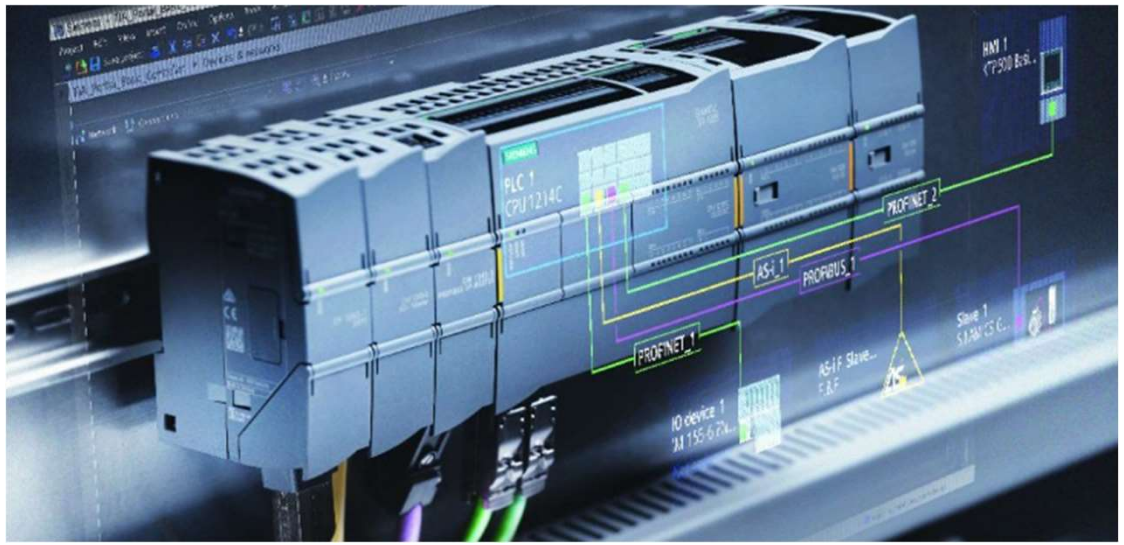
PROGRAMACIÓN SIEMENS



1- Ciclo scan

Los PLC presentan un funcionamiento cíclico en el que se diferencian 4 partes:

0- Test Inicial – Se realiza un test o chequeo interno. Esto sólo se realiza la primera vez que se enciende el PLC

1- Test: Test en cada ciclo, comprueba errores o tiempo máximo de ciclo. Test de hardware, *watchdog* (normalmente 1s, tiempo máximo de ciclo) , errores, etc

2- Lectura de las entradas. Se lee el estado en el que se encuentras las entradas (por ejemplo si un pulsador está a ON), un final de carrera, etc.

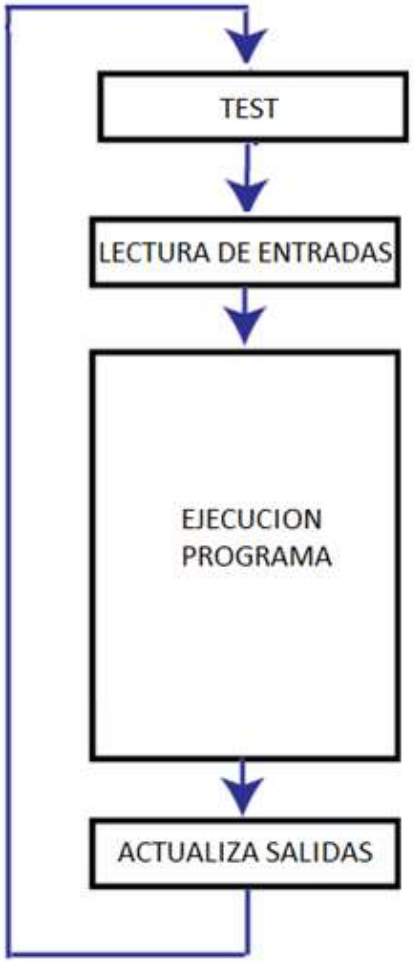
3- Ejecución del programa. El programa se ejecuta. El tiempo de ejecución es muy variable en dependencia de la longitud del mismo o del tipo de plc, aunque con las tecnologías actuales suele ser del orden de milisegundos. En este proceso se memoriza en que estado deben estar las salidas, **PERO TODAVIA NO SE ACTUA SOBRE ELLAS**

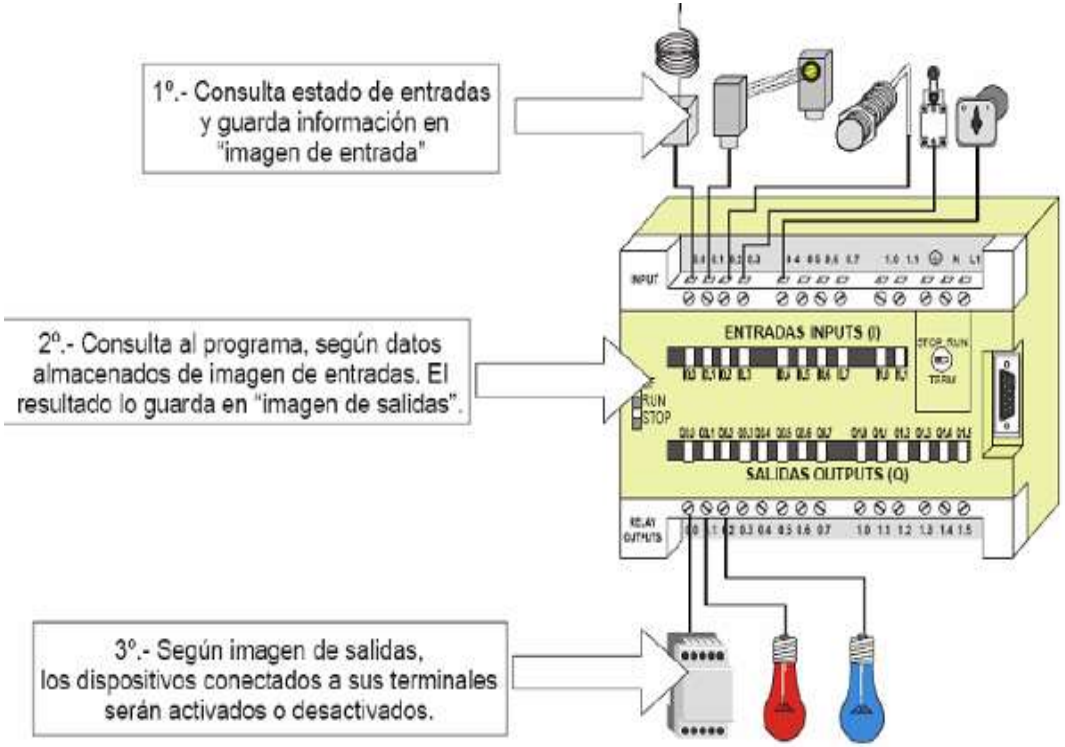
4- Actualización o escritura de salidas. Las salidas son actuadas al valor que corresponda (si por ejemplo el programa ha ordenado que se encienda un piloto o un relé se activa o desactiva la correspondiente salida). Una vez realizado este paso se retorna al primero, y así continuamente.

Las entradas se leen de manera sincrónica, en el inicio del ciclo

1- Ciclo scan

El esquema de funcionamiento básico es el siguiente





1- Ciclo scan

Ciclo de *scan*: tiempo de un ciclo de PLC, incluye: lectura de entradas, ejecución del programa y actualización de salidas, y su duración es de pocos milisegundos (frecuencia de kHz)

Ejemplo: un programa enciende un piloto cuando se presiona un pulsador.
El tiempo que tarda en encenderse el piloto desde que se pulsa el botón es de un ciclo de *scan* microsegundos, dependiendo del tiempo de ejecución del programa.

El tiempo de ejecución del programa depende del número y tipo de instrucciones que se ejecuten, por lo que puede variar de un ciclo a otro en función de como se haya realizado la programación.

1- Ciclo scan

Un PLC puede presentar los siguientes modos de operación:

- **RUN**: En este estado el programa del PLC se está ejecutando de forma cíclica.

- **STOP**: El programa del PLC está parado y en consecuencia no se leen las entradas ni se escribe sobre las salidas.

- **ERROR**: El PLC presenta un error, ya sea por un problema de cableado o de configuración. En dependencia del tipo de error que se trate el programa está parado o en funcionamiento. Normalmente el programa suele detenerse y las salidas se desactivan.

1- Ciclo scan

En algunos modelos de autómata se puede seleccionar el modo RUN o el modo STOP a través de un pequeño selector ubicado en el módulo de la CPU. En todos los casos se puede seleccionar el modo de trabajo a través del software de programación.

En algunos modelos de autómata puede haber otro modo de trabajo que es el **RUN PROTEGIDO**, que consiste en un modo de trabajo en el que el programa se está ejecutando, pero no se permite realizar modificaciones de software.

Cuando un PLC entra en Error, normalmente se debe a que detecta un fallo en la fase del **Test o Check** (fallo en alimentación, error de programación, etc.)

1- Ciclo scan

Hay dos tipos de test que se realizan en el PLC:

- **Test Inicial** (*Power-On Sequence*): Sólo se realiza una vez, cuando se da tensión al PLC. Suele verificar:

    - Las unidades internas de hardware del PLC
    - La memoria del PLC: borra aquellas posiciones de memoria que no son remanentes

  Si no se supera este test el PLC entra en error y se detiene. Si se supera, entra en modo cíclico (test cíclico, leer entradas, ejecución programa y actualizar salidas)

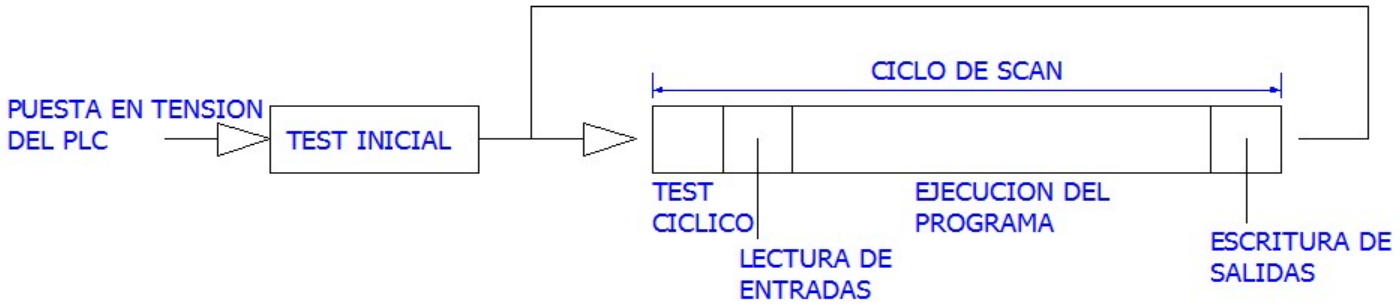
- **Test ciclo:** Se hace cada ciclo de *scan* y en este se comprueba:

    - Cumplimiento del *watchdog*
    - Test de hardware interno y de conexiones de las entradas/salidas

1- Ciclo scan

Watchdog (o tiempo de salvaguarda) es un temporizador interno que fija un tiempo máximo de ciclo de *scan*. Si el ciclo de *scan* es superior a este tiempo prefijado, la PLC entra en estado de error y el programa se detiene y se desactivan las salidas. Normalmente este tiempo esta prefijado entre 0,5 y 1s y en algunos modelos de PLC se puede configurar.

Teniendo en cuenta los dos tipos de test, el ciclo de funcionamiento de un PLC es:



1- Ciclo scan

El **tiempo de ejecución** del programa varía en función de las instrucciones programadas y del tiempo de ejecución de cada una de las instrucciones.
Puede variar en cada ciclo.

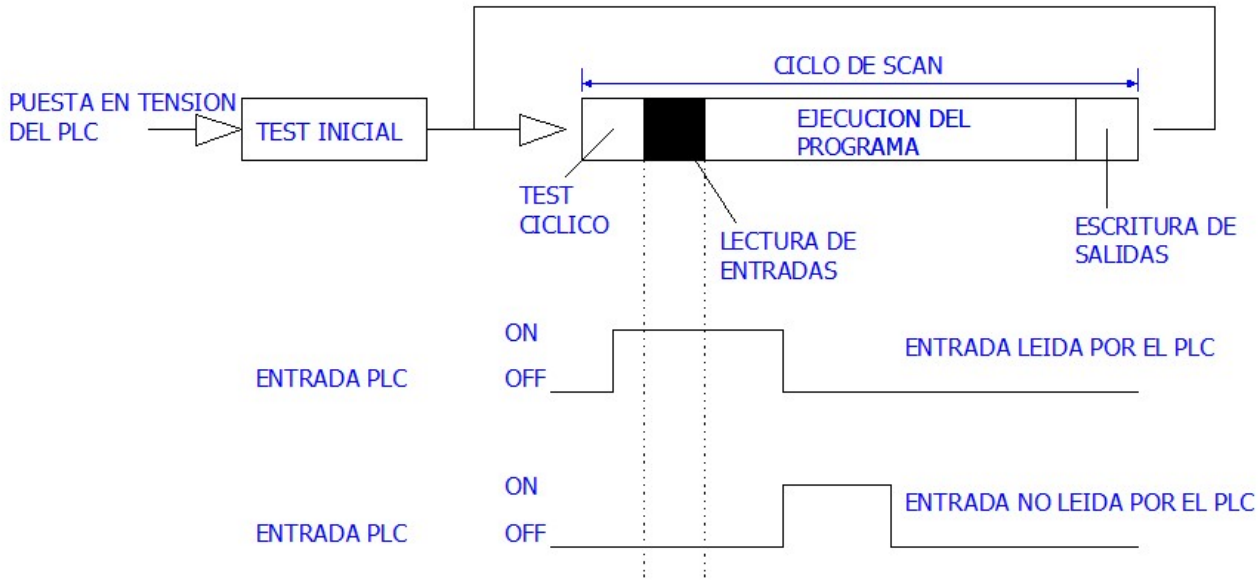
En caso de **error** en el test, en la mayoría de casos el PLC entra en modo de error y **se detiene** (dependerá del tipo de error). **Cada error tiene un código** para facilitar su identificación. Los códigos deben buscarse en una tabla de errores para obtener una descripción detallada.

El estado del PLC se puede consultar a través del software de programación.
En PLC SIEMENS se asignan unos determinados **bloques OB** que se ejecutan en determinados eventos, como los errores.

También, la mayoría de PLC disponen de una tabla de históricos que permite consultar los errores ocurridos a lo largo del tiempo.

1- Ciclo scan

En el ciclo de funcionamiento de un PLC puede ocurrir que una entrada se active durante un pulso de tiempo muy corto y no pueda ser leída en el ciclo de lectura de entradas.



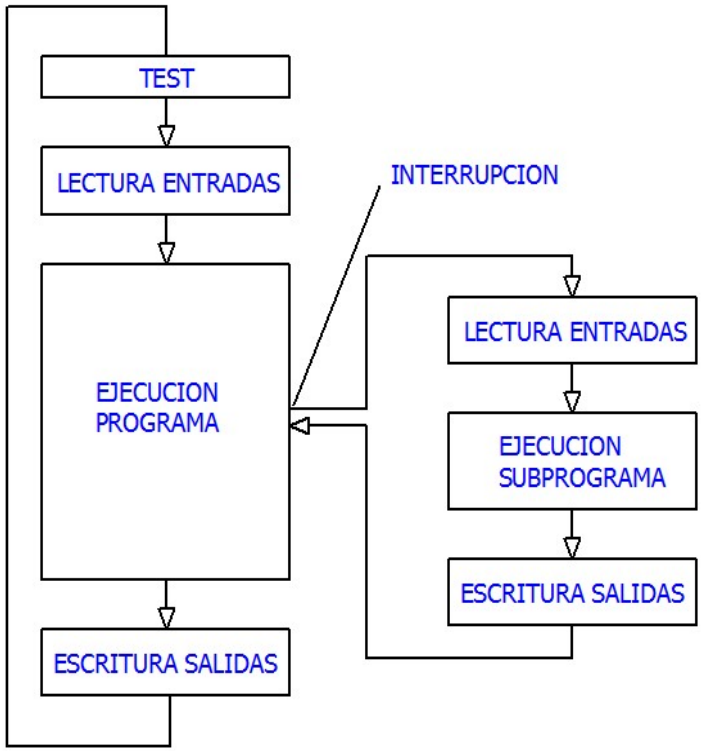
Para evitar este problema se pueden configurar **interrupciones** que pausan temporalmente el ciclo de funcionamiento del PLC para leer este tipo de entradas.

1- Ciclo scan

Cuando se produce una interrupción el ciclo de funcionamiento normal del PLC se interrumpe, de tal manera que:

- **Se vuelven a leer las entradas**
- Hay la posibilidad de **ejecutar un pequeño subprograma** (normalmente de poca extensión)
- **Se actualizan las salidas.**

Una vez realizado esto, se retorna al ciclo normal del PLC justo en el punto en que se produjo la interrupción.



1- Ciclo scan

Una **interrupción** en un PLC se puede producir de dos maneras: mediante un temporizador interno o mediante la activación de una señal externa.

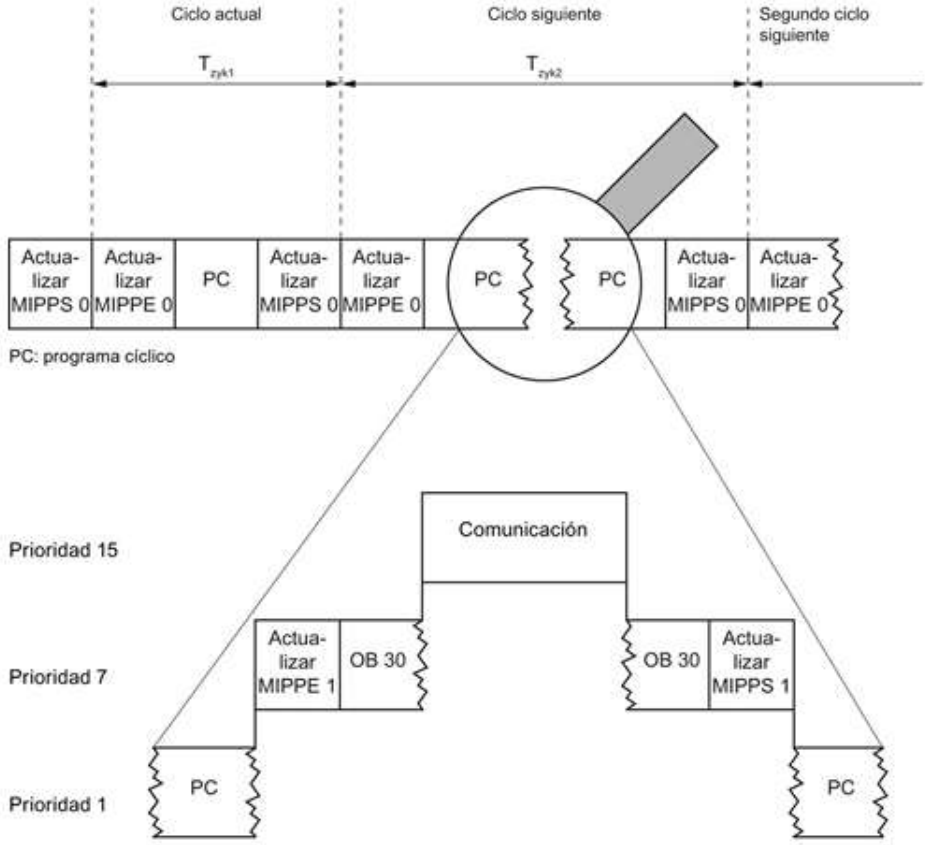
- **Señal externa** (asincrónica): En muchos modelos de PLC existe la posibilidad de configurar entradas para que provoquen una interrupción, de esta manera se asegura que esta entrada va a poder ser leída por el PLC
- **Temporizador interno** (sincrónica): En algunos modelos de PLC se puede configurar un temporizador para que ejecute una interrupción cada determinado tiempo (ms)

muy corta.

Tanto en un caso como en otro, se realiza una nueva lectura de las entradas donde se podrán leer pulsos de entradas que tengan una duración

En el ejemplo mostrado, se muestra una interrupción cíclica (OB30, prioridad 7) que, a su vez, está interrumpida por una comunicación.

Se observa el aumento de tiempo de *scan* en ese ciclo.



1- Ciclo scan

En el caso de que un PLC tenga que leer un tren de pulsos de alta frecuencia (por ejemplo de un *encoder*), se utilizará un hardware específico para este tipo de lecturas.

En muchos modelos de PLC y módulos se encuentran **entradas** de "**alta frecuencia**", preparadas para leer trenes de pulsos provenientes de encoders.

2- Programación

Programar un PLC consiste en introducir las instrucciones de control que permitan realizar las actuaciones necesarias en función del estado de las entradas.

La programación se realizará mediante un software concreto, normalmente desarrollado por el propio fabricante del PLC, que permita, entre otras funciones:

- Configurar el hardware del PLC.
- Configurar las entradas/salidas del sistema
- Editar el programa
- Compilar el programa y búsqueda de errores
- Cargar el programa en el PLC
- Conexión *online* con el PLC para ver el estado del mismo y sus variables internas (*debug*)

2- Programación

Ejemplos de *software* de programación pueden ser:

- Tia Portal para autómatas Siemens
- Unity Pro para autómatas Schneider/Telemecanique
- Cx-Programmer para autómatas Omron.
- Rs-Logix para autómatas Allen Bradley/Rockwell

Muchos de estos softwares incorporan utilidades de simulación que permiten trabajar en el PC mediante PLC virtual. Esto permite realizar la programación y testeos del programa antes de su ejecución en un PLC real.

Existen diversos lenguajes de programación al alcance del usuario: lenguajes de instrucciones, lenguaje de contactos, por bloques, SFC, de alto nivel, etc. Incluso dentro de un mismo programa se pueden emplear distintos lenguajes de programación. A nivel industrial el lenguaje más empleado es el **lenguaje de contactos**, también llamado diagrama escalera (***Ladder***) o **KOP** por ser el más parecido a una representación eléctrica.

2- Programación

Al programar un sistema automatizado hay que realizar los siguientes pasos.

- **Identificar que debe realizar el sistema**. Esto se puede realizar mediante un ***grafcet*** que nos muestre como evoluciona el sistema en el tiempo. Es equivalente al pseudocódigo en programación de microcontroladores.

- **Parametrizar el hardware del PLC**, identificando entradas/salidas, y asignar una dirección (muchas veces es el propio PLC quien de forma automática las asigna).
  En este punto debe realizarse la tabla de entradas/salidas y la realización de un mapa de memoria.

- **Asignar una identificación** o literal a cada una de las entradas/salidas (por extensión a todas las direcciones de memoria) que permita su identificación de una forma directa y clara en el programa

- **Realizar el código**, compilarlo, transferirlo al simulador (si es posible) para testearlo

- **Transferir el código compilado al autómata.**

- **Depurar el programa** y el funcionamiento del sistema. Conviene hacer copias de seguridad

2- Programación

En primer lugar, una vez definido el sistema y su funcionamiento, debe realizarse la tabla de entradas/salidas y su direccionamiento en el autómata.

Por ejemplo, supongamos la tabla de entradas/salidas de un sistema automatizado sobre un Siemens S7-300:

Disponemos de una instalación con 2 motores: uno principal y otro auxiliar, el operario tiene dos pulsadores para poner en marcha cada uno de los motores:

- Pulsador Prueba Marcha Auxiliar -> Al mantenerlo pulsado funciona el motor auxiliar. Es un pulsador NO
- Pulsador Marcha pral. -> Su pulsación provoca la puesta en funcionamiento del motor principal. Físicamente es un pulsador NO.
- Pulsador Paro -> Su pulsación provoca el paro de los motores. Físicamente es un pulsador NC

2- Programación

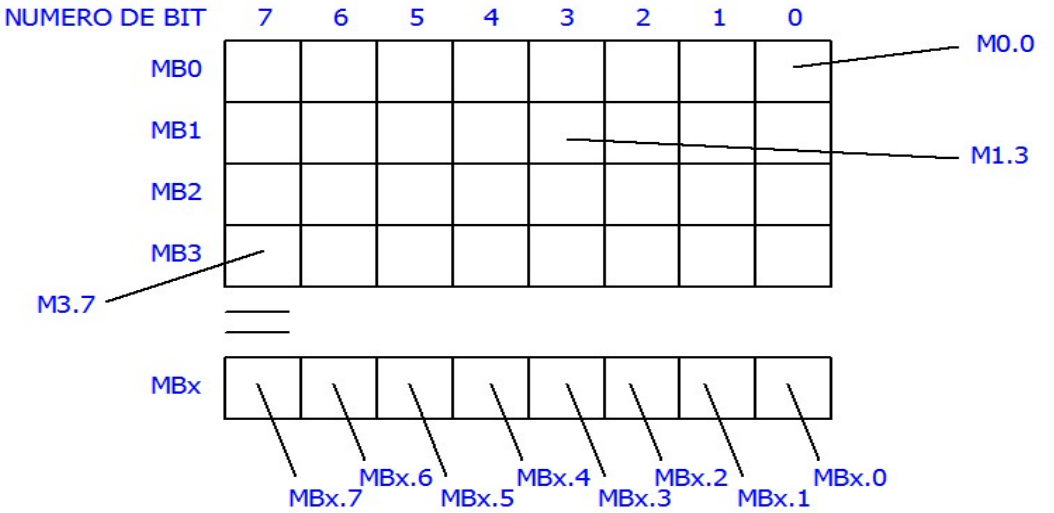
- Salida marcha motor principal
- Salida marcha motor auxiliar

Entrada	Descripción	Salida	Descripción
I0.0	Pulsador Prueba Motor Auxiliar	Q4.0	K Marcha Motor Auxiliar
I0.1	Pulsador Marcha Motores	Q4.1	K Marcha Motor Principal
I0.2	Pulsador Paro Motores		

2- Programación

Además de definir el direccionamiento de las entradas/salidas, se puede emplear la memoria interna del PLC. En el caso concreto de SIEMENS está organizada en grupos de bytes (8 bits), y cada uno es direccionado mediante la expresión 'MBx', siendo x el número de byte que va desde 0 hasta un máximo.

Para direccionar un bit de un registro se emplea la fórmula MBx.numero de bit, siendo un bit de 0 a 7.



2- Programación

A todas las variables del PLC conviene asignarles un literal (también llamado etiqueta o *tag*) que permita identificarlos de una forma clara y sin confusión. Esto se suele realizar en una tabla en la que se asocia la dirección de memoria con su etiqueta.

Nombre	Tipo de datos	Dirección	Rema...	Acces...	Visibl...	Com
Pulsador_Prueba_MotorAux	Bool	%I0.0		☑	☑	
Pulsador_Marcha	Bool	%I0.1		☑	☑	
Pulsador_Paro	Bool	%I0.2		☑	☑	
K_Marcha_Aux	Bool	%Q4.0		☑	☑	
K_Marcha_Motor_Principal	Bool	%Q4.1		☑	☑	
Bit_Marcha_Motor_Aux	Bool	%M0.0		☑	☑	

Esta tabla se puede ir rellenando mientras se va realizando la programación del autómata.

2- Programación

Los PLC se pueden programar mediante el uso de diversos lenguajes de Programación. Los más usuales son

- **Ladder (LAD) o Lenguaje de Contactos**. También llamado KOP (KOntaktsPlan). Es el más empleado por ser el más intuitivo, ya que permite implementar las funciones como si se estuviera realizando un esquema eléctrico en normativa ANSI.

- **Lenguaje de Instrucciones** (**STL**, StaTement List). También llamado AWL (AnWeisungs-Liste). Es poco intuitivo porque implementa las funciones mediante un literal o código.

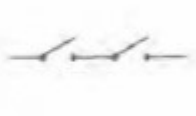
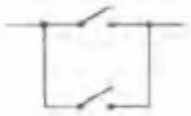
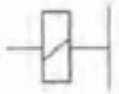
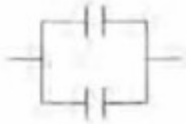
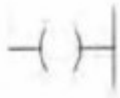
- **Bloques Funcionales**. También llamado **FUP** (FUnktionsPlan). La programación se realiza mediante bloques funcionales del tipo AND, OR, etc.

- **Alto Nivel**. La programación se realiza mediante sentencias del tipo IF…THEN

2- Programación

PROGRAMACION EN LADDER

Se programa abriendo y cerrando circuitos. Es la más intuitiva a nivel eléctrico porque implementa un esquema en normativa ANSI. Las funciones más usadas son las siguientes.

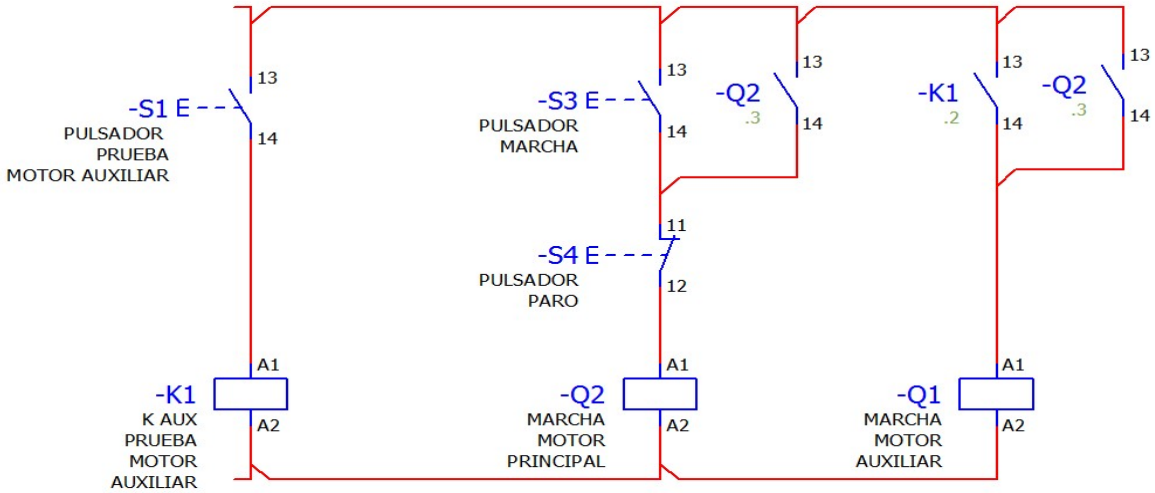
FUNCIÓN LÓGICA	Operación producto lógico	Operación suma lógica	Operación negación	Asignación de valor
INSTRUCCIONES BOOLEANAS	AND	OR	NOT	OUT
ESQUEMAS DE RELÉS (DIN 40713-16)				
DIAGRAMAS DE CONTACTOS (NEMA/DIN 19239)				

2- Programación

Siguiendo el ejemplo:

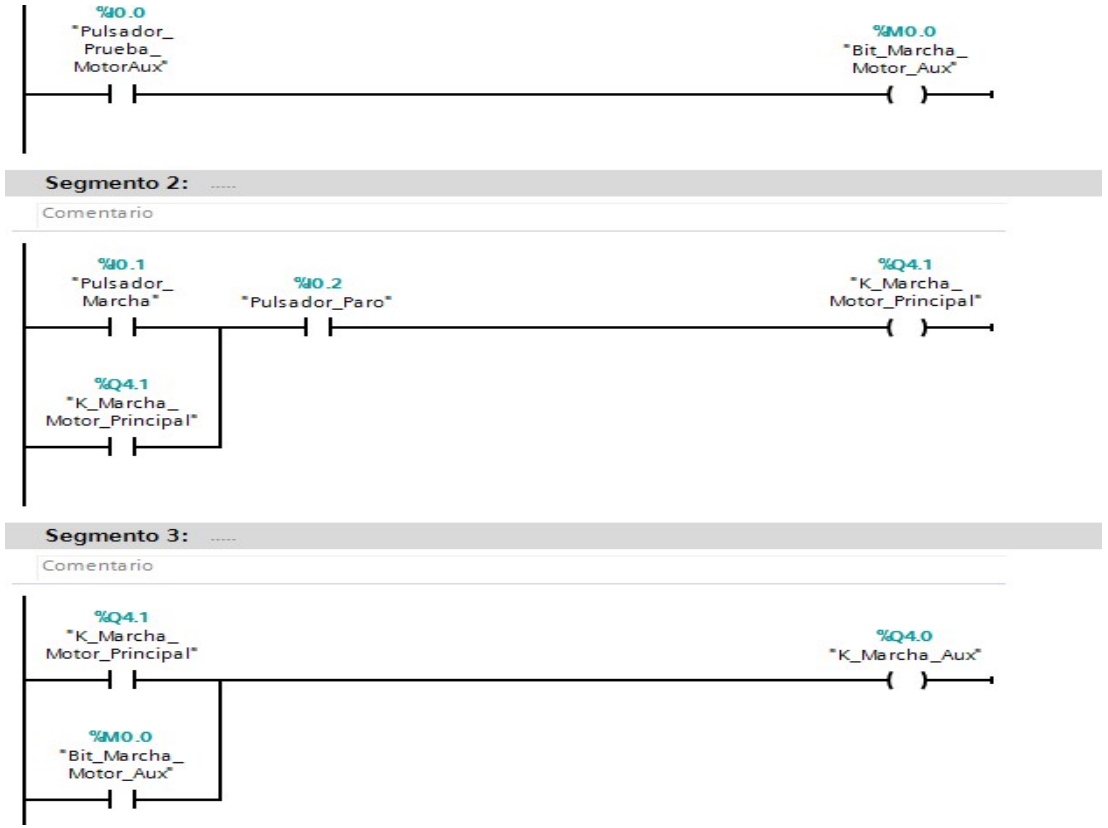
- Tenemos dos motores, uno auxiliar y otro principal, accionados por dos pulsadores.
- El pulsador “Marcha” (S3) pone en marcha ambos motores
- El pulsador “Prueba motor auxiliar” (S1) pone en marcha el motor auxiliar (sin enclavamiento)
- El pulsador de paro (S4) detiene ambos motores

Si hacemos los esquemas eléctricos tendremos:



2- Programación

Llevando el esquema a programación *ladder*



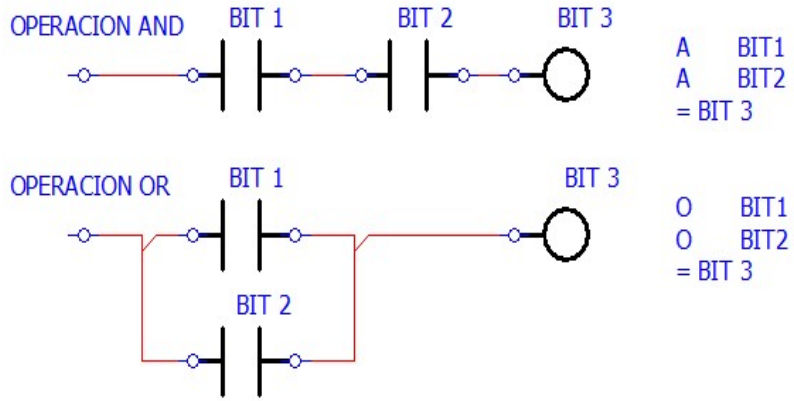
2- Programación

PROGRAMACION EN AWL

A cada función se le asigna un símbolo y se van escribiendo las funciones en modo secuencial. La tabla de asignación de bits es la siguiente

Bit logic	
A	And
AN	And Not
O	Or
ON	Or Not
X	Exclusive Or
XN	Exclusive Or Not
FN	Edge Negative
FP	Edge Positive
( )	Nesting
=	Assign

Por ejemplo:

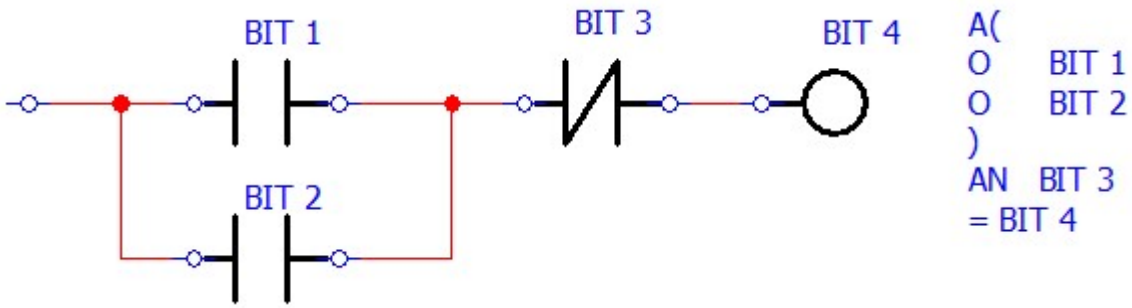


2- Programación

En el caso de que se quiera dar mayor prioridad a un operador, este se pone entre paréntesis. Por ejemplo, el siguiente código es una AND de dos operandos:

- El primero está formado por una OR de dos bits.
- El segundo en un bit

Se da prioridad a la OR, para después hacer la AND

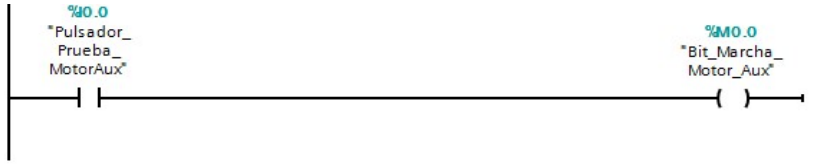



```
A(
O     BIT 1
O     BIT 2
)
AN   BIT 3
= BIT 4
```

2- Programación

El programa queda de la siguiente manera:

```
1     A     "Pulsador_Prueba_MotorAux"     %I0.0
2     =     "Bit_Marcha_Motor_Aux"         %M0.0
```



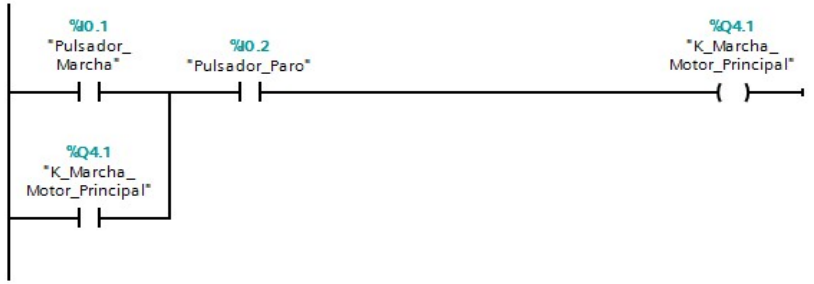
gmento 2: .....

mentario

```
1     A(
2     O     "Pulsador_Marcha"              %I0.1
3     O     "K_Marcha_Motor_Principal"     %Q4.1
4     )
5     A     "Pulsador_Paro"                %I0.2
6     =     "K_Marcha_Motor_Principal"     %Q4.1
```

Segmento 2: .....

Comentario



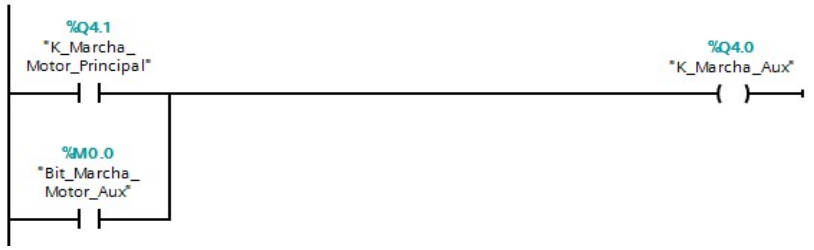
gmento 3: .....

mentario

```
1     O     "K_Marcha_Motor_Principal"     %Q4.1
2     O     "Bit_Marcha_Motor_Aux"         %M0.0
3     =     "K_Marcha_Aux"                 %Q4.0
```

Segmento 3: .....

Comentario

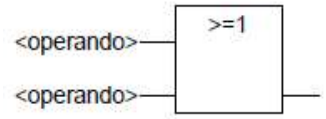


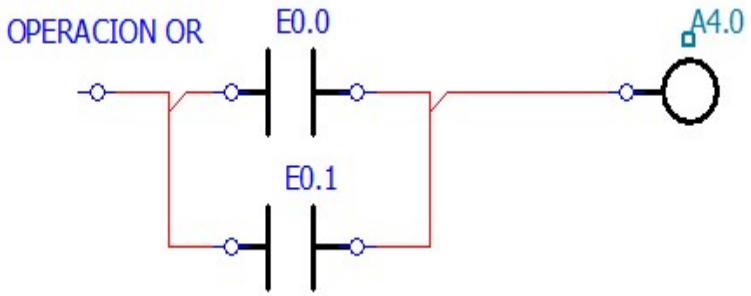
2- Programación

PROGRAMACION EN BLOQUES DE FUNCIONES

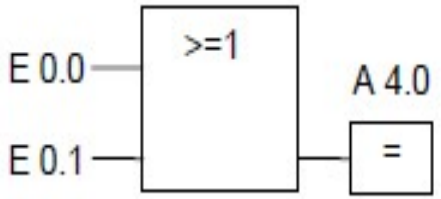
En Bloques de Funciones cada función lógica se representa mediante un bloque.

>=1 : Operación lógica O



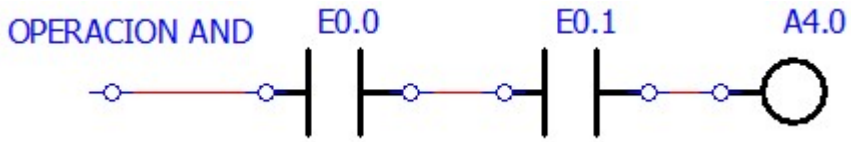


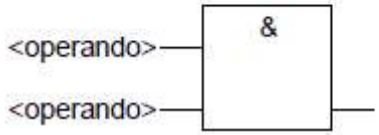
La asignación se realiza con un Bloque =

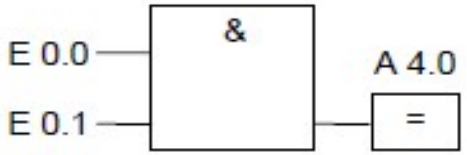


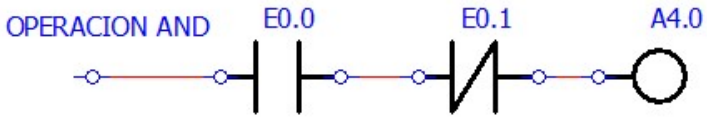
2- Programación

& : Operación lógica Y



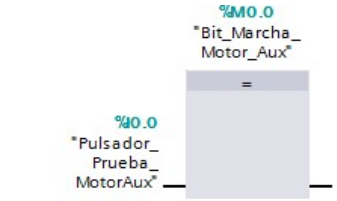


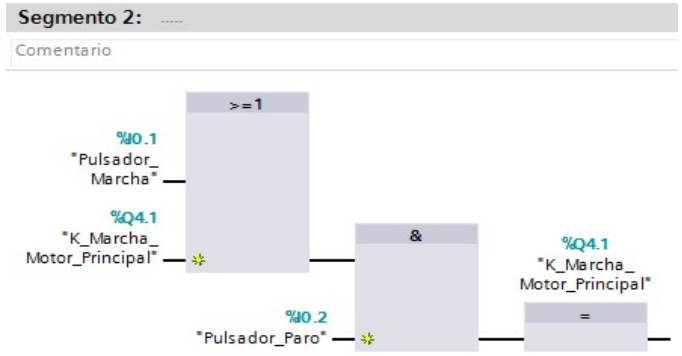


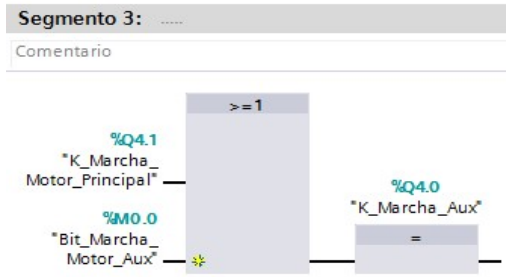


circul

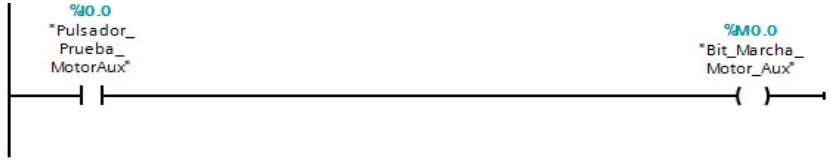
2- Programación







El programa queda de la siguiente manera:



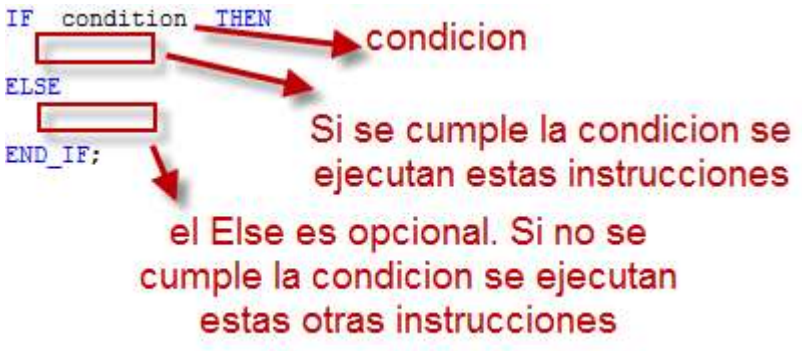
2- Programación

PROGRAMACION EN ALTO NIVEL

Es un tipo de programación estructurada que utiliza instrucciones de alto nivel. Las instrucciones más usadas son:

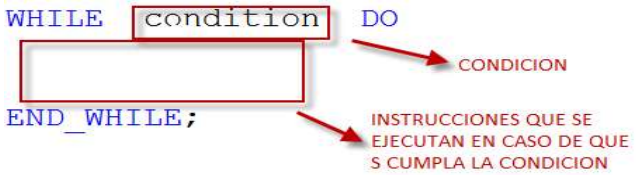
IF condición THEN: En esta instrucción si se cumple una condición lógica se ejecutan una serie de instrucciones. El ELSE es opcional e indica que si no se cumple la condición, se ejecutan estas otras

.

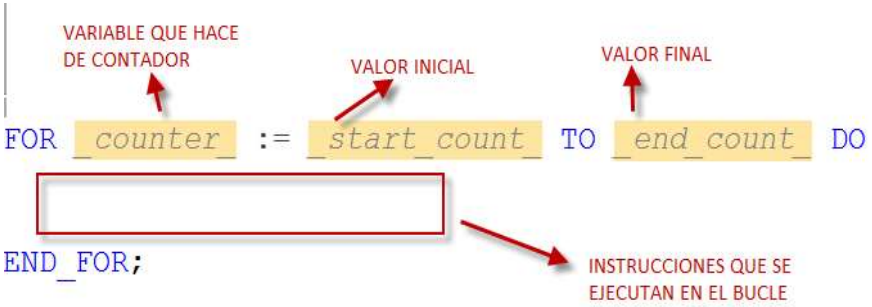


2- Programación

WHILE condicion DO: Se ejecuta un conjunto de instrucciones de forma cíclica mientras se cumpla la condición. En el momento en que deje de cumplirse sale de este bucle.



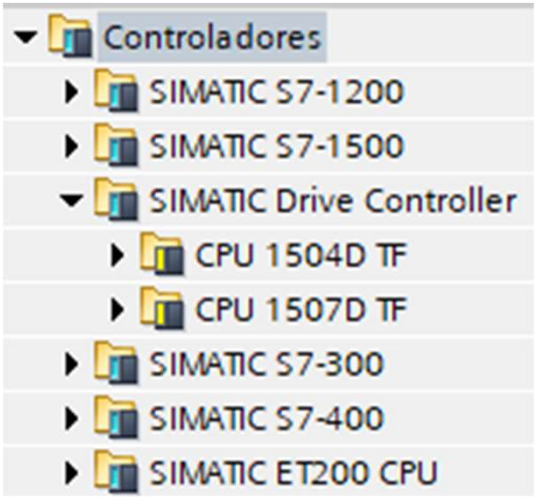
FOR contador TO valor final DO: Ejecuta un bucle un número de veces que viene condicionado por la variable contador: se declara un valor inicial para el contador, un valor máximo, y a cada ejecución del bucle el contador se incrementa en uno, mientras no se llegue al valor máximo no se sale del bucle.



3- Configuración de *hardware*

Uno de los productos industriales mas famosos de la marca siemens es su familia de autómatas, los cuales cuentan con una fuerte implantación en el mercado industrial. Actualmente dispone de las siguientes familias:

- **Familia 200:** Autómatas compactos muy sencillos. No programable desde TIA PORTAL. Descatalogado
- **Familia S7-300:** Autómatas modulares. Descatalogado
- **Familia s7-400:** Autómatas modulares de gama alta
- **Familia 1200:** Familia de autómatas semi compactos.
- **Familia 1500:** Nueva familia de autómatas modulares



3- Configuración de *hardware*

Todos estos autómatas, a excepción del 200, se programan mediante el *software* Tia Portal.

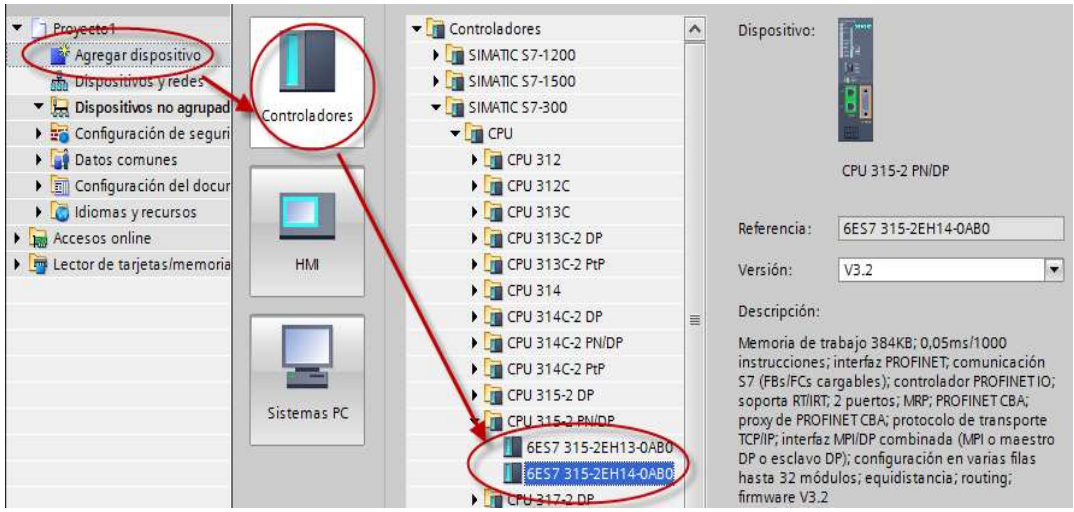
3- Configuración de *hardware*

Cuando se crea un proyecto nuevo en TIA PORTAL lo primero que hay que hacer es realizar la configuración del Hardware.

Configurar el Hardware consiste en informar en el programa sobre los dispositivos reales que existen: plc's, módulos de entrada salida, paneles táctiles, módulos especiales como pueden ser servos, módulos de comunicación, etc.

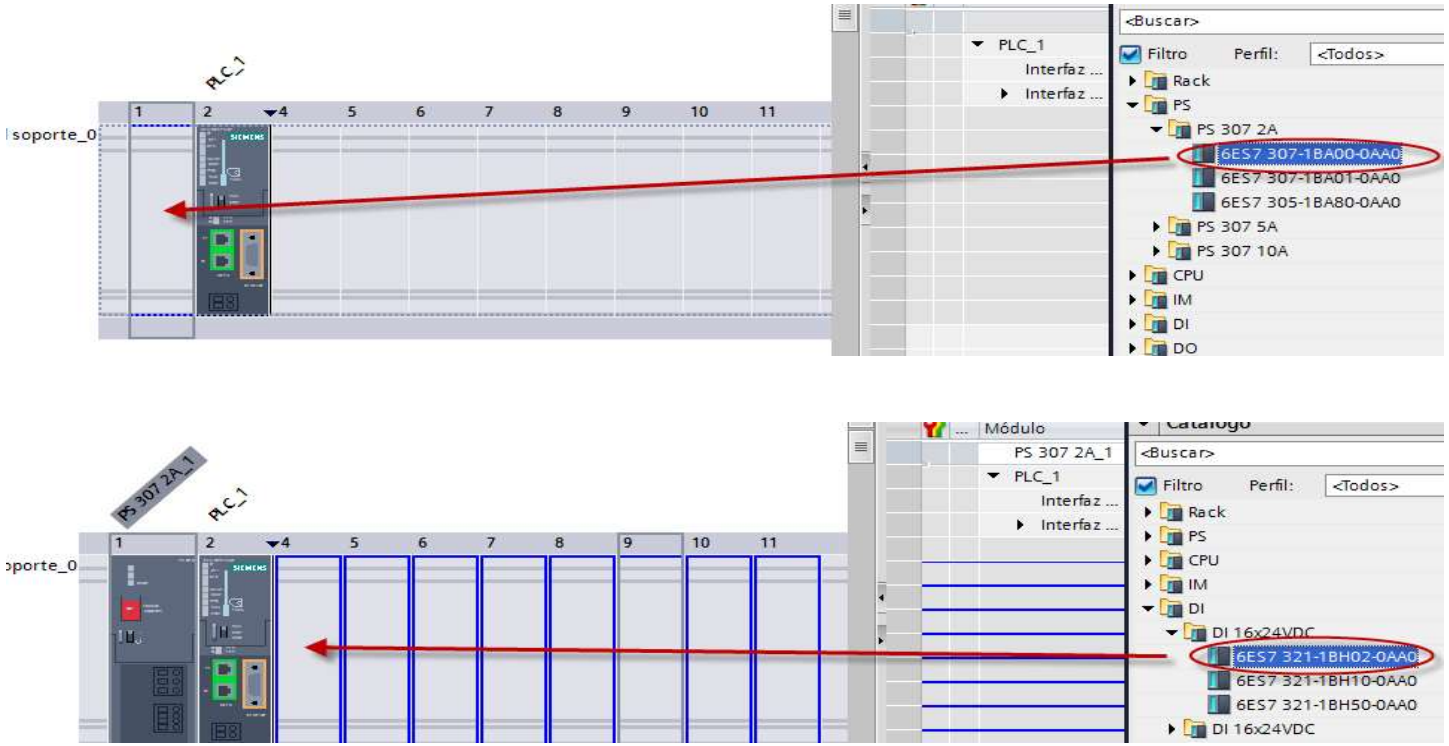
Deben seguirse los siguientes pasos

1- Seleccionar el PLC que se va a emplear.

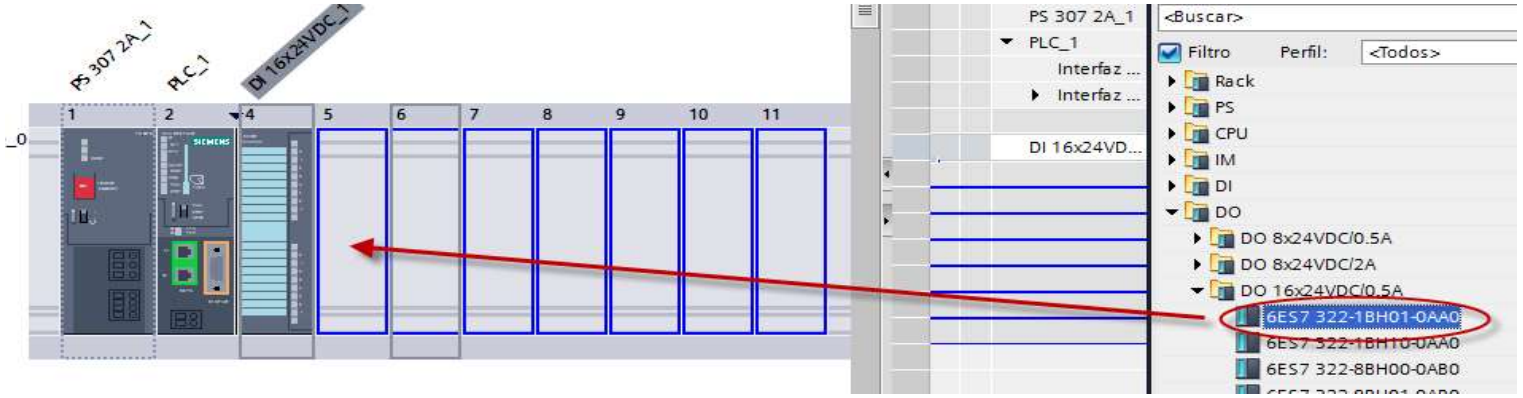


3- Configuración de *hardware*

2- Se añaden el resto de módulos al Rack. Por ejemplo se añade una fuente de alimentación, una tarjeta de 16 entradas digitales y una tarjeta de 16 salidas.

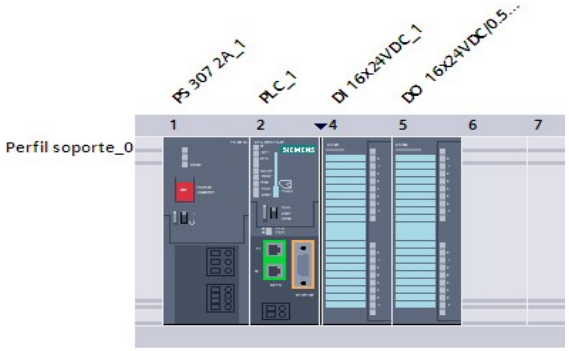


3- Configuración de *hardware*



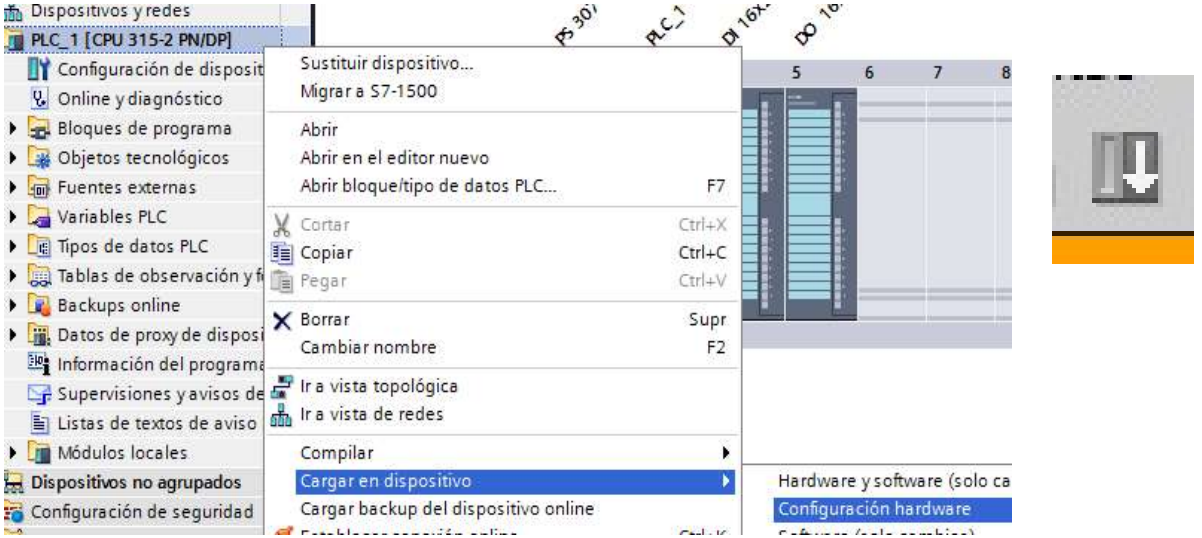
La configuración final queda de la siguiente manera

PLC: CPU 315 PN/DP Ref 6ES7 315-2EH14-0AB0
Fuente Alimentacion: 6ES7 307-1BA00-0AA0
Modulo 16 Entradas: 6ES7 321-1BH02-0AA0
Modulo 16 Salidas: 6ES7 322-1BH01-0AA0



3- Configuración de *hardware*

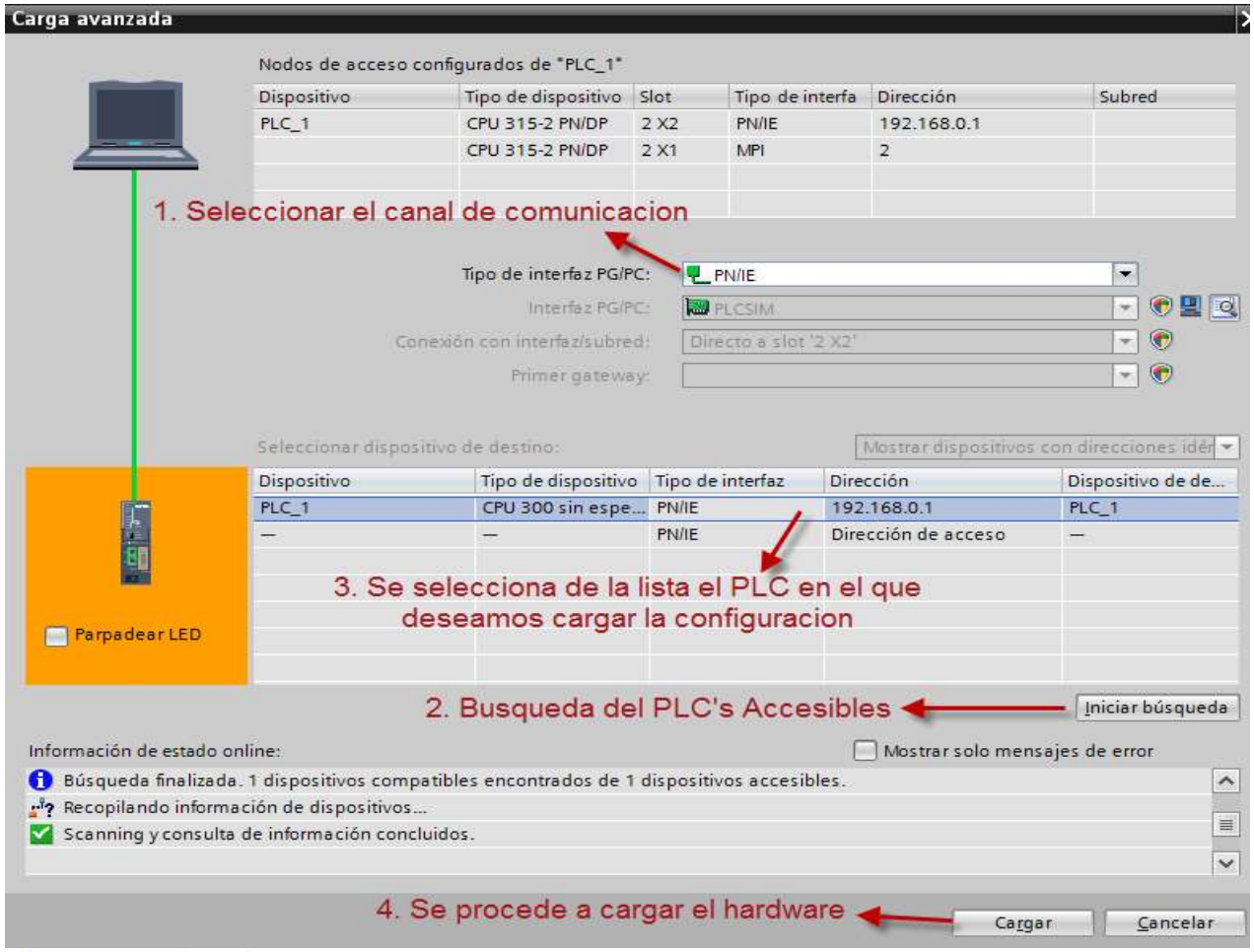
Por ultimo se realiza la carga del hardware sobre el PLC.



En el caso de utilizar un simulador (PLCSIM) debe ejecutarse el simulador antes de cargar el *software*

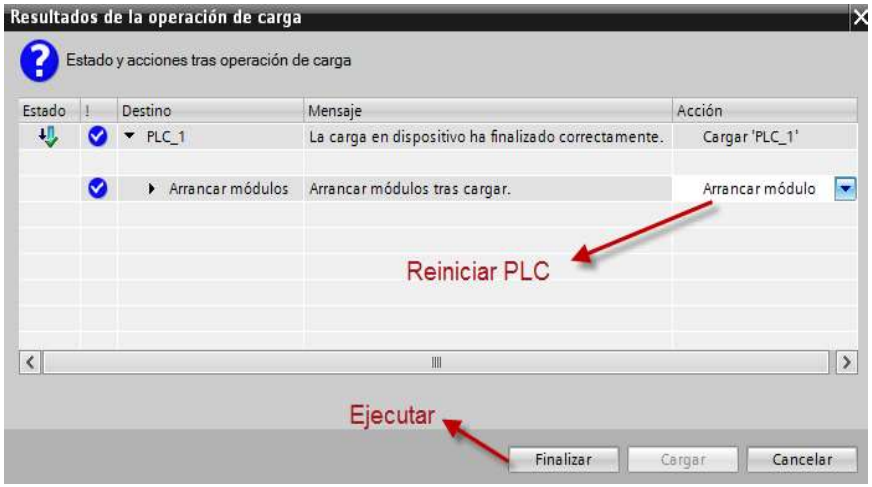
3- Configuración de *hardware*

En la pantalla que aparece, TIA PORTAL busca en toda la red el PLC configurado a través de la conexión utilizada, y una vez localizado transfiere programa al *hardware* encontrado.



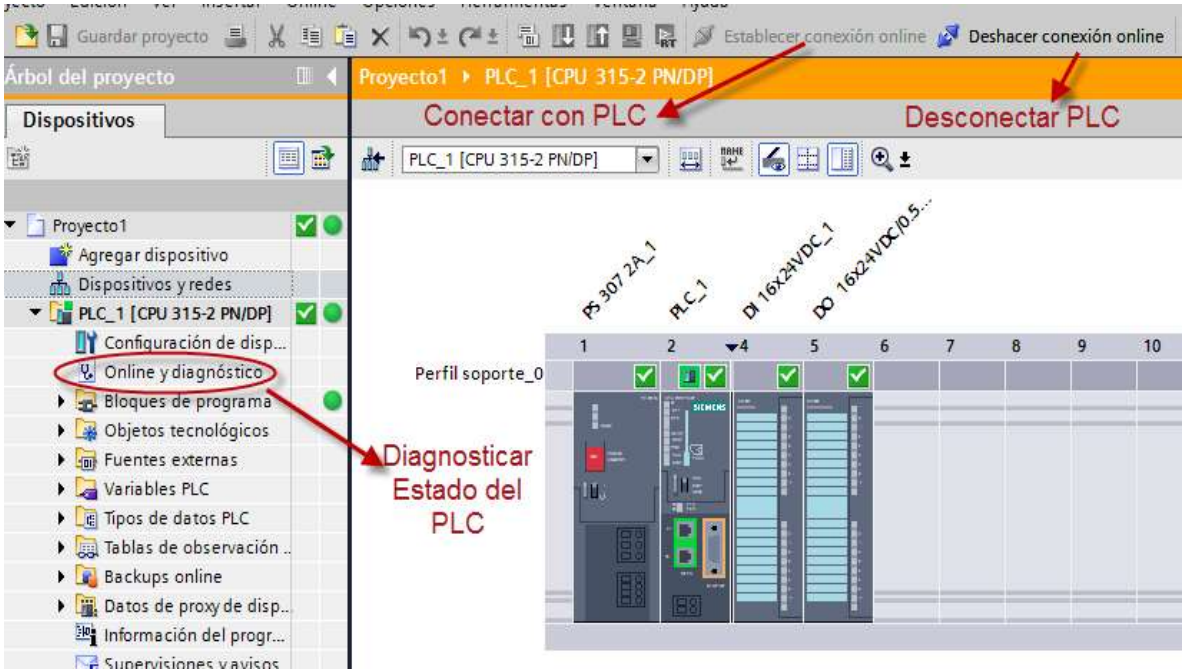
3- Configuración de *hardware*

Una vez realizada la carga hacer un reinicio del PLC (arrancar modulo)



3- Configuración de *hardware*

Una vez realizada la carga del Hardware en el PLC, nos podemos conectar Online con el mismo y comprobar el estado



4- Estructura de memoria

Con el hardware del PLC correctamente configurados debemos definir el mapeado de la memoria. Todo el mapeado de memoria en los autómatas siemens es a 8 bits, y en éste tendremos por un lado el direccionamiento de las entradas salidas y por otro lado el mapa de la memoria.

Deben tenerse en cuenta las siguientes consideraciones en el direccionamiento de las entradas digitales de los autómatas Siemens.

- Las direcciones de entradas y salidas comienzan con la dirección 0 para el primer modulo del rack.

- **A cada modulo se le asignan 4 bytes (se usen o no)**.

- La numeración de cada grupo de entradas se pondrá correlativa dependiendo no de la posición del rack sino del orden en el que insertemos las tarjetas.

4- Estructura de memoria

FUENTE ALIMEN.	CPU	IM emisor	0.0 a 3.7	4.0 a 7.7	8.0 a 11.7	12.0 a 15.7	16.0 a 19.7	20.0 a 23.7	24.0 a 27.7	28.0 a 31.7
		IM receptor	32.0 a 35.7	36.0 a 39.7	40.0 a 43.7	44.0 a 47.7	48.0 a 51.7	52.0 a 55.7	56.0 a 59.7	60.0 a 63.7

En cuanto al direccionamiento Analógico.

- Las direcciones analógicas para entradas y salidas empiezan con la dirección 256
- A cada módulo analógico, se le asignan 16 bytes (se usen o no).
- Cada señal analógica ocupa 2 bytes.

FUENTE ALIMEN.	CPU	IM emisor	256 a 271	272 a 287	288 a 303	304 a 319	320 a 335	336 a 351	352 a 367	368 a 383

4- Estructura de memoria

Desde el Tia Portal, consultando la configuración Hardware de las tarjetas de entrada/salida, se puede consultar el direccionamiento asignado.

En el caso de la configuración anterior tendremos

16 entradas

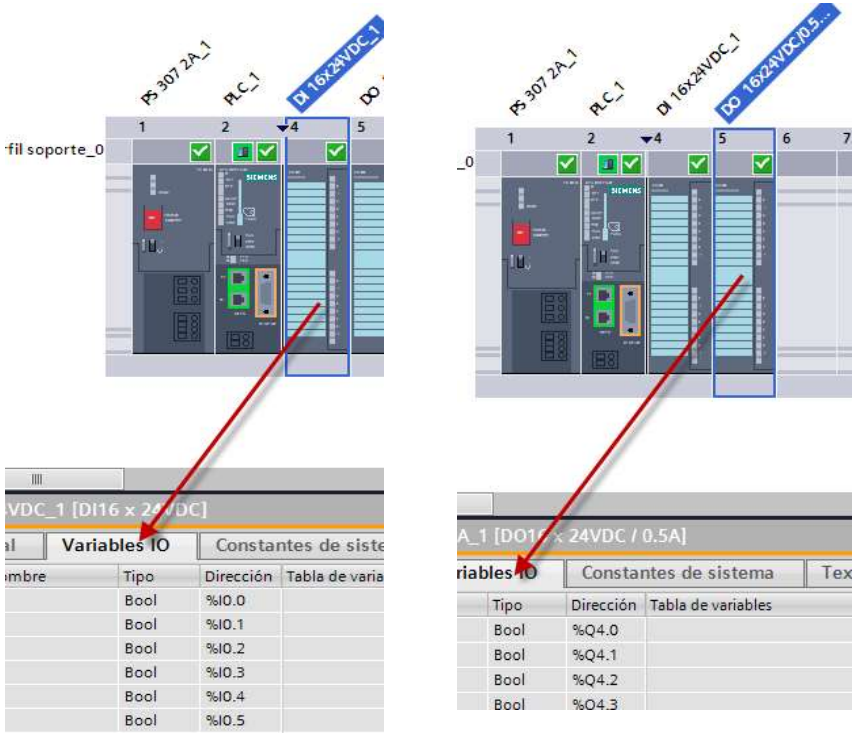
De %I0.0 a %I0.7

De %I1.0 a %I1.7

16 salidas

De %Q4.0 a %Q4.7

De %Q5.0 a %Q5.7



4- Estructura de memoria

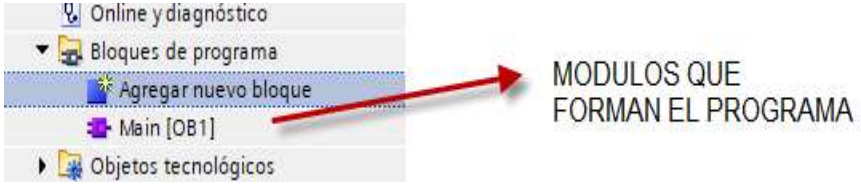
Por ejemplo, tenemos la siguiente configuración:

1er módulo: 16 bits (2 bytes)	2º módulo: 8 bits (1 byte)	3er módulo: 16 bits (2 bytes)
El direccionamiento será:	El direccionamiento será:	El direccionamiento será:
I0.0…….I0.7	I4.0…….I0.7	Q4.0…….Q0.7
I1.0…….I0.7	I5.0…….I0.7 (no usado)	Q5.0…….Q0.7
I2.0…….I0.7 (no usado)	I6.0…….I0.7 (no usado)	I6.0…….I0.7 (no usado)
I3.0…….I0.7 (no usado)	I7.0…….I0.7 (no usado)	I7.0…….I0.7 (no usado)

La memoria interna de datos, esta organizada en grupos de bytes (8 bits), y cada uno es direccionado mediante la expresión ‘MBx’, siendo x el número de byte que va desde 0 hasta un máximo que depende del modelo de PLC seleccionado.

5- Estructura de programa

La programación de los PLC Siemens es estructurada y separar las tareas.



Tenemos los siguientes módulos:

Bloque OB1: Es el módulo que se ejecuta cíclicamente y sobre el se implementa la tarea principal. Desde él podemos realizar llamadas a otros módulos.

5- Estructura de programa

Bloque OB100: El bloque OB100 es el módulo de arranque. Se ejecuta un solo ciclo de *scan* cuando el PLC entra en RUN y se ejecuta antes que el OB1. El OB100 es el primer módulo que se ejecuta.

Bloque OB de tareas rápidas o cíclicas: Los bloques OB30 a OB38 son las tareas periódicas que se ejecutarán cada intervalo de tiempo según la siguiente tabla.

Cyclic OBs	Time interval in ms
OB30	5000
OB31	2000
OB32	1000
OB33	500
OB34	200
OB35	100
OB36	50
OB37	20
OB38	10

Estos bloques tienen prioridad sobre el bloque principal (OB1) y lo interrumpen en el intervalo correspondiente. La prioridad aumenta con el número de OB, es decir el OB31 es prioritario al OB30. Se utiliza para realizar tareas de manera síncrona, por ejemplo, el control PID. Estos bloques realizan lecturas y escrituras en las E/S, como el OB100.

5- Estructura de programa

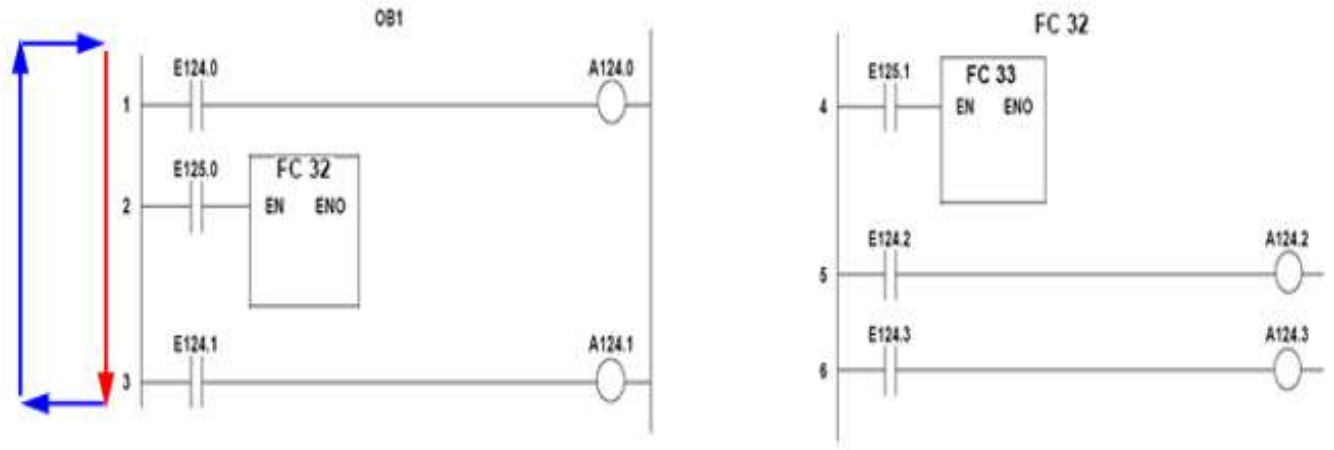
Otros Bloques OB

- OB81 Fallo de alimentación (S7-400) o Fallo de batería.
- OB83 Detección de presencia de módulo (Extraer/insertar). Si no está programado y detecta el error, el PLC pasa a STOP
- OB84 Avería de la CPU. Error de la interface MPI o de la periferia descentralizada.
- OB87: Detección de fallo en comunicación
- OB121: Detecta errores de programación.

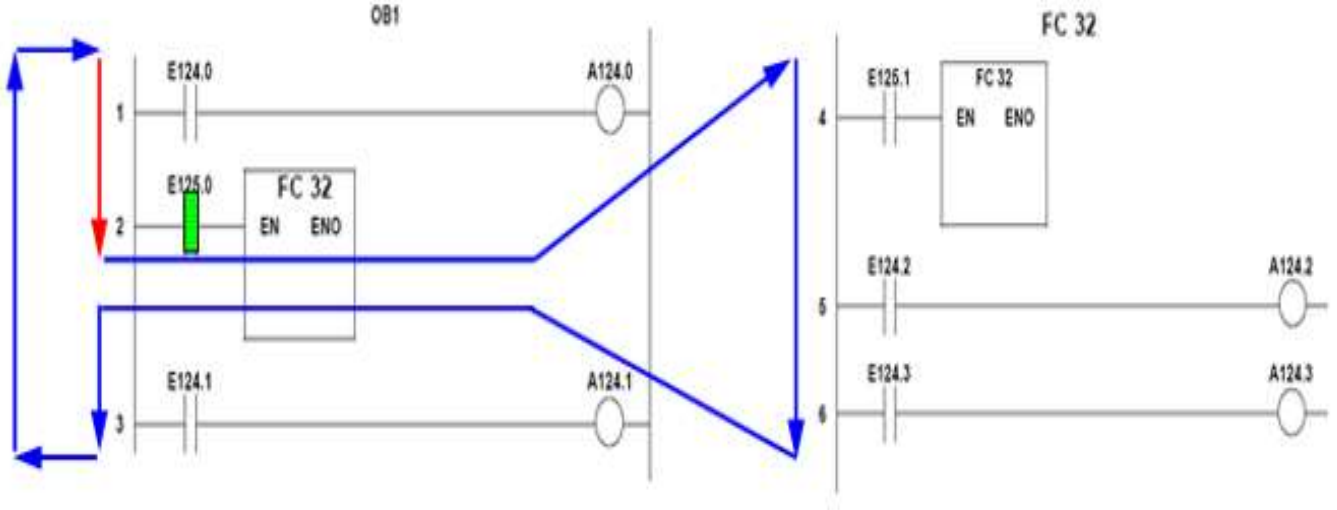
5- Estructura de programa

Bloques FC

Los bloques FC son los bloques de función y pueden ser llamados desde otros bloques. Reciben el nombre con formato FC + NUMERO. Por ejemplo si creamos el FC32



5- Estructura de programa

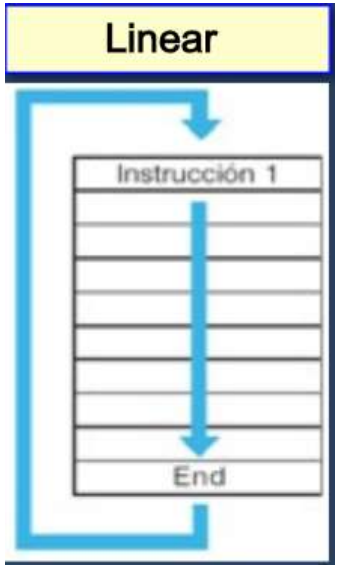


5- Estructura de programa

La estructura de un programa define el tipo de arquitectura que va a tener el programa, es decir, como se va a implementar el código del mismo. Esta arquitectura irá en función del modelo de PLC seleccionado, ya que no todos implementan exactamente las mismas funcionalidades.

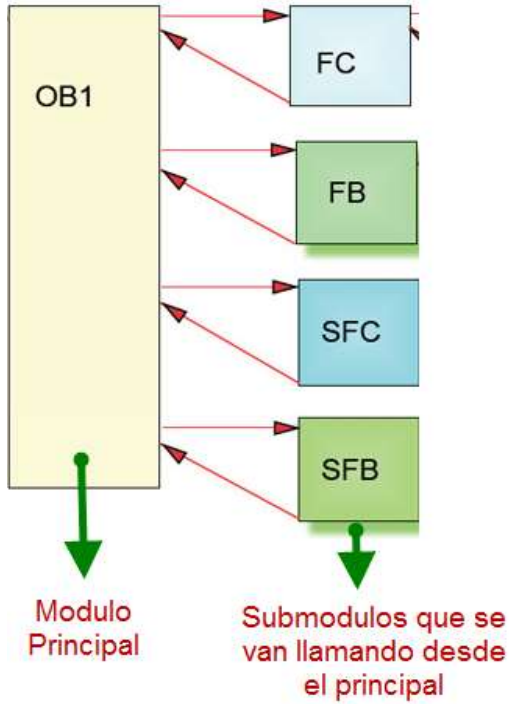
En un PLC a un programa se le llama tarea, y para una tarea podemos encontrar dos formas de organizar el código

- Lineal. Se caracteriza en que todo el programa se ejecuta de forma secuencial, una instrucción detrás de otra.



5- Estructura de programa

- Estructurada. El código se divide en módulos o subprogramas que son llamados desde un programa principal. El orden o número de llamada



Interrupciones:

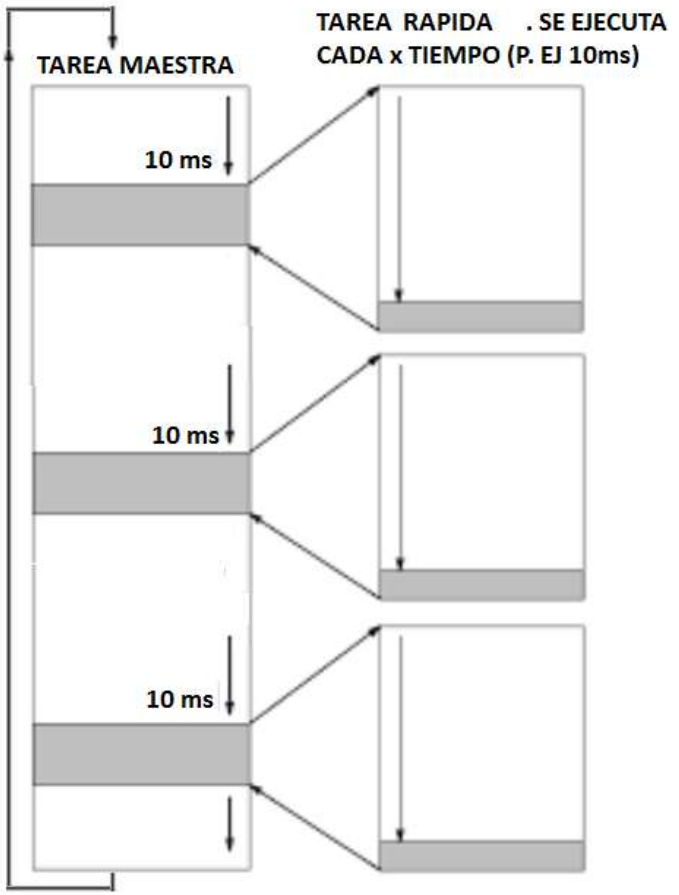
- Tareas Rápidas (síncronas). Son tareas que contienen un código pequeño que se ejecuta cada cierto tiempo. Son tareas prioritarias (OB)

- Tareas de Eventos (asíncronas). Se ejecutan cuando se activa una señal

5- Estructura de programa

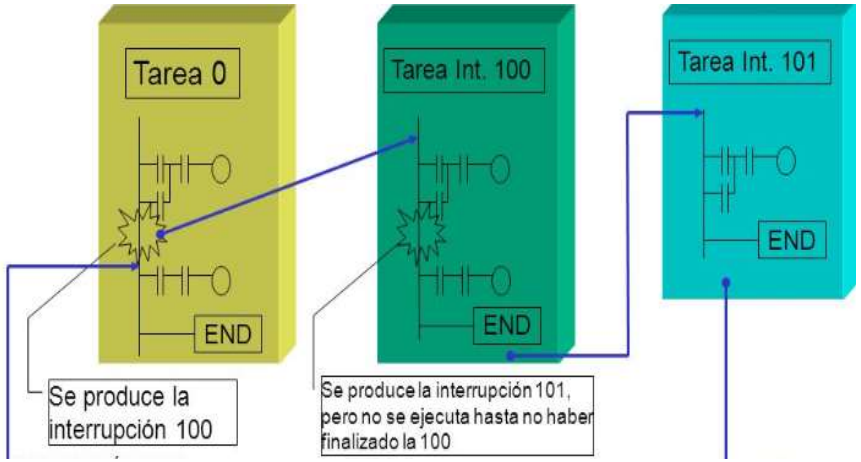
- **Tarea Rápida**: Se ejecuta de forma periódica (cada cierto tiempo).

  - Tiene prioridad alta por lo que interrumpe la tarea maestra.
  - Se ejecuta y retorna a la tarea maestra.
  - Contiene poco código porque penalizan el tiempo de *scan*
  - Si se asigna un tiempo de prioridad muy pequeño puede penalizar de forma importante el ciclo de scan



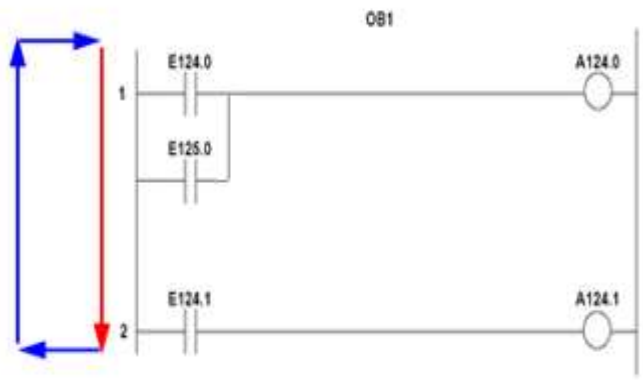
5- Estructura de programa

- **Tarea asíncrona.** Se ejecuta cuando ocurre un evento (por ejemplo, la activación de una entrada)
  - Son las de más alta prioridad, interrumpen todas las demás tareas
  - Una vez ejecutada retorna a las tareas que se había interrumpido.
  - Suelen contener un pequeño programa que necesita ser ejecutado de forma urgente en caso de que se active el evento.



5- Estructura de programa

Cuando una tarea se programa de forma lineal, todo el código se implementa en una sola sección o módulo, y todas las instrucciones se ejecutan de forma secuencial.



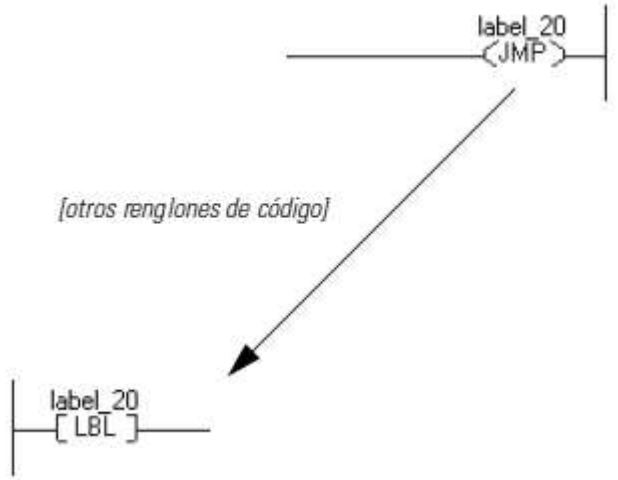
El ciclo de funcionamiento de esta tarea es la descrita anteriormente: lectura de entradas, ejecución del programa, y escritura de salidas.

Dentro de esta ejecución lineal del programa se pueden establecer saltos, de manera que se puede condicionar la ejecución de una parte del programa. Para hacer esto los PLC suelen implementar una serie de instrucciones que al ejecutarse realizan este salto. Se suele diferenciar entre:

5- Estructura de programa

- Instrucción de Salto: Instrucción que en caso de ejecutarse realiza ese salto.
- Etiqueta de Destino: Es un identificador que marca el punto de destino del salto.

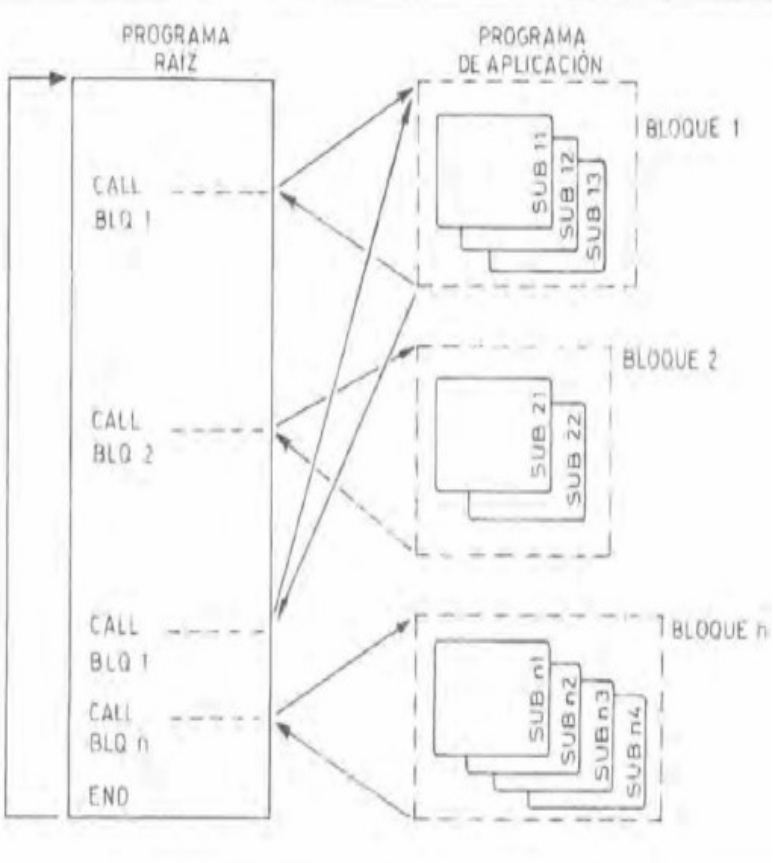
En este código en RSLogix (Allend Bradley) si se ejecuta el JMP se salta a la etiqueta 'label_20' y todo el código que hay en medio lógicamente no se ejecuta.



5- Estructura de programa

En la programación estructurada una tarea se divide en programas o módulos de tal manera que tendremos:

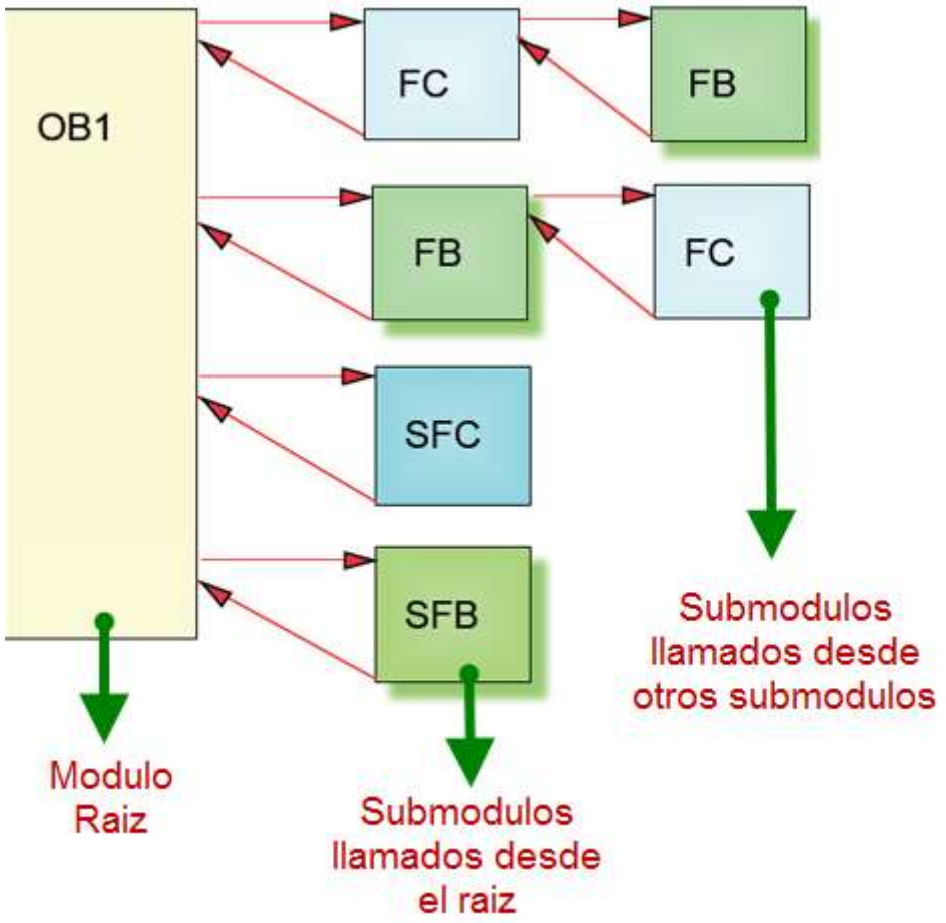
- Un módulo principal: es el módulo raíz desde el cual se realizan las llamadas a los otros submódulos.
- Uno o varios submódulos: Son los módulos secundarios que son llamados desde el principal. Un mismo submódulo puede no ser llamado o llamado varias veces, según esté hecho el programa.



5- Estructura de programa

Desde los submódulos se pueden llamar igualmente a otros submódulos, pudiéndose formar una estructura de programación en forma de árbol

La programación estructurada permite seleccionar en todo momento que submódulos serán ejecutados y, además, estructurar y organizar mejor la tarea asignando a cada módulo una función



5- Estructura de programa

Los autómatas incorporan todo un conjunto de funciones sencillas que cubren un gran abanico de operaciones elementales como por ejemplo los temporizadores, contadores, operaciones aritméticas, comparadores, etc. Se trata de bloques sencillos a partir de los cuales, combinándolos entre ellos, se pueden conseguir funcionalidades muchos mas complejas.

Dentro del *software* de programación de los autómatas hay módulos funcionales mucho más complejos que se utilizan como ayuda para la programación de tareas muy específicas: por ejemplo, podemos encontrar módulos para el control de servomotores, controladores de temperatura, controladores PID, etc.

Estos bloques ya están programados por el fabricante del PLC y suelen tener una interficie de datos compleja con un gran numero de entradas / salidas.

5- Estructura de programa

Ejemplo de Bloque para la programación de procesos PID