STEP 7

MANUAL BASICO DE PROGRAMACION

Exclusión de responsabilidad.

Se ha probado el contenido de este libro de programación, y la concordancia en el hardware descrito y el software. Sin embargo, es posible que se den algunas desviaciones con respecto a las diferentes posibilidades de los temas descritos, lo cual impide tomar garantía completa de los temas expuestos. Es necesario comprobar la concordancia del contenido de los temas con las aplicaciones propias a realizar, no aceptándose ninguna responsabilidad de ningún tipo por no adoptar dicha medida de precaución..

Autor: Juan Carlos Rubio Calín.
Fecha: 2001.
Versión: 1.0
Impreso en Barcelona, España.

Nociones básicas de S7

Prólogo

Este manual contiene las explicaciones base para el manejo y programación de un autómata S7 de Siemens. No es necesario que el usuario tenga una experiencia previa en la programación de los equipos Simatic S7 para entender el contenido de este manual, sólo se requerirá un conocimiento general de programación de PLC´s de manera genérica.

El objeto del presente manual también está enfocado a aquellos que aun disponiendo de nociones de programación, no se hayan adentrado minímamente en la programación de los equipos S7, debido a que no existiera un solo manual básico de referencia que permitiese adquirir en un periodo razonable de tiempo los conceptos al respecto de este sistema, dado que dichos conocimientos se encontraban repartidos en multitud de manuales de cursos de aprendizaje Simatic S7 nivel I y nivel II.

El manual está dividido en 3 partes: Nociones básicas de S7 (1ª parte), Conjunto de Operaciones (2ª parte) y Programación avanzada (3ª parte).Esta 1ª parte Nociones básicas de S7 está dividida en capítulos en que cada uno describe una función específica de trabajo así como las instrucciones necesarias para llevarla a cabo. Los capítulos se complementan entre sí y por tanto deberán leerse siguiendo el orden en que aparezcan en el manual, aunque si el usuario es experimentado y desea solamente conocer las instrucciones de programación de S7 en similitud con S5 u otros lenguajes de programación de autómatas, sólo será necesario leer la 2ª parte de este manual.

Las explicaciones de este manual se describen en los dos métodos más usuales de programación (KOP y AWL) y la CPU 314-IFM de Simatic para llevar a cabo cualquier proyecto habitual y se omiten aquellos aspectos que se consideran triviales, centrándose exclusivamente en los puntos conflictivos de los diferentes apartados tratados.

Espero que esta obra sea de utilidad a los diferentes usuarios y programadores que como yo, disfrutan viendo un sistema de automatización, pero siempre piensan que puede existir una forma mejor y más sencilla de hacerlo.

Juan Carlos Rubio Calín.
Barcelona 2001.

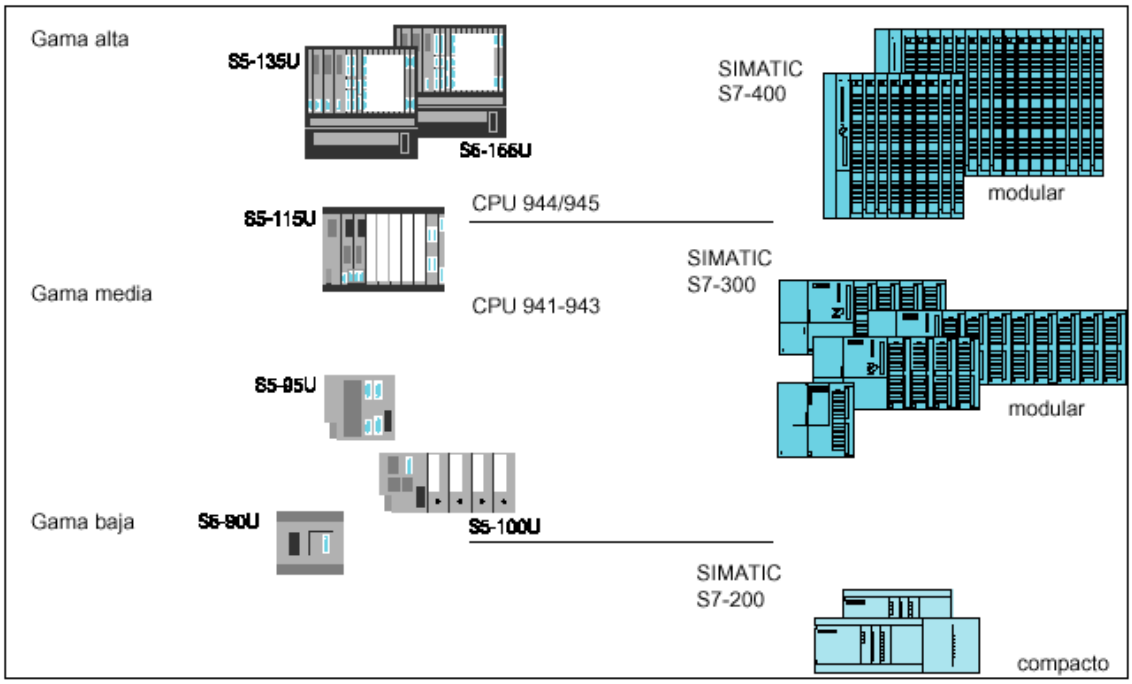
1
INTRODUCCIÓN A STEP7

1.1 STEP 7 ¿Qué es?. Familia S7. Comparativa entre S5 y S7

STEP 7 es el lenguaje de programación para la familia de autómatas SIMATIC S7-300/400 que se ejecuta en Windows, lo que permite una mayor facilidad de diseñar un programa de usuario para un autómata programable encargado de ofrecer una total solución a una tarea de automatización.

La familia S7 se compone de tres gamas de sistemas de automatización que se distinguen por sus prestaciones y su comparativa con la familia S5 serian:

- Gama baja: SIMATIC S7-200 (microPLC compacto con software propio) equivale a S5-90U
- Gama media: SIMATIC S7-300 (miniautómata modular) equivale a S5-95U
- Gama alta: SIMATIC S7-400 equivale a S5-115U S5-135U...

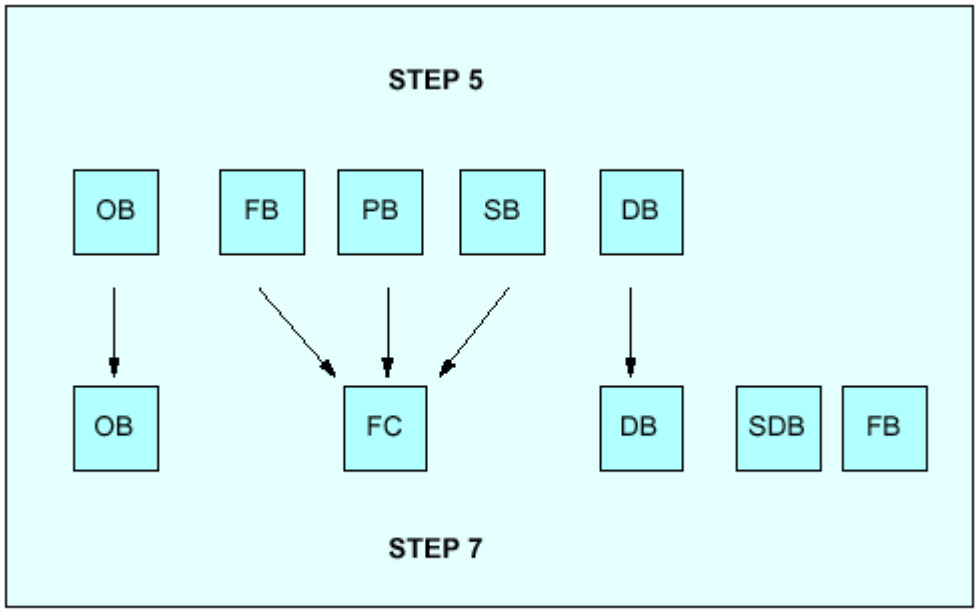


Sistemas de automatización SIMATIC

Para familiarizarse con el software STEP 7 adelantamos una visión general de las primeras diferencias entre STEP 5 y STEP 7 entre los siguientes campos:
a)Módulos: Lo que en S5 eran los módulos de programa, de datos, de funciones en S7 son bloques. Aunque no debería interpretarse como una asignación estricta 1 a 1 podríamos establecer la siguiente comparativa:

En S5	En S7
-Módulos de organización OB1	-Bloque de organización OB1
-Módulos OBs especiales (SFB)	-Bloques de función de sistema
	-Funciones de sistema (SFC)
-Módulos de programa PB	-Función (FC)
-Módulo de función FB	-Función (FC)
-Módulos de datos DB	-Bloques de datos (Globales DB o de instancia DI)
-Módulo de datos especiales DB1	-Bloques de datos de sistema (SDB)

Aunque los estudiaremos mas adelante no confundir en S7 los bloques de función FB con las funciones FC, y destacar que los bloques de función FB tienen unas características completamente distintas a los módulos de función FB de STEP 5.



Bloques con funciones comparables en STEP 5 y STEP 7

b)Tipos de datos y constantes:

STEP 7 utiliza nuevos formatos para los datos y constantes. Por ejemplo los datos locales que son datos que se asignan a un bloque lógico y que se declaran en el área de declaración de variables del mismo, pueden ser datos estáticos, temporales o de parámetros de bloques.
También utiliza nuevos formatos de constantes, como por ejemplo para valores de temporización (S5TIME)

c)Operaciones:

STEP 7 cambia el nombre de algunas de las variables de operaciones como sumas (+F por +I), comparación (>=F por >=I), abrir bloques (A por AUF), llamadas (SPA por CALL), conversión (KEW,KZW por INVI,NEGI)... operaciones de direccionamiento indirecto por memoria en vez de operaciones de procesamiento (B), así como ofrece nuevas operaciones, como por ejemplo:
-operaciones lógicas con bits X,XN,FP,FN,NOT
-operaciones de aritmética MOD
-operaciones de conversión: BTI,ITB
-operaciones de salto: SPBN
-operaciones master control relay: MCRA,MCRD
-direccionamiento completo de operandos de datos: L DB100.DW6

d)Direccionamiento:
El direccionamiemto absoluto en S7 es idéntico que en S5 salvo que los datos de los bloques de datos se direccionan byte a byte.

En S5	En S7
-DW 0,1,2,3,.....255	-DW 0,2,4,6,.....510
-DL 0,1,2,3,.....255	-DBB 0,2,4,6,.....510
-DR 0,1,2,3.....255	-DBB 1,3,5,7,.....511

El direccionamiento simbólico en S7 puede ser con símbolos globales o locales.

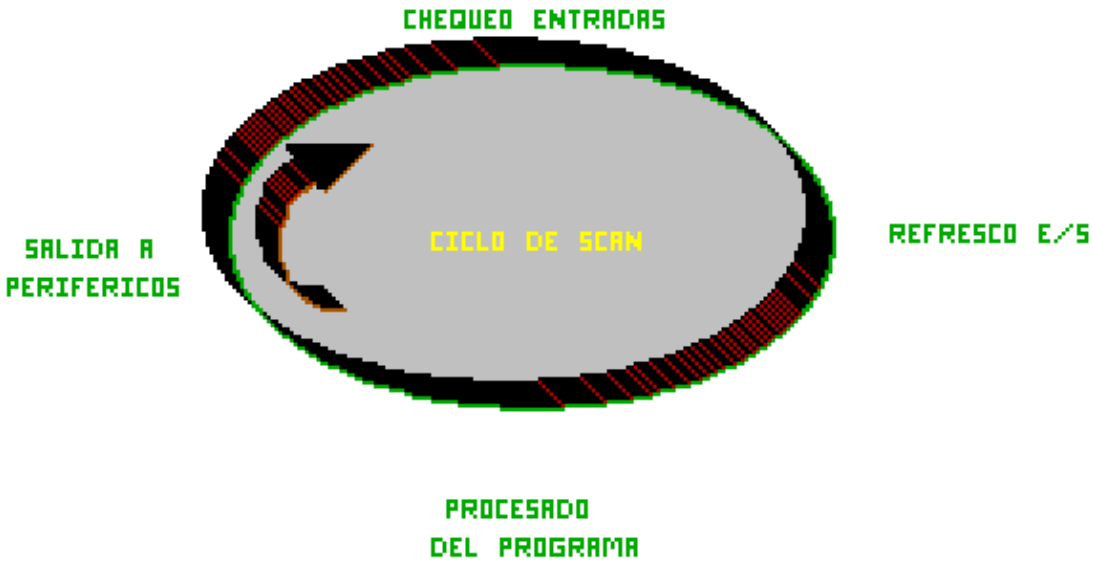
1.2 Principio de funcionamiento de un PLC.PAE-PPA.Ciclo de Scan

El autómata programable industrial (API o PLC) es un sistema de control, basado en el empleo de un procesador, que opera mediante la adquisición de información de las variables de entrada al sistema (datos), elabora una serie de instrucciones de programa de control una tras otra, para presentar los resultados (acciones) sobre las variables de salidas del sistema
Las variables de entrada al sistema que informan al autómata de señales externas al proceso (sensores, fotocelulas, pulsadores...) se denominan CAPTADORES.
Los elementos (reles, contactores, visualizadores...) que actúan según las salidas del autómata se denominan ACTUADORES.
Por tanto el autómata está formado por dos elementos básicos: La CPU (unidad central de proceso) y Sistema E/S.
La CPU consta de la memoria y el procesador donde corren principalmente dos programas: El sistema operativo y el programa de ususario.

El sistema operativo está integrado en la CPU y sus funciones son:

- Gestionar el arranque
- Actualizar la imagen de proceso de las entradas y emitir la imagen de proceso de las salidas
- Llamar al programa de usuario
- Detectar las alarmas y errores
- Administrar las áreas de memoria
- Comunicar con unidades de programación y otras estaciones de comunicación.

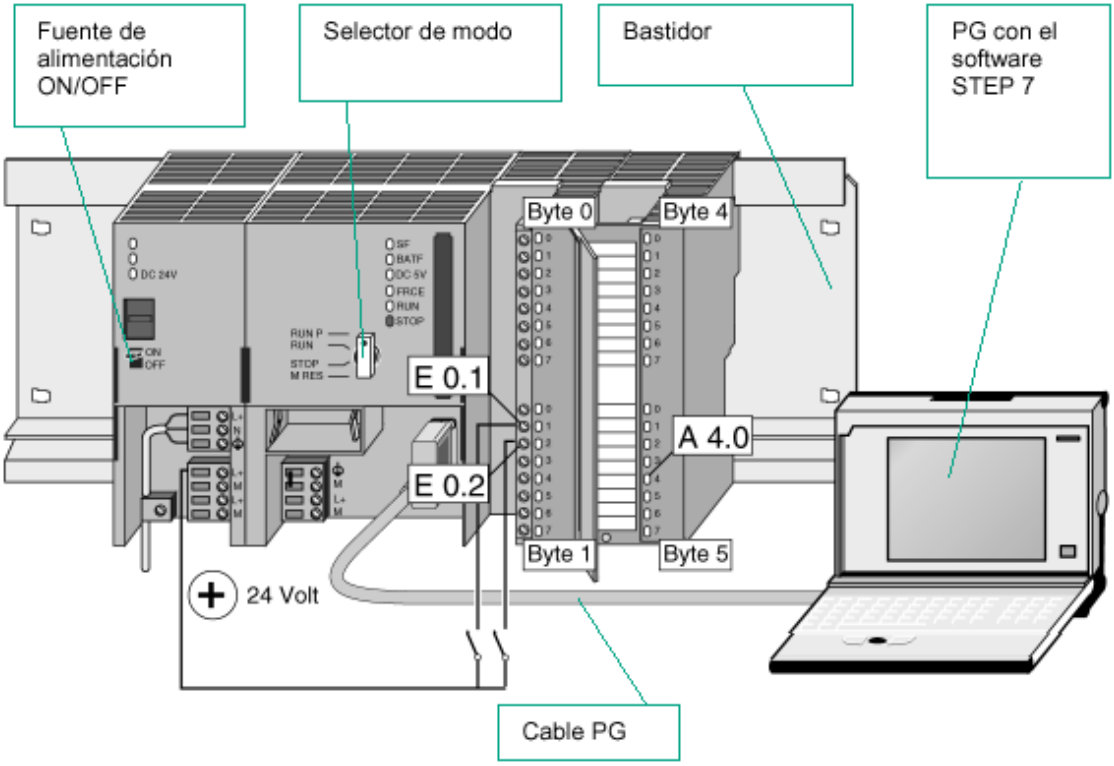
El funcionamiento del PLC es cíclico, al inicio (arranque) del programa los estados de señal de los módulos de entrada se transmiten a un área de memoria de la CPU conocida como imagen del proceso de las entradas y al final de cada ejecución cíclica del programa, los estados de señal de la imagen de proceso de las salidas se transfieren a los módulos de salida.
El proceso "Lectura de entradas-imagen de entrada del proceso-ejecución del programa-imagen de salida del proceso-salida de señales" se repite continuamente y se conoce como CICLO DE SCAN. El tiempo necesario para realizar un ciclo completo de programa se denomina SCANTIME.
En comparación con el acceso directo a los módulos E/S, el acceso a la imagen de proceso ofrece la ventaja de que la CPU dispone de una imagen consistente de las señales del proceso durante la ejecución cíclica del programa. Si durante la ejecución del programa varía un estado de señal en un módulo de entrada, dicho estado de señal se conserva en la imagen de proceso hasta que esta sea actualizada al inicio del próximo ciclo.



1.3 Características técnicas del autómata S7 CPU-314-IFM

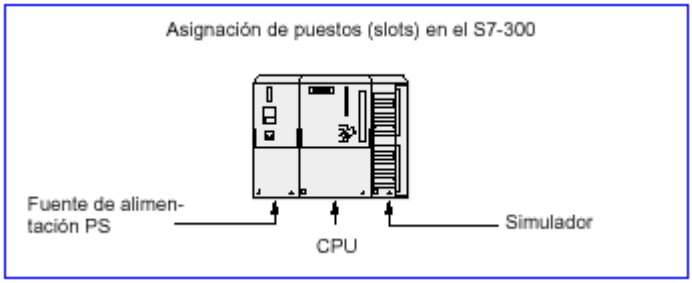
Los requisitos de hardware para el autómata S7 consta de:

- 1 Perfil soporte: Bastidor S7-314-IFM
- 1 Fuente de alimentación: PS transforma la tensión de red (220VAC) a la tensión de alimentación operativa 24VDC
- 1 CPU: ejecuta el programa de usuario.Alimenta con 5V el bus posterior del S7-314. Se comunica a través del cable MPI con otras CPU o PC
- Módulos E/S
- Cable MPI
- Ordenador personal con software STEP 7 bajo Windows



Reglas de colocación:

- La PS (fuente de alimentación) deberá colocarse siempre como primer módulo en la parte izquierda del perfil soporte.
- La CPU (módulo central) se colocará siempre como segundo módulo a la derecha de la fuente de alimentación
- En total es posible colocar a la derecha de la CPU un máximo de 31 módulos de señales (S7-300). En total suma E/S digitales 512, suma E/S analógicas 64



1.4 La memoria en S7

La memoria de las CPUs S7 se subdivide en tres áreas diferenciadas:

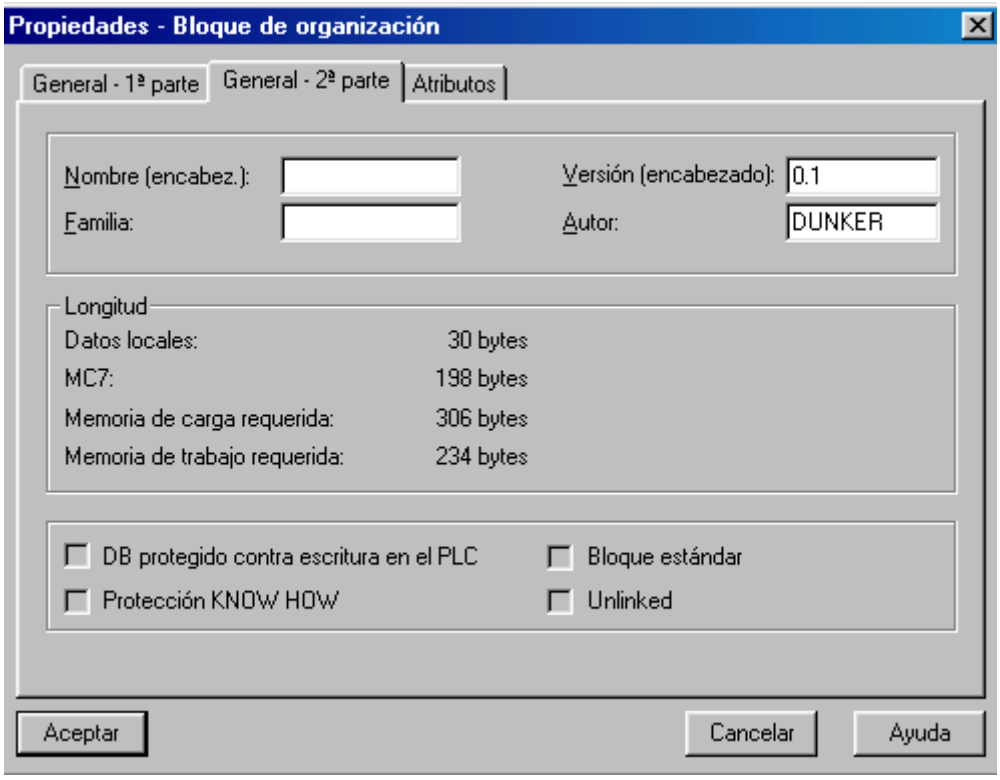
- **La memoria de carga**: permite almacenar el programa de usuario sin asignación simbólica de operandos o comentarios (éstos se almacenan en el disco duro del ordenador). La memoria de carga puede ser RAM o Flash-EPROM. En la memoria de carga se almacena no solo el programa sino también los datos del sistema.
- **La memoria de trabajo (RAM integrada)**: contiene la partes del programa S7 relevantes para la ejecución del programa. La ejecución del programa tiene lugar exclusivamente en el área correspondiente a las memorias de trabajo y del sistema.
- **La memoria del sistema (RAM)**: contiene los elementos de memoria que cada CPU pone a disposición del programa de usuario, tales como: la imagen de proceso de las entradas y salidas, marcas, temporizadores y contadores. Contiene además las pilas de bloques y de interrupción. La memoria del sistema de la CPU ofrece además una memoria temporal (pila de datos locales) asignada al programa para los datos locales del bloque llamado. Estos datos sólo tienen vigencia mientras esté activo el bloque correspondiente.

Por lo tanto, nuestro programa tendrá un consumo de memoria de carga y otro de memoria de trabajo. En ninguno deberemos de superar la memoria de trabajo, ya que no es posible su ampliación, por lo que nos veremos obligados a cambiar de CPU. La memoria de carga sí que puede ser ampliada mediante Flash o Ram externas.

1 instrucción de un programa equivale en promedio a 3 bytes

CPU	MEMORIA CARGA	MEMORIA TRABAJO
314 IFM	40 KB	24 KB

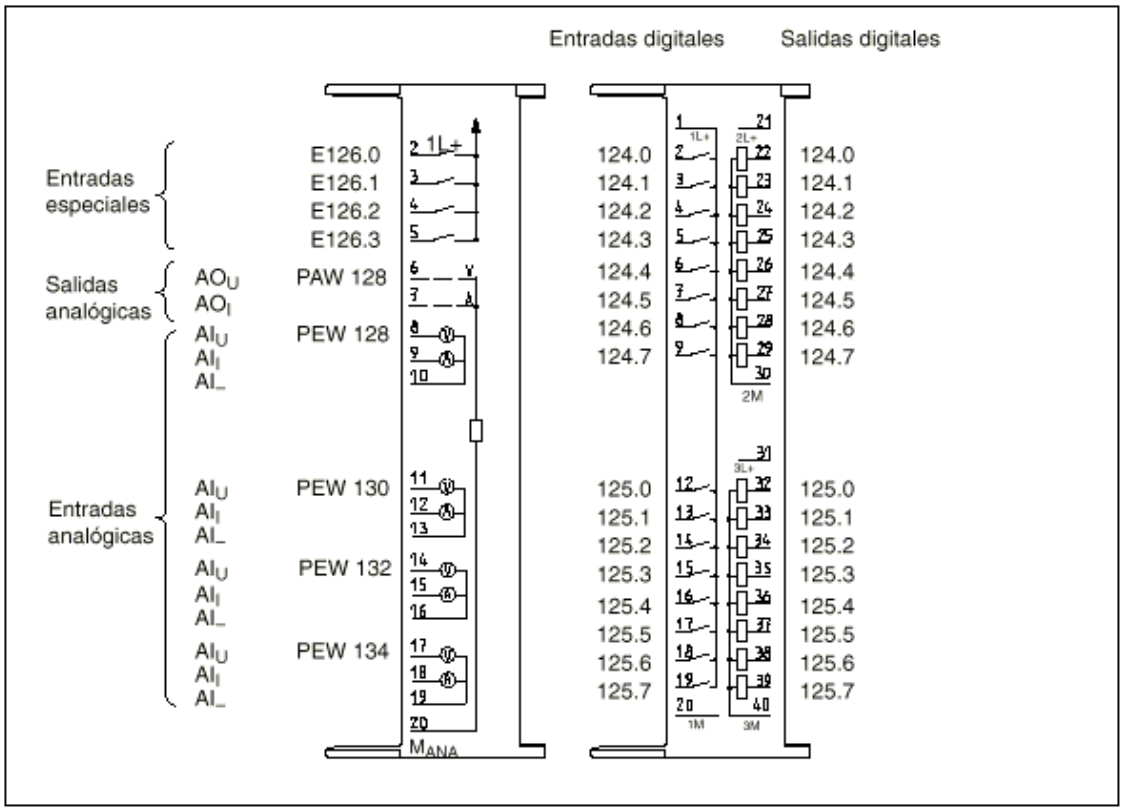
Si en el administrador de Step 7 seleccionamos un bloque de S7, presionando el botón derecho podremos seleccionar sus propiedades, que nos indicarán tanto el tamaño de la memoria de carga requiera, como el tamaño de la memoria de trabajo.



Para conocer cuanto nos ocupa todo el programa, incluyendo los datos de sistema, deberemos de seleccionar el subdirectorio bloques, y visualizar sus propiedades.

1.5 Direccionamiento de señales.Areas de operando y Tipos direccionamiento

La CPUs 314-IFM dispone de 20 entradas digitales, 16 salidas digitales, 4 entradas analógicas, 1 salida analógica integradas cuyo direccionamiento es:



CPU 314 IFM

ENTRADAS/SALIDAS	ÁREA
20 ED	E 124.0 a E 126.3
16 SD	E 124.0 a E 125.7
4 EA	EW 128 a EW 134
1 SA	AW 128

Area de operandos:
Además de las entradas (E) y salidas (S) disponemos de otros operandos que junto con el uso de las operaciones correspondientes forman el programa:
M-marcas 0.0 a 255.7 (Total 2048)
T-temporizadores 0 al 127 (10ms a 9990s)
Z-contadores 0 al 63 (1 a 999)
D-datos (total 127)
Obs (total 13)
Fc´s (128)
Fb´s (28)
Sfc´s (48)
Sfb´s (14)

Tipos de direccionamiento:

El direccionamiento indica una dirección donde la operación encuentra una variable con la que ejecuta una operación.
Las direcciones de estos operandos las podemos representar por su unidad de tamaño. Puede ser un bit,byte,palabra o doble palabra:
1 relé para1 Bit Ej: E 127.3
8 reles para 1 Byte Ej: AB100 (A100.7.....A100.0)
16 reles para 1 Word Ej: MW200 (MB200+MB201)
32 reles para 1 Dword Ej: MD80 (MW80+MW81+MW82+MW83)
El direccionamiento puede ser : Inmediato, absoluto o directo, simbólico,como completo de operando de datos o indirecto:
Si declaramos una constante como operando se denomina direccionamiento inmediato Ej.: L 50
Si declaramos una variable como operando, a este tipo se llama Direccionamiento absoluto o directo, y consta:

Identificador de operando	Tipo de dato	Nº del 1er Byte y en caso nº Bit
Ej: M (marca)	B (byte)	100
Ej: M		100.1

O solo área de memoria
Ej: T1 , Z5

El operando se compone de dos partes:

- un identificador (p. ej. "EB" para "byte de entrada")
- una dirección exacta dentro del área de memoria indicada por el identificador.

El operando indica directamente la dirección del valor.

Ejemplo	Descripción
U E 0.0	Ejecutar una operación lógica Y con el bit de entrada E 0.0.
S L 20.0	Activar el bit de datos locales L20.0.
= M 115.4	Asignar el RLO al bit de marcas M115.4.
L EB 0	Cargar el byte de entrada EB0 en el ACU 1.
L MW 64	Cargar la palabra de marcas MW64 en el ACU 1.
T DBD 12	Transferir el contenido del ACU 1 a la palabra doble de datos DBD12.

También si se asigna un símbolo al operando se llama Direccionamiento Simbólico
Ej: "Pulsador 1"

Por direccionamiento completo de operandos de datos se entiende la indicación conjunta del bloque de datos y del operando. Esto no era posible en S5.

L DB100.DBW6

Si el operando, en forma de puntero, indica la dirección del valor que deberá procesar la operación, se conoce como direccionamiento indirecto
Ej: L EB[MD2]
T AB[MD2]

1.6 Puesta en marcha del PLC.

El PLC dispone para su puesta en marcha de :

- Interruptor I/O para aplicar tensión
- Selector 4 posiciones con llave de modo de operación
- Leds de la CPU

En RUN-P puedo programar y forzar variables
En RUN se ejecuta el programa pero no puedo programar
En STOP se para la CPU
En MRES borra la CPU

En RUN y STOP podemos sacar la llave del selector de posiciones

Veamos cual es el significado de los leds de las CPU´s S7.

LED	INDICA	SIGNIFICADO
SF (ROJO)	Fallo o error de sistema.	Luce en caso de: Fallo hardware. Fallo del firmware del equipo. Instrucción de programa incorrecta. Asignación de parámetros de sistema erróneos. Errores aritméticos internos. Errores de tiempo. Flash-Eprom externa errónea. Fallo de la batería. Fallo de acceso a la periferia (no para la periferia integrada en la CPU).
BATF (ROJO)	Fallo de batería.	Luce si la batería no es capaz de salvaguardar el programa en caso de desconexión de la alimentación del equipo.
5 VDC (green)	Alimentación BUS	La CPU está alimentando al bus trasero a 5 V correctamente.
FRCE (AMARILLO)	Forzado activado.	Luce si se está forzando una señal.
RUN (VERDE)	Estado RUN.	El programa se está ejecutando.
STOP (ROJO)	Estado Stop.	El programa no se ejecuta.

En el caso de que la CPU disponga de tarjeta maestra de Profibús DP, el significado de los leds de la tarjeta son:

SF	DP	Significado	Solución
LED off	**LED off**	Configuración O.K.	
LED on	**LED on**	Fallo de bus de comunicaciones a nivel de hardware. Diferente configuración de velocidades de bus en modo multimaestro.	Chequee el cable y las resistencias terminadoras.
LED on	**Parpadea**	Fallo de estación. Cada vez que parpadea es que intenta acceder a un esclavo y este no responde.	Acudir a información del módulo->buffer de diagnóstico para saber que esclavo no responde.
LED on	**LED off**	Incorrecta configuración de los datos DP	Evaluar el buffer de diagnóstico.

2

Lenguajes de programación

2.1 Formatos de representación.

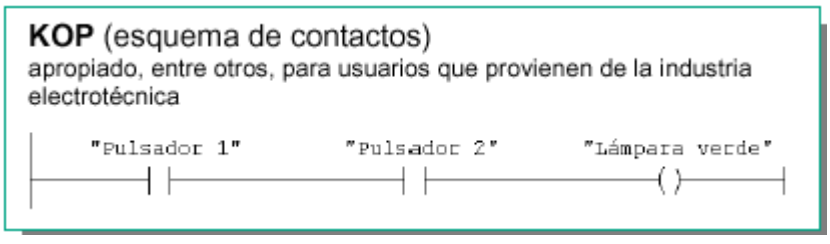
Para programar un autómata de la forma más simple posible basta con crear un programa de usuario y cargarlo en la CPU. Step 7 dispone de varios lenguajes de programación: KOP/FUP/AWL

2.1.1 Esquema de contactos KOP

KOP abreviatura de Kontaktplan (esquema de contactos) es el lenguaje grafico de Step 7 similar al de los esquemas de circuitos eléctricos.
Los elementos tales como contactos NO,NC, bobinas se agrupan en segmentos y este lenguaje permite un fácil seguimiento de la circulación de la corriente.
Las instrucciones KOP se componen de elementos y cuadros que se conectan gráficamente en segmentos.Tipos de operaciones en KOP:

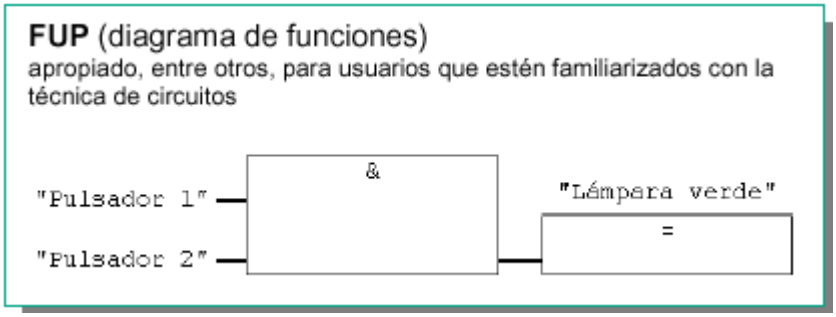
- Operaciones como elemento
- Operaciones como elemento con operando
- Operaciones como elemento con operando y valor
- Operaciones como cuadro con parámetros

En un programa en KOP la circulación de la corriente cuando pasa por diferentes entradas, salidas u otros elementos y cuadros siguen las reglas de la lógica booleana.
Cada operación lógica consulta si el estado de la señal de un contacto eléctrico es 0 (desactivado/off) o 1 (activado/on) y suministra a continuación un resultado. El resultado lógico se denomina RLO.



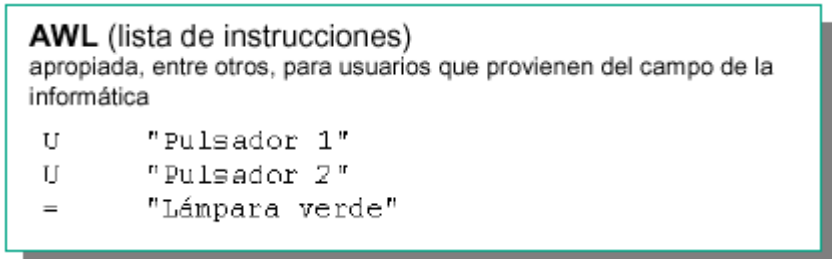
2.1.2 Diagrama de funciones FUP

Otro lenguaje gráfico que utiliza los símbolos del álgebra booleana para representar la lógica, como se haría con un esquema de conexiones en la técnica digital.



2.1.3 Lista de instrucciones AWL

Es un lenguaje de programación textual, muy próximo al lenguaje máquina, las instrucciones y las operaciones van seguidos de los operandos lo cual permite obtener programas ahorrando espacio de memoria y tiempo de ejecución.



2.2 Elementos de un programa de usuario.Tipos de bloques lógicos

El programa de usuario contiene todas las funciones requeridas para procesar la tarea especifica de automatización. Un programa de usuario S7 consta de bloques lógicos: OB´s,FB´s,FC´s,DB´s,VAT

Si desea crear programas en KOP,AWL,FUP seleccione el objeto Bloques del programa S7 y hacer click menú insertar para elegir bloques lógicos.

- Bloques de Organización OB´s:

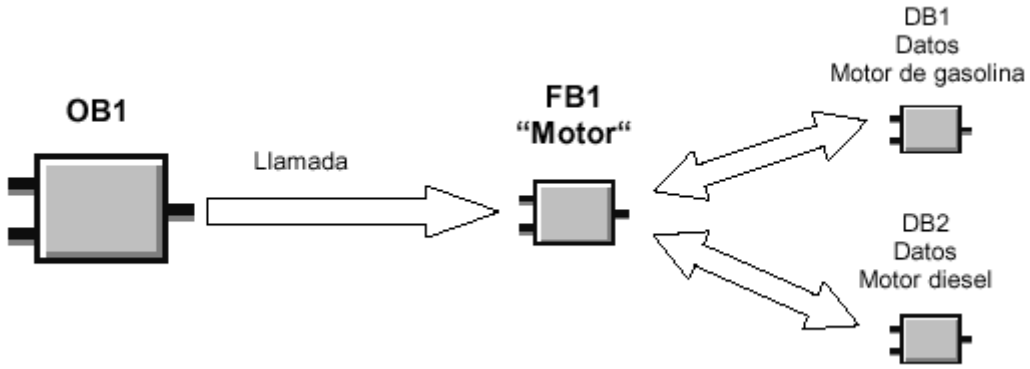
Constituye el interface entre el sistema operativo y el programa de usuario. En los AG se utiliza el tratamiento cíclico del programa.Por consiguiente el programa de usuario se trata cíclicamente en el OB1, que es la subrutina principal en la que se definirá la secuencia con la que se ejecutarán los bloques del programa de usuario.

- Funciones FC´s:

Son bloques lógicos programables sin memoria, que pueden transferir parámetros. Por ejemplo para devolver un valor de función a un bloque invocante. (en funciones matemáticas)

- Bloques de funciones FB´s:

Son bloques lógicos programables con memoria (asociados a un DB instancia).Simplifican la programación de funciones complejas de uso frecuente. Por ejemplo un FB para un tipo de motor, puede controlar por ejemplo diferentes motores utilizando datos de instancia diferentes para los diferentes motores.Los datos para cada motor (Ej: nº revoluciones) se pueden memorizar en unos o varios DB´s de instancia.

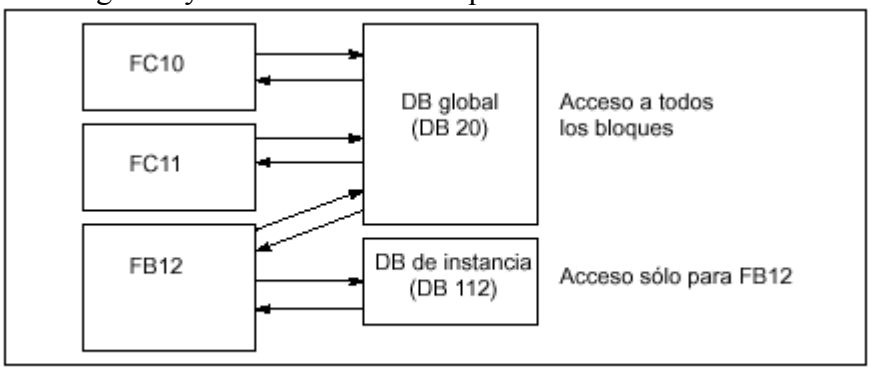


- Bloques de Datos DB´s:

No contienen instrucciones Step 7, sirven para depositar datos de usuario, es decir, contiene datos variables con los cuales puede trabajar el programa de usuario. 2 tipos

a)Globales: Pueden acceder todos los bloques lógicos

Un DB global y un DB de instancia pueden estar abiertos al mismo tiempo

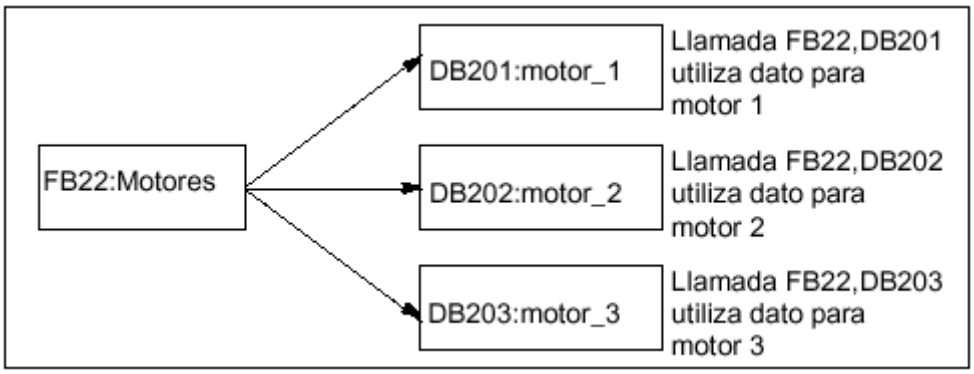


Accesos a DB globales y DB de instancia

b)Instancia: A cada llamada de un bloque de función que transfiere parámetros esta asignado un bloque de datos de instancia. en el DB de instancia están depositados los parámetros actuales y los datos estáticos del FB.

Antes de crear un bloque de datos de instancia debe existir el FB asociado. el número de dicho FB se debe indicar al crear el bloque de datos de instancia.

Por ejemplo si se asignan varios bloques de datos de instancia a un bloque de función FB que controla un motor, se puede utilizar este FB para controlar varios motores:



- VAT´s: Ver Tabla de variables

Una vez abierto el bloque (vacío) podrá introducir el programa en KOP,AWL o FUP.

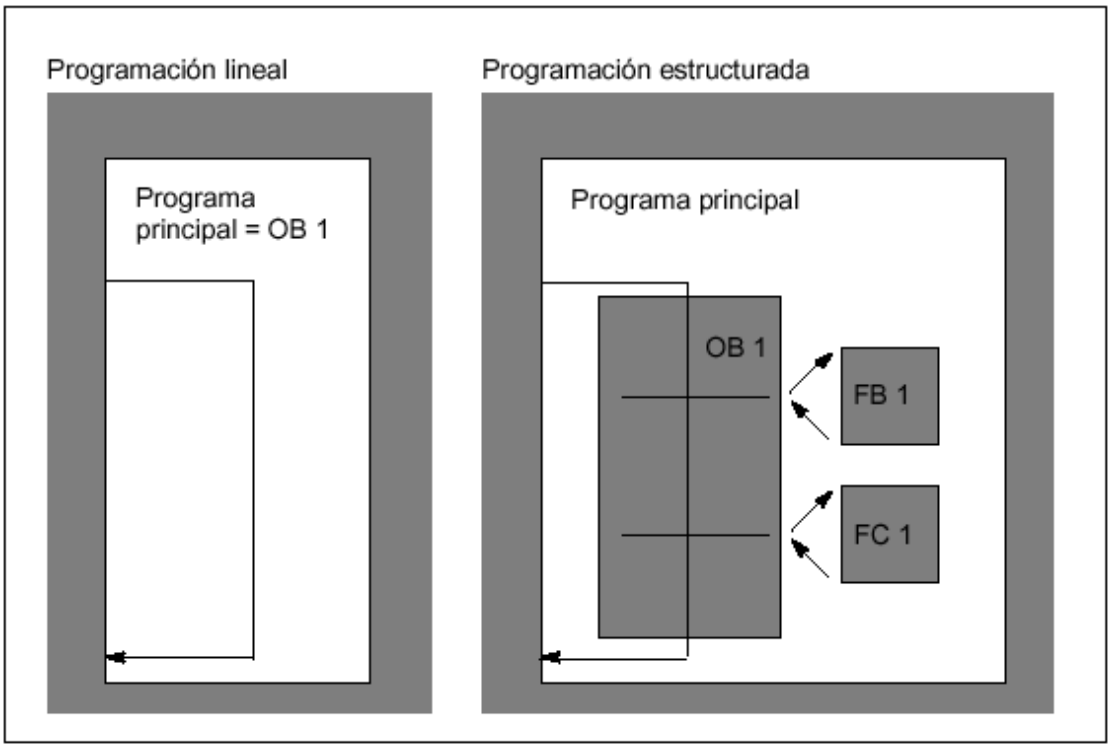
2.3 Estructuración de un programa: Programación lineal y Estructurada

El programa de usuario completo en el OB1 define una programación lineal (Para programas simples)

Para funciones complejas de automatización, Step 7 ofrece la posibilidad de estructurar el programa de usuario en tareas parciales más pequeñas, correspondientes a las funciones tecnológicas del proceso de automatización (Programación estructurada). Estas tareas parciales equivalentes a las partes del programa se definen como los bloques lógicos del programa

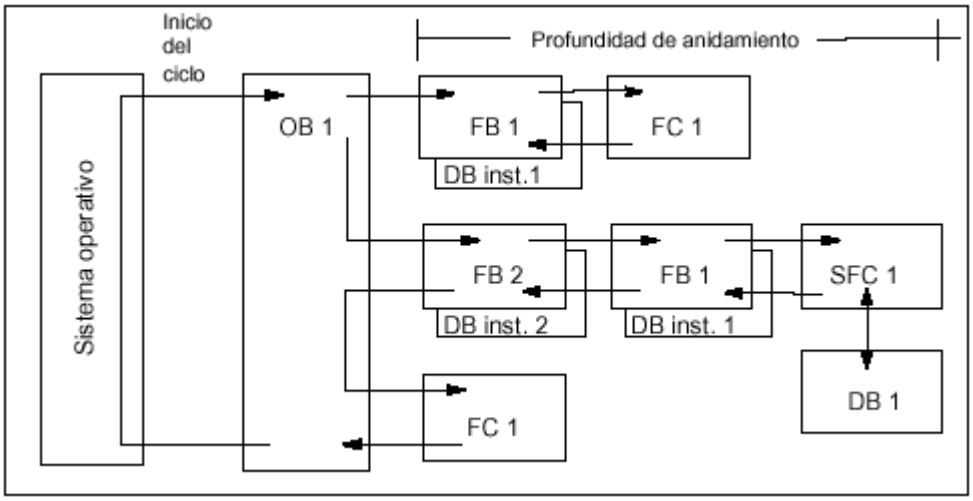
Ventajas de la programación estructurada:

- Programas voluminosos se pueden programar de forma clara
- Se pueden estandarizar partes de programas
- Se simplifica la organización
- Las modificaciones pueden efectuarse fácilmente
- Se simplifica las pruebas de los programas.



Orden de creación de los bloques:

- Los bloques se crean de arriba hacia abajo ,es decir, que se comienza con la fila superior de bloque
- Cada bloque que se llame ya deberá existir. Por tanto en una fila de bloques, el orden de creación deberá ser de derecha a izquierda
- El OB1 es el último bloque que se crea



Conforme a dichas reglas, el orden de creación de los bloques de la figura de ejemplo sería el siguiente:

FC 1 > FB 1 + DB 1 de instancia > DB 1 > SFC 1 > FB 2 + DB 2 de instancia > OB 1

La programación estructurada se utiliza cuando un programa es complejo y es necesario, por dar un carácter más panorámico o por comodidad, dividirlo en partes en donde cada una de estas partes se encargará de una tarea determinada dentro del conjunto del programa. Para ello será necesario la programación de tantos módulos como partes existan, fijándose la secuencia de funcionamiento de cada módulo programado en un módulo de organización OB1.
Desde cada módulo programado es posible ir a otro módulo, el cual una vez procesado se vuelve automáticamente al lugar de origen pudiéndose de esta forma realizar conexionados de hasta 16 módulos.
Resumiendo, se puede decir que la programación estructurada consiste en dividir la tarea de control en varias subtareas y realizar el programa de cada una de ellas independientemente. Cada uno de estos programas se realizará en bloques lógicos (FC,FB,DB...). Finalmente en el bloque de organización OB1 se establece el orden y las condiciones de procesamiento de cada uno de los bloques lógicos programados.
Este método de programación tiene las siguientes ventajas:

- Cada parte del proceso o cada máquina se puede programar independientemente, lo cual facilita la visión global del conjunto de máquinas o procesos a controlar
- Se pueden probar las distintas partes del proceso independientemente, lo cual facilita la labor de puesta en marcha y la localización de averías
- Potencia y rapidez en la ejecución del programa

2.4 Sistemas Numéricos. Tipos de Datos en S7

2.4.1 Sistemas de Numeración y Códigos de Representación

Las operaciones que han de realizarse con números en sistemas digitales (dos estados 0 y 1) pueden representarse en diversos sistemas de numeración, que se diferencian por su base

Para representar un número en general en cualquier sistema, sigue la expresión:

$N= dnB^n$ +..... $d3B^3$ + $d2B^2$ + $d1B^1$ + $d0B^0$

d= Es el dígito correspondiente a esa posición

n= Peso o exponente que indica el lugar que ocupa el digito

B= La Base de un sistema de numeración es el número de símbolos distintos utilizados para la representación de las cantidades del mismo.

B^n= Factor de posición es la base elevada a una cifra igual a la posición del digito dentro de la expresión del número, comenzando por la derecha que corresponde a la posición cero.

Cada digito tiene distinto valor en función de la posición que tenga dentro de la expresión del número. Dependiendo de la posición en que se encuentre el número tendrá un valor u otro.

El valor de la expresión es igual a la suma de los valores de todos los dígitos multiplicados por el factor de posición.

Se pueden formar sistemas de numeración con cualquier base, base 3, base 9, base 115,etc.Sin embargo solo tiene utilidad definirlo además del base 10 (sistema decimal) en base 2 (sistema binario) y en los derivados base 8 (sistema octal) y base 16 (sistema hexadecimal),nótese que 8 y 16 son potencia de dos.

- **Sistema de Numeración Decimal:**

En este sistema los números se forman por diez símbolos (las cifras comprendidas entre 0 y 9) y potencias de diez (Base decimal).

Por ejemplo el $1456 = 1*10^3 +4*10^2+5*10^1+6*10^0$

$347 = 3*10^2+4*10^1+7*10^0$

- **Sistema de Numeración Binario:**

Este sistema de numeración está constituido de forma semejante al sistema de numeración decimal pero emplea solamente dos cifras (0 y 1) y en potencia de dos (Base 2)

Por ejemplo los números binarios siguientes equivaldrían en decimal:

$11001 =1*2^4 + 1*2^3 +0*2^2+0*2^1+1*2^0 =16+8+1=25$

$10010 =1*2^4 + 0*2^3 +0*2^2+1*2^1+0*2^0 =16+2=18$

- **Sistema de Numeración Octal:**

Definamos un número octal con sus ocho símbolos 0,1,2,3,4,5,6,7 (Base –1) y obtengamos por el método ya conocido su valor en decimal:

6701 en base ocho$=6*8^3 +4*8^2+0*8^1+1*8^0 =3521$ en base diez

144 en base ocho$=1*8^2+4*8^1+1*8^0 =100$ en base diez

- **Sistema de Numeración Hexadecimal:**

El sistema de numeración hexadecimal es un sistema de numeración con base 16, que emplea 16 símbolos para la representación, los números 0 al 9 y las letras de la A a la F, ya que si empleáramos 10 en lugar de A al escribir el número no podríamos determinar si se trata de una cifra (10) o de dos cifras consecutivas (el 1 y el 0).

9A en base hexadecimal$= 9*16^1+10*16^0 =154$ en base diez

2F3A en base hexadecimal$= 2*16^3 +15*16^2+3*16^1+10*16^0 =12090$ en base diez

Cuando se trata de valores binarios grandes, con sólo las cifras 0 y 1, su escritura es muy engorrosa, para utilizar menos símbolos empleamos los signos del sistema de representación hexadecimal

- **Conversión entre sistemas de numeración:**

Del mismo modo que nos interesa pasar un número del sistema binario a decimal podremos encontrarnos en la circunstancia de querer transportar un número del sistema decimal a binario, hexadecimal u octal. Para ello realizamos divisiones sucesivas del número por la base quedándonos con los restos y con el último cociente cuya división daría cero. Veámoslo con varios ejemplos:

Conversión Decimal a binario:
123:2 => Cociente 61 Resto 1
61:2 => Cociente 30 Resto 1
30:2 => Cociente 15 Resto 0
15:2 => Cociente 7 Resto 1
7:2 => Cociente 3 Resto 1
3:2 => Cociente 1 Resto 1

Vemos que hemos alcanzado un cociente menor que el divisor (2) por tanto paramos y tomamos el último cociente y los restos en orden inverso de tal forma que el número 123 en base decimal quedaría expresado como 1111011 en base dos

Conversión Decimal a Hexadecimal:
154:16 => Cociente 9 Resto 10
El número 154 decimal es el 9ª en hexadecimal

Conversión Decimal a Octal:
25:8 => Cociente 3 Resto 1
El número 25 decimal es el 31 en Octal

La siguiente Tabla muestra la correspondencia de las 15 primeras cifras del sistema decimal con la de los sistemas binarios,hexadecimal y octal:

DECIMAL	BINARIO	OCTAL	HEXADECIMAL
0	0000	0	0
1	0001	1	1
2	0010	2	2
3	0011	3	3
4	0100	4	4
5	0101	5	5
6	0110	6	6
7	0011	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Vemos claramente como la conversión entre los sistemas octal y binario es inmediata, tan solo hay que sustituir ternas de dígitos binarios por el digito equivalente en octal (y viceversa).
De igual forma cada grupo de cuatro bits de un número binario (tétrada) , se puede representar por una cifra del sistema hexadecimal (y viceversa)
Ejemplo: 1 1011 en binario es 1 B en hexadecimal
A7E en hexadecimal es 1010 0111 1110 en binario

- **Códigos de Representación:**

Los códigos de representación establecen una ley de correspondencia entre las informaciones por representar y las configuraciones binarias.
Por ejemplo para establecer una comunicación entre el hombre (o equipos periféricos) y el ordenador, tenemos el código ASCII que asocia una secuencia de 0 y 1 a unos determinados caracteres (a, b,c..@,#.?...). O También si queremos visualizar en un display de 7 segmentos un número decimal utilizaremos el código BCD.

a) Código Decimal Codificado Binario (BCD)

Este código asocia a cada digito decimal su codificación binaria mediante una secuencia de 4 dígitos binarios.
Por ejemplo 235 en BCD basta con cambiar cada digito decimal por el conjunto de bits en binario que corresponda:

2	3	5
10	0011	0101

b) Código standard americano para intercambio de información ASCII (American Standard Code for Interchange of Information):

Para representar la información que necesitamos para la transmisión de datos entre un ordenador y sus periféricos (teclados, impresoras...)se ha desarrollado el código ASCII básico que compuesto por 7 bits representan los 128 caracteres fundamentales de números, letras y símbolos gráficos.
Por ejemplo la secuencia 0100 0001 se le ha asociado el carácter “A” con independencia del número que representa dicha secuencia de ceros y unos, que traducidos al decimal corresponden al 65.
Ej: la secuencia 0011 0011 seria el número decimal 3
Ej: 2B en hexadecimal corresponde al carácter +
Con 8 bits tenemos el código ASCII extendido para PC que representa 256 caracteres, como todos los caracteres del alfabeto en mayúsculas y minúsculas, los signos ortográficos y de puntuación, los símbolos correspondientes a las operaciones aritméticas (+,-,*,/), los números del 0 al 9 y además una serie de caracteres especiales que permiten realizar gráficos simples y recuadros en pantalla, así como otros que se reservan para las instrucciones de control utilizadas en las comunicaciones entre el ordenador y el exterior (Ej. Tecla ESC).

2.4.2 Formato y representación numérica de datos en S7

Las operaciones AWL, FUP o KOP utilizan objetos de datos de un tamaño determinado. Como su nombre lo indica, las operaciones lógicas de combinación de bits utilizan bits. Las operaciones de carga y transferencia (AWL), así como las operaciones de transferencia (FUP y KOP) utilizan bytes, palabras y palabras dobles.

Un bit es una cifra binaria ("0" o "1"). Un byte comprende 8 bits, en tanto que una palabra se compone de 16 bits y una palabra doble, de 32 bits.

Las operaciones aritméticas utilizan también bytes, palabras o palabras dobles. En estos operandos de bytes, palabras o palabras dobles se pueden codificar números de diversos formatos tales como enteros y números en coma flotante.

Los diversos tipos de datos tienen diferentes opciones de formato y representaciones numéricas.

Formato	Tamaño en bits	Representación numérica
Hexadecimal	8, 16 y 32	B#16#, W#16# y DW#16#
Binario	8, 16 y 32	2#
Fecha IEC	16	D#
Tiempo IEC	32	T#
Hora	32	TOD#
BYTE	8	'A'

Todas las variables utilizadas en el programa de usuario en Simatic S7 se deben identificar con un tipo de datos. Se distinguen entre:

- Tipos de datos simples
- Tipos de datos compuestos
- Tipos para definir los parámetros a transferir a los FBs o las FCs

Tipos de datos en S7	Tipos de datos
BOOL, BYTE, WORD, DWORD, INT, DINT, REAL, S5TIME, TIME, DATE; TIME_OF_DAY, CHAR	Tipos de datos simples
DATE_AND_TIME, STRING, ARRAY, STRUCT	Tipos de datos compuestos
TIMER, COUNTER, BLOCK_FC, BLOCK_FB, BLOCK_DB, BLOCK_SDB, POINTER, ANY	Parámetros

Veamos los tipos de datos de que podemos disponer, así como su rango de valores:

- Variables simples: aquellas que solo están formadas por una única variable.

Tipo	Bits	Fomatos	Rango	Ejemplo
BOOL	1	texto	TRUE/FALSE	TRUE
BYTE	8	hexadecimal	B#16#0 a B#16#FF	L B#16#23
WORD	16	binario	2#0 a 2#1111111111111111	L 2#00101
		hexadecimal	W#16#0 a W#16#FFFF	L W#16#234F
		Bcd	C#0 a C#999	L C#997
		Decimal sin signo	B#(0,0) a B#(255,255)	L B#(14,245)
INT	16	Decimal con signo	-32768 hasta 32767	L 345
DWORD	32	binario	2#0 a 2#11111111111111111111111111111111	L 2#11011

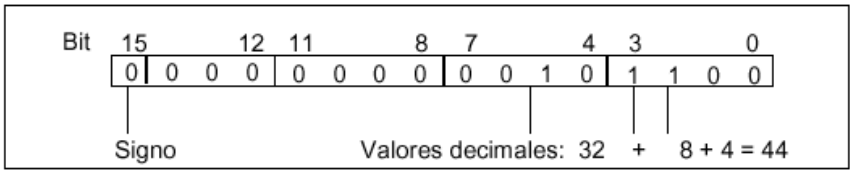
		hexadecimal	DW#16#0 a DW#16#FFFF_FFFF	L DW#16#3FT
		Decimal sin signo	B#(0,0,0,0) a B#(255,255,255,255)	L B#(0,1,2,3)
DINT	32	Decimal con signo	L# -2147483648 hasta L# 2147483647	L L#400000
REAL	32	Coma flotante	1.175 495e-38 a 3.402823e+38	L 23.5678
S5TIME	16	Tiempo S7	S5T#0H_0M_0S_10MS hasta S5T#2H_46M_30S_0MS	L s5t#2s
TIME	32	Tiempo IEC	- T#24D_20H_31M_23S_648MS hasta T#24D_20H_31M_23S_647MS	L T#2H
DATE	16	Fecha IEC	D#1990-1-1 hasta D#2168-12-31	L D#1994-3-15
TIME_OF_DAY	32	Hora del dia	TOD#0:0:0.0 hasta TOD#23:59:59.999	L TOD#1:10:3.3
CHAR	8	caracter	'A','B' etc.	L 'E'

Formato del tipo de datos INT (enteros de 16 bits)

El signo de un entero indica si se trata de un número entero positivo o negativo. El espacio que ocupa un entero (de 16 bits) en la memoria equivale a una palabra. La tabla siguiente muestra el margen de un entero (de 16 bits).

Formato	Grupo
Entero (16 bits)	–32 768 hasta +32 767

La figura siguiente muestra el entero +44 en forma de número binario.

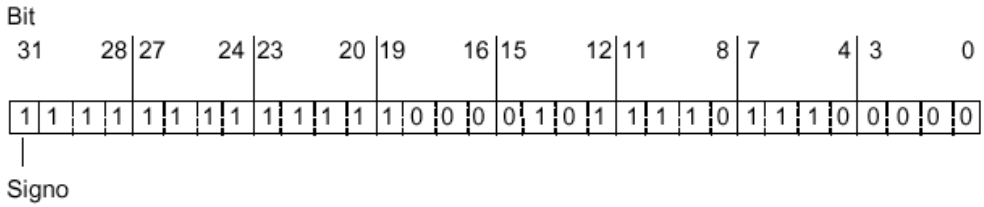


Formato del tipo de datos DINT (enteros de 32 bits)

El signo de un entero indica si se trata de un número entero positivo o negativo. El espacio que ocupa un entero (de 32 bits) en la memoria equivale a dos palabras. La tabla siguiente muestra el margen de un entero (de 32 bits).

Formato	Grupo
Entero (32 bits)	–2 147 483 648 hasta +2 147 483 647

La figura siguiente muestra el entero –500 000 en forma de número binario. En el sistema binario, la forma negativa de un entero se representa como complemento a dos del entero positivo. El complemento a dos de un entero se obtiene invirtiendo los estados de señal de todos los bits y sumando luego +1 al resultado.



Formato del tipo de datos REAL (números en coma flotante)

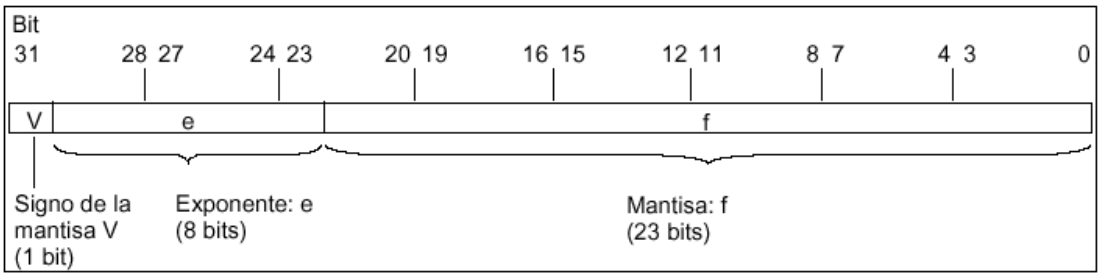
Un número en coma flotante es un número positivo o negativo que contiene un valor decimal como p. ej. 0,339 o –11,1. Un número en coma flotante también se puede dotar con un exponente para representar la potencia entera de 10 con la que se multiplica el número en coma flotante. Así se obtiene el valor que se desea representar. Por ejemplo, el número 1234 000 se puede representar mediante 1.234E6 ó 1.234e6 (es decir, 1.234 * 10^6). La tabla siguiente muestra los márgenes de los números en coma flotante.

Formato	margen[1)]
Números en coma flotante	–3.402 823E+38 hasta –1.175 495E–38 y " 0 y +1.175 495E–38 hasta +3.402 823E+38

1. Si el resultado de una operación en coma flotante se encuentra comprendido entre -1.175 495E-38 y -1.401 298E-45 o entre +1.401 298E-45 y +1.175 495E-45, significa que ha ocurrido un rebase por defecto (bits A1, A0, OV, OS de la palabra de estado). Este es el margen de los números desnormalizados.

En la memoria, un número en coma flotante ocupa el espacio de una palabra doble (32 bits). El bit más significativo indica el signo del número (bit 31, "0" significa "positivo", "1" significa "negativo"). Los demás bits representan el exponente y la mantisa.

La figura siguiente muestra los tres campos (V, e y f) de un número en coma flotante (de 32 bits).



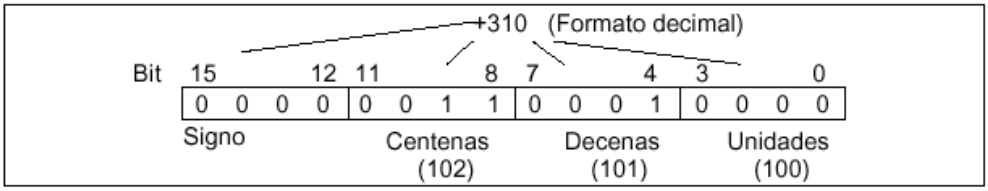
Formato de los tipos de datos WORD y DWORD en los números decimales codificados en binario

La notación codificada en binario (BCD) representa un número decimal en grupos de cifras binarias (bits). Un grupo de 4 bits representa una cifra de un número decimal o bien el signo de dicho número. Los grupos de 4 bits constituyen una palabra (16 bits) o una palabra doble (32 bits). Los cuatro bits más significativos indican el signo del número ("1111" significa "negativo" y "0000" significa "positivo"). Las instrucciones con operandos BCD sólo evalúan el bit más significativo (15 en el caso del formato de palabra o 31 en el de palabra doble). La tabla siguiente muestra el formato y el margen de los dos tipos de números BCD.

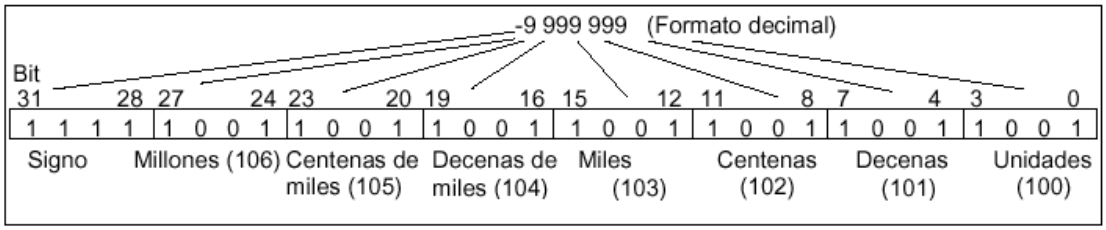
Formato	Grupo
Palabra (16 bits, número BCD de 3 cifras con signo)	–999 hasta +999
Palabra doble (32 bits, número BCD de 7 cifras con signo)	–9 999 999 hasta +9 999 999

Las figuras siguientes muestran ejemplos de un número decimal codificado en binario en los siguientes formatos:

- Formato de palabra

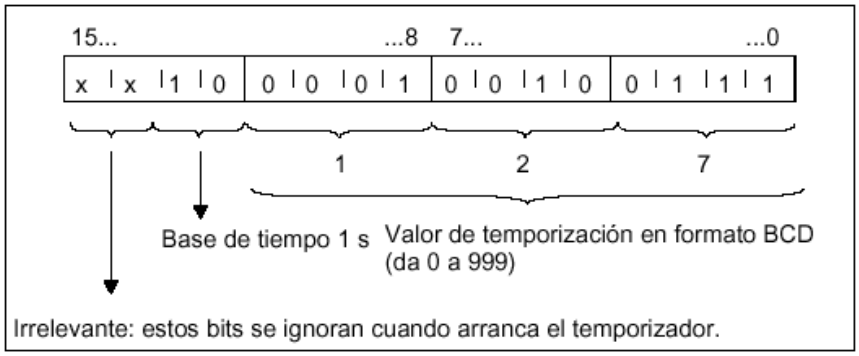


- Formato de palabra doble



Formato del tipo de datos S5TIME (intervalo)

Si introduce el intervalo con el tipo de datos S5TIME, sus entradas se guardarán en formato BCD. La figura siguiente muestra el contenido del operando de tiempo, donde el valor de tiempo es 127 y la base de tiempo es 1 s.



Al trabajar con S5TIME deberá introducir un valor de tiempo comprendido entre 0 y 999, así como determinar una base de tiempo (v. la tabla siguiente). La base de tiempo indica el intervalo en el que un temporizador reduce el valor de tiempo en una unidad hasta alcanzar "0".

Base de tiempo para S5TIME

Base de tiempo	Código binario para la base de tiempo
10 ms	00
100 ms	01
1 s	10
10 s	11

- Variables compuestas: aquellas que se basan en la agrupación de varias variables simples.

Tipo	Bits	Significado
DATE_AND_ TIME	64	Unión de una variable DATE y otra TIME_OF_DAY. Rango de valores: DT#1990-1-1-0:0:0.0 a DT#2089-12-31-23:59:59.999
STRING	-	Cadena de caracteres. Rango de valores: STRING[1] a STRING[254]
ARRAY	-	Array de un determinado tipo de variable simple. Ejemplo: ARRAY [1..20,1..10] of INT El número máximo de dimensiones de un array es de 6. E número máximo de índice por dimensión es de 32767.
STRUCT	-	Define un agrupamiento de tipos de datos compuestos o simples.

- Tipos de parámetros: además de los tipos de datos simples y compuestos, es posible definir tipos de parámetros para los parámetros formales que se transfieren entre los bloques

Parámetro	Capacidad	Descripción
TIMER	2 bytes	Designa un temporizador determinado que ha de ser utilizado por el programa en el bloque lógico llamado. Formato: `T1`
COUNTER	2 bytes	Designa un contador determinado que ha de ser utilizado por el programa en el bloque lógico llamado. Formato: `Z10`
BLOCK_FB BLOCK_FC BLOCK_DB BLOCK_SDB	2 bytes	Designa un bloque determinado que ha de ser utilizado por el programa en el bloque lógico llamado. Formato: `FC101` `DB42`
POINTER	6 bytes	Designa la dirección. Formato: `P#M50.0`
ANY	10 bytes	Se utiliza cuando el tipo de datos del parámetro actual no se conoce. Formato: `P#M50.0 BYTE 10` `P#M100.0 WORD 5`

TIMER o COUNTER: definen un determinado temporizador o contador que ha de ser utilizado en la ejecución. Si se utiliza un parámetro formal del tipo TIMER o COUNTER, el parámetro actual correspondiente debe ser un temporizador o un contador, es decir se debe indicar o una "T" o una "Z" seguida por un número entero positivo.

BLOCK: define un determinado bloque que ha de ser utilizado como entrada o como salida. La declaración del parámetro determina el tipo de bloque (FB, FC, DB etc.), que debe ser utilizado. Si se utiliza un parámetro formal del tipo BLOCK, se debe indicar la dirección del bloque como parámetro actual. Ejemplo: "FC101" (para direccionamiento absoluto) o "Válvula" (para direccionamiento simbólico).

POINTER: se refiere a la dirección de una variable. Un puntero contiene una dirección en lugar de un valor. Si se utiliza un parámetro formal del tipo POINTER, se debe indicar la dirección como parámetro actual. STEP 7 permite indicar un puntero en formato Pointer o simplemente como dirección (p. ej., M 50.0). Ejemplo para un formato Pointer para direccionamiento de los datos que comienzan en M 50.0:
`P#M50.0`

ANY: se utiliza cuando el tipo de datos del parámetro actual no se conoce o cuando se puede utilizar un tipo de datos cualquiera. Para obtener informaciones más detalladas acerca del parámetro ANY, consulte los apartados "Formato del tipo de parámetro ANY o Uso del tipo de parámetro ANY".

Uso del tipo de parámetro POINTER

Un puntero se utiliza para señalar a un operando. La ventaja de este tipo de direccionamiento es que el operando de la instrucción se puede modificar dinámicamente durante la ejecución del programa.

STEP 7 ofrece el formato de puntero: p#área de memoria byte.bit_dirección.

Los ejemplos siguientes muestran cómo introducir el tipo de parámetro POINTER para los datos que comiencen en M50.0:

- P#M50.0
- M50.0 (si el tipo de parámetro se ha declarado como POINTER)

Puntero para el direccionamiento indirecto por memoria

Las instrucciones del programa que utilizan el direccionamiento indirecto por memoria se componen de una operación, un identificador del operando y un desplazamiento (el desplazamiento se debe indicar entre corchetes cuadrados).

Ejemplo de un puntero en formato de palabra doble:

L	P#8.7	Cargar el valor del puntero en ACU1.
T	MD2	Transferir el puntero a MD2.
U	E [MD2]	Consultar el estado de señal en la entrada E 8.7,
=	A [MD2]	y asignar el estado de señal a la salida A 8.7.

Puntero para los direccionamientos intraárea e interárea

Las instrucciones de programación que utilizan estos tipos de direccionamiento se componen de una operación y las partes siguientes: identificador del operando, identificador del registro de direcciones, desplazamiento.

El registro de direcciones (AR1/2) y el desplazamiento se deben indicar juntos entre corchetes cuadrados.

Ejemplo del direccionamiento intraárea

El puntero no contiene una indicación del área de memoria:

L	P#8.7	Cargar el valor del puntero en ACU1.
LAR1		Cargar el puntero de ACU1 en AR1.
U	E [AR1, P#0.0]	Consultar el estado de señal en la entrada E 8.7 y
=	A [AR1, P#1.1]	asignar el estado de señal a la salida A 10.0.

El desplazamiento 0.0 no tiene efecto alguno. La salida 10.0 se calcula de 8.7 (AR1) más el desplazamiento 1.1. El resultado es 10.0 y no 9.8.

Ejemplo del direccionamiento interárea

En el caso del direccionamiento interárea, el área de memoria (en el ejemplo: E ó A) se indica en el puntero.

L	P# E8.7	Cargar el valor del puntero y el identificador de área en ACU1.
LAR1		Cargar el área de memoria E y la dirección 8.7 en AR1.
L	P# A8.7	Cargar el valor del puntero y el identificador de área en ACU1.
LAR2		Cargar el área de memoria A y la dirección 8.7 en AR2.
U	[AR1, P#0.0]	Consultar el estado de señal en la entrada E 8.7 y
=	[AR2, P#1.1]	asignar el estado de señal a la salida A 10.0.

El desplazamiento 0.0 no tiene efecto alguno. La salida 10.0 se calcula de 8.7 (AR2) más 1.1 (desplazamiento). El resultado es 10.0 y no 9.8

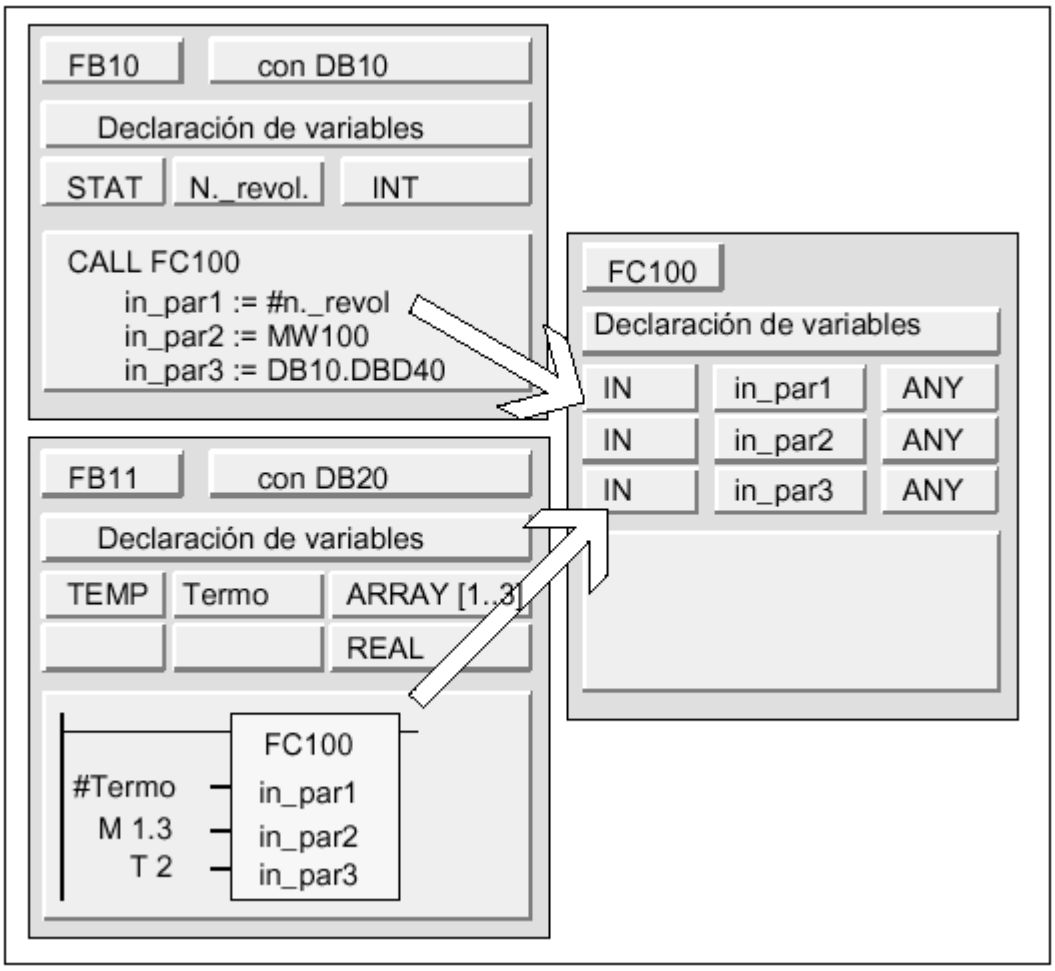
Uso del tipo de parámetro ANY

Para un bloque se pueden definir parámetros formales que sean apropiados para parámetros actuales con cualquier tipo de datos. Esto es sobre todo ventajoso cuando el tipo de datos del parámetro actual, que se suministra al llamar el bloque, es desconocido o puede variar (y cuando es admisible cualquier tipo de datos). En la declaración de variables del bloque se ha de declarar el parámetro con el tipo de datos ANY. En STEP 7 se puede asignar entonces un parámetro actual de un tipo de datos cualquiera.

Asignar un parámetro actual a un parámetro ANY

Si se declara un parámetro con el tipo de datos ANY, entonces se puede asignar a este parámetro formal un parámetro actual con un tipo de datos cualquiera. En STEP 7 se pueden asignar los siguientes tipos de datos como parámetros actuales:

- Tipos de datos simples: Se indica la dirección absoluta o el nombre simbólico del parámetro actual.
- Tipos de datos compuestos: Se indica el nombre simbólico de los datos con tipo de datos compuestos (p. ej., arrays y estructuras).
- Temporizadores, contadores y bloques: Introduzca el número (p. ej., T1, Z20 ó FB6).
- La figura siguiente muestra cómo se pueden transferir datos a una FC con parámetros del tipo de datos ANY.



En este ejemplo, FC100 tiene tres parámetros (*in_par1*, *in_par2* y *in_par3*), que fueron declaradas con el tipo de datos ANY.

- Cuando el FB10 llama la FC100, el FB10 entrega un número entero (la variable estática 'número de revoluciones'), una palabra (MW100) y una palabra doble en DB10 (DB10.DBD40).
- Cuando el FB11 llama la FC10, el FB11 entrega un campo de números reales (la variable temporal "termo"), un valor booleano (M 1.3) y un temporizador (T2).

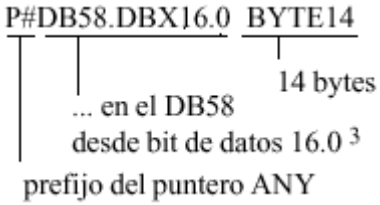
Indicar un área de datos para un parámetro ANY

A un parámetro ANY se le pueden asignar no sólo operandos individuales (p. ej. MW100), sino que es posible indicar también un área de datos. Si se desea asignar un área de datos como parámetro actual, entonces se debe utilizar el siguiente formato de una constante para indicar la cantidad de datos a transferir:

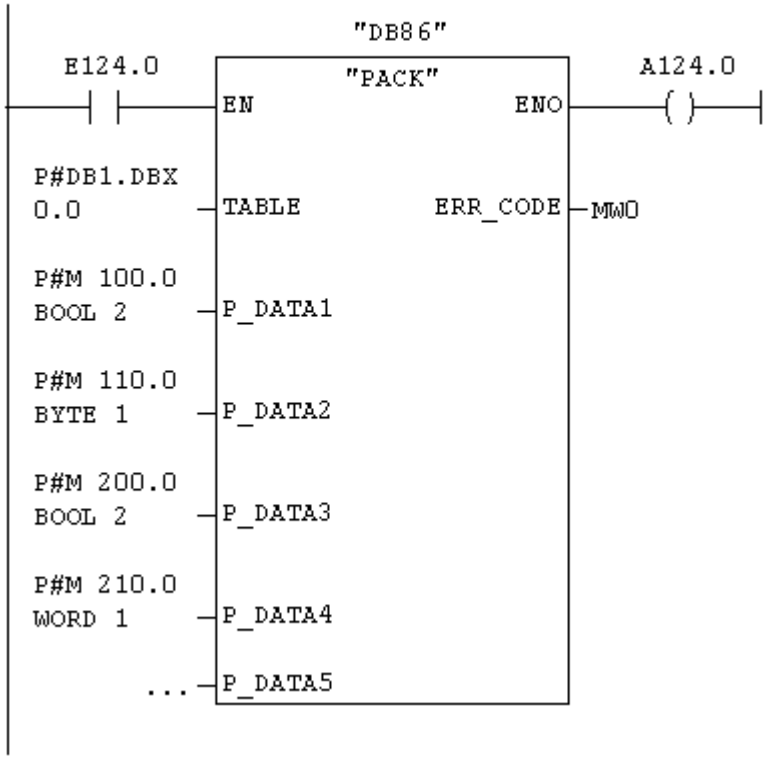
p# *identif. de área Byte.Bit* *tipo de datos* *factor de repetición*

Para el elemento *tipo de datos* en el formato para constantes se pueden indicar todos los tipos de datos simples y el tipo de datos DATE_AND_TIME. Si el tipo de datos no es BOOL, entonces debe indicarse la dirección de bit 0 (x.0). La tabla siguiente muestra ejemplos de formato para indicar las áreas de memoria que se desean transferir a un parámetro ANY.

Parámetro actual	Descripción
p# M 50.0 BYTE 10	Indica 10 bytes en el área de memoria Marcas: MB50 a MB59.
p# DB10.DBX5.0 S5TIME 3	Indica 3 unidades de datos del tipo S5TIME, memorizadas en el DB10: DB byte 5 a DB byte 10.
p# A 10.0 BOOL 4	Indica 4 bits en el área de memoria Salidas: A 10.0 a A 10.3.



Ejemplo de parametros ANY en la función FB86-PACK

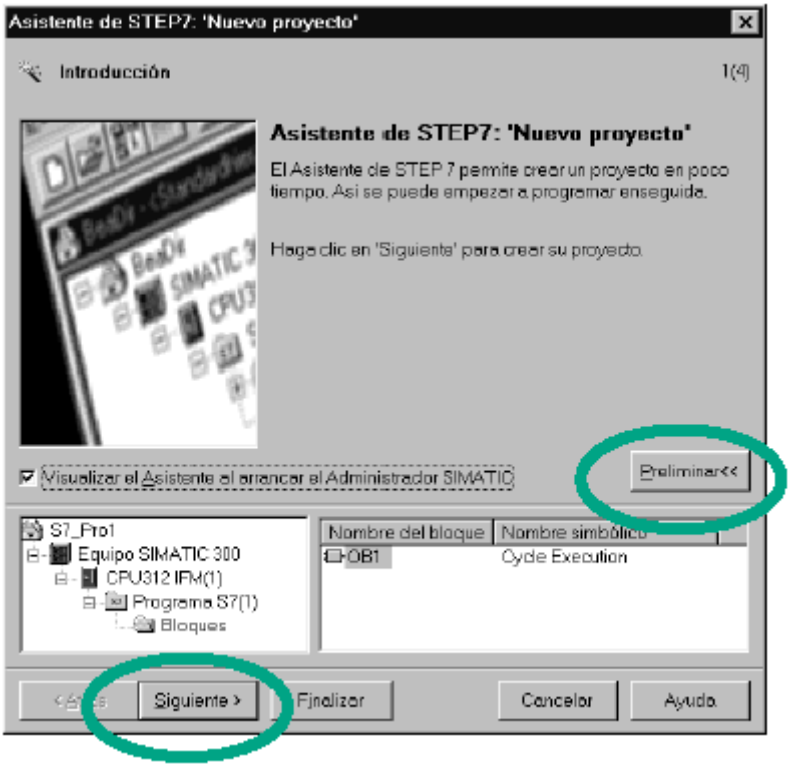


3

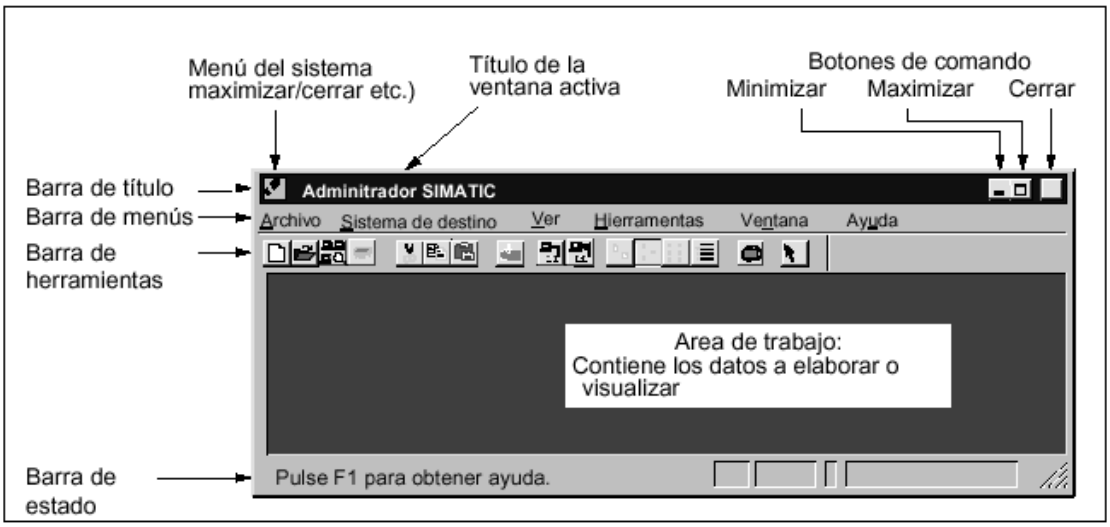
Descripción y manejo del software de programación STEP 7

3.1 Iniciar el software Step 7. EL administrador SIMATIC

Una vez iniciado Windows, aparece el icono ADMINISTRADOR SIMATIC  , que permite acceder al software Step 7, haciendo doble clic en el icono, se abre el asistente de Step 7 de forma estándar cada vez que se arranca el programa, para ayudarnos a crear un nuevo proyecto. El Asistente consulta los datos necesarios en diversos cuadros de diálogos y crea luego el proyecto.



Al pulsar el botón de comando finalizar se abre el Administrador SIMATIC, mostrando la ventana del proyecto creado, que es la pantalla inicial para programar con Step 7, y de donde se accede a todas las funciones y ventanas de STEP 7 mediante la barra de menús y herramientas :

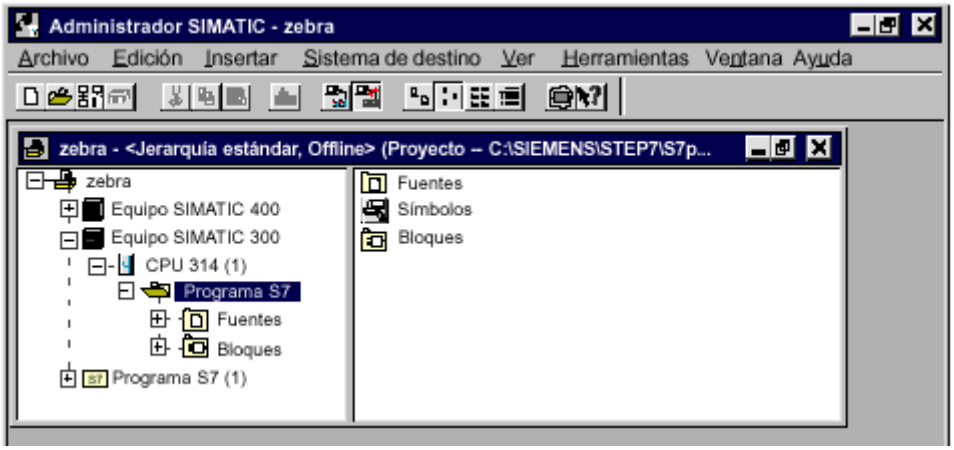


- Archivo: Abrir, organizar e imprimir proyectos
- Edición e Insertar: Editar bloques e insertar componentes del programa.
- Sistema de destino: Carga el programa y supervisa el Hardware.
- Ver, Herramientas y Ventana: Ajustar la representación y disposición de las ventanas, seleccionar un lenguaje y editar los datos de proceso.
- Ayuda: Llamar la ayuda de Step 7

3.2 Crear y elaborar proyectos

Los proyectos sirven para almacenar de forma ordenada los datos y programas necesarios para crear una tarea de automatización. Por ello la creación de un proyecto o de la estructura de un proyecto constituye el primer paso para trabajar con Step 7
El software para la CPUs se guarda en contenedores de programas. Para SIMATIC S7 los objetos de este tipo se denominan “Programa S7”
La manera más fácil de crear un nuevo proyecto es utilizando el asistente “Nuevo proyecto” o elegir “Nuevo” del menú Archivo. Para seguir elaborando el proyecto 2 alternativas:
1º) Configurar primero el Hardware. Una vez efectuada la configuración se habrán insertado ya los contenedores “Programa S7”
2º)Configurar primero el software sin haber configurado el hardware y realizar la configuración posteriormente, y luego asignar el programa S7 a una CPU

La ventana de proyecto se divide en dos partes:
En la parte izquierda de la ventana del proyecto se ve la estructura del proyecto.
En la parte derecha de la ventana del proyecto aparecen los objetos y carpetas que contiene la carpeta seleccionada en la ventana izquierda.



Los componentes principales que aparecen en un proyecto son:

- Icono del proyecto (nombre del proyecto)
- Equipo y modulo programable (CPU) contienen los datos de configuración y parametrización del software
- Programa S7 que contiene los siguientes componentes:
    1. -"Bloques" creados para programar en KOP,FUP.AWL
    2. -"Fuentes" para programar en lenguajes especiales
    3. -"Símbolos" para programar en vez de direcciones absolutas con símbolos.

Para insertar un programa S7 independientemente de la configuración del Hardware:

1. Seleccione el icono de la ventana de proyecto
2. Menú "insertar"_"Programa S7"
3. El programa S7 se crea debajo del proyecto

Para asignar a un módulo programable un programa S7 creado independientemente de la configuración del hardware:

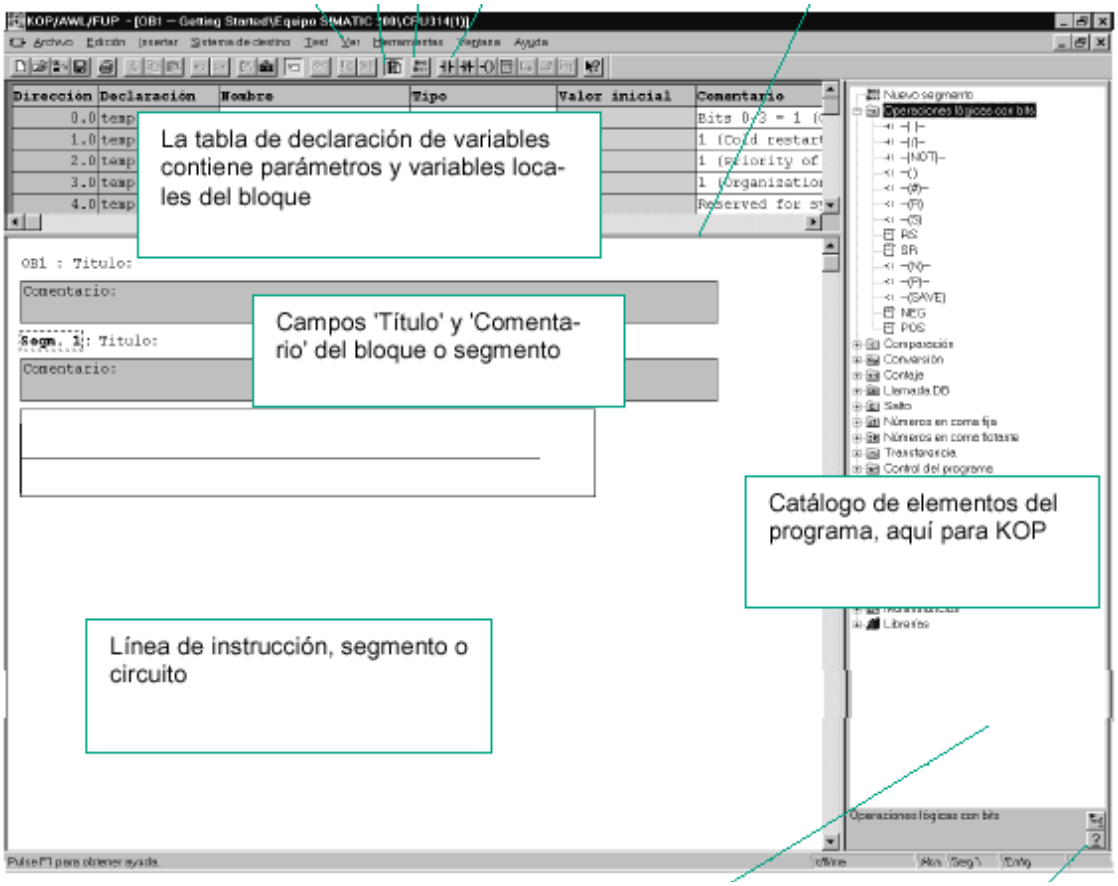
1. Seleccione el programa S7 no asignado
2. Arrastre luego el programa seleccionado hasta el modulo programable al que desea asignarlo.
3. Abrir la tabla de configuración del módulo programable y guarde la configuración de nuevo.

O también accediendo al sistema destino sin haber configurado el hardware:

1. Menú "Ver"_"Online"
2. Menú "Ventana"_"organizar"
3. En la ventana Online abra el programa S7
4. En la ventana Offline seleccione los objetos que desee cargar en el sistema destino
5. Arrastre los objetos seleccionados hasta el contenedor de "Bloques" en la ventana online

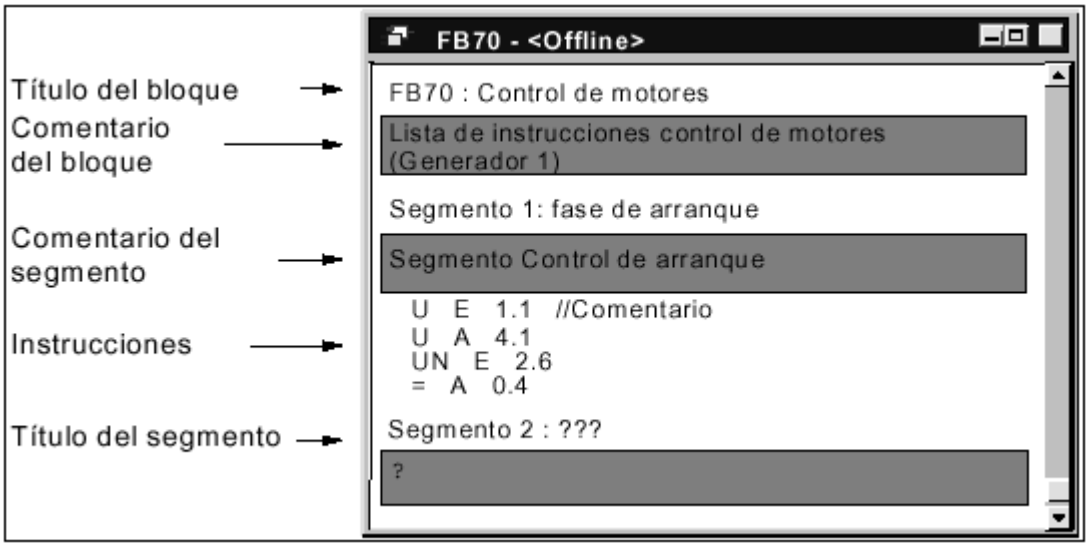
3.3 Editor de Bloques lógicos.Ventana KOP/AWL/FUP.Area de instrucciones

Una vez creado el bloque lógico, se arranca el lenguaje seleccionado (KOP,AWL o FUP) haciendo doble click sobre él, y aparecerá una ventana (Ventana KOP/AWL/FUP) dividida en dos áreas que se compone del área de instrucciones y del área de declaración de variables.



En el área de instrucciones se edita el ciclo del bloque lógico, es decir, aquí se introduce el programa en varios segmentos.Los componentes del área de instrucciones son:

- Titulo del bloque
- Comentario del bloque
- Titulo del segmento
- Comentario del segmento
- Instrucciones o elementos dentro de los segmentos

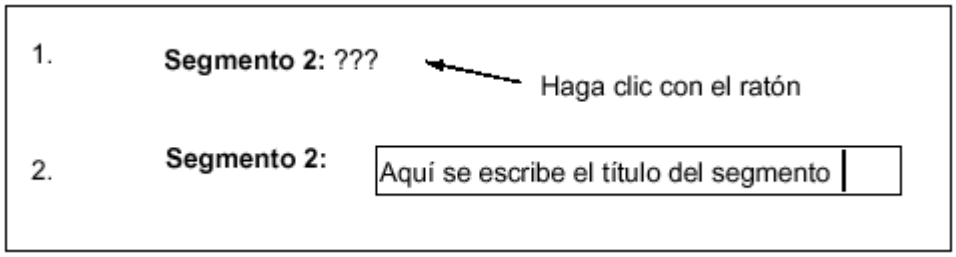


El editor comprueba las sintaxis inmediatamente después de introducir una instrucción y le indica las entradas erróneas en rojo o en cursiva.

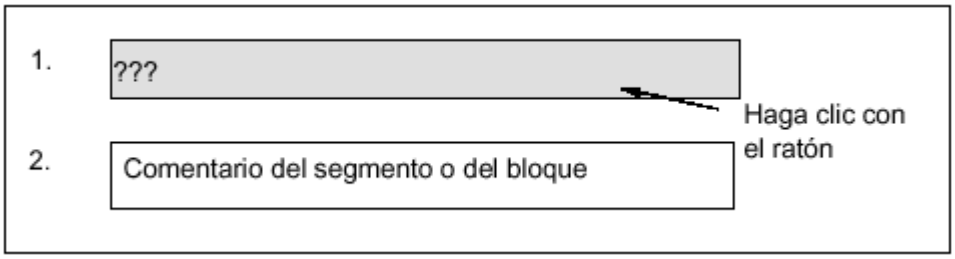
Introducir Títulos y comentarios:

Posicionar el cursor sobre ??? que esta a la derecha al laso del nombre del bloque o de segmento. Ej: segmento 1: ???

Se abrirá un cuadro de texto. Escribir nombre no más de 64 caracteres.



Para introducir comentarios hacer doble clic sobre uno de estos cuadros de comentario e introducir la explicación que se desee (se dispone de 64 Kb para los comentarios de bloque y segmento por cada bloque)



Se puede visualizar u ocultar el cuadro gris del comentario en menú Ver-mostrar-comentario.

Introducir instrucciones:

Para crear un nuevo segmento en menú Insertar_Segmento o clic en el icono de la barra de herramientas.

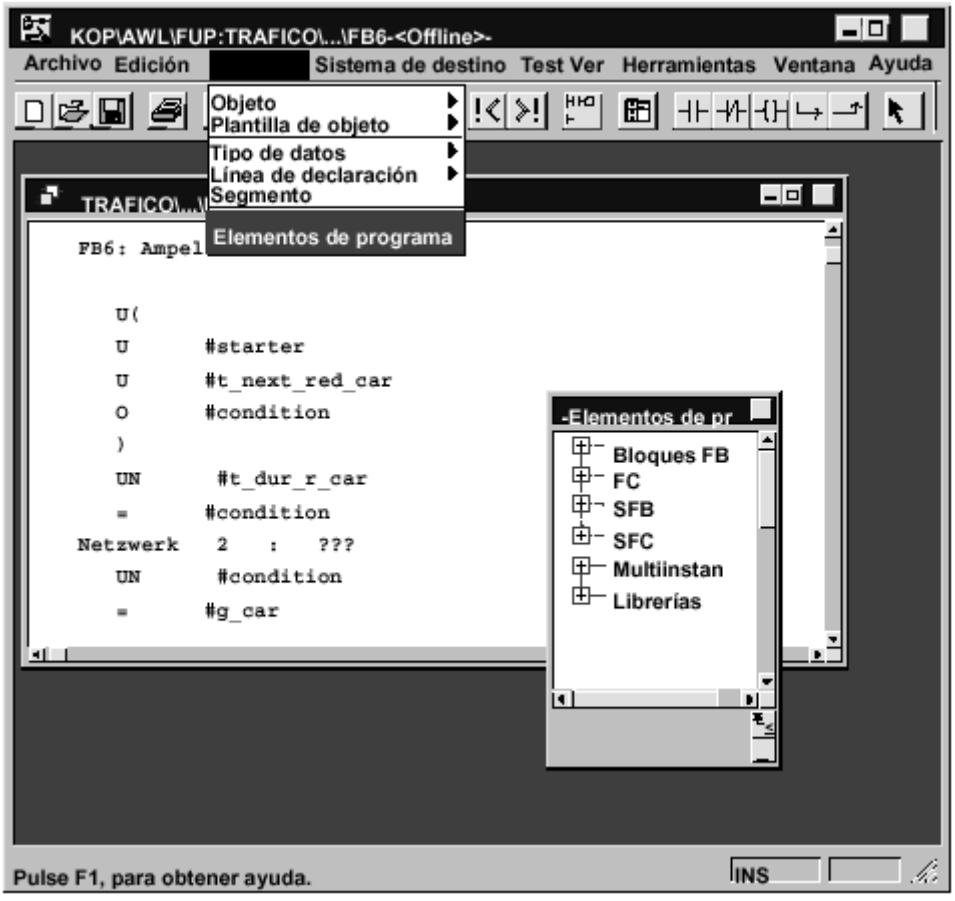
Para editar se dispone de las funciones usuales de edición en la barra de herramientas:

- Contacto normalmente abierto (F2) 
- Contacto normalmente cerrado (F3) 
- Bobina (F7) 
- Abrir rama (F8) 
- Cerrar rama (F9) 

Para buscar errores en edición en el comando de menú Edición >Ir a>error previo/error siguiente y se accede a los errores que se encuentran fuera del área actualmente visible.
Si se activa la barra de estado del menú Ver aparecerá allí indicaciones sobre los errores.

Instrucciones del catalogo de elementos de programa:

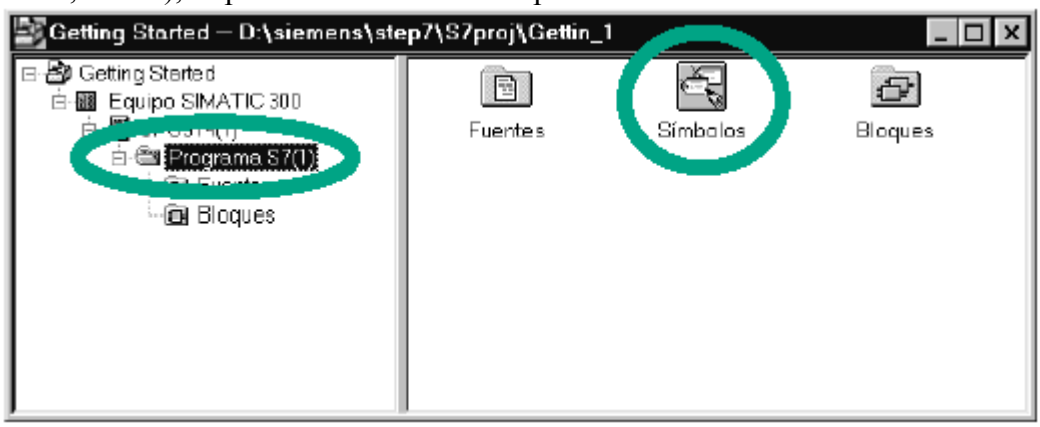
El catalogo “elementos de programa”  comprende elementos de los lenguajes KOP y FUP.



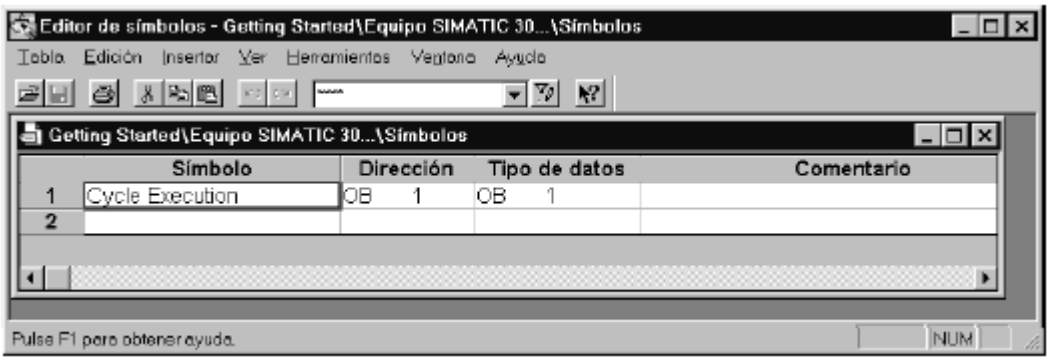
Así como multiinstancias ya declaradas, bloques ya listos y bloques de librerias.Dicho catálogo se puede llamar con el comando de menú Ver>Catálogo o el comando de menú Insertar>elementos de programa

3.4 Direccionamiento simbólico. Tabla de símbolos

En un programa de **STEP 7** se utilizan operandos tales como señales de Entradas,Salidas,Marcas,Contadores,Temporizadores,bloques de datos y bloques de función. Si así lo desea, en vez de direccionarlos de forma absoluta (Ej: E 126.1,FB21..), es posible direccionar un operando mediante un nombre simbólico.



Al crear un programa S7, se genera automaticamente una tabla de simbolos (vacia).Una vez seleccionado el objeto “simbolos” del contenedor de bloques, hacer doble click y aparece la ventana “Editor de simbolos”, visualizandose allí la Tabla de simbolos.



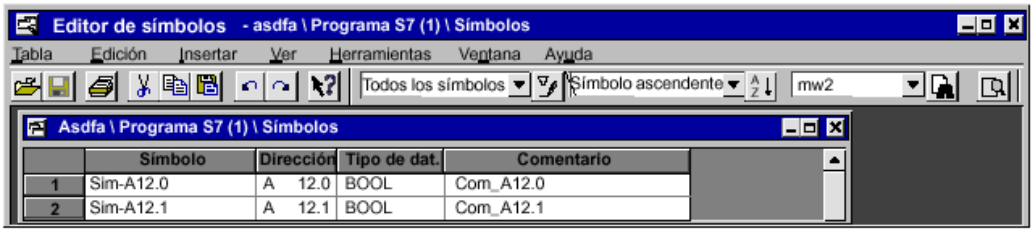
Se distinguen entre símbolos Globales y Locales:
Los símbolos **globales** se definen en la Tabla de símbolos y se representan escritos entre comillas “..” en el área de instrucciones. Pueden ser utilizados por todos los bloques del programa.
Los símbolos **locales** se definen en la Tabla de declaración de variables del bloque al introducir el programa.Se representan con el signo # por delante, esto es, antepuesto al símbolo. Se puede aplicar sólo en el bloque donde fueron definidos.
El usuario no tiene que introducir el distintivo (“..” o #) sino que el programa añade éste automáticamente al efectuar la comprobación.

La Tabla de símbolos:

Para poder usar símbolos globales tiene que registrarlos en la Tabla de simbolos.Hay 2 posibilidades para abrir una tabla de símbolos:

- Haciendo doble click en el objeto “simbolos” de la ventana de proyecto
- Seleccionar menú herramientas>Tabla de simbolos

Para introducir simbolos en la tabla, vaya a la primera línea vacía de la misma y rellene los campos:



Símbolo

El nombre de un símbolo puede comprender **24** caracteres como máximo. Una tabla de símbolos puede contener 16000 símbolos como máximo.

Dirección

Una dirección sirve para identificar un operando determinado.
Ejemplo: E 12.1

La sintaxis de la dirección se comprueba inmediatamente al introducirla. Además, se verifica si es posible asignar la dirección al tipo de datos indicado.

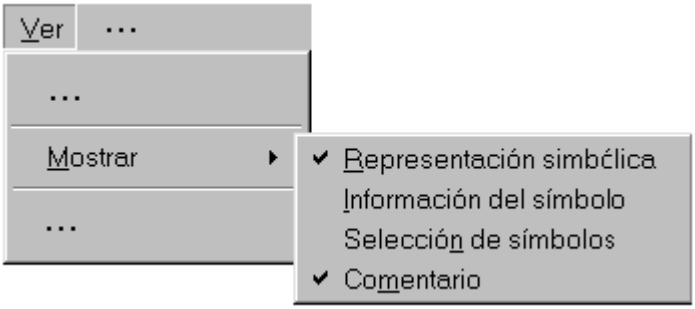
Tipo de datos

Puede elegir uno de los diversos tipos de datos disponibles en STEP 7. En el campo aparece un tipo de datos predeterminado que se puede cambiar en caso necesario. Si el cambio no concuerda con la dirección o si la sintaxis es errónea, aparecerá un mensaje de error al salir del campo.

Comentario

Es posible asignar comentarios a cada uno de los símbolos. Combinando símbolos breves con comentarios detallados se crea tanto un programa efectivo como una buena documentación del mismo. Los comentarios pueden comprender 80 caracteres como máximo.

En menú Ver>mostrar>representación simbólica, es posible mostrar u ocultar los símbolos



En menú Ver>mostrar>información del símbolo, se muestra la dirección absoluta, símbolo y comentario

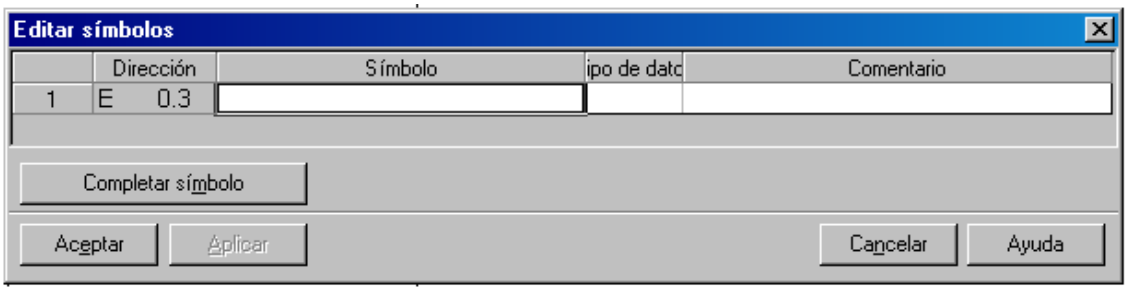
Información del símbolo:

```
E124.0          marcha                  pulsador on
E124.1          rearme                  defecto
E124.2          paro                    pulsador off
```

En menú Ver de la tabla de símbolos, con el comando “ordenar”, los registros de la tabla de símbolos se pueden ordenar alfabéticamente por simbolos,por direcciones, por tipo de datos o por comentario, y con el comando “símbolo” del menú Insertar se pueden insertar nuevas líneas para crear nuevos símbolos

Definir un solo símbolo en un cuadro de dialogo:

Para definir símbolos de forma individual, como por ejemplo, si mientras esta programando, se da cuenta que hace falta un símbolo o que es necesario corregir un símbolo ya existente, entonces no es necesario visualizar la Tabla de símbolos, sino en un cuadro de dialogo:



Para definir símbolos individuales durante la programación:

1. Ver>Representación simbólica
2. Seleccionar la dirección absoluta que desea asignar un simbolo en el area de instrucciones
3. Edición>Simbolo
4. Rellene el cuadro de dialogo

Resultado: el simbolo definido se introduce en la tabla de simbolos

3.5 La Tabla de declaración de variables.Datos locales de STEP 7

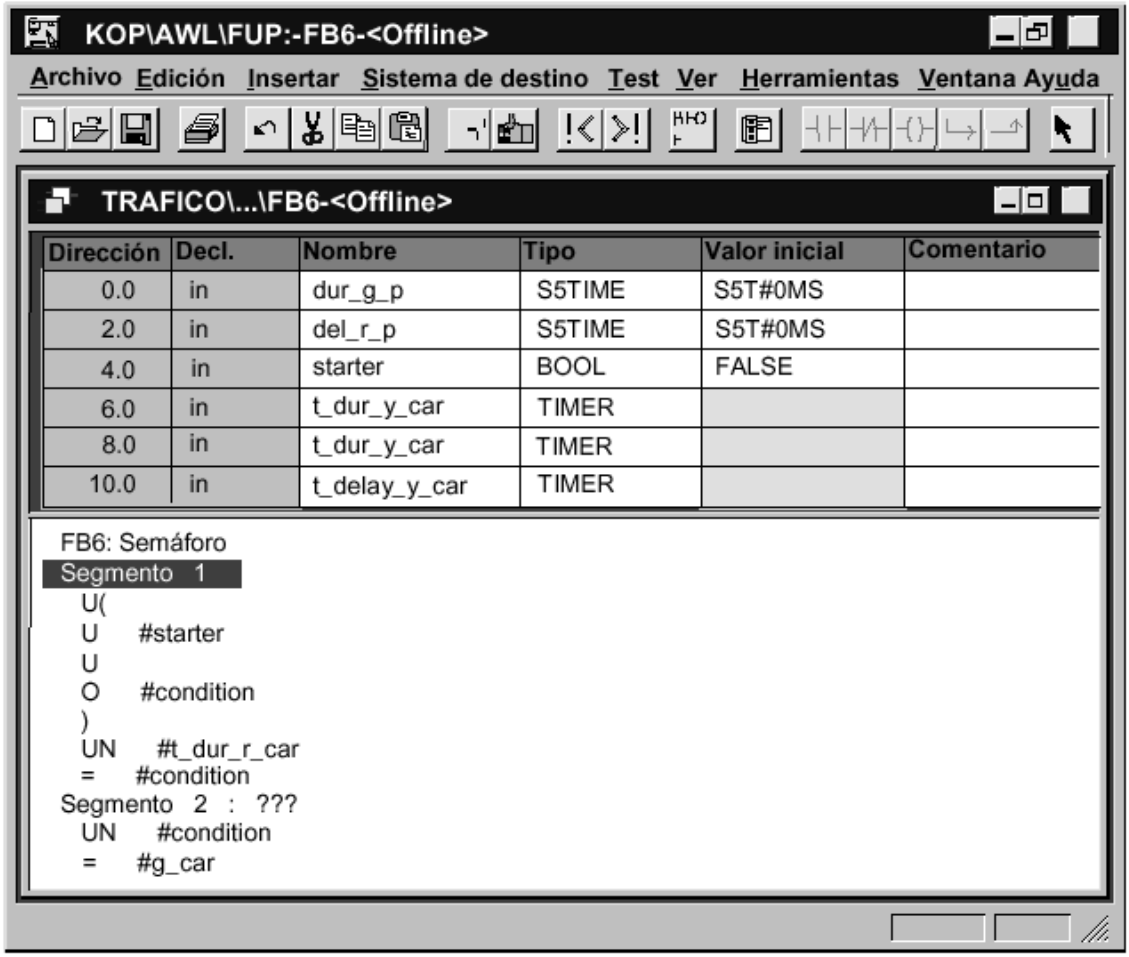
Una vez que haya abierto un bloque lógico recién creado, además de las instrucciones del programa de usuario, los bloques contienen variables que han de ser declaradas usando STEP 7 a fin de programar dichos bloques. En dicha declaración se pueden indicar las variables que el bloque ha de utilizar durante su tratamiento. Estas variables se visualizan en un tabla de declaración de variables predeterminada y son los datos locales de STEP 7 que dependiendo del bloque del que se trate pueden ser:

- Parámetros de bloques, que se transfieren entre los bloques lógicos
- Datos locales estáticos (STAT), pueden ser utilizados en cualquier bloque de función y se depositan en un bloque de datos de instancia asociado
- Datos locales temporales (TEMP) que solo están disponibles durante el tratamiento del bloque y sirven para guardar datos de forma intermedia en bloques.El espacio de memoria que ocupa se libera en cuanto se termina de ejecutar el bloque. El sistema operativo asigna un área de memoria propia a los datos temporales conocida como pila de datos locales

En la pila LSTACK se almacenan:

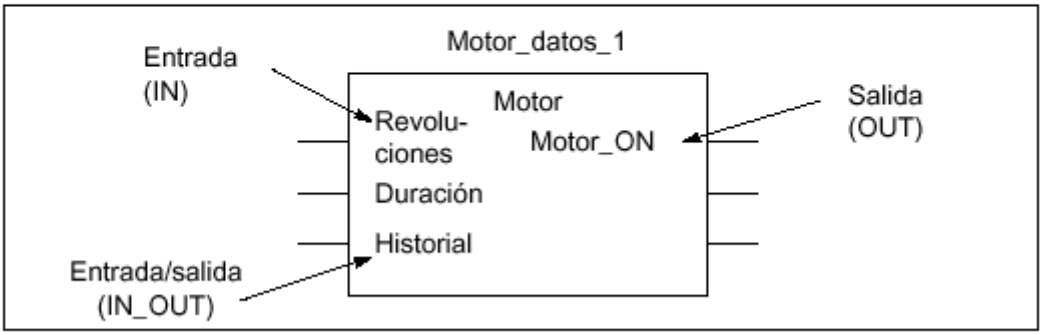
- las variables temporales de los datos locales de bloques
- la información de arranque de los bloques de organización
- informaciones para la transferencia de parámetros
- resultados intermedios de la lógica en programas escritos en Esquema de contactos.

Al declarar las variables se reserva suficiente espacio de memoria en la pila de datos locales para las variables temporales y en el caso de de los bloques de función, para las variables estáticas del DB de instancia que e asociará posteriormente.
Al declarar variables en un bloque de función dichas variables determinan también la estructura de todos los DBs de instancia que se asocien al FB.
La tabla de declaración de variables y el área de instrucciones de los bloques lógicos están directamente relacionados una con la otra, pues en el área de instrucciones se utilizan los nombres de la tabla de declaración de variables. Por tanto las modificaciones que se hacen en la tabla de declaración de variables afectan al área de instrucciones.
Los datos locales se direccionan generalmente con nombres simbólicos



Para cada parámetro formal se ha de indicar un tipo de declaración y un tipo de datos.

Es necesario indicar cómo un parámetro ha de ser utilizado por el bloque lógico. Los parámetros se pueden definir como valor de entrada o de salida. Sin embargo, también se pueden utilizar como variable de entrada/salida, que se transfiere al bloque y es emitida luego por dicho bloque.



Definición de los parámetros de entrada, salida y E/S de un bloque lógico

Tipos de declaración para parámetros y variables locales

Parámetro/ variable	Descripción	Admisible en
IN	Parámetro de entrada puesto a disposición por el bloque lógico invocante.	FB, FC
OUT	Parámetro de salida puesto a disposición por el bloque de datos llamado.	FB, FC
IN_OUT	Parámetro cuyo valor es puesto a disposición por el bloque invocante, modificado por el bloque llamado y devuelto al bloque invocante.	FB, FC
STAT	Variable estática que se memoriza en un DB de instancia.	FB
TEMP	Variable temporal que se memoriza en la pila de datos locales. Tras el tratamiento del bloque, el valor de las variables ya no está disponible.	FB, FC, OB

En los bloques de función FBs, el bloque de datos de instancia memoriza los datos que han sido declarados como IN, OUT, IN_OUT y como variables estáticas STAT. Las variables temporales TEMP no son memorizadas.

Las FCs no pueden contener variables estáticas. Los parámetros de entrada, salida y de entrada/salida se memorizan como punteros de los parámetros actuales suministrados por el bloque invocante.

La Tabla de declaración de variables comprende entradas para la dirección, el tipo de declaración, el nombre, el tipo de datos, el valor inicial y el comentario de la variable

Columna	Significado	Observaciones	Ejecución
Dirección	Dirección en formato BYTE.BIT.	Si se trata de tipos de datos que requieran más de un byte, la dirección muestra la asignación con un salto a la siguiente dirección de byte. Leyenda: * : tamaño de un elemento de campo en bytes. + : dirección inicial en relación con el comienzo de la estructura = : memoria total requerida por una estructura	Entrada de sistema: El sistema adjudica y visualiza la dirección al terminar de declarar una entrada.
Variable	Nombre simbólico de la variable	El nombre debe comenzar con una letra. No está permitido utilizar palabras clave reservadas.	necesaria
Declaración	Tipo de declaración, "Finalidad" de la variable	Dependiendo del tipo de bloque, se permiten: Parámetros de entrada "in" Parámetros de salida "out" Parámetros de entrada/salida "in_out" Variables estáticas "stat" Variables temporales "temp"	Estándar conforme al tipo de bloque
Tipo de datos	Tipo de datos de la variable (BOOL, INT, WORD, ARRAY etc.).	Los tipos de datos simples se pueden elegir del menú emergente al oprimir la tecla derecha del ratón.	necesaria
Valor inic.	Valor inicial si el software no debe adoptar el valor estándar.	Debe ser compatible con el tipo de datos. Al guardar por primera vez un bloque de datos, el valor inicial se adoptará como valor actual de la variable, a menos que defina expresamente un valor actual.	opcional
Comentario	Comentario explicativo		opcional

Tipos de datos

BOOL BYTE WORD DWORD	Combinaciones binarias desde 1 bit (tipo BOOL) hasta 32 bits (DWORD).
CHAR	Un carácter del juego de caracteres ASCII.
INT DINT REAL	Valores numéricos (p.ej.: para calcular expresiones aritméticas).
S5TIME TIME DATE TIME_OF_DAY	Valores de hora y fecha de STEP 7 (p.ej.: para ajustar la fecha o introducir la hora)

3.6 Cargar el programa de usuario.Ventana proyecto modo Online.

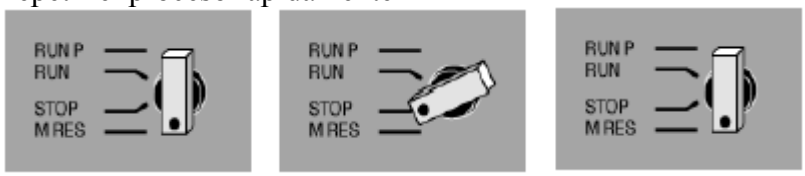
Para ejecutar el programa de usuario es necesario cargarlo previamente en la CPU del autómata. Para ello deberán cumplirse los siguientes requisitos:

1. El programa que se desea cargar se deberá haber compilado sin errores.
2. Existe un enlace ONLINE entre la PG y el AG.
3. La CPU ha de estar en modo STOP o RUN-P.
4. Borrar totalmente la CPU
5. Cargar el programa en la CPU (Todos o de bloque en bloque)

Borrar la CPU:

Para realizar un borrado total en un equipo S7 desde Hardware es necesario seguir los siguientes pasos:

- Girar el selector frontal de modo de la CPU a la posición MRES y mantenerlo durante al menos tres segundos hasta que el led STOP rojo parpade lentamente (luce alternativamente dos veces)
- Soltar el selector, que vuelve a la posición de STOP, y antes de tres segundos volver a ponerlo en MRES, al menos durante un segundo.Si el led STOP parpadea rápidamente significa que ha finalizado el borrado total de la CPU.Sino parpadea repetir el proceso rápidamente



Cargar el programa en la CPU:

- Con el selector de modo en posición STOP, abrir la ventana de proyecto (modo offline) del Administrador Simatic, y seleccionar el contenedor de bloques que se desea cargar y activar el comando cargar del menú “Sistema de destino”. Girar el selector de modo hasta la posición RUN, se enciende el led verde y la CPU está lista.
- Con el selector de modo en posición RUN-P, podemos cargar los bloques de uno en uno.Es necesario cargar siempre los bloques de un nivel inferior al superior, es decir, en el orden inverso a su llamada en el OB1

Ventana de proyecto modo ONLINE:

Con la ventana “Online” se visualizarán los bloques de programas después de la operación de cargar. Estos se pueden borrar o sobrescribir en online haciendo doble click sobre el bloque deseado.

- Para borrar elegir este comando del menú archivo o pulsar tecla supr.
- Los bloques existentes en la memoria RAM se pueden sobrescribir con una nueva versión.Esto remplaza el contenido anterior del bloque

Al borrar y sobrescribir bloques con frecuencia se forman huecos en la memoria que reducen el espacio de memoria aprovechado. Comprimiendo la memoria de usuario es posible reposicionar los bloques.

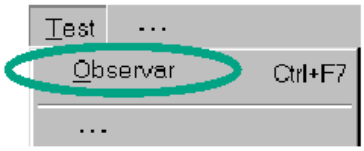
¿Diferencias entre Guardar , Cargar y Guardar Como?

El comando **"Guardar"** del menú archivo, permite guardar los cambios en modo OFFLINE, es decir se guarda el bloque siempre en el contenedor de "bloques" dentro del disco duro de la PG o PC, mientras que el comando **"Cargar"** del menú sistema de destino permite cargar en el AG los bloques que se hayan abierto.

El comando **"Guardar como"** del menú archivo permite almacenar los bloques de la CPU seleccionados en el contenedor de bloques de programa S7 del PC.

3.7 Test de Visualización del estado de programas

Con la función 'Status' (observar el programa) se puede probar el programa de un bloque. Para ello tiene que haberse establecido una conexión online con la CPU, la CPU tiene que estar en RUN o RUN-P y el programa tiene que haberse cargado en la CPU.



Active la función **Test > Observar**. 

Funciones de test permitidas:

- Observar/forzar variables
- Información del módulo
- Estado operativo

Antes de iniciar el Test para conocer el estado del programa debe determinar como desea visualizar la circulación de la corriente

La visualización del **estado de programa** se actualiza cíclicamente.

Identificadores preajustados

- El estado se cumple: líneas verdes continuas
- El estado no se cumple: líneas azules punteadas
- El estado es desconocido: líneas negras continuas

Este ajuste (tipo de línea y color) se puede modificar seleccionando el comando de menú **Herramientas > Preferencias** / Ficha KOP/FUP.

Estado de los elementos

- El estado de un contacto
- se cumple si el valor del operando es "1",
- no se cumple si el valor del operando es "0",
- es desconocido si el valor del operando es desconocido.
- El estado de elementos con una salida de habilitación (ENO) corresponde al estado de un contacto con el valor de la salida ENO como operando.
- El estado de elementos con salida Q corresponde al estado de un contacto con el valor del operando.
- El estado en CALLs se cumple si tras la llamada se ha puesto a 1 el bit RB.
- El estado de una operación de salto se cumple si se realiza el salto; es decir, si la condición del salto se cumple.
- Los elementos con salida de habilitación (ENO) se representan en negro si la salida de habilitación no está asignada a una conexión.

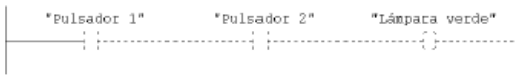
Estado de las líneas

- Las líneas son negras si no ha circulado corriente por ellas o si el estado es desconocido.
- El estado de aquellas líneas que comiencen en la barra de alimentación se cumple siempre ("1").
- El estado de aquellas líneas que se encuentren al comienzo de ramas paralelas se cumple siempre ("1").
- El estado de la línea situada tras un elemento se cumple si se cumplen tanto el estado de la línea situada antes del elemento como el estado del elemento.
- El estado de la línea situada tras NOT se cumple si no se cumple el estado de la línea situada antes de NOT (y viceversa).
- El estado de la línea situada **tras** la confluencia de varias líneas se cumple si
  - se cumplen tanto el estado de como mínimo una línea situada **antes** de la confluencia de líneas
  - como el estado de la línea situada antes de la rama.

Estado de los parámetros

- Los valores de los parámetros **en negrita** son actuales.
- Los valores de los parámetros en letra normal provienen de un ciclo anterior; este punto del programa no se ha ejecutado en el ciclo actual.

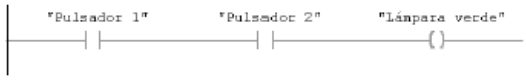
Probar el programa con KOP



Se visualiza la conexión en serie del segmento 1 en KOP. Hasta el pulsador 1 (E 0.1), el circuito se representa mediante una línea continua, es decir, se aplica tensión hasta el pulsador 1.

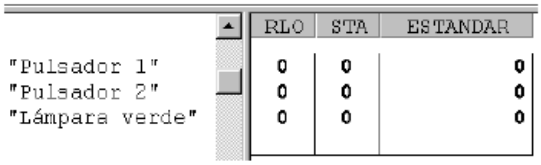
Probar el programa con FUP

El estado de señal se marca con un "0" o un "1". La línea discontinua significa que no hay resultado lógico.

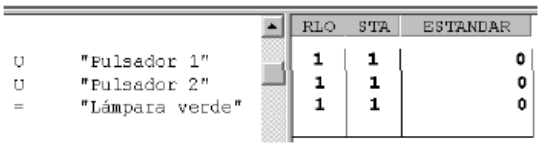


En los lenguajes de programación KOP / FUP puede seguir el test fijándose en el cambio de color del segmento programado. El color cambia cuando se cumple el resultado lógico.

Probar el programa con AWL



En AWL los
– resultados lógicos (RLO),
– bits de estado (STA) y
– estados estándar (STANDARD)
se visualizan en forma de tabla.



En el lenguaje de programación AWL, cambia el contenido de las columnas STA y RLO si se cumple el resultado lógico.

Forzar variables en el estado del programa

Requisito: El bloque online deberá estar abierto.

Las acciones descritas a continuación efectúan un forzado único e inmediato de las variables seleccionadas.

Forzar variables del tipo de datos BOOL:

1. Seleccione el operando que quiere forzar.
2. Elija el comando de menú **Test > Forzar a 1** o **Test > Forzar a 0**.

Forzar variables no booleanas:

1. Seleccione el operando que quiere forzar.
2. Elija el comando de menú **Test > Forzar**.
3. En el cuadro de diálogo visualizado, introduzca el valor que debe adoptar la variables (valor de forzado).
4. Cierre el cuadro de diálogo.

Método alternativo

1. Sitúe el cursor en el operando que desea forzar.
2. Pulse la tecla derecha del ratón y, en el menú emergente, elija el correspondiente comando para forzar.

3.8 La Tabla de variables VAT.Observar y forzar variables

Para comprobar programas de usuario se dispone de las siguientes posibilidades para intervenir en la ejecución del programa:

1. pueden hacerse mostrar (observar) los valores de las variables o asignar valores a las mismas (forzar). Ello es aplicable a las entradas, salidas, marcas, temporizadores, contadores, salidas periféricas, así como a los elementos de bloques de datos.
2. A excepción de los temporizadores y contadores, así como de los elementos de algunos bloques de datos, es posible predeterminar valores fijos que no puede modificar el programa de usuario (forzado permanente). Para ello es necesario que la CPU (p.ej. S7-400) ofrezca esta prestación.

Si el programa de usuario ya está compilado y se ha cargado en una CPU, es posible consultar el estado de los elementos de algunos bloques de datos y variables, p.ej. para

- poner en servicio una instalación o una parte de la misma,
- comprobar la interacción con otras partes de la instalación o del programa de usuario.

El test de las variables del programa consiste en observar y forzar dichas variables. La Tabla de variables o VAT se utiliza para observar o forzar variables.

Antes de poder observar o forzar variables, tiene que crear una tabla de variables (VAT) e introducir las variables deseadas. Para crear una tabla de variables puede elegir una de las alternativas siguientes:

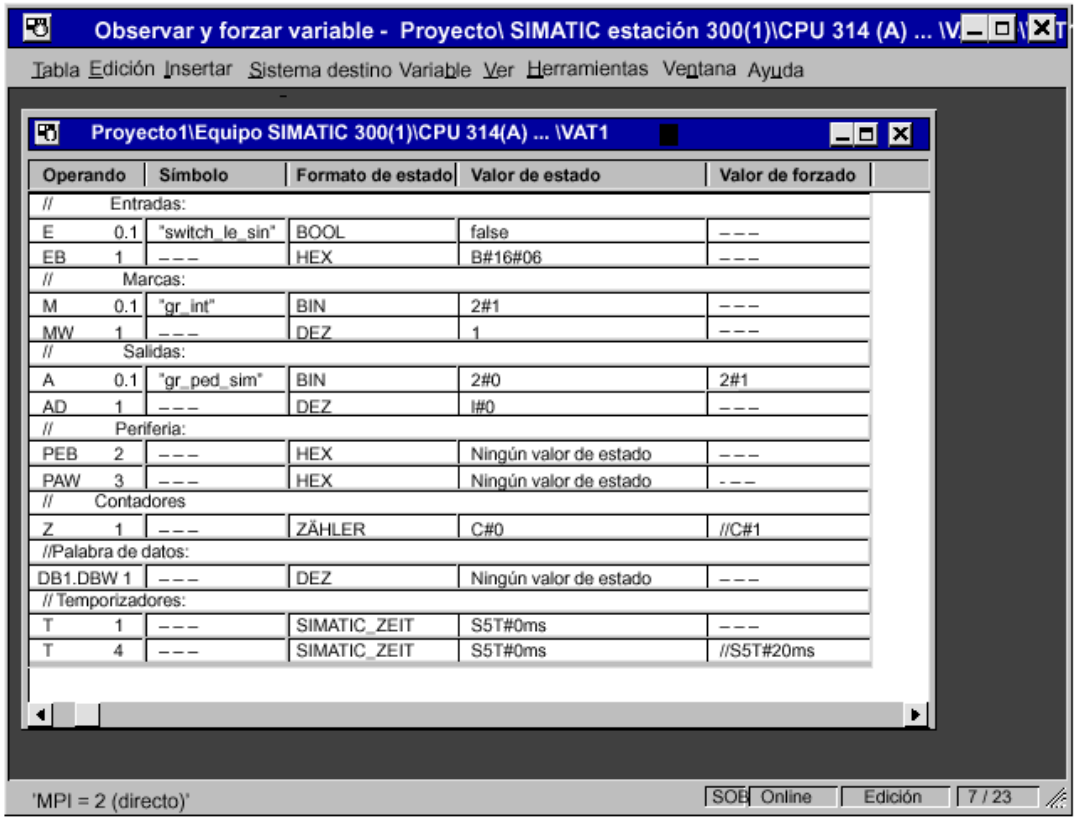
En el Administrador SIMATIC:

- Seleccione la carpeta "Bloques" y elija el comando de menú **Insertar > Bloque S7> Tabla de variables**. En el cuadro de diálogo siguiente puede adjudicar un nombre a la tabla. Para abrir la tabla de variables, haga doble clic en el objeto.
- Seleccione un enlace de la lista de las estaciones accesibles o un programa S7 de la vista online. Con el comando **Sistema de destino > Observar/forzar variable** es posible crear una tabla de variables sin nombre.

En "Observar/forzar variables":

- Eligiendo el comando **Tabla > Nueva** puede crear una tabla que no esté asociada todavía a ningún programa S7 . Para acceder a las tablas ya existentes, elija el comando **Tabla > Abrir**.

Ejemplo de una tabla de variables



Operando	Símbolo	Formato de estado	Valor de estado	Valor de forzado
// Entradas:				
E 0.1	"switch_le_sin"	BOOL	false	– – –
EB 1	– – –	HEX	B#16#06	– – –
// Marcas:				
M 0.1	"gr_int"	BIN	2#1	– – –
MW 1	– – –	DEZ	1	– – –
// Salidas:				
A 0.1	"gr_ped_sim"	BIN	2#0	2#1
AD 1	– – –	DEZ	I#0	– – –
// Periferia:				
PEB 2	– – –	HEX	Ningún valor de estado	– – –
PAW 3	– – –	HEX	Ningún valor de estado	– – –
// Contadores				
Z 1	– – –	ZÄHLER	C#0	//C#1
//Palabra de datos:				
DB1.DBW 1	– – –	DEZ	Ningún valor de estado	– – –
// Temporizadores:				
T 1	– – –	SIMATIC_ZEIT	S5T#0ms	– – –
T 4	– – –	SIMATIC_ZEIT	S5T#0ms	//S5T#20ms

En una tabla de variables es posible editar los campos ”Operando”, ”Símbolo”, ”Formato de estado” y ”Valor de forzado”.

- La variable a forzar se indica con su operando o como símbolo. Si se ha definido un símbolo correspondiente en la tabla de símbolos, la entrada en la columna ”Símbolo” u ”Operando” se complementa automáticamente.
- El formato de estado determina en qué formato se debe visualizar (en la columna que aparece a la derecha) el valor de estado determinado. Para elegir el formato, active el comando **Elegir formato de estado > ...** en el menú **Ver**, o haga varias veces clic en dicho campo de la tabla hasta que aparezca el formato deseado.

Al introducir las variables en la tabla, su sintaxis se comprueba antes de abandonar la línea. Las entradas erróneas se destacan en rojo. En este caso, un mensaje en la barra de estado indica que se ha cometido un error.

Las líneas de comentario se introducen con dos barras inclinadas ”//”. Si se activa el comando **Línea de comentario** en el menú **Edición** o el correspondiente botón en la barra de herramientas, es posible representar provisionalmente una línea de la tabla en forma de comentario.

Si en la columna ”Valor de forzado” se introduce un signo de comentario ”//” delante del valor de una variable que se desee forzar, dicho valor perderá su validez. Si las dos barras inclinadas se borran nuevamente, el valor volverá a ser válido y se podrá forzar.

Con el comando ***<Título de la columna>*** en el menú **Ver** es posible mostrar u ocultar las diversas columnas de la tabla. Se muestran sólo las columnas que lleven una marca de verificación en el menú **Ver**.

Para probar el programa con tablas de variables se dispone de las siguientes funciones:

- **Observar variables**
- Forzar variables
- Forzar variables de forma permanente

La función **Observar** permite observar las variables deseadas

La función **Forzar** permite asignar un valor a las variables deseadas y forzarlas directamente

---

Cuidado

Antes de ejecutar la función "Forzar", asegúrese de que no puedan presentarse situaciones peligrosas.

Esta función sólo se podrá ejecutar si el selector de modos de operación de la CPU se encuentra en "RUN-P" o "STOP".

Para observar variables puede elegir una de las alternativas siguientes:

- Active la función "Observar" con el comando **Observar** del menú **Variable**. los valores de las variables seleccionadas se visualizan en la tabla de variables. 
- Con el comando **Actualizar valores de estado** del menú **Variable**, se actualizan inmediatamente los valores de las variables seleccionadas una única vez. Los valores actuales de las variables seleccionadas se visualizan en la tabla de variables. 

Para forzar variables puede elegir una de las alternativas siguientes:

- Active la función "Forzar" con el comando **Forzar** del menú **Variable**, . 
- Con el comando **Activar valores de forzado** del menú **Variable** se actualizan inmediatamente los valores de las variables seleccionadas una única vey. 

Para poder realizar estas funciones tiene que existir una conexión online con la CPU, 

ésta tiene que estar en RUN-P y el programa se tiene que haber cargado ya.

Forzado permanente

Es posible asignar valores permanentes (fijos) a las variables de un programa de usuario, de manera que el programa que se ejecute en la CPU no los pueda cambiar ni sobreescribir. La condición es que la CPU (p.ej. S7-400) asista esta función. En caso contrario, los comandos de menú relacionados con el forzado permanente aparecerán atenuados.

Asignando valores permanentes a las variables, se pueden provocar determinados estados del programa y comprobar así las funciones programadas.

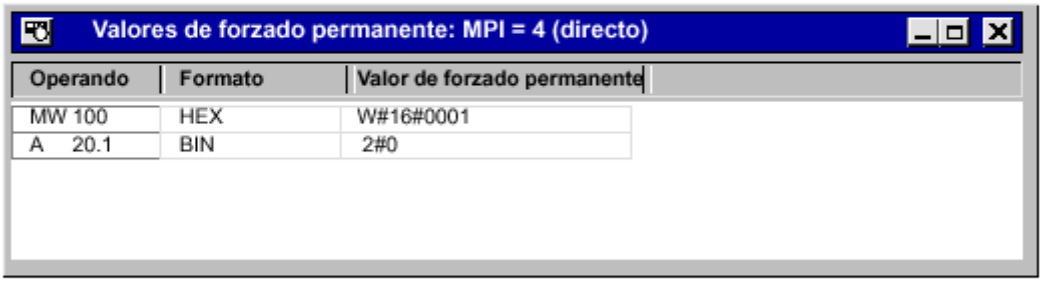
Cuidado

El forzado permanente es irrevocable, por lo que no se podrá activar el comando **Deshacer** del menú **Edición**.

Una petición de forzado permanente sólo se puede cancelar o finalizar con el comando **Anular forzado permanente** del menú **Variable**. Con sólo cerrar la ventana "Valores de forzado permanente" o salir de la aplicación "Observar y forzar variable" no se anula dicha petición.

Crear una petición de forzado permanente

1. Cree una nueva tabla de variables o abra una ya existente.
2. Establezca un enlace con la CPU deseada con el comando **Establecer enlace con** del menú **Sistema de destino**.
3. Abra la ventana donde se visualiza el estado actual de la CPU seleccionada con el comando **Mostrar valores de forzado permanente** del menú **Variable**.



La ventana estará vacía si no está activada ninguna petición de forzado permanente.

Si ya está activada una petición de forzado permanente, las variables y sus valores se representarán en negrita.

Para crear una petición de forzado permanente, proceda de la dorma siguiente:

1. Introduzca las variables a forzar en la columna "Operando" de la "Ventana de forzado permanente".
2. Introduzca los valores que desee asignar fijamente a las variables en la columna "Valor de forzado permanente", .
3. Inicie el forzado permanente con el comando **Forzado permanente** del menú **Variable**.

Diferencias entre el forzado permanente y el forzado "normal"

La tabla que se muestra a continuación muestra una comparativa entre el forzado permanente y el forzado "normal".

Característica / función	Forzado permanente con S7-400	Forzado permanente con S7-300	Forzado normal
Marcas (M)	•	–	•
Temporizadores y contadores (T, Z)	–	–	•
Bloques de datos (DB)	–	–	•
Entradas periféricas (PEB, PEW, PED)	•	–	–
Salidas de la periferia (PAB, PAW, PAD)	•	–	•
Entradas y salidas (E, A)	•	•	•
Ajustar las condiciones de disparo	siempre disparo inmediato	siempre disparo inmediato	único o cíclico
La función tiene efecto sólo en la variable que está en el área visible de la ventana activa.	tiene efecto en todos los valores de forzado perman.	tiene efecto en todos los valores de forzado perman.	•
El programa de usuario puede sobreescribir los valores de forzado normal o permanente.	–	•	•
El valor de forzado permanente se mantiene activo sin interrupciones.	•	•	–
Al salir de la aplicación, las variables conservan sus valores.	•	•	–
Al deshacer el enlace con la CPU, las variables conservan sus valores.	•	•	–
Se permiten errores de direccionamiento: p ej., EW1 Valor de forzado normal/permanente: 1 EW1 Valor de forzado normal/permanente: 0	–	–	el último tiene efecto

Forzar las salidas periféricas al estar la CPU en STOP

La función "Desbloquear salidas" desconecta el bloqueo de las salidas de periferia (PA). Ello permite forzar las salidas de periferia cuando la CPU se encuentra en el estado operativo STOP.

Proceda de la siguiente forma:

1. Elija el comando **Tabla > Abrir** para abrir la tabla de variables (VAT) que contiene las salidas periféricas a forzar, o active la ventana de la tabla de variables que corresponda.
2. Elija el comando **Sistema de destino > Establecer enlace con** para establecer una conexión con la CPU para poder forzar las salidas periféricas de la tabla de variables activa.
3. Con el comando **Sistema de destino > Estado operativo** abra el cuadro de diálogo Estado operativo y conmute la CPU al estado operativo STOP.
4. En la columna "Valor de forzado", introduzca los valores correspondientes a las salidas periféricas que desee forzar.

Ejemplos:	PAB	7	Valor de forzado:	2#00010011
	PAW	2		W#16#0027
	PAD	4		DW#16#00000001

5. Con el comando de menú **Variable > Desbloquear salidas** active el modo "Desbloquear salidas".
6. Con el comando **Variable > Activar valores de forzado** fuerce las salidas de periferia.
7. "Desbloquear salidas" permanece activado hasta que seleccione de nuevo **Variable > Desbloquear salidas**, desactivando así este modo.
8. Para predefinir nuevos valores, comience de nuevo en el punto 4.

Nota

- El comando **Variable > Desbloquear salidas** sólo tiene sentido si la CPU está en modo STOP.
- El modo "Desbloquear salidas" se finaliza al presentarse uno de los siguientes eventos:
- al cambiar el estado operativo de la CPU (se visualiza un mensaje)
- al elegir nuevamente el comando de menú Variable > Desbloquear salidas o al pulsar la tecla ESC (sin mensaje).

Si pulsa la tecla ESC mientras se está ejecutando la función "Desbloquear salidas", la función se cancelará sin consulta previa.

Ejemplos de introducción en tablas de variables

Ejemplo de introducción de operandos en tablas de variables

Operando permitido:	Tipo de datos:	Ejemplo (nemotécnica SIMATIC):
Entrada | Salida | Marca	BOOL	E 1.0 | A 1.7 | M 10.1
Entrada | Salida | Marca	BYTE	EB 1 | AB 10 | MB 100
Entrada | Salida | Marca	WORD	EW 1 | AW 10 | MW 100
Entrada | Salida | Marca	DWORD	ED 1 | AD 10 | MD 100
Periferia (Entrada | Salida)	BYTE	PEB 0 | PAB 1
Periferia (Entrada | Salida)	WORD	PEW 0 | PAW 1
Periferia (Entrada | Salida)	DWORD	PED 0 | PAD 1
Temporizadores	TIMER	T 1
Contadores	COUNTER	Z 1
Bloque de datos	BOOL	DB1.DBX 1.0
Bloque de datos	BYTE	DB1.DBB 1
Bloque de datos	WORD	DB1.DBW 1
Bloque de datos	DWORD	DB1.DBD 1

Ejemplo de introducción de un área de operandos conexos

Abra una tabla de variables y visualice en el cuadro de diálogo "Insertar grupo" con el comando de menú **Insertar > Grupo**.

Por lo que respecta a los datos del cuadro de diálogo, se insertan en la tabla de variables las siguientes líneas para las marcas:

- Desde operando: M 3.0
- Cantidad: 10
- Formato de estado: BIN

Operando	Formato de estado
M 3.0	BIN
M 3.1	BIN
M 3.2	BIN
M 3.3	BIN
M 3.4	BIN
M 3.5	BIN
M 3.6	BIN
M 3.7	BIN
M 4.0	BIN
M 4.1	BIN

En este ejemplo puede observar cómo varía la denominación en la columna "Operando" después de la octava entrada.

Ejemplos de introducción de valores de forzado normal y de forzado permanente

Operandos de bit

Operandos de bit posibles	Valores de forzado normal y forzado permanente permitidos
E1.0	true
M1.7	false
A10.7	0
DB1.DBX1.1	1
E1.1	2#0
M1.6	2#1

Operandos de byte

Operandos de byte posibles	Valores de forzado / de forzado permanente permitidos
EB 1	2#00110011
MB 12	b#16#1F
MB 14	1F
AB 10	'a'
DB1.DBB 1	10
PAB 2	-12

Operandos de palabra

Operandos de palabra posibles	Valores de forzado normal y forzado permanente permitidos
EW 1	2#0011001100110011
MW 12	w#16#ABCD
MW 14	ABCD
AW 10	b#(12,34)
DB1.DBW 1	'ab'
PAW 2	-12345
MW 3	12345
MW 5	s5t#12s340ms
MW 7	0.3s ó 0,3s
MW 9	c#123
MW 11	d#1990-12-31

Operandos de palabra doble

Operandos de palabra doble posibles	Valores de forzado / de forzado permanente permitidos
ED 1	2#00110011001100110011001100110011
MD 0	1.23e4
MD 4	1.2
AD 10	dw#16#abcdef10
AD 12	ABCDEF10
DB1.DBD 1	b#(12,34,56,78)
PAD 2	'abcd'
MD 8	l# -12
MD 12	l#12
MD 16	-123456789
MD 20	123456789
MD 24	t#12s345ms
MD 28	tod#1:2:34.567
MD 32	p#e0.0

Temporizador

Operandos posibles del tipo temporizador	Valores de forzado normal y forzado permanente permitidos	Explicación
T 1	0	Conversión en milisegundos (ms)
T 12	20	Conversión en milisegundos
T 14	12345	Conversión en milisegundos
T 16	s5t#12s340ms	
T 18	1.3	Conversión en 1s 300 ms
T 20	1.3s	Conversión en 1s 300 ms

El forzado de un temporizador sólo afecta su valor, mas no su estado. Por tanto, el temporizador T1 se puede forzar al valor 0, pero el resultado lógico en U T1 no se modifica.

Las literales "s5t" y "s5time" se pueden escribir tanto con mayúsculas como con minúsculas.

Contadores

Operandos posibles del tipo contador	Valores de forzado / de forzado permanente permitidos
Z 1	0
Z 14	20
Z 16	c#123

El forzado de un contador sólo afecta su valor, mas no su estado. Por tanto, el contador T1 se puede forzar al valor 0, pero el resultado lógico en U Z1 no se modifica.

3.9 La librería en S7 (Funciones estandar)

El software estándar STEP 7 contiene las librerías estándar

Las librerías sirven para depositar componentes reutilizables de programas para SIMATIC S7/M7. Los componentes de programas pueden copiarse de proyectos existentes a una librería, o bien, crearse directamente en la librería, independientemente de los proyectos.

La programación se puede simplificar en gran medida depositando en un programa S7 o en una librería los bloques que se deseen utilizar una y otra vez. De allí se podrán copiar siempre al programa de usuario en cuestión.

Las librerías estándar contienen los componentes siguientes:

- **builtin/Built In:** funciones de sistema (SFC) y bloques de función de sistema (SFB)
- **fblib1/FB Lib 1:** bloques para la conversión de programas STEP 5
- **fblib2/FB Lib 2:** funciones estándar utilizables en general
- **iec/IEC:** bloques para funciones IEC, tales como para editar indicaciones de fecha y hora, para operaciones de comparación, para el tratamiento de cadenas y para seleccionar el máximo y el mínimo
- **stdobs/Std OBs:** bloques de organización estándar (OB)

La librería estándar para la versión 3 contiene además los componentes siguientes:

- **PID Control:** bloques de función (FBs) para el PID Control
- **Net DP:** FCs para periféricos descentralizados y para enlaces FDL

Las funciones estandar de S5 se convierten automáticamente en funciones S7 que ofrecen la misma funcionalidad. Estas funciones se suelen sustituir en S7, por secuencias de instrucciones simples, lo que ahorra espacio de memoria y tiempo de ciclo.
Las funciones estandar que están incluidas en la librería S7 “Stdlibs” se encuentran en el contenedor de programas “Fblib1” y son:.

Aritmética en coma flotante

STEP 5	STEP 7		STEP 5	STEP 7	
Nombre del FB	Número	Nombre	Nombre del FB	Número	Nombre
GP:FPGP	FC 61	GP_FPGP	GP:MUL	FC 65	GP_MUL
GP:GPFP	FC 62	GP_GPFP	GP:DIV	FC 66	GP_DIV
GP:ADD	FC 63	GP_ADD	GP:VGL	FC 67	GP_VGL
GP:SUB	FC 64	GP_SUB	RAD:GP	FC 68	RAD_GP

Funciones de señales

MLD:TG	FC 69	MLD_TG	MLD:EZ	FC 75	MLD_EZ
MELD:TGZ	FC 70	MELD_TGZ	MLD:ED	FC 76	MLD_ED
MLD:EZW	FC 71	MLD_EZW	MLD:EZWK	FC 77	MLD_EZWK
MLD:EDW	FC 72	MLD_EDW	MLD:EDWK	FC 78	MLD_EDWK
MLD:SAMW	FC 73	MLD_SAMW	MLD:EZK	FC 79	MLD_EZK
MLD:SAM	FC 74	MLD_SAM	MLD:EDK	FC 80	MLD_EDK

Funciones integradas

STEP 5	STEP 7	
Nombre del FB	Número	Nombre
COD:B4	FC 81	COD_B4
COD:16	FC 82	COD_16
MUL:16	FC 83	MUL_16
DIV:16	FC 84	DIV_16

Funciones básicas

STEP 5	STEP 7		STEP 5	STEP 7	
Nombre del FB	**Número**	**Nombre**	**Nombre FB**	**Número**	**Nombre**
ADD:32	FC 85	ADD_32	REG:LIFO	FC 93	REG_LIFO
SUB:32	FC 86	SUB_32	DB:COPY	FC 94	DB_COPY
MUL:32	FC 87	MUL_32	DB:COPY	FC 95	DB_COPY
DIV:32	FC 88	DIV_32	RETTEN	FC 96	RETTEN
RAD:16	FC 89	RAD_16	LADEN	FC 97	LADEN
REG:SCHB	FC 90	REG_SCHB	COD:B8	FC 98	COD_B8
REG:SCHW	FC 91	REG_SCHW	COD:32	FC 99	COD_32
REG:FIFO	FC 92	REG_FIFO			

Funciones analógicas

STEP 5	STEP 7		STEP 5	STEP 7	
Nombre del FB	**Número**	**Nombre**	**Nombre del FB**	**Número**	**Nombre**
AE:460	FC 100	AE_460_1	AE:466	FC 106	AE_466_1
AE:460	FC 101	AE_460_2	AE:466	FC 107	AE_466_2
AE:463	FC 102	AE_463_1	RLG:AA	FC 108	RLG_AA1
AE:463	FC 103	AE_463_2	RLG:AA	FC 109	RLG_AA2
AE:464	FC 104	AE_464_1	PER:ET	FC 110	PER_ET1
AE:464	FC 105	AE_464_2	PER:ET	FC 111	PER_ET2

Funciones matemáticas

STEP 5	STEP 7		STEP 5	STEP 7	
Nombre del FB	**Número**	**Nombre**	**Nombre del FB**	**Número**	**Nombre**
SINUS	FC 112	SINUS	ARCCOT	FC 119	ARCCOT
COSINUS	FC 113	COSINUS	LN X	FC 120	LN_X
TANGENS	FC 114	TANGENS	LG X	FC 121	LG_X
COTANG	FC 115	COTANG	B LOG X	FC 122	B_LOG_X
ARCSIN	FC 116	ARCSIN	E^X	FC 123	E_H_N
ARCCOS	FC 117	ARCCOS	ZEHN^X	FC 124	ZEHN_H_N
ARCTAN	FC 118	ARCTAN	A2^A1	FC 125	A2_H_A1