UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE ESTUDIOS SUPERIORES ARAGON

"OPERACIÓN Y MANTENIMIENTO DE UN CONTROLADOR LOGICO PROGRAMABLE (PLC)".

T E S I S

QUE PARA OBTENER EL TITULO DE
INGENIERO MECÁNICO ELÉCTRICO
P R E S E N T A :

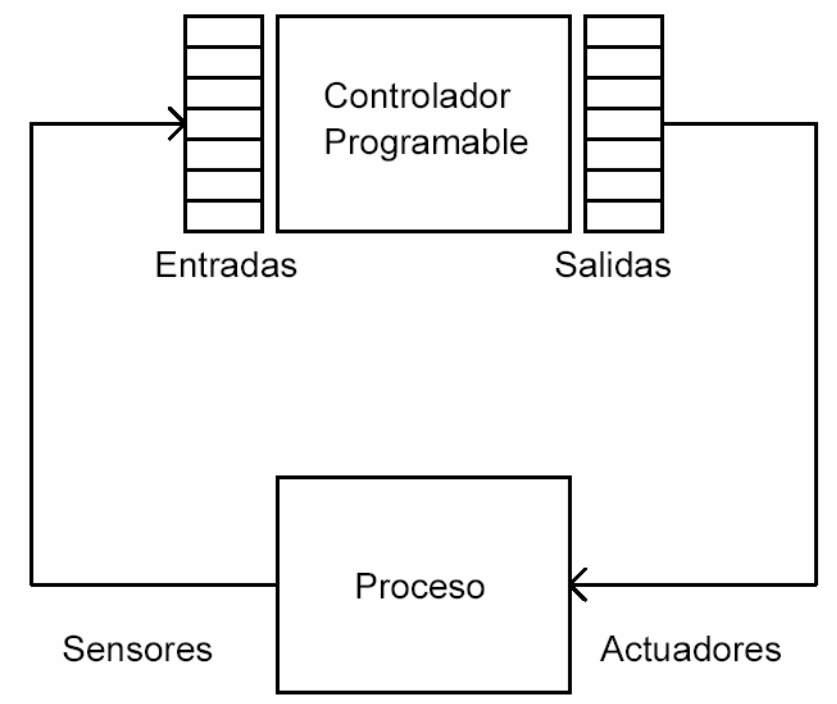
NERI MARÍN OLIVER.
Y
PALAFOX ESCOBEDO ALEJANDRO.

ASESOR: ING. BENITO BARRANCO CASTELLANOS

Estado de México 2008.

¿Qué es un PLC?

Un PLC (*Programable Logic Controller* - controlador lógico programable) es un dispositivo de estado sólido, diseñado para controlar secuencialmente procesos en tiempo real en un ámbito industrial.



Ejemplo del empleo de un PLC en un control de procesos.

Dentro de las funciones del PLC se puede mencionar:

. Adquirir datos del proceso por medio de las entradas digitales y analógicas.
. Tomar decisiones en base a reglas programadas.
. Almacenar datos en memoria.
. Generar ciclos de tiempo.
. Realizar cálculos matemáticos.
. Actuar sobre dispositivos externos mediante las salidas digitales y analógicas.
. Comunicarse con otros sistemas externos.

Desarrollo histórico

Los antecesores del PLC fueron los sistemas de control basados en relés (1960). Una aplicación típica de estos sistemas utilizaba un panel de 300 a 500 relés y miles de conexiones por medio de alambres, lo que implicaba un costo muy elevado en la instalación y el mantenimiento del sistema, estimado en US $30 a $50 por relé.

Luego surgieron los sistemas lógicos digitales construidos mediante circuitos integrados (1970), sin embargo eran productos diseñados para una aplicación específica y no eran controladores de propósitos generales. Muchos de ellos empleaban microprocesadores, pero su programación en un lenguaje poco familiar para los ingenieros de control (*Assembler*), hacía que el mantenimiento fuese inapropiado.

Los primeros controladores completamente programables fueron desarrollados en 1968 por la empresa de consultores en ingeniería Bedford y Asociados, que posteriormente pasó a llamarse MODICOM.

El primer Controlador Lógico Programable fue construido especialmente para la General Motors Hydramatic Division y se diseñó como un sistema de control con un computador dedicado.

Este primer modelo MODICOM, el 084, tuvo una gran cantidad de modificaciones, obteniéndose como resultado los modelos 184 y 384 desarrollados a principios de la década de los '70.

Con estos controladores de primera generación era posible:

. Realizar aplicaciones en ambientes industriales.
. Cambiar la lógica de control sin tener que cambiar la conexión de cables.
. Diagnosticar y reparar fácilmente los problemas ocurridos.

Los primeros PLC, que sólo incorporaban un procesador para programas sencillos y dispositivos de entrada/salida, evolucionaron hasta los equipos actuales, que integran:

. Módulos multiprocesadores.
. Entradas y salidas digitales de contacto seco, de relé o TTL.
. Entradas y salidas analógicas para corriente o voltaje.
. Puertas de comunicación serial o de red.
. Multiplexores análogos,
. Controladores PID.
. Interfaces con CTR, impresoras, teclados, medios de almacenamiento magnético.

Aplicaciones de los PLC

El PLC es usado en la actualidad en una amplia gama de aplicaciones de control, muchas de las cuales no eran económicamente posibles hace algunos años. Esto debido a:

. El costo efectivo por punto de entrada/salida ha disminuido con la caída del precio de los microprocesadores y los componentes relacionados.
. La capacidad de los controladores para resolver tareas complejas de computación y comunicación ha hecho posible el uso de PLC en aplicaciones donde antes era necesario dedicar un computador.

Existen 5 áreas generales de aplicación de PLC:

. Control secuencial.
. Control de movimiento.
. Control de procesos.
. Monitoreo y supervisión de procesos.
. Administración de datos.
. Comunicaciones.

Ejemplos de aplicaciones de PLC

- Annunciators
- Auto Insertion
- Bagging
- Baking
- Blending
- Boring
- Brewing
- Calendaring
- Casting
- Chemical Drilling
- Color Mixing
- Compressors
- Conveyors
- Cranes
- Crushing
- Cutting
- Digestors
- Drilling
- Electronic Testing
- Elevators
- Engine Test Stands
- Extrusion
- Forging
- Generators
- Gluing
- Grinding
- Heat Treating
- Injection Molding Assembly
- Motor Winding
- Oil Fields
- Painting
- Palltizers
- Pipelines
- Polishing
- Reactors
- Robots
- Rolling
- Security Systems
- Stretch Wrap
- Slitting
- Sorting
- Stackers
- Stitching
- Stack Precipitators
- Threading
- Tire Building
- Traffic Control
- Textile Machine
- Turbines
- Turning
- Weaving
- Web Handling
- Welding

CAPITULO I
CONTROL LÓGICO

1.1 Sistemas numéricos

Los sistemas numéricos son utilizados para la representación de números. Un sistema numérico de base n tiene n numerales, dígitos o símbolos distintos.
Mediante una combinación de los n dígitos es posible la representación de cualquier número. El sistema empleado por las personas es el decimal, debido al uso original de los diez dedos para contar. Sin embargo los sistemas digitales utilizan el sistema binario y sus derivados (octal y hexadecimal) ya que usan los *bits*: dígitos que sólo toman dos valores.

Sistema decimal

Está basado en 10 numerales o dígitos:

0, 1, 2, 3, 4, 5, 6, 7, 8 y 9.

Mediante estos dígitos es posible representar cualquier número. Por ejemplo la representación de 275910 es:

	10^3	10^2	10^1	10^0
	↓	↓	↓	↓
	1000	100	10	1
	↓	↓	↓	↓
	2	7	5	9
2759_{10} =	2 x 1000+	7x100+	5x10+	9x1

Los sistemas numéricos se basan en un sistema posicional ponderado. El valor del dígito depende de su posición.
El dígito de mayor ponderación es denominado MSD (*Most Significant Digit*), y se ubica en la primera posición de izquierda a derecha.
El dígito de menor ponderación se denomina LSD (*Least Significant Digit*), y se ubica en la posición del extremo derecho.

2 unidades 7 centenas 5 decenas 9 unidades de mil

Sistema binario.

Está basado en los dígitos 0 y 1, de modo que cualquier cifra entera puede ser representada por medio de estos 2 numerales. Por ejemplo la representación de 110112 es:

		2^4	2^3	2^2	2^1	2^0
		↓	↓	↓	↓	↓
		16	8	4	2	1
		↓	↓	↓	↓	↓
		1	1	0	1	1
11011_2	=	1x16 +	1x8 +	0x4 +	1x2 +	1x1
	=	27_{10}				

El número 11011 en base 2 es el número 27 en base 10.
Números típicos en sistema binario:

b7	b6	b5	b4	b3	b2	b1	b0	Bit
128	64	32	16	8	4	2	1	← Ponderación
0	0	0	0	0	0	0	1	$= 17_{10}$
0	0	0	0	0	0	1	0	$= 27_{10}$
0	0	0	0	0	1	0	0	$= 47_{10}$
0	0	0	0	1	1	1	1	$= 15_{10}$
1	0	0	0	0	0	0	0	$= 128_{10}$
1	1	1	1	1	1	1	1	$= 255_{10}$

En los computadores digitales se utilizan niveles de voltajes para las representaciones. Normalmente se adoptan los siguientes valores (niveles TTL).

Desde	Hasta		Representa
0.0 Volt	0.4 Volt	→	0 lógico
0.4 Volt	2.4 Volt	→	Incertidumbre
2.4 Volt	5.0 Volt	→	1 lógico

En general, 0 lógico = nivel bajo y 1 lógico = nivel alto.

Sistema BCD (*Binary - Coded - Decimal*)

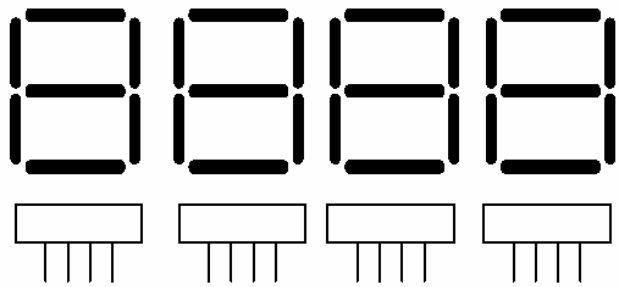
El sistema numérico BCD se basa en ponderaciones 8-4-2-1 empleando esta tabla.

Número	BCD
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001

Así por ejemplo el número 831 es representado en BCD como:

83110 = 1000 0011 0001BCD

El sistema BCD es ampliamente utilizado en visualizadores digitales (*Displays*), donde cada dígito es codificado por separado.



Display de 4 dígitos.

Sistema Octal

Se basa en ocho dígitos: 0, 1, 2, 3, 4, 5, 6 y 7. Por ejemplo la representación de 3758 es:

	8^2	8^1	8^0
	↓	↓	↓
	64	8	1
	↓	↓	↓
	3	7	5
375_8 =	3x64 +	7x8 +	5x1
=	253_{10}		

La transformación de octal a binario se obtiene como:

421	421	421	Ponderación binaria
3= 0+2+1	7= 4+2+1	5= 4+0+1	Representación octal
011	111	101	Representación binaria

Es decir,

3758 = 25310 = 01111110$_{12}$

Sistema Hexadecimal

Se basa en los 16 dígitos:

0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E y F.

Cuya equivalencia en el sistema decimal es:

Hexadecimal	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Decimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Por ejemplo la representación de B1316 es:

$$
\begin{array}{lll}
 & 16^2 & 16^1 & 16^0 \\
 & \downarrow & \downarrow & \downarrow \\
 & 256 & 16 & 1 \\
 & \downarrow & \downarrow & \downarrow \\
 & B_{16} & 1_{16} & 3_{16} \\
 & \downarrow & \downarrow & \downarrow \\
 & 11_{10} & 1_{10} & 3_{10} \\
B13_{16} = & 11 \times 256 + 1 \times 16 + & 3 \times 1 \\
= & 2832_{10} &
\end{array}
$$

La transformación de un número hexadecimal a uno binario se obtiene de la siguiente manera:

8421	8421	8421	Ponderación binaria
B=11= 8+0+2+1	1= 0+0+0+1	3= 0+0+2+1	Representación hexadecimal
1011	0001	0011	Representación binaria

Es decir,

$$B1316 = 283210 = 1011000100112$$

Conversión decimal

Para convertir un número de sistema decimal a sistema binario se divide el número por 2, el resto representa el dígito binario de menor ponderación, el resultado se divide nuevamente por 2 hasta que el resultado sea cero. Por ejemplo, se desea convertir 2910 a código binario:

Operación	Resultado	Resto
29/2=	14	1
14/2=	7	0
7/2=	3	1
3/2=	1	1
1/2=	0	1

Representación binaria: 11101_2

Si el microprocesador utiliza 8 ó 16 bits, se debe anteponer tantos ceros como sea necesario.

Para convertir de decimal a octal se utiliza el mismo método dividiendo por 8. Por ejemplo 22910 es:

Operación	Resultado	Resto
229/8=	28	5
28/8=	3	4
3/8=	0	3

Representación octal: 345_8

Para convertir de decimal a hexadecimal se divide por 16. Por ejemplo 22710 es:

Operación	Resultado	Resto
227/16=	14	3
14/16=	0	$14=E_{16}$

Representación hexadecimal: $E3_{16}$

1.2 Aritmética binaria

Las operaciones binarias básicas son la adición o suma y la sustracción o resta.

Suma binaria

La suma de dos números binarios es:

Operación	Arrastre	Resultado
0 + 0	0	0
0 + 1	0	1
1 + 0	0	1
1 +1	1	0

Por ejemplo:

$$\begin{array}{rr} 1\,0\,1\,0 & 10_{10} \\ +\ 0\,1\,1\,1 & +\ 7_{10} \\ \hline 1\,0\,0\,0\,1 & 17_{10} \end{array}$$

Resta binaria

Para restar números binarios se utiliza el método conocido como complemento dos:

El sustraendo se convierte en su equivalente negativo y luego se suma al minuendo. Es decir:

$$A + B = A + (-B)$$

Para formar el número negativo equivalente:

. *Complemento 1: Se cambia cada bit por su complemento. Por ejemplo:*

$$010110 \rightarrow 101001$$

. *Se adiciona 1 al resultado anterior.*

Por ejemplo:

Realizar la resta: 1110 - 710

1) Representación de 710 en binario: 00000111
2) Complemento 1 de 710: 11111000
3) Complemento 2 de 710: Adición de 1 11111001
4) Representación de 1110 en binario: 00001011
5) Adición de 1110 y (-710): [1] 00000100 = 410
(El arrastre 1 no se considera.)

1.3 Números binarios con y sin signo

Los números binarios pueden ser representados con y sin signo.

Números binarios sin signo

Un número binario de 8 bits sin signo se expresa:

b7	b6	b5	b4	b3	b2	b1	b0	Posición
2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0	Ponderación
128	64	32	16	8	4	2	1	

Por lo tanto el rango de variación del número binario sin signo, en su equivalente decimal es:

$$00000000_2 = 0_{10}$$
$$11111111_2 = 255_{10}$$

Números binarios con signo

En números binarios con signo se utiliza comúnmente la notación complemento 2. En esta notación para números de 8 bits, el bit b7 indica el signo. El rango es -128_{10} a 127_{10}.

Si b7 = 1, el número es negativo.
Si b7 = 0, el número es positivo.

Algunos valores típicos en complemento 2 son:

b7	b6	b5	b4	b3	b2	b1	b0	Número decimal
1	0	0	0	0	0	0	0	-128
1	0	0	0	0	0	0	1	-127
1	1	1	1	1	1	1	0	-2
1	1	1	1	1	1	1	1	-1
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	1	1
0	0	0	0	0	0	1	0	2
0	1	1	1	1	1	1	1	127

1.4 Circuitos lógicos

El diseño de circuitos lógicos se basa en la operación de variables digitales que sólo pueden tomar dos estados posibles:

ABIERTO o CERRADO
APAGADO o ENCENDIDO
BLANCO o NEGRO
OFF o ON

La expresión matemática de estos conceptos requiere de los números binarios:

A = 0 → FALSO, OFF, CONTACTO ABIERTO, RELÉ DESENERGIZADO, LÁMPARA APAGADA.
A = 1 → VERDADERO, ON, CONTACTO CERRADO, RELÉ ENERGIZADO, LÁMPARA ENCENDIDA.

El estado de un relé o contacto se identifica según su condición normal:

NO = *Normally open* - normalmente abierto
NC = *Normally close* - normalmente cerrado

Operadores NOT, AND y OR

Las operaciones matemáticas binarias se realizan con los operadores NOT, AND y OR.

NOT

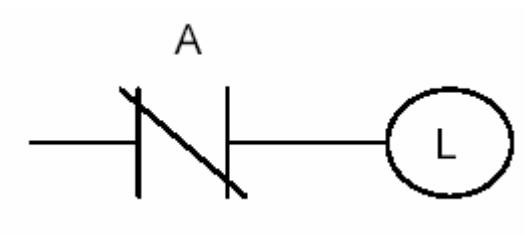
El operador NOT denota una salida verdadera si la entrada es falsa, y una salida falsa si la entrada es verdadera. Las distintas nomenclaturas son:

L = NOT A L = $\overline{A}$

Tabla de verdad para el operador NOT:

A	L
0	1
1	0

El circuito con un contacto NC representa el concepto lógico AND:



La lámpara L se encenderá sólo cuando A no esté conectado.

AND

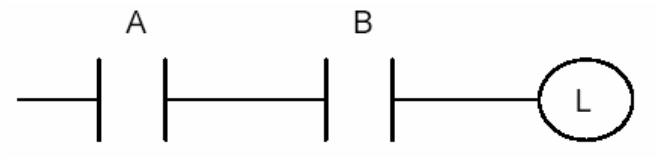
El operador AND denota una salida verdadera si y sólo si sus entradas son verdaderas. Las distintas nomenclaturas son:

L = A AND B L = AB L = A*B

Tabla de verdad para el operador AND:

A	B	L
0	0	0
0	1	0
1	0	0
1	1	1

El circuito serie representa el concepto lógico AND:



La lámpara L se encenderá sólo si los contactos A y B están cerrados.

OR

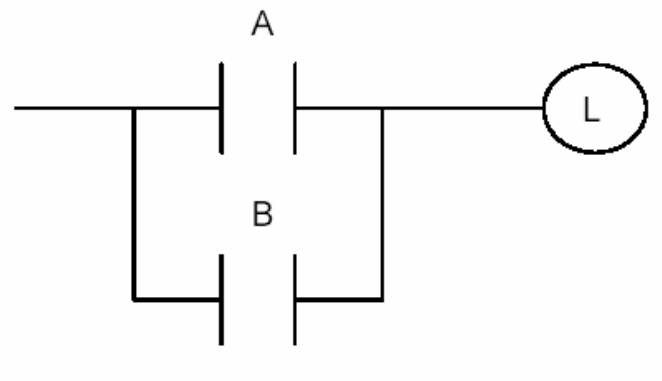
El operador OR denota una salida verdadera si hay alguna de las entradas (o ambas) verdaderas. Las distintas nomenclaturas son:

L = A OR B L = A + B

Tabla de verdad para el operador OR:

A	B	L
0	0	0
0	1	1
1	0	1
1	1	1

El circuito paralelo representa el concepto lógico OR:



La lámpara L se encenderá si alguno de los contactos A ó B (o ambos) está(n) cerrado(s).

Ejemplos

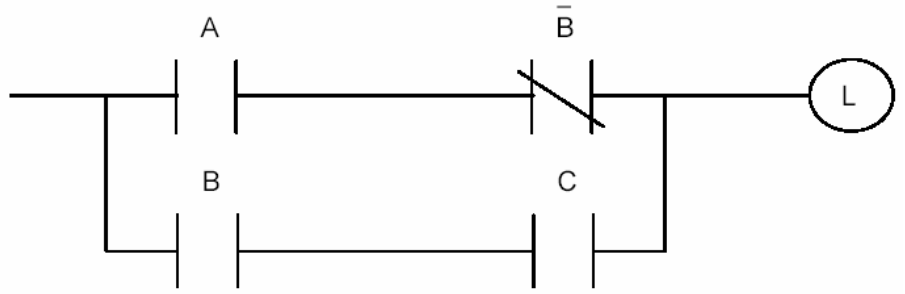
Existen operaciones binarias más complicadas, tales como:

1) $L = AB + \overline{BC}$

Esta función se basa en una combinación de operadores AND, OR y NOT. La tabla de verdad se representa a continuación:

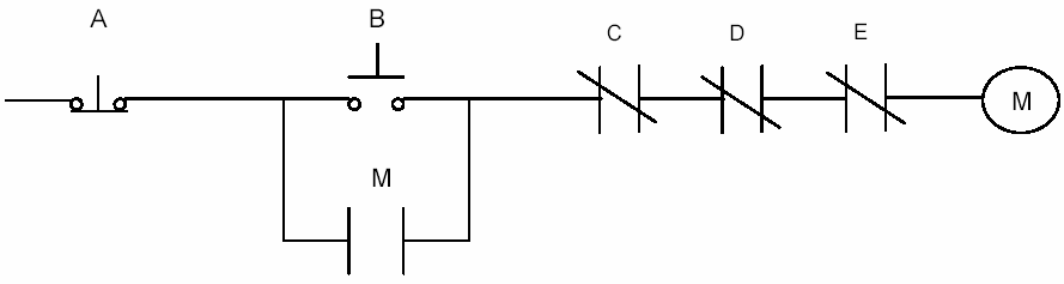
A	B	C	$A\bar{B}$	BC	L
0	0	0	0	0	0
0	0	1	0	0	0
0	1	0	0	0	0
0	1	1	0	1	1
1	0	0	1	0	1
1	0	1	1	0	1
1	1	0	0	0	0
1	1	1	0	1	1

El diagrama circuital es:



2) $M = A (B + M) C D E$

Esta función corresponde al sistema de partida/parada del motor M. El diagrama circuital se representa en el siguiente esquema:

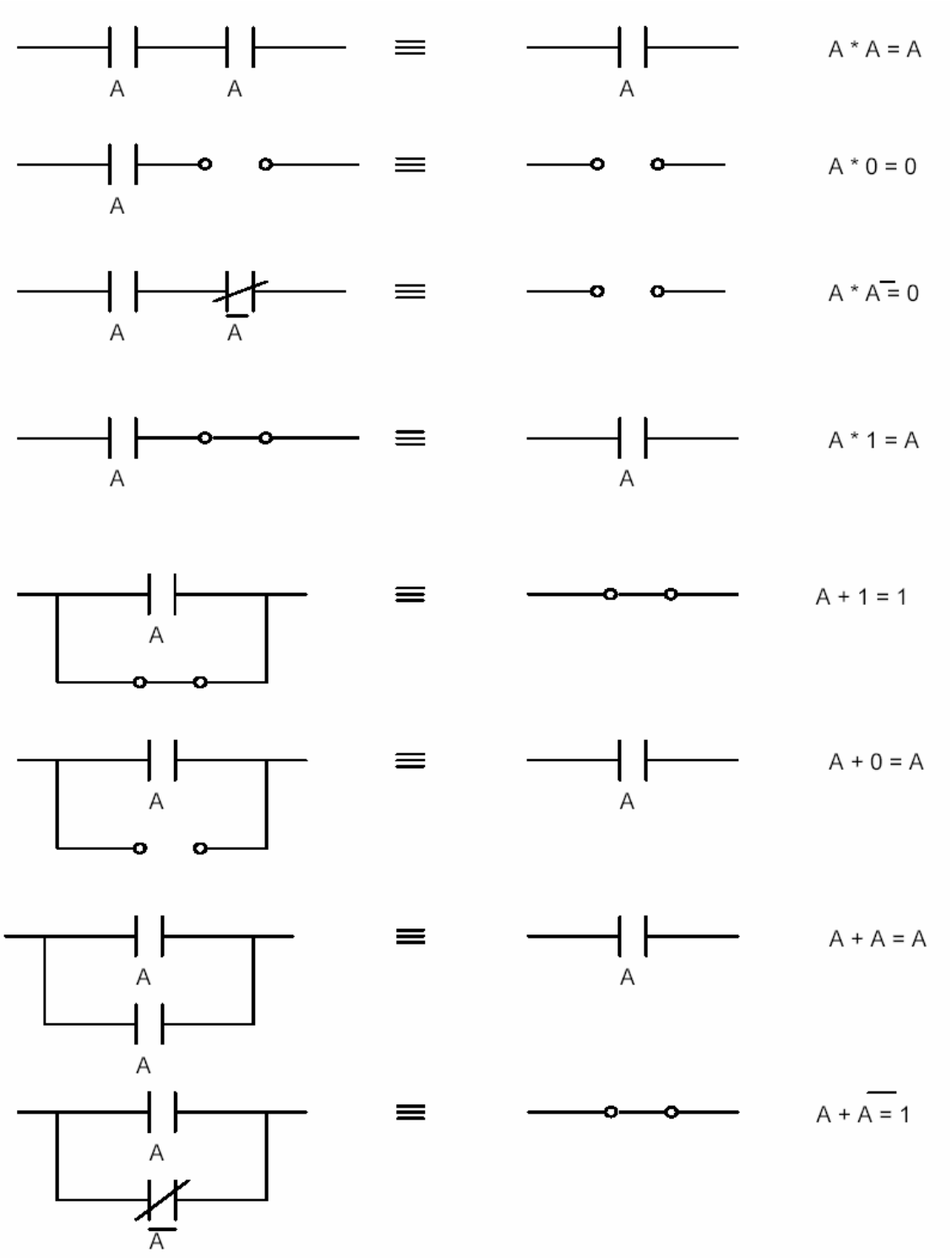


El motor (M) partirá sólo si el botón de parada (A) no está presionado y las sobrecargas (C, D, E) están cerradas y el botón de partida está presionado (B) o el contacto (M) está cerrado.

Se observa que en este sistema existe una realimentación de la variable M, que es la forma de realizar un elemento de memoria. Al encender el motor M presionando el pulsador B, se cerrará el contacto M. De esta forma al levantar B, el motor sigue energizado por el lazo B OR M.

Postulados lógicos Booleanos

Existen diversos postulados utilizados para simplificar las operaciones algebraicas. Entre ellos se puede mencionar.



Teoremas Booleanos

Los teoremas Booleanos permiten reducir la complejidad y extensión de las expresiones lógicas.

Teoremas de Morgan

Los teoremas de Morgan son:

Para la adición: $A + B = \overline{A * B}$ $\quad \overline{\quad} \quad \overline{\quad}$

Para la multiplicación: $\overline{A * B} = \overline{A} + \overline{B}$

Se pueden utilizar los teoremas de Morgan en operaciones más complejas, como por ejemplo:

$$\begin{aligned}\overline{(A+\overline{B}C)D\overline{E}} &= \overline{(A+\overline{B}C)} + \overline{D\overline{E}} \\ &= \overline{A}\overline{\overline{B}C} + \overline{D} + E \\ &= \overline{A}(B+\overline{C}) + \overline{D} + E\end{aligned}$$

Teorema del término incluido

El teorema del término incluido establece que si un término de una expresión algebraica binaria está totalmente incluido en otro término, entonces el término mayor es redundante. Es decir,

$$L = A + AB = A$$

Ejemplo:

$$L = AB + ABCD = AB$$

Teorema de los productos opcionales

Se dice que un producto es opcional si su presencia o ausencia no afecta al resultado.

Ejemplo:

$$L = AB + A\overline{B} = A\,(B + \overline{B}) = A$$

Teorema de las sumas opcionales

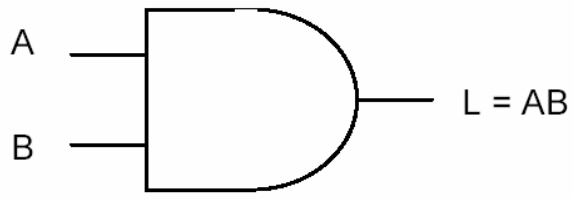
Una suma es opcional si su presencia o ausencia en la expresión Booleana no altera el resultado.

Ejemplo:

$$L = A\,(A + B) = A + AB = A\,(1 + B) = A$$

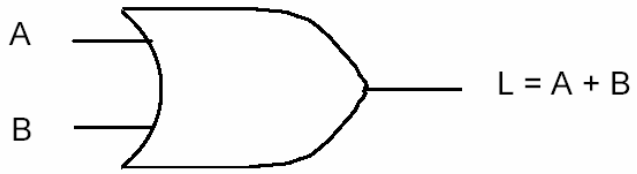
1.5 Compuertas lógicas

Puerta AND



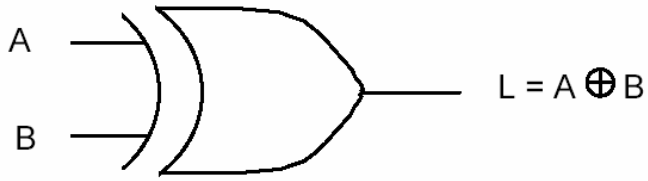
A	B	L
0	0	0
0	1	0
1	0	0
1	1	1

Puerta OR (inclusivo)



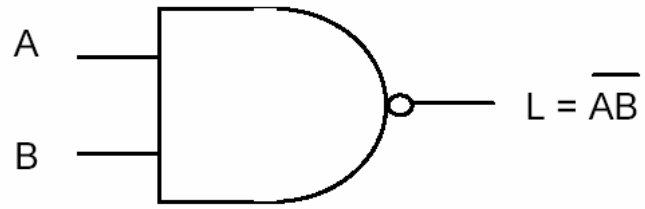
A	B	L
0	0	0
0	1	1
1	0	1
1	1	1

Puerta OR exclusivo



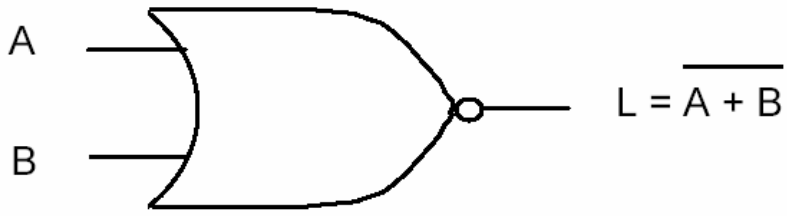
A	B	L
0	0	0
0	1	1
1	0	1
1	1	0

Puerta NAND



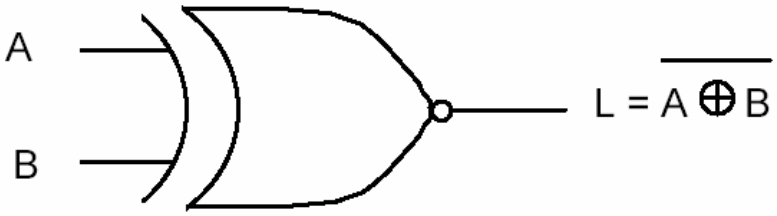
A	B	L
0	0	1
0	1	1
1	0	1
1	1	0

Puerta NOR



A	B	L
0	0	1
0	1	0
1	0	0
1	1	0

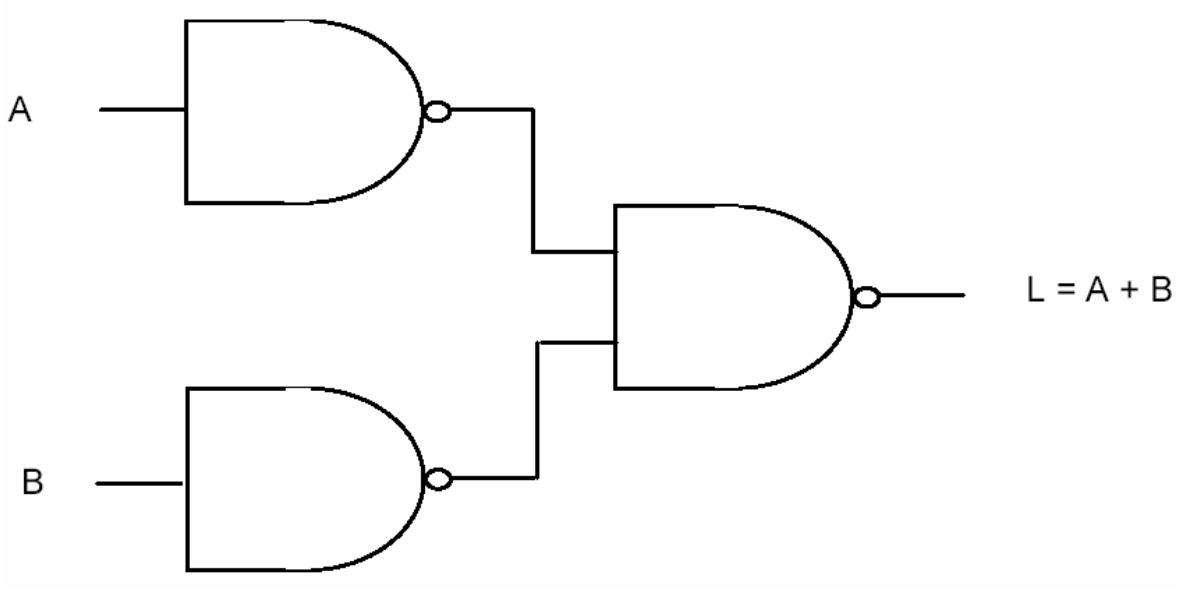
Puerta NOR exclusivo



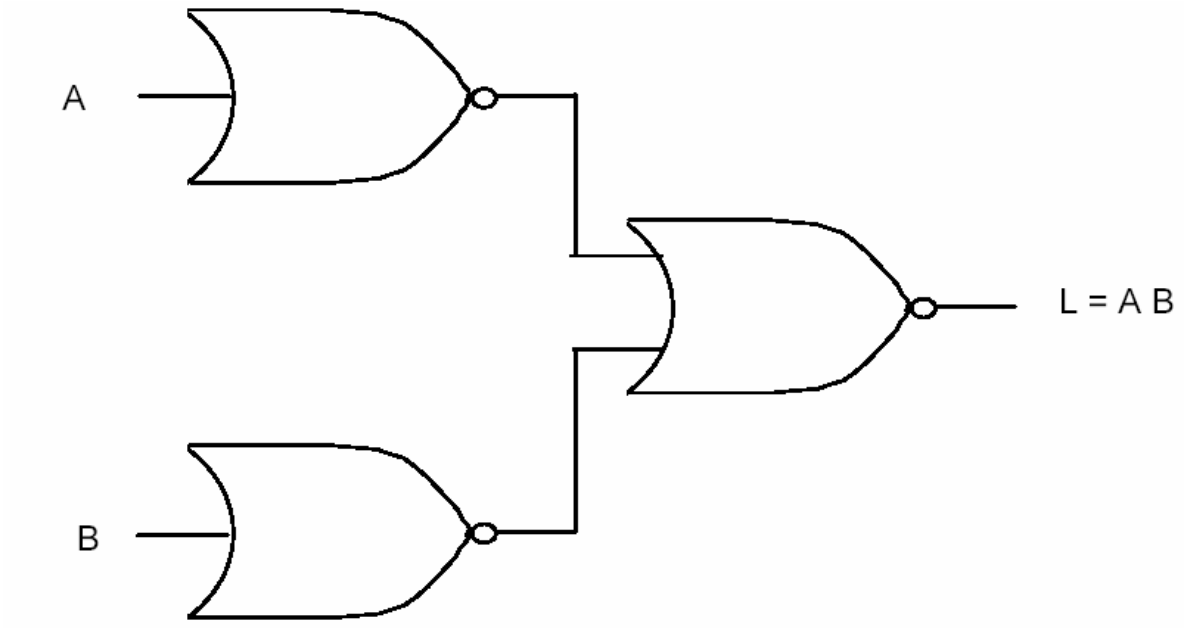
A	B	L
0	0	1
0	1	0
1	0	0
1	1	1

Ejemplos

Construcción de una puerta OR mediante puertas NAND:

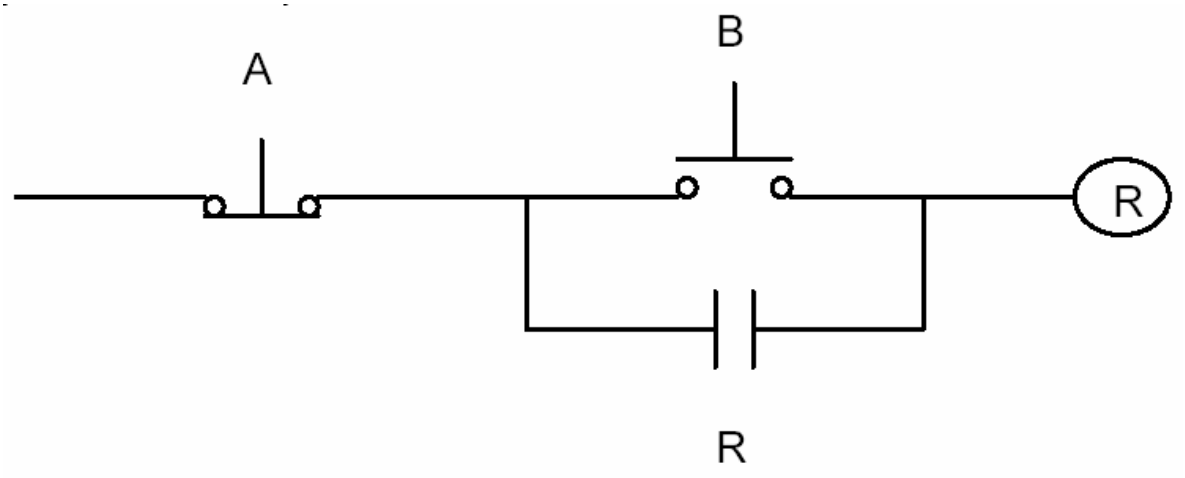


Construcción de una puerta AND mediante puertas NOR:



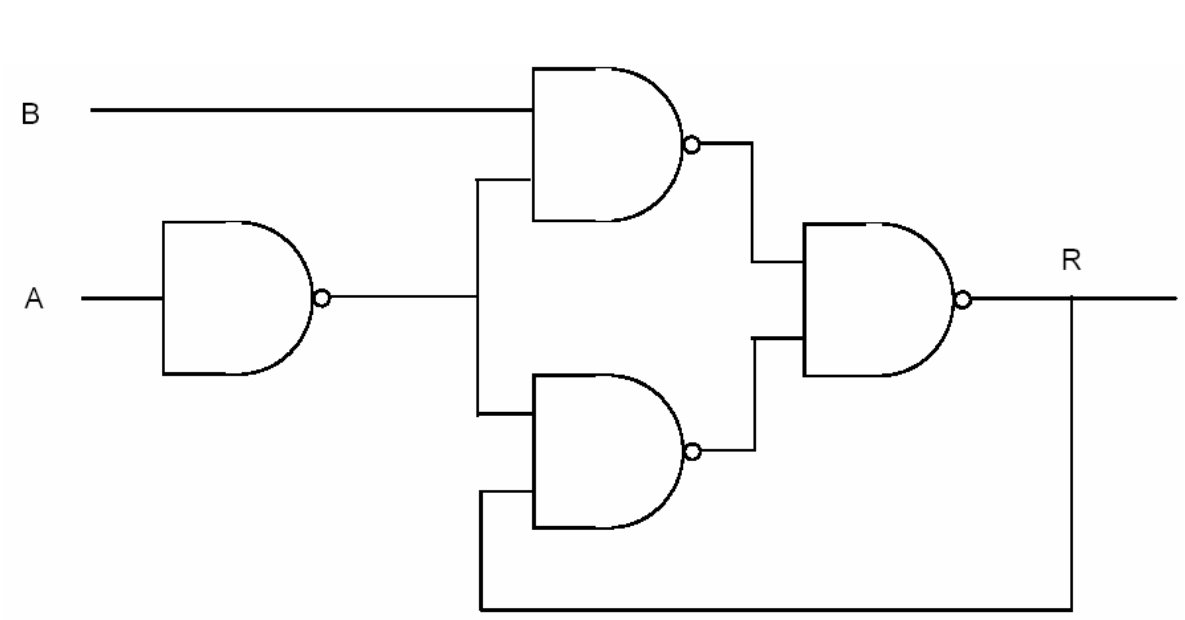
Ejemplo

Diseñar el circuito lógico correspondiente a la figura emplenado sólo puertas NAND.



Expresión lógica:

$$R = \overline{A}\,(B + R) = \overline{A}B + \overline{A}R$$

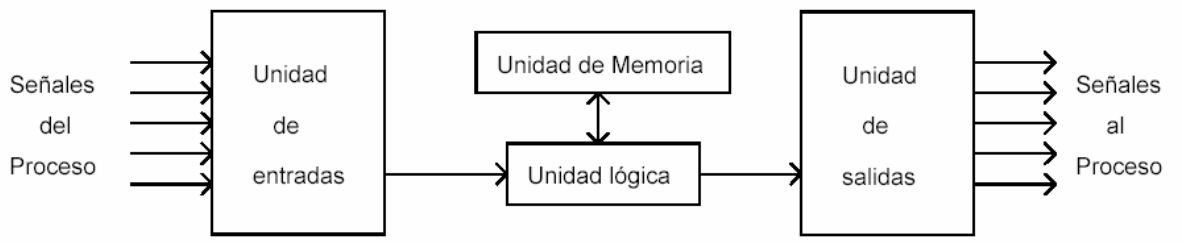


1.6 ESTRUCTURA BÁSICA DE UN PLC

Unidades funcionales

Un controlador lógico programable se compone de cuatro unidades funcionales:

- Unidad de entradas
- Unidad de salidas
- Unidad lógica
- Unidad de memoria



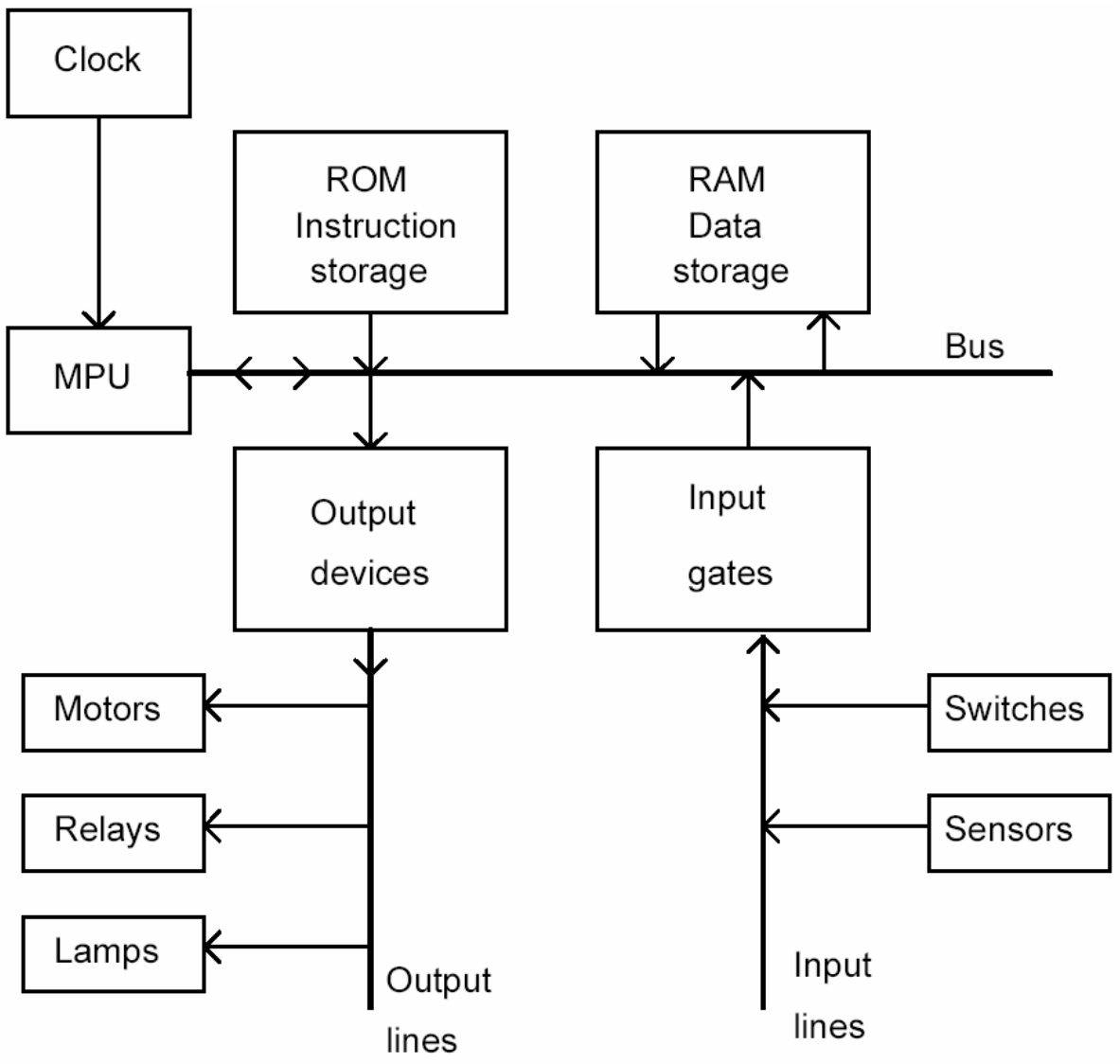


Diagrama de un PLC con dispositivos de entrada y salida.

Unidad de Entradas

Proporciona el aislamiento eléctrico necesario y realiza el acondicionamiento de las señales eléctricas de voltaje, provenientes de los *switches* de contactos ON-OFF de terreno. Las señales se adecúan a los niveles lógicos de voltaje de la Unidad Lógica.

Unidad de Salidas

Acepta las señales lógicas provenientes de la Unidad Lógica, en los rangos de voltaje que le son propios y proporciona la aislación eléctrica a los *switches* de contactos que se comandan hacia terreno.

Las unidades de entrada/salida del PLC, son funcionalmente iguales a los bancos de relés, que se empleaban en los antiguos controladores lógicos de tipo tambor. La diferencia radica en que las unidades de entrada/salida de los PLC son de estado sólido.

La eliminación de contactos mecánicos se traduce en una mayor velocidad de operación y mayor tiempo entre fallas (MTBF).

Unidad Lógica

El *corazón* de un PLC es la Unidad Lógica, basada en un microprocesador. Ejecuta las instrucciones programadas en memoria, para desarrollar los esquemas de control lógico que se especifican.

Algunos equipos antiguos implementan la unidad lógica en base a elementos discretos: compuertas NAND, NOR, FLIPFLOP, CONTADORES como máquinas de estado. Este tipo de controladores son *HARDWIRE*, versus aquellos que utilizan memorias, denominados *SOFTWIRE*.

Memoria

Almacena el código de mensajes o instrucciones que ejecuta la Unidad Lógica. La memoria se divide en PROM o ROM y RAM.

ROM: Memoria de sólo lectura (*Read Only Memory*). Memoria no volátil que puede ser leída pero no escrita. Es utilizada para almacenar programas y datos necesarios para la operación de un sistema basado en microprocesadores.

RAM: Memoria de acceso aleatorio (*Random Access Memory*). Memoria volátil que puede ser leída y escrita según sea la aplicación. Cualquier posición de memoria puede ser accesada en cualquier momento.

Por medio de ellas, se puede utilizar un PLC en procesos diferentes sin necesidad de readecuar o transformar el equipo; sólo se debe modificar el programa. Para el control de un proceso *BATCH*, se pueden almacenar varias recetas en la memoria y accesar aquélla que interesa.

Las PROM o ROM almacenan los programas permanentes que coordinan y administran los recursos del equipo.

La RAM guarda los programas de aplicación que pueden sufrir modificaciones. Esta memoria es respaldada con baterías, con el propósito de no perder la información al existir cortes de fluido eléctrico.

El sistema opera a través de la interacción con el procesador (Unidad Lógica) y la Memoria.

Cuando se enciende el equipo, el procesador lee la primera palabra de código (instrucción) almacenada en memoria y la ejecuta.

Una vez que termina de ejecutar la instrucción leída, busca en memoria la siguiente instrucción y así sucesivamente hasta que se completa la tarea.

Esta operación se llama ciclo de búsqueda-ejecución (FETCHEXECUTE CYCLE).

1.7 Interfaces de Estado Sólido

La función de los módulos de entrada y salida es conectar el PLC con el mundo exterior de los motores, *switches* límites, alumbrados, y dispositivos de medición. Estos módulos se realizan a través de elementos de estado sólido.

Las aplicaciones iniciales de dispositivos de estado sólido en el control de partida de equipos de potencia, se remontan a la década de 1950 con la utilización de diodos y transistores.

Sin embargo, en la práctica las aplicaciones comenzaron en 1957 con la aparición del primer rectificador controlado de silicio (SCR).

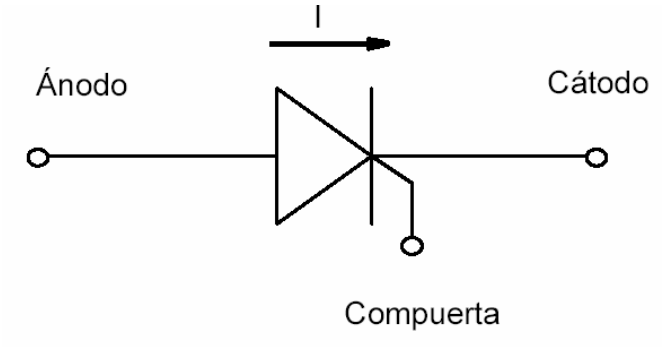
Los componentes de estado sólido empleados en las aplicaciones de control industrial, han reemplazando a los relés mecánicos en muchas de las funciones que llevaban a cabo.

Los dispositivos de estado sólido presentan muchas ventajas respecto a los relés, tales como alta velocidad de operación, pequeño tamaño y bajo consumo de potencia.

Sin embargo, son eléctricamente menos robustos y más sensibles a temperaturas elevadas y a la interferencia electromagnética (EMI).

Rectificador controlado de silicio SCR

El SCR, o denominado también tiristor, es utilizado como un interruptor electrónico que deja pasar corriente en un solo sentido.



El SCR, al recibir un pulso por la compuerta, deja pasar corriente sólo en el sentido ánodo → cátodo, en este caso su comportamiento es similar al del diodo.

Condiciones para el inicio de la conducción de un SCR:

1) Ánodo positivo respecto al cátodo.
2) Pulso positivo entre la compuerta y el cátodo.

El SCR permanecerá en el modo de conducción mientras el valor de la corriente esté por encima del valor crítico mínimo y se mantenga la diferencia de potencia positiva del ánodo con respecto al cátodo.

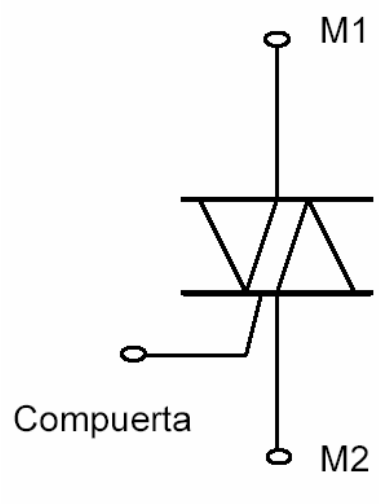
El SCR también entrará en conducción si la tensión ánodo cátodo sobrepasa los límites específicos del SCR (conducción por avalancha).

Generalmente, se emplea el SCR en circuitos de corriente alterna (AC). Mediante un pulso de control en la compuerta, que debe aplicarse durante el medio ciclo positivo, el SCR entra en conducción.

Existen diversos circuitos electrónicos utilizados para enviar los pulsos correspondientes a la compuerta del SCR. Algunos de ellos emplean microprocesadores, circuitos temporizadores, sensores de fase, UJT, etc.

El TRIAC

El TRIAC se utiliza como un interruptor electrónico que deja pasar corriente en ambos sentidos. Su construcción es la de dos SCR conectados en anti-paralelo.



El TRIAC tiene un amplio campo de uso en cargas de motores AC, ya que puede conducir en ambos semi-ciclos de voltaje alterno.

En comparación con los relés, el TRIAC resulta ser más sensible a la tensión aplicada, a la corriente y a la disipación interna de potencia. Una mala operación pude dañar el dispositivo para siempre.

Efectos del ruido

Se define el ruido como toda señal eléctrica indeseada, que puede entrar al equipo por diferentes vías. El ruido abarca el espectro completo de frecuencia y no presenta una forma de onda determinada.

El ruido eléctrico puede ocasionarles serios problemas de funcionamiento a los equipos de estado sólido, a causa de los bajos niveles de señal con que funcionan.

El ruido puede corresponder a alguno de los tres tipos básicos que se indican:

. Ruido transmitido, propio de la señal original.

. Ruido inherente, producto de los elementos que se integran en un sistema de adquisición de datos.
. Ruido inducido, originado por las fuentes de poder, acoplamientos magnéticos y acoplamientos electrostáticos.

Algunas medidas que deben tenerse en cuenta para reducir el acoplamiento del ruido eléctrico son:

. Usar encapsulados metálicos adecuados (jaula Faraday).
. Canalizar las líneas de control de los dispositivos de estado sólido en forma separada de las líneas de poder.
. Utilizar cables apantallados y trenzados, que proporcionan un escudo adecuado contra el acoplamiento electrostático y magnético.

El empleo de filtros adecuados permitirá eliminar el ruido indeseado de la señal.

Consideraciones especiales

Los componentes de estado sólido presentan una alta confiabilidad cuando se utilizan en los rangos y condiciones de operación adecuados.

La vida media de un TRIAC puede ser, por ejemplo, de 450.000 horas o 50 años, considerando condiciones de operación típicas. Sin embargo, puede fallar en forma aleatoria, incluso si se emplea dentro de los rangos de operación de diseño.

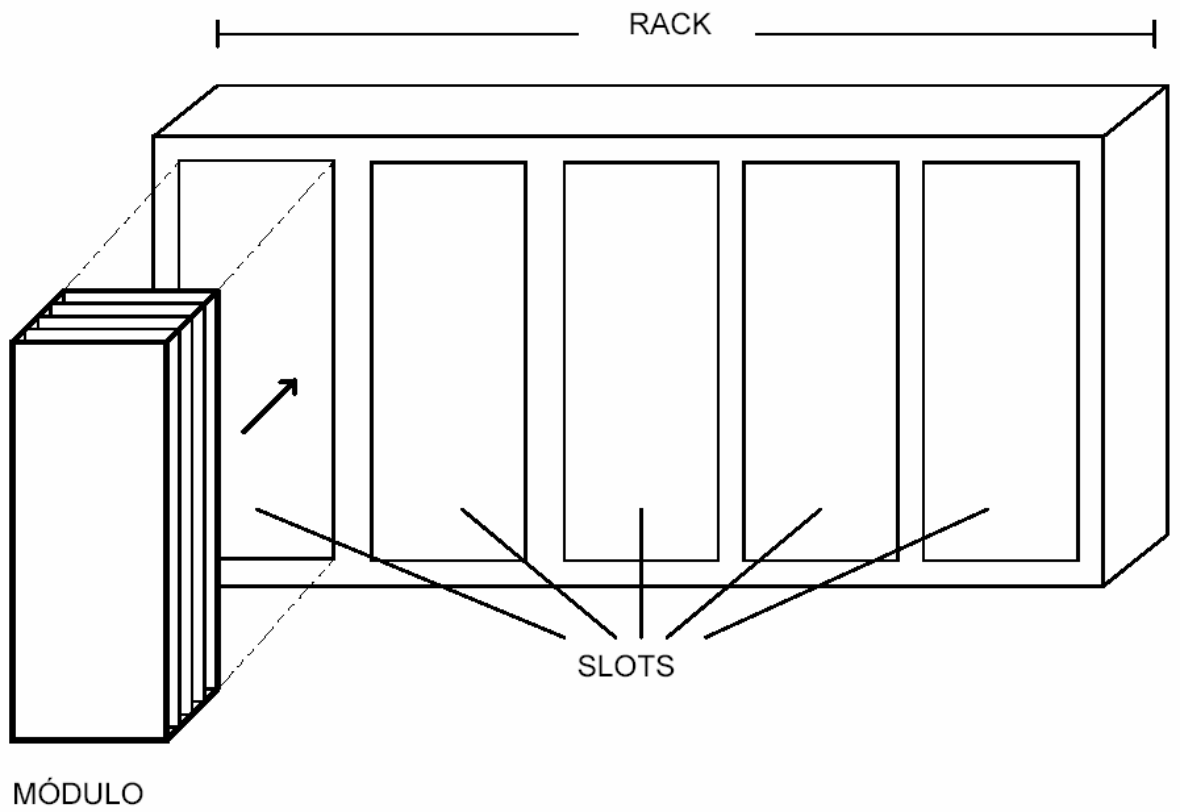
No es posible predecir cuándo va a fallar un componente de estado sólido cualquiera, como en el caso de los relés mecánicos, en los que observando su comportamiento se puede conocer el estado operacional.

Los controladores lógicos programables consideran las limitaciones y ventajas de los elementos de estado sólido, de modo que se minimizan los efectos del ruido. Generalmente, los PLC emplean rutinas de autodiagnósticos y verifican constantemente el funcionamiento correcto de los dispositivos de I/O.

1.8 Administración de entradas y salidas de un PLC

Bases del montaje

El montaje de los diversos módulos del PLC se realiza en *slots* ubicados en *racks*.



Los módulos básicos de un PLC son:

. Fuente de poder
. CPU
. Interfaces de entrada y salida

Dependiendo del modelo y la marca, existen en el mercado *racks* de diversos tamaños, como por ejemplo 4, 6, 8, 12, 14 y 16 *slots*. Según la aplicación se debe escoger el tamaño adecuado. Es posible instalar un módulo de ampliación, el que permite la conexión de un *rack* adicional.

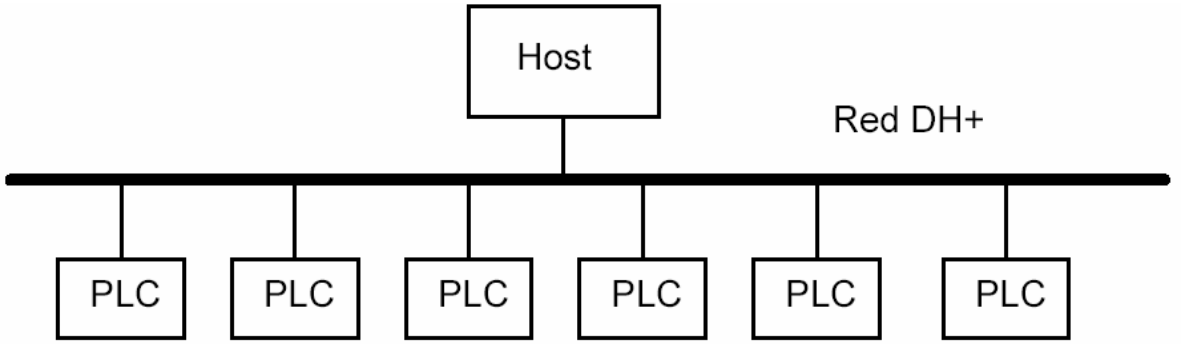
Otros módulos existentes son:

. Módulos de comunicaciones (TCP/IP, DH+, etc.)
. Módulos de control de redundancia
. Módulos para conexión de *racks* remotos

. Módulos de interfaz hombre-máquina (teclado, monitores, etc.)
. Módulos de almacenamiento de información
. Módulos controladores PID

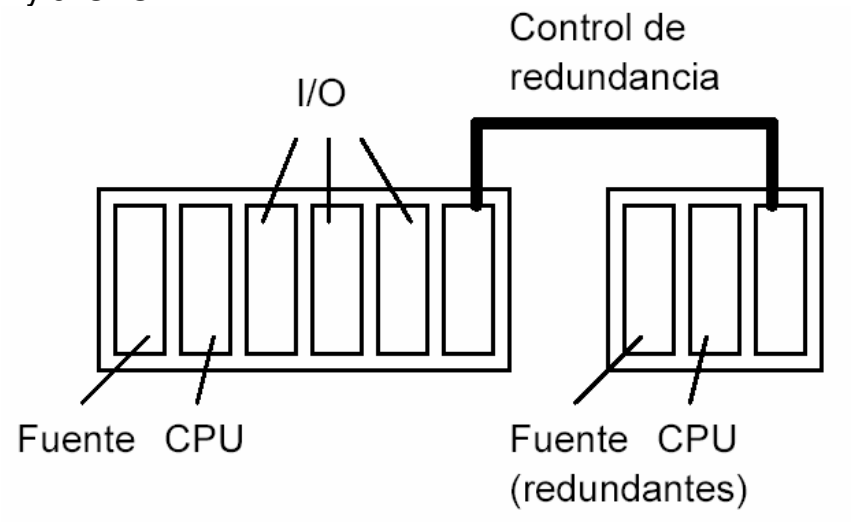
Módulos de comunicaciones

Permite la conexión del PLC a otros sistemas de información, tales como computadores y otros PLC. Existen por ejemplo redes tipo DataHiway para establecer una red de PLC conectados a un computador *Host*, utilizada comúnmente en sistemas de control distribuido.



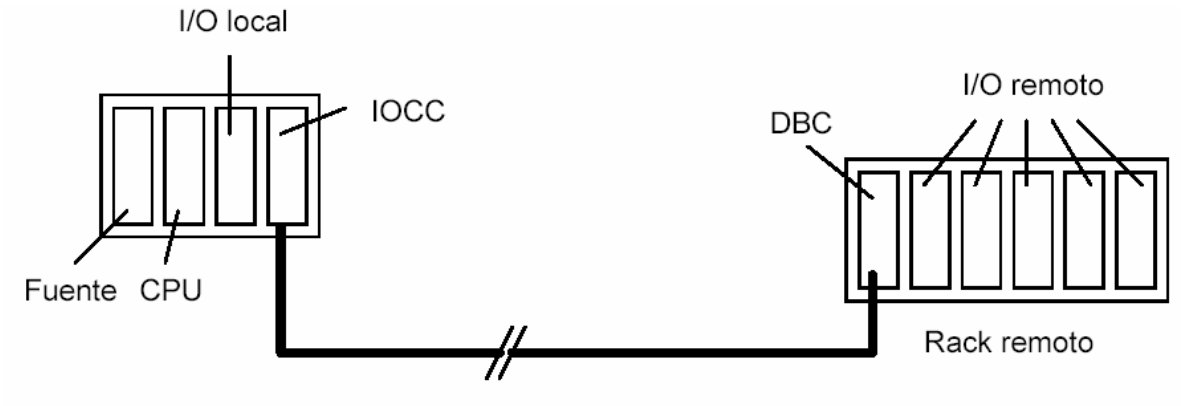
Módulos de control de redundancia

Son utilizados para asegurar la operación de un módulo redundante en caso de fallas. Generalmente se utiliza redundancia para el módulo de fuente de alimentación y el CPU.

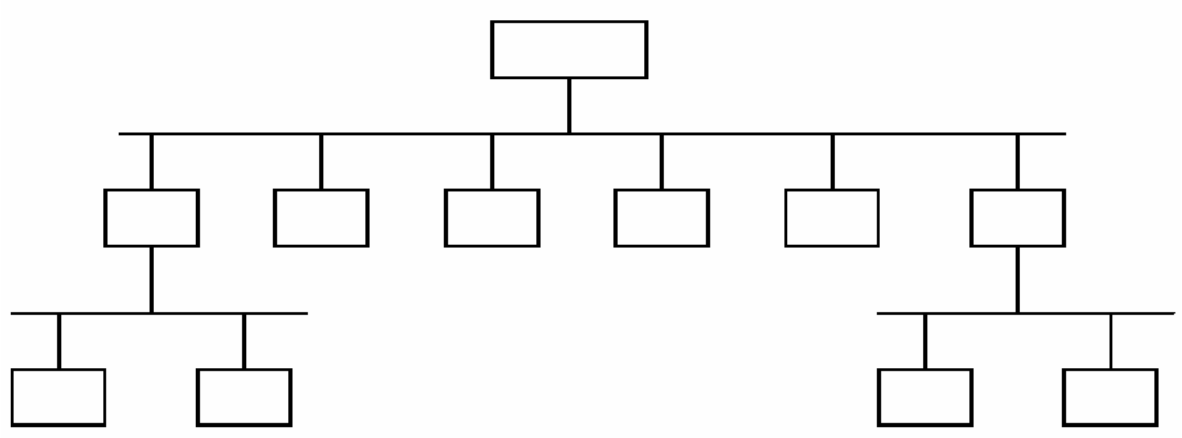


Módulos para conexión de *racks* remotos

En muchas aplicaciones los sensores y los actuadores están localizados a gran distancia del PLC. En estos casos se utilizan los *racks* remotos, los que son conectados por medio de un cable al *rack* central del PLC. Se consiguen distancias de 300 metros.

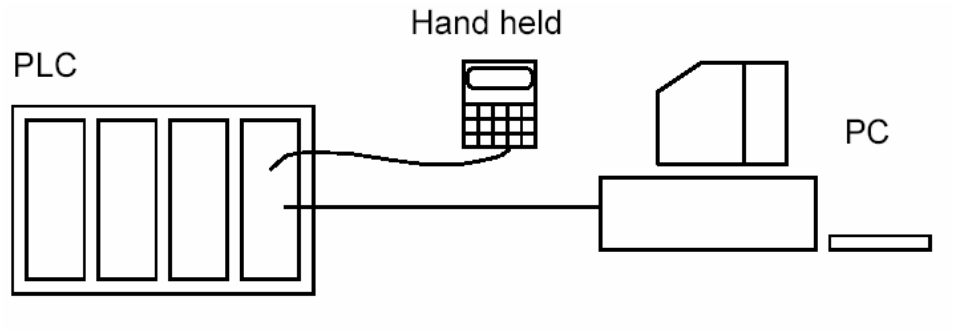


Para establecer esta comunicación se utiliza un módulo denominado canal controlador de entradas y salidas (IOCC) en el *rack* local y otro llamado controlador de base (DBC) en el *rack* remoto, al que se le pude conectar otro *rack* remoto, estableciéndose así una arquitectura distribuida con distintos niveles de jerarquía:



Módulos de interfaz hombre-máquina

Se utilizan para establecer la comunicación entre el PLC y el usuario. En la mayoría de los casos se emplea con este fin, un computador PC conectado serialmente, desde el cual se puede programar el PLC y ver los estados de los registros internos y los puntos de entrada/salida. En otros casos se usa un *Hand held monitor*, que es un dispositivo pequeño con teclas funcionales y pantalla de caracteres.

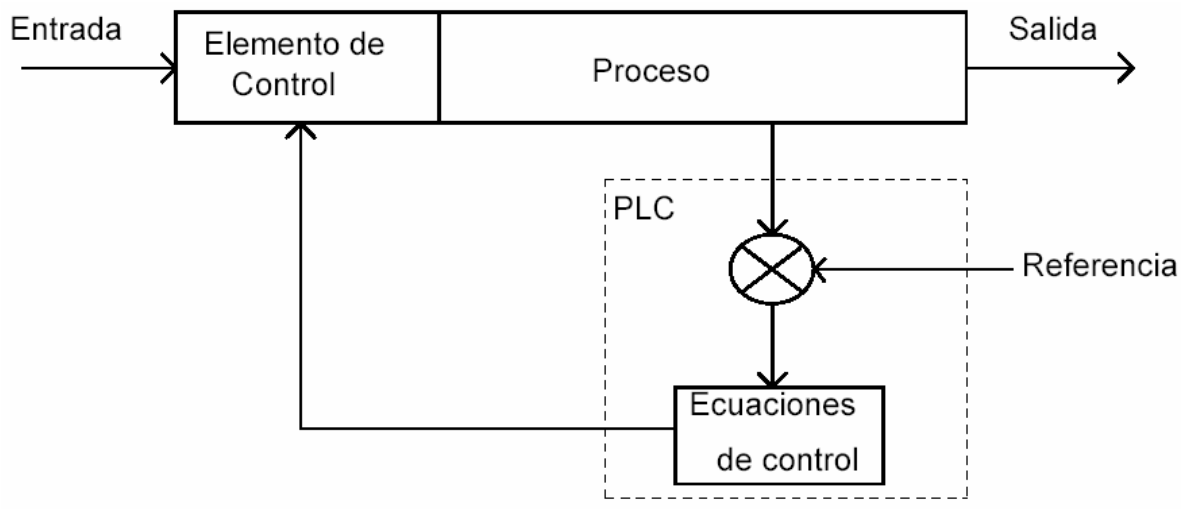


Módulos de almacenamiento de información

Por lo general se utilizan medios de almacenamiento magnéticos tales como cintas y discos, en los que se puede guardar información de los valores de los puntos de entrada/salida y registros internos.

Módulos controladores PID

Se utilizan en el control de procesos, en el que se pretende igualar una variable de salida de un proceso a una variable de referencia.

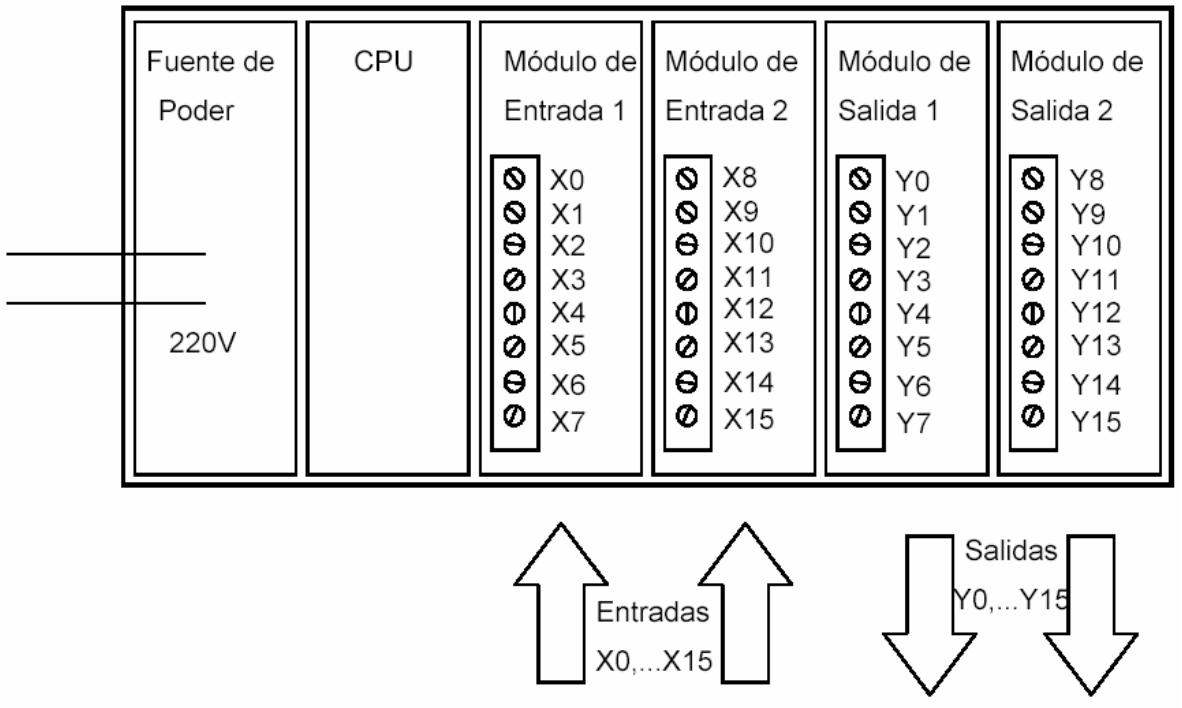


Puntos de entrada/salida

Los puntos del PLC son las entradas/salidas físicas que éste puede manejar. Cada punto tiene su representación interna en la memoria del PLC, en la que se utilizan números para identificarlos. Por lo general los módulos de entrada/salida vienen configurados en grupos de 8 puntos y pueden llegar hasta 1024, ampliables a más.

Los puntos de entrada son designados como X0, X1, X2, X3..., mientras que los puntos de salida se identifican como Y0, Y1, Y2, Y3...

A continuación se muestra una configuración básica de un PLC de 16 entradas y 16 salidas:

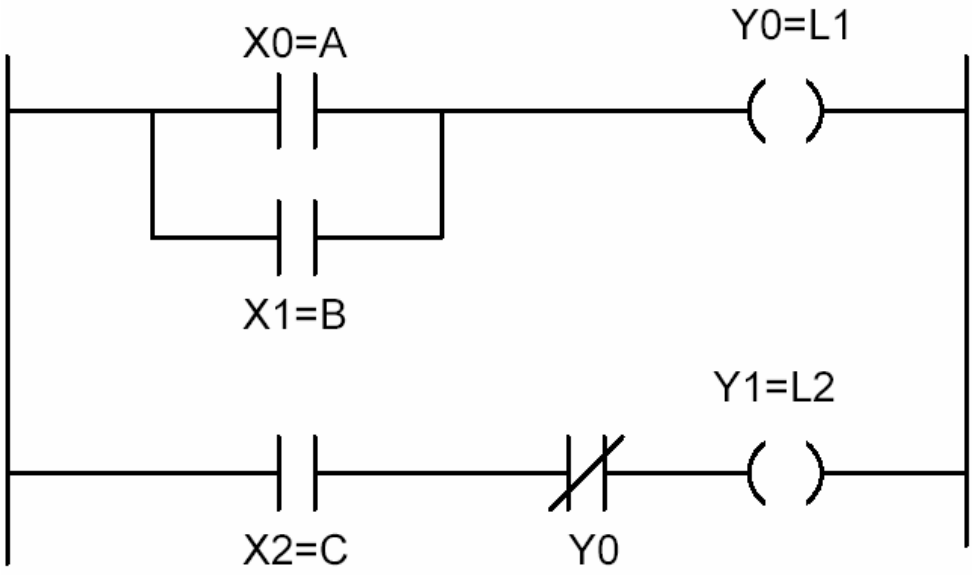


Al diseñar el programa se debe hacer referencia a las variables de entrada/salida que identifican los puntos del PLC.

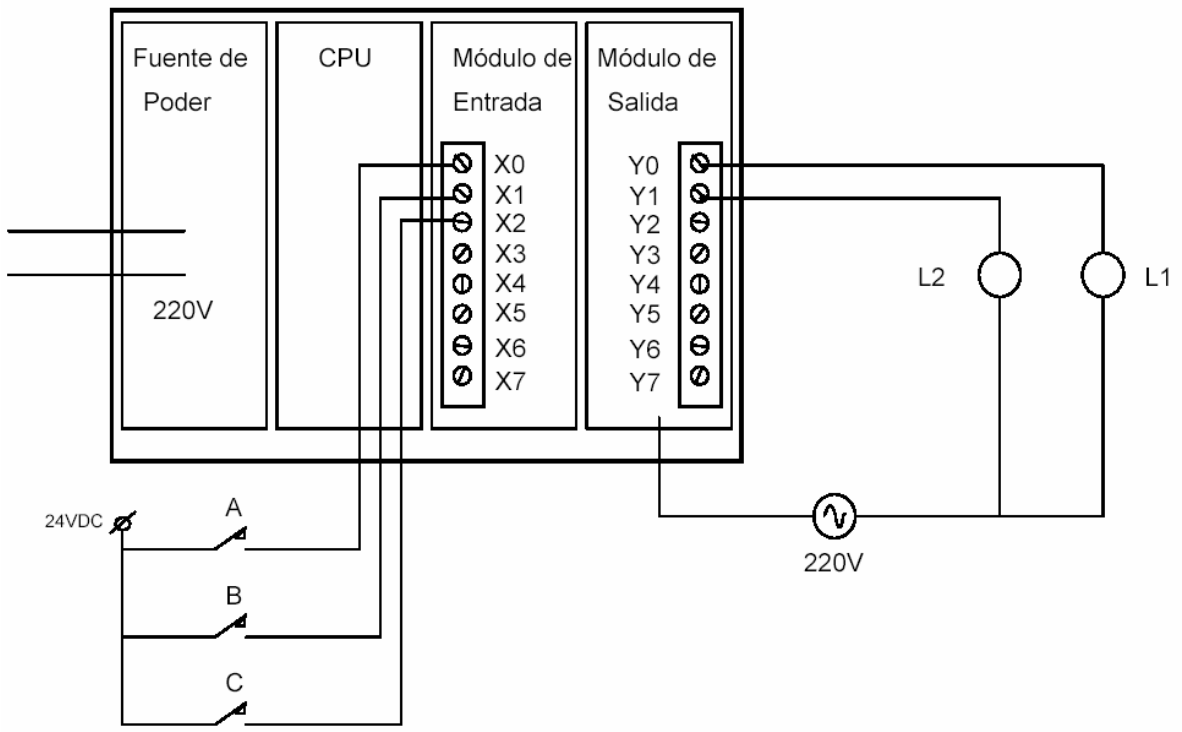
Ejemplo

. Se desea encender una lámpara L1 cuando se conecte el interruptor A o el interruptor B, y encender una lámpara L2 cuando L1 esté apagada y el interruptor C esté conectado.

Se distinguen las variables de entrada A, B y C, las que serán designadas como X0, X1 y X2; y las variables de salida L1 y L2, las que se identificarán como Y0 y Y1.



Las conexiones para este ejemplo se muestran a continuación:



La asignación de entradas y salidas se efectúa por medio del dispositivo de programación del PLC. Por lo general se utiliza un PC con interfaz gráfica que permita visualizar el diagrama escalera RLL (*Relay Ladder Logic*).

Registro imagen

Es un área de memoria del PLC reservada para mantener el estado de todas las entradas y salidas. Este registro se actualiza en forma permanente. Existen diversos registros:

. **Registro imagen discreto**

Corresponde a localizaciones de bits, donde se almacena el estado de todas las entradas/salidas digitales.

. **Registro imagen de relé control**

Son localizaciones de memoria de bits donde se guarda el estado de los Relés control.

. **Registro imagen de palabra**

Consiste en localizaciones de memoria, donde se registra el valor de cada palabra de entrada y salida.

En la programación de un PLC se utiliza también registros internos, que son de gran ayuda para almacenar datos intermedios. Estos registros son designados comúnmente como C0, C1, C2,...

CAPITULO II

LENGUAJES DE PROGRAMACION

2.1 Lenguajes de programación

Los lenguajes de programación ofrecen un conjunto de instrucciones con una determinada sintaxis para ejecutar una función.

Existen lenguajes de nivel bajo, intermedio y superior dependiendo del grado de comunicación que se tiene con la unidad de control de procesos (CPU) y el grado de complejidad de las instrucciones.

Otra clasificación de los lenguajes de programación son los lenguajes estructurados y los no estructurados, que se refieren a la forma de escribir y agrupar las instrucciones.

Un buen lenguaje de programación debe ser de fácil entendimiento, de tal forma que permita su modificación posterior si es que existen nuevos requerimientos.

Lenguajes de bajo nivel

Son los lenguajes que operan con instrucciones que controlan cada bit del CPU. Éstos son los lenguajes *Assembler* y de máquina. A manera de ejemplo, con estos lenguajes sólo se pueden sumar números de 8 ó 16 bits. Para realizar una suma de números de más bits es necesario descomponer el número en números primarios, sumarlo uno por uno guardando el arrastre de cada suma primaria para sumarlo con el siguiente número más significativo.

Ejemplo:

Suma 2+3 en Assembler de Z80

```
LD A, 03H        Carga 3 al acumulador A (A=3)
ADD A, 02H  Suma 2 al acumulador A (A=5)
```

Lenguajes de nivel intermedio

Estos lenguajes ofrecen un conjunto de instrucciones que pueden tanto comunicarse a nivel de bit con el microprocesador como ejecutar funciones de mayor grado de complejidad.

En los lenguajes de nivel intermedio se incorporan las funciones aritméticas, algunas funciones matemáticas (trigonométricas, raíz cuadrada, logaritmos, etc.) y funciones de manipulación de archivos en dispositivos de almacenamiento externo.

Ejemplos de lenguajes de nivel medio: C, FORTH.

Ejemplo:

Cálculo de 20! en C:

```
s=1;
for(i=2;i<=20;i++)
     s=s*i;
```

Lenguajes de nivel superior

Los lenguajes de nivel superior realizan con tan solo una instrucción una operación que con lenguajes de otro nivel se necesitaría fácilmente una docena de ellos.

Por ejemplo, con un lenguaje de nivel superior orientado al manejo de bases de datos, se puede con una sola instrucción ordenar alfabéticamente una lista de 10,000 nombres.

Ejemplos de lenguajes de nivel superior: PASCAL, FORTRAN, BASIC, dBASE, COBOL, SQL.

Ejemplo:

Ordenamiento de un directorio telefónico en dBASE

```
use teléfono
index on nombre to telenom
```

Lenguajes estructurados y no estructurados

La diferencia fundamental entre la programación estructurada y la no estructurada radica en que la primera no acepta el comando de bifurcación. De esta forma, el programa se ejecuta sólo por secciones. Para realizar una bifurcación, es necesario recurrir a instrucciones condicionales que ejecutarán una sección del programa sólo si se cumple una determinada condición.

Por otra parte, el lenguaje no estructurado permite la bifurcación desde y hacia cualquier línea del programa.

Ejemplos de lenguajes no estructurados: BASIC, FORTRAN, *Assembler*.

Ejemplos de lenguajes estructurados: C, PASCAL, dBASE.

Ejemplo:

Cálculo de 20! en BASIC

```
10 S=1
20 I=2
30 S=S*I
40 IF I<=20 THEN 30
```

NO ESTRUCTURADO

Cálculo de 20! en C

```
s=1;
for(i=2;i<=20;i++)
   s=s*i;
```

ESTRUCTURADO

Lenguajes de programación orientados a PLC

El lenguaje de programación de un PLC permite la creación del programa que controlará su CPU.

Mediante este lenguaje el programador podrá comunicarse con el PLC y así dotarlo de un programa que controle las actividades que debe realizar.

Dependiendo del lenguaje de programación, es posible la realización del programa con distintos grados de dificultad.

Junto con el lenguaje de programación, los fabricantes suministran un software de ambiente de trabajo donde el usuario puede escribir sus programas. Estos softwares son amistosos y corren sobre computadores tipo PC bajo plataformas DOS o Windows.

Los métodos de programación más utilizados para PLC son:

. Programación con diagrama escalera
. Programación con bloques funcionales
. Programación con lógica boolena

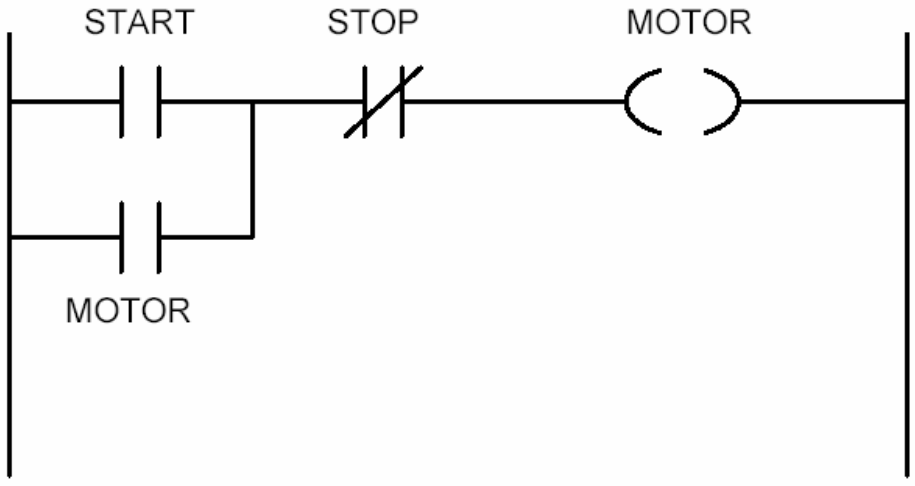
2.2 Programación con diagrama escalera

El diagrama escalera es uno de los más utilizados en la programación de PLC. Fue desarrollado a partir de los sistemas antiguos basados en relés. La continuidad de su utilización se debe principalmente a dos razones:

. Los técnicos encargados en darle mantenimiento a los PLC están familiarizados con este lenguaje.

. A pesar del desarrollo de los lenguajes de alto nivel, han sido pocos los lenguajes que han cumplido satisfactoriamente los requerimientos de control en tiempo real que incluyan la representación de los estados de los puntos de entrada y salida.

El nombre escalera proviene del uso de "rieles" y "peldaños" en el diagrama, como en este ejemplo de arranque de un motor.



En la gran mayoría de casos, las instrucciones para programar PLC pueden ser separadas en *básicas* y *expandidas*.

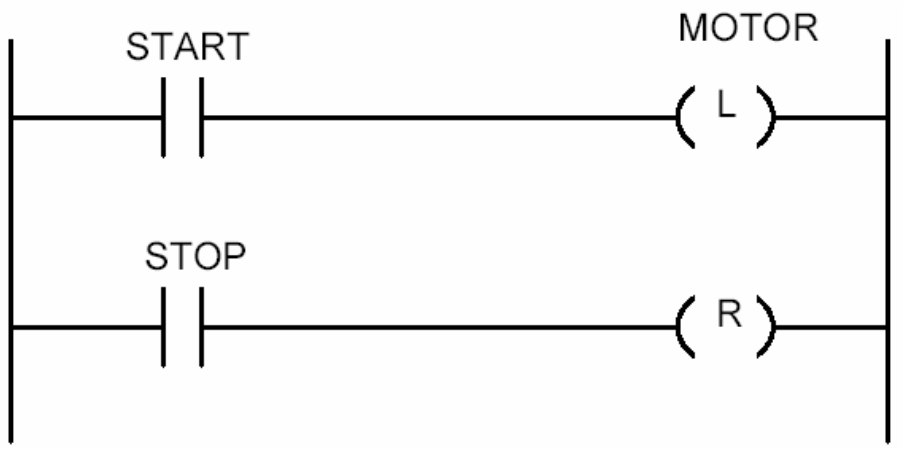
Instrucciones básicas:

Instrucciones básicas
RELAY
TIMER
COUNTER
LATCH
ONESHOT
I/O REG
REG I/O
BIN - BCD
BCD - BIN
ADD
SUB
COMPARE
MCR
SKIP

A continuación se explican algunas de ellas:

LATCH

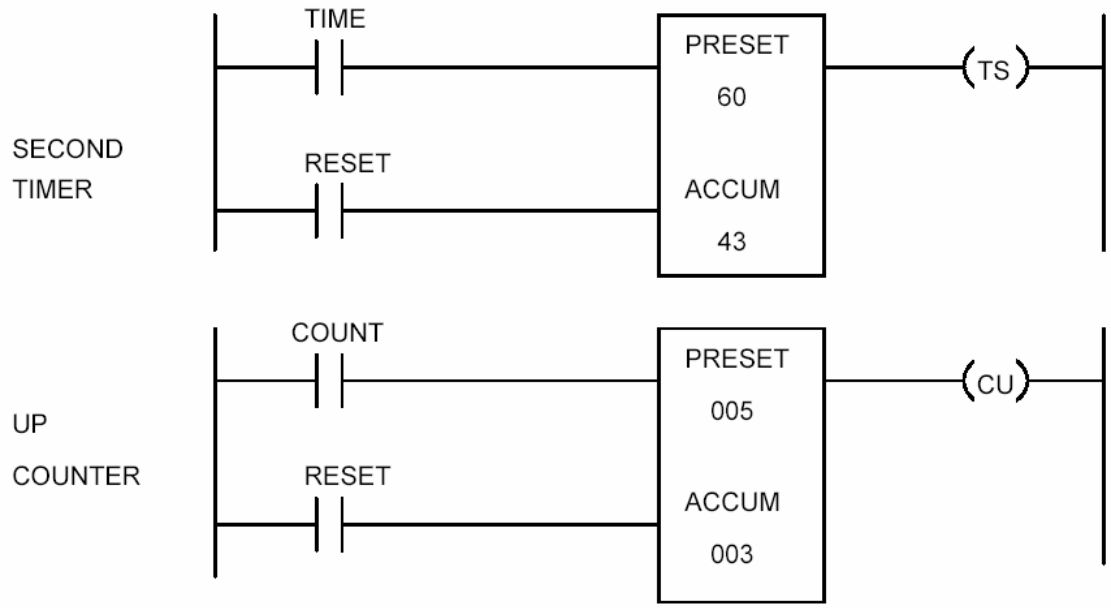
Mediante el empleo de *latchs* es posible desarrollar el mismo diagrama anterior de arranque de un motor, en el que un simple contacto energiza y mantiene energizado el motor.



El *latch* retiene su estado lógico cuando se abre el contacto, es decir, basta un solo contacto momentáneo para que el *latch* quede energizado. Esta función es de gran uso en sistemas de seguridad, en los que por precaución un circuito lógico no debe empezar en el estado *on* después de reactivarse una falla eléctrica, sino que debe conectarse en forma manual.

TIMER y COUNTER

Estas instrucciones remplazan los contadores electromecánicos en aplicaciones que requieren de contadores y temporizadores de eventos discretos.



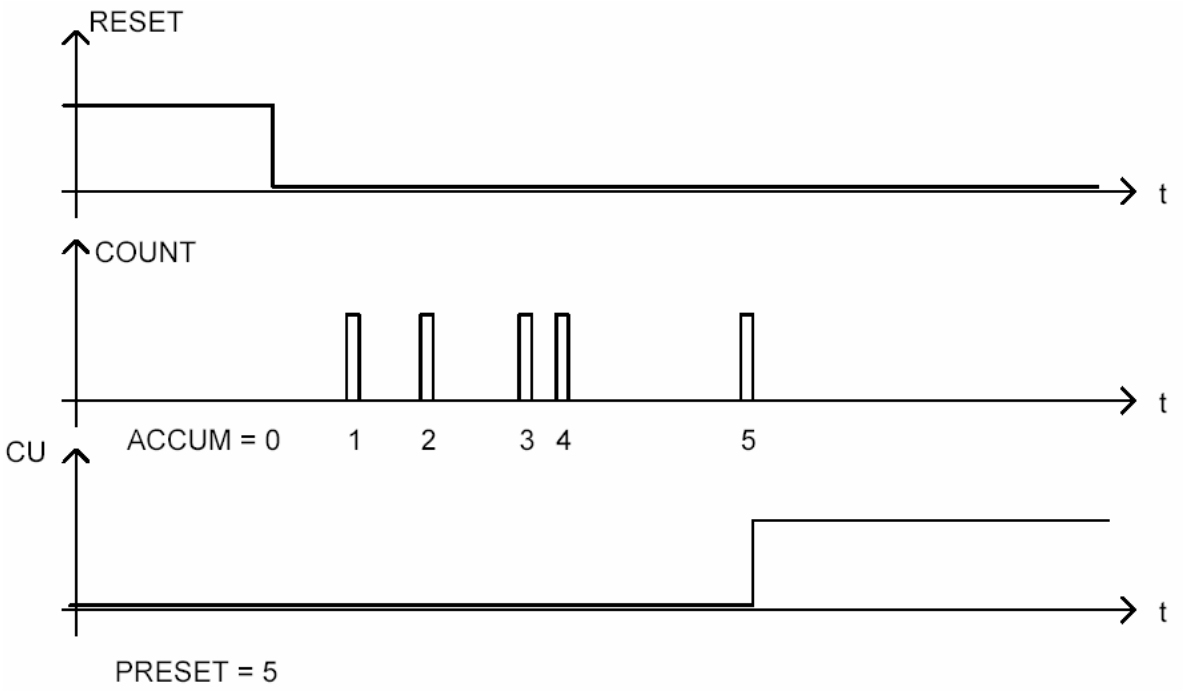
El **temporizador** opera de una manera similar al contador. Mientras el contacto TIME permanece cerrado, el valor del acumulador ACCUM se incrementa en uno por cada unidad de tiempo que pase. Esta unidad de tiempo es en algunos

PLC 0.1 seg, mientras que en otros puede se una unidad configurable.

Cuando el temporizador alcance el valor PRESET activará la salida TS. El contacto RESET hace que el valor del acumulador vuelva a 0.

El **contador** cuenta el número de contactos producidos en la entrada COUNT. Los contadores pueden contar hacia arriba: 0, 1, 2... ó hacia abajo 10, 9, 8, 7... El valor de la cuenta actual se almacena en el acumulador ACCUM. El valor del acumulador se hace 0 si el contacto RESET se cierra.

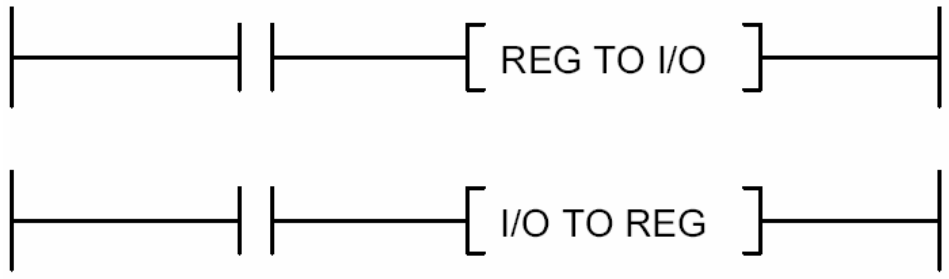
El contador cuenta hasta un valor de PRESET, y cuando lo alcanza activará la salida CU.



Ejemplo de un diagrama de tiempos de un contador

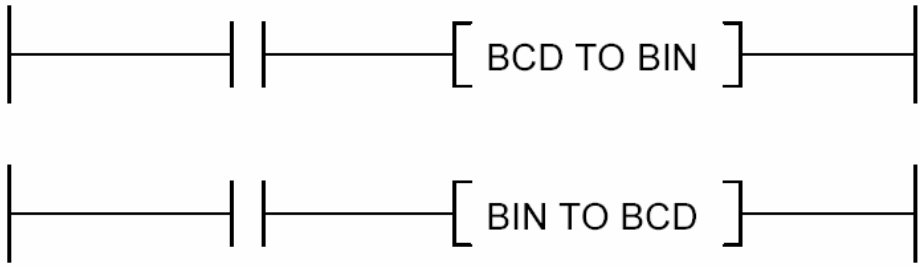
Instrucciones de entrada/salida

La instrucción I/O TO REG es utilizada para ingresar un punto de entrada a un registro del PLC, mientras que la instrucción REG TO I/O hace la operación contraria: pasa un registro a un punto de salida del PLC.



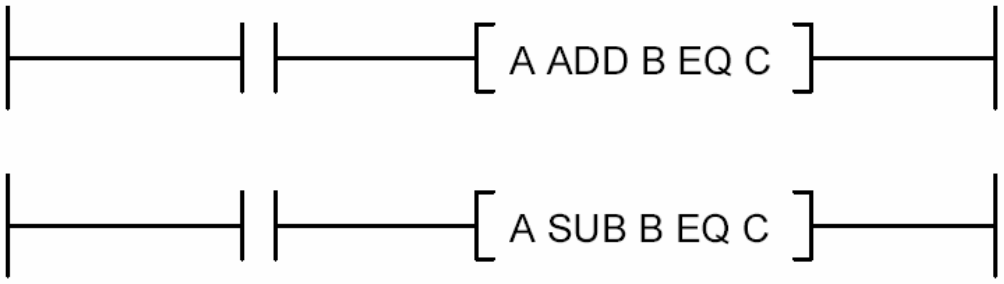
Instrucciones de conversión

Las instrucciones BCD TO BIN y BIN TO BCD son empleadas para convertir de código BCD a binario y binario a BCD respectivamente. Estas instrucciones se combinan con las de entrada y salida explicadas anteriormente.



Instrucciones aritméticas

Los PLC incluyen dentro de sus instrucciones, operaciones aritméticas sin signo. Los comandos utilizan los nemónicos ADD y SUB para la adición y substracción respectivamente. En la figura los registros A y B son sumados o substraídos, y el resultado se almacena en el registro C.



Instrucciones expandidas

Instrucciones expandidas	
MOVE	REM -FM - TOP
MOVE RIGHT 8	SORT
MOVE LEFT 8	AND
DP ADD	IOR
DP SUB	EOR
ADD X	INV
SUB X	MATRIX COMPARE
MPY	BIT SET
DVD	BIT CLEAR
GREATER THAN	SHIFT RT
TABLE - DEST	SHIFT LT
SRC - TABLE	DO SUB
MOVE TABLE	RETURN
ADD-TO-TOP	DO I/O
REM-FM-BOT	

Las instrucciones básicas contienen normalmente relés, *latches*, temporizadores, contadores, manipulación de registros y puntos de entrada/salida, conversiones y funciones matemáticas.

Debido a que los PLC contienen un microprocesador, es posible la incorporación de funciones más sofisticadas que las utilizadas en la lógica de relés.

Las instrucciones expandidas incluyen funciones tales como movimiento de datos, movimiento de tablas, administradores de listas, aritmética con signo y doble precisión, cálculos matriciales y ejecución de subrutinas.

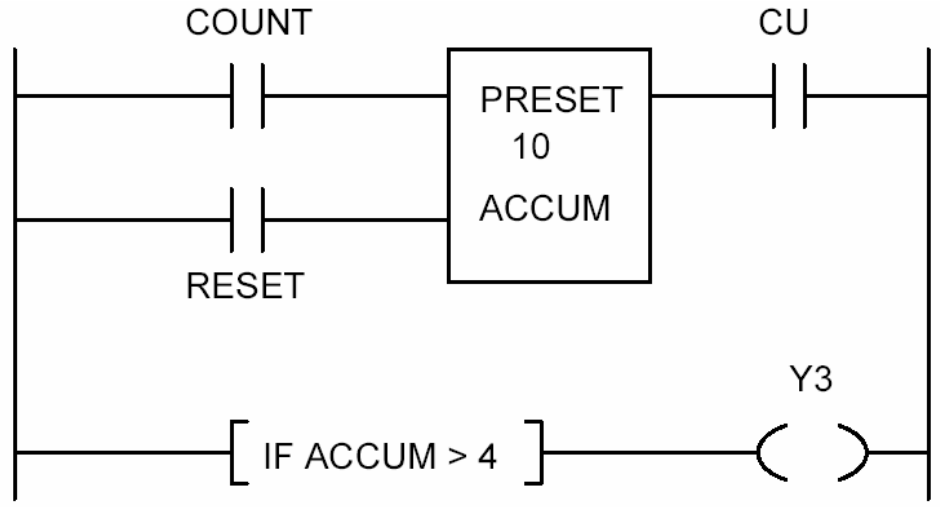
Instrucciones de movimiento de datos

Son utilizadas para copiar un registro o una porción de memoria a alguna localización de la memoria.

Instrucciones matemáticas avanzadas

Se incluyen operaciones aritméticas (suma, resta, multiplicación y división) de doble precisión con signo. Algunos PLC tienen otras funciones como raíz cuadrada y funciones trigonométricas.

La instrucción matemática GRATER THAN energiza la línea si la condición es verdadera. En el siguiente ejemplo, el acumulador ACCUM contiene la cuenta que lleva la caja *counter*. Si la cuenta es mayor que 4 se activa la salida Y3.

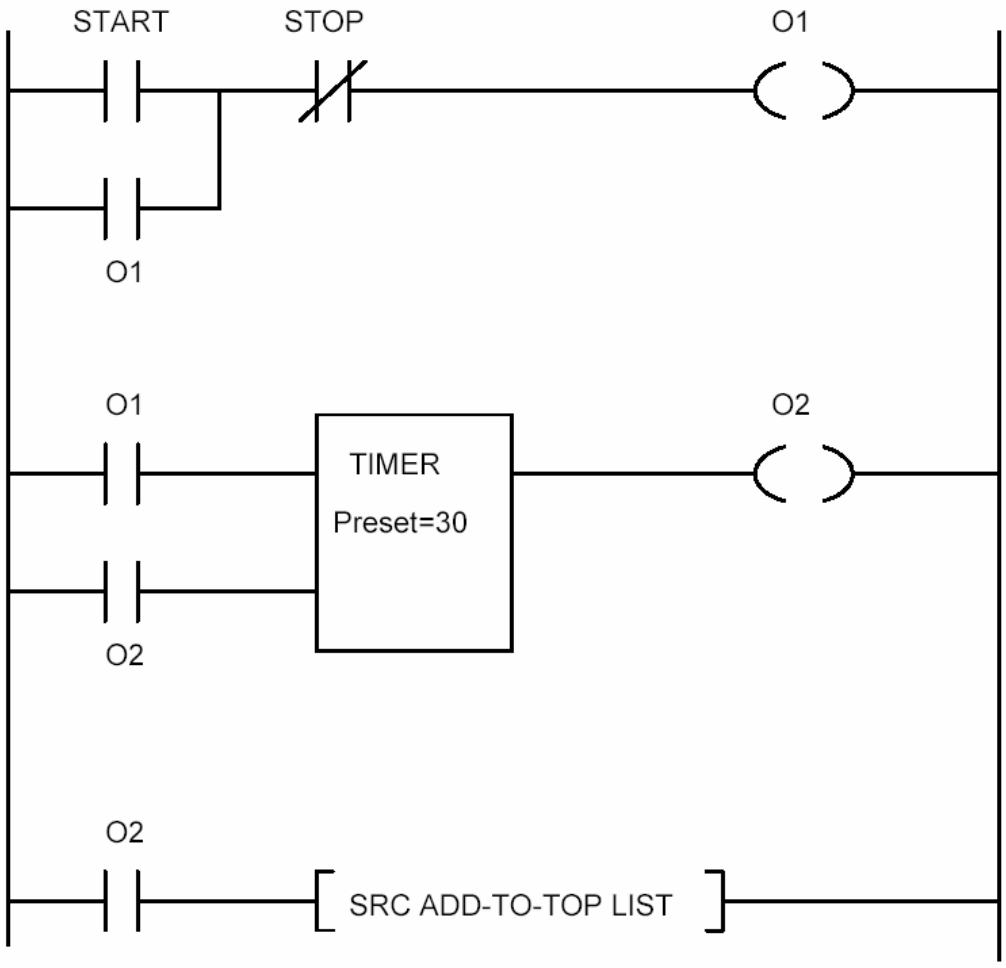


Instrucciones de tabla y de lista

Estas instrucciones permiten la creación y edición de tablas numéricas. De esta forma se pueden almacenar datos en una tabla, para que posteriormente puedan ser leídos y procesados. La información es almacenada en formato binario, pero puede ser convertida a decimal o a ASCII.

Las instrucciones de lista accesan los datos de forma secuencial con un puntero (FIFO o LIFO), mientras que las instrucciones de tabla permiten un acceso aleatorio.

Ejemplo:



En este ejemplo, al presionar el botón START se energiza la salida O1, que habilita el temporizador TIMER. Al pasar 30 segundos se activa la salida O2. Esta salida realimenta el TIMER para que reinicie su cuenta y activa la instrucción SRC ADD-TO-TOP LIST, que agregará al inicio de la lista el dato indicado por SRC, que podría ser un registro analógico de una temperatura de un proceso. Así se almacena cada 30 segundos la temperatura en una lista.

Instrucciones matriciales

Son utilizadas para realizar la operación OR (inclusivo y exclusivo) y AND de dos matrices binarias, el resultado se almacenará en una tercera matriz. Asimismo se pueden comparar dos matrices y tomar alguna decisión si son iguales.

Instrucciones de subrutina

Una subrutina es una porción del programa que puede ser ejecutada varias veces con distintos parámetros, desde distintas partes del programa.

2.3 Programación con bloques funcionales

Una de las formas más recientes de programar un PLC es a través de una carta gráfica de bloques funcionales. Este tipo de programación ha sido diseñado para describir, programar y documentar la secuencia del proceso de control.

En Europa, se ha comenzado a utilizar el lenguaje de programación llamado GRAFCET (creado en FRANCIA), orientado a la programación de PLC mediante bloques funcionales.

En la lógica combinacional, la programación con bloques funcionales es muy superior a otras formas de programación, mientras que los diagramas escalera y booleanos son mejores en lógica combinacional.

Debido a que hoy en día el control de procesos se programa principalmente con lógica secuencial, la programación con bloques funcionales será pronto el estándar para programar PLC.

Este lenguaje incluye un conjunto de símbolos y convenciones tales como pasos, transiciones, conectividades (también llamados enlaces) y condiciones.

Pasos

Son símbolos secuenciales individuales, representados por cuadrados numerados, los que pueden contener nemónicos que describen la función del paso.

Transiciones

Las transiciones describen movimiento de un paso a otro. Su representación es una línea horizontal corta.

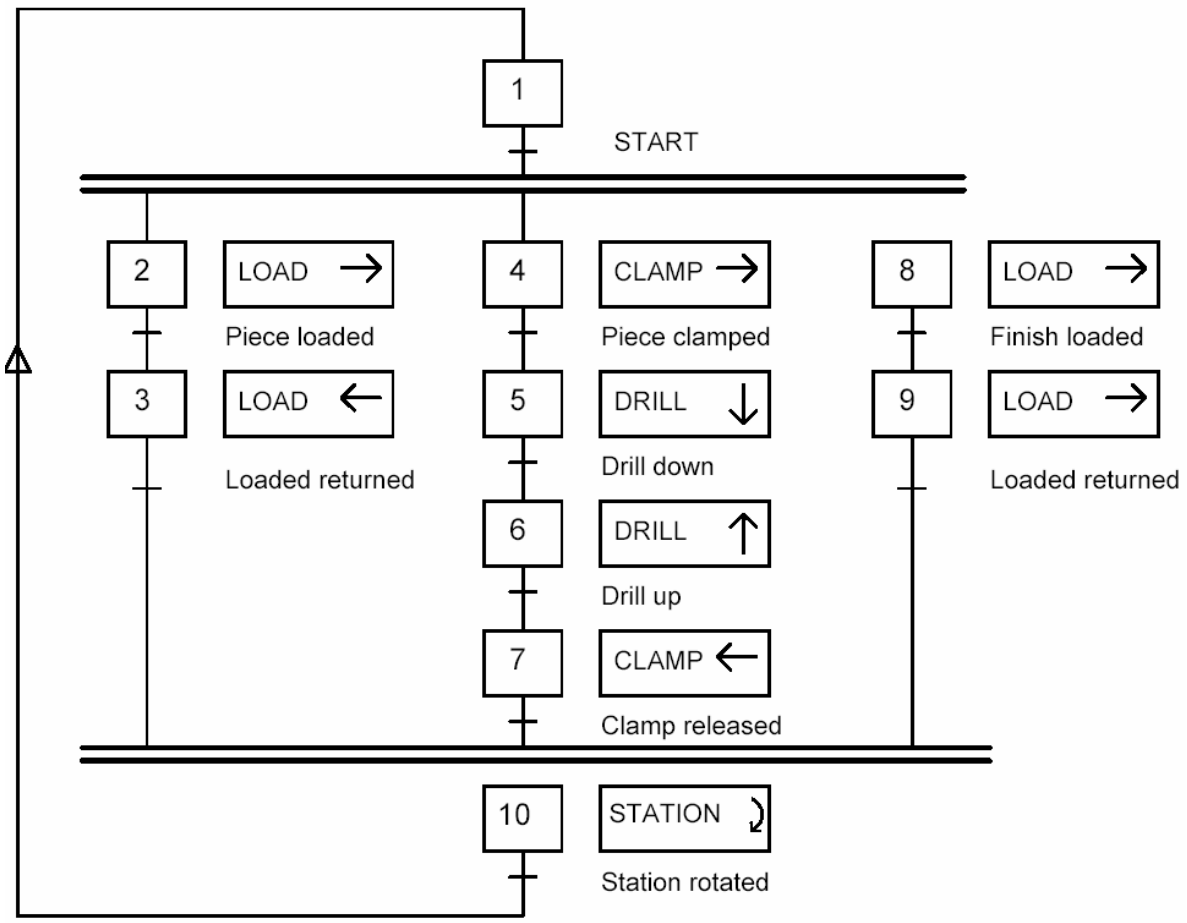
Enlaces

Los enlaces muestran el flujo del control, el que va desde arriba hacia abajo, salvo que se indique lo contrario.

Condiciones

Las condiciones están asociadas a las transiciones y deben ser escritas a la derecha.

Ejemplo



El ejemplo muestra lo fácil que puede ser programar y describir un control de un proceso de perforación por medio de un taladro.

La operación comienza con una pieza que es cargada, luego sujetada, perforada y removida, seguida de una estación que rota la pieza antes de que el proceso comience nuevamente.

Cada cuadrado contiene comandos de control que describen la entrada/salida discreta y/o las operaciones aritméticas que son programadas.

Este tipo de programación representa un gran vínculo entre el programador y el diseñador del proceso. Asimismo es una gran herramienta para:

- . describir esquemáticamente el proceso,
- . localizar fallas rápidamente,
- . integrar fácilmente el sistema de control y el usuario

2.4 Programación con lógica booleana

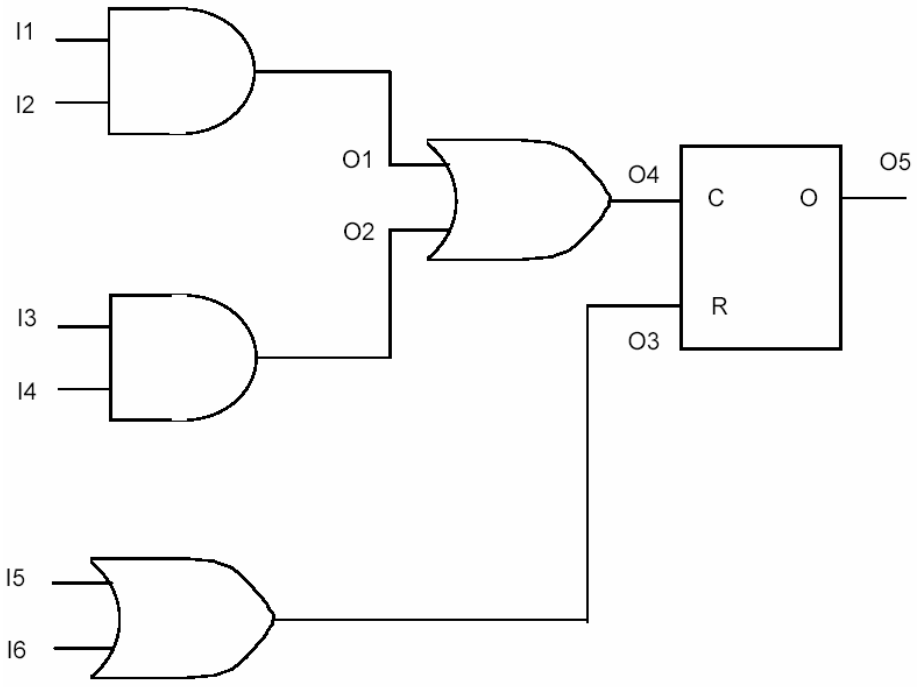
La programación con lógica booleana incluye las funciones AND, OR y NOT para la lógica secuencial y las funciones TIMER, COUNTER y LATCH para la lógica combinacional.

Estas funciones son muy similares a las utilizadas en la programación con diagrama escalera. Específicamente:

AND: Contactos en serie.
OR: Contactos en paralelo.
NOT: Contacto normalmente cerrado.

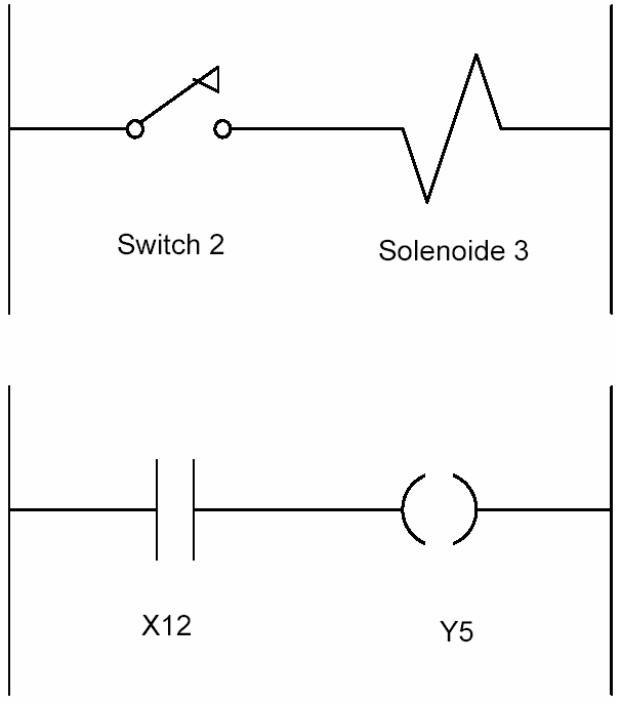
Las cajas tipo TIMER, COUNTER y LATCH son empleados de similar manera.

Algunas industrias europeas han optado por la programación booleana como estándar para el diseño del control lógico.



2.5 PROGRAMACIÓN DE UN PLC

Una forma usual de programar el PLC es utilizando el esquema *Relay Ladder Logic* (RLL), que es muy similar en forma e interpretación a los diagramas de escalera de relés.



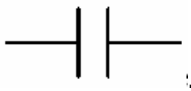
De acuerdo al diagrama escala, cuando el interruptor 2 se cierra, el solenoide 3 se energiza.

Para el programa RLL, el interruptor 2 está conectado a un terminal de módulo de entrada identificado como X12.

Cuando el PLC ejecuta el programa, envía una señal al terminal de módulo de salida identificado como Y5, el cual se encuentra conectado el solenoide 3.

En la figura del ejemplo, se utilizaron las instrucciones RLL de contacto y solenoide.

2.6 Contactos

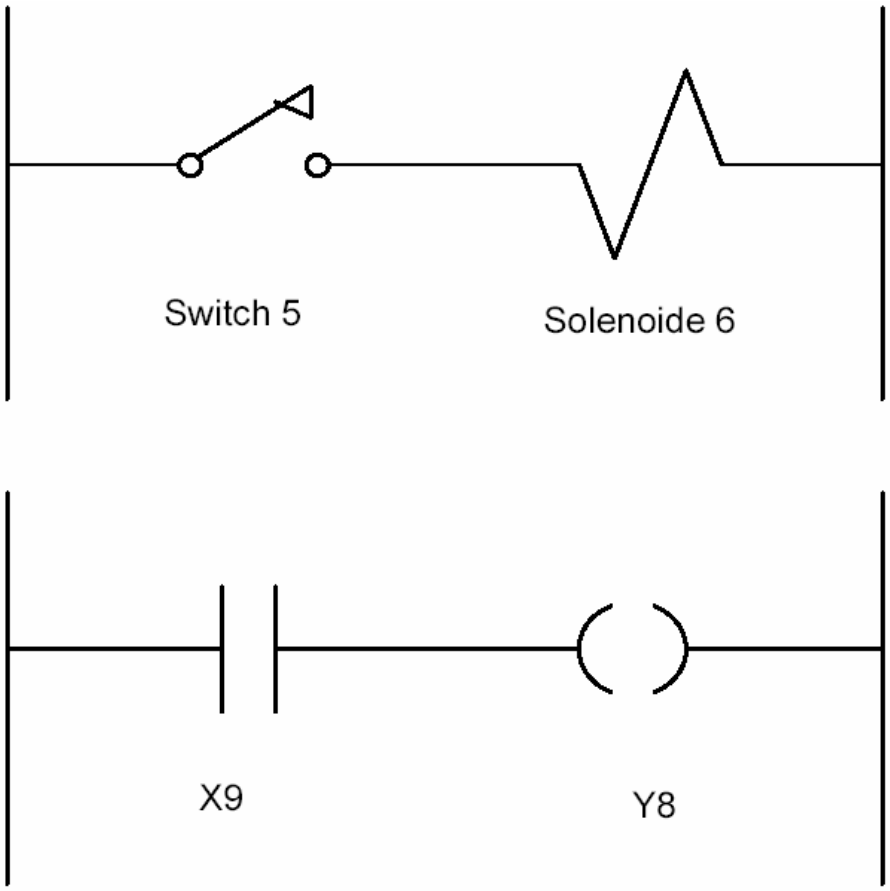
En el diagrama escala anterior, el solenoide está energizado cuando el interruptor se encuentra cerrado. En el programa RLL el *switch* se representa con el símbolo  , que se denomina CONTACTO NORMALMENTE ABIERTO **(NO)**. Esto significa que establece el flujo de energía cuando el interruptores cierra. Si el *switch* se abre, no fluye corriente a través del contacto.

El módulo de entrada al cual se ha conectado el *switch*, detecta si éste se encuentra abierto o cerrado.

Un CONTACTO NORMALMENTE CERRADO **(NC)** puede representar la entrada de cualquier sensor, *switch* o el estado de la salida de otra etapa del programa.

Un contacto NC se representa con el símbolo  , y conceptualmente invierta el estado de la entrada.

Por ejemplo en la figura, si el switch 5 está abierto, el solenoide 6 se encuentra energizado. En el programa RLL, el switch 5 está conectado al módulo de entrada X9 y la salida Y8 entrega el poder al solenoide 6.

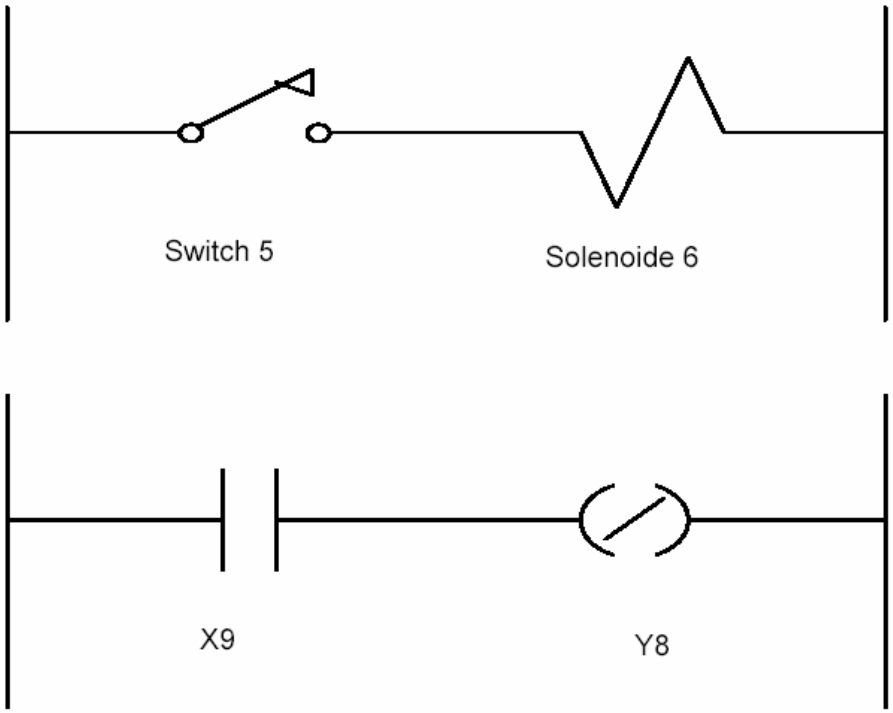


El PLC reconoce solamente si un contacto está abierto o cerrado, pero no puede determinar su concepción NA o NC.
Por lo tanto, si la función debe ocurrir cuando el *switch* está abierto, la entrada de contacto se debe programar como normalmente cerrado NC.

2.7 Bobinas (solenoide)

En un Programa RLL, el dispositivo de salida es el solenoide cuyo símbolo es. Este símbolo es usado tanto para un dispositivo físico de salida externa, como para una salida interna que se emplea posteriormente en el programa.

La salida invertida se indica como. En este caso, al recibir la señal de salida, se desenergiza.

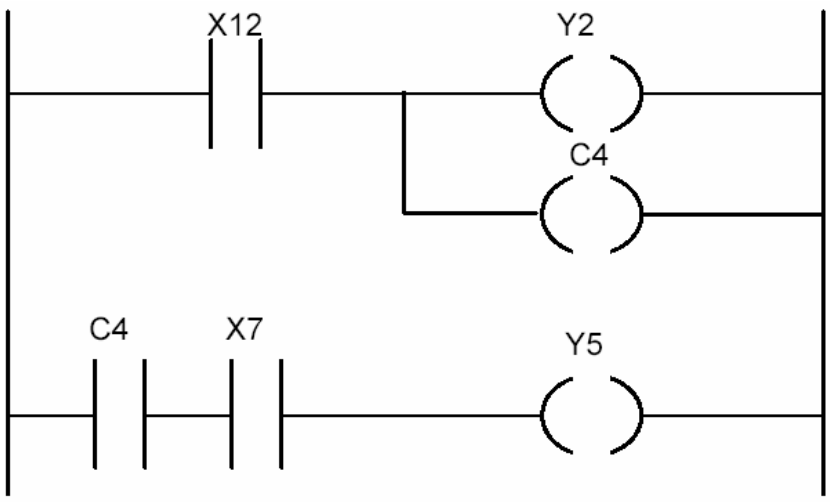


En la figura, si el *switch* se cierra (entrada X9), entonces se desenergiza el solenoide 6 (salida Y8).

2.8 Relés de control

Estos elementos no existen físicamente como dispositivos de entrada o salida. Sin embargo, se ubican en la memoria del PLC y sirven como herramientas de programación para simular las entradas y salidas en el programa.

En un Programa RLL se representan mediante los mismos símbolos que las bobinas y los contactos.

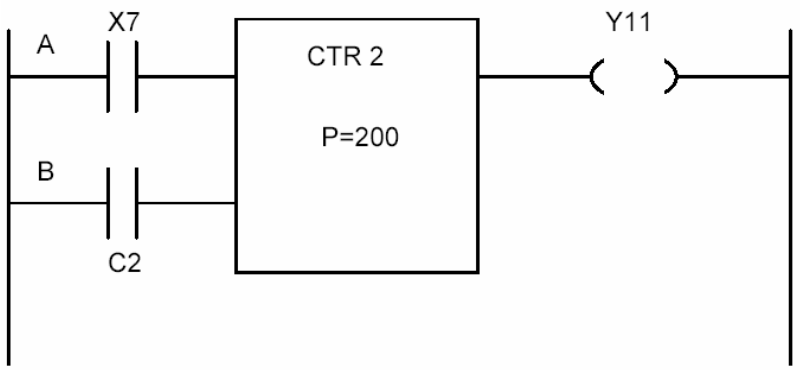


En el ejemplo, el relé de control C4 se energiza cuando X12 cambia a *ON*. El cambio de estado de C4 se registra en memoria. De igual forma, un relé de control se puede emplear como entrada en el Programa RLL; tal es el caso de C4 en la segunda línea. Si C4 y X7 se energizan, también lo hará Y5.

2.9 Cajas lógicas *RELAY LADDER*

Las cajas de instrucciones son funciones preprogramadas que amplían las capacidades de un programa más allá del conjunto de instrucciones RLL estándar.

Estas funciones permiten un empleo eficiente de la memoria del PLC y ahorran tiempo de programación.



El contador de la figura corresponde a un ejemplo de caja de función.

El contador CTR 2 se habilita por medio de la línea de entrada inferior B. Las transiciones *Off-On* de la línea de entrada superior A se cuentan como pulsos. Una vez que la cuenta alcanza el valor prefijado, P=200, la bobina de salida Y11 es energizada.

Ejercicio 1

Diseñar un diagrama RLD para un motor de 3 velocidades (V1, V2 y V3). El sistema cuenta con tres interruptores (S1, S2 y S3) que controlarán respectivamente cada una de las velocidades. Si están conectados 2 o más interruptores simultáneamente deberá activarse sólo la velocidad de menor rango. Adicionalmente el sistema debe contemplar un interruptor de apagado S0.

Ejercicio 2

Diseñar un diagrama RLD para un sistema que controle el encendido y apagado de un motor (M). El sistema debe contar con un pulsador de encendido (E) y uno de apagado (A). El motor se encenderá, y permanecerá encendido, cuando se presione el pulsador por un instante. De igual forma, el motor se apagará, y permanecerá apagado, cuando se presione el pulsador (A) por un instante.

Ejercicio 3

Diseñar un diagrama RLD para un sistema de alarma de una oficina. La oficina cuenta con una puerta (P) y una ventana (V). La alarma debe activar una sirena (S) cuando se abra la puerta o la ventana, y deberá permanecer activa si es que la puerta o la ventana se cierran. El sistema de alarma debe tener una luz indicadora Lp que señalará que fue la puerta la que activó la alarma y una luz Lv para la ventana. Adicionalmente deberá contar con un botón de encendido y apagado de alarma.

2.10 Diseño y documentación de programas

Definición de la aplicación

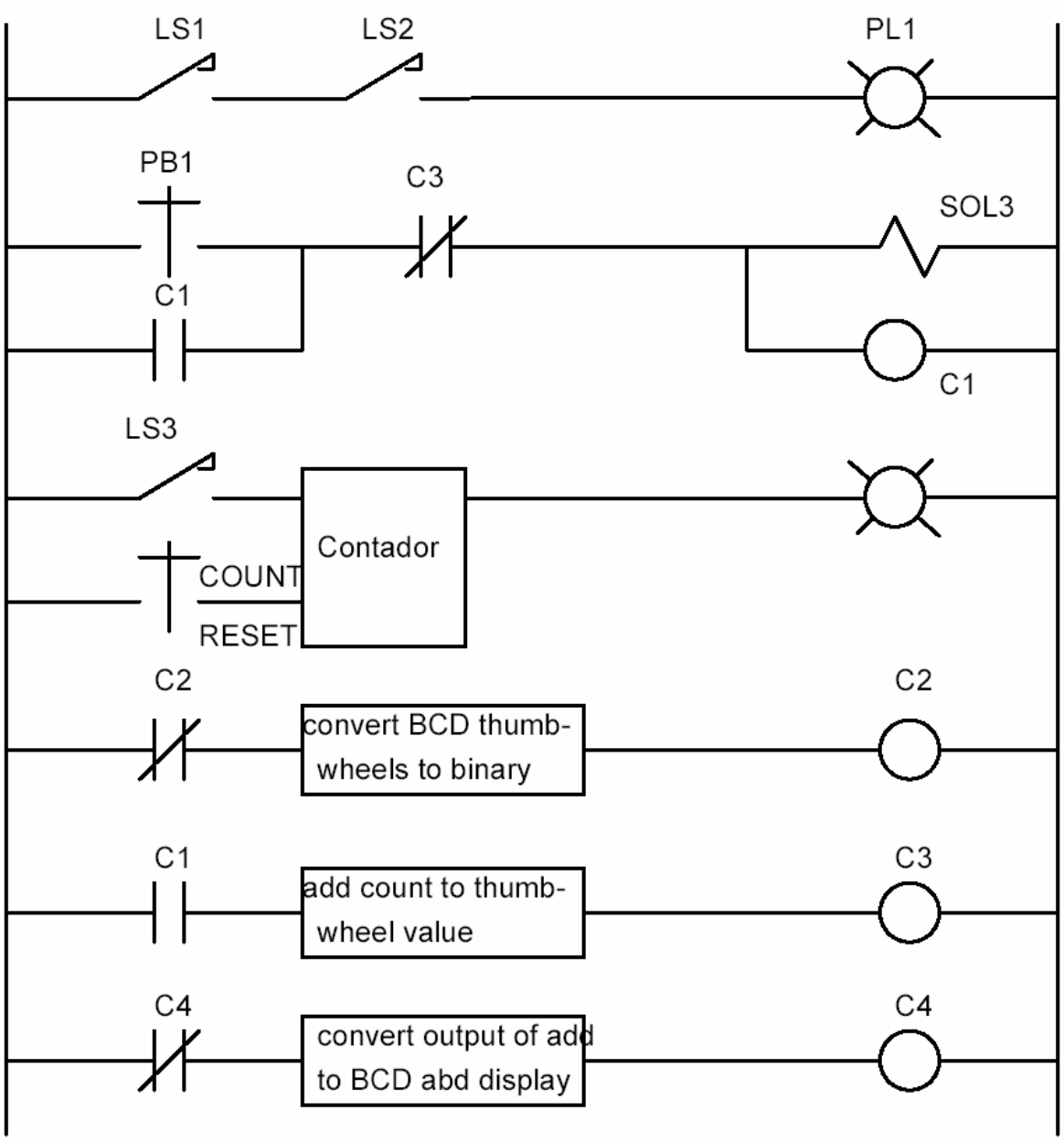
Lo primero que debe realizarse, es:

. Determinar las tareas que se requiere del equipo.
. Estimar los requisitos de tiempo.
. Estimar el orden en que se deben efectuar las tareas.

Previo a escribir el Programa RLL, se debe estar familiarizado con el equipo y su operación, y así determinar cómo automatizarlo.

Construcción del diagrama de relés RLD

Una vez definida la aplicación, se construye el diagrama RLD (Relay Ladder Diagram), donde se reúnen los requerimientos de operación. Este diagrama es una representación estándar de relés, interruptores, solenoides, motores, retardos de tiempo, lámparas, etc. que realizan la operación que se desea controlar.



Ejemplo de un diagrama RLD.

Asignación de identificadores

Se debe asignar la identificación a cada punto físico representado en el diagrama RLD, mediante una letra (X ó Y) y un número.

Una vez que el identificador se ha asignado al terminal, el dispositivo físico ahí conectado retiene su denominación hasta que se cambie a otro terminal.

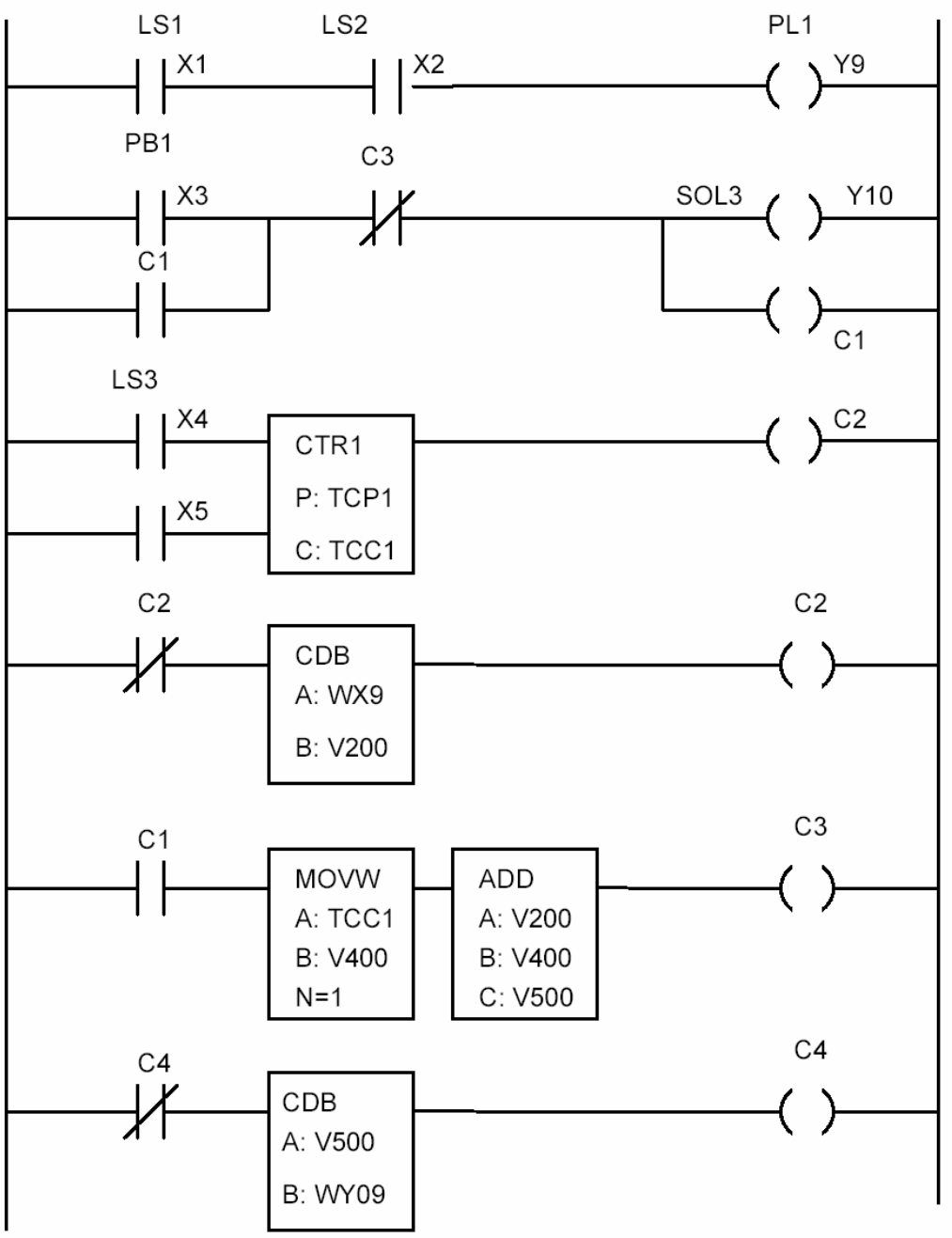
Base Assembly No. 01			
Identifier		Terminal Name	Slot Number
I/O Point type	Terminal designation		
X	1	LS!: part detect	1
X	2	LS2: part in place	1
X	3	PB1: cycle start	1
X	4	LS3: part count	1
	5		
	*		
Y	6	PL1: put part in place	2
Y	7	SOL3: clamp solenoid	2
Y	8	PL1: end of cycle	2
	*		
	*		

Construcción del diagrama RLL

Una vez asignadas las I/O y las localizaciones de memoria, se construye el diagrama RLL equivalente al diagrama RLD. Se cambian las designaciones mecánicas por los identifica-dores asignados.

El programa se almacena en memoria RAM. Cada instrucción se guarda, por lo general, como una palabra de 16 bits. El número de palabras por instrucción depende de:

. Tipo de instrucción y el número de referencia asignado.
. Localizaciones de variables V de memoria que se accesan dentro de la instrucción.
. Número de referencia de los relés de control que se accesan en la instrucción.
. Número de referencia de inicialización de temporizadores o valores que son accesados en la instrucción.



Ejemplo de un diagrama RLL.

Un bloque de memoria, memoria V, se asigna para las operaciones de cálculo interno.

Es recomendable llevar un registro con las posiciones de memoria V a medida que se diseña el programa. En la siguiente figura se muestra una manera conveniente de registrar las posiciones de memoria empleadas durante el diseño del programa.

		Comments
V	200	Binary value of thumbwheel input
V	201	
V	209	Value of current count in CTR1
V	213	Sum of thumbwheel input and CTR1 current
V	216	
V	217	
V	218	
V	219	

Diseño de diagnósticos en el programa.

El PLC posee la capacidad de entregar información sobre el estado del software y del hardware. Esta información se guarda en formato de palabra y puede accesarse desde el dispositivo de programación.

De igual forma, las palabras de estado se pueden utilizar en el Programa RLL para facilitar la detección temprana de errores y dificultades en el hardware.

A manera de ejemplo se ilustra a continuación cómo un PLC Texas Instruments de las serie 500 informa sobre su estado de operación.

El PLC TI-500 posee un conjunto de palabras de estado de 16 bits. Cada palabra informa el estado de una operación específica. En muchos de los casos es necesario estudiar cada bit de la palabra. Algunas palabras de uso común son:

Palabra de estado 1 (STW01)

Informa sobre el estado de la batería del PLC, problemas de muestreo, puerta de comunicaciones, estado de los módulos I/O, y de módulos de funciones especiales. Bit = 0 indica que no hay problema, Bit = 1 señala que hay problema.

Bit	Problema
15	Batería baja.
14	Tiempo de scan muy corto.
13	Falla en puerta de comunicaciones.
12	Falla en I/O.

Palabra de estado 2 (STW02)

Informa sobre el estado de hasta 16 *racks*. LSB corresponde al *rack* 0 y MSB al *rack* 15. El bit respectivo toma el valor 1, cuando el *rack* ha fallado o no está, y 0 cuando no hay problema.

Palabra de estado 6 (STW06)

Informa el estado de la programación de la EPROM/EEPROM del PLC.

Palabras de estado 7 a 9 (STW07 - STW09)

STW07 entrega la dirección absoluta de memoria donde se detecta el primer error al tratar de programar la EPROM (EEPROM).
STW08 muestra el valor calculado de *checksum* para el programa RLL almacenado en la memoria EPROM.
Este número se emplea para verificar que las copias de un programa sean iguales.
STW09 muestra el valor calculado de *checksum* para la EPROM completa: Programa RLL e información de memoria de configuración de I/O.

Palabra de estado 10 (STW10)

STW10 muestra, en código binario, el tiempo de muestreo del PLC.

Palabras de estado 11 a 18 (STW11 - STW18)

Informan el estado de los módulos I/O instalados en los *racks*. Cada bit corresponde a un módulo del *rack*. Bit = 0 indica que no hay módulo en el slot o está funcionando bien. Bit = 1 señala que el módulo del slot está en

mal estado. Si existe un módulo que no corresponde a su configuración, éste se informa como si estuviese fallado.

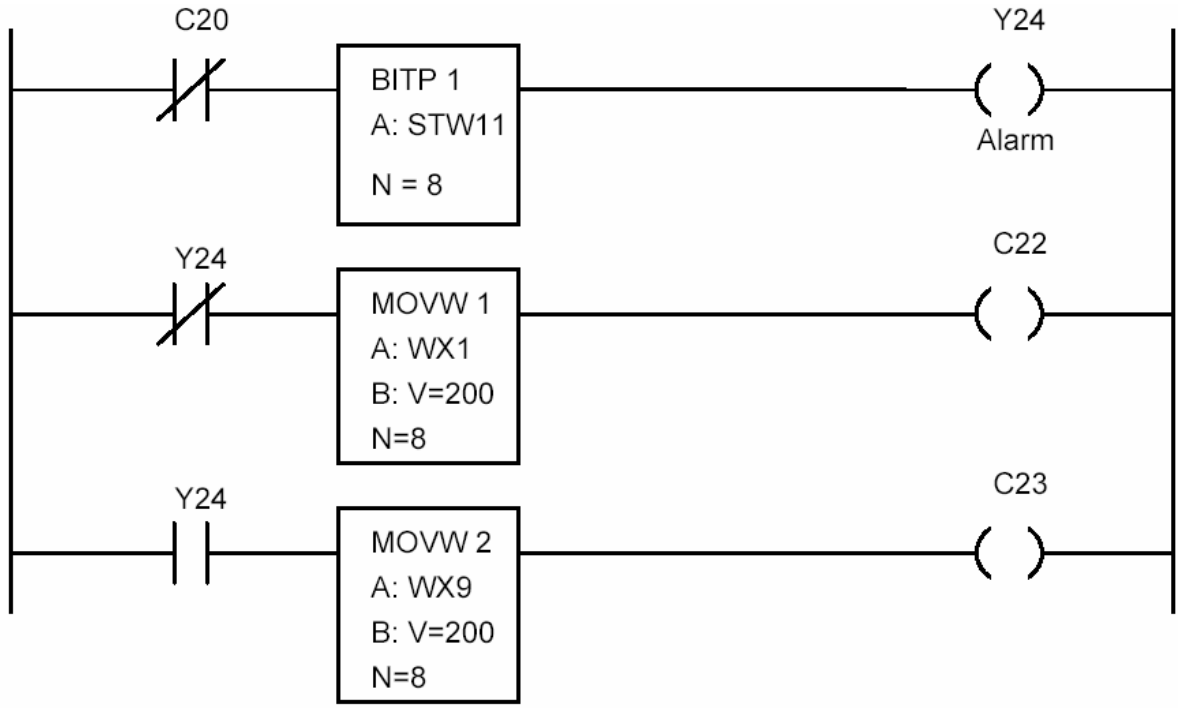
En una configuración distribuida, si algún *rack* pierde la comunicación con el PLC, en STW02 aparecerá 1 para el correspondiente bit y los bits en STW12 - STW18 muestran 0, incluso si los módulos del *rack* están fallados o mal configurados.

Empleo de las palabras de estado en el programa RLL

El programa que se muestra a continuación muestra un método para desconectar un módulo fallado y conectar el módulo de respaldo, ubicado en la misma base.

Módulo 1 en slot 1	Rack 1: WX1-WX8-STW11	BIT 8
Módulo 2 en Slot 2	Rack 2: WX9-WX16-STW12	BIT 15
Módulo 3 en Slot 3	Rack 1: Y17 - Y24	

Y24 : Módulo de alarma de falla

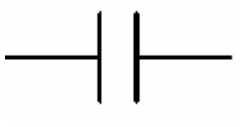


El estado del módulo de I/O Nº 1 se verifica con la instrucción BITP. Si BITP indica falla (bit 8 de STW11 en 1), Y24 se conecta (alarma). El programa desconecta el segundo módulo, para conectar el módulo de respaldo.

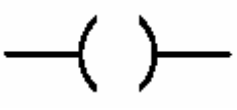
2.11 Funciones RLL

Contactos y bobinas

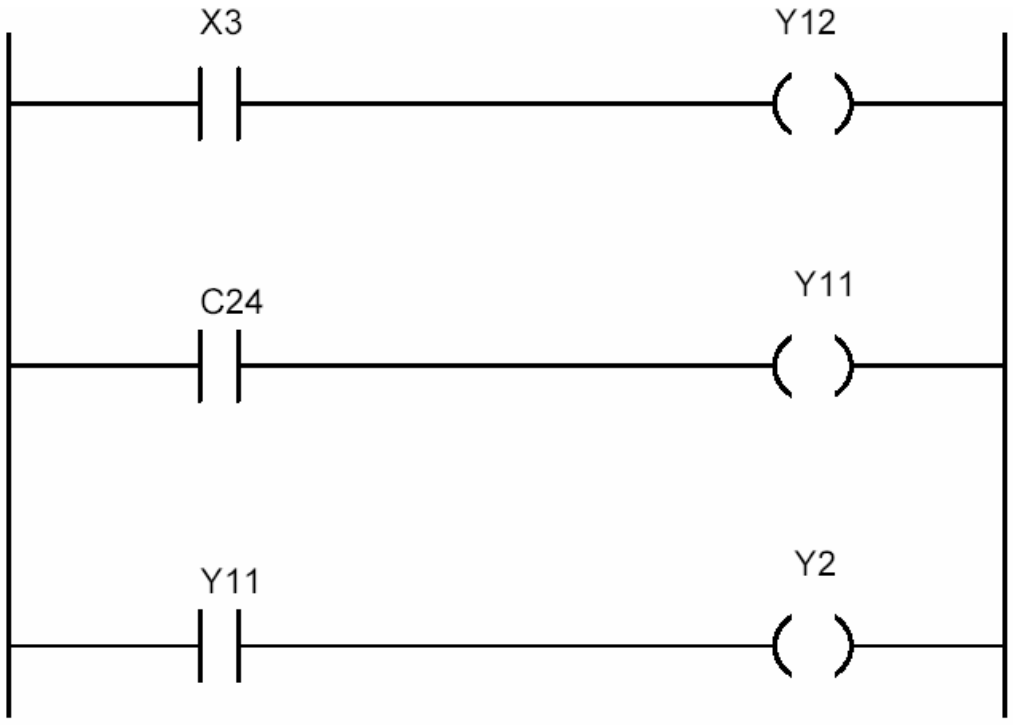
Representación de contactos:



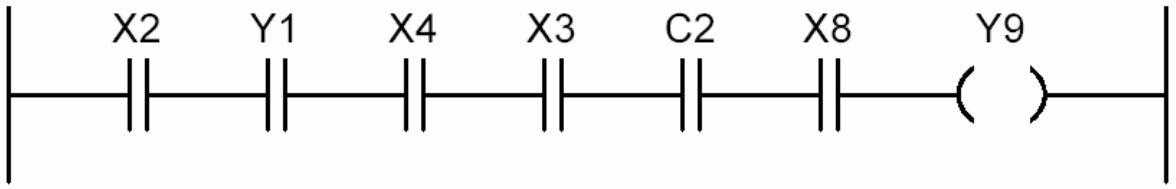
Representación de bobinas:



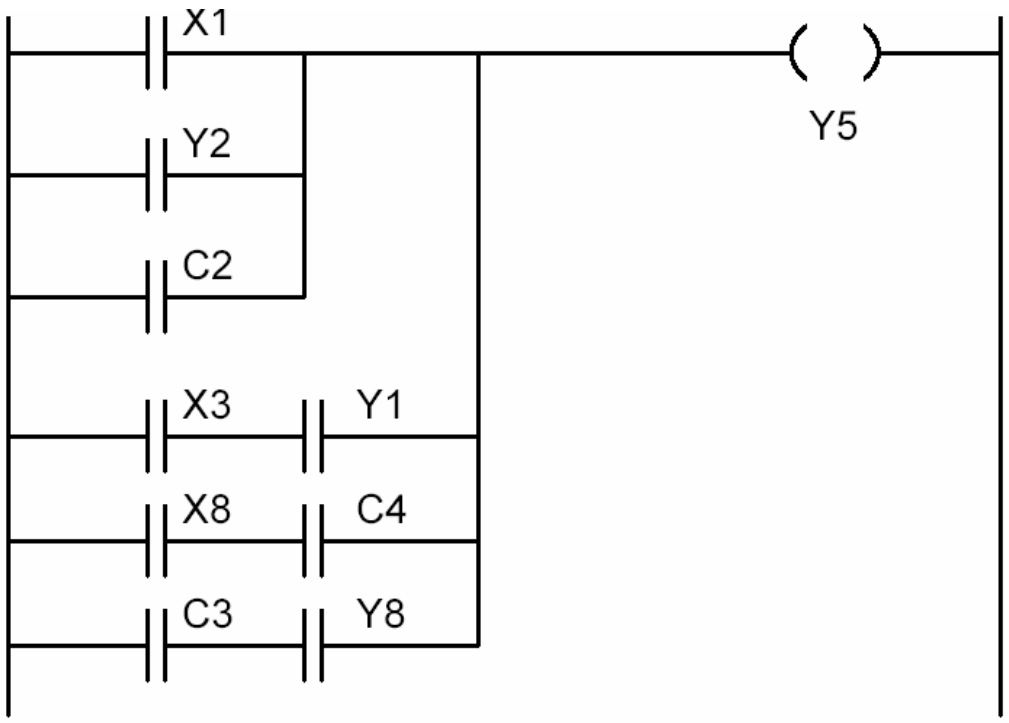
Las entradas y salidas físicas de módulos I/O se designan como Xn y Yn respectivamente, donde n es el número de referencia, por lo general n: 0-1023. Mientras que los relés de control utilizados en lazos intermedios se denotan como Cn.



Conexión serie de contactos: ANDs



Conexión paralelo de contactos: ORs



Funciones en caja

Se representan por un rectángulo que contiene en su interior un identificador descriptor de la función (código nemónico) seguido por un número de de referencia. Dependiendo de la instrucción, este número puede variar de 1 a 32767.

Línea de 1 entrada con caja pequeña

Debe tener a lo menos un contacto antes de la caja y no más

CAPITULO III

PROGRAMACION, INSTALACION Y MONTAJE

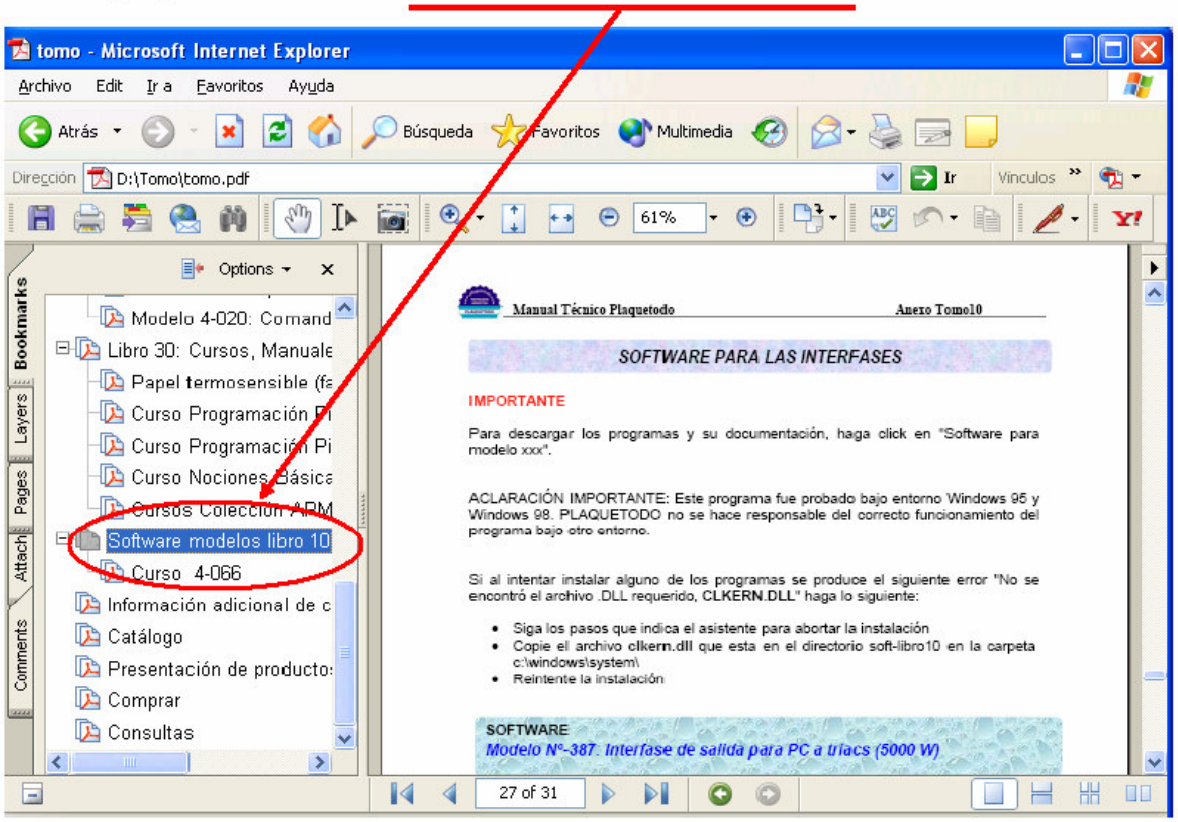
El objetivo del presente curso es el aprendizaje de la programación y puesta en marcha de nuestro PLC, modelo 4-036/1 del libro 10.
En principio se mostrará la instalación y uso del soft que acompaña al modelo. El mismo fue realizado por la Editorial Técnica Plaquetodo y es de muy fácil uso, sólo requiere saber utilizar Windows. Luego se mostrará la forma de conexión y puesta en marcha.

3.1INSTALACIÓN DEL SOFT:

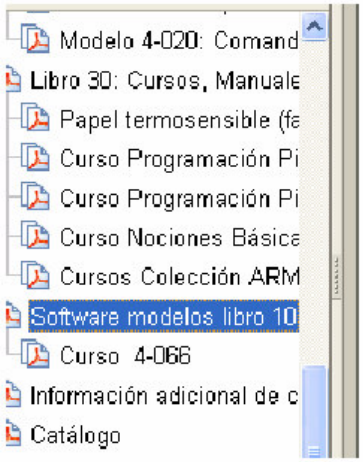
El modelo 4-036/1 se acompaña de un CD. En él encontrará el soft necesario para su programación.

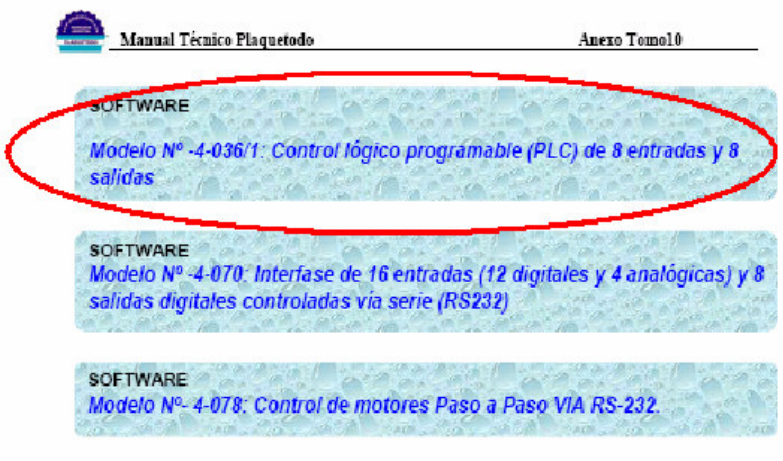
Ingrese el Cd al reproductor. La primera pantalla con la que se encontrará es la siguiente, haga clic sobre el icono de comenzar:

A la izquierda aparece el bookmark, en donde encontrará todo el contenido del CD. Debe bajar y hacer clic sobre "SOFTWARE MODELOS LIBRO10":

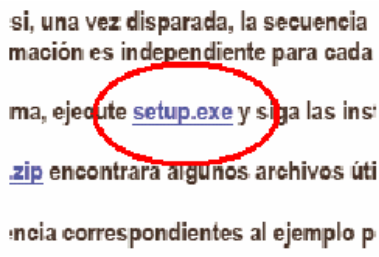


Aparecerán a la derecha todos los modelos correspondientes al libro 10 del Manual Técnico Plaquetodo. Baje hasta encontrar el modelo 4-036/1, y haga clic:



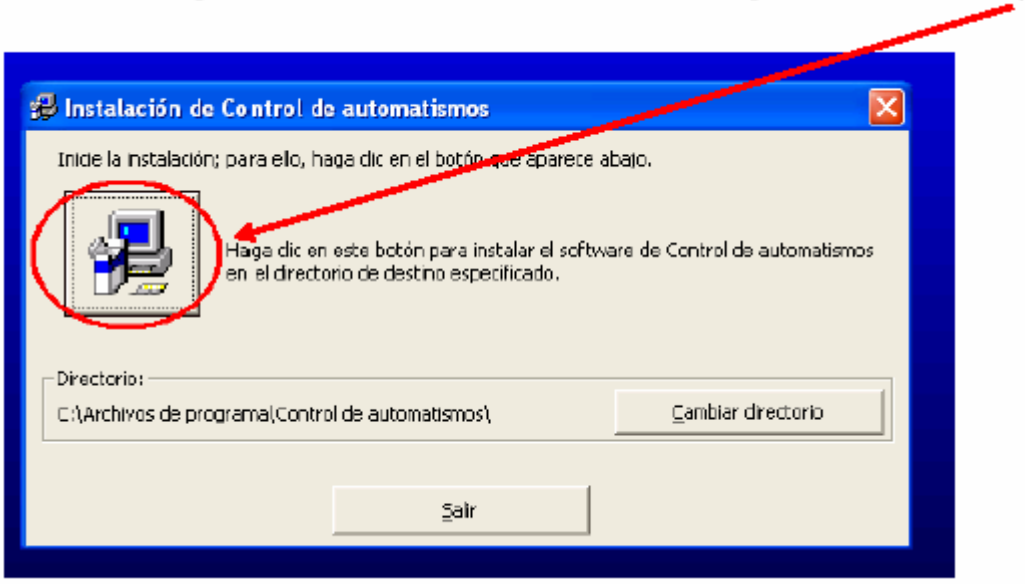


Para instalar el programa debe ejecutar, “setup.exe” (haciendo clic sobre él).

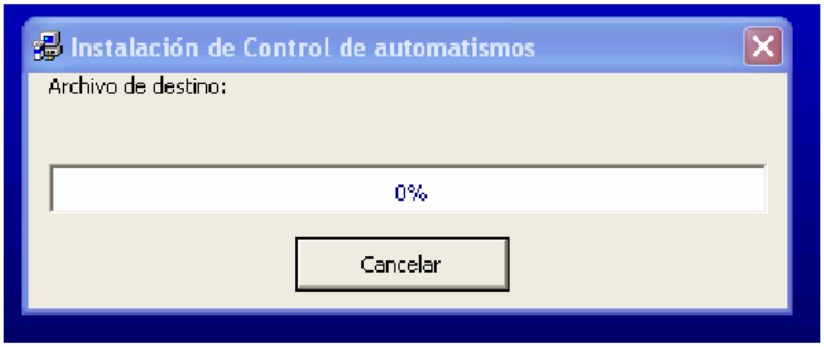


En la siguiente ventana, haga clic en Aceptar:

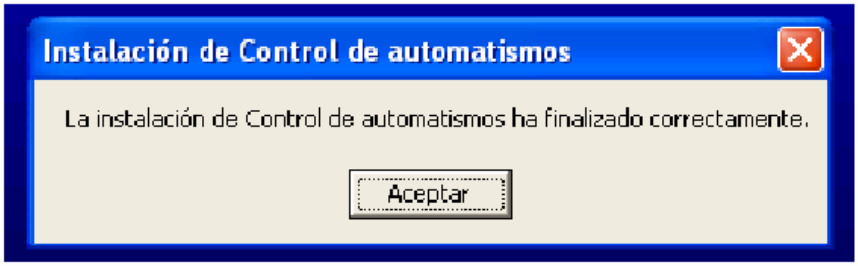
Para iniciar la instalación, haga clic sobre el icono con la imagen de una computadora:



La instalación comenzará, e irá mostrando el porcentaje.



Una vez finalizada la instalación, aparecerá la siguiente ventana:

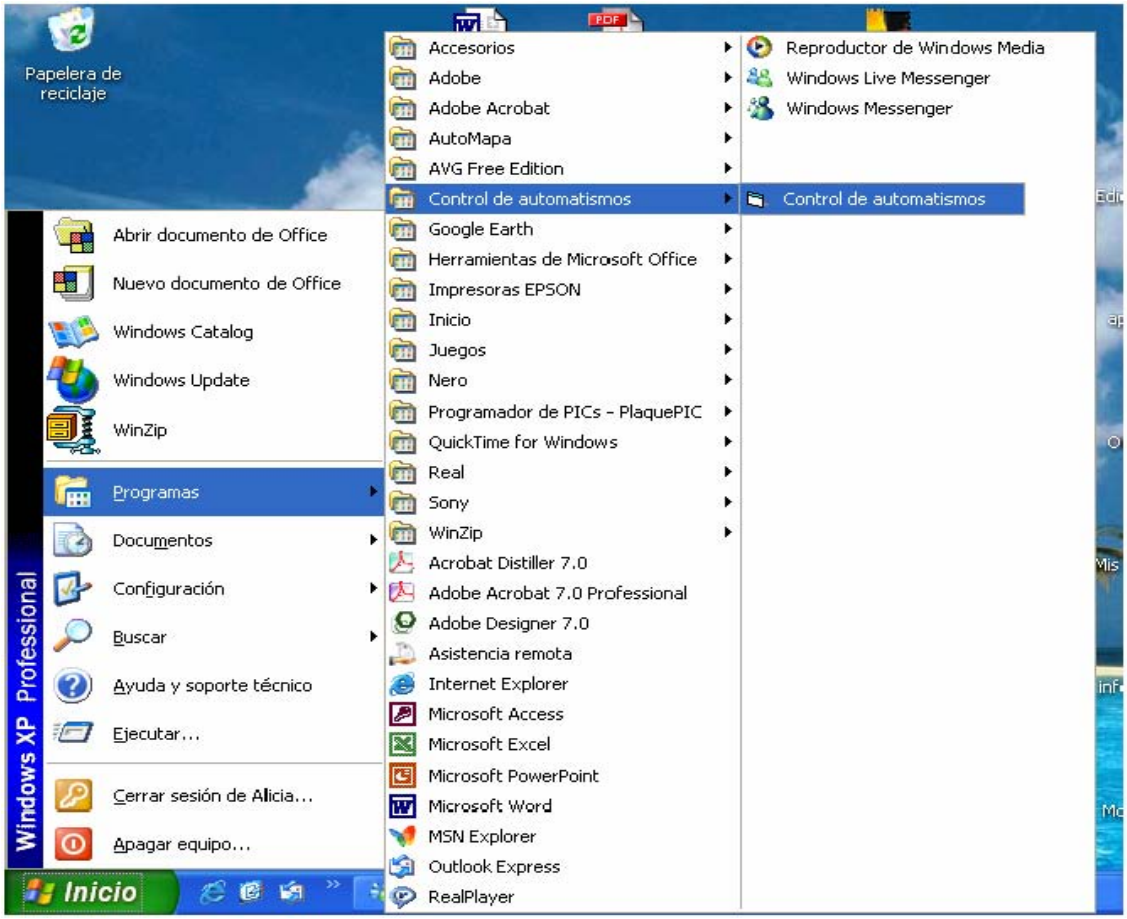


SI SU SISTEMA OPERATIVO ES WINDOWS XP

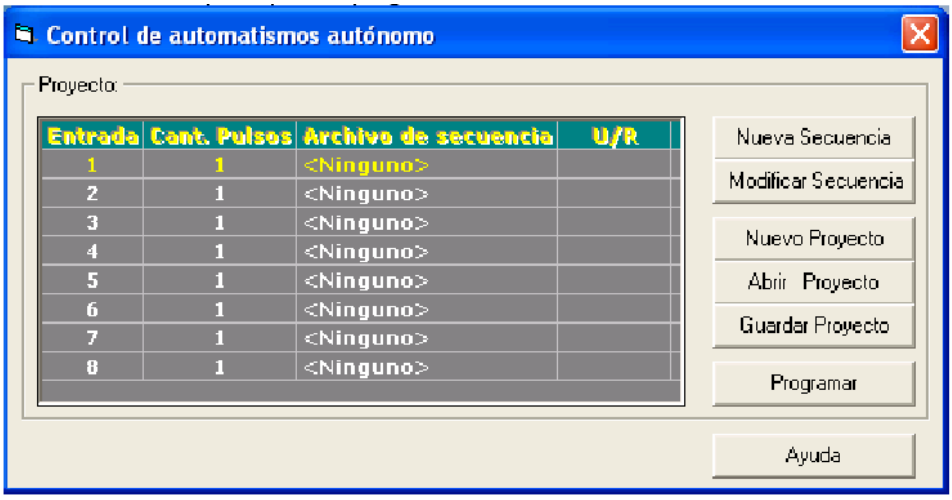
En caso de ser utilizado con Windows XP, debe instalar previamente en su PC el Paquete de redistribución de archivos de tiempo de ejecución para Visual Basic 6 **vbrun60sp6.exe** que puede ser descargado gratuitamente de la página de Microsoft.

3.2 PROGRAMACIÓN

Para abrir el soft. Se debe ir a:
INICIO > PROGRAMAS > CONTROL DE AUTOMATISMOS.



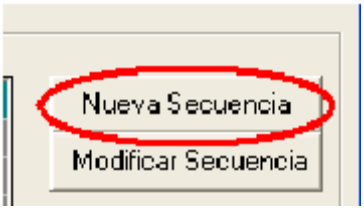
Se abrirá la ventana principal del programa:



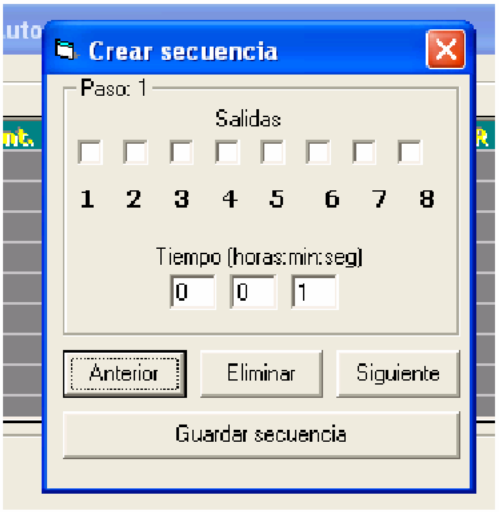
En principio debemos crear las diferentes secuencias que deseamos que realice el PLC.
Luego asignaremos cada una de ellas a una entrada particular.

Cada vez que el PLC detecta un pulso en cualquiera de sus entradas, se activa la secuencia correspondiente, cortando la que estaba realizando.

Para crear una secuencia, se debe hacer clic sobre:



Se abrirá la siguiente pantalla:



Simplemente se debe hacer clic sobre la ó las salidas que se desean activar en el primer paso de esta secuencia.

Luego colocar el tiempo de duración de encendida/as dicha/s salida/s.

Si hace clic en siguiente, programará el segundo paso de esta misma secuencia.

Veamos un ejemplo concreto. Supongamos querer simular la presencia de una persona es nuestro hogar.

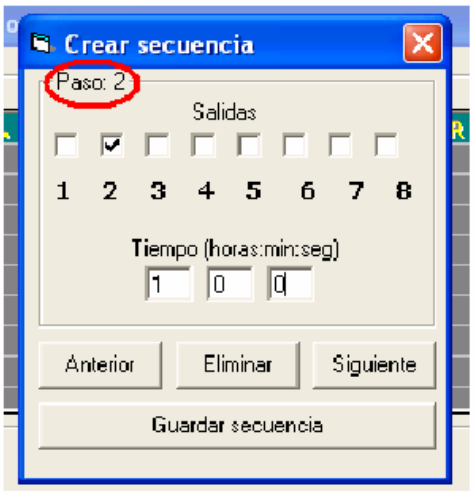
Podríamos, por ejemplo, presionar un botón al salir de la casa que active una secuencia de prendido/apagado de diferentes elementos que simulen la presencia de una persona (TV, lámparas, radio, etc.).

Entonces lo primero que deseamos es encender una lámpara exterior conectada a la salida1, durante 10 minutos. Luego que esta se apague, y prender la radio (conectado a la salida2). Y ahora mientras la radio esta encendido, que se prenda otra lámpara en el interior de la casa conectada a la salida3.

Cabe aclarar que el ejemplo sencillo que se esta dando es sólo ilustrativo, para enseñar el uso del PLC. Aconsejamos al lector leer el ejemplo que se brinda en el Informe técnico del modelo. Allí encontrará una aplicación más acorde con todos los beneficios que ofrece el PLC (Automatización de una planta envasadora de pintura)

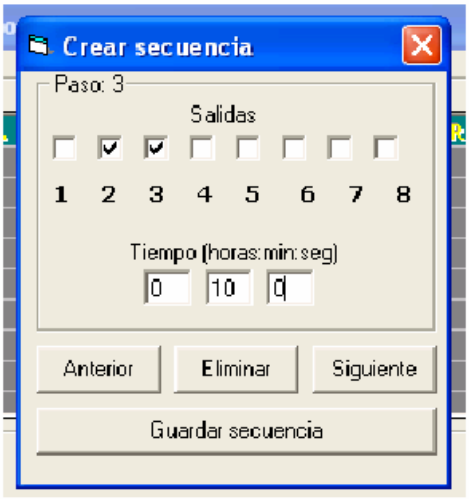
Volviendo al ejemplo, la programación sería:

Hacemos clic en siguiente, y se programa el paso 2:



Marcamos la salida2 (donde se había conectado la radio). Sin marcar ninguna otra salida.

Luego, el paso3:



Notar que volvimos a tildar la salida2, para que la radio continúe encendida mientras la lámpara conectada a la salida3 se prende (el tiempo de encendido de 10 minutos fue elegido arbitrariamente).

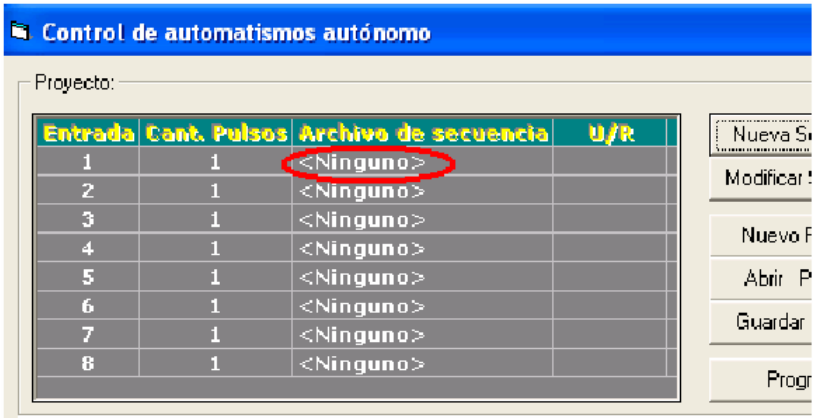
Una vez creado todos los pasos que uno desea, se debe guardar la secuencia. Haciendo clic sobre:

Los pasos de las secuencias pueden ser modificados, o eliminados mediante los controles: anterior, eliminar y siguiente.

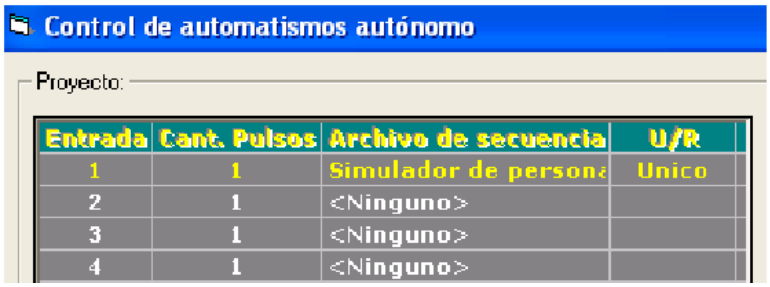
Las secuencias son guardadas en la carpeta “Control de automatismos” con la extensión “. aut”. En nuestro caso fue guardada con el nombre de “Simulador de personas”.
Se le aconseja al lector guardarlas con un nombre que describa el proceso que esta realizando.

Se deben crear todas las secuencias que se deseen asignar a cada entrada (máximo 8).

Para asignar una secuencia a una entrada particular basta con hacer doble clic sobre
<Ninguno> de la entrada correspondiente. En nuestro ejemplo, seleccionamos la entrada1:

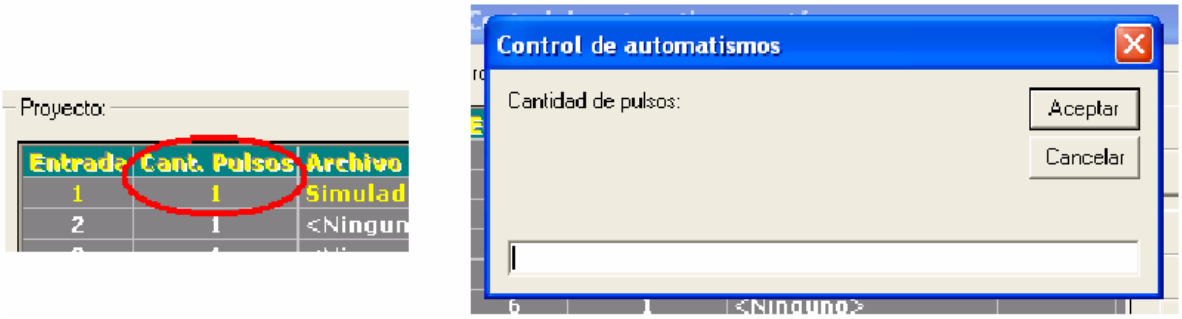


Seleccionamos la secuencia, y esta aparecerá en lugar de <Ninguno>:

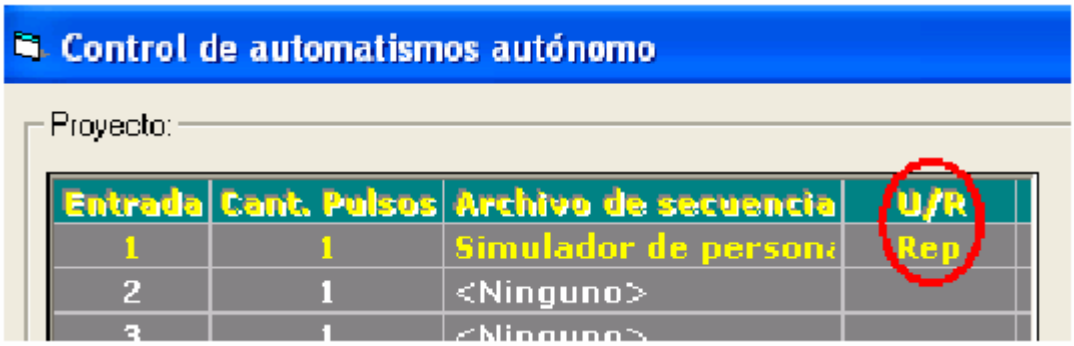


Así sucesivamente vamos asignando a cada entrada, la secuencia que deseamos que se active.

En cuanto a las otras columnas. La segunda (Cant. Pulsos) es la cantidad de pulsos que deseamos que ingresen para que esta secuencia se active. En nuestro caso, deseamos que al presionar un botón (conectado en la entrada1) se active esta secuencia. Por ende sólo colocamos un 1. Si quisiéramos que se active luego de apretar 3 veces el botón. Entonces colocamos un 3.



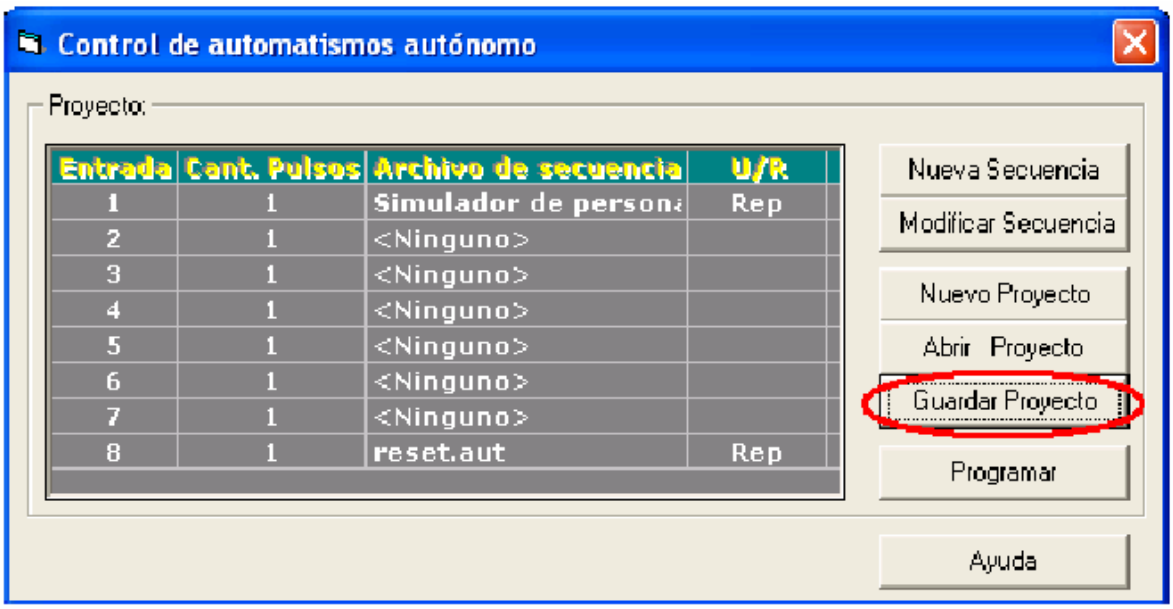
La última columna (U/R) significa repetitivo o único. Para nuestro ejemplo, si colocamos Repetitivo, luego del paso3 comenzaría de nuevo esta secuencia. Es decir se prende la lámpara del exterior 10 minutos, luego la radio 1 hora, y luego 10 minutos más junto a la lámpara del interior. Una vez terminado este proceso vuelve a comenzar: 10 minutos encendida la lámpara exterior, etc.



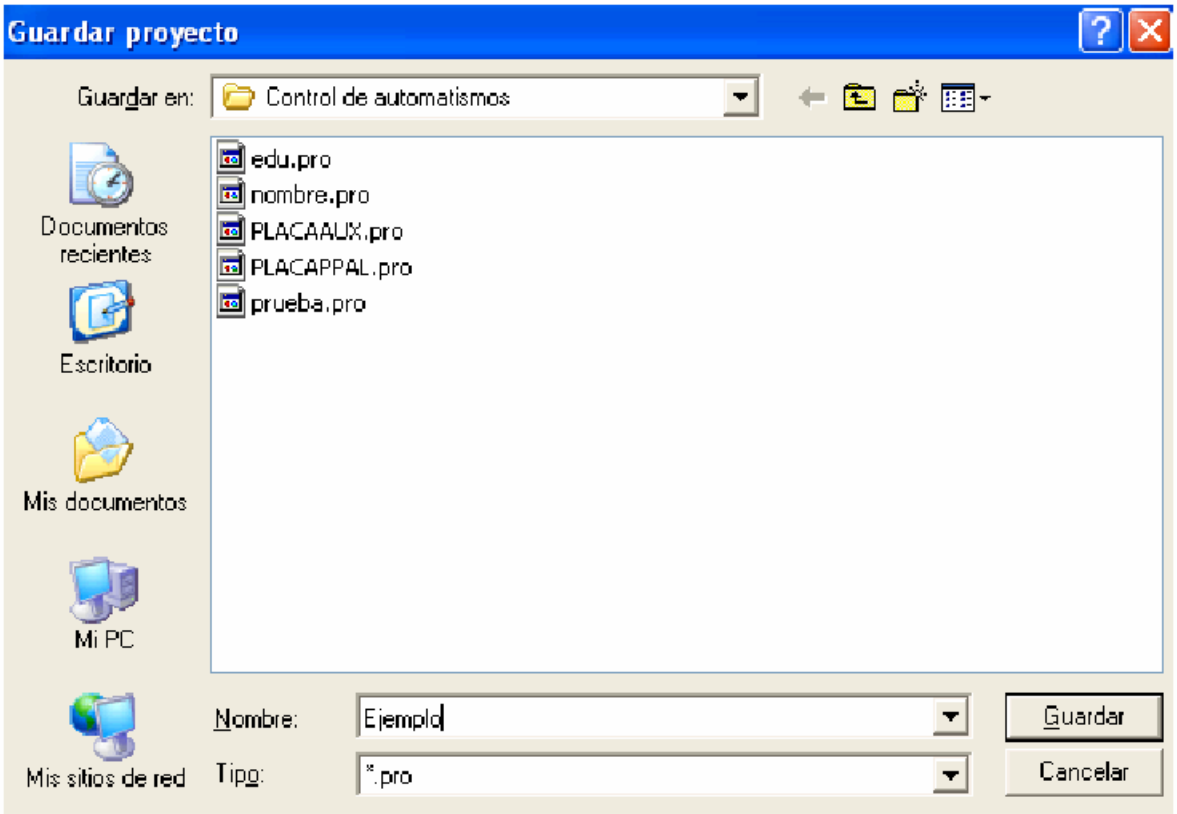
Si colocamos Único, entonces una vez terminada la secuencia el PLC se quedará stand-by esperando que otra entrada se active.

Es muy común utilizar el modo repetitivo como “Reset”. Creando una nueva secuencia en la cual no se active ninguna salida. Teniendo entonces la posibilidad de colocar un pulsador como comando manual de “Parada de emergencia”.

Una vez que se asignó una secuencia a cada entrada, debemos guardar el proyecto. En nuestro ejemplo utilizamos la secuencia “Simulador de personas.aut” para la Entrada1 y “Reset.aut” para la Entrada8. Para guardar el proyecto se debe hacer clic sobre “Guardar proyecto”:



El mismo queda guardado también, dentro de la carpeta "Control de automatismos", con la extensión ".pro".



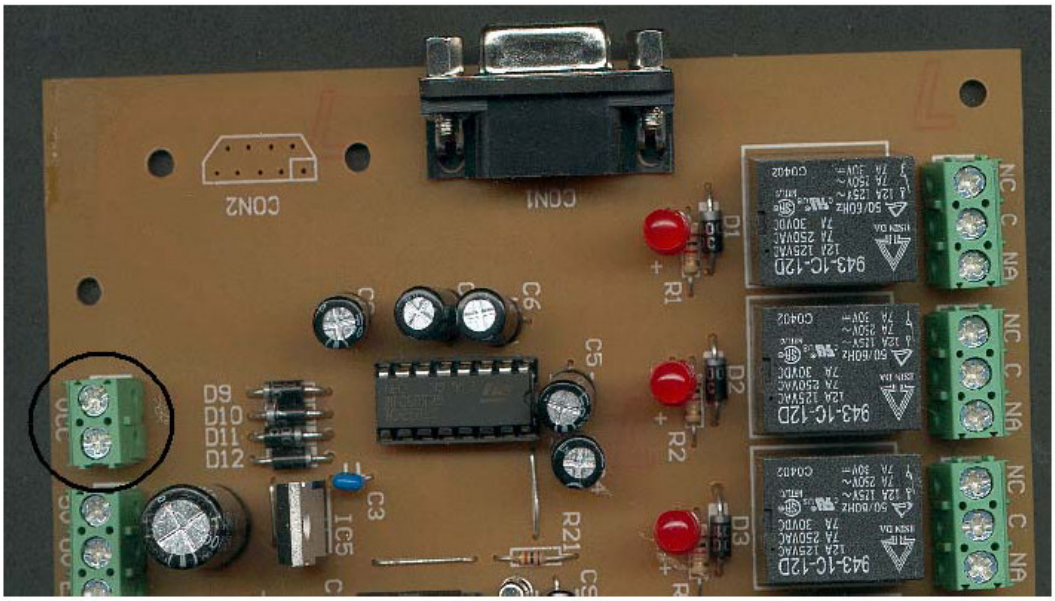
3.3 PASAJE DEL PROYECTO, DE LA PC A LA PLACA

Una vez guardado el proyecto, debemos volcar el programa al módulo. Lo primero que debemos hacer es conectar el cable DB9 (macho-hembra pin a pin) al COM de la PC.

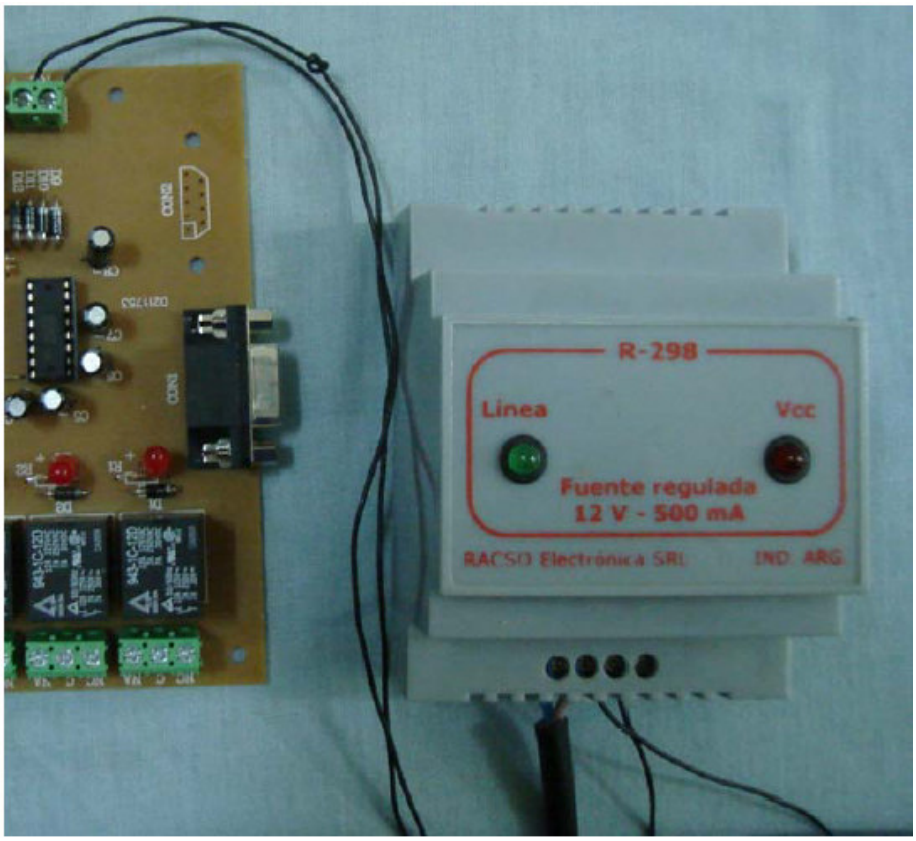
El conector hembra se conecta a la Pc:

Y el macho a l CON1 de la placa:

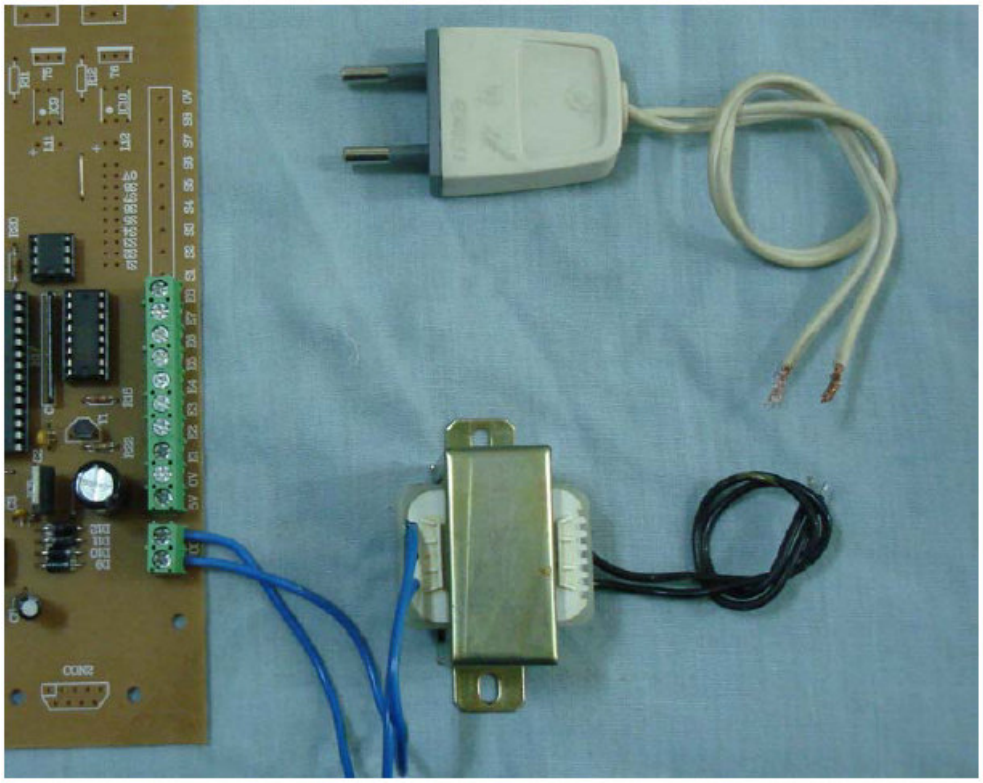
Luego se debe alimentar la placa, en la bornera indicada con Vcc:



Con una fuente de 12Vcc, 500mA. No tiene importancia la polaridad. Se puede colocar de cualquier forma (ya que el modelo incluye el puente rectificador).



O directamente con un transformador de 220/9Vca 500mA, ya que la placa incluye la rectificación:



Una vez alimentada, podemos proceder a programar. Para ello basta con hacer clic en:

El led8 (salida8) debe titilar 8 veces. Si esto no se produce, es porque hubo alguna falla en la comunicación. Desconecte la fuente y vuelva a conectarla.

Cabe aclarar que este modelo puede programarse tantas veces como uno lo desee. Ya sea porque se necesita modificar el programa, o porque se quiera realizar un nuevo automatismo.

Cuando la programación finaliza se abre una ventana indicando la correcta programación.

PUESTA EN MARCHA

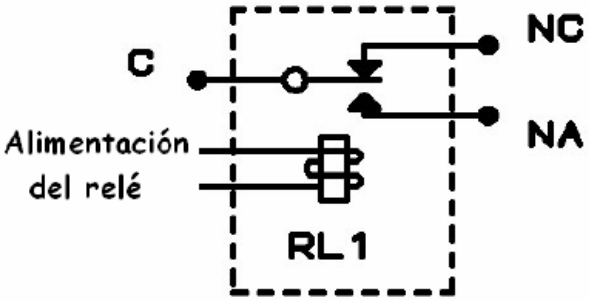
Una vez programada la placa, debe desconectar el cable de conexión a la PC (Se aconseja cortar la alimentación para esta acción).

El modelo ya esta listo para colocarlo en nuestro sistema.

SALIDAS:
Las salidas constan de 8 relés (llaves electromecánicas) que pueden utilizarse para activar cualquier tipo de carga. En particular estos soportan hasta 5A en 220Vca y 10A en 12Vcc.

Siguiendo con el ejemplo, veamos las conexiones que deberíamos implementar. A la salida1 y 3 debemos conectar lámparas y a la salida2 una radio.

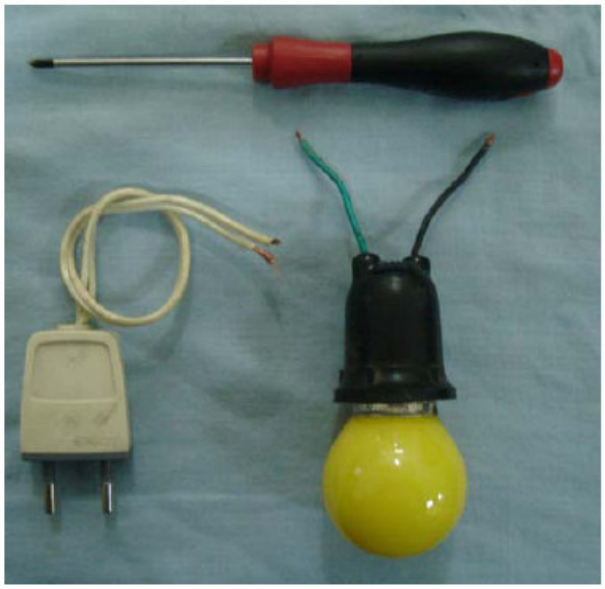
Cada salida a relé esta indicada con NA, NC y C. Estas se refieren a **N**ormalmente **A**bierto, **N**ormalmente **C**errado y **C**entral respectivamente. Se puede pensar como 2 llaves, una formada por NA-C y la otra por NC-C



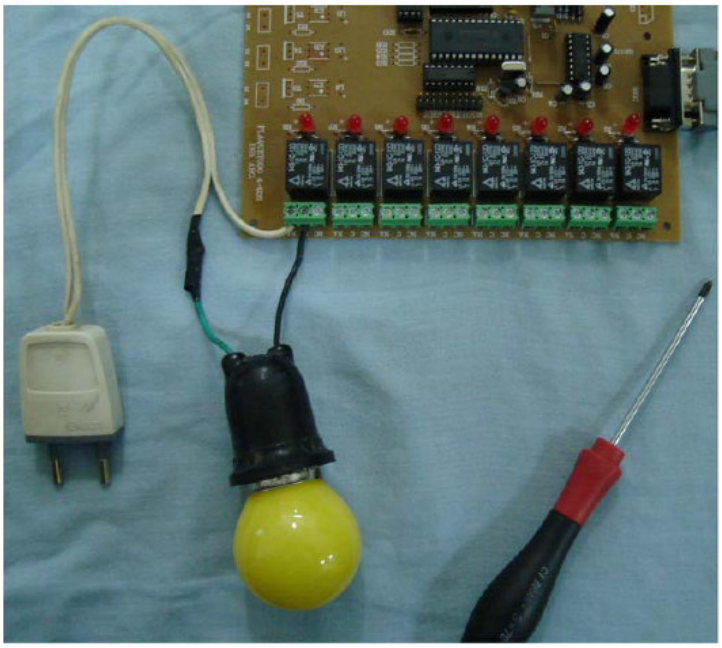
Cuando un relé se activa, el contacto NA (normalmente abierto) se une con C. Es decir la llave formada por NA-C se cierra. Y la llave formada por NC-C (normalmente cerrada) se abre.

Entonces, las conexiones tanto de las lámparas como la de la radio las haremos entre NA y C del relé correspondiente. La siguiente figura indica la conexión de la lámpara a la entrada1. De la misma manera debemos conectar la radio y la otra lámpara, al relé correspondiente.

Las herramientas necesarias son:

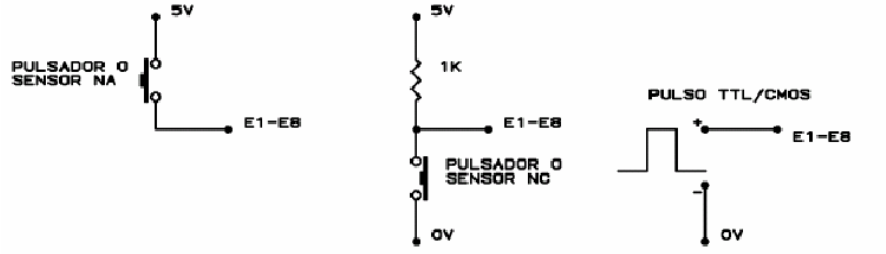


Y la conexión es la siguiente:

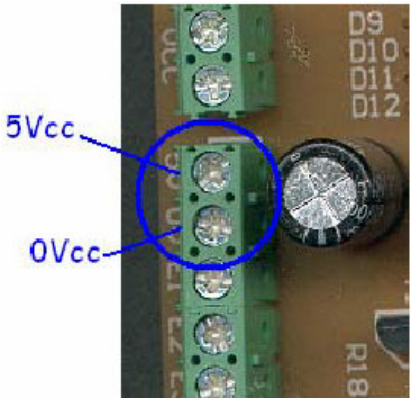


ENTRADAS:

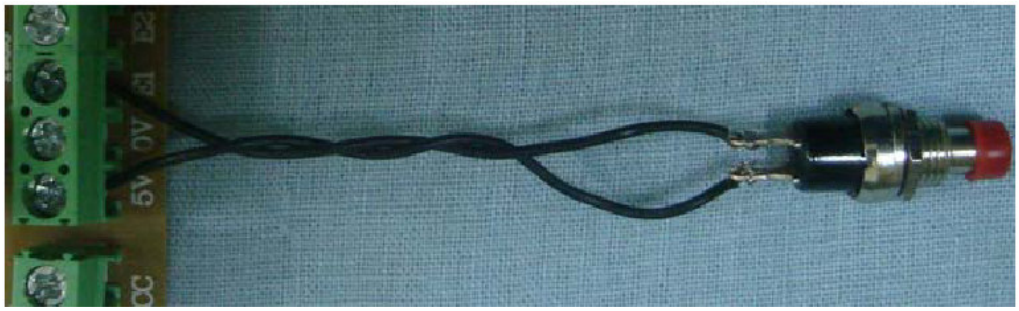
Las entradas son del tipo digital (si/no). Soportan tanto señales TTL como CMOS. Se puede conectar pulsadores y la salida de cualquier tipo de sensor:



Para ello la placa brinda específicamente dos bornes especiales con 5Vcc y 0Vcc, a continuación de las borneras de las entradas (E1-E8).

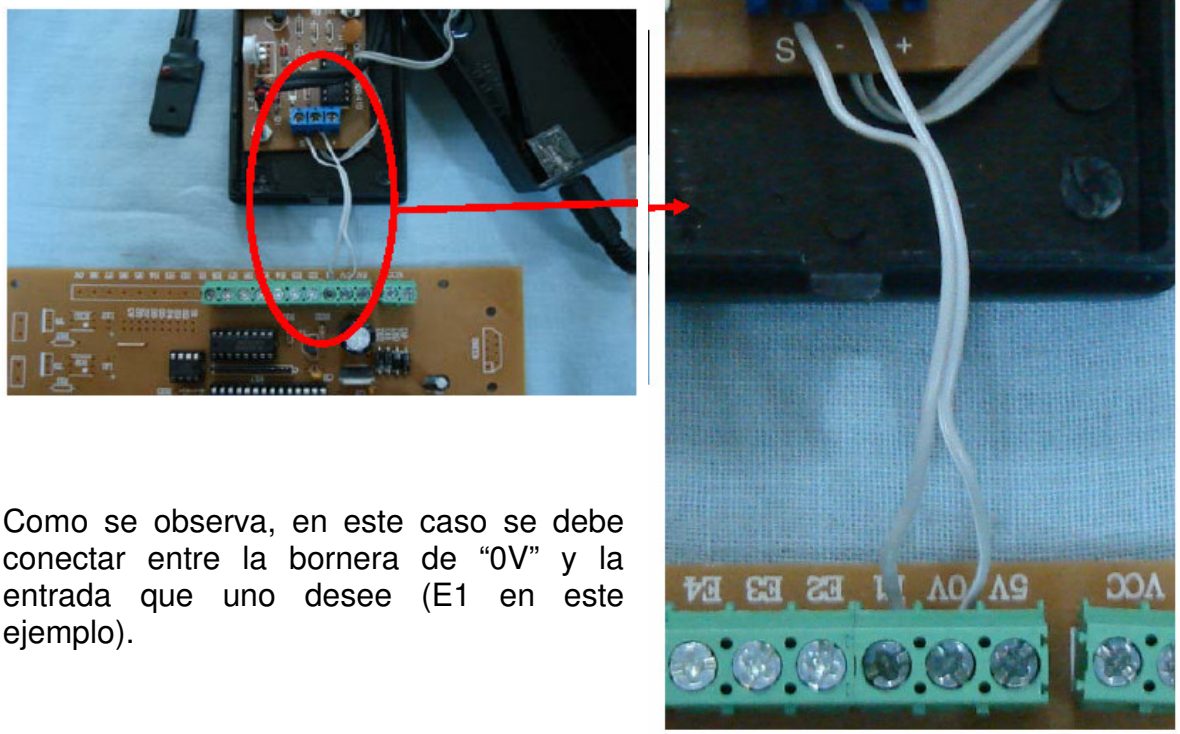


Entonces la conexión con un pulsador NA a la entrada1 es de la siguiente forma:



Al ser un pulsador del tipo NA, la conexión se realiza entre la bornera de “5V” y la entrada que uno desee (E1 en este ejemplo)
Para el caso de utilizar un sensor en lugar de un pulsador, la conexión es la siguiente.

En la imagen se muestro la conexión con un sensor magnético



Como se observa, en este caso se debe conectar entre la bornera de “0V” y la entrada que uno desee (E1 en este ejemplo).

3.4 CONSIDERACIONES DE INSTALACION Y MONTAJE

Preparación del lugar de instalación

. Definición de los requerimientos de control.
. Determinar el número de PLC requeridos.
. Determinar disposición de paneles y tierras.

Los requerimientos de control se definen en términos del número de entradas y salidas. Posteriormente, se calculan los módulos de I/O y los *racks* que se necesitan. Una vez que se conocen los PLC, módulos de I/0 y *racks* requeridos, se deben determinar la potencia necesaria para el funcionamiento correcto del PLC.

La potencia total requerida en la instalación, considerando el PLC, módulos de I/0 y módulos controladores, no debe exceder la capacidad disponible de la fuente de poder.

Consideraciones de seguridad

Al diseñar el sistema, se deben tener en cuenta las condiciones de seguridad del personal durante fallas. Los equipos conectados al PLC deben incluir *interlocks* y *switches* de seguridad, que prevengan la operación al producirse una falla.

. Debe existir un medio para desconectar la alimentación de energía a las cargas (salidas), independiente del PLC, para operaciones de rutina.

. Debe existir un medio para desconectar la alimentación de energía a las salidas, para condiciones de emergencia.

. Se deben utilizar circuitos *by-pass* externos para operaciones de partida o inicialización (cargas críticas).

Encapsulado (*Enclosure)*

Requerimientos mínimos

. Fácil acceso a componentes.

. Potencial de tierra común para el gabinete.

. Instalación en rieles o paneles verticales de seguridad.

. Cumplir estándares o normas eléctricas.

. Protección EMI.

. Restringir acceso a los equipos.

. Protección contra polvo y suciedad.

. Normas NEMA.

Consideraciones de temperatura

Se debe asegurar un adecuado flujo de aire, de modo que se obtenga una buena refrigeración del equipo.

Si la temperatura ambiente es alta, se debe utilizar ventilación forzada o acondicionamiento de aire. La temperatura máxima de operación típica es 60º C.

3.5Consideraciones eléctricas

Tierras

Para obtener una operación adecuada, es fundamental contar con un buen sistema de conexión a tierra. Se recomienda la utilización de cable trenzado de cobre Nº12 AWG o de mayor grosor en el retorno de tierra.

Algunas reglas para lograr un buen contacto eléctrico:

. Se deben emplear terminales adecuados en los extremos de los cables de tierra.
. Es recomendable utilizar pernos de cobre para realizar la conexión al punto de tierra.
. La pintura, recubrimientos y el óxido impiden un buen contacto en los puntos de tierra. Se deben remover y emplear golillas dentadas para asegurar una buena continuidad y baja impedancia.

Alambrado

Algunas consideraciones que se deben tener en cuenta en el alambrado:

. Emplear cables de largo mínimo.
. No añadir cables.
. Evitar la proximidad de cables de alta potencia.
. Instalar cablería de entrada, salida y de otro tipo en paneles separados.
. Cuando sea posible, canalizar por separado los cables con señales DC y AC.
. Una impedancia de 0.1. o menor debe haber en la conexión a tierra de todos los componentes del sistema.
. Utilizar guías de cable.
. Proteger los cables desnudos.
. No utilizar el mismo cable de retorno de alimentación cuando las líneas son muy largas; de esta forma se minimiza la caída de voltaje.

Minimización del ruido eléctrico

Fuentes de ruido

El ruido puede ser conducido a través de los cables de señal o de alimentación, o puede ser irradiado por ondas electromagnéticas.

El acoplamiento electrostático se produce a través de las capacitancias parásitas existentes entre la línea de ruido y la línea de alimentación o señal. Este es el caso típico cuando se canalizan cables largos en un mismo *conduit*.

El acoplamiento magnético ocurre a través de las inductancias mutuas parásitas entre líneas.

El ruido electromagnético irradiado es generalmente de alta frecuencia. Se debe tener especial cuidado en el sistema de control y su alambrado, ya que pueden actuar como antenas.

Las fuentes primarias de ruido en ambientes industriales son:

. Motores grandes.
. Máquinas soldadoras.
. Contactores (*switch* con cargas electromagnéticas).
. Máquinas de estado sólido.

Eliminación del ruido

El empleo de supresores de ruido *snubbing* permite reducirlo en su origen. Son aplicables en dispositivos comandados por contactos mecánicos, y suprimen el arco en los contactos eléctricos (cargas inductivas).

Un tipo alternativo de supresor se logra con circuitos RC o varistores.

Aislamiento del ruido

Otra forma de manejar el problema de ruido, consiste en aislar el dispositivo que presenta problemas de ruido, de los cables y componentes electrónicos. Adicionalmente y en casos extremos, se emplean escudos electrostáticos.

Una medida complementaria, especial para cables con señales de valores bajos (TTL), se consigue con protecciones de malla y trenzado (12 vueltas/pie). Además, se debe mantener la separación física con los emisores.

CONCLUSIONES

Como se pudo notar tanto la programación como la puesta en marcha es muy sencilla. No requiere de mayores conocimientos. Por ello, este modelo es especial para todo tipo de automatismos que no requieran de una lógica complicada, se puede aplicar para control de máquinas, herramientas, sistemas de domótica para el hogar, etc.

ALGUNAS RECOMENDACIONES

Si quiere finalizar su proyecto satisfactoriamente recomendamos seguir los siguientes pasos:

1- Plantee el proyecto en una forma clara tratando de identificar cada parte, sus etapas y procesos. Si utiliza sensores, primero obtenga la información de cómo funcionan, estúdielos y luego avance al siguiente punto. Tenga presente que los pulsos de entrada deben ser positivos y de una duración mayor a 100 milisegundos. El pulso es detectado en el flanco de subida (dispara la secuencia asociada) y mientras en una entrada haya un nivel positivo (por ejemplo, un pulsador trabado), los pulsos que entren en otras entradas serán ignorados (no se disparará otra secuencia).

2- Elabore un diagrama eléctrico con los sensores, entradas y salidas utilizados. Asigne a cada uno el nombre de la entrada (E1, E2 etc.) y salida que corresponda.

3- Elabore el listado con las variables que requiera el programa, para esto utilice la hoja de proyecto que se provee en el CD adjunto al modelo.

4- Verifique el funcionamiento eléctrico de cada parte en forma manual (entradas y salidas) antes se conectar al equipo.

GLOSARIO

!=F
: Comparar si son iguales.

)
: Cerrar un circuito complejo.

****
: Segmento

+F
: Suma

<=
: Comparar si es igual o menor

<F
: Comparar si es menor

=
: Conectar bobinas

><F
: Comparar si son diferentes

>=F
: Comparar si es igual o superior

>F
: Comparar si es superior

A
: Salida de bit

AB
: Salida de byte

ADB
: Llamada a un módulo de datos

AG
: Autómata o PLC
Ver también: FD, PG

AKKU1
: Zona de memoria donde se guarda los valores cargados con la instrucción de carga L o el resultado de operaciones aritméticas

AKKU2
: Zona de memoria donde se deposita el valor del AKKU1 cuando este es cargado con otro valor.

AW
: Salida de palabra

AWL
: Programar en lista instrucciones.
Ver también: FUP, KOP

BCD
: Código Binario Decimal.

BE
: Fin módulo

BEA

Fin de módulo de forma absoluta

BEB

Fin de módulo condicional (Si =1) VKE

DB

Módulo de datos

DWn

Palabra *n* de un módulo de datos (podemos cargar L y transferirla T)

DE

Salida digital con en contenido del contador en BCD

DU

Salida digital con el valor del contador en binario

E

Entrada de bit

EB

Entrada de byte

EW

Entrada de palabra

-F

Resta

FB

Módulo de funciones

FD

Define la el PC como consola
Ver también: AG, PG

FUP

Programar puertas lógicas
Ver también: AWL, KOP

K

Constantes de bit

KB

Constante de byte

KB

Formato constante de byte en decimal

KF

Formato en coma fija (palabra). Disponemos de signo

KH

Formato en hexadecimal (palabra)

KM

Formato muestra de bit

KOP

Programar esquema de contactos
Ver también: AWL, FUP

KT

Constante de tiempo.
Formato: unidades_de_tiempo.base_de_tiempo
unidades_de_tiempo: como máximo 3 dígitos
base_de_tiempo:

0 = centésimas de segundo
1 = décimas de segundo
2 = Segundos
3 = 10 segundos

KY

Formato nº absoluto

KZ

Constante para un contador
Formato: 000 a 999

L

Carga valor en AKKU1

M

Marcas (merker)

O

Paralelo

O(

Poner en paralelo un circuito complejo

OB

Módulo de organización

PB

Byte de periferia

PB

Módulo de programa

PG

Consola de programación
Ver también: AG, FD

PW

Palabra de periferia

PY

Byte de periferia

R

Reset (poner a 0 un bit)

S

Set (poner a 1 un bit)

SA

Temporizador con retardo a la desconexión

SE

Temporizador con retardo a la conexión

SI

Temporizador de impulsos

SPA

Llamada a módulo incondicional

SPB

Llamada a módulo condicional

SPM =

Salto cuando el resultado de la operación es negativo

SPN =
Salto cuando el resultado de la operación no es 0

SPO =
Salto cuando el resultado de la operación produce un desbordamiento

SPP =
Salto cuando el resultado de la operación es positivo

SPZ =
Salto cuando el resultado de la operación es 0

SS
Temporizador con retardo a la conexión memorizado

SV
Temporizador de impulsos prolongados

T
Transfiere valor del AKKU1 al byte o palabra especifica

Tn
Define temporizador n indica el nº de temporizador
Ejemplo de uso:

```
U E 1.0     Con la entrada E1.0 activamos el temporizador
L KT 35.1   Le asignamos una constante de tiempo
SE T1       Definimos el temporizador 1 como temporizador a la conexión
U T1        El contacto del temporizador activará la salida A2.0
= A2.0
```

U
Serie

U(
Poner en serie un circuito complejo

VKE
Zona de memoria donde se guarda el resultado intermedio de operaciones con contactos.

Zn
Define un contador donde n es el número de contador
Ejemplo de uso:

```
U E 32.0    Con la entrada E32.0 mandamos un impulso al contador
ZR Z1       El impulso lo mandamos al contador para que descuente

U E 32.1    Con la Entrada E32.1 activamos el contador.
L KZ10      Cargamos el valor a contar en el acumulador.
S Z1        Y por último cargamos el contador con el valor anterior.

U Z1        Cuando el contador deje de tener el valor 0
=A2.0       Activamos la salida A2.0
```

IMPORTANTE: Para contar hasta 10 tenemos primero que poner el contador a 10 y luego decrementar hasta 0.

ZR
Contar hacia atrás

ZV

Contar hacia adelante

USTACK

Diagnostico de averías

BIBLIOGRAFIA.

**Aprenda a programar un PLC-
Libro 10.**

**Curso PLC.
Siemens**