Universidad De Oriente

Núcleo de Anzoátegui

Escuela de Ingeniería y Ciencias Aplicadas

Departamento de Electricidad

Controlador Lógico Programable (PLC).

Curso Tutorial.

Realizado por:

Prof. Danilo A. Navarro G.

Trabajo Presentado Como Requisito Parcial Para Ascender a la Categoría de Profesor Agregado.

Puerto La Cruz, Julio de 2001.

os sistemas automatizados han evolucionado desde el control a relés hasta los que usan facilidades computacionales desarrolladas en los tiempos presentes. Actualmente el corazón del desarrollo de los sistemas automáticos lo representa esencialmente los Controladores Lógicos Programables (PLC). Cotejar la idea que tiene cada estudiante o profesional del área de automatización respecto a lo que es un PLC arrojaría un cúmulo de ideas distintas dependiendo por supuesto del conocimiento previo y el paradigma que cada individuo se haya formado respecto a este mismo tema. En este trabajo no se trata de unificar estas ideas o conceptos, sino más bien se trata de enriquecer el conocimiento que estudiantes y profesionales poseen hasta este momento.

De manera especial para estudiantes y profesionales del área de eléctrica, electrónica, computación y sistemas; es de primera necesidad tener conocimientos y un mediano entrenamiento en el tópico de los PLCs, ya que estos forman parte fundamental en sistemas de control, de adquisición de datos y de manejo de la información de un gran número de procesos y máquinas que se encuentran en su campo de trabajo.

Aunque existen textos (muy escasos) especializados en la temática de los PLCs, para no dejar el entrenamiento en lo meramente teórico, es necesario que el aprendiz acuda a los cursos de entrenamiento dictados por los propios fabricantes de los equipos. Estos cursos además de ser costosos, los ofrecen muy esporádicamente a grupos especiales, y por lo general son poco amplios ya que tratan a un determinado tipo o marca de PLC.

Este trabajo combina los aspectos teóricos del software y del hardware de los PLCs, con paquetes computacionales de simulación para ofrecer al interesado la posibilidad de cumplir con un programa de auto entrenamiento que lo prepare mejor para ejecutar trabajos en el área de los PLCs.

También, el rápido avance de los sistemas de control distribuido no es posible sin las facilidades de comunicación con las que cuentan los PLCs. De aquí que como parte final de este trabajo, en el capítulo 7, se detalle en forma básica la interfase de comunicación más comúnmente usada por los PLCs: La RS-232.

n PLC (Controlador Lógico Programable) es un dispositivo electrónico de estado sólido que puede controlar un proceso o una máquina y que tiene la capacidad de ser programado o reprogramado rápidamente según la demanda de la aplicación. Fue inventado para remplazar los circuitos secuenciales basados en relés que eran necesarios para el control de las máquinas. El PLC funciona monitoreando sus entradas, y dependiendo de su estado, activando y desactivando sus salidas. El usuario introduce al PLC un programa, usualmente vía Software, lo que ocasiona que el PLC se comporte de la manera deseada.

Fig. 1 – Aspecto físico de un PLC.

Los PLCs son usados en muchas aplicaciones: Maquinado de piezas, Embaladoras, Manipulación de materiales, ensamblado automático, y en general cualquier tipo de aplicación que requiera de controles eléctricos puede usar más bien un PLC.

Fig. 2 – Aplicación típica de un PLC.

Por ejemplo asúmase que cuando un switch se activa, deseamos también activar una válvula solenoide por un período de 5 segundos y luego apagarla sin importar el tiempo que el switch estuvo activado. Esto se puede hacer con un simple temporizador externo; pero, ¿Qué tal si el proceso incluye 10 switch y 10 selenoides?. También, ¿Qué si en el proceso es necesario contar cuantas veces cada switch se activo?. Obviamente se necesitarían una gran cantidad de contadores externos. Como se ve, mientras más grande el proceso, mayor es la necesidad de un PLC, y por ejemplo en el caso descrito bastaría con simplemente programar el PLC para que cuente sus entradas y mantenga activadas sus salidas por un cierto período de tiempo.

os PLCs fueron introducidos por primera vez a finales de 1960. La razón principal para introducir tal dispositivo fue la de eliminar el gran costo que representaba remplazar los sistemas de control basados en lógica de relés. En 1968, una expresa consultora llamada Bedford Associates (Bedford, MA) diseño para la General Motors un dispositivo de control que llamaron Controlador Digital Modular (Modular Digital Controller, MODICON) 084. Otras compañías al mismo tiempo propusieron esquemas de control basados en computadoras, uno de los cuales se basó en el PDP-8. El MODICON 084 representó el primer PLC en el mundo dentro de la producción comercial.

La razón principal que impulsó este nuevo tipo de control fue que cuando cambiaba los requerimientos de producción, también lo hacia el sistema de control, y esto se tornaba costoso sobre todo cuando los cambios eran frecuentes. También, como los relés son elementos mecánicos, ellos tienen un período de vida limitado y además requieren de un estricto programa de mantenimiento. Igualmente, la resolución de problemas en la lógica de control era muy tediosa sobre todo cuando estaban involucrados gran cantidad de relés; y los paneles de control de las máquinas incluían cada vez más funciones que si se utilizaba lógica a relés, estos incluirían cientos de ellos, lo que ocasiona el problema inicial del difícil cableado de los paneles.

Estos nuevos controladores también tenían que ser fáciles de programar por los ingenieros de planta y de mantenimiento. El tiempo de vida tenia que ser largo y los cambios en la programación de las funciones debía ser fácilmente realizable. También, los nuevos controladores debían poseer cualidades para resistir a los severos ambientes industriales. La respuesta a este lote de planteamientos era usar técnicas de programación que ya le fueran familiares a los técnicos de plantas(diagramas de contacto: LADDER) y a la par remplazar los relés electromecánicos por unos que fueran de estado sólido.

A mediados de los 70 la tecnología dominante en los PLCs eran las maquinas secuenciadoras de estados y los bit-Slice based CPU. El AMD 2901 y 2903 eran bastante populares en los PLCs de Allen Bradley y en los de MODICON. Los microprocesadores convencionales carecían de la potencia para satisfacer los requerimientos de lógica en todo los PLCs excepto en los más pequeños. Según como los microprocesadores convencionales evolucionaron, en esa misma medida se construyeron PLCs cada vez más grandes y potentes.

Las posibilidades de comunicación comienzan a aparecer aproximadamente en 1973. El primero de tales sistemas fue el ModBus de MODICON. Los PLCs pueden a partir de aquí comunicarse con otros PLCs distantes e intercambiar con ellos datos de las máquinas controladas. Igualmente se pueden usar para enviar y recibir voltajes variables lo que les permite entrar al mundo analógico. Desdichadamente, la carencia de estandarización acoplada con los continuos cambios tecnológicos hicieron la comunicación entre los PLCs un mar negro de redes y protocolos incompatibles.

En los 80 se vio el intento por estandarizar las comunicaciones con el Protocolo de Automatización de la Manufactura de la General Motors (MAP). En este tiempo también se redujo el tamaño de los PLCs y se hicieron programables mediante la programación simbólica desde computadores personales PCs en vez de mantener los terminales de programación dedicados o programadores "handheld". Hoy, el PLC más pequeño en el mundo es del tamaño de un simple relé de control.

En los 90 se vio una gradual reducción en la introducción de nuevos protocolos, y en la modernización de las capas físicas de alguno de los más populares protocolos que sobrevivieron a la década de los 80. El ultimo standard (IEC 1131-3) ha tratado de unificar los lenguajes de programación de los PLCs bajo un único standard internacional. Actualmente hay PLC que son programables en diagramas de Bloques de Funciones, Lista de instrucciones, “C++” y texto estructurado, Diagrama de Contactos(LADDER) y GRAFCET al mismo tiempo.

Fig. 3 – Tipos de memorias en un PLC.

- Capacidad modular de entradas / salidas**.** Esto permite la combinación de distintos niveles y tipos de señal de entrada, así como también el manejo de salidas para distintos tipos de carga. Igualmente si la aplicación crece, y se requiere mayor número de entradas / salidas, casi sin ningún problema los PLCs pueden adecuarse al nuevo requerimiento.

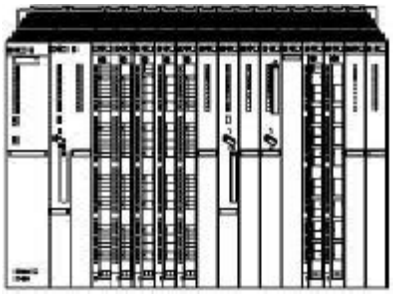


Fig. 4 – Capacidad modular de los PLCs.

- Autodiagnóstico de fallas. El PLC monitorea el funcionamiento de su CPU, Memoria y circuito de interfases de entrada y de salida, e igualmente

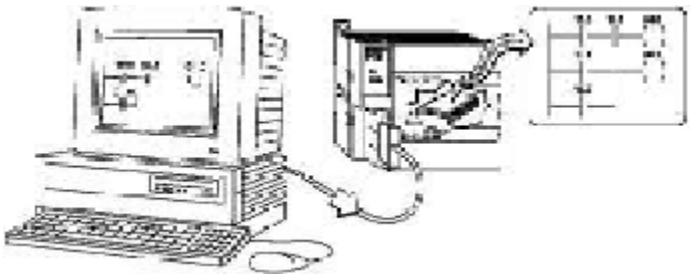


Fig. 6 – Lógica programada.

- Capacidad para generar reportes y comunicarse con otros sistemas**.** Con esta facilidad se pueden integrar interfaces de explotación Hombre-Máquina, sacándole al sistema mayor cantidad de información. Igualmente los PLCs pueden participar en redes de datos comunicándose con otros PLCs para formar sistemas de control distribuidos, o integrándose a las redes administrativas de la producción.

Alta confiabilidad. Elementos de estado sólido ❖ Múltiples contactos NO, NC ❖ Consumo de energía reducido ❖ Reducción del costo a medida que aumenta la complejidad del proceso	⌦ Máximo de 4 a 6 contactos ⌦ Mayor consumo de energía ⌦ A partir de 15 o 20 relés, el costo comparativo supera el costo con PLCs

Control Microsystems

Crouzet Automatismes

Control Technology Corporation

Cutler Hammer/IDT : http://www.ch.cutler-hammer.com/

Divelbiss

EBERLE gmbh

Elsag Bailey

Entertron

Festo/Beck Electronic

Fisher & Paykel

Fuji Electric

GE-Fanuc : http://www.gefanuc.com/

Gould/Modicon : http://www.modicon.com/

Grayhill

Groupe Schneider

Hima

Omron : http://oeiweb.omron.com/

Opto22

Pilz

PLC Direct/Koyo/AutomationDirect : http://www.automationdirect.com/

Reliance

Rockwell Automation : http://www.automation.rockwell.com/

Rockwell Software : http://www.software.rockwell.com/

SAIA-Burgess

Samsung

Schleicher : http://www.schleicher-de.com/

Schneider Automation : http://www.schneiderautomation.com/

Sharp

Siemens : http://www.aut.sea.siemens.com/

Sigmatek

Sixnet

SoftPLC/Tele-Denken : http://www.softplc.com/

os PLCs constan principalmente de un CPU, área de memoria, y circuitería apropiada de entrada /salida de datos. Se puede considerar al PLC como una caja llena de cientos o miles de Relés independientes, contadores, temporizadores y locaciones para almacenamiento de datos. Estos contadores, temporizadores, etc.; ¿realmente existen?. No, Ellos no existen físicamente pero en vez de eso son simulados y se pueden considerar como contadores, temporizadores, etc. hechos a nivel de software. También los Relés internos son simulados mediante bits en registros del hardware del PLC.

Fig. 3 – Estructura general simplificada de un PLC.

- **Relés DE ENTRADA:** Están conectados al mundo externo. Físicamente existen y reciben señal de los switches, sensores, etc. Típicamente no son relés pero si son transistores que funcionan como relés estáticos.

- **Relés INTERNOS:** Estos no reciben señal desde el mundo exterior ni existen físicamente. Ellos son relés simulados y permiten al PLC eliminar los relés externos. También hay relés especiales que el PLC usa para realizar una tarea única. Algunos están siempre activados mientras que otros su estado

- **Relés DE SALIDA:** Estos se conectan al mundo exterior al PLC. Físicamente existen y funcionan enviando señales de encendido / apagado a selenoides, luces, etc. Basados en hardware, pueden estar construidos con transistores, relés electromecánicos o TRIACS, según el modelo que se escoja.
- **ALMACENAMIENTO DE DATOS:** Típicamente hay registros del PLC que están asignados al simple almacenamiento de datos. Usualmente se usan para almacenamiento temporal para manipulación matemática o de datos. También son usados para almacenar datos cuando se corta el suministro de energía al PLC. Una vez regresa la energía, los registros disponen de los mismos datos que tenían cuando se corto la energía.

Un diagrama de bloques más completo que describe la estructura de un PLC sería el siguiente.

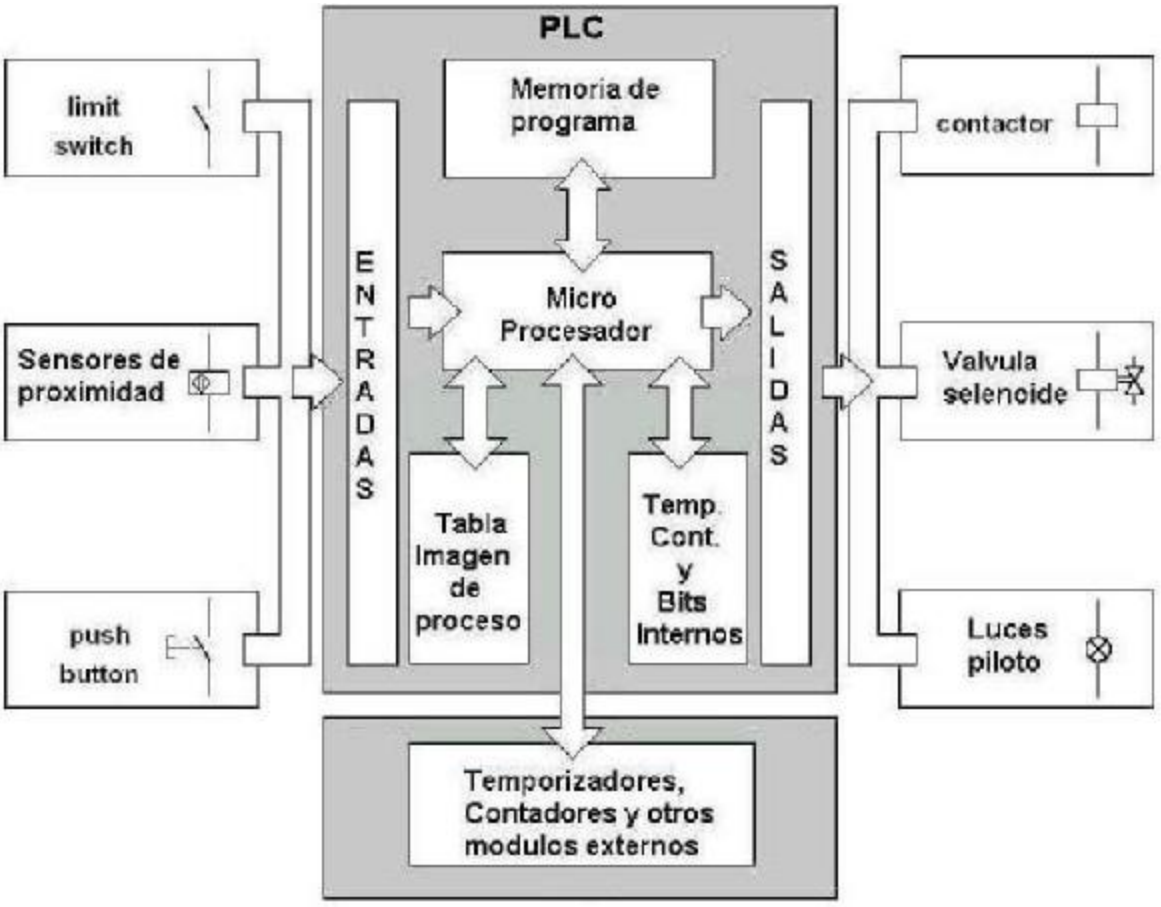


Fig. 4 – Estructura general de un PLC.

- **Unidad Central de procesamiento (CPU)**: Esta formada por la unidad de control, la tabla imagen de proceso, y por los temporizadores, contadores y bits internos. La CPU se encarga del tratamiento de los datos internamente (sumas, operaciones lógicas, transferencias, etc), busca o escribe operandos en la memoria, lee o escribe datos en las unidades de entrada y salida, etc.

- **Memoria:** Es la circuitería electrónica capaz de almacenar el programa de aplicación escrito por el usuario, y los datos provenientes de la máquina o proceso controlado. También es la encargada de almacenar las variables internas generadas por la CPU y las variables de salida a ser transferidas a los periféricos.

Corresponden a la circuitería de entrada / salida del PLC, o lo que representa lo mismo: su comunicación con el proceso o máquina a controlar y con el usuario u operador del sistema. Las señales de entrada provenientes de los sensores son de naturaleza diversa: Voltaje AC, Voltaje DC, Corriente, señales binarias, señales analógicas, etc. Es así como los periféricos son los encargados de convertir estas señales a información capaz de ser interpretada por la CPU, y de convertir las señales provenientes de la CPU a señales capaces de excitar los preaccionadores de las máquinas.

Fig. 6 – Ciclo de trabajo de un PLC.

Paso 1-**DIAGNÓSTICO INTERNO:** En este paso el PLC revisa su circuitería interna en busca de defectos de entradas, salidas, CPU, memorias y batería. También revisa el WATCHDOG y los desbordamiento de memoria para revisar fallas en el programa de aplicación.

Paso 2-**CHEQUEAR EL ESTADO DE LAS ENTRADAS:** Al principio el PLC accede cada una de las entradas para determinar si están activadas o desactivadas (on / off). Es decir, ¿ Esta activado el sensor conectado a la primera entrada?, ¿El segundo?, ¿El tercero? … Luego el PLC graba estos datos en la tabla imagen de proceso para usarlos en el próximo paso.

Paso 3-**EJECUTAR EL PROGRAMA DE LA APLICACIÓN:** El PLC ejecuta el programa de la aplicación creada por el usuario una instrucción a la vez. Por ejemplo, si el programa especifica que si la primera entrada esta “on” se debe activar la salida numero 2, el PLC graba este resultado para tomarlo en cuenta en el próximo

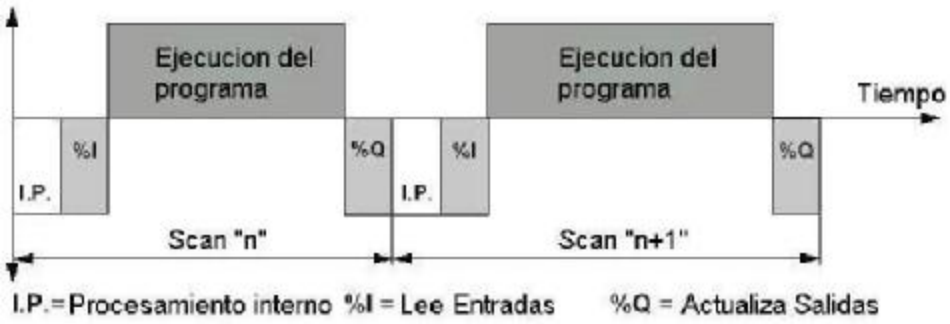


Fig. 7 – Esquema de tiempos relativos dentro del SCAN del PLC.

Chequea para verificar que todas las tarjetas estén libres de falla, restaura el perro de guardia(WATCHDOG TIMER), etc. (El “WATCHDOG” causará un error e interrumpirá el funcionamiento del PLC sino es restaurado dentro de un período corto de tiempo. Esto indicaría que la lógica del programa no esta siendo escaneada normalmente).

Barrido de Entradas: Lee los valores de entrada disponibles en los chips de las tarjetas de entrada y copia sus valores en la memoria. Esto hace al PLC más rápido y evita casos donde una entrada cambia entre el principio y el final del programa. Existen también funciones especiales de los PLCs que leen las entradas directamente y evitan el uso de las tablas de imagen.

Ejecución de la Lógica: Basado en la tabla de imagen de entradas, el programa es ejecutado un paso a la vez, y al mismo tiempo se va actualizando en memoria la tabla de imagen de salida.

Fig. 9 – Error en el barrido de entradas.

En el diagrama, la entrada 1 no es vista sino hasta el scan 2. Esto se debe a que cuando la entrada 1 esta en alto, el scan 1 ya ha finalizado su tiempo de chequeo de entradas. La entrada 2 no es vista sino hasta el scan 3 por la misma razón anterior. La entrada 3 nunca es vista o validada como activa ya que cuando el scan 3 estaba haciendo el barrido sobre las entradas, la misma no estaba aun en on, y además cambia a off antes de que el scan 4 active su tiempo de chequeo de entradas. Por tanto, la entrada 3 nunca es vista por el PLC.

Para evitar lo que ocurre con la entrada 3 del ejemplo anterior, se debe establecer que la entrada este activa al menos por **1 tiempo de barrido de entradas + un tiempo de barrido de programa o Scan.**

Fig. 10 – Tiempo de entrada activa.

Función interrupción: Esta función interrumpe el scan para procesar una rutina especial que el usuario haya programado. Esto es que tan pronto como la entrada se activa, sin importar en que parte del scan este, el PLC inmediatamente para lo que esta haciendo y ejecuta una rutina de interrupción. (Una rutina puede ser interpretada como un mini programa aparte del programa principal). Después de realizar la rutina de interrupción, el PLC regresa al mismo punto donde dejo el hilo principal y continúa el proceso normal del scan.

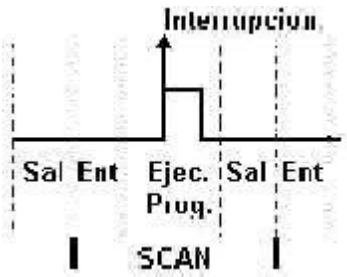


Fig. 12 – Interrupción.

Las entradas discretas, también conocidas como entradas digitales, son las que poseen dos estados: ON u OFF. Provienen de Pushbottons, detectores de proximidad, interruptores de posición, etc. En la condición de ON, una entrada discreta puede ser llamada como un 1 o como un ALTO, mientras que en la condición de OFF se conoce como un 0 o como un BAJO.

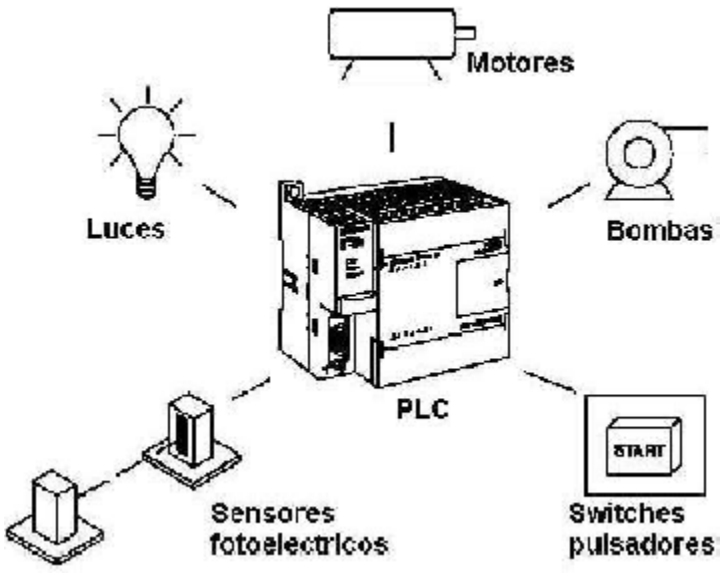


Fig. 14 – Entradas – Salidas discretas al PLC.

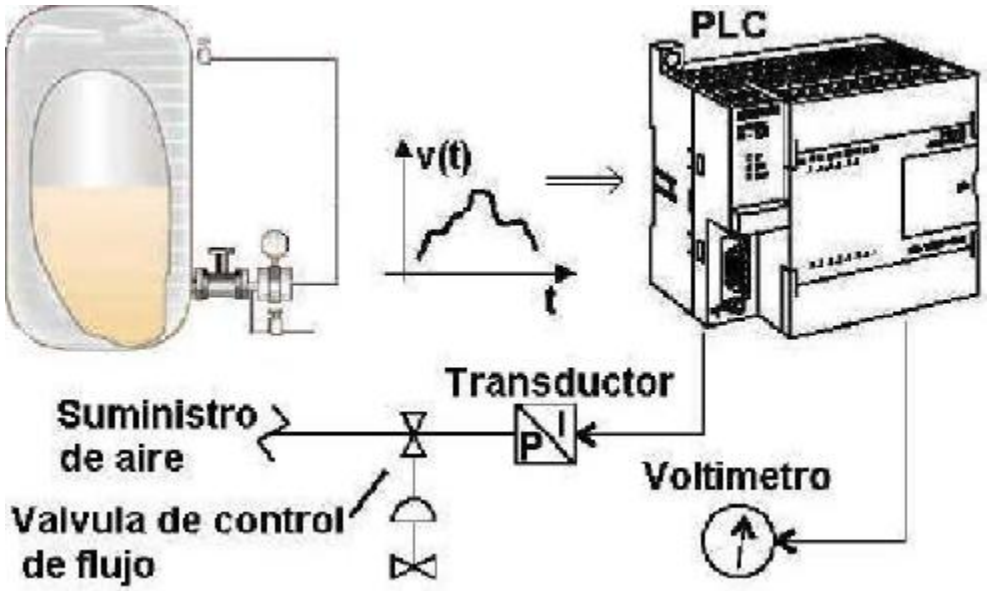


Fig. 15 – Entradas – Salidas analógicas al PLC.

Las salidas analógicas son señales de corriente o voltaje continuo. Pueden ser tan simples como un nivel de 0 a 10 voltios que maneje un voltímetro analógico, o un poco más complejas como señales de corriente que manejen convertidores corriente - presión de aire que a su vez sirvan a actuadores como lo son Servo válvulas para el control de flujo. Igualmente, con la interfase adecuada, servirían a otros tipos de actuadores dentro de esos mismos procesos como lo son: servomotores, controles de potencia de hornos, etc.

24 Vac

El PLC debe convertir esta variedad de niveles lógicos de voltaje a niveles de voltaje de lógica TTL (5 Vdc). Para lograr esto utiliza dos interfaces circuitales típicas: DC a TTL, y AC a TTL.

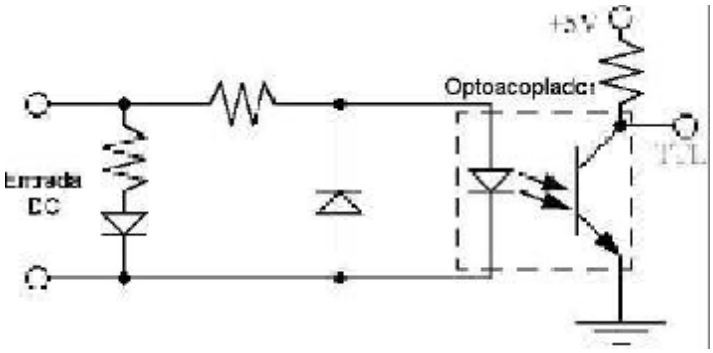


Fig. 16 – Interfase típica de conversión DC a TTL.

Los optoacopladores son usados para aislar la circuitería interna de las tensiones de alimentación externas. Esto elimina la posibilidad de que cualquier voltaje dañino o cualquier ruido alcance los circuitos lógicos internos del PLC. Los optoacopladores convierten la señal eléctrica de corriente o voltaje a una señal

Las entradas DC son muy rápidas. Las entradas AC requieren de un tiempo mayor para ser reconocida.

- Los voltajes DC pueden ser conectados a una gran variedad de equipos y sistemas eléctricos.
- Las señales AC son más inmunes al ruido que las señales DC, por eso pueden cubrir mayor distancia y ambientes ruidosos.
- El suministro AC es más fácil y menos costoso al momento de alimentar equipos eléctricos.
- Las señales AC son muy comunes en muchos equipos de automatización.

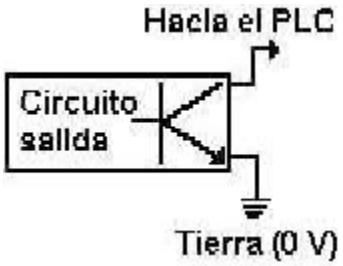


Fig. 17 – Etapa de salida de un sensor tipo NPN (Sumidero).

En este tipo de sensor se conecta uno de los terminales de salida al PLC, mientras que el otro se conecta a la referencia de la fuente de alimentación. Si el sensor no es alimentado de la misma fuente que alimenta al PLC, los negativos de ambas fuentes deben unirse entre sí para formar un terminal de referencia común. Los sensores NPN son de uso común mas que todo en Norte América.

En los sensores tipo PNP se conecta uno de los terminales de salida al positivo de la fuente, mientras que el otro se conecta a la correspondiente entrada del PLC. Si el sensor no es alimentado por la misma fuente que alimenta al PLC, se deben conectar ambos V+'s entre sí. Los sensores tipo PNP son más comúnmente usados en Europa.

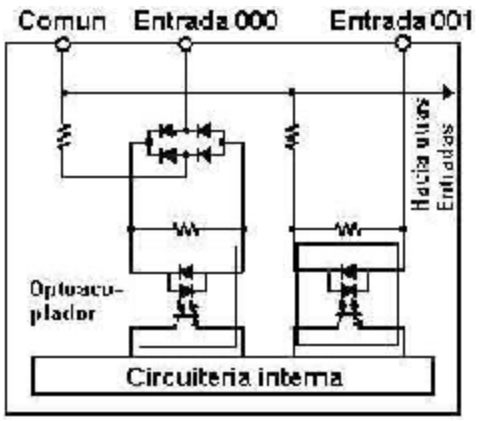


Fig. 19 – Interfase DC de entrada al PLC.

En el PLC, lo único accesible al usuario son los terminales nombrados como COMÚN, ENTRADA 0000, ENTRADA 0001, ENTRADA xxxx... El terminal común debe conectarse a V+ o a tierra, dependiendo del tipo de sensor que se este usando. Cuando se usan sensores tipo NPN el terminal se conecta a V+, mientras que cuando se usan sensores tipo PNP el terminal común se conecta a 0V (tierra).

Un switch ordinario como por ejemplo un limit switch, pushbutton, selector, etc; debe ser conectado a las entradas del PLC de una manera similar a la conexión de los sensores descritos anteriormente. Un terminal del switch debe ser conectado directamente a V+, mientras que el otro se debe conectar a la entrada del PLC, si se asume compatibilidad PNP. Es decir esta conexión asume que el común esta

Salida tipo TTL: Transistor Transistor Logic.

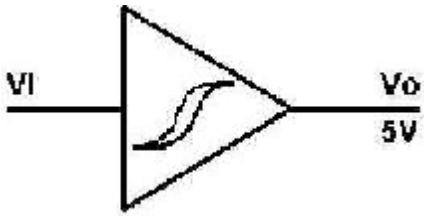


Fig. 21 – Salida tipo TTL.

- Salida DC tipo Sumidero: Conmutan corriente a tierra.

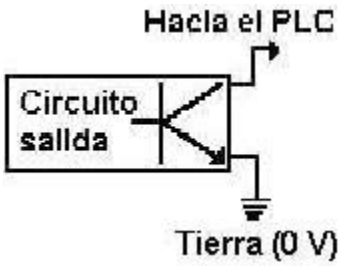


Fig. 22 – Salida tipo NPN.

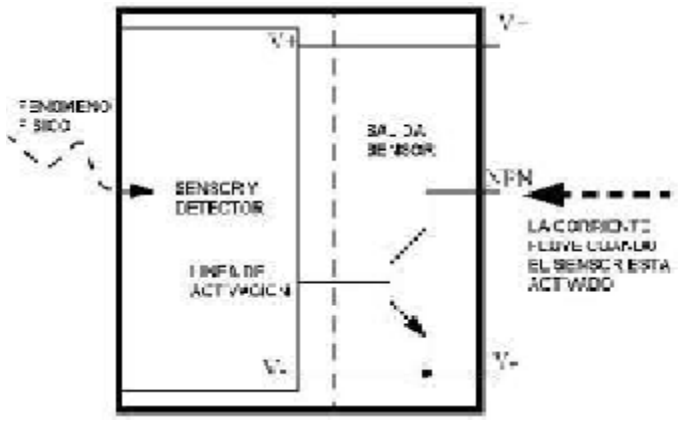


Fig. 24 – Sensor NPN simplificado.

Si el sensor ha detectado algún fenómeno, activa el transistor permitiendo así el flujo de corriente hacia el común.

Cuando se tiene una tarjeta de entrada que tiene un +V (no un común), entonces se pude usar sensores tipo NPN. En este caso la corriente sale de la tarjeta (Fuente) y el sensor la conmuta a tierra.

Cuando el sensor detecta un cambio lógico, permite que desde el +V salga una corriente que fluyendo a través de él, active una carga o la entrada de un PLC. Complementario al sensor tipo sumidero, la salida de un sensor tipo fuente consta de un transistor tipo PNP. En forma simple estos sensores se conocen como tipo PNP, e igualmente necesitan de una fuente de alimentación para poder operar.

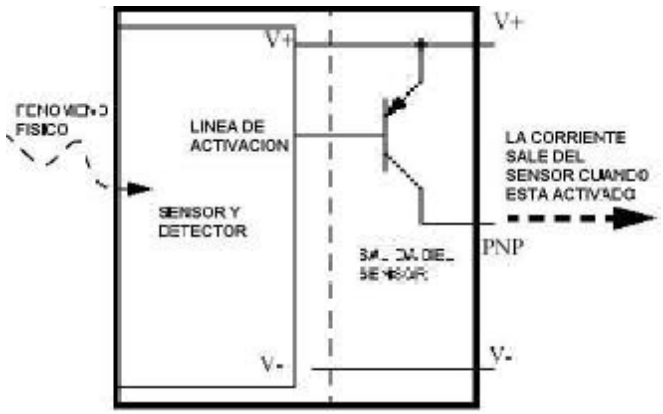


Fig. 26 – Sensor tipo PNP simplificado.

Si el sensor ha detectado algún fenómeno, activa el transistor permitiendo así que el flujo de corriente salga por el transistor.

Cuando se tiene una tarjeta de entrada que tiene un COM (común), entonces se pude usar sensores tipo PNP. En este caso la corriente fluye hacia la tarjeta buscado el común de la fuente de alimentación.

Los sensores NPN o PNP a dos hilos se han hecho populares ya que ellos reducen el cableado en las aplicaciones de PLCs. Un sensor a dos hilos puede ser usado como Fuente o como sumidero. Necesita sólo de una pequeña corriente para mantenerse polarizado, auque cuando se activa permite un mayor flujo de corriente.

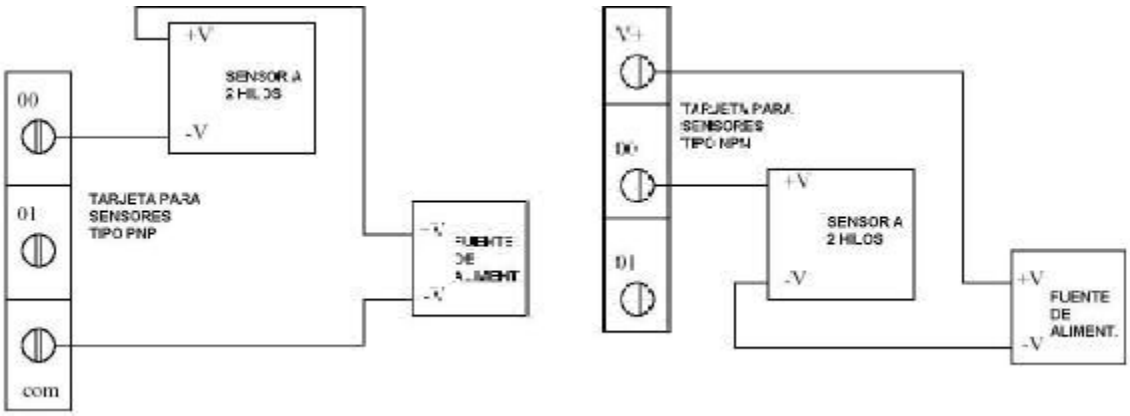


Fig. 28 – Conexión de sensores de 2 hilos tipo PNP y NPN.

Finalmente, la conexión de sensores tipo 2 hilos requiere que la tarjeta de entrada de los PLCs permita una cierta corriente de fuga, que sería la necesaria para que el sensor opere en stand-by.

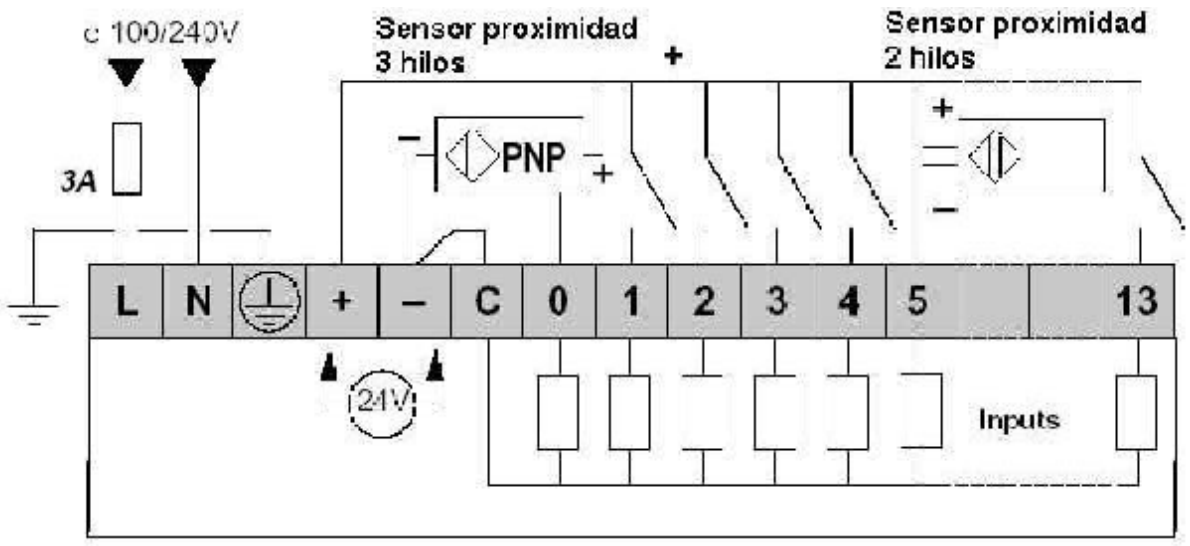


Fig. 30 – Resumen de conexión de sensores DC tipo PNP.

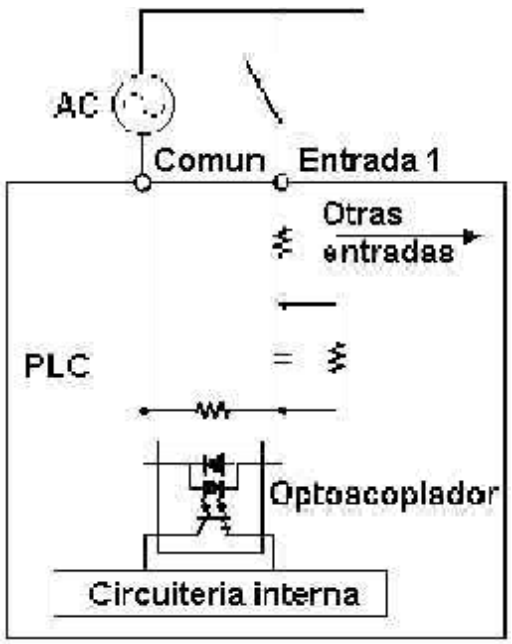


Fig. 31 – Entrada AC al PLC.

La conexión típica de los elementos de entrada AC a los módulos del PLC se muestra en la figura de arriba. Comúnmente la línea activa (fase) se conecta a los switch, mientras que el neutro se conecta a la entrada común del PLC. El terminal de aterramiento de la red AC debe ser conectado a la carcasa del PLC.

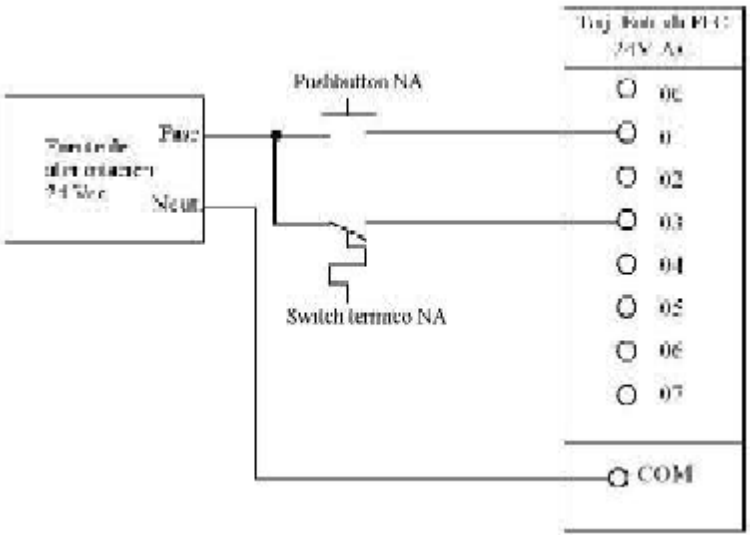


Fig. 32 – Tarjeta de entrada AC típica.

La tarjeta de entrada compara el voltaje con la referencia (COM), si está dentro de cierto rango, la entrada se activa. En este caso, el neutro es el punto de referencia de voltaje(COM), por lo que sí existen otras fuentes hay que unir todos los neutros. También, se debe tomar en cuenta que **Tierra NO es igual a COM**, el aterramiento es usado para prevenir "shocks" eléctricos y daños de los equipos.

5 VDC (TTL)

Los módulos de salida normalmente tienen de 8 a 16 salidas de un mismo tipo: a relés, a transistores, o a TRIACs. Los PLCs deben convertir los niveles lógicos TTL (5 VDC) presente en el bus de datos a niveles de voltaje externos. Esto se logra con el uso de circuitos de interfase como los mostrados a continuación, los cuales además de usar básicamente un optoacoplador para conmutar la circuitería externa, también utilizan algunos componentes para proteger la circuitería de voltajes excesivos y de polaridad inversa.

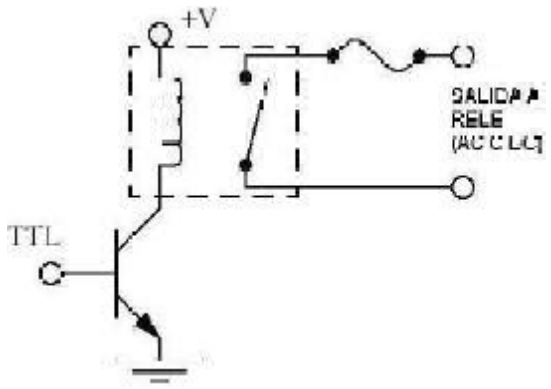


Fig. 33 – Interfase típica para salida a relé.

Fig. 34 – Interfase típica para salida a transistor.

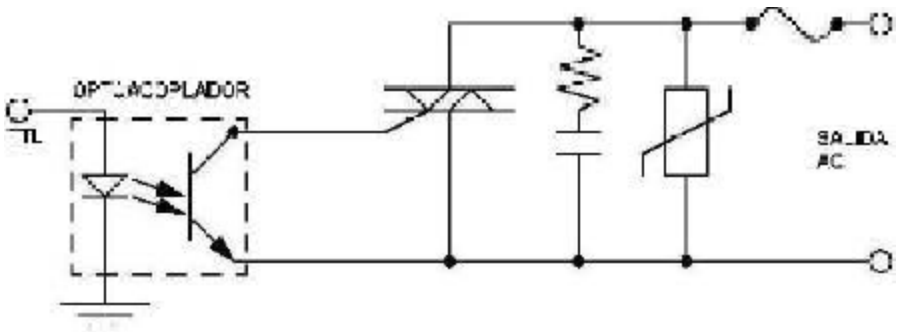


Fig. 35 – Interfase típica para salida a TRIAC.

Las salidas a transistores usan del tipo NPN o PNP y están limitadas a servir cargas DC hasta 1 amp, mientras que las salidas a TRIACs sirven a cargas AC típicamente hasta 1 Amp. Las salidas a transistores o a TRIACs son conocidas como salidas conmutadas o salidas estáticas, y su tiempo de conmutación esta normalmente por debajo de 1 ms..

Existe un tipo de carga a las que se le debe prestar especial atención: las llamadas cargas inductivas. Este tipo de carga tiene la tendencia de desarrollar una sobrecorriente al energizarlas, y lo que es peor, desarrollan un sobre impulso de voltaje inverso cuando son desactivadas. Esta corriente y voltaje inverso propensa el daño de la salida a relés del PLC. Típicamente se deben usar diodos, varistores o circuitos "snubber" para ayudar a combatir el daño de los relés de salida del PLC.

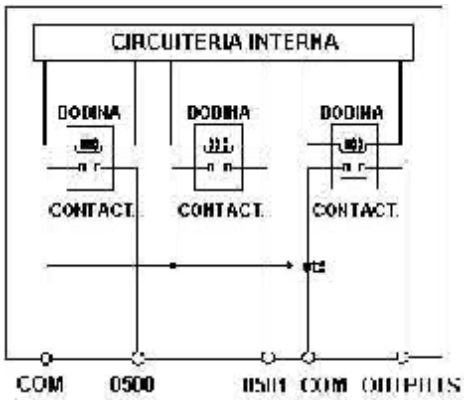


Fig. 36 – Módulo de salidas a relés.

Los relés de salida están dentro del PLC. La figura de arriba muestra un diagrama circuital típico de las salidas a relés. Cuando la lógica del programa de aplicación indica que se debe activar una salida física, entonces el PLC aplica un

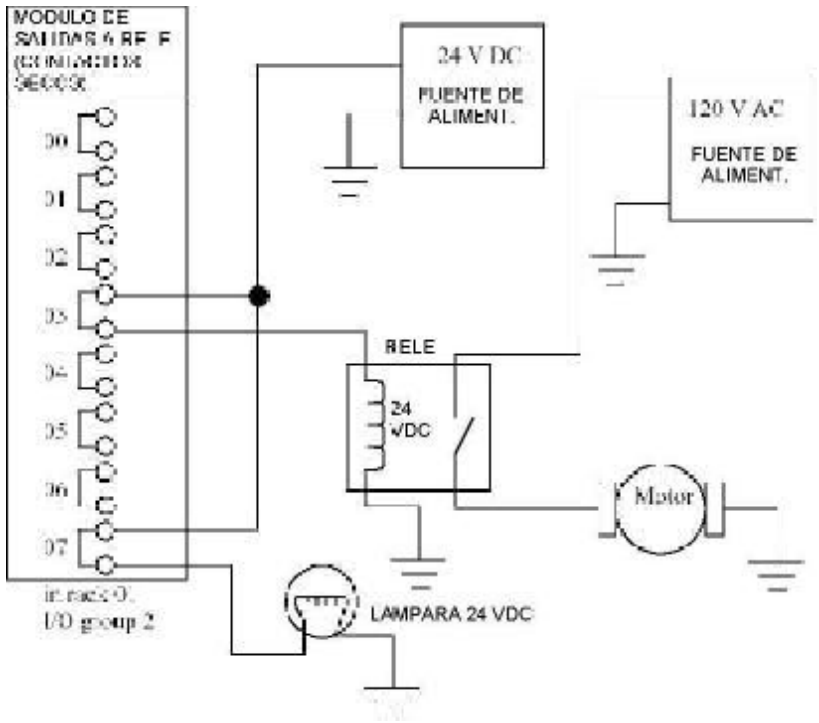


Fig. 37 – conexión típica de un módulo de salidas a relés.

La figura de arriba muestra el modo típico de conexión de las salidas a relés de los PLCs. Aunque la figura muestra sólo la conexión en circuitos DC, también se puede conectar de manera similar en circuitos AC; ya que un relé es un elemento de salida no polarizado y en consecuencia él puede conmutar tanto AC como DC. En este caso se trata de salidas a contactos secos.

Un resumen de las salidas a relés es el siguiente: **son relativamente lentas, pueden conmutar corrientes algo grandes, tiene tiempo de vida relativamente corto y trabajan tanto en AC como en DC.**

En general existen dos tipos de transistores usados en la etapa de salida de los PLCs: Transistores NPN y transistores PNP. El tipo "físico" de transistor usado también varía de fabricante a fabricante. Algunos de los tipos mas comúnmente usados son los BJT y los MOSFET. Un transistor tipo BJT(Bipolar Junction Transistor) generalmente tiene menos capacidad de conmutación (Puede manejar menos corriente) que uno tipo MOS-FET(Metal Oxide Semiconductor- Field Effect Transistor). Sin embargo, el BJT tiene un tiempo de conmutación ligeramente más pequeño que el tiempo de los MOS-FET. Al igual que con las salidas a relés, hay que chequear las especificaciones dadas por el fabricante acerca de un grupo de salidas a transistores en particular, a fin de verificar que la máxima corriente de carga no exceda la del transistor.

La figura que se muestra a continuación incluye un típico diagrama circuital de salida para una del tipo NPN.

Fig. 38 – Salida a transistor.

Aquí también se muestra un foto acoplador cuya función es aislar los voltajes y corrientes del mundo exterior de la circuitería interna del PLC. Cuando la lógica programada indica que se debe activar esta salida, el circuito interno aplica un pequeño voltaje al LED del foto acoplador el cual emite entonces una luz que causa que el fototransistor permita el flujo de una pequeña corriente hacia la base del transistor conectado a la salida 0500. De aquí que lo que este conectado entre el COM y el terminal 0500 se active. Cuando la lógica programada indica que se debe desactivar la salida 0500, entonces se deja de aplicar el voltaje al foto acoplador lo que causa que ya el LED pare de emitir luz y así el transistor de salida conectado entre 0500 y COM se desactivara "abriendo" sus contactos.

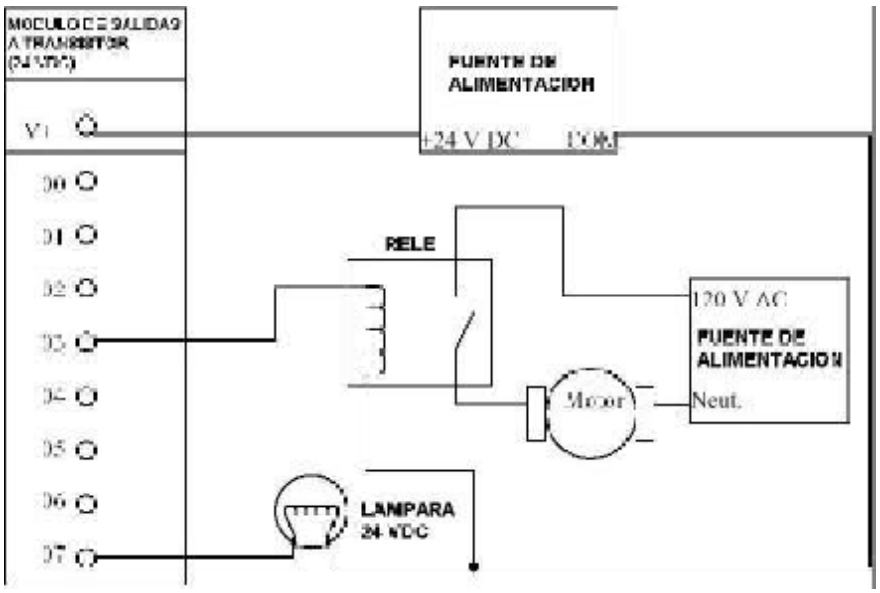


Fig. 39 – Conexión típica de un módulo de salidas a transistores.

Fig. 40 – Relé electromecánico.

Aquí se muestra el simple encendido de una campana cuando se cierra el switch. Existe 3 partes del mundo real: Un switch, un relé y una campana. Siempre que el switch se cierre se aplica una corriente eléctrica a la campana ocasionando que la misma suene.

Ahora, usemos un PLC en lugar de un relé. Lo primero por hacer es crear lo que se conoce como un diagrama escalera (LADDER). Aunque los PLC son computadoras que solo reconocen códigos de programación, afortunadamente muchos de ellos poseen software con el cual puede convertir el diagrama escalera a código de máquina.

Primer paso: se deben convertir todos los ítem del diagrama de control de la aplicación a símbolos que el PLC entiende. El PLC no entiende de switch, relés o campanas, mas bien él interpreta entradas, bobinas de salida, contactos, etc.

Primero se remplazará la batería con un símbolo. Este símbolo es común en todos los diagramas escalera y son llamados barras de potencial. Las mismas son simplemente dos líneas verticales, una a cada lada del diagrama. Se puede pensar que la de la izquierda es el potencial "+", mientras que la de la derecha es el potencial "-". Así el flujo lógico será de izquierda hacia la derecha.

Luego se colocará el símbolo de las entradas. En el ejemplo que se trata se tiene una entrada del mundo real (el switch) el cual se representará con el símbolo

El suministro de AC es una fuente externa y por lo tanto no se coloca en el diagrama escalera. El PLC sólo tiene que ver con encender sus salidas sin importar que está físicamente conectado a ellas.

Segundo paso: Debemos decir al PLC donde están ubicadas cada una de las entradas y salidas mencionadas. En otras palabras debemos dar todas las direcciones correspondientes a cada símbolo. Esto es: ¿Dónde esta físicamente conectado el switch?, ¿Dónde esta conectada la campana?. Cada fabricante de PLCs tiene su manera diferente de indicar estas direcciones, pero por ahora asúmase que la entrada tiene la dirección “0000” y que la salida tiene la dirección “500”

Paso final: Se debe convertir el esquemático de control en una secuencia lógica de eventos. Esto es que el programa que se escribirá dirá al PLC que hacer cuando ciertos eventos aparezcan en su contexto. En este ejemplo, se debe indicar al PLC que hacer cuando el operador active el switch.

00003	AND	
00004	LD	00003
00005	LD	00004
00006	AND	
00007	OR	
00008	ST	00107
00009	END	

Tabla 1 – Ejemplo de programación en lista de instrucciones.

2. **Diagrama Escalera:** Es un lenguaje de programación gráfica que conserva la estructura de los diagramas eléctricos de control.

4. **Diagrama de bloques funcionales:** Es un lenguaje gráfico que permite al usuario construir procedimientos complejos mediante la unión de bloques funcionales prediseñados.

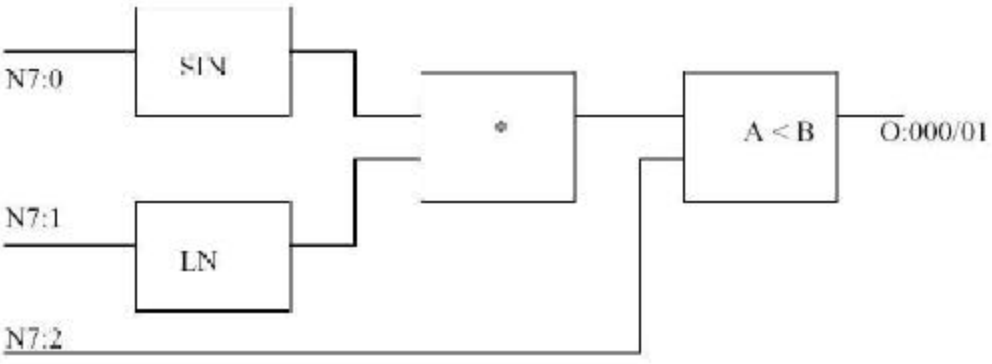


Fig. 46 – Programación con bloques funcionales.

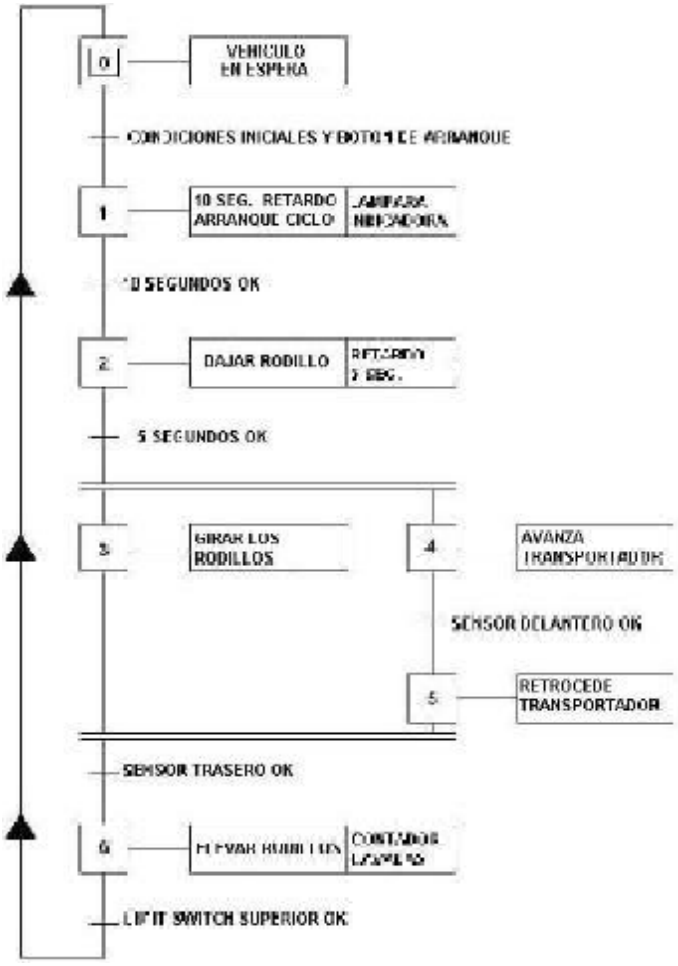


Fig. 47 – Ejemplo de diagrama GRAFCET.

5. **Texto estructurado:** Este es un lenguaje estructurado de alto nivel parecido al PASCAL, pero más intuitivo para el ingeniero de control. Este lenguaje es usado principalmente para implementar procedimientos complejos que no pueden ser expresados mediante lenguajes gráficos.

```
N7:0 := 0;
REPEAT
N7:0 := N7:0 + 1;
UNTIL N7:0 >= 10;
END_REPEAT;
```

Fig. 48 – Ejemplo de código de programación con texto estructurado.

Este símbolo es usado cuando se requiere que la señal este presente para que el símbolo este lógicamente activo. Cuando la entrada física esta activa o en alto se puede decir que la condición lógica del símbolo es “Cierta”. El PLC examina si la señal de entrada esta en “on”, es decir, si la entrada física está “on” entonces el símbolo también estará “on”. Una condición “on” también se conoce como un estado lógico “1” o un “alto”.

Este símbolo normalmente puede ser usado con entradas internas, entradas externas y contactos de salida externos.

Cargar Barra(LoadBar): La instrucción “LoaDBar” se refiere a un contacto cerrado. También es conocida como “examinar si esta cerrado” (examine if closed: XIC) y como “Cargar negado” (LoaDNot: LDN). Esto es examinar si la entrada física esta desactivada. El símbolo para la instrucción LoadBar es el siguiente:

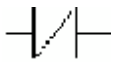


Fig. 50 – Símbolo para la instrucción LoaDBar.

Es usado cuando se requiere que el símbolo este activo en ausencia de señal. Esto es, cuando la señal física esta en “off” o en un nivel “bajo”, entonces el símbolo

Fig. 51 – Símbolo OUT (Bobina de salida)

Cuando en un “peldaño” existe una secuencia de símbolos con condición lógica “cierta” que preceden esta instrucción, entonces se activa la salida física correspondiente a la dirección indicada en la instrucción OUT. Esto es equivalente a una salida que esta normalmente desenergizada. Esta instrucción es usada tanto en bobinas internas que son simuladas (no existen físicamente) como en salidas hacia el mundo exterior.

SalidaBarra(Outbar): La instrucción Outbar se conoce también como salida negada (OutNot). Es similar a la bobina de un relé que esta normalmente energizada. El símbolo usado para esta instrucción corresponde a:

Comparemos un diagrama escalera simple con circuitos de relés interconectados como el de la figura para apreciar las diferencias.

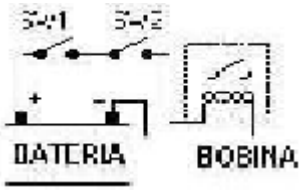


Fig. 53 – Circuito con lógica a relés.

En el circuito anterior, la bobina se energizara cuando exista un circuito cerrado entre los terminales positivo y negativo de la batería. Este mismo circuito se puede representar mediante un diagrama escalera que consiste de peldaños individuales justo como en una escalera real. Cada peldaño debe contener una o más entradas, y una o más salidas. La primera instrucción en un peldaño debe ser siempre una instrucción de entrada, y la última instrucción del peldaño debe ser siempre una salida o su equivalente.

Fig. 55 – Diagrama escalera con dirección de operandos.

Nótese que a cada símbolo (o instrucción) ahora se le asigno una dirección. Esta dirección especifica una cierta área de almacenamiento en los registros de datos del PLC, de tal manera que el estatus de la instrucción (verdadero o falso) puede ser almacenado. Muchos PLCs usan registros de 16 bits. En el ejemplo anterior se esta asumiendo que SW1 estará en principio físicamente abierto (off) y que el SW2 estará en principio físicamente cerrado (on). Igualmente se esta asumiendo que el registro 00 almacena el estatus de las entradas y que el registro 05 es la tabla de imagen de las salidas.

REGISTRO 00															
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
														1	0

REGISTRO 05															
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
															0

Tabla 4 – Registros del PLC.

En las tablas anteriores se puede apreciar que en el registro 00, el bit 00 (esto es entrada 0000) tiene un estatus bajo, mientras que el bit 01 (entrada 0001) tiene un estatus alto. Por otra parte el registro 05 muestra que el bit 00 (salida 0500) posee un estatus bajo.

De aquí que mirando la tabla anterior se concluya que en el ejemplo el SW1 debe poseer un estatus y estado lógico alto (1), mientras que el SW2 debe poseer un estatus bajo, para que así el PLC asigne a la salida un estado lógico 1(cierto), energizando la salida física. Si alguna instrucción en el peldaño es Falso, entonces el estado lógico de la salida será Falso y de esa manera no se activará la salida física.

En la siguiente tabla de la verdad se exploran las distintas combinaciones a las que puede dar lugar el ejemplo que se esta analizando.

Estatus de entradas		Estatus Salida	Estado Lógico del símbolo		
SW1	SW2	BOBINA	LD 0000	LDB 0001	OUT 0500
Abierto(0)	Abierto(0)	Desenergizada	0	1	0
Abierto(0)	Cerrado(1)	Desenergizada	0	0	0
Cerrado(1)	Abierto(0)	energizada	1	1	1
Cerrado(1)	Cerrado(1)	Desenergizada	1	0	0

Tabla 6 – Tabla de la verdad del ejemplo.

Se desea que la bomba de llenado inyecte aceite al tanque hasta que el sensor superior se active. Alcanzada la condición anterior se debe apagar el motor de la bomba y mantenerlo desactivado hasta que el tanque se vacié y caiga por debajo del nivel del sensor inferior. Entonces se debe encender nuevamente la bomba de llenado y repetir el proceso según lo descrito anteriormente.

En esta aplicación son necesarios 3 I/O (Entradas / salidas): 2 entradas para los sensores y 1 salida para el motor de la bomba. Asúmase por ejemplo que ambos sensores serán normalmente cerrados (NC) del tipo de sensores de nivel de fibra óptica. Cuando ellos NO están inmersos en el liquido estarán activados. Contrariamente, cuando están inmersos en el liquido estarán desactivados.

Por otra parte, hay que asignarle a cada una de las Entradas / salidas una dirección en el PLC, para que el mismo conozca donde están físicamente conectadas tanto las entradas como las salidas. Las direcciones escogidas se muestran en la siguiente tabla:

Fig. 57 – Diagrama escalera para el control de nivel.

Se debe recordar que la principal razón para usar los PLCs en la mayoría de las aplicaciones es para sustituir los relés del mundo real, y en ese sentido los relés internos del PLC hacen esta acción por lo demás posible. Es imposible indicar cuantos relés internos vienen en cada tipo o marca de PLC. Algunos incluyen cientos de relés, otros miles de relés. Típicamente, el tamaño del PLC (no es tamaño físico) es el factor que marca la pauta. Si se esta usando un micro-PLC con pocas entradas / salidas, normalmente no se necesitarán muchos relés internos, pero si en cambio se esta usando un PLC mayor con cientos o miles de entradas / salidas, obviamente se necesitaran gran cantidad de relés internos.

Fig. 58 – a) Scan 1. b) Scan 2 - 100.

Gradualmente el tanque se va llenando gracias a que 0500 esta activa y con esto se activa el motor de la bomba.

Después de 100 SCANs el nivel de aceite alcanza el sensor de bajo nivel por lo que el se activa y abre su contacto. De este manera pasa a un estado lógico FALSO .

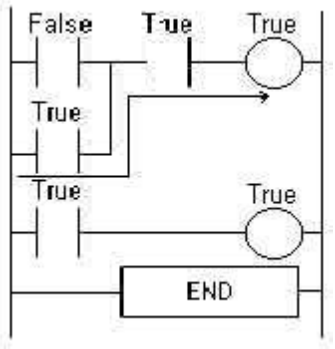


Fig. 59 – Scan 101 – 1000.

Fig. 60 – a) Scan 1001. b) Scan 1002.

Ya que no existe ningún camino lógico hacia la salida 0500, la misma se desactivara apagando así el motor de llenado.

Después del SCAN 1050 el nivel de aceite cae por debajo del sensor de nivel alto y él se hace CIERTO otra vez.

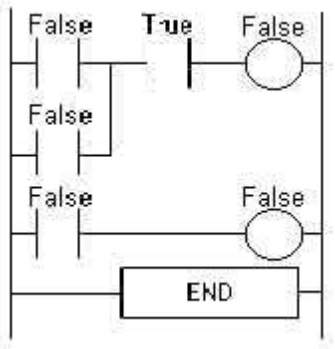


Fig. 61 – Scan 1050.

Resultado = A *AND* B		
A	**B**	**Resultado**
0	0	0
1	0	0
0	1	0
1	1	1

Tabla 8 – Tabla de la verdad función AND.

Aquí se puede observar que la operación AND esta relacionada con la multiplicación ya que la única vez que el resultado es cierto es cuando los dos operándoos son ciertos al mismo tiempo.

La función AND es útil sobretodo cuando el PLC no cuenta con funciones de enmascaramientos de registros. Cuando se trabaja a nivel de bits, la función de enmascaramiento permite dejar en un registro dado un único bit. Esto se debe a que cualquier bit que sea operado a través de una AND con sigo mismo

Tabla 9 – Tabla de la verdad función OR.

Aquí se puede ver que la función OR esta relacionada con la adición ya que mientras A o B sean ciertos, el resultado será cierto.

- **EXOR**- Esta función sigue la tabla de la verdad presentada a continuación.

Resultado = A *XOR* B		
A	**B**	**Resultado**
0	0	0
1	0	1
0	1	1
1	1	0

Tabla 10 – Tabla de la verdad función XOR.

Aquí se puede ver que la función no esta relacionada a nada. Una nemotécnica para recordar sus resultados es pensar que los operando deben ser opuestos para que el resultado sea cierto. Si por el contrario los operando son iguales el resultado será falso.

Esta operación es algunas veces útil cuando se desea comparar dos bits en dos

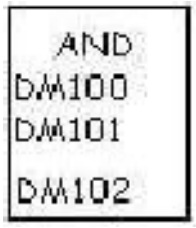


Fig. 62 – Símbolo AND

Las instrucciones AND, OR, y XOR en cualquier PLC típico tendrían un símbolo parecido al de arriba. Remplazando por supuesto la etiqueta AND por OR o XOR según corresponda. En el símbolo anterior, El operando A esta en la dirección de memoria DM100, El operando B esta en la dirección de memoria DM101 y el resultado se almacenará en la dirección de memoria DM102. Por lo tanto aquí se ha creado simplemente la ecuación Booleana DM100 AND DM101 = DM102. El resultado es almacenado automáticamente en la dirección DM102.

El siguiente diagrama escalera muestra el uso de la función Boolena AND.

Fig. 64 – Símbolo AND(método de la instrucción dual)

El método de la instrucción dual usa un símbolo similar al que se muestra arriba. En este método sólo se suministra con el símbolo la dirección del operando B. La dirección del operando A se suministra con la operación LDA, y la dirección del destino del resultado se suministra con la operación STA.

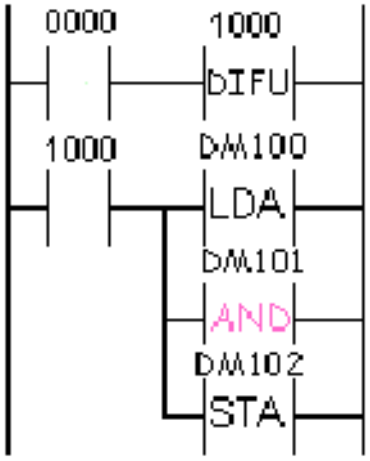


Fig. 65 – Uso de la instrucción dual para el AND en el diagrama escalera.

La instrucción de retención comúnmente se llama SET, mientras que la instrucción de liberación se llama RESET. El diagrama siguiente muestra el uso de estas dos instrucciones.

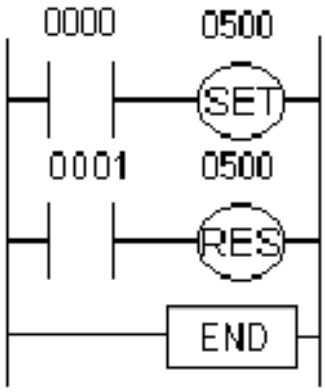


Fig. 66 – Uso de la instrucción SET y RESET.

Aquí se están usando dos switches tipo “push button”. **Uno de ellos esta físicamente conectado a la entrada 0000 del PLC, mientras que el otro esta conectado a la entrada 0001 del mismo.** Cuando el operador oprime el switch 0000 la instrucción "set 0500" se hace cierta por lo que entonces se activa la salida física 0500. Aunque el operador deje de oprimir el switch, la salida 0500 permanecerá

Fig. 67 – Figura animación uso de la instrucción SET y RESET.

Por otra parte, ¿Qué pasaría si se activan las dos entradas(0000 y 0001) al mismo tiempo?, ¿La salida 0500 estará enclavada en ON o restaurada en OFF?

Para responder a esta pregunta se debe pensar en la secuencia del scan que ejecuta el PLC. El diagrama de escalera siempre es barrido de arriba hacia a bajo y de izquierda a derecha. Lo primero que se ejecuta en el scan es el monitoreo de las entradas física. Esto arrojará que tanto 0000 como 0001 están activas. Luego el PLC ejecuta el programa y resuelve: Comenzando por el tope izquierdo, como 0001 esta activa entonces la salida 0500 debe estar lógicamente activa. En segundo lugar, como en el próximo peldaño 0001 esta activa, entonces la salida 0500 debe estar lógicamente inactiva o en cero. Finalmente, en la última parte del scan que es donde se actualizan las salidas, se mantendrá el último estado lógico (reset 0500) de la salida 0500 que no es otro distinto a que la salida física se mantenga desactivada.

También, muchos fabricantes incluyen un numero limitado de contadores de alta velocidad denotados usualmente como HSC (high-speed counter). Típicamente el contador rápido es un dispositivo del hardware del PLC, mientras que los contadores mencionados anteriormente son implementados mediante software. Es decir, mientras que los contadores ordinarios no existen físicamente sino que son simulados en el programa monitor del PLC, los contadores de alta velocidad si existen como elemento del hardware y funciona de esta manera en forma independiente del tiempo de scan del PLC.

Una buena regla práctica es simplemente usar los contadores normales (software) cuando los pulsos que se estén contando arriben con periodos mayores a 2 veces el tiempo del SCAN. Por ejemplo, si el tiempo de SCAN es 2 ms y los pulsos que se están contado llegan cada 4 ms o más, es posible usar los contadores normales. Si al contrario los pulsos arriban cada 3 ms hay que hacer uso de los contadores rápidos.

Típicamente los contadores de 16 bits pueden contar desde 0 hasta 9999 usando BCD (decimal codificado en binario), -32,768 hasta +32,767 o 0 hasta 65535 usando codificación binaria normal. Cuando el programa esta corriendo en el PLC, el

Cxxx es la dirección o el nombre del contador. Si se desea nombrarlo como el contador cero (0), se debe colocar ahí "C000".

yyyyy es el preset o el número de pulso que se desean contar antes de que el contador active su salida. Por ejemplo, si se desean contar 5 piezas antes de que se active la salida física que activa el mecanismo de embalaje de las piezas, habrá que colocar ahí el número 5 en decimal. Cuando el contador ha alcanzad el valor del preset entonces él activara un conjunto de contactos etiquetados también con la dirección C000 y que pueden ser utilizados en cualquier parte del programa de la aplicación.

.

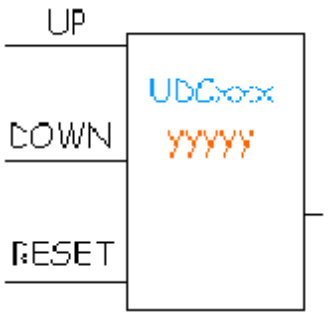


Fig. 70 – Símbolo típico de un contador bidireccional .

En este tipo de contador se requieren 3 entradas: La entrada de RESET la cual tiene la misma función que en el contador mencionado anteriormente, la entrada UP para contar ascendentemente y la entrada DOWN para avanzar descendentemente. En este ejemplo se llamara al contador como UDC000 y se le pondrá un valor de preset igual a 1000. (se desean contar 1000 piezas. El diagrama escalera correspondiente se muestra a continuación.

(VER ANIMA3.PPT EN EL CD ANEXO)

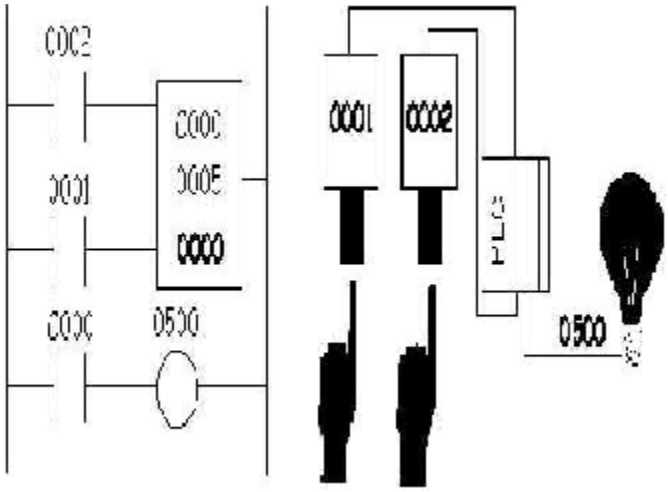


Fig. 72 – Figura animación uso del bloque contador.

- Este temporizador retarda la desactivación de una salida. Por ejemplo, después que un sensor detecta un objetivo, se activa inmediatamente una salida, y luego cuando ya el sensor no esta detectando más el objetivo, la salida se mantiene encendida por un tiempo determinado antes de desactivarla. El símbolo para este tipo de temporizadores es TOF (timer off-delay) y es menos común que el temporizador ON-DELAY.

- **Temporizador acumulativo o de retención**- Este tipo de temporizador requiere de dos entradas. Una de las entradas inicia la temporización y la otra la restaura a cero. La temporización de los mencionados anteriormente es restaurada a cero una vez que la entrada del sensor que los activa cambia de estado sin que haya concluido la temporización, mientras que este tipo de temporizador mantiene el tiempo de temporización que haya transcurrido cuando el mismo sea desactivado a mitad del ciclo de temporización. Por ejemplo, si se desea conocer cuanto tiempo estuvo un sensor activado durante el intervalo de una hora, hay que usar temporizador acumulativo ya que si se usan los ordinarios (on / off delay) el temporizador que lleva la

Fig. 73 – Símbolo básico de una instrucción de temporización.

Este temporizador es uno del tipo on-delay direccionado como **Txxx**. Cuando la entrada de activación (enable) se pone en un nivel alto comienza la temporización. Cuando los pulsos de temporización han alcanzado el valor de preselección **yyyyy**, él activará sus contactos que se podrán usar en cualquier forma y punto del programa de aplicación. Hay que recordar que la duración de cada pulso de reloj depende del tiempo base usado el cual se ajusta para cada temporizador con la ayuda de la consola de programación.

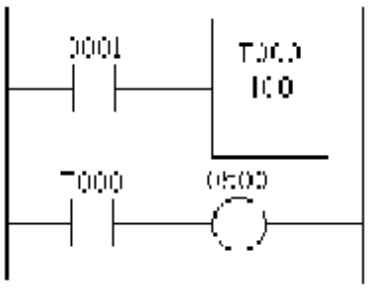


Fig. 74 – Uso del temporizador on delay en el diagrama escalera.

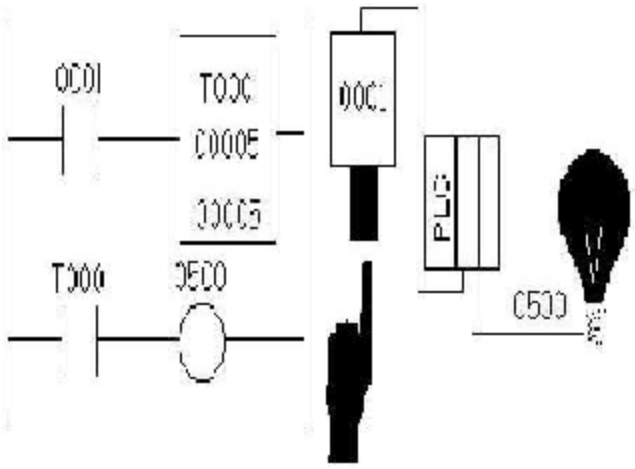


Fig. 75 – Figura animación uso del bloque temporizador on delay.

Un temporizador de acumulación típico sería como el siguiente.

Fig. 76 – Símbolo básico de un temporizador acumulativo.

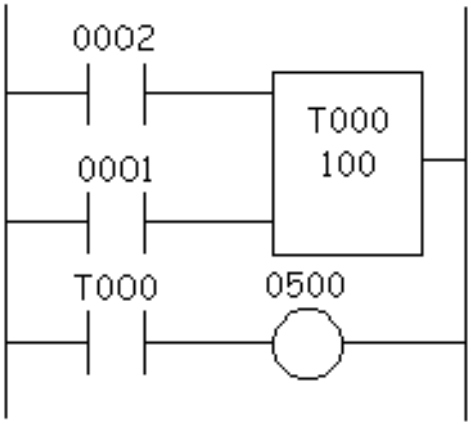


Fig. 77 – Uso del temporizador acumulativo en el diagrama escalera.

En este diagrama cuando la entrada 0002 se activa, el temporizador T000 de base de tiempo 10 ms comienza a contar pulsos hasta alcanzar 100 unidades que corresponden a 1 segundo (100 pulsos X 10ms = 1,000ms= 1 segundo). Cuando se alcanza la temporización de 1 segundo se cierra el contacto T000, activando de esta manera la salida 0500. Si la entrada 0002 cae hasta cero el tiempo contabilizado hasta el momento se retendrá hasta que la misma vuelva al nivel alto continuando la temporización en el punto que se dejó, o hasta que se active la entrada 0001 (RESET) con lo que se restauraría a cero el tiempo contabilizado y se iniciaría desde cero cuando se active 0002.

- Este tipo de error ocurre dependiendo del momento del ciclo de scan del PLC donde se active la entrada del temporizador. Si la entrada del temporizador se activa inmediatamente después que el PLC ha monitoreado o capturado el status de las entradas durante el ciclo de scan, el error de entrada será máximo (mayor o igual al tiempo de scan). Esto se debe a que las entradas son capturadas solamente una vez durante el ciclo de scan, y a que además hay que esperar hasta que la instrucción de temporización sea ejecutada. Si la instrucción de temporización es la última del peldaño, el error será aun mayor.

- **Error de salida**- Este tipo de error depende del momento dentro del programa cuando el temporizador agote su tiempo de temporización, y de cuando el PLC finalice de ejecutar la parte del programa del scan para entonces ir a la parte donde actualiza las salidas. Esto se debe a que el temporizador finaliza durante el tiempo de ejecución del programa pero el PLC debe primero finalizar de ejecutar el resto del programa antes de activar la salida apropiada.

Abajo hay un diagrama donde se ilustra el peor caso de error de entrada. Ahí se nota que el mismo corresponde a **1 tiempo completo de ejecución del scan + 1 tiempo de ejecución del programa**. También hay que recordar que el tiempo de

Fig. 80 – Diagrama error de salida.

Basado en lo expuesto anteriormente, se puede concluir que el peor tiempo de error posible correspondería a:

1 tiempo de scan + 1 tiempo de ejecución de programa + 1 tiempo de scan

= **2 tiempos de scan + 1 tiempo de ejecución de programa.**

Todo lo anterior significa que aunque muchos fabricantes actualmente ofrecen temporizadores con resolución de 1 ms, estos no deben ser usados para temporizaciones menores que algunos milisegundos, y esto asumiendo que el tiempo de scan sea de 1 ms. Si el tiempo de scan de la aplicación es por ejemplo 5 ms, es preferible no usar temporizaciones inferiores a 15 ms.

. Una instrucción **difd** usaría el mismo símbolo excepto que la etiqueta interna sería “difd". Algunos fabricantes usan como símbolo las mismas barras exteriores cambiándole la etiqueta interna por una “P” para la activación por flanco ascendente y una “N” para flanco descendente.

Esta instrucción es a menudo usada en conjunto con otras instrucciones avanzadas para realizar una actividad que debe tomar lugar solamente una vez. Por ejemplo, considérese la implementación de un flip-flop. Se desea que la primera vez que un operador accione un pulsador se active una salida y se mantenga memorizada en ese estado hasta tanto el operador vuelva a accionar el pulsador.

- **Peldaño 3**- NO 1001 es cierto, entonces la salida física OUT 500 es cierta.

Próximo Scan

- **Peldaño 1**- NO 0000 permanece cierto. DIFU 1000 ahora cambia a falso. Esto se debe a que la instrucción DIFU es cierta sólo durante un scan
- **Peldaño 2**- NO 1000 es falso, NO 1001 permanece cierto, NC 1001 es falso, NC 1000 cambia a cierto. Ya que aun existe un camino cierto (NO 1001 & NC 1000), entonces OUT 1001 permanece en cierto.
- **Peldaño 3**- NO 1001 es cierto y por lo tanto OUT 500 permanece cierto.

Después de por ejemplo 100 scan, el operador libera el pulsador y así NO 0000 cambia a falso. La lógica permanece en el mismo estado según como se describió en los peldaños 2 y 3 de arriba. Si por ejemplo en el scan 101 el operador acciona el pulsador, entonces NO 0000 cambia a cierto, y produciéndose los siguientes cambios:

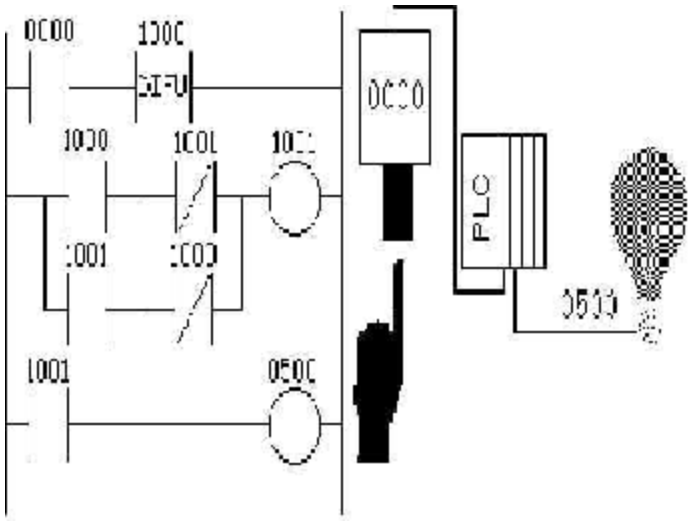


Fig. 83 – Figura animación implementación del FLIP-FLOP.

Fig. 84 – a) Control Maestro. b) Reset del Control Maestro.

Para ver como funciona, considere el siguiente ejemplo:

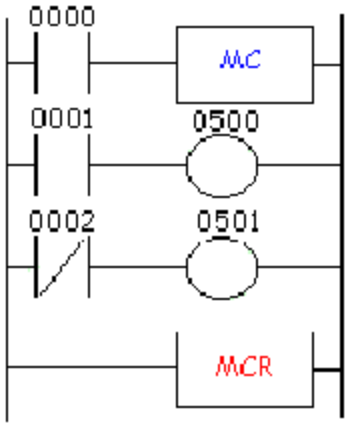


Fig. 85 – Uso del Control Maestro en un diagrama escalera.

En este ejemplo los peldaños 2 y 3 se ejecutan solamente cuando la entrada 0000 es cierta. De no ser así, el PLC asume que las instrucciones lógicas entre *MC y MCR* no existen y por lo tanto él hará un by-pass en este bloque de instrucciones e ira al peldaño ubicada inmediatamente después del la instrucción MCR.

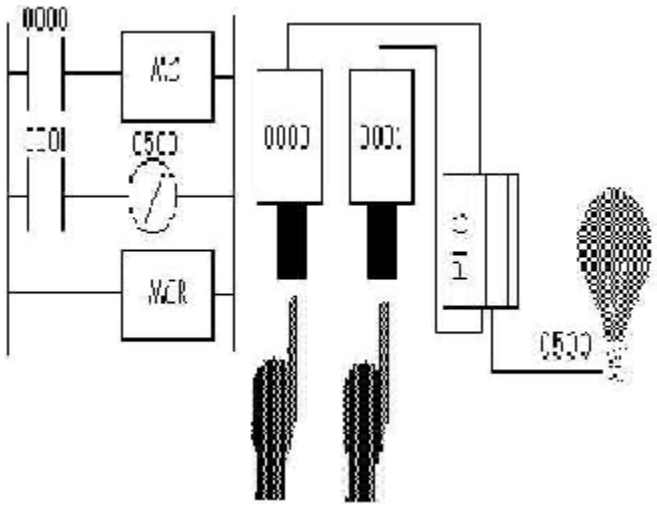


Fig. 86 – Figura animación uso Control Maestro.

n muchas aplicaciones es necesario almacenar el estado de uno o varios eventos que ha ocurrido previamente. En este caso el uso de registros o grupo de registros para formar un tren de bits que almacene el estado (on / off) de los eventos reseñados, es una técnica ventajosa. Cada nuevo cambio de estado se almacena la primera posición y los bits restantes avanzan una posición dentro del registro.

El registro de desplazamiento se etiqueta con variados nombres: SFT (ShiFT), BSL (Bit Shift Left), SFR (Shift Forward Register) son algunos de los más comunes. Estos registros desplazan los bits hacia la izquierda. BSR (Bit Shift Right) y SFRN (Shift Forward Register Not) son algunos ejemplos de instrucciones que desplazan los bits hacia la derecha. Pocos fabricantes ofrecen registros de desplazamiento hacia la derecha; mientras que la mayoría ofrece registros de desplazamiento hacia la izquierda.

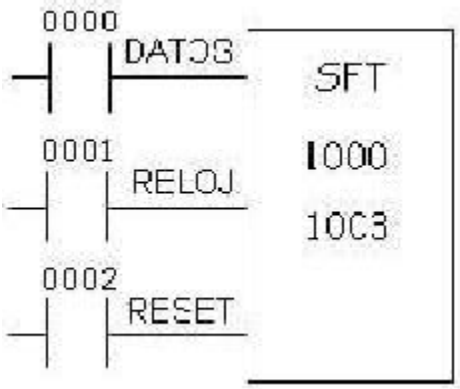


Fig. 87 – Símbolo típico de registro de desplazamiento.

- **Datos**- La entrada de datos suministra los estados cierto / falso que serán almacenados en el tren de bits dentro del registro. Cuando la entrada de datos es cierta el primer bit en el tren se pondrá en 1 y se almacenara en el registro con el flanco subiente de la entrada de reloj.

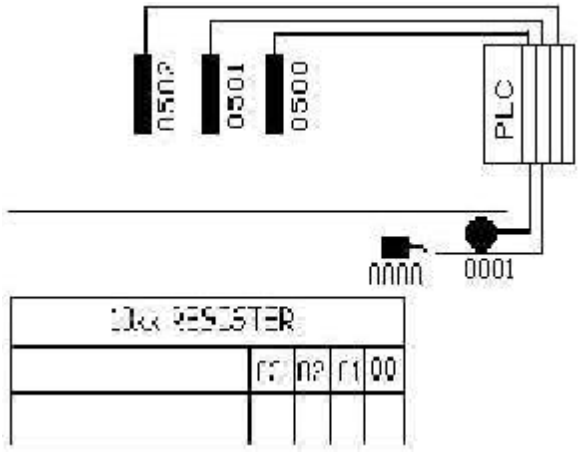


Fig. 88 – Figura del ejemplo de la máquina de barquillas.

Considérese una maquina de 4 estados destinada a la elaboración de barquillas. Primero se verifica que el cono base no esté roto. Luego se coloca el helado sobre el cono activando la salida 0500, luego se le adiciona maní activando la salida 0501 y finalmente se le rocía con caramelo al activar la salida 0502. Si el cono esta roto no se le adiciona ni el helado ni los items restantes. De aquí que se le deba hacer un seguimiento al cono dañado a medida que él avanza dentro del proceso a fin de informar a la máquina para que no adicione el ítem correspondiente a cada estado. Aquí se sugiere utilizar un sensor óptico para verificar si el fondo del cono esta dañado (entrada 0000), un codificador rotativo para seguir el viaje del cono a través de la línea transportadora (entrada 0001), y un push button para reiniciar el registro y

Fig. 89 – Uso del registro de desplazamiento en el ejemplo anterior.

A continuación se muestra el seguimiento del proceso según van aconteciendo las acciones y hechos. El registro 1000 inicialmente estaría en cero.

10xx Register															
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
												0	0	0	0

Tabla 11 – Estado inicial del registro de desplazamiento.

Un cono en buen estado llega y el sensor (entrada 0000) cambia a cierto. El registro 1000 no almacenará el dato hasta tanto no reciba el flanco subiente proveniente del codificador (entrada 0001). Finalmente cuando el codificador genera la entrada requerida, el estado de la entrada de datos es transferida hacia el registro, el cual se vería como:

10xx Register															
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
												0	0	0	1

Tabla 12 – Estado del registro al llegar un cono en buen estado.

Tabla 13 – Estado del registro al llegar un cono en mal estado.

Como el registro muestra que el bit 1001 es cierto, entonces la lógica del diagrama escalera indica que hay que activar la salida 0500 que es la que agrega el helado al cono bueno.

Según que el transportador sigue avanzando, un cono bueno alcanza al sensor y este cambia su estado a cierto. Nuevamente el codificador genera el pulso de reloj con lo que ingresa el nuevo dato al registro mientras que los datos previos son desplazados hacia la izquierda. Ahora el registro luce de la siguiente forma:

10xx Register															
15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
												0	1	0	1

Tabla 14 – Estado del registro al llegar el próximo cono en buen estado.

Dado que el registro muestra el bit 1002 como cierto, se activa la salida que adiciona el maní. Igualmente como el bit 1001 esta en falso mostrando el cono roto, la lógica del diagrama escalera no esta habilitada para adicionar el helado sobre este cono. Al seguir en movimiento el sistema de transportación, un cono en buen estado alcanza el sensor y luego el codificador genera la entrada de validación del reloj. El nuevo estado del registro sería:

15	14	13	12	11	10	09	08	07	06	05	04	03	02	01	00
												0	1	1	0

Tabla 16 – Estado del registro al llegar el próximo cono en mal estado

Nótese que el estado del primer cono que ingreso ha desaparecido, aunque realmente ha pasado a la posición 1004 la cual no se está usando en la lógica programada en diagrama escalera y por lo tanto carece de utilidad. La operación descrita continua sucediéndose con cada flanco ascendente proveniente del codificador.

(VER ANIMA8.PPT EN EL CD ANEXO)

. Considérese que en una aplicación se está utilizando un módulo especial de entrada como por ejemplo uno de conversión Análogo / Digital. Este módulo adquiere las señales analógicas del mundo exterior (una corriente o un voltaje variable) y las convierte a un formato que el PLC pueda entender (una señal digital: 1's y 0's), al mismo tiempo que las almacena en la memoria. Sin embargo se debe acceder los datos almacenados y moverlos a otra posición de memoria o de lo contrario la próxima muestra remplazará la previamente existente, produciéndose la perdida irremediable de esos datos. En otros casos simplemente se desea almacenar una constante, acceder algún dato binario no proveniente de las entradas discretas (por ejemplo datos provenientes de un programador cíclico), realizar alguna operación matemática y almacenar el resultado en una dirección de memoria diferente, etc.

Típicamente existen dos instrucciones que se utilizan para el acceso y la manipulación de datos (*LDA & STA)*, aunque algunos fabricantes utilizan una instrucción única para realizar enteramente esta operación (*MOV)*.

Fig. 90 – Símbolo de la instrucción MOV.

El símbolo del par de instrucciones LDA (LoaD Accumulator) y STA (STore Accumulator) se muestra a continuación. Es de hacer notar que el acumulador es simplemente un registro dentro del CPU donde el PLC almacena datos temporalmente mientras ejecuta algunas operaciones.

- Esta es la dirección de memoria a donde se moverá el dato. Por ejemplo si se escribe DM201 significa que el dato se moverá a la dirección de memoria 201. también se puede escribir por ejemplo 0500, lo que significaría que el bit menos significativo del dato debe ser movido a la salida física 0500, el siguiente a la 0501 … y el más significativo a la salida 0515. Esto puede ser útil si se tiene un display binario conectado a las salidas y se desea desplegar el valor de un contador para que lo use el operador de la máquina.

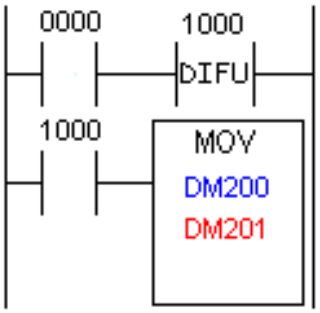


Fig. 92 – Uso de la instrucción MOV.

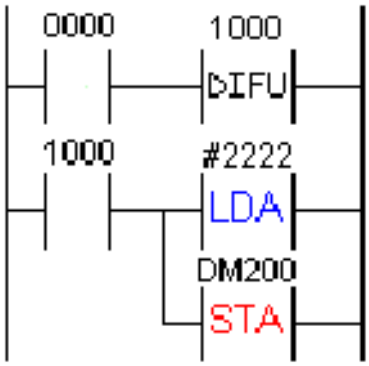


Fig. 93 – Uso de la instrucción a dos símbolos.

Para usar estas instrucciones se debe suministrar también 2 cosas:

- **LDA** - Esta instrucción es similar a la fuente de la instrucción MOV. Esto se refiere a la dirección donde esta localizada la data que se desea mover. Si se escribe DM100 significa que se desea mover el dato que esta localizado en la dirección de memoria 100. también se puede escribir aquí directamente el dato que se desea mover. Por ejemplo se puede escribir una constante (2222 por ejemplo). Esto último significa que el dato fuente es el 2222.

- **STA**- Esta instrucción es similar al destino de la instrucción MOV. Por ejemplo si se escribe DM200 significa que el dato se moverá a la dirección de memoria 200. también se puede escribir por ejemplo 0500, lo que significaría que el bit

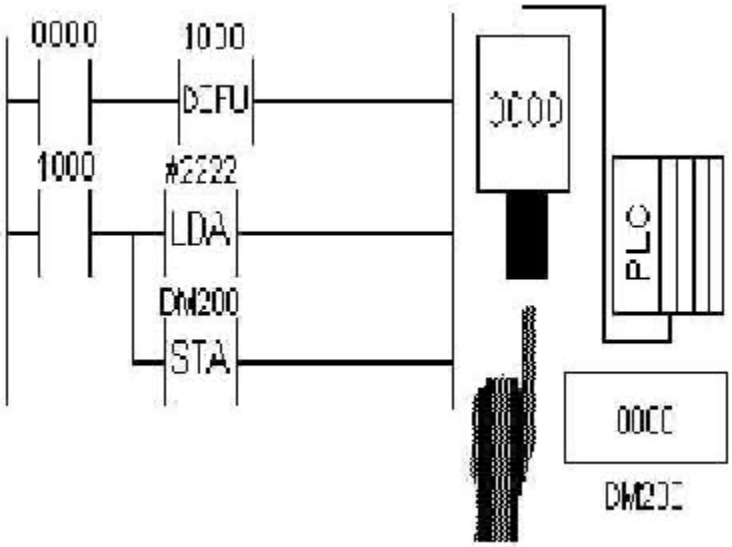


Fig. 94 – Figura animación manipulación de datos.

- Corresponde a la capacidad de sumar unos datos a otros. Comúnmente se denota con el símbolo ADD.

- **Sustracción**- Corresponde a la capacidad de sustraer unos datos de otros. Comúnmente se denota con el símbolo SUB.

- **Multiplicación**- Corresponde a la capacidad de multiplicar unos datos por otros. Comúnmente se denota con el símbolo MUL.

- **División**- Corresponde a la capacidad de dividir unos datos entre otros. Comúnmente se denota con el símbolo DIV.

Similar a como ocurre con la instrucción MOV, hay generalmente dos métodos que son usados por los fabricantes para realizar estas operaciones. El primer método incluye el uso de un sólo símbolo que requiere de tres entradas para ejecutar la instrucción:

Las operaciones aritméticas básicas se simbolizarían como se muestra arriba, intercambiando la etiqueta de ADD, SUB, MUL y DIV según se trate de una suma, resta, multiplicación o división respectivamente. En el símbolo anterior se ve que la operación es la suma, el operando 1 se encuentra en la dirección de memoria 100, el operando 2 se encuentra localizado en la dirección de memoria 101, y el resultado se almacenara en la dirección de memoria 102.

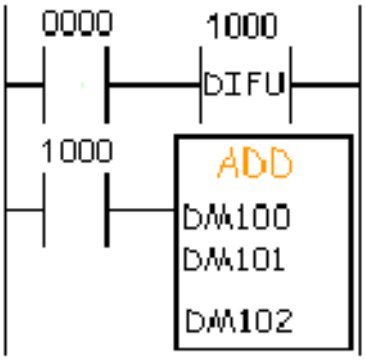


Fig. 96 – Uso de la operación Suma en el diagrama escalera.

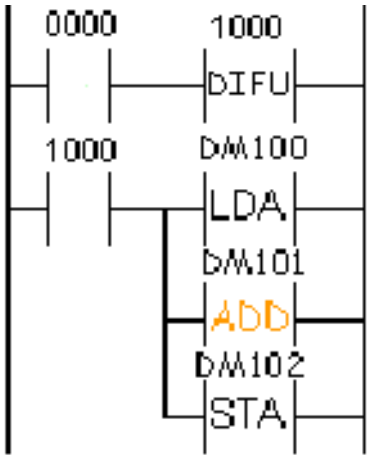


Fig. 98 – Uso de la instrucción de suma(ADD método dual).

El diagrama escalera muestra el uso de las instrucciones anteriormente indicadas. Obviamente el resultado es similar al obtenido con la instrucción simple mostrada antes.

Una interrogante interesante es preguntarse ¿Qué pasaría si el resultado de una operación matemática es más grande del máximo tamaño que se puede almacenar en una locación de memoria?

Típicamente las locaciones de memoria son de 16 bits. Esto significa que el mayor número que se puede almacenar corresponde a 65535 (2^16=65536). Si el resultado es mayor a este número ocurre un desbordamiento (overflow), y típicamente el PLC activa un relé indicando lo que ha pasado.

Algunos PLCs operan a 32 bits con lo cual resuelven este problema excepto para números realmente muy grandes. Otra causa de desbordamiento es la división por cero lo que también causa la activación del relé de error matemático.

Muchos PLCs también incluyen otras capacidades matemáticas tales como:

- **Extracción de raíz cuadrada.**
- **Escalamiento.**
- **Valor absoluto.**

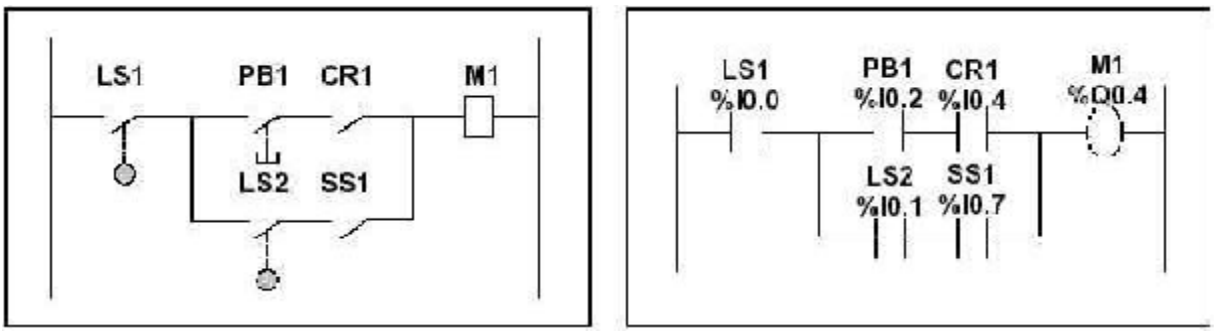


Fig. 100 – Equivalencia entre lógica a relés y diagrama escalera.

La figura de arriba ilustra un diagrama simplificado del control con lógica a relés y su equivalente en diagrama de escalera. Nótese que en el diagrama de escalera, todas los contactos de entrada están asociados con uno de los elementos de conmutación del diagrama de lógica a relés. La bobina de salida M1 en el diagrama de relés está representada con un símbolo de bobina de salida en el diagrama escalera. La dirección numérica que aparece en la parte superior de cada símbolo de contacto o de bobina en el diagrama escalera, hace referencia a la

Estas instrucciones gráficas se relacionan mediante líneas de conexión horizontales y verticales, que al final conducen a una o varias salidas y/o acciones. Un peldaño no puede soporta más que un grupo de instrucciones enlazadas. Por ejemplo el siguiente diagrama esta compuesto por dos peldaños.

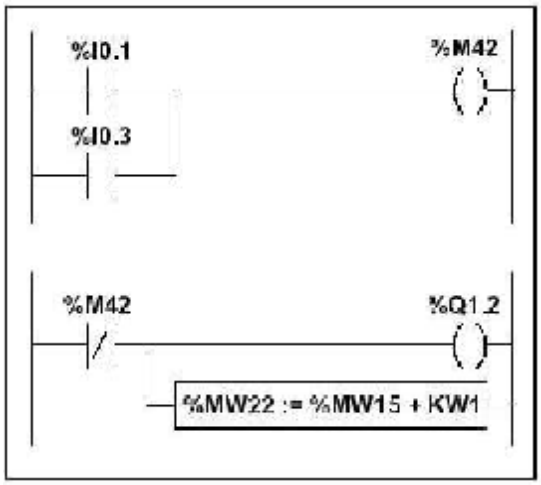


Fig. 101 – Ejemplo de peldaños.

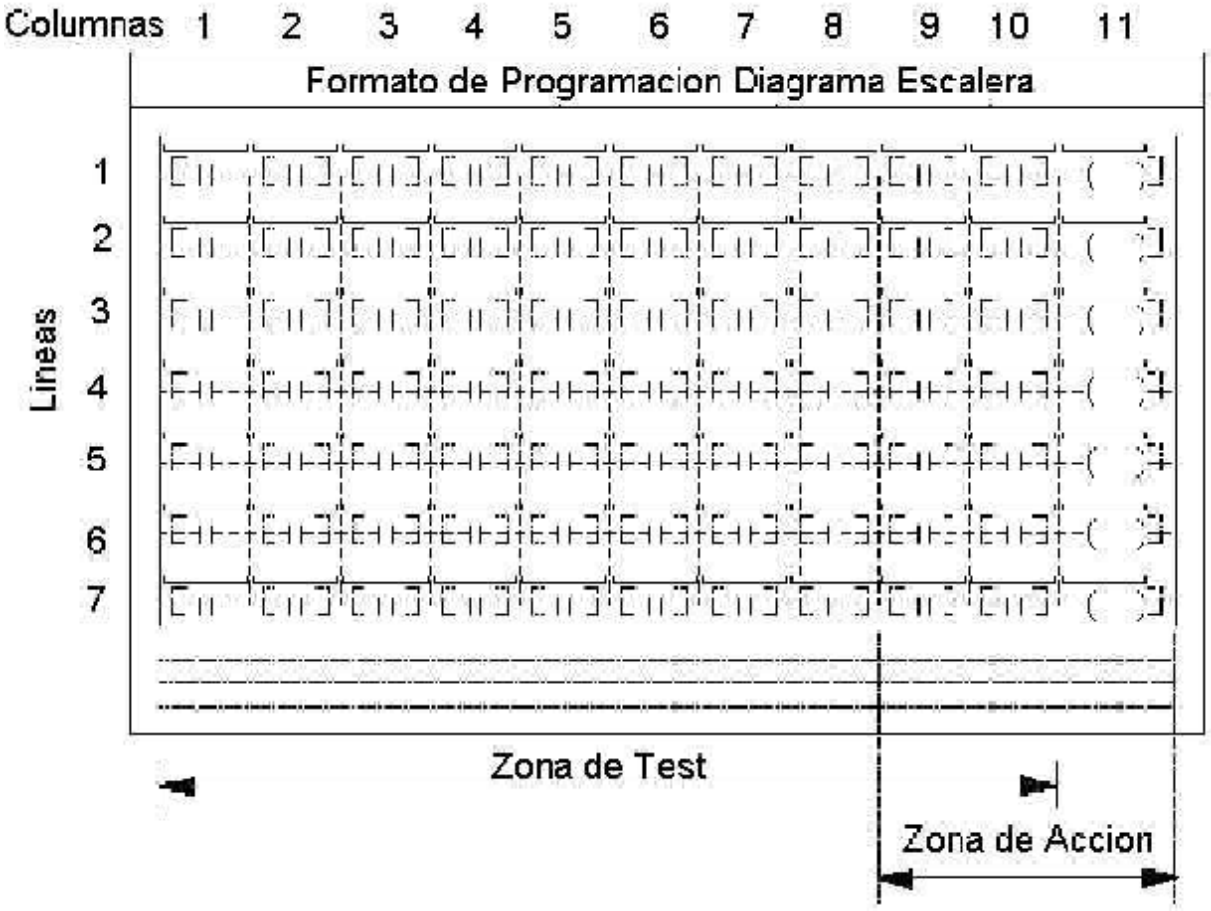


Fig. 102 – Formato de programación.

De esta forma el peldaño se visualiza como un formato de programación de 7 por 11, comenzando la programación a partir del tope de la izquierda. Aquí, se deben

Control de acceso de vehículos: Tan pronto como se detecte la presencia de un vehículo en la rampa de entrada, el sistema debe abrir automáticamente la barrera de entrada y/o de salida, de acuerdo a lo que corresponda. De igual forma, el numero de puestos disponibles, el control de horario de trabajo y otros eventos especiales, debe ser completamente administrado por el sistema.

- Seguridad de personas: Un detector de gases de escape debe iniciar el arranque de un gran extractor con la finalidad de evacuar el exceso de CO^2 en el estacionamiento. Asimismo, por razones económicas, las luces deben ser controladas por temporizadores.

- Arrancar / Parar un extractor al detectar exceso de CO^2.
- Programador semanal para horario de funcionamiento del estacionamiento.
- Control manual de apertura y cierre de la barrera de acceso.
- Posibilidad de modificar los parámetros de la cantidad de puestos disponibles en caso de eventos especiales.
- Indicar si hay disponibilidad de puestos o no en el estacionamiento.

Lo anterior conduce a la siguiente tabla de requerimientos:

Número de entradas discretas:	6
Número de entradas analógicas:	1
Numero de salidas discretas:	4
Contadores:	1
Temporizadores:	1
Reloj de tiempo real:	1

Tabla 17 – Requerimientos de entrada /salida del ejemplo del estacionamiento.

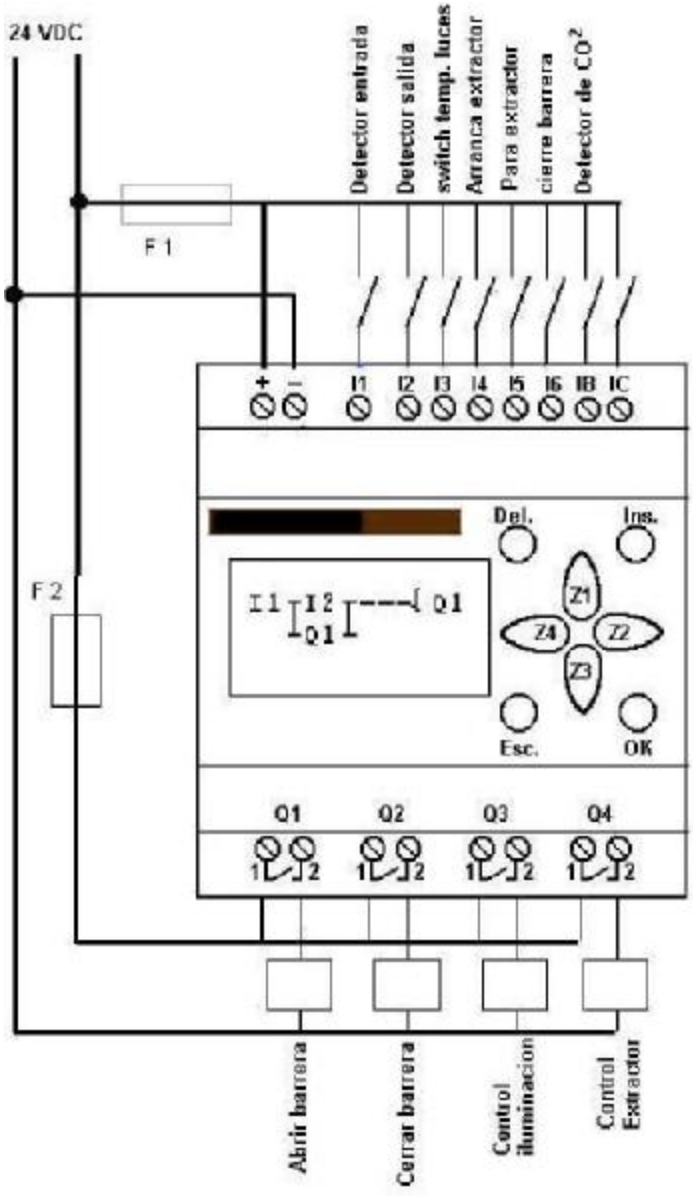


Fig. 104 – Diagrama de Entradas / salidas del PLC.

Tabla 18 – Variables de entrada / salida.

Nº	Función	Etiqueta	Tipo	Unidad	Valor	Bloqueo	Visible	Notas
01	Tempo.	T1	B: Pulse width Pulse On	M:S	[illegible]	NO		Control de iluminación
02	Tempo.	T2	B: Pulse width Pulse On	s	[illegible]	NO		
03	Tempo.	T3	A: On Delay	s	[illegible]	SI		
04	Tempo.	T4	B: Pulse width Pulse On	s	[illegible]	SI		
05	Tempo.	T5	B: Pulse width Pulse On	s	[illegible]	SI		
06	Tempo.	T6	B: Pulse width Pulse On	s	[illegible]	SI		Activa con la entrada
07	Tempo.	T7	D - Flashing relay	s	[illegible]	SI		
08	Contador	C1			[illegible]	NO	[illegible]	Numero de vacantes
09	Reloj	1				NO		Comanda las horas de funcionamiento
10	Reloj	2				NO		Comanda la apertura del estacionamiento
11	Reloj	3				NO		Comanda el cierre sobre del estacionamiento
12	Analog	A1	Z: [illegible]	Volt	[illegible] = 3.0	NO		
13	Vist.	Ib						Detector CO2

Tabla 19 – Parametrización de temporizadores y entradas analógicas.

Fig. 105 – Parametrización de bloques de texto.

d) Programa.

El listado de programa se muestra en las siguientes dos figuras. Para la simulación, cargar el software (EVA-SOFT) del CD anexo a este trabajo y abrir el archivo **parking.zel.**

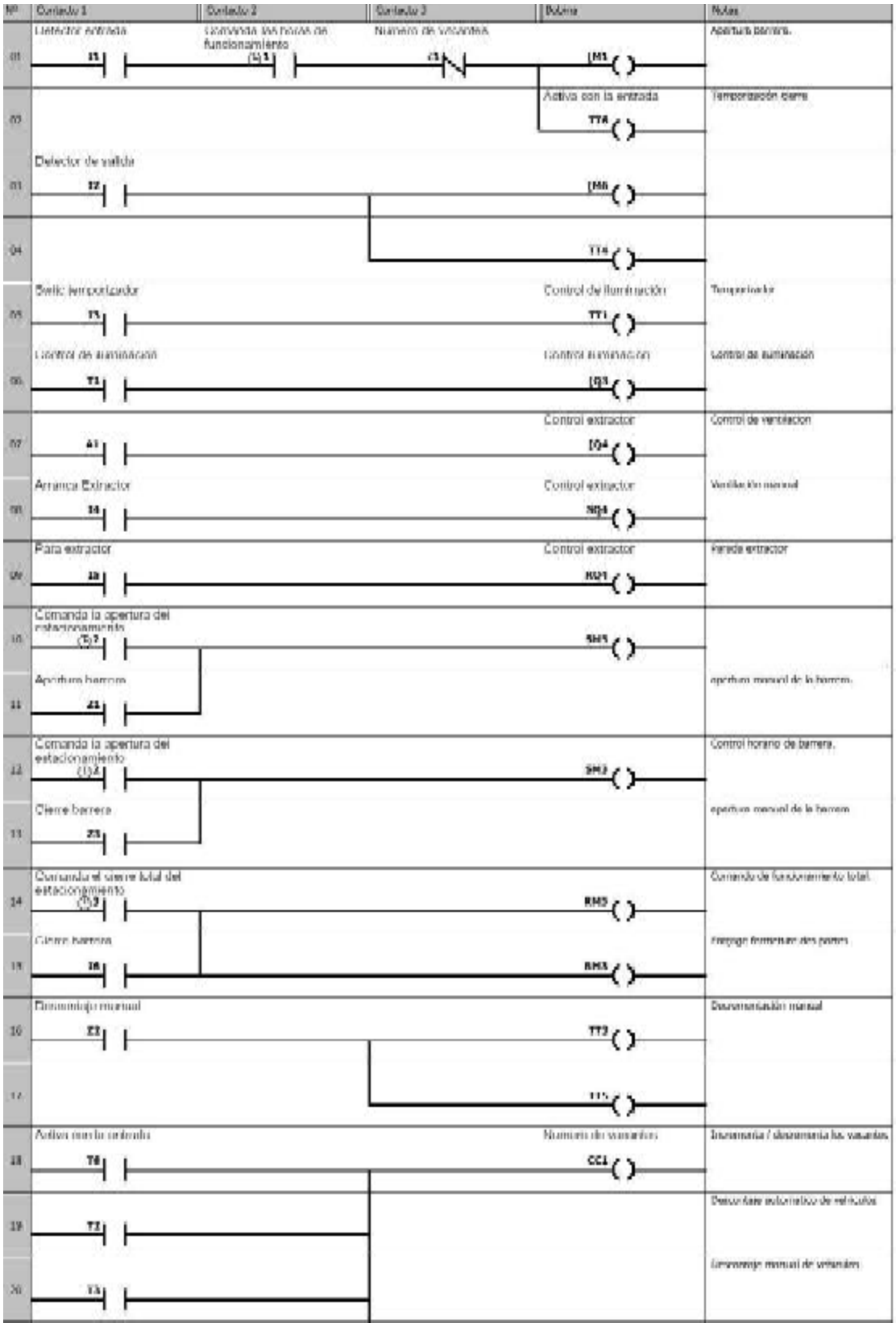


Fig. 106a – Diagrama escalera del ejemplo del estacionamiento.

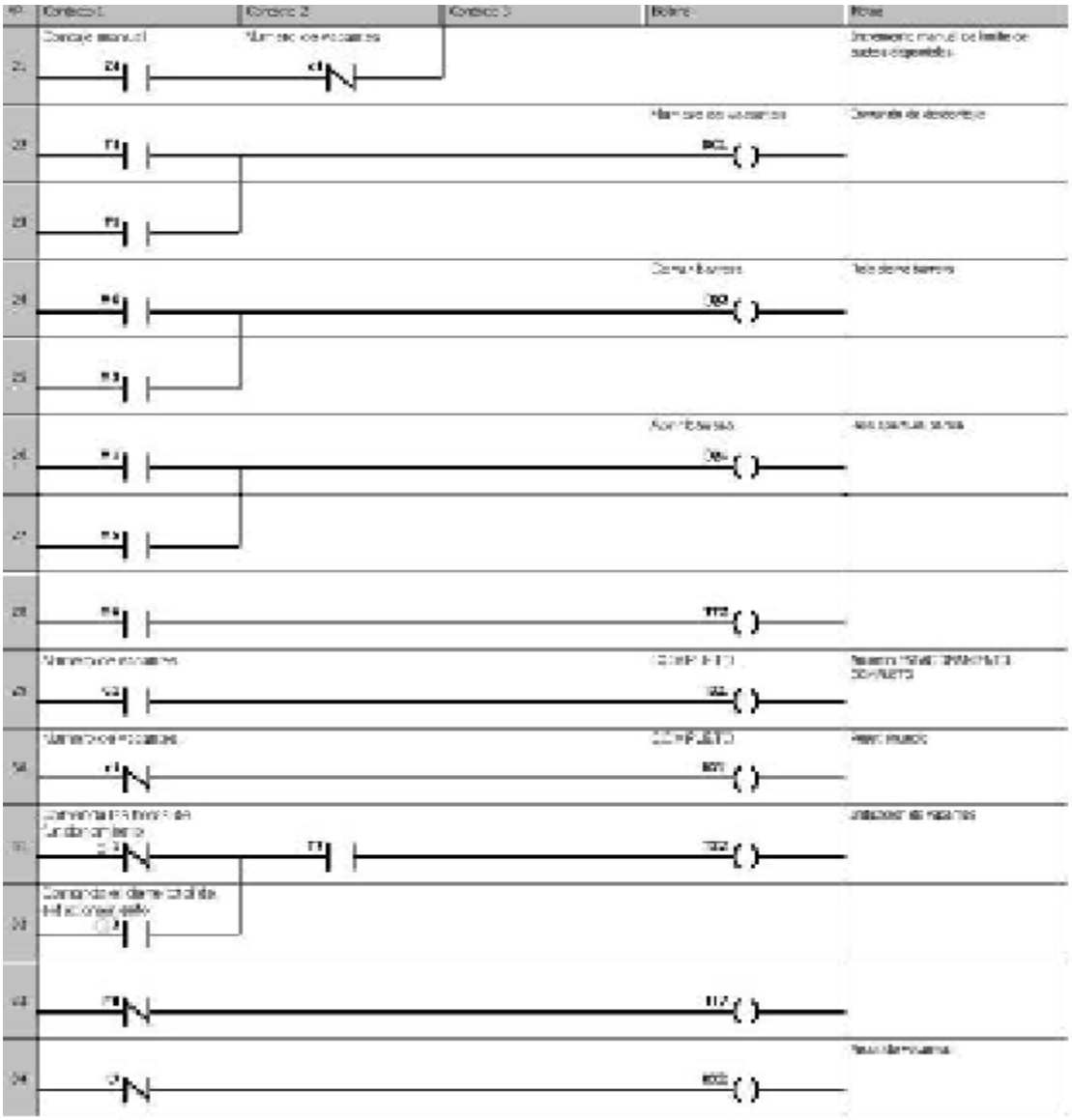


Fig. 106b – Diagrama escalera del ejemplo del estacionamiento.

Fig. 107 – Diagrama escalera.

En el diagrama escalera anterior contiene 4 entradas y 1 salida. Esta red puede ser representada mediante la ecuación Booleana que aparece justo abajo.

O:01 = I:00 AND (I:01 OR (I:02 AND NOT I:03))

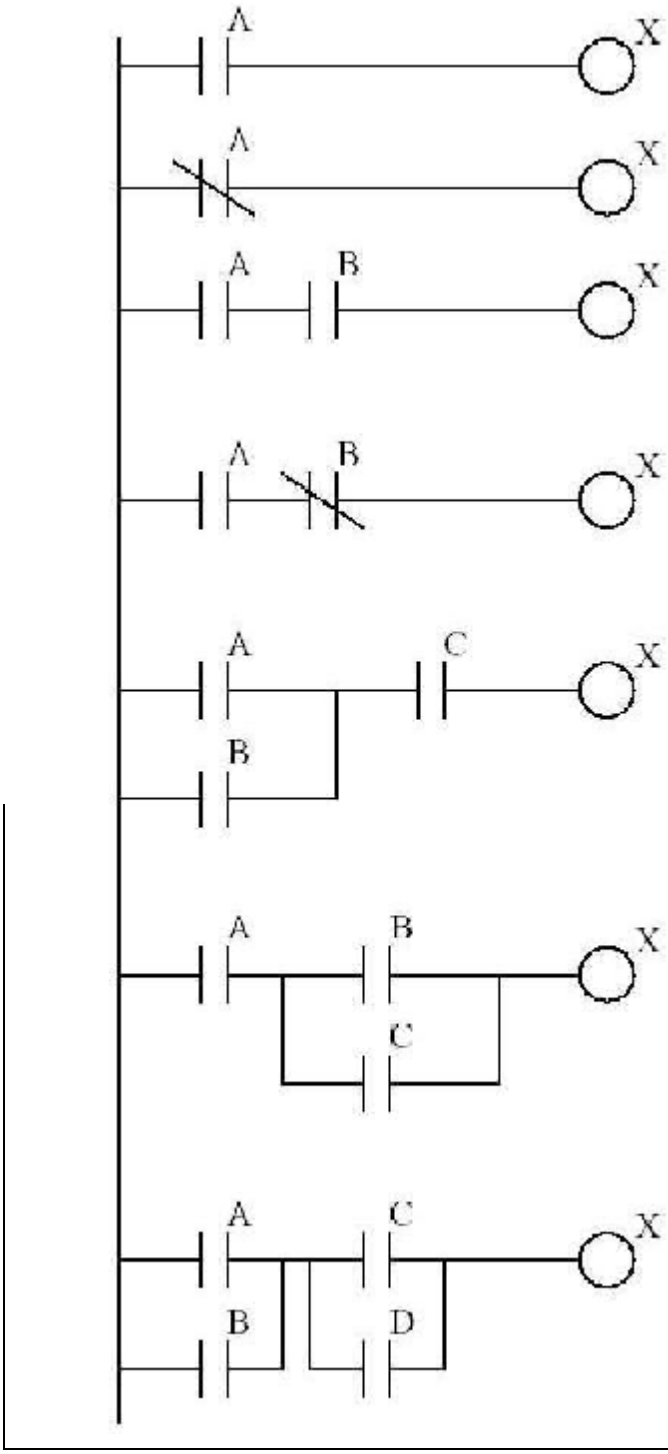
Igualmente, esta ecuación puede ser convertida a una lista de instrucciones usando los Nemónicos que aparecen a continuación:

Etiqueta	Cod. Ope.	Operando	Comentario
START:	LD	% I:00	(* Carga la entrada 00 *)
	AND(	% I:01	(* Comienza una rama y carga la entrada 01 *)
	OR(	% I:02	(* Carga la entrada 02 *)
	ANDN	% I:03	(* Carga la entrada 03 y la invierte *)
	)		
	)		
	ST	% O:01	(* Asigna la salida 00 *)

Fig. 108 – Lista de Nemónicos del ejemplo anterior.

DIV	(	VARIADO	División
GT	(	VARIADO	Mayor que (>)
GE	(	VARIADO	Mayor o igual que (>=)
EQ	(	VARIADO	Igual que(=)
EN	(	VARIADO	Distinto que (<>)
LE	(	VARIADO	Menor que (<)
LT	(	VARIADO	Menor o igual que (<=)
JMP	C, N	ETIQUETA	Salto a la dirección
CAL	C, N	NOMBRE	Llamada a subrutina
RET	C, N		Retorno subrutina.
)			Leer del “stack”

Tabla 21 – Operadores y Nemónicos que establece el estándar IEC 1131.



LD	A
LD	B
OR	C
ANB	
ST	X
LD	A
OR	B
LD	C
OR	D
ANB	
ST	X

Tabla 22 – Tabla de equivalencias Diagrama escalera - Lista de instrucciones.

Fig. 109 – Función compleja programada en Lista de Instrucciones.

En esta instrucción se suma el valor decimal 3 con el contenido del acumulador del temporizador T4:0, y se almacena en la dirección de memoria N7.0

EJEMPLO DE APLICACIÓN:

a) Descripción de la aplicación.

Un lavado de autos esta compuesto:

- Un puente que soporta un rodillo horizontal y dos rodillos verticales. El Puente es movido en dos direcciones (adelante y atrás) por un motor reversible.
- Un motor que rota tanto el rodillo horizontal como los rodillos verticales.
- Un motor que sube y baja el rodillo horizontal.
- Un limit switch que detecta cuando el rodillo horizontal esta en la posición alta.
- Un limit switch que detecta cuando el puente ha llegado al tope posterior de su recorrido.
- Un limit switch que detecta cuando el puente ha llegado al top5 delantero de su recorrido.

Fig. 110 – Puente de lavado.

b) Especificaciones funcionales.

Condiciones iniciales: El puente esta en la posición trasera, y el rodillo horizontal en su posición alta. Hay un vehículo presente en el área de lavado. Cuando se llenan estas condiciones, se presiona el botón de marcha y comienza el siguiente ciclo automático.

- Enciende la lámpara de proceso en marcha, y es seguida por una temporización de 10 segundos.
- El rodillo se mueve hacia abajo por un período de 5 segundos.
- Los rodillos comienzan a rotar y el puente se mueve hacia delante. Se presume que la bomba de agua se activa al unísono con la activación del puente.
- El avance del puente es detenido cuando toca el limit switch delantero, y en ese momento se pone en reversa.

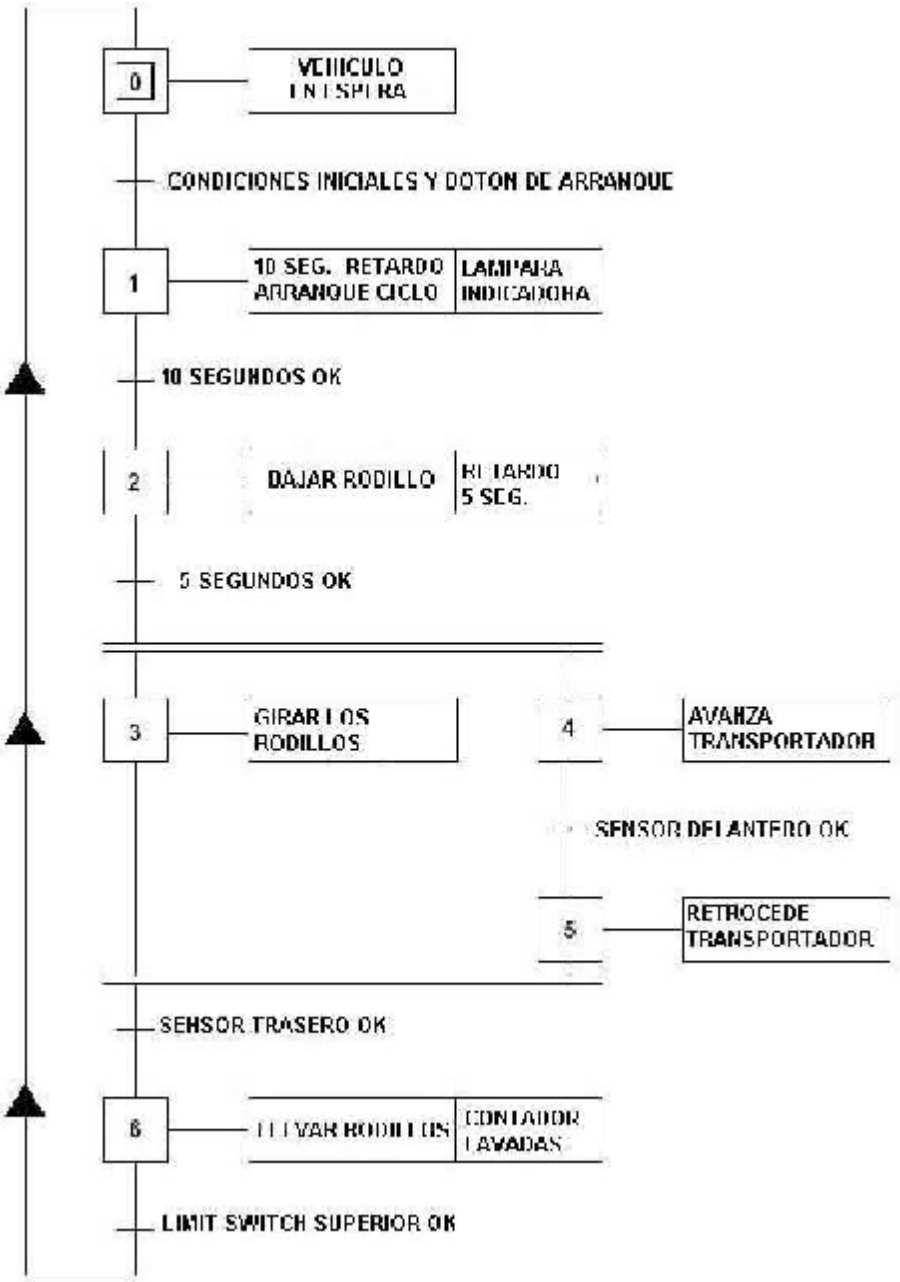


Fig. 111 – Descripción grafica del proceso.

d) Entradas / salidas al PLC.

VARIABLES INTERNAS		
Temporizador arranque	**T0**	ON DELAY (10 seg)
Temp.bajar rodillo	**T1**	ON DELAY (5 seg)
Contador de lavadas.	**C0**	ASCENDENTE(9999)

Tabla 23 – análisis de entrada / salida del ejemplo del lavado de vehículos.

e) Conexión del PLC.

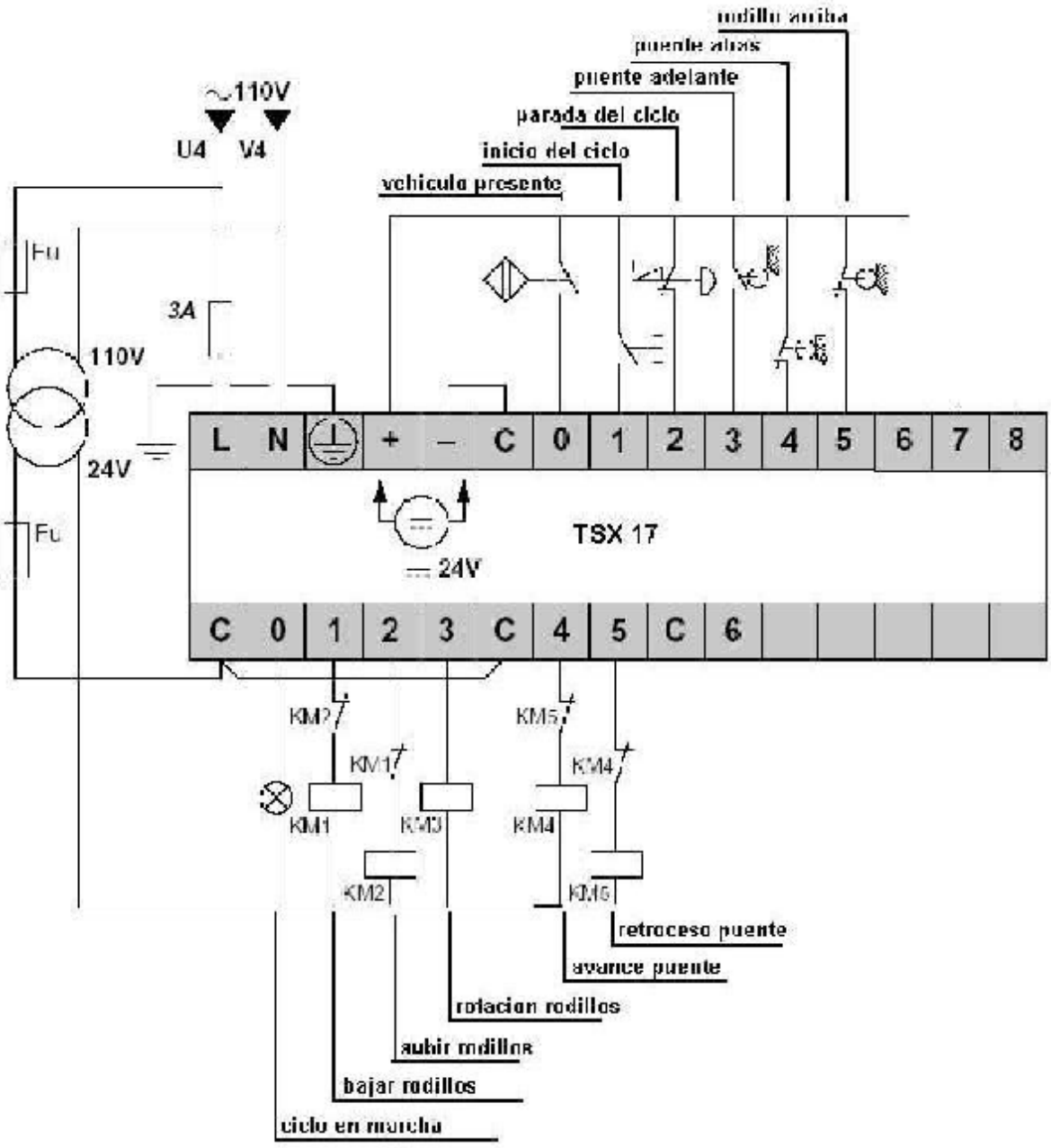


Fig. 112 – Conexión del PLC.

f) Programa.

n sistema es combinacional cuando para cada combinación de estado de los elementos de entrada al sistema, existe una y sólo uno combinación de estado de los elementos de salida. En forma distinta, un sistema es secuencial cuando para una combinación de estado de los elementos de entrada al sistema, pueden existir más de una combinación de estado de los elementos de salida; o bien cuando además de considerar el estado actual de los elementos de entrada y de salida, también se considera su estado anterior o la evolución que hayan tenido.

La mayoría de los procesos funcionan secuencialmente pero con un solo estado activo a la vez. Sin embargo, existen máquinas más complejas que son diseñadas para realizar varias operaciones al mismo tiempo. Este último tipo de máquina o proceso amerita que el controlador sea capaz de realizar procesos concurrentes. Es decir, el procesador debe ser capaz de realizar las actividades correspondientes a dos o más etapas que estén activas al mismo tiempo. Este último requerimiento puede lograse con técnicas tales como los Gráficos de Funciones Secuenciales (SFC: Sequencial Function Charts), conocido también como el estándar IEC 848, o sencillamente como GRAFCET (**Gráf**icos **F**uncionales de **C**ontrol de **E**tapas y **T**ransiciones).

El GRAFCET es un método mediante el cual se describe en forma gráfica las especificaciones de cualquier automatismo. Dentro de sus ventajas se encuentra que permite la programación directa desde el mismo gráfico sin tener que traducirlo a lenguaje de contactos o LADDER. También permite hacer seguimiento etapa por etapa dentro del automatismo, de forma que la localización de algún problema se hace sistemática y por lo tanto conduce al ahorro de tiempo.

El GRAFCET permite la representación de los sistemas secuenciales mediante la sucesión alternada de etapas y transiciones. Los elementos básicos de una carta GRAFCET típica se muestran a continuación.

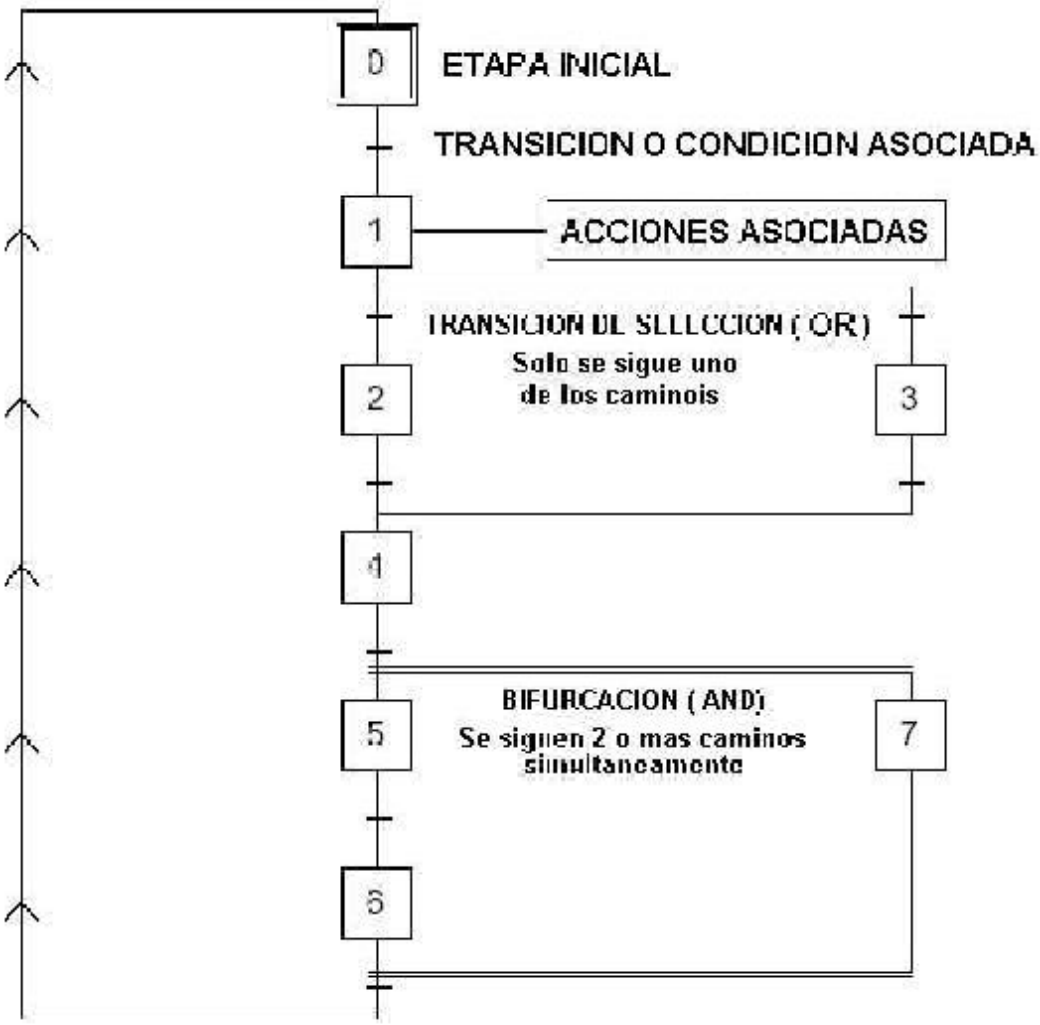


Fig. 113 – Principales elementos del GRAFCET.

ETAPAS: En la carta GRAFCET todos los estados estables del sistema tienen asociado un elemento de memoria llamado etapa. Las etapas se representan con un cuadro, o bien con un cuadro doble en el caso de etapas iniciales. Además, las etapas están numeradas en forma ordenada de acuerdo al desarrollo del automatismo.

a) Etapas iniciales.

- ❖ Se activan al iniciar el GRAFCET.
- ❖ Una vez iniciado, tienen el mismo tratamiento que otras etapas.

: Toda transición tiene una condición o condiciones lógicas que son las que van marcando la evolución del sistema. Una transición marca el paso de una etapa a la otra, ya que es una barrera que separa dos etapas y que se supera si estando activa la etapa o etapas anteriores o de entrada a la transición, se cumplen las condiciones lógicas en ella impuesta.

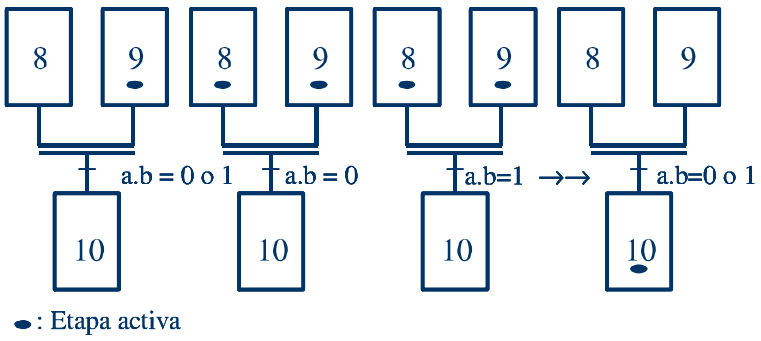


Fig. 114 – Ejemplo de evolución del GRAFCET.

Validar la transición implica un cambio en las etapas activas del GRAFCET. Al pasar una transición, el sistema deja de estar en una etapa e inmediatamente va a la siguiente.

Una receptividad asociada a alguno otra etapa GRAFCET.

ESTADO: Se refiere a una de las formas o maneras en la que puede encontrarse el sistema: Parado / operando, encendido / apagado, etc.

ACCIONES ASOCIADAS: Están asociadas a las etapas del GRAFCET y son las normalmente modifican el estado de algunos elementos de salida del sistema.

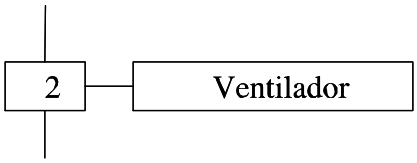


Fig. 115 – Acciones asociadas.

La figura anterior indica que al estar activa la etapa 2, se enciende el ventilador.

Existen acciones asociadas que son condicionadas. En este caso en el rectángulo donde se representa la acción, hay un campo de entrada para las condiciones.

S	Acción mejorada

Tabla 25 – Modificación de las acciones.

LÍNEAS DE ENLACE: Son líneas verticales u horizontales que unen con una dirección significativa, las distintas etapas con las transiciones, y las transiciones con las etapas.

6.3.1 Reglas para construcción del GRAFCET.

a) El diagrama debe dibujarse en **sucesión alternada de etapas y transiciones**. En ese sentido no puede haber ni dos etapas ni dos transiciones seguidas.

cuando a la salida de una etapa hay una selección de secuencias. En este punto, el flujo seguirá por una sola de ellas. Aunque no es necesario que las diferentes secuencias posean igual número de etapas y transiciones, lo que si es conveniente es que sean excluyentes entre si.

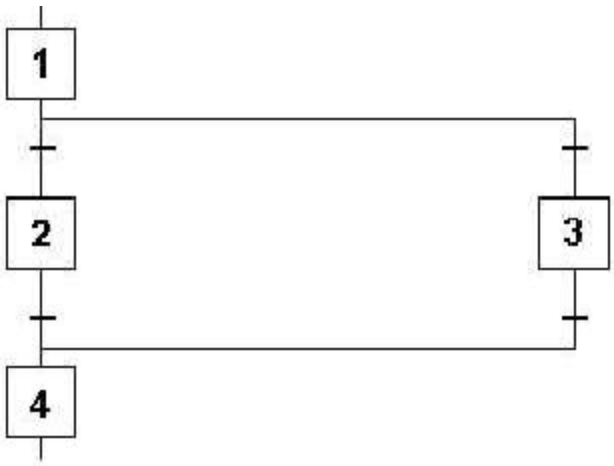


Fig. 118 – GRAFCET tipo OR.

Fig. 119 – GRAFCET tipo Paralelo

e) Una modalidad de los puntos de bifurcación lo constituyen los de etapas. Esta modalidad da la posibilidad de que se ejecute o no una secuencia completa, o de que la evolución del flujo siga a partir de la etapa indicada en el salto.

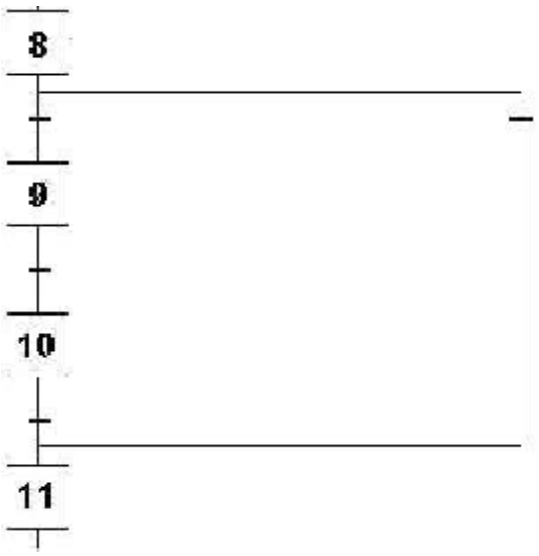


Fig. 120 – Saltos en cartas GRAFCET.

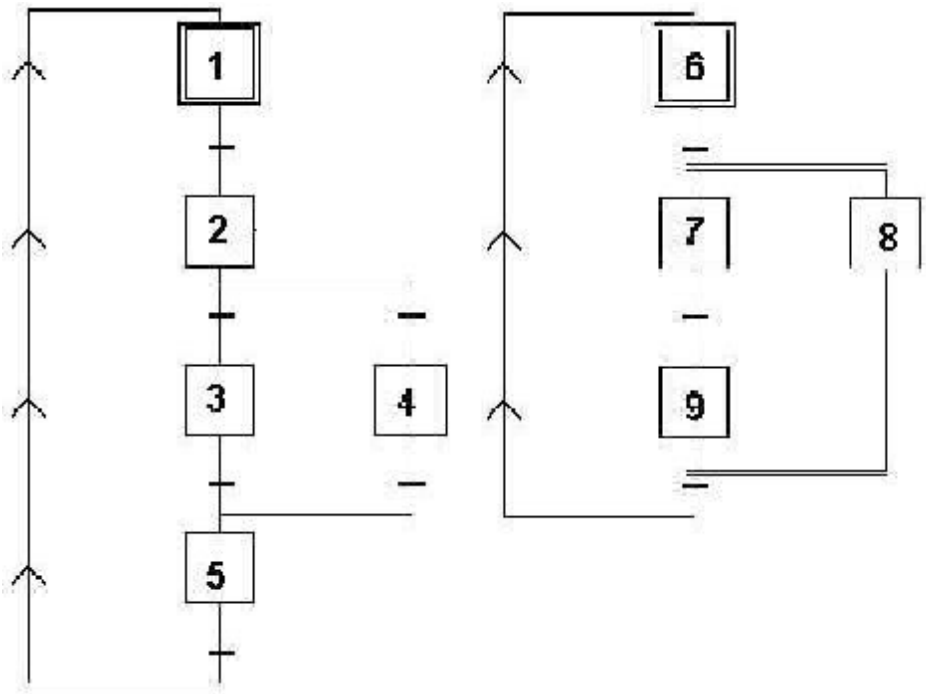


Fig. 121 – Representación de procesos concurrentes.

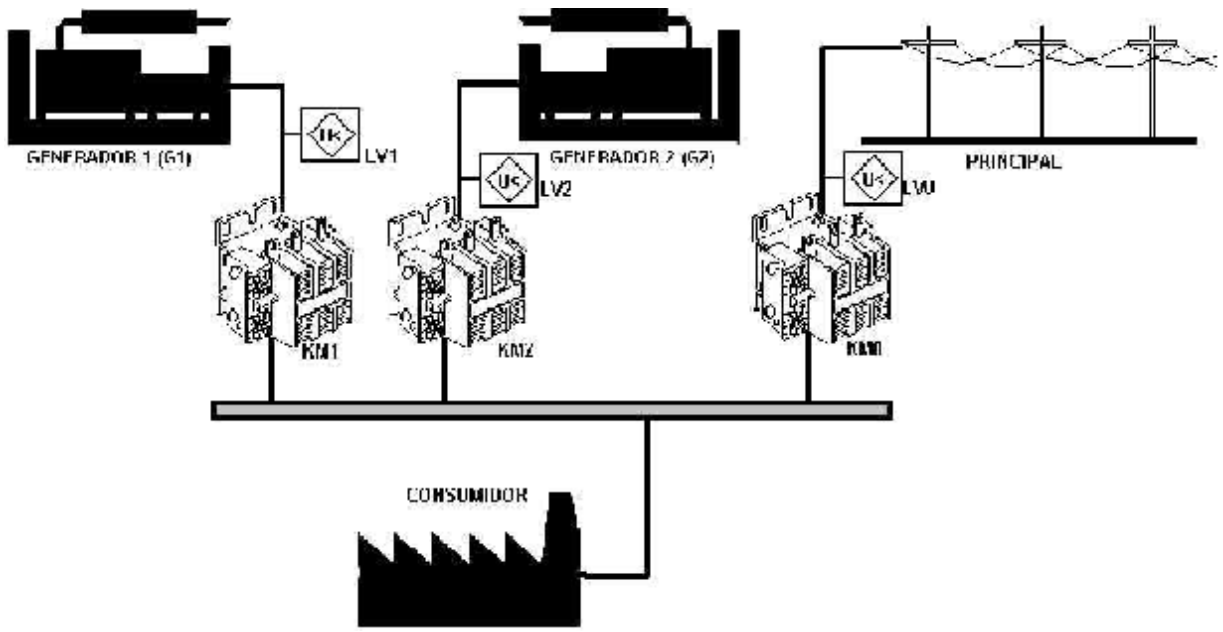


Fig. 122 – Esquemático del sistema de respaldo de energía.

b) **Especificación funcional.**

Condiciones iniciales: La línea de suministro principal debe estar OK, con G1 y G2 apagados. En consecuencia KM0 debe estar cerrado y KM1 junto a KM2 abiertos.

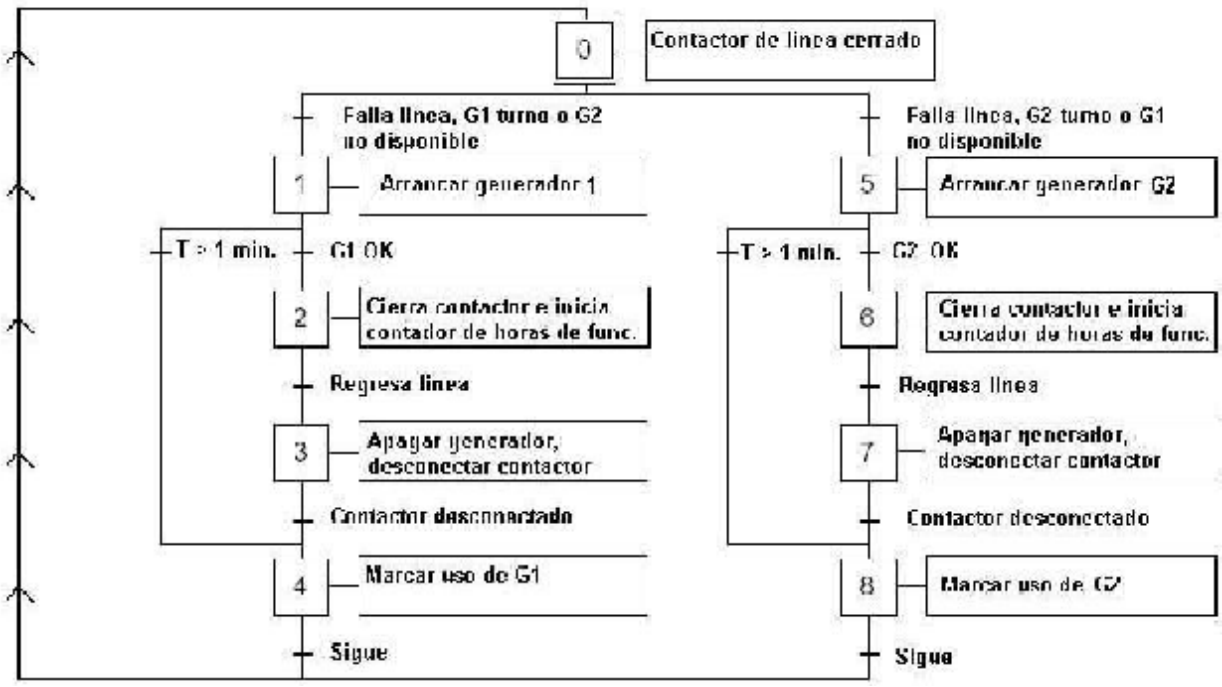


Fig. 123 – Descripción funcional a través de un esquema GRAFCET.

Mantenimiento G1	**O0,4**	Luz piloto LP1
Mantenimiento G2	**O0,5**	Luz piloto LP2
VARIABLES INTERNAS		
Tiempo falla línea	**T0**	ON DELAY (1 seg)
Tiempo retorno línea	**T1**	ON DELAY (5 seg)
Tiempo enfriamiento G1	**T2**	OFF DELAY (10 min)
Tiempo enfriamiento G2	**T3**	OFF DELAY (10 min)
Funcionamiento G1	**T4**	ON DELAY (60 min)
Horas G1	**C1**	ASCENDENTE(200)
Funcionamiento G2	**T5**	ON DELAY (60 min)
Horas G2	**C2**	ASCENDENTE(200)
Alternador G1 – G2	**SC0**	PASO a PASO (2 pasos)

Tabla 26 – Análisis de entrada / salida del sistema de respaldo.

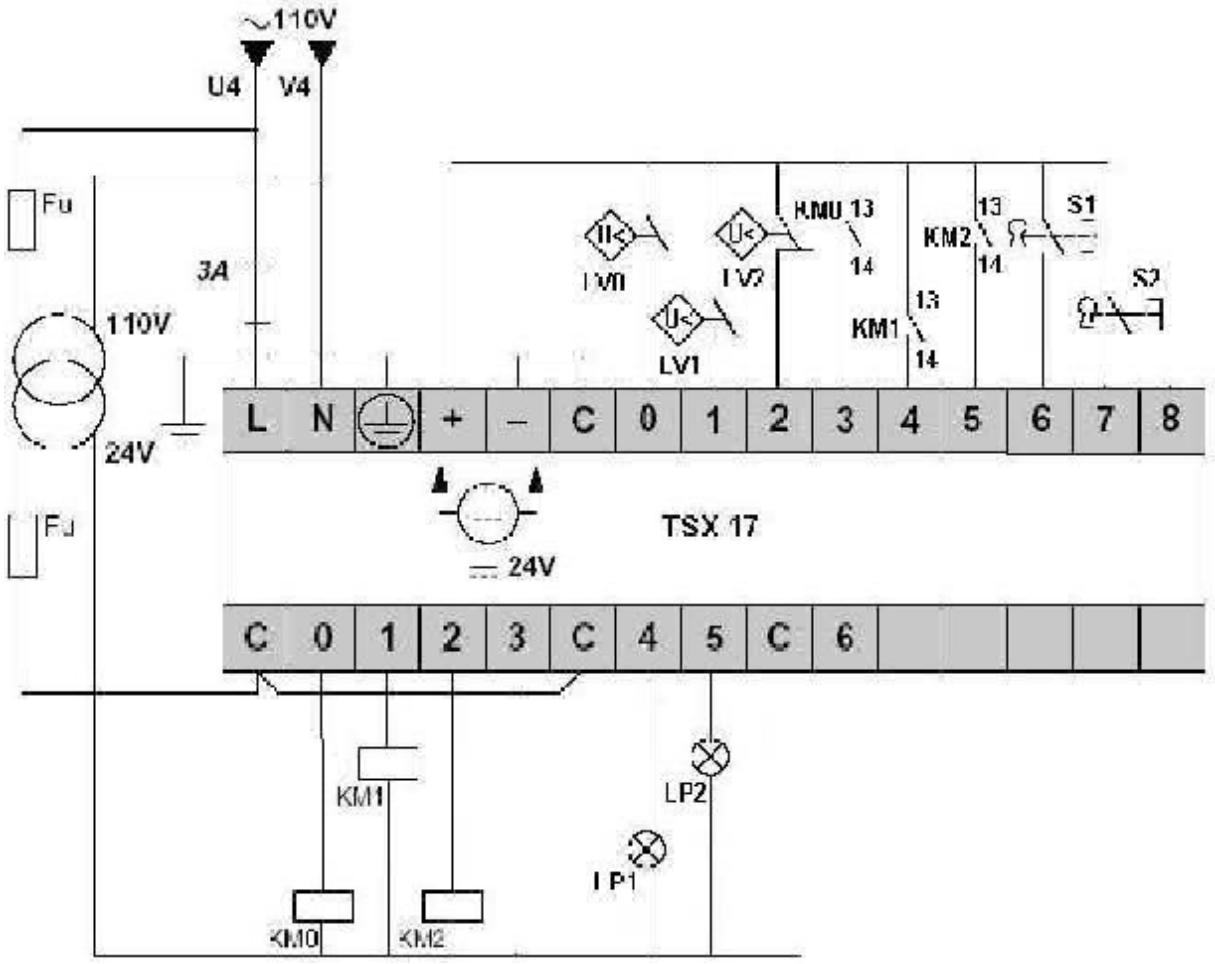


Fig. 124 – Conexión del PLC para el sistema de transferencia.

e) **Programa.**

DECLARACIÓN	DESCRIPCIÓN
VAR	Declaración general de variables.
VAR_INPUT	Declaración de variables de entradas a una función.
VAR_OUTPUT	Declaración de variables de salida de una función.
VAR_IN_OUT	Declara variables de entrada / salida de una función.
VAR_EXTERNAL	
VAR_GLOBAL	Declaración de variable global.
VAR_ACCESS	
RETAIN	Variable memorizada ante un corte de energia.
CONSTANT	Valor que no cambia.

Tabla 27 –Declaración de variables.

Booleano	0, FALSE, TRUE

Tabla 28 –Sistema numérico.

h) Las cadenas de caracteres usables se muestran a continuación:

Ejemplos	Descripción
‘’	Cadena vacia.
‘ ‘, ‘a’, ‘$’’, ‘$$’	Un espacio, un carácter, una comilla, símbolo $(dólar)
‘RL’, ‘rl’,‘$0D$0A’	Produce la combinación ASCII <CR>, <LF>
‘$P’, ‘$p’	Avance página.
‘$T’, ‘4t’	Tabulador <TAB>

Tabla 29 –Cadenas de caracteres válidos.

Fecha y hora	DATE_AND_TIME#1996-12-25-12:42:50.92, DT#1996-12-25-12:42:50.92

Tabla 31– Variables tipo calendario.

k) Funciones matemáticas básicas.

Función	Descripción
:=	Asigna una variable.
+	Suma
-	Sustracción.
/	División.
+	Multiplicación.
MOD(A,B)	Módulo. Provee el entero resultate de dividir A/B.
SQR(A)	Raiz cuadrada.

A**B	“A” elevado a la “B”

Tabla 32– Funciones matemáticas.

l) Funciones lógicas de comparación.

Función	Descripción
>	Mayor que.
>=	Mayor o igual que.
=	Igual que.
<=	Menor o igual que.
<	Menor que.
<>	Diferente de.

Tabla 33– Funciones lógicas de comparación.

m) Funciones de lógica Booleana.

Tabla 35– Estructuras de ejecución y lazos.

o) Instrucciones especiales.

Instrucción	Descripción
RETAIN()	Causa que un bit sea memorizado.
IIN();	Actualiza una entrada (Immediate Input Update)
EXIT;	Sale de un lazo FOR o WHILE
EMPTY	

Tabla 36– Instrucciones especiales.

Fig. 125 – Ejemplo de programación con bloques funcionales.

b) Las entradas y las salidas se pueden “negar” mediante la adición de un bloque inversor.

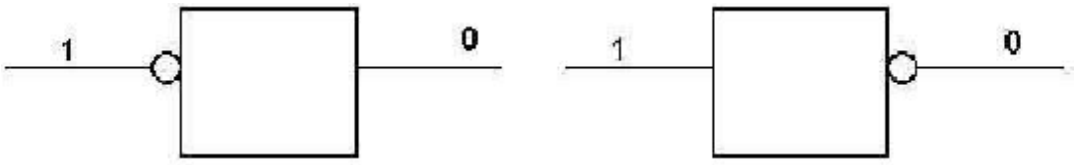


Fig. 126 – Bloques de negación.

c) Las funciones en los diagramas están basadas en otras funciones disponibles. Las entradas a una función van por la izquierda, mientras que la salida emerge por la derecha.

Fig. 128 – Funciones de tres argumentos.

e) Se pueden desarrollar bloques función usando texto estructurado, LADDER u otros lenguajes de programación.

Texto estructurado	Equivalente bloques funcionales

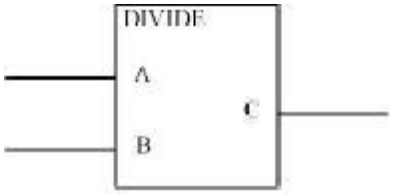
FUNCTION_BLOCK DIVIDE
VAR_INPUT
A: INT;
B: INT;
END_VAR
VAR_OUTPUT
C: INT;
END_VAR
IF B <> 0 THEN
C := A / B;
ELSE
C:= 0;
END_IF;
END_FUNCTION_BLOCK
	

Fig. 129 – Desarrollos de bloques funcionales.

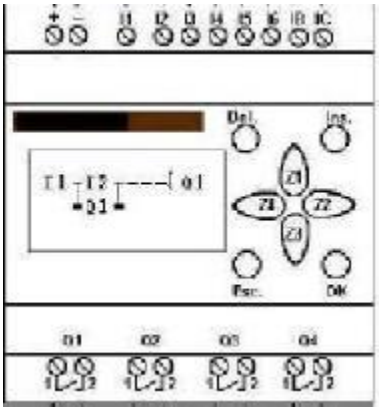


Fig. 130 – Equipos integrados.

Fig. 131 – Hand Held.

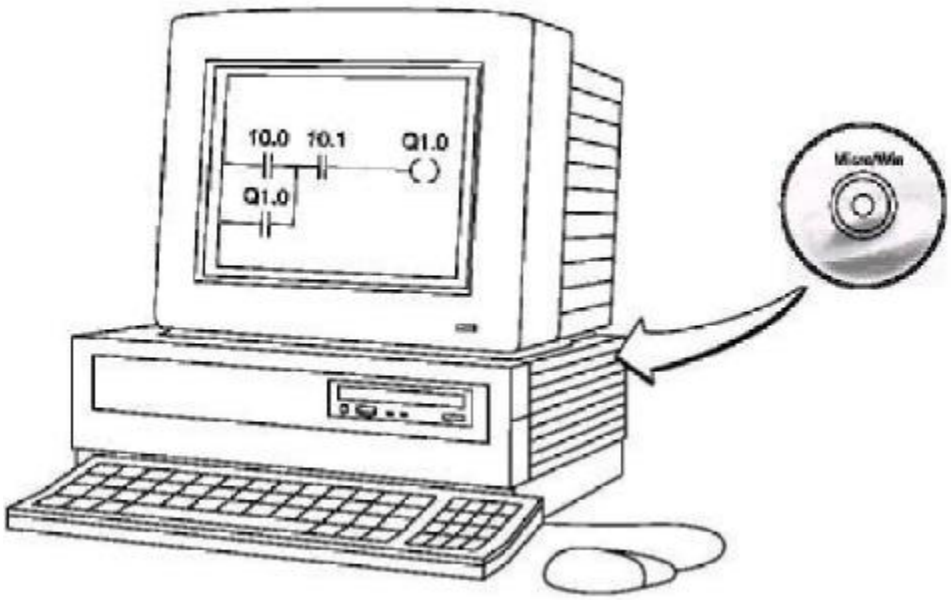


Fig. 133 – Programación mediante PCs + Software especializado.

a comunicación a través de un puerto de comunicaciones RS-232 es el método más popular para la comunicación de los PLCs con dispositivos externos. Este es un método de comunicación asincrónica que usa el sistema binario (1's y 0's) para transmitir los datos en un formato ASCII(American Standard Code for Information Interchange). Este código traduce el código humano (letras / números) a un código legible por las computadoras(1's y 0's). La transmisión y recepción de los datos se hace a través del puerto serial de los PLCs. Este puerto trabaja enviando y recibiendo señales de voltaje. **Un voltaje positivo se conoce como una MARCA, mientras que un voltaje negativo es un ESPACIO.** Típicamente los PLCs trabajan con +/- 15 voltios.

Existen 2 tipos de dispositivos RS-232. El primero es llamado DTE (Data Terminal Equipment) y un ejemplo de él es un computador. El segundo tipo de dispositivo es llamado DCE (Data Communications Equipment) y un ejemplo de él es un MODEM (Modulador / Demodulador). Los PLCs pueden ser tanto DTE como DCE.

El puerto serial del PLC trabaja poniendo algún pin en on (ALTO) mientras pone algún otro en off (BAJO). Cada uno de estos pines esta dedicado a un propósito específico. El puerto serial viene en dos presentaciones: un tipo de 25 pines y otro de 9 pines. Los pines y sus propósitos se muestran a continuación. (esta tabla asume que el equipo es un DTE).

- Este pin debe ser conectado internamente al chasis del dispositivo.

- **Recepción de datos**- este pin es por entra los datos provenientes de un dispositivo externo.

- **Transmisión de datos**- Este pin es por donde salen los datos rumbo a un dispositivo externo.

- **Terminal listo**- Este pin es el control maestro para el dispositivo externo. Cuando este pin esta en 1 el dispositivo externo ni transmite ni recibe datos.

- **Referencia de la señal**- Ya que los datos son enviados como voltajes positivos o como voltajes negativos, este pin sirve como referencia a ambas señales.

- **Equipo OK**- Usualmente los dispositivos externos mantienen este pin permanentemente en 0 y el PLC lo usa básicamente para determinar si el dispositivo externo esta encendido y listo.

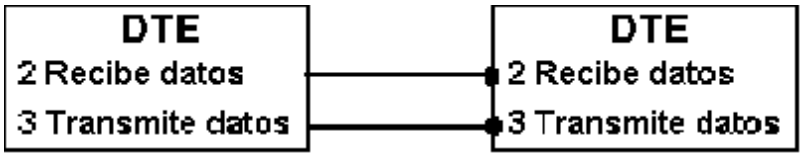


Fig.134 – Conexión incorrecta.

Nótese que en la figura de arriba la línea de recepción de datos (pin 2) del primer dispositivo esta conectada con la línea de recepción de datos del segundo dispositivo. Con la línea de transmisión de datos pasa algo similar excepto que es para la transmisión sobre el pin 3.La solución a este problema es usar una conexión tipo null-modem como las mostrada a continuación.

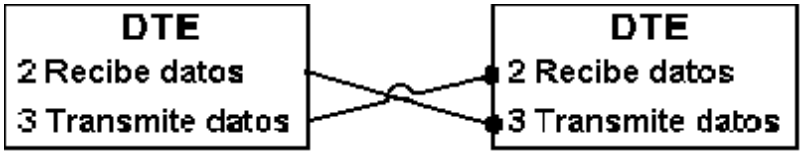


Fig. 135 – Conexión Null-Modem.

El dispositivo externo activa su pin DSR lo cual dice al PLC que está ahí y que está encendido. El PLC activa su pin RTS lo cual es igual que preguntar al dispositivo externo “Estas listo para recibir algunos datos”. El dispositivo externo responde activando su pin CTS con lo cual dice que esta OK que el PLC envíe datos. El PLC envía datos usando su pin TD terminal y el dispositivo externo la recibe por su pin RD. Algunos datos son enviados y recibidos. Después de un tiempo, El dispositivo externo no puede procesar los datos tan rápidamente, y por eso pone a 0 su pin CTS y el PLC congela el envío de datos. El dispositivo externo se pone al corriente y entonces pone nuevamente a 1 su pin CTS . El PLC reanuda el envío de datos a trabes del terminal TD terminal y el dispositivo externo los recibe a trabes de su terminal RD. El PLC se le acaban los datos por enviar y entonces pone a cero su pin RTS. El dispositivo externo para de recibir y entonces aguarda por mas datos futuros.

Bits Menos significantes	**5**		NAK	%	5	E	U	e	u
	6	ACK		&	6	F	V	f	v
	7			'	7	G	W	g	w
	8			(	8	H	X	h	x
	9			)	9	I	Y	i	y
	A	LF		*	:	J	Z	j	z
	B			+	;	K	[	k	{
	C			,	<	L	\	l	|
	D	CR		-	=	M	]	m	}
	E			.	>	N	^	n	~
	F			/	?	O	_	o	

Tabla38 – Tabla símbolos ASCII.

Paridad Ninguna: Como el bit de paridad es siempre 0, entonces se envía 1000101**0**.

Paridad par: Se debe tener una cantidad par de 1's. Como el carácter original tiene 3 1's (**1**000**1**0**1**) entonces el bit de paridad a adicionar debe ser 1 (1000101**1**). Ahora lo enviado si tiene una cantidad par de 1's.

Paridad impar: Se debe tener una cantidad impar de 1's. Como el carácter original tiene 3 1's (**1**000**1**0**1**), osea una cantidad impar, entonces el bit de paridad a adicionar debe ser 0 (1000101**0**).

Durante la comunicación el dispositivo que transmite calcula el bit de paridad y lo envía. Por su parte el receptor calcula también la paridad para el carácter de 7 bits y la compara con el bit de paridad recibido. Si los bits calculado y recibido no son iguales, ha ocurrido un error y entonces se ejecuta la rutina apropiada en este caso.

El método de chequeo de paridad no es muy popular. Esto se debe a que el mismo es efectivo sólo la mitad de las veces, ya que el chequeo de la paridad puede sólo detectar errores que afectan una cantidad impar de bits. Si el error

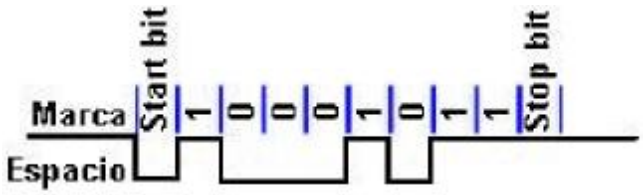


Fig. 36 – Tren de bits correspondiente al carácter “E”

Otra importante característica que algunas veces es usada para la sincronización entre dispositivos es el “control de flujo”, el cual se usa para asegurar que ambos dispositivos estén listos para enviar / recibir datos. El más popular “control de flujo de caracteres” se conoce como XON/XOFF. Simplemente cuando el receptor desea que el transmisor haga una pausa, él envía el carácter XOFF. Cuando el receptor desea recibir datos otra vez, él envía al transmisor el carácter XON.

Una última característica son los “delimitadores”, los cuales se agregan al final del mensaje para indicar al receptor que procese los datos que él ha recibido. El más popular es el CR(carriage return) o el par CR y LF(line feed). Cuando el PLC u otro dispositivo externo recibe estos delimitadores, sabe que tiene que tomar los datos del

- ¿Cómo indicarle al PLC que es el momento para enviar los datos por el puerto serial?
- ¿Como se sabrá que se ha recibido datos desde un dispositivo externo?

Como se nota, para establecer la comunicación con dispositivos externos sólo hay que escoger en la memoria disponible un área de trabajo, y seleccionar unos relés internos para enviar y recibir datos. Pero antes de realizar esto, considérese las siguientes definiciones.

- **Buffer**- Es una locación de memoria usada para almacenamiento temporal donde el PLC o el dispositivo externo hace lo propio con los datos recibidos o con los datos que están aguardando para ser enviado vía RS-232.
- **Cadena**- es una forma sencilla de referirse a un grupo de caracteres. La palabra “hola” es una cadena, ya que es un grupo de caracteres (h-o-l-a) que están relacionados entre sí para lograr un significado útil. "43770" también es una cadena aunque por ahora carezca de sentido alguno. Sin embargo la

Se debe asignar por ejemplo el relé interno 1000 para que funcione como nuestra bandera de arranque de transmisión. En otras palabras, cuando se active el relé 1000 el PLC enviará los datos contenido en DM100-DM102 por el puerto serial hacia el dispositivo externo. Nótese nuevamente que muchos PLCs cuentan con relés especiales para ejecutar lo mencionado.

En el siguiente ejemplo se enviará la cadena "alr" por el puerto serial del PLC hacia una interfase hombre-máquina cuando un sensor de temperatura se active indicando que el horno se ha sobrecalentado. Cuando el dispositivo externo que funciona como HMI reciba esta cadena, desplegará un mensaje de alarma para que sea vista por un operador. Al mirar la tabla **ASCII** se nota que “alr” en hexadecimal es igual a 61, 6C, 72 (a=61, l=6C, r=72). De aquí que se deba escribir en forma individual estos caracteres ASCII (en forma Hexadecimal) en las locaciones de memoria ya antes seleccionadas (DM100-102). Para ello se utilizarán las instrucciones LDA o MOV. Posteriormente, se activara el relé de envío cuando el sensor de temperatura (entrada 0000) se ponga en alto. El diagrama escalera resultante se muestra a continuación.

Algunos PLCs pueden no tener relés internos dedicados al envió de datos a trabes de puerto RS-232, y en consecuencia se deberá asignar manualmente por configuración. También, algunos PLCs cuentan con instrucciones especiales para indicar donde los datos han sido almacenados y para indicar cuando enviar los datos. Estas instrucciones son llamadas comúnmente como AWT (ASCII Write) o RS (Request to Send).

de los PLCs, siendo esta la característica que permite a los usuarios crear aplicaciones de control imposibles de realizar con los viejos sistemas a relés, además de que les permite la fácil reprogramación de los cambios que surjan en la lógica de control. Para lograr esta versatilidad de programación de la lógica de control, los PLCs disponen de varios lenguajes mediante el cual se pueden crear las aplicaciones en cuestión, pero de ellos el de más amplio uso es el lenguaje a contactos o diagrama escalera ya que le es el más familiar a los profesionales que están envuelto en las operaciones con PLCs.

Las herramientas de comunicación integrada a los PLCs no sólo mejora la posibilidad de explotación de los sistemas de control, sino que además abre sus aplicaciones hacia los sistemas integrados de manufactura y producción tales como: CIM, CAM, etc.

Finalmente, todo lo relacionado a los Controladores Lógicos Programables o PLCs, esta reunido bajo el estándar IEC-1131.

Julio 1986.

11. Terminal TSX T317. Programación PL7-1. Lenguaje Booleano. Telemecanique. Manual 1989.