica imprescindible hoy en día.

La ingeniería automática es un área multidisciplinar que se encarga de la concepción y desarrollo de autómatas y de otros procesos automáticos. La ingeniería automática se encarga de la automatización de procesos técnicos en las siguientes áreas:

- Electrónica y electricidad
- Automatización de edificios (domótica)
- Procesos químicos
- Ingeniería mecánica
- Automóviles
- Aeronáutica y astronáutica
- Sistemas
- Robótica
- Biología
- Medicina
- Mecatronica

Dentro de la ingeniería automática se encuentran, entre otras, las siguientes subdisciplinas:

- Instrumentación automática
- Tecnología de sensores
- Regulación automática
- Control de procesos
- Ingeniería automática
- Vigilancia
- Diagnóstico de fallos
- Optimización
- Visualización de procesos

El diseño, implementación y puesta en marcha de sistemas automáticos es un proceso muy metódico. Estos métodos de la ingeniería automática están en parte divididos en procesos.

Hoy en día, la ingeniería electrónica es una parte integrante de la ingeniería de control. Casi todos los sistemas automáticos funcionan con ayuda de la electrónica, quedando los sistemas automáticos basados en la mecánica en un segundo plano. Por otra parte, los sistemas digitales están tomando cada vez más importancia en esta área, en especial los microprocesadores y los convertidores digital-analógicos (D/A) así como los analógico-digitales (A/D).

La mayoría de los métodos generales de la ingeniería de control se basan en el uso de modelos analíticos del proceso que se quiere estudiar obtenidos de forma teórica o experimental. A partir de estos modelos se pueden usar métodos científicos para obtener sistemas de control para los mismos. Esta parte de la automática tiene una gran importancia, contando con los siguientes métodos:

- Identificación y estimación de parámetros
- Control adaptativo
- Vigilancia y diagnóstico de fallos
- Lógica Fuzzy
- Algoritmos evolutivos
- Redes neuronales

Con estos métodos se pueden diseñar sistemas inteligentes con reguladores basados en modelos que se auto-actualizan y con control de fallos, que pueden tomar decisiones en función de la información que obtienen a través de sus sensores. Los mismos son también de gran importancia en mecatrónica y son usados también en el control digital de robots, máquinas herramienta, motores, coches y sistemas neumáticos e hidráulicos.

El control es un área de la ingeniería y forma parte de la Ingeniería de Control. Se centra en el control de los sistemas dinámicos mediante el principio de la realimentación, para conseguir que las salidas de los mismos se acerquen lo más posible a un comportamiento predefinido. Esta rama de la ingeniería tiene como herramientas los métodos de la teoría de sistemas matemática. Las bases de esta ingeniería se sentaron a mediados del Siglo XX a partir de la cibernética. Sus principales aportaciones corresponden a Norbert Wiener, Rudolf Kalman y David G. Luenberger.

La ingeniería de control es una ciencia interdisciplinar relacionada con muchos otros campos, principalmente las matemáticas y la informática. Las aplicaciones son de lo más variado: desde tecnología de fabricación, instrumentación médica, Subestación eléctrica, ingeniería de procesos, robótica hasta economía y sociología. Aplicaciones típicas son, por ejemplo, el piloto automático de aviones y barcos y el ABS de los automóviles. En la biología se pueden encontrar también sistemas de control realimentados, como por ejemplo el habla humana, donde el oído recoge la propia voz para regularla.

El control de temperatura en una habitación es un ejemplo claro y típico de una aplicación de ingeniería de control. El objetivo es mantener la temperatura de una habitación en un valor deseado, aunque la apertura de puertas y ventanas y la temperatura en el exterior hagan que la cantidad de calor que pierde la habitación sean variables (perturbaciones externas). Para alcanzar el objetivo, el sistema de calefacción debe modificarse para compensar esas perturbaciones. Esto se hace a través del termostato, que mide la temperatura actual y la temperatura deseada, y modifica la temperatura del agua del sistema de calefacción para reducir la diferencia entre las dos temperaturas.

La ingeniería de control moderna se relaciona de cerca con la Ingeniería eléctrica y la electrónica, pues los circuitos electrónicos pueden ser modelizados fácilmente usando técnicas de la teoría de control. En muchas universidades, los cursos de ingeniería de control son dictados generalmente por la Facultad de Ingeniería Eléctrica.

Anterior a la electrónica moderna, los dispositivos para el control de procesos eran diseñados por la ingeniería mecánica, los que incluían dispositivos tales como levas junto con dispositivos neumáticos e hidráulicos. Algunos de estos dispositivos mecánicos siguen siendo usados en la actualidad en combinación con modernos dispositivos electrónicos.

El control aplicado en la industria se conoce como control de procesos. Se ocupa sobre todo del control de variables como temperatura, presión, caudal, etc, en un proceso químico de una planta. Se incluye como parte del plan de estudios de cualquier programa de ingeniería química. Emplea muchos de los principios de la ingeniería de control. La ingeniería de control es un área muy amplia y cualquier ingeniería puede utilizar los mismos principios y técnicas que esta utiliza. La ingeniería de control se ha diversificado a tal punto que hoy se aplica incluso en campos como la biología, las finanzas, e incluso el comportamiento humano.

El estudiante de ingeniería de control comienza el curso con los llamados sistemas de control lineal que requieren del uso de matemática elemental y la transformada de Laplace (llamada teoría de control clásica). En el control lineal, el estudiante hace análisis de los sistemas en el dominio de la frecuencia y del tiempo mientras que en los sistemas no lineales y en el control digital se requiere el uso del álgebra lineal y de la transformada Z respectivamente. A partir de aquí hay varias ramas secundarias.

La Ingeniería de control es una disciplina que se focaliza en modelizar matemáticamente una gama diversa de sistemas dinámicos y el diseño de controladores que harán que estos sistemas se comporten de manera deseada. Aunque tales controladores no necesariamente son electrónicos y por lo tanto la ingeniería de control es a menudo un subcampo de otras ingenierías como la mecánica. Dispositivos tales como circuitos eléctricos, procesadores digitales y los microcontroladores son muy utilizados en todo sistema de control moderno. La ingeniería de control tiene un amplio rango de aplicación en áreas como los sistemas de vuelo y de propulsión de los aviones de aerolíneas, militares, en la carrera espacial y últimamente en la industria automotriz.

El objetivo del control automático es poder manejar con una o más entradas (o referencia), una o más salidas de una planta o sistema, para hacerlo, la idea más primitiva es colocar entre la referencia y la planta, un controlador que sea inverso de la función de transferencia de la planta, de tal manera que la función de transferencia de todo el sistema (la planta más el controlador), sea igual a uno; logrando de esta manera que la salida sea igual a la entrada; esta primera idea se denomina control en la lazo abierto.



Fig 1.2 PLC dentro de su cabina.

Un ejemplo clásico de control en lazo abierto es una lavadora de ropa ya que ésta funciona durante un ciclo predeterminado sin hacer uso de sensores. Las desventajas que tiene el control por lazo abierto son:

- A) No se conoce la planta, a lo más se puede conocer un modelo aproximado, por lo que no se puede lograr el inverso perfecto.
- B) No se puede usar para controlar plantas inestables.
- C) No compensa perturbaciones en el sistema.

- D) Si la planta tiene grado relativo mayor que cero, no se puede crear un controlador que la invierta, ya que no se puede hacer una función de transferencia con grado menor que cero.
- E) Es imposible invertir perfectamente una planta, si esta tiene retardos, ya que su inverso sería un adelanto en el tiempo (se debería tener la capacidad de predecir el futuro).

Una idea más avanzada, y más ampliamente implementada, es el concepto de feedback o realimentación, en que se usa la medición de la salida del sistema, como otra entrada del mismo, de tal forma que se puede diseñar un controlador que ajuste la actuación para variar la salida y llevarla al valor deseado. Por ejemplo el cuerpo humano realiza un control por realimentación para mantener la homeostasis, tiene sensores para cada elemento en el cuerpo y si es que se detecta una cantidad anormal.

El cuerpo tiene sistemas para compensarlo (estos sistemas serían el controlador), los que produce una actuación (cierra válvulas, produce más sustancia, etc) hasta que los sensores le indican al cuerpo que ya se alcanzó el equilibrio.

Otro ejemplo : un automóvil con control de cruceo la velocidad se sensa y se retroalimenta continuamente al sistema que ajusta la velocidad del motor por medio del suministro de combustible al mismo, en este último caso la salida del sistema sería la velocidad del motor, el controlador sería el sistema que decide cuanto combustible echar de acuerdo a la velocidad y la actuación sería la cantidad de combustible suministrado.

Las ventajas que tiene el control por retroalimentación son:

- A) Puede controlar sistemas inestables.
- B) Puede compensar perturbaciones.
- C) Puede controlar sistemas incluso si estos tienen errores de modelado.

Desventajas:

- A) El uso de sensores hace más caro (en dinero) el control.
- B) Se introduce el problema del ruido, al hacer la medición.

1.4 COMPONENTES DE CONTROLADORES DE LÓGICA PROGRAMABLE PLC.

Los controladores lógicos programables o PLC's juegan un papel vital en la industria, se pueden utilizar en la manufactura para repuestos para vehículos o en el control de procesos tales como la preparación de alimentos. Comparados con los sistemas tradicionales alambrados los PLC's pueden ser re programables, pueden ser monitorizados directamente y pueden encontrar fallas dentro de si mismos o en los equipos a los cuales están conectados.

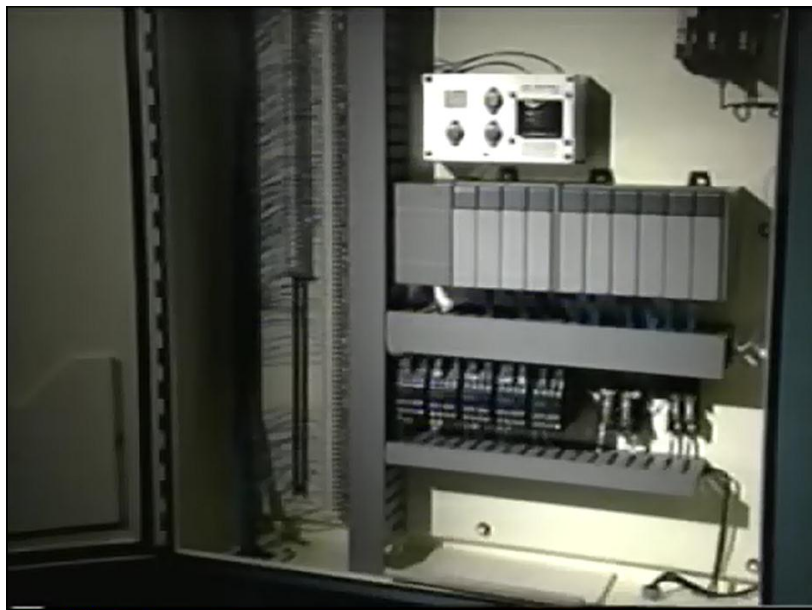


Fig 1.3 PLC, fuente de alimentación y relevadores.

El entender los diferentes aspectos fundamentales de los PLC's nos dan las bases sobre las cuales construir las técnicas para programar y diagnosticar. Debemos conocer los

componentes de los controladores lógicos programables, los diagramas discretos, las funciones de lógica y los diversos sistemas numéricos.

Los PLC's tienen una gran variedad de opciones y configuraciones para aplicaciones específicas. Son equipos duraderos capaces de aguantar las condiciones de humedad, ruido, temperatura y vibración del ambiente industrial. Todos los sistemas tienen componentes básicos que manipulan y procesan los datos importados y controlan varios puntos de salida. Como vemos aquí los elementos básicos de un PLC son: la unidad central procesadora CPU, la unidad de Memoria, la interface de entrada y la interface de salida.

Las señales externas son convertidas por la interface de entrada en un lenguaje que la unidad central puede entender, la cual manipula la información según el programa desarrollado por el usuario que está en la memoria. Los resultados de la ejecución de este programa pasan a la interface de salida la cual los convierte en un lenguaje utilizable por los equipos que estén conectados.

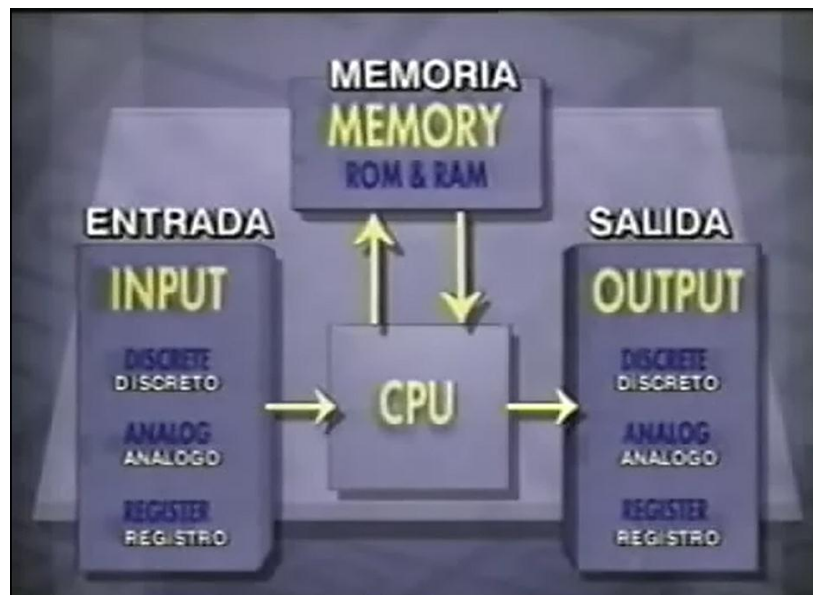


Fig 1.4 PLC, partes principales del CPU del PLC.

La unidad central procesadora CPU es el cerebro del PLC, consiste de un microprocesador cuya función es manejar los datos recibidos por puntos de entrada según lo indique el

programa y controlar los puntos de salida afectados. La unidad central está ubicada dentro del módulo procesador dentro de los PLC modulares.

La memoria se puede clasificar en 3 categorías ROM o memoria tipo Read Only Memory, RAM o Random Acces memory, EEPROM memoria reprogramable o reborrable electrónicamente. La memoria ROM contiene información permanente del fabricante instaladas por el fabricante, la cual es usada para implementar el programa del usuario almacenado en la porción del RAM o EEPROM en la memoria.

La memoria tipo RAM es usada para programas desarrollados por el usuario y puede ser manipulada para manejar registros de almacenamiento, relojes de control, registradores y otras funciones dentro del PLC. La memoria tipo RAM también es conocida como memoria volátil ya que pierde la información al ser desenergizada. Esto se puede evitar instalando una batería que mantenga a la memoria tipo RAM energizada cuando se desconecta la fuente de energía principal.

La memoria tipo EEPROM es usada de la misma manera que el RAM excepto que no requiere batería.

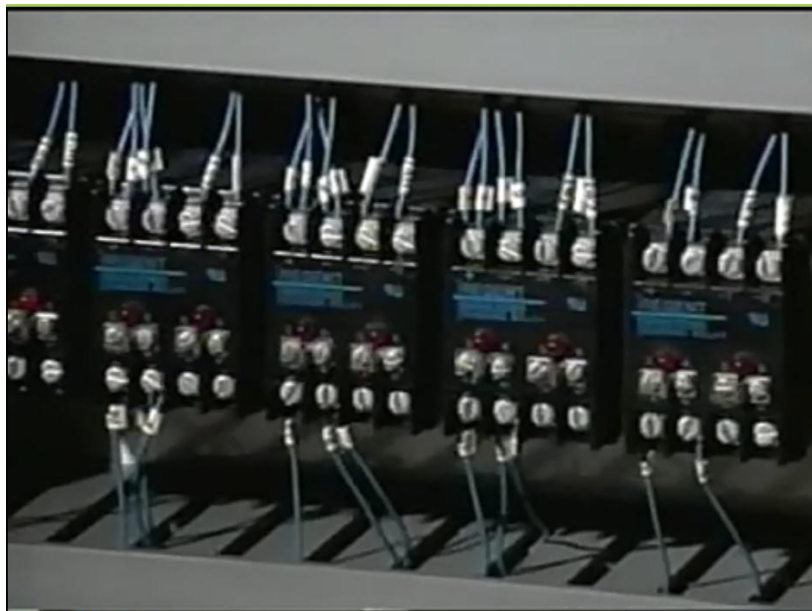


Fig 1.5 Grupo de relevadores del PLC.

Los puntos de entrada al PLC son de dos tipos: discretos y análogos, los discretos están prendidos o apagados ON, OFF respectivamente y se representan con un uno o un cero respectivamente en la memoria. Un interruptor y un contacto relevador son ejemplos de puntos de entrada discretos. Este tipo de puntos de entrada discretos de entrada pueden ser agrupados para formar registros de entrada que presentan un número binario a la unidad central. Como por ejemplo las calibraciones de los contadores digitales o los valores numéricos de sistemas externos.

Los voltajes variables tales como las salidas de transductores o potenciómetros, son ejemplos de puntos de entrada análogos, este voltaje es convertido a un número proporcional a la magnitud del voltaje del circuito análogo de entrada del PLC. Este voltaje puede ser leído por la unidad central cuando lo requiera el programa.

Los puntos de salida del PLC son similares a los puntos de entrada, excepto que proveen voltajes específicos, ya sean discretos o análogos a los sistemas a los cuales están conectados. Sistemas que pueden ser controlados por puntos de salida discretos pueden ser una luz, un control de válvulas hidráulicas, un solenoide, o un alarma o chicharra. Los puntos de salida análogos pueden proveer de voltajes de control a sistemas externos tales como motores de velocidad variable o pantallas análogas.

Un registro de salida consiste en una serie de salidas agrupadas para presentar datos numéricos que controlan pantallas LED ó LCD ó a sistemas externos.

2 PROGRAMACIÓN

Hay algunos conceptos que deben ser entendidos para programar un PLC efectivamente, un programa PLC es documentado generalmente en un programa discreto o de escalera, este sistema es usado porque los PLC generalmente reemplazan un sistema de relevos de control, como podemos ver en este diagrama eléctrico, el contacto positivo y el contacto neutral son los rieles de la escalera, mientras que las líneas del programa representan los escalones. Diagramas como el que vemos aquí usan símbolos eléctricos para identificar interruptores, sobrecargas, motores de arranque y motores, a esto se le conoce generalmente como relevos o relevadores lógicos.

2.1 DIAGRAMAS DISCRETOS

Si tenemos que alambrear este sistema estamos limitados a utilizar el voltaje requerido por el motor, el cual comúnmente excede los 120 voltios, esto es peligroso para el operador y reduce la durabilidad de los contactos. Si tenemos que utilizar los mismos sistemas de entrada y de salida para cumplir la misma función con un PLC podemos utilizar voltajes menores que no son peligrosos para el operador.

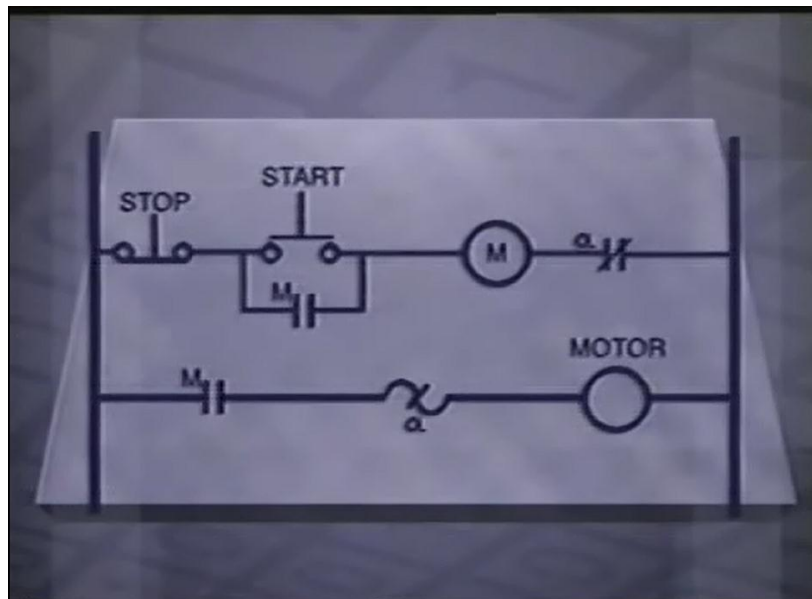


Fig 2.1 Diagrama de paro y arranque de motor.

Este rectángulo representa el PLC, aquí vemos la fuente de energía de 24 voltios DC o corriente directa con la salida negativa conectada al terminal modular de entrada común, este diagrama indica que se requiere conectar 3 entradas, un botón de arranque, un botón de parada y la sobrecarga para el motor.

El lado positivo de la fuente de energía de 24 voltios DC está conectado a un lado de cada entrada, luego se conecta un cable desde cada sistema al modulo de entrada del PLC, ahora la salida del PLC, el motor y el motor de arranque tienen que ser conectados. Esto representa la fuente de energía de 120 voltios AC por un lado conectado al módulo de salida, debido que el único sistema de interés al PLC es el motor de arranque, este es conectado al módulo de salida y al otro lado de la fuente de energía, luego el motor, sus sistemas de sobrecarga y el contacto de corriente principal del motor de arranque son conectados en serie a la fuente de energía, esto completa el alambrado del sistema.

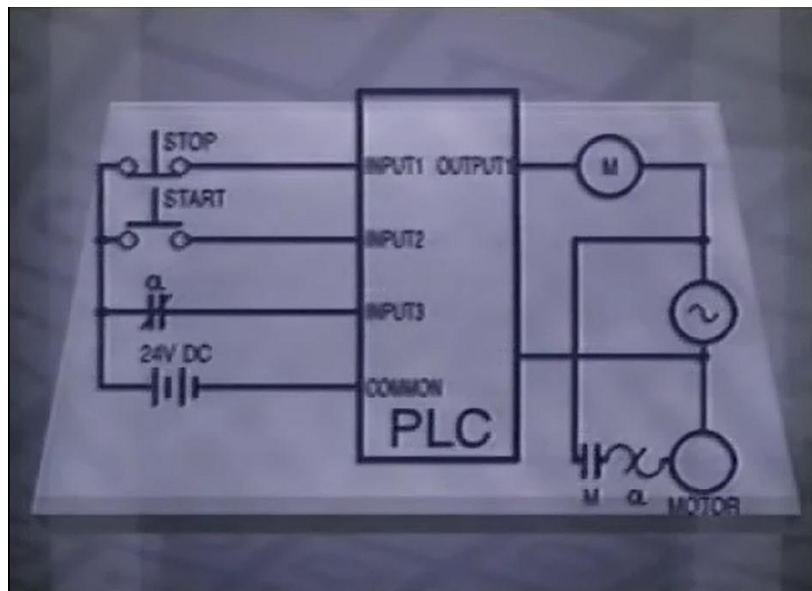


Fig 2.2 Conexión de paro y arranque al PLC.

Aunque hay muchos tipos de sistemas conectados al PLC este únicamente reconoce si una entrada discreta está encendida o apagada, las entradas son representadas en el diagrama discreto como contactos normalmente cerrados o abiertos.

Un contacto normalmente abierto es un contacto que está cerrado cuando la entrada asociada con él está conectada y abierto cuando la entrada asociada con él está desconectada. Un contacto normalmente abierto significa que cuando la entrada está desconectada no hay paso de corriente a través del contacto. Algunos controladores se refieren a este símbolo como normalmente abierto, lo cual significa que si la entrada está conectada, implementará la orden dada en el diagrama discreto.

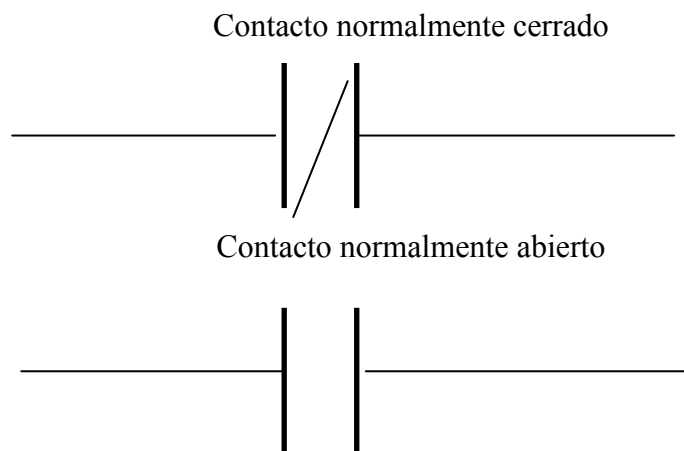


Fig 2.3 Contactos normalmente cerrado, abierto y símbolos.

Cuando el contacto aparece resaltado indica que está energizado y que hay corriente pasando por el contacto, este diagrama discreto representa una simple función de encender una luz, cuando se oprime el botón interruptor el contacto se cierra energiza la bobina o carrete de salida, esto a su vez le provee voltaje a la luz.

Un contacto normalmente cerrado es un contacto que está abierto cuando la entrada asociada con él está energizada y cerrado cuando la entrada asociada con él está desconectada. Un contacto normalmente cerrado indica que cuando no hay señal por parte del sistema de entrada hay corriente que pasa a través del contacto. Algunos controladores se refieren a este símbolo como normalmente cerrado lo cual significa que si la entrada está desconectada implementará la orden dada en el diagrama discreto.

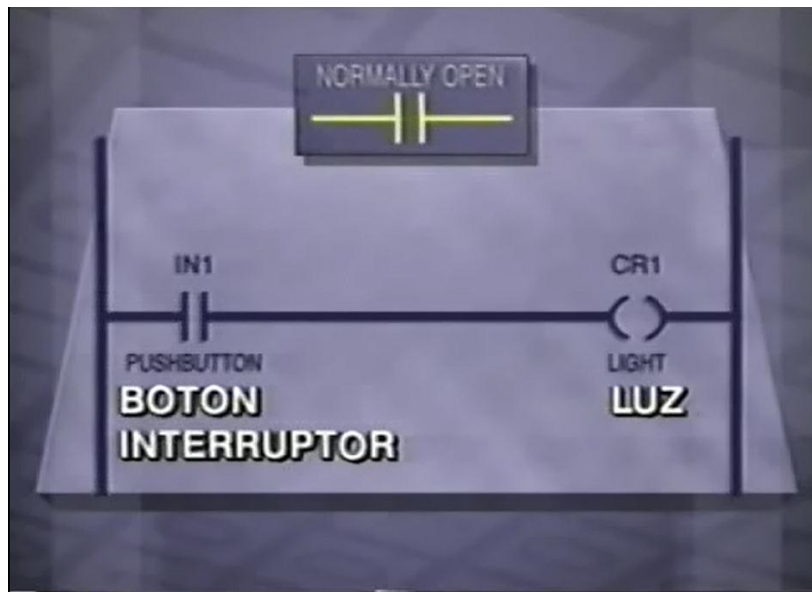


Fig 2.4 –control de luz por un botón normalmente abierto.

Cuando el contacto aparece resaltado indica que no está energizado y que hay corriente pasando por el contacto, en este diagrama discreto la luz estará encendida hasta que el interruptor sea activado.

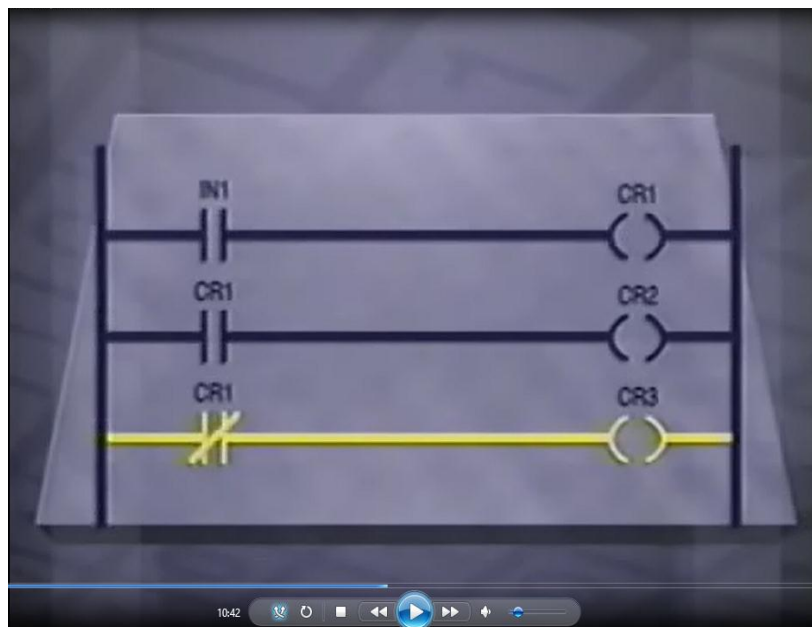


Fig 2.5 Control de una carga con un contacto normalmente cerrado peldaño.

Un contacto puede estar asociado con cualquier sistema de entrada, bobina de salida, reloj de control, registrador, relevo de control, al igual que con banderas de estado. Si el contacto está asociado con una bobina o carrete de relevo se muestra en la condición cuando la bobina está desenergizada, esto significa que un contacto normalmente cerrado está cerrado cuando la bobina asociada con él está desconectada. Un contacto normalmente abierto estará abierto cuando la bobina asociada con él está desconectada.

Una salida PLC se indica con este símbolo el cual representa una bobina o carrete relevo, es un sistema electromagnético que usa un contacto móvil con resortes y bisagras para cerrar o abrir el espacio entre los contactos de una fuente de energía.

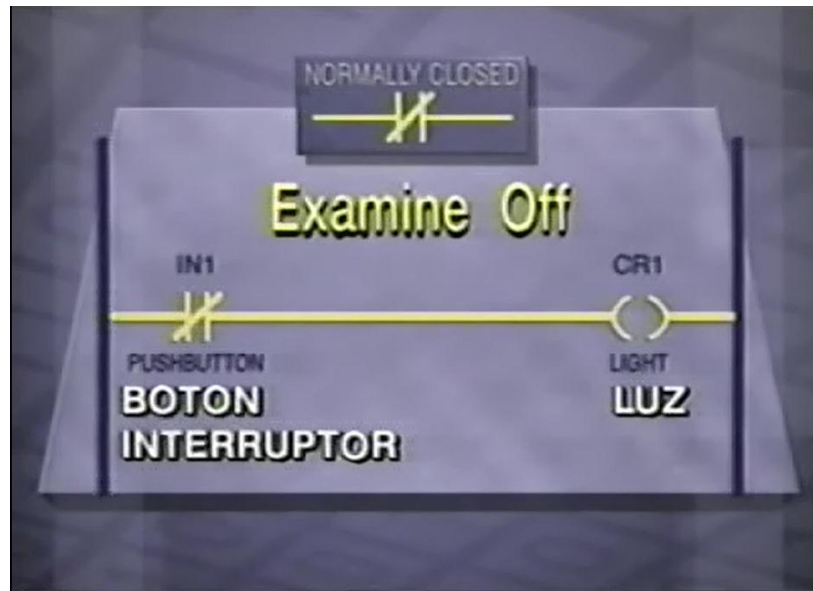


Fig 2.6 Peldaño que controla una luz con un botón normalmente cerrado.

El mecanismo es movido por un electroimán, cuando el electroimán es energizado el mecanismo de contacto se mueve haciendo que los contactos normalmente abiertos se cierren y que todos los contactos normalmente cerrados se abran. Cuando el electroimán o bobina de contactos son energizados se dice que el relé está energizado. Cuando se elimina la energía de la bobina se dice que el relé está desconectado.



Fig 2.7 Fotografía de la estructura de un relevador.

Pueden haber grupos múltiples de salidas normalmente abiertas o normalmente cerradas asociados con una bobina o carrete de relevo, estos contactos pueden ser usados a lo largo del diagrama discreto para controlar otros escalones según lo requiera el programa. Estos 3 símbolos pueden ser usados conjuntamente para crear lógica.

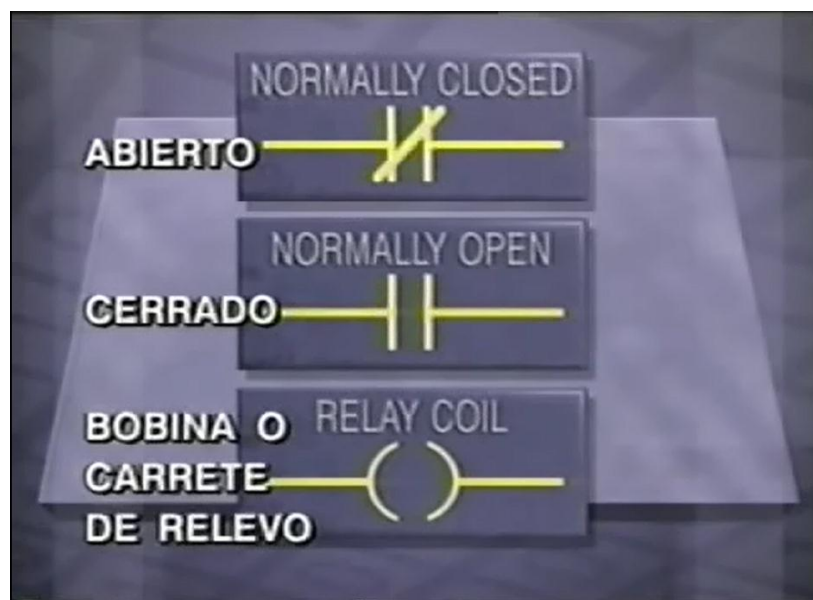


Fig 2.8 Símbolos para contactores y bobina de relevo.

Diferentes combinaciones de contactos normalmente abiertos o cerrados, en serie o en paralelo tienen salidas diferentes.

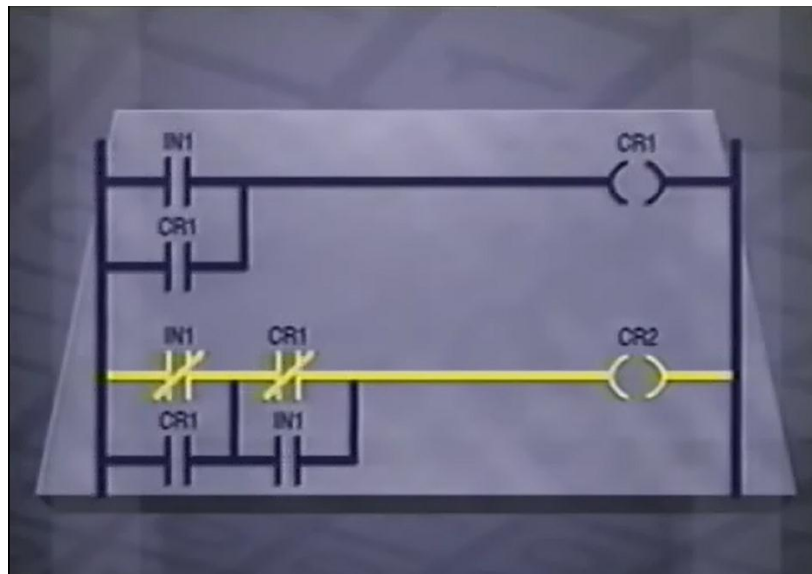


Fig 2.9 Circuito combinacional para un carrito de relevo.

La respuesta de este sistema a las diversas entradas, dependerá del programa del PLC según el diagrama. La programación comenzaría con el botón de parada conectado a la entrada 1 ó IN 1, luego programamos la sobrecarga IN 3, luego programamos el botón de arranque o IN 2, la bobina OUT 1 controlará el motor de arranque o bobina de relevo M, un contacto asociado con esta bobina puede ser programado en paralelo con IN 2, la entrada del botón de arranque para mantener la bobina energizada después de soltar el botón de arranque. El escalón es completado con la programación de la bobina OUT 1.

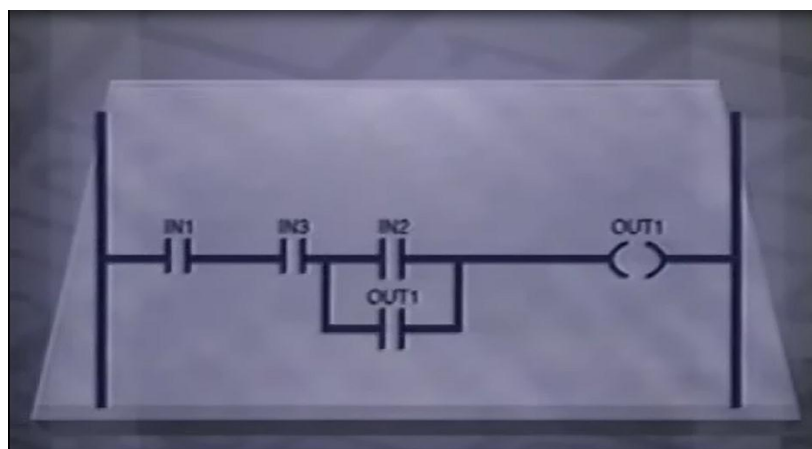


Fig 2.10 Diagrama de escalera y peldaño con auto alimentación OUT1.

2.2 FUNCIONES DE LÓGICA

Hay 3 tipos de lógica asociada con la programación de los PLC, son AND serie, OR paralelo y NOT negativo.

La lógica AND es tan simple como su nombre lo indica y se refiere a grupo de contactos en serie, considere estos 2 contactos normalmente abiertos en serie, cuando el contacto IN 1 e IN 2 están cerrados la bobina de relevo es energizada, la expresión booleana para esta expresión en serie se escribe como vemos aquí, el relevo es igual al contacto 1 multiplicado por el contacto 2, la multiplicación tiene el símbolo de la cruz, es usado para indicar la función de serie.

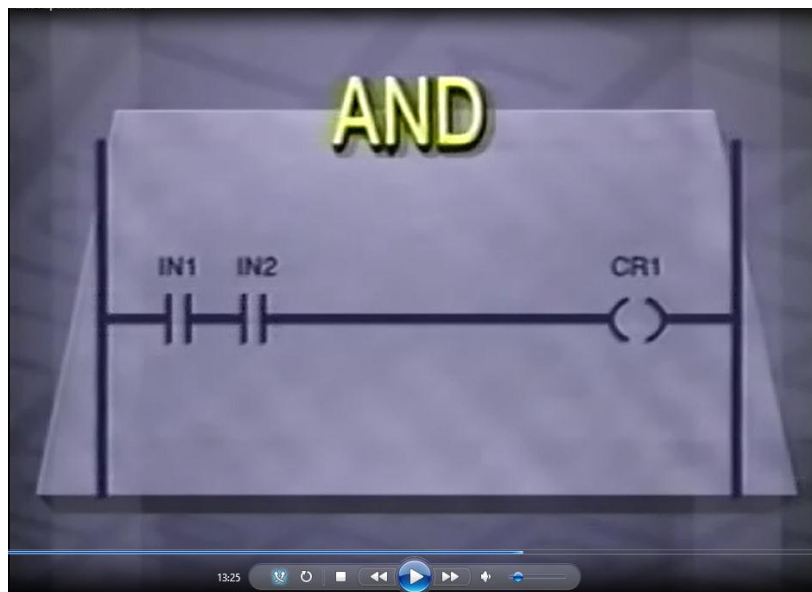


Fig 2.11 Contactos normalmente abiertos en serie con CR1 salida del peldaño.

La lógica OR o de paralelo también funciona como lo indica su nombre, se refiere a un grupo de contactos en paralelo, consideremos estos dos contactos en paralelo normalmente abiertos, cuando el contacto IN 1 ó IN 2 está cerrado la bobina es energizada. La expresión booleana para esta función de paralelo se escribe relevo es igual a contacto 1 mas el contacto 2, el símbolo + se usa para indicar la función de paralelo.

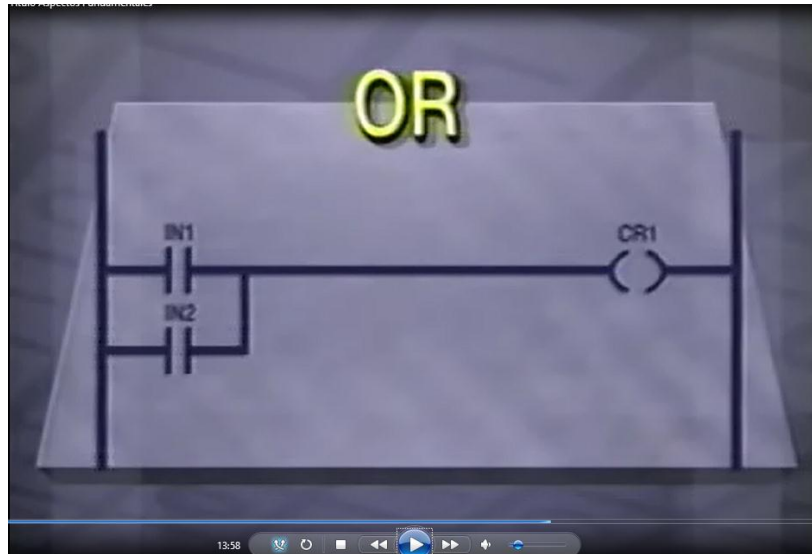


Fig 2.12 Contactos normalmente abiertos en paralelo.

La lógica de NOT o de negativo se refiere a un contacto normalmente cerrado, al energizar el contacto asociado con el contacto en este escalón del diagrama, la bobina no será energizada. el símbolo booleano para la lógica NOT se escribe con una barra sobre el identificador, relevo igual a $\overline{IN1}$. En este caso NOT significa que el contacto normalmente cerrado si la entrada o la bobina asociada con el contacto no está energizada.

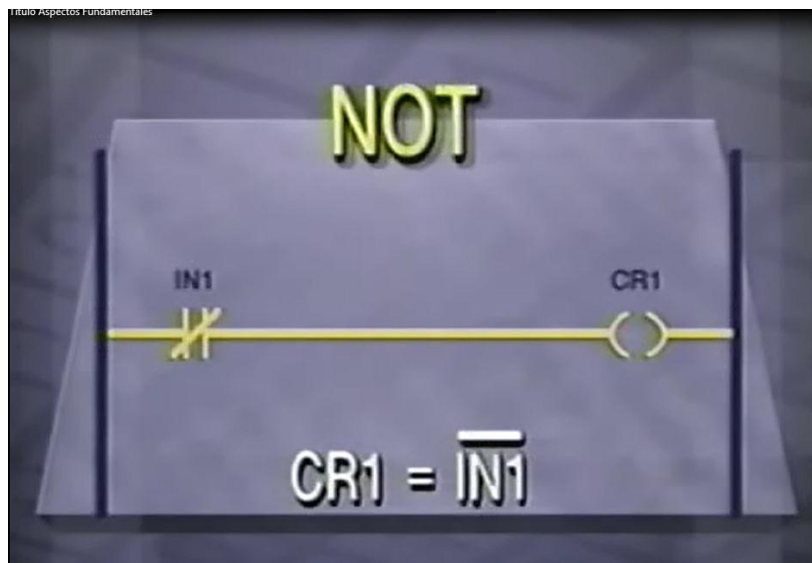


Fig 2.13 Contacto normalmente cerrado y control de salida del peldaño en CR1.

Es importante notar que en comparación a los diagramas eléctricos el flujo de corriente en un diagrama discreto PLC debe seguir ciertas normas. La corriente únicamente puede ir de izquierda a derecha y de arriba hacia abajo. Como regla general la corriente no puede fluir de derecha a izquierda en un diagrama discreto de PLC, a esta convención se le conoce como SCAN o barrida.

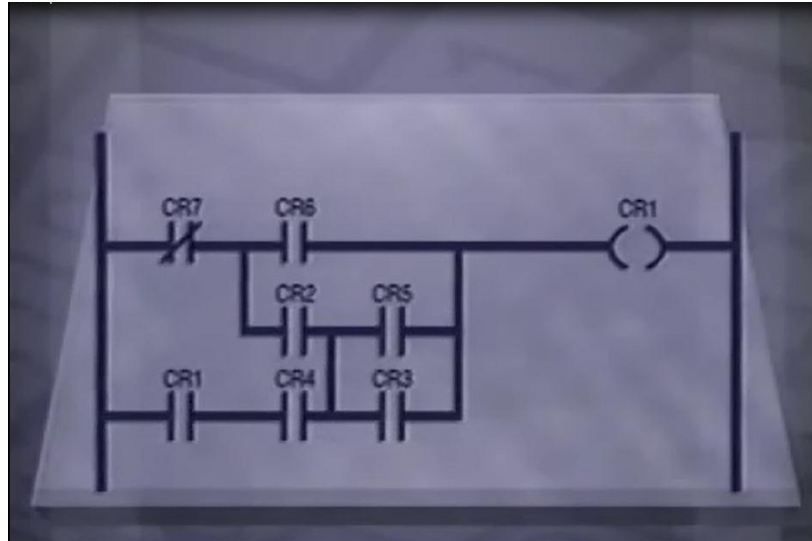


Fig 2.14 Estructura combinacional de 7 contactores y un carrete de salida CR1.

Cuando el PLC lee todo el programa en la memoria es una barrida o SCAN del programa, el PLC hace una barrida de las entradas y mantiene los resultados en la tabla de condición interna, una vez completada la barrida del programa la tabla de salida es actualizada y los datos son leídos y transferidos al terminal o módulo de salida.

2.3 SISTEMAS NUMÉRICOS

Frecuentemente se requiere que los PLC's actúen sobre los datos presentados como valores numéricos, estos datos están en forma binaria, octal, hexadecimal o en formato BCD decimal con código binario. La forma o formas usadas dependen de la capacidad del PLC.

Los datos binarios vienen en forma de un número estándar binario de base 2 representado por una serie de unos y ceros. El sistema octal requiere de tres dígitos binarios o bits, que le permiten contar de 0 al máximo de 7 para un total de 8 dígitos. Para convertir del sistema binario al octal consideramos el equivalente binario 30_{10} es igual a 16 por 1 mas 8 por 1 mas 4 por 1 mas 2 por 1 mas 1 ó 11110_2 .



2.15 Secuencia del procedimiento de detector del PLC para variables.

El equivalente octal es determinado agrupando los bits en grupos de 3 igualmente 011 y 110 el cero inicial se añade para ilustrar la agrupación de los bits. El grupo a la izquierda de 011 tienen un equivalente binario de 3 y el grupo a la derecha o grupo menos significativo 110 tiene un equivalente binario de 6 . El equivalente octal es 36_8 . Así tenemos que los equivalentes en las bases son $30_{10} = 11110_2 = 36_8$.

Para convertir un número octal a binario decodificamos cada dígito del octal al equivalente binario y los combinamos, si queremos decodificar el valor octal 127_8 a binario, reemplazamos cada dígito en octal por su equivalente binario. El 1 equivale a 001 , 2 es igual a 010 y 7 equivale a 111 . El equivalente binario del octal $127_8 = 001010111$, los ceros iniciales pueden ser eliminados produciendo el número binario 1010111_2 .

La conversión del sistema binario al hexadecimal se hace de la misma manera como para la conversión de un octal.

Cada dígito hexadecimal puede contener cualquiera de 16 números 0 a F, cualquiera de los símbolos numéricos 0 a 9, 10 a 15 es representado por las letras A hasta F. Debido que se requieren 4 bits para representar del 0 al 15 con base 2, el valor hexadecimal equivalente puede ser obtenido colocando los bits en grupos de 4, comenzando con el bit a la derecha o sea el menos significativo, por ejemplo al colocar los bits en grupos de 4 el número binario 1011011 nos da 0101 y 1011, de nuevo se ha colocado un 0 inicial para hacer que el número de bits sea igual a un múltiplo de 4.

El grupo más significativo 0101 es el equivalente binario del número decimal 5 y el grupo a la derecha menos significativo es equivalente al número decimal 11. El número 11 es representado en hexadecimal como la letra B. Como resultado el equivalente hexadecimal del número binario 1011011_2 es $5B_{16}$. La pantalla del programador portátil puede indicar el valor de un registro de 16 bits como 4 dígitos hexadecimales.

El sistema BCD o de decimales en código binario usa bits para representar los dígitos decimales, el mayor valor de un solo dígito en el sistema decimal es 9, se requieren 4 bits para representar el valor 9_{10} es 1001_2 , el usar 4 bits permite representar todos los valores numéricos del 0 al 9. BCD usa 4 bits por cada dígito decimal que va a ser representado, por ejemplo un número decimal de 999 requiere 12 bits, 3 grupos de 4 para representar el número en BCD, el equivalente BCD para 999_{10} es 1001, 1001, 1001. El valor BCD 1000, 0101, 0011 es 853_{10} el valor decimal.

Este tipo de sistema BCD es comúnmente utilizado para islar los valores de los switches controladores digitales en el PLC y para transferir los datos digitales a las pantallas LED o LCD desde el PLC. Normalmente una calculadora de bolsillo tiene estas conversiones incluidas, pero el saber como usar este sistema numérico requiere saber distinguir entre ellos, puede ser muy útil al trabajar con los PLC. Aquí hemos visto los componentes de un PLC típico, la estructura y los símbolos de un diagrama discreto o de escalera, aspectos

básicos de la lógica de escalera y los diferentes sistemas numéricos que podemos encontrar al trabajar con PLC's. Al entender los aspectos fundamentales de los PLC's podemos programar y conseguir objetivos mas avanzados.

2.4 LENGUAJE LADDER

Lenguaje de escalera o LADDER es muy fácil desarrollar un programa en esta clase de lenguaje, es el lenguaje de escalera y así es conocido regularmente, este concepto de visual es muy extendido a los PLC que son conocidos como autómatas programables, debido que su estructura de escalera del LADDER es igual a los diagramas de control clásico o convencional. De este modo el conocimiento de nivel técnico eléctrico se tienen para el control convencional.

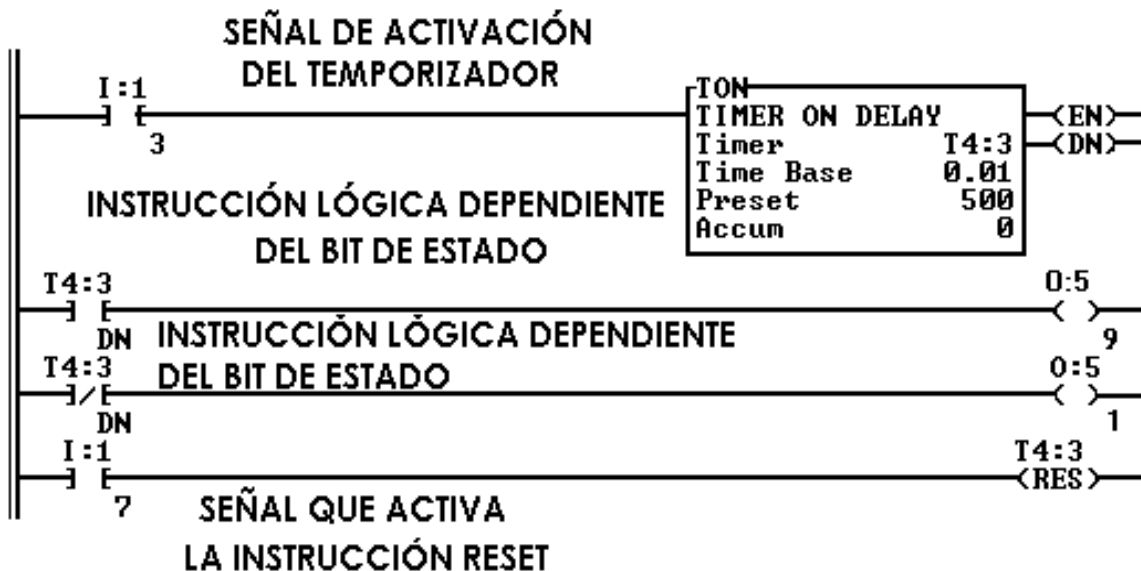


Fig 2.16 Programa básico de un temporizador de demora activa.

Los elementos de programación para programar un PLC con LADDER debemos entender y conocer las normas de los circuitos de conmutación, es indispensable conocer por separado cada componente del que consta el lenguaje del diagrama de escalera. En esta tabla ponemos los símbolos de dispositivos más utilizados en la programación.



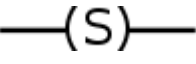
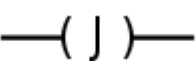
Elementos básicos en LADDER		
Símbolo	Nombre	Descripción
	Contacto NA	Se activa cuando hay un uno lógico en el elemento que representa, esto es, una entrada (para captar información del proceso a controlar), una variable interna o un bit de sistema.
	Bobina NC	Se activa cuando la combinación que hay a su entrada (izquierda) da un cero lógico. Su activación equivale a decir que tiene un cero lógico. Su comportamiento es complementario al de la bobina NA.
	Bobina SET	Una vez activa (puesta a 1) no se puede desactivar (puesta a 0) si no es por su correspondiente bobina en RESET. Sirve para memorizar bits y usada junto con la bobina RESET dan una enorme potencia en la programación.
	Bobina JUMP	Permite saltarse instrucciones del programa e ir directamente a la etiqueta que se desee. Sirve para realizar subprogramas.

Fig 2.17 elementos básicos del diagrama de escalera.

Su nomenclatura diversa, dependiendo siempre de tipo de autómeta y fabricante. Existe una gran variedad, siendo los más importantes los de arranque y los de reloj, que permiten que empiece la ejecución desde un sitio en concreto y formar una base de tiempos respectivamente. Los bits de sistema son contactos que el propio autómeta activa cuando conviene o cuando se dan unas circunstancias determinadas.

Su utilidad fundamental es la de almacenar información intermedia para simplificar esquemas y programación. Su número de identificación suele oscilar, en general, entre 0 y 255. Se suele indicar mediante los caracteres B ó M y tienen tanto bobinas como contactos asociados a las mismas de los tipos vistos en el punto anterior.

Temporizadores

El símbolo elemental del temporizador es distinto de uno a otro PLC, pero siempre encontramos un grupo de señales fundamentales aunque eso si con nombres totalmente diferentes. El temporizador es un elemento que posibilita la cuenta en el tiempo con el objeto de accionar bobinas pasado un determinado tiempo desde la activación.

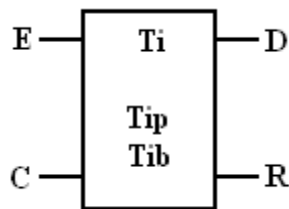


Fig 2.18 Símbolo para un contador en el tiempo.

Podemos observar, en la figura el esquema de un temporizador, **Tii**, con 2 salidas (D y R a la derecha con las siguientes características) y 2 entradas (E y C a la izquierda):

- **Entrada Enable (E):** ya que si se desactiva (puesta a cero lógico) se interrumpiría la cuenta de tibia (puesta a cero temporal), tiene que estar activa (a 1 lógico) en todo momento durante el intervalo de tiempo.

Contadores

Es un elemento que puede contabilizar el cómputo de los accionamiento de sus entradas, es muy útil para registrar en memoria eventos que no tengan que ver con el tiempo, tiene varias entradas de activación y algunas salidas hasta 3, pero se requieran realizar una cierta cantidad de veces.

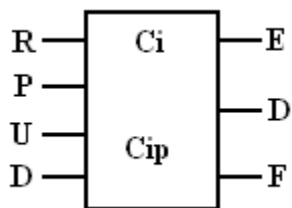


Fig 2.19 Símbolo para un contador con terminales de control.

En la figura puede verse el esquema de un contador, Ci, bastante usual, donde pueden distinguirse las siguientes entradas y salidas:

- **Entrada RESET (R):** En cada activación de esta lleva a cero el contador. Se suele emplear en el inicio de la ejecución dándole bits de arranque, de modo que quede a cero cada vez que se arranca el sistema.
- **Entrada PRESET (P).** Permite llevar la cuenta del contador a un valor determinado distinto de cero, que previamente está programado en Cip.
- **Entrada UP (U):** Cada vez que se acciona o solicita produce un incremento en una unidad de la cuenta que posea en ese momento el contador.
- **Entrada DOWN (D):** Cada vez que se activa produce un decremento en una unidad de la cuenta que posea en ese momento el contador.
- **Salida FULL (F):** Se activa al producirse un desbordamiento del valor del contador contando en sentido ascendente.
- **Salida DONE (D):** Se activa cuando el valor del contador se iguala al valor preestablecido Cip.
- **Salida EMPTY (E):** Se activa al producirse un desbordamiento del valor del contador contando en sentido descendente.

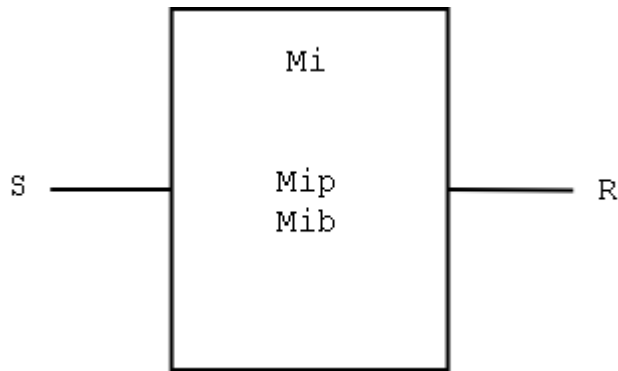


Fig 2.20 Símbolo de un monoestable que retiene su salida un tiempo definido.

Una de sus mejores características es su simplicidad al poseer una única entrada y una sola salida según se observa en la siguiente figura. Es un elemento que puede mantenerse en un determinado estado activando un a entrada durante el tiempo programado, desactivándose automáticamente al concluir tal intervalo de tiempo.

- **Entrada START (S):** Cuando se activa o se le proporciona un impulso comienza la cuenta que tiene programada.
- **Salida RUNNING (R):** Se mantiene activada mientras dura la cuenta y se desactiva al finalizarla. Al igual que con el temporizador, para programar la cuenta hay que introducir los valores de Mip y Mib.

En la programación en esta parte llevamos o exponemos lo mejor posible los conceptos básicos de la programación en escalera. Como ya hemos visto los elementos del diagrama de escalera ofrece según se soliciten al programar, hay que poner el claro la estructura de un programa y su secuencia de ejecución.

El siguiente diagrama representa la estructura general de la distribución de todo programa LADDER, contactos a la izquierda y bobinas y otros elementos a la derecha en renglones o líneas o peldaños.

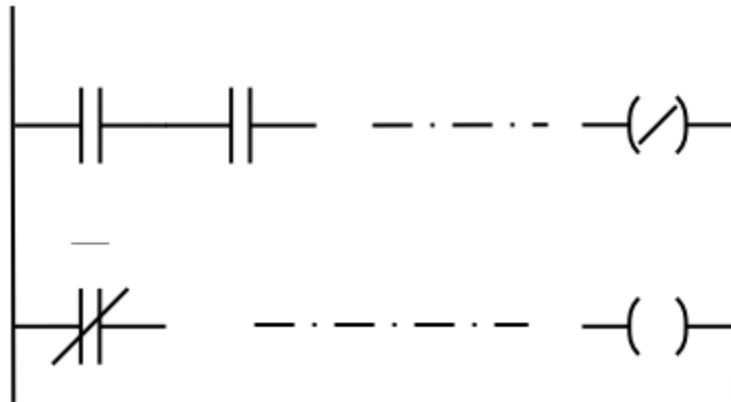


Fig 2.21 Diagrama de escalera de 2 peldaños y combinacional.

La distribución de un programa, la barra vertical ó línea izquierda equivale al lado de alimentación o vivio de la fuente de energía eléctrica, mientras que el lado de la derecha es una línea vertical que representa el terminal de alimentación.

El orden de realización del programa es de arriba a abajo y de izquierda a derecha conforme se va leyendo, primero los elementos de la izquierda del peldaño y luego el elemento de salida a la derecha o bobina, de manera que ya se conoce el valor de los elementos y se activan si procede al llegar a la bobina. La secuencia de realización del programa puede variar de un PLC a otro de manera que se ejecuta lo que primero se introduce, pero siempre se respetará el orden de introducción del programa.

Sistemas combinacionales

En la conmutación se aplica el álgebra de Boole en ecuaciones, la suma es el contacto en paralelo u operación OR, la multiplicación queda definida por los contactos en serie y es AND, la operación de negación NOT está definida por contactos normalmente cerrados. Al conseguir la función lógica de un grupo de contactos asociados es un combinacional, el diagrama de escalera o esquema de contactos es sencillo. En la siguiente figura se muestra un ejemplo de digrama de escalera para una ecuación de Boole.

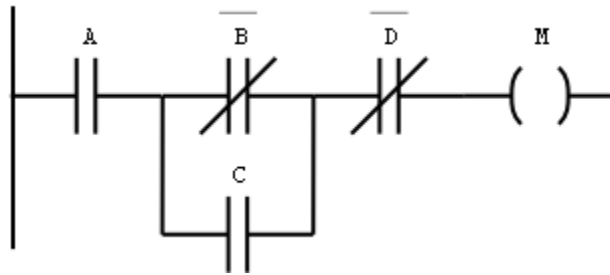


Fig 2.22 Diag. De escalera para la función $M = A(B'+C)D'$.

En los sistemas industriales aunque la programación se centra en procesos secuenciales es necesario conocer la lógica combinacional, no teniendo demasiado interés en los procesos combinacionales es necesaria la programación secuencial no teniendo demasiado interés la lógica combinacional.

Elementos de memoria

La conexión tradicional para realizar una función de memoria en los circuitos con relés, es el circuito con autoalimentación. Esto se consigue mediante la conexión de un contacto NA del relé (o contactor) en paralelo con el pulsador de marcha. A continuación puede observarse las dos variantes de este circuito: con prioridad a la Desconexión (figura a) y con prioridad a la conexión (figura b).

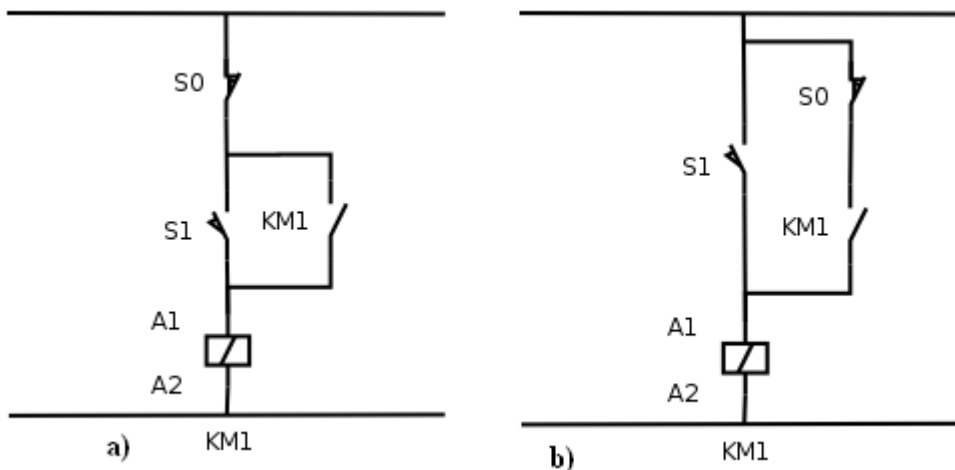


Fig 2.23 Diagrama de circuito de retención de estado o memoria.

Circuitos con autoalimentación con prioridad a la desconexión a) y a la conexión b)

En la siguiente figura se pueden observar los sus esquemas equivalente en LADDER:

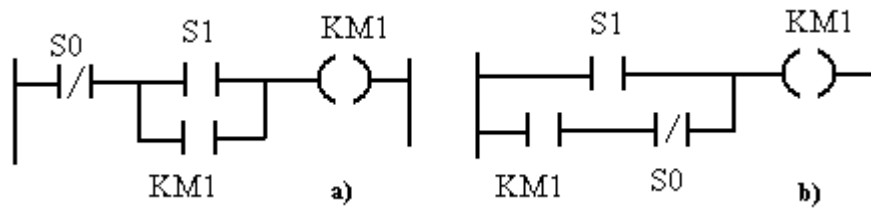


Fig 2.24 Circuitos LADDER con autoalimentación.

Sin embargo, con LADDER el esquema puede quedar mucho más sencillo si empleamos las bobinas de SET para la marcha y RESET para paro:

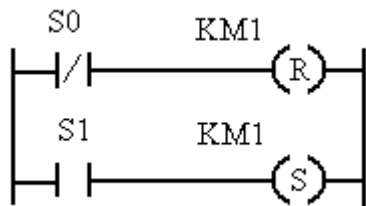


Fig 2.25 Circuito de retención de estado usando bobina R.

Circuito de marcha y paro con bobinas SET y RESeT

En este caso la prioridad dependerá del PLC utilizado, aunque usualmente la función RESeT tiene prioridad sobre la SET.

Autómata programable

En electrónica un **autómata** es un sistema secuencial, aunque en ocasiones la palabra es utilizada también para referirse a un robot. Puede definirse como un equipo electrónico programable en lenguaje no informático y diseñado para controlar, en tiempo real y en ambiente industrial, procesos secuenciales. Sin embargo, la rápida evolución de los autómatas hace que esta definición no esté cerrada.

Aplicaciones

Como ya se ha comentado, las primeras aplicaciones de los autómatas programables se dieron en la industria automotriz para sustituir los complejos equipos basados en relés. Sin embargo, la disminución de tamaño y el menor costo han permitido que los autómatas sean utilizados en todos los sectores de la industria. Sólo a modo de ejemplo, se mencionan a continuación algunos de los múltiples campos de aplicación.

Automóvil

- Cadenas de montaje, soldadura, cabinas de pintura, etc.
- Máquinas herramientas: Tornos, fresadoras, taladradoras, etc.

Plantas químicas y petroquímicas

- Control de procesos (dosificación, mezcla, pesaje, etc).
- Baños electrolíticos, oleoductos, refinado, tratamiento de aguas residuales, etc.

Metalurgia

- Control de hornos, laminado, fundición, soldadura, forja, grúas, entre otros.

Alimentación

- Envasado, empaquetado, embotellado, almacenaje, llenado de botellas, etc.

Papeleras y madereras

- Control de procesos, serradoras, producción de conglomerados y de laminados, etc.a

Producción de energía

- Centrales eléctricas, turbinas, transporte de combustible, energía solar, etc.

Tráfico

- Regulación y control del tráfico, ferrocarriles, líneas de metro, etc

Domótica

- Iluminación, temperatura ambiente, sistemas anti robo, comodidad y bienestar en el hogar, etc.

Fabricación de Neumáticos

- Control de calderas, sistemas de refrigeración , prensas que vulcanizan los neumáticos.
- Control de las máquinas para el armado de las cubiertas, extrusoras de goma.
- Control de las máquinas para mezclar goma.

El autómata programable estructura general

Un autómata programable se puede considerar como un sistema basado en un microprocesador, siendo sus partes fundamentales la Unidad Central de Proceso (CPU), la Memoria y el Sistema de Entradas y Salidas (E/S).

La CPU realiza el control interno y externo del autómata y la interpretación de las instrucciones del programa. A partir de las instrucciones almacenadas en la memoria y de los datos que recibe de las entradas, genera las señales de las salidas. La memoria se divide en dos bloques, la memoria de solo lectura o ROM (Read Only Memory) y la memoria de lectura y escritura o RAM (Random Access Memory).

En la memoria ROM se almacenan programas para el correcto funcionamiento del sistema, como el programa de comprobación de la puesta en marcha y el programa de exploración de la memoria RAM.

La memoria RAM a su vez puede dividirse en dos áreas:

- A. Memoria de datos, en la que se almacena la información de los estados de las entradas y salidas y de variables internas.
- B. Memoria de usuario, en la que se almacena el programa con el que trabajará el autómata.

El sistema de Entradas y Salidas recoge la información del proceso controlado (Entradas) y envía las acciones de control del mismo (salidas). Los dispositivos de entrada pueden ser pulsadores, interruptores, finales de carrera, termostatos, presostatos, detectores de nivel, detectores de proximidad, contactos auxiliares, etc.

Por su parte, los dispositivos de salida son también muy variados: Pilotos indicadores, relés, contactores, arrancadores de motores, válvulas, etc. En el siguiente punto se trata con más detalle este sistema.

Sistema de entradas y salidas

En general, las entradas y salidas (E/S) de un autómatas pueden ser discretas, analógicas, numéricas o especiales.

Las E/S discretas se caracterizan por presentar dos estados diferenciados: presencia o ausencia de tensión, relé abierto o cerrado, etc. Su estado se puede visualizar mediante indicadores tipo LED que se iluminan cuando hay señal en la entrada o cuando se activa la salida. Los niveles de tensión de las entradas más comunes son 5 V cc, 24 V cc/ca, 48 V cc/ca y 220 V ca.

Los dispositivos de salida más frecuentes son relés, transistores y triacs.

Las E/S analógicas tienen como función la conversión de una magnitud analógica (tensión o corriente) equivalente a una magnitud física (temperatura, presión, grado de acidez, etc.) en una expresión binaria de 11, 12 o más bits, dependiendo de la precisión deseada. Esto se realiza mediante conversores analógico-digitales (ADC's).

Las E/S numéricas permiten la adquisición o generación de información a nivel numérico, en códigos BCD, Gray u otros (véase código binario). La información numérica puede ser entrada mediante dispositivos electrónicos digitales apropiados. Por su parte, las salidas numéricas suministran información para ser utilizada en dispositivos visualizadores (de 7 segmentos) u otros equipos digitales.

Por último, las E/S especiales se utilizan en procesos en los que con las anteriores E/S vistas son poco efectivas, bien porque es necesario un gran número de elementos

adicionales, bien porque el programa necesita de muchas instrucciones. Entre las más importantes están:

- A. Entradas para termopar y termorresistencia: Para el control de temperaturas.
- B. Salidas de trenes de impulso: Para el control de motores paso a paso (PAP).
- C. Entradas y salidas de regulación P+I+D (Proporcional + Integral + Derivativo): Para procesos de regulación de alta precisión.
- D. Salidas ASCII: Para la comunicación con periféricos inteligentes (equipo de programación, impresora, PC, etc.).

Cuando se pone en marcha el PLC se realizan una serie de comprobaciones:

- A. Funcionamiento de las memorias.
- B. Comunicaciones internas y externas.
- C. Elementos de E/S.
- D. Tensiones correctas de la fuente de alimentación.

Una vez efectuadas estas comprobaciones y si las mismas resultan ser correctas, la CPU inicia la exploración del programa y reinicializa. Esto último si el autómatas se encuentra en modo RUN (marcha), ya que de estar en modo STOP (paro) aguardaría, sin explorar el programa, hasta la puesta en RUN.

Al producirse el paso al modo STOP o si se interrumpe la tensión de alimentación durante un tiempo lo suficientemente largo, la CPU realiza las siguientes acciones:

- Detiene la exploración del programa.
- Pone a cero, es decir, desactiva todas las salidas.

Mientras se está ejecutando el programa, la CPU realiza en sucesivos intervalos de tiempo distintas funciones de diagnóstico (watch-dog en inglés). Cualquier anomalía que se detecte se reflejará en los indicadores de diagnóstico del procesador y dependiendo de su importancia se generará un código de error o se parará totalmente el sistema.

El tiempo total del ciclo de ejecución viene determinado por los tiempos empleados en las distintas operaciones. El tiempo de exploración del programa es variable en función de la cantidad y tipo de las instrucciones así como de la ejecución de subrutinas. El tiempo de exploración es uno de los parámetros que caracteriza a un PLC y generalmente se suele expresar en milisegundos por cada mil instrucciones. Para reducir los tiempos de ejecución, algunas CPU's constan de dos o más procesadores que operan simultáneamente y están dedicados a funciones específicas. También se puede descargar de tareas a la CPU incorporando módulos inteligentes dedicados a tareas específicas.

Equipos de programación, la misión principal de los equipos de programación, es la de servir de interfaz entre el operador y el autómatas para introducir en la memoria de usuario el programa con las instrucciones que definen las secuencias de control.

Dependiendo del tipo de autómatas, el equipo de programación produce unos códigos de instrucción directamente ejecutables por el procesador o bien un código intermedio, que es interpretado por un programa residente en el procesador (firmware).

Las tareas principales de un equipo de programación son:

- A. Introducción de las instrucciones del programa.
- B. Edición y modificación del programa.
- C. Detección de errores.
- D. Archivo de programas (cintas, discos).

Básicamente existen tres tipos de equipos de programación:

- A. Consola con teclado y pantalla de tubo de rayos catódicos (CRT) o de cristal líquido (LCD).

- B. Programador manual, semejante a una calculadora de bolsillo, más económico que la anterior.
- C. Ordenador personal con el software apropiado.

La conexión de la consola u ordenador al autómata programable se realiza mediante una conexión en serie (generalmente la RS-232C o la RS-422).

Equipos periféricos

Además de los equipos de programación, existen numerosos dispositivos que sin formar parte directa del autómata, pueden conectarse al mismo para realizar distintas funciones. Normalmente se conectan a las salidas ASCII o a los canales de comunicación del autómata.

Seguidamente se describen algunos de los equipos periféricos más comunes:

- A. Módulos de ampliación de entradas y salidas: Necesarios para aquellos procesos en los que la estructura de E/S del autómata sea insuficiente.
- B. Módulos de tratamiento de datos: Son pequeños ordenadores que manejan distintos datos (contaje, tiempo, estado de E/S, etc.), para la elaboración de informes, gráficos, etc.
- C. Impresoras.
- D. Visualizadores alfanuméricos.
- E. Lectores de código de barras.

La forma de comunicarse el autómata con sus periféricos puede ser unidireccional, cuando se establece en un sólo sentido, o bien bidireccional, cuando se establece en los dos sentidos. Los enlaces para ambos tipos de comunicación suelen ser por lo general del tipo serie, siendo los más empleados los anteriormente mencionados RS-232C y RS-422, ambos de acuerdo con las normas de la EIA (Electronic Industries Association). El RS-232C es el

método de transmisión de datos más difundido, pero tiene la limitación de la distancia máxima de transmisión a 15 metros y la velocidad máxima de transmisión de 19.200 baudios (1 baudio = 1 bit/segundo). El RS-422 resuelve en parte las limitaciones del RS-232C. La distancia de transmisión puede superar un kilómetro y la velocidad puede llegar a 10 Mbaudios.

Programación del autómata, controlar un determinado proceso, el autómata realiza sus tareas de acuerdo con una serie de sentencias o instrucciones establecidas en un programa. Dichas instrucciones deberán haber sido escritas con anterioridad por el usuario en un lenguaje comprensible para la CPU. En general, las instrucciones pueden ser de funciones lógicas, de tiempo, de cuenta, aritméticas, de espera, de salto, de comparación, de comunicación y auxiliares. Dependiendo del fabricante, los lenguajes de programación son muy diversos, sin embargo, suelen tener alguna relación más o menos directa con los lenguajes Ladder o GRAFCET.

Los programas para autómata pueden realizarse de forma lineal o de forma estructurada. En la programación lineal el programa consta de una serie de instrucciones que se van ejecutando una tras de otra de modo cíclico. Este modo de programación se suele emplear en programas no demasiado complejos o en autómatas que no posean el modo estructurado. Cuando los programas son muy complejos, la programación estructurada es más aconsejable ya que puede dividirse el proceso general en subprogramas con diferentes subprocesos tecnológicos. Otras de las ventajas de este modo de programación es que da un carácter más panorámico al programa, lo que conlleva una más fácil identificación de errores así como una mayor facilidad de comprensión por otros programadores.

Programar un autómata no es realmente algo imposible, pero sí se necesita paciencia. Como ejemplo tenemos un enlace en la sección de enlaces externos que conduce a una página que nos lleva a donde se encuentra un archivo hecho en java con código fuente para que se pueda analizar y comprender de una manera más sencilla cómo funciona un autómata finito determinístico (AFD).

2.5 PROGRAMACIÓN DE CONTACTOS

Un PLC al igual que cualquier computador únicamente puede cumplir las funciones para las cuales fue programado. La programación se logra al desarrollar un diagrama discreto al introducirlo a la memoria del PLC. Conocemos como programar contactos normalmente abiertos, normalmente cerrados, como programar varias funciones de lógica, como delegar puntos de entrada, salida y relevos internos y la documentación de programas.

Para programar información en un PLC se debe tener un diagrama discreto completo y libre de errores, en PLC's mas complejos el diagrama discreto puede ser dibujado en la pantalla de un computador. El programa puede traducir el diagrama a un código que es reconocido por el controlador, son este programa se requiere un programador portátil, al usar un programador portátil tendremos que traducir el diagrama discreto o de escalera en un código que el PLC pueda reconocer y luego pasarlo a la memoria del PLC. La mayoría de los PLC usan un lenguaje de programación similar pero esto no quiere decir que todos son iguales, algunos PLC Allen Bradley por ejemplo usan símbolos en el programador portátil que representan los diferentes elementos en el diagrama discreto.

Marcas tales como Texas Instruments usan códigos mnemónicos, que representan símbolos del diagrama discreto, el código mnemónico utilizado puede ser adaptado a un PLC específico porque cada función representada tiene una función similar en todos los PLC's. Se recomienda consultar el manual de referencia del fabricante para comparar los códigos del programa y la numeración de entradas y salidas para el PLC en particular.

Al programar un escalón primero debemos decirle al PLC que estamos comenzando un nuevo escalón. El comando LD le comunica este concepto al controlador, como resultado el comando para un contacto normalmente abierto identificado como IN 1 se logra al oprimir la tecla LD seguido por la tecla IN luego el número 1.

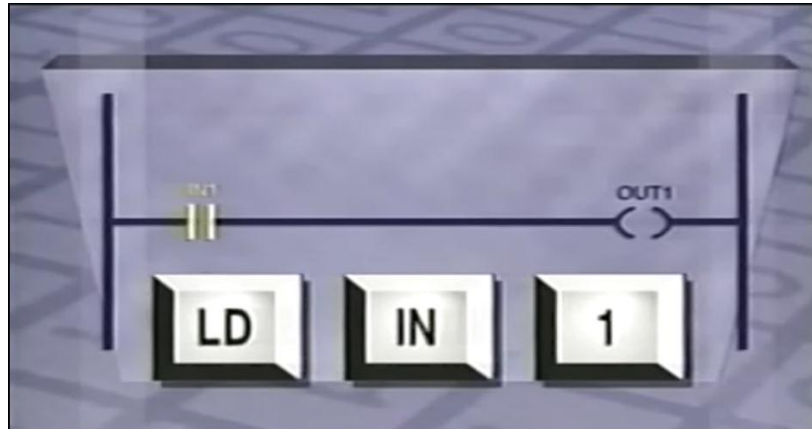


Fig 2.26 Código mnemónico primer renglón con OUT1.

El siguiente paso es programar la salida, la tecla STO le dice al PLC toda la información previa en los controles de escalón controla la bobina de escalón el comando de la tecla STO, para programar la salida en este escalón oprimimos la tecla STO seguido de la tecla OUT seguido por la tecla 1. Para decirle al PLC que el comando ha sido completado se usan códigos de terminación o finalización, los comandos LD son finalizados cuando otro comando LD o un comando STO son encontrados.

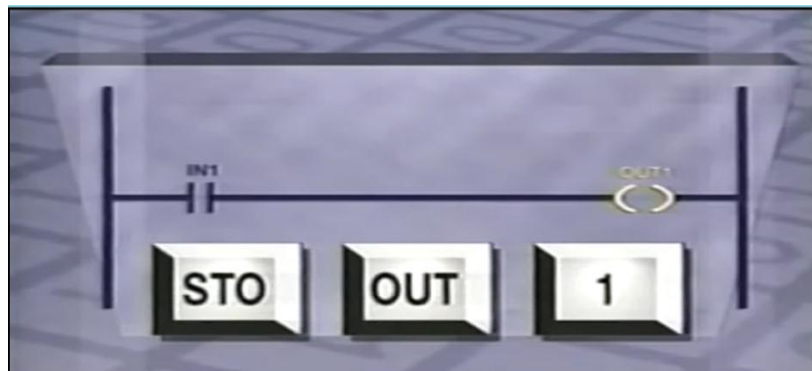


Fig 2.27 Símbolo mnemónico segundo renglón para OUT1.

ejemplo del diagrama discreto con la ecuación $OUT\ 1 = IN\ 1$ el código entero del programa sería LD IN 1 ENTER STO OUT 1 ENTER, si el contacto normalmente abierto es reemplazado por un circuito con un contacto normalmente cerrado, la ecuación booleana se

vería como complemento o barra. El programa queda escrito así LD NOT IN 1 ENTER STO OUT 1 ENTER.

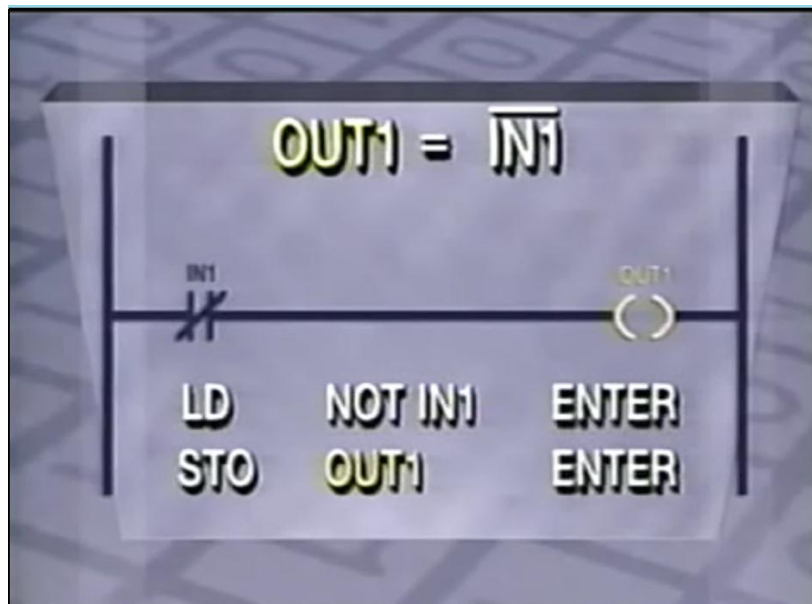


Fig 2.28 Códigos mnemónicos para un peldaño con interruptor normalmente cerrado.

Los códigos de programación para contactos normalmente abiertos y cerrados, pueden ser usados para implementar operaciones en serie o paralelo, usando la ecuación booleana para el diagrama discreto $OUT 1 = IN 1 \times IN 2$, el código mnemónico se vería así LD IN 1 ENTER AND IN 2 ENTER STO OUT 1 ENTER.

2.6 PROGRAMACIÓN DE FUNCIONES DE LÓGICA

Al reemplazar la función de serie por una función de paralelo, esta función se programa así, $OUT 1 = IN 1 + IN 2$, el código mnemónico se vería así LD IN 1 ENTER OR IN 2 ENTER STO OUT 1 ENTER.

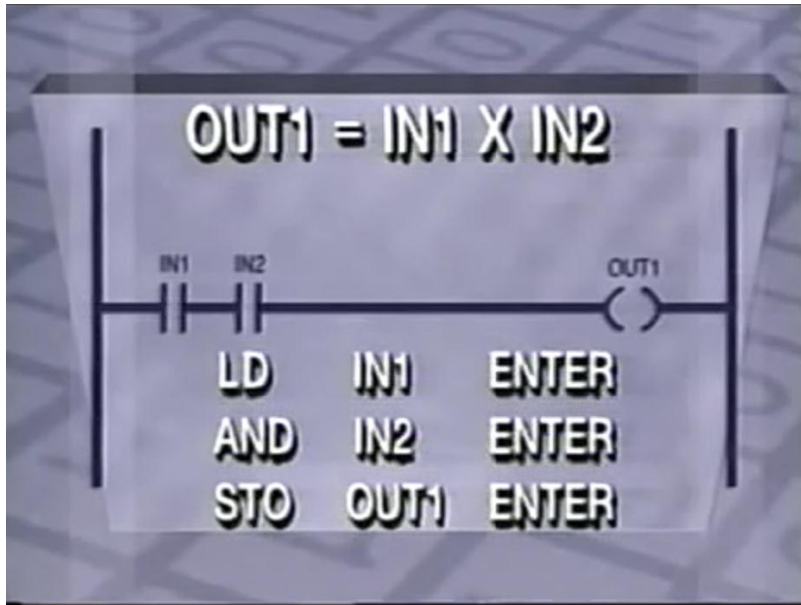


Fig 2.29 Código mnemónico para AND con 2 interruptores normalmente abiertos.

La ecuación lógica cuya bobina es la ecuación booleana $OUT\ 1 = NOT\ IN\ 1 + NOT\ IN\ 2$, el código mnemónico se vería así LD NOT IN 1 ENTER OR NOT IN 2 ENTER STO OUT 1 ENTER.

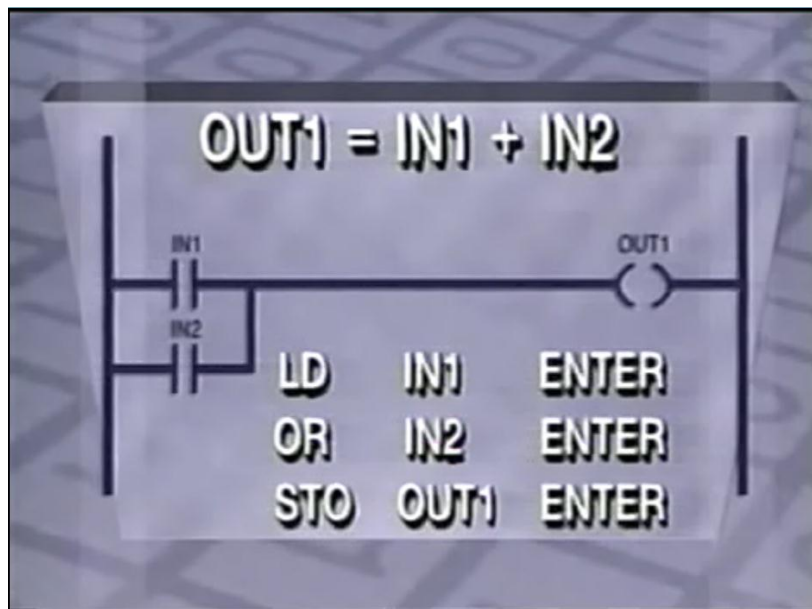


Fig 2.50 Código mnemónico para OR con dos interruptores normalmente abiertos.

Algunos fabricantes de PLC usan una sola tecla $OUT\ 1 = NOT\ IN\ 1 + NOT\ IN\ 2$, el código mnemónico se vería así LDI IN 1 ENTER ANI IN 2 ENTER STO OUT 1 ENTER.

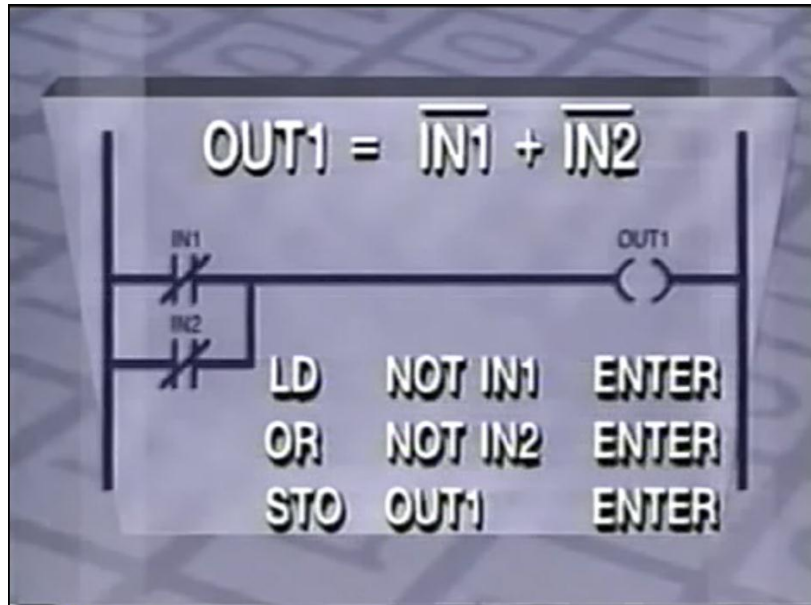
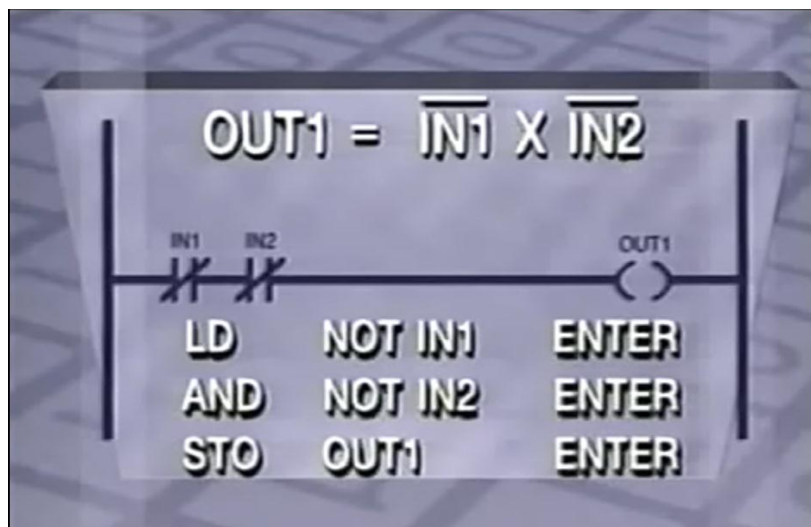


Fig 2.31 Código mnemónico para OR con 2 interruptores normalmente cerrados.

Una función lógica representada por esta ecuación booleana $OUT\ 1 = NOT\ IN\ 1 \times NOT\ IN\ 2$, el código mnemónico se vería así LD NOT IN 1 ENTER AND NOT IN 2 ENTER STO OUT 1 ENTER.



Código mnemónico para AND con 2 interruptores normalmente cerrados.

La memoria lógica o la función tipo latch como vemos aquí tiene un contacto controlado por la salida de la bobina en paralelo con el contacto IN 1. La ecuación booleana para esta función es $OUT\ 1 = IN\ 1 + OUT\ 1$, el código mnemónico se vería así LD IN 1 ENTER OUT OR 1 ENTER STO OUT 1 ENTER.

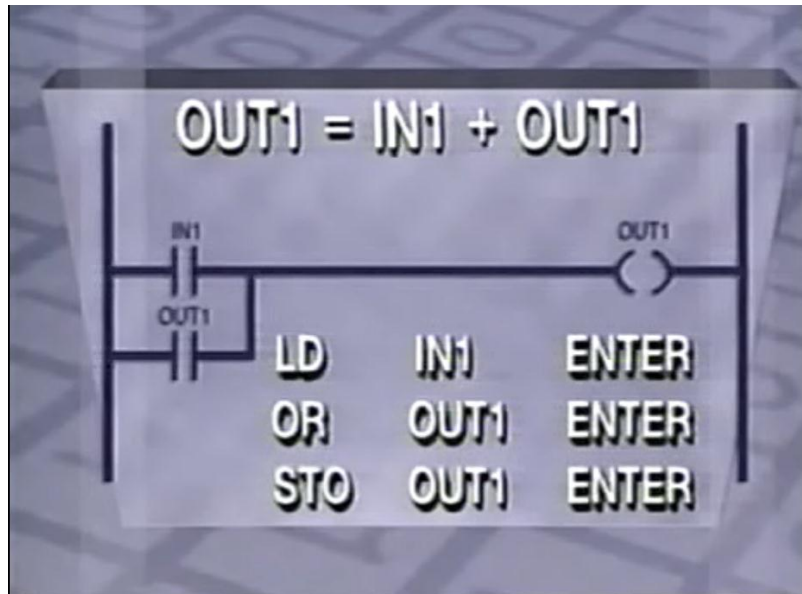


Fig 2.33 Código mnemónico para una autoalimentación.

Antes de ver programaciones mas complejas es necesario conocer la sección de memoria conocida como el STAK o unidad aisladora, esta es una sección de memoria reservada para operaciones mas complejas, tales como funciones de paralelo y serie de dos o mas grupos.

Consideramos el siguiente escalón considerando que IN 1, IN 2 e IN 3 están conectados en serie, IN 4, IN 5 e IN 6 están conectados en serie, finalmente los dos grupos de funciones en serie están conectados en paralelo y controlan la condición de la bobina OUT 1. La ecuación booleana para este escalón se ve $OUT1=(IN1 \times IN\ 2 \times IN3) + (IN4 \times IN5 \times IN6)$. Para programar este escalón o RAM programamos la primera función en serie LD IN 1 ENTER AND IN 2 ENTER AND IN 3 ENTER, luego introducimos la segunda función en serie comenzando con el comando LD, al introducir el comando LD toda la información previa para este RAM es colocada en el STACK, las diferentes secciones de comando LD asociadas con el, ahora programamos la segunda función AND como LD IN 4 ENTER AND IN 5 ENTER AND IN 6 ENTER.

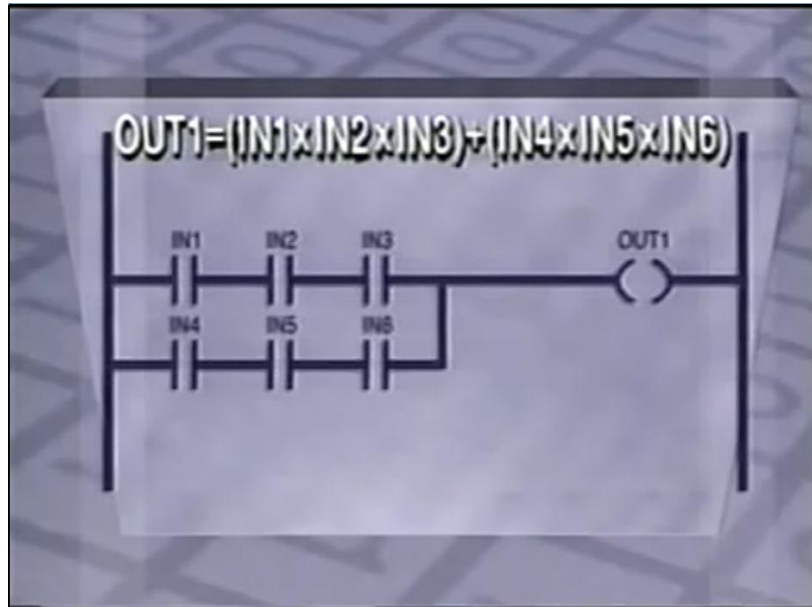


Fig 2.34 Diagrama de un peldaño con 2 ramas y 6 interruptores normalmente abiertos.

La segunda función AND y la primera función AND están ahora en el STACK, el comando para poner en paralelo lo que ha sido programado en el STACK es OR LD ENTER el comando OR LD le dice al PLC que ponga en paralelo lo que fue programado con todo lo que hay en el STACK hasta la primera función LD. El RAM es completado programando la bobina STO OUT 1 ENTER.

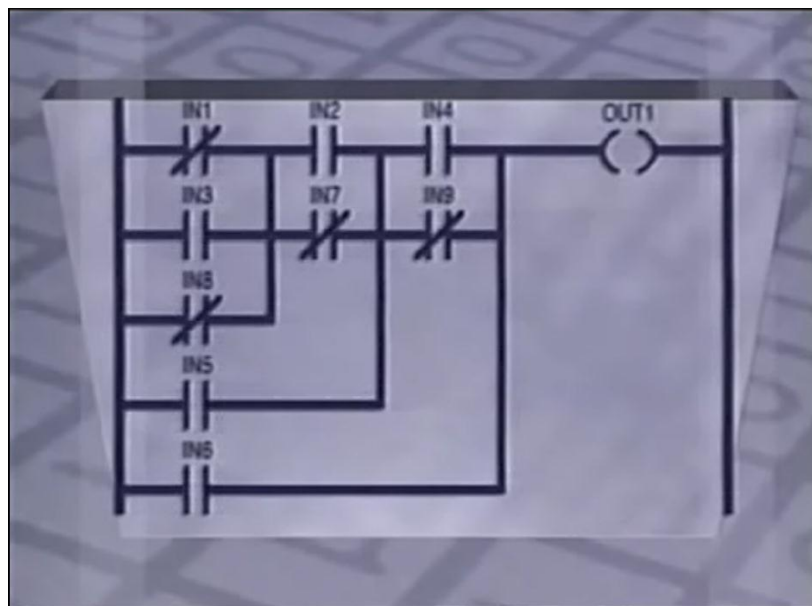


Fig 2.35 Diagrama combinacional de 1 peldaño con 9 interruptores.

Consideremos este diagrama usando los conceptos que ya vimos hasta ahora para programar un RAM complejo esta es la lista mnemónica completa.

```
LD NOT IN 1 ENTER
OR IN 3 ENTER
OR NOT IN 8 ENTER
LD IN 2 ENTER
OR NOT IN 7 ENTER
AND LD ENTER
OR IN 5 ENTER
LD IN 4 ENTER
OR NOT IN 9 ENTER
AND LD ENTER
OR IN 6 ENTER
STO OUT1 ENTER
```

El comando AND LD le dice al PLC que la información incluida debe ser conectada en serie con la información que acababa de ser incluida en el STACK . Las funciones mas comúnmente usadas en un PLC son los relojes controladores y los contadores, debido que los métodos de programación para estas funciones varían según el fabricante, usaremos un método de programación genérico. Los relojes de control son usados para controlar funciones que requieren intervalos de tiempo específicos para cada operación. Aparatos programables son usados para operaciones como la soldadura, lubricación, tratamiento por calor y expulsores de partes para moldes de plástico inyectado.

Las funciones básicas para el control de tiempo son el retardador de tiempo de encendido Timer ON Delay y el retardador de tiempo de apagado Timer OFF Delay. Para programar un punto de salida que enciende una luz 5 segundos después de que el botón es oprimido se escoge un retardador de tiempo de encendido, esto significa que después de 5 segundos el reloj controlador activa la bobina asignada y permanece activa hasta que el interruptor es apagado.

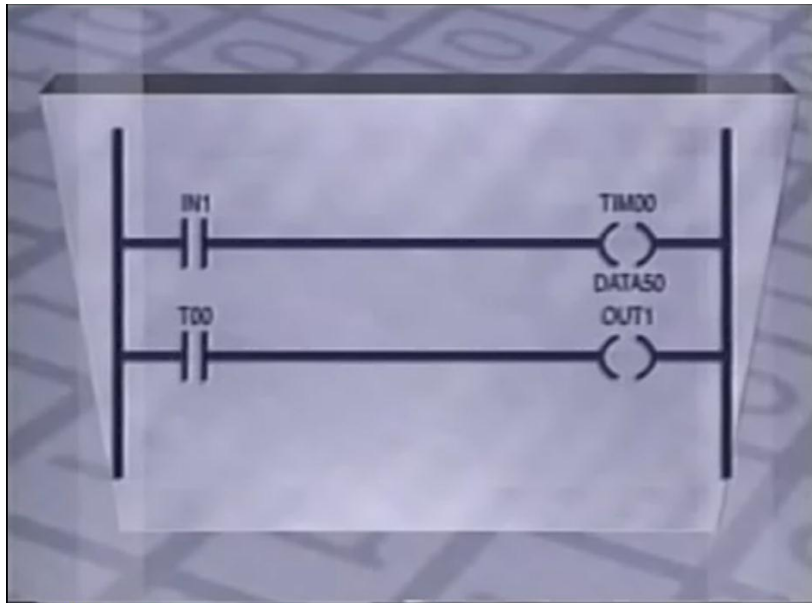


Fig 2.36 Diagrama de 2 peldaños con bobina de retardo.

Aquí vemos el diagrama discreto para el proceso, el código mnemónico para un retardador de tiempo de encendido es LD IN 1 ENTER TIM 00 ENTER TIM 00 DATA 50 ENTER, los PLC comúnmente requieren valores de control de tiempo con una décima de segundo y no aceptan valores de un solo segundo, por lo que se debe multiplicar la cantidad de segundos por 10 antes de programar esta información LD T00 ENTER STO OUT1 ENTER.

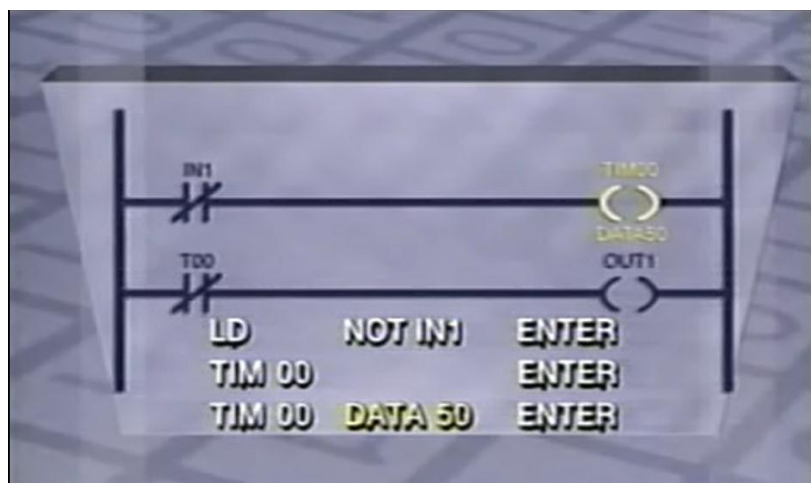


Fig 2.37 Código mnemónico para los 2 peldaños con bobina de retardo.

Si queremos que la luz se apague 5 segundos después de oprimir el botón usaríamos un retardador de tiempo de apagado Timer OFF Delay, el diagrama discreto para este elemento aparece así, la única diferencia entre este y el retardador de encendido es que los contactos están cerrados, los códigos mnemónicos son LD NOT IN 1 ENTER TIM 00 ENTER TIM 00 DATA 50 ENTER LD NOT T00 ENTER STO OUT1 ENTER.

Los contadores o registradores son usados en operaciones de manufactura cuando un número específico de señales debe estar presente para que un PLC active un sistema de salida, supongamos que queremos que una luz se encienda al oprimir un botón 4 veces, este sería el diagrama para el programa contador, el código mnemónico LD IN 1 ENTER CTR00 ENTER CTR DATA 4 ENTER LD CTR00 ENTER STO OUT1 ENTER. Cuando el contador llega al punto especificado activa el punto de salida en este caso una luz.

Los contadores pueden tener un número preprogramado o leer un número de un registro de datos, en vez de programar CTR00 DATA 4 podríamos usar el número de un registro de datos en la memoria, que tendría el número requerido para este contador, con este método el valor máximo puede ser cambiado sin reprogramar el PLC. Lo mismo se aplica a los relojes controladores.

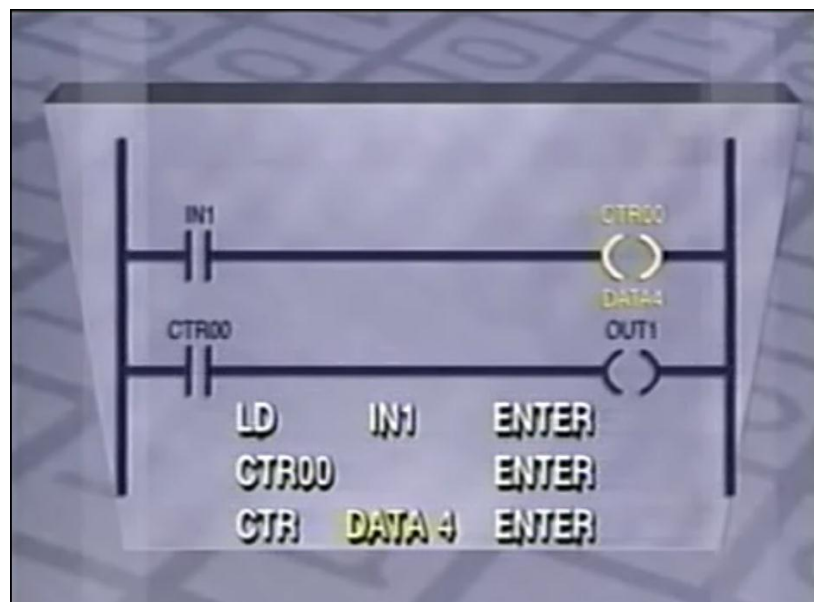


Fig 2.38 Código mnemónico para el diagrama de 2 peldaños.

Las bobinas o carretes internos son muy útiles para el operador del PLC ya que ofrecen más versatilidad al programar, las bobinas internas no están conectadas al circuito, lo que no son reales, estas bobinas internas no tienen resortes o bisagras como un relevo real, se mantienen en la memoria del computador como un bit de encendido o apagado y no tienen un esquema de delegación diferente a los esquemas de entradas y salidas reales.

Consideremos esta operación en la que 5 entradas diferentes determinan la función de 3 temporizadores diferentes, cuando los INPUT'S 1, 4, 5 están conectados y los INPUT'S 2 y 3 están desconectados todos los timers se activarán, el programar esta función puede hacerse mas fácilmente usando bobinas internas. Si se colocan todos los INPUT'S en un escalón controlando un carrete interno, se pueden usar los contactos de relé interno en los otros escalones, este es el diagrama simplificado usando una bobina o carrete interno. Si tenemos que añadir o remover otro carrete interno únicamente tendríamos que editar la primera línea y no las 3 líneas como en el diagrama original. El usar relés internos aumentará drásticamente las capacidades de programación.

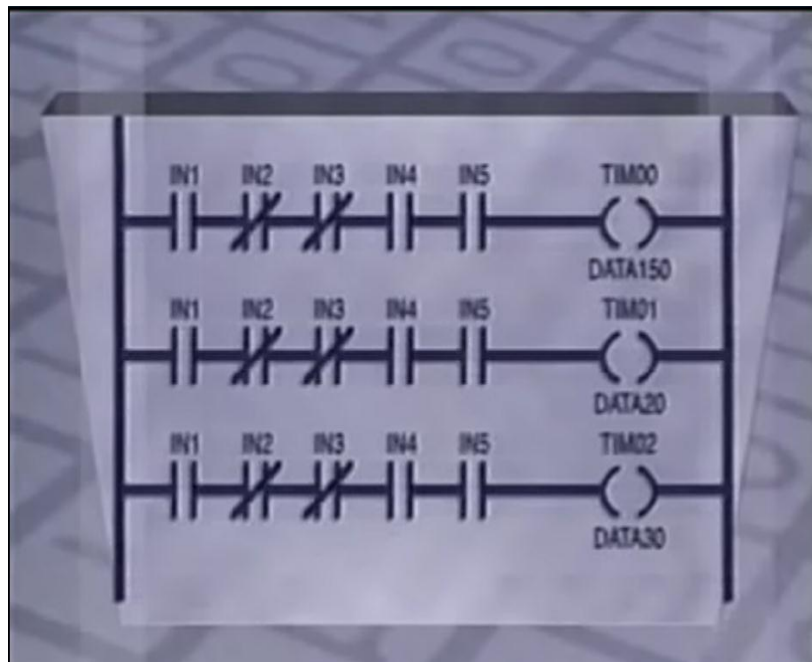


Fig 2.39 Diagrama de escalera de 3 peldaños y 15 interruptores.

3 TABLERO DEL PLC

Las funciones que están marcadas en este esquema están a cargo de un control lógico programable, ahora serán asumidas por un control lógico programable, un PLC esta constituido básicamente de 3 partes, La parte de la entrada de las señales la unidad central y la parte de la salida de las señales. La unidad central esta compuesta de un procesador y de una memoria del programa. Un ejemplo un detector de proximidad constata si al final de la cinta transportadora hay una válvula para el siguiente paso de trabajo, en caso positivo emite una señal de tensión a una pinza. Dicha señal pasa por la pinza y llega hasta el módulo de entrada, el módulo de entrada recibe la señal eléctrica y la transforma para que la unidad central que no es otra cosa que un pequeño pero potente ordenador la pueda elaborar.

3.1 PLC

Las funciones que asumen los cables conectados a una máquina también pueden estar a cargo de un controlador lógico programable, un PLC el cual está a cargo de los enlaces y del procesamiento de las señales.

Desde la unidad central con el programa correspondiente se transforma dicha señal en una señal de salida, que es transportada al módulo de salida, ahí es reportada y adaptada y por medio de una pinza es dirigida hacia la electro válvula. La electro válvula conecta y procura que el cilindro avance para que la pinza pueda coger la válvula con la pinza de la cinta transportadora. El programa para SPS se elabora normalmente con un programa y la ayuda de un software instalado en el ordenador y se memoriza en el control de mando.

SPS es un elemento de control ideal para los equipos electro neumáticos, electro hidráulicos debido a su fiabilidad y poco espacio, a esto hay que añadir además el mando sencillo y la extraordinaria versatilidad.

3.2 DELEGAR PUNTOS DE ENTRADA, SALIDA Y RELEVOS INTERNOS

La delegación de direcciones, la localidad es un término común al trabajar con PLC's es el lugar en la memoria donde el PLC almacena información sobre la condición de los INPUT'S ó bobinas internas, temporizadores, contadores y OUPUTS, si un INPUT es activado se coloca un 1 en la localidad de la memoria que presenta la condición de ese INPUT, si el INPUT está desconectado se coloca un cero en dicha localidad, recordemos que la mayoría de los fabricantes usan diferentes esquemas para las localidades, así que hay que consultar con el manual del operador para ver cual es el esquema particular del PLC.

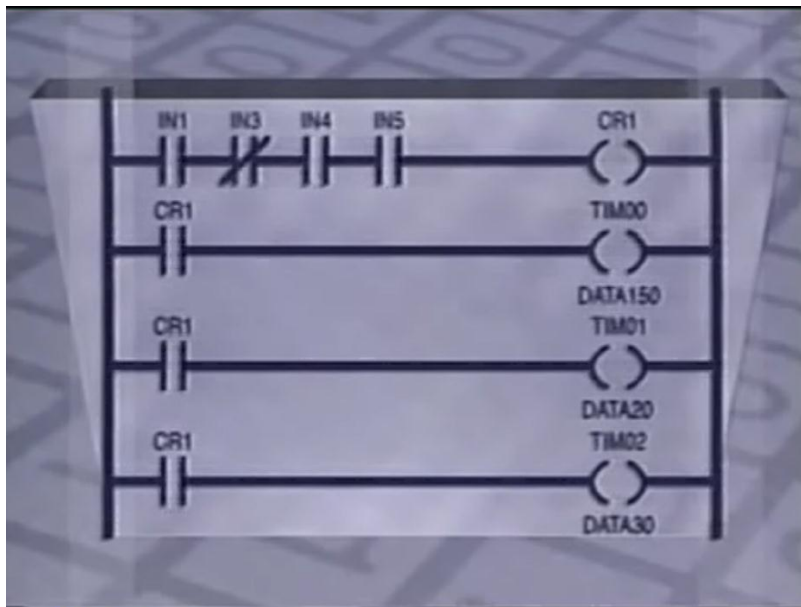


Fig 3.1 Diagrama de escalera de 5 peldaños.

Veamos el esquema para un sistema de PLC modular, la localidad para el primer INPUT en el primer módulo es 0000 el último INPUT tendrá la localidad 0015. el octavo INPUT en el quinto módulo sería 0407 contando desde cero para el módulo y el INPUT. Este Tipo de sistema es conocido como localización de palabra y bit. Este término se refiere a la localidad del módulo de entrada o salida, los primeros 2 números de una localidad son un apalabra, y los 2 números siguientes son el bit.

Supongamos que tenemos un PLC con 8 INPUT's 7, 6, 5, 4, 3, 2, 1, 0, la palabra es el número asociado con los INPUT'S 10010110, supongamos que usamos los INPUT's 1, 2, 4, 7 en nuestro programa y todos son activados, la palabra 00 tiene un equivalente binario de 10010110 y recordemos que el dígito menos significativo es el INPUT 0. La localidad del INPUT para el primer bit es 0001, la segunda localidad es 0002, la tercera 0004, y la última es 0007 o palabra bit 0000 0007.

3.3 DOCUMENTACIÓN DE PROGRAMAS

Es importante documentar todo el programa que generemos, algunos programas para LAP TOP's y PLC's avanzados imprimen la documentación si lo deseamos, si el programa desarrollado es muy largo, una copia del diagrama existente es muy útil, ya que nos ayuda a ver todo el programa a la vez, en vez de parcialmente en la pantalla. Los diagramas discretos o listas mnemónicas pueden ser útiles para diagnosticar el programa y detectar errores, sin el diagrama discreto o lista mnemónica nosotros no tenemos idea de que es lo que el programa tiene que hacer. Las listas de los temporizadores y contadores es importante, sobre todo en programas extensos, si queremos incrementar el tiempo de un timer en particular revisamos la lista de temporizadores para ver en que RAM fue ubicado el timer, dirigimos al RAM adecuado y cambiar la calibración del timer.

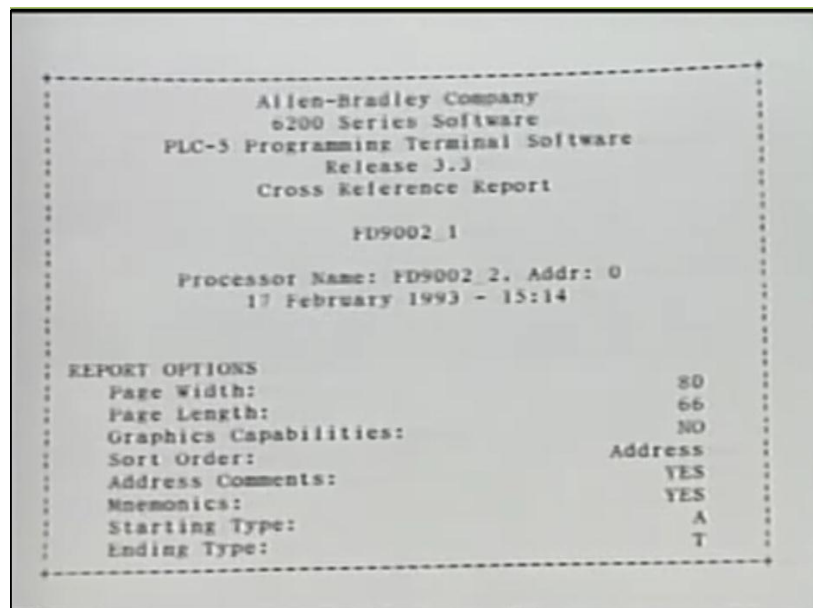


Fig 3.2 hoja de datos de un programa en PLC.

Las listas con las definiciones de los INPUTS y OUTPUTS ayuda al programador a llevar un recuento de los diferentes equipos conectados a las entradas y salidas, para determinar el INPUT asociado con un switch particular la lista de INPUT's puede ser usada para encontrar la señal al número de INPUT asociado con él. El mismo proceso se aplica para la lista de OUTPUT's. Una referencia cruzada indica los escalones del diagrama discreto o de escalera el cual contiene salida, entrada, timer, contador y bobina interna. La referencia cruzada es útil cuando hay una entrada o bobina interna colocada en RAM's múltiples.

Algunos contactos de bobina múltiple pueden ser usados docenas o cientos de veces en el programa, así que es útil saber como el cambiar una bobina puede afectar la demás lógica en el programa. El usar todos los tipos de documentación disponibles y el mantener información precisa, nos ahorrará tiempo para diagnosticar y modificar los programas del PLC.

En esta parte vimos como programar funciones de lógica en el PLC, como programar controladores de tiempo, registradores y relevos internos, como hacer un diagrama de delegación y como documentar el programa. Una combinación de estas 4 técnicas, la práctica y la creatividad hacen a un programador altamente eficiente.

3.4 ENTRADAS Y SALIDAS DISCRETAS

Los módulos de entrada y de salida proveen de una interface al PLC que le permiten comunicarse a este con los sistemas que esté conectado, es importante entender los diferentes tipos de módulos de salida y de entrada porque una instalación incorrecta puede estropear los módulos o los equipos conectados a ellos y aún herir al trabajador. Los 4 temas son entradas y salidas discretas, entradas y salidas analógicas, módulos de alambrado para entrada y salida y como conectar un sistema de multiplexing.

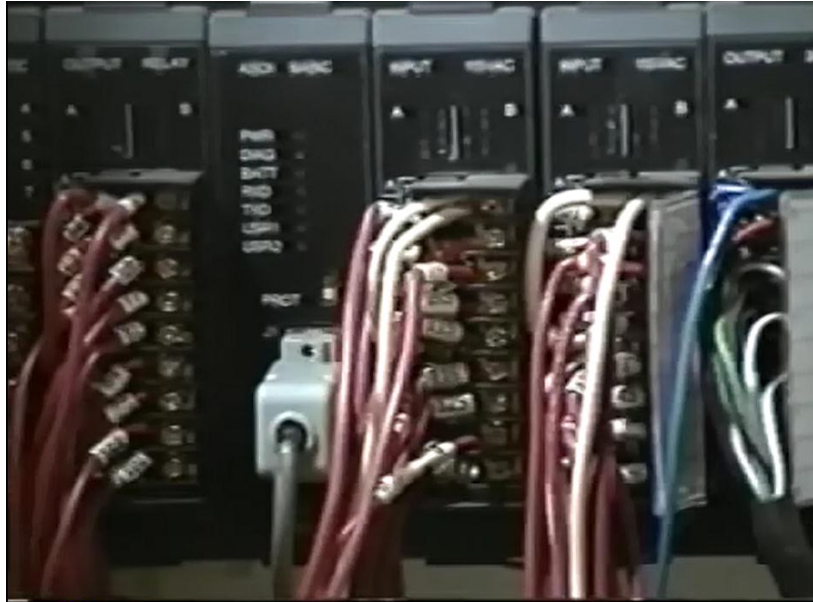


Fig 3.3 Módulos de terminales de tarjetas del PLC.

Las funciones que asumen los cables conectados a una máquina también pueden estar a cargo de un controlador lógico programable, un PLC el cual está a cargo de los enlaces y del procesamiento de las señales.



Fig 3.4 Módulos de entrada de DC de baja tensión.

Los módulos de entrada DC de bajo voltaje son utilizados en aplicaciones comunes a la industria de hoy, los interruptores de botón, los interruptores de límite y los switches de

temperatura son algunos ejemplos, los módulos de entrada DC de bajo voltaje están disponibles en aplicaciones de 12 a 24 voltios CD.

Los módulos de entrada tienen diferentes configuraciones, un módulo típico contiene 4, 6, 8, 12, 16, 32 terminales sin incluir el terminal común y a tierra. Estas entradas o INPUT's reciben señales digitales o análogas de varias fuentes de voltaje tales como 120 voltios 24 voltios DC, o valores análogos tales como 0 a 5 voltios DC, algunos PLC tienen una fuente de energía de 24 voltios DC que podemos usar para energizar las entradas. Se utilizan 24 voltios porque no hay peligro al trabajar con ellos y no electrocutarse. También es ventajoso utilizar entradas de bajo voltaje porque requieren menos corriente y menos alambrado.



Fig 3.5 Módulos de entrada de tensión alta 120 V.

Los módulos de entrada únicamente pueden leer los voltajes dentro del alcance para el cual fueron diseñados. Un INPUT diseñado para voltajes entre 5 y 30 voltios DC no podrá reconocer voltajes fuera de este intervalo. Una entrada de bajo voltaje puede dañarse si un alto voltaje es conectado a su terminal, es importante revisar siempre las calibraciones de voltaje antes de conectar los sistemas de entrada al módulo.

Los módulos de entrada de alto voltaje DC son usados cuando el sistema que esta siendo controlado requiere voltajes altos, tales como 110 o 240 voltios DC, el alto voltaje es usado a veces cuando sistemas alambrados son actualmente reemplazados por un PLC, ya que no pueden funcionar con menos voltaje y es mas económico dejar la fuente de energía de 120 voltios en vez de reemplazar todos los sistemas e instalar un transformador.

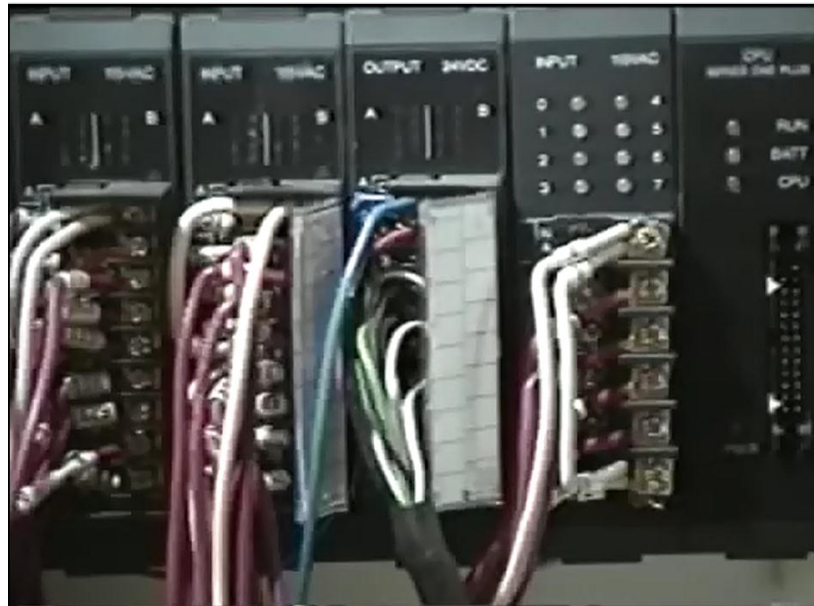


Fig 3.6 Módulos de entrada para 240 VCD.

Los voltajes AC también son usados por los módulos de entrada del PLC y se usan cuando son la única fuente de energía disponible. Algunas aplicaciones usan alto voltaje AC debido a los equipos controlados o a la edad del sistema.

Debido a que las fuentes de energía de AC son usualmente de 120, 240 ó 460 voltios. Un transformador que provee 24 voltios AC ó una fuente de energía de 24 voltios debe ser instalada, esto requiere equipos adicionales y el alambrado de alto voltaje conectado al transformador debe ser instalado de tal forma que no interfiera con los bajos voltajes presentes en el alambrado de bajo voltaje del sistema de entrada.

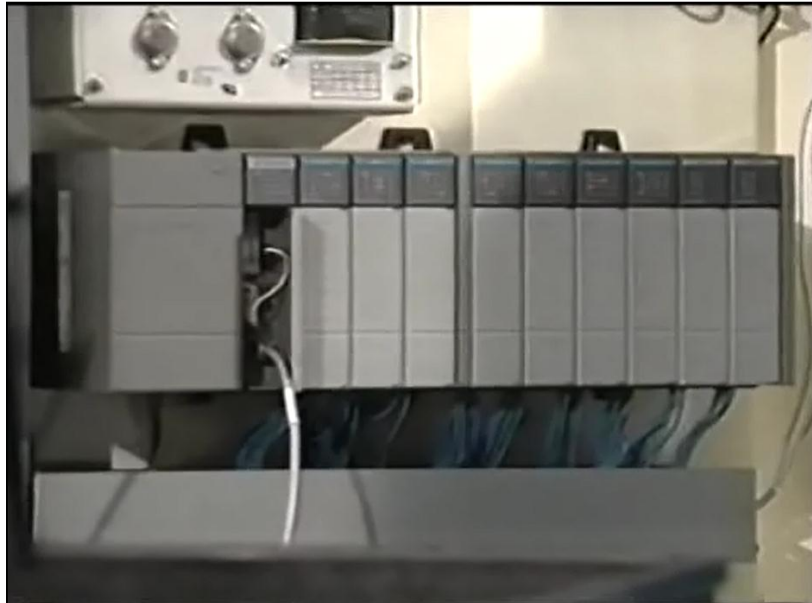


Fig 3.7 Módulos de 24 VCA o bajo voltaje.

Instalar cables para alto voltaje AC requiere mas atención que instalar alambrado o insulación. Los módulos de entrada están generalmente diseñados para responder igualmente bien a voltajes AC y DC. Un módulo de 24 voltios DC generalmente también puede aceptar una señal de 24 voltios AC. Y lo mismo se aplica a un módulo AC de 120 voltios que acepta una entrada DC de 120 voltios.

Los módulos de entrada discreta leen una señal que está prendida o apagada, pero que se hace si se requiere de un equipo que pueda leer una presión o una temperatura variables, para este tipo de señales usa un sistema de entrada análogo. Los sistemas análogos no producen una señal de ON u OFF, sino que miden un voltaje o una corriente variable entre 2 límites. Los sistemas análogos típicos miden distancia, presión, temperatura, electricidad, intensidad de luz, niveles de líquido.

3.5 ENTRADAS Y SALIDAS ANÁLOGAS

Supongamos que el nivel de agua en un tanque debe ser monitorizado, digamos que el tanque tiene 256 pulgadas de profundidad y una acción específica de salida debe ocurrir cuando el nivel sube o baja una pulgada.

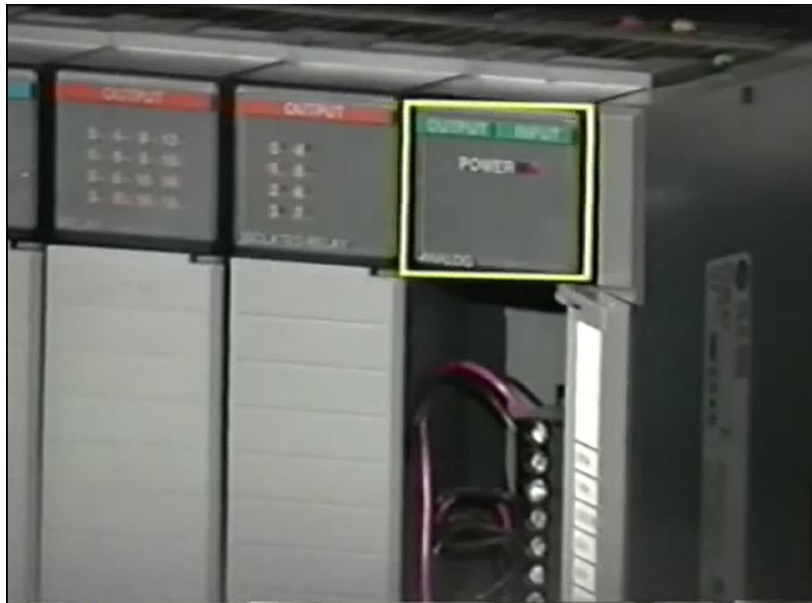


Fig 3.8 Módulo con entrada analógica.

Para cumplir esta función con sistemas de entrada discreta, switches flotantes tendrían que ser instalados en 256 niveles diferentes dentro del tanque para proveer una lectura precisa. Esto requeriría de 32 módulos de entrada discreta con 8 entradas cada uno. Con un sistema análogo únicamente se requieren 1 transductor análogo y un módulo de entrada.

Los módulos de entrada análoga vienen con un alcance entre 0 y 5 voltios DC, 0 a 10 voltios DC, 2.5 voltios DC negativo o positivo y 5 voltios DC negativo o positivo. Y corrientes entre 2 – 10 miliamperios, 4 - 20 miliamperios y 10 – 50 miliamperios.

El módulo usado dependen de la aplicación y del transductor o fuente análoga que estén presentes, los módulos de entrada análoga convierten el voltaje del equipo a un número binario que es leído por el PLC, el alcance de la profundidad del transductor en el tanque es de 1 a 256 pulgadas máximo, se requiere un transductor con alcance de profundidad de por lo menos 256 pulgadas. Si el máximo de salida del transductor equivale a 10 voltios cuando hay 256 pulgadas de profundidad, se aplica una señal de entrada de 10 voltios cuando el tanque está lleno, supongamos que la profundidad de l tanque es de 10 pies 8 pulgadas ó 128 pulgadas, en este punto el transductor lee la mitad de su valor máximo produciendo la

mitad de su voltaje máximo o sea la mitad de 10 voltios, para esta profundidad el transductor representa 5 voltios a la entrada análoga del PLC. Si el nivel es de 64 pulgadas la señal es de un cuarto del voltaje máximo o sea 2.5 voltios.

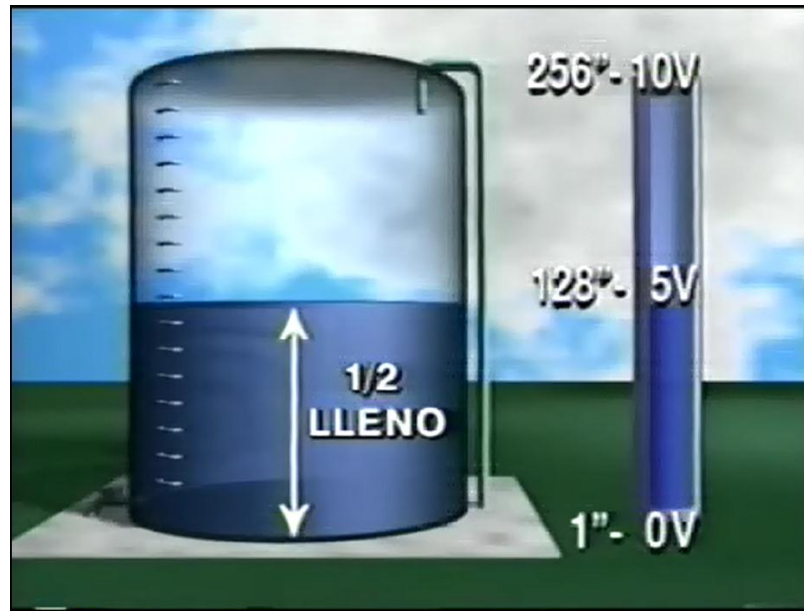


Fig 3.9 Diagrama del tanque y el equivalente altura y voltaje.

El módulo de entrada convierte el voltaje a un formato binario con un número de bits determinado por la capacidad del módulo, cero voltios al módulo producen un valor convertido a binario de 0000 0000 ó cero decimal. 10 voltios producen un valor de 1111 1111 ó 255 decimal. El número binario resultante es ubicado en un lugar de la memoria determinado por las especificaciones para el módulo y puede ser leído por el PLC como lo requiera el programa. El registro de entrada finalmente es actualizado en cada barrida del PLC.

Supongamos que una alarma tiene que sonar si el nivel del agua es menor de 64 pulgadas, un diagrama similar a este se usaría, la primera línea contiene un contacto normalmente abierto y el comando de desplazamiento MOV, al entrar el comando MOV se debe usar un registro de memoria temporal, en este programa el registro del PLC es el HR100 y el módulo análogo colocará el valor del voltaje convertido al registro de entrada IR01, la información debe ser movida del registro de entrada IR01 al registro de memoria temporal

HR100, este escalón mueve la información cuando el contacto del controlador normalmente abierto está cerrado.

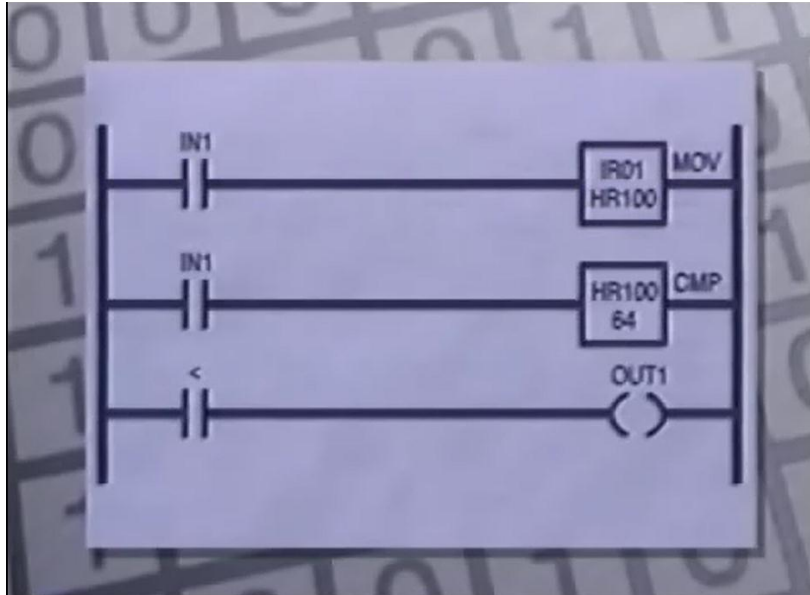


Fig 3.10 Diagrama de escalera para el tanque.

El valor está ahora en la memoria y puede ser usado en un escalón de comparación con un número. La segunda línea muestra la función de comparación, aquí el PLC compara el valor en el registro 100 con las constante 64_{10} , el resultado de la función de comparación controla ciertos bits en la memoria del PLC llamados banderas, cuando se efectúa la función de comparación se establece una de 3 banderas representando mayor que $>$, menor que $<$ ó igual a $=$ según el resultado de la comparación. Debido que la bandera es un bit en la memoria puede ser usado como un contacto que actúa según el resultado de la comparación, en este caso la alarma debe sonar si el nivel del agua es menor de 64 pulgadas y se usa la bandera menor que $<$.

Si el nivel de agua en el tanque es 60 pulgadas esta bandera se activaría como resultado de la función de comparación, haciendo que el contacto normalmente abierto asociado con la bandera es cierto, lo cual activa la alarma. Las entradas análogas también pueden ser usadas para especificar un alcance dentro del cual algo debe ocurrir, supongamos que una luz debe encenderse cuando la profundidad del tanque está entre 140 y 180 pulgadas, esto es el

comando MOV para transferir el valor de entrada al registro 100 se puede usar la función de comparación CMP para comparar el valor a 140 pulgadas, un contacto normalmente abierto de la bandera mayor que $>$, podría ser usado entonces para permitir una función busque un valor de comparación mayor a 180, de nuevo un contacto normalmente abierto de la bandera menor que $<$ podría ser usado para hacer que la luz se encienda.

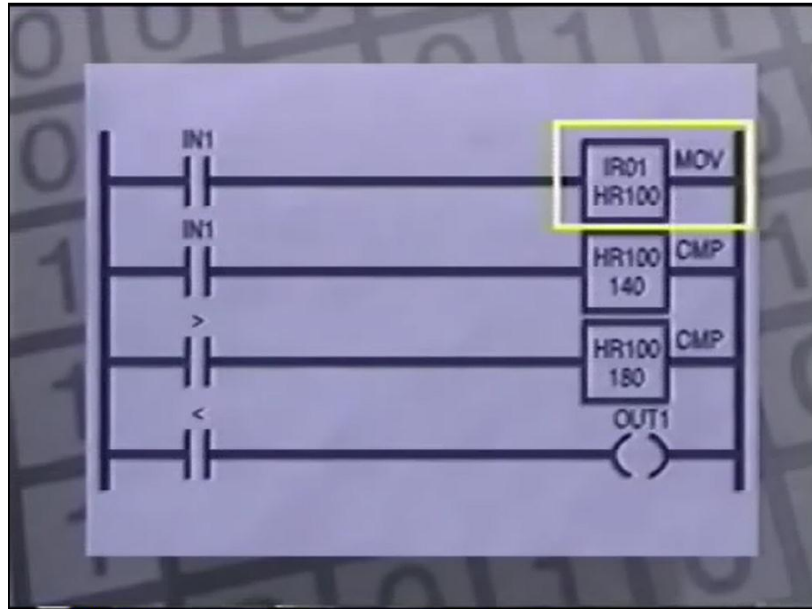


Fig 3.11 Diagrama de escalera con condicionales para el tanque.

Observamos que las banderas mayor que $>$ y menor que $<$ son afectadas cada vez que la función de comparación se efectúa, así que las banderas deben ser usadas antes que otra función de comparación, ya que estas siguiente función ubicará las banderas en otro valor.

3.6 MÓDULOS DE ALAMBRADO PARA ENTRADAS Y SALIDAS

Los módulos de salida al igual que los módulos de entrada sirven de interface entre el PLC y el mundo exterior, el módulo de salida convierte datos de la unidad central a un formato que controla el equipo conectado. Las salidas pueden ser sistemas transistorizados ó contactos secos. Ejemplos de salidas tipo estado sólido podrían ser un transistor, un TRIAC, un SCR. Sistemas de bajo voltaje como luces pueden ser activados con una salida transistorizada.

Sistemas de alto voltaje AC como motores de arranque pueden ser controlados usando salidas TRIAC, un contacto seco es un relevo que es cambiado mecánicamente, los contactos secos son de 3 configuraciones o formas: forma A, forma B, forma C. La configuración de forma A es un contacto normalmente abierto, una configuración de forma B es un contacto normalmente cerrado, y la configuración de forma C es un contacto que contiene uno de cada uno.

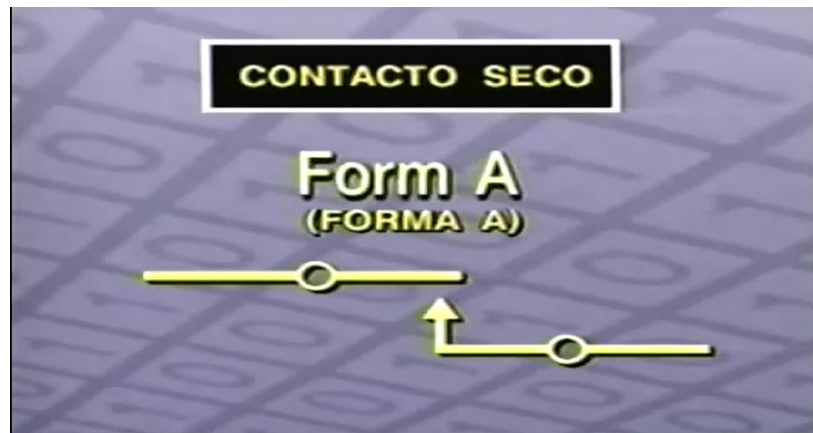


Fig 3.12 Contacto normalmente abierto y símbolo.

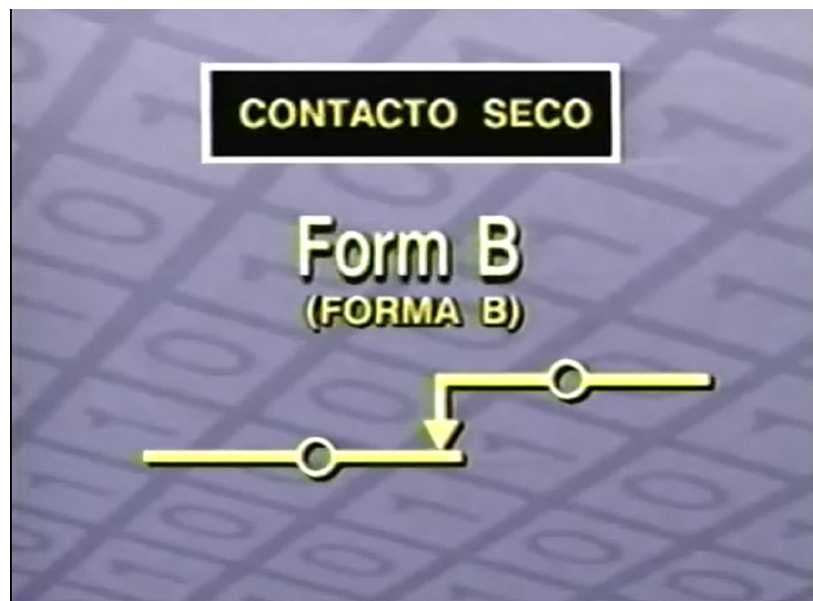


Fig 3.13 Contacto normalmente cerrado y símbolo.

Los contactos secos son mas adecuados para salidas que manejan alto voltaje e impulsos de corriente demasiado intensos para equipos transistorizados, una vez seleccionado el tipo de módulos de entrada y salida, se debe desarrollar el alambrado mas eficiente posible. Las condiciones de alambrado son muy importantes para la operación del PLC y es un aspecto que no debe ser menospreciado, el usar malas técnicas para el alambrado puede comprometer la seguridad del sistema resultando en lesiones personales o daños al PLC.

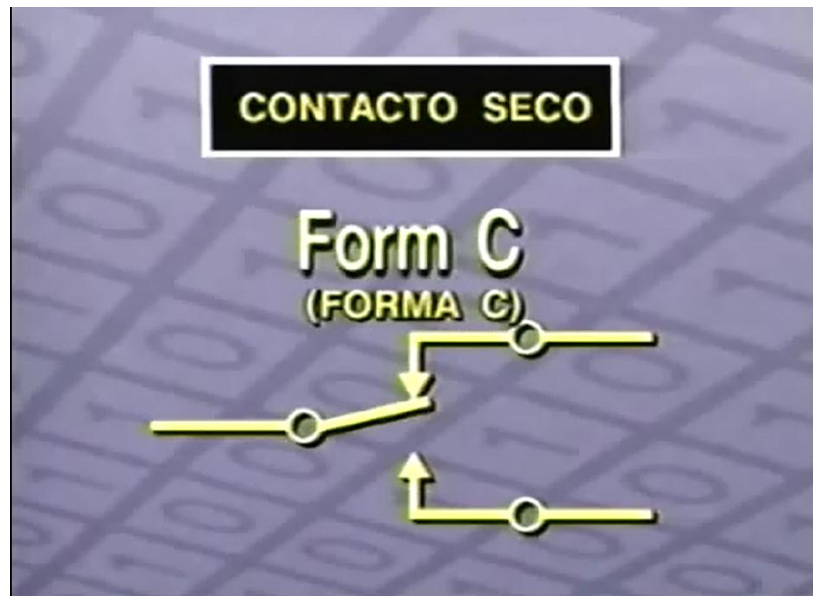


Fig 3.14 contacto dual normalmente cerrado / abierto.

Un PLC generalmente maneja diferentes voltajes de operación, como las señales de control de salida de bajo voltaje y de alto voltaje, al alambiar los módulos de entrada y salida separamos el alambrado de bajo voltaje del alambrado de alto voltaje, el código nacional de electricidad describe los procedimientos con respecto a la ruta de alambrados de alto y bajo voltaje en un sistema.

Al alambiar sistemas de entrada o salida localizadas a cierta distancia del PLC puede ser necesario utilizar una unidad remota de entradas y salidas, esta unidad consiste en un sistema de comunicación, módulos de entrada y salida y una fuente de energía.

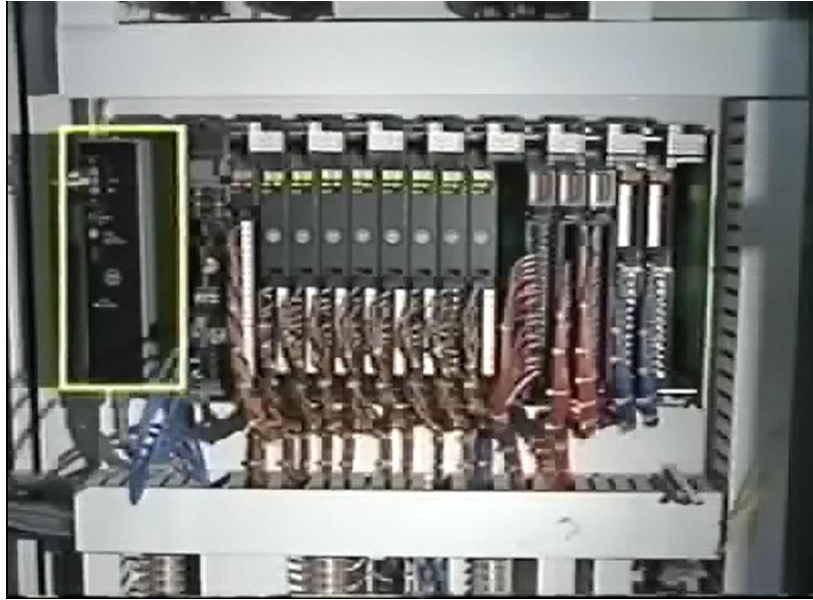


Fig 3.15 Unidad de comunicación remota del PLC.

3.7 MULTIPLEXING

En un sistema que tiene 8 entradas y 8 salidas a 150 metros del PLC tienen que ser conectados, esta conexión requeriría 18 o más cables para recorrer la distancia de 500 pies, esto podría resultar en interferencia, resistencia o disminución en el voltaje si las salidas requieren un nivel alto de corriente. La unidad remota elimina estos problemas según el sistema que se escoja únicamente hay que conectar de 2 a 5 cables a la unidad remota. Los sistemas de comunicación en la unidad remota y en el PLC transfieren información relacionada a los módulos de entrada y salida entre el PLC y la unidad remota. Algunas unidades remotas utilizan sistemas de fibra óptica para comunicarse con el PLC eliminando el riesgo de saltos en el voltaje o interferencia en la línea de comunicación.

Los sistemas multiplexing pueden ser usados si no hay suficientes entradas y salidas, al observar este diagrama de alambrado notamos que cada lado de cada uno de los 4 interruptores está conectado a una entrada y que las salidas 1, 2, 3 y 4 están conectadas al otro terminal de cada interruptor. Este tipo de diagrama de alambrado puede ser usado para reducir el alambrado cuando se requiere que un gran número de equipos sean conectados al PLC o si los equipos están a gran distancia y el uso de una unidad remota no es posible.

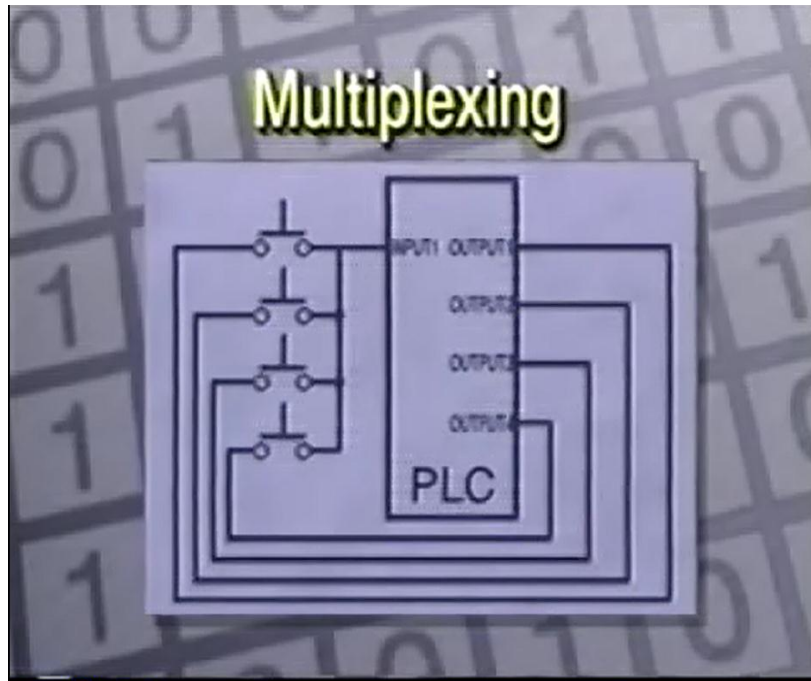


Fig 3.16 Diagrama de multiplexado de 4 cables.

En estos casos hemos visto que las entradas de voltaje discreto y análogo y la diferencia entre los dos, también los comandos para mover y comparar para manipular entradas análogas, como alambrear correctamente entradas y salidas tomando en consideración el ruido, la interferencia y el voltaje y como diseñar un sistema de alambrado con sistema multiplexing para aprovechar al máximo las entradas y salidas. Estos conceptos ayudan a instalar un sistema PLC nuevo y a conectar correctamente módulos de entrada y salida a sistemas establecidos.

3.8 ARRANQUE Y PARO

Primero verificamos que el PLC este prendido, esta la interface, esta conectado hacia el cable, el cable hacia la computadora, para un arranque y paro tenemos el interruptor termomagnético, el contactos y el motor. Verificamos que estemos conectados a la alimentación de 110 voltios.

Una vez que ya vimos los elementos que componen la práctica de un arranque y paro del motor con un PLC Allen Bradley Micrologic1100, la siguiente etapa consiste en comunicar lo que es el PLC con la computadora. Para esto el software que vamos a utilizar es Rslinx, lo abrimos, lo maximizamos y vamos a la parte de configuración de drivers. Buscamos entre todos los protocolos que nos podemos comunicar con los drivers que puedo agregar, en este caso el que vamos a utilizar es RS 232 entre otros como Ethernet, Device Net, etc, el driver que utilizamos es típicamente conocido como comunicación serial.

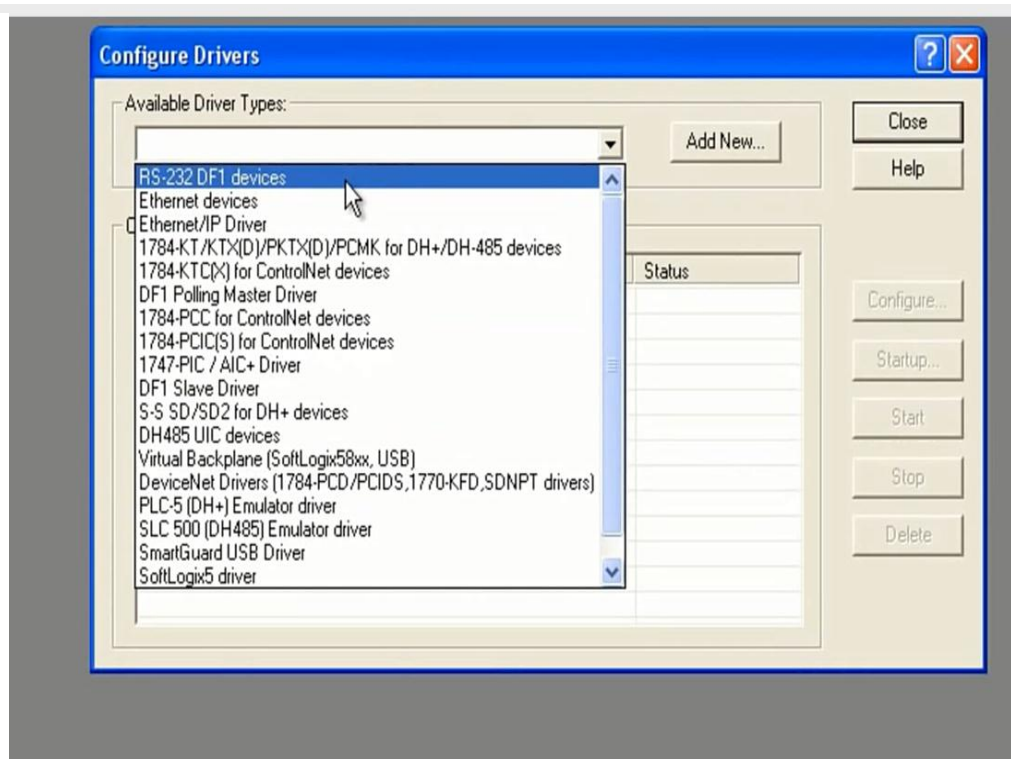


Fig 3.17 Ventana para configurar Drivers al PLC.

Empezamos en auto configurar, ya que en esta parte de aquí esta buscando la configuración del PLC si nos aparece o nos marca una falla pueden ser varias las causas como que el cable este desconectado, el cable esté dañado o que el puerto como que hemos elegido no es el correcto. Para verificar la primera parte verificamos que nuestro cable este conectado y en la siguiente parte verificaríamos que el puerto com que hemos seleccionado es el correcto. Oprimimos la tecla de Windows pausa del hardware, el administrador de

dispositivos, vamos a puertos com, si la computadora es reciente ya no tiene en la placa un puerto serial RS 232 solamente cuento con puerto USB utilizo convertidores USB que está detectado como com 2.

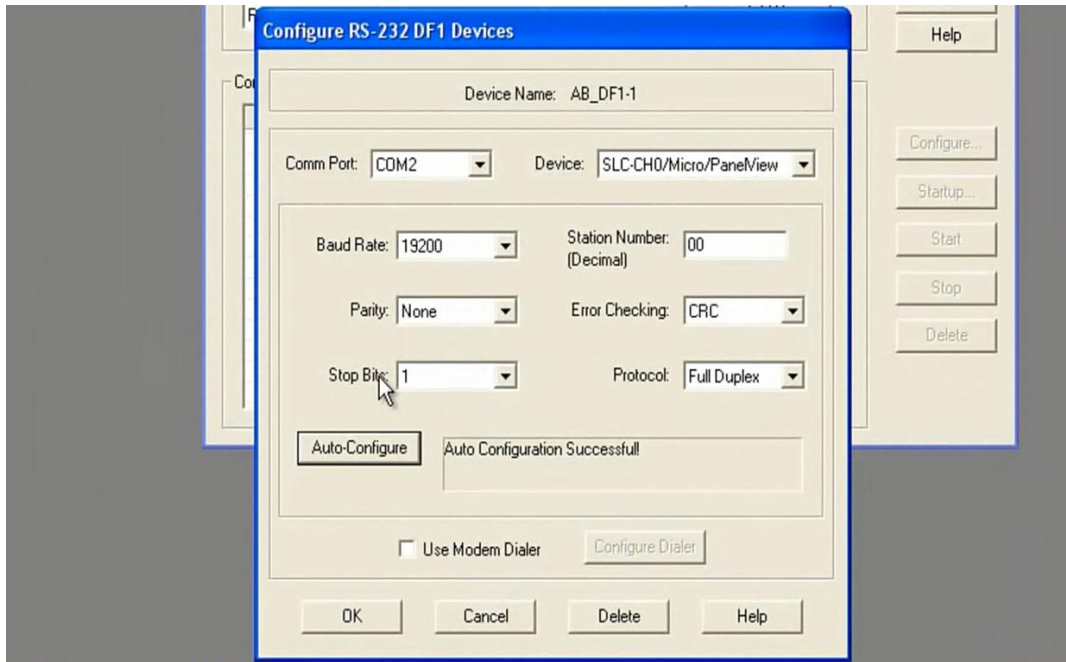


Fig 3.18 Ventana con puerto seleccionado para el PLC es RS 232.

Cambiamos a com 2 y damos auto configurar. Como detalle en el puerto USB serie los que son muy largos que tienen un metro aproximadamente de largo se detectan fallas seguido, es mejor utilizar un cable de 20 cm de largo que es mas corto suelen trabajar mejor. Vemos la configuración en com, la velocidad, el bit de parada 1, full duplex.

El PLC ya está en runing, damos close y posteriormente vamos a la parte de RSLinx, seleccionamos el PLC, vemos que aunque quitamos la autobúsqueda y lo removamos damos autobúsqueda y vuelve a aparecer, si el cable se desconectara o fallara aparecería un tache en el icono de conexión que encontramos con autobúsqueda.

Lo primero que verificamos es que tengamos comunicación con el PLC ya que si esta etapa no funciona lo demás no podremos hacerlo, una vez que tenemos configurado y detectado el PLC en la red abrimos el programa.

El programa que abrimos es RSLogix Micro para Allen Bradley en versión micro para PLC's 1000 y 1100 ambos creados por la compañía Rockwell automation, en este caso utilizamos una versión 8.30. lo siguiente que hacemos es hacer un respaldo del programa que tiene el PLC y esto es muy útil para cualquier modificación que vayamos ha hacerlos al tener respaldado el programa original. Esto lo podemos hacer desde load ó system com cualquiera de las dos opciones verificamos que tenemos conectado el PLC además verificamos el protocolo, ahora aparece una ventana con una tecla load que la oprimimos para descargar el programa del PLC hacia la computadora, después de la descarga una nueva ventana aparece preguntándonos si queremos ponerlo en línea seleccionamos afirmativo.

Vemos que estamos en línea, en Run, además estamos activando las forzadas, y ahora vemos en pantalla el programa que tiene el PLC que consta de una entrada y de una salida. Ahora que ya respaldamos el programa además estamos en línea, hacemos un programa para el arranque y paro del motor, ya tenemos entrada normalmente cerrado I : 0/1 como el arranque, agregamos el botón de paro I : 0 / 0 arrastrando y colocando el icono de la ventana. El carrete de control interno lo ponemos como O: 0/0.

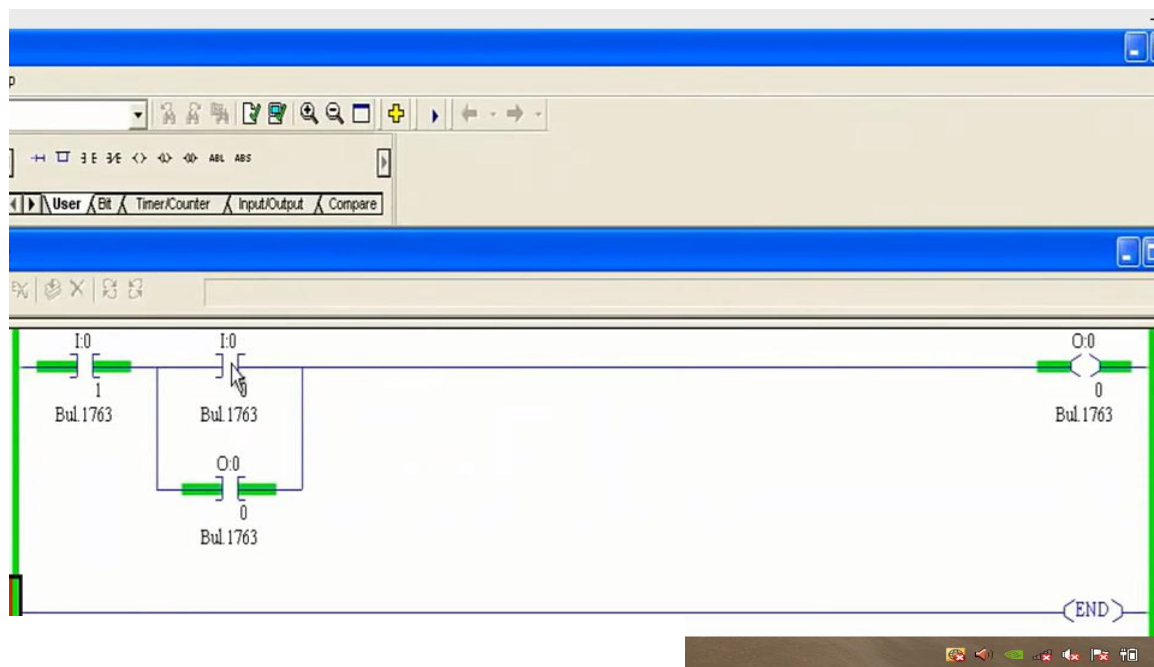


Fig 3.19 Diagrama en Rs Logix para paro y arranque.

colocado así si fuera un sensor. Ahora nos salimos de línea para cambiarlo por un interruptor normalmente abierto, volvemos a descargar el programa.

Lo seleccionamos dándole down load, ya estamos cargando el programa al PLC, con la modificación de cambiar el I: 0/1 normalmente cerrado a normalmente abierto, ya que en esta lógica al llegar al energía al contacto invierte el estado del mismo entonces se abre y el motor no arranca, por eso debe ir normalmente abierto para que al energizarse sea cerrado, podemos directamente arrancar el motor.

Presionamos arrancar y arranca el motor ya entrando el contactor en juego, pasan las 3 líneas trifásicas y el motor arranca a través de los relevadores electromecánicos.

3.9 DIAGRAMA DE ESCALERA

con diagrama de escalera, para programar un PLC necesitamos básicamente 3 elementos: una computadora, un PLC y un software que nos permita desarrollar el diagrama de escalera. Tal diagrama después será enviado al PLC para que este lo ejecute de manera constante e independiente. Antes de hablar de lo que es la programación del diagrama de escalera vamos a hablar de lo que es el ciclo de programa, que son los 4 pasos que ejecuta el PLC cada vez o cada iteración del programa.

El primer paso es el barrido de entradas que es el proceso en el que el PLC toma la información conectada a las señales de campo y la manda a la memoria. Segundo es el barrido del programa, una vez que la información de las señales de entrada está en la memoria entonces se puede ejecutar el programa renglón a renglón que hemos desarrollado y programado en el diagrama de escalera.

Como tercer paso es el proceso de enviar la información resultado de la ejecución del programa que está almacenado en la memoria del PLC a las tarjetas de salida para que estos activen sensores, válvulas, motores o cualquier otro elemento conectado. El paso 4 es el

mantenimiento interno, le va dar lugar a ciertas operaciones como locación de memoria, limpieza de variables, todo aquello que puede ser necesario por el PLC para mantener la integridad del sistema.

Ahora dibujamos el diagrama eléctrico de un sistema de arranque y paro que es el que vamos a estar utilizando, tenemos el botón de paro, el botón de arranque, el motor y la bobina en el clásico circuito de arranque y paro de motores. Vamos ahora a copiar este diagrama a convertir el diagrama de escalera y ahí lo tenemos, podemos darnos cuenta que el diagrama de escalera es muy similar al diagrama anterior de escalera, que tienen los mismos elementos y la misma estructura, ese es un diagrama de escalera.

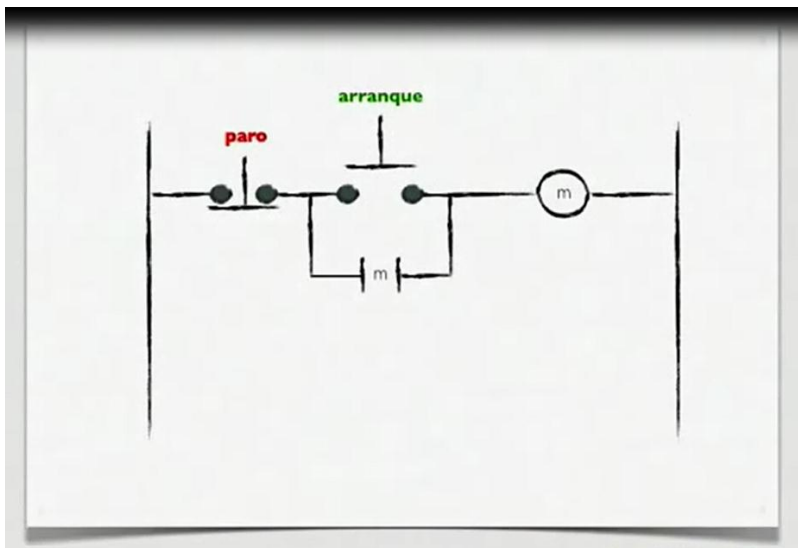


Fig 3.20 diagrama sencillo de paro y arranque de un motor.

Como podemos darnos cuenta el diagrama de escalera tiene varios elementos, como podemos observar que tiene renglones, todo diagrama de escalera está separado por líneas o renglones que son leídos durante la ejecución del programa de arriba hacia abajo y de izquierda a derecha de la misma manera que en las culturas occidentales leemos.

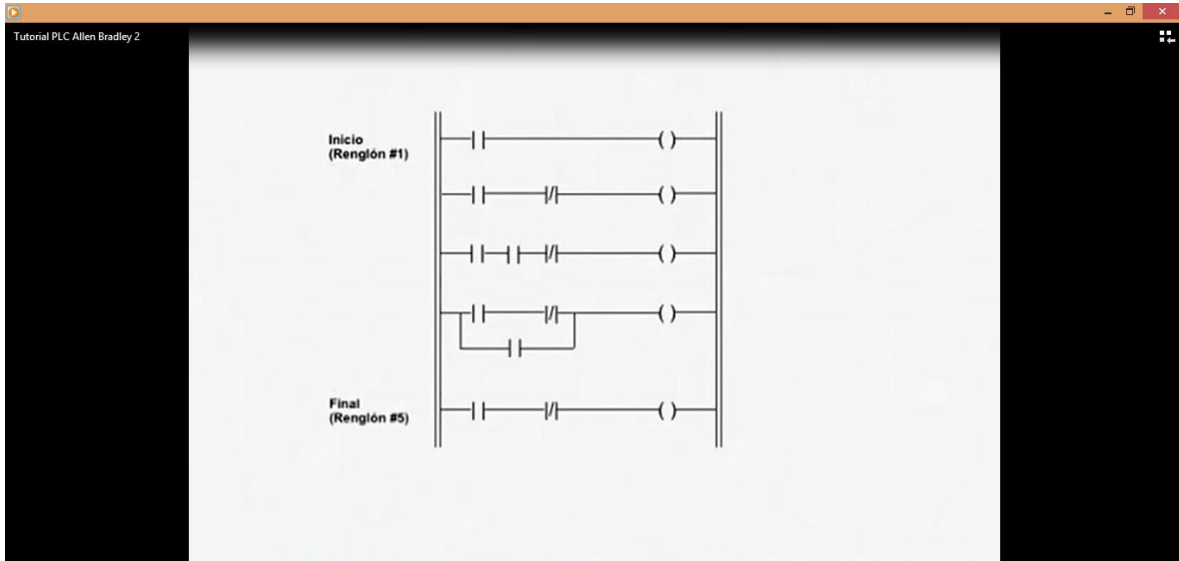


Fig 3.21 Ejemplo de un diagrama de escalera de 5 peldaños combinacional.

Podemos ver también en este diagrama que existen un conjunto de elementos del lado izquierdo que son instrucciones de lectura o las condiciones cuando estemos evaluando un programa, el PLC recorrerá de izquierda a derecha primero se encontrará con lo que son las condiciones, que en lenguaje podríamos decir si el contacto, si esto, si lo otro, son las condiciones de entrada que están del lado izquierdo.

Y podemos inferir que del lado derecho entonces estarán las instrucciones de escritura o de control, que son el resultado de una evaluación. Veamos ahora un ejemplo muy sencillo de lo que sería un diagrama de escalera, si pensamos en que el contacto está abierto que sería tener un estado falso, el flujo de información no podría cruzar por este contacto y entonces el resultado sería en la bobina falso. Sin embargo si cerramos el contacto que sería tener un contacto verdadero, el flujo de información podría transitar por ese contacto llegar a la bobina, ponerla en valor verdadero y el resultado sería verdadero.

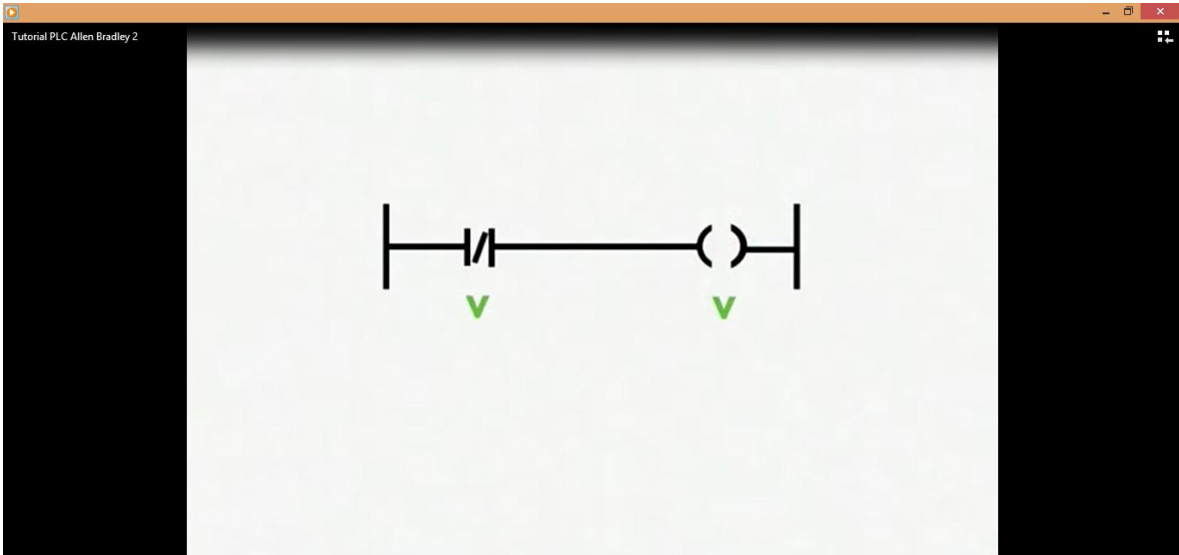


Fig 3.22 Operación NOT con contacto normalmente cerrado y salida V.

Que pasaría si ahora tenemos una estructura un poco mas compleja de contactos en serie, podríamos pensar en ellos como en una tubería donde tenemos dos llaves en serie y para que el fluido pueda pasar necesitamos tener las dos llaves abiertas, que de lo contrario no pasaría el flujo y esto es una estructura de tipo AND.

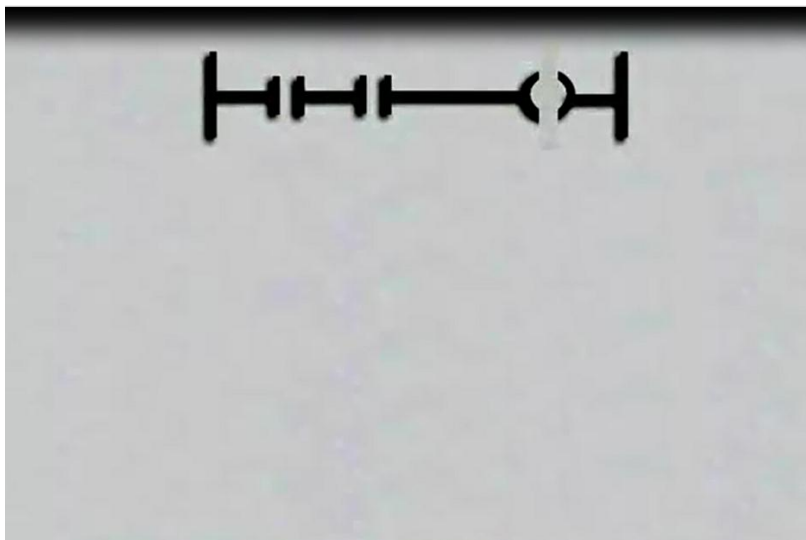


Fig 3.23 Operación AND con 2 contactos normalmente abiertos.

Si ahora tenemos las estructuras de información en paralelo y no en serie, pues sería lo mismo que si tuviéramos un sistema hidráulico donde tenemos dos llaves, ya sea que una o la otra llave este abierta tendremos flujo y esto es una estructura de tipo OR.

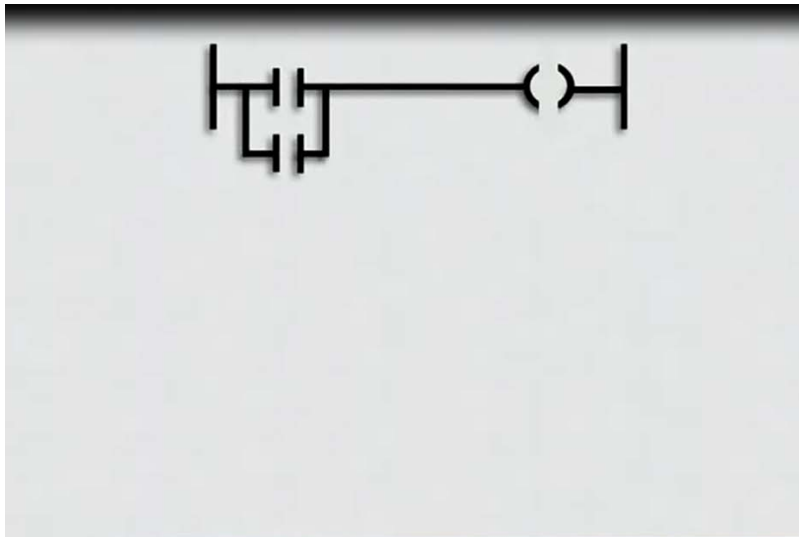


Fig 3.24 Operación OR con 2 contactos normalmente abiertos o en paralelo.

Este es un diagrama de escalera un poco mas complejo donde podemos darnos cuenta que tenemos un renglón con varios renglones anidados, contactos normalmente abiertos, normalmente cerrados, podemos identificar normalmente cerrados porque tienen una diagonal dentro de ellos.

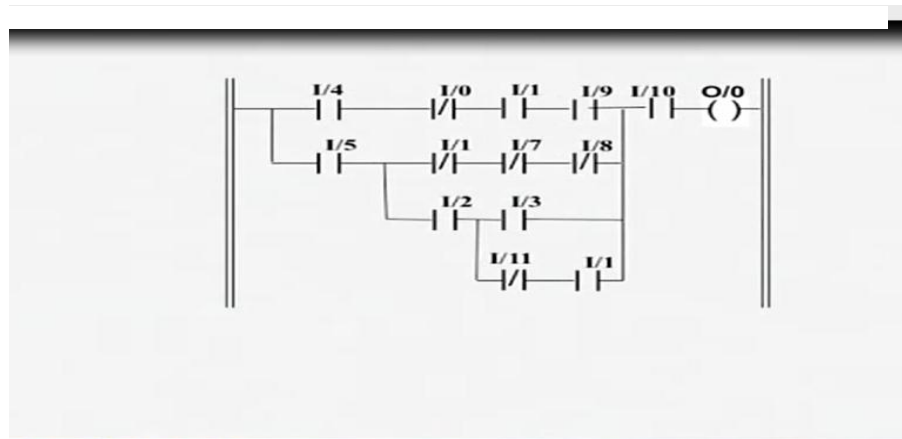


Fig 3.25 Diagrama de escalera complejo con 1 peldaño y 4 ramas con 13 interruptores.

Vamos a analizar ahora el diagrama eléctrico de cómo sería la evaluación que un PLC haría, en la primera iteración donde todo está falso el botón de arranque y paro están en falsos que sucede, empezaría el flujo la evaluación llegando al contacto de start como esta abierto no puede pasar, toma entonces el camino de la bobina de retorno como esta abierta no puede pasar, por consiguiente el motor estaría apagado.

En la siguiente iteración podemos observar que ahora ya tenemos el botón de start encendido o presionado, por lo tanto el contacto de start o encendido se hace verdadero, y en la segunda iteración podemos ver que el flujo de información puede transitar, llegar a la bobina, por consiguiente hacer que el motor se encienda.

Entonces que pasaría en la tercera iteración en que el botón de encendido ya no está presionado, sin embargo la bobina y el motor si están encendidos. Veamos entonces como el flujo de información llegaría al start no puede cruzar porque el botón ya no está presionado, sin embargo toma el camino de la bobina de retorno y como ya estaba energizado el motor puede transitar y llegar a la bobina, energizarla nuevamente y encender el motor y en la siguiente iteración se mantendría encendido.

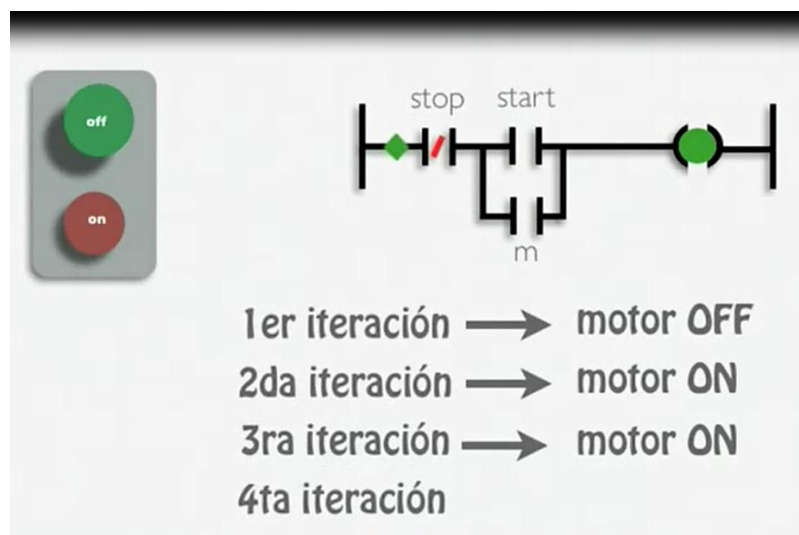


Fig 3.26 Exploración de un peldaño mostrando 4 iteraciones del PLC.

Y así podemos mantener la condición hasta que presionáramos el botón stop y el flujo de información para la cuarta iteración sería de la manera siguiente, llegaría al contacto de stop y encontraría que no puede pasar por consiguiente el motor se apaga.

Hemos puesto un ejemplo de cómo se programa el arranque y paro de motor es muy sencillo, sin embargo son los conceptos básicos para poder entender cualquier programación en diagramas de escalera.

4 PROGRAMACIÓN BÁSICA

4.1 TIMERS

Los elementos de tiempo como son conocidos, los dos elementos básicos de tiempo son el temporizador y el monoestable. A continuación veremos un ejemplo de programación de un automatismo temporizado. El esquema siguiente se corresponde con el mando de un motor con marcha temporizada (vea su funcionamiento aquí):

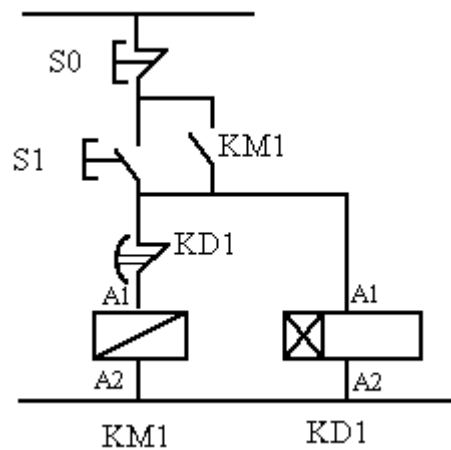


Fig 4.1 Diagrama para el motor con marcha temporizada.

Automatismo temporizado.

Un posible programa equivalente en LADDER podría ser el siguiente:

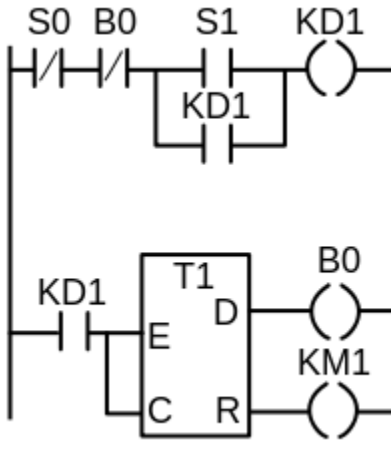


Fig 4.2 Uno de las soluciones en diagrama de escalera del motor temporizado.

Aplicación de un temporizador en LADDER.

Elementos de cómputo

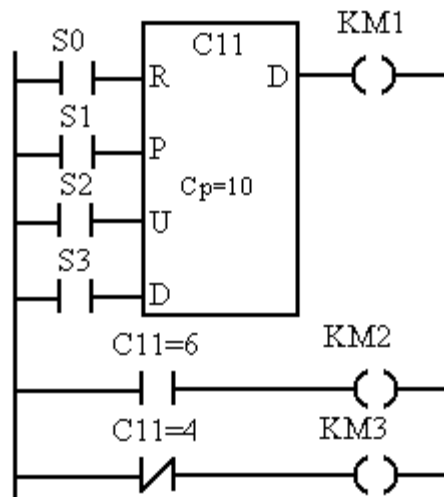


Fig 4.3 Diagrama de escalera de un contador.

Ejemplo de programa LADDER de cómputo

Para aclarar la programación con elementos de cómputo, se explicará el funcionamiento del esquema de la derecha:

Como se puede observar, el programa consta de un contador C11 que ha sido programado con el valor 10 ($Cp=10$). Con la entrada S0 ponemos a cero el contador y con la entrada S1 se preselecciona con el valor de Cp , esto es, 10. Cada impulso dado en S2 incrementa en una unidad el contador y cada impulso en S3 lo disminuya.

Las bobinas KMI y KM2 se activan cuando el contador posee el valor 10 y 6 respectivamente, en cambio, la bobina KM3 está continuamente activada excepto cuando el contador se encuentra con el valor 4.

Sistemas secuenciales

Aunque es posible programar sistemas secuenciales en LADDER, sólo se suele utilizar para el control de sistemas sencillos. En aquellos más complejos se utiliza la programación modular o el GRAFCET.

Los timers que son, como se usan y como se implementan en un diagrama de escalera, para entenderlo es necesario tener los conceptos básicos del diagrama de escalera como contactos, bobinas, renglones y conocer también el circuito de arranque y paro de motores. El agrega un cronómetro al motor para que este se encienda durante un tiempo y luego pararlo, tendríamos que poner algo en el diagrama de escalera que funcionara como un cronómetro, una función que nos permita temporizar o llevar el tiempo que ha transcurrido a partir de un momento.

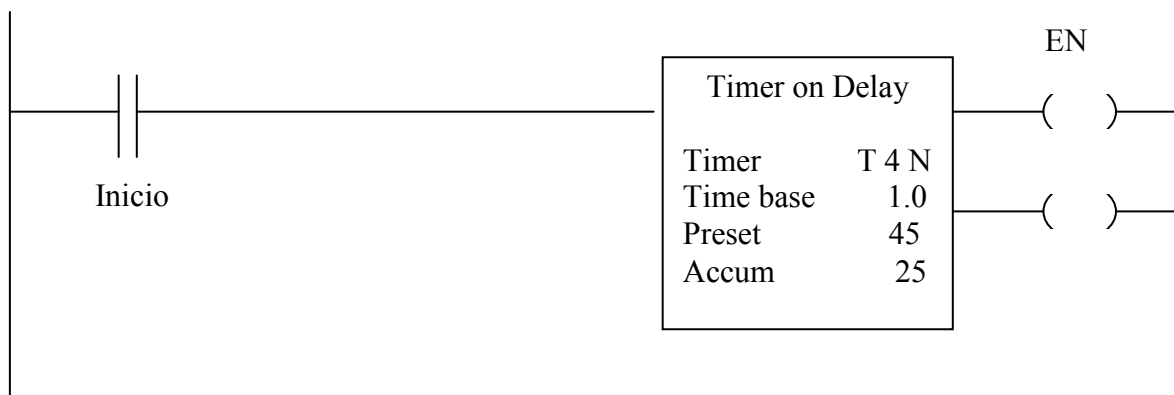
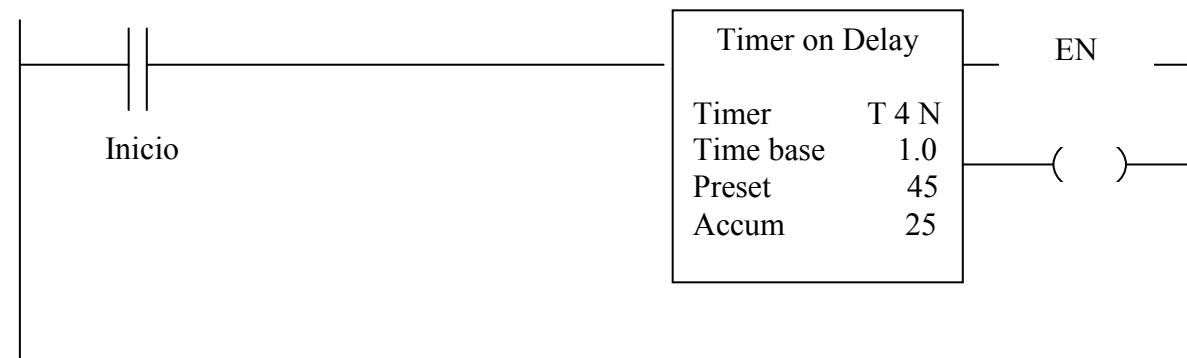


Fig 4.4 Diagrama de escalera de un Timer on Delay.

Para ello existe el timer ON Delay que es una función de salida que hace las veces de cronómetro, empezar a contar el tiempo una vez que el renglón está colocado como verdadero hasta llegar al tiempo meta y finalmente cuando el renglón se hace falso el timer se reinicia. Estas funciones de timer utilizan 3 palabras, podemos entender por palabra un espacio de memoria del PLC de 16 bits, la primer palabra que utiliza el timer es el preset en la cual se va a establecer el tiempo meta al que queremos llegar, la segunda palabra es el acumulador que es donde se almacena el tiempo que ha transcurrido desde que el renglón se hizo verdadero y finalmente la tercer palabra son los bits de control es una palabra de 16 bits donde se utilizan básicamente los 3 primeros bits para establecer banderas que nos permiten colocar o controlar el timer a lo largo del proceso indicado.

La primer bandera EN de enable que mantendrá un 1 mientras el timer esté energizado, existe la de TT que es la del timer está temporizando, la cual estará en 1 siempre y cuando esté energizado el timer y no se haya llegado al tiempo meta. Finalmente tenemos el DONE que indica que el tiempo meta ha sido alcanzado. Veamos ahora como integraríamos estos bits de control en el programa, imaginemos que tenemos este diagrama y hemos dado el botón de inicio subiendo esta palanquita en ON, y vemos que el renglón se hizo verdadero por lo tanto el bit de control

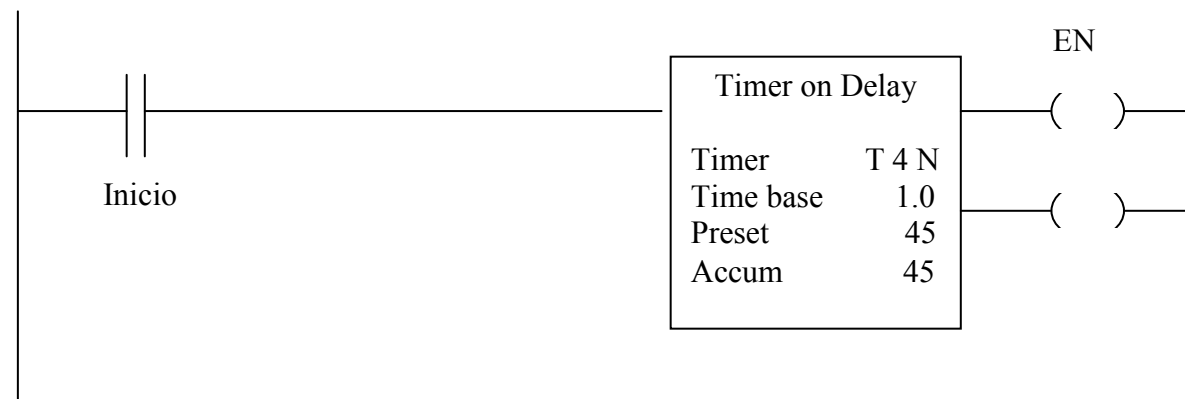


Bit	Estado	Descripción
EN	1	Renglón verdadero
TT	1	No ha llegado a preset
ON	0	No ha terminado

Fig 4.5 Diagrama de escalera del Timer on Delay y estado e bits.

EN está en 1, y el TT está en 1 ya que no hemos llegado al tiempo meta que para este ejemplo es de 45 segundos ya que en el preset marca 45 segundos y el tiempo base es 1.0 segundos, lo que establece que el tiempo está dado en segundos.

Podemos observar que el DONE aún está en ceros, ya que no hemos llegado al tiempo meta de 45 segundos, esperemos un poco y observamos como cuando las dos manecillas se equiparen cambian los estados, sigue energizado dado que mantenemos el switch en on, sin embargo TON se fué a 1 y TT a 0 dado que hemos llegado ya al valor meta, cuando se va a falso los timers se reinician y el proceso ha terminado en un diagrama de escalera.



Bit	Estado	Descripción
EN	1	Renglón verdadero
TT	0	Ya no se temporiza
ON	1	Se ha llegado al preset

Fig 4.6 diagrama al final de l estado para bits del Timer on Delay.

Aquí tenemos ahora un diagrama de escalera modificado de lo que era el circuito de arranque y paro de motores, para que podamos energizar el motor y después de un tiempo determinado este se apague. Entonces vamos a ver como se evaluaría esto, se presiona el botón de inicio y entonces empieza ha hacer la evaluación, dado que se presionó el botón de inicio el control de flujo puede pasar sin ningún problema y podemos observar como el timer TON o contacto que está referenciado al timer que es normalmente cerrado permite el flujo ya que el timer no está energizado. En ese momento se enciende el motor y vamos a ver ahora como se haría la evaluación por parte del PLC para el diagrama de escalera.

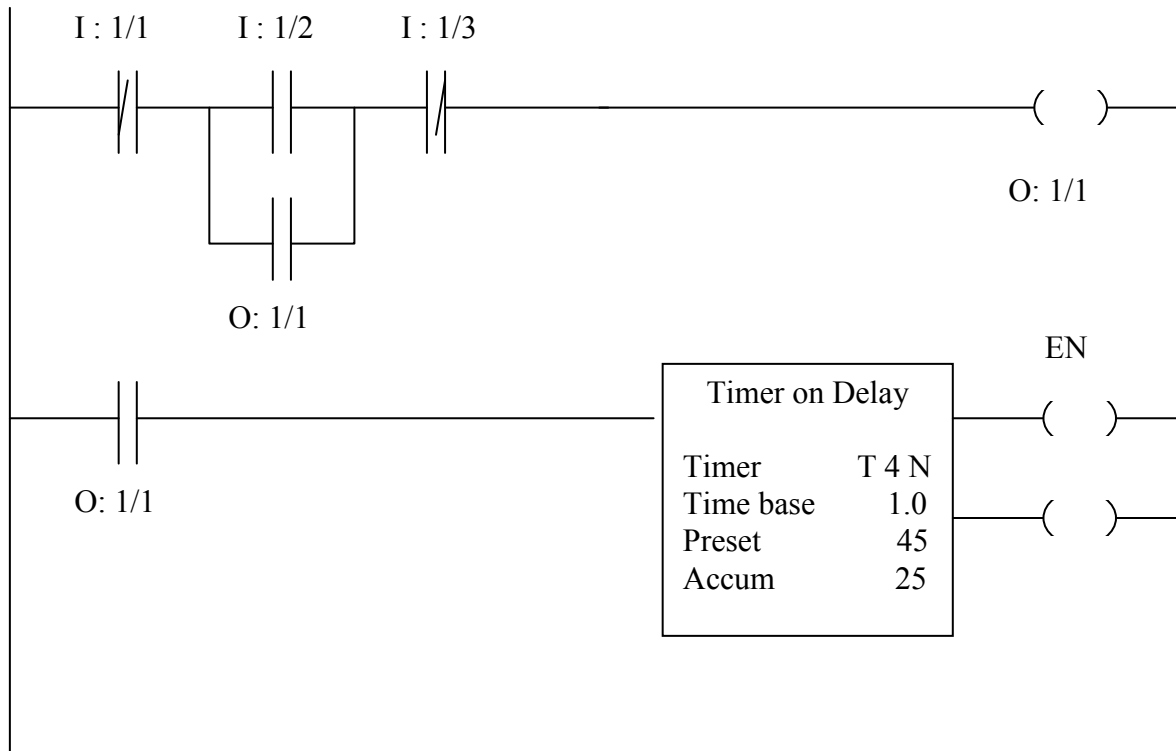


Fig 4.7 Diagrama de escalera de paro y arranque de un Timer on Delay.

Es importante recordar que debido al ciclo del programa en la primera iteración se ha encendido la bobina, sin embargo el contacto de referencia de bobina permanece desenergizado hasta que actualicemos las salidas de tal manera que el timer lo vamos a poder ver hasta la siguiente iteración, que sería esta al momento de evaluar, ya no está presionado el botón de inicio, sin embargo como el motor ya se energizó la bobina permanece ahí, se evalúa la segunda línea y empieza a contar el tiempo.

Y podemos ver como el tiempo va a estar manteniéndose cronometrado, hasta alcanzar el tiempo meta o tiempo de preset, y vemos como una vez alcanzado ese tiempo se evalúa, se llega al contacto de DONE que por ser normalmente cerrado al ponerlo en 1 se abre y el motor ya no puede ser energizado, de igual manera el timer se va a falso por lo tanto el timer se inicializa.

Hemos presentado al timer ON Delay, sin embargo es importante comentar que existe una segunda instrucción del timer llamada timer OF Delay, que funciona muy similar, sin embargo el comportamiento del bit de control es diferente, ya que esta función va a temporizar o va a cronometrar cuando el renglón sea falso.

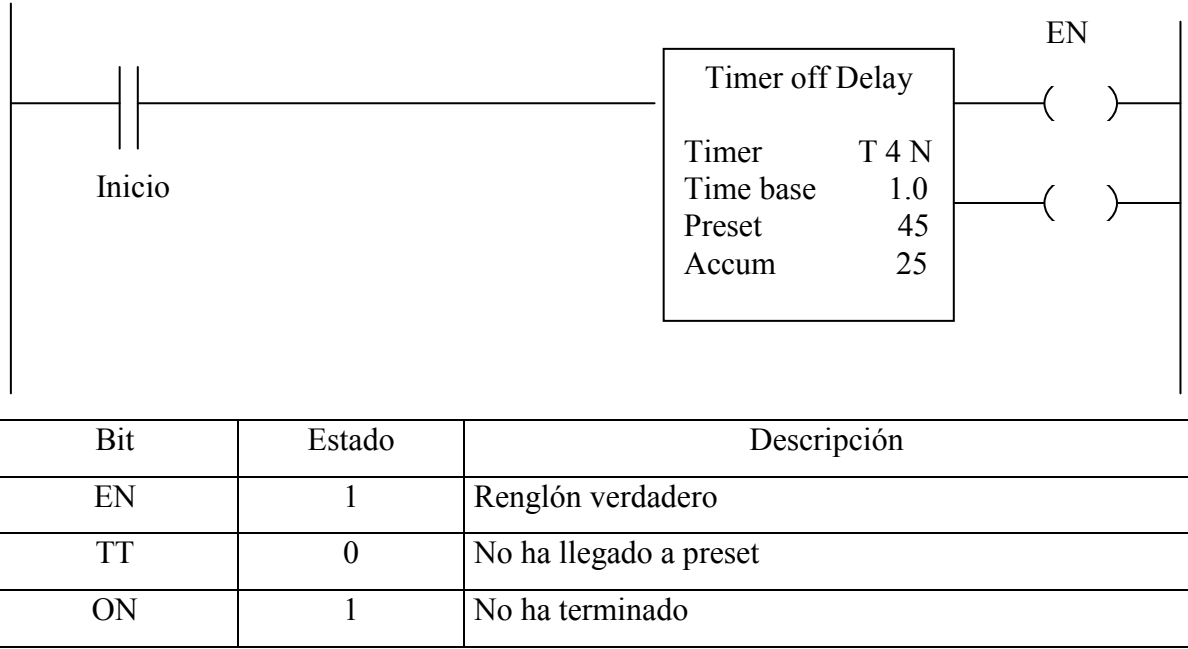
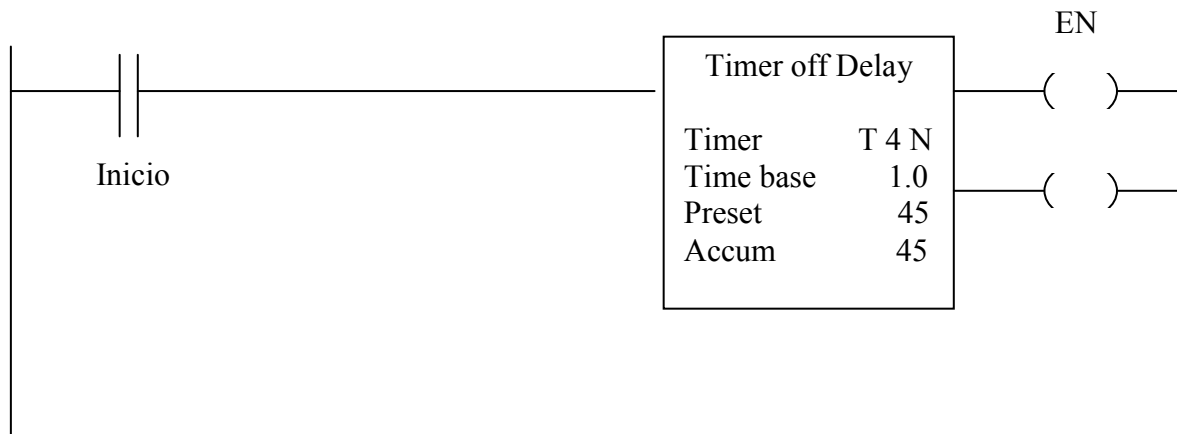


Fig 4.8 Diagrama de estado de bits de un Timer off Delay.

Como podemos observar en este ejemplo el TT esta en 1 está contando, sin embargo el bit de control EN es cero porque el renglón es falso, vamos a esperar para ver que sucede cuando llegamos al tiempo meta establecido en la palabra preset, y al llegar al tiempo meta veremos que DONE se va a 1, el TT a cero, y así se mantendrá hasta que se manden a verdadero los controles y entonces se re inicialice el timer, yéndose DONE a 0, TT a 0 y energizado BLE en inglés a 1.



Bit	Estado	Descripción
EN	0	Renglón verdadero
TT	0	Llega a preset
ON	1	Ya ha terminado

Fig 4.9 Diagrama de estado final de bits del Timer of Delay.

4.2 CONTADORES

Principios de programación, la función de contadores que ofrece la capacidad de integrar un contador al diagrama de escalera de un circuito sencillo, en ocasiones durante los procesos industriales es necesario contar eventos por ejemplo número de botellas, número de veces que algo ha sucedido, entonces podemos preguntarnos que hacer en ese caso, bueno para ello existe una función dentro del diagrama de escalera llamada counter UP o contadores que nos van a permitir llevar el conteo de eventos que se hayan suscitado, estos contadores al igual que los timers utiliza 3 palabras, recordemos que cada palabra es un espacio de memoria en el PLC compuesto por 16 bits.

La primera palabra que utilizamos para el contador es PRESET que es la palabra donde se almacenan el número de cuentas o número de eventos al que deseamos llegar, la siguiente palabra es ACUM que es el número actual de esos eventos que se han dado que estaremos esperando llegar al PRESET y finalmente los bits de control que al igual que con el timer,

son bits que nos van a permitir conocer el estado del contador y utilizarlo en la lógica de nuestro programa.

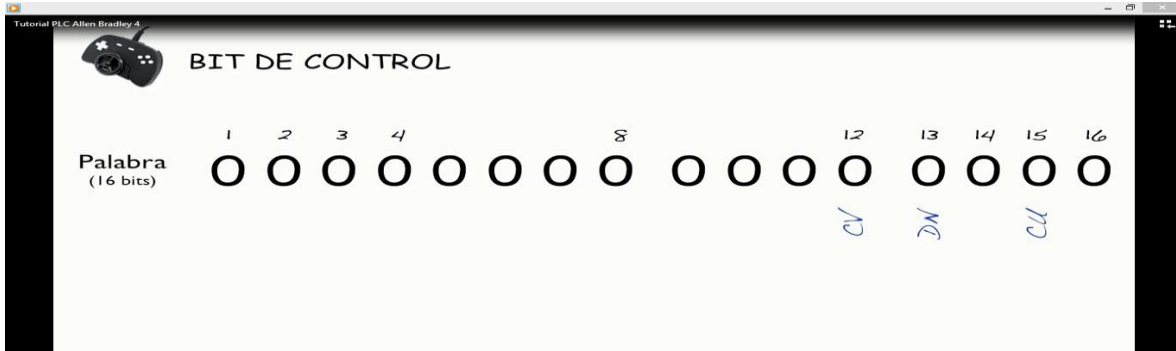


Fig 4.10 Representación de la palabra de 16 bits indicando bits de control 12, 13, 15.

Tenemos básicamente 3 contactos que nos van a servir pero el primero es el CV nos indica si el acumulado ha rebasado el límite del contador o rango de operación es de - 32, 768 hasta + 32, 768 este rango es porque 2 a la 15 nos daría ese valor y el bit 16 avo se utiliza únicamente para indicar si es un signo positivo o negativo.

El siguiente bit que podemos utilizar es DONE que al igual que los timers una vez que el preset ha sido alcanzado o de otra manera que el acumulador es igual al preset, entonces este bit se pondrá en 1.

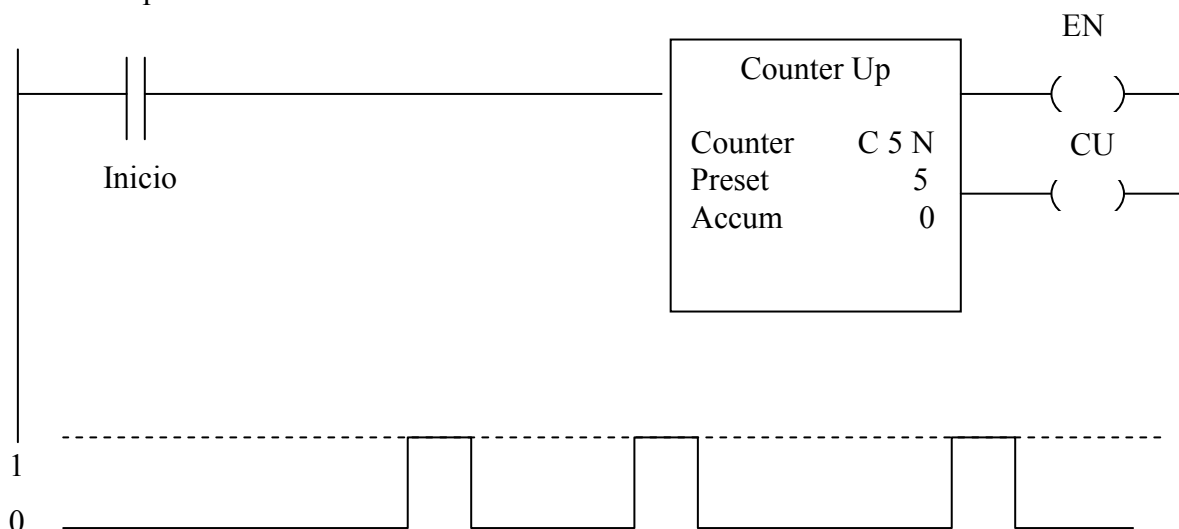


Fig 4. 11 diagrama de escalera de un contador ascendente.

Existe el otro bit de control que es counter UP que lo que indica es cuando el timer está habilitado cuando el renglón está en verdadero. Vamos ahora a ver en este diagrama de escalera que tenemos aquí como se va haciendo la secuencia, tenemos ahí un objeto que va subiendo hasta llegar a un sensor y notamos que cuando llega al sensor se va a 1 el valor y el contador a incrementado en 1, nuevamente vuelve a subir y al llegar al sensor incrementa la cuenta en 1, de tal manera que estamos dándonos cuenta que el contador únicamente va a incrementar la cuenta cuando se detecte el flanco de subida, es decir cuando la señal del renglón vaya de falso a verdadero.

Vamos a ver ahora como podemos integrar un contador a un diagrama de circuito de arranque, a un diagrama de arranque y paro de motores, ahora cada vez que presionamos el botón de START encenderíamos el motor e incrementaríamos la cuenta la primer cuenta, la segunda cuenta y la tercera cuenta. Que sucede que una vez que hemos alcanzado o llegado al PRESET como está configurado en este ejercicio, el contacto referenciado a este contador al contacto de TON se ha cerrado, por lo tanto si volvemos a presionar lo que podemos ver es que el flujo llega ahí y no se puede cruzar, por lo tanto no podríamos prender mas de 3 veces este sistema.

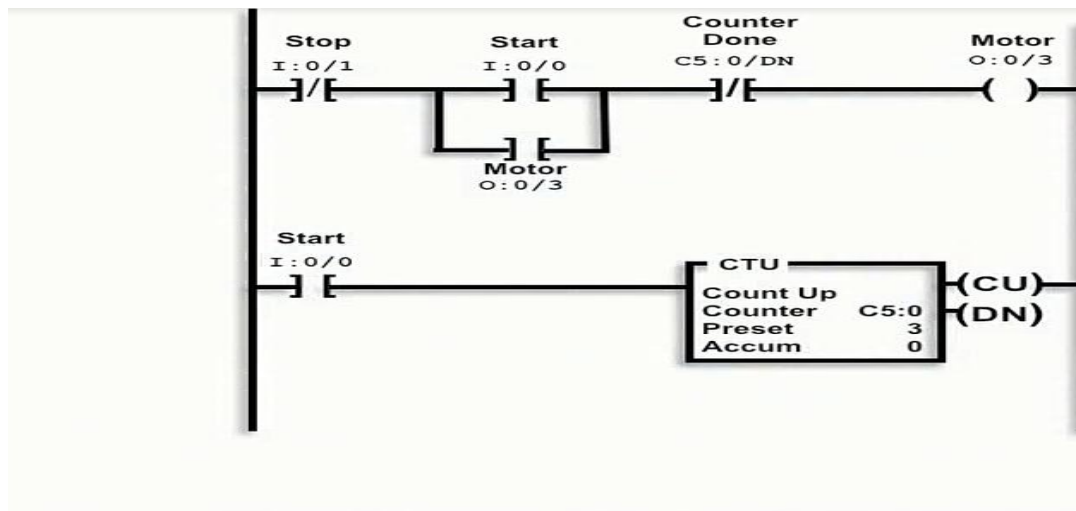


Fig 4.12 diagrama en paro y arranque de un contador ascendente.

Bien ahora como hacemos para poderlo realizar, pues seguramente hemos llegado al número 3 y queremos volver a prenderlo sin tener que apagar el PLC para poder reiniciar esta función poniendo valores a 0.

Para ello existe una función de RESET la cual va a permitir que los contadores se vayan a cero y los bits de control también se vayan a cero. A diferencia con el timer, que el timer se inicia únicamente mandando a falso el renglón aquí es necesario utilizar esta función de RESET, ya sea porque la conectamos a un botón o alguna otra condición dentro de nuestra lógica.

Con las funciones de contadores tenemos 2 funciones una que nos va a ayudar a incrementar la cuenta durante la lógica que es la llamada counter UP, y otra función que nos va a permitir decrementar esa cuenta que es la función llamada counter DOWN.

Vamos a ver un ejemplo de cómo podríamos ya en un programa estar utilizando los contadores y los timers relacionados en una lógica de un proceso industrial, en este caso tenemos 2 pistones, una banda, un sensor y un cabezal de llenado.

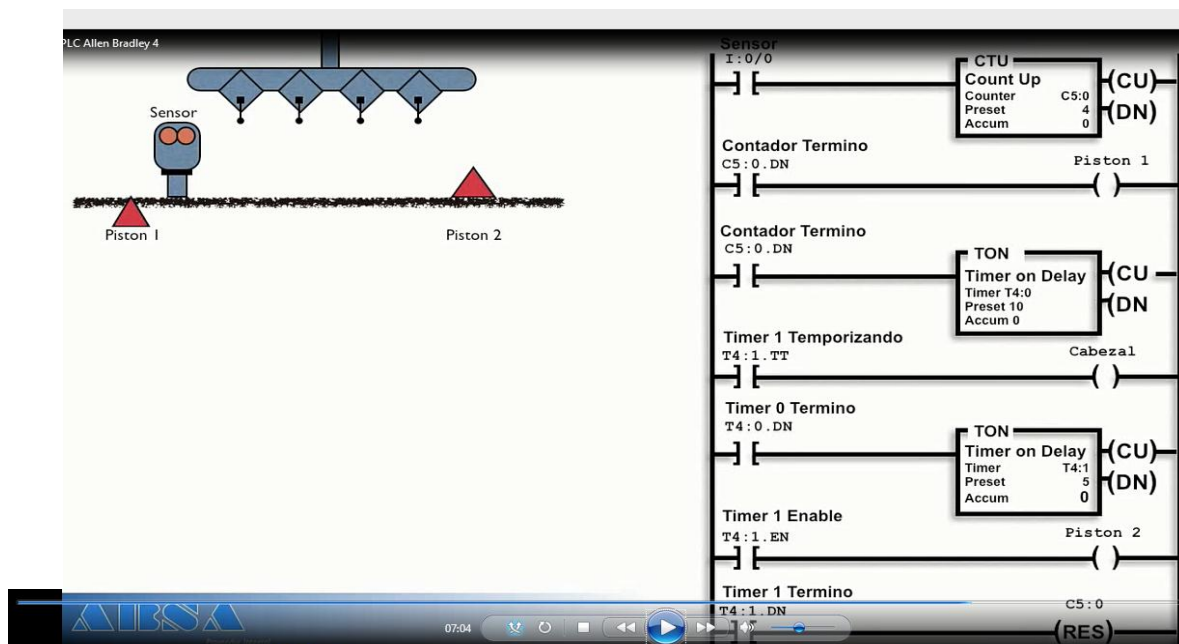


Fig 4.13 Primer peldaño del diagrama de escalera y el proceso industrial.

Esta es una línea de llenado típica en la que se pretende llenar 4 botellas a la vez por lo que es necesario utilizar el pistón 1 para detener la llegada de nuevas botellas, y el pistón 2 para detener aquellas botellas que están por llenarse.

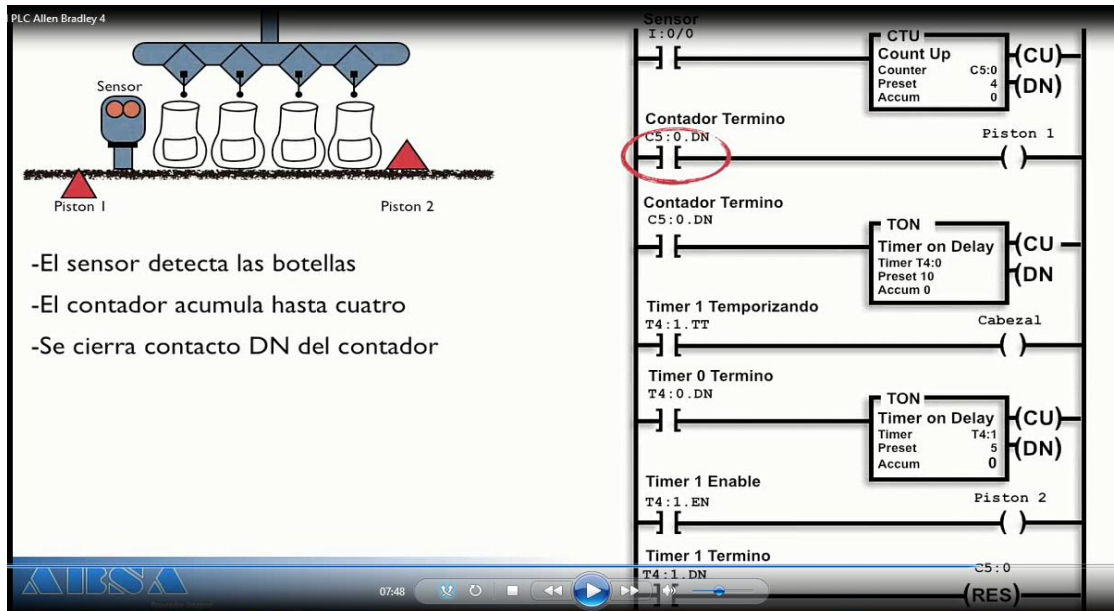


Fig 4.14 Segundo peldaño del diagrama de escalera y el proceso industrial.

Entonces que ocurrirá, el sensor empieza a detectar las botellas y una vez que se detectan y se ha llegado al acumulado esperado el contacto del contador se cierra lo que permitiría entonces activar el pistón 1 para detener las demás botellas que puedan venir detrás de ellas, por lo tanto también iniciaría el conteo del timer 1.

Habilitando el bit de temporizado que haría que el cabezal bajara durante el tiempo que se especificó en el timer, posteriormente el cabezal subiría y se habilitaría entonces el segundo timer, estamos dándonos cuenta ahora que podemos poner timers en cascada, es decir el segundo timer se va a activar cuando se de la señal de TON del primer timer y eso permitirá que el pistón 2 baje, para permitir que las botellas salgan.

Habilitando el bit de temporzizado que hace que el cabezal bajara, durante el tiempo que se especificó en el timer 1.

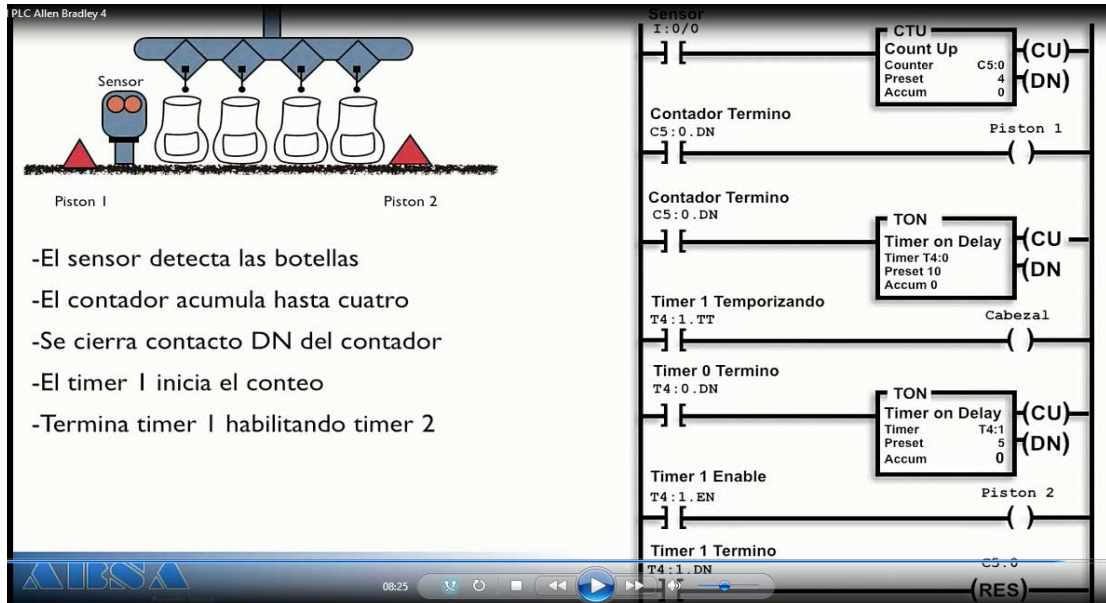


Fig 4.15 Quinto peldaño del diagrama de escalera y el proceso industrial.

Y posteriormente el cabezal subiría y se habilitaría entonces el segundo timer, estamos viendo ahora que podemos poner timers en cascada, es decir el segundo timer se activa cuando se da la señal de DN del primer timer y eso permitirá que el pistón 2 baje y eso permitirá que las botellas.

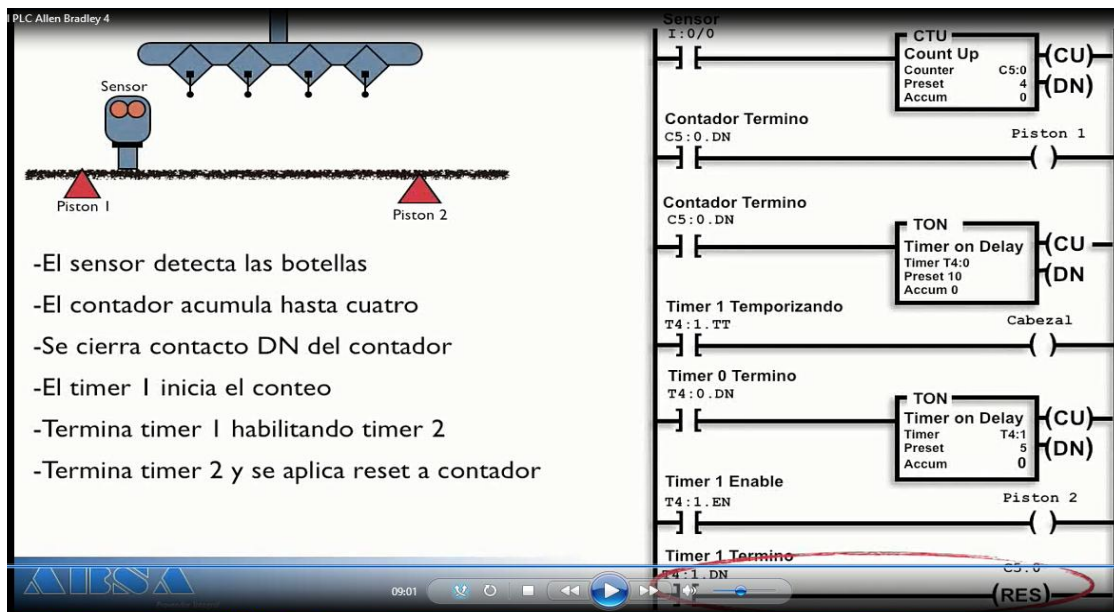


Fig 4.16 Séptimo peldaño del diagrama de escalera y el proceso industrial.

Y así durante el tiempo especificado hasta que se de la señal de DONE que lo que va a hacer es poner una función de RESET que como ya habíamos dicho va a inicializar todo en 0 y todo vuelve a su estado original.

4.3 CONFIGURACIÓN DE DRIVERS

Los drivers son para podernos conectar al PLC para descargar un programa o hacer algún mantenimiento del PLC, para esto sabemos que es un PLC, sabemos como hacer un diagrama de escalera, pero como hacemos que el PLC lo asimile o mandárselo.

Para ello es importante tener primero una computadora, en la cual deben estar instalados 2 software, el RS – Logix 500 / 5000 dependiendo el hardware que estemos utilizando, por ejemplo el PLC microLogix utiliza el software PLC 500 y el control Logix que es un PLC mas grande con mas capacidad utiliza el Logix 5000. También es necesario tener el RS Linx Classic que es el Software que nos permite administrar la comunicación con el PLC a través de los drivers.

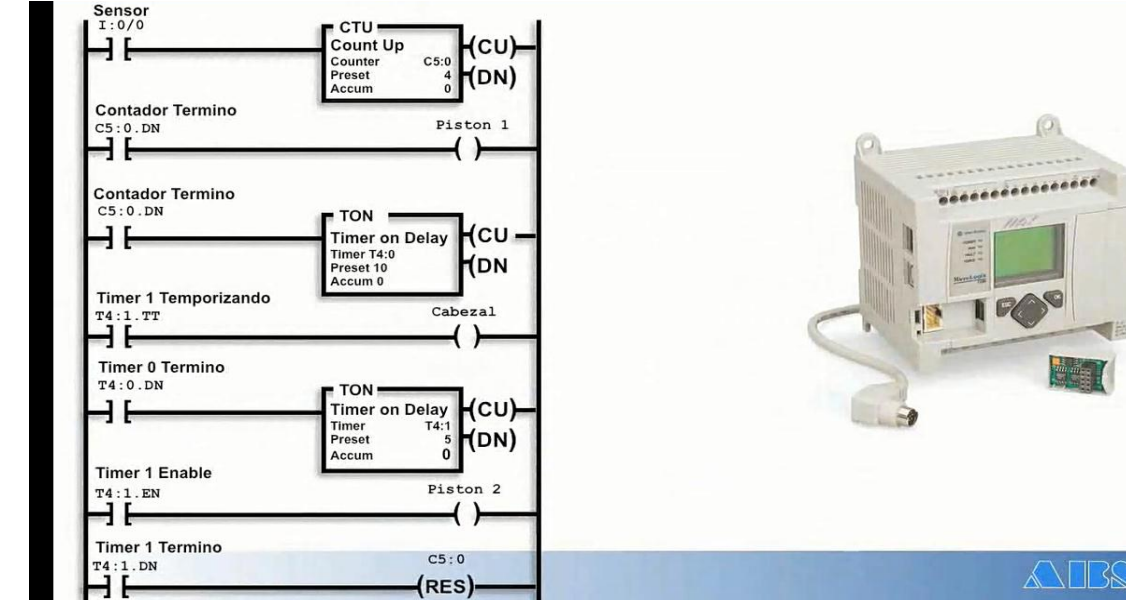


Fig 4.17 PLC mostrando el contenido del programa del proceso de llenado de botellas.

Por ejemplo tenemos una computadora conectada por algún medio físico a un cable a un PLC, y ya sea que se esté utilizando un protocolo Ethernet, Device, DH+, Serial, Control NET, lo que estamos viendo es que la computadora se conecta al PLC por diferentes protocolos de comunicación industriales, pero necesitamos que exista alguien que administre esa comunicación, eso es un driver.

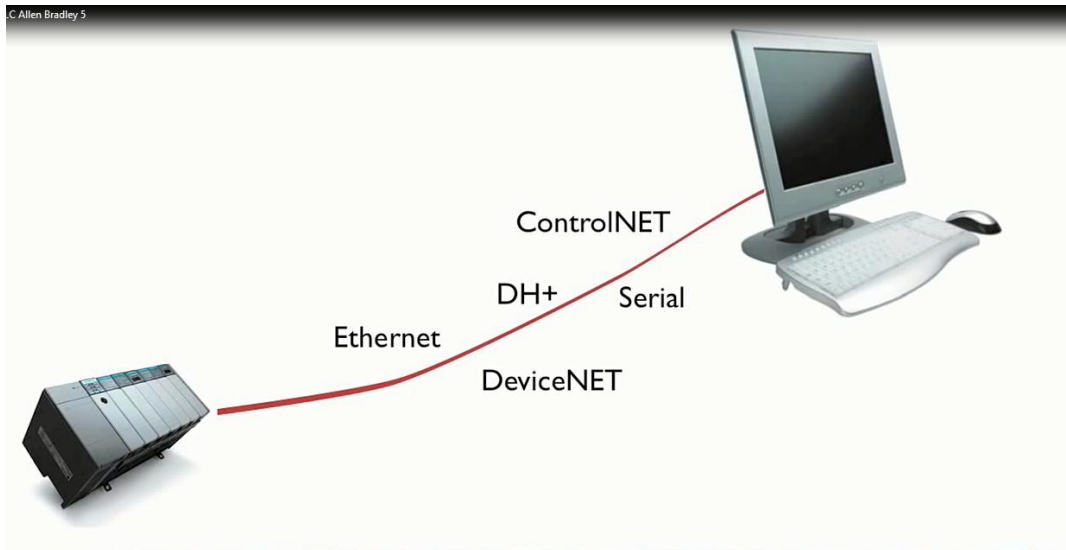


Fig 4.18 Conexión del puerto RS 232 al PLC y la computadora.

Una computadora y un PLC son los destinos, tenemos un cable que conecta a ambos como si fuera una carretera virtual, el driver administra los protocolos de comunicación, el driver es una instancia de software que se encarga de administrar por donde y porque protocolo de comunicación se va a comunicar el PLC para que la información se pueda mandar directamente al PLC. Vamos a entrar a RS Linx Classic dándole un clic al icono, nos muestra esta pantalla donde tenemos en el toolbar básicamente 3 botones o 3 íconos, el primero que es RS Hub que nos va a permitir visualizar a manera de un explorador de windows en un diagrama de árbol los diferentes drivers que tenemos y a su vez del lado derecho los dispositivos que estén conectados a través de ese driver o ese puerto de comunicación.

El siguiente icono que es al que vamos a hacer clic en este momento es el que nos permite configurar los drivers que tengamos disponibles en la computadora y que necesitamos

utilizar, para ello vamos a seleccionar este combo desplegable, y va a desplegar todos los drivers disponibles, que tenemos en esta computadora, por ejemplo podemos darnos cuenta que tenemos el RS 232, tenemos el driver del Ethernet, un DH485, o un emulador de driver para el PLC, Soft Logic5, DeviceNET, ControlNET, tenemos diferentes protocolos de comunicación disponibles, en este caso vamos ahora a seleccionar el RS 232, una vez que lo hemos seleccionado presionamos Add New y nos va a poder permitir ponerle un nombre que tenga algún significado para nosotros o poner el nombre que por default nos propone el software.

Para este caso vamos a ponerle serial 232 y presionamos el botón de OK, una vez que hemos seleccionado el nombre y que nos va a abrir una nueva pantalla donde podemos seleccionar y configurar algunos parámetros. Esta pantalla va a cambiar dependiendo del driver que estemos conectando o configurando en este caso es un serial tenemos que seleccionar que puerto de comunicación queremos usar en la computadora, este es el com 3, a que dispositivo queremos conectar si es un PLC en nuestro caso. Y datos clásicos de paridad, baudios / segundo, bien podemos dar clic en autoconfigurar, lo que va a hacer es un escaneo sugiriendo la mejor combinación, una vez que tenemos el driver configurado le damos OK.

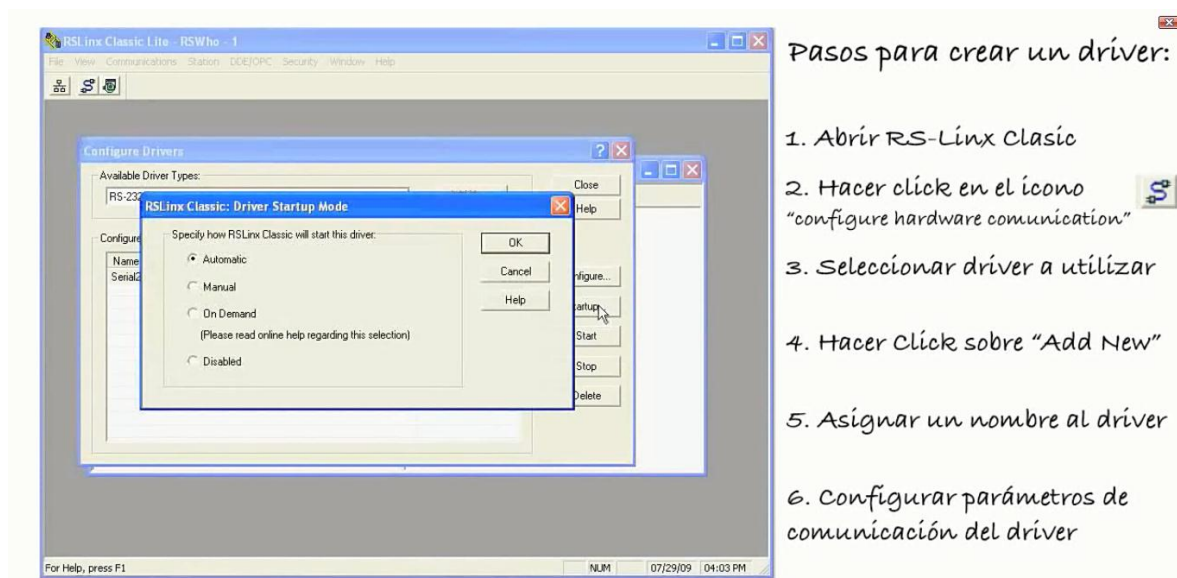
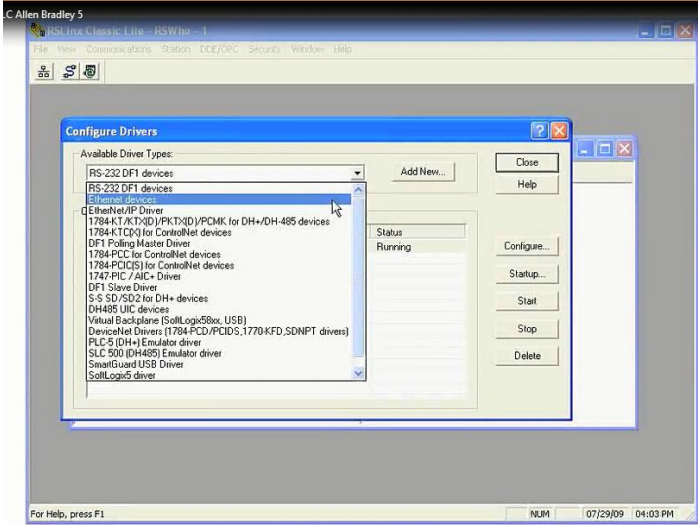


Fig 4.19 Ventana con RSLinx en Automático.

Nos damos cuenta que en este momento ya aparecen aquí en los drivers configurados, bien podemos configurarlo nuevamente y olvidar la nueva pantalla, bien podemos modificar sus parámetros de cómo queremos que arranque este driver, si queremos que sea de manera automática siempre que RS Linx arranque, o bien sea manual. Podemos forzar un arranque, pararlo, vamos ahora a crear otro driver de tal manera que con esto estamos dándonos cuenta que en la misma computadora podemos tener una infinidad de drivers configurados sin necesidad de trabajar con uno o con otro, podemos trabajar con 2 o más a la vez. Vamos a trabajar ahora con Ethernet, vamos a presionar nuevamente Add New y vamos a modificar el nombre como dijimos, es importante que el nombre no tenga caracteres especiales ya que nos va a marcar un error.

Podemos seleccionar si queremos que busque todos los PLC en la misma red local o bien especificar una IP diferente a la que queremos, esto lo que nos va a permitir es que podamos utilizar nuestra computadora en una red, y probablemente nuestros equipos de campo estuvieran en otra red. Para mas información es importante consultar al administrador de redes de cada una de las empresas para saber que IP's podemos asignar.



Pasos para crear un driver:

1. Abrir RS-Linx Clasic
2. Hacer click en el icono "configure hardware communication"
3. Seleccionar driver a utilizar
4. Hacer Click sobre "Add New"
5. Asignar un nombre al driver
6. Configurar parámetros de comunicación del driver

Fig 4.20 Ventana para configurar drivers para el PLC.

En este momento tenemos configurados ya 2 protocolos de información de 2 drivers, es decir nos podemos comunicar con equipos seriales o bien con equipos que estén en la red local. Vamos a dar Close y ahora podemos ver como en nuestro árbol pantalla de RS hub del explorador del lado izquierdo aparecen ya nuestros diferentes drivers y del lado derecho aparecerán al momento de expandir estos drivers los PLC's que tengamos conectados en este momento a estos dispositivos. Con esto hemos configurado ya el protocolo de comunicación ya le hemos dicho a la computadora por donde queremos que se comunique con el PLC y nos permita entonces crear un proyecto nuevo de diagrama de escalera y descargarlo así como monitorear las señales del PLC.

4.4 PROGRAMA RS LOGIX 500

Principios de programación, como se utiliza el RS Logix software, debemos saber que son los timers y contadores, diagramas de escalera, como crear un driver. El primer paso para crear un proyecto de automatización en el PLC es la creación del driver, vamos a ver como lo creamos, como ya dijimos el primer paso para crear un proyecto en SR Logix 500 es la creación de los vínculos de comunicación conocidos como drivers, para repasarlo hacemos clic en RS hub para poder ver que drivers tenemos configurados, hacemos clic en configuración de hardware, para seleccionar que protocolo de comunicaciones es el que vamos a utilizar. Nosotros ahora vamos a utilizar Ethernet IP, vamos a crear al hacer clic en Add New y podremos cambiar el nombre si así queremos, en este caso le vamos a poner Ethernet.

Vamos a dejar seleccionado el Browse local Net para que busque en todos los equipos que estén conectados en la misma red que nuestra computadora está. Una vez que ya aparece el driver en esta parte diciéndonos que esta corriendo hacemos clic en close y aparece en el árbol del explorador de dispositivos Ethernet dado que así le llamamos a nuestra conexión y debajo de ella van a aparecer todos los dispositivos conectados a esa red.

El segundo paso para desarrollar un proyecto con el PLC es desarrollar ya el diagrama de escalera para lo cual es necesario crear un nuevo proyecto, desarrollar el nuevo programa,

es decir construir la lógica con la que queremos que ese proyecto vaya desarrollándose o ejecutándose y finalmente descargar esa lógica o ese programa al PLC.

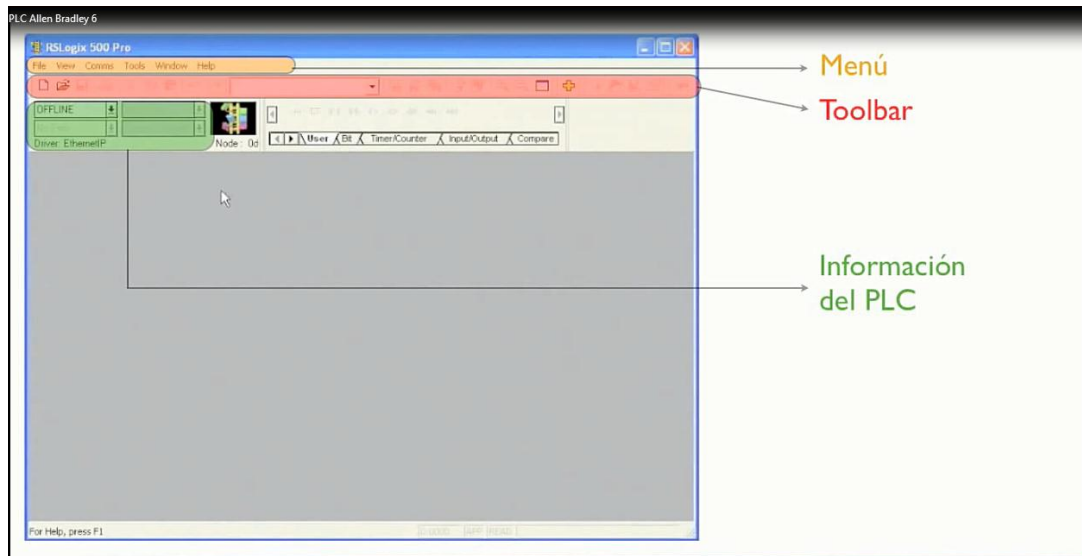


Fig 4.21 Ventana para diagramas de escalera en RSLogic500.

Una vez que ya hemos creado el driver en RS Linx vamos ha hacer doble clic en RS Logix 500 para iniciar nuestro proyecto con un PLC. Al igual que casi en cualquier otro programa, la pantalla principal de RS Logix 500 tiene un menú desde el que podemos acceder a diferentes funciones que son necesarias para la operación de este programa, existe también el toolbar o estos íconos que podemos utilizar para acceder a las instrucciones del programa que mayormente se utilizan. Este apartado o parte de la pantalla nos va a dar información específica sobre lo que está sucediendo con el PLC, si está en línea, si está corriendo un programa. En este apartado de la pantalla o parte del programa lo que vamos a tener es una bandeja donde se contienen todas las funciones que podremos utilizar para el diagrama de escalera.

El primer para crear un proyecto o el diagrama de escalera es indicar un nuevo procesador a utilizar y para ello vamos ha hacer clic en new, y nos va a preguntar como primer punto que nombre le queremos poner, tenemos que seleccionar que tipo de controlador estaremos utilizando, esto es muy importante porque dependiendo del controlador que seleccionemos son algunas de las funciones que podremos tener disponibles o no. En nuestro caso tenemos

un MicroLogix 1100 serie b, vamos a decir incluso aquí podemos seleccionar Hub active, vamos a darnos cuenta que nos muestra una especie del RS Linx, donde podemos seleccionar con qué PLC nos queremos conectar, en nuestro caso es el PLC que queremos, entonces aparece ya la pantalla lista para empezar a programar el primer diagrama de escalera.

Vamos ha hacer algo muy sencillo y vamos a poner en nuestro diagrama de escalera un contacto, vamos a poner ahora una bobina de salida, ahora vamos a descarga este breve programa que hemos hecho al PLC y para ello seleccionamos download, nos pide que gravemos este proyecto generalmente es un archivo que es muy importante tener guardado y es parte de los procedimientos, lo vamos a guardar en el Escritorio.

En este momento podemos observar que el PLC está en modo de run o modo de correr y tenemos las líneas verticales del diagrama de escalera en verde indicando que se está evaluando el PLC o el flujo de información. Vamos a presionar ahora físicamente el botón y podemos observar como el contacto se pone en verde y el foco se pone en verde. Al momento de dejar de presionar el botón, el indicador o contacto se pone nuevamente en abierto, se apaga el foco.

PLC.

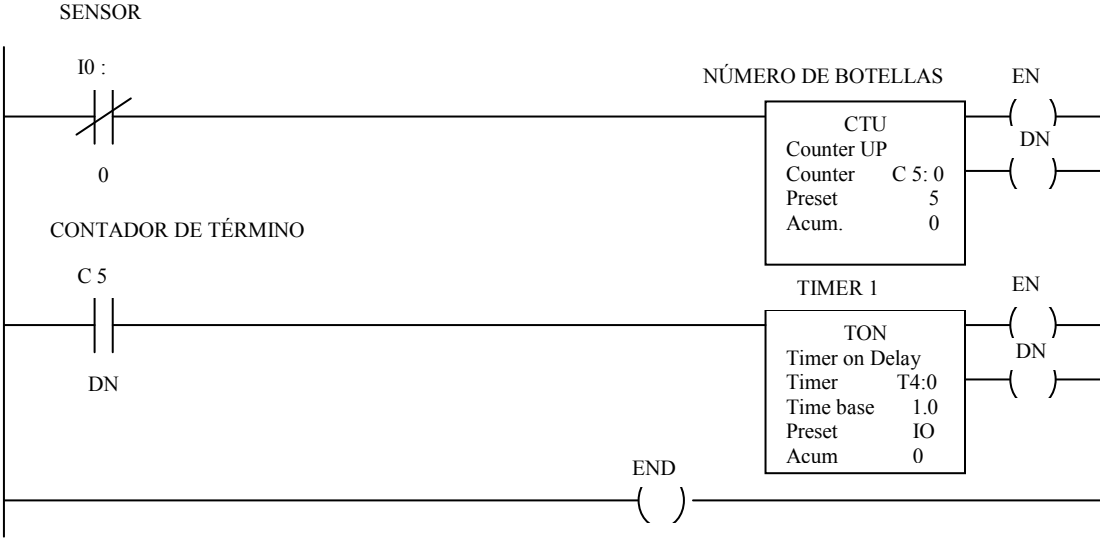


Fig 4.22 diagrama de escalera en RSLogix 500 de contadores.

Vamos a crear un nuevo proyecto que vamos a nombrar banda, seleccionamos el PLC que es el MicroLogix 1000 serie b, seleccionamos con que PLC vamos a querer estar trabajando, damos OK y tenemos el sistema listo para empezar. Vamos a poner un contacto, vamos a agregar un contacto, vamos a agregar un contador que lo podemos sacar de la categoría de contadores, le ponemos un preset de 5.

Vamos a agregar una nueva línea que la sacamos de la categoría de user, vamos a colocar un contacto normalmente abierto, una vez que el contador haya terminado lo que queremos es que se energice el timer colocamos otro contacto normalmente abierto nombrado con la salida del primer timer en esta segunda línea, por lo que vamos a colocar un TON tiempo base 1, y que remos que cuente hasta 10 segundos.

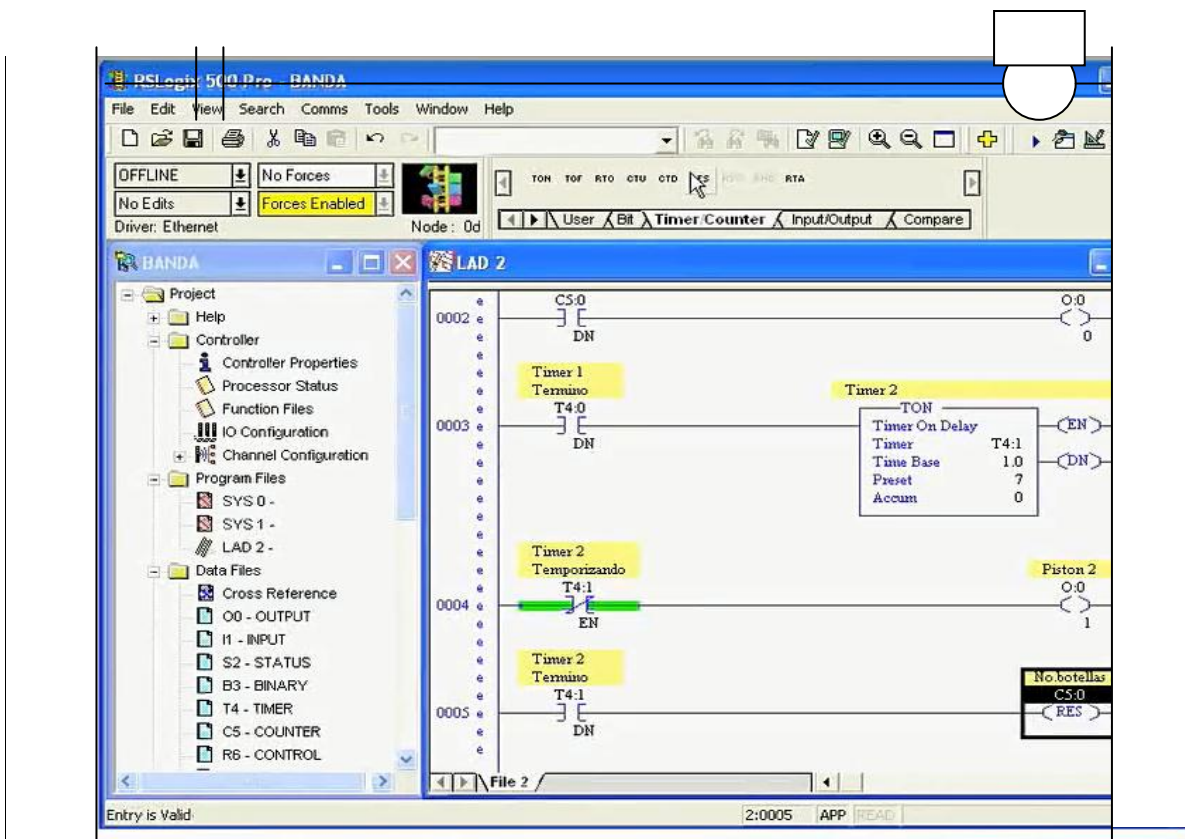


Fig 4.23 Diagrama de escalera mostrando el timer 2.

Vamos a colocar una nueva línea que obtenemos de la categoría de user y ponemos un contacto normalmente abierto, mediante otro vamos a copiar el segundo contacto y pegarlo en la tercera línea y vamos a colocar una bobina de salida. Ahora vamos a colocar una nueva línea donde colocaremos un contacto normalmente abierto, y colocaremos un timer que va a estar en cascada con una nueva línea vamos a colocar un contacto normalmente cerrado, ahora colocamos una bobina de salida. Ahora queremos que esto se reinicie cuando el timer llegue al tiempo deseado y vamos a poner una función de RESET.

4.5 DIRECCIONAMIENTO EN RS LOGIX 500

Sobre direccionamiento en RS Logix 500 como se direccionan las entradas, las salidas y los espacios de memoria cuando utilizamos RS Logix 500. En el ciclo del programa el primer paso se pasan los datos de campo a las tarjetas de entrada las cuales convierten a ceros y unos la información y lo mandan a los espacios de memoria para que de ahí se vaya a la lógica que hemos programado en el diagrama de escalera y los resultados de esta ejecución, se almacenen nuevamente en un espacio de memoria el cual sea enviado a las tarjetas de salida, de tal manera que finalmente podamos actuar sobre los elementos de campo como pueden ser bombas, motores, etc.

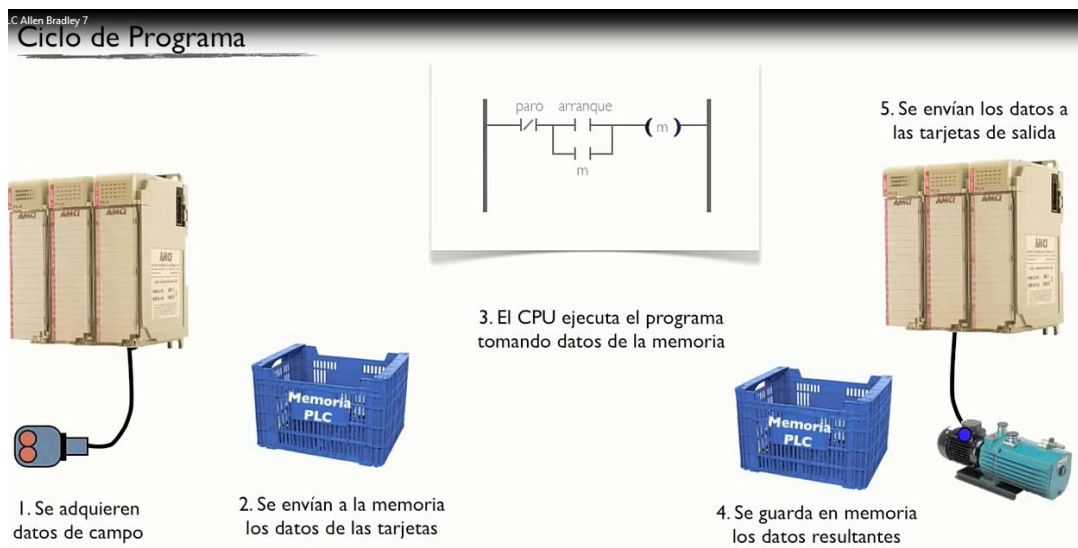


Fig 4.24 Como controla el PLC un programa; ejemplo sencillo.

Pero que son estos espacios de memoria, en los PLC tenemos espacios de memoria que son archivos de datos, archivos de funciones y archivos de programa. Ahora vamos a conocer los archivos de datos es donde se almacena la información de números de datos enteros, archivos de control, contadores, timers, archivos de bits, entradas, salidas.

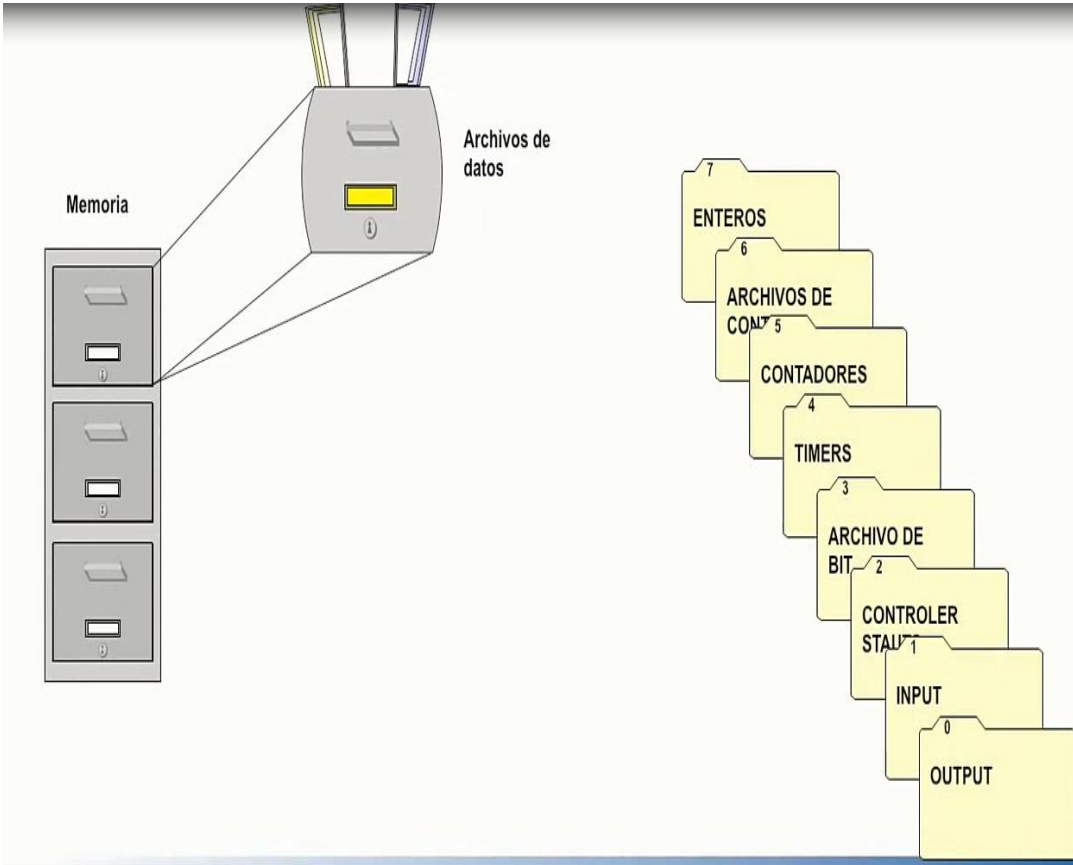


Fig 4.25 El PLC ubica distintos tipos de datos en diferentes archivos.

El número de la tarjeta o palabra y el bit será el número del canal o pin que queremos usar de la tarjeta, de igual manera los archivos de entrada estarán en el archivo 1, el número de archivo es 0 o palabra que es la tarjeta, el bit es el 0 o pin de la tarjeta. Estos son espacios físicos.

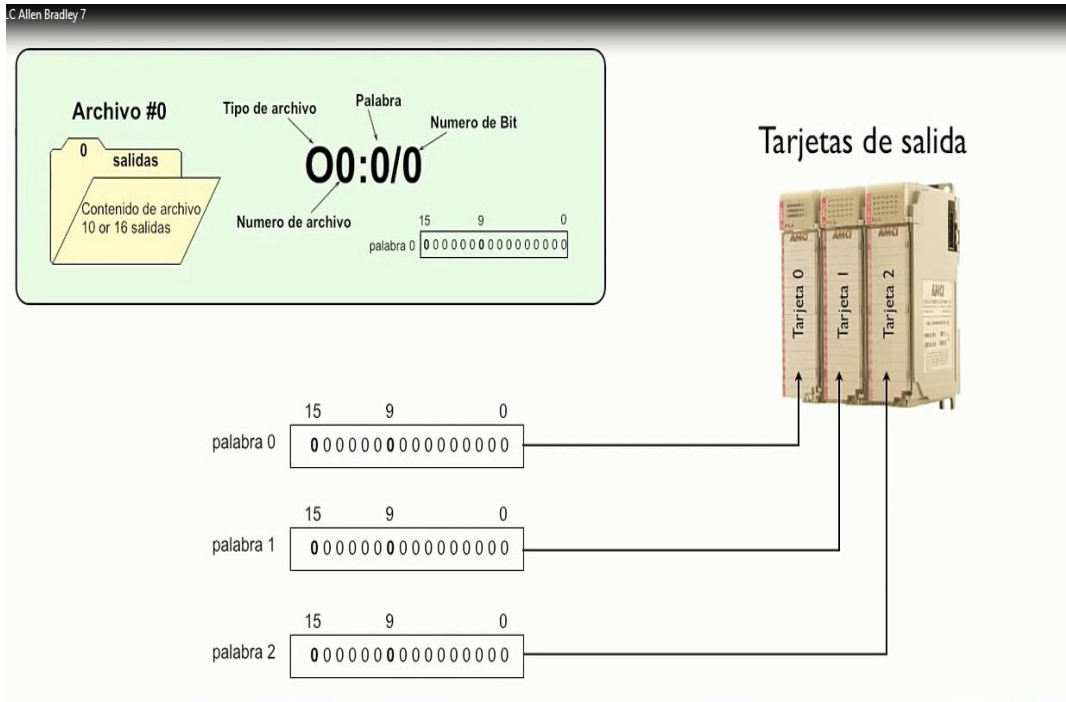


Fig 4.26 Representación de las palabras de bits en las tarjetas del PLC.

Sin embargo existen espacios de memoria donde necesitamos tener alguna bandera para indicar el estado de una cosa, almacenar cierta información o ciertos bits, para ello existe el archivo número 3 donde la nomenclatura es B3 la palabra, debemos saber que dependiendo del PLC típicamente podemos tener hasta 256 palabras, y hemos dicho que cada palabra contiene 16 bits, podemos imaginar el archivo de B3 como una hoja de excell donde cada fila representa una palabra y cada columna representa uno de los bits de esa palabra.

The diagram illustrates the mapping of bit words to PLC memory. At the top right, a file structure for 'Archivo #3' (Bits) is shown, containing a folder '3' and a document 'Contenido de archivo'. The address 'B3:0/0' is displayed, with labels for 'Tipo de archivo', 'Palabra', and 'Numero de bit'. Below this, a table shows the bit values for 'Palabra 0' through 'Palabra 3' across 16 bits. The table is titled 'BITS' and has columns numbered 1 to 15. The values are as follows:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Palabra 0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Palabra 1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Palabra 2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
Palabra 3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Fig 4.27 Tipo de archivo B y diferentes palabras almacenadas.

Existen el archivo 4 y el 5 para el temporizador y para contadores en los procesos y los programas del PLC de utilizar números enteros ya sea para algún cálculo o para llevar el acumulado de alguna cosa. Para ello existe el archivo número 7 donde su nomenclatura es N entero siguiéndole el número de archivo N7 : 0.

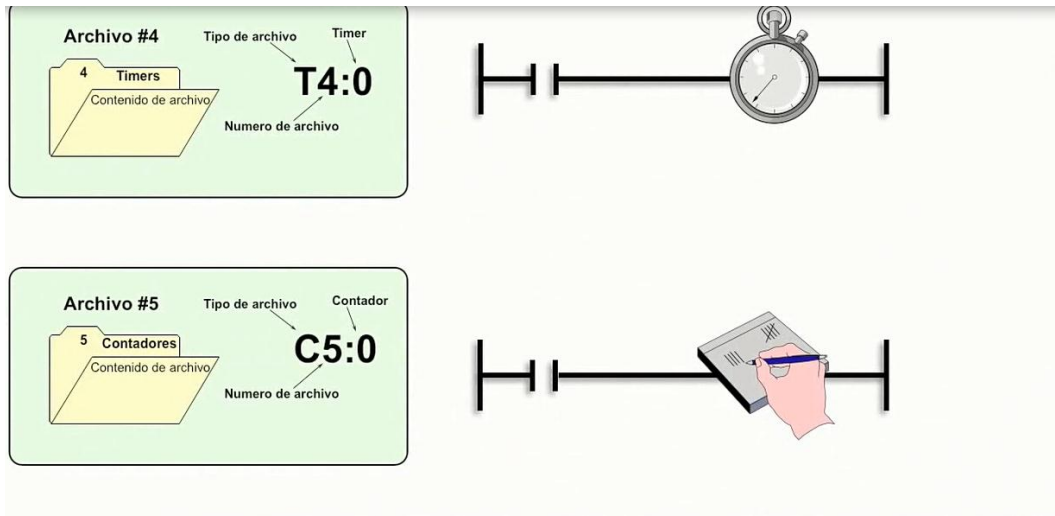


Fig 4.28 Archivo de datos tipo contador, timer y almacena el estado del mismo.

Notamos que aquí no estamos llegando al nivel del bit, nos estamos quedando en el nivel de la palabra para almacenar ese número entero.

Y paro con las entradas y el foco que hay que prender cuando se active la tarjeta de salida. Vemos que tenemos el botón de arranque que es un tipo de entrada que estará conectado a la tarjeta 0 del pin 0 por lo tanto su direccionamiento tendrá que ser I 1 : 0 / 0. Mientras que el botón de paro que también es una entrada está también en la misma tarjeta pero este estará conectado en el pin 1, su direccionamiento sería I 1 : 0 / 1, a diferencia del foco que esta conectado en una salida y ya es una salida de la tarjeta 1 pero en el pin 0 de esa tarjeta y su direccionamiento sería O 0 : 1 / 0. Si nos damos cuentas este número lo tenemos en el diagrama de escalera.

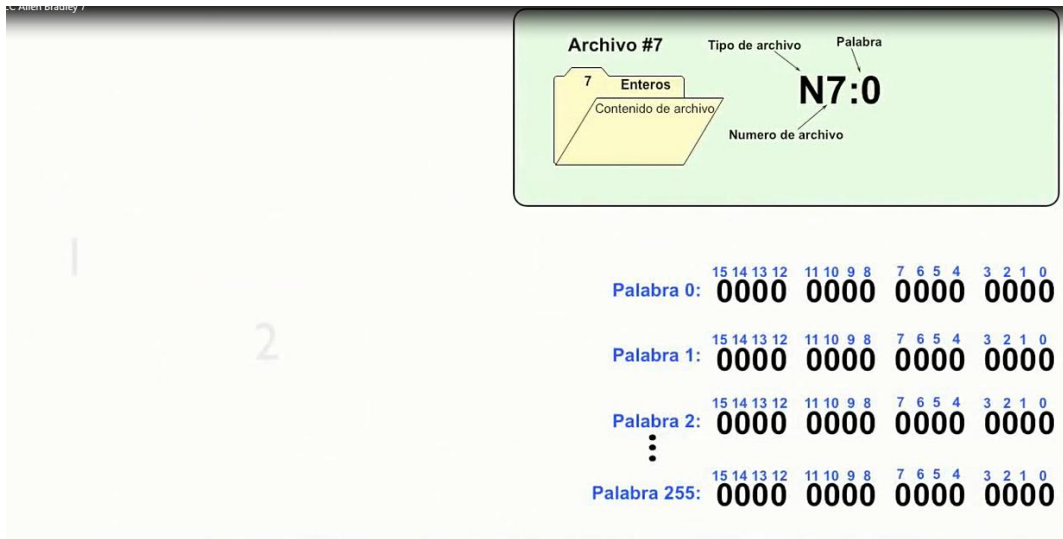


Fig 4.29 Archivo de tipo numérico para diferentes palabras.

Ahora vemos un diagrama de escalera un poco mas complejo donde lo único que hace es cuando presionamos el botón de inicio se enciende una torreta y va a mantenerse durante 10 segundos y luego se va a apagar. Vamos a ver como lo hacemos, primero en el primer renglón estamos evaluando que el timer haya terminado el timer 0 que es T 4 : 0. DN el número de timer, punto ON para utilizar el número del bit de control cuando haya terminado.

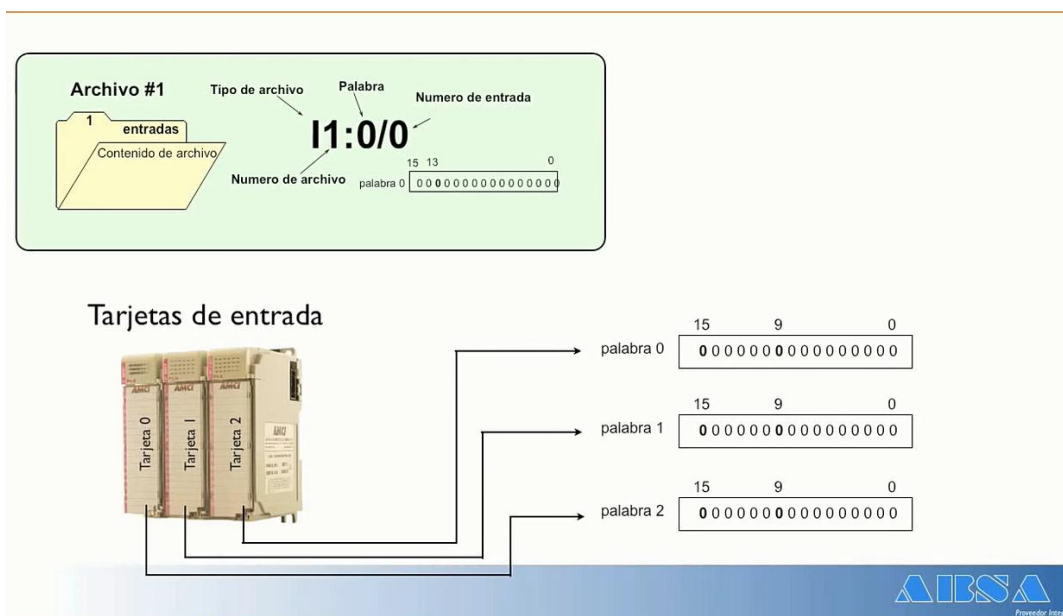


Fig 4.30 Representación de cada palabra en cada tarjeta del PLC.

Después evaluamos la entrada que está conectada en la entrada I : 0 / 0 indicando que está conectado, y evaluamos el resultado que en este caso no lo vamos a poner como enclavar una señal física, simplemente estamos guardando el resultado de una evaluación de este renglón en un espacio de memoria que hemos denominado en el nivel B 3 : 0 / 1 quiere decir en el archivo de bit de la palabra 0 en el pin de bit 1.

Después utilizaremos esa señal para evaluar ese enclavamiento, de tal manera que podemos decir que el renglón 1 es utilizado para enclavar el inicio del proceso que se va a utilizar en los demás renglones. Por ejemplo cuando se evalúa si B 3 : 0 / 1 está encendido entonces el timer se va a temporizar por lo que podemos darnos cuenta que el segundo renglón se encarga del temporizador. Y en el tercer renglón lo que vamos a estar evaluando es cuando ese timer termine, si vemos es el T 4 : 0. DN la información del bit 0 su bit de control DN, cuando este se vaya de falso a verdadero, este detecte el flanco de subida, entonces el contador incrementará su cuenta en 1 y podremos decir entonces que el renglón 3, su objetivo es incrementar la cuenta una vez que el timer haya terminado.

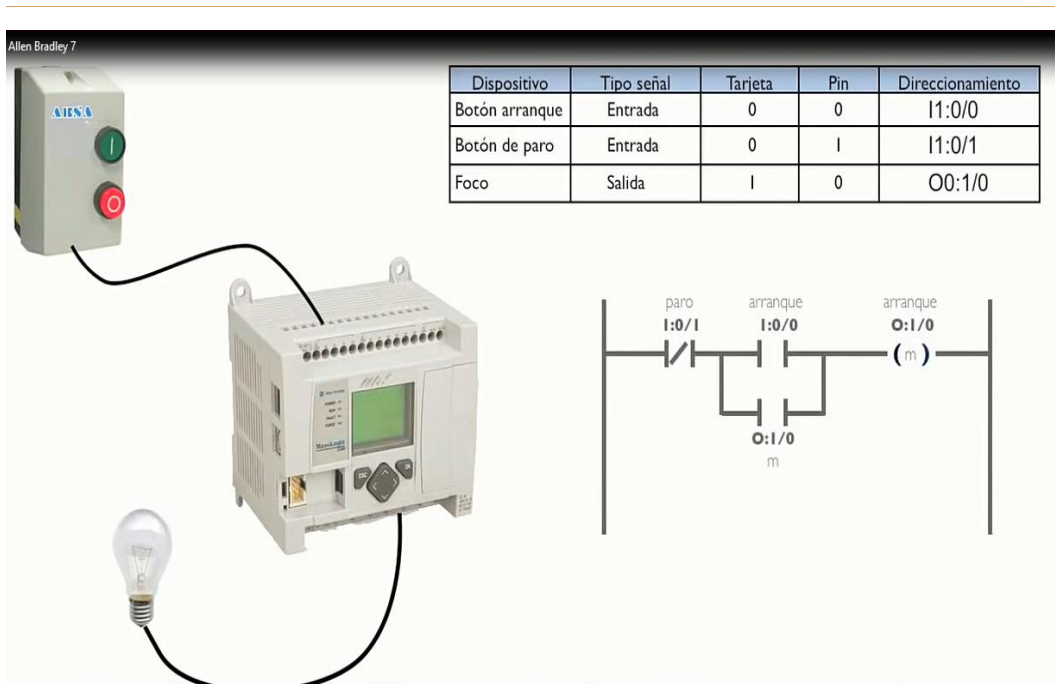


Fig 4.31 Conexión de los elementos del ejemplo de paro y arranque al PLC.

En el renglón 4 nos damos cuenta que lo que estamos haciendo es evaluar cuando el timer 0 está temporizando, estamos utilizando el bit de control TT de T 4 : 0. TT para energizar la torreta que está conectada a una salida física que la hemos puesto en la tarjeta 1, por ello la nomenclatura del O: 1 / 0 que es la tarjeta 1 ocupando el pin de bit 0. Con ello decimos que el renglón 4 tiene como objetivo encender la torreta siempre y cuando esté temporizado y con ello regresamos al renglón 1 y cuando se ha dado el tiempo dado que este es un interruptor normalmente cerrado, al cerrarse al terminar el tiempo se abre y por lo tanto B 3 se va a falso, todo vuelve a su estado inicial.

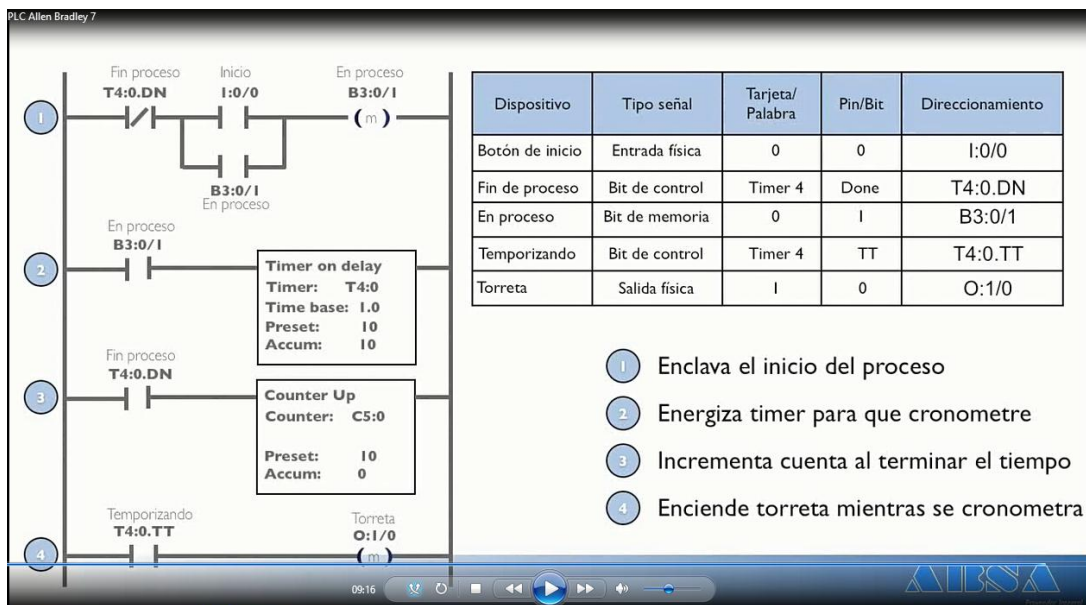


Fig 4.32 Diagrama de escalera de un timer y un contador.

4.6 SÍMBOLOS DE RS LOGIX 500

Los símbolos en RS Logic500 y sus usos, como crear un diagrama de escalera que conforme este va creciendo se va haciendo mas complejo e incluirá mas elementos, conforme esto vaya creciendo vamos a empezar a cual es el botón de inicio, que era el B 3 : 0 / 2, para que era el timer 3, la salida 3 de la tarjeta 4 que es, empezamos a dudar porque si tuviéramos un sistema con 4 ó 5 tarjetas ya tenemos 256 señales de campo y andarnos acordando puede ser mas complicado.

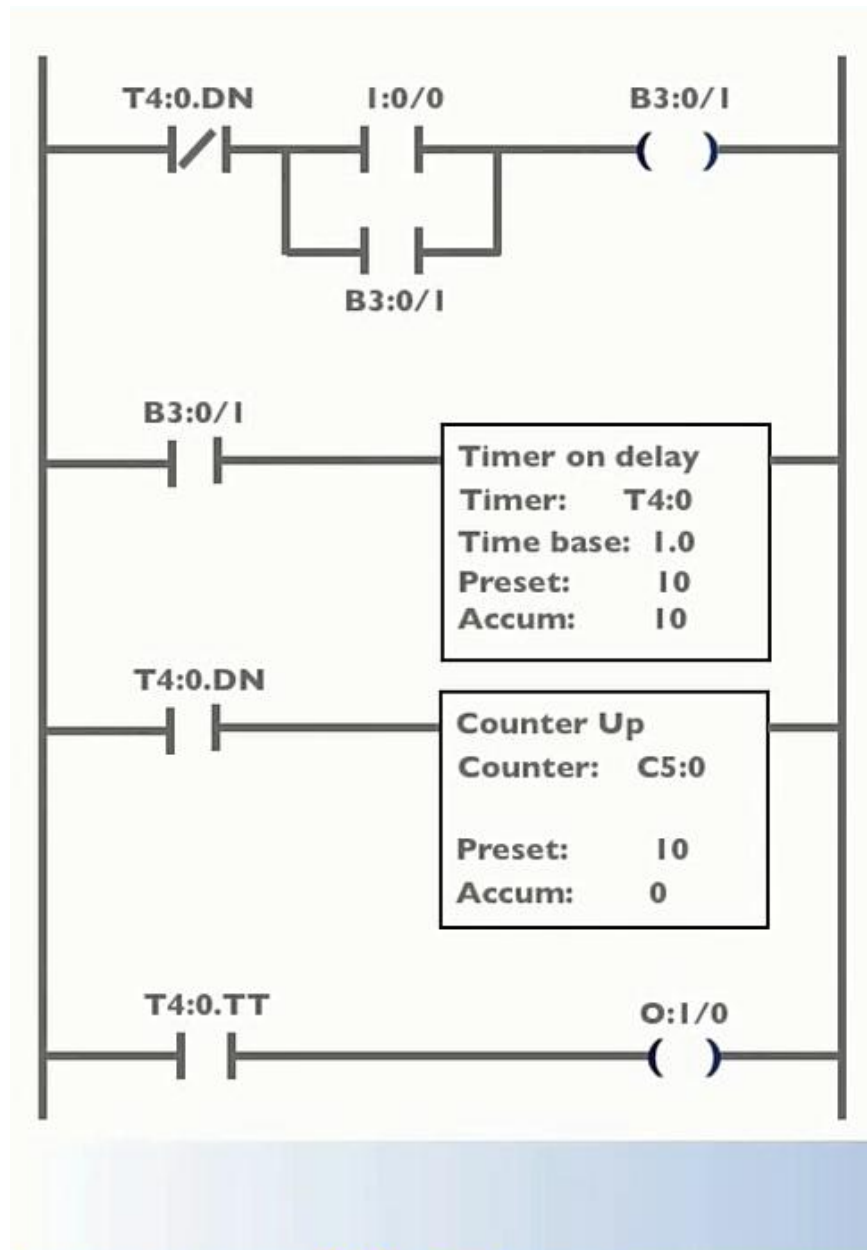


Fig 4.33

Podemos ir las apuntando en un cuaderno e ir haciendo una lista del número de entrada, salidas, parte o contador, timer, bobina y para que sirve, para esto RS Logix 500 posee el etiquetado de cada uno de los elementos del diagrama con una etiqueta que se llama símbolo, veamos como lo hacemos en RS Logic500.

Los símbolos son etiquetas asociados a los elementos del diagrama de escalera, por ejemplo en el renglón 0 vamos a colocar un contacto normalmente abierto que va a estar vinculado a la tarjeta de entrada 0 al pin 0, como podemos ver nos pone la dirección, sin embargo que pasaría si necesitamos utilizar esta señal durante muchas partes del programa, si nosotros damos clic derecho y seleccionamos edit symbol y ponemos por ejemplo botón de inicio. Vamos a dar cuenta como la dirección la mantiene y agrega una etiqueta en verde para indicar que ese botón o ese contacto es el botón de inicio.

Si ahora colocamos un botón normalmente cerrado y este lo vinculamos a la entrada 1 de la tarjeta 0, si le damos nuevamente un clic derecho y seleccionamos edit symbol, escribimos botón paro, podemos notar que ya tenemos el botón de paro y bien vemos que estamos haciendo un diagrama de escalera de botón de arranque y paro de motores. Ahora queremos conectar la bobina del renglón 0 a una salida en la tarjeta 0 en el pin 0, nuevamente le damos clic derecho en edit symbol y vamos a escribirle motor.

Nos falta poner el enclavamiento para lo cual vamos a poner un branch, aquí es donde viene algo interesante vamos a poner un contacto normalmente abierto, podríamos direccionarlo poniéndole O : 0 / 0, tal vez hemos avanzado mucho en un proyecto y no nos acordamos cuál era la dirección, pero es mas fácil que recuerde que este es el motor, y va a desplegar las opciones de las funciones que vengan con el contacto, de las opciones seleccionamos la que determina motor y automáticamente queda vinculado.

Esa es una de las grandes ventajas de usar los símbolos, que no hay que llevar o recordar en un momento las entradas, las salidas a que pertenecen, simplemente es mas fácil recordar los elementos del proceso y a través de ellos estarlos utilizando.

Ahora donde almacenemos estos datos es en el árbol del proyecto, tenemos una carpeta llamada Database, debajo de ella existe un apartado o una base de datos llamada Adress / symbols, si le damos doble clic aparece una pantalla donde podemos ver el listado de contactos o funciones y el símbolo que se le asigna en la tarjeta 0 en el pin 0. Donde vemos que la entrada 0 / 0 tiene el botón de inicio, la entrada 0 / 1 tiene el botón de paro, si

decidimos cambiar el nombre de la etiqueta lo ponemos hacer desde aquí, poniéndole botón de stop por ejemplo, notamos como en cuanto modificamos en la ventana que aparece también se modifica en el diagrama de escalera el nombre de la etiqueta el símbolo, si yo quiero modificar algo directamente se va a ver afectado en todo el programa.

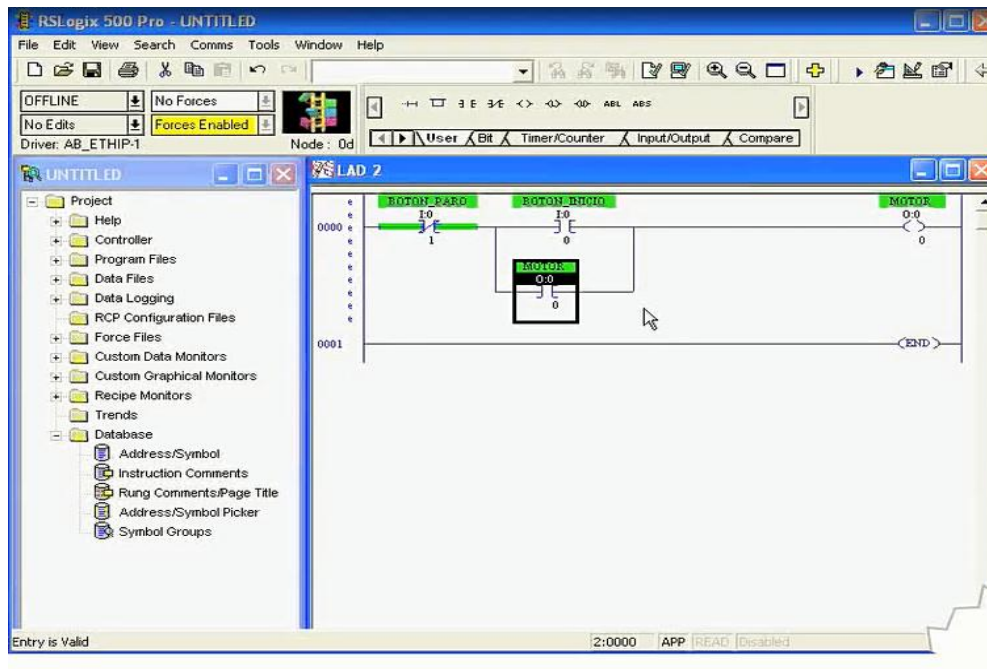


Fig 4.34 diagrama de paro y arranque en RSLogix 500 ventana de programación.

Hemos llegado a esta pantalla nuevamente con el árbol del proyecto en apartado Database, en la opción despegada Ardes/symbols al darle doble clic nos muestra todos los símbolos y sus direcciones que tenemos.

4.7 SECUENCIADOR

Vemos que es un secuenciador de cómo entenderlo y aplicarlo en un proyecto y en un diagrama de escalera, que ocurre si tenemos 12 leds que deben encenderse en un movimiento circular hacia la derecha, que es ir formando un reloj, seleccionamos un intervalo de 2 segundos entre cada transición, al terminar el ciclo deberá iniciarse. Ahí

vamos a ver como va avanzando ese led, es decir va conmutando un led hacia el otro cada dos segundos, es cuando nos preguntamos que debe haber un timer implícito en esta lógica, o también la opción de 12 contactos utilizándolos, la pregunta es como creemos que sería adecuado utilizar la lógica para poder hacerlo, como poder darnos cuenta del modo sencillo de hacerlo.

No es tan fácil como parece, sin embargo existe un secuenciador que lo que hace es tomar una etapa, un pedazo de una matriz o de una palabra y la deposita en una salida, pudiera ser una tarjeta de salida o bien una palabra de archivos de memoria y cuando al etapa 2 o siguiente punto se active toma la palabra 2 y la deposita en la misma salida y así sucesivamente con la palabra siguiente.

De tal manera que el secuenciador lo que hace es va tomando palabra por palabra, de una matriz o de un conjunto de palabras para depositar esa palabra o esos bits en una palabra destino que siempre será la misma palabra destino.

Como estamos usando un Micro Logix 1100 nosotros vamos ha hacer este ejemplo utilizando únicamente 6 leds, como ya sabemos el secuenciador va a tomar los datos de una matriz o de un conjunto de palabras y los va a colocar dependiendo la etapa será la palabra que envíe a una salida, que en nuestro caso queremos que esa salida sea directamente la palabra de salida de tarjeta.

Lo primero que vamos ha hacer es crear esa matriz para lo cual vamos a entrar al archivo de datos B3 que se puede encontrar en el proyecto directamente en Datafiles tenemos el B3 que como ya sabemos en ocasiones anteriores en un tipo de archivo binario, si hacemos un zoom para que se vea bien que estamos haciendo, podemos observar como tenemos todas las palabras del archivo B3 y como sabemos que debemos utilizar 6 palabras para tener diferentes etapas.

Cada etapa va a corresponder a un led encendido vamos a poner la configuración de esas palabras como queremos que se manifieste la salida, para ello vamos a poner un 1 en el

primer bit en la palabra 1, nos podemos preguntar porque en la palabra 1, porque el secuenciador va iniciar siempre en la palabra 0, pero cuando termine su ejecución regresa a la palabra 1 en el siguiente ciclo, entonces es conveniente iniciar todo en la palabra 1. Y bien vamos a poner unos en los siguientes bits que queremos que se vayan energizando, si nos damos cuenta estamos haciendo un corrimiento del bit, con ello estamos configurando ya la matriz que vamos a estar utilizando para que el secuenciador vaya de etapa en etapa.

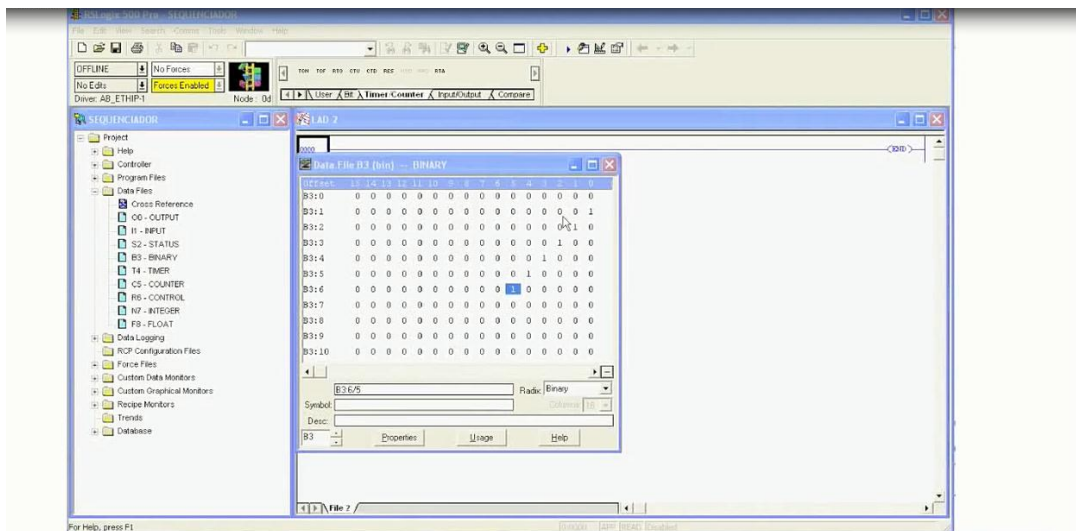


Fig 4.35 Ventana de RSLogix 500 para un secuenciador.

Vamos ahora a crear el diagrama de escalera que como ya sabemos debemos de crear primero una línea, sabemos que debemos utilizar un timer, para tener el control y sea cada 2 segundos el cambio.

Vamos a nombrarlo T 4 : 0, como es el primer timer vamos a ponerle timer 0, vamos a ponerle el tiempo base en 1 y un preset de 2 segundos para que se haga el cambio. Bueno ahora vamos a poner en esta misma línea un contacto normalmente cerrado es direccionar al mismo timer T 4 : 0 en DON, que es un timer ON Delay. Que estamos haciendo con esto, que cada vez que este timer llegue a 2 este bit se va a abrir se reset el time y vuelve a contar ya que estamos haciendo sea un ciclo infinito el timer llegue a 2 y se reset, llegue a 2 y se reste y así sucesivamente. Ahora necesitamos una nueva línea para poner el control del

secuenciador, y el secuenciador lo podemos sacar de la categoría file shift secuencer SQO y nos esta pidiendo un archivo que es la matriz de donde queremos que tome los datos que en nuestro caso lo pusimos en B 3 : 0.

Mas que nada es utilizada para filtrar los bits de la palabra que si queremos que se envíen, es decir si usamos la máscara podemos determinar cuales bits van a ser procesados y cuales no. El destino va a estar tomando un pedazo de esa matriz que es un apalabra para cada etapa y a donde la va a mandar, le ponemos directamente en la salida, es O : 0.0 normalmente cuando hablamos del bit ponemos diagonal 0 pero aquí hay una diferencia, sin embargo como aquí queremos que mande toda la palabra cuando especificamos la dirección como 0.0 le estamos diciendo al programa que va a ser toda la palabra.

Luego nos pide un archivo de control , estos archivos de control son los bits donde va a almacenar la información en que etapa va, que secuencia está haciendo, algún bit de si ya terminó o no terminó.

Es un archivo que le ayuda al secuenciador como auxiliar, que longitud queremos utilizar, nosotros tenemos 6 leds por lo tanto tenemos 6 etapas, pero como dijimos que utiliza de la 1 en adelante vamos a decirle que queremos 7 etapas, y la posición es un valor en que nos va a estar reportando en que etapa vamos, bien vamos a colocarle un contacto normalmente abierto al renglón 1 del mismo timer, pero ahora un TT, por efectos de tiempo ya hemos descargado y puesto en línea el PLC.

Vamos a abrir ahora el archivo de salidas para que podamos observar como la salida que es la palabra 0 en nuestra tarjeta del PLC, como está modificando su bit, como nos damos cuenta es 1 luego brinca al siguiente bit, al siguiente bit, con un orden de 2 segundos, incluso podemos escuchar de fondo los relevadores del PLC activándose. Que se activan cada 2 segundos conforme lo tenemos programado en el PLC. Como podemos darnos cuenta los secuenciadores son muy importantes al momento de desarrollar proyectos en el PLC.

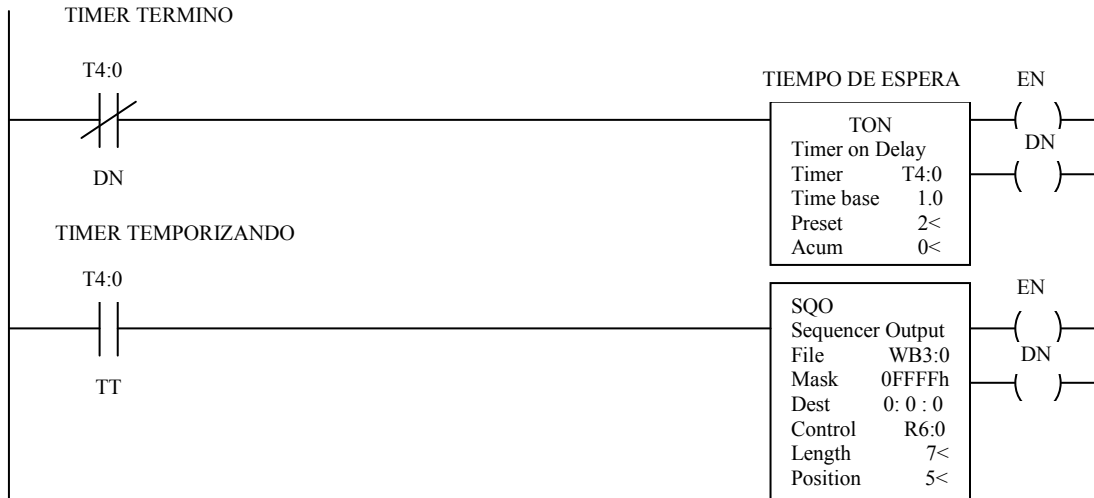


Fig 4.36 salidas Timer Término y Timer Temporizado.

Se tiene un Sistema de llenado de Botellas con tres productos diferentes designados como “A”, “B” y “C”, cada producto ésta almacenado en un recipiente por separado las botellas son transportadas sobre una Banda Ancha accionada por Motor el cual consta de sus Botones de “Arranque” y “Paro” Manual, El ciclo de llenado de cada botella inicia cuando una botella al ir avanzando es detectada por la fotocelda correspondiente, parando automáticamente el motor y activando la electroválvula “A” a la vez para descargar el volumen programado del producto “A”; cuando es completada la descarga del producto “A” se cierra su válvula de descarga, y al mismo tiempo activando la apertura de la electroválvula “B” para descargar el volumen de producto “B” una vez que sea descargado el volumen total requerido del producto “B” su válvula se cierra y ahora con esto se activa simultáneamente la electroválvula “C” permitiendo la descarga del volumen del producto “C” requerido.

Una vez que ha finalizado el ciclo de llenado de la botella en turno, se requiere se active automáticamente el motor de la banda transportadora y se detenga el motor cuando nuevamente la siguiente botella llegue a la fotocelda detectora de estas y comience un nuevo ciclo.

El sistema de llenado de botellas, tiene una fotocelda detectora de sobrenivel en el caso de presentarse ésta situación mandara activar la válvula de seguridad que impedirá el flujo de descarga de cualquier producto; la adaptación de la válvula de seguridad se lleva a cabo pensando en que por alguna causa alguna fallara el mecanismo de cierre de las electroválvulas de descarga, con lo cual se pudiera presentar la descarga de dos productos a la vez, originando con esto el sobrellenado de la botella en turno.

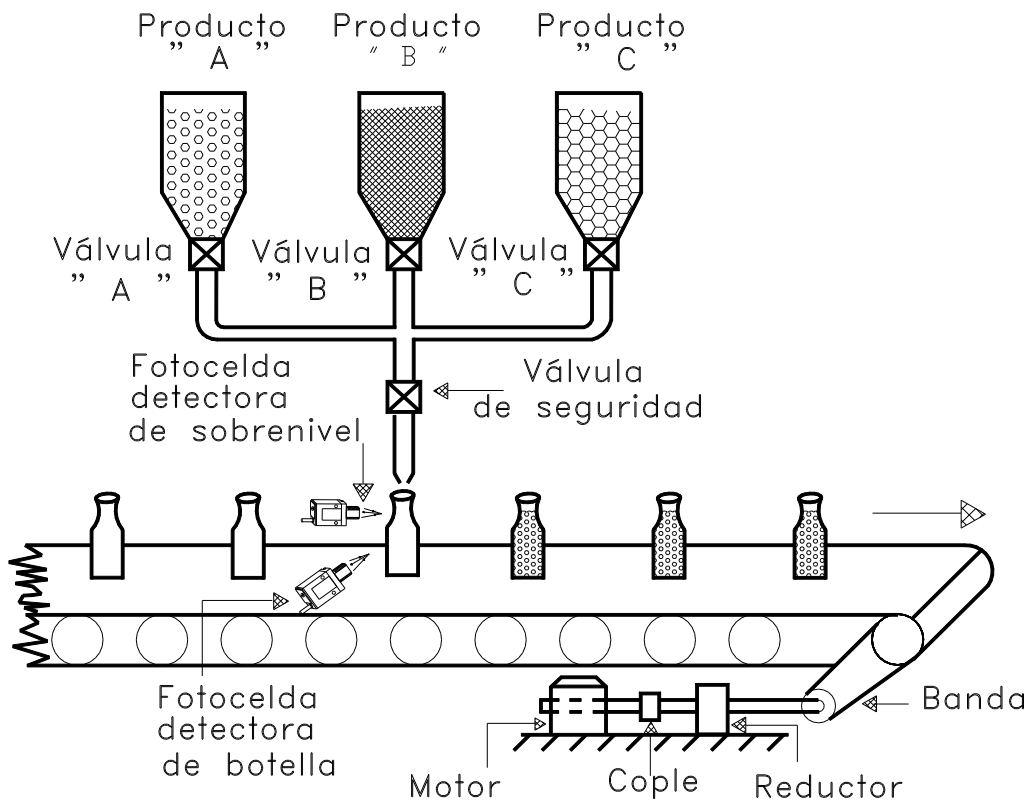


Fig. 4.37 Esquema de Llenado de Botellas.

La figura 4.39 representa el diagrama funcional del ciclo de llenado para cada botella, el volumen de cada uno de los productos requerido por botella ésta dado por una instrucción de tiempo, la descripción e interpretación del diagrama funcional es la siguiente:

Cuando se cumplen las condiciones previamente mencionadas para iniciar el ciclo; primeramente se activa la electroválvula "A" y el volumen en la botella empieza a subir

al cabo de un tiempo base “ T_1 ” elegido en nuestro caso en 4 seg. alcanza su volumen “ $4V_1$ ” por lo que se cierra la válvula “A” y se abre inmediatamente la válvula de “B” al transcurrir un tiempo de 2 seg., que es la mitad de “ T_1 ” alcanza la botella su volumen $2V_1$ establecido del producto “B” que es la mitad del volumen de “A”, cerrándose ahora “B” y abriéndose simultáneamente la válvula correspondiente del producto “C” y al transcurso de un tiempo de apertura de 1seg. Se cierra ésta última válvula pues alcanzo su volumen V_1 e iniciándose un nuevo ciclo.

El tiempo de apertura de cada válvula es proporcional al volumen respectivo requerido de los tres diferentes productos por las botellas, considerando que las electroválvulas y los depósitos son iguales entre si.

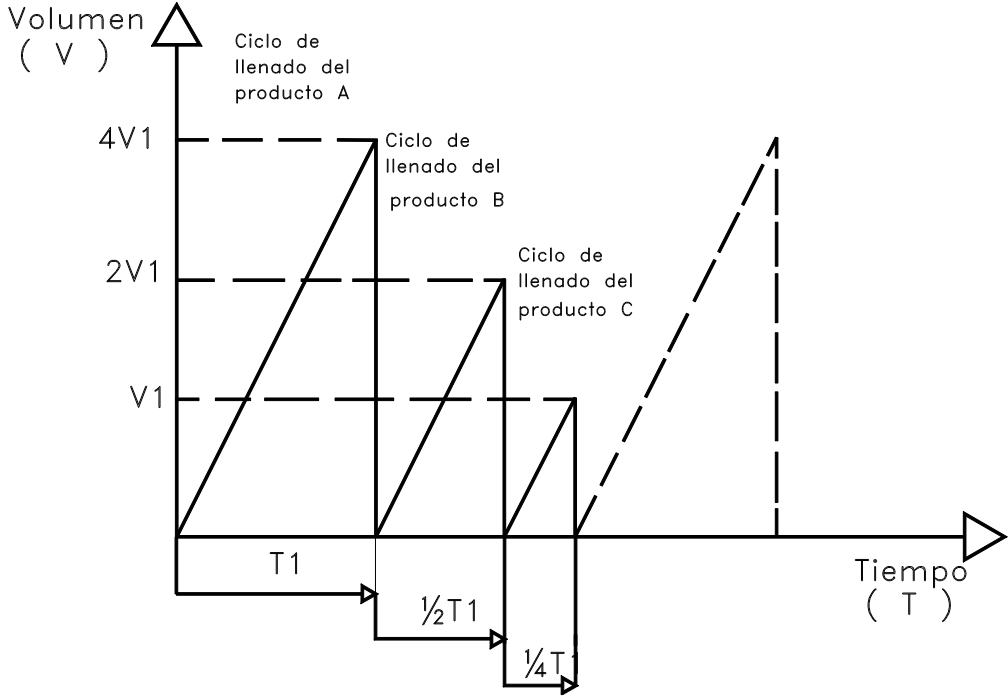


Fig. 4.38 Diagrama Funcional del Llenado de Botellas.

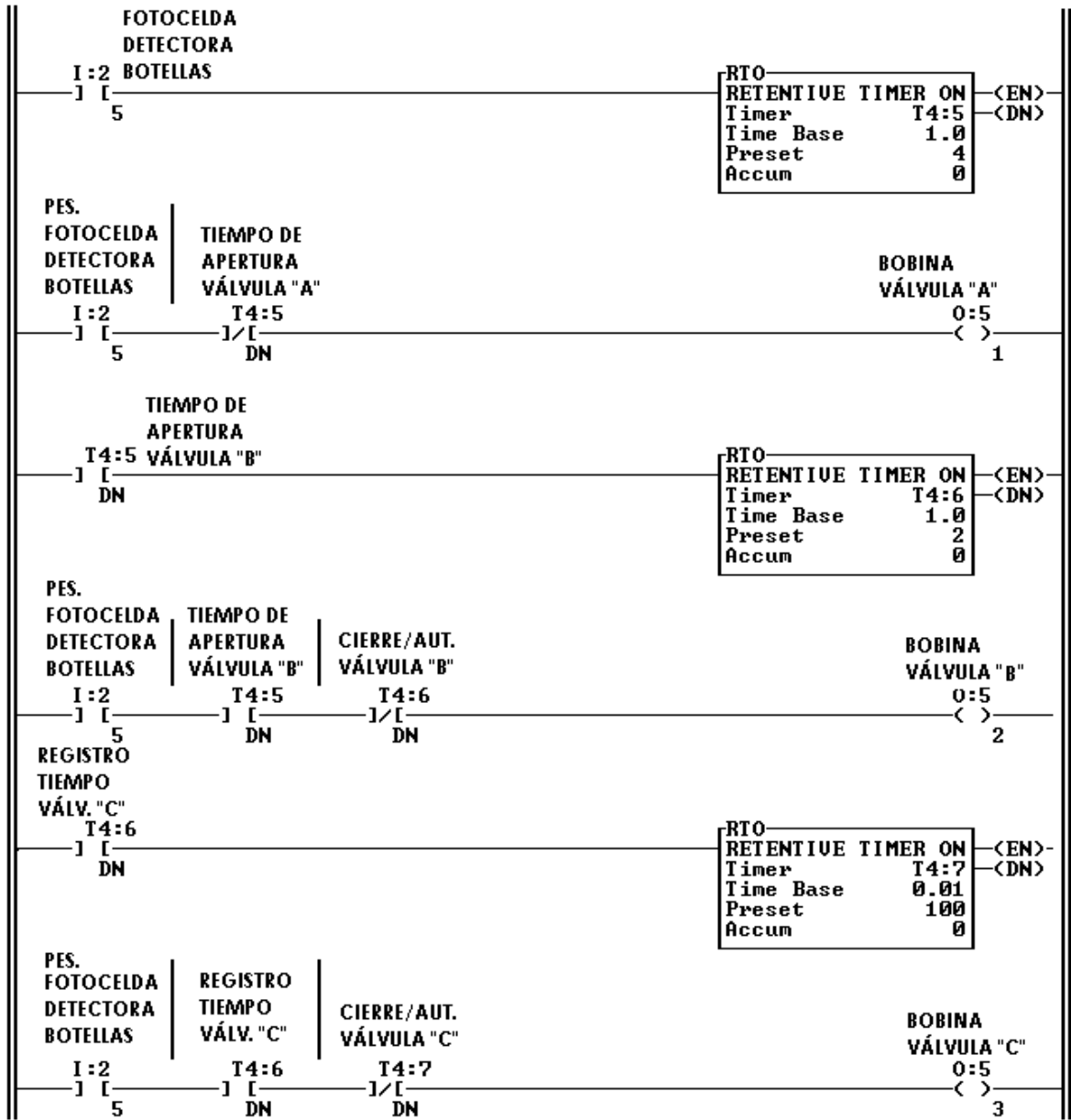


Fig. 4.39 Programa de Llenado de Botellas Parte I.

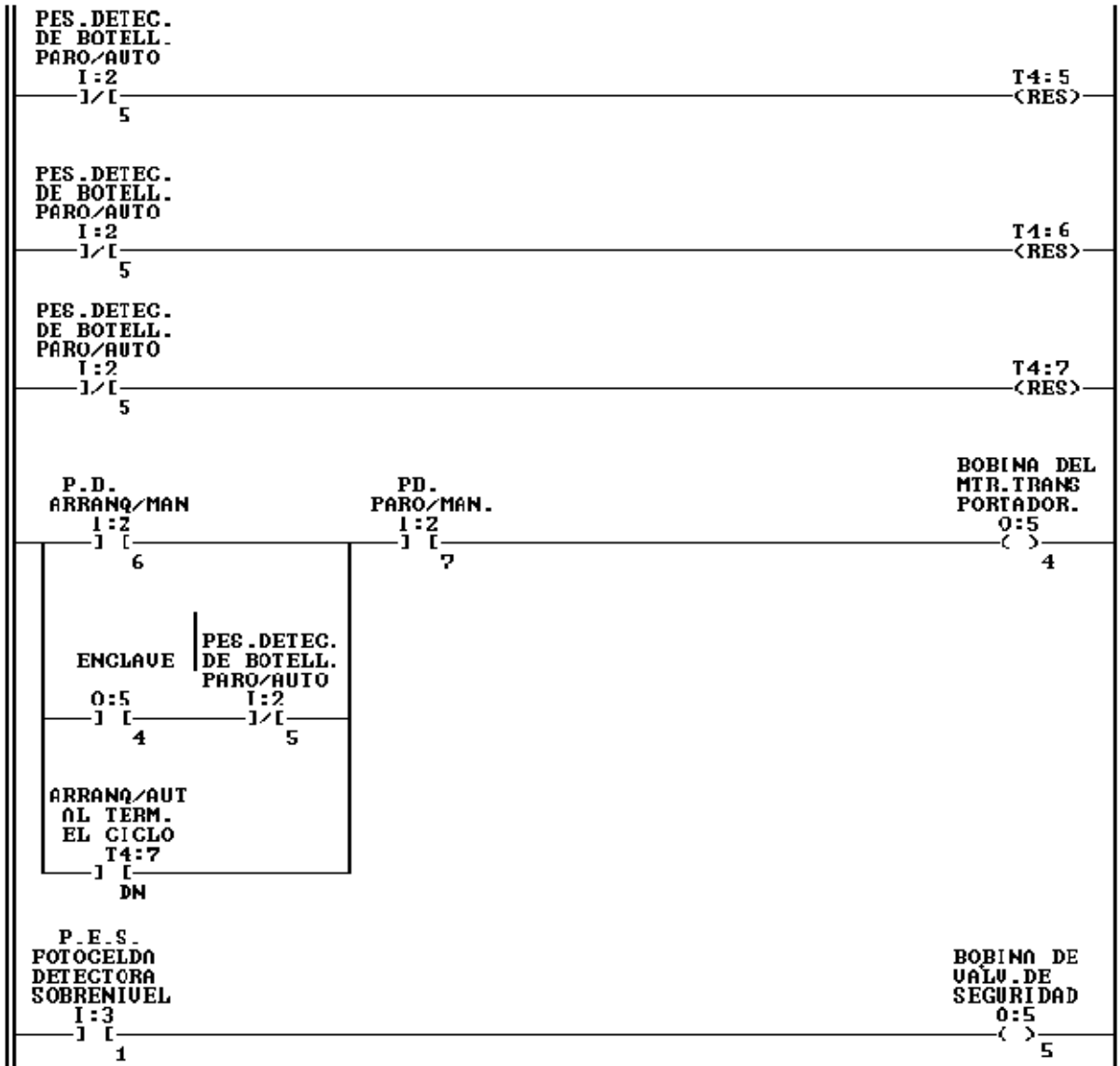


Fig. 4.40 Parte II del Ciclo de Llenado de Botellas.

Algunos controladores utilizan en sus instrucciones de tiempo “TIMERS” y de contar “COUNTERS” varios Bits de control de comportamiento especial, para efectuar el control de los dispositivos de salida, estos Bits en los programas llevan una etiqueta numérica; la familia de controladores PLC-2 y PLC-3 de Allen-Bradley en sus distintas versiones de procesadores maneja ésta situación.

5 FUNCIONES AVANZADAS

Los PLC's son una herramienta vital en el control de procesos de casi toda industria, la razón por la cual son tan populares es debido a su capacidad de programabilidad y de comunicación, el conocer las técnicas avanzadas de programación y de comunicación de los PLC's nos ayudará a incrementar la productividad y eficacia. Debemos conocer los diferentes tipos de sistemas de comunicaciones, como programar las diferentes funciones matemáticas, como programar las funciones de lógica avanzada y como usar los comandos de programación encontrados en la mayoría de los PLC en funciones avanzadas de programación.

5.1 SISTEMAS DE COMUNICACIÓN

Los PLC's pueden comunicarse con sistemas periféricos tales como impresores, motores impulsores, LAP TOPS, programadores de mano, otros PLC's y sistemas configurados para comunicación por serie. Algunos PLC's tienen los puertos de serie y los puertos de RS 232 incluidos en la porción de la unidad central procesadora del sistema modular, otros requieren de un módulo de comunicación adicional, el puerto periférico es otro RS 232 o un puerto de serie compatible usado para comunicaciones con un sistema externo. En algunos PLC's el puerto programador también puede ser usado para comunicarse con un sistema periférico secundario, hay esquemas de localización que permiten que varios sistemas de serie sean conectados en cadena con una ubicación distinta para cada sistema.

La comunicación con estos equipos es precedida con una dirección que hace que la comunicación con una unidad afecte únicamente al equipo con dicha dirección.

Al programar sistemas avanzados con un PLC puede ser necesario manipular datos matemáticamente. La mayoría de los PLC's pueden hacer funciones matemáticas tales como sumar, restar, multiplicación, división, multiplicación al cuadrado y raíz cuadrada.

Los números que encontramos al usar funciones matemáticas usualmente son enteros, aunque algunos sistemas son capaces de manejar matemáticas con fracciones. Los datos numéricos para las diferentes funciones matemáticas pueden ser almacenadas en registros de entrada, registros temporales, registros de datos o registros de salida dependiendo de la capacidad del PLC. Los datos numéricos también pueden provenir de entradas análogas tales como tacómetros ó transductor análogo, o entradas digitales como switches digitales o codificadores.

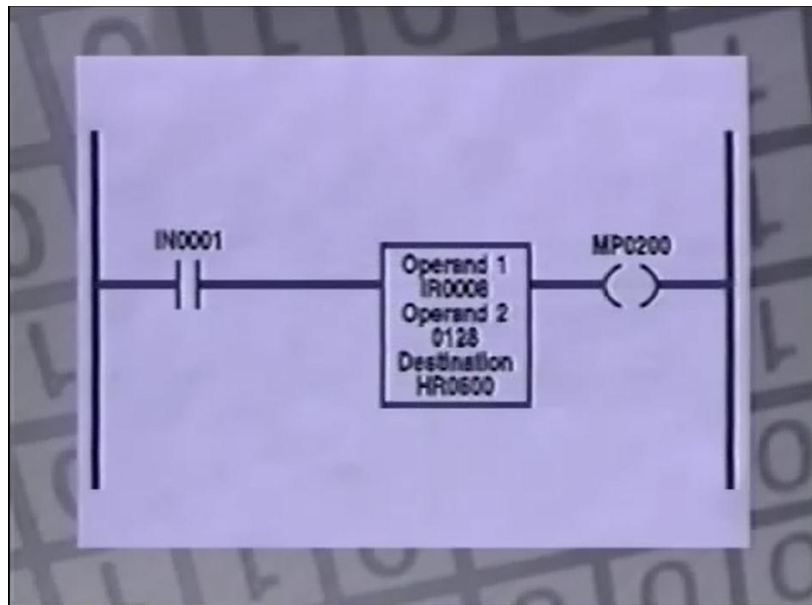


Fig 5.1 Operadores de entrada y destino habilitados por IN0001.

5.2 FUNCIONES DE COMPARACIÓN

Funciones de comparación del RS Logix 500, en muchas ocasiones dentro de la lógica de programación es necesario que comparemos dos elementos, si son iguales, si son diferentes, si uno es mayor que el otro, para ello podemos usar las funciones de comparación. Estas funciones son elementos del diagrama de escalera de entrada, es decir van a estar dentro del lado izquierdo que tienen ciertas características como que se deben utilizar 2 elementos. El elemento A puede ser una constante o bien una entrada o salida física, o bien un espacio de memoria, sin embargo el elemento B no puede ser direccionado a una salida o entrada física.

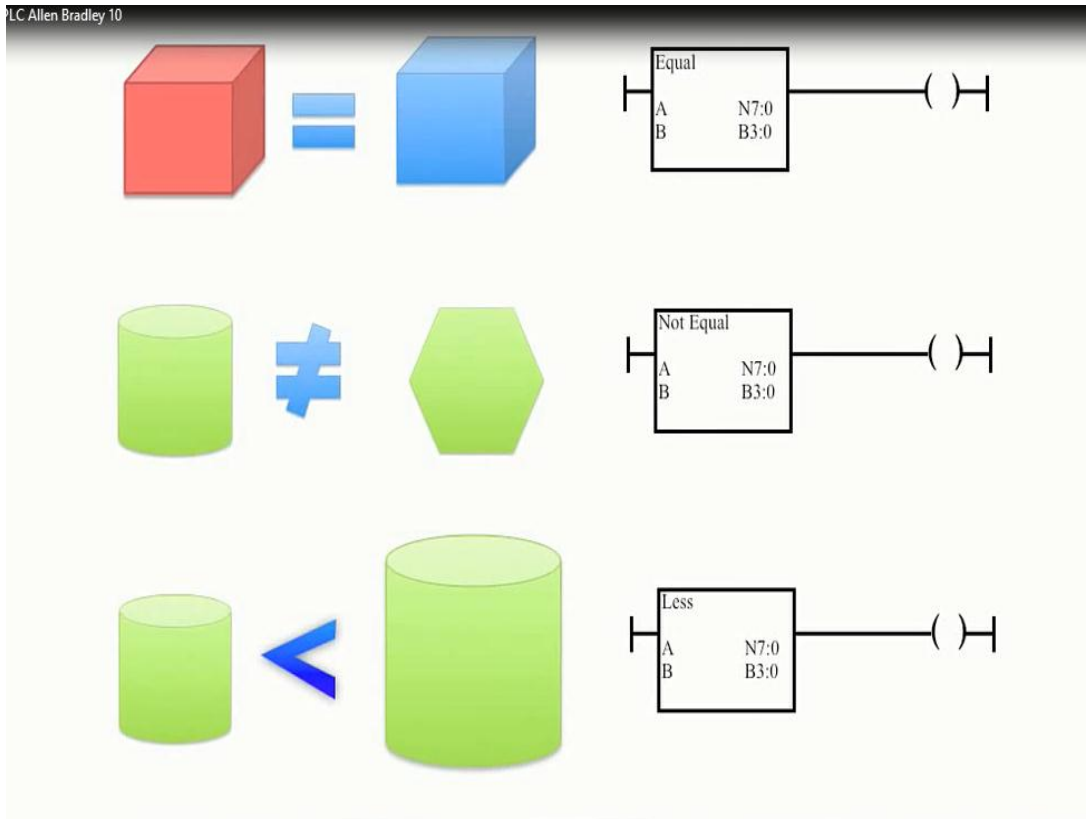


Fig 5.2 Funciones de comparación del PLC.

Vamos a ver como se utilizan, en este ejemplo vamos a ver como se utiliza la función de comparación mayor que $>$. Vemos que es lo que contiene el programa.

En el renglón 0 tenemos el botón 1 direccionado a la salida física 0 que acciona el contador 1, y en el renglón 1 tenemos otro botón también de una salida física que es la I: 0/1 con el contador 2 y ahora lo que queremos es que se energice una bobina siempre que el contador 1 sea mayor que el 2, vamos a direccionarlo a una salida O : 0/0, ahora vamos a colocar de la paleta de comparación la instrucción greater than, el origen A es decir que valor va a comparar $A > B$, bueno nosotros queremos que sea C 5 : 0. ACC, si recordamos en los contadores una de las 3 palabras que utilizan es el acumulador.

Luego nos pide el origen B donde queremos compararlo mayor que, y ahora vamos a decirle que sea el C 5 : 1. ACC nuevamente, y ahí ya le hemos dicho a nuestro comparador que queremos que sea del contador 0 del acumulado mayor que el contador, y cuando el acumulado del contador 1 sea mayor que el contador 2 entonces nuestra bobina se va a energizar. Al seguir programando ahora podemos observar que nuestra bobina de salida está apagada y vamos a ver ahora que sucede cuando energizamos el contador 1. Como podemos observar el acumulado del contador 1 es mayor que el acumulado del contador 2, automáticamente la evaluación de esta instrucción que es mayor que nos da un verdadero y la bobina se energiza.

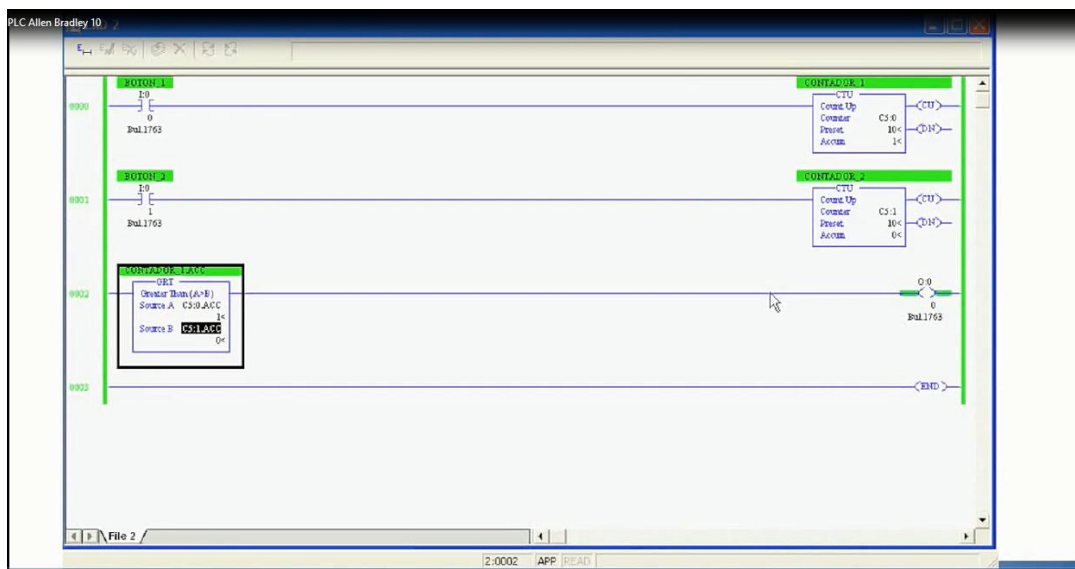


Fig 5.3 Diagrama de comparación en RSLogix 500.

Sin embargo si energizamos el contador 2 nuevamente ya no es mayor el 1, por lo tanto se apaga, es importante saber que las funciones de comparación pueden ser contenidas en una estructura de tipo serie, donde si por ejemplo $N 7 : 0 > B 3 : 0$ y $N 7$ diferente a un $B 3$ entonces el renglón será verdadero. Como podemos darnos cuenta al ponerlos en serie estamos armando una estructura de tipo AND o de tipo y. Vamos a utilizar este ejemplo anterior y ahora vamos a colocar una función NOT Equal de la paleta de comparación, el origen A lo vamos a llamar $C 5 : 0.ACC$, y el origen B vamos a ponerle un $B 7 : 0$, tenemos el archivo Data file del número 7 de enteros.

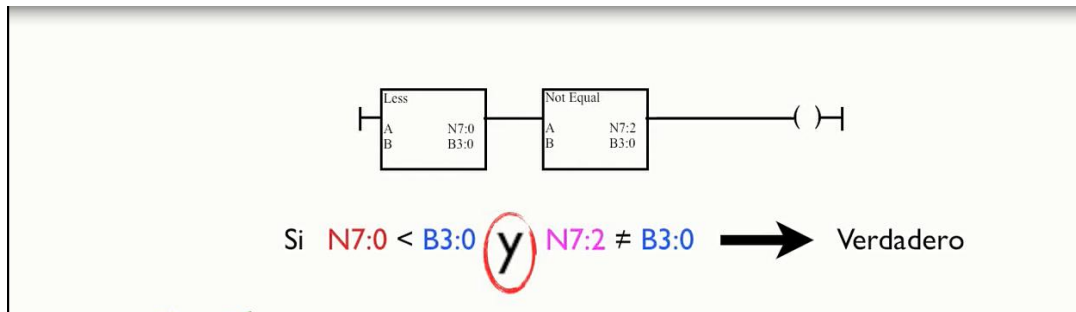


Fig 5.4 Comparadores en serie para un peldaño.

De la pantalla podemos observar ahora como N7 : 0 está con un valor como podemos ver dentro de la función, de tal manera que lo que queremos ver es que nuestra rutina va a estar energizada cuando el acumulado del contador 1 sea mayor que el 2 y el acumulado no sea 2.

De tal manera que si energizamos la primera del renglón 0 como el acumulador 1 es 1 y el acumulado del contador 2 es cero, por consiguiente se hace verdadero, si ahora volvemos a energizar el botón 1 nos damos cuenta que el acumulado 1 es mayor que el 2 sin embargo el acumulado 2 es igual a N 7 por consiguiente el renglón se hace falso.

También podemos obtener una lógica OR de tipo paralelo que de tal manera que podemos ir viendo que en una comparación de un N 7 menor que un B 3, ó un N 7 diferente de un B 3, entonces el renglón será verdadero. Como nos damos cuenta estamos usando la estructura O o de tipo OR.

Ahora vamos a crear una estructura en paralelo para lo cual vamos a colocar un branch o una nueva ramificación y en esta ramificación vamos a colocar una estructura de comparación, le vamos a colocar less than, aquí le vamos a decir que queremos que se compare un C 5 : 1.ACC contra por ejemplo una constante que le damos el valor de 4.

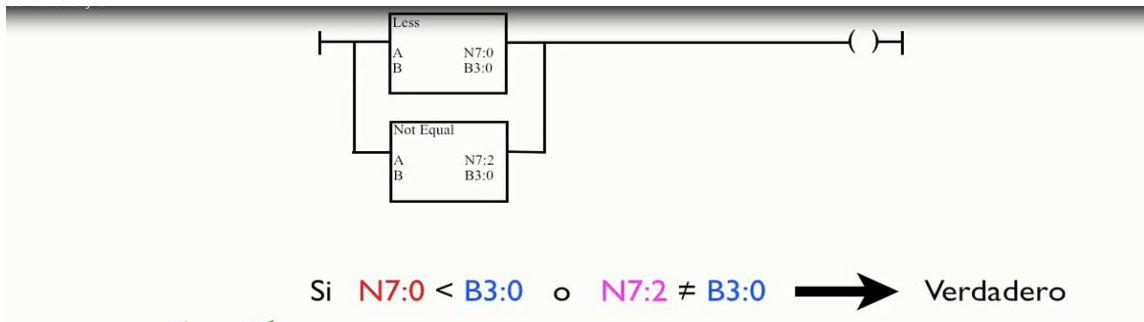


Fig 5.5 Comparadores en paralelo para un peldaño en 2 ramas.

Y como podemos observar ahora la bobina ya está energizada, dado que el acumulado del contador 1 es menor que 4, y como es una estructura en paralelo ya sea que se cumpla del renglón de arriba o del branch.

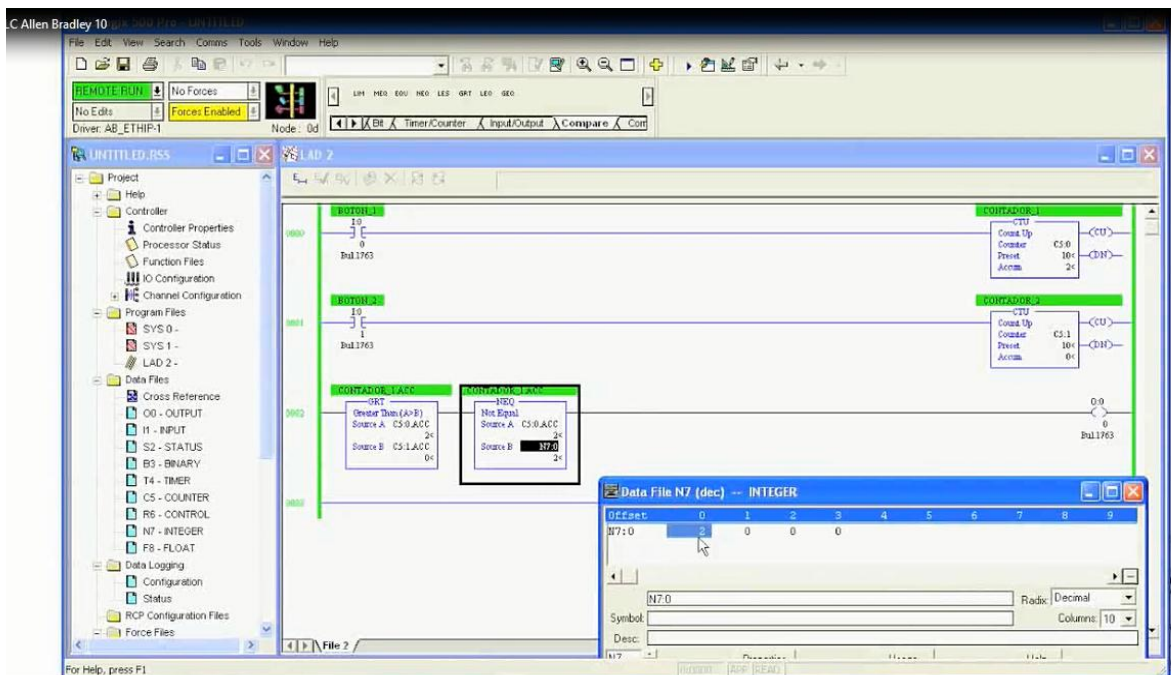


Fig 5.6 Diagrama en RSLogix 500 del comparadores en serie.

Entonces vamos a ver como funcionaría cuando empezamos a energizar, ahí podemos darnos cuenta que el renglón de arriba es falso, sin embargo el branch la ramificación es verdadera. Si energizamos varias veces el botón 2 podemos observar ahí que el renglón se hizo falso, dado que el acumulador del contador 1 ya no es menor o igual que 4 como lo teníamos en la función de less than, y tampoco el acumulado 0 es mayor que el acumulado de valor 1.

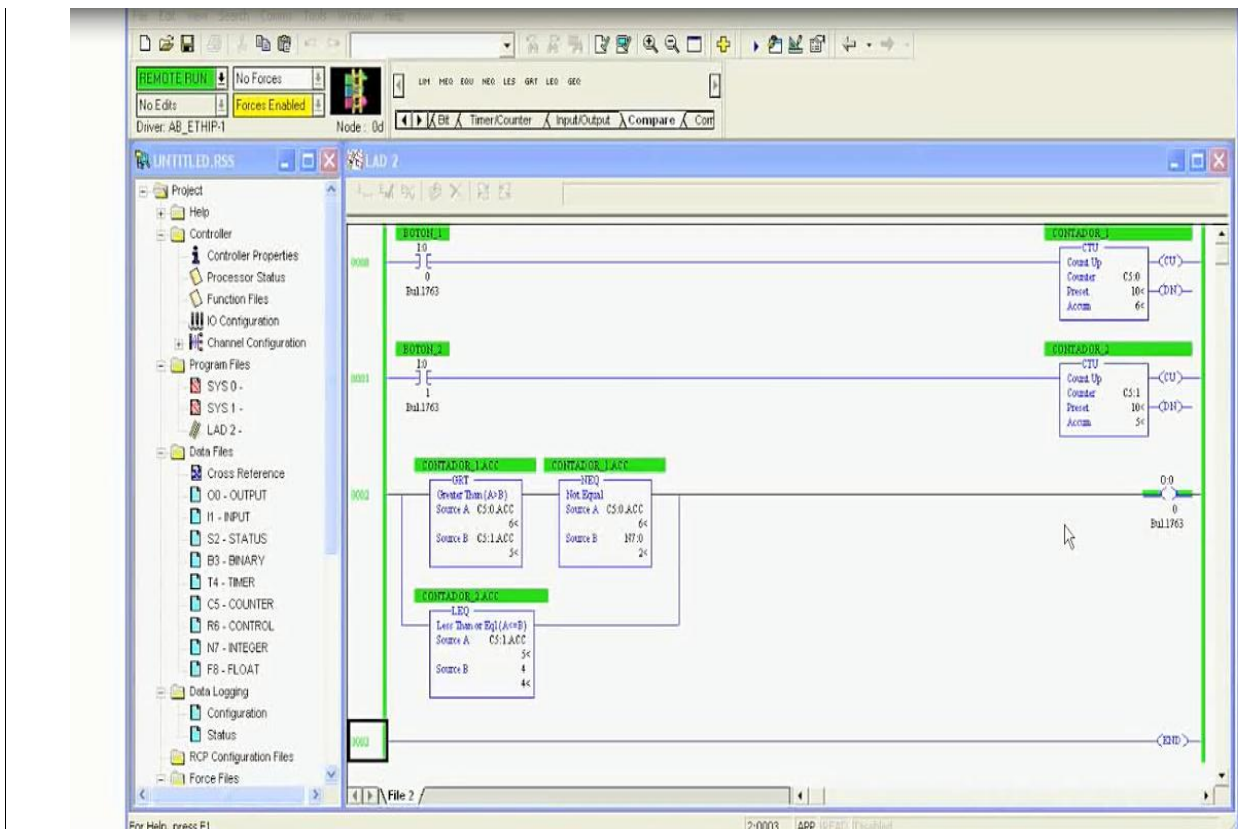


Fig 5.7 Diagrama en RSLogix 500 de 3 comparadores serie / paralelo.

Vamos a ver ahora que pasa si incrementamos el acumulador 1 como ya es mayor el renglón se hace verdadero, y como el acumulador 0 que es 6 es mayor y a la vez diferente a 2 el renglón se hace verdadero, con esto nos estamos dando cuenta que las funciones de comparación también las puede uno utilizar en una lógica ya sea en serie o en paralelo como instrucciones OR ó AND.

5.3 FUNCIONES MATEMÁTICAS

Las funciones matemáticas generalmente son implementadas durante el barrido del programa que la lógica de control cambia de falso a verdadero, esto es conocido como una función de transición, si el INPUT permanece apagado la función no va a efectuarse. Si el control lógico es verdadero la función matemática es efectuada y la respuesta se transfiere a la localidad especificada donde es mantenida hasta que se hace el cálculo de nuevo.

Este diagrama de escalera representa la función de sumar del PLC, el operador 1 es mantenido en IR011 en el registro de entrada, el operador 2 es mantenido en IR0023 registro de entrada y el resultado de la función matemática o destino es colocada en el registro de salida OR0001.

La bobina de salida para la función de sumar comienza con el prefijo ADD y es seguido por el número de relevo que debe ser energizado. La bobina de salida será energizada si la solución excede el número máximo permitido para este tipo de función y no será energizada si la lógica de control no es verdadera.

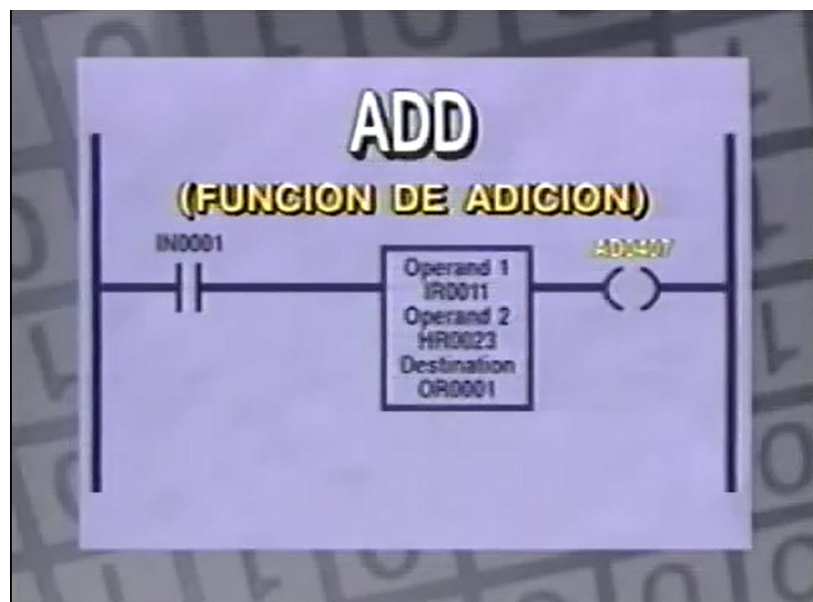


Fig 5.8 Función de suma del PLC.

La función de restar usa el mismo tipo de programación excepto que el prefijo de la bobina es SB, generalmente la bobina será energizada si el resultado de la resta es menor a cero, las funciones de programación de multiplicación y división son similares a las funciones de sumar y restar con algunas excepciones.

La función de multiplicación multiplica dos valores individuales de registro y da un resultado que es un valor de 2 registros. Considerar este diagrama de escalera de multiplicación, el primer operador es mantenido en el registro de almacenamiento IR0008, el operador segundo es la constante 0128, el resultado de esta función será mantenido en el registro HR0600 y registro HR0601, únicamente el primero de los registros consecutivos necesita recibir una localización y es el 0600.

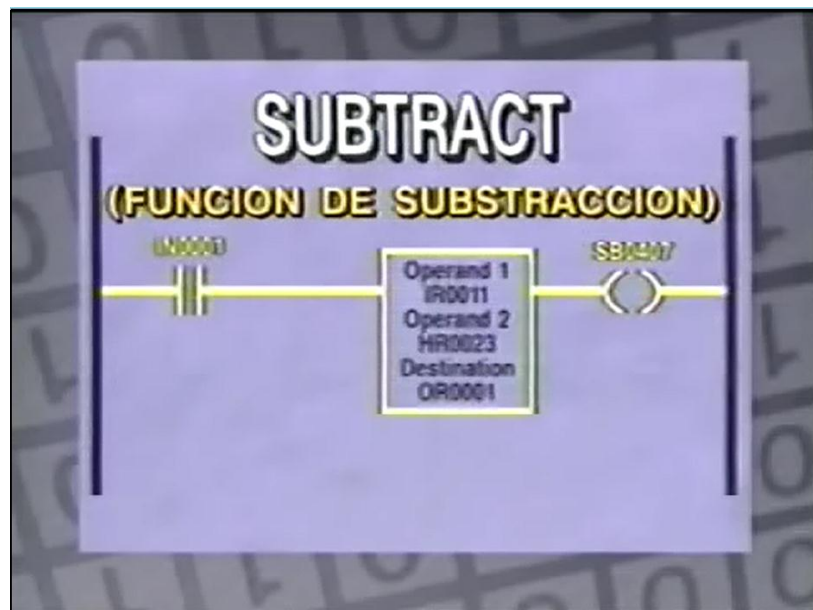


Fig 5.9 función de resta del PLC.

Notamos que se necesitan 2 registros para el resultado de la multiplicación, cuando se hace una multiplicación utilizando 2 valores de registro individuales el resultado puede ser tan grande que requiere dos registros con suficientes dígitos para representar el valor, consideremos el multiplicar dos valores digitales individuales como 2 y 2 el resultado es 4 un valor de un solo dígito.

Supongamos que dos valores individuales multiplicados son 9 y 3 el resultado es 27 el cual requiere del doble de capacidad de almacenamiento.

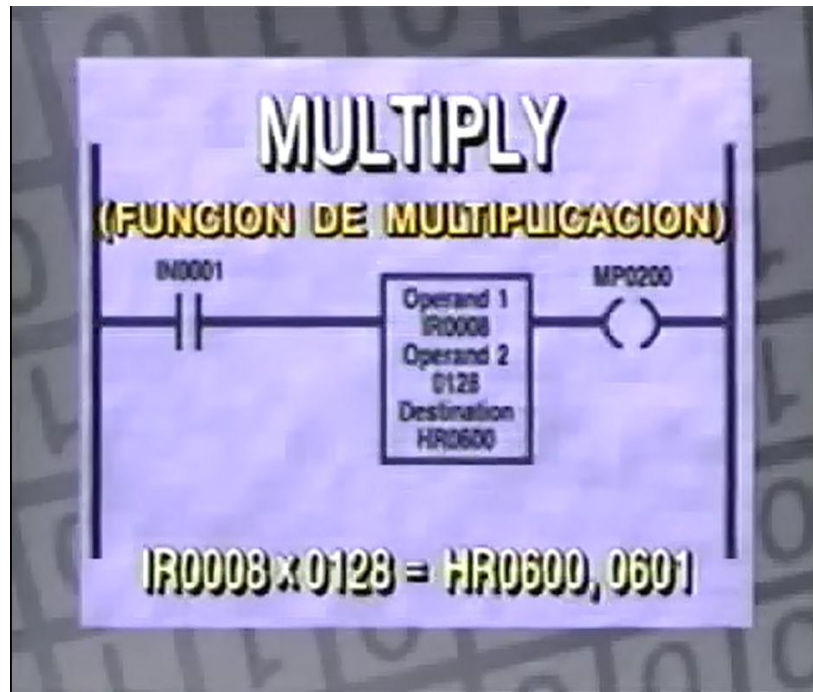


Fig 5.10 Función de multiplicación del PLC.

La función de división divide un valor de dos registros por un valor de registro de un valor y produce un resultado almacenado en 2 registros, con 1 registro siendo el número entero que resulta de la división y el segundo registro con el resto. Notamos que la bobina asociada con la función de división se indica como DV.

Supongamos que tenemos que dividir un número de 2 dígitos, que es 27 entre 7 por un número de un solo dígito, el número entero que resulta por esta división es 3, porque 7 cabe 3 veces en 27, el resto es 6, el 3 es almacenado en el primer registro HR0600 y el resto es almacenado en el registro segundo HR0601 de la división. La bobina DV0100 será energizada para la barrida en la cual la función es implementada.



Fig 5.11 función de división del PLC.

5.4 FUNCIONES AVANZADAS DE LÓGICA

Las funciones de lógica común al implementar algunas funciones de lógica avanzadas puede ayudar al programador a ejecutar su labor mas eficientemente. Las funciones típicas de lógica avanzada incluyen la función de “ ONE SHOT” o de un solo disparo, el flip flop R – S, el flip flop D y el flip flop T. Casi todo programa de PLC usa una de estas funciones avanzadas.

La función ONE SHOT o de un solo disparo es una función es una operación en una función de varios escalones que una bobina seleccionada sea energizada para una barrida, este tipo de bobina puede ser usada para controlar funciones que requieren una bobina para una sola barrida, consideremos este diagrama de escalera de un disparo que hará que CR1 se energice durante una barrida cada vez que IN 1 cambie de OFF a ONN, CR1 la bobina de un solo disparo será energizada por el contacto normalmente cerrado CR 2 controlado

por CR 2 y el contacto normalmente abierto controlado por IN 1. CR 1 será energizado con la primera barridas después de que IN 1 ha sido cerrado, ya que CR 2 será des energizado en este punto, IN 1 también hace que CR 2 se energice, cuando el segundo escalón es ejecutado.

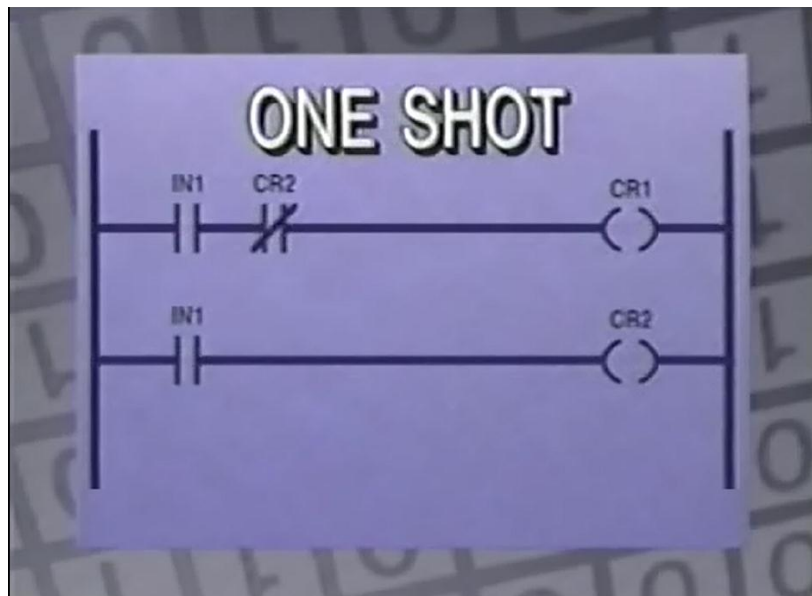


Fig 5.12 Diagrama de escalera para una bobina de un solo barrido.

Cuando la barrida comienza de nuevo en la parte superior de la escalera, el contacto normalmente cerrado CR 2 estará abierto, porque la bobina CR 2 está energizada, esto hará que CR 1 se desconecte. En el segundo escalón CR 2 permanece energizado si IN 1 está energizado, cuando IN 1 es apagado CR 2 se apaga y reajustará los 2 escalones para esperar al siguiente cerrada del IN 1.

El flip flop R – S está en sistemas que requieren un botón de arranque y parada momentáneo, la entrada R es el reset o botón de parada y la entrada S es el set o puesta de botón de arranque. Como podemos ver en el diagrama de escalera de un flip flop R – S, cuando el IN 1 es energizado momentáneamente la corriente continúa fluyendo hacia la bobina a través del contacto normalmente abierto CR 1 y el contacto normalmente cerrado IN 2, la bobina se mantendrá energizada hasta que el IN 2 es activado.

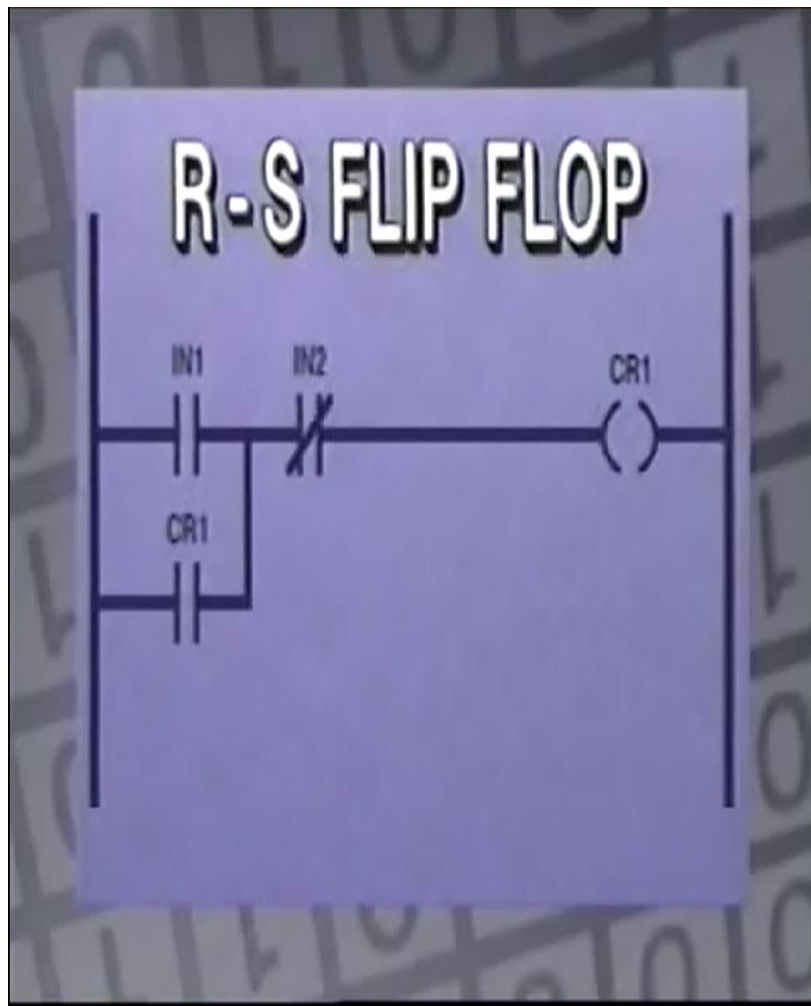


Fig 5.13 Diagrama de escalera nuestra como construir un flip flop R – S.

El flip flop D es usualmente utilizado en situaciones donde 2 acciones momentáneamente simultáneas se requieren para activar la salida, la función lógica requiere una entrada tipo D activada por el operador y una entrada activada externamente por un reloj o una función de monitorización. Como vemos en este diagrama la entrada IN 1 e IN 2 el gatillo deben ser activadas para activar la bobina CR 1, cuando esto sucede la bobina CR 1 permanece energizada hasta que IN 2 es activada de nuevo, una vez que CR 1 es energizado IN 1 no afecta la lógica del escalón. Si I 2 se activa de nuevo el escalón es reajustado.

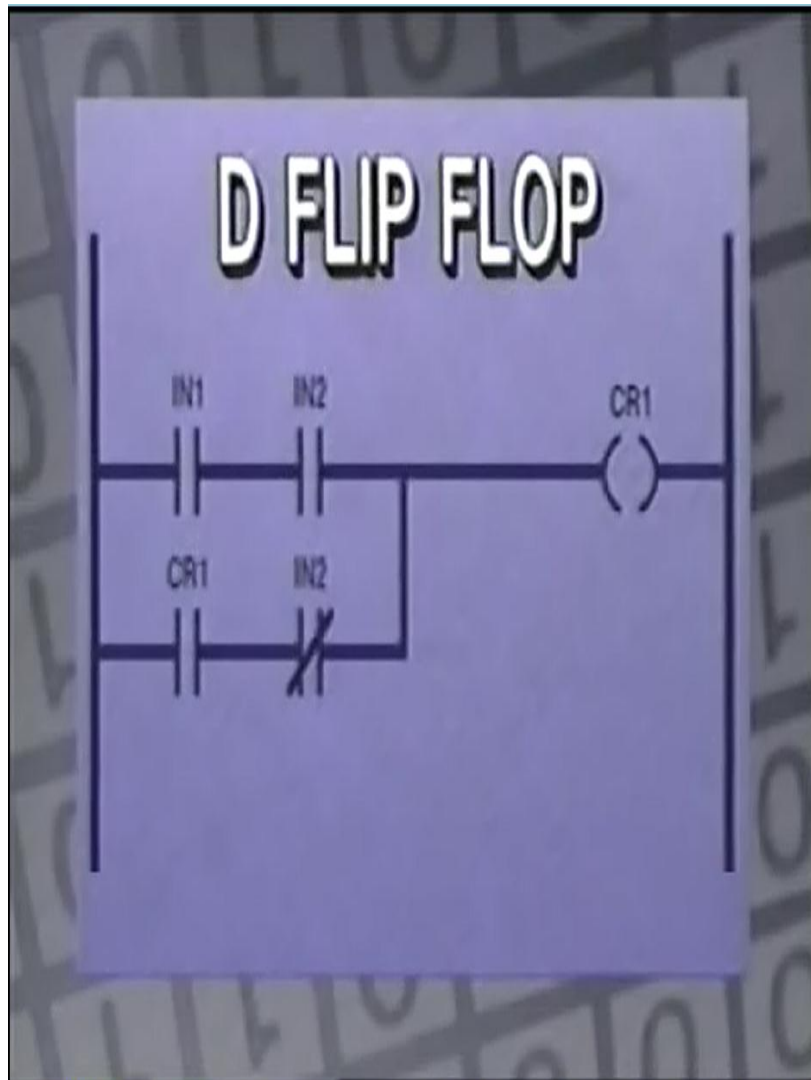


Fig 5.14 Diagrama de escalera nuestra como construir un flip flop D.

El flip flop T también consta de dos entradas, en este diagrama de dos escalones el primer escalón es un reloj controlador TIM 1 calibrado para 0.5 segundos y el siguiente escalón es el flip flop T, el primer escalón controla el gatillo y el segundo es la función del flip flop, al oprimir IN 1 el reloj controlador calibrado para 0.5 segundos comienza su conteo porque el contacto lógico será verdadero. En el primer escalón después de que el reloj llega a 0.5 segundos la bobina TIM 1 será energizada. TIM 1 se energiza por una barrida cada vez que el reloj expira, en este caso cada 0.5 segundos.

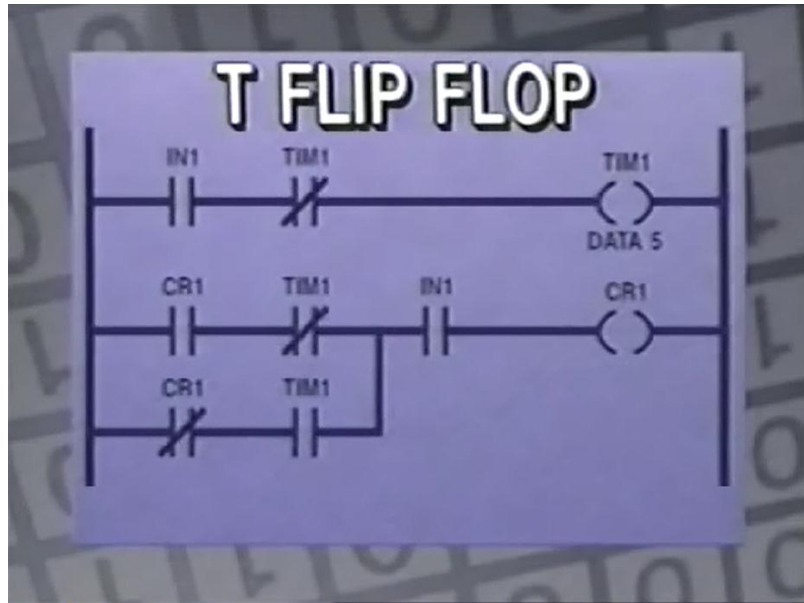


Fig 5.15 Diagrama de escalera nuestra como construir un flip flop T.

Vamos a asumir que la primera vez que la bobina TIM 1 se energiza con una barrida CR 1, está de energizado, en el segundo escalón la rama que contiene el CR 1 normalmente cerrado y el contacto normalmente abierto TIM 1 será verdadero y CR 1 será energizado, en la siguiente barrida ambas ramas de la lógica de contacto serán falsas, porque CR 1 será energizado en ese momento, el resultado es que CR 1 será de energizado.

5.5 FUNCIONES AVANZADAS DE PROGRAMACIÓN

Es conveniente entender algunas de las funciones de programación avanzada disponibles en muchos PLC's, las funciones avanzadas son Subrutina, JUMP o puentes, SKIP o saltos, control central tipo relevo MCR, programación de secuencia y desplazamiento de bits.

Una subrutina puede ser una línea o líneas múltiples de lógica que controlan una función específica o que efectúan un cálculo en algún proceso, se usa cuando ciertas condiciones requieren la salida de una subrutina. Se puede tener acceso a una subrutina a través del progreso convencional de escalera o puede ser alcanzado usando las funciones de puente, salto ó MCR.

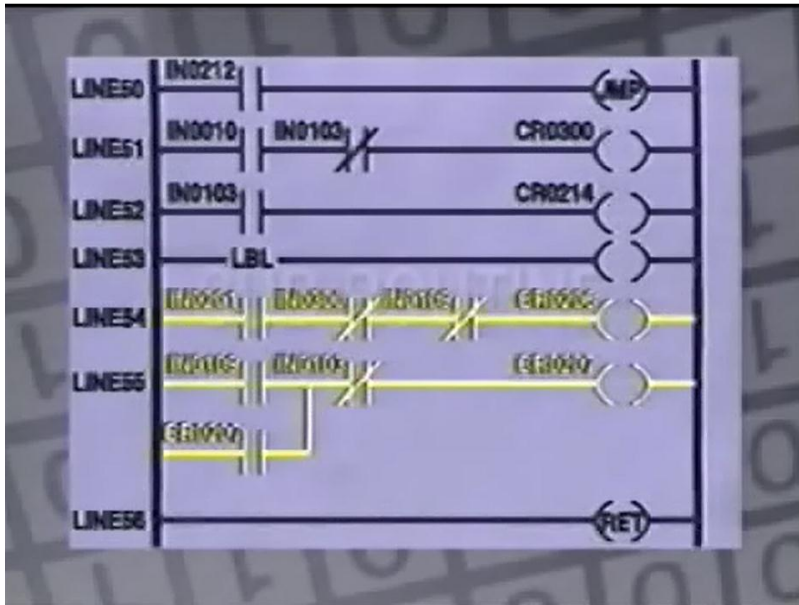


Fig 5.16 Diagrama de escalera que muestra funciones de salto de peldaños con LBL.

Las subrutinas se usan para ahorrar el tiempo del proceso del PLC ya que los escalones de subrutina son solucionados cuando son llamados y son saltados cuando no son llamados, en casos de que la subrutinas consten de muchos escalones se puede ahorrar una cantidad considerable de tiempo de barrida.

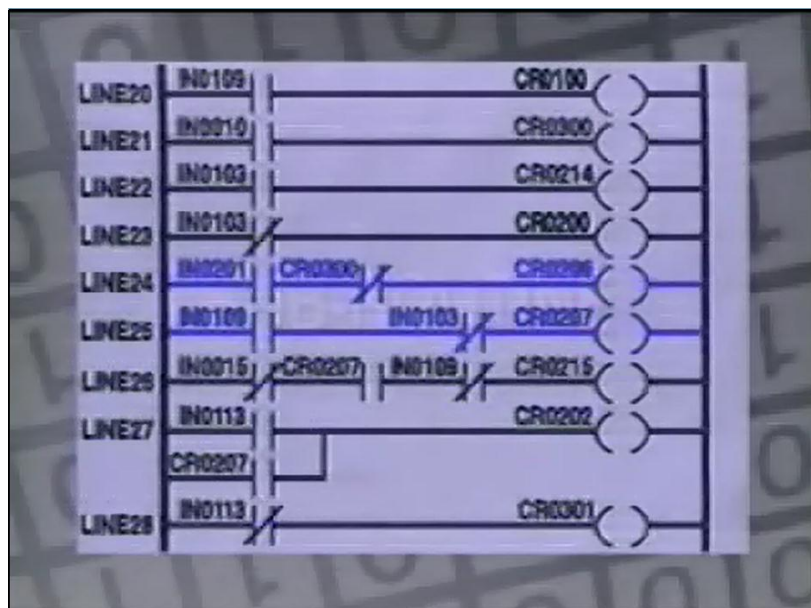


Fig 5.17 funciones de salto con subrutina de llamado con MCR.

La función de puente o JUMP se utiliza para saltar por encima de líneas en cualquier dirección, este diagrama de escalera demuestra la función de puente, sin IN0212 es energizado se activa la función de puente, las líneas que son saltadas permanecen en la condición que estaban durante la última barrida del programa. Todos los escalones son saltados hasta que el comando LBL es leído por el PLC el cual comienza a leer el programa de nuevo.

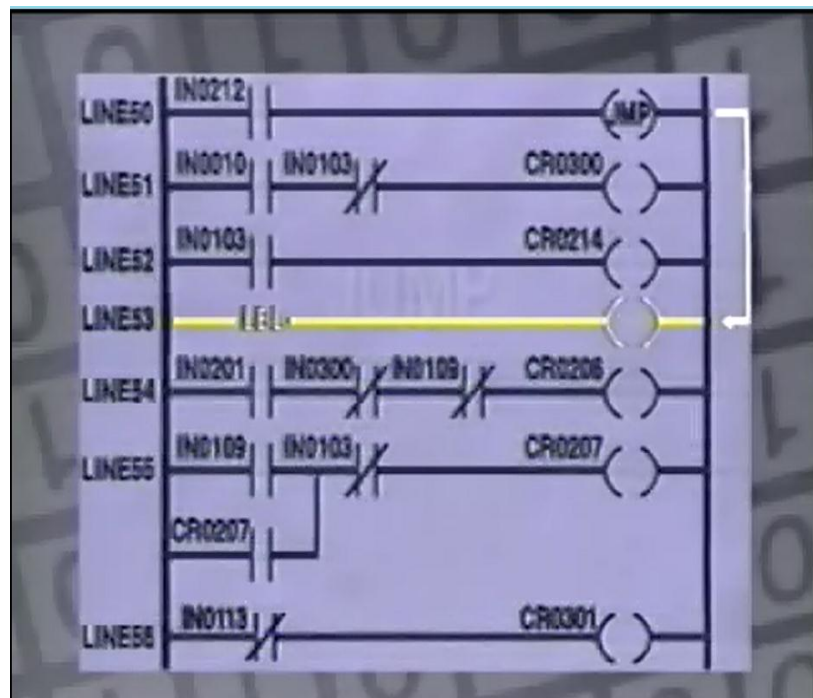


Fig 5.18 Función de salto programada con el comando LBL.

El comando de puente puede continuar con el eslabón LBL que recoja o puede ser programado para regresar a la función de puente.

Para regresar al escalón de puente se debe programar una bobina de retorno. Si reemplazamos la línea 56 por una bobina de retorno, el PLC sabrá que debe regresar a la línea inmediatamente después del comando de salto de puente.

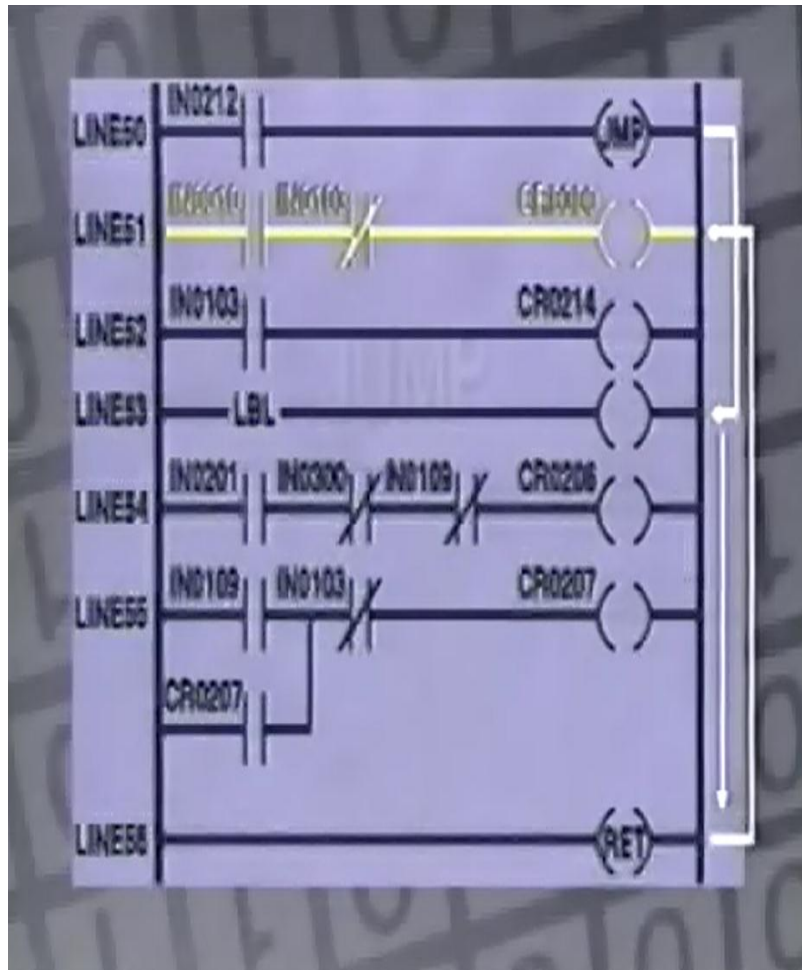


Fig 5.19 Diagrama de saltos en programación avanzada JMP, LBL y RET.

Cuando IN0212 es energizado el PLC saltará al escalón efectuando la subrutina y regresará a a línea después del comando de puente continuando por la escalera. Si no se activa el puente el programa será leído en la convención normal.

El comando de salto SKIP es similar al comando de puente excepto que especifica cuantos escalones de lógica debe saltar y no puede ir hacia atrás en el programa. Este diagrama demuestra la función de salto, cuando IN0109 es energizado la función de salto le dice al PLC que ignore los 4 escalones siguientes y que continúe con el siguiente escalón, las líneas que son saltadas se mantienen en la condición en que estaban durante la última barrida del programa.

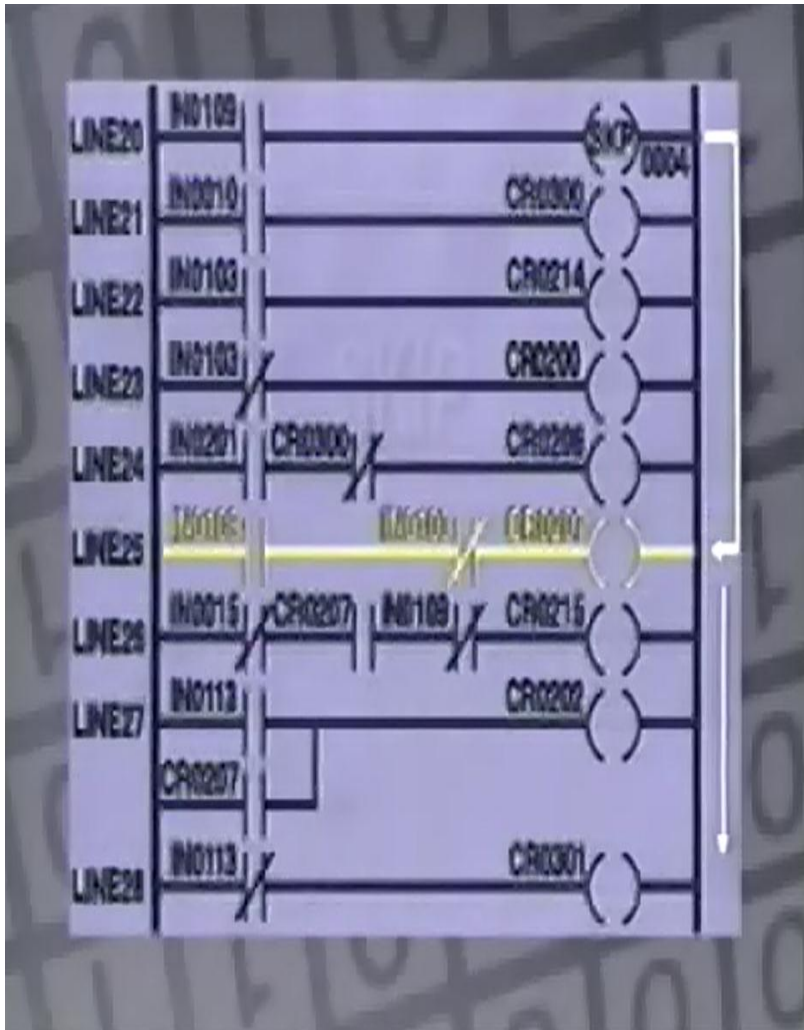


Fig 5.20 Saltos de peldaños con SKIP.

La función de master control relay MCR o de control central de tipo relevo es casi lo opuesto a la función de salto, este diagrama de escalera demuestra la función MCR, cuando IN0102 está desconectado la función MCR desconecta la cantidad de entradas especificada sin tomar en cuenta la condición de los equipos conectados a esas entradas. En este caso los 4 escalones siguientes estarán desconectados hasta que IN0102 es energizado, después de los 4 escalones los siguientes son leídos normalmente, cuando IN0102 está conectado el PLC lee la escalera normalmente, esta función no deja la condición de los escalones saltados como estaban en la barrida anterior.

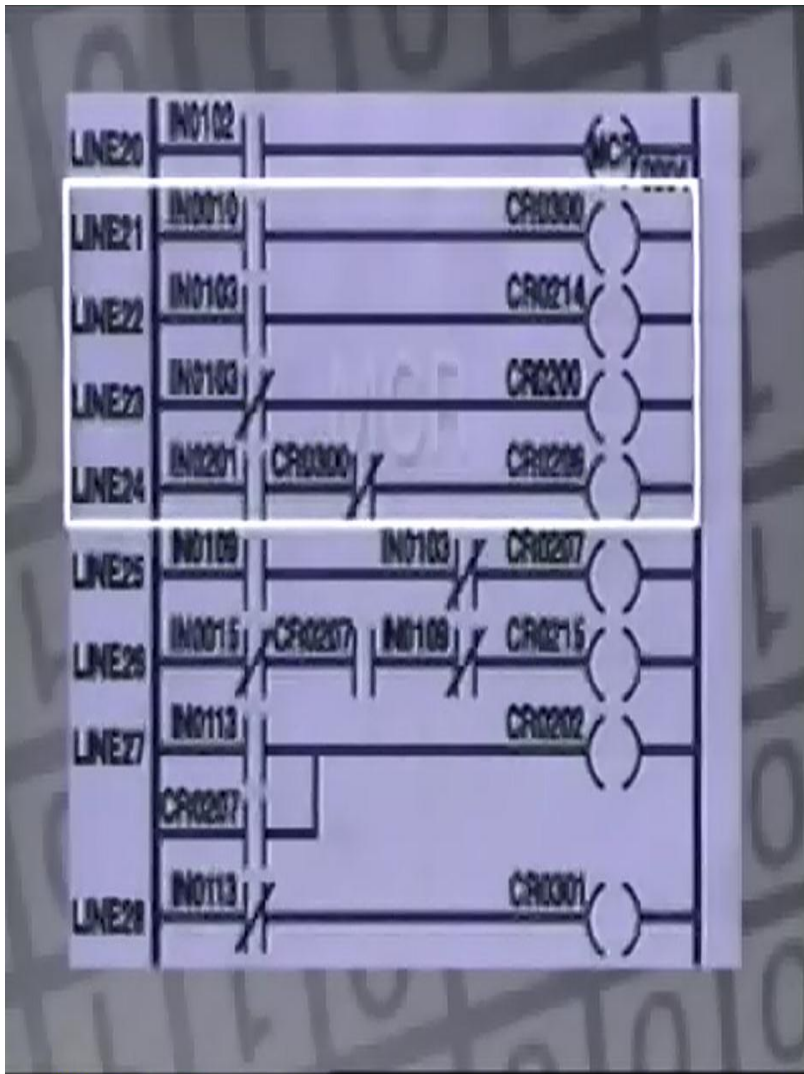


Fig 5.21 Saltos de peldaños con MCR.

La función de programación de secuencias ahorra tiempo, recibe entradas e inicia una serie de eventos de salida, controla un bloque de bobinas en una secuencia preprogramada bajo el control de una función que hace que la secuencia pase de una configuración de bobina a la siguiente OG001. Estas configuraciones son generalmente requeridas en registros HRD000 de la memoria del PLC que pueden ser cambiadas al modificar los valores en estos registros de memoria, este diagrama ilustra la programación del comando de secuencias. 3 líneas de comando se requieren para ejecutar la secuencia, cada línea puede tener entradas múltiples o una sola entrada como IN0101, en este caso solo hay una entrada para cada línea.

La primera entrada IN0101 se conoce como el circuito de paso o STEP cada vez que la línea se energiza ya sea manualmente o automáticamente o con reloj el secuenciador pasa a la siguiente configuración de bobina.

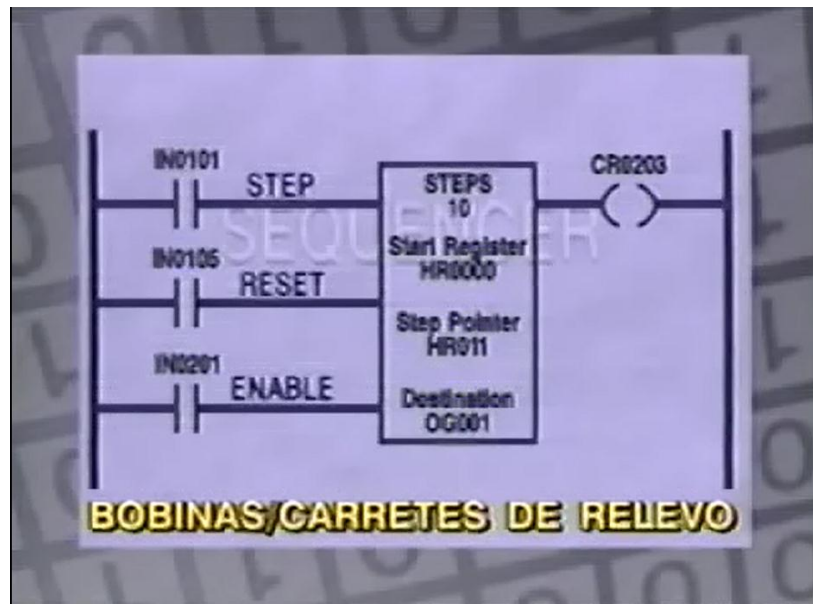


Fig 5.22 Entradas IN0101, 0105, 3201 con salida en CR0203.

La segunda línea de entrada IN0105, es el circuito de RESET, cada vez que se activa esta entrada el secuenciador de entrada es reajustado para comenzar en el primer paso designado. La tercera línea de entrada IN0201 se conoce como la línea activadora, básicamente el switch de encendido.

El comando de secuencias puede ocurrir únicamente cuando el comando de secuencias IN0201 es activado representa el comando de secuencias, para programar este comando se debe incluir el número de pasos u operaciones, un registro inicial el cual es el primer registro en el que los comandos son almacenados. Un director de pasos y un destino para los datos. Ya que se requieren localidades múltiples de registros para almacenar cada paso de datos un grupo de salida OG es definido en el campo de destino.

Debemos especificar los números de registros de memoria temporal los cuales deben estar en secuencia al igual que las bobinas de relevo. Ahora hemos visto los diferentes equipos de comunicación a los que un PLC se puede conectar, operaciones matemáticas de multiplicar, sumar y restar para manipular datos importados, programación de función de un solo disparo, los flip flop RS, T, D para aplicaciones comunes y como usar comandos avanzados de puentes, saltos, MCR, subrutinas de secuencia. Con esto podemos entender como aprovechar al máximo un sistema de PLC.

5.6 BOBINAS DE TIPO RETENTIVO

Programación avanzada con bobinas de tipo retentivo, para distinguir una bobina normal y las bobinas de tipo LATCH, en un diagrama de escalera en el que tenemos un contacto y una bobina, el contacto es normalmente abierto y la bobina es como las que ya hemos visto. Si energizamos el contacto y prendemos el botón para apagar la bobina tendríamos que regresar al mismo botón, por ejemplo de la sala de una casa. Si utilizamos ese botón en otras partes de una casa por ejemplo una escalera, prenderíamos la luz de la sala o la escalera e irnos, entonces para apagar la luz desde otro sitio nos damos cuenta que tendríamos que poner otro apagador ahí mismo. Con esta estructura tenemos un contacto normalmente abierto que cuando sea energizado cerrara y activará la bobina, pero para apagar esa bobina tendremos que hacerlo del mismo renglón.

A diferencia cuando se utiliza la estructura con bobinas tipo LATCH que son las que van a energizar ese bit, y con bobinas de tipo ON LATCH que son las que van a apagar ese bit, querer decir que si tenemos el renglón n al inicio del programa y energizamos una bobina esta se va a quedar retenida o enclavada hasta que en renglón m energicemos otra vez esa bobina de tipo ON LATCH y entonces se apague, lo que nos permite tener en una parte del programa el encendido de una bobina y en otra parte del programa el apagado, el diagrama de escalera que estamos viendo es conocido para nosotros, es el diagrama típico de arranque y paro de motores con su enclavamiento. Donde si energizamos el botón de inicio que es la I: 0 / 0 se enciende el motor y si energizamos la entrada 1 se apaga y este es el diagrama típico que ya conocemos.

Vamos ahora a ver como podemos lograr lo mismo utilizando las bobinas retentivas tipo LATCH, para lo cual nos vamos fuera de línea a otro diagrama y colocamos un contacto normalmente abierto y una bobina de tipo LATCH. Ahora colocamos un segundo renglón y vamos a colocar también un contacto normalmente abierto pero ahora una bobina de tipo ON LATCH, llamemos al primer contacto I : 0 / 0 que es el botón de inicio, la de abajo vamos a establecerla como I : 0 / 1 que es el botón de paro, y las bobinas vamos a llamarlas las O : 0 / 0 y ahora seleccionamos y arrastramos la dirección para copiarla de esta manera. Se arrastramos de este modo una dirección hacia otro elemento o contacto esta se copia, lo que hicimos aquí fue seleccionar y arrastras esa misma dirección.

Ahora tenemos la evaluación del botón de inicio para dar como salida una bobina de tipo LATCH del motor 1, y en el siguiente renglón tenemos la evaluación del botón de paro para dar como salida de la evaluación del renglón una bobina de tipo ON LATCH direccionado al mismo motor. Por efectos de tiempo ya hemos descargado este programa al PLC, podemos observar como los 2 botones están apagados y el motor está apagado, sin embargo si energizamos el botón de inicio el contacto se cierra, el botón se hace verdadero y la bobina LATCH se hace verdadera y se energiza el motor. Podemos observar como se mantiene en verde el motor, quiere decir que está encendido aunque ninguno de los 2 botones esté energizado, aunque el botón de inicio ya no está energizado pero el motor sigue estando energizado.

Y esto va a suceder así hasta que presionemos el botón de paro o energicemos ese contacto y entonces el renglón 1 se hace verdadero y se ejecuta la función ON LATCH, que lo que esto hace es mandar a cero ese bit que en nuestro caso es el O : 0 / 0. Ahora podemos observar que hemos agregado una línea para incluir un timer en esta pequeña lógica de control donde nos damos cuenta que el botón de inicio va a energizar el botón 1 energizando una bobina de LATCH para que cuando el motor este encendido el timer temporice hasta su preset que es de 10 segundos y cuando este timer llegue al tiempo definido se energice la bobina ON LATCH y con ello se apague el motor reiniciando el timer también. Vamos a ver, vamos a energizar el botón de inicio y vemos ahí como

nuestro timer va contabilizando va temporizando y cuando llega a 10 podemos observar que el motor se apaga.

5.7 JUMPO TO LABEL

Tema acerca de los brincos de etiquetas, cuyo objetivo es saber como se controla la ejecución de un programa utilizando la función JUMP TO LABEL, sabemos que el diagrama de escalera se ejecuta de arriba hacia abajo en un orden secuencial, sin embargo que pasaría si quisiéramos alterar un poco el orden de ir renglón a renglón, para ello vamos a ver un ejemplo donde podemos sentar la instrucción de JUMP TO LABEL y lo que hace es brincar de un renglón a renglones mas adelante hasta donde se encuentre la etiqueta de dicho renglón, dependiendo del PLC será que podamos utilizar 256 etiquetas dentro del programa en el caso de que sea un SLC, o bien si fuese un Micro Logix podríamos utilizar hasta 1, 000 etiquetas en un programa.

Como toda instrucción dentro del diagrama de escalera esta debe tener un direccionamiento y el direccionamiento de las etiquetas es Q 2 : y el número de la etiqueta. En la herramienta de SR Logix 500 podemos encontrar bajo la categoría de Program Control, la función JUMP con su mnemónico JMP que es la función que va a brincar a una misma etiqueta que también la encontramos en la misma categoría con el mnemónico LBL que es la etiqueta que indica el punto donde continuará el programa. En este código que estamos observando ya pre hecho por efectos de tiempo, lo que vamos a estar demostrando es como la etiqueta JUMP trabaja en conjunto con la etiqueta LBL de tal manera que cuando la función JMP recibe un valor verdadero en el renglón automáticamente brinca hasta la etiqueta que contiene la misma dirección.

Lo contrario cuando el renglón es falso la etiqueta no se ejecuta por lo que sigue el programa del renglón inmediato siguiente. Para ello elaboramos este ejercicio donde lo que estamos haciendo es utilizar un botón para habilitar o deshabilitar el brinco a una etiqueta de tal manera que cuando se brinque a la etiqueta no se ejecute el renglón 1 y 2. Y que es lo que hace el renglón, este renglón evalúa el valor de una salida y va a poner a esa misma

salida el valor inverso de tal manera que cada iteración va a estar encendido apagado, encendido, apagado. Y dependiendo del estado de esa salida será si contabiliza o no este contador. Vamos a verlo como podemos observar ahora el botón 1, está energizado por consiguiente estamos brincando de la etiqueta JMP a la etiqueta LBL y podemos observar como el contador no se está incrementando.

En el momento que desactivamos el botón I : 0 / 0 podemos ver como el contador se incrementa, y en el momento que lo volvamos a activar automáticamente se detiene el conteo dado que estamos brincando de la línea 0 a la línea 3, sin ejecutar las dos líneas intermedias que son las que hacen que se incremente el contador, nuevamente deshabilitamos el botón de JMP por lo que se deja de hacer verdadero este renglón 0 haciendo que se ejecute el renglón 1 y 2, si volvemos a presionar el botón del renglón 0 activándolo, entonces el renglón se hace verdadero y del renglón 0 se brinca al renglón 3 sin ejecutar las líneas intermedias.

Con esto nos podemos dar cuenta que el utilizar las etiquetas nos ayudan a omitir la ejecución de unas partes del código. En muchos programas se tiene la parte de inicialización, pero esta parte de inicialización solamente se desea que se ejecute como su nombre lo dice al inicio o previo ya a una operación constante del proyecto. Muchas veces se requiere que el operador presione el botón de inicio y en ese momento se verifican cortinas de seguridad, se puede mandar información a una pantalla, se pueden hacer movimiento de registros, movimientos de datos, es decir se prepara el sistema para una operación continua, pero no quisiéramos que cada ciclo del programa es decir cada iteración cada escaneo estuviéramos perdiendo tiempo mínimo y despreciable probablemente pero es un tiempo que puede hacer más eficiente nuestro proyecto.

En esas operaciones que solo queremos que se ejecuten cuando el operador da un botón de inicio, como podemos darnos cuenta en este ejemplo lo que tenemos es una bandera una B 3 que hemos titulado o llamado en operación. De tal manera que cuando el proyecto o el programa ya este operando de manera continua o constante es decir ya le dieron el botón de inicio, no este evaluando la parte de inicialización que para nosotros es la línea 1 y 2, pero

vamos a decir que se energiza por primera vez el PLC por lo tanto la bandera va a estar apagada y si se va a estar ejecutando la línea 1 y 2 de tal manera que como lo vamos a ver ahora que cuando se energiza a través de el botón de inicio se prende la bandera a través de la función LATCH, y automáticamente para el siguiente scan la función JMP va a brincarse hasta la línea 3 logrando con esto se eficiente el tiempo con el que ejecutamos el programa.

Con esto nos damos cuenta que cuando el programa se va ejecutando llega una función JMP Q 2 : y el número de la etiqueta, entonces brincará hasta donde se encuentre la etiqueta, para eso pues continúa con la ejecución, sin embargo es importante que los brincos a etiquetas pueden ser en reversa, es decir hacia líneas anteriores regresando en el código. Sin embargo es muy importante tener cuidado con esto porque podríamos vernos en un loop en el que estamos regresando siempre a líneas anteriores y no le permitimos al programa terminar lo que ocasionaría un problema y un error pues no permitimos que el PLC continúe con su ciclo de programa.

6 PROGRAMACIÓN AVANZADA

La programación avanzada emplea subrutinas para controlar de mejor manera la ejecución de un programa. Hemos visto como se van creando los diagramas de escalera para que se ejecuten constantemente iteración tras iteración. Sin embargo con el paso del tiempo vamos agregando funciones y nuestro código o diagrama de escalera va creciendo, hasta que se pone muy largo y complejo entender ese diagrama. Sin embargo podríamos dividir ese gran código en pedacitos que fueran mas fácil al manejar a los cuales llamaríamos subrutinas. Esos pedazos de código o de programa van a ser elementos que uno llamará al otro y una vez que se ejecuta regresa a la rutina de la que fue llamado.

6.1 SUBRUTINAS

Por ejemplo si el programa 2 está ejecutándose y en una de las líneas se encuentra la función JS y queremos que el programa brinque al programa 3, pues entonces el procesador brincará al programa 3 y ejecutar el diagrama de escalera y al encontrarse la etiqueta END regresará al renglón siguiente de donde fue llamada. Podemos encontrar la función JS en la paleta Program Control. En el apartado Programfiles podemos encontrar las subrutinas. Hasta este momento solo se ha creado la que por default es creada ladder 2 sin embargo nosotros podemos crear diferentes subrutinas y como lo hacemos, damos clic derecho sobre program files, hacemos clic en new y nos va a preguntar que número de subrutina, ladder o escalera queremos utilizar, y podemos asignar un nombre por ejemplo inicio.

Podemos darnos cuenta ahora como ya ha aparecido una nueva ladder 3 o una nueva subrutina, como podemos ver lo que esta dentro de una subrutina, simplemente haciendo doble clic en la subrutina, y si nos fijamos en la parte inferior del diagrama de escalera nos muestra las diferentes subrutinas o diferentes programas que tenemos abierto de este proyecto. Vamos entonces a crear ahora nuestro programa , para ello vamos a empezar en el ladder 2 que para ello vamos a empezar en el Micro Logic el ladder 2 es la subrutina que por default siempre se va a ejecutar, en cuanto el PLC inicie la operación una vez que esté

encendido, vamos a colocar de la categoría de users un contacto al I : 0 / 0 que va a ser nuestro botón de inicio.

Nosotros queremos que cuando se presione el botón de inicio disparemos una subrutina diferente y esto nos va a ayudar a que si durante el proceso de inicialización necesitamos crear muchas operaciones, necesitamos hacer ciertas funciones como manejo de datos, pues queremos tener una rutina para que siempre se ejecute cuando esto sea y no esté ocupando espacio en otra parte de nuestro código. Para ello vamos a irnos a la categoría con program control, vamos a tomar la función JTS, nos pregunta que file o nombre de archivo queremos usar, escribimos que vamos a utilizar el 3, ahora podemos programar lo que en ladder 3 se da. Nosotros lo que vamos a querer es que cada vez que se meta el programa a ladder 3 se active un timer que nos permita ver como si estamos ejecutando la rutina 3.

Colocamos un timer que va a ser el T 4 : 1, tiempo base de 1 y un preset de 2 segundos, y luego lo que queremos es que ese contacto funcione como un reset automático por lo tanto al botón reset le ponemos normalmente abierto un T 4 : 1- DONE. De tal manera que cuando automáticamente se llegue al preset de 2 segundos este contacto se va a abrir, por lo tanto el renglón se hace falso y con ello estaremos reiniciando el TON. Regresamos a nuestro ladder 2 y lo que vamos a hacer ahora es colocar una nueva línea en donde vamos a poner un contacto direccionado al timer número 1 pero ahora al bit de temporizado para que cada vez que se temporice podamos incrementar la cuenta de un contador. El cual va a ser el C 5 : 0 y por un preset de 20 únicamente para efectos de demostración.

Tenemos aquí un botón de inicio que cuando hacemos que se presione ese contacto normalmente abierto tengamos un botón de verdadero, el renglón se hace verdadero y por consiguiente se ejecute la función JS, la cual nos va a hacer brincar al ladder 3, en el ladder 3 lo que tenemos es un timer que estará contando hasta 2 segundos y cada vez que llegue a 2 segundos se reinicia. Por efectos de tiempo ya hemos descargado el programa, podemos observar que ya está corriendo el programa sin embargo nuestro contador no está incrementado, vamos a ver que sucede cuando energizamos el botón de inicio.

En ese momento el botón de inicio está energizado y podemos ver como el contador está incrementando.

Esto porque el botón de inicio hace verdadera la instrucción de JS y nos envía a ejecutar lo que está dentro de esta subrutina y una vez que termina o se encuentra con la función END regresa a la rutina que lo disparó a la rutina siguiente. Si dejamos de presionar el botón de inicio el renglón ya no es verdadero y por lo tanto no ejecutamos la rutina 3, por lo que no se va a incrementar nuestro contador.

6.2 FUNCIONES MOVE Y COPY

Las funciones MOVE y COPY conociendo ya los archivos de datos, que es y como se utiliza la herramienta RS Logic, en ocasiones tenemos datos que necesitamos duplicar o copiar, para que después podamos sustituir el dato original con algún otro dato, esto lo podemos lograr con la función MOVE, lo que hace es copia a una palabra a una palabra destino sin eliminar la palabra de origen. Ya dijimos la función MOVE va a mover el dato contenido de una palabra a otra palabra sin eliminar el dato de origen, para demostrar la función MOVE vamos a colocar ahora una nueva línea, y vamos a colocar un contacto normalmente abierto porque la función MOVE únicamente copiará el dato si el renglón en el que está colocado es verdadero. Y entonces vamos a utilizar este contacto normalmente abierto para mover la palabra cada vez que sea presionado.

La función MOVE la podemos encontrar en la carpeta o en la categoría MOVE / logic, entonces me pide SOURCE o el origen de donde va a tomar los datos, para este ejemplo le decimos que lo queremos de un N 7 : 0 y queremos que mueva o copie el dato a un N 7 : 10, vamos a abrir el archivo de datos N 7 para poder ver que es lo que está sucediendo, lo que estaríamos haciendo con la función MOVE o esperando hacer es que copie la palabra N 7 : 0 sea copiada hasta la palabra N 7 : 10 entonces queremos ver que los bits de la palabra N 7 : 0 sean colocados en los bits de la palabra N 7 : 10. Ahora esperamos el botón de inicio se copien todos los 1's y todos los 0's.

Ahora ya hemos abierto el contacto por lo tanto ya no se están moviendo los datos, para ello vamos a colocar todos los bits en 1 de la palabra de 16 bits. Ahora nos damos cuenta que N 7 comprende puros 1's y al momento de que cerremos el contacto normalmente abierto el renglón se hará verdadero y por consiguiente la función MOVE moverá los datos del registro copiándolos del N 7 : 0 al registro N 7 : 10 y al momento que lo cerramos podemos observar como N 7 : 10 es llenado con los valores de N 7 : 0. Ya vimos como la función MOVE lo que hace es mover un dato, sin embargo existe una función llamada COPY que lo que hace es mover varias palabras en una sola instrucción.

Ya sabemos que lo que hace la función COPY hace copiar una palabra origen a una palabra destino pero a diferencia de la instrucción MOVE podemos mover varias palabras con una sola instrucción, palabras que estén consecutivas. Y para que esto quede un poco mas claro vamos a hacer un pequeño ejemplo, vamos a colocar la función COPY y esa la podemos sacar de la categoría File / misceláneos al igual que la función MOVE solo se ejecutará cuando sea verdadero el renglón. Vamos a decir que nuestro origen N 7 : 0 y queremos que lo copie de igual manera que cuando lo hicimos con la función MOVE N 7 : 10. La diferencia aquí es que vamos a poner una longitud de 5 y con esto, lo que le estamos diciendo es que queremos que copie 5 palabras consecutivas, a partir del N 7.

Y si nos damos cuenta nos pone un símbolo de gato antes de la dirección N 7 : 0 esto nos indica que está indexado, es un conjunto de palabras la que va a estar operando la función, vamos a ver que la mejor forma de hacerlo es con un ejemplo, N 7 : 1 hasta N 7 : 4 ya tienen arreglos, cuando hagamos el copy lo que podemos ver que de N 7 : 0 hasta N 7 : 4 que son 5 palabras se copien hasta N 7 : 10 en adelante que es N 7 : 14. Vamos a ver un poco mas esta pantalla para que lo podamos ver de la mejor manera, ahora vamos a presionar el botón 0 como podemos darnos cuenta cuando el botón 0 es presionado el botón se hace verdadero, por lo tanto se ejecuta la función copy y lo que sucede es que del renglón 0 se copien los datos de toda la palabra al renglón 10. Del renglón 1 se copia toda la palabra al renglón 11, del renglón 2 al renglón 12 consecutivamente acumular 5 palabras.

Pues 5 palabras fue el valor definido de campo length, una vez que se deja de presionar el botón la función copy se hace falso por lo tanto no se van a estar copiando los valores, ahora si yo coloco un 0 al inicio de cada uno de estos valores de palabra, al momento que se van a copiar a presionar el botón vamos a observar a presionar como estos valores se harán cero. Vamos a ver está presionado el botón y han sido copiadas nuevamente las primeras 5 palabras a partir del N 7 : 0 que fue como lo definimos al N 7 : 10 porque está definida aquí y a partir del N 7 : 0 son 5 palabras dado que esa es la longitud que definimos para este ejemplo.

6.3 FUNCIÓN FILL

En programación avanzada ahora hacemos uso de la función FILL apoyado en funciones MOVE, COPY y el manejo de subrutinas en RS Logic, en un resumen de lo que es el manejo de datos a través de las subrutinas y la función MOVE.

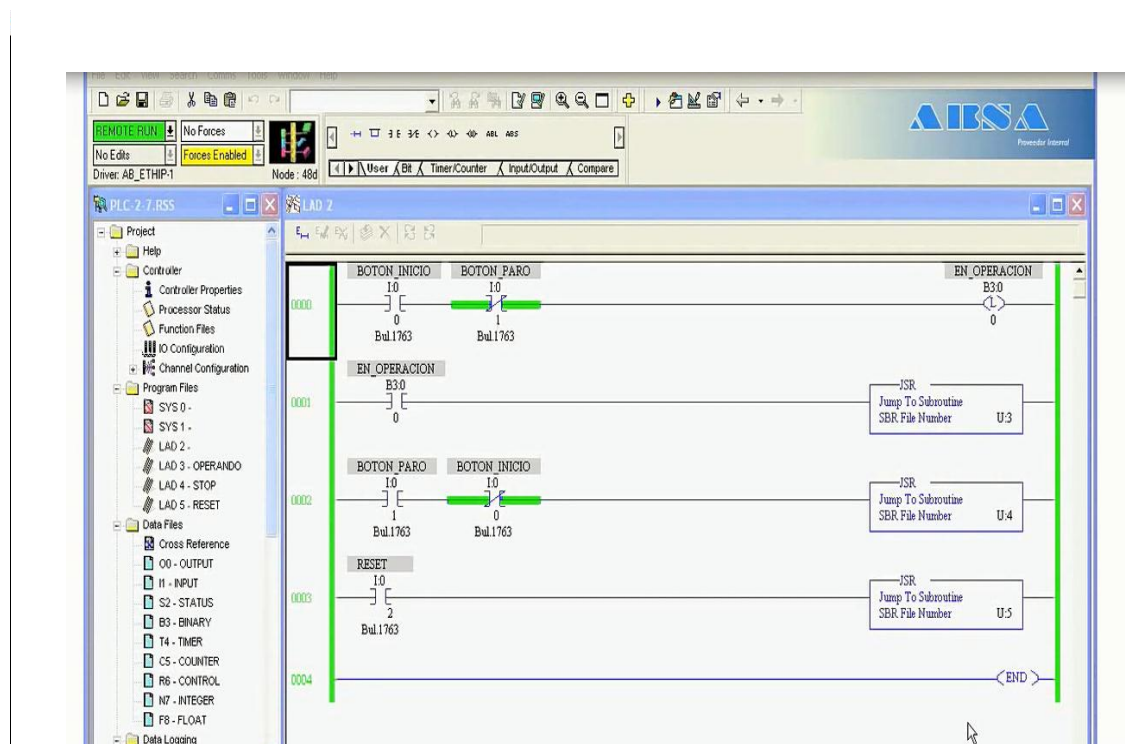


Fig 6.1 Diagrama de RS Logix 500 con 3 subrutinas una en cada peldaño.

Del siguiente diagrama tenemos el ladder 2 que por default ejecutarse como el principal, el renglón 0 se utiliza para cuando se presiona el botón de inicio no está presionado el botón de paro la bandera en operación se haga 1 a través de una función de salida de tipo LATCH, luego en el renglón 1 si ya está en operación es decir si ya se presionó el botón 1, hace un brinco a la subrutina número 3, lo que vemos aquí es que una vez que está energizado o activada la bandera en operación el timer va a empezar a contabilizar, al mismo tiempo en la subrutina 3 se evalúa el sensor de las botellas, y el censor de la botella que no está bien, porque este podría ser bien el control de la línea de llenado de botellas, donde si la botella esta bien llena se pasa a la banda de empaquetado mientras que no a la banda de descarga.

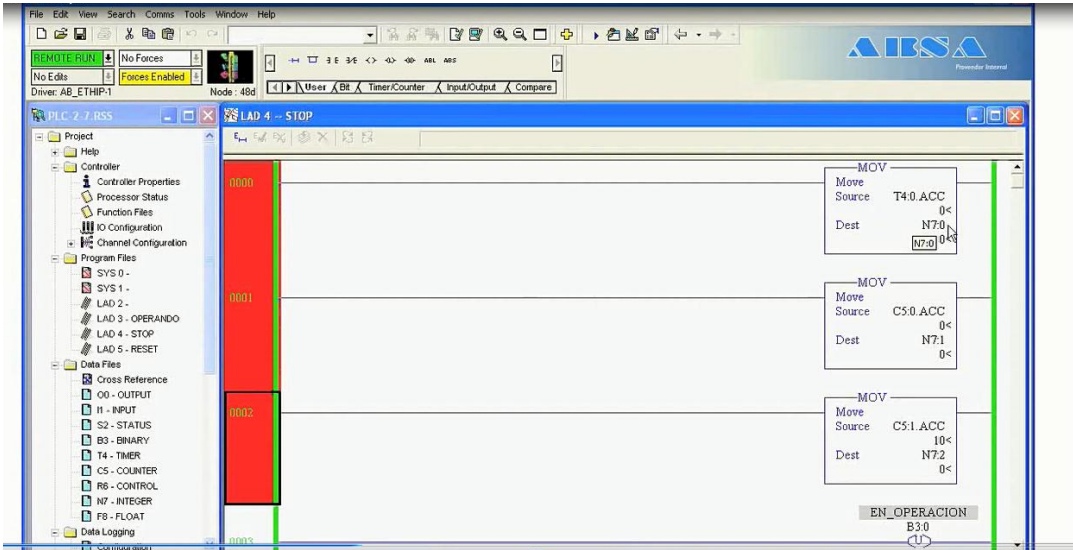


Fig 6.2 Subrutinas para evaluar el llenado de botellas.

Lo que queremos con este programa es poder controlar cuantas botellas han sido buenas y cuantas han sido malas en un determinado tiempo para ello tenemos un censor de botellas buenas y un censor de botellas malas, una vez que se termina de ejecutar la rutina 3 se regresa a la subrutina 2 al renglón 2 donde se evalúa si se ha presionado el botón de paro y no el de inicio que estaba presionado, para entonces entrar a la rutina 4 que hemos nombrado STOP, que sucede cuando se presiona el botón de paro, bueno se mueve el valor o se copia el valor del acumulado del timer 0 al N 7 : 0 y el acumulado de las botellas

buenas que es el del contador 0 al N : 7 número uno. Se energiza la bobina ON latch por lo que la bandera en operación se va a hacer 0.

Y regresamos a la subrutina 2 y se evalúa del renglón 3 si el botón de reset ha sido presionado, en caso de que si se haya presionado el botón de reset, nos envía a la subrutina 5 donde lo que se hace es se pone un 0 a la N 7 : 0, N 7 : 1, N 7 : 2 y un 0 al acumulado del timer, lo que podemos entender es que con este lo que estamos haciendo es inicializar nuevamente nuestros registros de memoria, nuestros valores acumulados que hemos guardado para poder tener los valores en 0, pues bien vamos a ver como funciona esto, para ello vamos a hacer desde la rutina 2 vamos a presionar el botón de inicio como no está presionado el botón de paro, se energiza la bobina LATCH se pone en 1 la bandera de en operación, y se empieza a ejecutar con cada scan del programa la rutina 3 y podemos ver en la rutina 3 como dado que la bandera de en operación está energizada.

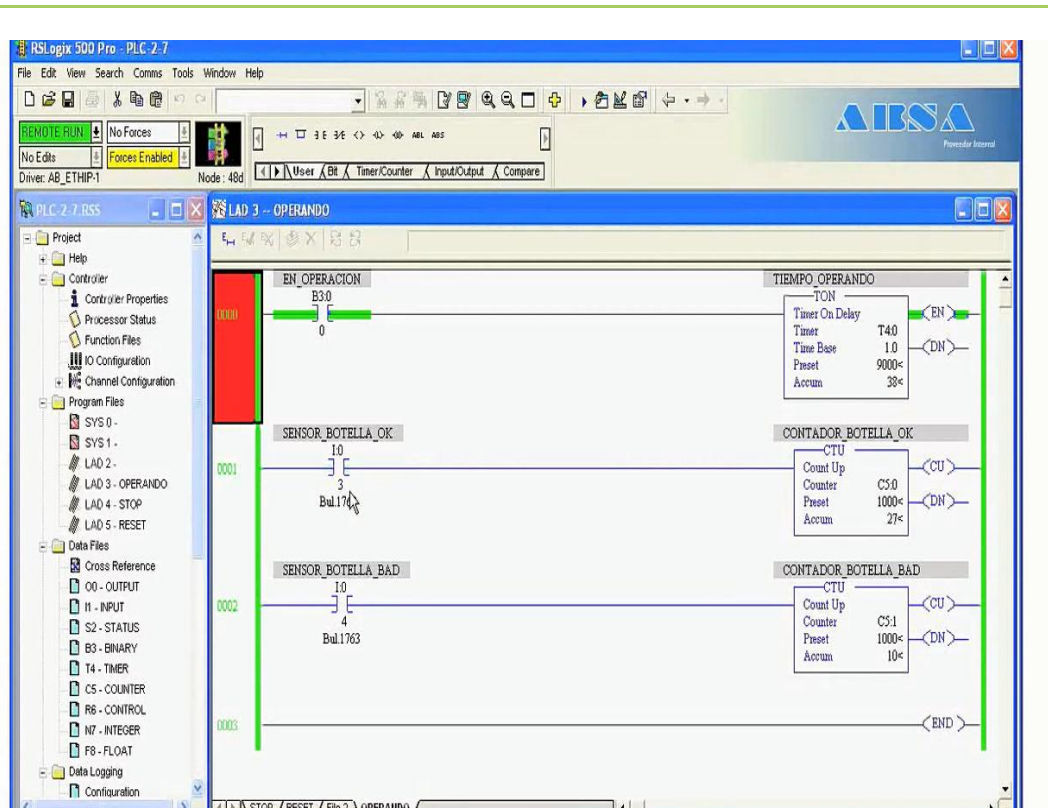


Fig 6.3 contenido de la subrutina 3 de llenado de botellas.

Como la bandera de en operación está energizada el timer está temporizando, ahora podemos presionar repetidas veces el sensor de las botellas y vemos como se va incrementando el contador número 0. Si ahora regresamos al ladder 2 y presionamos el botón de STOP, lo que vamos ha hacer es ejecutar rutina 4 al momento que sea presionado el botón 2, se ejecuta la rutina 4 y se energiza la bobina ON LATCH, se apaga esa bandera y se copia los registros acumulados del timer y de los 2 contadores a los N 7 que se han asignado. Y podemos revisar ahora los valores que tenemos en N 7 donde el valor N 7 : 0 es el acumulado en segundos, el N 7 : 1 es el acumulado en el contador 0 y el N 7 : 2 es el acumulado del contador 1. Podemos ver como ya fueron copiados a estos espacios de memoria.

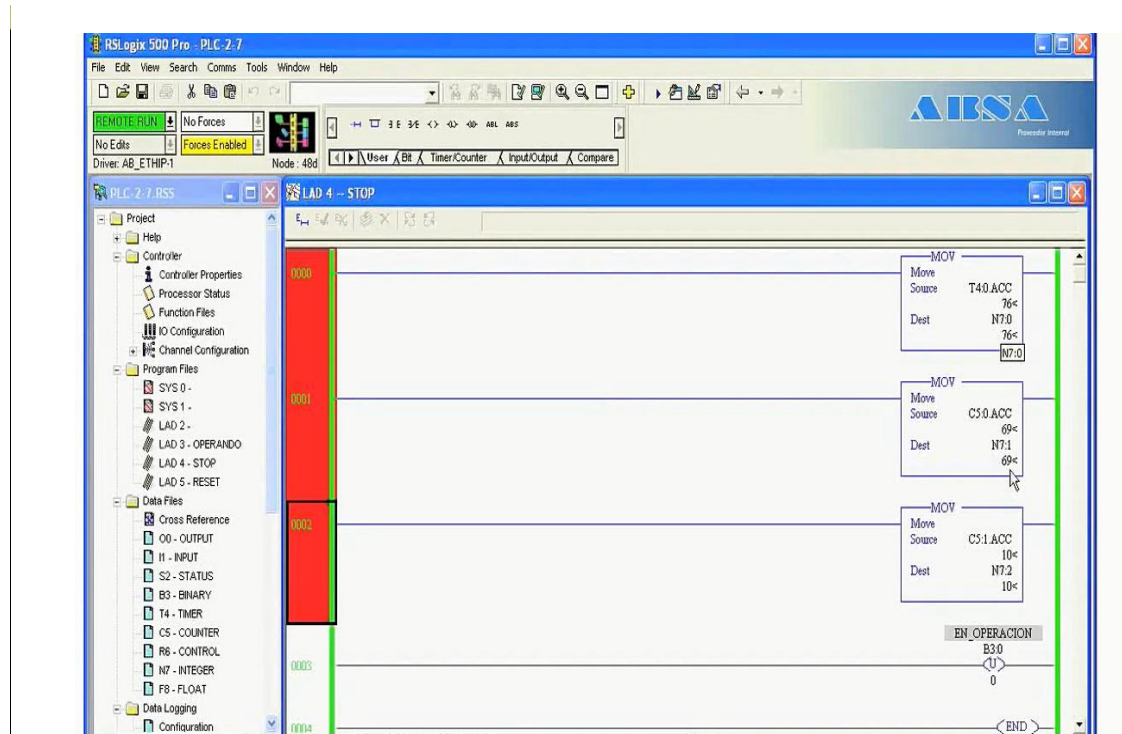


Fig 6.4 Contenido de la subrutina 2 de llenado de botellas.

Sin embargo si regresamos al ladder 2 vemos que cuando presionemos el botón de RESET se ejecutará la rutina 5, esta rutina lo que hace es colocar o copiar el valor 0 a N 7 : 0, N 7 : 1, N 7 : 2 e incluso copia un 0 al acumulado del timer 4 para cerciorarnos que el timer también sea reseteado. Por eso vamos a ver como si presionamos el botón de RESET todos

los espacios y los valores de memoria se van a 0, porque tenemos aquí la función. Vamos a hablar ahora sobre la función FILL y para qué la podemos utilizar.

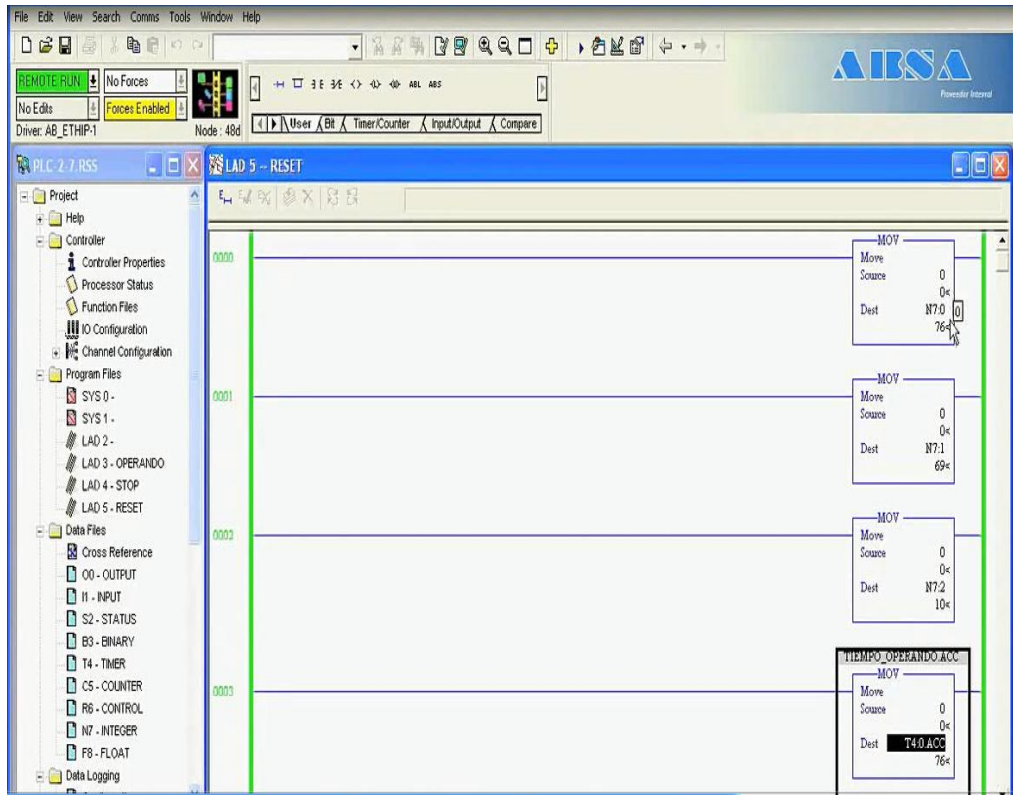


Fig 6.5 Contenido de la subrutina 1 de llenado de botellas.

La función FILL es muy usada al inicializar o resetear valores, aquí estamos utilizando 3 renglones para inicializar los valores, voy a eliminar de este programa los primeros 3 renglones, vamos a agregar una nueva línea y de la paleta FILL vamos a poner la función FILL, me pide el origen que quiero copiar quiero mandar un 0, a donde a N 7 : 0 y me pide el valor length, vamos a indicarle que queremos copiar 3 valores, o 3 palabras porque tenemos la de la 0, 1, 2. automáticamente con esto que le estoy especificando es que quiero es que copie el valor SOURCE a todas las palabras que haya determinado en este caso a partir del N 7 : 0. Cuando damos el botón de reset automáticamente todos los N 7 que estamos utilizando son llenados con 0 porque tenemos la función FILL que lo que está haciendo es tomar este 0 y mandarlo al N 7 : 0, ese misma valor los N 7 : 1, N 7 : 2.

Como podemos observar la función FILL uno de sus mayores usos es para la inicialización de datos.

6.4 OPERACIONES MATEMÁTICAS

Como utilizamos las funciones matemáticas mas comunes, como en todos los proyectos a veces es necesario utilizar operaciones matemáticas como la suma, resta, multiplicación y división. Para ello vamos a entrar a la paleta de Compute / Mat, donde podemos encontrar todas las operaciones matemáticas básicas como suma, resta, multiplicación, división, raíz cuadrada, negación de signo, convertir a BCD, de BCD a decimal.

Vamos a colocar una suma podemos notar que nos esta pidiendo el origen A, el origen B y un destino. Como origen A vamos a seleccionar un N 7 : 0, como origen B vamos a decirle que queremos sumarle una constante de 10. Queremos almacenar el resultado de la suma en N 7 : 1. Las funciones como las operaciones matemáticas son instrucciones de salida por lo que dependiendo de si el renglón es verdadero o falso se ejecutarán o no.

En caso de que sea verdadero se ejecutará por lo que vamos a colocar un contacto normalmente abierto, para que este pueda decir cuando queremos que se esté ejecutando la suma, vamos a abrir el archivo de datos N 7 para poder ver como ahora N 7 : 0 está en 0, al igual que N 7 : 1, vamos a presionar el botón de suma y nos damos cuenta como ahora N 7 : 1 tiene el valor de 10 dado que 0 mas 10 nos da 10. Bien vamos por ejemplo ahora a modificar el valor de N 7 : 0 vamos a ponerle un 120 y al momento que presionamos el botón entonces se hace verdadero el renglón, 120 mas 10 es almacenado en N 7 : 1 donde podemos observar ahora tenemos un 130.

Que pasaría si quisiéramos usar una estructura de $A + B = A$ es decir donde el destino sea el mismo espacio de memoria, esto sería como tener un N 7 : 0 mas una constante igual al mismo N 7 : 0.

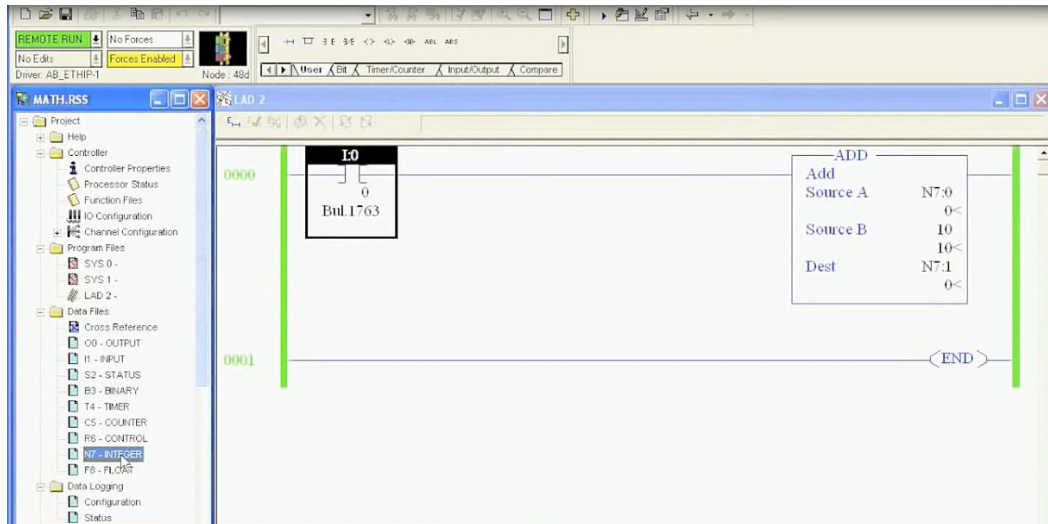


Fig 6.6 Función de sumador.

Vamos a abrir nuestro archivo de datos nuevamente y vamos a colocar un 0 en N 7 : 0 y un 0 en N 7 : 1, vamos a ver como al momento que se presione el botón de entrada se incrementa la cuenta de N 7 : 0, se pueden dar cuenta que incrementa muy rápido esto es porque el programa tarda prácticamente nada en ejecutarse.

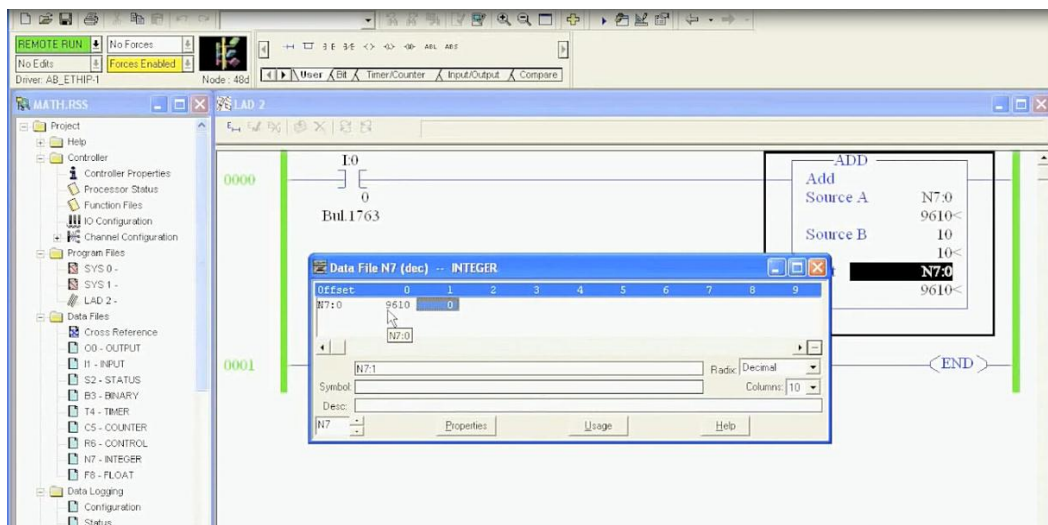


Fig 6.7 Resultado del sumador en la ventana de RSLogix 500.

Es decir ejecuta muy rápido, quiere decir que al presionar el botón con el dedo el programa alcanza a dar muchas vueltas y entonces la cuenta se incrementa rápidamente. Entonces

cada ciclo de programa o cada vuelta de programa se está sumando el valor que en ese momento está en N 7 : 0 más un 10 y almacena el valor en N 7 : 0 nuevamente.

6.5 DIRECCIONAMIENTO INDIRECTO

Explicamos como utilizar el direccionamiento indirecto en RS Logic, para entenderlo lo mejor posible sabemos como utilizar los secuenciadores, MOVE para entender el siguiente ejemplo. Sabemos con anterioridad como podemos utilizar la función MOVE para mandar un dato a una palabra a otra donde como origen hemos determinado un N 7 : 0 donde N hace referencia al tipo de archivo, el 7 es número de archivo de datos que estemos utilizando y el 0 es el número de palabra.

Sin embargo a veces existe la necesidad de utilizar un direccionamiento indirecto que dependiendo de las condicionantes del programa podamos identificar.

Imaginemos una carrera y nos han pedido que hagamos un proyecto en donde podamos cronometrar el tiempo, que tenga un censor en la meta y no indique quién es el ganador, hemos decidido poner el valor del a medalla de oro el tiempo en N 7 : 1 el tiempo del ganador de la medalla de plata en N 7 : 2 y el tiempo del perdedor será almacenado en N 7 : 3.

Veamos ahora en RS Logic como podríamos hacer esto con las funciones MOVE para después ver como utilizando la función o como utilizando el direccionamiento indirecto podemos eficientar el programa.

Este programa lo que hace es que cuando inicia la carrera se presiona el botón de inicio para energizar una bobina B 3 : 0 / 0 la cual va a hacer que el timer empiece a temporizar y cronometrar para llevar el registro de la carrera.

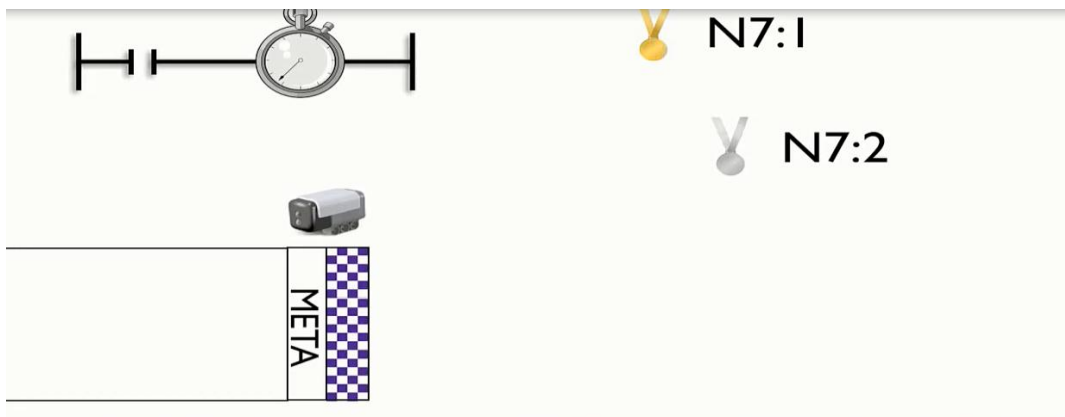


Fig 6.8 Elementos para el control de 1, 2 lugar.

Mas debajo de nuestro programa lo que vamos ha hacer es que cuando a la meta llegue el sensor se va a activar incrementando la cuenta de este contador y al mismo tiempo haríamos una comparación de que nuestro contador sea el igual a 1, el cual sería el primer lugar y en caso de que eso suceda es cierto vamos a trasladar el valor del timer y lo vamos a guardar el valor en el N 7 : 1. en caso que el contador sea igual que 2 entonces ahora el valor del timer será almacenado en N 7 : 2, o bien en caso que el contador sea igual a 3 que será el tercer lugar de los concursantes el timer será enviado al N 7 : 3. Vamos a ver como funcionaría este programa, vamos a dar inicio a nuestro timer como pueden ver ya se esta temporizando y entonces vamos a bajar un poco este programa para que podamos ver como cuando pasa el primer concursante.

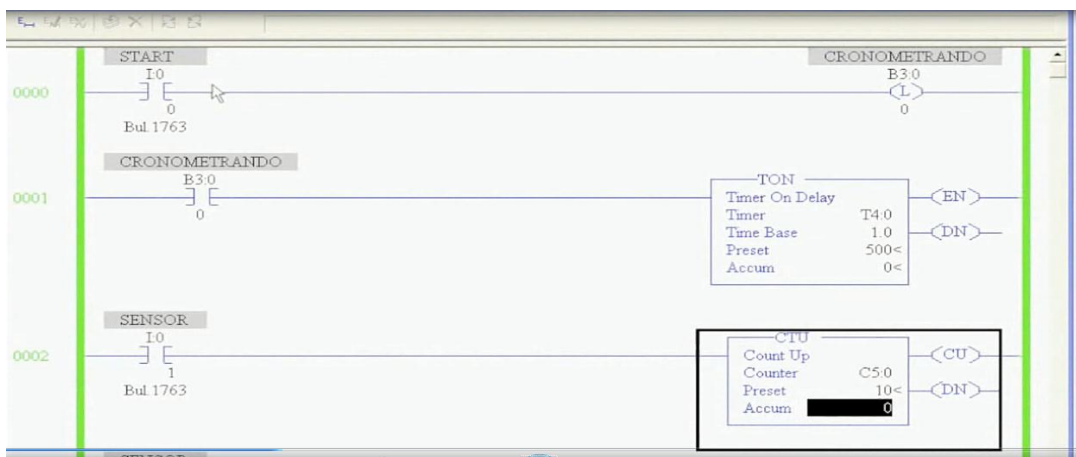


Fig 6.9 Estado del programa al pasar el primer lugar.

Abrimos nuestro archivo de datos vemos cuando se presiona el botón el contador va a incrementar su cuenta en 1, al mismo tiempo el sensor va a leer si ya se presionó, si el contador es igual que 1 entonces vamos a poner ese valor en N 7 : 1, presionamos el botón y el contador se hizo 1 y entonces podemos ver como ahora N 7 : 1 tiene el valor de 92 segundos. Si volvemos a presionar el sensor en N 7 : 2 con 107 segundos, y si volvemos a presionar por último para nuestro tercer concursante el botón o el sensor podemos darnos cuenta como el N 7 : 3 se ha llenado con el valor 120 segundos que tenía el acumulado de nuestro timer. Algo importante es que hemos puesto 3 líneas para lograr hacer esto, este es un programa que se ha utilizado mas utilizando las instrucciones indirectas.

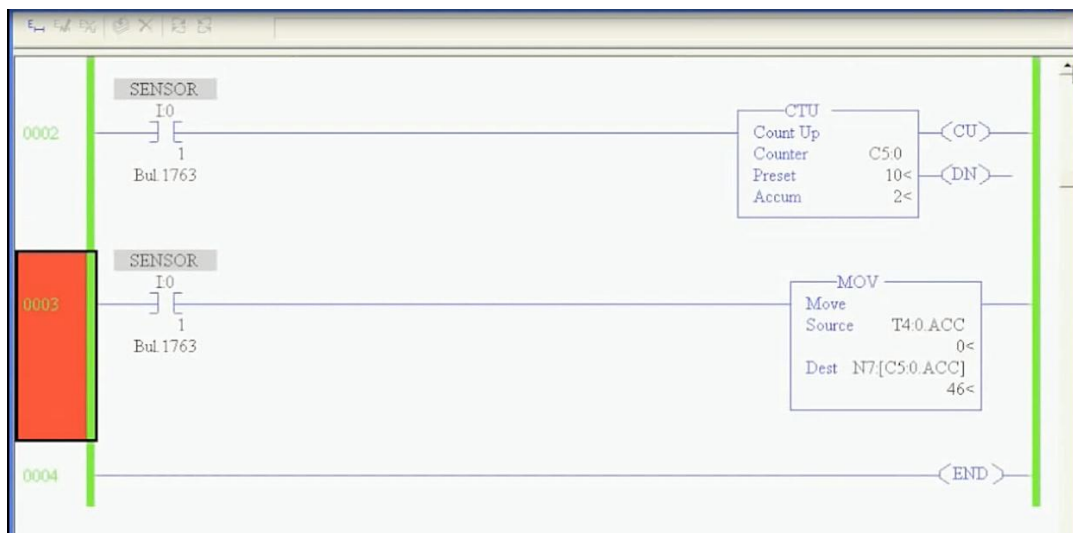


Fig 6.10 Momento del programa al pasar el segundo lugar.

Lo que nos ha permitido reducir el número de líneas y número de funciones que utilizamos para lograrlo.

Vamos a revisar este código de igual manera tenemos un botón de inicio que va a enclavar o va a prender la bobina B 3 : 0 / 0, el cual va a utilizar el timer el cual va a utilizarse como cronómetro, de igual manera tenemos un contador para saber cuantos concursantes ya han cruzado la meta, pero fíjense como la parte que registra los datos en N 7 es simplemente una línea, y que hacemos una vez que se presiona el sensor, se mueve el valor que esté temporizado es decir acumulado del timer a una dirección, sin embargo podemos observar

como en la parte de almacenar los datos solo tenemos una línea o instrucción dado que estamos haciendo una dirección indirecta es decir tenemos un N 7 : [C 5 : 0.ACC] que es el valor acumulado y cerramos corchete.

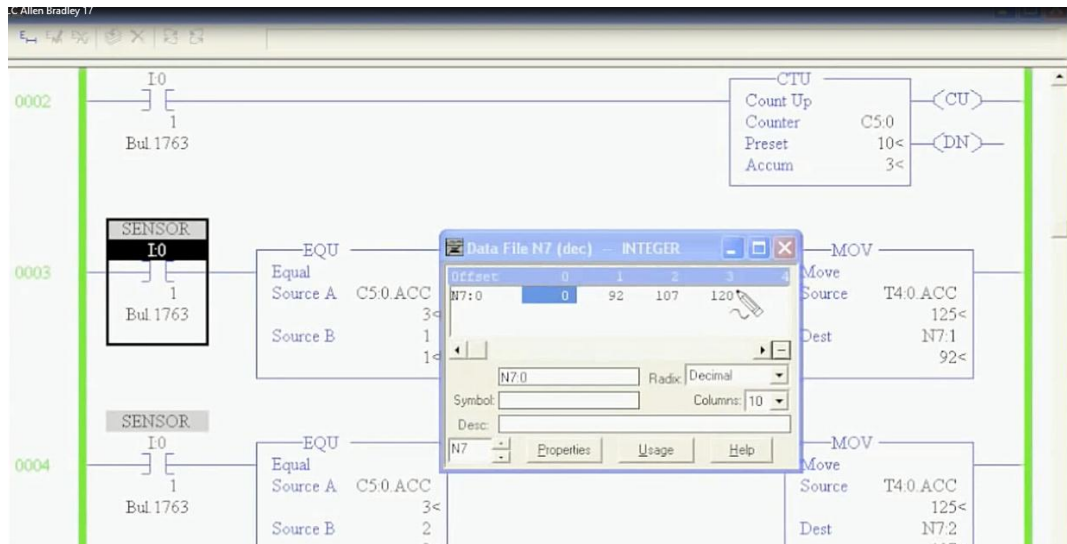


Fig 6.11 Diagrama del segundo lugar con su registro de tiempo.

De tal manera que en cuanto el contador esté en 0 se va a guardar en N 7 : 0, cuando el contador esté en 1 se va a guardar en N 7 : 1 y cuando el contador esté en 2 se va a guardar en N 7 : 2. y entonces con esto como si fuera una función matemática se sustituye la variable del corchete por el valor que se tenga en el contador acumulado y entonces eso hace que podamos reducir significativamente nuestro código. Vamos a ver como funciona ahí vamos a abrir nuestro archivo de datos en N 7 para ver como cuando presionemos el botón que representa el sensor se va a activar la cuenta, al mismo tiempo se va a mover el valor que hay en el acumulado del timer a un N 7 y dependiendo del valor del contador será el N 7 que se utilice.

Por segunda vez nos damos cuenta que el acumulador del contador era 1 y por lo tanto el valor del timer se ha registrado en N 7 : 1. si ahora volvemos a presionar el sensor nos damos cuenta que lo registra en N 7 : 2 y si lo presionamos nuevamente nos damos cuenta que lo va a registrar en N 7 : 3.

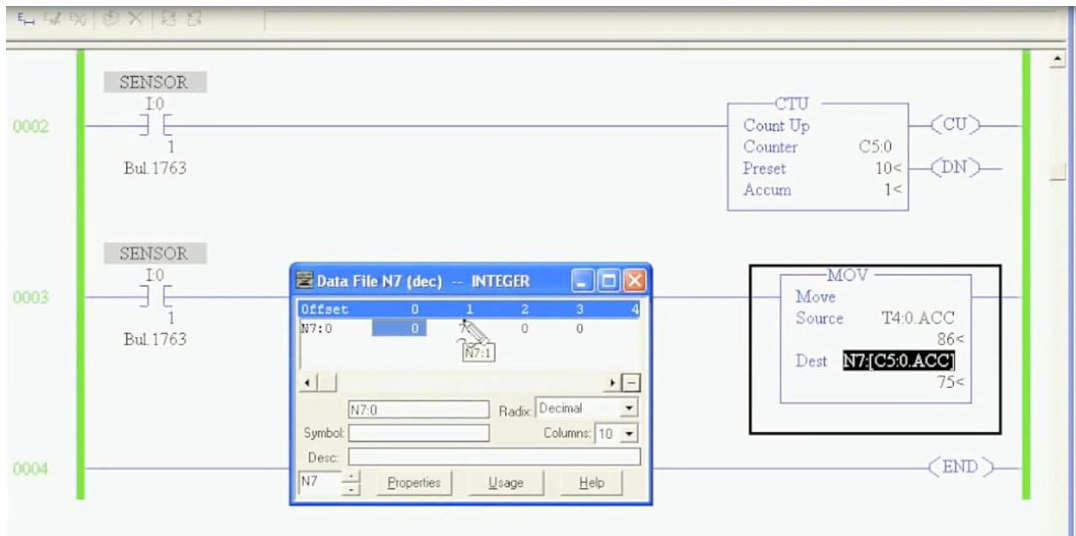


Fig 6.12 Registro de tiempo del primer, segundo lugares.

6.6 FUNCIÓN BITSHIFT

La instrucción bitshift como podemos utilizarla, muchas veces tenemos en la industria líneas de llenado o de etiquetado en los que desarrollan proyectos que en algún lugar se esté evaluando la pieza y en otro lado se esté descartando, lo que necesitamos es alguna palabra o lugar donde se vaya almacenando ese dato, en este ejemplo tenemos una línea que es donde llegan piezas altas y piezas bajas, necesitamos expulsar aquellas piezas que sean altas, para eso vamos a utilizar la palabra B 3 : 1 donde vamos ir almacenando si hay que descartar o no cuando llegemos a la posición de descarte, por ejemplo ir a la primer pieza y dado que es una pieza baja no hay que descartarla en la palabra que se va a ir almacenando, un 0 colocamos e indicará que no es necesario activar el pistón de expulsión.

La siguiente pieza que llega es una pieza alta por lo que esa si habrá que descartarla del sistema, así observamos que en la palabra B 3 se ha insertado un 1, se ha desplazado el 0 que teníamos anteriormente hacia la izquierda, así nuevamente cuando llega una pieza alta y el contador ya ha incrementado a dos su cuenta de piezas altas, hay que desplazar hacia la izquierda el valor de descarte de las piezas, otro cero si llega un cajón vacío donde no hay ni pieza alta ni pieza baja pues no necesitamos descartar nada

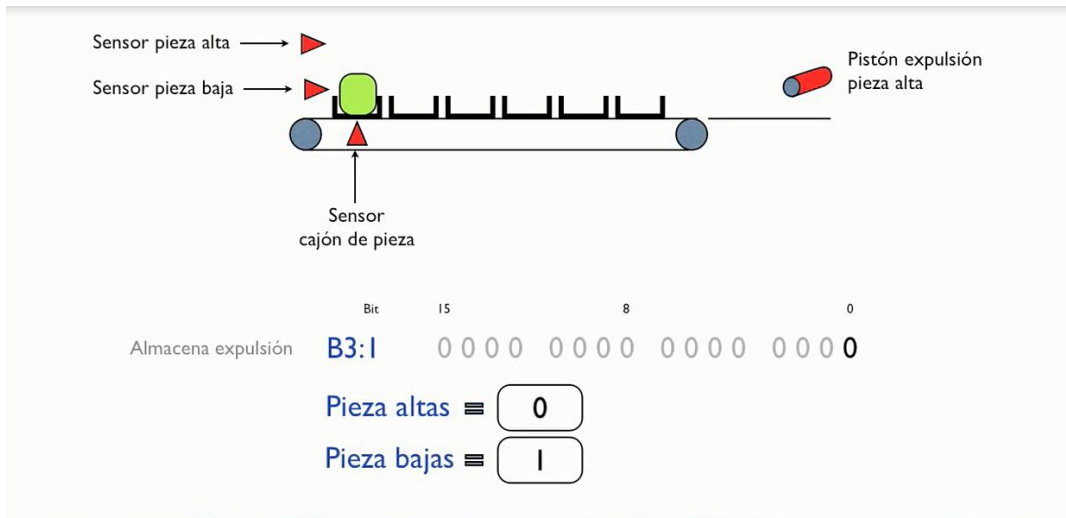


Fig 6.13 Palabra de estado en bits según sensores de pieza.

La siguiente pieza es una pieza baja por lo que se inserta un 0 en la palabra de almacenamiento de tal manera que no habrá que descartarla, llega la siguiente pieza que es una pieza alta por lo que se coloca un 1, podemos darnos cuenta que los bits se han ido desplazando hacia la izquierda sin importar si son ceros o unos.

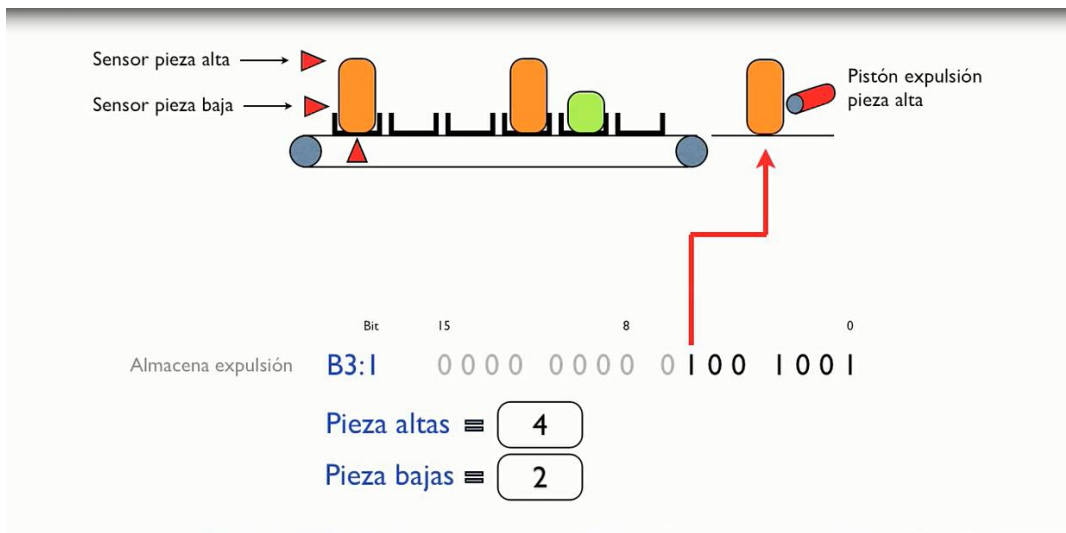


Fig 6.14 palabra de estado de 16 bits que muestra algunos bits ya ocupados.

Lo interesante viene cuando la pieza llega a la posición de descarte y entonces se evalúa si es un 0 no se activará la pieza de descarte, en este ejemplo cuando la pieza llega ya a la posición de descarte, y el valor que tenemos es 1 entonces si se energiza el pistón de expulsión. Nuevamente tiene un 1 y se energiza el pistón de expulsión y como a la entrada ya llegó una pieza alta se vuelve a agregar a la palabra de almacenamiento un 1, como se puede hacer esto, bueno utilizando la función bit shift left donde el array que nos pide es la palabra donde se modificarán los datos, donde vamos a estar almacenando esos datos, luego nos pide control que al igual que otras funciones conocidas es una palabra de control donde se almacena información necesaria interna para la operación de estas funciones.

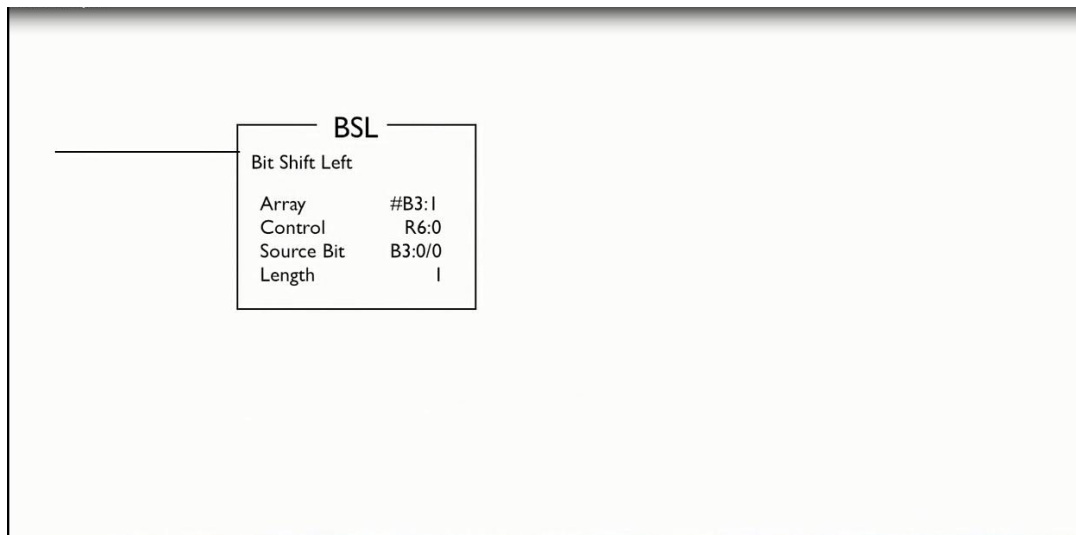


Fig 6.15 Diagrama de escalera de un BLS del ejemplo de palabra de bits.

Luego viene el source bit que es el bit que se ingresará a la regla, es decir cuando se desplace hacia la izquierda los bits y se introduzca un nuevo bit de donde va a tomar ese bit ya sea una entrada digital o un espacio de memoria, finalmente length que es el número de bits que se van a desplazar hacia la izquierda. Veamos como lo podemos utilizar utilizando un RS Logic 500.

El programa lo hemos dividido en 3 subrutinas, la principal es la que lleva el control del programa, la subrutina de descarte es la que se encarga de las botellas altas y bajas, la

subrutina 4 es la que se encarga de llevar el conteo de las piezas altas y bajas que han pasado por la banda. En la subrutina 2 en la línea 0 lo que tenemos es un sensor que determina cuando la banda transportadora ya ha llegado a la siguiente posición o siguiente cajón y entonces pone en 1 un bit utilizando one shot righth que esta instrucción lo que hace es cuando se detecta el flanco de subida, pone un bit en un solo escan en 1 y después lo apaga, en la línea 1 lo que se está haciendo es una vez que el cajón ya llegó a su posición se utiliza el bit shift left para introducir el bit cero de la palabra B 3 : 1 el valor de si debe ser descartada o no.

Ya hemos dicho que van a ser descartadas aquellas piezas que sean altas, por lo tanto se pondrá un 1 si la pieza es alta, se pondrá un 0 si la pieza es baja o si no hubiera pieza. Y después en las líneas 2, 3 se hacen llamados a las subrutinas de descarte y de contadores y al final de la línea 4 lo que se hace es evaluar si la pieza en posición 6 debe ser descartada para entonces activar la señal de descarte. Vamos a ver como funciona el programa, para ello vamos a entrar a la subrutina de contadores y vamos a abrir para verlo de mejor manera nuestro archivo de datos.

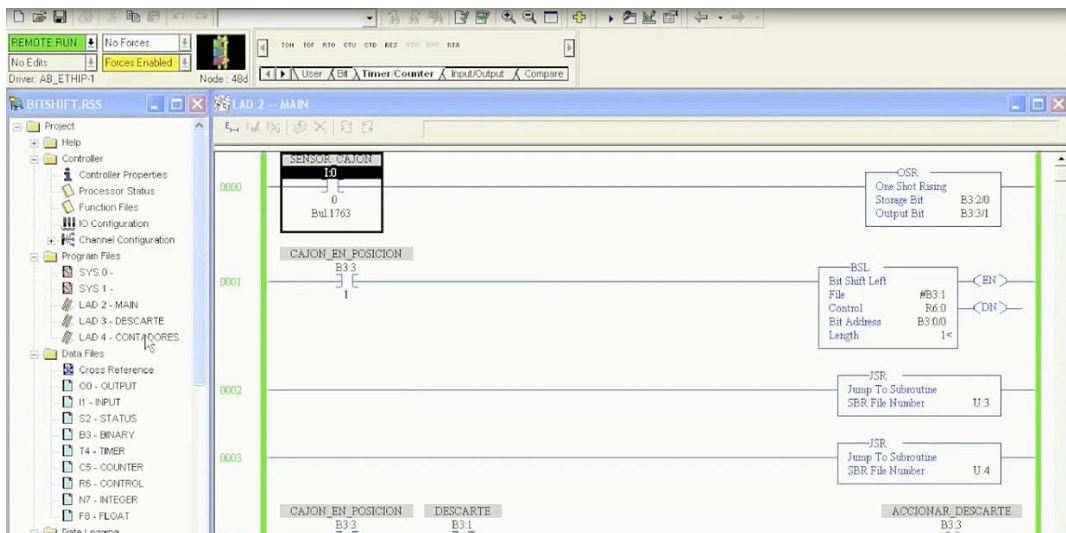


Fig.6.16 subrutinas del programa de selección por altura de piezas.

Vamos a decir ahora que queremos poner que ya haya una pieza en la posición de pieza alta el sensor de pieza alta ya está detectando, y ahora vemos como cuando volvemos a

energizar esa pieza el contador ya ha contabilizado 2 piezas y podemos observar como ya se han introducido a la palabra del almacenado de botellas los 2 unos, que representa que esos 2 bits han sido botellas altas, ahora vamos a apagar el de botella alta y vamos a poner el de botella baja, ahora al energizar el cajón o el sensor del cajón podemos observar como el contador de piezas bajas ha incrementado en 1 y al mismo tiempo se insertó en B 3 : 1 que es la palabra que estamos usando para almacena cuales botellas van a ser descartadas y cuales no, se insertó un 0 y se han desplazado los unos al lado izquierdo, nuevamente volvemos a energizar y podemos observar que se siguen desplazando.

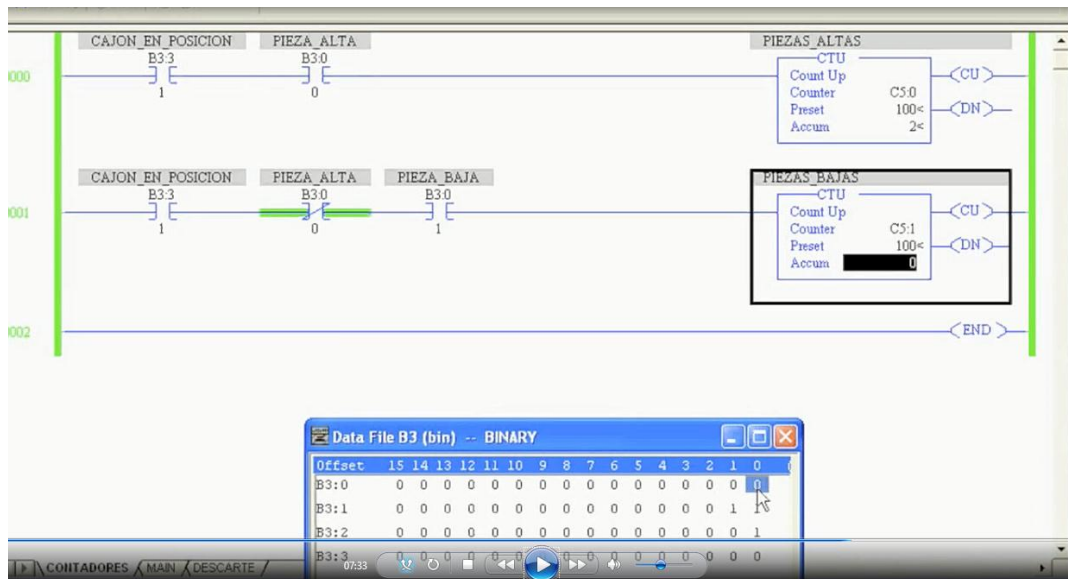


Fig 6.17 Subrutina para piezas altas y bajas.

Nuevamente ponemos en 1 el sensor de la botella alta y vemos como se están desplazando, algo importante es que empleamos ya la rutina de descarte sin perder de vista nuestros archivos de datos, observamos del diagrama de escalera como el pistón de descarte se va a energizar durante 4 segundos, dado que el valor que está en el bit 6 de la palabra B 3 : 1 que es la que estamos utilizando para este análisis es un 1, energizamos dado que el valor de B 3 : 1 es 1 se activa durante 4 segundos y ya después se apaga el pistón de descarte, nuevamente en esta ocasión el bit es 0 pues no se activa el pistón de descarte dado que solo se activa cuando es necesario y es en las botellas altas lo que se indica cuando haya un 1.

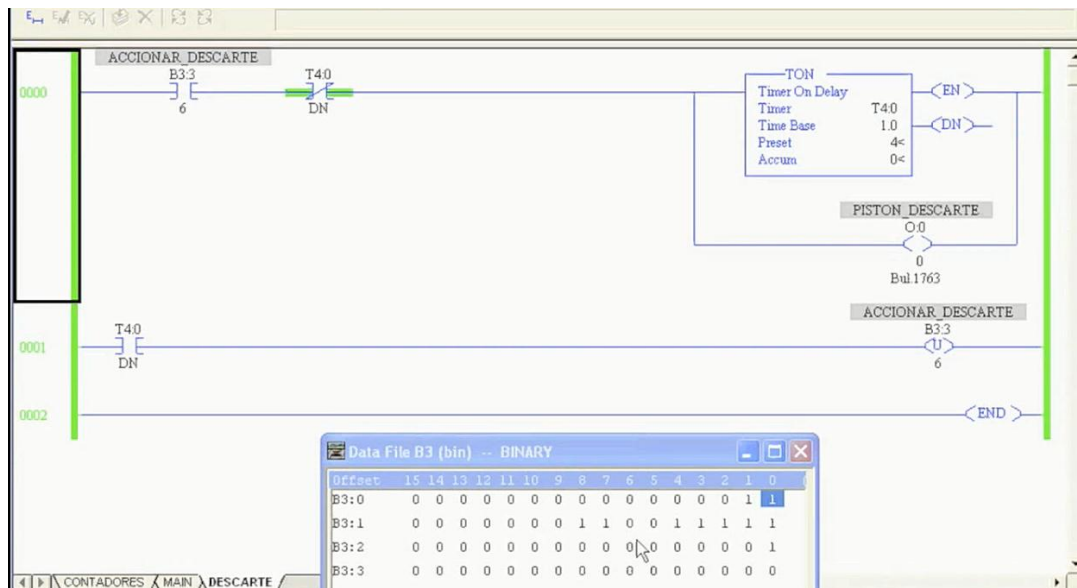


Fig 6.18 Subrutina para descartar piezas.

Nuevamente con esto como ya es un 1 ahora se vuelve a activar el pistón de descarte y podemos observar como los contadores han ido incrementando la cuenta de piezas altas y piezas bajas.

6.7 COMO PROGRAMAR

Ahora vamos a dar 5 punto para programar, utilizar los símbolos o etiquetas que van a ir asociadas a cada uno de los elementos para que no tengamos que recordarlos por su dirección. En este diagrama de escalera lo que podemos entender o inferir que lo que se está haciendo en el renglón 0 es el sistema de arranque y paro de algún motor o el enclavamiento de una señal de inicio. O bien en el renglón 1 deben estar no activados ciertos valores o ciertos bits de la palabra B 3, sin embargo si este fuera un programa mucho mas grande que tuviéramos 30 o 200 puntos y después ya no sabríamos que la palabra B 3 : 0 la estamos utilizando para almacenar que cosa, entonces simplificando los símbolos es mucho mas entendible y nos va a ayudar.

El punto 1: por ejemplo en el siguiente diagrama ya hemos colocado los símbolos y podemos darnos cuenta como es ya un poco mas entendible tenemos línea 0 que es efectivamente el paro y arranque de algún motor, y en la línea 1 estamos energizando o prendiendo la torreta roja si el estatus de la banda o el horno es apagado y si el sistema no está en operación ni tampoco hay un error en activo. Una ventaja de utilizar los símbolos es que si voy a editar una nueva línea y coloco un contacto en vez de crear B 3 : 0 para no tener la necesidad de estar recordando esos datos, lo que queremos es que ese contacto sea el horno, y vemos cuando empezamos a teclear la letras del símbolo ya me permite seleccionar el horno estados y ya no es necesario para mi recordarle el direccionamiento de la dirección a donde esta vinculado esta señal sino simplemente como.

El punto 2 es utilizar comentarios, que son campos de texto donde podemos poner o decir del diagrama algunas etiquetas que van asociadas a los renglones donde podemos explicar que es lo que está sucediendo y con ello documentar el programa. Una buena práctica de programación es utilizar comentarios ya que es el espacio donde puedes ir escribiendo texto con palabras que es lo que esta haciendo cierta parte del programa. Por ejemplo tenemos una línea amarilla que nos dice que el renglón 0 esta siendo ocupado para el paro y arranque del motor 1. por ejemplo nosotros podemos agregar a esta línea y le damos clic derecho y seleccionamos edit coment, podemos poner aquí el comentario como por ejemplo enciende la torreta roja si no hay error y la válvula esta apagada como el horno, damos aceptar y se pone una etiqueta.

A través de etiquetas es muy fácil poder documentar el código que estamos desarrollando, de tal manera que posiblemente mientras estamos programando no recordemos que es lo que estamos haciendo, pero que pasa si 6 meses después regresamos ha hacer alguna modificación, entonces es muy probable que ya no nos acordamos y tengamos que entender que hicimos que en solucionar el problema.

El punto 3 es recomendable poner en un renglón del lado izquierdo aquellas funciones o contactos que sean mas probable que sean falsos, esto es recomendable cuando estamos desarrollando proyectos dentro de RS Logic 500, es dentro de la misma línea colocar del

lado izquierdo aquellas funciones o aquellos contactos que mayormente vayan a ser falsos, en este ejemplo es mas factible que el botón de arranque sea falso que lo sea el botón de paro, ya que el botón de paro es un contacto normalmente cerrado casi siempre cuando se evalúe será verdadero. Sin embargo el botón de arranque normalmente estará en falso, al permitir que el sistema evalúe primero aquellas funciones que tengan mayor probabilidad de ser falsas lo que estamos haciendo es eficientar el tiempo de procesamiento.

Ya que cuando un renglón encuentra evaluaciones en falso de estructuras en serie pues como la primera es falsa automáticamente los demás elementos del renglón van a ser falsos, ya no se evalúa ese renglón y se continúa con el siguiente.

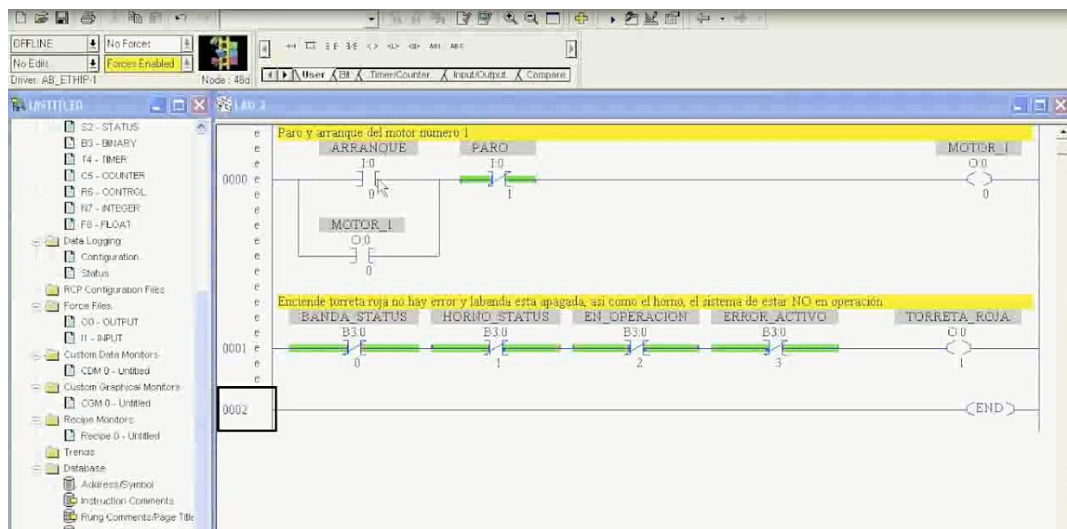


Fig 6.19 programa para dividir en subrutina.

El punto 4 es muy recomendable emplear las subrutinas de tal manera que podamos dividir el programa y sea mucho mas fácil administrarlo, programarlo, por ejemplo aquí tenemos un programa donde en el diagrama de escalera 2 tenemos el botón de arranque y estamos usando una bobina tipo latch para ese botón normalmente abierto. Cuando se presione el botón de arranque se va a energizar esa bobina y en el renglón 1 se va a cerrar el contacto en operación, y mientras la bobina o el contacto que hace referencia a que ya terminó la etapa 1 está energizado vamos a estar entrando a la subrutina 3 que hemos llamado etapa 1.

De tal manera que cuando ciertas condicionantes hagan que el proceso llegue al fin de la etapa 1, la bobina B 3 : 1 / 0 se va a poner en 1, por lo tanto este renglón va a ser falso y al momento de ser falso ya no vamos a entrar a la subrutina 3, sin embargo el renglón 2 estaremos en operación la etapa 1 ya debe de haber terminado por lo tanto es verdadero sin embargo la etapa 2 no habrá terminado y entraremos a la subrutina 4, y así dentro de la subrutina 4 cuando se termine esa etapa nos mandará en 1 el bit de la etapa 2 diciendo que ya terminó y por lo tanto en este renglón será falso, pero el renglón 3 será verdadero, lo que aquí haremos será finalizar y vemos como que la etapa 2 ya haya terminado está temporizando, al finalizar este tiempo, se desactiva la señal de en operación tanto como las dos señales que ya habían terminado.

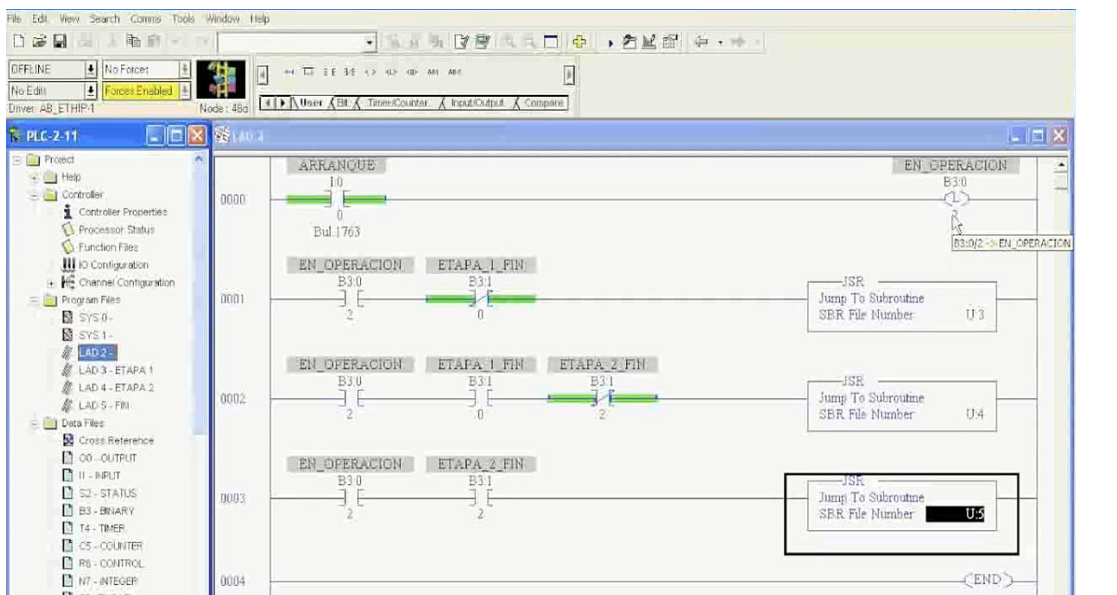


Fig 6.20 División de un programa en 3 subrutinas.

El número 5 es importante que aunque parezca muy sencillo tengamos un método de respaldo de los proyectos que hemos realizado para futuras mejoras o mantenimientos, pues al regresar a darle mantenimiento al sistema y si no guardamos ese respaldo a la hora que descarguemos del PLC el programa no contendrá ni símbolos ni comentarios, entonces puede ser un poco más tardado dar el mantenimiento.

Es recomendable tener un adecuado control de las versiones ya que con el paso del tiempo se puede volver confuso la versión adecuada.

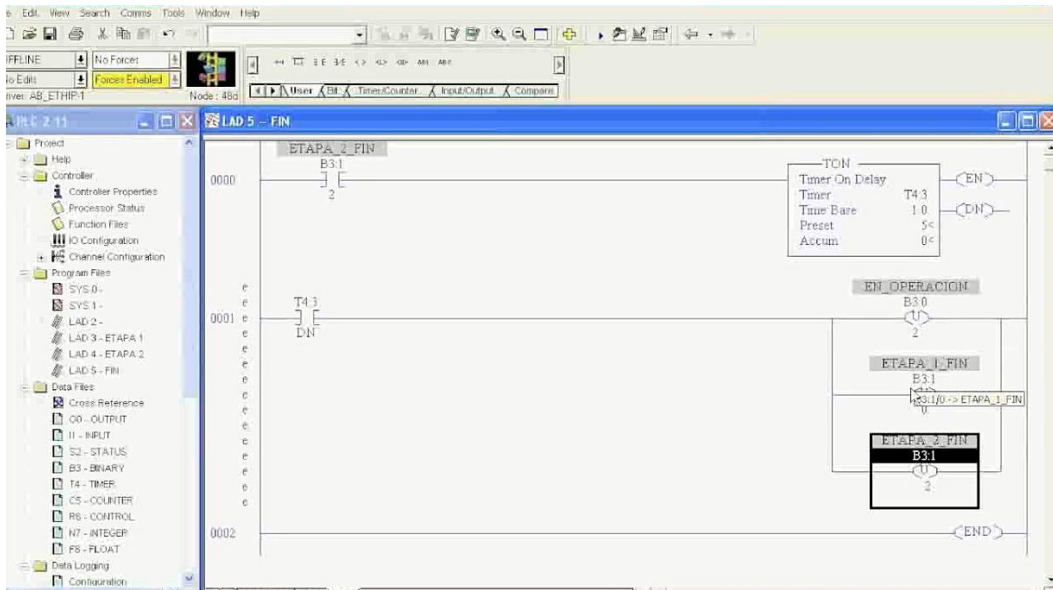


Fig 6.21 subrutina del programa.

6.8 MÁSCARAS

Este materia explica para que son las máscaras y como se utilizan en el manejo de RS Logic, en ocasiones es necesario mover una palabra a otra palabra, pero también es necesario en ocasiones mover únicamente un segmento de una palabra, para ello podemos utilizar las máscaras que no son otra cosa mas que un filtro para decidir que bit de la palabra que queremos mover queremos que sea movido.

La máscara funciona de la siguiente manera, únicamente va a permitir que sean copiados o movidos según la función los bits que en la máscara tengan un 1. en ocasiones en necesario copiar algún dato o manejar alguna palabra pero no toda la palabra, por ejemplo si tuviéramos que copiar de un B 3 : 0 a un B 3 : 1, únicamente 4 bits o media palabra, vamos a ver como podemos hacer esa parte.

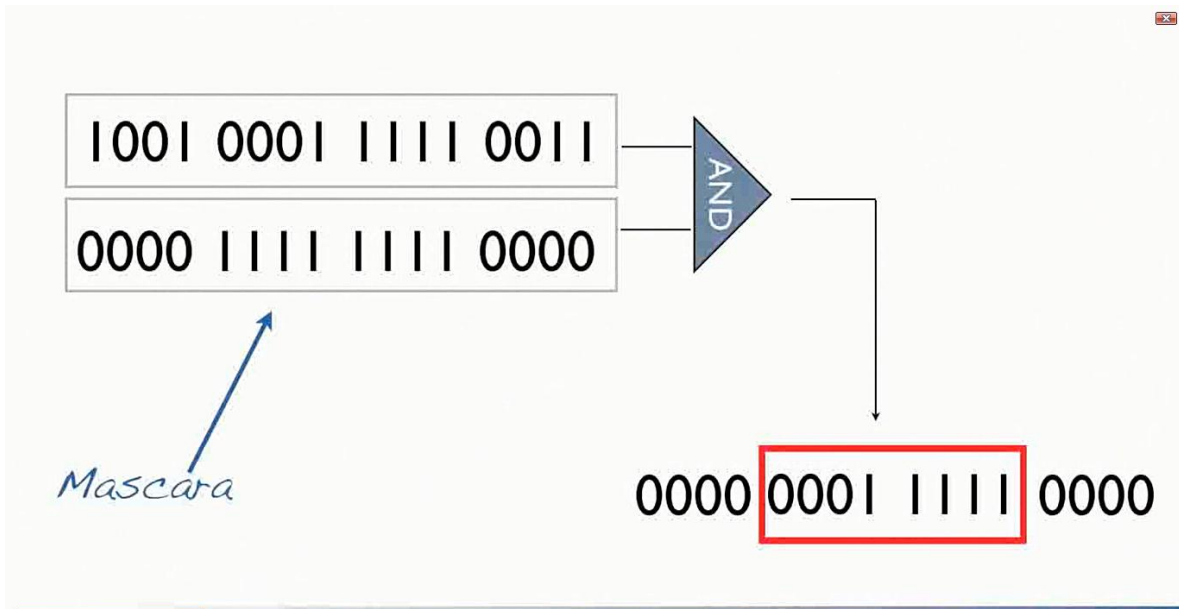


Fig 6.22 operación AND con una máscara en 2 palabras de 16 bits cada una.

Vamos a colocar un MOVE MASC si nos fijamos es una instrucción muy parecida a MOVE sin embargo ahora nos pone otra opción que es MASC, le ponemos que queremos mover los datos de B 3 : 0 y la máscara es el filtro aquí es donde vamos a establecer el filtro que queremos que se utilice. El filtro va a ser utilizando B 3 : destino es B 3 : vamos a colocar un contacto normalmente abierto para indicar cuando queremos que se haga el movimiento de los datos y este estará vinculado a la entrada física I : 0 / 0 que ahora vamos a descargar.

Y para tener una mejor idea de que es lo que está sucediendo vamos a abrir el archivo de datos B 3, lo que va a estar haciendo la función es mandar los datos de B 3 a B 2 lo que vamos a hacer es colocar algunos unos, ya hemos colocado puros unos en B 3 : 0, y ahora vamos a definir de esos bits cuales son los que si queremos que se pasen, y eso lo vamos a hacer poniendo un 1 en la palabra que vamos a remarcar, únicamente le vamos a poner que los primeros 8 dígitos de la palabra sean copiados. Vamos a necesitar el contacto, fijémonos como cuando se energiza el contacto del B 3 : 0 se han copiado al B 3 : 2 únicamente aquellos bits que la máscara permitía, es decir solo se van a copiar los bits que tengan un 1 en la máscara.

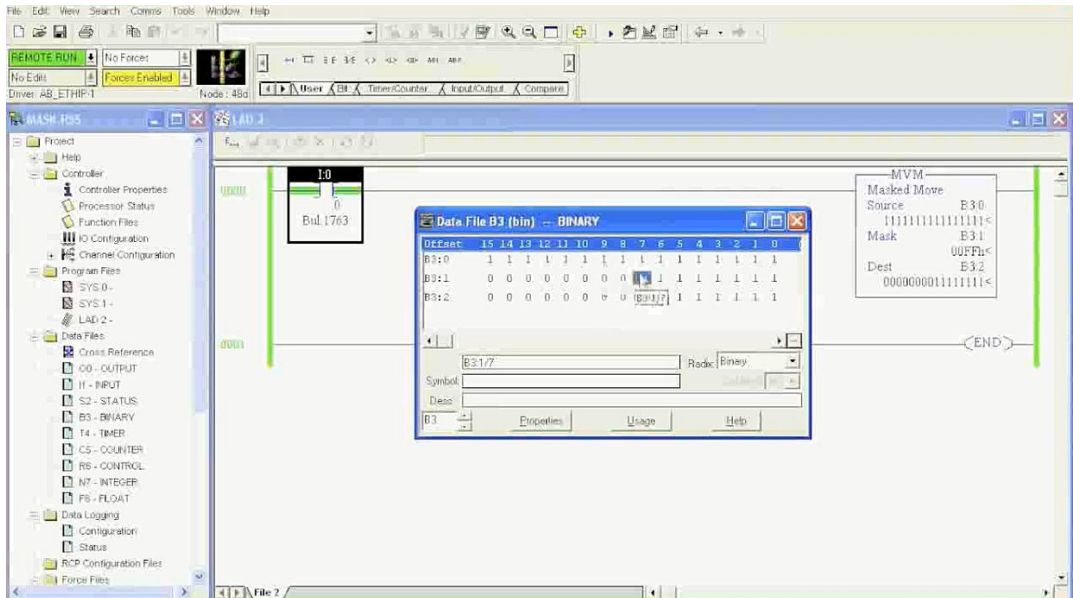


Fig 6.23 Ventana RSLogix 500 para una operación de máscara de bits.

Ahora si ponemos un 0 en la palabra original o fuente fuera de los 8 bits primeros que seleccionamos de la máscara no se copiará, es decir únicamente se van a reemplazar los bits que estén dentro de la máscara. Ahora en el siguiente ejemplo en la palabra destino vamos a poner unos, y en la palabra origen vamos a poner ceros, de igual manera lo que vamos a estar observando es que únicamente se copiarán los bits dentro de la máscara.

Ahora hemos colocado unos bits en la palabra destino pues lo que queremos observar es como cuando se energiza el contacto y se copian los datos, solamente se sustituyen aquellos que si tenían permitido ser sustituidos en los campos, y no se sustituyen aquellos datos que la máscara no los permitió. Al introducir una máscara podemos hacerlo sea direccionando a una palabra como en este caso a la B 3 : 1.

O bien escribiendo directamente en el campo en formato binario, decimal o hexadecimal, por ejemplo queremos poner una máscara donde únicamente copiamos los primeros 4 dígitos, pues ponemos 0000 0000 0000 1111 en la máscara, y una vez para indicarle al sistema que lo estaremos viendo en binario y una vez que demos enter fijémonos como lo coloca en hexadecimal.

Podemos introducirlo en binario, en decimal o en hexadecimal, pero siempre va a mostrar en hexadecimal, bien podemos afirmar solamente que si son los primeros dígitos le ponemos un 15 que es la D de decimal y nuevamente lo muestra en hexadecimal o bien si quisiera, por ejemplo si quiero seleccionar que 4 bits si pasen y 4 bits no pasen alterno, le ponemos a la máscara un F0F0, le ponemos la H para que sepamos que es en hexadecimal y nos lo va a mostrar.

Existen varias funciones en las que podemos utilizar la máscara como los secuenciadores, los movimientos y otras funciones que al utilizar la máscara podamos incrementar la capacidad de nuestros programas.

CONCLUSIONES

En toda esta tesis hemos visto como programar ejemplos típicos como llenado de botellas, control de lugares, sumadores, restadores, multiplicaciones, entre otros, sin embargo se muestran los programas típicos de enseñanza de PLC, para continuar con una formación de PLC mas en forma definitiva es necesario completar otra serie de cursos, pero ya no dictados por nosotros mismos, con los elementos básicos que hemos mostrado y que satisfacen las necesidades de una enseñanza introductoria al manejo y la programación del PLC desde sus componentes externos hasta módulos, entradas, salidas y programaciones de ejemplo.

La enseñanza básica del PLC comprende su estructura externa, CPU, tarjetas, módulos, así como de la programación del PLC con algunos programas seleccionados como básicos o de iniciación. La capacitación para una programación mas a fondo solamente puede ser obtenida directamente del ramo industrial y suele tener un coste económico alto. Aquí en la tesis solo hemos mostrado las herramientas básicas de programación.

Con esta conclusión finalizamos este trabajo de tesis que ha llegado al término establecido de un temario que cubrir en la que nosotros mismos nos contestamos las preguntas básicas del PLC acerca de la programación y abre una nueva ventana a otra serie de preguntas mas avanzadas, de las que por menos proveemos de las herramientas de abordaje necesario.

BIBLIOGRAFIA

1. AUTÓMATAS PROGRAMABLES
EDITORIAL Mc GRAW HILL
A. PORRAS I A.P. MONTANERO
2. MANUAL KEYENCE MODULO PS2-6 1
P JAPON 1995.
3. USER MANUAL SLC500 BULLETIN 1745
ALLEN BRADLEY COMPANY USA 1996.
4. HAND BOOK OF APPLIED INSTRUMENTATION CONSODININE & ROSS MC
GRAW HILL USA 1994
5. INDUSTRIAL CONTROL ELECTRONICS JOHN WEBB -@ KEVIN GRFSHHOCK
ED MERRIL USA 1992.
6. BULLETIN 1775
PLC-3 MAINTENANCE TRAINING MANUAL
ALLEN BRADLEY COMPANY USA 1988
7. MANUAL DE ENTRENAMIENTO FAMILIA PLC-2 PARTE 1
CENTRO DE ENTRENAMIENTO ALLEN BRADLEY MEXICO 1985..
8. MANUAL DE ENTRENAMIENTO FAMILIA PLC-2 PARTE 11
CENTRO DE ENTRENAMIENTO ALLEN BRADLEY MEXICO 1985.
9. INDUSTRIAL ELECTRONICS DEVICES AND SYSTEMS
PRENTICE HALL USA 1986.
10. USER MANUAL AUTOMAX2 P,ELIANCE ELECTRIC & ENGINEERING CO DE
MEXICO S.A DE C.V. MEXICO 1992.

SITIOS DE INTERNET

www.absa.com

www.allenbradley.com.