APUNTES GENERALES DE M.M.S.A.P.

AUTOMATAS PROGRAMABLES SIEMENS S7 – 300

DPTO. DE MANTENIMIENTO - C.I.P. ETI

INDICE

Págs.

1 – INTRODUCCION - SOFTWARE STEP - 7

Hay gran cantidad de manuales y documentación del STEP-7 de Siemens. Con el propio programa de instalación vienen 15 o 20 manuales electrónicos en PDF y después una vez instalado el programa, tenemos un menú de Ayuda muy extenso y la ayuda contextual (F1). El problema de documentación referente al S-7 por lo tanto, no es la falta de documentación; al contrario; puede ser que hay mucha, en varios manuales y varias versiones y casi lo mismo podemos decir para el Hardware y el Software. Debido a esto ultimo, estos apuntes, son solo un breve resumen para el inicio a la programación con STEP-7.

STEP 7 es el software estándar para configurar y programar los sistemas de automatización SIMATIC. STEP 7 forma parte del software industrial SIMATIC. El software estándar STEP 7 presenta las siguientes variantes:

- STEP 7-Micro/DOS y STEP 7-Micro/WIN para aplicaciones independientes sencillas en sistemas de automatización SIMATIC S7-200. Los S7-200, digamos que son los autómatas de gama baja; los S7-300 los de gama media y los S7-400 los de gama alta.
- STEP 7 para aplicaciones en sistemas de automatización SIMATIC S7-300/400, SIMATIC M7-300/400 y SIMATIC C7 con funciones ampliadas y ampliable con los productos de software opcionales integrados en el Software Industrial SIMATIC.

Es fácil de instalar por que además de autoarranque, tiene un asistente de instalación bueno. Actualmente el Software viene en uno o dos CDs, con varios lenguajes de programación y un Simulador (el PLC-SIM); la licencia o licencias van en disquetes.

2 - LENGUAJES DE PROGRAMACIÓN

Lenguajes de programación usuales

Los lenguajes de programación KOP, AWL y FUP para S7-300/400 son parte integrante del software estándar.

- KOP (esquema de contactos) es un lenguaje de programación gráfico. La sintaxis de las instrucciones es similar a la de un esquema de circuitos. KOP permite observar la circulación de la corriente a través de contactos, elementos complejos y bobinas.
- AWL (lista de instrucciones) es un lenguaje de programación textual orientado a la máquina. En un programa creado en AWL, las instrucciones equivalen en gran medida a los pasos con los que la CPU ejecuta el programa. Para facilitar la programación, AWL se ha ampliado con estructuras de lenguajes de alto nivel (tales como accesos estructurados a datos y parámetros de bloques).
- FUP (diagrama de funciones) es un lenguaje de programación gráfico que utiliza los cuadros del álgebra booleana para representar la lógica. Asimismo, permite representar funciones complejas (p.ej. funciones matemáticas) mediante cuadros lógicos.

Lenguajes de alto nivel

Para programar los sistemas de automatización SIMATIC S7-300/400 se dispone de los siguientes lenguajes opcionales:

- S7-GRAPH es un lenguaje de programación que permite describir cómodamente controles secuenciales (programación de cadenas secuenciales) dividiendo el proceso en diferentes etapas. Estas últimas contienen sobre todo acciones para controlar las salidas. El paso de una etapa a otra se controla mediante condiciones de transición.
- S7-HiGraph es un lenguaje de programación que permite describir cómodamente los procesos asíncronos y no secuenciales en forma de grafos de estado. Para ello se divide la

instalación en unidades funcionales que pueden adoptar diversos estados. Las unidades funcionales se pueden sincronizar mediante el intercambio de mensajes.

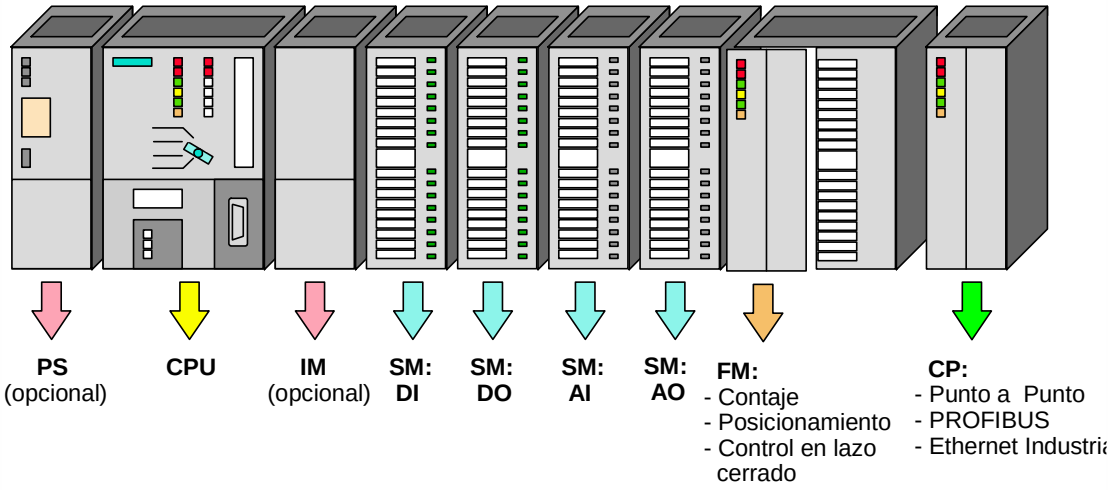
- S7-SCL es un lenguaje textual de alto nivel según la norma DIN EN 61131-3. Contiene estructuras similares a las de los lenguajes de programación Pascal y C. Por consiguiente, S7-SCL es especialmente apropiado para los usuarios que ya estén acostumbrados a utilizar lenguajes de nivel superior. S7-SCL se puede utilizar p.ej. para programar funciones repetitivas o muy complejas.

Software opcional

Hay varios: Borland C++ (sólo M7), S7-PDIAG, S7-PLCSIM, etc; con el S7-PLCSIM que es el que vamos a ver, es posible simular sistemas de automatización S7 conectados al sistema de origen (PC/PG) para someterlos a un test.

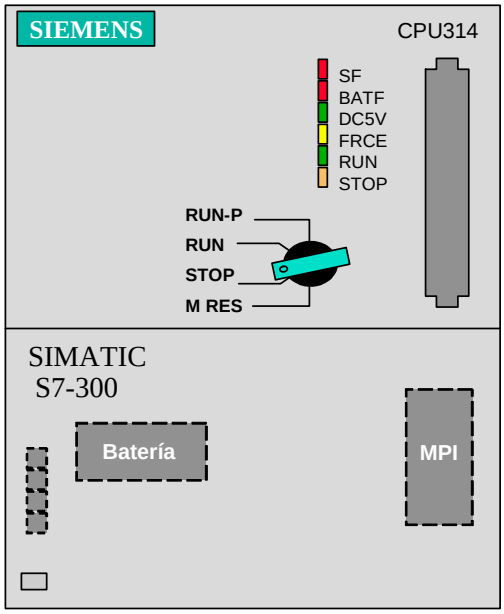
3 - AUTOMATA S7 – 300. MODULOS QUE LO COMPONEN

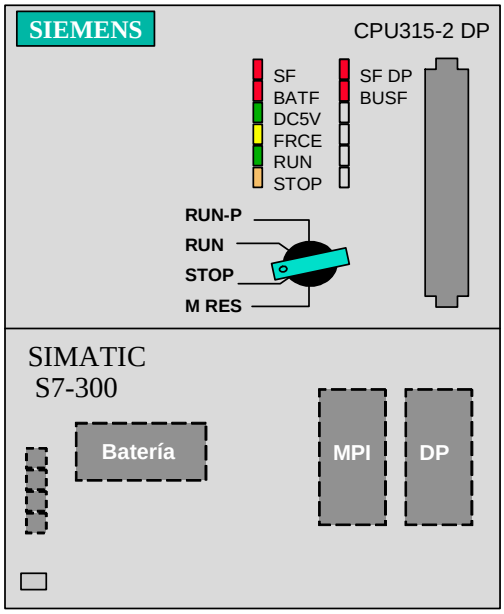
Los autómatas programables de serie media y alta en su mayoría son modulares, por lo tanto antes de empezar a programarlo hay que configurarlo con los elementos que lo van a componer. Los elementos principales de un PLC son:



- **Fuente de alimentación (PS).**
- **Unidad central de proceso (CPU)**
- **Módulos de Interfase (IM).-** La IM360/IM361 y la IM365 hacen posible configuraciones multifila. Enlazan el bus a través de una fila a la siguiente.
- **Módulos de Señal (SM).-** Pueden ser: Módulos de entradas digitales de 24V DC, 120/230V AC**;** Módulos de salidas digitales a 24V DC, a Relé**;** Módulos de entradas analógicas configurables como: Tensión, corriente, resistencia, termopares y Módulos de salidas analógicas: configurables como: Tensión, corriente.
- **Módulos de Función (FM).-** Realizan "funciones especiales": Contaje Posicionamiento**,** Control en lazo cerrado.
- **Procesadores de Comunicación (CP).-** Proporciona las siguientes posibilidades de montaje en red: Enlaces Punto a Punto, PROFIBUS, Industrial Ethernet.
- **Accesorios:** Cables, Conectores de bus y conectores frontales...

CARATULA-DISEÑO DE LA CPU





En las figuras anteriores, tenemos las carátulas de dos CPUs distintas, en ellas podemos distinguir:

Selector de Modo

- MRES = Función de reset de memoria (Module Reset).
- STOP = Modo Stop; el programa no se ejecuta.
- RUN = Ejecución del programa, es posible el acceso sólo lectura desde la PG.
- RUN-P = Ejecución del programa, es posible el acceso lectura/escritura desde la PG

Indicadores de estado (leds)

- SF = Error de grupo; fallo interno de la CPU o fallo en un módulo con capacidad de diagnóstico.
- BATF = Fallo de batería; Batería vacía o no presente.
- DC5V = Indicador de tensión interna de 5 V DC.
- FRCE = FORCE; indica que, al menos, una entrada o salida está forzada.
- RUN = Parpadea cuando la CPU está arrancando, luce fija en modo RUN.
- STOP = Muestra una luz fija en modoSTOP
  Parpadea lentamente durante una petición de reset de memoria,
  Parpadea rápidamente cuando se está efectuando un reset de memoria,
  Parpadea lentamente cuando se necesita un reset porque se ha insertado una memory card.

Memory Card

Existe un slot para una memory card . La memory card salva el contenido del programa en caso de caída de alimentación sin necesidad de batería. Hay CPUs que no llevan memory card y otras nuevas que ahora llevan Tarjeta SD

Compartimento de la batería

Existe un receptáculo para una batería de litio bajo la cubierta. La batería proporciona energía de respaldo para salvar los contenidos de la RAM ante una supuesta caída de alimentación.

Conexión MPI

Conexión para un dispositivo de programación u otro dispositivo con interfase MPI.

Interfase DP

Interfase para la conexión directa de periferia descentralizada a la CPU.

Datos tecnicos de algunas CPUs S7 - 300

A continuacion tenemos un cuadro con algunos datos técnicos de distintas CPUs; como dentro de las CPUs, tambien hay versiones, estos datos varian entre las distintas versiones. Para hacernos una idea aproximada, con la CPU – 314 podemos programar unas 8 000 instrucciones.

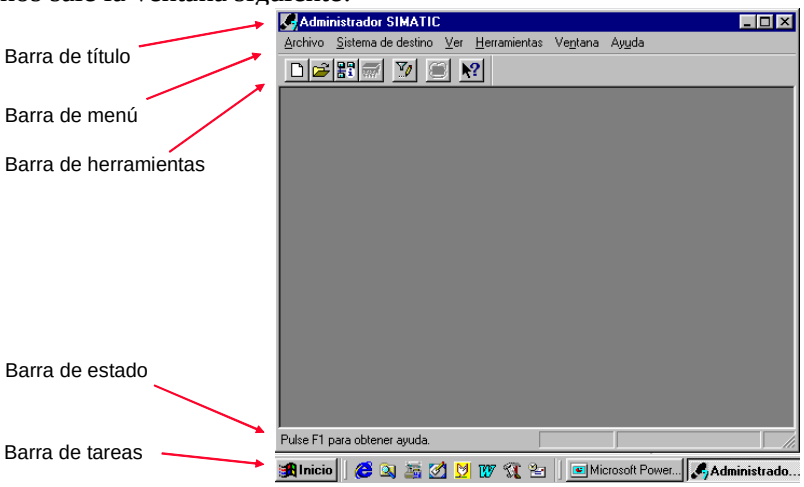
CPU	312 IFM	313	314	314 IFM	315	315-2 DP	316-2 DP	318-2 DP
Tiempo de ejec. en µs								
Instrucciones de bit	0.6 - 1.2	0.6 - 1.2	0.3 - 0.6	0.3 - 0.6	0.3 - 0.6	0.3 - 0.6	0.3 - 0.6	0.1
Instrucción de Palabra	2.0	2.0	1.2	1.2	1.0	1.0	1.0	0.1
Entero (+/-)	3.0	3.0	2.0	2.0	2.0	2.0	2.0	0.1
Real (+/-)	60.0	60.0	50.0	50.0	50.0	50.0	50.0	0.6
Memoria de Usuario								
Memoria de Trabajo	6 KB	12 KB	24 KB	32 KB	48 KB	64 KB	128 KB	512 KB
Memoria de carga int.	20 KB	20 KB	40 KB	48 KB	80 KB	96 KB	192 KB	64 KB
Memoria de carga ext.	-	4 MB	4 MB	(4 MB)	4 MB	4 MB	4 MB	4 MB
Direcciones								
Marcas	1024	2048	2048	2048	2048	2048	2048	8192
Marcas de Ciclo	8	8	8	8	8	8	8	8
Temporizadores	64	128	128	128	128	128	128	512
Contadores	32	64	64	64	64	64	64	512
Nº/Tipos de Bloque								
FBs	32	128	128	128	192	192	256	1024
FCs	32	128	128	128	192	192	512	1024
DB's	63	127	127	127	255	255	511	2047
Tamaño imagen proceso I/O en bytes	32 cada una	128 cada una	128 cada una	124 cada una	128 cada una	128 cada una	128 cada una	256 cada una (2048)
Área de direcc. max. I/O en bytes	32 cada una	32 cada una	768 cada una	752 cada una	768 cada una	1024 cada una	1024 cada una	8192 cada una
Interfases	MPI	MPI	MPI	MPI	MPI	MPI, DP	MPI, DP	MPI/DP, DP

4 - CONFUGURACION DEL SIEMENS S7 – 300

El software de programación de Step-7, tiene la misma estructura y manejo que la mayoría de los programas con sistema operativo Windows.

El administrador Simatic

Una vez instalado el programa STEP-7, nos aparece en el escritorio el icono del programa con el nombre de Administrador Simatic; con doble Clic en él, entramos en el programa y nos sale un asistente para crear un nuevo proyecto sencillo. Para proyectos mas complejos, salimos del asistente y nos sale la ventana siguiente:



En SIMATIC S7 todos los requerimientos hardware y software de un proceso de automatización se manejan dentro de un proyecto. Al examinar un proyecto que queramos automatizar, encontraremos que está constituido por multitud de secciones y subprocesos más pequeños, que están interrelacionados y dependen unos de otros. La primera tarea es, por tanto,

dividir el proceso en subtareas más sencillas.
Cada subtarea define ciertos requerimientos hardware y software que debe cumplir el sistema de automatización:
Hardware:

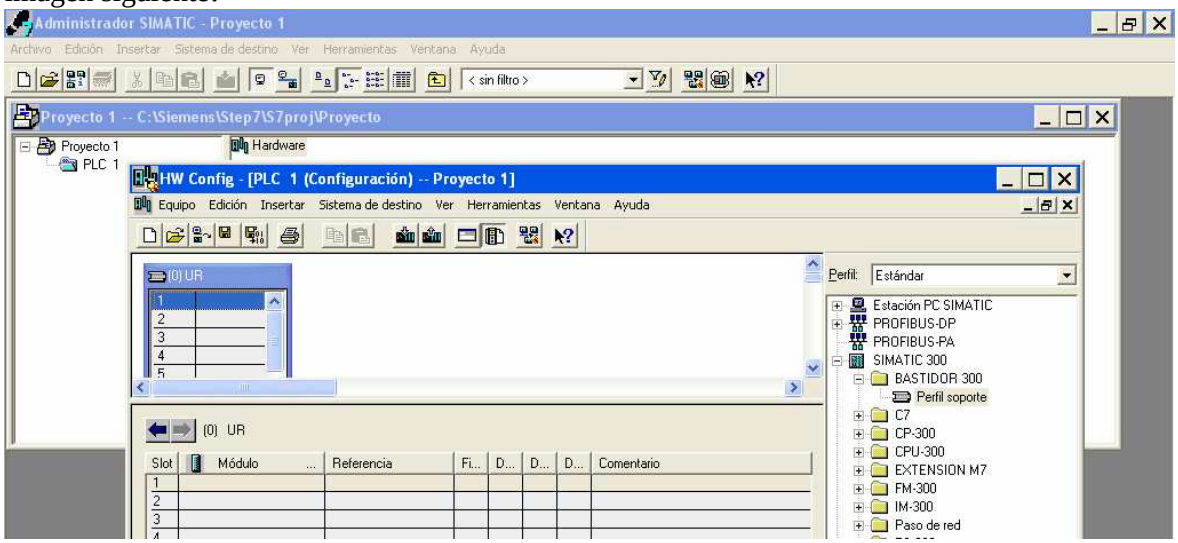
- Número y tipo de entradas y salidas.
- Número y tipo de módulos - Número de bastidores.
- Capacidad y tipo de CPU.
- Sistemas HMI (OPs, etc)
- Sistemas de comunicación en red

Software:

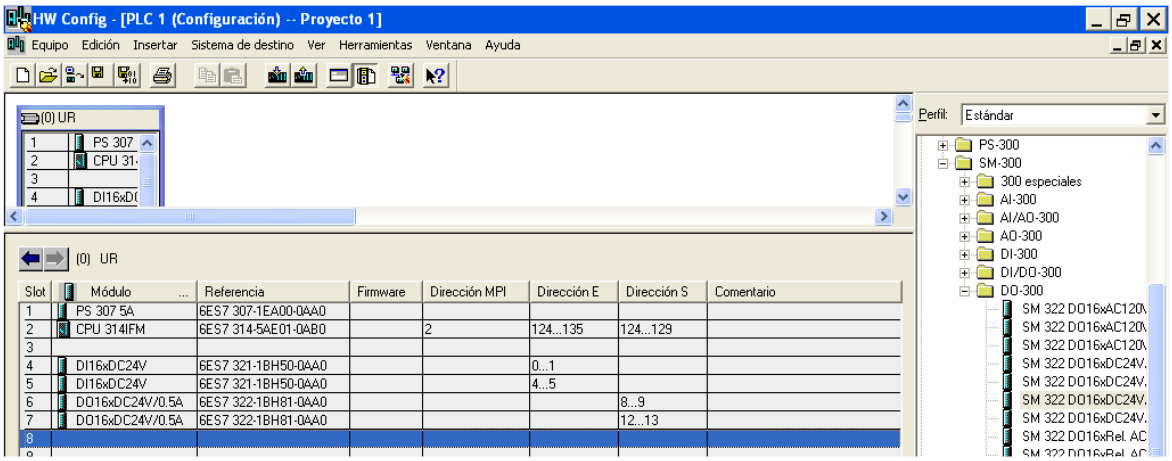
- Estructura del programa.
- Manejo de datos para el proceso de automatización.
- Datos de configuración.
- Datos de comunicación.
- Documentación del proyecto y del programa.

Creación de un Proyecto

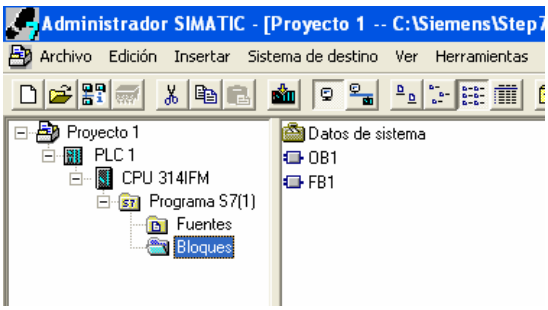
A partir de la ventana del Administrador Simatic, seleccionar la opción del menú *Archivo -> Nuevo,* o el icono "nuevo" en la barra de herramientas, así abrimos el cuadro de diálogo para la creación de un nuevo proyecto o una nueva librería. Las librerias se usan para guardar trozos o bloques de programa estandar, que despues podemos insertar en nuestro proyecto. Step-7 tiene varios bloques estandar, alguno de los cuales vamos a usar. Introducir el nombre del proyecto, "Proyecto 1" en este caso y confirmar haciendo click en el botón "Aceptar". Despues con *Insertar -> Equipo -> Simatic 300 ->Nombre* (PLC 1, en este caso), ya tenemos un equipo; con *click en este equipo* (PLC 1) y despues *doble click en Hardware*, entramos en el **HW Config**, que es donde vamos a configurar el Hardware de nuestro PLC. En la nueva ventana sale una subventana con el menú *catalogo* de Hardware, si no saliese, seleccionar la opción de menú *Ver -> Catálogo* o hacer click en el icono correspondiente de la barra de herramientas. (Si hemos seleccionado el perfil de catálogo "Estándar", aparecen todos los bastidores, módulos y módulos de interfase en la ventana "Catálogo Hardware". Puede crear sus propios perfiles de catálogo con los elementos que use más frecuentemente seleccionando la opción de menú *Herramientas -> Editar perfiles de catálogo).* Estando en esta nueva ventana y desplegando *SIMATIC 300 -> BASTIDOR 300 -> Perfil soporte (doble click),* nos encontramos ya en la imagen siguiente:

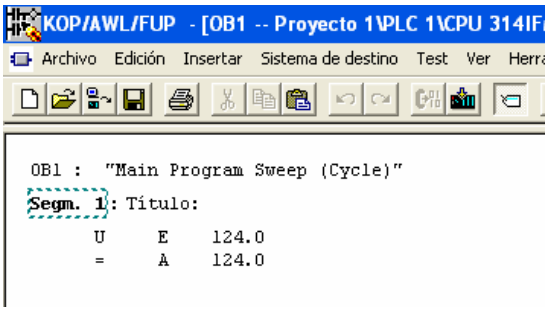


Desplegando las pestañas: PS, CPU y SM vamos insertando en el bastidor la: F.A., CPU y entradas y salidas que tengamos que configurar. Según vamos seleccionando cada una de las opciones, nos aparece debajo informacion con la referencia y caracteristicas de ese elemento concreto. En este ejemplo el slot 3 queda libre por que no hay interfases IM, y hemos configurado una F.A. de 5 A; una CPU 314-IFM con entradas y salidas propias en las direcciones 124 y siguientes; dos tarjetas de 16 Entradas cada una con las direccciones 0, 1 y 4, 5; y otras dos tarjetas de Salidas con las direcciones 8, 9 y 12, 13 respectivamente. Hay CPUs como las 315-2DP que dejan elegir las direcciones; esta 314-IFM, nos las ha asignado directamente y no existe la posibilidad de poderlas cambiar.



Si guardamos y compilamos, ya tenemos configurado el Hardware. Saliendo del HW Config vamos otra vez al Administrador Simatic, aquí, si desplegamos la estructura de árbol del Proyecto (ventana de la izquierda) y seleccionamos los Bloques con: *Insertar -> Bloque S7*, ademas del OB1 que ya sale por defecto, podemos insertar los bloques de programacion que necesitemos, en este caso hemos insertado un FB. Entrando en los bloques, podemos empezar a





programar, y una vez programado todo, cargaremos **todos** los bloques, bien con el icono de cargar (el de la flecha), o bien a traves del menú: *sistema de destino -> cargar*.

El icono de la "nube" (activar/desactivar simulacion) sale si tenemos cargado el programa S7 - PLCSIM; si este icono esta activado, el programa se carga en el simulador, si no lo esta y el PC esta conectado con el PLC entonces se carga en el PLC real.

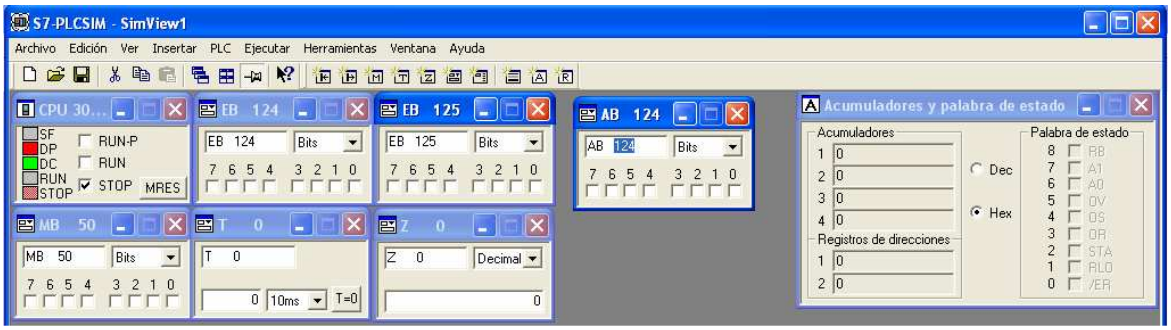

S7wnfwlx.exe


S7wsvapx.exe

5 – VER / SIMULAR EL PROGRAMA. SIMULADOR S7 – PLCSIM

Activando el Icono Observar si/no (representa a unas gafas), podemos ver como se van activando

partes del programa que se esta ejecutando. También podemos crear una tabla de variables con el menú: *sistema de destino -> observar/forzar variable,* y ver los cambios y valores actuales de estas variables. Esto lo podemos hacer tanto si se esta ejecutando el programa en el PLC, como si lo estamos probando en el simulador.
El funcionamiento del simulador es muy intuitivo, cuando activamos su icono, se activa la ventana del simulador y ya sale el simulador de la CPU. En funcion de lo que necesitemos, tambien podemos activar (Insertar) pequeñas ventanas de: Entradas, Salidas, Marcas, Temporizadores, Contadores, Acumuladores, etc. Es muy adecuado para probar programas o partes de programas, con la ventaja de que podemos ver el funcionamiento y los valores de: bits, Bytes, Palabras, contadores, acumuladores, etc. El funcionamiento de los temporizadores se realiza bien con la unica desventaja de que el tiempo, no es real.



6 - ESTRUCTURA DE UNA INSTRUCCIÓN

Instrucción.- Se entiende por instrucción, cada una de las líneas de un programa. Son instrucciones: >=I, ITB, SPB, U E 124.0, L S5T#5S, T MW 30, S "Motor1"...

Tipos de instrucciones.- Hay dos tipos, las que constan solo de una operación (SPB, BEB, NOT, ...) y las que constan de una operación y de un operando.

Operación.- La operación es la parte de la instrucción que especifica lo que tiene que hacer el procesador. Son operaciones: U, O, L, CALL, NOT, SE,...

Operando.- El operando especifica con que tiene que operar el procesador. El operando se puede direccionar de forma absoluta o de forma simbólica. Son operandos: E 124.0, S5T#3MS, MW 10, B#16#6F, "marcha",...

Identificador y dirección.- El operando a su vez puede constar de un identifícador que especifica el área de memoria con la que se trabaja y a veces su tamaño. La dirección indica la dirección del bit o byte del área de memoria especificada en el identificador.

INSTRUCCION		
Operación	**Operando** (direccionamiento)	
=	**A**	**124.0**
	Identificador	Dirección

Direccionamiento.- El direccionamiento de un operando se puede hacer de forma absoluta (U E 124.5) o de forma simbólica (U "marcha"). A su vez en la forma simbólica, el **símbolo** puede ser de dos tipos:

- **Global,** que se define en la tabla de símbolos al asignarle a cada dirección absoluta una dirección simbólica, vale para todos los bloques del programa, y step-7 lo pone entre comillas: "marcha", "tiempol", "A+", "fc_al"...
- **Local,** que se define en la tabla de declaración de variables de los bloques, se emplean en sus bloques respectivos y step-7 los escribe con almohadilla delante: #marchal, #condición; estos símbolos también se usan cuando en un bloque se hace una llamada a otro bloque, en el bloque que hace la llamada aparecen estos símbolos sin ningún distintivo y se ofrecen como parámetros formales que serán sustituidos por parámetros actuales en el bloque que los llama.

7 – ESTRUCTURA DEL PROGRAMA

Programa Lineal .- Todo el programa se encuentra en un modulo (OB1) con todas las instrucciones juntas. Este modelo se asemeja a un esquema de relés, que se reemplaza por un controlador lógico programable. La CPU procesa las instrucciones individuales y da un resultado de entradas y de salidas en cada ciclo de scan.

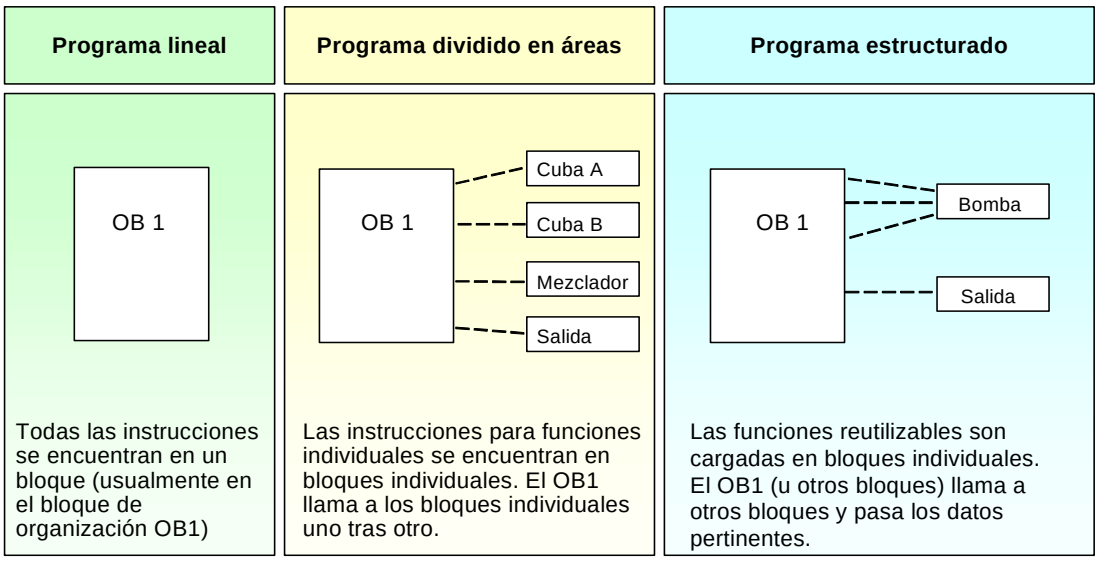
Programa Dividido.- El programa está dividido en bloques, cada bloque solo contiene el programa para resolver una tarea parcial. Es posible dividir aún más el programa en segmentos dentro de un bloque. Podemos generar plantillas de segmento para segmentos del mismo tipo.

El bloque de organización OB 1 contiene instrucciones que llaman a los otros bloques en una secuencia definida.

Programa Estructurado.- Un programa estructurado contiene bloques con parámetros, llamados bloques parametrizables. Estos bloques se diseñan para que puedan usarse de forma universal. Cuando llamamos a un bloque parametrizable, le damos los parámetros actuales (las direcciones exactas de entradas y salidas así como los valores de los parámetros).

Ejemplo:

- Un “bloque bomba" contiene instrucciones para el control de una bomba.
- Los bloques de programa, que son responsables del control de bombas especiales, llaman al “bloque bomba“ y le dan información sobre qué bomba va a ser controlada y con qué parámetros.
- Cuando el “bloque bomba" haya completado la ejecución de sus instrucciones, el programa retorna al bloque que ha realizado la llamada (por ej. el OB 1), que continua con el procesamiento de sus instrucciones.



8 – TIPOS DE BLOQUES DE PROGRAMA

El controlador lógico programable proporciona varios tipos de bloques, donde podemos almacenar el programa de usuario y demás datos relacionados. Dependiendo de los requerimientos del proceso, el programa puede ser estructurado en bloques diferentes.

Bloques de Organización (OB‘s)

Los bloques de organización (OB‘s) constituyen la interfase entre el sistema operativo del PLC y el programa de usuario. El programa completo puede almacenarse en el OB, que es ejecutado cíclicamente por el sistema operativo (programa lineal) o puede dividirse y almacenarse en distintos bloques (programa estructurado).

Funciones FC, SFC

Una función (FC) contiene parte de la funcionalidad del programa. Es posible programar funciones a las que se les pueda asignar parámetros. Cómo resultado, las funciones también se pueden utilizar para tareas repetitivas o funcionalidades complejas tales como cálculos.
Las Funciones de Sistema (SFC) son funciones integradas en el sistema operativo de la CPU; en

Bloques de función FB, SFB

Básicamente, los bloques de función ofrecen la misma funcionalidad que las funciones. La diferencia radica en que los bloques de función poseen su propia área de memoria en forma de bloques de datos de instancia. Como resultado, los bloques de función están concebidos para tareas muy repetitivas o funcionalidades complejas, como tareas de control en lazo cerrado.
Los Bloques de Función de Sistema (SFB) son funciones parametrizables integradas en el sistema operativo de la CPU..

Bloques de datos DB

Los bloques de datos (DB) son áreas de datos del programa de usuario en las que los datos son distribuidos de forma estructurada.

Dentro del programa Step-7, en “elementos de programa“ -- “librerias“, podemos ver e insertar bloques FCs, SFCs, FBs y SFBs ya programados genericamente; seleccionando uno en concreto y pulsando la ayuda F1 obtenemos informacion.

9 - CICLO DE LA CPU

Introducción

La CPU comprueba el estado de las entradas y las salidas en cada ciclo. Existen áreas de memoria específicas en las que se almacenan los datos binarios de los módulos: la PAE y la PAA. El programa accede a estos registros durante el procesamiento.

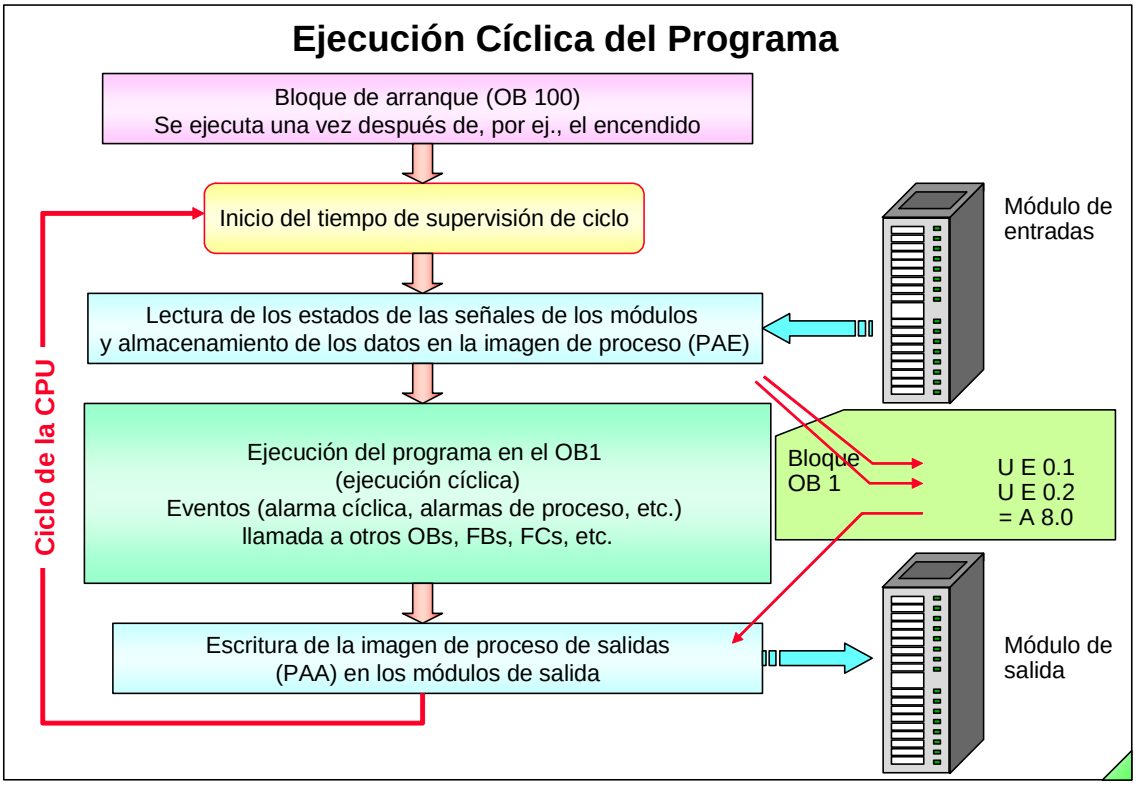
PAE

La tabla de imagen de proceso de entrada se encuentra en el área de memoria de la CPU. Allí se almacena el estado de señal de todas las entradas.

PAA

La tabla de imagen de proceso de salida contiene los valores de salida resultantes de la

ejecución del programa. Estos se envían a las salidas reales (A) al final del ciclo



Programa de Usuario

Cuando comprobamos las entradas en el programa de usuario, por ejemplo con la instrucción U E 2.0, se evalua el último estado de señal desde la PAE. Esto garantiza la llegada del mismo estado de señal cuando realizamos consultas múltiples de la entrada dentro de un ciclo.

Arranque

La CPU lleva a cabo un rearranque completo (con el OB100) cuando alimentamos o cuando pasamos de STOP --> RUN. Durante un rearranque completo, el sistema operativo borra las marcas, temporizadores y contadores no remanentes, borra la pila de interrupciones y la pila de bloques, resetea todas las alarmas de proceso y diagnóstico almacenadas e inicia el tiempo de vigilancia del ciclo.

Ciclo de Scan

El funcionamiento cíclico de la CPU se compone de tres secciones principales, como se muestra en el diagrama de arriba:

- La CPU comprueba el estado de las señales de entrada y actualiza la tabla de imagen de proceso de entrada.
- Ejecuta el programa de usuario con sus respectivas instrucciones.
- Escribe los valores de la tabla de imagen de proceso de salida en los módulos de salidas.

10 - COMBINACIONES BINARIAS

Las operaciones lógicas con bits operan con dos dígitos, 1 y 0. Estos dos dígitos constituyen la base de un sistema numérico denominado sistema binario. Los dos dígitos 1 y 0 se denominan dígitos binarios o bits. En el ámbito de los contactos y bobinas, un 1 significa activado (”conductor”) y un 0 significa desactivado (”no conductor”).

Las operaciones lógicas con bits interpretan los estados de señal 1 y 0, y los combinan de acuerdo con la lógica de Boole. Estas combinaciones producen un 1 ó un 0 como resultado y se denominan ”resultado lógico” (RLO). Las operaciones lógicas con bits permiten ejecutar las más diversas funciones.

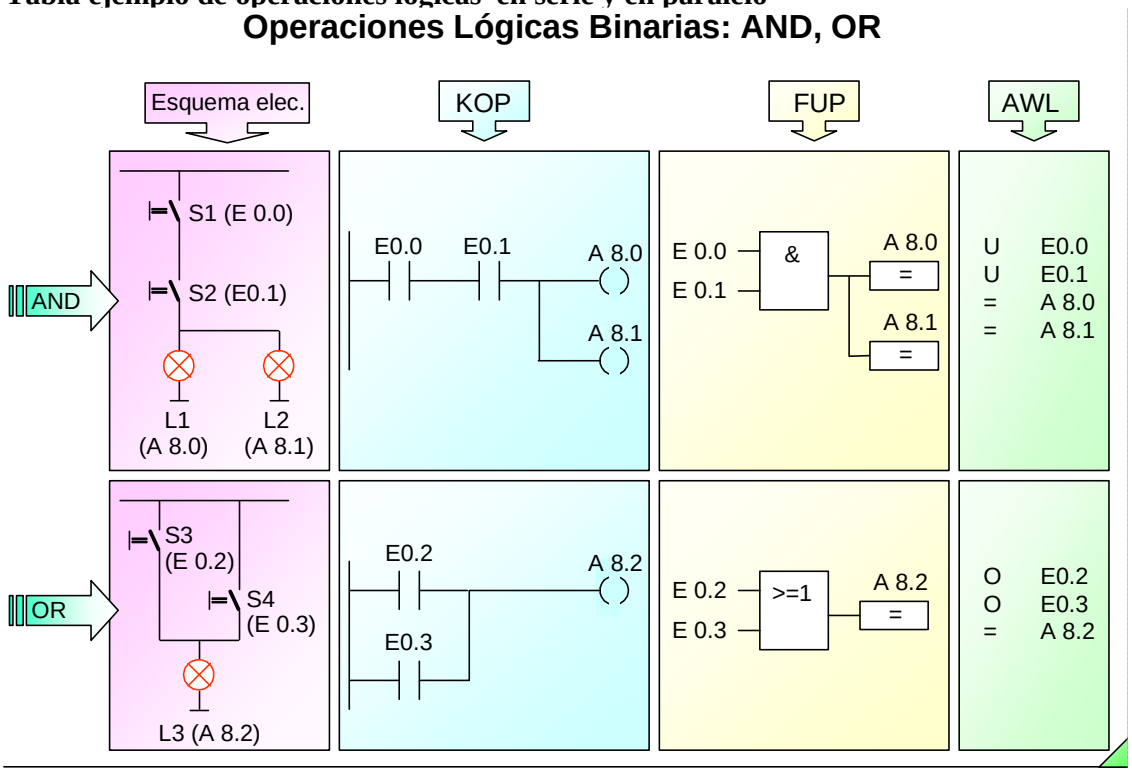
Las **operaciones básicas** para las operaciones lógicas con bits, están en la tabla resumen del final, **son**: U (Y), UN (Y-No), O (O), ON (O-No), X (O-exclusiva), XN (O-exclusiva-No). Todas estas operaciones permiten ejecutar una cadena lógica encerrada entre paréntesis.

Para terminar una cadena lógica se puede utilizar una de las tres operaciones:
· = Asignar
· R Desactivar
· S Activar

Las operaciones siguientes permiten modificar el resultado lógico (RLO):
· NOT Negar el RLO
· SET Activar el RLO (=1)
· CLR Desactivar RLO (=0)
· SAVE Memorizar el RLO en el registro RB

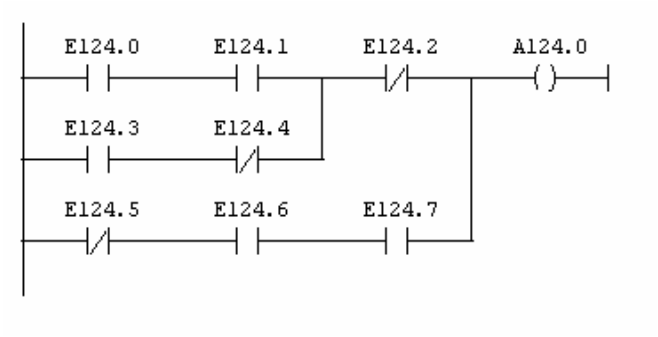
Otras operaciones detectan cambios en el resultado lógico y reaccionan correspondientemente:
· FN Flanco negativo
· FP Flanco positivo

Tabla ejemplo de operaciones lógicas en serie y en paralelo



El principio básico para programar estas operaciones, es que si programamos un contacto en serie (U), o en Paralelo (O); es en serie, o en paralelo con todo lo anterior. Si con este

principio, no es posible programar el automatismo de que se trate, hay que usar paréntesis o marcas que almacenen resultados intermedios.
Ejemplo KOP, AWL de un automatismo genérico:

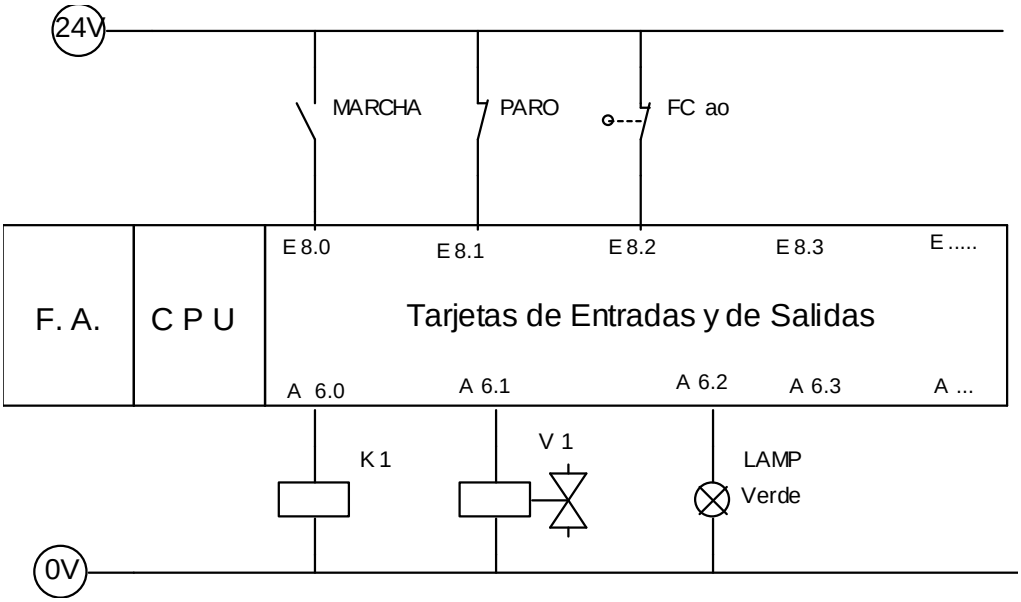



```
U(
U     E     124.0
U     E     124.1
O
U     E     124.3
UN    E     124.4
)
UN    E     124.2
O
UN    E     124.5
U     E     124.6
U     E     124.7
=     A     124.0
```

La de la derecha es la solución AWL que el propio programa ha generado, siguiendo la regla de que: "Y" antes de "O", no necesita paréntesis; pero puede haber otras soluciones de programación también validas aunque no sean convertibles a KOP

11 - CONTACTOS n.a y n.c. - CONEXIONES DE E/S AL PLC

En automatismos de cierta complejidad, la lógica cableada que se hacía antes con pulsadores, relés, temporizadores...., se hace ahora con Autómatas Programables (PLCs). Toda la lógica la lleva el programa de Autómata, y solo hay que hacer las conexiones directas de las Entradas y las Salidas, según se ve en la figura. Los 0-24 V son de la propia FA del Autómata si el consumo no es elevado; en caso contrario, las salidas del Autómata irían a relés y a través de los contactos de estos, se hace la alimentación externa de las válvulas, lámparas, etc. . Aparte de estas conexiones, cada autómata, lleva un esquema de las conexiones que hay que hacer de la F.A. a las tarjetas y otros módulos que pueda llevar el autómata.

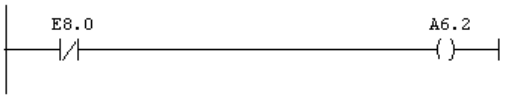


Hay "cierta" dificultad a la hora de programar lo que son contactos n.a. y n.c.. Aunque los manuales de programación (sobre todo para KOP) hablan de contactos n.a. y n.c; nosotros, vamos a considerar que son contactos n.a. y n.c los de las Entradas reales que conectamos al

autómata; y **para programar, vamos a considerar que tenemos contactos afirmados (los que se representan abiertos) y contactos negados (los que se representan cerrados o negados).**

Ejemplo.- Si tenemos las conexiones del Autómata de la figura anterior y programamos lo siguiente:

```
UN    E    8.0
=     A    6.2
```



La lámpara verde lucirá si no pulsamos la marcha. El razonamiento es el siguiente: si la marcha, no esta pulsada, no llega corriente a la E 8.0 (RLO = 0), pero como por programa, estamos negando lo que ocurre realmente, el RLO = 1 y se realiza la activacion de la A 6.2 y por lo tanto de la lámpara verde.

12 - FORMATOS Y TIPOS DE DATOS SIMPLES

Bit, byte, palabra, doble palabra

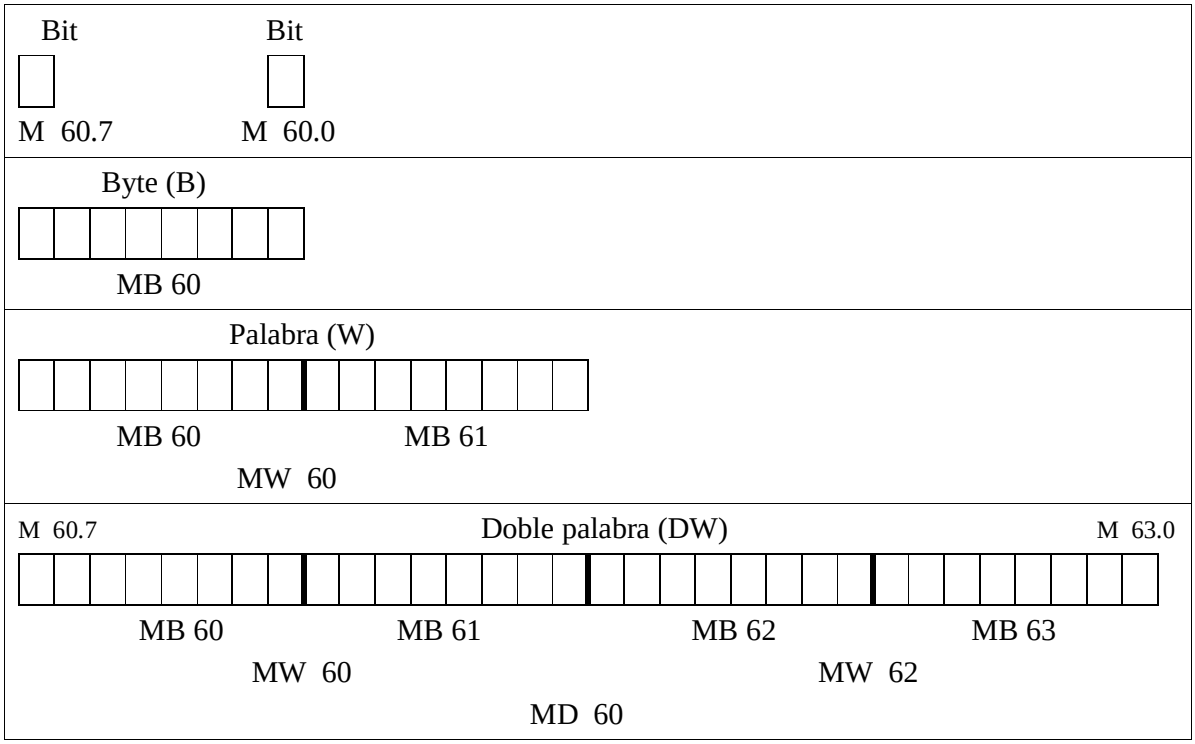
Para entender como procesa el programa el PLC, es necesario conocer previamente los conceptos de: bit, byte, palabra y doble palabra.

Un **bit** es la unidad para dígitos binarios o caracteres binarios. Es la unidad más pequeña de información. Permite los estados: "0" ó "1". Los bits pueden agruparse para formar las unidades mayores siguientes

Un **byte** es un grupo de 8 bits. Se emplea para la representación de valores numéricos de 8 dígitos binarios. También permite representar el valor numérico de 8 canales consecutivos de entradas, salidas o marcas. Representa un valor comprendido entre -128 y +127.

Una **palabra** esta compuesta por dos bytes, o 16 bits que suponen por ejemplo: 16 Entradas ó 16 salidas o 16 marcas. Su valor en entero va de -32.768 a + 32.767)

Una **doble palabra** está constituida por 2 palabras (o 4 bytes, o 32 bits). Representa un valor numérico comprendido entre -2.147.483.648 y +2.147.483.647.



Formato de datos

Los datos utilizados en un PLC, pueden ser muy diferentes, por lo que se debe seleccionar el tipo exacto de formato a utilizar. Para asignar una constante debemos dar dos características, que son la anchura del dato y el formato del dato.

Los anchos de datos se dan de la siguiente manera:	**Los formatos de datos y su representación en STEP-7 son:**
- Byte: **B#** - Palabra: **W#** - Doble palabra**: DW#** - Binario: no se especifica anchura - Entero: no se especifica anchura - Real: no se especifica anchura	- Binario: **2#** - Hexadecimal: **16#** - Doble entero: **L#** - BCD: **C#** - Decimal: **B#** - Tiempo SIMATIC: **S5T#.....** - Tiempo IEC: **T#.....** - Fecha IEC: **D#.....** - Hora:**TOD#....** - Caracteres: " " - Enteros: Sin identificador de formato - Reales: Sin identificador de formato

Ejemplo de formato de datos para una palabra:

- Binario, iría desde: 2#0 hasta 2#1111 1111 1111 1111
- Hexadecimal, iría desde: W#16#0 hasta W#16#FFFF
- BCD, iría desde: C#0 hasta C#999
- Entero, iría desde: -32768 hasta 32767
- Decimal, (bytes sin signo) iría desde: B#(0, 0) hasta B#(255, 255)

13 –TEMPORIZADORES

Área de memoria y componentes de un temporizador.- Los temporizadores tienen un área reservada en la memoria de la CPU. Esta área de memoria reserva una palabra de 16 bits para cada operando de temporizador. Los bits 15 y 16 no tienen significación, los bits 12 y 13 indican la base de tiempo en que va a trabajar el temporizador y el resto de bits, almacenan (cuentan) el tiempo.

Valor de temporización.- Podemos ver los valores de temporización en binario o en BCD (formato decimal codificado en binario: cada grupo de cuatro bits contiene el código binario de un valor decimal). Los bits 0 a 9 de la palabra de temporización contienen el valor de temporización en código binario. Para BCD, son los bits 0 a 11 de la palabra de temporización los que almacenan el valor de temporización. El arranque o la actualización del temporizador, decrementa el valor de temporización en una unidad y en el intervalo indicado por la base de tiempo hasta alcanzar el valor 0. El valor de temporización se puede cargar en los formatos binario, hexadecimal o decimal codificado en binario (BCD). El valor de temporización puede ir de 0 a 9 990 segundos.

Lista de operaciones de temporización.- Se dispone de las operaciones de temporización siguientes:

- L Cargar valor actual del temporizador en ACU 1 como entero
- LC Cargar el valor actual de temporización en ACU 1 como número BCD

- R Desactivar temporizador
- SI Temporizador como impulso
- SV Temporizador como impulso prolongado
- SE Temporizador como retardo a la conexión
- SS Temporizador como retardo a la conexión con memoria
- SA Temporizador como retardo a la desconexión.
- FR Habilitar temporizador. El FR solo sirve para reiniciar un temporizador si tenemos a 1 la entrada de arranque y esta funcionando el temporizador. Idem para contadores (cuenta si esta a 1 la entrada de contar).

Base de tiempo.- Los bits 12 y 13 de la palabra de temporización contienen la base de tiempo en código binario.
- 10 ms , bits 13-12 0-0.
- 100 ms, bits 13-12 0-1.
- 1 s , bits 13-12 1-0.
- 10 s , bits 13-12 1-1.

La base de tiempo define el intervalo en que se decrementa en una unidad el valor de temporización. La base de tiempo más pequeña es 10 ms, la más grande 10 s.
El valor de temporización se puede cargar en cualesquiera de los siguientes formatos:

- **w#16#wxyz** siendo: w= la base de tiempo (es decir, intervalo de tiempo o resolución) y xyz = el valor de temporización en formato BCD.

- **S5T#aH_bM_cS_dMS** siendo: H (horas), M (minutos), S (segundos), MS (milisegundos); a, b, c, d los define el usuario**.** La base de tiempo se selecciona automáticamente y el valor de temporización se redondea al próximo número inferior con esa base de tiempo. Este es el formato mas fácil de usar y el que mas se emplea. El valor de temporización máximo que puede introducirse es de 9 900 segundos ó 2H_46M_30S.

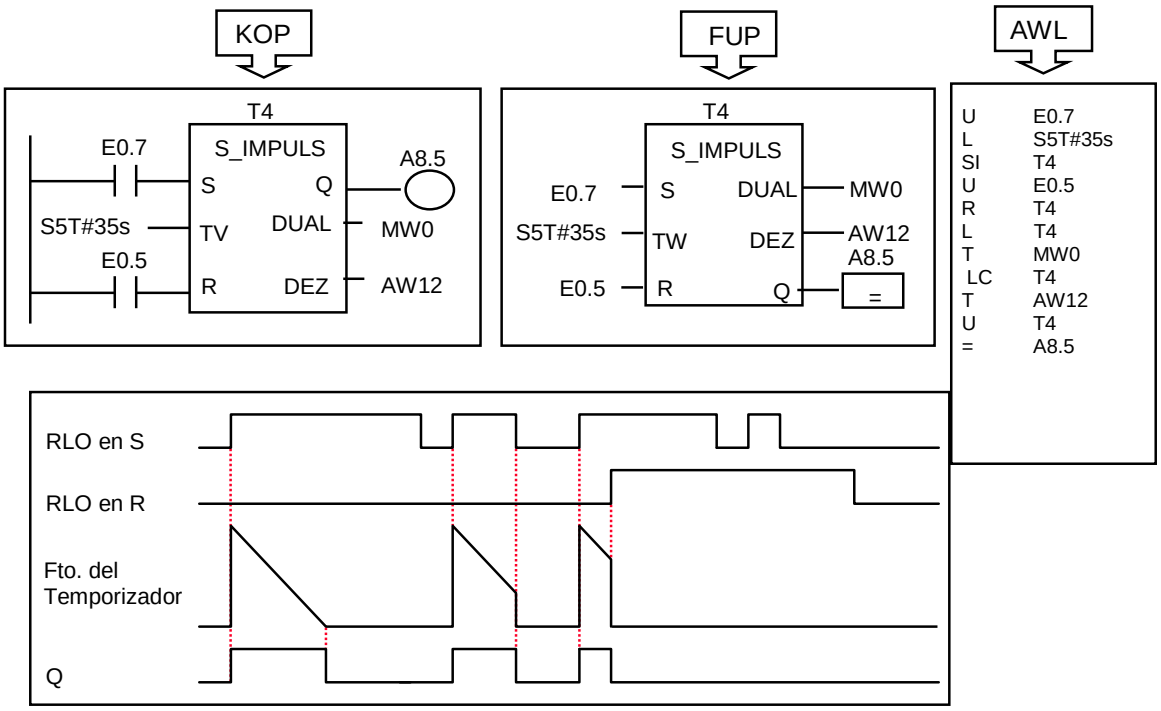
Ejemplo de los dos formatos en AWL

```
U   E   124.0          U   E   124.1
L   S5T#9S             L   W#16#1090
SE  T   1              SE  T   0
```

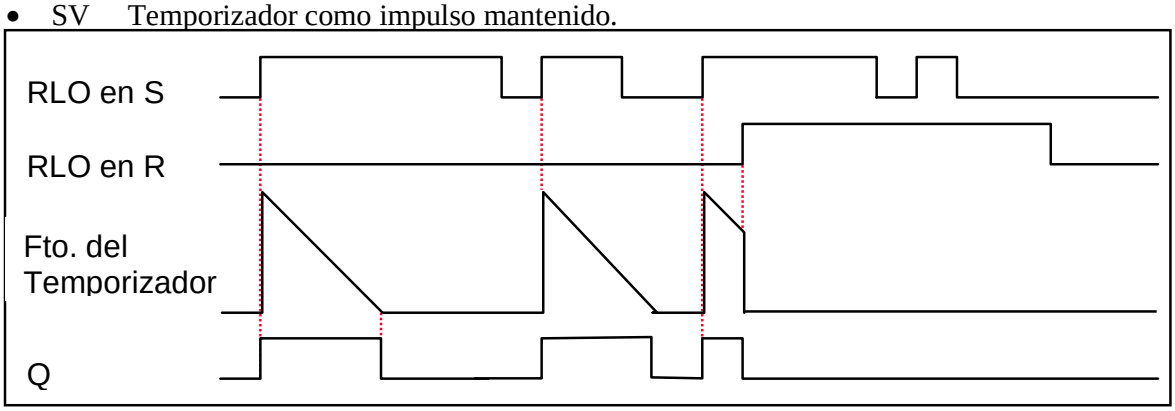
El temporizador trabaja con el dato que tenga el acumulador, luego el L puede estar antes del U y funcionaria igual o podemos programar solo U E 124.4 / SE T 4 y el Temporizador funcionaria con el valor que tenga el ACU si es valido.
Para programar temporizadores u otro tipo de elementos en KOP o en FUP, previamente con el menú: *ver -> KOP o FUP*, elegimos la opción y después insertamos un segmento de programación, con su icono o con el menú: *insertar*; además si estamos en KOP, pinchamos-seleccionamos con el ratón la línea de programa del segmento. Después, sacamos la subventana de "elementos de programa" con su icono o con el menú: *insertar -> elementos de programa*; esta ventana varia un poco si estamos en KOP o en FUP; en ella y dentro de los temporizadores, podemos elegir varias opciones; para insertar-programar cualquiera de ellas, o bien hacemos doble clic con el ratón o la arrastramos hasta el segmento; a partir de aquí, podemos programar lo que se necesite, dentro de las posibilidades que tiene cada opción. Como normalmente no se programan todas las posibilidades, es mas rápido programar los temporizadores en AWL.
En la primera figura del apartado siguiente, esta programado el mismo temporizador completo en: KOP, FUP y AWL.

Comportamiento de los distintos tipos de temporizadores:

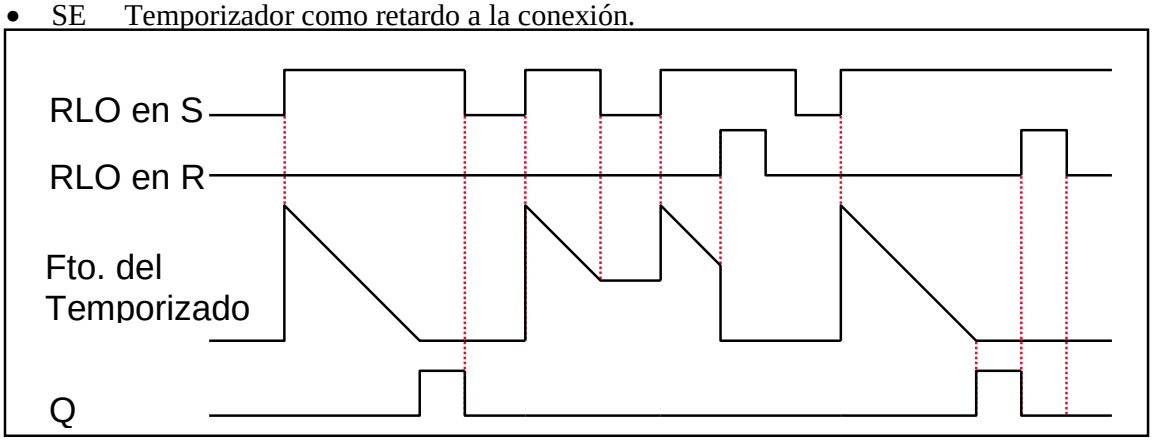
- SI Temporizador como impulso.



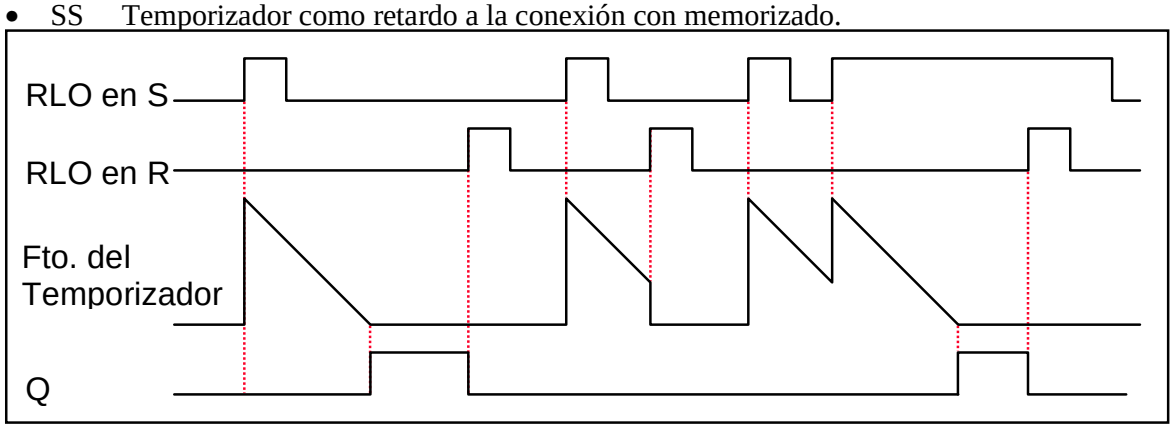
- SV Temporizador como impulso mantenido.



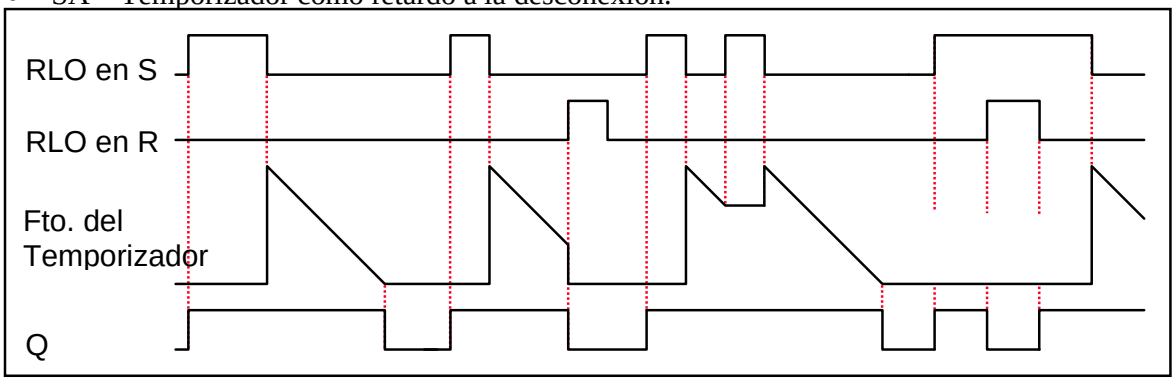
- SE Temporizador como retardo a la conexión.



- SS Temporizador como retardo a la conexión con memorizado.



- SA Temporizador como retardo a la desconexión.



14 – CONTADORES

Descripción y programación.

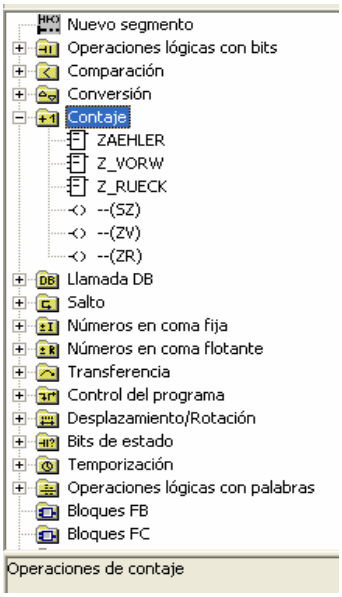
Un contador es un elemento funcional del software de programación STEP 7. Los contadores tienen reservada un área de memoria propia en la CPU. Dicha área de memoria reserva una palabra de 16 bits para cada contador. La programación con AWL asiste un máximo de 256 contadores. En los datos técnicos de la CPU encontrará la cantidad de contadores de que puede disponer.

Las operaciones de contaje son las únicas funciones que tienen acceso al área de memoria reservada para contadores.

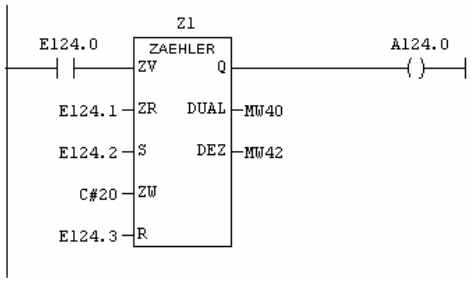
Se dispone de las operaciones de contaje siguientes:

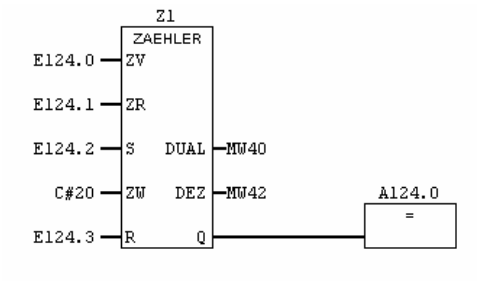
- L Cargar valor actual del contador en ACU 1 en forma de entero
- LC Cargar valor actual del contador en ACU 1 como número BCD
- R Desactivar contador
- S Poner contador al valor inicial
- ZV Incrementar contador
- ZR Decrementar contador
- FR Habilitar contador. El FR solo sirve para contar si esta a 1 la entrada de contar.

Dentro de los temporizadores hemos visto la programación en KOP y FUP. Para programar contadores en KOP y FUP, seguimos los mismos pasos, (*ver -> KOP o FUP* / insertar segmento / "elementos de programa" / insertar el contador elegido). En la figura de la derecha, vemos la subventana "elementos de programa" con el menú de las opciones de programación de contadores desplegado.



En el ejemplo siguiente, hemos insertado la opción ZAEHLER, que es la que tiene todas las posibilidades de un contador. Cambiando entre KOP, FUP y AWL tenemos las tres formas en las figuras siguientes:





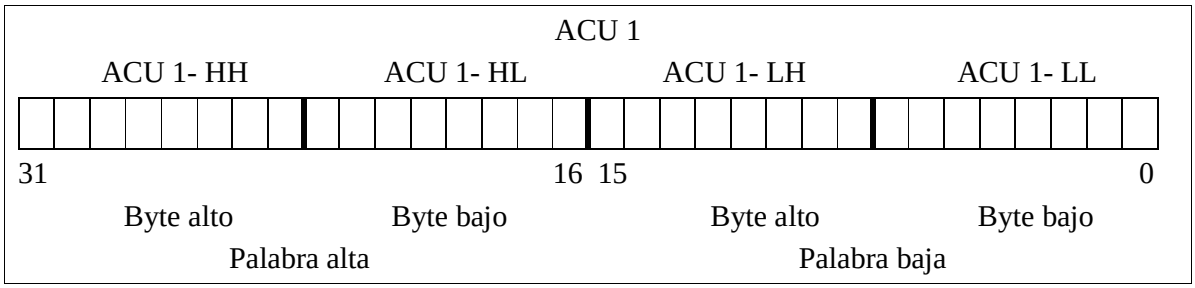
A esta forma AWL, se le han añadido los comentarios que van detrás de //

```
U     E    124.0              //Si el RLO es 1
ZV    Z      1                //Sube una unidad la cuenta del contador Z1

U     E    124.1              //Si el RLO es 1
ZR    Z      1                //Baja una unidad la cuenta del contador Z1

U     E    124.2              //Si el RLO es 1
L     C#20
S     Z      1                //Mete un 20 en la cuenta del contador Z1

U     E    124.3              //Si el RLO es 1
R     Z      1                //Pone a cero la cuenta del Z1

L     Z      1
T     MW    40                //En MW  40 se ve la cuenta del Z1 en decimal

LC    Z      1
T     MW    42                //En MW  42 se ve la cuenta del Z1 en BCD

U     Z      1
=     A    124.0              //Si la cuenta de Z1 es distinta de cero, se activa la salida
```

Como normalmente no se programan todas las funciones de un contador, para programar contadores, la opción más rápida y sencilla es hacerlo en AWL, en la que podemos programar por ejemplo, solo incrementar la cuenta con las dos primeras instrucciones. Para cargar un numero en el contador, la forma habitual es la del ejemplo anterior. Para hacer comparaciones con contadores, se emplea más la forma decimal para cargar el contador y después comparar.

15 – ACUMULADORES. Operaciones de carga y transferencia

Los acumuladores son zonas de memoria auxiliares de la CPU en las que podemos escribir o leer datos. Se emplean para el intercambio de datos entre varias direcciones y para realizar operaciones matemáticas y de comparación.
El S7-300 tiene dos acumuladores de 32 bits cada uno (ACU1 y ACU2) y el S7-400 tiene 4 acumuladores de 32 bits cada uno.
La instrucción de carga (L, de load), carga el contenido del byte, palabra o doble palabra especificado en el ACU 1; si volvemos a repetir la instruccion de carga, lo que habia en el ACU1, pasa al ACU2.
Con la instrucción de Transferencia, (T) pasamos-copiamos los datos del ACU1 en la dirección especificada en la instrucción, el contenido del ACU 1 se mantiene; por tanto, podemos transferir la misma información a otros destinos. Si sólo transferimos un byte, se cogen los 8 bits situados más a la derecha del ACU.
El resultado de las operaciones lógicas entre el ACU 1 y el ACU 2 (operaciones aritméticas, de comparación, Y, O, ...) siempre se almacena en el ACU 1.



Hay varias operaciones con acumuladores, se especifican en las tablas del resumen de operaciones.

16 - COMPARADORES

Descripción

Las operaciones Comparar enteros (de 16 bits) comparan el contenido del ACU2-L con el contenido del ACU1-L . Las operaciones Comparar enteros dobles y números en coma flotante (32 bits, IEEE-FP), comparan el contenido del ACU 2 con el contenido del ACU 1. Los tipos de comparación están en la tabla resumen del final.
RLO = 1 indica que el resultado de la comparación es verdadero. RLO = 0 indica que el resultado de la comparación es falso. Los bits A1 y A0 de la palabra de estado indican la relación "menor que", "igual que" o "mayor que".
Para hacer una comparación, tenemos que “cargar” en los acumuladores los dos datos o variables que queremos comparar y activar una salida que nos indique el resultado de la comparación. Ejemplo:

```
L    Z     1
L    40
>=I
=    M     20.0     // si el valor de contaje de  Z1 es  >=  40;  se activa la marca   M  20.0
```

Nota: Ver operaciones aritméticas en las tablas-resumen de operaciones del STEP-7

17 - COMPARADORES DE RELOJ EN TIEMPO REAL

Los bloques de organización y en concreto el OB1, en su Tabla de Declaración de Variables, llevan el dato DATE_AND_TIME (fecha y hora actuales). Este dato o variable temporal ocupa 8 Bytes con el valor de: año – mes – día – h – min – s – mseg. (2 primeras cifras) - mseg y día de la semana. Como en la TDV empiezan en la dirección 12, el byte 0 corresponde a la dirección 12

Byte (Direccion)	**Contenido**	**Margen posible de valores**
12	Año	90-89, es decir, los años 1990 hasta 2089
13	Mes	1 – 12
14	Día	1 – 31
15	Hora	00 – 23
16	Minuto	00 – 59
17	Segundo	00 – 59
18	Las dos cifras más significativas de mseg	0 – 99
19 (4MSB)	La cifra menos significativa de mseg	0 - 9
19 (4LSB)	Día de la semana 1 = Domingo 2 = Lunes ... 7 = Sábado	1 - 7

El tipo de datos Date_And_Time se guarda en formato BCD. El margen va desde DT#1990-01-01-0:0:0.0 hasta DT#2089-12-31-23:59:59.999. Como estos Bytes, son datos Locales, para cargarlos se emplea la letra L de Local. Ejemplo:

```
L    LB   17
L    C#55
>=I
=    A    124.0    //Esta Salida, se activa los 5 últimos segundos de todos los minutos.
```

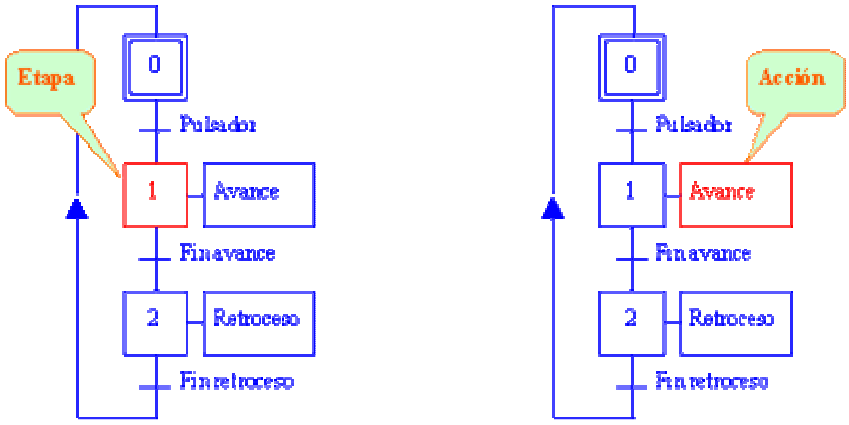
18 - ELEMENTOS Y PRINCIPIOS BÁSICOS DEL GRAFCET

INTRODUCCIÓN

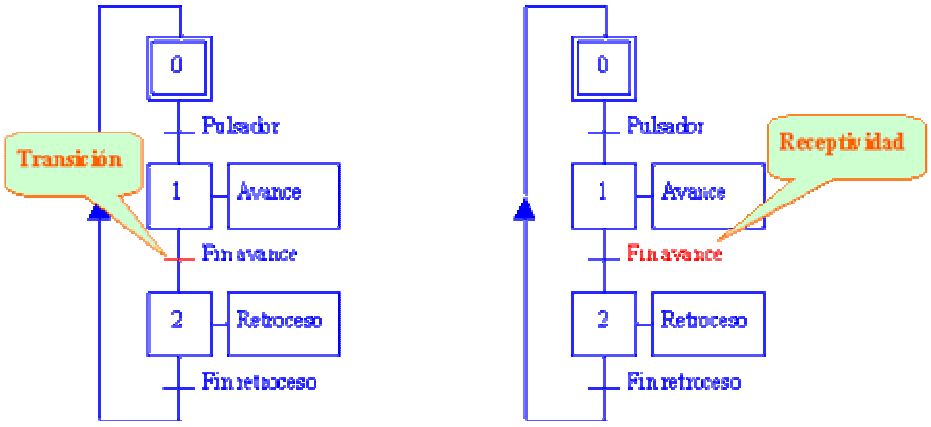
El GRAFCET es un sistema de representación grafica en el que se relacionan los estados de entradas y salidas de un automatismo secuencial, según va evolucionando su estado. La creación del GRAFCET fue necesaria, entre otros motivos, por las dificultades que comportaba la descripción de automatismos con varias etapas simultáneas utilizando el lenguaje normal. Nació en el año 1977 en un grupo de trabajo de la AFCET (*Association Française pour la Cybernétique Economique et Technique).* En el año 1988, el GRAFCET es reconocido por una norma internacional, la IEC-848.
Un sistema combinacional es aquel en que las salidas en un instante sólo dependen de las entradas en aquel instante. En cambio, un automatismo secuencial es aquel en el que las salidas en cada instante no dependen sólo de las entradas en aquel instante sino que también dependen de los estados anteriores y de su evolución.

ELEMENTOS Y PRINCIPIOS DEL GRAFCET

El GRAFCET es una sucesión alternada de **etapas y transiciones**. Cada etapa tiene sus **acciones** asociadas de forma que cuando una etapa está activa se realizan las correspondientes acciones; pero estas acciones no podrán ejecutarse nunca si la etapa no está activa.
Las etapas iniciales, que se representan con línea doble, se activan en la puesta en marcha. Las etapas activas se suelen señalizar poniendo un punto debajo del nº de etapa.



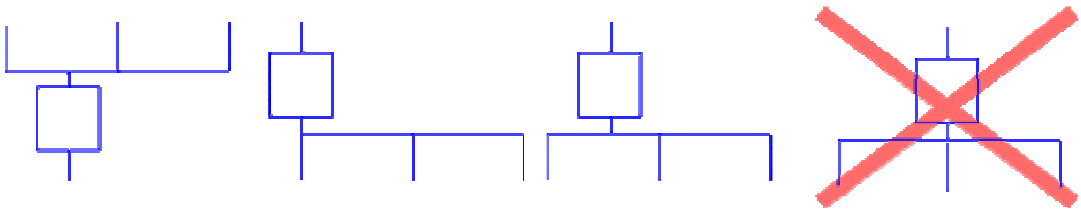
Entre dos etapas solo puede haber una **transición**. A cada transición le corresponde una **receptividad**, es decir la condición que se ha de cumplir para poder pasar la transición; en caso de tener que cumplirse varias condiciones, estas se agrupan en una sola receptividad. Mediante líneas (**uniones orientadas**) se indican las vías de evolución del GRAFCET, el sentido general es de arriba hacia abajo, la llegada o entrada a una etapa es por la parte de arriba y la salida por la de abajo; en caso contrario, hay que señalizarlo con flechas. Para ir de una etapa a otra hay que pasar obligatoriamente por una y solo una transición (receptividad), si deben cumplirse varias condiciones, estas se agrupan en una sola receptividad.



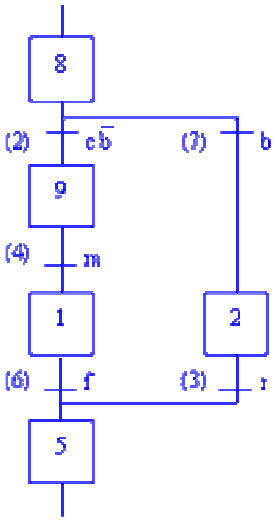
Una transición es **válida** cuando la etapa inmediatamente anterior a ella está activa. Cuando una transición es válida y su receptividad asociada se cumple se dice que la transición es **franqueable.** O sea: para que se realice el paso de una etapa a la siguiente, tiene que estar activa la etapa precedente y cumplirse la receptividad; en ese momento se activa la etapa siguiente y se desactiva la precedente. En GRAFCETs con varias ramas, al franquear una transición (o mas si están en paralelo) se desactivan sus etapas anteriores y se activan las posteriores. Si una etapa del Grafcet se activa y se desactiva al mismo tiempo, debe quedar activa. Si la descripción de un GRAFCET lo requiere, pueden numerarse las transiciones con un número entre paréntesis a la izquierda del trazo que representa la transición.

GRAFCET CON VARIAS RAMAS O CAMINOS (secuencias multiples)

En estos casos algunas etapas pueden tener varias entradas y/o salidas. En la representación, es conveniente hacer las uniones evitando que confluyan mas de tres caminos pues se suele entender que en un punto en el que hay cuatro caminos, representa un cruce y no una unión.



SELECCIÓN DE SECUENCIA (ramas en "o")

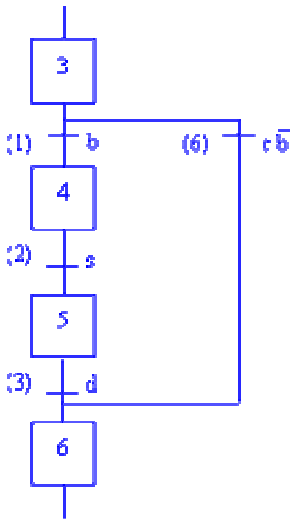


A partir de una determinada etapa, hay dos (o más) secuencias entre las que se escogerá en función de las transiciones. No es necesario que las distintas secuencias tengan el mismo número de etapas. En la figura, si estamos en la etapa 8 y b es cierta iremos por la secuencia de la derecha; si c es cierta y b es falsa iremos por la de la izquierda Las dos secuencias confluyen en la etapa 5.
En la selección de secuencia es imprescindible que las receptividades asociadas a las transiciones de selección, en el ejemplo las transiciones (2) y (7), sean excluyentes, es decir no puedan ser ciertas simultáneamente; por lo tanto las secuencias son alternativas.

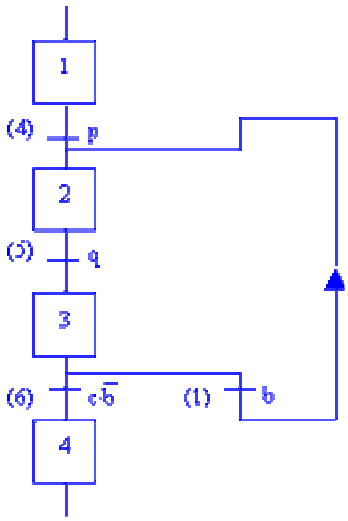
SALTO DE ETAPAS Y REPETICIÓN DE ETAPAS (salto adelante y salto hacia atrás)

Ambos son casos particulares de la selección de secuencia

Salto adelante
Es un caso particular de selección entre dos secuencias en el que una de las secuencias no tiene ninguna etapa.

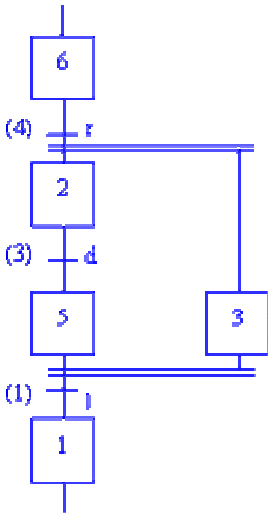


Salto hacia atrás
En la figura, se irá repitiendo la secuencia formada por las etapas 2 y 3 hasta que b sea falsa y c cierta.



SECUENCIAS SIMULTANEAS (Ramas en "y")

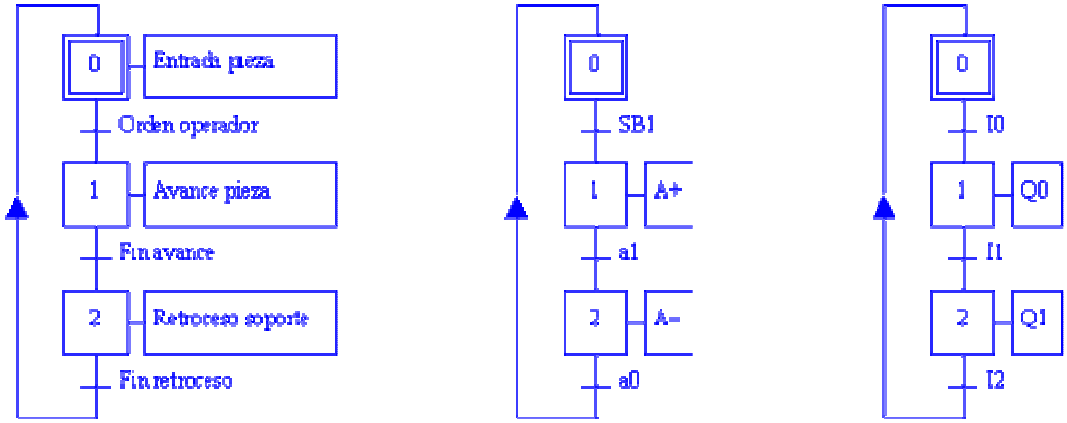
A partir de una determinada etapa, hay dos (o más) secuencias que se ejecutan simultáneamente. No es necesario que las distintas secuencias tengan el mismo número de etapas. El inicio de secuencias paralelas se indica con una línea horizontal doble después de la transición correspondiente. De forma similar, el final de las secuencias paralelas se indica con otra línea horizontal doble antes de la transición correspondiente; esta transición sólo es válida cuando todas las etapas inmediatamente anteriores están activas. En la figura, al franquear la transición (4), se activarán las etapas 2 y 3 y las dos secuencias trabajarán simultáneamente. La transición (1) sólo será válida cuando estén activas las etapas 3 y 5.



NIVELES DE ESPECIFICACIÓN / CONCRECIÓN DEL GRAFCET

El GRAFCET puede utilizarse para describir los tres niveles de especificaciones de un automatismo. Estos tres niveles son los que habitualmente se utilizan para diseñar y para describir un automatismo.
En el primer nivel interesa una descripción global (normalmente poco detallada) del automatismo que permita comprender rápidamente su función. Es el tipo de descripción que haríamos para explicar lo que queremos que haga la máquina a la persona que la ha de diseñar o

el que utilizaríamos para justificar a las personas con poder de decisión en la empresa, la necesidad de esta máquina. Este GRAFCET no debe contener ninguna referencia a las tecnologías utilizadas; es decir no se especifica cómo hacemos avanzar la pieza (cilindro neumático, motor y cadena, cinta transportadora, etc.), ni cómo detectamos su posición (final de carrera, detector capacitivo, detector fotoeléctrico, etc.), ni tan solo el tipo de automatismo utilizado (autómata programable, neumática, ordenador industrial, etc.



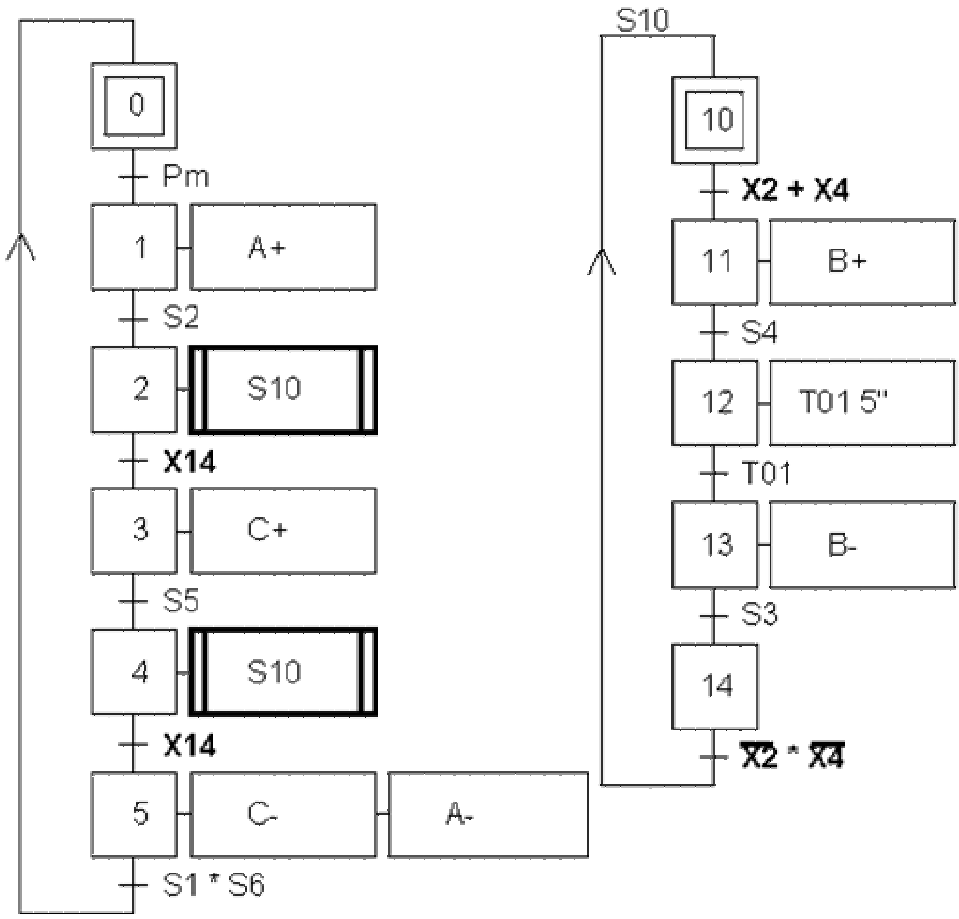
ñEn un segundo nivel se hace una descripción a nivel tecnológico y operativo del automatismo. Quedan perfectamente definidas las diferentes tecnologías utilizadas para cada función. El GRAFCET describe las tareas que han de realizar los elementos escogidos. En este nivel completamos la estructura de la máquina y nos falta el automatismo que la controla.
En el tercer nivel se implementa el automatismo. El GRAFCET definirá la secuencia de actuaciones que realizará este automatismo. En el caso de que se trate, por ejemplo, de un autómata programable, definirá la evolución del automatismo y la activación de las salidas en función de la evolución de las entradas.

GRAFCET – AUTÓMATA PROGRAMABLE.

A la hora de convertir un GRAFCET de tercer nivel en programa de PLC para STEP-7, las propias estructuras del GRAFCET, nos ayudan a hacer los agrupamientos necesarios para no repetir el Set de las etapas que tienen mas de un camino de entrada; ya sean en “y” en el agrupamiento de secuencias simultaneas, o en “o” en el agrupamiento de selección de secuencias y en los saltos.
Para las activaciones, también hay que agrupar en “o” todas las etapas que activan una misma salida de autómata. Aunque los Reset dan menos problemas que los Set e = , también es conveniente agruparlos en lo posible pues el programa queda mas claro.

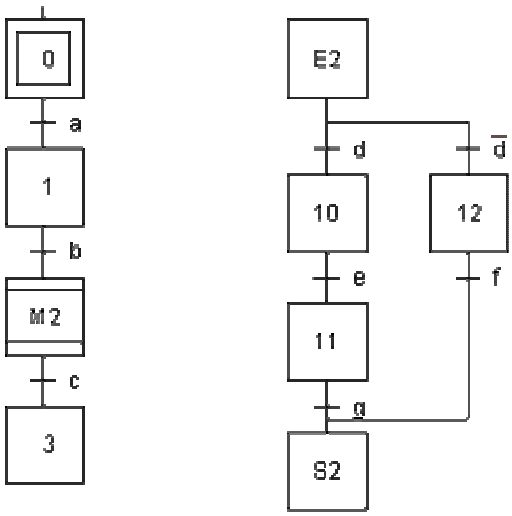
GRAFCET CON SUBRUTINAS.

Una subrutina es una parte de un programa que realiza una tarea concreta, a la que se puede invocar una o varias veces por parte del programa principal. Un vez realizadas las acciones de la subrutina el programa continua en el punto donde estaba.
Los trabajos a desarrollar en un automatismo se pueden dividir entre diferentes diagramas. Por ejemplo, puede haber un diagrama principal con etapas X0 a X5 y otro secundario con etapas X10 a X14, una vez que en estas últimas realizan determinadas funciones devuelven el control al diagrama principal.



Al llegar a la etapa X2 o X4 del primer diagrama, se valida la transición X2+X4 y empieza la subrutina. Al llegar a la etapa 14 se valida la transición X14 y continua la evolución del diagrama principal a las etapas 3 o 5 respectivamente. La receptividad de la ultima transición de la subrutina, también puede ser X3 + X5.

GRAFCET CON MACROETAPAS



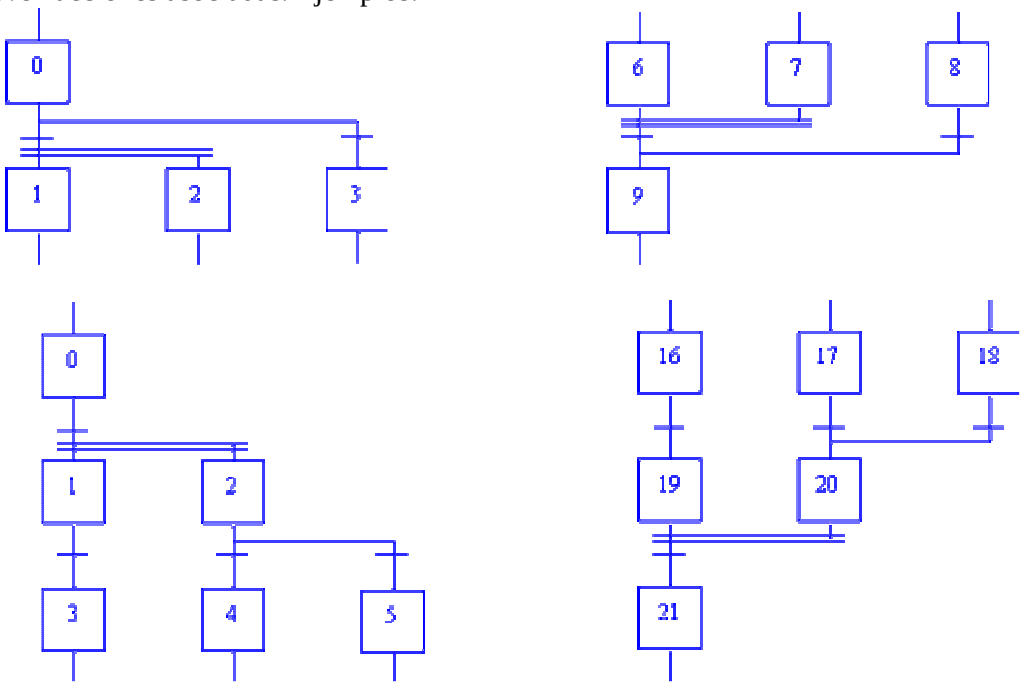
Al hacer la descripción del automatismo, el Grafcet permite empezar desde un punto de vista muy general y a partir de él hacer descripciones cada vez más concretas del proceso de control. Una Macro-etapa es la representación mediante una única etapa, de un conjunto de etapas, transiciones y acciones asociadas, a las que llamamos expansión de la macro-etapa. Podríamos decir que al hacer la expansión de la macro etapa, en realidad lo que hacemos es una especie de zoom, que nos enseña en detalle, etapas, transiciones y acciones concretas, a las que antes nos hemos referido de forma general.

El diagrama principal evoluciona a partir de la etapa 0 y la transición a; una vez que esta activa la etapa 1, la transición b estará receptiva, y al validarse,

entraremos a la macro etapa M2, la etapa E2 estará activa, y según el estado de la transición d, evolucionara hacia la etapa 10 o la 12, y al llegar a la etapa S2 volverá al diagrama principal. La etapa E2 es la etapa de entrada a la macro 2, la etapa S2, es la etapa de salida de la macro 2.

COMBINACIÓN DE ESTRUCTURAS BASICAS EN UN GRAFCET.

Hay automatismos que requieren un GRAFCET con diversas ramas o combinaciones de: saltos, ramas en "o", en "y", GRAFCETs paralelos, etc. En general no hay problema para combinar ramas en "o" con ramas en "y" si hay etapas intermedias entre ellas. Si estas ramas tienen que ir seguidas; después de "o" puede ir "y" y después de "y" "o" sin problemas; en cambio para poder hacer "o" después de "y" y "y" después de "o" es mejor meter etapas intermedias aunque no lleven acciones asociadas. Ejemplos:



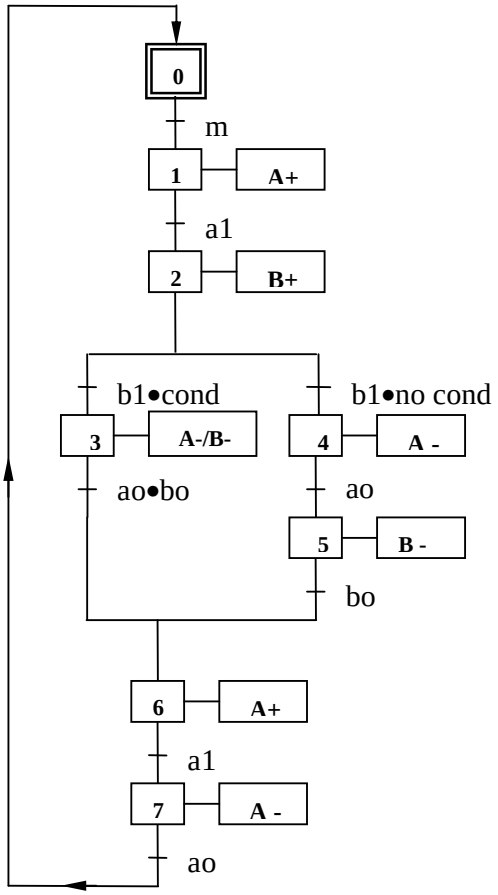
Las etapas 1, 2, 19 y 20 de los grafcets de abajo, se han puesto por motivos estructurales y para dar mayor claridad pero no llevarán ninguna acción asociada. La 1, 19 y 20 pueden no ser necesarias, pero la 2 es imprescindible pues si falta habrá dos transiciones seguidas. Si no se ponen la 19 y 20, se realiza el paso a la 21 si esta activa la 16 y se cumple su receptividad y además esta activa la 17 y se cumple su receptividad o esta activa la 18 y se cumple su receptividad.
Estos apuntes, son solo una introducción y más aún lo expuesto en este último apartado. Para completarlos y ampliarlos, hay mucha bibliografía y diversas paginas Web referentes a este tema; simplemente con buscar GRAFCET aparecen varias direcciones de Internet; en concreto de las siguientes se han extraído partes de esta introducción.

http://gpds.uv.es/plc/
http://perso.wanadoo.es/kiko2000/
http://www.grupo-maser.com/PAG_Cursos/Cursos.htm

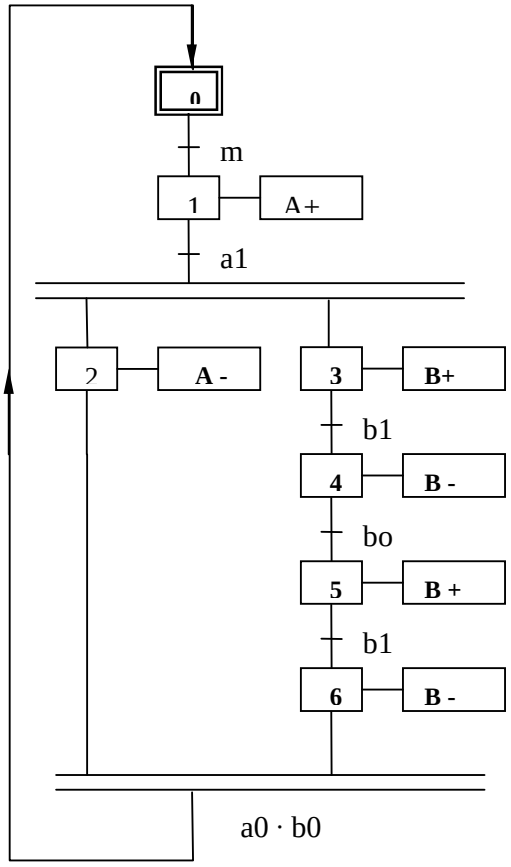
EJEMPLOS PARA PASAR DE GRAFCET A PROGRAMA STEP-7

En los tres casos la válvula A, es 4/2 monoestable y la B, 4/3 monoestable con dos bobinas

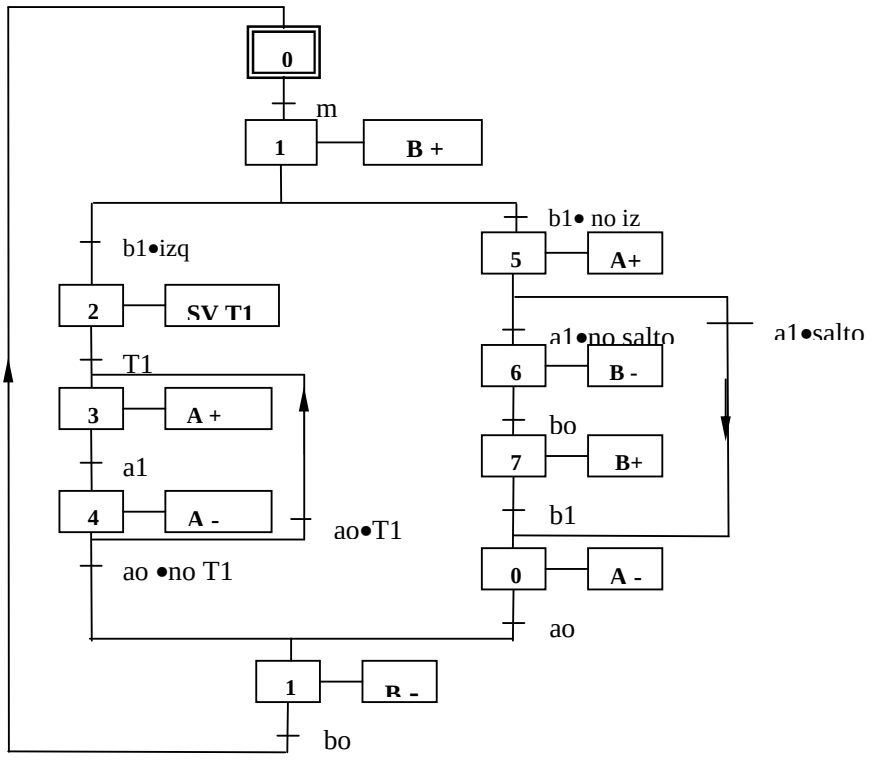
A) Dos ramas en "o"



B) Dos ramas en "y"



C) Dos ramas en "o" mas saltos



19 – PROGRAMACION GENÉRICA ESTRUCTURADA

La programación genérica consiste en programar las instrucciones con direccionamiento simbólico local, para lo cual, primero tenemos que generar (declarar) las direcciones simbólicas que vayamos a emplear; esto se hace en la "tabla de declaración de variables" que tienen todos los bloques de programa; una vez hecho esto, podemos programar las instrucciones necesarias en la zona de programación de usuario que también tienen todos los bloques. Este tipo de programación, se emplea para facilitar la programación cuando hay que repetir partes de programa con la misma estructura.
En las tablas de declaración de variables y de parámetros de los bloques, además de los datos locales que algunos bloques llevan ya incluidos para su funcionamiento particular, podemos definir o declarar otros nuevos como: parámetros y variables temporales en los FCs; parámetros, variables estáticas y variables temporales en los FBs; y variables temporales en los OBs;

PARÁMETROS Y VARIABLES DE UNA TDV.

Parámetros formales.- Son "variables" que se definen o crean en la tabla de declaración de un bloque lógico mediante nombres simbólicos; pueden ser de tres tipos: in, out, e in_out. En general hay que declarar como in lo que hace de E, T, Z, como out lo que hace de A y como in_out lo que hace de E y de A.

Parámetros actuales:- Son los valores o datos que tenemos que escribir para sustituir a los parámetros formales que nos ofrece step-7, al hacer llamadas desde un bloque lógico a otro bloque genérico. Estos parámetros actuales pueden ser direcciones simbólicas especificadas en la tabla de símbolos, direcciones absolutas ó también a su vez parámetros formales declarados como variables temporales en el propio bloque que hace la llamada. El bloque trabaja realmente con estos parámetros y si hacemos varias llamadas, como en cada llamada asignamos distintos parámetros actuales, no hay interferencias entre ellos.

Tipos de variables.- En las tablas de declaración de variables de los bloques lógicos, podemos declarar también variables temporales (temp) y estáticas (stat); estas últimas solo en FBs. Al contrario que los parámetros, las variables no intervienen en las llamadas entre bloques.

Variables temporales.- Son variables que solo están disponibles durante el tratamiento de su propio bloque, estas variables son datos locales que se almacenan en el área de memoria local o pila de datos locales. Son variables intermedias que no van a tener una asignación real de Es, As, Ts, Zs y por tanto no importa que actúen como in, out o in_out; su declaración es temp. Digamos que pueden hacer la función de una marca, pero en forma de dato local.

Variables estáticas.- Son variables que se almacenan en DBs de Instancia, por lo tanto solo se pueden declarar en los FBs con DBs asociados o de Instancia. Estas variables también están disponibles aunque no se esté ejecutando su FB, y pueden contener valores de: estado 1 ó 0, contaje, revoluciones, nº de arranques, etc. Las misma variable estática puede tener distintos valores en cada uno de los DBs y aunque en su FB y en los DBs tiene el mismo nombre y dirección, no hay interferencias entre estos valores.

BLOQUES EMPLEADOS EN LA PROGRAMACIÓN GENÉRICA ESTRUCTURADA

En las instalaciones que hay varios elementos o equipos análogos repetidos, como pueden ser: válvulas, motores, automatismos, etc; el programa total, se divide en varios bloques de programa y en estos bloques hacemos un programa genérico para las válvulas, otro para los motores, etc; para ello en este Bloque o bloques, empleamos variables locales como por ejemplo: U #marcha, U #temporizador, = #motor; en vez de: U E 124.0, U T1, = A 124.0; después desde otro Bloque de programa, podemos llamar al bloque o bloques de programa genéricos las veces que este elemento o equipo se repita y decirle en cada llamada que Es, Ss, Zs, Ts,... utiliza en particular cada uno de esos elementos o equipos repetidos; o sea hacemos un programa formal que vale para varios elementos o partes de programa repetidas y después concretamos las direcciones actuales o de trabajo real de cada uno de ellos.
En este tipo de programación, se pueden emplear:

- Bloques lógicos de organización OBs, (el OB1 no puede faltar en ningún caso).

- Funciones FCs.- Son Bloques que no tienen "memoria" particular, se puede programar en ellos todo lo habitual excepto variables genéricas que puedan o deban tener distintos valores en cada una de las llamadas que se hagan a este FC. La llamada a un bloque FC (FC3 por ejemplo) se hace con la instrucción: Call FC3

- Bloques de función FBs.- Se puede hacer en ellos lo mismo que con los FCs, y además como tienen "memoria" particular, se pueden programar variables genéricas declaradas como Stat, y cada una de ellas puede tener distintos valores. En cada llamada Call a un FB, se puede asociar un bloque de datos de instancia, DB1, DB2, DB3,.... y en él se guarda o memoriza el valor particular de estas variables y del resto de los parámetros que se programen en el FB; en estos DBs, solo puede "leer" o "escribir" ese FB concreto. La llamada a un bloque FB (FB3 por ejemplo) se hace con la instrucción: Call FB3, DB1 para la primera llamada; Call FB3, DB2 para la segunda, etc.
  La llamada a un FB con Call, exige que lleve asociado un DB, pero si no lleva variables stat entonces el FB equivale prácticamente a un FC, pero gasta mas recursos del PLC.

- DBs de instancia, se pueden crear de dos formas, insertándolos en el Administrador Simatic y después al abrilos poner a que FB va asociado, o al hacer una llamada con Call a un FB, poner que DB lleva asociado. En ambos casos, al generar estos DBs, ellos solos se "copian" para su funcionamiento la Tabla de Declaración de Variables del FB, excluyendo las variables temp.

- DBs generales o globales, pueden tener datos en los que pueden "leer" o "escribir" todos los bloques lógicos programados. Hay que insertarlos, abrirlos y rellenar la tabla con la estructura de las las variables que necesitemos.

- También para uso más especifico se pueden emplear: FCs, FBs, SFCs, SFBs, de las Librerías del sistema; UDTs, etc.

ALGUNAS RESTRICCIONES PARA LAS LLAMADAS ENTRE BLOQUES

Podemos mezclar programación absoluta con genérica y hacer el mismo programa con distintas combinaciones entre partes genéricas y absolutas; declarar mas o menos variables en las TVD

para usar mas o menos datos locales; emplear mas o menos Bloques; hacer llamadas a bloques que a su vez han llamado a otros Bloques, etc.
En las llamadas entre boques, a los parámetros formales se les pueden asignar parámetros actuales (o reales), pero también se les pueden asignar a su vez parámetros formales (que harían de actuales) de la tabla de declaración del bloque que hace la llamada, pero en general con las siguientes restricciones:

- Cuando un FC llama a otro FC.- No se pueden asignar variables de tipo compuesto o parámetros.
- Cuando un FC llama a un FB.- No se pueden asignar variables de tipo compuesto y dentro de los parámetros soto se puede con: Timer, Counter y Block pero siendo la declaración in-in.
- Cuando un FB llama a un FC.- No se pueden asignar parámetros ni datos compuestos con declaración in-out.
- Cuando un FB llama a un FB.- Restringido solo a datos compuestos si la declaración es in-out. Permitido a tipo parámetro solo si la declaración es in-in y los parámetros Timer, Counter o Block.

Seleccionando: Ayuda – temas de ayuda – índice -- "Tipo de datos admisibles al transferir parámetros", hay información mas detallada, también accedemos a la misma ayuda si estando dentro de un bloque de programación, seleccionamos: Ayuda – ayuda de AWL – índice – ponemos lo que queramos – mostrar -- en la nueva ventana seleccionamos ayuda de step7 -- – índice -- "Tipo de datos admisibles al transferir parámetros".

20 - TABLAS RESUMEN DE OPERACIONES DEL STEP-7

1. OPERACIONES BINARIAS
2. OPERACIONES DE TEMPORIZACION
3. OPERACIONES DE CONTAJE
4. OPERACIONES DE CARGA/TRANSFERENCIA
5. OPERACIONES DE COMPARACIÓN
6. OPERACIONES ARITMÉTICAS
7. OPERACIONES LÓGICAS CON PALABRAS
8. OPERACIONES DE TRANSFORMACIÓN
9. OPERACIONES DE DESPLAZAMIENTO Y ROTACIÓN.
10. OPERACIONES DE SALTO
11. OPERACIONES DE CONTROL DEL PROGRAMA
12. OPERACIONES DE CONVERSIÓN DE TIPOS DE DATOS.
13. OPERACIONES CON ACUMULADORES

T.1 OPERACIONES BINARIAS

T.1.1 Operaciones lógicas con bits.- Consulta el estado de señal del operando direccionado según la función lógica deseada.

U	Realiza la operación lógica "Y" entre dos o más operandos
UN	Realiza la operación lógico "NO-Y" entre dos o más operandos
O	Realiza la operación lógica "O" entre dos o más operandos

ON	Realiza lo operación lógica "NO-0" entre dos o más operandos
X	Realiza la operación lógica "0-EXCLUSIVA" entre dos o más operandas
XN	Realiza la operación lógica "NO-O-EXCLUSIVA" entre dos o más operandos.

T.I.2 Operaciones lógicas de expresiones entre paréntesis.- Se emplean para que el conjunto de varios operandos que están dentro de los paréntesis se consideren como un solo operando para realizar una función. La pila de paréntesis puede contener un máximo de 7 entradas.

U(	Función "Y-Abrir paréntesis"
UN(	Función "NO-Y-Abrir paréntesis"
O(	Función "0-Abrir paréntesis"
ON(	Función "NO-0-Abrir paréntesis"
X(	Función "0-EXCLUSIVA-Abrir paréntesis"
XN(	Función "NO-0-EXCLUSIVA-Abrir paréntesis"
)	Función "Cerrar paréntesis”

T.1.3 Activar/borrar (set/reset) bits.- Sirven para asignar el valor" 1" o "O" al operando direccionado.

S	Realiza lo operación "SET" [poner o 1} de un operando
R	Realiza la operación "RESET' (poner a 0) de un operando
=	Asigna el "RLO" a un operando. Envía el resultado de una operación lógica a un operando

T.1.4 Operaciones con flancos.- Sirven para detectar un cambio de flanco, es decir, los cambios de señal de"0" a "l" y de "l" a "0" y lo indican con RLO = 1. Necesitan una marca asociada ya que para poder ejecutar la comparación hay que almacenar el estado del bit RLO anterior en la dirección de la marca de flancos (**<bit>**). Las operaciones de flancos, se pueden utilizar con: Entradas, "E"; Salidas, "A"; Marcas, "M"; Bit de datos locales, "L"; Bit de datos, "DBX"; Bit de datos, "DBI"; Temporizadores "T" y Contadores "Z".

FP	Detecta el flanco ascendente de un operando
FN	Detecta el flanco descendente de un operando

T.1.5 Bits de la palabra de estado.- Cada vez que se ejecuta una instrucción, la CPU además de realizar la acción que hemos programado lee o escribe varios bits para poder realizar otras acciones optativas. Estos bits conforman la llamada palabra de estado que esta compuesta por los siguientes bits:

Bit 0	/ER	La próxima operación da inicio a una nueva cadena lógica.
Bit 1	RLO	Almacena el resultado de una cadena de operaciones lógicas o de comparación.
Bit 2	STA	Almacena el valor del operando direcionado.
Bit 3	OR	Una expresión entre paréntesis ha dado como resultado 1.
Bit 4	OS	Desbordamiento memorizado.
Bit 5	OV	Desbordamiento
Bit 7	A0	Información sobre resultados de operaciones aritméticas, de comparación, digitales y desplazamiento.
Bit 8	A1	Igual que A0
Bit 9	RB	Bit de resultado binario.

El bit mas importante es el RLO (Resultado Lógico de la Operación), si al final de un conjunto de operaciones es 1, se realiza la activación u operación programada. El resultado del RLO podemos asimilarlo en un circuito eléctrico al paso o no de la corriente eléctrica.

Con estos bits podemos realizar las operaciones: U, UN, O, ON, X y XN. Los bits que podemos consultar son, el resultado del A1 comparado con el resultado del A2 y los resultados del: OS, OV, RB por separado
En la siguiente tabla se ven ejemplos de consulta de los bits de estado con la operación "U":

U ==0	"Y" resultado=0 (A1=0 y A0=0)
U >0	"Y" resultado >0 (Al =1 y A0=0)
U <0	"Y" resultado <0 (Al = 0 y A0 = 1)
U <>0	"Y" resultado distinto 0 ((Al = 0 y A0 = 1) o (Al = 1 y A0 = 0))
U <=0	"Y" resultado <=0 ((A1 = 0 y A0 = 1) o (A1= 0 y A0 = 0))
U >=0	"Y" resultado > =0 ((Al = 1 y A0 = 0) o (Al = 0 y A0 = 0))
U UO	"Y" resultado no admisible (Al =1 y A0 = 1)
U OS	“Y” si desbordamiento memorizado
U 0V	“Y” si desbordamiento

U RB U BIE	"Y" si RLO almacenado es 1

T.1..6 Operaciones que afectan directamente al RLO.- Son operaciones que actúan directamente sobre el bit RLO (Resultado Lógico de la Operación)

CLR	Pone el RLO a "0"
SET	Pone el RLO a 1
NOT	Invierte RLO
SAVE	Almaceno el RLO en el bit RB de la palabra de estado

T.2 OPERACIONES DE TEMPORIZACIÓN

Los temporizadores se denominan con la letra "T" seguida del número de temporizador. El número de temporizadores que se pueden utilizar depende del modelo de CPU.
Hay 5 modelos de temporizadores:

- **SI:** arranque de un temporizador como impulso.
- **SV:** arranque de un temporizador como impulso mantenido.
- **SE:** arranque de un temporizador como retardo a la conexión.
- **SS:** arranque de un temporizador como retardo a la conexión memorizado.
- **SA:** arranque de un temporizador como retardo a la desconexión.

T.3 OPERACIONES DE CONTAJE

Los contadores se denominan con la letra "Z" seguida del número de contador. El número de contadores que se pueden utilizar depende del modelo de CPU. Hay 3 modelos de contadores:

- contador ascendente,
- contador descendente
- contador ascendente/descendente.

T.4. OPERACIONES DE CARGA/TRANSFERENCIA

T.4.1 Operaciones de carga

Sirven para cargar o memorizar un operando en el ACU1, memorizando previamente el contenido del ACU1 en el ACU2. La palabra de estado permanece inalterada.

L	Carga el operando en el ACU1; esta es la forma que se usa habitualmente
LC	Carga el valor de un temporizador o contador en BCD

T.4.2 Operaciones de transferencia

Sirven para transferir el contenido del ACUl al operando direccionado. La palabra de estado permanece inalterada.

T	Transfiere el contenido del ACU1 a la dirección que especifiquemos

Los operandos posibles a utilizar en los anteriores juegos de operaciones son todos los tipos de datos de STEP-7.

T.5. OPERACIONES DE COMPARACIÓN

T.5.1 Operaciones de comparación con números enteros (16 bits)

Compara los enteros depositados en ACU1-L y ACU2-L. Si se cumple la condición el RLO pasa a 1. Se pone la comparación I de Integer o Entero. Ejemplo:

==I	Comparación a igual (ACU2-L = ACU1-L)
<>I	Comparación a distinto (ACU2-L distinto ACU1-L)
<I	Comparación a menor (ACU2-L < ACU1-L)
<=I	Comparación a menor o igual (ACU2-L < = ACU1-L)
>I	Comparación a mayor (ACU2-L > ACU1-L)
>=I	Comparación a mayor o igual (ACU2-L > = ACU1-L)

T.5.2 Operaciones de comparación con números enteros (32 bits)

Compara los enteros de 32 bits depositados en ACU1 y ACU2. Es lo mismo que con enteros de 16 bits, pero empleando la D de doble entero

T.5.3 Operaciones de comparación con números reales (32 bits)

Compara los números reales depositados en ACU1 y ACU2. Ídem pero con la R de Real. Ejem:

```
L    MD   50
L    3.400000e+002
==R
=    M    20.7      // si se cumple la comparación se activa la marca  M  20.7
```

T.6 OPERACIONES ARITMÉTICAS

T.6.1 Operaciones aritméticas con números enteros (16 bits)

El resultado de la operación se deposita en ACU1; como los acumuladores son de 32 bits, para estas operaciones de 16 bits, se emplea la palabra baja (Low) del acumulador ACU1-L. La palabra alta (Hight) es ACU1-H

+ I	Suma dos enteros de 16 bits. Resultado en ACU1-L
- I	Resta dos enteros de 16 bits. Resultado en ACU1-L
* I	Multiplica dos enteros de 16 bits. Resultado en ACU1-L
/ I	Divide dos enteros de 16 bits. Resultado en ACU1-L. Resto en ACU1-H.

Ejemplo:

```
L    5
L    8
*I
T    MW   20        // El resultado de la operación que es 40, lo pasamos del ACU1 a la MW 20
                       para poderlo ver
```

T.6.2 Operaciones aritméticas con números enteros (32 bits)

Son análogas a las anteriores pero con D de doble entero y además tienen la operación, MOD que divide 2 enteros dobles y carga el resto de la división en el ACU1.

T.6.3 Operaciones aritméticas con números reales (32 bits)

Lo mismo que las anteriores pero con: + R - R *R y /R

T.6.4 Raíz cuadrada y cuadrado (32 bits)

SQRT	Calcula la raíz cuadrada de un número real
SQR	Calcula cuadrado de un numero real

Ejemplo:

```
L    2.000000e+000
SQRT
T    MD   20              // En MD 20, vemos en formato real el resultado: 1.414214e+000
```

T.6.5 Funciones logarítmicas y exponenciales (32 bits) ***(No son posibles con*** *la CPU 312 IFM*)

LN	Forma el logaritmo natural (base e) de un número real.
EXP	Calcula el valor exponencial (e^n) de un número real.

Se programa de forma análoga a la raíz y al cuadrado.

T.6.6 Funciones trigonométricas (32 bits) ***(No son posibles con*** *la CPU 312 IFM*)

SIN	Calcula el seno de un numera real.

ASIN	Calcula el arcoseno de un numero real.
COS	Calcula el coseno de un número real.
ACOS	Calcula el arcocoseno de un número real.
TAN	Calcula la tangente de un numero real.
ATAN	Calcula el arcotangente de un numero real.

Ejemplo:
L 1.047200e+000 // ángulo expresado en radianes; 60º = π/3 rad =1.0472 rad
SIN
T MD 20 // En MD 20, vemos en formato real el resultado: 8.660266e-001

T.7. OPERACIONES LÓGICAS CON PALABRAS, O CON DOBLES PALABRAS.

Combinan el contenido con una palabra (W) o doble palabra (D), mediante la función deseada.

UW	Combinación "Y" bit a bit de una palabra.
OW	Combinación "O" bit a bit de una palabra.
XOW	Combinación "0-EXCLUSIVA" bit a bit de una palabra.

T.8 OPERACIONES DE TRANSFORMACIÓN CON ENTEROS (I) Y DOBLES ENTEROS (D)

INVI (INVD)	Forma el complemento a 1. Se invierte bit a bit el número
NEGI (NEGD)	Forma el complemento o 2. Se hace el negativo de un número.

T.9 OPERACIONES DE DESPLAZAMIENTO Y ROTACIÓN

T.9.7 Operaciones de Desplazamiento

Las operaciones de desplazamiento sirven para desplazar el contenido de la palabra baja del ACU1 o de todo el ACÜ1 bit a bit a la izquierda o a la derecha.

SLW	Desplaza el contenido del ACU1-L a la izquierda.
SLD	Desplaza el contenido del ACU1 a la izquierda.

SRW	Desplaza el contenido del ACU1-L a la derecha.
SRD	Desplaza el contenido del ACU1 a la derecha.
SSI	Desplaza el contenido del ACU1-L a la derecha rellenando las posiciones vacantes con el valor del bit de signo.
SSD	Desplaza el contenido del ACU1 a la derecha rellenando las posiciones vacantes con el valor del bit de signo.

Ejemplo:
```
L    2# 100000000000111
SLW  5
T    MW   20                  // En Mw  20,  vemos: 0000_0000_1110_0000
```

T.9.2 Operaciones de rotación

Las operaciones de rotación hacen circular todo el contenido del ACU1 (los 32 bits) bit a bit a la izquierda o a la derecha. Las posiciones vacantes se rellenan con los estados de señal de los bits que se desplazan fuera del acumulador.

RLD	Rota el contenido del ACU1 a la izquierda.
RRD	Rota el contenido del ACU1 a la derecha.
RLDA	Rota el contenido del ACU1 una posición a la izquierda según el bit Al.
RRDA	Rota el contenido del ACU1 una posición a la derecha según el bit A1

Ejemplo:
```
L    2#1111000000001111    // MW 22
RRD  4
T    MD   20          // En MD  20,  vemos: 1111_0000_0000_0000    0000_1111_0000_0000
                                             ( MW 20          mas          MW 22)
```

T.10 OPERACIONES DE SALTO

T.10.1 Operaciones de salto incondicional

Las siguientes operaciones de salto se utilizan para interrumpir el desarrollo normal del programa sin condiciones.

SPA	Salto incondicional
SPL	Distribuidor de saltos o metas

T 10.2 Operaciones de salto condicional

Las siguientes operaciones de salto interrumpen el desarrollo del programa dependiendo del resultado lógico (RLO) dado en la instrucción anterior.

SPB	Salto si el anterior RLO=1.
SPBN	Salto si el anterior RLO=0.
SPBB	Salto si el anterior RLO = 1; el RLO se almacena en el bit RB.
SPBNB	Salto si el anterior RLO = 0; el RLO se almacena en el bit RB.

T.10.3 Operaciones de salto en fundón de RB, OV, OS

Las siguientes operaciones de salto interrumpen el desarrollo del programa en función del estado de señal de determinados bits de la palabra de estado.

SPBT	Salto si RB=1.
SPBIN	Salto si RB=0.
SPO	Salto si OV=1.
SPS	Salto si OS =1.

T10.4 Operaciones de salto en función de Al y A0

Las siguientes operaciones de salto interrumpen el desarrollo del programa en función del resultado de una operación anterior.

SPZ	Salto si resultado = 0.
SPN	Salto si no es 0.
SPP	Salto si es positivo (es decir, mayor que 0).
SPM	Salto si es negativo (es decir, menor que 0).
SPMZ	Salto si es < = O (es decir, menor o igual que 0).
SPPZ	Salto si es > =0 (es decir, mayor o igual que 0).
SPU	Salto si no es valido.

T 10.5 Bucles

La operación LOOP (bucle) sirve para llamar varias veces un segmento del programa. La operación LOOP decrementa el ACU1-L en 1. Después comprueba el valor del ACU1-L . Si no

es igual a O, se ejecuta un salto a la meta indicada en la operación LOOP. En caso contrario, se ejecuta la siguiente operación.

LOOP	Decremento ACU1-L y salta si ACU1-L no es 1

T.11 OPERACIONES DE CONTROL DEL PROGRAMA

T 11.1 Operaciones de llamada de bloques

CALL	Llamada incondicional de bloques con parámetros o DB de instancia.
UC	Llamada incondicional de bloques sin parámetros.
CC	Llamada condicional de bloques sin parámetros

T.11.2 Operaciones de fin de bloques

BE	Fin de bloque
UCBEA	Fin incondicional de bloque.
BEB	Fin condicional de bloque si RIO = 1.

T.11.3 Operaciones para abrir bloques

Con esta instrucción se pueden abrir bloques de datos DB o bloques de datos de instancia DI y en los direccionamientos no tenemos que poner el Bloque abierto. Cuando se trabaja con varios bloques, es mejor no abrirlos y en el direccionamiento incluir el numero de bloque. Ejemplos: U DB1.DBX 40.2 ; L DB2.DBB 10 ; T DB3.DBB 1 .....

AUF	Abrir bloque de datos.

T.11.4 Operaciones para el Master Control Relay (MCR)

El Master Control Relay (MCR) se utiliza para activar o desactivar circuitos. Las operaciones "=", "S", "R" y "T" dependen del MCR.

BCRSA	Activa el área MCR.
MCRD	Desactiva el área MCR.
MCR(	Salva el RLO y comienza el área MCR.

)MCR	Fin de área MCR.

T .12 OPERACIONES DE CONVERSIÓN DE TIPOS DE DATOS

T.12.1 Operaciones para convertir números decimales codificados en binario y enteros a otros tipos de números.

BTI	Convertir de BCD a entero.
BTD	Convertir de BCD a doble entero.
ITB	Convertir de entero a BCD.
ITD	Convertir de entero a doble entero.
DTB	Convertir de doble entero a BCD.
DTR	Convertir de doble entero a real.

T. 12.2 Operaciones para convertir números reales en dobles enteros

RND	Convertir un numero real en doble entero.
RND+	Convertir un numero real en doble entero redondeando hada arriba.
RND-	Convertir un numera real en doble entero redondeando hada
TRUNC	Convertir la parte entera de un numero real en doble entero.

T .13 LISTA DE OPERACIONES CON ACUMULADORES

Para operar con el contenido de uno o varios acumuladores o registros de direcciones se dispone de las siguientes operaciones:

Introducir pila de ACU
Salir de la pila de ACU
Incrementar ACU 1-L-L
Decrementar ACU 1-L-L
Sumar el ACU 1 al registro de direcciones 1
Sumar el ACU1 al registro de direcciones 2
Estructuración de imagen (operación nula)
Operación nula 0
Operación nula 1

TAK	Intercambiar ACU 1 y ACU 2

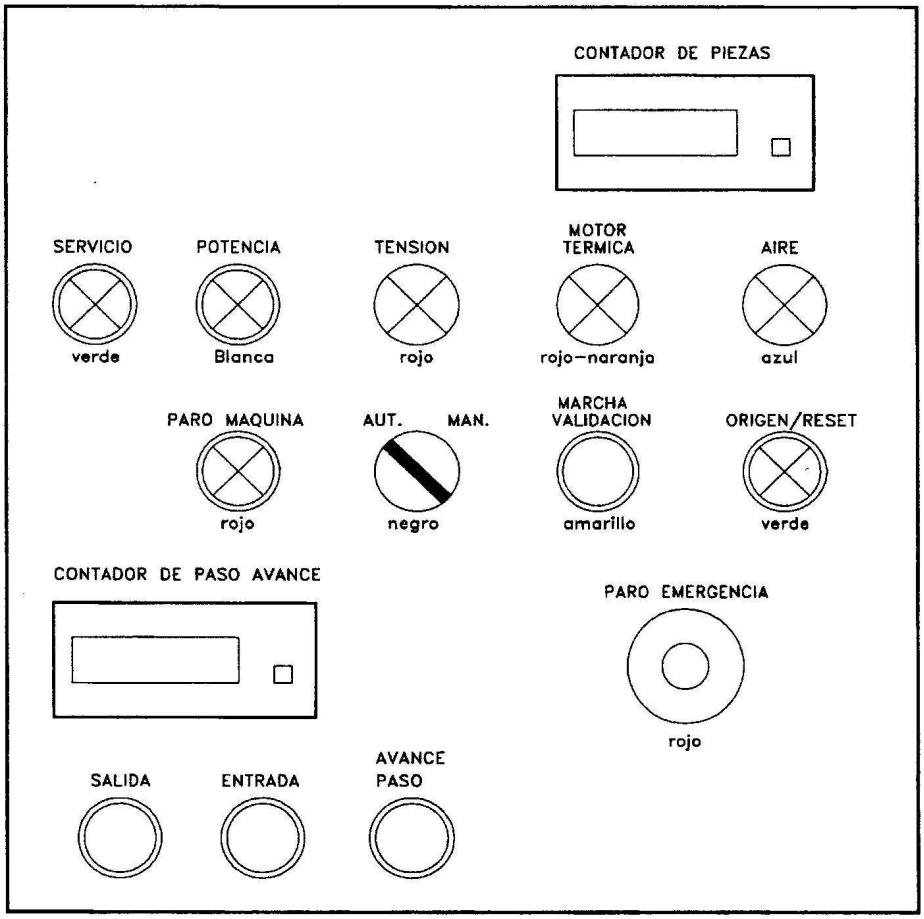
PUSH	CPU con dos acumuladores. copia el contenido completo del ACU 1 al ACU 2. El ACU 1 no se altera
PUSH	CPU con cuatro acumuladores. copia el contenido del ACU 3 al ACU 4, el contenido del ACU 2 al ACU 3 y el contenido del ACU 1 al ACU 2. El ACU 1 no se altera.
POP	CPU con dos acumuladores. copia el contenido completo del ACU 2 al ACU 1. El ACU 2 no se altera
POP	CPU con cuatro acumuladores. copia el contenido del ACU 2 al ACU 1, el contenido del ACU 3 al ACU 2 y el contenido del ACU 4 al ACU 3. El ACU 4 no se altera.
ENT	Introducir pila de ACU. Copia el contenido del ACU 3 al ACU 4 y el contenido del ACU2 al ACU 3.
LEAVE	Salir de la pila de ACU. Copia el contenido del ACU 3 al ACU 2 y el contenido del ACU 4 al ACU 3
INC	Incrementar ACU 1-L-L; o sea solo el byte bajo de la palabra baja (0 a 255)
DEC	Decrementar ACU 1-L-L; o sea solo el byte bajo de la palabra baja (0 a 255)
+AR1	Sumar el ACU 1 al registro de direcciones 1
+AR2	Sumar el ACU1 al registro de direcciones 2
BLD	Estructuración de imagen (operación nula)
NOP 0	Operación nula 0
NOP 1	Operación nula 1

21 - EJEMPLO DE PROGRAMA ESTRUCTURADO PARA UN AUTOMATISMO SECUENCIAL.

Antes de empezar con este ejemplo, vamos a hacer una breve descripción de las funciones de algunos mandos, del cuadro de mando, de un automatismo secuencial. También se incluyen los esquemas de un relé de seguridad, y de un relé de seguridad junto al esquema eléctrico de potencia para una máquina o automatismo. Los relés de seguridad, suelen llevar los contactos por duplicado y con mayor poder de corte. En función del tipo de instalación, las condiciones de seguridad deberán ser mayores o menores.
Las normas (IEC 61508, EN 954-1 ....) exigen que los cuadros eléctricos de mando, lleven relé de seguridad, pues en caso de emergencia este relé asume las principales funciones del Paro de Emergencia; entre ellas pueden estar las que vemos en estos esquemas, que son: **a)** cortar la alimentación a algunas entradas del autómata (+A). **b)** cortar la línea de potencia de los motores (relé KM0). **c)** cortar la línea de presión del aire (EVG).

CUADRO DE MANDO TIPICO DE UN AUTOMATISMO

Hay diversos cuadros de mando para automatismos que se adaptan a cada caso en concreto; uno típico puede ser el siguiente en el que hay: pulsadores, interruptores, lámparas, pulsadores con lámpara, y un contador o un Panel de Operador (OP).

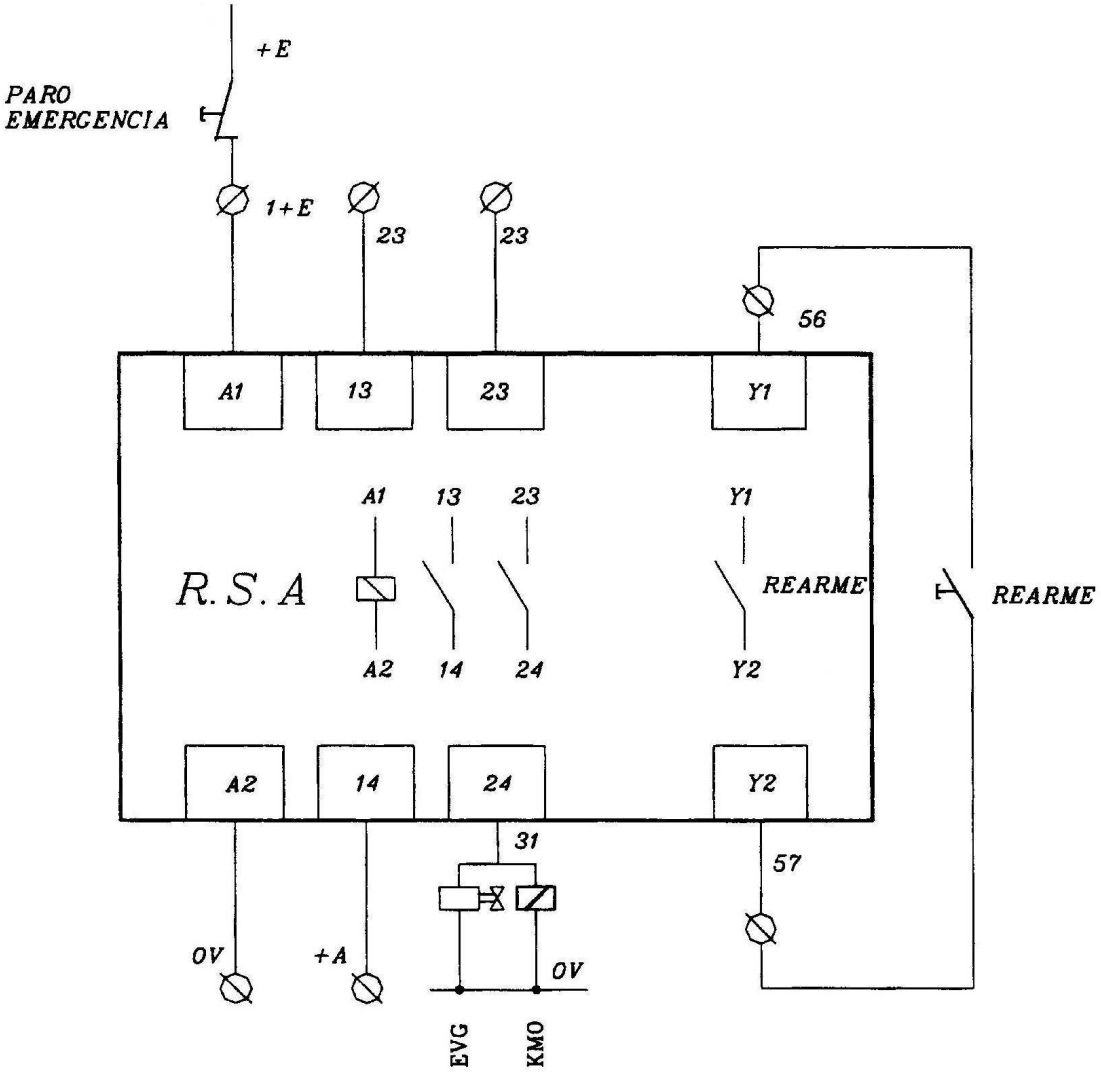


Breve descripción de los mandos con las funciones más normales:

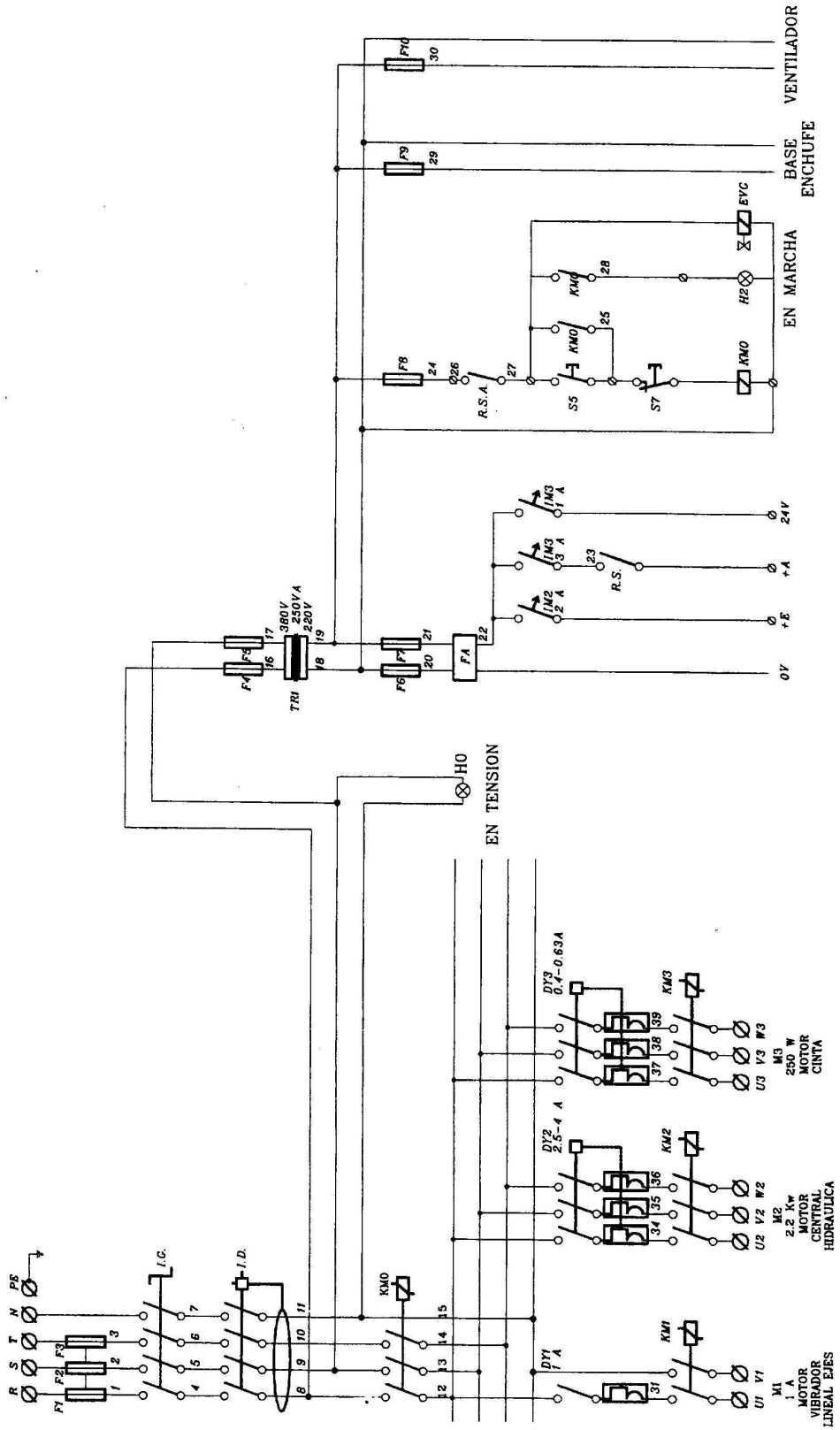
- **Pulsador-piloto de Servicio**.- Al activarlo se tiene al PLC dispuesto para empezar a ejecutar el programa.
- **Potencia**.- Activa un relé que conecta la corriente eléctrica hacia los motores, u otros elementos. Puede llevar también piloto incorporado.
- **Tensión.-** Es una lámpara que nos indica que se ha activado el interruptor general de corriente; puede estar ubicado en el panel de mandos o en el armario eléctrico general.
- **Térmico.-** Lámpara que nos avisa que se ha desconectado algún relé térmico.
- **Aire.-** Lámpara de señalización de que hay suficiente presión neumática.
- **Paro maquina.-** Pulsador-piloto para parar el automatismo o máquina; suele llevar dos contactos, uno para la maniobra eléctrica y otro para conectar al PLC. Esta acción puede tener variaciones; a continuación se detallan tres:
  1. Que pare la máquina al acabar todos los movimientos de la secuencia; supone un paro retardado y mientras tanto se suele hacer que luzca un piloto intermitentemente.
  2. Que se acabe solo el movimiento que se esta realizando; para ello se "tira" el servicio (con esto se deja de "llamar" al bloque que tiene el programa de funcionamiento en automático) y después al accionar el servicio y la marcha, prosigue la secuencia en el punto en que se dejó.
  3. Que además del servicio, se "tire" la potencia eléctrica; es casi un paro de emergencia, pero sin efectuar las acciones especiales que alguna emergencia puede requerir.
- **Marcha.-** Produce la validación de marcha, la maquina arranca si se cumplen las condiciones. También puede llevar un contacto que haga las funciones del pulsador de potencia.
- **Selector Automático/Manual**.- Con él elegimos entre trabajo en automático o en manual. En manual activamos individualmente cada uno de los movimientos de la secuencia; para ello disponemos de un contador de pasos y pulsadores de: avance de paso, salida y entrada (o activado y desactivado). En automatismos complejos o con varios movimientos, en vez de un contador, puede haber un terminal de operador, (sin o con pantalla táctil), en el que además de contadores, podemos programar otras funciones, dar, y recibir datos e información del automatismo.
- **Origen /reset**.- Es un pulsador-piloto que nos pone a cero el contador y lleva a origen los cilindros y otros elementos.
- **Rearme.**- En los automatismos que llevan relé de seguridad, este pulsador produce su rearme. También puede hacer funciones de origen/reset.
- **Paro de emergencia**.- Es un pulsador con enclavamiento y contacto n.c.; su función depende mucho del tipo de maquina. Hay casos, en los que al activar el P.E., se debe realizar una secuencia, o invertir los movimientos que se estaban produciendo; otros en los que se cortan la energía eléctrica, neumática e hidráulica, ... Lo mas normal es parar los motores, "tirar" el aire y la presión de aceite; y poner a cero el grafcet del automático. Un orden de actuación lógico después de accionado el P.E. es el siguiente:
  1. Solucionar las causas y efectos que han motivado el P.E..
  2. Desenclavar el pulsador de P.E..
  3. Rearmar el relé de seguridad si lo hay.
  4. Pasar el selector Aut/Man a Manual.
  5. Activar servicio y potencia.
  6. Estando en manual, hacer los movimientos individuales que sean necesarios y el reset/origen (solo se puede hacer en manual) para poder volver a trabajar en cuanto sé este en condiciones iniciales.
  7. Pasar el selector a automático y accionar la marcha/validación.

RELÉ DE SEGURIDAD

Leyenda:
RSA : Relé de seguridad (contactos propios o del relé auxiliar activado)
+ E: Tensión 24 V. para la alimentación a las tarjetas de entrada en el autómata
+ A : Tensión 24 V. para la alimentación a las tarjetas de salida del autómata.
0 V. : Tensión 0 V. (masa)
EVG : Electroválvula de entrada de aire a la máquina.
KSA : Relé auxiliar para abrir/cerrar contactos RSA (para la acción del P Emerg.)
KM0 : Relé de potencia.
S5 : Pulsador de activación potencia.
S7 : Pulsador de Paro de Maquina.



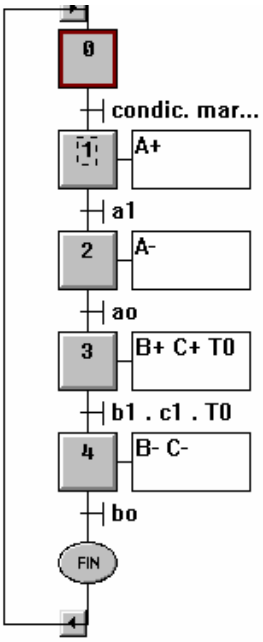
CIRCUITO ELECTRICO DE POTENCIA CON RELE DE SEGURIDAD



EJEMPLO DE PROGRAMACIÓN DE DE UN AUTOMATISMO SECUENCIAL PARA PLC

Un automatismo consta de tres cilindros: A, B y C; que tienen que hacer los siguientes movimientos automáticamente: A+ / A- / B+ y C+ a la vez **y** esperar un tiempo / meterse a la vez estos dos cilindros. En manual se debe poder accionar cada cilindro individualmente con las siguientes condiciones: a) Los cilindros A y B, no deben poder estar fuera a la vez por que chocarían entre sí. b) Para que el cilindro C pueda entrar, debe estar previamente el A dentro.
Los tres cilindros son accionados por electroválvulas 4/2 monoestables. Las asignaciones de entradas y salidas para el PLC son las que figuran en la tabla de símbolos creada con el programa Step-7.
Para no mezclar conceptos del programa Step-7, en el programa no se van a usar parámetros, ni variables temporales, ni estáticas; por lo tanto no se van a rellenar las tablas de declaración de variables; aunque haciéndolo, nos ahorraríamos por ejemplo, casi todas las marcas.

La siguiente figura, muestra el GRAFCET de los movimientos en automático



A continuación, van la tabla de símbolos y los bloques de programación para este ejemplo.

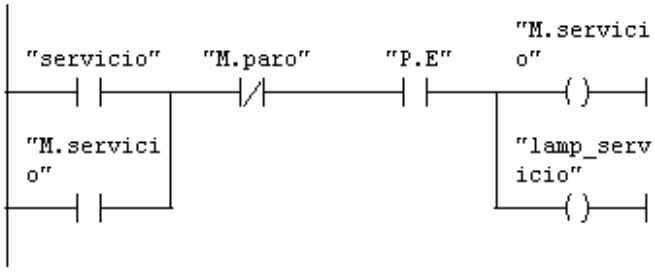
TABLA DE SIMBOLOS

Símbolo	Dirección	Tipo de datos	Comentario
Cycle Execution		OB 1	
válvulaA	A 124.0	BOOL	válvula que activa al cilindro A
válvulaB	A 124.1	BOOL	válvula que activa al cilindro B
válvulaC	A 124.2	BOOL	válvula que activa al cilindro C
lamp_servicio	A 125.0	BOOL	Lámpara de señalización de servicio.
Aire	A 125.1	BOOL	piloto de presión neumática
lam-paro	A 125.2	BOOL	lámpara de paro
lamp_termico	A 125.3	BOOL	lámpara de señalización de fallo térmico.
ao	E 124.0	BOOL	f.c. cilindro A dentro
a1	E 124.1	BOOL	f.c. cilindro A fuera
bo	E 124.2	BOOL	f.c. cilindro B dentro
b1	E 124.3	BOOL	f.c. cilindro B fuera
co	E 124.4	BOOL	f.c. cilindro C dentro
c1	E 124.5	BOOL	f.c. cilindro C fuera
rele_termico	E 124.6	BOOL	contacto del relé térmico
reset_origen	E 125.0	BOOL	reset para el contador y los cilindros
servicio	E 125.1	BOOL	Pulsador de servicio
marcha	E 125.2	BOOL	pulsador de marcha-validación
paro	E 125.3	BOOL	pulsador de paro
manual	E 125.4	BOOL	Selector en manual
automatico	E 125.5	BOOL	selector en automático
presostato	E 125.6	BOOL	Señal de presión de aire suficiente
P.E	E 125.7	BOOL	pulsador de paro de emergencia
avan-paso	E 126.0	BOOL	avance del contador de manuales
salir	E 126.1	BOOL	sale el cil. seleccionado
entrar	E 126.2	BOOL	entra cil. seleccionado
M.reset	M 6.0	BOOL	marca de reset-origen
1ªetapa	M 10.0	BOOL	1ª marca del grafcet del automático
ultima_M.Auto.	M 10.4	BOOL	última marca del grafcet del automático
ultima_M.Man.	M 20.3	BOOL	última marca de comparaciones del manual
M. c.ini.	M 25.0	BOOL	marca de todo en origen (condiciones iniciales)
M.condic.mar	M 30.0	BOOL	marca de en condiciones de marcha
M.paro	M 40.0	BOOL	marca de paro
M.paro interm	M 40.1	BOOL	marca intermedia de paro
M.validación	M 50.0	BOOL	marca de validación
M.servicio	M 70.0	BOOL	marca de servicio
	MB 10	BYTE	marcas del grafcet automático
	MB 20	BYTE	marcas del manual
	T 0	TIMER	tiempo de espera de los cilindros en B+ y C+
pasos_man.	Z 1	COUNTER	contador de pasos en manual

BLOQUE DE SERVICIOS - FC 1

Segm. 1: Marca de servicio

Si esta activa la marca de servicio, se pueden ejecutar los bloques del Automático y del Manual. Desactivan a esta marca la marca de paro y el paro de emergencia.

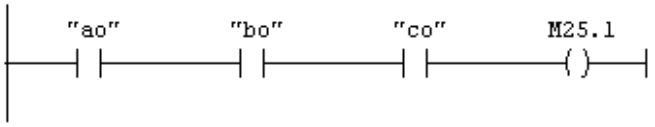


Información del símbolo:

E125.1	servicio	Pulsador de servicio
M70.0	M.servicio	marca de servicio
M40.0	M.paro	marca de paro
E125.7	P.E	pulsador de paro de emergencia
A125.0	lamp_servicio	Lámpara de señalización de servicio.

Segm. 2 : Marca de condiciones iniciales

Entradas de "todo" en origen. Estas entradas, tambien pueden activar un piloto de señalización.

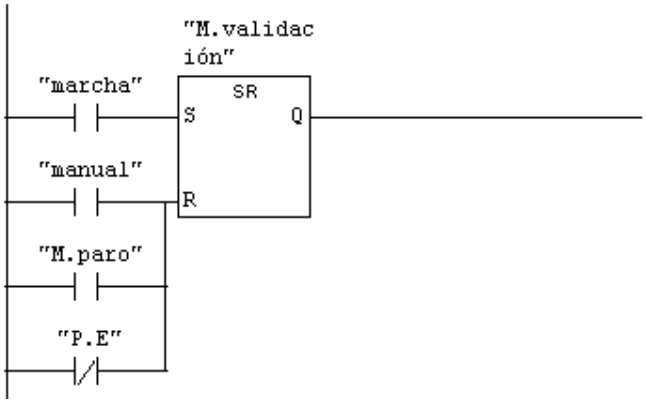


Información del símbolo:

E124.0	ao	f.c. cilindro A dentro
E124.2	bo	f.c. cilindro B dentro
E124.4	co	f.c. cilindro C dentro

Segm. 3 : Marca de validación

El pulsador de marcha, activa la marca de validación. Si pasamos a manual o hemos solicitado paro o pulsamos P.E, se desactiva la marca de validación. Si tenemos contador de ciclos de trabajo, una marca de ciclos alcanzados, tambien nos reseteaia la validación.

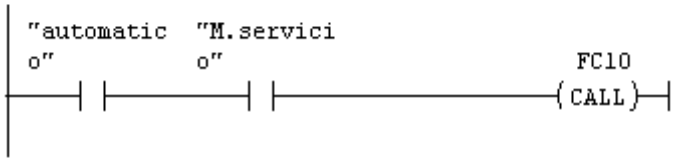


Información del símbolo:

E125.2	marcha	pulsador de marcha-validación
E125.4	manual	Selector en manual
M40.0	M.paro	marca de paro
E125.7	P.E	pulsador de paro de emergencia
M50.0	M.validación	marca de validación

Segm. 4 : Llamada condicional al bloque de automatico

Si el selector está en automático y hay servicio, se produce la llamada al bloque del automático. En ese bloque se deberán cumplir otras condiciones para que la máquina se ponga en marcha.

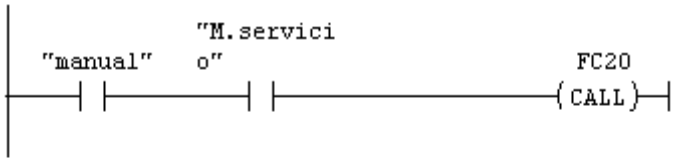


Información del símbolo:

E125.5	automatico	selector en automático
M70.0	M.servicio	marca de servicio

Segm. 5: Llamada condicional al bloque de manuales

Si el selector está en manual y hay servicio se produce la llamada al bloque del manual. Podemos poner tambien la condición de que para pasar a manual se haya acabado la secuencia del automático (que no esté ya la 1ª marca del grafcet, para esto, en este segmento pondremos además el contacto M10.0 en serie y negado y tambien deberemos ponerlo afirmado en paralelo con "automático" en el segmento anterior.

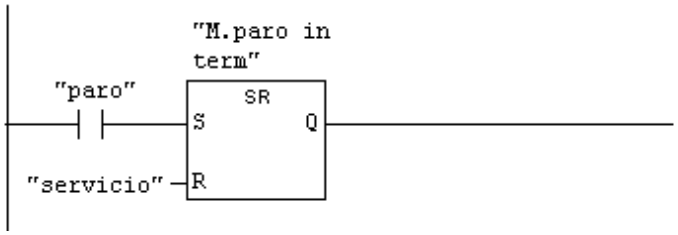


Información del símbolo:

E125.4	manual	Selector en manual
M70.0	M.servicio	marca de servicio

Segm. 6 : Marca intermedia de paro

Al pulsar "paro de la máquina", se activa esta marca intermedia hasta que finalice la secuencia de trabajo. Pulsando servicio se anula esta marca. Si hacemos las conexiones de "paro" a un contacto n.a., esta bien programado; si el contacto es n.c., lo deberemos programar negado.

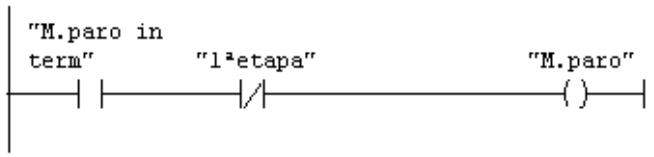


Información del símbolo:

E125.3	paro	pulsador de paro
M40.1	M.paro interm	marca intermedia de paro
E125.1	servicio	Pulsador de servicio

Segm. 7: Marca de paro

Si esta la marca intermedia de paro y no esta la 1ª etapa del grafcet del automático, implica que hemos solicitado parar y se ha acabado la secuencia de trabajo; luego se activa la marca de paro, que nos quita la validación con lo cual, no comenzará otro ciclo de trabajo.

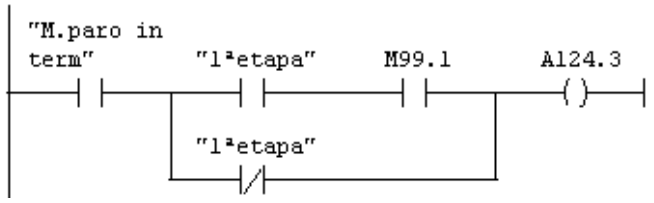


Información del símbolo:

M40.1	M.paro interm	marca intermedia de paro
M10.0	1ªetapa	1ª marca del grafcet del automático
M40.0	M.paro	marca de paro

Segm. 8 : Lámpara de paro

Si esta trabajando la máquina en automático, y se ha activado la marca intermedia de paro, lucirá la lámpara de paro intermitentemente en función de la frecuencia de la marca de intermitencias M 99.1,(o de la de otra que queramos utilizar). Al finalizar la secuencia se desactiva la 1ª etapa del Grafcet y por la rama de abajo se queda fija la lámpara de paro. AL volver a pulsar servicio, se desactivan las marcas de paro y se apagará la lámpara de paro.

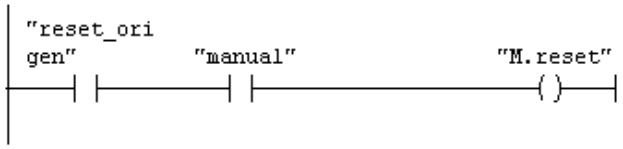


Información del símbolo:

M40.1	M.paro interm	marca intermedia de paro
M10.0	1ªetapa	1ª marca del grafcet del automático

Segm. 9 : Marca de reset-origen

Despues de haber hecho algun trabajo en manual, y estando en manual, es conveniente poner a cero el contador de pasos de manual, y que "todo" vaya a origen para poder empezar sin problemas en automático. Tanto los movimientos en manual como el reset, pueden llevar condiciones: por ejemplo que unos cilindros entren o salgan antes que otros. Este segmento, tambien podria ir en el bloque manual, en él se programan las acciones de esta marca de reset.



Información del símbolo:

E125.0	reset_origen	reset para el contador y los cilindros
E125.4	manual	Selector en manual
M6.0	M.reset	marca de reset-origen

Segm. 10 : Presostato y señalización de presión de aire.

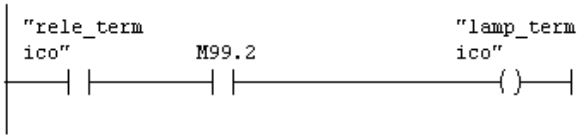
Si hay suficiente presión, se señaliza con una lámpara azul

Información del símbolo:

E125.6	presostato	Señal de presión de aire suficiente

Segm. 11: Aviso de fallos

En este segmento van las entradas en paralelo de todos los fallos que deban tener señalización. Se pueden poner intermitencias.



Información del símbolo:

E124.6	rele_termico	contacto del relé térmico
A125.3	lamp_termico	lámpara de señalización de fallo termico.

Segm. 12 : Reset del contador exterior

Con una salida (A 125.4) ponemos a cero el contador exterior. Con las mismas condiciones, que esté la marca de reset origen o que este la ultima marca de comparaciones del manual; ponemos a cero el contador del PLC en el bloque manual

Información del símbolo:

M20.3	ultima_M.Man.	última marca de comparaciones del manual
M6.0	M.reset	marca de reset-origen

Segm. 13: Contador de ciclos o de piezas

Si hay que contar el número de ciclos en un contador exterior, empleamos la salida A 125.5 para darle los impulsos de contaje. Estos impulsos se dan con la última marca del grafcet del automático.

Información del símbolo:

M10.4	ultima_M.Auto.	última marca del grafcet del automático

BLOQUE DEL AUTOMATICO - FC 10

En este bloque programamos los movimientos o secuencia de movimientos que tiene que hacer la máquina

Segm. 1: Puesta a cero del grafcet manual, y reset contador

Mandamos tres ceros a las marcas de activación de los cilindros en manual, y un 1 para que la última marca haga el reset del contador de pasos.

```
L     2#1000
T     MB    20
```

Segm. 2 : Tiempo de retardo de inicio de ciclo, con paro solicitado

Hay veces que no le da tiempo a la marca de paro a quitar la validación y se haria un ciclo mas antes de parar. Con este temporizador y su contacto en el siguiente segmento, se asegura el paro al final del ciclo en el que se ha pulsado el paro

```
UN    "1ªetapa"                M10.0          -- 1ª marca del grafcet del automático
L     S5T#10MS
SE    T      1
```

Segm. 3 : Marca de todas las condiciones de marcha

Comentario:

```
U     M     25.1
U     "M.validación"           M50.0          -- marca de validación
U     T      1
=     "M.condic.mar"           M30.0          -- marca de en condiciones de marcha
```

Segm. 4 : Grafcet del automático

Comentario:

```
UN    "1ªetapa"                M10.0          -- 1ª marca del grafcet del automático
U     "M.condic.mar"           M30.0          -- marca de en condiciones de marcha
S     "1ªetapa"                M10.0          -- 1ª marca del grafcet del automático
S     M     10.1


U     M     10.1
U     "a1"                     E124.1         -- f.c. cilindro A fuera
R     M     10.1
S     M     10.2

U     M     10.2
U     "ao"                     E124.0         -- f.c. cilindro A dentro
R     M     10.2
S     M     10.3

U     M     10.3
U     T      0
U     "b1"                     E124.3         -- f.c. cilindro B fuera
U     "c1"                     E124.5         -- f.c. cilindro C fuera
R     M     10.3
S     "ultima_M.Auto."         M10.4          -- última marca del grafcet del automático

U     "ultima_M.Auto."         M10.4          -- última marca del grafcet del automático
U     "bo"                     E124.2         -- f.c. cilindro B dentro
R     "ultima_M.Auto."         M10.4          -- última marca del grafcet del automático
R     "1ªetapa"                M10.0          -- 1ª marca del grafcet del automático
```

Segm. 5 : Activaciones en automático

```
U     M     10.1
S     "válvulaA"                A124.0          -- válvula que activa al cilindro A
U     M     10.2
R     "válvulaA"                A124.0          -- válvula que activa al cilindro A

U     M     10.3
S     "válvulaC"                A124.2          -- válvula que activa al cilindro C
S     "válvulaB"                A124.1          -- válvula que activa al cilindro B
U     "ultima_M.Auto."          M10.4           -- última marca del grafcet del automático
R     "válvulaB"                A124.1          -- válvula que activa al cilindro B
R     "válvulaC"                A124.2          -- válvula que activa al cilindro C

U     M     10.3
U     "b1"                      E124.3          -- f.c. cilindro B fuera
U     "c1"                      E124.5          -- f.c. cilindro C fuera
L     S5T#3S
SE    T      0
```

BLOQUE DEL MANUAL - FC 20

En este bloque programamos las activaciones para hacer los movimientos de cada actuador individualmente, pulsando avance o retroceso.

Segm. 1: Puesta a cero del grafcet del automático

Comentario:

```
L     0
T     MB    10
```

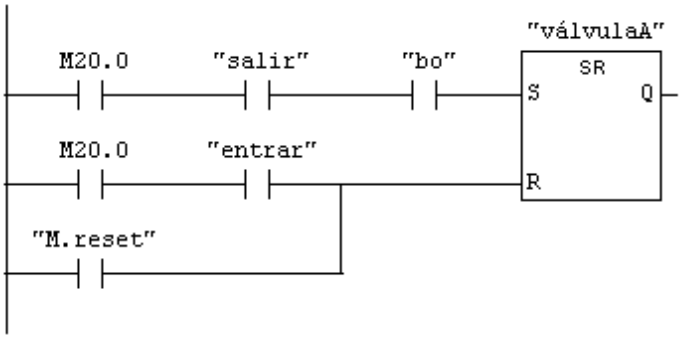
Segm. 2: Contador de pasos en manual.

Con el pulsador de avance-paso, vamos metiendo valores en el contador. Haciendo comparaciones, activamos cada una de las marcas que usaremos despues, para accionar un cilindro u otro. Con la última comparación y su marca o la de reset, volvemos a poner a cero el contador.

```
U     "avan-paso"               E126.0          -- avance del contador de manuales
ZV    "pasos_man."              Z1              -- contador de pasos en manual
U     "ultima_M.Man."           M20.3           -- última marca de comparaciones del manual
O     "M.reset"                 M6.0            -- marca de reset-origen
R     "pasos_man."              Z1              -- contador de pasos en manual

L     "pasos_man."              Z1              -- contador de pasos en manual
L     1
==I
=     M     20.0

L     "pasos_man."              Z1              -- contador de pasos en manual
L     2
==I
=     M     20.1

L     "pasos_man."              Z1              -- contador de pasos en manual
L     3
==I
=     M     20.2

L     "pasos_man."              Z1              -- contador de pasos en manual
L     4
==I
=     "ultima_M.Man."           M20.3           -- última marca de comparaciones del manual
```

Segm. 3 : Movimientos en manual del cilindro A

Si con el contador y las comparaciones, tenemos activada la marca M 20.0 correspondiente al cilindro A, y pulsamos "salir", saldra el cilindro A si el B esta metido (bo). Con la M 20.0 y "entrar", se mete el cilindro A. Al pulsar reset, se activa la marca de reset, y entrarian el cilindro A; el B (Segm.4) y el C, si el A está ya dentro (Segm. 5)

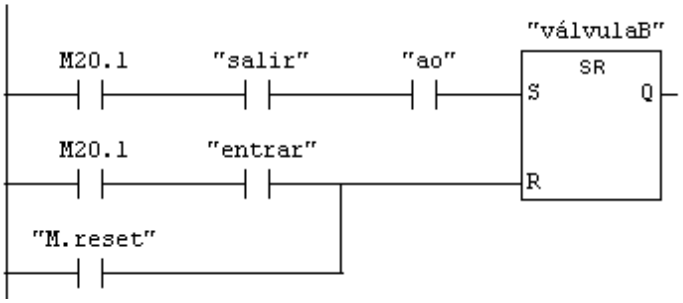


Información del símbolo:

E126.1	salir	sale el cil. seleccionado
E124.2	bo	f.c. cilindro B dentro
E126.2	entrar	entra cil.seleccionado
M6.0	M.reset	marca de reset-origen
A124.0	válvulaA	válvula que activa al cilindro A

Segm. 4 : Movimientos en manual del cilindro B

Si con el contador y las comparaciones, tenemos activada la marca M 20.1 correspondiente al cilindro B, y pulsamos "salir", saldra el cilindro B si el A esta metido (ao). Con la M 20.1 y "entrar", se mete el cilindro B

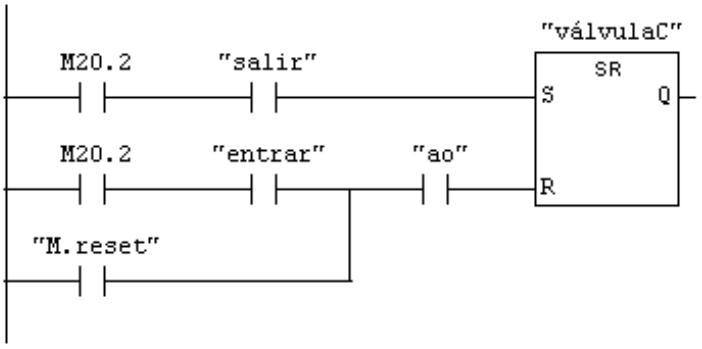


Información del símbolo:

E126.1	salir	sale el cil. seleccionado
E124.0	ao	f.c. cilindro A dentro
E126.2	entrar	entra cil.seleccionado
M6.0	M.reset	marca de reset-origen
A124.1	válvulaB	válvula que activa al cilindro B

Segm. 5 : Movimientos en manual del cilindro C

Si con el contador y las comparaciones, tenemos activada la marca M 20.2 correspondiente al cilindro C, y pulsamos "salir", saldra el cilindro C. Con la M 20.0 y "entrar", se mete el cilindro C, pero con la condición de que el A esté dentro (ao).



Información del símbolo:

E126.1	salir	sale el cil. seleccionado
E126.2	entrar	entra cil.seleccionado
M6.0	M.reset	marca de reset-origen
E124.0	ao	f.c. cilindro A dentro
A124.2	válvulaC	válvula que activa al cilindro C

BLOQUE DE INTERMITENCIAS - FC 5

Segm. 1: Generador de impulsos de una de las formas que hay

```
UN    T     10
L     S5T#10S
SE    T     10
L     T     10
T     MB    99
```

BLOQUE DE ORGANIZACIÓN OB 1

OB1 : bloque de organización

En este caso, este bloque tiene pocas instrucciones por que no hay condiciones para hacer las LLamadas, ni se han programado otras condiciones. En caso de no haber relé de seguridad, podríamos programar un bloque para emergencias en el que básicamente tranferiríamos un "cero" a las marcas del bloque Automático y a las del Manual; tambian mandaríamos entrar o salir a algún cilindro según el caso. En este bloque OB1, al pulsar el P.E se haría la llamada condicional a ese supuesto bloque de Emergencias.

Segm. 1: Lamadas incondicionadas

Las llamadas a los FCs, se pueden hacer con UC o con CALL

```
UC    FC    1
CALL  FC    5
```

TABLA DE REFERENCIAS CRUZADAS.

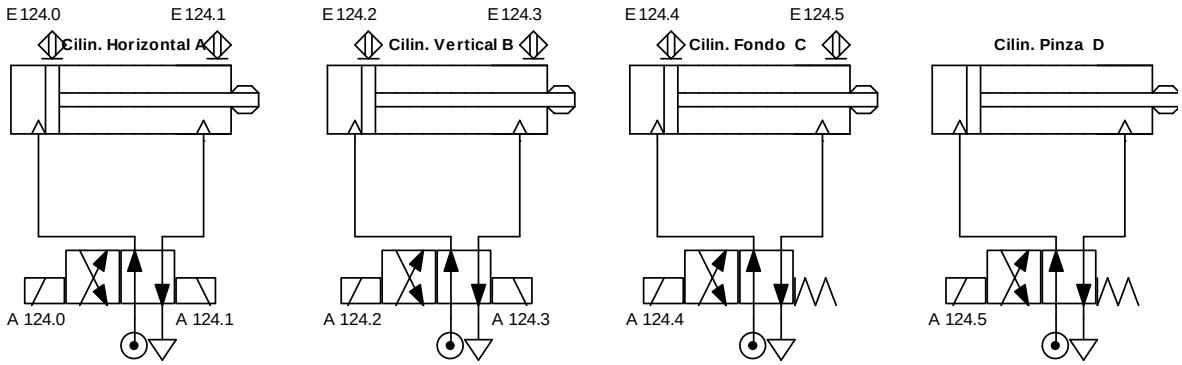
Esta tabla la da el programa Step-7 (Herramientas – datos de referencia – mostrar); sirve para ver en que bloque o bloques hemos programado cada operando y como. Con la opción de Filtrar, se pueden seleccionar los operandos y datos que queremos ver. A continuación van parte de estas referencias

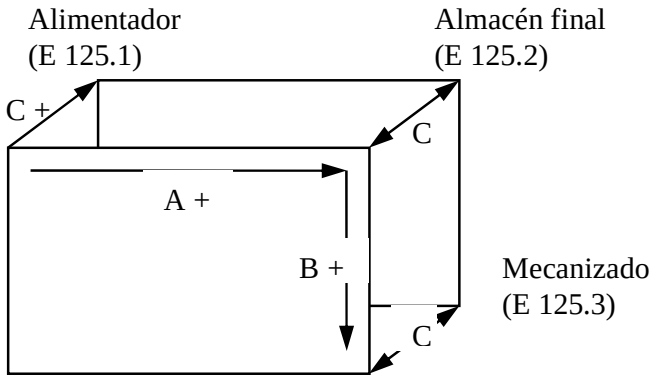
0PERANDO	SIMBOLO	BLOQ.	LENGU.	DETALLES
E 124.0	ao	FC1	KOP	Seg 2 /U
E 124.0	ao	FC10	AWL	Seg 4 Ins 10 /U
E 124.0	ao	FC20	KOP	Seg 5 /U
E 124.0	ao	FC20	KOP	Seg 4 /U
E 124.1	al	FC10	AWL	Seg 4 Ins 6 /U
E 124.2	bo	FC1	KOP	Seg 2 /U
E 124.2	bo	FC10	AWL	Seg 4 Ins 20 /U
E 124.2	bo	FC20	KOP	Seg 3 /U
E 124.3	bl	FC10	AWL	Seg 4 Ins 15 /U
E 124.3	bl	FC10	AWL	Seg 5 Ins 12 /U
E 124.4	co	FC1	KOP	Seg 2 /U
E 124.5	c1	FC10	AWL	Seg 4 Ins 16 /U
E 124.5	c1	FC10	AWL	Seg 5 Ins 13 /U
E 124.6	relé térmico	FC1	KOP	Seg 11 /U
E 125.0	reset origen	FC1	KOP	Seg 9 /U
E 125.1	servicio	FC1	KOP	Seg 1 /O
E 125.1	servicio	FC1	KOP	Seg 6 /U
E 125.2	marcha	FC1	KOP	Seg 3 /U
E 125.3	paro	FC1	KOP	Seg 6 /U
E 125.4	manual	FC1	KOP	Seg 9 /U
E 125.4	manual	FC1	KOP	Seg 3 /O
E 125.4	manual	FC1	KOP	Seg 5 /U
E 125.5	automático	FC1	KOP	Seg 4 /U
E 125.6	presostato	FC1	KOP	Seg 10 /U
E 125.7	P.E	FC1	KOP	Seg 3 /ON
E 125.7	P.E	FC1	KOP	Seg 1 /U
E 126.0	avan-paso	FC20	AWL	Seg 2 Ins 1 /U
A 126.1	salir	FC20	KOP	Seg 3 /U
A 126.1	salir	FC20	KOP	Seg 4 /U
A 126.1	salir	FC20	KOP	Seg 5 /U
A 126.2	entrar	FC20	KOP	Seg 4 /U
A 126.2	entrar	FC20	KOP	Seg 3 /U
A 126.2	entrar	FC20	KOP	Seg 5 /U
A 124.0	válvulaA	FC10	AWL	Seg 5 Ins 4 /R
A 124.0	válvulaA	FC10	AWL	Seg 5 /R
A 124.0	válvulaA	FC20	KOP	Seg 3 Ins 2 /S
A 124.0	válvulaA	FC20	KOP	Seg 3 /S
A 124.1	válvulaB	FC10	AWL	Seg 5 Ins 9 /R
A 124.1	válvulaB	FC10	AWL	Seg 5 Ins 7 /S
A 124.1	válvulaB	FC20	KOP	Seg 4 /R
A 124.1	válvulaB	FC20	KOP	Seg 4 /S
A 124.2	válvulaC	FC10	AWL	Seg 5 Ins 10 /R
A 124.2	válvulaC	FC10	AWL	Seg 5 /R
A 124.2	válvulaC	FC20	KOP	Seg 5 Ins 6 /S
A 124.2	válvulaC	FC20	KOP	Seg 5 /S

22 – EJERCICIO PARA PROGRAMAR UN MANIPULADOR

Programar el manipulador que se especifica a continuación. Tener en cuenta las condiciones de funcionamiento que se reseñan.
Las entradas y salidas necesarias, y que aún no están asignadas, asignarlas de las AW124, EW 124, y E 126.1, E 126.2, E 126.3, que no estén asignadas ya. Los temporizadores, contadores y marcas asignarlos análogos a los del ejemplo. Se adjunta una tabla de símbolos para copiar, si se quiere.

OBJETIVO DE ESTE MANIPULADOR:
Este manipulador coge piezas desde el alimentador y las lleva al plato de amarre de la máquina de mecanizado, y después de un tiempo de mecanizado, las retira al almacén de piezas terminadas.





CONDICIONES DE FUNCIONAMIENTO:

Selector Automático/Manual (Aut/Man)

MANUAL
En esta posición del interruptor se anula el funcionamiento automático de la alimentación de piezas a la máquina, pudiendo realizar el movimiento de cada uno de los actuadores independientemente. Este es el procedimiento adecuado para el proceso de regulación del manipulador o cuando se pretenden reajustar las posicione de los actuadores después de una parada de emergencia. El movimiento de los cilindros A y B no se efectuará si el cilindro C no está introducido.

AV. P.- Avance de Paso. Con este pulsador se selecciona el valor del contador, tanto externo como interno. Cada uno de los valores del contador, o paso, se corresponde con un actuador. El último paso pone a cero los contadores para poder comenzar un nuevo proceso si se desea.
AV. C..- Avance del cilindro, cierre de pinza,.., según el paso correspondiente.
RET. C..- Retroceso del cilindro, según el paso correspondiente.
ORIGEN-RESET.- Su accionamiento sólo es posible en modo manual. Al accionarlo todos los actuadores vuelven a su posición inicial teniendo en cuenta que el cilindro C debe estar dentro para que el resto retorne a su posición de origen.

AUTOMÁTICO:
Este es el funcionamiento ordinario para la manipulación de las piezas. Coge una pieza del alimentador y la deposita en el puesto de mecanizado. La máquina tiene un tiempo de mecanizado de 3 segundos, al cabo de los cuales el manipulador vuelve a coger la pieza mecanizada de la máquina y la deposita en el almacén de piezas terminadas.
Realiza 4 ciclos y se para la manipulación hasta que se active nuevamente el Pulsador de servicio.
Para poder realizar un ciclo es necesario que el alimentador disponga de pieza (E 125.1) que esté libre el puesto del almacén de piezas terminadas (E 125.2) y que la máquina no tenga pieza mecanizando (E 125.3). En el caso de no existir pieza en el alimentador el manipulador deberá esperar a que haya pieza. Así mismo en el caso de encontrarse la máquina con pieza, el manipulador espera a que sea retirada manualmente para poder comenzar el ciclo. La presencia de pieza en el almacén de piezas terminadas no será impedimento para que comience el ciclo, pero sí para que se ejecute la acción de depositar pieza en dicho almacén.

P. VAL- Pulsador de validación de automático. Si estando el interruptor en posición manual se pasa a posición automática, requerirá accionar el "pulsador de validación", para que se efectúe el modo automático y se pueda ejecutar la secuencia.
P. SER.- Pulsador de servicio. Al accionarlo el manipulador se prepara para su puesta en marcha. Dicha Marcha se activará al accionar el pulsador de validación-marcha si se cumplen las siguientes condiciones previas:

- El selector AUT/MAN debe estar en automático.
- Debe existir pieza en el alimentador
- No debe existir pieza en la máquina.
- Todos los actuadores (cilindros), deben de estar en su posición inicial.

2. Condiciones de parada:
P. E.- Pulsador de paro de emergencia. El conjunto máquina-manipulador debería responder coordinadamente al Paro de emergencia, tanto si el problema es causado por la máquina como si es originado por el manipulador. Como en este caso no vamos a controlar a la máquina de mecanizado, el paro de emergencia se limita a los movimientos del manipulador.
Al accionar el pulsador de Paro de emergencia, entra el cilindro C. Si coincidiera que la orden de emergencia se efectúa cuando se está ejecutando el cerrado de la pinza sobre la pieza, la actuación del paro hará retroceder al cilindro pinza D, evitando el amarre de la pieza y realizara posteriormente el retroceso del cilindro C.
P. MAQ.- Paro de máquina. Este pulsador permite terminar la secuencia de la pieza que se está manipulando, pero evita un siguiente ciclo al margen de que no se hayan efectuado las 4 piezas previstas. Para poder realizar una nueva puesta en marcha, previamente habrá que accionar el pulsador de servicio.
Para visualizar el accionamiento del Paro de Máquina, la lámpara correspondiente se activará intermitentemente durante el tiempo que transcurre entre el accionamiento del pulsador y la

parada real de la máquina. En cuanto dicha acción se ejecuta, la lámpara pasa a estado de activación permanente.

Símbolo	Dirección		Tipo de datos	Comentario
Cycle Execut.			OB 1	
lamp.servicio	A	124.0	BOOL	Lámpara de señalización de servicio.
lamp.marcha	A	124.1	BOOL	Lámpara de señalización de marcha.
lamp.rearme	A	124.2	BOOL	Lámpara de señalización de todo en origen.
lamp.paro	A	124.3	BOOL	Lámpara de señalización de parada.
contar.ext.	A	124.4	BOOL	Salida para activar el contaje exterior
reset.contador	A	124.5	BOOL	Salida para poner a cero el contador exterior
val.A+	A	124.6	BOOL	Bobina de avance del cilindro A
val.A-	A	124.7	BOOL	Bobina de retroceso del cilindro A
val.B+	A	125.0	BOOL	Bobina de avance del cilindro B
val.B-	A	125.1	BOOL	Bobina de retroceso del cilindro B
val.C+	A	125.2	BOOL	Bobina de avance del cilindro C
val.C-	A	125.3	BOOL	Bobina de retroceso del cilindro C
val.D+	A	125.4	BOOL	Bobina de avance del cilindro D
pul.servicio	E	124.0	BOOL	Pulsador de servicio
marcha	E	124.1	BOOL	Pulsador de marcha-validación
paro	E	124.2	BOOL	Pulsador de paro
rearme	E	124.3	BOOL	Pulsador de reset-rearme
avan.z	E	124.4	BOOL	Pulsador contar ZV pasos manual
auto/man	E	124.5	BOOL	Selector de modo Automático (E 124.5) / Manual (no E 124.5)
avan.cil.	E	124.6	BOOL	Pulsador de avance de un cilindro en manual
retro.cil.	E	124.7	BOOL	Pulsador de retroceso de un cilindro en manual
P.E.	E	125.0	BOOL	Paro de emergencia
fca0	E	125.1	BOOL	Final de carrera cilindro A dentro
fcfa1	E	125.2	BOOL	Final de carrera cilindro A fuera
fcb0	E	125.3	BOOL	Final de carrera cilindro B dentro
fcb1	E	125.4	BOOL	Final de carrera cilindro B fuera
fcc0	E	125.5	BOOL	Final de carrera cilindro C dentro
fcc1	E	125.6	BOOL	Final de carrera cilindro C fuera
fcd0	E	125.7	BOOL	Final de carrera cilindro D dentro
fcd1	E	126.0	BOOL	Final de carrera cilindro D fuera
pieza1	E	126.1	BOOL	Detector inductivo
pieza2	E	126.2	BOOL	Detector capacitivo
libre	E	126.3	BOOL	
intermitencias	FC	5	FC 5	Bloque de intermitencias
coger	FC	12	FC 12	bloque o subrutina de coger pieza
dejar	FC	14	FC 14	bloque o subrutina de dejar pieza
Manuales	FC	20	FC 20	Bloque de accionamientos en manual
1ªetapa	M	20.0	BOOL	1ª marca del grafcet del automático
ultima_M.Auto.	M	21.0	BOOL	última marca del grafcet del automático
ultima_M.Man.	M	22.5	BOOL	última marca de comparaciones del manual
M.c.ini.	M	25.0	BOOL	marca de todo en origen (condiciones iniciales)
M.ciclos	M	25.1	BOOL	marca de nº ciclos concluidos
M.condic.mar	M	30.0	BOOL	marca de en condiciones de marcha
M.paro	M	40.0	BOOL	marca de paro
M.paro interm	M	40.1	BOOL	marca intermedia de paro
M.reset	M	40.2	BOOL	marca de reset-origen
M.emergencia	M	40.3	BOOL	marca de emergencia
M.servicio	M	50.0	BOOL	marca de servicio
M.validacion	M	50.1	BOOL	marca de reset-0rigen
	MB	22	BYTE	marcas del manual

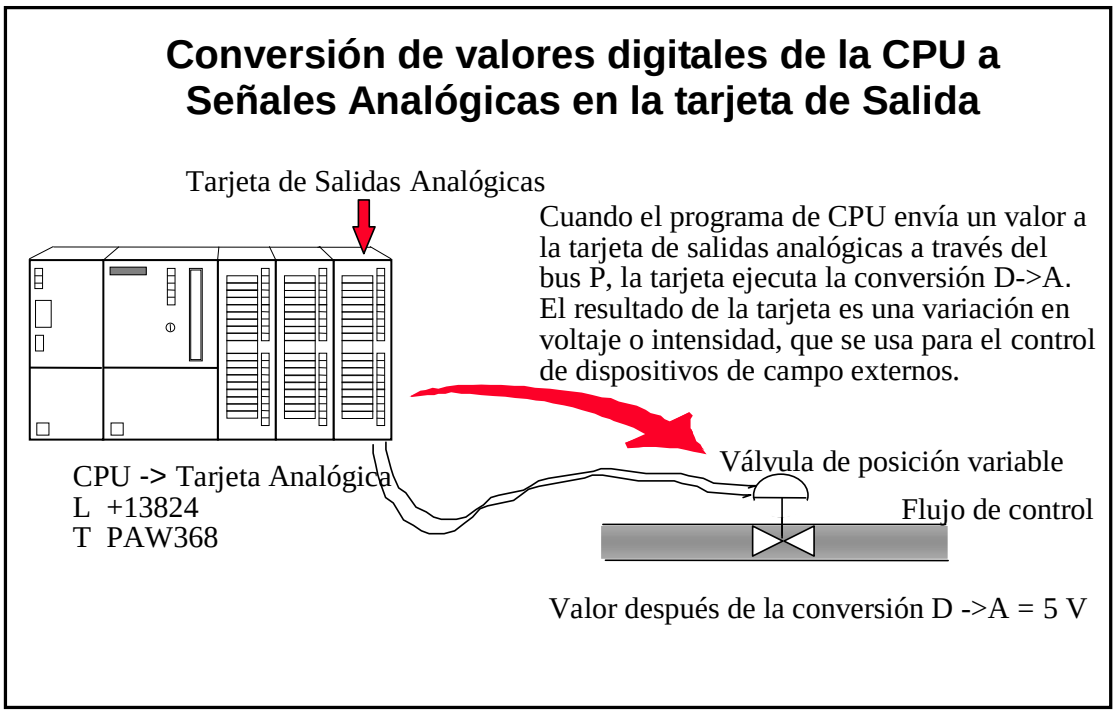
23 - ENTRADAS Y SALIDAS ANALÓGICAS

INTRODUCCIÓN

Además de los valores digitales 1 y 0 (todo/nada) con los que hemos trabajado hasta ahora; en automatismos, hay muchas magnitudes que presentan un margen de variación continuo, o sea, pueden tener distintos valores entre un mínimo y un máximo. Para procesar estos valores, los autómatas pueden tener tarjetas de entradas y salidas analógicas configurables: las de entradas en función de la señal que nos den los sensores y las de salidas en función de la señal de mando que vayamos a emplear. Son señales muy normales: ± 10 V, 0 ÷10 V, ± 80mV, ± 20 mA, 0 ÷20 mA, 4 ÷ 20 mA ....

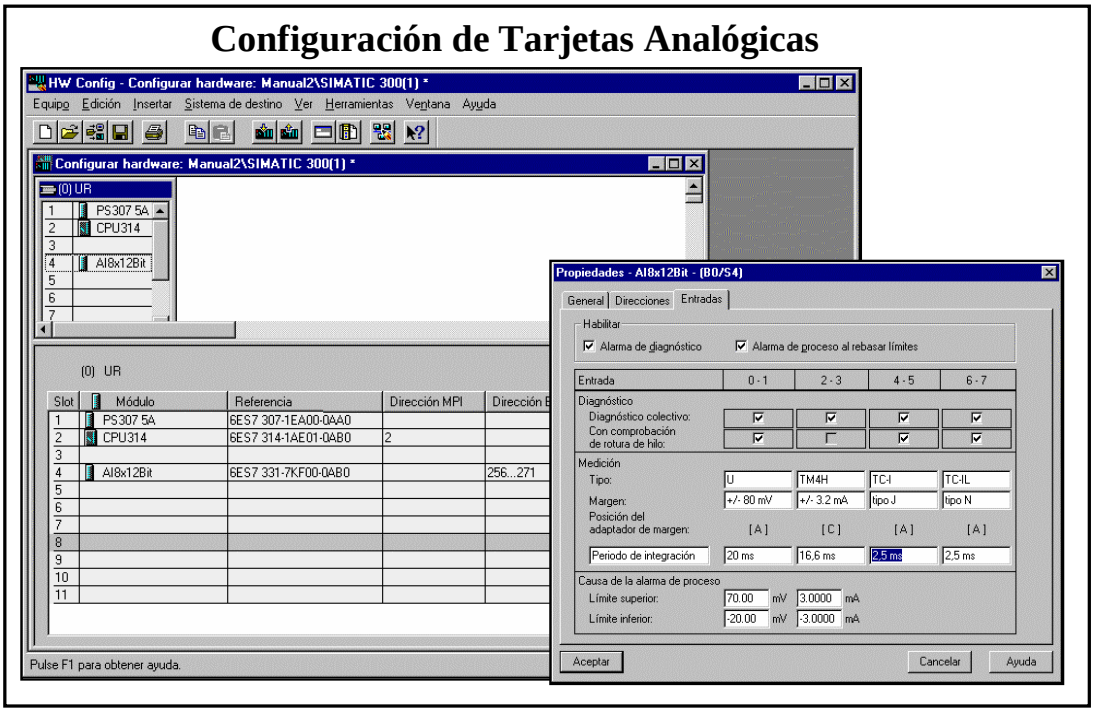
Las tarjetas analógicas de entradas, normalmente convierten el valor analógico del sensor a un valor proporcional digital, comprendido entre 0 y 27648 si la entrada es unipolar; y entre –27648 y 27648 si es bipolar. Las tarjetas de salida hacen lo contrario, convierten estos valores digitales a valores proporcionales analógicos adecuados a la acción que tienen que realizar. La CPU del PLC trabaja con estos valores digitales.

En la siguiente figura vemos este paso de Analógica a Digital para el caso de una tarjeta de salida analógica, 0 – 10 V. que controla a una válvula de posicionamiento variable. Si queremos que la válvula se abra a medias, programaremos que la CPU envíe al puesto de salida el 50% del margen disponible (+13824). La tarjeta debería entonces convertir este número en una señal de +5V que se usaría para abrir la válvula al 50%.



CONFIGURACIÓN DE ESTAS TARJETAS

Con: Equipo ---- Hardware --- entramos en el HV Config para realizar la configuración, al igual que para las tarjetas digitales, insertamos la tarjeta analógica deseada en un slot, hacemos doble clic en esa línea y nos sale una ventana de configuración análoga a la de la siguiente figura:



La figura muestra una tarjeta analógica de 8 entradas distribuidas en 4 canales. Cada uno de estos canales es configurable de forma individual. Así, en el ejemplo, uno de los canales medirá tensión, otro intensidad (transductor de 4 hilos) y los otros dos distintos tipos de termopares. Todos estos ajustes se realizan únicamente por software; esto significa que con una misma tarjeta, tendremos acceso a todo tipo de señales analógicas sin necesidad de tener que colocar ningún hardware adicional, ya que podemos configurar el tipo de señal y el margen.
En el caso de trabajar con medidas de tensión o corriente, tendremos que realizar conversiones de valores digitales a analógicos y viceversa. Con algunos termopares, la PEW correspondiente, nos da directamente la medida en grados centígrados.

PROGRAMACIÓN

El programa de usuario, basándose en los valores digitales 0 a 27648 ó –27648 a 27648 y mediante: cargas (L), comparaciones, saltos, ...realizará las activaciones pertinentes. Para trabajar con valores reales referentes a un automatismo en concreto, podemos hacer conversiones operando matemáticamente; o lo que es más sencillo, podemos usar bloques estándar que nos convierten los valores digitales internos de la CPU en valores de uso práctico. Estos bloques están en: elementos de programa – librerías estándar – TI S7 – FC 105, FC 106.
El bloque FC 105 nos escala: pasa el valor digital (0 a 27648 ó –27648 a 27648 en formato entero) a un valor en formato real comprendido entre unos limites elegidos por el programador que se adaptan al caso en concreto.
El bloque FC 106 nos desescala: pasa un valor en formato real comprendido entre unos limites elegidos por el programador que se adaptan al caso en concreto, a un valor digital apto para la CPU (0 a 27648 en formato entero). Seleccionando estos bloques y pulsando la ayuda de F1 se ven los detalles de cómo rellenar los parámetros de estos bloques.

EJEMPLOS

1º) - Vamos a hacer un programa, para que los 0-10 V. del potenciómetro de la entrada PEW 128 activen un rango de 3V. a 6V. en la salida PAW 130 del entrenador de prácticas. Tenemos que cargar en el PLC el OB1 y el FC 105 que nos va a generar el programa de usuario.

OB 1:
Segm. 1: Escalar
Comentario: Los 0-27648 valores digitales con los que trabaja internamente la entrada analógica PEW 128, los vamos a pasar (escalar) a 8294,4-16588,8 valores, que corresponden a 3 y 6 en una escala de 0-10 V. Cuando en el potenciómetro, seleccionemos 0 V. en MD50 tendremos 8294 valores (en nº real); y cuando seleccionemos 10 V. tendremos en MD50 16588 valores (en nº real). Esto se puede hacer con cálculos matemáticos, pero emplearemos el bloque FC 105 que nos ofrece el programa S-7 para esto; esta en: elementos de programa - librerías - estándar - TI S7 - FC 105; lo insertamos en el programa y después rellenamos los parámetros que nos ofrece:

```
CALL  "SCALE"                 / Call FC 105
 IN    :=PEW128               / entero
 HI_LIM :=1.658880e+004       / real
 LO_LIM :=8.294400e+003       / real
 BIPOLAR:=FALSE
 RET_VAL:=MW54
 OUT   :=MD50                 / real
```

Segm. 2:
Comentario: Cargamos lo que tenga MD 50 (en nº real), lo convertimos a valor en entero (RND) y lo cargamos en la salida analógica PAW 130. A las E/S analógicas solo les podemos meter valores en formato entero.

```
L    MD   50
RND
T    PAW  130
```

2º) – Hacer el programa para que mediante un sensor de temperatura de 0 a 40 ºC conectado a la PEW 128 (0 a 10 V), se conecte la salida A 124.0 a partir de los 25 ºC, y la A 124.1 a partir de los 35 ºC.
Comentario: Por regla de tres sacamos que a 25 ºC, le corresponden 17.280 valores y que a 35 ºC le corresponden 24.192. Con programar lo siguiente, nos valdría:

```
L     PEW 128
L     17280
>= I
=     A 124.0

L     PEW 128
L     24192
>= I
=     A 124.1
```

3º) – Un ventilador gira entre 0 y 1400 rev/min en función de los 0-10 V. de salida de una tarjeta analógica. Hacer el programa para que al accionar la E 124.1 gire a 500 rev/min, y al

accionar la E 124.2 gire a 1000 rev/min

```
      U    E   124.1
      SPB  quin
      U    E   124.2
      SPB  mil
       L    0.000000e+000
       T    MD   50
       SPA  uns
quin: L    5.000000e+002
       T    MD   50
       SPA  uns
mil:  L    1.000000e+003
       T    MD   50
       SPA  uns

uns:  CALL  "UNSCALE"
      IN    :=MD50
      HI_LIM :=1.400000e+003
      LO_LIM :=0.000000e+000
      BIPOLAR:=FALSE
      RET_VAL:=MW54
      OUT   :=PAW128
```

TIPOS DE EJERCICIOS CON ANALÓGICAS

Básicamente hay tres tipos de ejercicios:

A) Que se vayan conectando salidas digitales a partir de unos valores determinados de una entrada analógica; se hacen mediante comparaciones. (Ejercicio 2º)

B) Que una salida analógica, de distintos valores en función de los pulsadores (entradas digitales) que se activen; se hacen mediante saltos a unas cargas y transferencias a un Unscale. (Ejercicio 3º)

C) En general si nos piden que entre unos valores determinados de una entrada se generen valores proporcionales entre otros valores determinados en la tarjeta de salida; podremos resolver el ejercicio empleando un Scale y un Unscale. Vamos a ver una de las formas de hacerlo basándonos en el siguiente ejemplo:

Un detector de posición de 0 a 100 mm (0-10 V), se conecta a una entrada analógica y queremos que para 40 mm; un equipo hidráulico (0 – 2000 N) mandado por una salida (0 - 10 V) produzca ya una fuerza de 1000 N, que seguirá aumentando y será de 1800 N a los 60 mm.

- Programamos un Scale para pasar los 0-27648 valores enteros a 0 - 100 mm reales en la salida Out: = MD 50, por ejemplo.
- Dibujamos una línea con la escala 0-100; partiendo del 40 bajamos a otra línea de escala y le ponemos el valor 1000, partiendo del 60 mm bajamos también a la segunda línea y le ponemos el valor 1800.
- Completamos esta segunda línea hasta sus extremos 0 y 2000 y a partir de estos extremos, subimos gráficamente (u operando matemáticamente) hasta la primera línea; los valores de llegada a esta primera línea, nos dan los limites Hi y Low del Unscale, que en este caso son 65.0 y 15.0. Si completamos los parámetros del Unscale siendo In igual a MD 50 y asignamos en Out la salida analógica; estaría resuelto el ejercicio.

24 - PRACTICA DE PROGRAMACIÓN DE LA MAQUETA DE SELECCIÓN DE RODAMIENTOS

Hacer el programa de PLC para la estación SMC de selección de rodamientos. En esencia el sistema consta de: A) Un cilindro que alimenta rodamientos hasta un detector de presencia de rodamiento. B) Un cilindro giratorio con pinza que traslada el rodamiento hasta la zona de medición-selección. C) El sistema de medición compuesto por un centrador, un cilindro elevador con potenciómetro de medición analógico y un expulsor de rodamiento no correcto. D) Sistema de traslado de rodamiento bueno, compuesto por un cilindro vertical que traslada a un cilindro giratorio, y este a su vez a una pinza.
Debe funcionar en Manual y en Automático para numero indeterminado de ciclos. Las E/S de PLC disponibles son: **a)** EW 124 y E 126.0 a 126.3. **b)** AW 124. **c)** Analógicas, PAW 128, PEW 130, PEW 132 y PEW 134.
Observando el funcionamiento de dicha estación, hay que determinar el GRAFCET práctico y las posibles interferencias que se puedan producir en el funcionamiento manual.
En el bloque de servicios, hay que programar lo habitual excepto el paro de emergencia (P.E.), que lo lleva la estación independientemente del programa de autómata.
A continuación van el esquema neumático y la tabla de símbolos si se quiere usar esta.

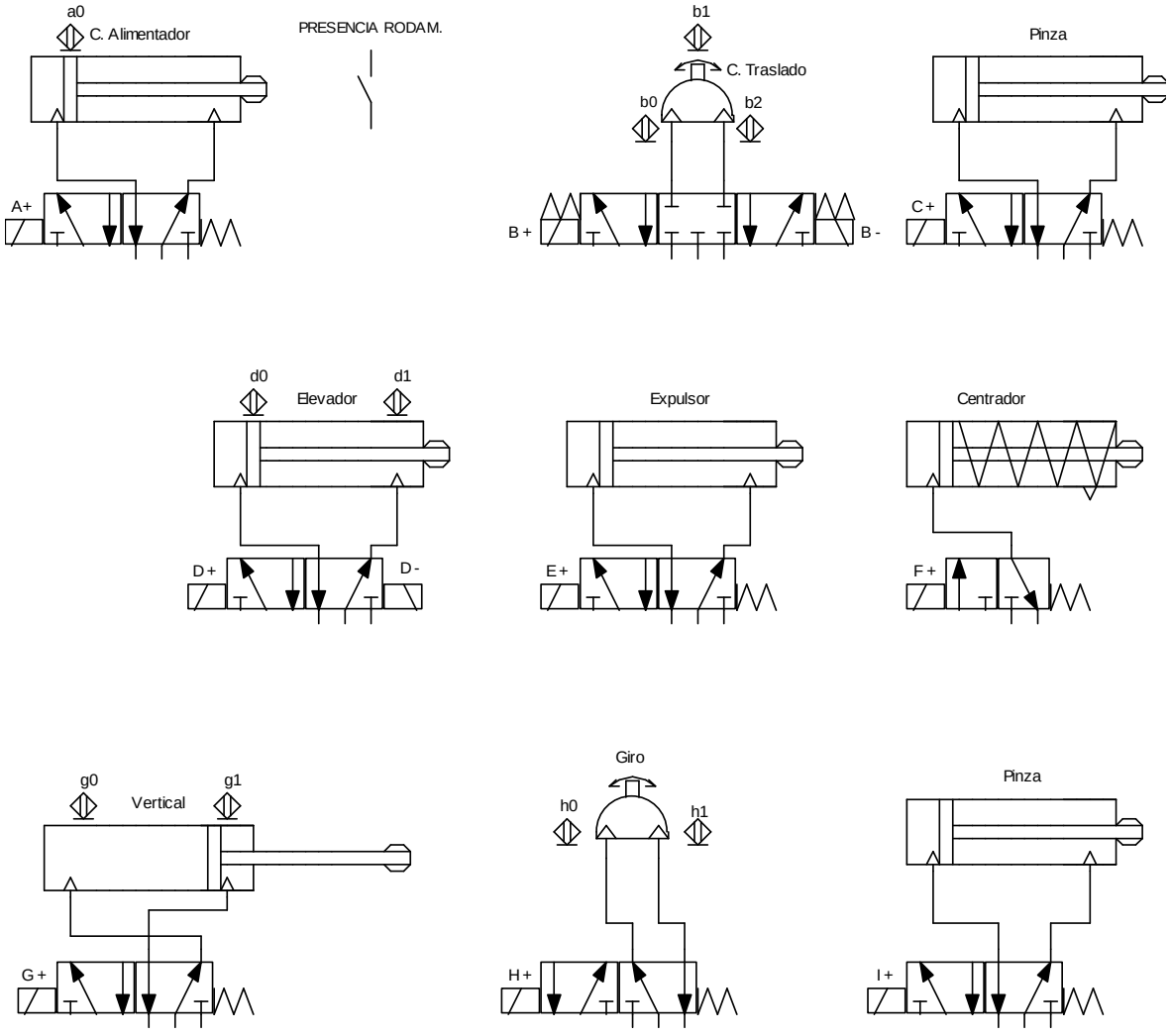


TABLA DE SIMBOLOS

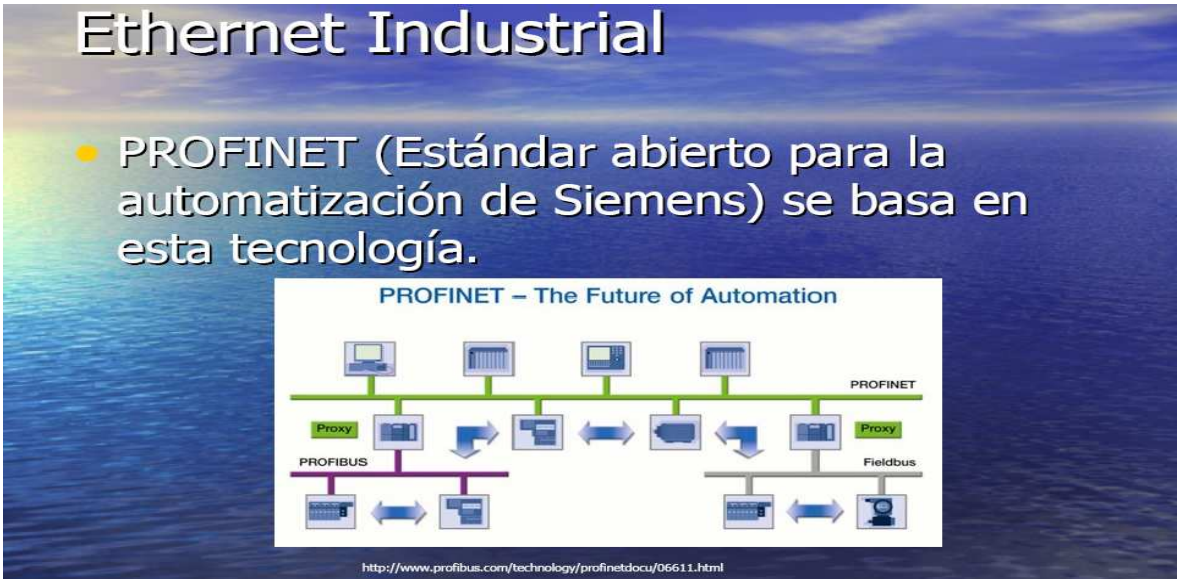
Símbolo	Dirección	Tipo de datos	Comentario
Cycle Execution		OB 1	
cilin. A	A 124.0	BOOL	set, alimenta; reset retrocede
B+	A 124.1	BOOL	B+ (va a por rod.)
B-	A 124.2	BOOL	B- (trasladar rod. para medir)
pinza C	A 124.3	BOOL	set, coger; reset dejar
D+	A 124.4	BOOL	sube cil. elevador
D-	A 124.5	BOOL	baja el cil. elevador
cilin. E	A 124.6	BOOL	set expulsa roda.; reset retrocede
cilin. F	A 124.7	BOOL	set, centra roda.; reset retrocede
cilin. G	A 125.0	BOOL	set, baja; reset sube
cilin. H	A 125.1	BOOL	set, gira a coger; reset gira a dejar
pinza I	A 125.2	BOOL	set, coger; reset dejar
lam-paro	A 125.3	BOOL	lámpara de paro
marcha	E 124.0	BOOL	pulsador de marcha
paro	E 124.1	BOOL	pulsador de paro
auto/man	E 124.2	BOOL	selector automático/manual
rearme	E 124.3	BOOL	reset para el contador y los cilindros
ao	E 124.4	BOOL	cilindro A atras
pr	E 124.5	BOOL	presencia de rodamiento
bo	E 124.6	BOOL	cilindro B atras (dejar)
b1	E 124.7	BOOL	cilindro B centro
b2	E 125.0	BOOL	cilindro B adelante (coger)
do	E 125.1	BOOL	cilindro elevador D, abajo
d1	E 125.2	BOOL	cilindro elevador D, arriba
go	E 125.3	BOOL	cilindro G, abajo
g1	E 125.4	BOOL	cilindro G, arriba
ho	E 125.5	BOOL	cilindro H, atrás (dejar)
h1	E 125.6	BOOL	cilindro H, adelante (coger)
servicio	E 125.7	BOOL	Pulsador de servicio
avan-paso	E 126.0	BOOL	avance del contador de manuales
salir	E 126.1	BOOL	sale el cil. seleccionado
entrar	E 126.2	BOOL	entra cil. seleccionado
SCALE	FC 105	FC 105	
m.reset	M 6.0	BOOL	marca de reset-origen
	M 10.0	BOOL	1ª marca del grafcet del automático
M. c.ini.	M 25.0	BOOL	marca de todo en origen (condi. iniciales)
condic.mar	M 30.0	BOOL	marca de en condiciones de marcha
m.paro	M 40.0	BOOL	marca de paro
m.interm paro	M 40.1	BOOL	marca intermedia de paro
m.validación	M 50.0	BOOL	marca de reset-0rigen
M.servic	M 70.0	BOOL	Marca de servicio
	MB 7	BYTE	marcas para discriminar tipo de rodamiento
	MD 10	DWORD	marcas del grafcet automático
	MW 20	WORD	marcas del manual
t14	T 14	TIMER	tiempo para coger las dos medidas analógicas
t15	T 15	TIMER	tiempo espera para asegurar el reset validación
	Z 1	COUNTER	contador de pasos en manual

25 – COMUNICACIÓNES INDUSTRIALES - GENERALIDADES

INTRODUCCION – REDES ESTANDAR

Los altos niveles de automatización industrial, se deben al nivel tecnológico que han alcanzado las maquinas y sistemas programables, y también a la posibilidad de comunicación entre estas maquinas y sistemas, de tal forma que desde ordenadores remotos podemos obtener información de procesos de fabricación, y controlarlos hasta en detalles como el accionamiento de una simple válvula. Las distintas redes de comunicación, necesitan su propio Hardware (tarjetas, cables, conectores) y también software; por ejemplo para Internet necesitamos un programa, la tarjeta (MODEM) y los cables y conectores. Intranet e Internet son redes de uso global, abiertas (WAN); nos vamos a centrar ahora en redes de uso local LAN, que se emplean en procesos industriales, (aunque a su vez, con los medios apropiados también se pueden comunicar con las anteriores).
La mayoría de los autómatas incorporan la posibilidad de comunicarse entre ellos o bien con otros sistemas a través de redes estándar de comunicación. Las redes estándar de comunicación industrial mas usuales son:

- **AS-I**. La red AS-I también llamada Interface Actuador Sensor, se utiliza para conectar los distintos elementos (actuadores y sensores) al autómata a través de un solo cable de 2 hilos. El tiempo de reacción es de 5 ms. y se pueden conectar hasta 124 entradas binarias y 124 salidas en una sola red.
- **PROFIBUS**. La red PROFIBUS tiene 3 divisiones que son PROFBUS-DP, PROFIBUS-PA y PROFBUS-FMS. La FMS, se utiliza para la interconexión de autómatas programables con otros autómatas, con paneles de operador, con ordenadores, etc. realizando tareas de comunicación universal. La PROFIBUS-DP (Periferia Descentralizada) se utiliza para la conexión de estaciones de entradas y salidas remotas, islas de válvulas, etc, con el autómata programable; es una red muy rápida para el manejo de entradas y salidas binarias. La PROFIBUS-PA es igual que el PROFIBUS-DP pero manejando transmisores analógicos, en lugar de señales binarias; además se utiliza en zonas explosivas.
- **ETHERNET INDUSTRIAL.** Se utiliza para conectar un autómata programable con otro autómata o con ordenadores, cuando el volumen de información a intercambiar es muy alto (entorno de Megabytes) y cuando es necesario cubrir grandes distancias. Una variante de esta red es la **PROFINET** que se esta empezando a emplear y es fácilmente compatible con PROFIBUS



TOPOLOGIA DE LAS REDES

La topología o forma física en la que se conectan los elementos o estaciones de una red, puede ser de varias formas, las topologías más normales son:

- **Estrella**; hay una estación central maestra a la que se conectan individualmente el resto de estaciones. Lógicamente la comunicación entre las estaciones periféricas, tiene que ser a través de la estación central o maestra.
- **Anillo**; las estaciones, se conectan una seguida de otra y la "última" con la "primera", cerrando así el circulo; la comunicación pasa por todas las estaciones pero la procesan solo las que se están comunicando en cada instante.
- **Bus**; hay un cable común al que se conectan en derivación cada una de las estaciones; esta es la topología más usual en la conexión entre autómatas, y con la que vamos a trabajar.

MODALIDADES DE TRANSMISION EN LAS REDES LAN

La clasificación se puede hacer atendiendo a diversos criterios; algunos de ellos son:

A) Por la forma de enviar y recibir los bits de información.

- Paralelo; se envían a la vez 8 bits de información mas algunos bits de acompañamiento por cables individuales en paralelo. Esta forma es rápida, pero debido al número de cables que emplea, solo es apropiada para distancias cortas, la emplean la comunicación del PC con la impresora y las comunicaciones internas dentro del PC.
- Serie; aunque lleva 2 o 3 cables, se envían por el mismo cable los bits de información uno detrás de otro intercalados entre bits de control. Esta forma es más lenta, pero más fiable y de mas alcance; es la forma más usada en comunicaciones industriales; el ratón y el teclado se comunican vía serie con el PC.

B) Por el tipo de enlace o transmisión emisor-receptor.

- Simple; comunicación entre dos terminales que solo permite el flujo de datos en un solo sentido, del emisor al receptor.
- Semiduplex; comunicación en ambos sentidos por el mismo cable, pero no simultáneamente.
- Duplex; comunicación en ambos sentidos simultáneamente, pero por cables independientes para emisor y receptor.

C) Por el numero de estaciones

- Conexión punto a punto; es solo entre dos estaciones.
- Conexión multipunto; se pueden conectar varias estaciones a la misma línea o bus.

NIVEL FISICO DE LA RED

A) Tipos de cables

Uno de los problemas que hay en las comunicaciones, son las interferencias de campos eléctricos y magnéticos que pueden afectar a los bajos niveles de tensión y corriente que se emplean en la transmisión de los datos; por este motivo los cables son especiales, los mas empleados son:

- Pares de cables trenzados y apantallados; lo emplea Profibus, el final de este cable suele llevar una resistencia para evitar las distorsiones producidas en los empalmes y finales de línea
- Cable coaxial, es uno de los que utiliza Ethernet.
- Cable de fibra óptica; tiene la ventaja de que no le afectan las interferencias eléctricas, pero es caro y las conexiones las deben hacer especialistas en ello. Para distancias largas se utiliza fibra óptica de tipo mineral (cuarzo), y para las cortas fibra de tipo plástico que es más barata. Los datos se transmiten excitando la fibra óptica con luz infrarroja producida por diodos láser; al final de la línea, otro diodo pasa la señal de luminosa a eléctrica.

B) Enlaces Standard para comunicación serie (Interfaces)

- **RS-232**, es un enlace con dos cables, uno para datos y el otro de masa o común; se emplea en conexiones punto a punto y para distancias de hasta unos 15 m. La velocidad de transmisión pueden llegar hasta los 20.000 baudios (Bits/seg).
- **RS-485;** ha sustituido a la RS-422l original; lleva al menos dos cables de transmisión de datos, (transmisión diferencial) y además trenzados, con lo cual le afectan menos las interferencias. Se emplea en conexiones, punto a punto de distancias superiores a la RS-232 y en conexiones multipunto. La velocidad de transmisión, puede llegar hasta los 10 Mbits/s. Este tipo de enlace o interfaz es el que usa MPI y PROFIBUS.
- **Ethernet**; utiliza cable coaxial y es para conexiones multipunto de mas "densidad" de datos.

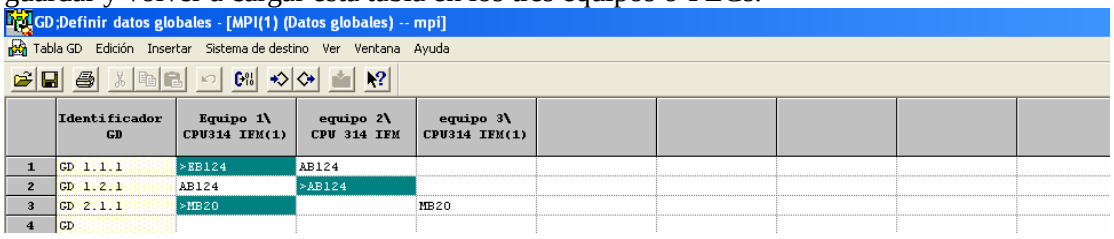
Todos estos enlaces tienen unas normas en cuanto a: normas de montaje, terminales de conexión, etc.

26 - COMUNICACIÓN MPI

La comunicación MPI es especifica de Siemens, no es tan potente como la Profibus, pero es sencilla y barata por que no se necesita mas que el cable de conexión, pues el resto lo lleva el propio PLC. Para realizar esta comunicación, seguimos los pasos siguientes:

- Configurar cada PLC, darle una dirección MPI y seleccionar la red MPI para que se integre en esa Red MPI que se crea.
- Cargar la configuración en cada equipo individualmente.
- Conectar los equipos con un cable Profibus por medio de los puertos MPI y conectar la tarjeta MPI encima en cualquiera de los equipos.
- Subir en el árbol del Administrador Simatic hasta el inicio (proyecto) y seleccionar la red MPI.

- Vamos al menú Herramientas y nos metemos en: definir datos globales, y nos aparecerá una tabla como la siguiente pero vacía. Primero con doble clic en las celdas de los equipos, nos sale una ventana nueva y en esta ventana, también con doble clic en la CPU del equipo que queramos poner en esa celda se va rellenando la cabecera de la Tabla; después seleccionando las celdas de debajo metemos los bits bytes o words que queremos comunicar entre cada PLC. En el menú Ver – factores de Ciclo, podemos ajustar cada cuantos ciclos de Scan se intercambian los datos de comunicación. Después de esto, hay que compilar dos veces, guardar y volver a cargar esta tabla en los tres equipos o PLCs.



	Identificador GD	Equipo 1\ CPU314 IFM(1)	equipo 2\ CPU 314 IFM	equipo 3\ CPU314 IFM(1)					
1	GD 1.1.1	>EB124	AB124						
2	GD 1.2.1	AB124	>AB124						
3	GD 2.1.1	>MB20		MB20					
4	GD								

Arriba, vemos los GDs (Datos Globales) de comunicación. Los signos > indican que se envía; por ejemplo en el GD 1.1.1 los estados en que estén los bits del EB 124 del primer equipo, se envían a los Bits AB 124 del segundo (se podrían enviar también por ejemplo al MB 100 del tercer equipo). Después al hacer el programa de usuario hay que tener en cuenta lo que programamos pues se pueden producir interferencias con estos datos de comunicación.

- Realizar el programa de usuario y cargar los Bloques de programación (al estar montada la red, cada programa se va a su equipo correspondiente).

27 - COMUNICACIÓN PROFIBUS DP

INTRODUCCION

Algunas de las tarjetas de comunicación que tiene Siemens son las: CP 342-2 para comunicaciones AS-I, la CP 343-1 para Ethernet y la 342-5 para Profibus, dentro de esta última hay variantes para protocolos FDL, FMS, S 7, Profibus-DP etc. En los apartados siguientes, vamos a ver fundamentalmente la configuración del hardware para la comunicación Profibus-DP que es una comunicación Maestro - Esclavo en Bus.

1 - PERIFERIA DESCENTRALIZADA

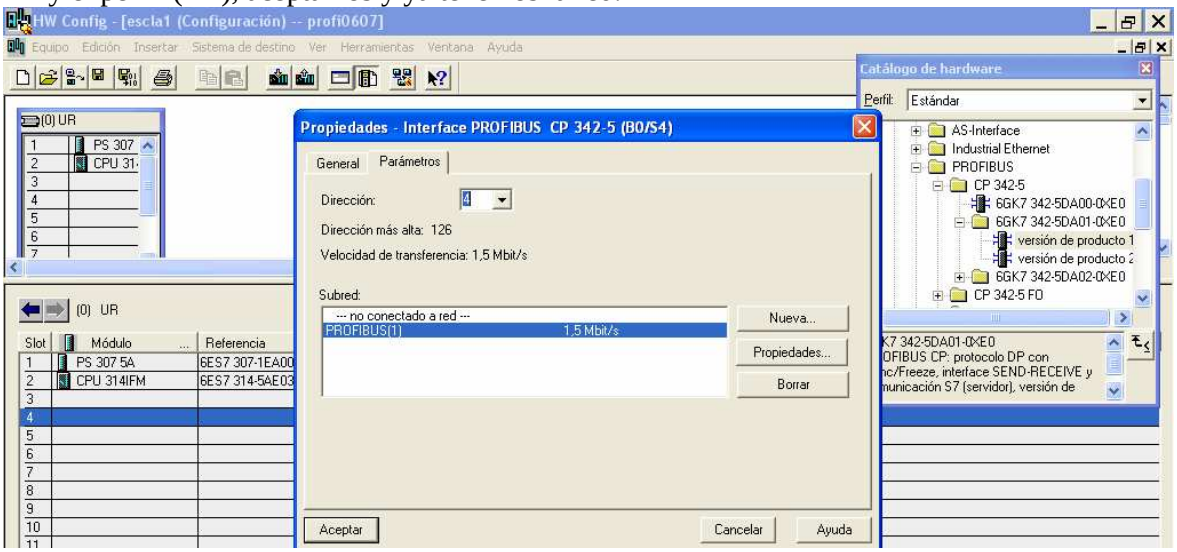
La comunicación Profibus mas sencilla es la Profibus DP; DP viene de Periferia Descentralizada porque consiste en descentralizar lo que son las entradas y salidas de la CPU; es común en las instalaciones poner la CPU del PLC en un armario de control y las entradas y salidas en las zonas de la maquina o instalación en que se encuentran los actuadores, los finales de carrera y los sensores. Los módulos que se instalan en estas zonas son las ETs o estaciones de trabajo que pueden tener entradas y salidas integradas o tarjetas independientes con entradas y salidas o módulos especiales.
Si la CPU tiene la opción DP, simplemente con configurar la CPU como maestra de la red y acoplarle estaciones ET con las E/A correspondientes, podremos hacer el programa de usuario como si todas las E/A estarían en el mismo rack (perfil) que la CPU; para esto, hay que conectar la CPU con las ETs por medio de un cable Profibus y por medio de un selector o microinterruptores, poner en cada ET el nº de estación que le hemos dado al configurarla. Si la estación maestra es una tarjeta de comunicaciones CP, se hace lo mismo, pero en el programa de usuario además hay que programar bloques Send y Recv.
De forma análoga a las ETs se trabaja con las islas de válvulas u otro tipo de elementos pero como pueden ser de otro fabricante que el del PLC, se necesita también un archivo GSD que hay que instalar para que el PLC reconozca a estas islas de válvulas o elementos para acoplarlas al Maestro.

2 – CONFIGURACION HARDWARE DE PROFIBUS ENTRE PLCs

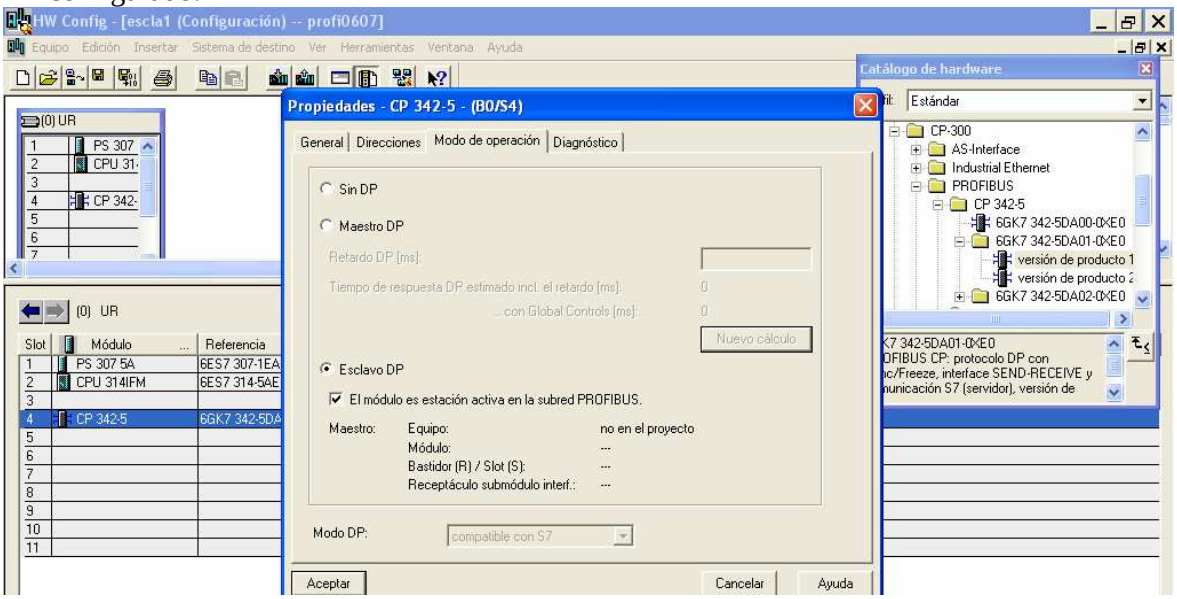
La comunicación Profibus, también se emplea para comunicar PLCs entre si y con otro tipo de equipos; en general para comunicar PLCs entre sí, tenemos que configurar por hardware y por software las direcciones y zonas o áreas de memoria que vamos a emplear para efectuar las comunicaciones de cada PLC; después el método de comunicación general tiene tres pasos: A) En el PLC emisor, pasar por programa la información o datos a comunicar, desde las áreas y direcciones reales de trabajo a sus áreas y direcciones de comunicación ó "buzón" de comunicaciones. B) Profibus, pasa la información del "buzon" ó área de comunicación del emisor al "buzon" del receptor. C) En el PLC receptor, pasar por programa la información o datos que le comunican, desde su "buzón" ó áreas de comunicación a sus áreas y direcciones reales de trabajo.
Podemos configurar los equipos en otro orden, pero el siguiente es adecuado:

A) Configuramos el primer esclavo; al meter la CP, sale una ventana para crear la red, asignamos dirección de red a este equipo o estación, "pinchamos" en nueva o en la red

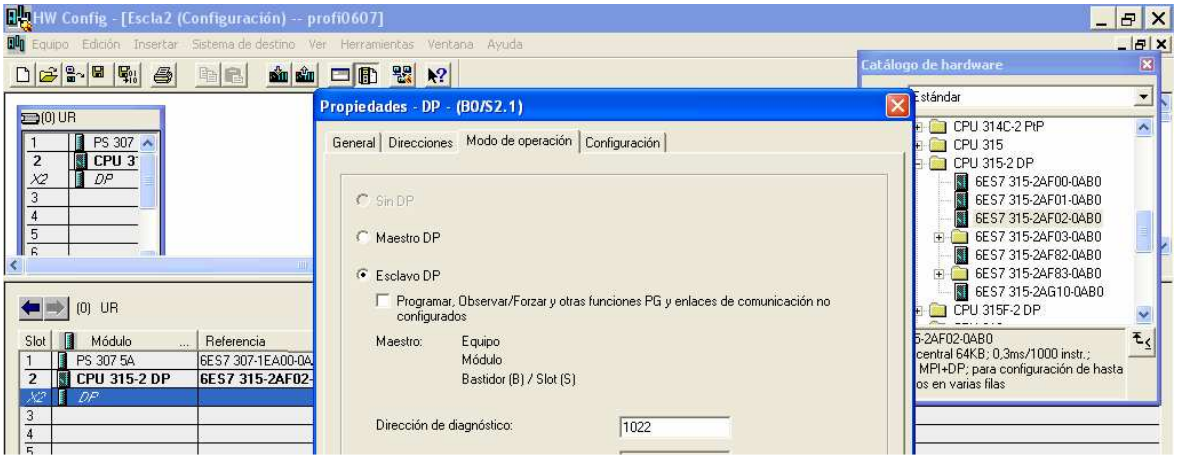
Profibus(1); en propiedades le ponemos nuevo nombre si se quiere y ajustamos la velocidad y el perfil (DP), aceptamos y ya tenemos la red.



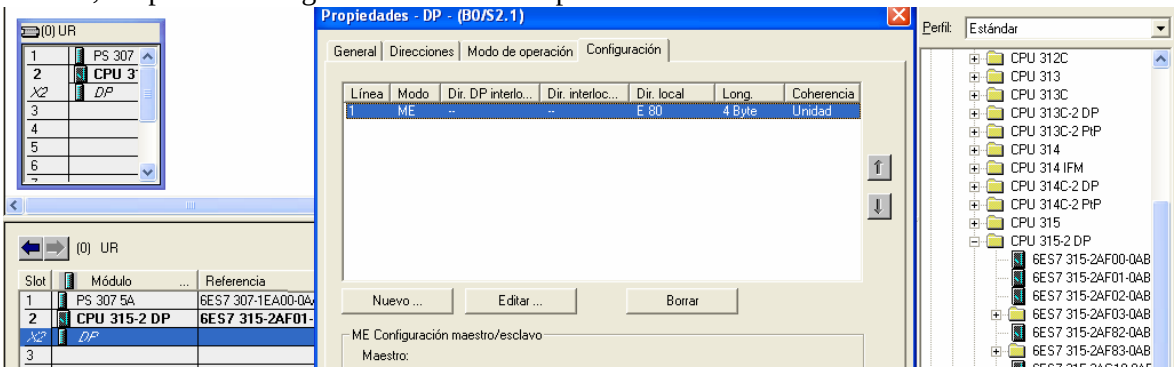
B) Con doble click en el slot de la nueva CP, nos sale otra ventana en la que fundamentalmente nos interesan la dirección de la CP (hay que meterla en hexadecimal en el parámetro CPLADDR de los bloques SEND y RECV, que después tendremos que programar) y el modo de operación (Esclavo DP en este caso); aceptamos, guardamos-compilamos y ya está configurado.



C) Configuramos el segundo esclavo, pero en la primera ventana de la CP, en vez de nueva red, ahora “pinchamos” en la red ya creada y aceptamos, en la segunda ventana hacemos lo mismo. Si los esclavos son 315-2DP, al meter en el slot la CPU y asociarla a la red, por defecto se autoconfiguran como maestros; con dclick en el slot X2-DP nos sale la segunda ventana y podemos cambiarle el modo de operación (la 6ES7 315-2AF00-0AB0, no admite el modo esclavo).

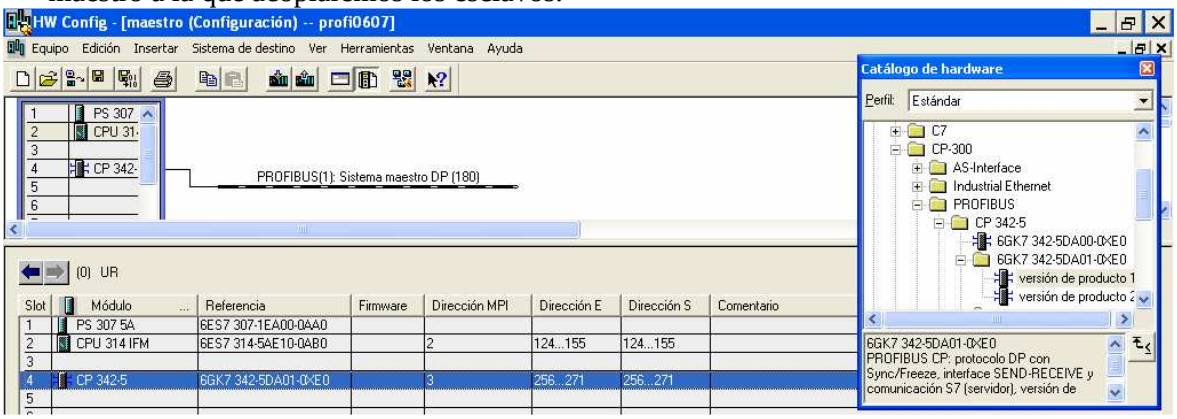


D) Si en la ficha de "Configuración" no metemos alguna dirección de comunicación con el maestro, no se puede guardar la configuración; podemos meter direcciones locales ahora, o después al configurar también las de el maestro y volver a guardar-compilar.
En la figura , hemos configurado que este esclavo va a recibir datos por las direcciones EB 80 a 83; después al configurar el maestro completaremos esta ficha.



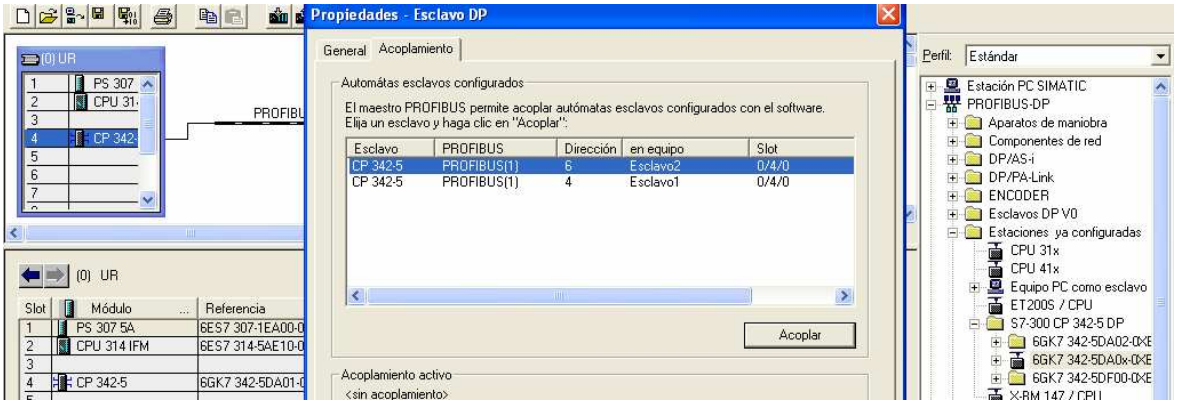
Si los equipos son análogos, una vez configurado uno, podemos copiarlo para los otros y cambiar lo necesario en las fichas que salen al hacer doble click en el slot pertinente.

E) Configuramos el maestro; "pinchamos" en la red ya creada, aceptamos y en la segunda ventana elegimos el modo de operación Maestro DP; al aceptar nos sale la línea del sistema maestro a la que acoplaremos los esclavos.

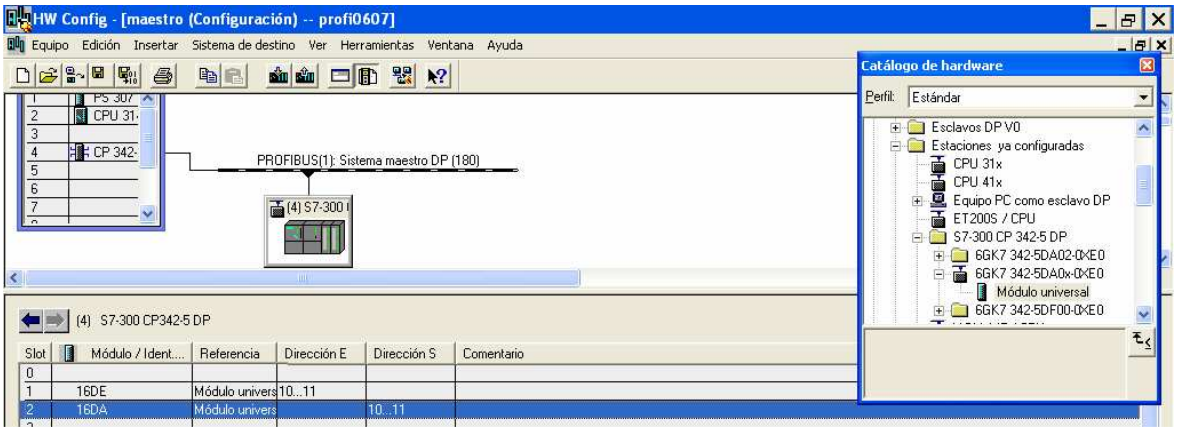


F) Acoplar los esclavos al Maestro: en HV Config del maestro --- catálogo --- Profibus DP --- estaciones ya configuradas; "pinchar" en las CPs (CP 342-5DA0x) que hemos configurado y

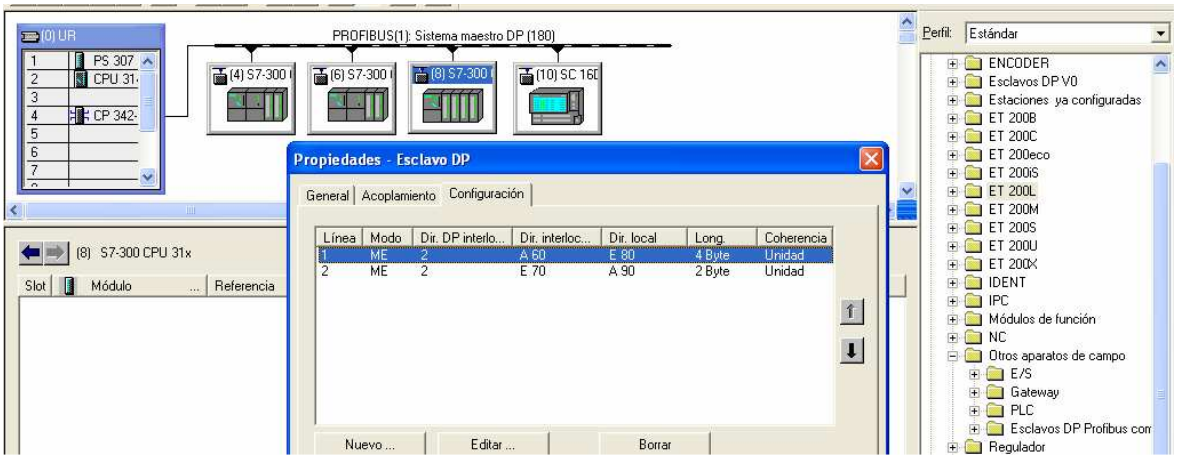
arrastrar hasta la línea del sistema maestro, sale la ventana de acoplamiento: seleccionar esclavo, acoplar y aceptar; también se hace lo mismo activando la línea del sistema maestro y dclick en las CPs,. Si hemos configurado esclavos 315-2 DP, se "pincha" en CPU 31x y se hace lo mismo.



Al maestro, también le podemos acoplar ET 200 (periferia de entradas y salidas) directamente desde aquí, sin haber hecho nada previamente, y también otros aparatos de campo no Siemens si hemos metido previamente su archivo GSD. Hay varios tipos de ET 200, las compactas llevan un numero de E/S determinado y las modulares llevan una "cabeza" IM a la que se le pueden agregar varias tarjetas de E, S o E/S.
Después o a la vez, pinchando en los iconos de los esclavos, metemos módulos universales (desplegando más las estaciones ya configuradas) en los bastidores que se crean y con dclick en los módulos, parametrizamos las direcciones de entrada y salida que el maestro va a usar en la comunicación con cada esclavo: nº, longitud, coherencia,..



Si hemos configurado esclavos 315-2DP; para configurar las direcciones de comunicación, hacemos dclick en estos esclavos y en la ficha de configuración ya nos sale la dirección del esclavo que hemos configurado antes; o bien editamos y completamos las direcciones para el maestro, ó con "nuevo" damos direcciones tanto al Maestro (local) como al Esclavo (interlocutor).



Si hemos acoplado ETs-200, no hay que hacer nada de esto último, la comunicación Profibus – DP, fue diseñada para estas ETs y una vez acopladas, sus E/S se comportan como el resto de E/S que lleva anexas el PLC.

3- ALGUNOS FACTORES A TENER EN CUENTA AL PROGRAMAR COMUNICACIONES PROFIBUS

1) Para pasar un bit, byte, word o dword de un PLC a otro, primero lo pasamos por programa en el PLC-1 a sus direcciones de comunicación (CP o DP); Profibus lo pasa a las direcciones de comunicación del PLC-2 y finalmente por programa en este PLC-2 lo pasamos de sus direcciones de comunicación a la dirección deseada.

2) En la comunicación lo normal es que se envíen y reciban el mismo numero de bytes (words o dwords) ya que estos van emparejados en paralelo byte a byte y bit a bit, por ejemplo si enviamos 4 bytes de marcas de la 40 a la 43 y se reciben por las EB 15 a la EB 18; van emparejados MB 41 con EB 16, o en bits; el M 40.0 con el E 15.0 ó el bit M 42.6 con el E 17.6, etc.

3) En los módulos universales de las CP-342-5 esclavas, configuramos las direcciones (inicio + bytes, words ó dwords) que el maestro va a emplear para comunicarse con ellas y su PLC. Si los esclavos son 315- 2DP, en: propiedades (doble clic en el esclavo), ficha de configuración; además de las direcciones que va a emplear el Maestro (interlocutor), se determinan también las direcciones directas de comunicación del propio esclavo (Local).
   - Si el Maestro es CP, tenemos que meter en su programa los DP_Send y DP-Recv, y en los punteros determinamos: a) El área de memoria por la que se va a comunicar con los esclavos. b) Si las direcciones de las instrucciones de programación en el maestro, son las de los módulos universales en el caso de las CPs, ó las de la ficha de configuración en el caso de los DPs; y si en ambos casos hay que sumarles offset si la dirección del puntero es distinta de 0.0. c) El numero total de bytes de envío o recibo, que tienen que ser respectivamente al menos igual a la dirección mas alta (inicio + bytes) de envío o recibo del conjunto de los esclavos; ya que el Maestro CP envía el paquete completo de bytes desde la dirección 0.
   - Si el Maestro es 315- 2DP, como en los programas de los 315- 2DP no hay ni Send ni Recv, las direcciones configuradas se emplean directamente en el programa de usuario del maestro y las áreas de memoria son de A para enviar y de E para recibir.
4) En los esclavos CP, tenemos que meter DP_Send y DP-Recv y en los punteros respectivos, determinamos: el área de memoria, la primera dirección de envío o recibo que ese esclavo en

concreto puede emplear en sus instrucciones de programación y el numero de bytes de comunicación; o sea: estos esclavos mandan y reciben directamente por las direcciones y áreas que indica el puntero, no hay offset ni otras direcciones a tener en cuenta.

5) En los esclavos 315-2DP, al configurar las direcciones para el maestro (interlocutor), ya hemos determinado también las direcciones directas para el esclavo (local), el numero de bytes y las áreas que son de E/A

4 - EJEMPLO 1 DE COMUNICACIÓN PROFIBUS DP

- El maestro envía el bit E 124.0 al bit A 124.0 del Esclavo1 y al A 124.0 del Esclavo2.
- El Esclavo1 envía el byte EB 125 al byte AB 125 del Maestro.
- El Esclavo2 envía el byte EB 124 al byte AB 124 del Maestro

Programaríamos lo siguiente en el maestro

```
OB1 :  "Main Program Sweep (Cycle)"

Hemos configurado los equipos y la comunicacion, en este caso el maestro coge la
dirección 10-13 para enviar datos y las mismas para
coger datos del esclavo1; las 14-17 las emplea con el esclavo2).

Segm. 1: Título:

Creamos los bloques send y recv por los que va a enviar y recibir los datos el
maestro. Los párametros importantes son el CPLADDR, que es la dirección que ha
adoptado el CP al configurarlo expresada en hexadecimal (en este caso 256 =
100 Hex). Con el SEND y RECV, asignamos que tipo de area de memoria vamos a
utilizar: de imagen de proceso (E ó A), de marcas M, o de bloque de datos;
estas areas tienen que estar libres, si por ejemplo el equipo tiene tarjeta de
salidas con dirección 80, esta dirección no la podemos utilizar como A en
la comunicación.
Tambien en estos parametros ponemos offset de direcciones y n° de bytes
necesarios como mínimo.

      CALL  "DP_SEND"
       CPLADDR:=W#16#100                // la 256 decimal equivale a la 100 Hex.
       SEND   :=P#A 0.0 BYTE 18         // usamos area de salidas para la comunicación, sin offset(0.0) y
       DONE   :=M0.0
       ERROR  :=M0.1
       STATUS :=MW2

      CALL  "DP_RECV"
       CPLADDR :=W#16#100
       RECV    :=P#M 12.0 BYTE 18       //usamos area de marcas y offset 12 que tendremos que sumar a las
       NDR     :=M0.2
       ERROR   :=M0.3
       STATUS  :=MW4
       DPSTATUS:=MB6

Segm. 2 : Título:

Comentario:

      U     E    124.0                 // al pulsar E124.0 en el maestro
      =     A     10.0                 //se activa la A 10.0 (sin offset)que es la dirección que el maestro
      =     A     14.0                 // se activa A 14.0 hacia el esclavo2

      L     MB    23                   //carga el byte 23 (2° de los cuatro del esclavo1=11, 11 más 12 de of
      T     AB   125                   //transfierelo al byte de salidas 125 con tarjeta física de A del mae

      L     MB    26                   //Carga el byte 26(el primero de los cuatro asignados al esclavo2 es
      T     AB   124                   //Los datos anteriores pasalos a las salidas reales AB 124
```

Programaríamos lo siguiente en el Esclavo1

```
OB1 :  "Main Program Sweep (Cycle)"
```

Los esclavos, no saben la dirección que les ha asignado el maestro. Ellos, en su párametro SEND o RECV se ponen un dirección directa (sin offset) y el tipo de memoria (E,A,M,DB)que van a gastar para comunicarse (coger y dejar datos en su buzón), de lo demás se encarga el maestro o la comunicació DP. Por ejemplo si el maestro, ha asignado la dirección 10 de A para enviar al esclavo1 y este en su párametro RECV tiene la dirección 40 de M; la comunicación pasa automaticamente lo que hay en A 10 a M 40, si son 4 bytes pasaria correlativamente todos los datos bit a bit hasta A13 con M 43

Segm. 1: Título:

Comentario:

```
CALL  "DP_SEND"
 CPLADDR:=W#16#100
 SEND    :=P#A 0.0 BYTE 4          //El esclavo1 usa (gasta en su buzón)4 bytes (del 0 al 3) del area
 DONE    :=M0.0
 ERROR   :=M0.1
 STATUS  :=MW2

CALL  "DP_RECV"
 CPLADDR :=W#16#100
 RECV    :=P#M 40.0 BYTE 4         //El esclavo1 usa 4 bytes (del 40 al 43) del area de M para recibir
 NDR     :=M0.2
 ERROR   :=M0.3
 STATUS  :=MW4
 DPSTATUS:=MB6
```

Segm. 2 : Título:

Comentario:

```
U     M     40.0                  //Primer bit del area de datos propia del esclavo1
=     A    124.0                  //Activar la A 124.0 de la tarjeta de salidas real

L     EB   125                    //Cargar el byte de entradas 125 del esclavo1
T     AB     1                    //Lo transferimos al 2° byte de comunicación que emplea el esclavo1
```

Programaríamos lo siguiente en el Esclavo2

```
OB1 :  "Main Program Sweep (Cycle)"
```

Segm. 1: Título:

```
CALL  "DP_SEND"
 CPLADDR:=W#16#100
 SEND    :=P#E 0.0 BYTE 4          //El esclavo2, usa los bytes 0 a 3 del area de entradas para dejar datos
 DONE    :=M0.0
 ERROR   :=M0.1
 STATUS  :=MW2

CALL  "DP_RECV"
 CPLADDR :=W#16#100
 RECV    :=P#E 4.0 BYTE 4          //El esclavo2, usa los bytes 4 a 7 del area de entradas para cojer datos
 NDR     :=M0.2
 ERROR   :=M0.3
 STATUS  :=MW4
 DPSTATUS:=MB6
```

Segm. 2 : Título:

```
U     E      4.0                  //primer bit de datos de comunicación que coge el esclavo2
=     A    124.0

L     EB   124                    //Cargar el byte 124 de entradas fisicas del esclavo2
T     EB     0                    //Transferirlo al primer byte que usa el esclavo2 para dejar datos
```

5 - EJEMPLO 2 DE COMUNICACIÓN PROFIBUS DP

- El maestro envía el byte EB 124 al byte AB 124 del Esclavo y el bit E 125.0 al bit A 124.7 del esclavo.
- El Esclavo envía el byte EB 124 al byte AB 124 del Maestro y el bit E 125.1 al bit A 125.5 del Maestro.

Programaríamos lo siguiente en el maestro

```
OB1 :  Comunicación Profibus DP mediante DBs entre maestro y un esclav
```

```
Profibus DP es rapida y se emplea para comunicar pocos datos; por esto es mas
normal usar E, A ó M para la comunicación, que emplear DBs con los que se
podrian comunicar muchos datos; no obstante en este ejemplo usaremos DBs.
Configuramos-parametrizamos 10 bytes de la dirección 20 a la 29 para enviar
datos al esclavo y 3 bytes de la dirección 5 a la 7 para recibirlos.
Los DBs tienen que tener como mínimo igual nímero de bytes que la dirección mas
alta respectiva; en el DB, se crean facilmente estos bytes con un ARRAY o
matriz de datos.Si las direcciones comenzasen por la 0, seria mas sencillo el
programa y los DBs podrían tener solo los bytes justos.
```

Segm. 1 : FC1 de envio

```
En vez de P#DB1.DBX0.0 BYTE 30; tambien se puede poner con offset por ejem:
P#DB1.DBX25.0 BYTE 30; tendria que ser el DB1 al menos de 25 bytes mas, y en el
segmento 3 sumar este 25 al 20 y 23.5
```

```
      CALL  "DP_SEND"
       CPLADDR:=W#16#100
       SEND   :=DB1.b2                  // este parametro es P#DB1.DBX0.0 BYTE 30, pero se autoes
       DONE   :=M1.1
       ERROR  :=M1.2
       STATUS :=MW4
```

Segm. 2: FC2 de recepción

```
      CALL  "DP_RECV"
       CPLADDR :=W#16#100
       RECV    :=P#DB2.DBX0.0 BYTE 8    //El DB2 tiene 11 bytes, pero con poner byte 8, ya se
       NDR     :=M2.2
       ERROR   :=M2.3
       STATUS  :=MW10
       DPSTATUS:=MB12
```

Segm. 3 : Enviar al Esclavo

```
      L     EB   124
      T     DB1.DBB   20               //Transfierelo al primer byte de comunicación que es

      U     E    125.0
      =     DB1.DBX   23.5             // Pon a 1 el bit 5 (6°) del byte 23 del DB1.
```

Segm. 4 : Recibir del Esclavo

```
      L     DB2.DBB    6               //Carga el 2° byte de comunicación (5 a 7) del DB2
      T     AB   124

      U     DB2.DBX    7.7             //Ultimo bit de comunicación del DB2
      =     A    125.5
```

A la hora de cargar los programas en el PLC, tener en cuenta que hay que cargar todos los bloques, en este caso, además de este OB1, también los FC1, FC2, DB1 y DB2.

Programaríamos lo siguiente en el Esclavo

Comentario: Las direcciones y áreas de memoria usadas para comunicar por los esclavos DP se definen en los parámetros SEND y RECV de sus FC1 y FC2, el nº de bytes tiene que ser igual a los configurados-parametrizados para comunicación**.**

```
OB1 :  "Main Program Sweep (Cycle)"
Segm. 1 : Título:
      CALL  "DP_RECV"
       CPLADDR :=W#16#100
       RECV    :=P#DB1.DBX0.0 BYTE 10   //Este DB1 tiene 13 bytes pero con que tendria los
       NDR     :=M2.2
       ERROR   :=M2.3
       STATUS  :=MW10
       DPSTATUS:=MB12

Segm. 2 : Título:
      CALL  "DP_SEND"
       CPLADDR:=W#16#100
       SEND    :=P#M 40.0 BYTE 3        //Envia los 3 bytes del MB 40 al MB 42
       DONE    :=M1.1
       ERROR   :=M1.2
       STATUS  :=MW4

Segm. 3 : Título:
      L     DB1.DBB    0               //Los esclavos con DB, empiezan su dirección en le
      T     AB   124

      U     DB1.DBX    3.5             //Bit 5 (6°) del byte 3 del DB1
      =     A    124.7

Segm. 4 : Título:
      L     EB   124
      T     MB    41

      U     E    125.1
      =     M     42.7
```

28 - INTRODUCCIÓN A LOS PANELES DE OPERADOR

Para que los operadores de las maquinas y automatismos puedan manejar, controlar y visualizar los procesos que realizan estos automatismos; como mínimo tienen un panel (botonera) de mando con los pulsadores, selectores, lámparas y contadores con los que hemos trabajado hasta ahora. Un escalón superior a esto, son los paneles de operador programables (OP); dentro de estos tenemos los básicos en los que mediante líneas de texto, se pueden programar: avisos, variables, alarmas, funciones equivalentes a las de los pulsadores, etc. En los OPs de gama alta se pueden programar además: gráficos, históricos, iconos, etc y algunos ya disponen de pantalla táctil TP y la posibilidad de representar controlar y visualizar de forma grafica todo el proceso. Una función análoga a estos últimos paneles, la realizan los programas SCADA pero con mas potencia y posibilidades.
Ahora nos vamos a centrar en los OPs básicos y en concreto en el OP-7 que es el que tienen algunos puestos de la Célula Flexible; este OP se programa mediante el Programa "ProTool" de Siemens y dentro de este nos vamos a basar en la versión ProTool/Pro CS V6.0.
Básicamente con este programa hacemos lo siguiente, en este u otro orden:

- Programamos variables en el proyecto o programa del OP y las asociamos a determinados bits de un proyecto del PLC.
- Programamos punteros de área para imágenes con los que se determina que bit del PLC es activado por la imagen que aparece en la pantalla del OP.
- Programamos punteros de área para avisos con los que se determina que bit del PLC activa un determinado aviso para que aparezca en la pantalla del OP; o sea: lo inverso de lo que se hace en el puntero de imágenes.
- Programamos las imágenes y subimágenes que van a salir en la pantalla o display del OP. En estas imágenes además de textos de información, programamos: A) Teclas de función que después al accionarlas nos cambiaran a otra imagen, activarán un bit del PLC ó realizarán otras funciones. B) Campos de entrada y/o salida de datos en los que podemos ver o meter datos en la variable del PLC asociada a este campo del OP. Los cambios entre imágenes y subimágenes, también se pueden hacer con las teclas del cursor y ESC según los casos.
- Programamos los avisos, que son los textos que van a salir en la pantalla o display del OP si el programa del PLC lo manda por medio del puntero de avisos. En los avisos, también podemos programar Campos de salida de datos. Después los cambios a otros avisos se pueden hacer con las teclas ENTER ó ACK según los casos.

Con lo anterior:

- El operador desde el OP básicamente puede: A) visualizar datos del programa del PLC y meter datos al programa del PLC por medio de los campos de datos. B) manejar en modo manual el programa y su automatismo asociado; por ejemplo, si en la pantalla del OP metemos la imagen (subimagen) de activación del cilindro X, automáticamente se activan dos bits del puntero de imágenes, y si en esa subimagen hay programada una función de activación de un bit, al pulsar la tecla correspondiente, se activa otro bit mas; con estos tres bits, de forma análoga a como se programan las activaciones en manual con un contador de pasos, activamos el cilindro X.
- Automáticamente el programa del PLC nos puede activar avisos en la pantalla del OP; por ejemplo, si en el PLC hemos programando que si esta pulsado el P.E. se active un bit del puntero, al pulsarlo se activará el aviso correspondiente a ese bit.

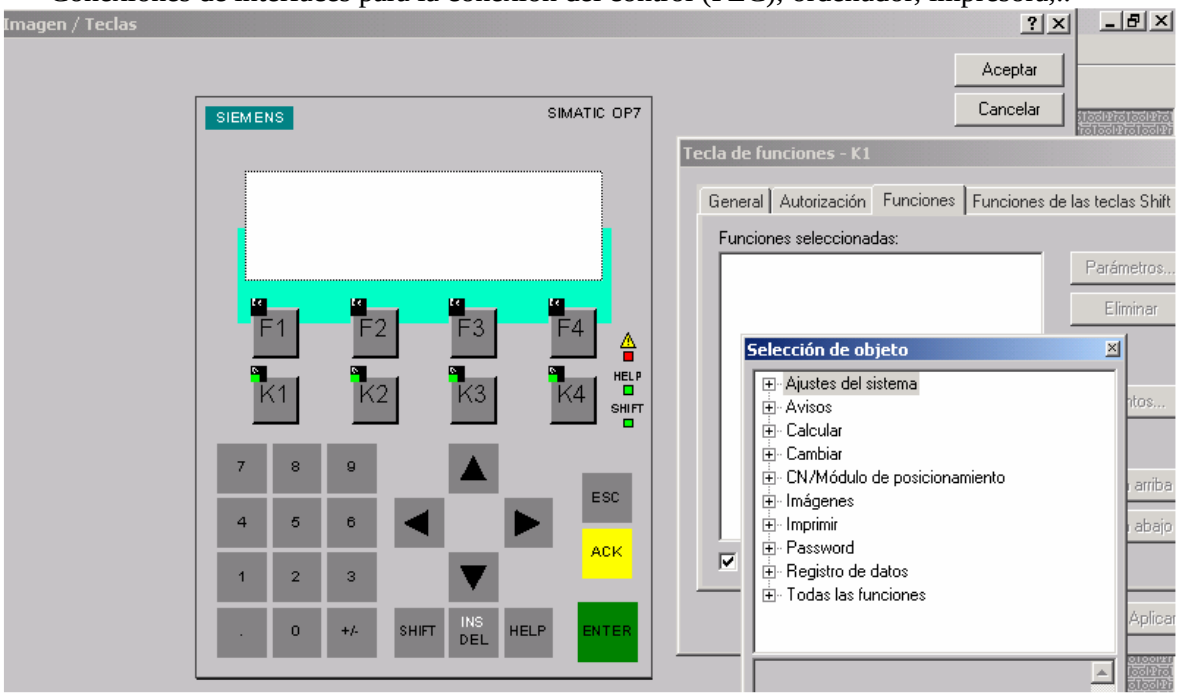
1 – ESTRUCTURA Y FUNCIONES DEL OP-7

Estructura del Panel de Operador OP-7

Es una "caja" rectangular preparada para su acoplamiento en un armario o botonera, su cara frontal o panel quedaría a la vista y en posición de manejo. En la siguiente figura se ve la cara

frontal junto con la ficha de programación de teclas K de forma global. Dispone de:

- Display o pantalla.
- Teclado de funciones; pulsando en ellas, se accionan las funciones de control programadas.
- Teclado de sistema compuesto por: bloque numérico, teclas de función (Fs y Ks), teclas de cursor, enter ...
- Conexiones de interfaces para la conexión del control (PLC), ordenador, impresora,..



Funciones básicas que se pueden realizar con un OP-7

Por medio de imágenes que previamente hemos programado, podemos:

- Visualizar: estados de servicio, valores actuales del proceso, alarmas de proceso, anomalías del control (PLC), etc.
- Manejar en manual el sistema, meterle datos de proceso, etc.
- Hacer funciones de diagnóstico de la máquina, funciones de borrado del programa…

2 – PROGRAMA ProTool

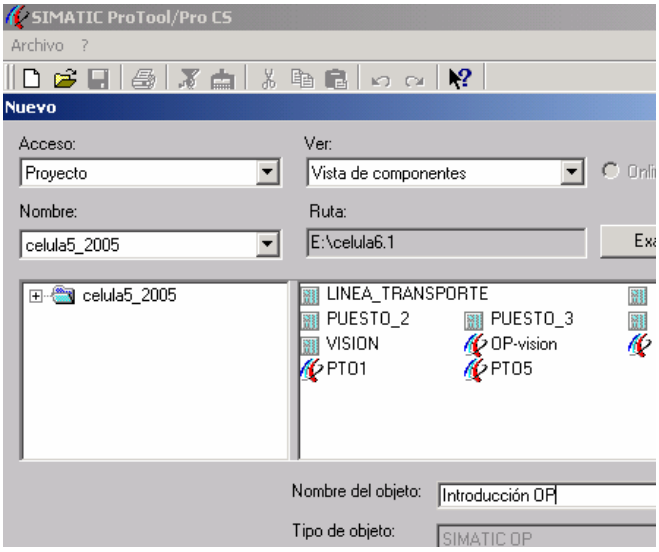
Lo normal es que al programar un OP, tengamos también en el PC el Programa STEP 7, en este caso al instalar ProTool es conveniente instalarlo con la opción "integrado en STEP 7", esto también se puede hacer después al crear el programa del OP, en el menú archivo seleccionando la opción "integración de ProTool en STEP 7", con esto tenemos las siguientes ventajas:

- Podemos administrar los proyectos de ProTool también con el administrador de SIMATIC,
- Se pueden seleccionar como variables los símbolos y los módulos de datos de STEP 7 de una lista de símbolos. El tipo de datos y la dirección son registrados automáticamente.
- Cuando hacemos un nuevo proyecto de ProTool, este presenta una lista de todos los controles en su proyecto STEP 7 y, tras la elección de un control, determina los correspondientes parámetros de direccionamiento.

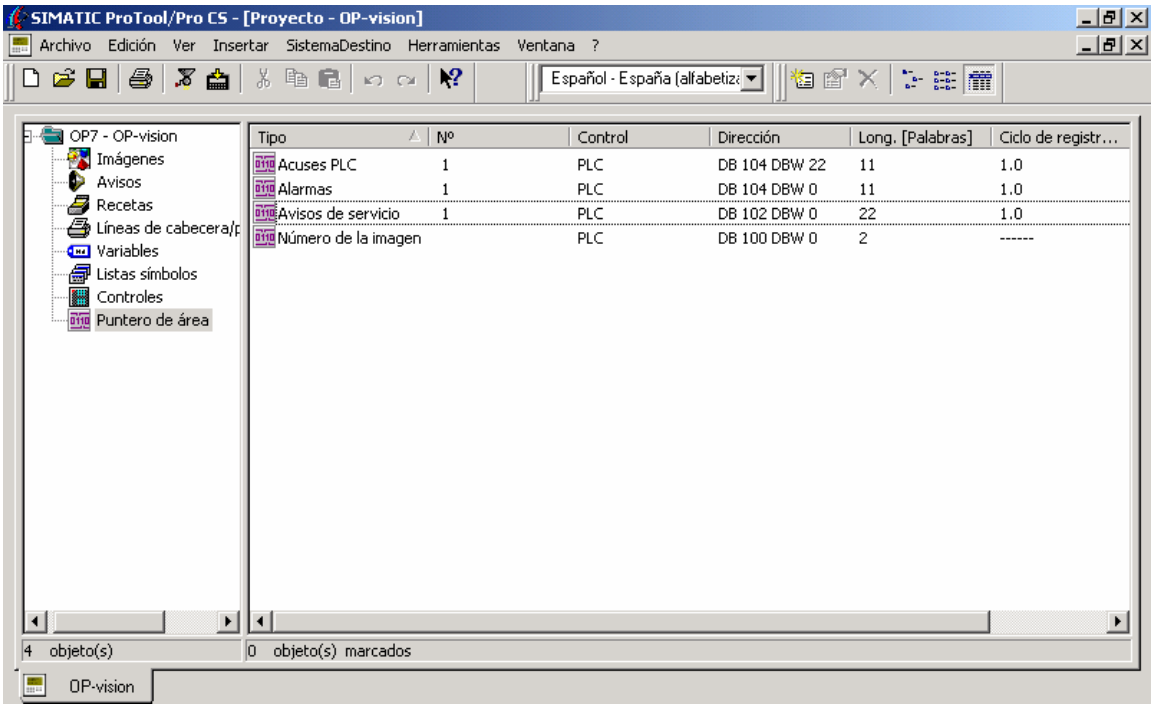
En nuestro caso, tenemos ya el programa STEP 7 y después o a la vez que el proyecto de STEP 7 para el autómata, podemos hacer el programa-proyecto para la OP desde el programa ProTool.
Sin entrar en muchos detalles, vamos a ver a continuación lo más básico de este programa y una aplicación al Puesto 1 de la Célula Flexible. Podemos apoyarnos en las ayudas que tiene, que son análogas a las del Step-7

3 - CREAR EL PROGRAMA DE LA OP.

Se siguen los pasos habituales: *Archivo – nuevo,* y sale la ventana de la figura siguiente. En la casilla nombre, seleccionamos el proyecto de STEP 7 al que va a ir asociado el OP, en la de



nombre del objeto, ponemos el nombre del programa-proyecto del OP que vamos a programar; aceptamos y sale otra ventana, en ella, seleccionamos: display de líneas que es donde están las OPs básicas, elegir la 7 – siguiente – poner nombre al control – seleccionar el protocolo: Simatic S-7 300/400. En parámetros seleccionar el interfase IF1 B, la comunicación MPI o Profibus, la red o PLC al que va a pertenecer este OP y otros que se necesiten – aceptar – siguiente (aquí se puede elegir si se quieren utilizar proyectos estándar con varias cosas ya programadas y programar lo demás que se necesite)– finalizar. Así, ya tenemos un nuevo proyecto ProTool como el de la figura siguiente, de él "cuelgan" los editores de: imágenes, avisos, recetas, líneas de cabecera, variables, listas de símbolos, controles y punteros de área. Todas las funciones de visualización y manejo se programan a partir de esta ventana del programa ProTool:



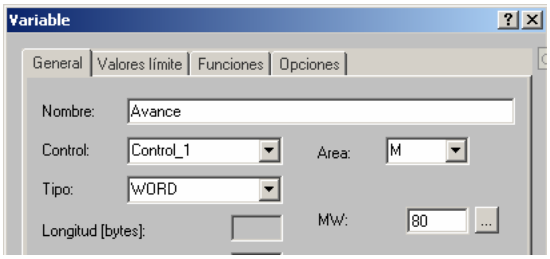
4 – EDITORES DE LOS ELEMENTOS DEL PROYECTO-PROGRAMA DEL OP

Para hacer el programa de la OP se pueden seguir distintos ordenes y volver a programar o reprogramar elementos del programa-proyecto; puede ser una buena opción comenzar programando los punteros y las variables. A continuación se hace una somera descripción de los menús de edición:

Punteros de área

En los punteros de área para: imágenes, avisos de servicio, alarmas…, determinamos en que área de memoria del PLC se van codificar las respectivas imágenes y subimágenes del OP; o sea: establecemos la relación entre las imágenes y/o subimágenes del OP con los valores de una variable en el PLC. Al hacer doble clic (seleccionar y ejecutar) en puntero de área, el programa nos pide que seleccionemos puntero para: numero de imágenes, avisos, alarmas, etc, y después zona de memoria (DB, M), dirección y nº de palabras del PLC que vamos a usar para este cometido. El puntero de imágenes, tiene fija una zona de memoria de 2 palabras en el PLC; después cuando programamos las imágenes, se autonumeran por orden y este numero se guarda en el segundo byte de la primera palabra del puntero, (si queremos se puede cambiar esta numeración). Lo mismo ocurre con las subimágenes de cada imagen, pero esta numeración se guarda en el primer byte de la segunda palabra del puntero.

Variables

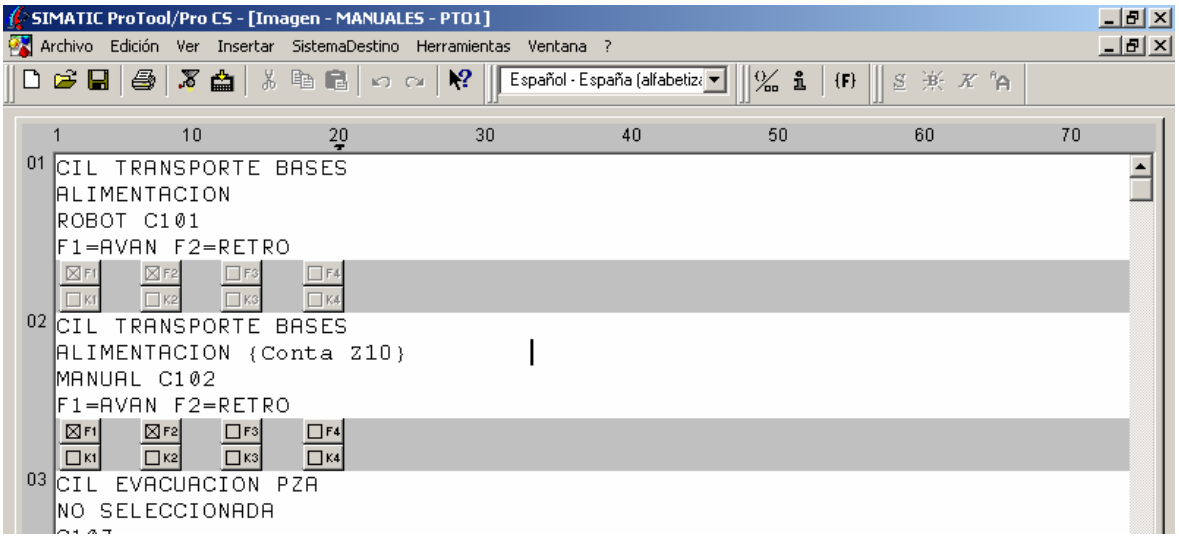


Mediante variables, se establece la conexión entre el control (PLC) y el OP, para ello creamos variables en el OP y al asociarlas a: marcas, contadores, DBs etc. del programa del autómata, podemos hacer que se ejecuten las instrucciones del programa relativas a estas variables. Podemos programar las variables que vamos a emplear para relacionar teclas de función y campos de E/S del OP con acciones del programa del PLC. Al igual que con los otros editores, haciendo doble-click en variables nos sale la ventana de la figura, en la que tenemos las fichas de configuración; la fundamental es la ficha “General”; en ella el programa nos pide fundamentalmente: nombre, control, tipo (Bool, Word, Timer...) área y dirección; aceptamos y ya tenemos creada la variable. En la ficha "Funciones" a las variables de los campos de entrada/salida se les pueden asignar funciones, por ej. las funciones "Selección de imagen", “conversión de variables”, etc; en cuanto se modifique el valor de la variable, se ejecuta la función que le hayamos programado. Las fichas “valores limites” y “opciones”, tienen menos aplicación.

Imágenes:

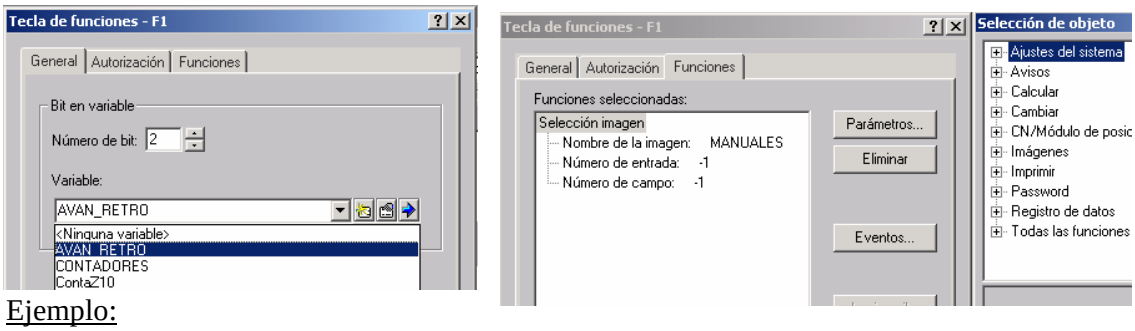
Una imagen es lo que vemos en la pantalla del OP, un programa tiene varias imágenes, además cada imagen que programamos, puede tener varias subimágenes (varias pantallas).
La relación fundamental entre el OP y el autómata, se establece por medio de las imágenes del OP con su puntero y las Funciones y Campos que podemos programar en estas imágenes.
Al hacer doble clic en el editor de “imágenes” se crea una imagen nueva y el programa le asigna un numero de imagen y de subimagen, que se codifica en el puntero de imágenes; después cuando aparezca una subimagen en el display de la OP, se ponen a 1 los bits del puntero correspondiente a esa imagen y subimagen. Una vez programada, (seleccionándola y botón dcho -- propiedades), podemos asignarle: nombre, numero, si es imagen inicial, adonde retorna si en la OP pulsamos escape, si la imagen va metida en el índice, etc. A la primera imagen que queremos que salga en la pantalla del OP, tenemos que asignarla como imagen inicial.
Cada imagen o subimagen de un OP-7 tiene dos zonas; en la parte superior, muestran lo que

hemos escrito en ellas; además de texto, en esta zona podemos programar campos de Entrada/Salida, después en este lugar concreto podremos ver valores del programa-proceso (valor de Salida del contador Z 10 en el caso de la subimagen 02 de la siguiente fig.) o meter valores al programa (Entrada).
En la zona inferior de cada subimagen podemos programar las funciones y dependiendo de lo que les programemos, al accionarlas después desde la pantalla del OP, se ejecutarán unas acciones u otras como: accionamientos en manual, cambios a otras imágenes, puesta a cero de contadores, borrar el programa, etc.



Si estamos en la zona de texto de una imagen y pinchamos en el icono {F} o con: *Insertar* --- *Campo de datos*, aparece una ventana con datos para ese campo, en la que podemos seleccionar: utilización, tipo de campo, representación, variable asociada y lista de símbolos si en representación hemos seleccionado texto (ver las figuras del apartado "Lista de símbolos" de la pag. 8). Si lo asociamos a una de las variables programadas y aceptamos, se crea el campo con el nombre de la variable seleccionada, ejemplo: {vari2}.
Para introducir valores decimales en los campos, pulsamos el/los números y ENTER, al hacer esto, los valores pasan a la variable del PLC. Si el tipo de representación es texto, pulsando Shift + arriba o abajo, se cambia el texto y el bit de la variable asociada a él; se introduce también con ENTER.
En cuanto a las funciones, tenemos dos funciones básicas: A) Las que se asocian a un cambio de imagen; al pulsar su tecla, el OP cambia a la imagen que hemos asignado a la función. B) Las que se asocian a un bit de una variable; al pulsar su tecla se activa este bit del PLC; previamente hemos tenido que crear esta variable.
Para programar o reprogramar una función, se "pincha" esta y el programa nos ofrece una doble ventana; si seleccionamos la ficha "general" podemos asignarle una variable a esta función y dándole las propiedades correspondientes, al pulsar la tecla correspondiente en el OP, el autómata ejecuta la instrucción relativa a la variable. Si seleccionamos una autorización distinta de cero, en la ayuda F1 tenemos la explicación de los códigos y password necesarios para acceder al control de esa pantalla. Si seleccionamos la ficha de funciones, el programa nos ofrece una extensa lista de acciones que se le pueden asignar a la tecla de función, seleccionando cada una de estas, el programa nos ofrece una ayuda debajo sobre que es lo que hace; por ejemplo si seleccionamos: imágenes – selección de imagen – agregar, nos sale una nueva ventana en la que podemos seleccionar de entre las imágenes programadas a cual de ellas (nombre) queremos que salte el OP al pulsar esta tecla de función desde la imagen actual – aceptar. (evento: pulsar tecla)
Las teclas de función se pueden programar para que actúen de forma local o de forma global; de forma local se programan desde el editor de imágenes (estando en una imagen) y realizan la

función programada solo cuando la OP está en la imagen concreta en que han sido programadas, en otra imagen realizarán la nueva función que se les haya programado; de forma global se programan desde el menú: *sistema de destino -- imagen /teclas* y realizan su función al ser accionadas, independientemente de la imagen que tenga la OP; en la figura del apartado 1, (página 2), se ven las ventanas de programación de funciones globales. En general se emplean las teclas F (se les llama teclas soft) para programar funciones locales y las K para programar funciones globales; por ejemplo en el OP-7 todas las Fs y Ks se pueden programar como funciones locales o como globales. En las dos siguientes figuras vemos pantallas de programación de funciones de forma local.



Ejemplo:

Supongamos que hemos programado el puntero de imágenes asociado a la MD 90. Si en el OP seleccionamos la imagen de las activaciones en manual que por ejemplo puede ser la 3, se carga un 3 en MB 91, y si dentro de esta imagen estamos en la subimagen 4, se cargará un 4 en MB 92. Programando comparaciones y otras instrucciones en el FC de manuales, podemos tener una marca de manuales y otra de la subimagen 4 en la que activamos al cilindro 4; si también hemos programado la variable Avan/Retro asociada a la MB 100, y programamos en esta subimagen 4 una F1 asociada al bit 0 (M100.0) y otra F2 asociada al bit 1 (M100.1); estando en esta subimagen, tenemos la marca de manuales y la del cilindro 4, si además pulsamos F1 (se activa M 100.0), podemos hacer que avance el cilindro 4 y si pulsamos F2 que retroceda. En otras imágenes o subimágenes se activan otras salidas de forma análoga, y/o podemos programar cambios a otras imágenes y/o campos de E/S.

Textos de información:

Son informaciones adicionales e indicaciones de manejo que se pueden disponer para cada una de las subimágenes, avisos y alarmas. Como cada subimagen del OP-7 solo puede tener 80 caracteres, si necesitamos más información, se da en un texto adjunto a la subimagen. Seleccionando *texto de ayuda* en el menú "*ver*", podemos escribir lo que deseemos, después estando en la subimagen correspondiente, si pulsamos "HELP" en la OP, nos sale el texto.

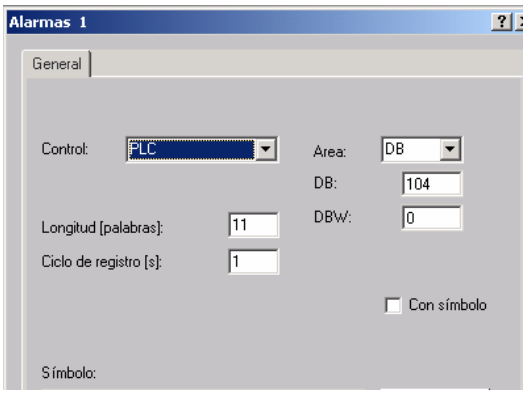
Avisos:

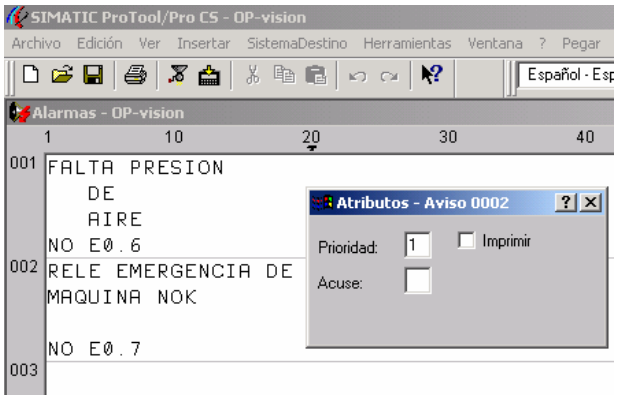
Son subimágenes que aparecen en la pantalla del OP si hay alguna disfunción en el proceso, nos dan información de esa disfunción si previamente lo hemos programado, también pueden ser informaciones e indicaciones de manejo sobre estados actuales de máquinas y procesos. Los avisos, se programan en una sola imagen con las subimágenes necesarias; al programarlos (subimágenes 001, 002, 003,...) el programa ProTool asocia cada uno de ellos a un bit del puntero. Si se proyecta una subimagen de avisos en la zona o subimagen 000, ésta es interpretada como aviso de reposo, no tiene bit asociado y se visualiza en el display, siempre que el OP trabaje en el nivel de avisos y no haya ningún aviso de servicio o alarma. En las subimágenes de avisos no hay funciones, solo hay texto y campos de Salida.

En la ventana de atributos del editor de avisos (ver las figuras siguientes de: puntero de área y avisos) se puede establecer la Prioridad del aviso programado; si coinciden al mismo tiempo varios avisos, se visualizan con arreglo a las prioridades programadas; para el OP-7; 1 es la prioridad mínima y 4 la máxima; si todos tienen la misma se visualizan por orden inverso a lo

programado en el PLC. En esta ventana también se puede activar la opción de imprimir los avisos.
En: *sistema de destino ---Avisos --- Ajustes*, se pueden establecer varios parámetros para los avisos.
Hay dos clases de avisos: **Avisos de servicio y alarmas**. Se programan prácticamente igual pero en distinta imagen y con distinto puntero de área, debido a esto, tienen algunas diferencias:

A) **Avisos de servicio**; accionando la tecla ENTER, pasamos al siguiente aviso; en la ventana de atributos de cada aviso, sale desactivada la casilla de "Acuse"
B) **Alarmas**; a diferencia de los avisos de servicio, las alarmas indican estados críticos de la máquina durante el proceso de producción y deben acusarse con la tecla ACK antes de que sean posibles otras acciones. Si la casilla de "Acuse" la dejamos vacía o ponemos 0, después cuando se activen las alarmas, hay que acusar cada una individualmente por orden de prioridad; si ponemos un numero (del 1 al 4 para el OP-7), al pulsar ACK, se acusan a la vez todas las alarmas que lleven ese mismo numero pues con este numero hacemos un Grupo de acuse.





También hay **avisos del sistema** pero estos no se programan, se realiza su ajuste en: *sistema de destino ---Avisos --- Avisos del sistema.*
El funcionamiento OP-PLC de los avisos es a la inversa que para las imágenes generales; ahora es el PLC el que activa los bits:1, 2, 3, 4, 5, 6, ..., del puntero que a su vez activan los avisos correspondientes en el display del OP; para esto programamos en el Proyecto del PLC un Bloque en el que en función del estado de los elementos (finales de carrera, pulsadores, salidas,..) que queramos que produzcan avisos, se activen los correspondientes bits del puntero.
Ejemplo:
Supongamos que hemos programado el puntero de avisos de servicio asociado a las dos primeras palabras del DB 102 y queremos que el aviso 004 salga en el display del OP cuando no esté accionado el final de carrera a0; programando en el bloque del PLC: (UN a0) (= DB102. DBX 1.3); saldrá ese aviso cuando se cumpla la condición.

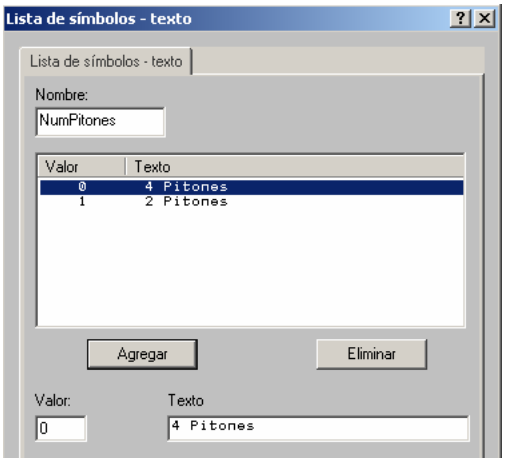
Recetas:
Si por ejemplo se están fabricando distintos productos con una serie de componentes en distintas proporciones, se pueden hacer "recetas" para cada uno de estos productos, los datos de las recetas forman el registro de datos de la receta que se almacena en el OP y cuando haga falta se trasfiere a la unidad de control, así se ahorra espacio de memoria en el Control. La finalidad de las recetas es transferir al control varios datos interrelacionados "juntos" y de forma "síncrona". En la ayuda podemos ver como se configuran las recetas con sus variables, como crear y transferir un registro de datos desde el OP 7, un ejemplo, etc.

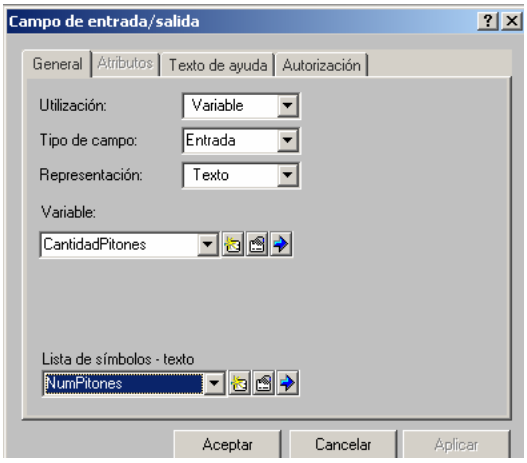
Líneas de cabecera y pies de página
En el OP, se pueden activar opciones para imprimir y podemos editar las líneas de cabecera/pie

de página de lo que se va a imprimir; para esto, se pueden programar "texto estático" y "campos de salida" que sirven para configurar opciones de impresión de avisos, alarmas etc.

Lista de símbolos

En los campos de Entrada podemos meter valores, directamente o por medio de símbolos, para hacerlo de esta segunda forma, además de la variable asociada a este campo de Entrada, también tenemos que programar una lista de textos (símbolos) asociados a un valor determinado de esa variable. Con doble click en "listas símbolos", nos aparece una ventana; damos: nombre, escribimos el texto y agregamos; el programa va asignando valores de bit a cada uno de los textos que pongamos, con aceptar salimos y ya hemos creado una lista de símbolos, que después activaran unos bits para introducir determinados valores en un campo.





Después desde la OP, estando en la subimagen correspondiente, con SHIFT y flechas arriba o abajo y ENTER cambiamos el texto que va asociado a un valor de la variable, y al cambiar este valor, se harán unas acciones u otras en el programa de usuario del PLC .

Controles

Tenemos que seleccionar el PLC o red que controla a este OP, el protocolo y algunos parámetros para conectar el OP a la red; esto se puede hacer ahora, "pinchando" en controles, pero se suele hacer al inicio cuando se crea el nuevo programa, tal como lo hemos visto en el apartado 3.

5 –FORMAS DE TRANSFERIR LOS PROGRAMAS AL OP.

Si el OP no esta metido en una red Profibus hay que conectar un cable (sin la tarjeta MPI) del puerto serie (conector hembra de 9 polos) del PC al conector MPI del OP (Sub-D 15 macho), las conexiones van de la siguiente manera: colorA del pin 3 PC al pin 3 del OP, colorV del pin 2 del PC al pin 4 OP, colorM del pin 5 del PC al pin 12 del OP, además hay que puentear los pines 1-4-6 del conector del PC.

A) Quitar la tensión del OP, pulsar a la vez las teclas de desplazamiento abajo, la de despl. dcha. y la de escape, sin soltarlas meter la tensión. Con lo anterior se borra completamente el OP, ahora conectar el cable serie PC/OP y transferir el programa desde la opción correspondiente del ProTool o con el icono transferir.

B) Si nos hemos creado una imagen con una tecla de función para transferir programas entrar en esta pantalla pulsar la tecla F y activar el icono de transferir en el ProTool. La función se programa con: click en la F elegida – funciones – cambiar – cambio modo servicio – agregar – transferencia serie – aceptar – aceptar; con otras versiones de ProTool, se crea esta pantalla y su F con: F – funciones – tooggle – mode – serial download – agregar.

6 - EJEMPLOS DEL P-1 DE LA CÉLULA

1) Creamos la variable avan/retro y le asignamos la MW 100, después en las imágenes de manuales, a las teclas F1 se le asigna la variable avan/retro, bit 0 (M101.0) y a las F2 la variable avan/retro pero el bit 1 (M101.1). En el FC 15 se han programado comparaciones entre números y valores del DB100.DBW0 para tener la marca de imagen-pantalla de manuales y lo mismo con el DB100.DBW2 para las marcas de subpantallas; con estas marcas y las que se activan al pulsar F1 o F2, tenemos básicamente programado el FC de manuales y su manejo desde el OP. Eligiendo tipo bool, se puedan programar las funciones F1 y F2 directamente asociadas a los bits M 101.0 y M 101.1 respectivamente y no emplear toda una MW100.
   Análogamente en los segm: 15 y 16 del FC1 obtenemos marcas para trabajar con las pantallas de lámparas y contadores
2) Se programa una variable de nombre contador (u otro nombre), tipo Counter con área C y dirección 0 (C0 = Z0); se mete el campo {contador} en el texto de la pantalla de contadores como salida. Se programa también la F1 asociada a la variable avan/retro con bit 0 (M 101.0). Como en el FC1 hay programado un segmento en el que la M101.0 pone a cero al contador Z0, al pulsar la F1, el contador se pone a cero y en el campo {contador} de la pantalla va apareciendo el valor actual de Z0 =C0.
3) Programamos los campos de entrada y salida: {referencia_E} y {referencia_A}; previamente ya teníamos las variables respectivas del mismo nombre asociadas al DB115.DBW2 y DB115.DBW0, después cuando estemos en la pantalla de selección de modelos, parpadea el digito que representa a {referencia_E}, si pulsamos un numero (1 AL, 2 Metacri. 3 Ambos) y confirmamos con ENTER, este nº pasa al DB115.DBW2, y en el FC115 mediante comparaciones activamos marcas que utilizadas después en el modulo de Automático, hacen que el programa trabaje con aluminio, metacrilato o ambos. También en el FC 115 se carga lo del DB115.DBW2 al DB115.DBW0 y esta palabra que es la del campo de salida hace que aparezca en la pantalla el nº de la selección de modelos vigente.
4) Avisos.- Los avisos, se reciben estando el selector en manual. En el FC 105, estando en manual y consultando el estado de: finales de carrera, pulsadores, presostato o pantalla de selección de modelos; y programando los correspondientes saltos SPB y SPA, cargamos un numero determinado en el DB105.DBW0 que es la zona de memoria determinada por el puntero de avisos de servicio (mas fácil que hacer esta serie de saltos y cargas es activar directamente el bit correspondiente con la operación =). Cada numero que se carga y transfiere, produce el aviso correspondiente. Como está hecha la estructura de saltos, solo se puede producir un aviso a la vez. Pero una vez que pulsamos ENTER y hemos eliminado la causa de ese aviso, puede aparecer otro de los avisos siguientes.
5) En el Puesto 5, tenemos campos con texto; por ejemplo en una de las pantallas tenemos los campos de E y de S, {CantidadPitones}; primero hemos creado la variable CantidadPitones asociada al DB101.DBX20.0 (ver FC1 segm 18 y FC10 segms 9 y 17) y en la lista de símbolos, el símbolo NumPitones; en este símbolo le hemos puesto 4Pitones al bit 0 y 2Pitones al bit 1. Después hemos cumplimentado la ventana que sale al seleccionar este campo en su imagen. Finalmente con Shift + ⁂ o Shift + y después ENTER, cambiamos entre el bit 0 o el 1.

29 - INSTRUCCIONES BÁSICAS DE UN AUTÓMATA OMRON, COMPARADAS CON STEP-7 DE SIEMENS

Entradas y salidas

Para las entradas y salidas, OMRON, utiliza normalmente tarjetas de 16 bits (W) que denomina "canales"; utiliza tres cifras para denominar el canal y otras dos para el nº de bit. Los números bajos son entradas (000, 001, ...) y los (010, 011, ...) salidas. En el autómata modular CQM1 se suelen asignar los canales 100, 101,.... a las salidas.
Son entradas: 000.00, 000.01, ... 000.15; 001.00, 001.01.....
Son salidas: 010.00, 010.01, ...010.15; 011.00, 011.01,... ; 100.00, 100.01,...100.15; ...
Para los ejemplos que vamos a ver aquí, el canal 000 es de entradas y el 001 de salidas.

Marcas

Las marcas, también van por canales; hay que saber que canales (IR) concretos usa cada tipo de autómata; son canales habituales los: 200, 201.... Además, Omron utiliza canales específicos como remanentes (HR) otros como especiales (SR); por ejemplo son relés o marcas especiales el 253.02 que da un impulso cada seg. o los 255.05, 255.06 y 255.07, que se ponen a "1" según que una comparación sea: mayor, igual o menor respectivamente; otro de estos contactos esta abierto siempre, otro cerrado siempre.....

Temporizadores y contadores

Fundamentalmente hay dos tipos de Temporizadores: retardo a la conexión de 100 en 100 ms (TIM) y de 10 en 10 ms (TIMH). Los contadores son uno descendente (CNT) y otro ascendente/descendente (CNTR). Los temporizadores y contadores se numeran con tres cifras a partir del 000, pero esta numeración es para ambos y un temporizador y un contador no pueden tener el mismo número.

Bloques de datos

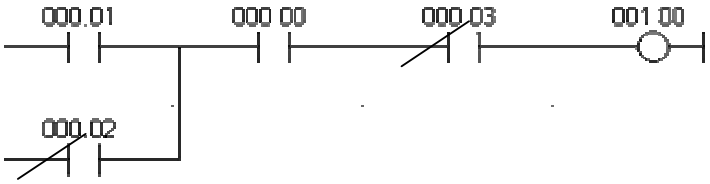
Como bloque de datos emplea palabras DM, (DM0000, DM0001, ...)

Instrucciones básicas con ejemplos.

Se emplea el programa SysWin o el CX-Programmer y los programadores, habitualmente programan mas en KOP que en AWL, para entender mejor los ejemplos vamos a ver las dos formas. En AWL la primera instrucción de una cadena lógica de instrucciones comienza por LD (Load); también se emplea el LD para comenzar un paréntesis y AND LD para cerrarlo y que esté en serie con lo anterior, cerrándolo con OR LD estaría en paralelo con lo anterior. Las instrucciones provienen del Ingles, son: AND, OR, NOT, OUT, MOVE, TIM (Timer), CNT (Counter) CMP (Compare), JMP (Jump, salto).... En Step-7 podemos elegir nemotecnia de programación inglesa y las instrucciones se parecerían a estas.

CONTACTOS SERIE-PARALELO

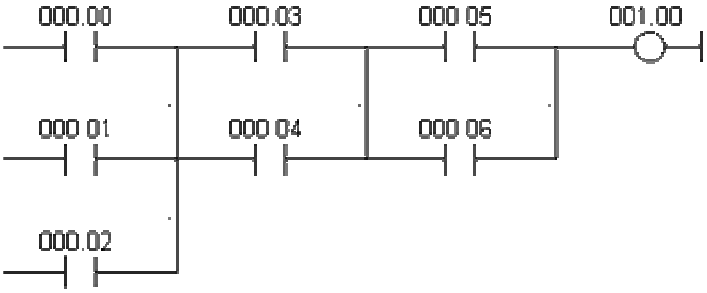
```
LD          000.01
OR NOT      000.02
AND         000.00
AND  NOT    000.03
OUT         001.00
```



CONTACTOS Y PARÉNTESIS (LD)

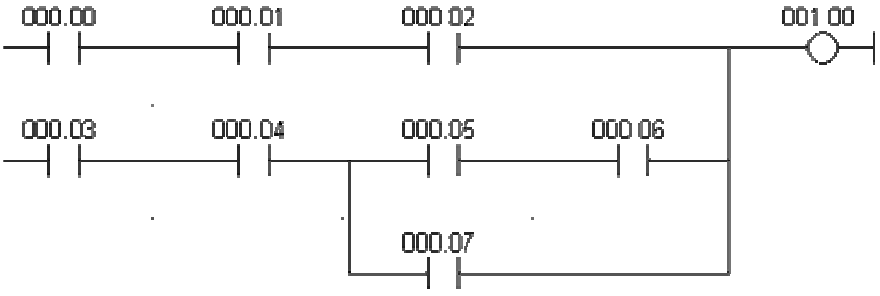
AND LOAD (AND LD)

```
LD        000.00
OR        000.01
OR        000.02
LD        000.03
OR        000.04
AND LD
LD        000.05
OR        000.06
AND LD
OUT       001.00
```



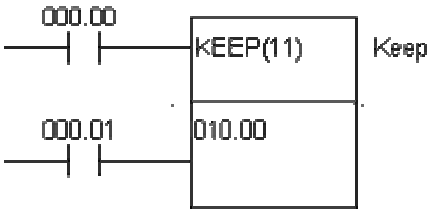
OR LOAD (OR LD)

```
LD        000.00
AND       000.01
AND       000.02
LD        000.03
AND       000.04
LD        000.05
AND       000.06
OR        000.07
AND LD
OR LD
OUT       001.00
```



RELÉ DE ENCLAVAMIENTO S-R.- Las instrucciones Set y Reset van por separado, solo con el relé especial KEEP pueden ir juntas.

```
LD      000.00
LD      000.01
KEEP 010.00
```

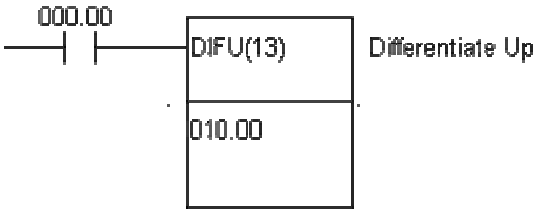


FLANCOS.

FLANCO ASCENDENTE (DIFU)/ FLANCO DESCENDENTE (DIFD)

Activan una salida durante un ciclo de SCAN. DIFU se activara cuando la entrada 0 se active.

```
LD      000.00
DIFU    010.00
```



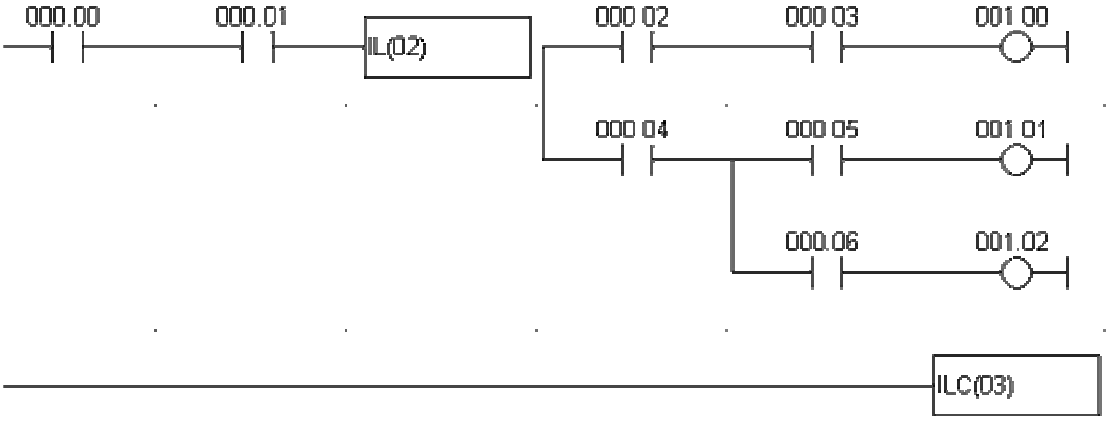
Para la desconexión (DIF Down) seria:

```
LD      000.00
DIFD    010.00(P)
```

SALTOS.- Hay dos formas básicas de hacerlos, IL/ ILC y JMP/JME.

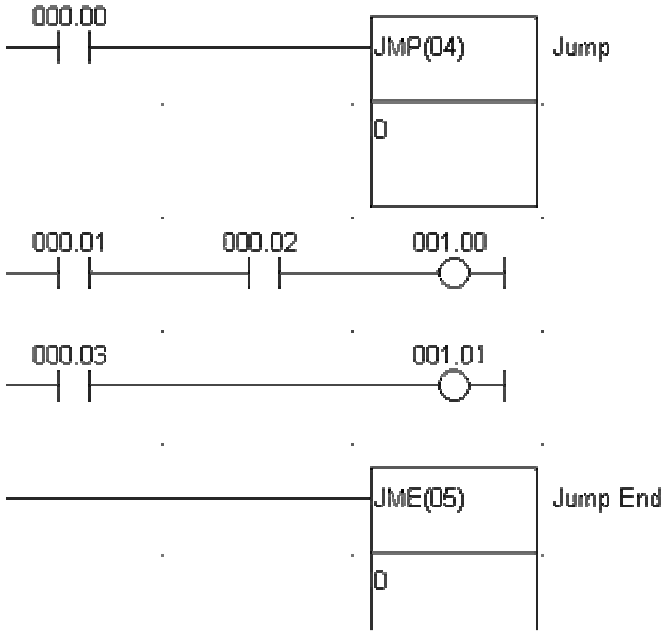
ENCLAVAMIENTO (IL)/ BORRA ENCLAVAMIENTO (ILC).IL y ILC van siempre juntos.

```
LD      000.00
AND     000.01      Si el resultado es OFF todas las salidas hasta ILC están desactivadas.
IL                  Si el resultado es ON las salidas toman el valor que corresponde.
LD      000.02
AND     000.03
OUT     001.00
LD      000.04
AND     000.05
OUT     001.01
LD      000.04
AND     000.06
OUT     001.02
ILC
```



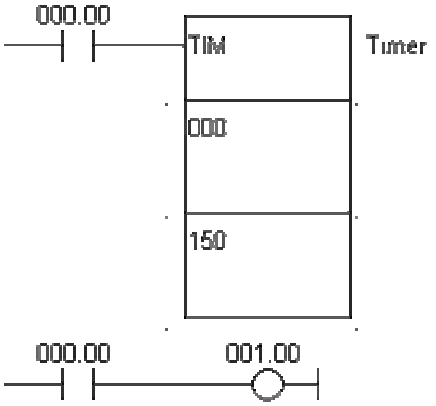
SALTO (JMP)/ FIN DE SALTO (JME)
Van siempre juntos. Hacen que las instrucciones entre JMP y JME se ignoren o se ejecuten normalmente.
Las instrucciones entre JMP y JME se ejecutan si la instrucción anterior a JMP es 1(R=1)(no salta) si R=0 no se ejecutan (salta).
La diferencia entre IL y JMP es que con JMP los valores OUT se mantienen si no se ejecuta.

```
LD      000.00
JMP     0(N)
LD      000.01
AND     000.02
OUT     001.00
LD      000.03
OUT     001.01
JME     0
```



TEMPORIZADORES.- El temporizador se inicia al activarse la entrada 000.00. Cuando se cumpla el tiempo se activa la salida 001.00 y si desactivamos la entrada, se ponen el temporizador y la salida a 0. El temporizador TIMH es análogo pero con base de tiempo de 10 ms. El tiempo va con #tiempo pero también podemos meter el valor que tenga una palabra DM

```
LD      000.00
TIM     000(n)   #0150(SV)
LD      TIM000
OUT     001.00
```

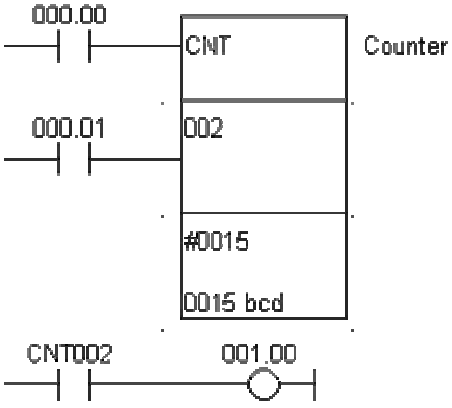


Se pueden encadenar timers para temporizaciones mas largas.

CONTADORES.- Hay dos tipos; el descendente (CNT) que pone su salida a uno cuando el valor de contaje llega a cero y el ascendente / descendente (CNTR) que pone su salida a 1 cuando llega a 0 descendiendo o cuando llega a su valor de contaje ascendiendo. Este último cuenta de forma continua, o sea descendiendo pasa de 0 al valor de contaje y sigue descendiendo y ascendiendo pasa del valor de contaje a 0 y sigue ascendiendo. La entrada de Reset pone el contador a su valor de contaje (en algunos casos a 0)y ambos son remanentes.

CONTADOR (CNT)
Es un contador descendente. La única diferencia es que en el contador se mantiene el valor si hay un fallo de alimentación y en el timer no.

```
LD      000.00
LD      000.01
CNT     002(n)   #0015(SV)
```



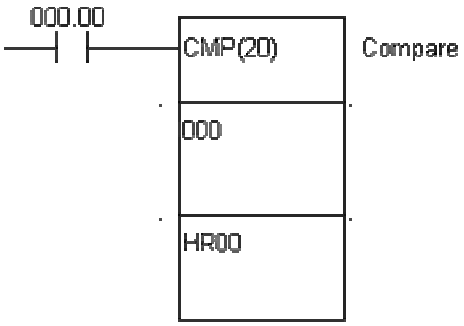
CONTADOR REVERSIBLE (CNTR)

```
LD        000.00(II)
LD        000.01(DI)
LD        000.02(Rt)
CNTR      000(n)  #9999(SV)
LD CNT    000
OUT       001.00
```

IL- Pulsos de cuenta ascendente.
DI- Pulsos de cuenta descendente.

COMPARACIONES.- En el S-7, se hacen automáticamente, con Omron hay que activar la comparación Hay varias comparaciones, en la mas usual (CMP) se comparan dos valores (canales) y en el caso del autómata CQM1, el resultado de la comparación (>, =, <) pone a "1" respectivamente los bits especiales 255.05, 255.06 ó 255.07

```
LD        000.00
CMP       000(C1)  HR00(C2)
```



TRANSFERENCIAS.- La mas usual y directa es MOVE. Mueve el dato de un canal (16 bits), a otro. Copia el contenido de un canal fuente (S) en un destino (D). Si delante de la instrucción MOVE o de otras instrucciones como las de calculo se pone la @, esta transferencia solo se hace en el primer ciclo de scan.

```
LD        000.01
MOV       011(S)   HR01(D)
```

NOTA: Lo anterior de OMRON, es solo una pequeña introducción, hay manuales y diversos medios para profundizar algo mas; por ejemplo en las paginas:
http://www.grupo-maser.com/PAG_Cursos/Cursos.htm ,
http://olmo.pntic.mec.es/~jmarti50/enlaces/automatas.html , http://gpds.uv.es/plc/ .....