LABORATORIO I DE CONTROL

CONTROLADOR LÓGICO PROGRAMABLE **PLC**

Prof. Gerardo Torres - gerardotorres@ula.ve - Cubículo 003
Escuela de Ingeniería Eléctrica de la Facultad de Ingeniería de la Universidad de Los Andes

INTRODUCCIÓN

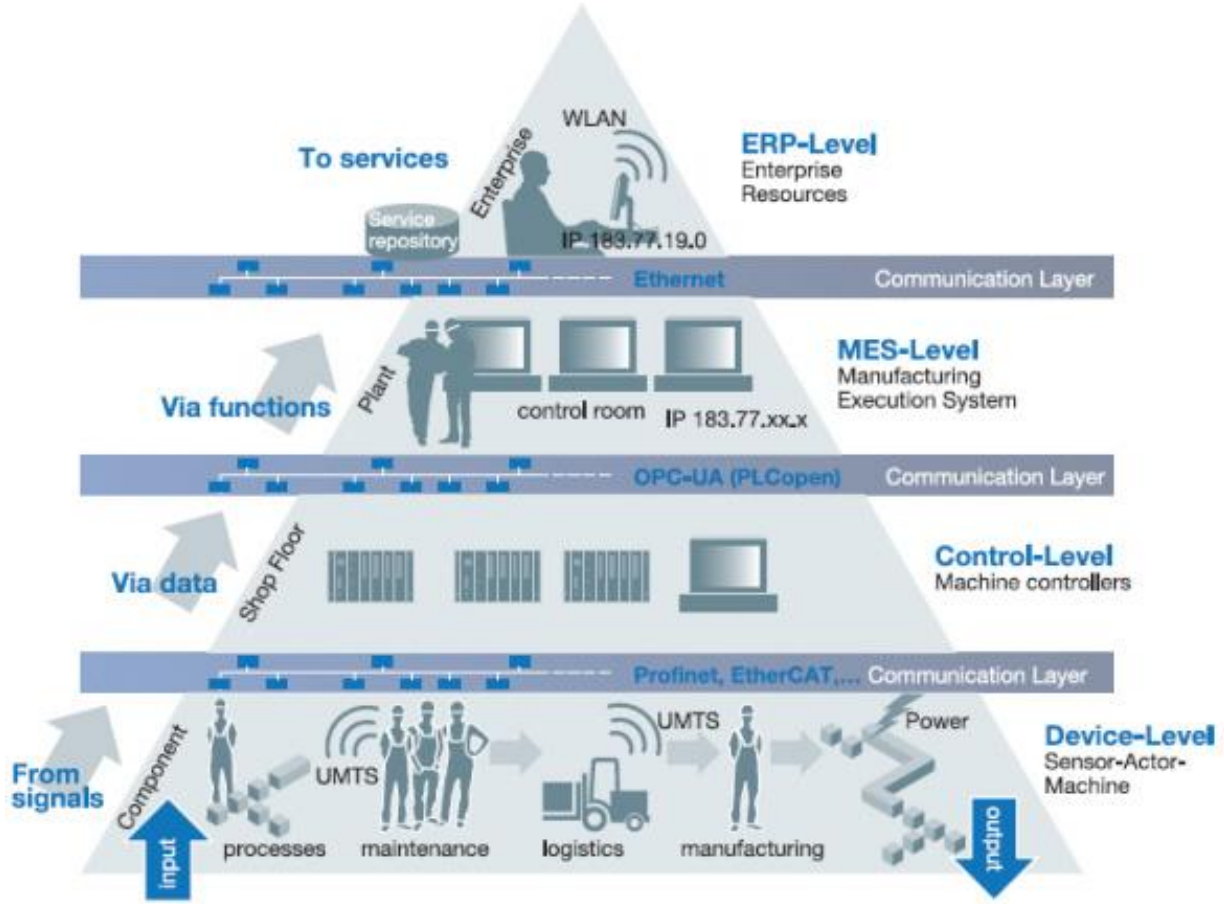


Fig. 1. Pirámide de automatización.

INTRODUCCIÓN

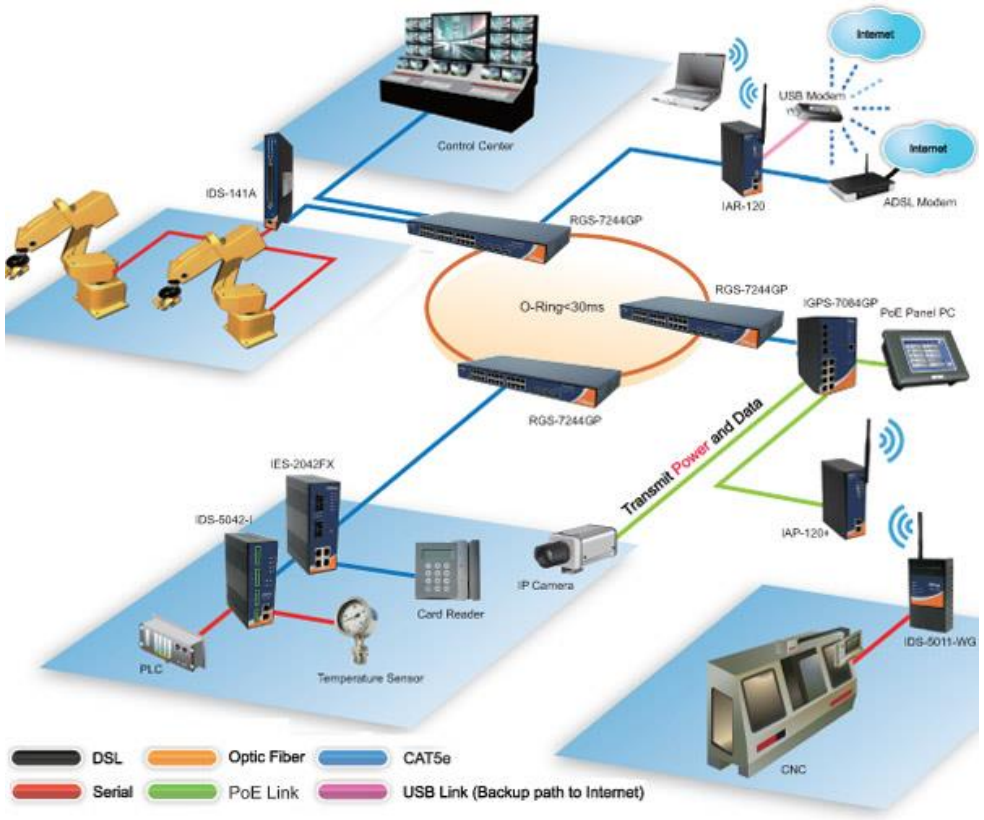


Fig. 2. Diagrama de automatización.

INTRODUCCIÓN

Fig. 3. Panel de control basado en relés

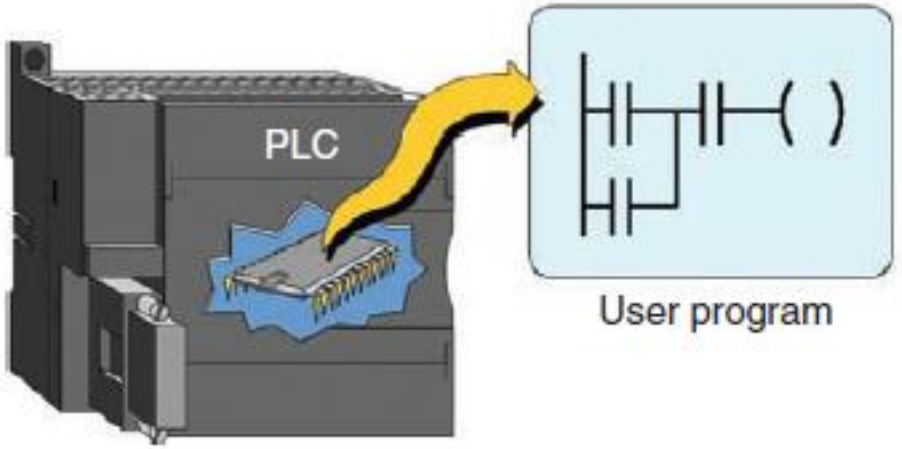
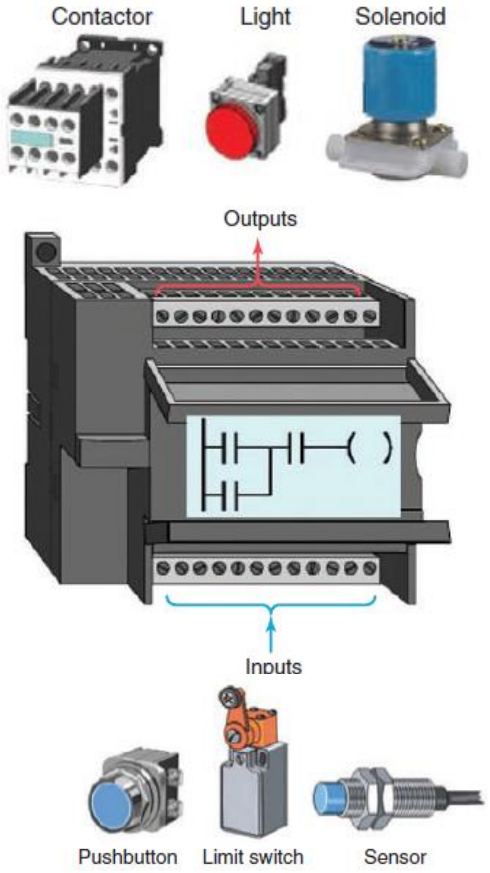


Fig. 4. La lógica de control está contenida en el PLC

INTRODUCCIÓN



- Múltiples entradas y salidas.
- Rangos de temperatura amplios.
- Inmunes al ruido eléctrico.
- Resistencia a la vibración y a impactos.

Fig. 5. Relación entre entradas y salidas con respecto al PLC y al programa.

VENTAJAS DEL PLC

- Menor cableado comparado con el control convencional de relés.
- Es mas pequeño y económico comparado que el control convencional de relés.
- Incrementa la confiabilidad.
- Mayor flexibilidad.
- Capacidad de comunicaciones.
- Rápida respuesta de tiempo.
- Fácil para solucionar problemas.
- Capacidad de desarrollo de aplicaciones básicas y complejas.

DEFINICIÓN DE PLC

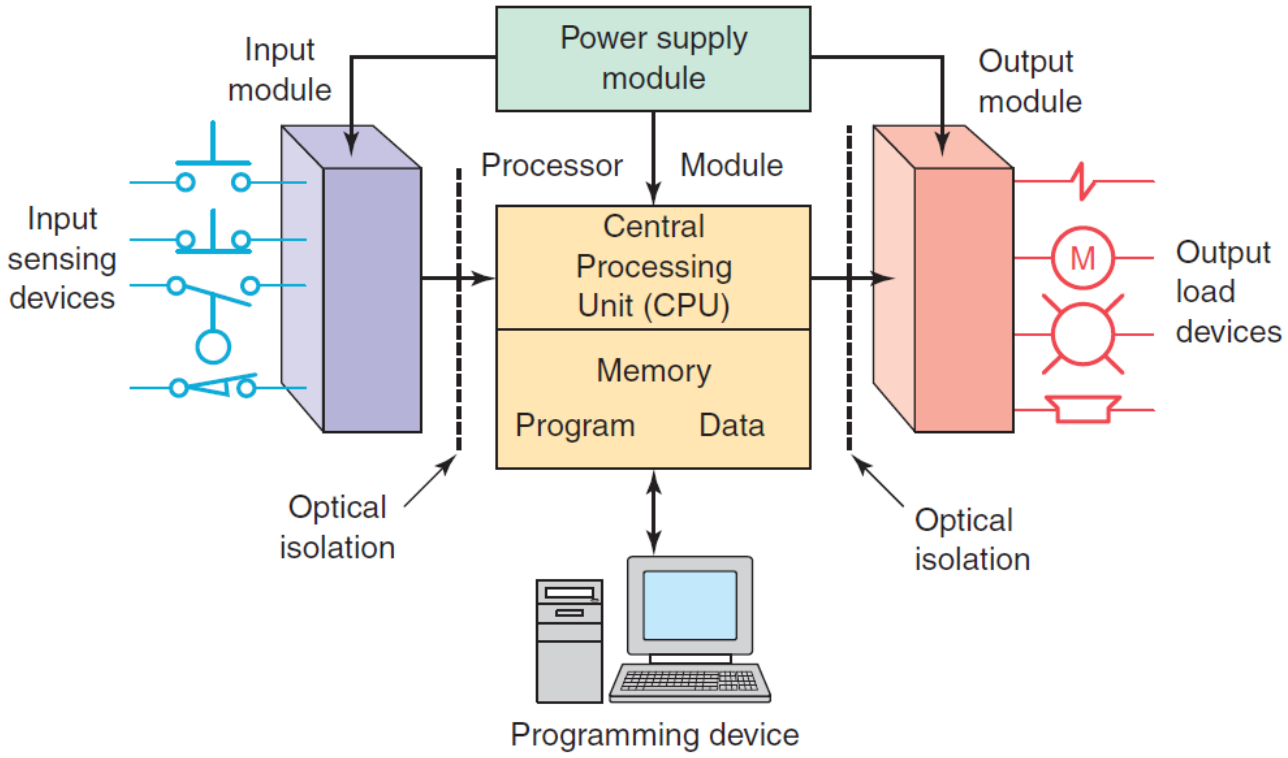
Un Controlador Lógico Programable (PLC) es una computadora de grado industrial capaz de ser programada para realizar diversas funciones de control.

PARTES DE UN PLC

Un PLC típico puede estar dividido en las siguientes partes:

- Unidad central de procesamiento (UCP).
- Sección de entradas y salidas (I/O).
- Fuente de poder.
- Dispositivo de programación.
- Sección de comunicaciones.

PARTES DE UN PLC



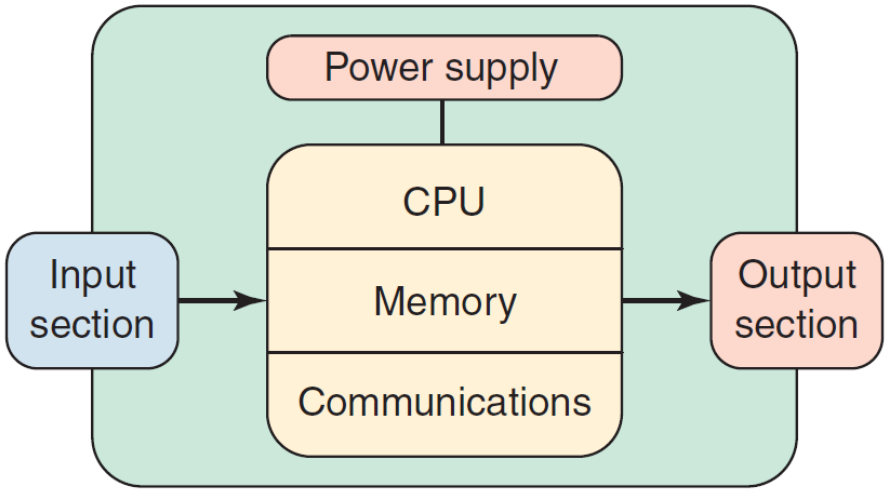


Fig. 6. Típicas partes de un PLC.

PARTES DE UN PLC

Fig. 6. Módulos de procesadores típicos (Rockwell Automation, Inc.).

PARTES DE UN PLC

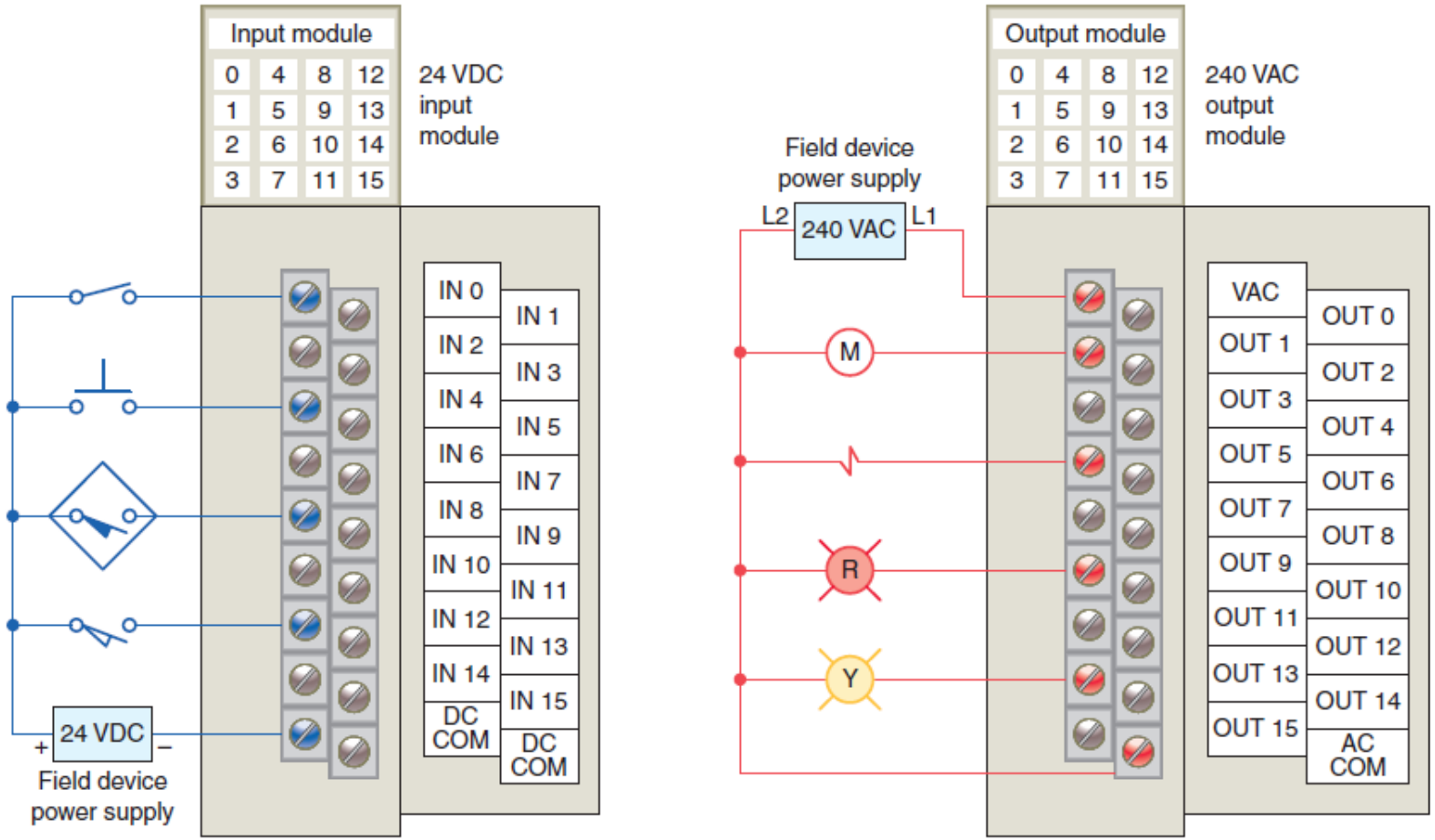


Fig. 7. Sistema de conexión típico de entradas y salidas.

PARTES DE UN PLC

Fig. 8. La fuente de poder alimenta todos los módulos que están conectados en el rack (Schneider Electric).

PARTES DE UN PLC

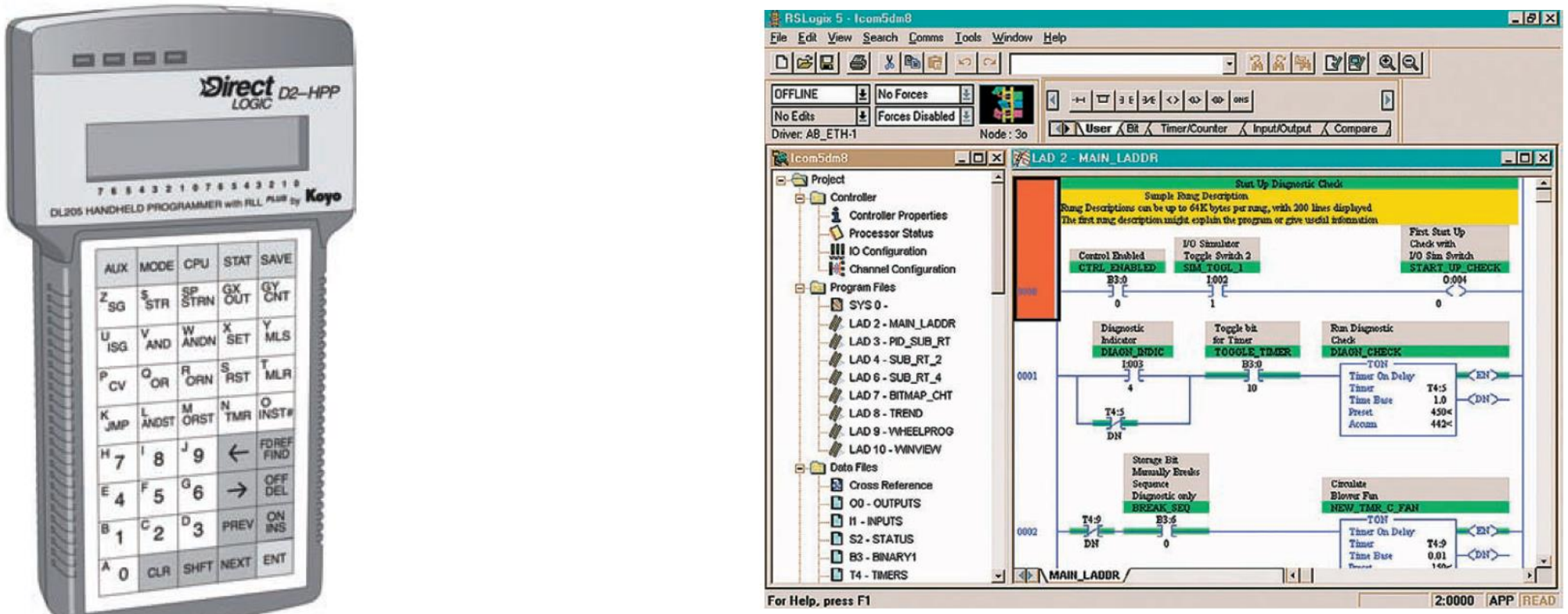


Fig. 9. Dispositivo típico de programación: Hand-held izquierda, PC software a la derecha.

PARTES DE UN PLC

Fig. 10. Modulo típico de comunicaciones (Automation Direct).

ARQUITECTURA DE UN PLC

El termino arquitectura se puede referir al hardware, al software o a una combinación de ambos.

Una arquitectura abierta permite incorporar componentes de otros fabricantes, ya que su diseño es estándar, mientras que una arquitectura cerrada no permite incorporación de componentes de otros fabricantes.

ESTRUCTURA DE UN PLC

Existen dos modos en los cuales las entradas y salidas con incorporadas al PLC:

- Compactas.
- Modular.

ESTRUCTURA DE UN PLC

Existen dos modos en los cuales las entradas y salidas con incorporadas al PLC:

- Compactas.
- Modular.

PLC COMPACTO

Fig. 11. PLC's compactos Rockwell Automation, Inc.

PLC COMPACTO

Fig. 12 PLC modular GE intelligent platforms.

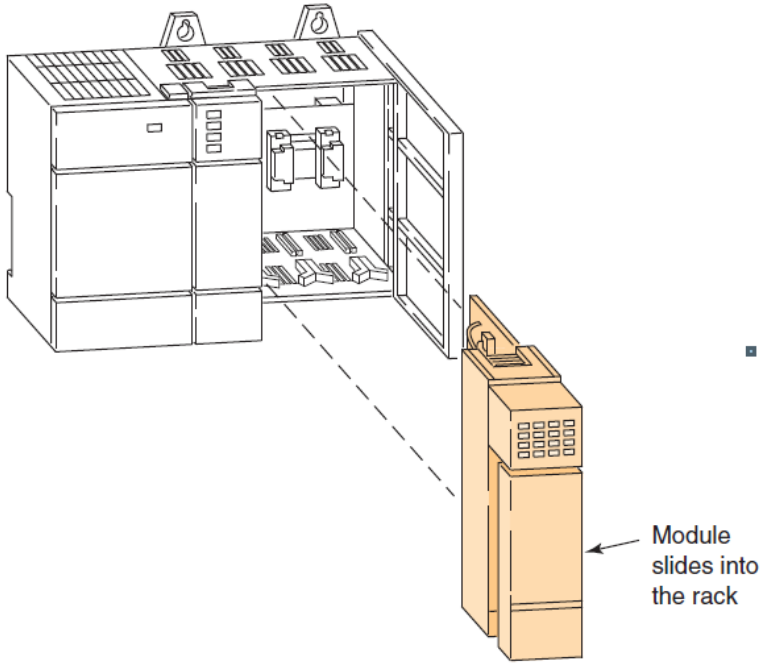


Fig. 13 Configuración modular.

PLC COMPACTO

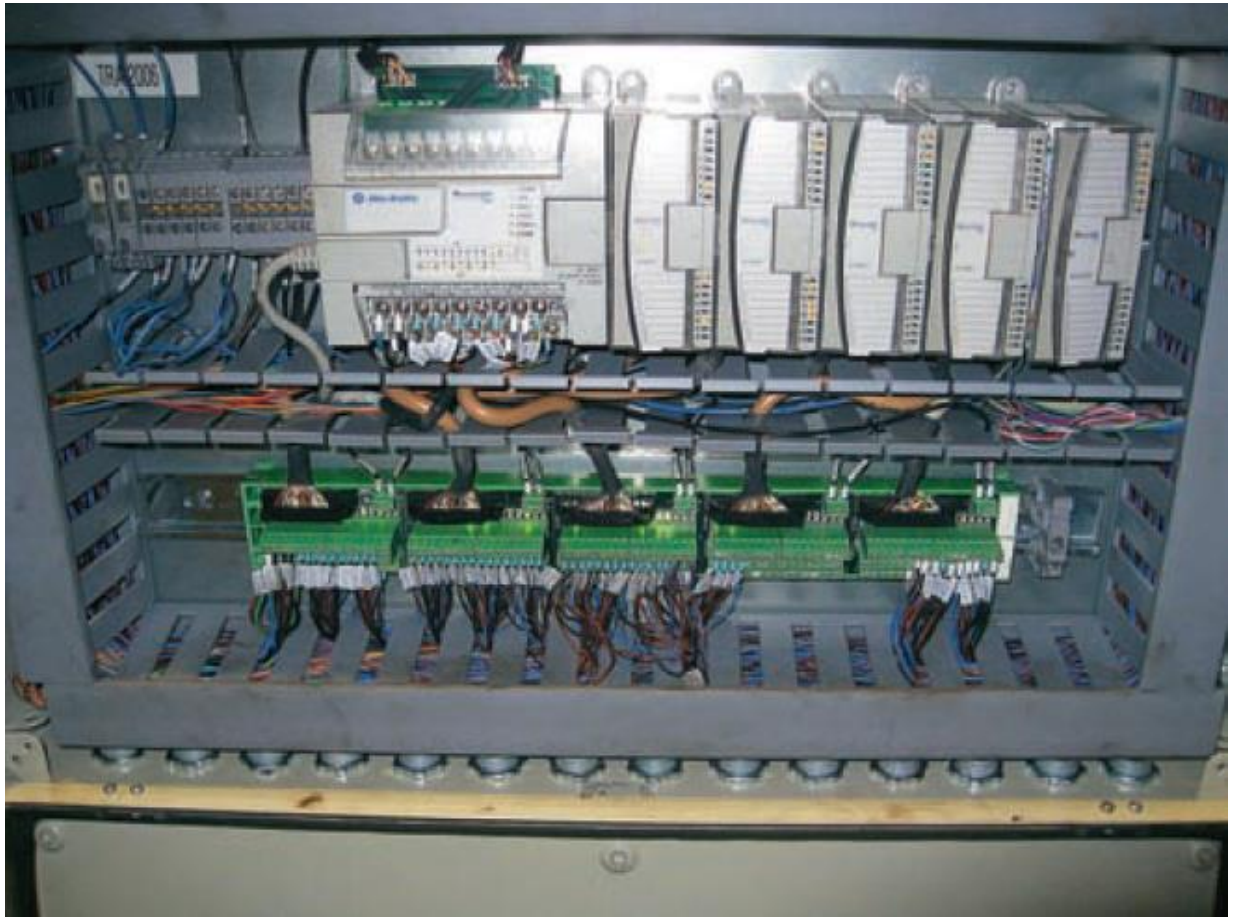


Fig. 14 PLC modular instalado en la industria (Automation IG).

CICLO DE EJECUCIÓN DEL PROGRAMA DEL PLC

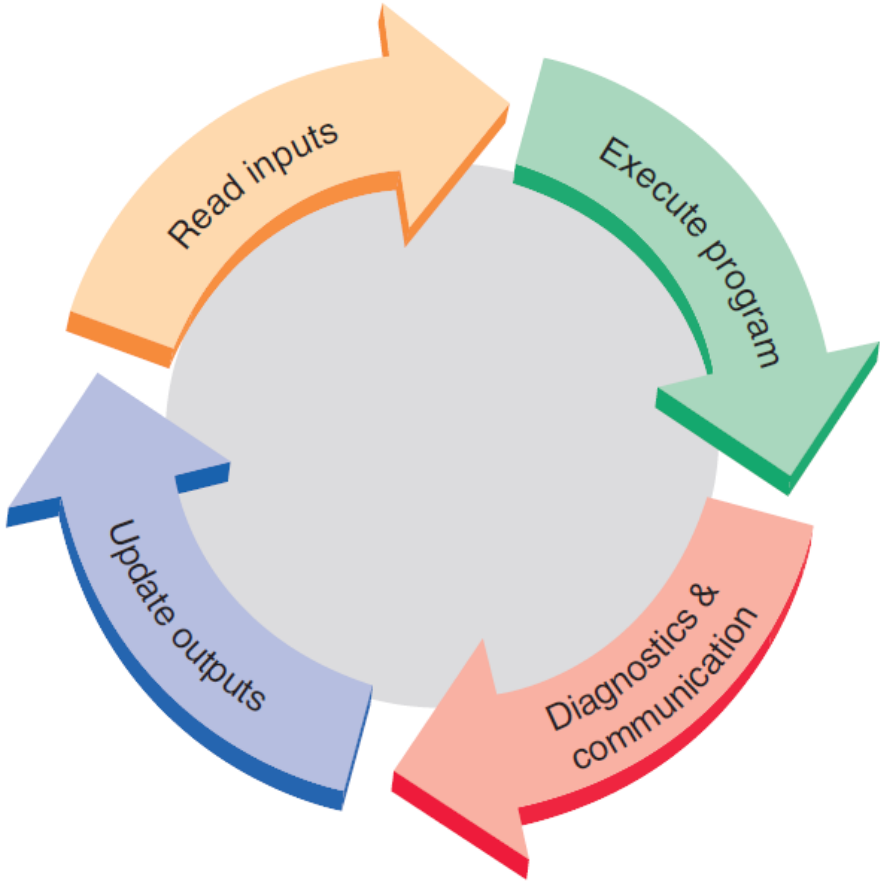


Fig. 15 Ciclo de ejecución del programa del PLC.

PRINCIPIOS DE OPERACIÓN

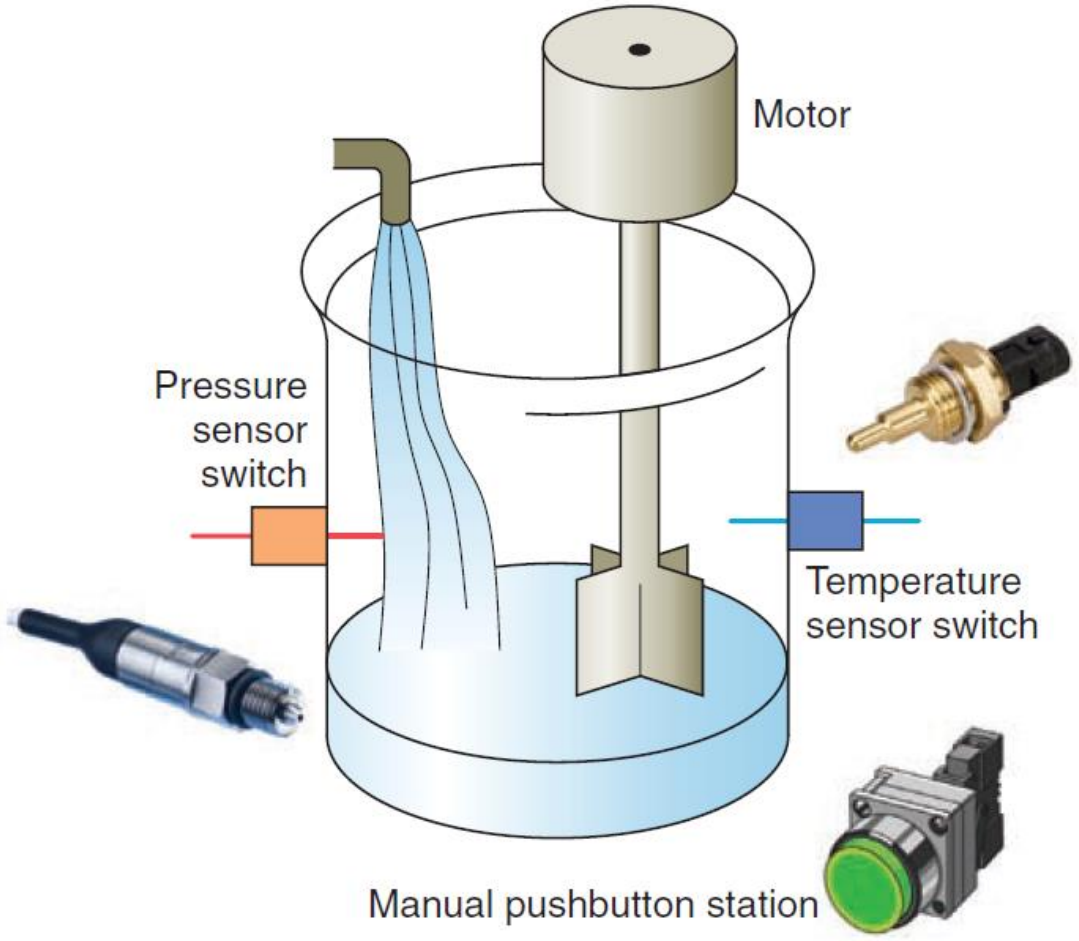


Fig. 16 Control de un proceso de mezclado.

PRINCIPIOS DE OPERACIÓN

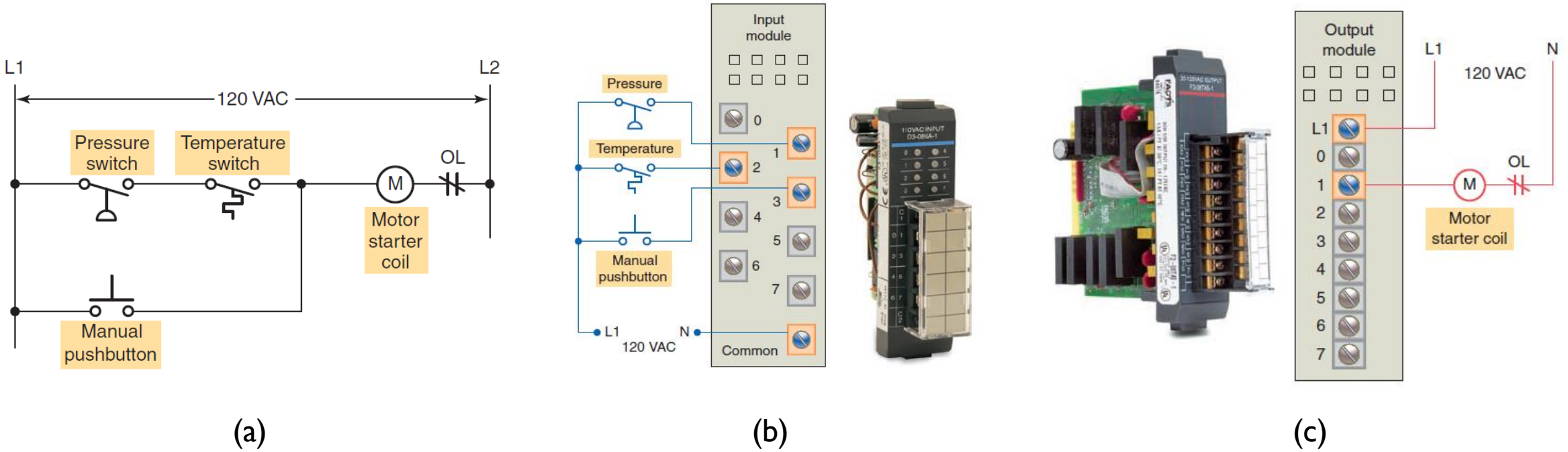
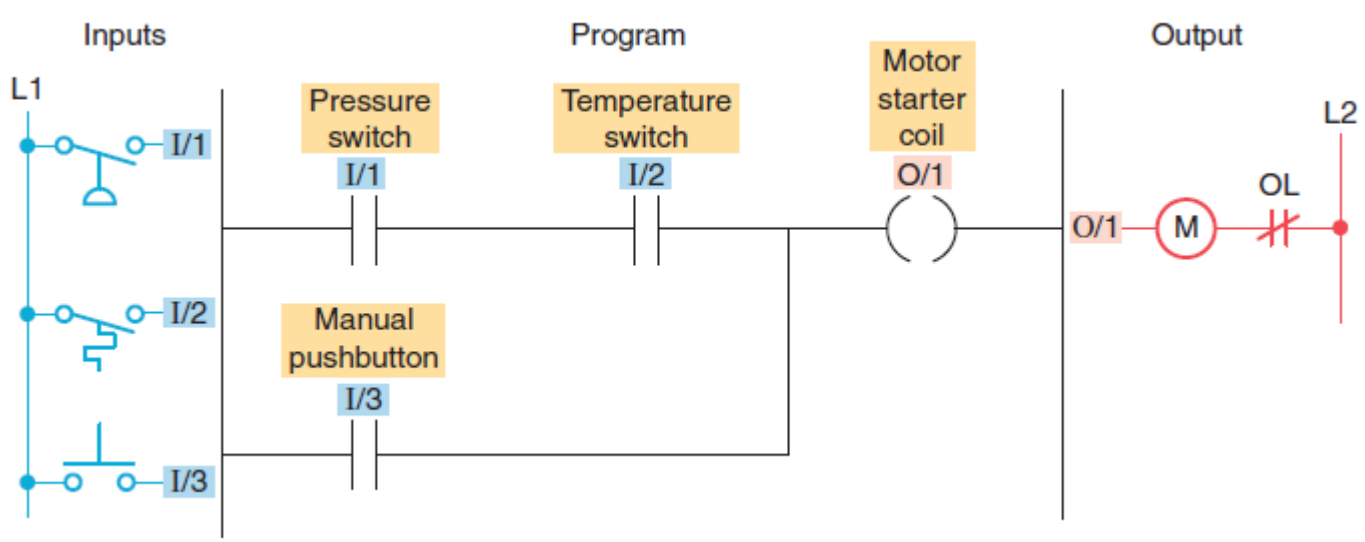
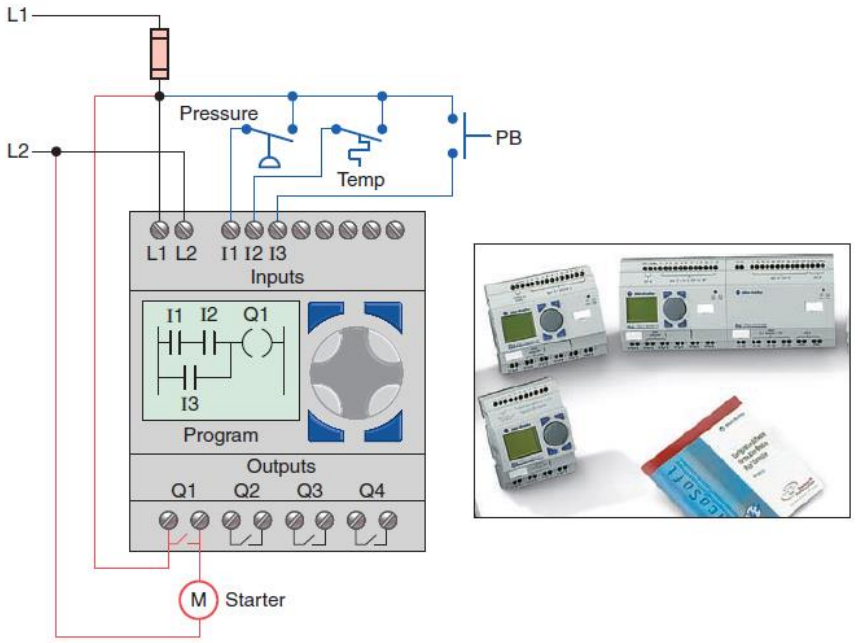


Fig. 17 (a) Control de procesos en diagrama escalera, (b) Conexiones típicas en 120V en corriente alterna de un modulo de entradas, (c) Conexiones típicas en 120V en corriente alterna de un modulo de salidas. (Automation Direct.)

PRINCIPIOS DE OPERACIÓN



(a)



(b)

Fig. 18 (a) Programa en diagrama escalera con configuraciones típicas de direccionamiento, (b) Esquema de control implementado en un PLC compacto. (Rockwell Automation, Inc.)

TAMAÑOS DEL PLC Y APLICACIONES

Para categorizar los PLC se utiliza l funcionalidad, numero de entradas y salidas, costo y tamaño físico. El numero de entradas y salidas es el factor más importante.

TAMAÑOS DEL PLC

- El nano es el más pequeño con menos de 15 I/O.
- Le sigue el micro con un numero de (15 a 125) I/O.
- Medio con (128 a 512) I/O.
- Por último están los Grandes o Large que tiene un numero superior a 512 I/O.

APLICACIONES DEL PLC

- Terminación única o stand alone: involucra un PLC para controlar un proceso.
- Multitarea: un PLC para controlar diversos procesos, puede pertenecer a un subsistema de un gran proceso y se puede comunicar con un PLC central.
- Manejo de control: un PLC controlando otros PLC's, por lo tanto requiere de un gran procesador.

APLICACIONES DEL PLC

Fig. 19 Rangos típicos de tamaños de PLC's. (Siemens)

APLICACIONES DEL PLC

Fig. 20 PLC usado como terminación única (Rogers Machinery Company, Inc.).

APLICACIONES DEL PLC

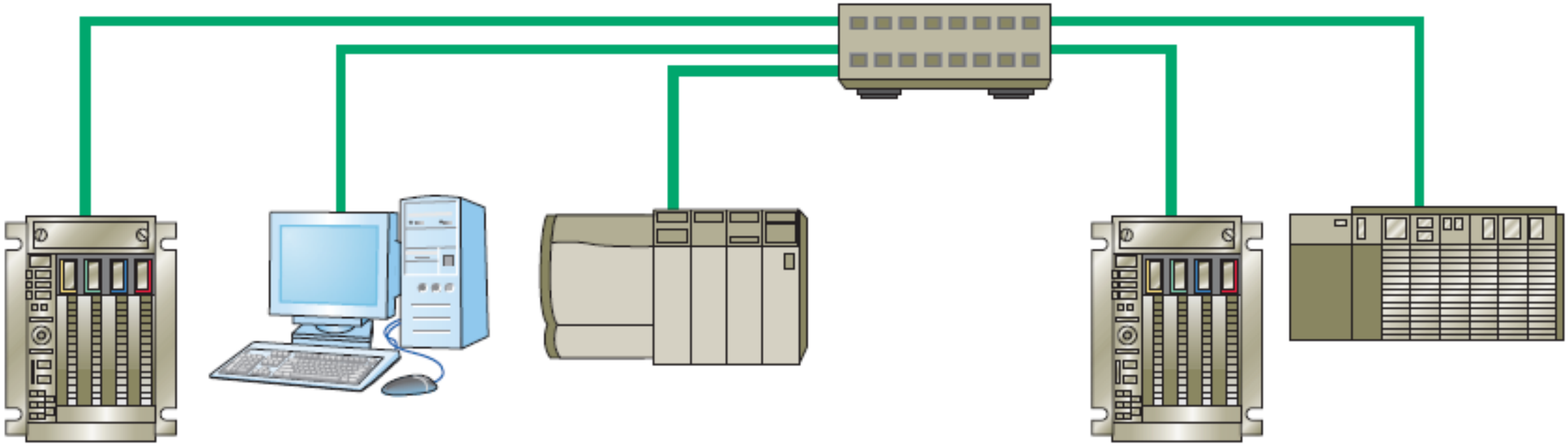


Fig. 21 PLC usado como manejo de control

FACTORES A CONSIDERAR EN LA DECISIÓN DE ESCOGER UN PLC

- Numero de entradas y salidas.
- Tamaño del programa de control.
- Requerimientos de colector de datos.
- Funciones de supervisor.
- Futuras expansiones.

LENGUAJES DE PROGRAMACIÓN ESTÁNDAR IEC 61131-3

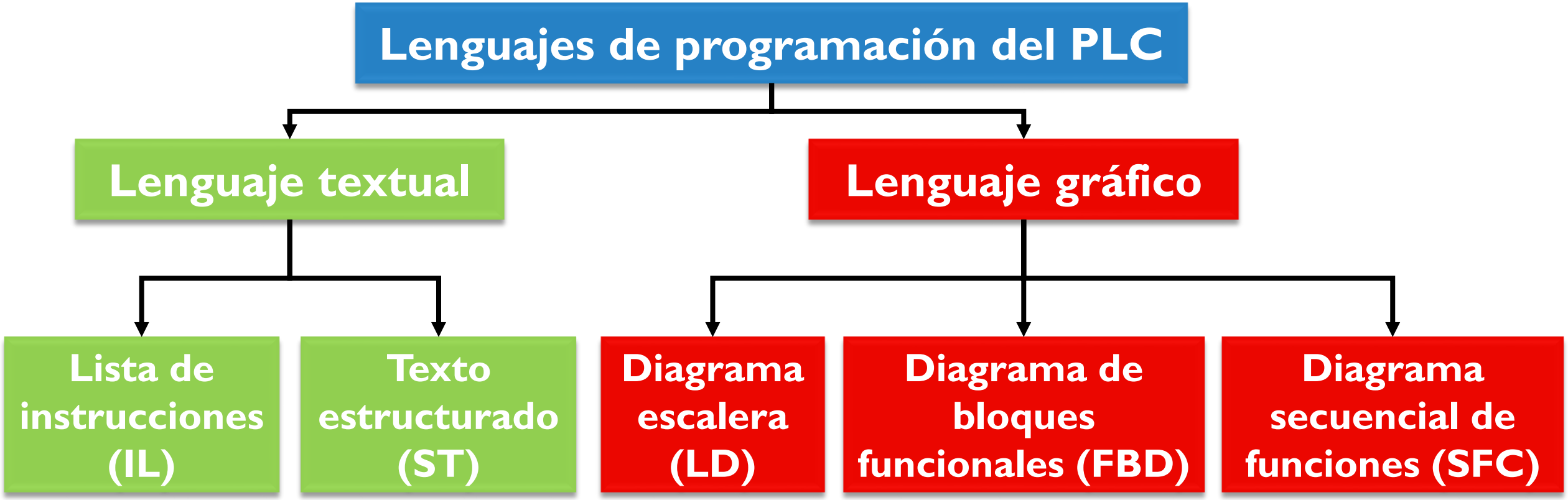


Fig. 22 Lenguajes de programación según el estándar IEC 61131-3

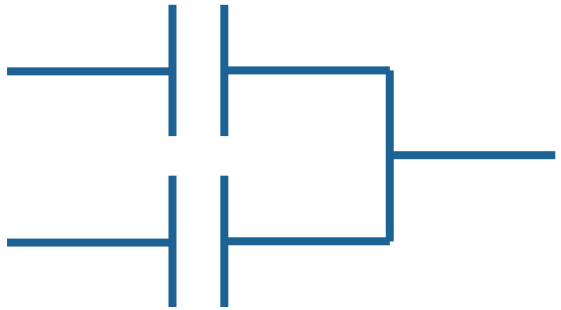
DIAGRAMA ESCALERA

Contacto normalmente abierto

Contacto normalmente cerrado

DIAGRAMA ESCALERA

OR



OR NOT

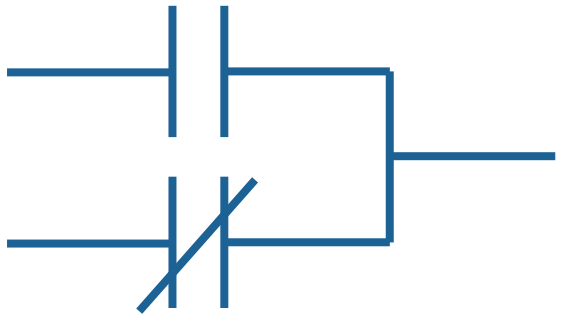


DIAGRAMA ESCALERA

AND

AND NOT

DIAGRAMA ESCALERA

OUT

OUT NOT

DIAGRAMA ESCALERA

SET

RESET

DIAGRAMA ESCALERA

FLAG

BLOQUES
FUNCIONALES

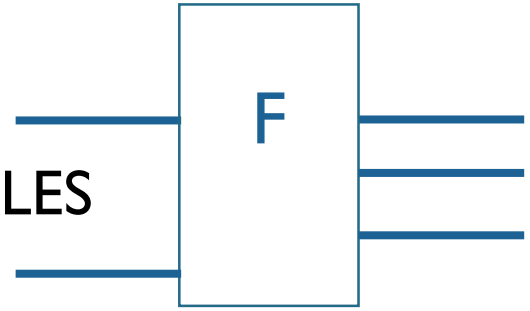


DIAGRAMA ESCALERA

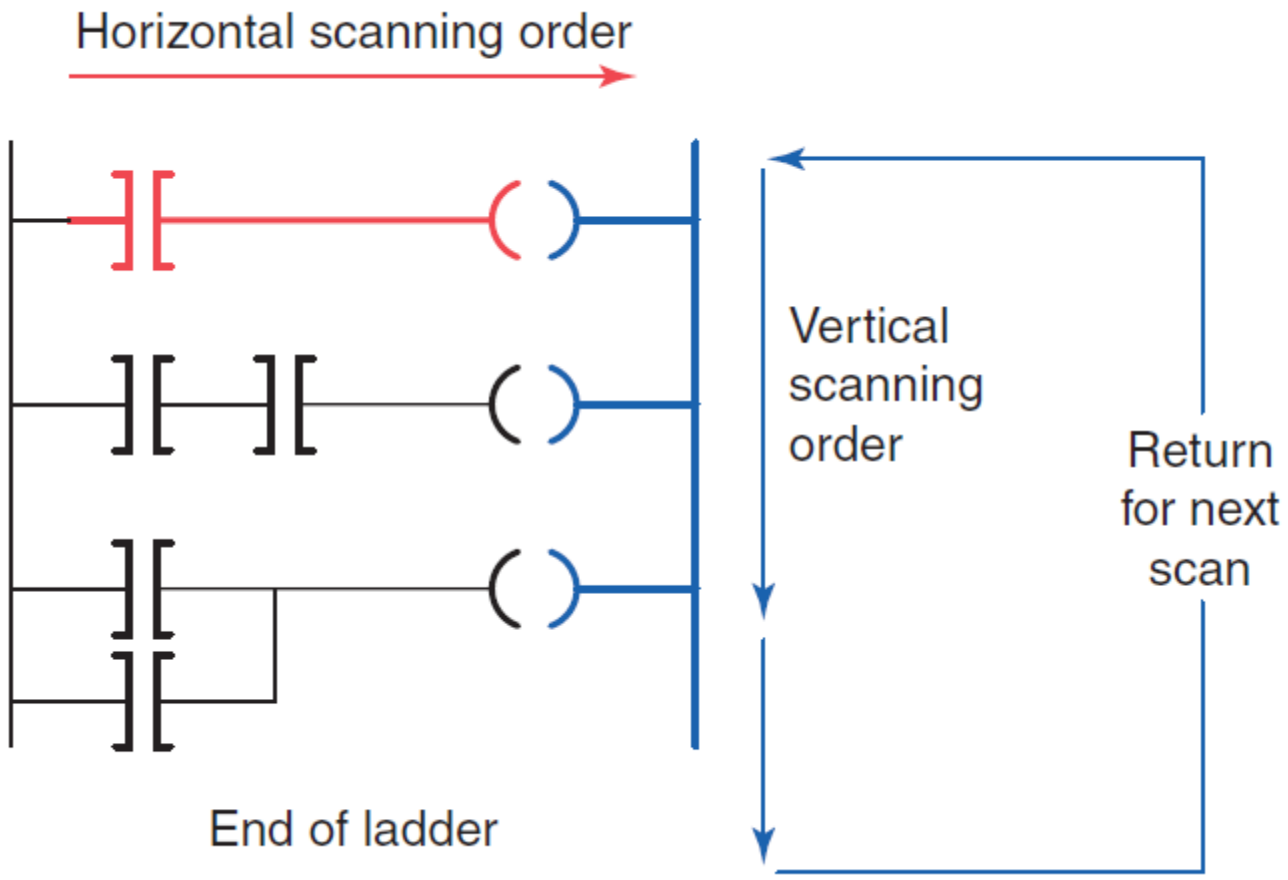


Fig. 23 La ejecución del programa en escalera puede ser tanto en sentido horizontal como en sentido vertical.

DIAGRAMA DE BLOQUES FUNCIONALES (FBD)

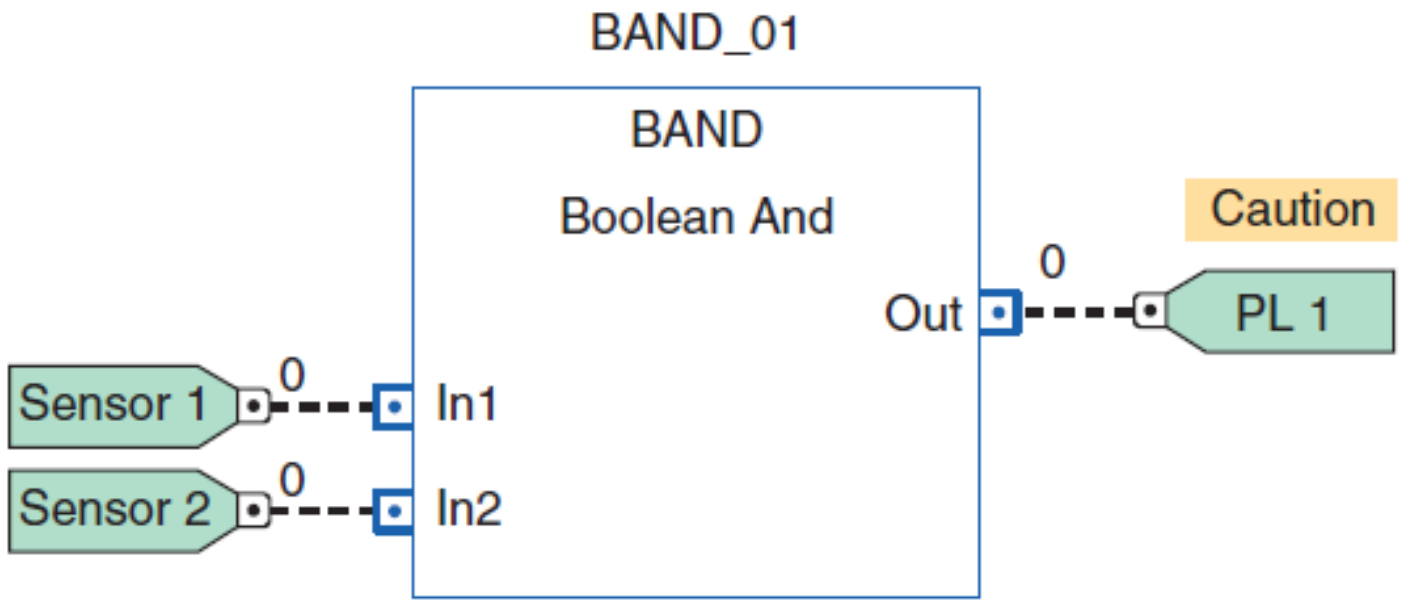


Fig. 24 Ejemplo de compuerta And usando FBD.

DIAGRAMA SECUENCIAL DE FUNCIONES (SFC)

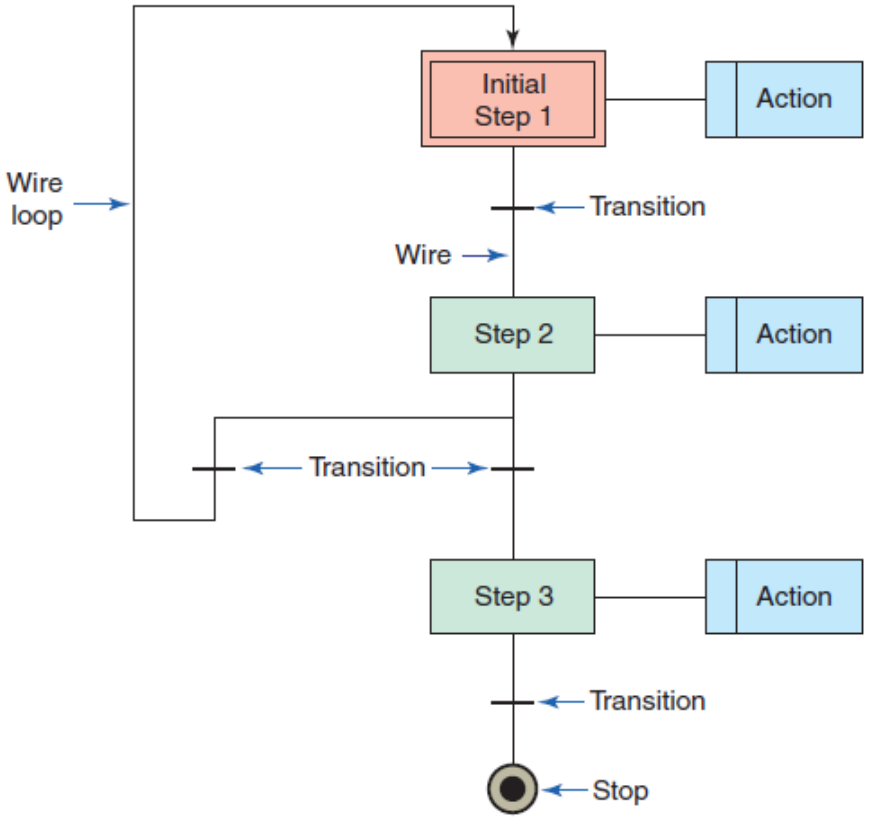
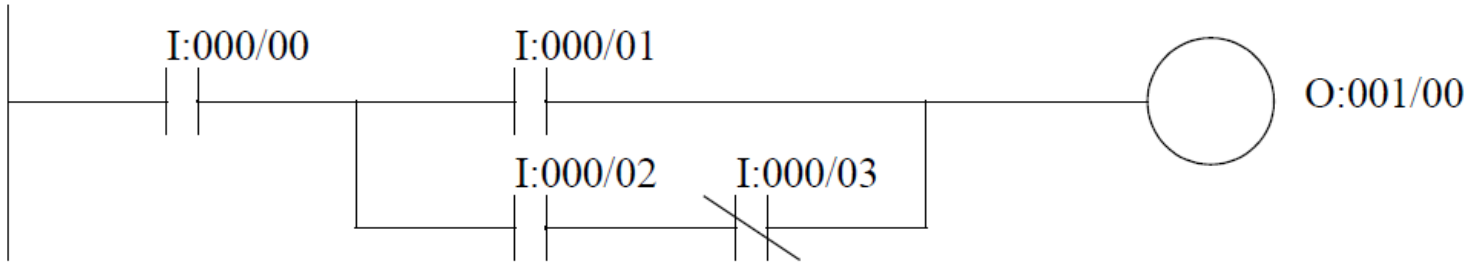


Fig. 25 Ejemplo de como se programa en SFC.

LISTA DE INSTRUCCIONES (IL)



read as O:001/00 = I:000/00 AND (I:000/01 OR (I:000/02 AND NOT I:000/03))

Label	Opcode	Operand	Comment
START:	LD	%I:000/00	(* Load input bit 00 *)
	AND(	%I:000/01	(* Start a branch and load input bit 01 *)
	OR(	%I:000/02	(* Load input bit 02 *)
	ANDN	%I:000/03	(* Load input bit 03 and invert *)
	)		
	)		
	ST	%O:001/00	(* SET the output bit 00 *)

Fig. 26 Ejemplo de programación en IL.

LISTA DE INSTRUCCIONES (IL)

Operator	Modifiers	Data Types	Description
LD	N	many	set current result to value
ST	N	many	store current result to location
S, R		BOOL	set or reset a value (latches or flip-flops)
AND, &	N, (	BOOL	boolean and
OR	N, (	BOOL	boolean or
XOR	N, (	BOOL	boolean exclusive or
ADD	(	many	mathematical add
SUB	(	many	mathematical subtraction
MUL	(	many	mathematical multiplication
DIV	(	many	mathematical division
GT	(	many	comparison greater than >
GE	(	many	comparison greater than or equal >=
EQ	(	many	comparison equals =
NE	(	many	comparison not equal <>
LE	(	many	comparison less than or equals <=
LT	(	many	comparison less than <
JMP	C, N	LABEL	jump to LABEL
CAL	C, N	NAME	call subroutine NAME
RET	C, N		return from subroutine call
)			get value from stack

Fig. 27 Lista de instrucciones.

TEXTO ESTRUCTURADO (ST)

>	greater than
>=	greater than or equal
=	equal
<=	less than or equal
<	less than
<>	not equal

AND(A,B)	logical and
OR(A,B)	logical or
XOR(A,B)	exclusive or
NOT(A)	logical not
!	logical not (note: not implemented on AB controllers)

:=	assigns a value to a variable
+	addition
-	subtraction
/	division
*	multiplication
MOD(A,B)	modulo - this provides the remainder for an integer divide A/B
SQR(A)	square root of A
FRD(A)	from BCD to decimal
TOD(A)	to BCD from decimal
NEG(A)	reverse sign +/-
LN(A)	natural logarithm
LOG(A)	base 10 logarithm
DEG(A)	from radians to degrees
RAD(A)	to radians from degrees
SIN(A)	sine
COS(A)	cosine
TAN(A)	tangent
ASN(A)	arcsine, inverse sine
ACS(A)	arccosine - inverse cosine
ATN(A)	arctan - inverse tangent
XPY(A,B)	A to the power of B
A**B	A to the power of B

Fig. 28 Comandos de texto estructurado

TEXTO ESTRUCTURADO (ST)

```
SBR();
    IF S:FS THEN
        state := 0;
        green_EW.TimerEnable := 1;
        yellow_EW.TimerEnable := 0;
        green_NS.TimerEnable := 0;
        yellow_NS.TimerEnable := 0;
    END_IF;

    TONR(green_EW); TONR(yellow_EW);
    TONR(green_NS); TONR(yellow_NS);

    CASE state OF
    0:  IF green_EW.DN THEN
            state :=1;
            green_EW.TimerEnable := 0;
            yellow_EW.TimerEnable := 1;
        END_IF
    1:  IF yellow_EW.DN THEN
            state :=2;
            yellow_EW.TimerEnable := 0;
            green_NS.TimerEnable := 1;
        END_IF
    2:  IF green_NS.DN THEN
            state :=3;
            green_NS.TimerEnable := 0;
            yellow_NS.TimerEnable := 1;
        END_IF
    3:  IF yellow_NS.DN THEN
            state :=0;
            yellow_NS.TimerEnable := 0;
            green_EW.TimerEnable := 1;
        END_IF

light_EW_green := (state = 0);
light_EW_yellow := (state = 1);
light_EW_red := (state = 2) OR (state = 3);
light_NS_green := (state = 2);
light_NS_yellow := (state = 3);
light_NS_red := (state = 0) OR (state = 1);

RET();
```

Fig. 29 Ejemplo de subrutina de un semáforo en ST.

PROGRAMACIÓN DE TEMPORIZADORES

- On-delay timer

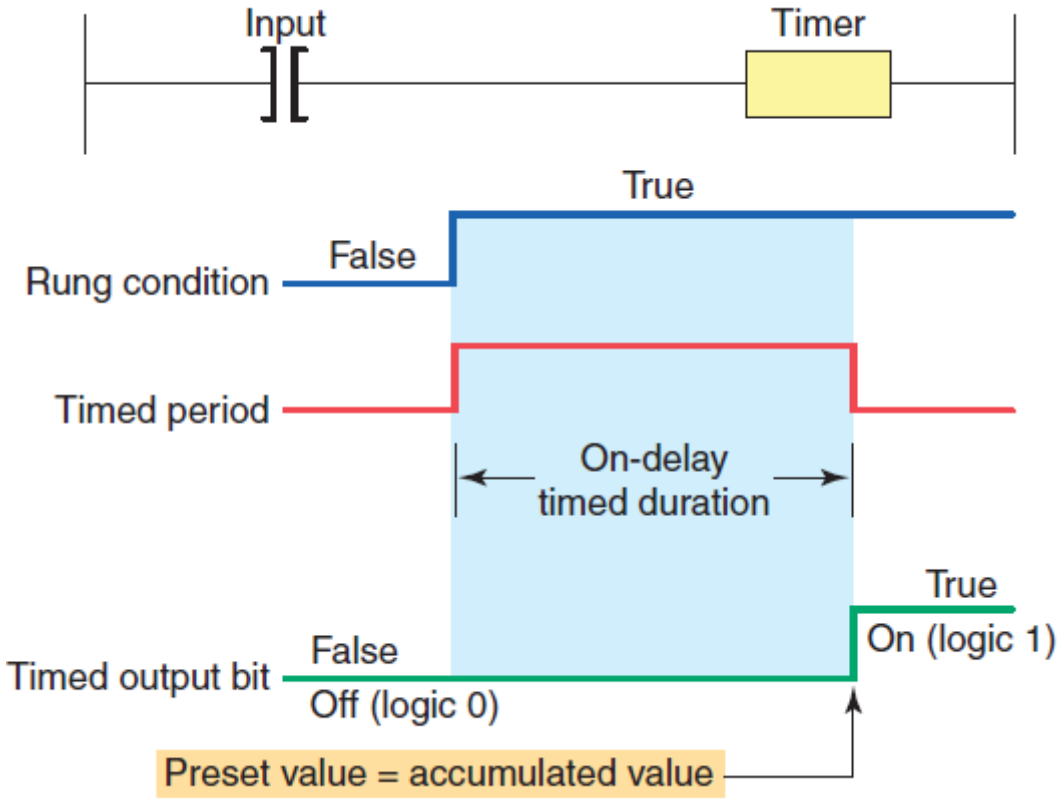


Fig. 30 Principio de operación de un temporizador on-delay.

PROGRAMACIÓN DE TEMPORIZADORES

- Off-delay timer

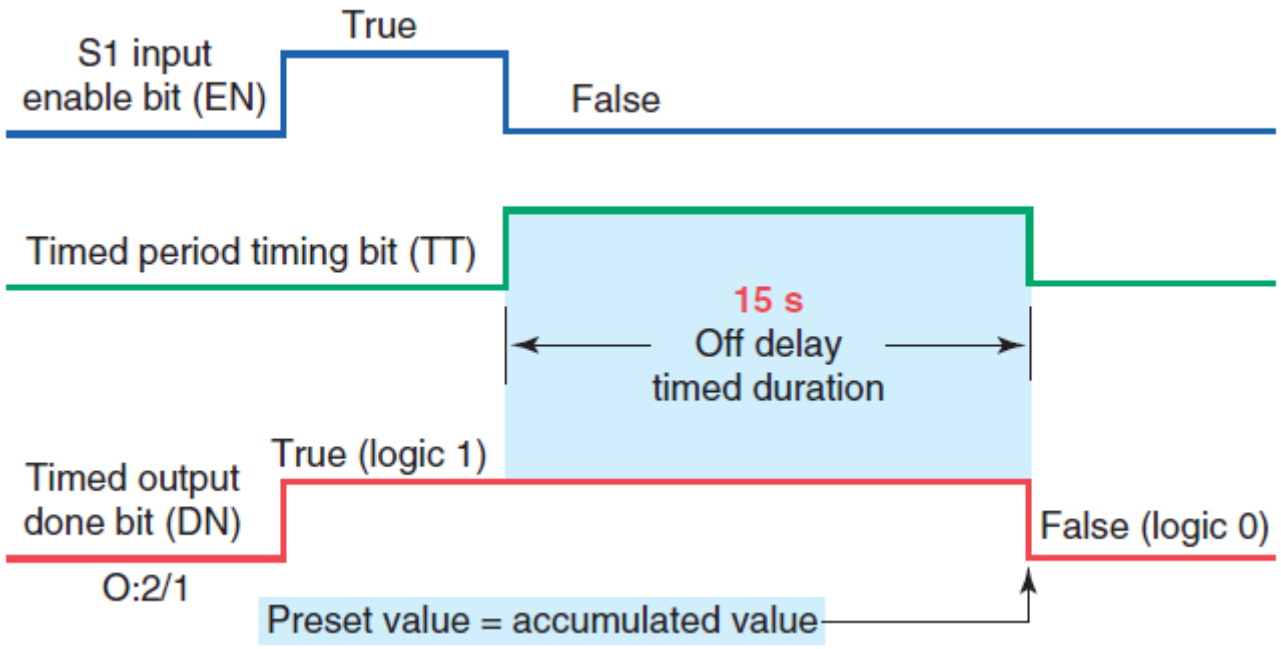


Fig. 31 Principio de operación de un temporizador off-delay.

PROGRAMACIÓN DE TEMPORIZADORES

- Retentive on-delay timer

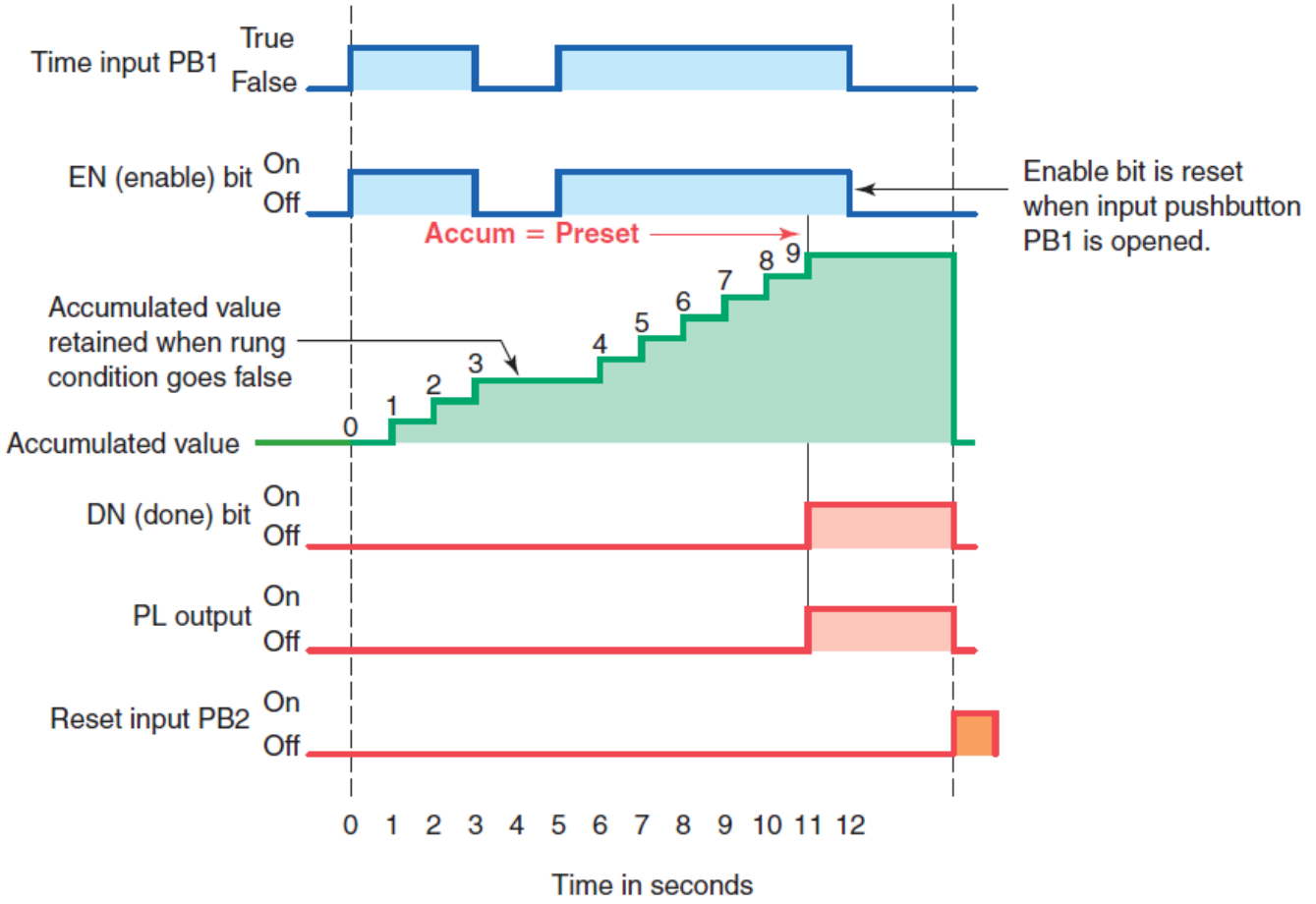


Fig. 32 Principio de operación de un temporizador on-delay de retención.

PROGRAMACIÓN DE CONTADORES

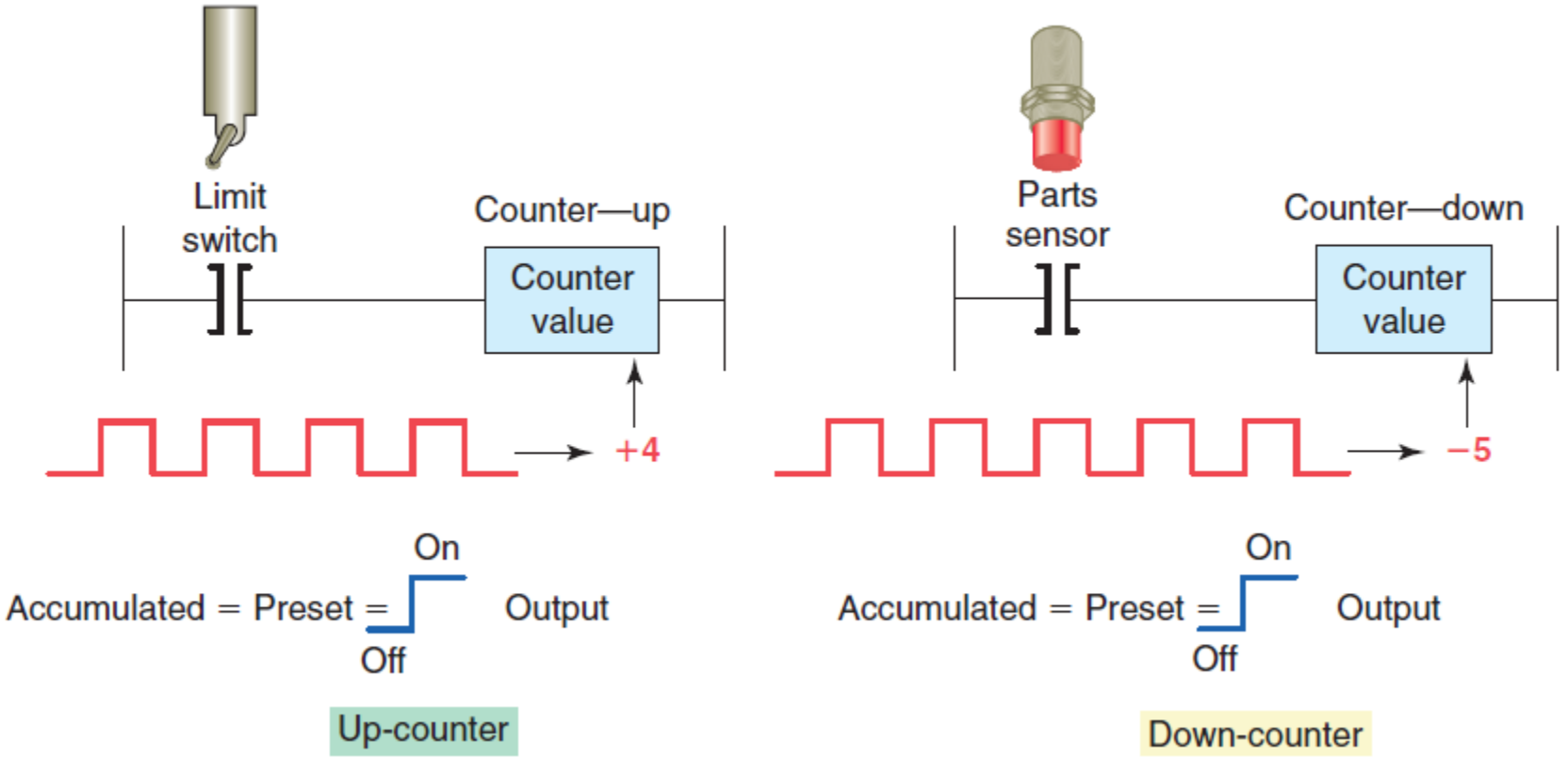


Fig. 33 Uso de contadores.

PROGRAMACIÓN DE CONTADORES

- Up-counter

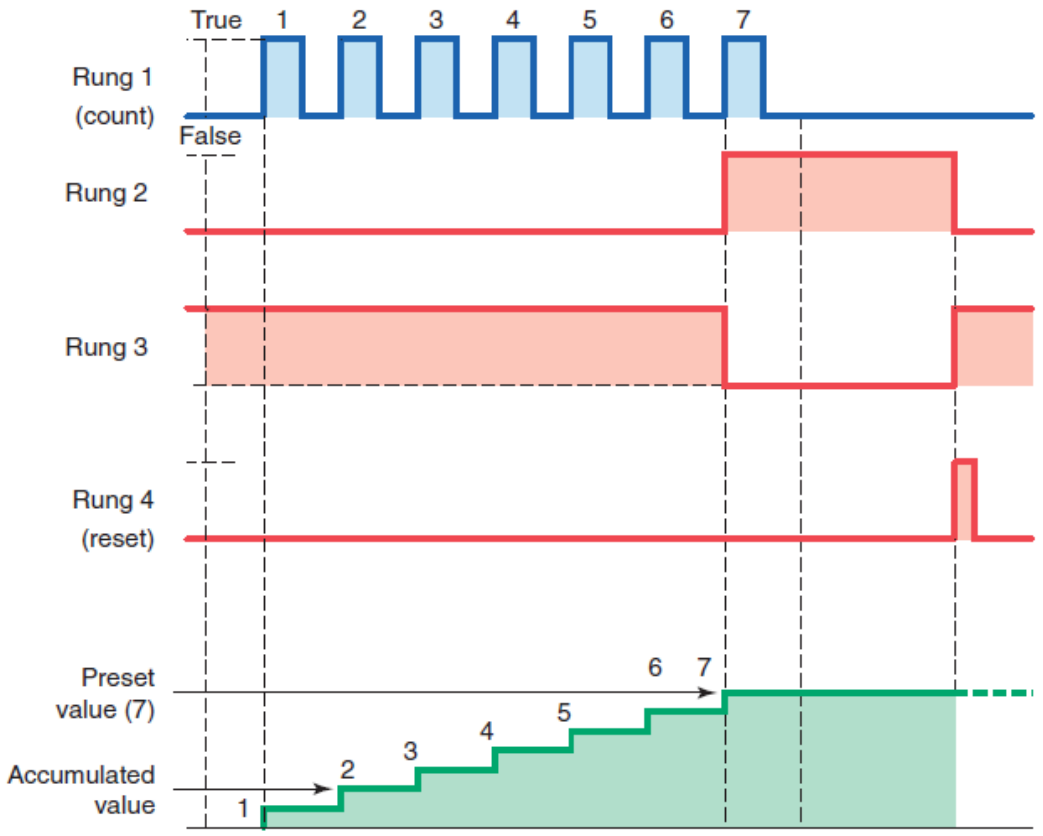


Fig. 34 Principio de operación de un contador up-counter.

PROGRAMACIÓN DE CONTADORES

- Down-counter

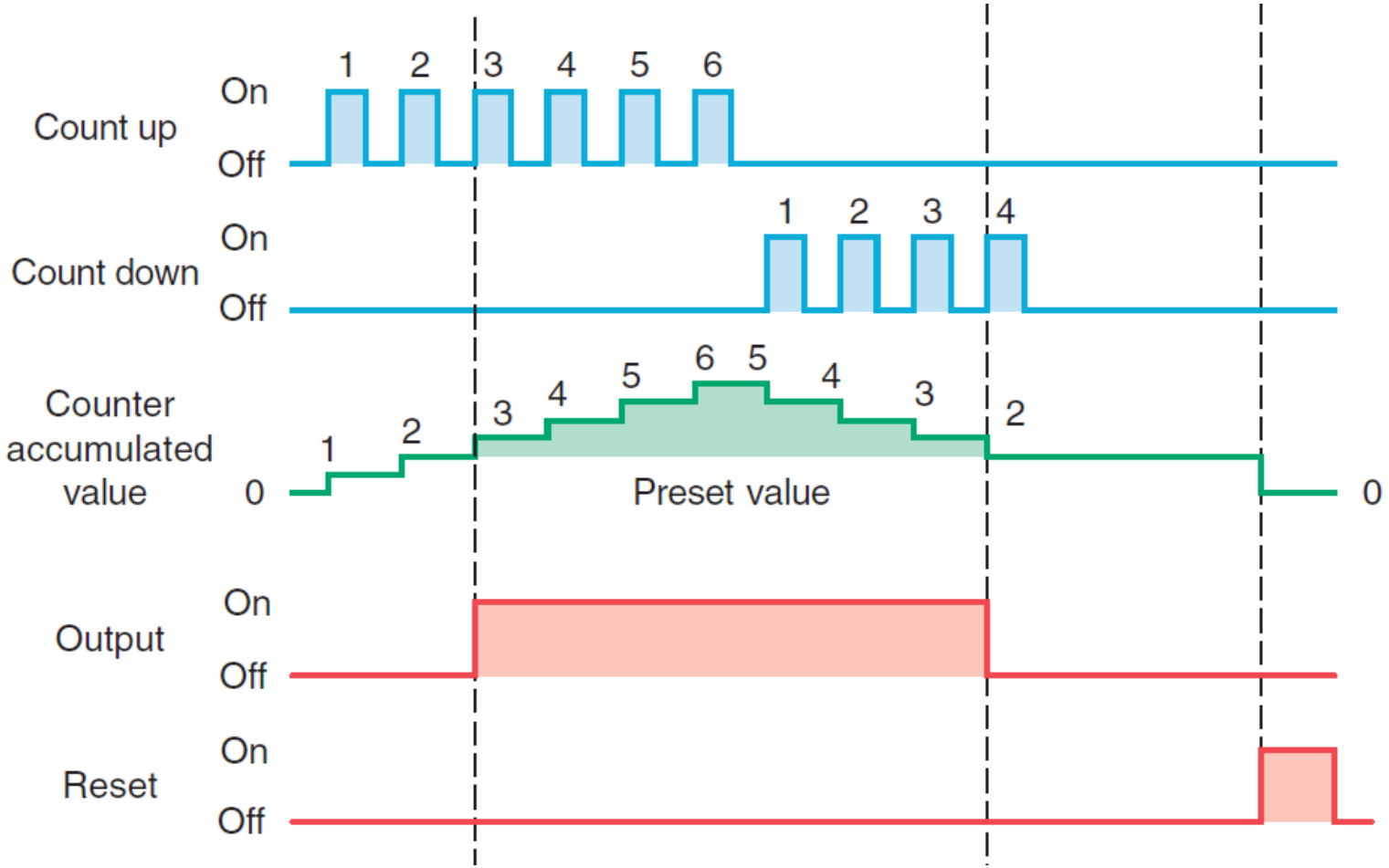


Fig. 35 Principio de operación de un contador down-counter.

DISEÑO DE LÓGICA ESTRUCTURADA

La mayoría de los ingenieros que escriben programas por lo general no se toman el tiempo necesario para realizar el diseño del mismo, por lo tanto la calidad del programa es pobre o regular, por esta razón, para obtener un programa de buena calidad se debe dedicar al menos un 30% del tiempo al diseño estructurado o secuencial, tomando en cuenta que los procesos se ejecutan de manera secuencial y estructurada.

DISEÑO DE LÓGICA ESTRUCTURADA

Técnicas de diseño secuencial

- Diseño simple/pequeño
  - Secuencia de bits (Pasos bastante claros).
  - Diagrama de Flujo (Pasos con algunas desviaciones).

DISEÑO DE LÓGICA ESTRUCTURADA

Técnicas de diseño secuencial

- Diseño complejo/largo
  - Proceso único.
    - Diagramas de estado.
      - Lógica de bloques (Tiempo de desarrollo corto).
      - Ecuaciones (Rendimiento es importante).
  - Múltiples procesos.
    - Redes de Petri (Con estados paralelos).
    - SFC/Grafset (Con estados únicos).

DISEÑO DE LÓGICA ESTRUCTURADA

Secuencia de Bits

Una máquina típica usa una secuencia de pasos repetitivos que pueden ser claramente identificados. La lógica escalera se puede escribir siguiendo esta secuencia. Los pasos para el diseño utilizando este método son:

1. Entender el proceso.
2. Escribir los pasos de la operación en secuencia y darle un número a cada paso.
3. Para cada paso asignar un bit.

DISEÑO DE LÓGICA ESTRUCTURADA

4. Escribir la lógica escalera para convertir los bits en on/off como el proceso se mueve a través de los estados.
5. Escribir la lógica escalera para ejecutar las funciones de la máquina para cada paso.
6. Si el proceso es repetitivo haga que el último paso vuelva al primero.

DISEÑO DE LÓGICA ESTRUCTURADA

Ejemplo:

- **Descripción:** Un izador de bandera que la levanta cuando el botón up es presionado y la baja cuando el botón down es presionado, ambos botones son momentáneos. Existen limit switches en el tope y en la base del poste. Cuando se enciende por primera vez, la bandera debe bajar hasta la base del poste.

DISEÑO DE LÓGICA ESTRUCTURADA

Pasos:

1. **La bandera se mueve hacia abajo esperando la señal del limit switch de la base.**
2. **La bandera se encuentra en la base del poste esperando por el botón up.**
3. **La bandera se mueve hacia arriba, esperando el limit switch del tope.**
4. **La bandera se encuentra en el tope esperando por el botón down.**

DISEÑO DE LÓGICA ESTRUCTURADA

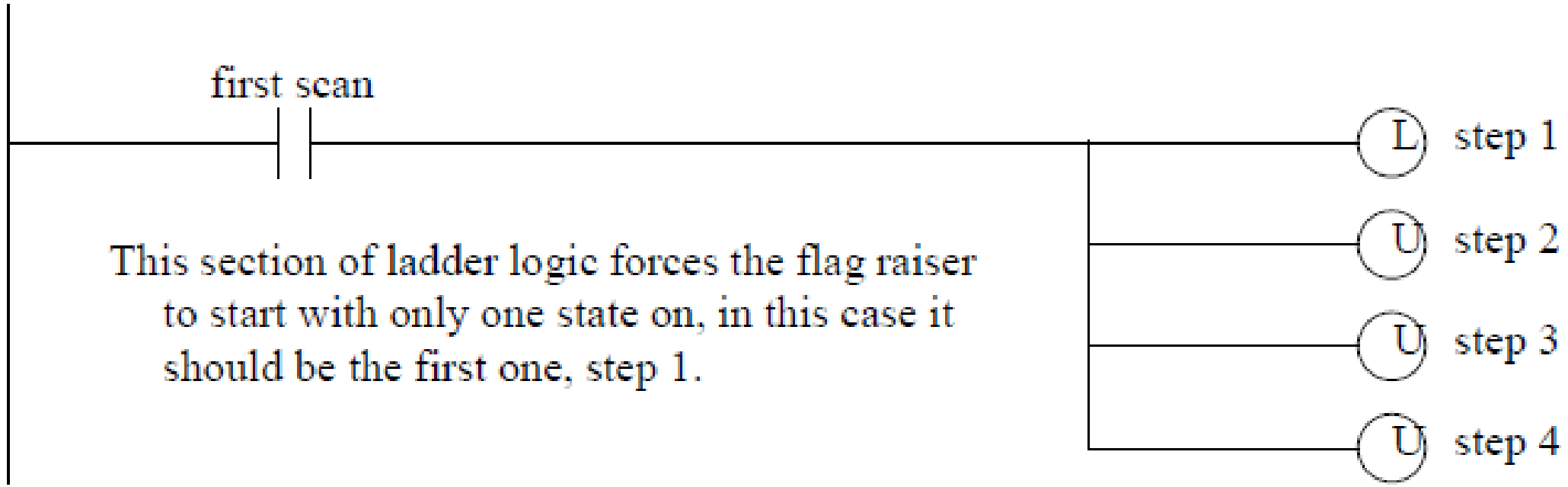


Fig. 36 Ejemplo de proceso diseñado con el método de secuencia de bits.

DISEÑO DE LÓGICA ESTRUCTURADA

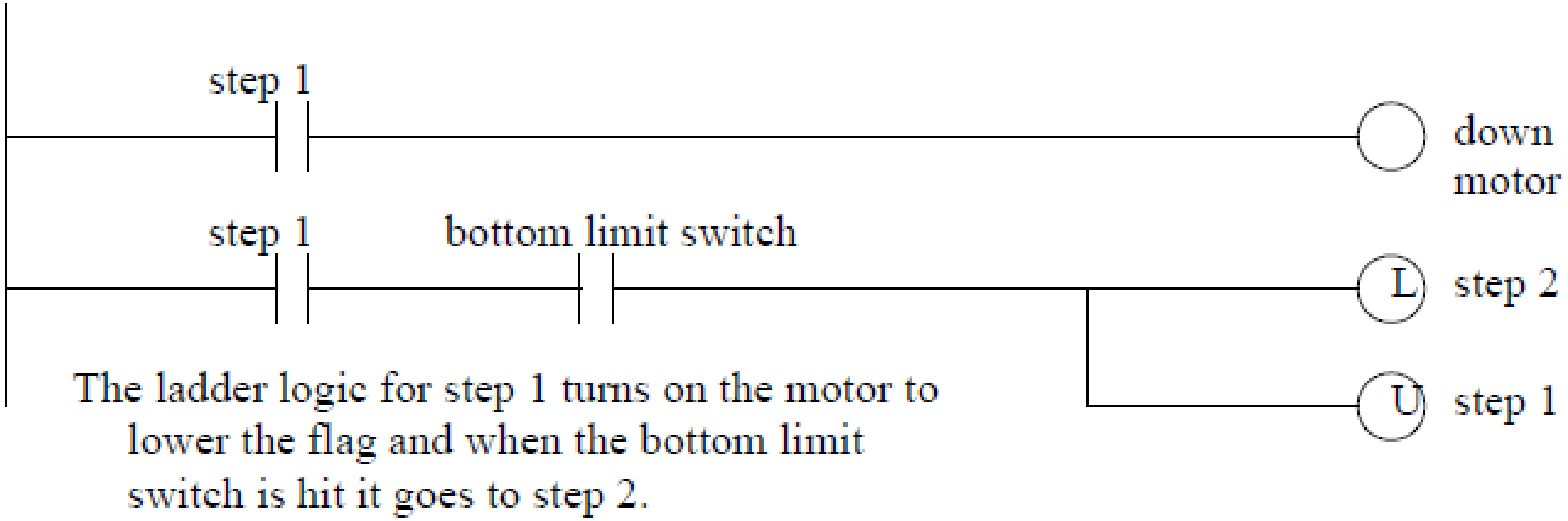


Fig. 37 Ejemplo de proceso diseñado con el método de secuencia de bits (cont.).

DISEÑO DE LÓGICA ESTRUCTURADA

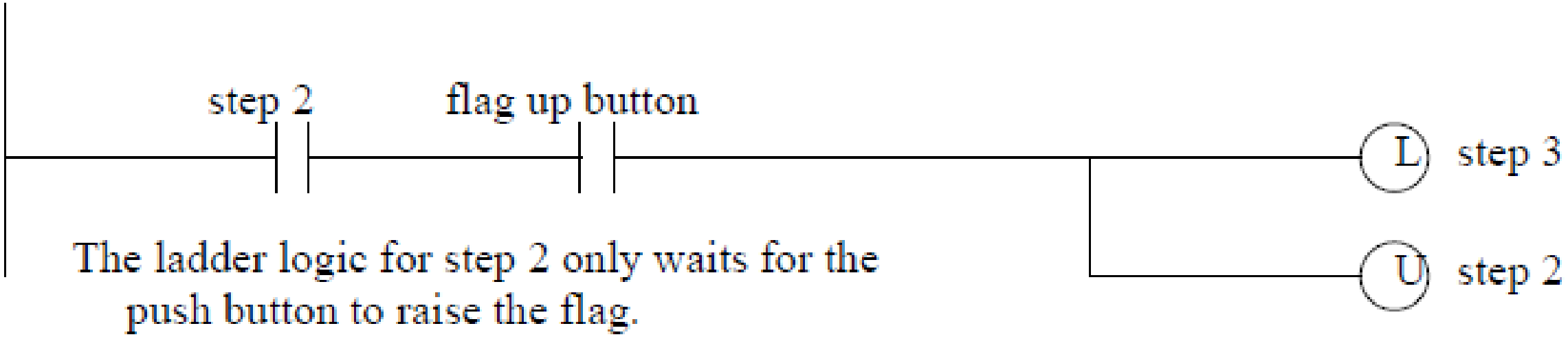


Fig. 38 Ejemplo de proceso diseñado con el método de secuencia de bits (cont.).

DISEÑO DE LÓGICA ESTRUCTURADA

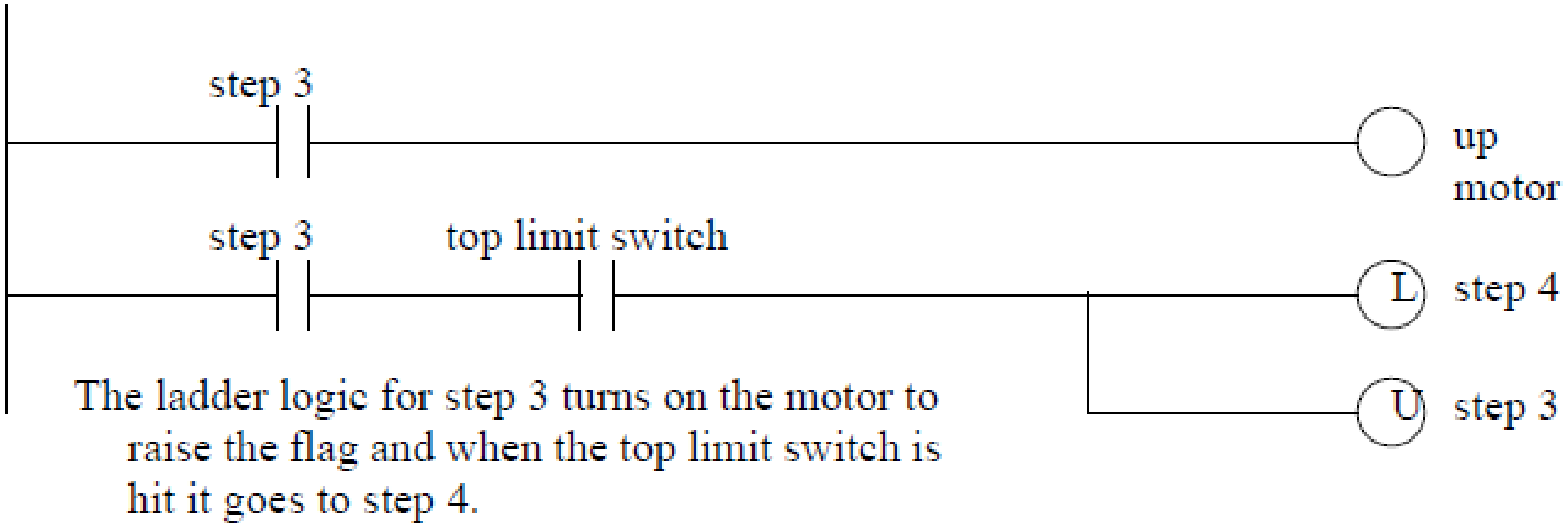


Fig. 39 Ejemplo de proceso diseñado con el método de secuencia de bits (cont.).

DISEÑO DE LÓGICA ESTRUCTURADA

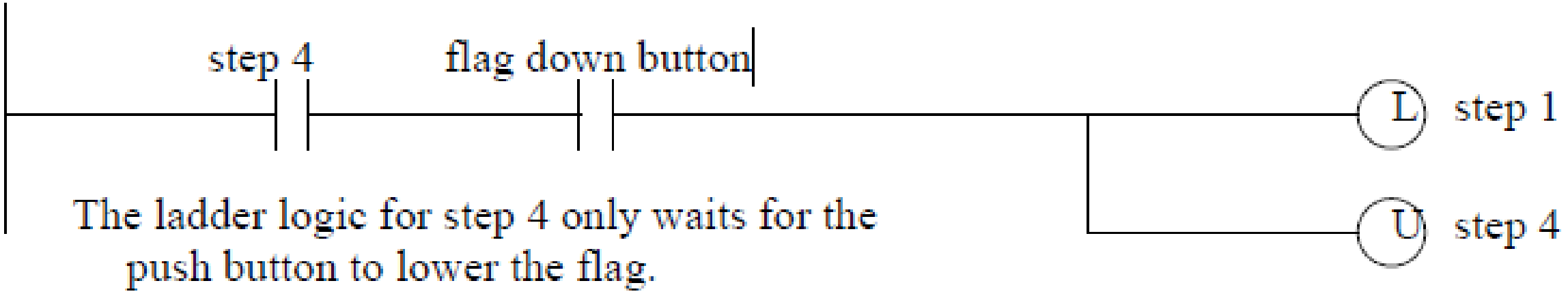


Fig. 40 Ejemplo de proceso diseñado con el método de secuencia de bits (cont.).

DISEÑO DE LÓGICA ESTRUCTURADA

Diseño basado en diagrama de flujo

El diagrama de flujo es ideal para aquellos procesos que presentan pasos secuenciales. Es descrito por bloques, los cuales están unidos por flechas las cuales indican la secuencia de los pasos.

En la siguiente figura se muestran los bloques usados en el diagrama de flujo.

DISEÑO DE LÓGICA ESTRUCTURADA

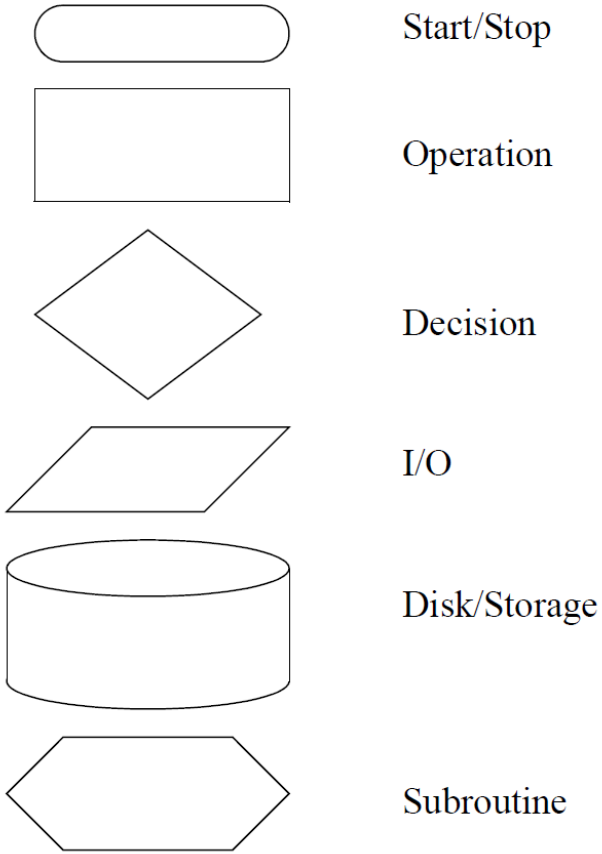


Fig. 41 Símbolos del diagrama de flujo.

DISEÑO DE LÓGICA ESTRUCTURADA

Método general para construir diagramas de flujo

1. Entender el proceso.
2. Determinar las principales acciones, ellas se dibujaran en bloques.
3. Determinar las secuencias de operación, ellas serán dibujadas con flechas.
4. Cuando las secuencias puedan cambiar se debe usar bloques de decisión.

Una vez que esta hecho el diagrama de flujo se puede escribir la lógica escalera.

DISEÑO DE LÓGICA ESTRUCTURADA

Ejemplo:

- **Descripción:** Diseñe el diagrama de flujo y la lógica escalera para un tanque de agua, cuando se inicia el proceso se abre la válvula de vaciado y se cierra la válvula de llenado, de cuando se presiona el botón de encendido se abre la válvula de llenado y el flujo de salida debe detenerse, cuando este lleno el tanque o se presiona el botón de parada, se cierra la válvula de llenado y se abre la válvula de salida. Tomar en cuenta que el botón de arranque es normalmente abierto y el botón de parada es normalmente cerrado.

DISEÑO DE LÓGICA ESTRUCTURADA

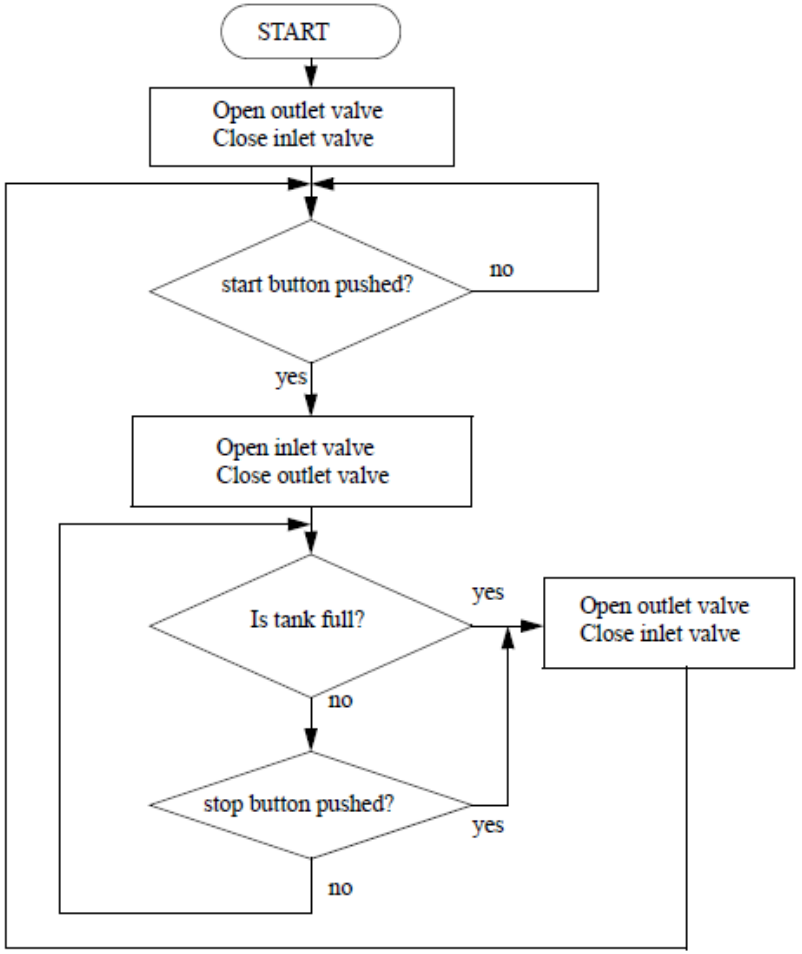


Fig. 42 Diagrama de Flujo del proceso.

DISEÑO DE LÓGICA ESTRUCTURADA

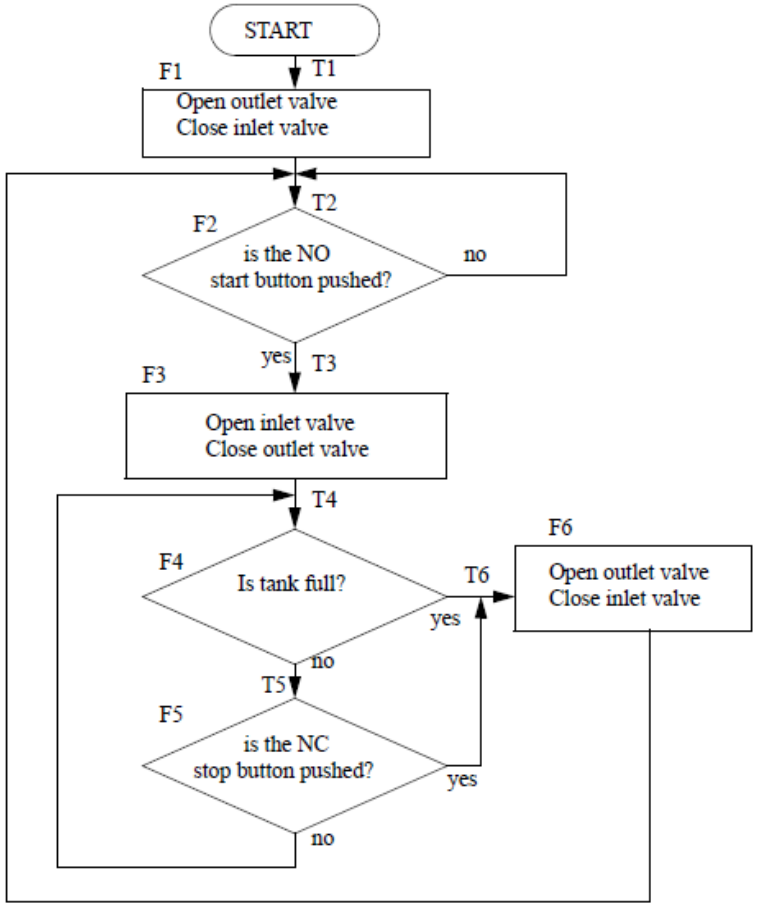


Fig. 43 Diagrama de Flujo del proceso utilizando secuencia de bits.

DISEÑO DE LÓGICA ESTRUCTURADA

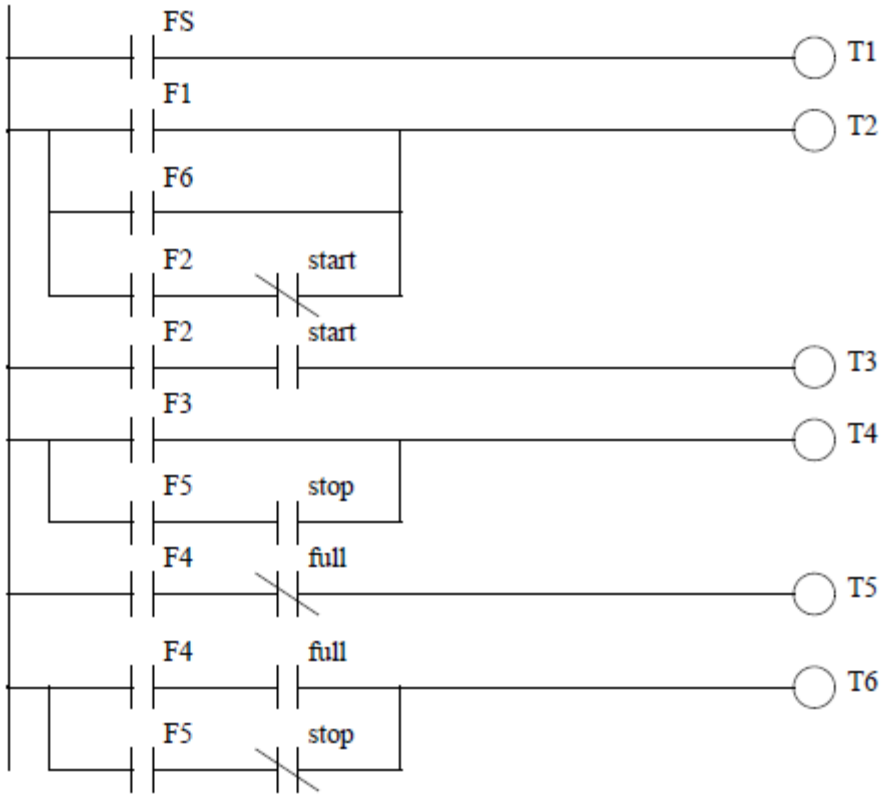


Fig. 44 Lógica escalera utilizando secuencia de bits.

DISEÑO DE LÓGICA ESTRUCTURADA

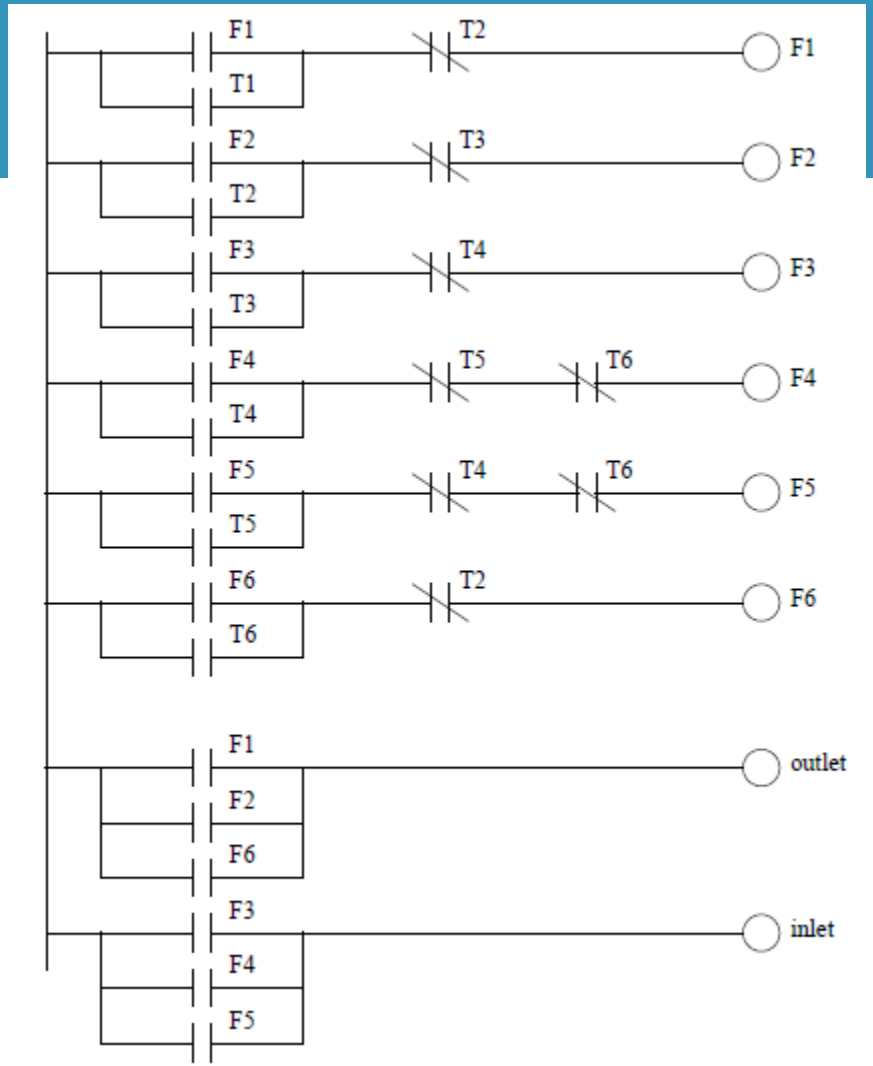


Fig. 45 Lógica escalera utilizando secuencia de bits (cont).

BASE BIBLIOGRÁFICA PARA ESTA PRESENTACIÓN

Para desarrollar la estructura y contenido de esta presentación, así como las imágenes aquí proporcionadas, se utilizó la siguiente bibliografía.

[1]	**Frank D. Petruzella, Programmable Logic Controllers, 4th ed., Career Education, 2010.**
[2]	Hugh Jack, Automatic Manufacturing Systems with PLCs, 2007.