SIEMENS

SIMATIC

S7-SCL V5.3 para S7-300/400

Manual

Edición 01/2005
A5E00324652-01

Consignas de seguridad

Este manual contiene las informaciones necesarias para la seguridad personal así como para la prevención de daños materiales. Las informaciones para su seguridad personal están resaltadas con un triángulo de advertencia; las informaciones para evitar únicamente daños materiales no llevan dicho triángulo. De acuerdo al grado de peligro las consignas se representan, de mayor a menor peligro, como sigue**:**

Peligro
Significa que, si no se adoptan las medidas preventivas adecuadas se producirá la muerte, o bien lesiones corporales graves.

Advertencia
Significa que, si no se adoptan las medidas preventivas adecuadas puede producirse la muerte o bien lesiones corporales graves.

Precaución
Con triángulo de advertencia significa que si no se adoptan las medidas preventivas adecuadas, pueden producirse lesiones corporales.

Precaución
Sin triángulo de advertencia significa que si no se adoptan las medidas preventivas adecuadas, pueden producirse daños materiales**.**

Atención
Significa que puede producirse un resultado o estado no deseado si no se respeta la consigna de seguridad correspondiente.

Si se dan varios niveles de peligro se usa siempre la consigna de seguridad más estricta en cada caso. Si en una consigna de seguridad con triángulo de advertencia se alarma de posibles daños personales, la misma consigna puede contener también una advertencia sobre posibles daños materiales.

Personal cualificado

El equipo/sistema correspondiente sólo deberá instalarse y operarse respetando lo especificado en este documento. Sólo está autorizado a intervenir en este equipo el personal cualificado. En el sentido del manual se trata de personas que disponen de los conocimientos técnicos necesarios para poner en funcionamiento, conectar a tierra y marcar los aparatos, sistemas y circuitos de acuerdo con las normas estándar de seguridad.

Uso conforme

Considere lo siguiente:

Advertencia

El equipo o los componentes del sistema sólo se podrán utilizar para los casos de aplicación previstos en el catálogo y en la descripción técnica, y sólo asociado a los equipos y componentes de Siemens y de tercera que han sido recomendados y homologados por Siemens.

El funcionamiento correcto y seguro del producto presupone un transporte, un almacenamiento, una instalación y un montaje conforme a las prácticas de la buena ingeniería, así como un manejo y un mantenimiento rigurosos.

Marcas registradas

Todos los nombres marcados con ® son marcas registradas de Siemens AG. Los restantes nombres y designaciones contenidos en el presente documento pueden ser marcas registradas cuya utilización por terceros para sus propios fines puede violar los derechos de sus titulares.

Exención de responsabilidad

Hemos comprobado la concordancia del contenido de esta publicación con el hardware y el software descritos. Sin embargo, como es imposible excluir desviaciones, no podemos hacernos responsable de la plena concordancia. El contenido de esta publicación se revisa periódicamente; si es necesario, las posibles correcciones se incluyen en la siguiente edición**.**

Siemens AG
Automation and Drives
Postfach 4848, 90327 Nürnberg, Alemania

Prólogo

Finalidad del manual

Este manual proporciona un completo resumen acerca de la programación con S7-SCL. Sirve de ayuda durante la instalación y la puesta en marcha del software. En él se explican los pasos que hay que seguir para crear el programa, la estructura de los programas de usuario y los elementos de lenguaje individuales.

Este manual está dirigido a programadores de programas S7 y al personale encargado de la configuración, la puesta en marcha y el mantenimiento de sistemas de automatización.

Le recomendamos familiarizarse con el ejemplo del capítulo 2 "Diseño de un programa S7-SCL". Este ejemplo ofrece una sencilla introducción a la programación con S7-SCL.

Conocimientos básicos necesarios

Para comprender el contenido de este manual se requieren conocimientos generales en el ámbito de la técnica de automatización.

Además son necesarios conocimientos sobre el uso de ordenadores o medios de trabajo similares a un PC (p. ej. unidades de programación) en el sistema operativo MS Windows 2000 Professional y MS Windows XP Professional. Dado que S7-SCL situa encima del software básico STEP 7, el usuario ya debe tener conocimientos en relación con el software básico, descrito en el manual "Programación con STEP 7 V5.3".

Ámbito de validez del manual

El manual es válido para el paquete de software S7-SCL V5.3 a partir del Service Pack 1.

Los paquetes de documentación de S7-SCL y del software básico STEP 7.

En la siguiente tabla aparece resumida la documentación de STEP 7 y S7-SCL:

Manuales	**Finalidad**	**Número de referencia**
Información S7-SCL básica y de referencia con	Información básica y de	El manual no se puede adquirir por separado. Éste se encuentra disponible en el CD del producto, en la Manual Collection y en Internet.
Información básica de STEP 7 compuesta por: • STEP 7 V5.3: Introducción y ejercicios prácticos, Getting Started • Programar con STEP 7 V5.3 • Configurar el hardware y la comunicación con STEP 7 V5.3 • De S5 a S7, Guía para facilitar la transición	Nociones básicas para el personal técnico. Describe cómo realizar soluciones de control con el software STEP 7 para los sistemas S7-300/400.	6ES7810-4CA07-8DW0
Información de referencia STEP 7 con • Manuales KOP/FUP/AWL para S7-300/400 • Funciones estándar y de sistema para S7-300/400	Información de consulta que describe los lenguajes de programación KOP, FUP y AWL y las funciones estándar y de sistema que integran STEP 7.	6ES7810-4CA07-8DW1

Ayudas en pantalla	**Finalidad**	**Número de referencia**
Ayuda acerca de S7-SCL	Información básica y de	Componente del
Ayuda acerca de STEP 7	Información básica sobre cómo	Componente del
Ayudas de referencia acerca de AWL/KOP/FUP Ayuda de referencia acerca de SFB/SFC Ayuda de referencia acerca de bloques de organización Ayuda de referencia acerca de funciones IEC Ayuda de referencia acerca de atributos de sistema	Información de referencia contextual	Componente del paquete de software STEP 7

Ayuda en pantalla

La ayuda en pantalla le ofrece información en el lugar donde la necesite. Así podrá obtener información de forma rápida y específica sin tener que buscar en los manuales. En la ayuda en pantalla encontrará:

- **Temas de Ayuda**: ofrece diferentes accesos para visualizar información de ayuda.
- **Ayuda contextual** (tecla F1): muestra información sobre el objeto seleccionado o el cuadro de diálogo y/o ventana activos.
- **Introducción**: proporciona un breve resumen sobre el uso, las características esenciales y la funcionalidad de la aplicación.
- **Primeros pasos**: describe los primeros pasos que debe realizar para obtener buenos resultados.
- **Uso de la Ayuda**: explica las posibilidades de las que dispone el usuario para encontrar determinada información en la Ayuda.
- **Acerca de**: proporciona información sobre la versión actual de la aplicación.

Asistencia adicional

Si tiene preguntas relacionadas con el uso de los productos descritos en el manual a las que no encuentre respuesta, diríjase a la sucursal o al representante más próximo de Siemens, en donde le pondrán en contacto con el especialista.

Encontrará a su persona de contacto en la página de Internet:

http://www.siemens.com/automation/partner

Encontrará una guía sobre el conjunto de la información técnica correspondiente a los distintos productos y sistemas SIMATIC en la página de Internet:

http://www.siemens.com/simatic-tech-doku-portal

Encontrará el catálogo y el sistema de pedidos on-line en:

http://mall.automation.siemens.com/

Centro de formación SIMATIC

Para ofrecer a nuestros clientes un fácil aprendizaje de los sistemas de automatización SIMATIC S7, les ofrecemos distintos cursillos de formación. Diríjase a su centro de formación regional o a la central en D 90327 Nuernberg.

Teléfono: +49 (911) 895-3200.

Internet: http://www.sitrain.com

Technical Support

Podrá acceder al Technical Support de todos los productos de A&D

- a través del formulario de Internet para el Support Request
  http://www.siemens.com/automation/support-request
- Teléfono: + 49 180 5050 222
- Fax: + 49 180 5050 223

Encontrará más información sobre nuestro Technical Support en la página de Internet
http://www.siemens.com/automation/service

Service & Support en Internet

Además de nuestra documentación, en Internet le ponemos a su disposición todo nuestro know-how.

http://www.siemens.com/automation/service&support

En esta página encontrará:

- "Newsletter" que le mantendrán siempre al día ofreciéndole informaciones de última hora,
- La rúbrica "Servicios online" con un buscador que le permitirá acceder a la información que necesita,
- El "Foro" en el que podrá intercambiar sus experiencias con cientos de expertos en todo el mundo,
- El especialista o experto de Automation & Drives de su región,
- Bajo la rúbrica "Servicios" encontrará información sobre el servicio técnico más próximo, sobre reparaciones, repuestos etc.

Índice

1 Presentación del producto

1.1 Campo de aplicación de S7-SCL

S7-SCL (Structured Control Language) es un lenguaje de alto nivel orientado a PASCAL para la programación de autómatas programables con SIMATIC S7.

Certificado PLCopen

S7-SCL es un lenguaje textual de alto nivel ST (Structured Text) según lo establecido en la norma IEC 61131-3 ST y ha sido preparado para la certificación para el Reusability Level.

Campo de aplicación

S7-SCL está optimizado para la programación de autómatas programables y contiene tanto elementos del lenguaje de programación PASCAL como elementos típicos de PLCs, como entradas y salidas, temporizadores y contadores.

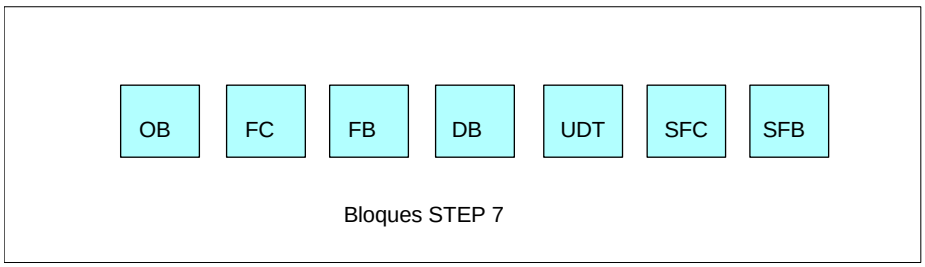
S7-SCL resulta especialmente apropiado para resolver las siguientes tareas:

- Programación de algoritmos complejos
- Programación de funciones matemáticas
- Gestión de datos y recetas
- Optimización del proceso

1.2 Funcionamiento de S7-SCL

Integración en STEP 7

S7-SCL soporta el concepto de bloques de STEP 7.



Con S7-SCL se pueden crear los siguientes bloques de STEP 7:

- OB
- FC
- FB
- DB
- UDT

En un programa S7 también se pueden combinar bloques de S7-SCL con bloques de otros lenguajes de programación de STEP 7. Estos bloques se pueden llamar unos a otros. Los bloques de S7-SCL también se pueden almacenar en librerías y utilizarlos desde allí en otros lenguajes.

Como los programas de S7-SCL se pueden programar como fuentes ASCII, son fáciles de importar y exportar.

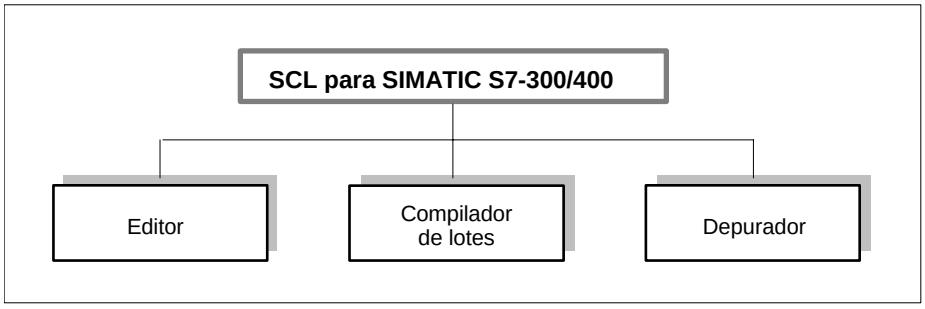
Los bloques de S7-SCL se pueden recompilar al lenguaje de programación AWL de STEP 7 (lista de instrucciones). Sin embargo, tenga en cuenta que una vez guardados en AWL ya no se pueden editar en S7-SCL.

Entorno de desarrollo

S7-SCL ofrece en la práctica un potente entorno de desarrollo adaptado tanto a las características específicas de S7-SCL como a las de STEP 7. El entorno de desarrollo está formado por los siguientes componentes:

- Un **editor** para elaborar programas compuestos de funciones (FC), bloques de función (FB), bloques de organización (OB), bloques de datos (DB) y tipos de datos de usuario (UDT). El programador cuenta con la ayuda de potentes funciones.
- Un **compilador por lotes** para compilar a código máquina MC7 el programa previamente editado. El código MC7 generado puede ejecutarse en todas las CPU del sistema de automatización S7-300/400 a partir de la CPU 314.
- Un **depurador** que permite buscar errores lógicos de programación en una compilación sin errores. La búsqueda de errores se realiza en lenguaje fuente.

La figura siguiente ofrece una visión de conjunto de los componentes del entorno de desarrollo:



1.3 ¿Qué funciones ofrece S7-SCL?

S7-SCL ofrece todas las ventajas de un lenguaje de programación de alto nivel. Además, S7-SCL dispone de características que han sido diseñadas especialmente para facilitar la programación estructurada:

Librerías de bloques

Con S7-SCL se suministran bloques ya preparados en librerías, p. ej.:

- Funciones de sistema
- Funciones IEC
- Funciones de conversión

Un cuadro de diálogo le ayudará a navegar por la librería. Al seleccionar un bloque se copia automáticamente la plantilla de parámetros de la función en el archivo que se está editando. El usuario no tiene más que introducir los parámetros deseados.

Plantillas de programa

El editor de S7-SCL ofrece diversas plantillas que se pueden insertar y que el usuario tan solo tiene que rellenar:

- Plantillas para bloques (p. ej. bloques de función, bloques de datos) y sus llamadas
- Plantillas para comentarios de bloques, parámetros de bloques y constantes
- Plantillas para estructuras de control (IF, CASE, FOR, WHILE, REPEAT)

Elementos de lenguaje para la programación de alto nivel

Una programación sencilla, rápida y segura gracias a la utilización de construcciones de elementos muy potentes, p. ej.:

- Bucles
- Ramificaciones condicionales o alternativas (IF ... THEN ... ELSE)
- Saltos

Fácil comprensión del programa

Las siguientes prestaciones aumentan la legibilidad del programa:

- Programación completamente simbólica
- Comentarios
- Tipos de datos básicos y tipos de datos definidos por el usuario
- Visualización de referencias cruzadas
- Formateo automático de la entrada realizada mediante sangrado
- Coloración de los elementos del lenguaje orientada a la sintaxis

Depurador a nivel de lenguaje de alto nivel

El depurador o debugger permite probar el programa fácilmente a nivel de lenguaje de alto nivel. Ofrece las siguientes funciones:

- Observación continua de la ejecución del programa
- Observación paso a paso con ayuda de puntos de parada que se pueden posicionar individualmente.
- Funcionalidad Step-in (posibilidad de saltar a los bloques llamados durante el test)

1.4 Novedades de la versión V5.3 SP1

Ampliación del lenguaje

S7-SCL V5.3 SP1 ha sido ampliado con medios del lenguaje que se definen en la norma IEC 61131-3 :

- Funciones para procesar valores numéricos como funciones internas de S7-SCL (SEL, MAX, MIN, LIMIT, MUX)
- Soporte de la representación BCD de números entereso mediante funciones de conversión (BCD_TO_INT, INT_TO_BCD, etc.)
- Operador de asignación => para parámetros de salida de funciones
- Inicialización de array con paréntesis
- Nuevas funciones de conversión (BYTE_TO_INT, INT_TO_BYTE, etc.)

Ajustes del compilador en la fuente

Los ajustes de compilador se pueden almacenar en fuentes S7-SCL o en archivos de control de compilación. De este modo es posible almacenar las propiedades de la compilación de una fuente determinada.

Funciones de test ampliadas

- Las incoherencias en bloques y los conflictos de fecha y hora de bloques S7-SCL se pueden determinar y corregir con la función de test "Comprobar coherencia de bloques" de STEP 7. Esta función de test está disponible a partir de la versión de 5.3 SP2 de STEP 7.
- La función de test "Observar" se puede utilizar de manera más eficaz mediante definición del entorno de llamada.
- El área de observación para la función de test "Observar" se puede delimitar marcando una sección de la fuente.

Imprimir en color

Las fuentes de S7-SCL se pueden imprimir a partir de ahora en color.

Función de búsqueda ampliada

S7-SCL permite ahora buscar desde la posición del cursor hacia arriba, así como buscar dentro de un área seleccionada.

Posicionar marcadores en el texto fuente

Gracias a los marcadores ahora disponibles podrá navegar rápidamente por la fuente.

Creación de bloques S7-SCL con juegos de caracteres de otros idiomas

Las fuentes S7-SCL pueden contenter textos de juegos de caracteres de otros idiomas. De este modo es posible crear bloques para el mercado internacional, en los que las partes más importante que puede ver el usuario aparezcan en los juegos de caracteres de su idioma (p. ej. nombres simbólicos de bloques, atributos y comentarios).

Para más información sobre los juegos de caracteres de otros idiomas, consulte el archivo Léame.

2 Instalación

2.1 Automation License Manager

2.1.1 Autorización de uso con el Automation License Manager

Automation License Manager

Para la utilización del software de programación se requiere una clave de licencia (autorización de utilización) específica para del producto, cuya instalación se ejecuta a partir de la versión 5.3 de S7-SCL con el Automation License Manager.

El Automation License Manager es un producto de sofware de Siemens AG. Se utiliza en todos los sistemas para el procesamiento de claves de licencia (representantes técnicos de licencias).

El Automation License Manager puede encontrarse:

- En el soporte de instalación de STEP 7
- En las páginas de Internet de A&D Customer Support de Siemens AG como WebDownLoad.

En el Automation License Manager se ha integrado una ayuda en pantalla, a la que puede acceder contextualmente tras la instalación pulsando la tecla F1 o a través del comando de menú **Ayuda > Ayuda sobre License Manager**. En esta ayuda encontrará la información detallada sobre la funcionalidad y el uso del Automation License Managers.

Licencias

Para la utilización de paquetes de software de STEP 7 protegidos con licencia se requieren licencias. Una licencia se otorga a modo de derecho a la utilización de productos. Los representantes de este derecho son:

- El CoL (**C**ertificate **o**f **L**icense) y
- La clave de licencia.

Certificate of License (CoL)

El "Certificate of License" (certificado de licencia) suministrado con el producto es la prueba jurídica el derecho de utilización. El producto correspondiente sólo puede utilizarlo el propietario del CoL o personas autorizadas.

Claves de licencia

La clave de licencia es el representante técnico de una licencia (sello de licencia electrónico).

Para cada software protegido por licencia SIEMENS AG otorga una clave de licencia. Sólo si al abrir el software se detectar una clave de liencia válida podrá utilizarse el software correspondiente de acuerdo con las condiciones de licencia y de utilización relacionadas con la licencia en cuestión.

Notas

Puede utilizar el software básico sin clave de licencia para familiarizarse con la interfaz de usuario y con sus funciones.

El uso ilimitado bajo la consideración de las determinaciones jurídicas de la licencia sólo es posible con una clave de licencia instalada.

Si no ha instalado la clave de licencia le aparecerá regularmente un aviso que le solicitará la instalación de la clave.

Las claves de licencia pueden guardarse y transferirse a soportes individuales de la siguiente manera:

- En disquetes de claves de licencia,
- En discos duros locales y
- En memorias de disco duro de ordenadores y de redes

Encontrará información detallada sobre el uso de claves de licencia en la ayuda en pantalla sobre el Automation License Manager.

Tipos de licencia

Para productos de software de Siemens AG se distingue entre los siguientes tipos de licencia orientadas a la aplicación. El comportamiento del software depende de las claves de licencia de los diferentes tipos de licencia. El tipo de utilización depende a su vez del correspondiente Certificate of License.

Tipo de licencia	**Descripción**
Single License	La utilización del software es ilimitada en tiempo y es posible en cualquier ordenador.
Floating License	Derecho de utilización temporalmente ilimitada sujeta a la utilización en red ("remote"Nutzung) de un software.
Trial License	El uso del software está limitado a:· • Una validez máxima de 14 días, • Un número determinado de días a partir de la primera utilización, • El uso para pruebas y para validación (exclusión de responsabilidades).
Rental License	El uso del software está limitado a : • una validez de 50 días como máximo • un determinado número de horas al ser utilizado
Upgrade License	Para una actualización pueden ser necesarios determinados requisitos en cuanto al estado del sistema: • Con una licencia de actualización puede convertirse la licencia de una antigua versión x a una versión >x+.... • Una actualización puede ser necesaria p.ej. por la ampliación del equipamiento.

2.1.2 Instalar el Automation Licence Manager

El Automation License Manager se instala con un programa de instalación. El software de instalación del Automation License Manager puede encontrarse en el CD de producto STEP 7.

Puede instalar el Automation License Manager junto con S7-SCL o hacerlo posteriormente.

Notas

Consulte información detallada sobre el procedimiento de instalación del Automation License Manager en el archivo Léame.wri actual del Automation License Manager.

En la ayuda en pantalla sobre Automation License Manager encontrará toda la información necesaria sobre la funcionalidad y el uso de claves de licencia.

Instalar claves de licencia posteriormente

Si abre el software S7-SCL sin disponer de una clave de licencia, aparecerá el aviso correspondiente.

Notas

Puede utilizar el software sin clave de licencia para familiarizarse con la interfaz de usuario y con sus funciones.

El uso ilimitado bajo la consideración de las determinaciones jurídicas de la licencia sólo es posible con una clave de licencia instalada.

Si no ha instalado la clave de licencia le aparecerá regularmente un aviso que le solicitará la instalación de la clave.

Para instalar la clave de licencia posteriormente dispone de las siguientes posibilidades:

- Instalar las claves de licencia desde diquets
- Instalar las claves de licencia a través de WebDownLoad (es necesaria una solicitud previa)
- Utilizando las claves de Floating License disponibles en la red.

Encontrará información detallada sobre el procedimiento en la ayuda en pantalla sobre el Automation License Manager, a la que puede acceder tras la instalación pulsando la tecla F1 o con el comando de menú **Ayuda > Ayuda sobre el License Manager**.

Notas

Las claves de licencia sólo funcionarán en Windows 2000/XP si se encuentran en un soporte de disco duro con acceso de escritura.

Las Floating Licenses también puede utilizarse a través de una red, es decir en modo "remote".

2.1.3 Reglas para el uso de claves de licencia

Cuidado

Tenga en cuenta las indicaciones sobre el uso de claves de licencia especificadas en la ayuda en pantalla así como en el archivo Léame.wri referente al Automation License Manager. Si no las observa es posible que pierda las claves de licencia de forma irrecuperable.

La ayuda en pantalla referente al Automation License Manager se puede abrir contextualmente pulsando la tecla F1 o con el comando de menú **Ayuda > Ayuda sobre el Automation License Manager**.

En esta ayuda encontrará toda la información necesaria sobre la funcionalidad y el uso de las claves de licencia.

2.2 Instalación

2.2.1 Requisitos de instalación

Requisitos de sistema

El paquete opcional S7-SCL V5.3 SP1 se ejecuta en una PG o en un PC con una instalación del paquete básico STEP 7 V5.3 o superior.

Para más información sobre los requisitos que debe cumplir el sistema operativo, consulte el archivo Leame.wri.

Requisitos de hardware

Para S7-SCL rigen los mismos requisitos que para el paquete básico STEP 7. La capacidad de memoria que requiere el paquete opcional S7-SCL V5.3 SP1 se indica en archivo Léame.wri.

2.2.2 Instalación de S7-SCL

Iniciar el programa de instalación

S7-SCL incluye un programa de instalación (setup) que instala el software automáticamente. En la pantalla aparecerán mensajes que le solicitarán que introduzca datos y le guiarán paso a paso durante todo el proceso de instalación.

Procedimiento:

1. Inicie en Windows el diálogo de instalación de software haciendo doble clic en el icono "Agregar o quitar programas" del "Panel de control".
2. Haga clic en "Instalar".
3. Introduzca el soporte de datos y haga clic en "Continuar". Windows buscará el programa de instalación "Setup.exe".
4. Siga paso a paso las instrucciones que le indique el programa de instalación.

Autorización de claves de licencia

Durante la instalación se comprueba si en el disco duro existe la correspondiente clave de licencia. Si ésta no se detecta, aparece un mensaje indicando que el software sólo se puede utilizar con la debida clave de licencia. Si lo desea, puede transferir la clave de licencia inmediatamente o continuar con la instalación de STEP 7 y transferirla posteriormente. En el primer caso, introduzca el disquete con la clave de licencia suministado en la correspondiente unidad cuando se le solicite.

3 Diseñar un programa S7-SCL

3.1 Bienvenido al ejemplo de iniciación "Adquisición de valores medidos"

¿Qué aprenderé?

El ejemplo de iniciación ilustra cómo configurar S7-SCL de forma eficaz. Al principio, las preguntas más frecuentes son p. ej.:

- ¿Cómo se trabaja con S7-SCL al diseñar programas?
- ¿Qué medios ofrece el lenguaje S7-SCL para solucionar la tarea planteada?
- ¿De qué funciones de test dispongo?

En este capítulo se dará respuesta a éstas y otras preguntas.

Medios utilizados en el lenguaje S7-SCL

En el ejemplo aparecen, entre otros, los siguientes elementos de lenguaje S7-SCL:

- Estructura y utilización de los diferentes tipos de bloques en S7-SCL
- Llamada del bloque con transferencia y evaluación de parámetros
- Distintos formatos de entrada y salida
- Programación con tipos de datos simples y arrays
- Inicialización de variables
- Estructuras con ramificaciones y bucles

Hardware necesario

El programa de ejemplo se puede ejecutar en un SIMATIC S7-300 o SIMATIC S7-400 con la siguiente periferia:

- un módulo de entrada de 16 canales
- un módulo de salida de 16 canales

Funciones de test disponibles

El programa está estructurado de manera que el usuario pueda testearlo rápidamente mediante los pulsadores a la entrada y los indicadores en la salida. Para realizar un test exhaustivo, es mejor utilizar las funciones de test de S7-SCL.

Además, el usuario dispone de todas las posibilidades del paquete básico STEP 7 en varios idiomas.

3.2 Planteamiento

Resumen

La tarea consiste en adquirir valores medidos a través de un módulo de entrada y clasificarlos y procesarlos con un programa S7-SCL. Los resultados deben visualizarse en el módulo de salida.

Adquisición de valores medidos

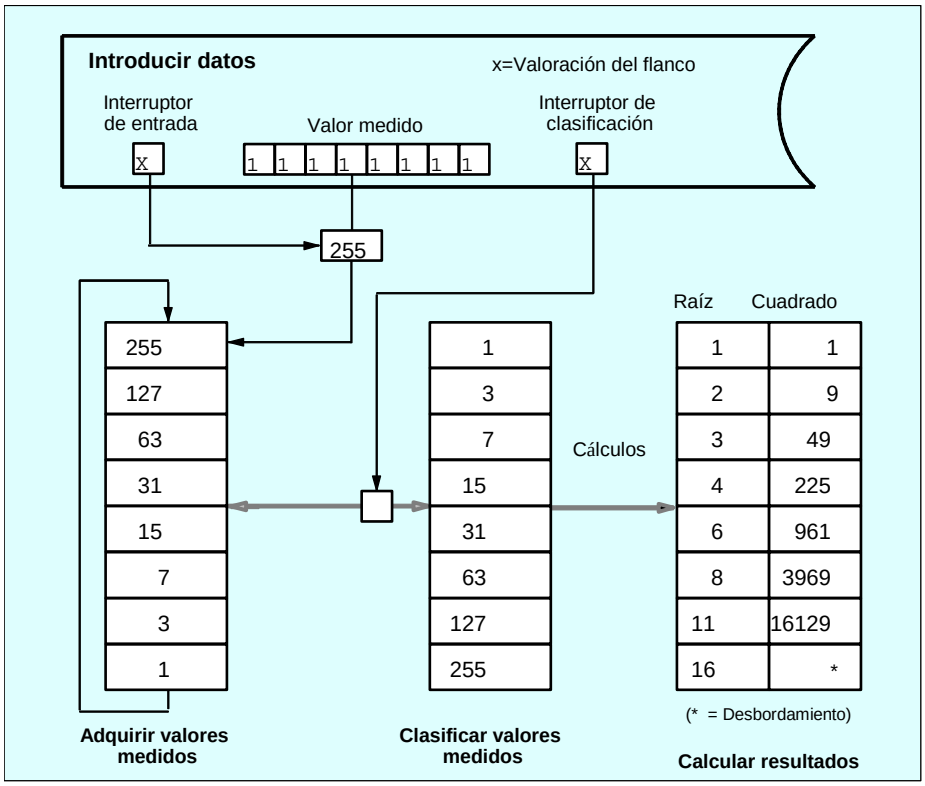
Un valor medido se ajusta a través de los 8 interruptores de entrada. Cuando en uno de los interruptores de entrada se detecte un flanco (v. figura siguiente), habrá que registrar dicho valor en el array del valor medido.

El margen permitido para los valores medidos oscila entre 0 y 255. Por consiguiente se requiere un byte para la entrada.

Procesamiento de valores medidos

El array del valor medido debe ser un búfer anular de 8 entradas como máximo.

Si en un interruptor de clasificación se detecta un flanco, habrá que clasificar los valores guardados en el array de valor medido en orden ascendente. Después se calculará la raíz y el cuadrado de cada uno de los valores. Para las funciones de procesamiento se necesita una palabra.



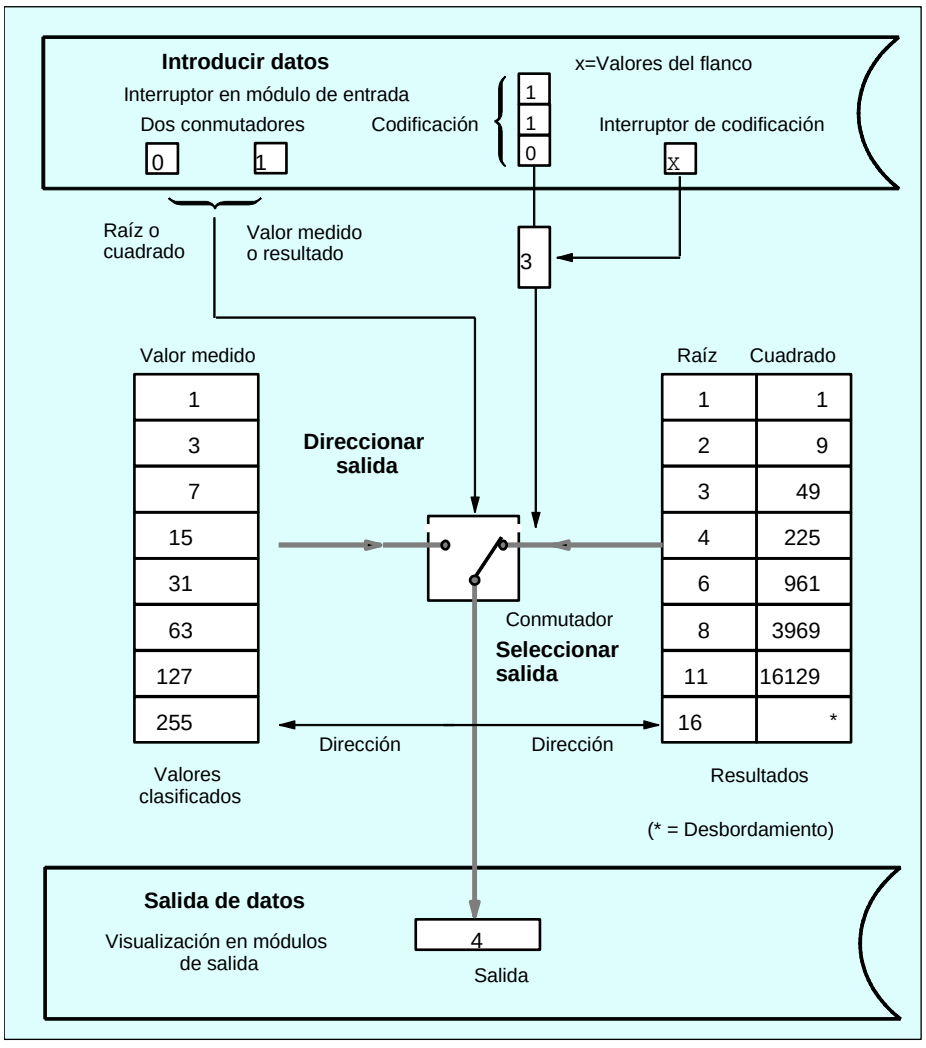
Salidas ajustables

En el módulo de salida sólo se puede mostrar un valor. Por eso se programarán las siguientes opciones:

- Selección de un elemento de una lista
- Selección entre valor medido, raíz y cuadrado

La selección del valor mostrado se realiza de la manera siguiente:

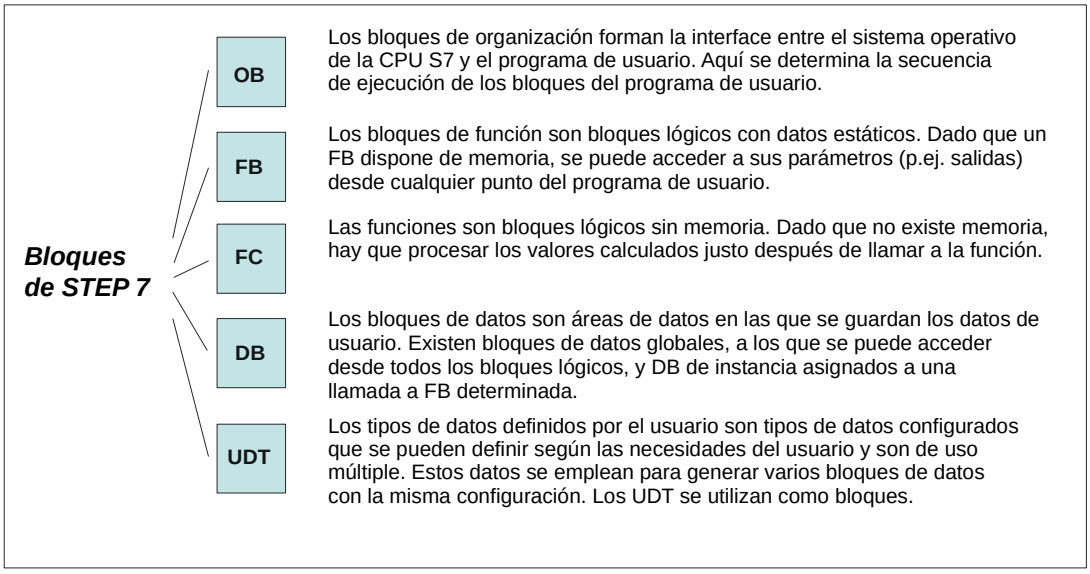
- Con tres interruptores se ajusta una codificación que se acepta al detectarse un flanco en el cuarto interruptor, el interruptor de codificación. A partir de aquí se calcula la dirección con la que se direccionará la salida.
- Con la misma dirección se ponen a disposición tres valores para la salida: valor medido, raíz y cuadrado. Para seleccionar uno de estos valores, hay que programar dos conmutadores.



3.3 Crear un programa estructurado con S7-SCL

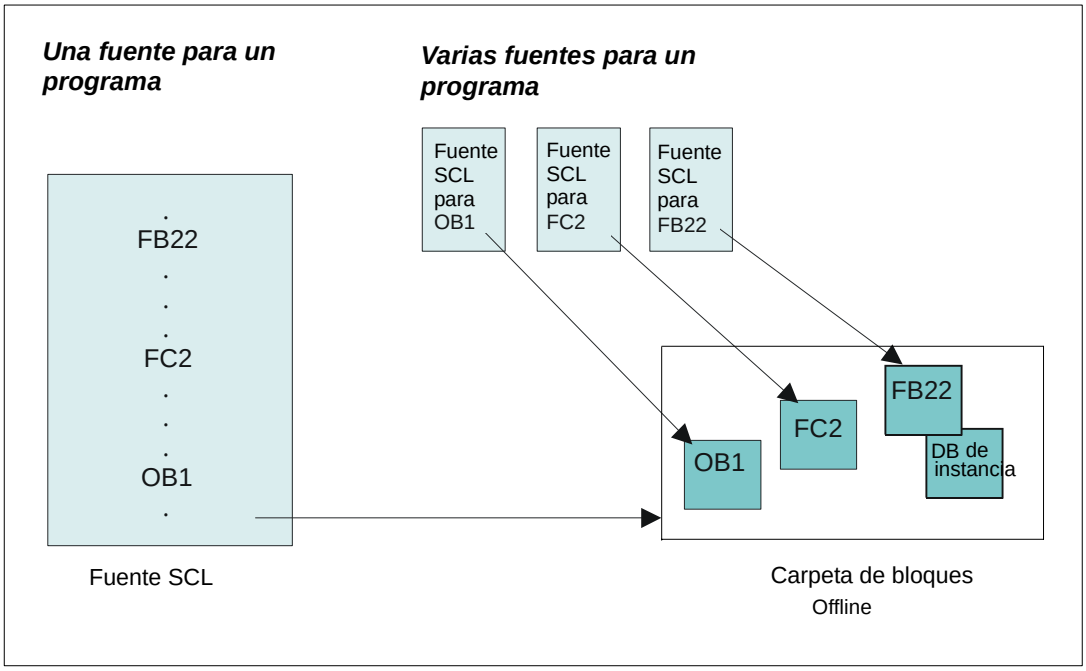
Tipos de bloques

La mejor manera de solucionar la tarea planteada es mediante un **programa S7-SCL estructurado**. Dicho programa tiene estructura modular; es decir, se compone de bloques que procesan una o varias tareas parciales. Al igual que los lenguajes de programación de STEP 7, S7-SCL ofrece los siguientes tipos de bloques.



Organización de los bloques en fuentes S7-SCL

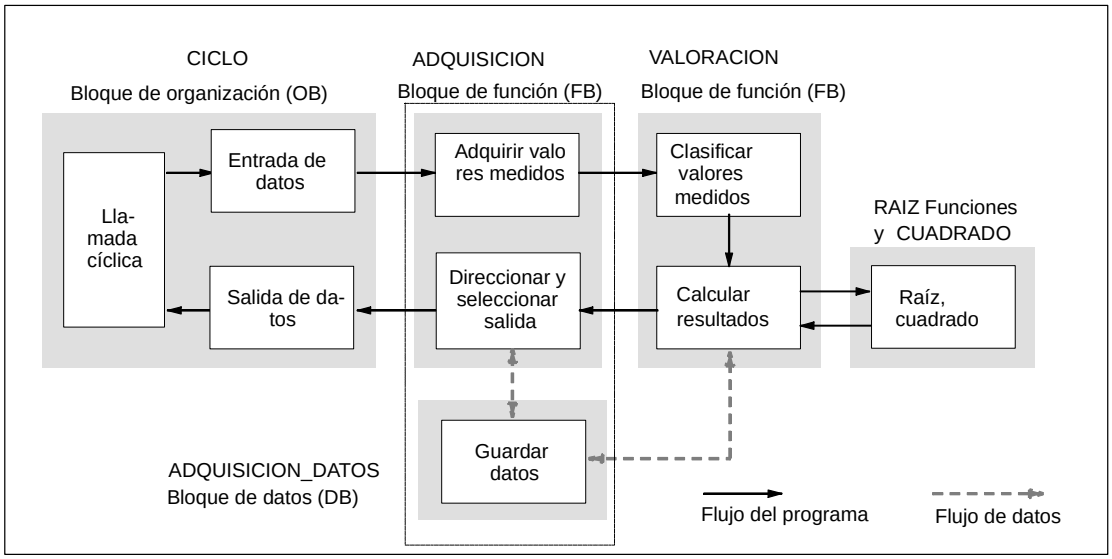
Un programa S7-SCL se compone de una o más fuentes. Una fuente puede contener un único bloque o un programa completo compuesto de varios bloques.



3.4 Definir las tareas parciales

Tareas parciales

Las tareas parciales aparecen representadas en forma de cuadros en la siguiente figura. Las rectángulos sobre fondo gris representan los bloques. La disposición de los bloques lógicos de izquierda a derecha se corresponde con la secuencia de llamada.



Selección y asignación de los tipos de bloque

Los bloques se seleccionan de acuerdo con los siguientes criterios:

Función		**Nombre de bloque**
Los programas de usuario sólo se pueden iniciar desde un OB. Dado que los valores medidos se adquieren de manera cíclica, se requiere un OB para *llamadas cíclicas* (OB1). Una parte del procesamiento - *entrada de datos* y *salida de datos* - se programa en el OB.	⇒	OB "Ciclo"
Para la tarea parcial *Adquisición de valores medidos* se requiere un bloque con memoria, es decir un FB, ya que hay que mantener ciertos datos locales de bloque (p.ej. el búfer anular) de un ciclo de programa en el siguiente. El lugar donde se *almacenan los datos* (memoria) es el bloque de datos de instancia `ADQUISICION_DATOS`. El mismo FB puede asumir la tarea parcial Direccionar salida o Seleccionar salida, puesto que se dispone de los datos necesarios.	⇒	FB "ADQUISICION"
Al seleccionar el tipo de bloque para resolver las tareas parciales *Clasificación valores medidos* y *Cálculo de resultados* hay que tener en cuenta que es necesario crear un búfer de salida que contenga los resultados de cálculo raíz y cuadrado de cada valor medido. Por este motivo el único bloque posible es un FB. Dado que el FB es llamado por un FB de orden superior, no necesita ningún DB propio. Sus datos de instancia se pueden crear en el bloque de datos de instancia del FB invocante.	⇒	FB "EVALUACION"
La FC resulta idónea para resolver la tarea parcial *Cálculo de raíz o cuadrado* porque se puede producir el retorno del resultado como valor de la función. Además, para el cálculo no se requiere ningún dato que deba conservarse más de un ciclo de ejecución del programa. Para calcular la raíz se puede utilizar la función estándar de S7-SCL `SQRT`. Para calcular el cuadrado debe crearse una función CUADRADO, que también comprobará los límites del rango.	⇒ ⇒	FC "SQRT" (raíz) y FC "CUADRADO"

3.5 Definir los interfaces entre bloques

Resumen

El interface de un bloque se crea mediante parámetros a los que se puede acceder desde otros bloques.

Los parámetros declarados en el bloque son comodines cuyos valores se determinan al efectuarse la llamada al bloque. Estos comodines se denominan parámetros formales y los valores asignados al llamar el bloque, parámetros actuales. Cuando se llama a un bloque se le transfieren datos de entrada en forma de parámetros actuales. Al retornar al bloque invocante se ponen a disposición los datos de salida para su transferencia. Una función (FC) puede transferir su resultado como valor de la función.

Los parámetros de bloque se pueden dividir en las siguientes categorías:

Parámetros de bloques	Significado	Declaración con
Parámetros de entrada	Los parámetros de entrada asumen los valores de entrada actuales cuando se llama al bloque. Sólo permiten accesos de lectura.	`VAR_INPUT`
Parámetros de salida	Los parámetros de salida transfieren los valores de salida actuales al bloque invocante. Permiten accesos de lectura y escritura.	`VAR_OUTPUT`
Parámetros de entrada/salida	Cuando se efectúa la llamada, los parámetros de entrada/salida asumen el valor actual de una variable, lo procesan y a continuación devuelven los resultados a la misma variable.	`VAR_IN_OUT`

OB Ciclo

El OB `CICLO` no dispone de ningún parámetro formal. Llama al FB `ADQUISICION` y le asigna el valor medido y los datos de control para sus parámetros formales.

FB ADQUISICION

Nombre del parámetro	Tipo de datos	Tipo de declaración	Descripción
intr_val_med	INT	VAR_INPUT	Valor medido
nue_val	BOOL	VAR_INPUT	Interruptor para aceptar el valor medido en el búfer de anillo
nue_clas	BOOL	VAR_INPUT	Interruptor para clasificar y evaluar valores medidos
sel_funcion	BOOL	VAR_INPUT	Conmutador para seleccionar raíz o cuadrado
seleccion	WORD	VAR_INPUT	Código para seleccionar valor de salida
nue_sel	BOOL	VAR_INPUT	Interruptor para aceptar la codificación
sal_resultado	DWORD	VAR_OUTPUT	Salida del resultado calculado
sal_v_med	DWORD	VAR_OUTPUT	Salida del valor medido correspondiente

Evaluar

El FB ADQUISICION llama al FB EVALUACION. Ambos FB reciben como dato común el array del valor medido que se clasificará. Por ello se declara como parámetro de entrada/salida. Para el resultado de cálculo de la raíz y del cuadrado se crea un array estructurado en forma de parámetro de salida. En la siguiente tabla se indican los parámetros formales:

Nombre	Tipo de datos	Tipo de declaración	Descripción
bufer_clasif	ARRAY[..] OF REAL	VAR_IN_OUT	Array de valor medido, se corresponde con el búfer anular
bufer_calculo	ARRAY[..]OF STRUCT	VAR_OUTPUT	Array para resultados: estructura con los componentes "RAIZ" y "CUADRADO" del tipo INT

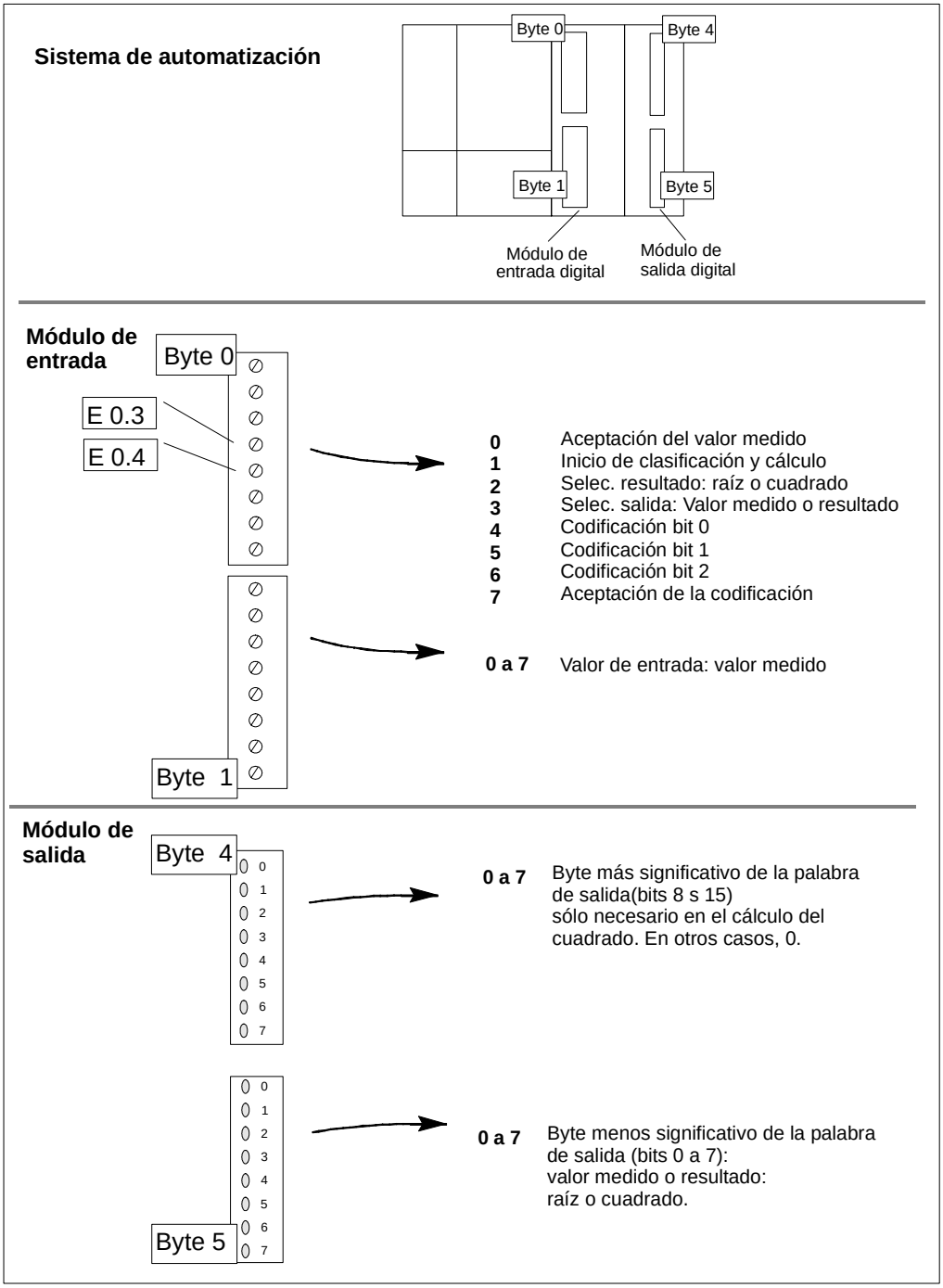
SQRT y cuadrado

Las funciones se llaman desde EVALUACION. Necesitan un valor de entrada y proporcionan un valor de función como resultado.

Nombre	Tipo de datos	Tipo de declaración	Descripción
Valor	REAL	VAR_INPUT	Entrada para RAIZ
SQRT	REAL	Valor de función	Raíz del valor de entrada
Valor	INT	VAR_INPUT	Entrada para CUADRADO
CUADRADO	INT	Valor de función	Cuadrado del valor de entrada

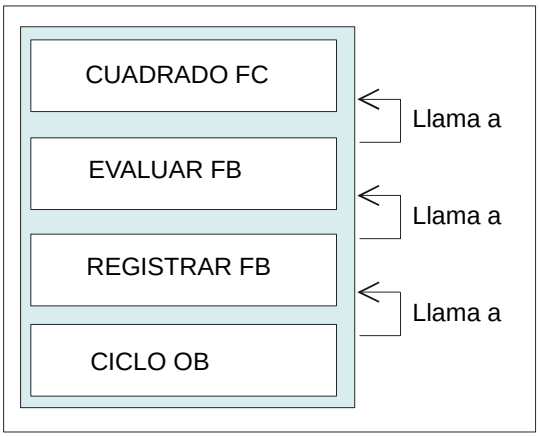
3.6 Definir el interface de entrada/salida

En la siguiente figura se muestra el interface de entrada/salida. Tenga en cuenta que en la entrada/ salida byte a byte, el byte menos significativo aparece arriba y el byte más significativo aparece abajo. Sin embargo, en la entrada/salida palabra a palabra sucede todo lo contrario.



3.7 Definir la secuencia de bloques en la fuente

En cuanto a la secuencia de los bloques en la fuente S7-SCL, hay que tener en cuenta que un bloque debe existir antes de poderlo utilizar, es decir, de llamarlo desde otro bloque. En la fuente S7-SCL, los bloques deben estar ordenados de esta forma:



3.8 Definir los símbolos

El programa resulta más comprensible asignando nombres simbólicos a las direcciones de los módulos y a los bloques. Los nombres simbólicos se asignan en la tabla de símbolos.

La siguiente figura contiene la tabla de símbolos del ejemplo. En ella se describen los nombres simbólicos que hay que declarar en la tabla de símbolos para poder compilar la fuente sin que se produzca ningún error:

	Símbolo	Dirección	Tipo de dato
1	ADQUISICION	FB 10	FB 10
2	ADQUISICION_DATOS	DB 10	FB 10
3	CICLO	OB 1	OB 1
4	codificacion	EW 0	WORD
5	CUADRADO	FC 41	FC 41
6	Entrada	EB 1	BYTE
7	Entrada 0.0	E 0.0	BOOL
8	Interr_clasif	E 0.1	BOOL
9	Interr_codif	E 0.7	BOOL
10	Interr_funcion	E 0.2	BOOL
11	Interr_salida	E 0.3	BOOL
12	salida	AW 4	INT
13	EVALUACION	FB 20	FB 20

3.9 Crear la función CUADRADO

3.9.1 Área de instrucciones de la función CUADRADO

Área de instrucciones

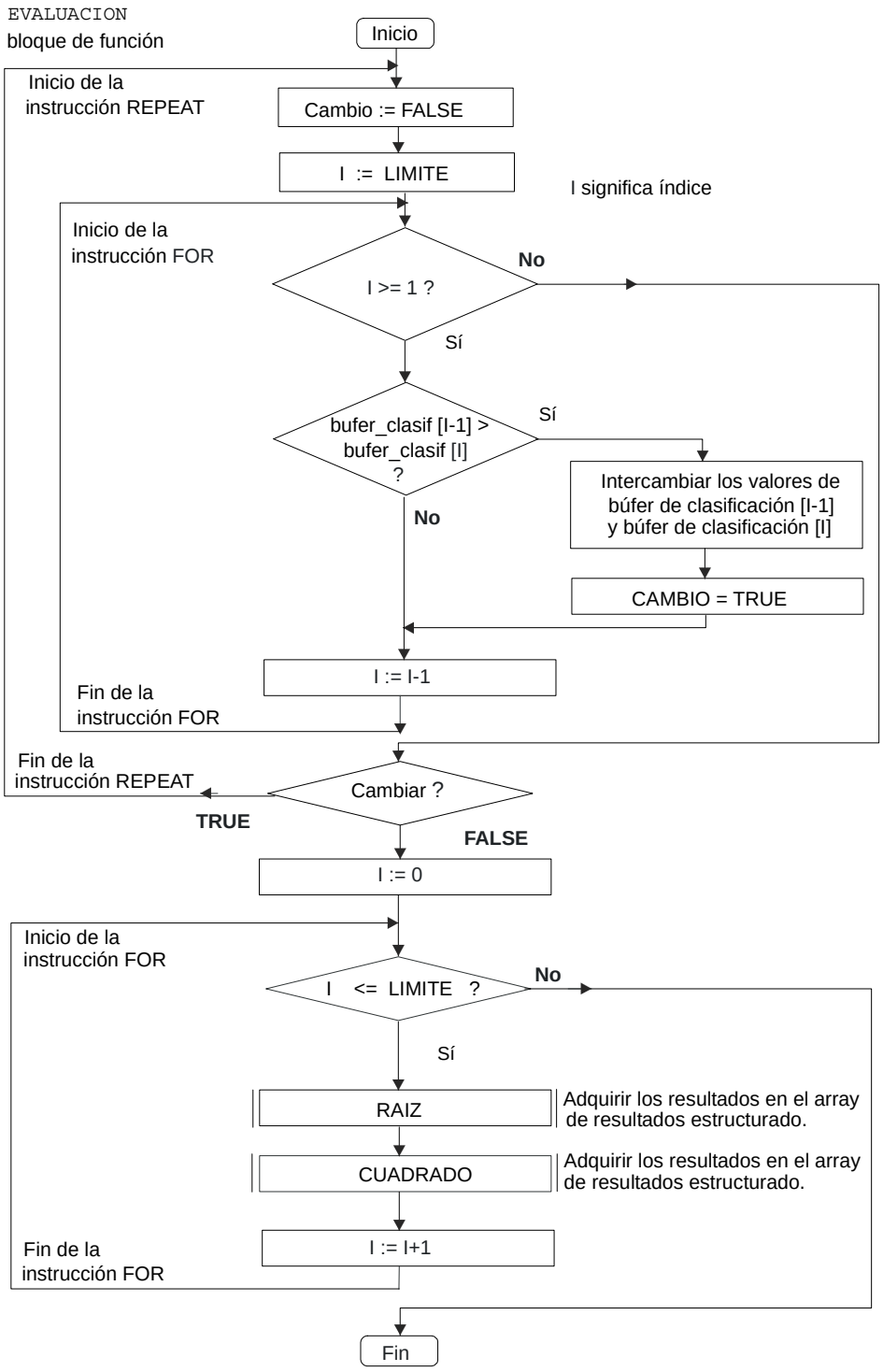
En primer lugar se comprueba si el valor de entrada excede el límite en el que el resultado es superior al rango permitido para enteros. En este caso se registraría el valor máximo para Integer. En caso contrario, se efectuaría la operación de elevar al cuadrado. El resultado se transfiere como valor de función.

```
FUNCTION CUADRADO : INT
(*****************************************************************
Esta función proporciona como valor de función el cuadrado de
entrada,
o, en caso de desbordamiento, el valor máximo que puede
representarse con enteros.
*****************************************************************
*)
VAR_INPUT
    valor : INT;
END_VAR
BEGIN
IF valor <= 181 THEN
    CUADRADO   := valor * valor; //Cálculo del valor de la función
ELSE
    CUADRADO   := 32_767; // Definir valor máximo en desbordamiento
END_IF;
END_FUNCTION
```

3.10 Crear el bloque de función EVALUACION

3.10.1 Diagrama de flujo de EVALUACIÓN

La figura representa el algoritmo en forma de diagrama de flujo:



3.10.2 Área de declaración del FB EVALUACION

Estructura del área de declaración

El área de declaración de este bloque está formada por las siguientes partes:

- Constantes: entre CONST y END_CONST
- Parámetros de entrada/salida: entre VAR_IN_OUT y END_VAR,
- Parámetros de salida: entre VAR_OUTPUT y END_VAR
- Declaración de las variables temporales: entre VAR_TEMP y END_VAR

```
CONST
    LIMITE := 7;
END_CONST
VAR_IN_OUT
    bufer_clasif : ARRAY[0..LIMITE] OF INT;
END_VAR
VAR_OUTPUT
    bufer_calculo  : ARRAY[0..LIMITE] OF
        STRUCT
            raiz  : INT;
            cuadrado : INT;
        END_STRUCT;
END_VAR
VAR_TEMP
    cambiar      : BOOL;
    indice, ayuda   : INT;
    n_valor, n_resultado: REAL;
END_VAR
```

3.10.3 Área de instrucciones del FB EVALUACION

Ejecución del programa

El parámetro de entrada/salida "búfer_clasif" se combina con el búfer anular "valores_medidos", es decir, el contenido original del búfer se sobrescribe con los valores medidos clasificados.

Para los resultados de cálculo se crea el nuevo array "bufer_calculo", configurado como parámetro de salida. Sus elementos están estructurados de tal forma que conservan la raíz y el cuadrado de cada valor medido.

En la figura siguiente puede encontrar la relación entre los arrays descritos:

EVALUACION

Este interface muestra el núcleo del intercambio de datos para procesar los valores medidos. Los valores se guardan en el bloque de datos de instancia `ADQUISICION_DATOS` ya que en el `FB ADQUISICION` invocante se ha creado una instancia local para el `FB EVALUACION`.

Área de instrucciones de EVALUACION

En primer lugar se clasifican los valores medidos en el búfer anular y a continuación se realizan los cálculos:

- Método del algoritmo de clasificación
  Aquí se utiliza el método de intercambio permanente de valores para la clasificación del búfer de valores medidos, es decir, se comparan dos valores consecutivos y se intercambian hasta que se alcanza la secuencia de clasificación deseada. El búfer utilizado es el parámetro de entrada/salida "bufer_clasif".
- Inicio del cálculo
  Cuando termina la clasificación se ejecuta un bucle para el cálculo en el que se llama a las funciones `CUADRADO` para elevar al cuadrado y `SQRT` para calcular la raíz. Sus resultados se guardan en el array estructurado "bufer_calculo".

Área de instrucciones de EVALUACION

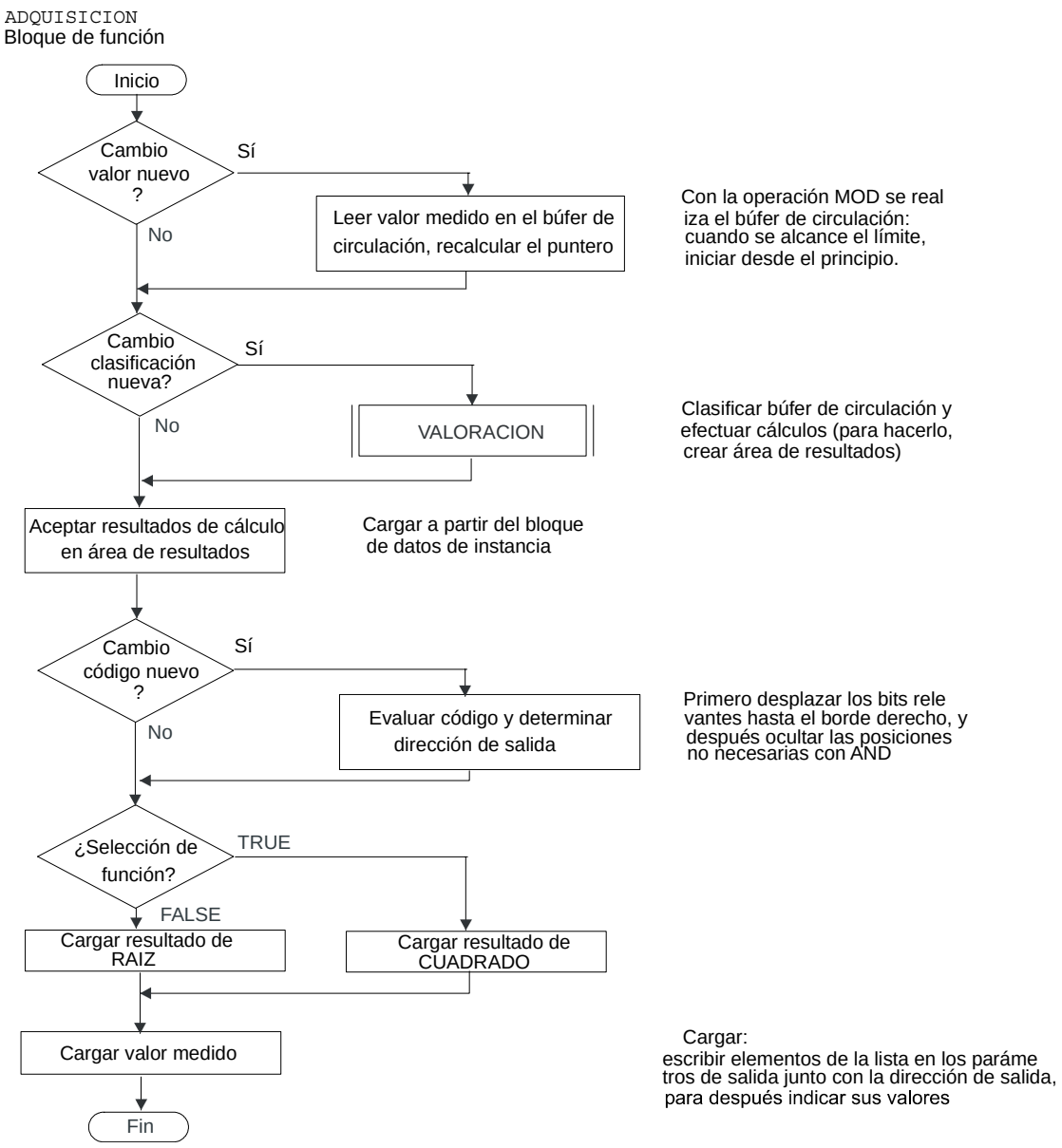
El área de instrucciones del bloque lógico presenta la siguiente estructura:

```
BEGIN
(* Parte 1  'Clasificacion' :
Clasificación según el proceso "bubble sort": intercambiar de dos en
dos los valores hasta que el búfer de valores medidos esté
clasificado ******************************************************)
REPEAT
    cambiar := FALSE;
        FOR indice := LIMITE TO 1 BY -1 DO
            IF bufer_clasif[indice-1] > bufer_clasif[indice] THEN
                ayuda                   := bufer_clasif[indice];
                bufer_clasif[indice]   := bufer_clasif[indice-1];
                bufer_clasif[indice-1] := ayuda;
                cambiar               := TRUE;
            END_IF;
        END_FOR;
    UNTIL NOT cambiar
END_REPEAT;
(*****************************************************************
*
Parte 2  'Calculo' :
Cálculo de la raíz con la función estándar RAIZ y
obtención del cuadrado con la funcion CUADRADO.
*************************************************)
FOR indice       := 0 TO LIMITE BY 1 DO
    n_valor       := INT_TO_REAL(bufer_clasif[indice]);
    n_resultado := SQRT(n_valor);
    bufer_calculo[indice].raiz        := REAL_TO_INT(n_resultado);
    bufer_calculo[indice].cuadrado :=
CUADRADO(bufer_clasif[indice]);
END_FOR;
```

3.11 Crear el bloque de función ADQUISICIÒN

3.11.1 Diagrama de flujo de ADQUISICION

La siguiente figura representa el algoritmo en forma de diagrama de flujo:



3.11.2 Área de declaración del FB ADQUISICION

Estructura del área de declaración

El área de declaración de este bloque está formada por los siguientes bloques de declaración:

- Constantes: entre CONST y END_CONST.
- Parámetro de entrada: entre VAR_INPUT y END_VAR
- Parámetros de salida: entre VAR_OUTPUT y END_VAR.
- Variables estáticas: entre VAR y END_VAR.
  Incluida la declaración de la instancia local para el bloque EVALUACION.

```
CONST
    LIMITE := 7;
    NUMERO:= LIMITE + 1;
END_CONST
VAR_INPUT
    intr_val_med: INT ; // Nuevo valor medido
    nue_val     : BOOL; // Aceptar valor medido en búfer de
circulación
// "valores_medidos"
    nue_clas     : BOOL; // Clasificar valores medidos
    sel_funcion  : BOOL; // Seleccionar la función de cálculo
raiz/cuadrado
    nue_sel      : BOOL; // Aceptar dirección de salida
    seleccion: WORD; // Dirección de salida
END_VAR
VAR_OUTPUT
    sal_resultado : INT; // valor calculado
    sal_v_med: INT; // valor medido correspondiente
END_VAR
VAR
    valores_medidos      : ARRAY[0..LIMITE] OF INT := 8(0);
    bufer_resultado      : ARRAY[0..LIMITE] OF
    STRUCT
        raiz       : INT;
        cuadrado   : INT;
    END_STRUCT;
    puntero        : INT  := 0;
     ant_val       : BOOL := TRUE;
    ant_clas       : BOOL := TRUE;
    ant_sel        : BOOL := TRUE;  //Dirección de salida convertida
    valorar_instancia: evaluacion;  //Declarar instancia local
END_VAR
```

Variables estáticas

Se ha seleccionado el tipo de bloque FB porque hay datos que se deben mantener de un ciclo de programa al siguiente. Estos datos son las variables estáticas declaradas en el bloque de declaración "VAR, END_VAR".

Las variables estáticas son variables locales cuyo valor se mantiene en todos los recorridos del bloque. Sirven para guardar los valores de un bloque de función y se almacenan en el bloque de datos de instancia.

Inicialización de las variables

Tenga en cuenta los valores de inicialización que se registran en las variables al inicializar el bloque (después de cargar el programa en la CPU). En la tabla de declaración VAR, END_VAR también se declara la instancia local del FB EVALUACION. El nombre se utilizará posteriormente para la llamada y para acceder a los parámetros de salida. Como memoria de datos se utiliza la instancia global ADQUISICION_DATOS.

Nombre	**Tipo de datos**	**Pre-ajuste**	**Descripción**
valores_medidos	ARRAY [..] OF INT	8(0)	Búfer anular para valores medidos
bufer_resultado	ARRAY[..] OF STRUCT	-	Array para estructuras con los componentes "raíz" y "cuadrado" del tipo INT
puntero	INT	0	Índice para búfer anular, allí se registra el valor medido
ant_val	BOOL	FALSE	Anterior valor para la validación del valor medido con "nue_val"
ant_clas	BOOL	FALSE	Valor precedente para clasificar con "nue_clas"
ant_sel	BOOL	FALSE	Valor precedente para aceptar la codificación con "nue_sel"
direccion	INT	0	Dirección para la salida de valores medidos o resultados
valorar_instancia	Instancia local	-	Instancia local para el FB EVALUACION

3.11.3 Área de instrucciones del FB ADQUISICION

Estructura del área de instrucciones

El área de instrucciones de `ADQUISICION` se divide en tres partes:

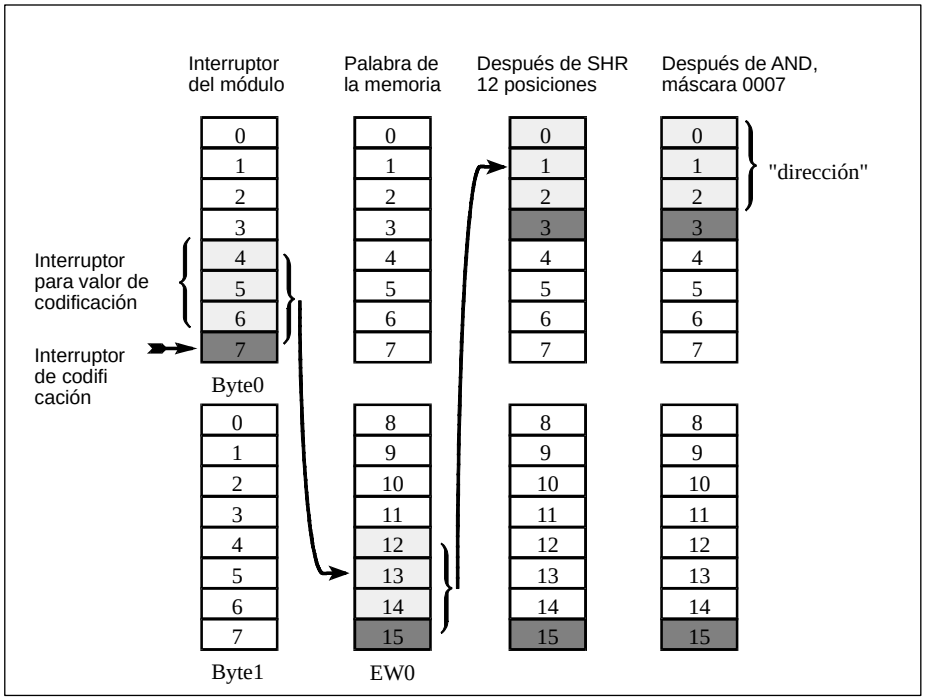
- adquirir valores medidos:
  cuando el parámetro de entrada cambia de "valor antiguo" a "valor nuevo" se carga un valor medido nuevo en el búfer anular.
- iniciar clasificación y cálculo
  Llamando al bloque de función `EVALUACION`, cuando el parámetro de entrada cambia de "clasificación antigua" a "clasificación nueva".
- evaluar codificación y preparar salida
  La codificación se lee por palabras: según las convenciones de SIMATIC, esto significa que el grupo de interruptores superior (Byte0) contiene los 8 bits más significativos de la palabra de entrada y el grupo de interruptores inferior (Byte1), los menos significativos. La siguiente figura indica dónde se encuentran los interruptores en los que se puede ajustar la codificación:

Cálculo de la dirección

La siguiente figura muestra el cálculo de la dirección: la palabra de entrada EW0 contiene en los bits 12 a 14 el código que se acepta cuando en el interruptor de codificación (bit 15) se detecta un flanco. Desplazando a la derecha con la función estándar SHR e inhibiendo los bits relevantes con una máscara AND se calcula la "dirección".

Con esta dirección se escriben los elementos de array (resultado de cálculo y valor medido correspondiente) en los parámetros de salida. Que la salida sea raíz o cuadrado depende de la "selección de función".

Un flanco en el interruptor de codificación se detecta porque el "valor nuevo" cambia respecto al "valor antiguo".



Área de instrucciones

El área de instrucciones del bloque lógico presenta la siguiente estructura:

```
BEGIN
(*****************************************************************
Parte 1 : 'Adquisicion' de 'valores_medidos'
Al cambiar "nue_val" se produce la entrada del valor medido.
Con la operación MOD se ejecuta un búfer de circulación para valores
medidos.
*****************************************************************)
IF nue_val <> ant_val THEN
    puntero                  := puntero MOD NUMERO;
    valores_medidos[puntero] := Intr_val_med;
    puntero                  := puntero + 1;
END_IF;
ant_val := nue_val;
(*****************************************************************
Parte 2 : Iniciar 'Clasificación' y 'Cálculo'
  Al cambiar "nue_clas" se inicia la clasificación del búfer de
circulación y la ejecución
  de los cálculos con los valores medidos. Los resultados se guardan
en un nuevo array, "bufer_calculo". *)
 IF nue_clas <> ant_clas THEN
    puntero := 0;      //Inicializar puntero del búfer
                       //de circulación
    valorar_instancia(bufer_clasif := valores_medidos);
                       //Llamar evaluacion
END_IF;
ant_clas                     := nue_clas;
bufer_resultado := valorar_instancia.bufer_calculo; //cuadrado y
raiz
(*****************************************************************
(* Parte 3 : Valorar código y preparar salida:
Al cambiar "nue_sel" se determina de nuevo el código para el
direccionamiento del elemento de array para la salida: Los datos
relevantes de 'seleccion' se ocultan y se transformen en
entero. Dependiendo de la posición del interruptor de "sel_funcion"
en la salida se dispondrá     "RAIZ" o "CUADRADO" *)
*****************************************************************)
IF nue_sel <> ant_sel THEN
    direccion := WORD_TO_INT(SHR(IN := seleccion, N := 12) AND
16#0007);
END_IF;
ant_sel := nue_sel;
IF sel_funcion THEN
    sal_resultado:= bufer_resultado[direccion].cuadrado;
ELSE
    sal_resultado:= bufer_resultado[direccion].raiz;
END_IF;
sal_v_med   := valores_medidos[direccion]; //indicación del valor
medido
END_FUNCTION_BLOCK
```

3.12 Crear el bloque de organización CICLO

Tareas del OB CICLO

Se ha seleccionado un OB1 porque se llama de forma cíclica. Con él se ejecutan las siguientes tareas del programa:

- Llamar y transferir datos de entrada y datos de control al bloque de función ADQUISICION.
- Aceptar los resultados del bloque de función ADQUISICION
- Salida de los valores para su visualización.

Al principio del área de declaración aparece el array de datos temporal con 20 bytes de "datos del sistema".

Código de programa del OB CICLO

```
ORGANIZATION_BLOCK CICLO
(******************************************************************
*  CICLO equivale al OB1, es decir, el sistema S7 lo llama
cíclicamente
Parte 1 :  llamada del bloque de función y transferencia de los
valores de entrada
Parte 2 :  aceptación de los valores de salida con conmutación de
salida
******************************************************************)
VAR_TEMP
    datos_del_sistema  : ARRAY[0..20] OF BYTE; // Area para OB1
END_VAR
BEGIN
(* Parte 1 : ************************************************)
ADQUISICION.ADQUISICION_DATOS(
    intr_val_med := WORD_TO_INT(entrada),
    nue_val      := "entrada 0.0",//Interruptor de entrada como
símbolo
    nue_clas     := "Interr_clasif",
    sel_funcion  := "Interr_funcion",
    nue_sel      := "Interr_codif",
    seleccion    := codificacion);
(* Parte 2 :
*****************************************************)
IF Interr_salida THEN          //Conmutación de salida
    salida    := ADQUISICION_DATOS.sal_resultado;     //Raíz o
cuadrado
ELSE
    salida    := ADQUISICION_DATOS.sal_v_med;     //Valor medido
END_IF;
END_ORGANIZATION_BLOCK
```

Conversión de tipos de datos

El valor medido en la entrada es del tipo BYTE. Hay que convertirlo a INT: Se transforma de WORD a INT (el compilador realiza la conversión previa de BYTE a WORD de forma implícita). No es necesario realizar ninguna conversión para la salida, ya que ésta está declarada como INT en la tabla de símbolos.

3.13 Datos del test

Requisitos

Para el test se necesita un módulo de entrada con la dirección 0 y un módulo de salida con la dirección 4.

Antes de efectuar el test, conmutar los 8 interruptores superiores del módulo de entrada hacia la izquierda ("0") y los 8 interruptores inferiores hacia la derecha ("1").

Cargue previamente los bloques en la CPU, ya que también se comprueban los valores iniciales de las variables inmediatamente después.

Pasos del test

Realice los pasos del test como se indica en la tabla.

Paso	Acción	Resultado
1	Active el código "111" (E0.4, E0.5 y E0.6) y acéptelo con el interruptor de codificación (E0.7).	Todas las salidas del módulo de salida (byte menos significativos) se activan y se encienden los indicadores.
2	Visualice la raíz correspondiente conmutando el interruptor de salida (E0.3) a la posición "1".	Los indicadores de salida corresponden al valor binario "10000" (=16).
3	Visualice el cuadrado correspondiente conmutando el interruptor de función (E0.2) a la posición "1".	En la salida se iluminan 15 indicadores. Esto significa que se produce un rebose, dado que con 255 x 255 se obtiene un valor demasiado elevado para el dominio de enteros.
4a	Vuelva a conmutar el interruptor de salida (E0.3) a la posición "0".	Vuelve a aparecer el valor medido: todas las indicaciones en las salidas del byte de salida de menor valor están activadas.
4b	Introduzca el valor 3, es decir, el valor binario "11" como nuevo valor medido en la entrada.	La salida no cambia todavía.
5a	Observe la escritura en memoria del valor medido: ajuste la codificación a "000" y confírmela con el interruptor de codificación (E0.7) para poder observar más tarde la entrada de valores.	En la salida se visualiza 0; es decir, no se enciende ningún indicador.
5b	Conmute el interruptor de entrada "Entrada 0.0" (E0.0). De esta forma se escribirá en la memoria el valor ajustado en el cuarto paso del test.	A la salida se muestra el valor medido 3, valor binario "11".
6	Inicie la clasificación y el cálculo conmutando el interruptor de clasificación (E0.1).	A la salida vuelve a aparecer 0, dado que por medio del proceso de clasificación, el valor medido se ha vuelto a desplazar hacia arriba en el array.
7	Visualizar el valor medido después de clasificar: ajuste el código "110" (E0.6=1, E0.5=1, E0.4= 0 de EB0, corresponde a bit 14, bit 13, bit 12 de EW0) y confírmelo conmutando el interruptor de codificación.	A la salida aparece otra vez el valor medido "11", ya que se trata del segundo valor más alto del array.

Paso	Acción	Resultado
8a	Visualizar los resultados correspondientes: conmutando el interruptor de salida (E0.3) se visualiza el cuadrado del valor medido obtenido en el paso 7.	Se visualiza el valor de salida 9 o el valor binario "1001".
8b	Conmutando el interruptor de función (E0.2) obtendrá también la raíz.	Se visualiza el valor de salida 2 o el valor binario "10".

Test adicional

En las siguientes tablas aparecen los interruptores del módulo de entrada y patrones de test para la raíz y el cuadrado. Además, estas tablas le ayudarán a definir sus propios pasos de test:

- La entrada se realiza a través de interruptores: el programa se puede controlar mediante los 8 interruptores superiores y los valores medidos se pueden ajustar con los 8 inferiores.
- La salida se realiza a través de indicadores: en el grupo superior aparece el byte de salida más significativo, y en el grupo inferior el menos significativo.

Interruptores de manejo	Nombre del parámetro	Explicación
Canal 0	Interruptor de entrada	Conmutación para la validación de valores medidos
Canal 1	Interruptor de clasificación	Conmutación para clasificación/evaluación
Canal 2	Interruptor de función	Interruptor hacia la izquierda ("0"): raíz, Interruptor hacia la derecha ("1"): cuadrado
Canal 3	Interruptor de salida	Interruptor hacia la izquierda ("0"): valor medido, Interruptor hacia la derecha ("1"): resultado
Canal 4	Codificación	Dirección de salida bit 0
Canal 5	Codificación	Dirección de salida bit 1
Canal 6	Codificación	Dirección de salida bit 2
Canal 7	Interruptor de codificación	Conmutación para confirmación de código

La siguiente tabla contiene a título de ejemplo 8 valores medidos en una secuencia ya clasificada.

Introduzca los valores en cualquier orden. Ajuste la combinación binaria deseada y acepte el valor correspondiente conmutando el interruptor de entrada. Una vez introducidos todos los valores inicie la clasificación y evaluación conmutando el interruptor de clasificación. A continuación puede visualizar los valores clasificados o los resultados - raíz o cuadrado.

VALOR_MEDIDO	Raíz	Cuadrado
0000 0001 = 1	0, 0000 0001 = 1	0000 0000, 0000 0001 = 1
0000 0011 = 3	0, 0000 0010 = 2	0000 0000, 0000 1001 = 9
0000 0111 = 7	0, 0000 0011 = 3	0000 0000, 0011 0001 = 49
0000 1111 = 15	0, 0000 0100 = 4	0000 0000, 1110 0001 = 225
0001 1111 = 31	0, 0000 0110 = 6	0000 0011, 1100 0001 = 961
0011 1111 = 63	0, 0000 1000 = 8	0000 1111, 1000 0001 = 3969
0111 1111 = 127	0, 0000 1011 = 11	0011 1111, 0000 0001 = 16129
1111 1111 = 255	0, 0001 0000 = 16	0111 111, 1111 1111 = Indicación de desbordamiento

4 Manejo de S7-SCL

4.1 Iniciar el software S7-SCL

Iniciar S7-SCL desde el interface de Windows

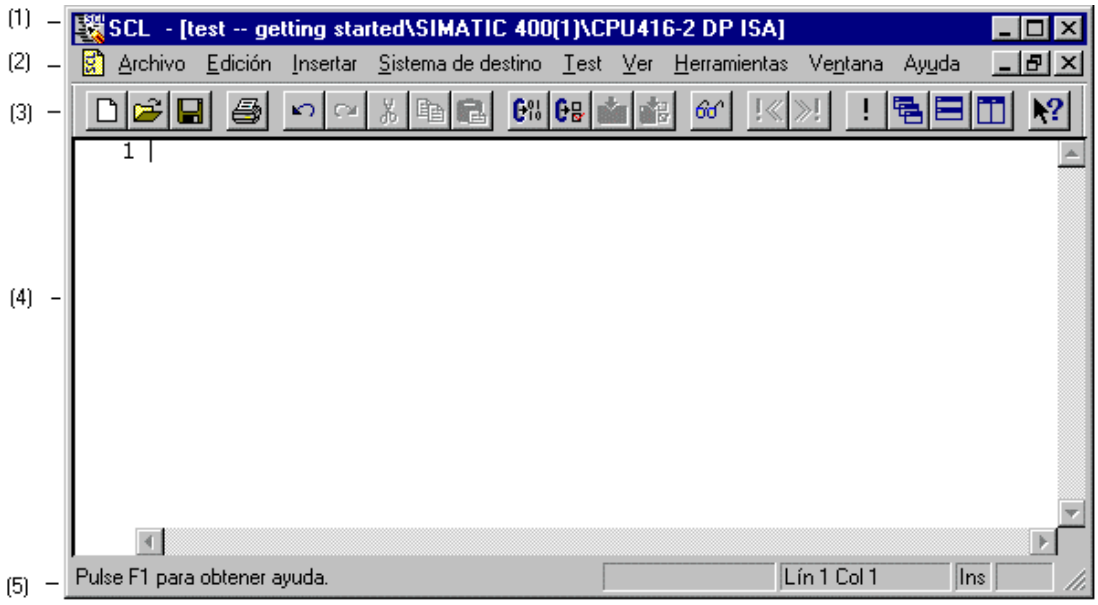
Después de instalar el software S7-SCL en la PG (o en el PC), S7-SCL se puede iniciar desde el menú "Inicio" de la barra de tareas de Windows (bajo "SIMATIC/STEP 7").

Iniciar S7-SCL desde el Administrador SIMATIC

La forma más rápida de iniciar S7-SCL en el Administrador SIMATIC es posicionando el puntero del ratón sobre una fuente S7-SCL y haciendo doble clic.

4.2 Interface de usuario

Las ventanas de S7-SCL están formadas por los siguientes componentes estándar:



1. **Barra de título:**
   incluye el título de la ventana y símbolos para el control de ventanas.
2. **Barra de menús:**
   incluye todos los menús de que se dispone en la ventana.
3. **Barra de herramientas:**
   incluye botones que le permitirán ejecutar rápidamente los comandos de uso más frecuente.
4. **Área de trabajo:**
   incluye las distintas ventanas para editar el texto del programa o leer datos del compilador o datos de test.
5. **Barra de estado:**
   indica el estado y otros datos sobre el objeto seleccionado.

4.3 Personalizar el interface de usuario

Personalización del editor

Para realizar ajustes en el editor seleccione el comando de menú **Herramientas > Preferencias** y haga clic en la ficha "Editor" del cuadro de diálogo "Preferencias". En esta ficha puede realizar los siguientes ajustes:

Opciones de la ficha "Editor"	Significado
Tipo de letra	Establece el tipo de letra para todo el texto fuente.
Ancho del tabulador	Establece el ancho de columna de los tabuladores.
Mostrar números de línea	Muestra números de línea al comienzo de cada línea.
Guardar antes de compilar	Pregunta al usuario si desea guardar la fuente antes de iniciar la compilación.
Consultar antes de guardar	Solicita una confirmación antes de guardar

Personalización del tipo y color de letra

Para modificar el tipo y el color de letra de los distintos elementos de lenguaje, seleccione el comando de menú **Herramientas > Preferencias** y haga clic en la ficha "Formato" del cuadro de diálogo "Preferencias". En esta ficha puede realizar los siguientes ajustes:

Opciones de la ficha "Formato"	Significado
Palabaras clave en mayúsculas	Formatea en mayúsculas las palabras clave de S7-SCL como FOR, WHILE, FUNCTION_BLOCK, VAR o END_VAR al realizar la entrada.
Sangrar en palabras clave	Sangra las líneas en las distintas secciones de declaración y en las instrucciones de control IF, CASE, FOR, WHILE y REPEAT al introducirlas .
Sangrar automáticamente	Tras un salto de línea, sangra la línea siguiente bajo la línea precedente, respetando el mismo margen. Este ajuste sólo es válido para las líneas recién introducidas.
Estilo/color de letra	Determina el estilo y el color de letra de los elementos de lenguaje individuales.

Las preferencias de esta ficha sólo tienen efecto si está activada la opción "Utilizar los siguientes formatos" de la ficha "Formato" y se realizan los ajustes deseados.

Barra de herramientas, barra de puntos de parada, barra de estado

La visualización de la barra de herramientas, la barra de puntos de parada y la barra de estado Se puede activar o desactivar por separado. Para ello, active o desactive el comando correspondiente del menú **Ver**. Una marca de verificación delante del comando de menú indica si el comando está activado o desactivado.

Ventana "Resultados"

La ventana "Resultados" muestra los errores y advertencias que aparecen al compilar una fuente. Esta ventana se puede mostrar y ocultar con el comando de menú **Ver > Resultados**.

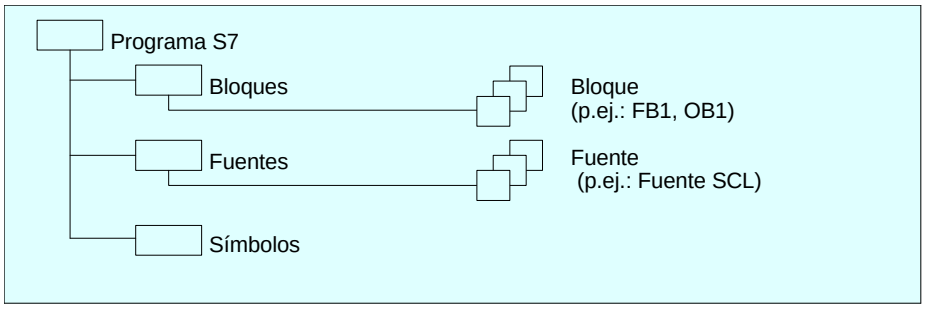
4.4 Crear y manejar fuentes S7-SCL

4.4.1 Crear una fuente S7-SCL

Antes de poder escribir otro programa S7-SCL es necesario crear una fuente S7-SCL. Esta fuente debe crearse en la carpeta "Fuentes" de un programa S7.

Estructura de un programa S7 en el Administrador SIMATIC

Las fuentes creadas con S7-SCL se pueden integrar en la estructura de un programa S7:



Proceda de la siguiente forma:

1. Abra el cuadro de diálogo "Nuevo":
   - haciendo clic en el botón "Nuevo" de la barra de herramientas, o bien
   - seleccionando el comando de menú **Archivo > Nuevo**.
2. En el cuadro de diálogo "Nuevo":
   - un proyecto,
   - el filtro "Fuente S7-SCL" y
   - la carpeta "Fuentes" del programa S7.
3. Introduzca el nombre del objeto fuente en el cuadro de texto correspondiente. El nombre puede tener como máximo 24 caracteres.
4. Confirme con "Aceptar".

El objeto fuente se creará con el nombre que se le haya asignado y se visualizará en una ventana de trabajo para su posterior edición.

Nota

También puede crear una fuente S7-SCL con el Administrador SIMATIC seleccionando una carpeta de fuentes y eligiendo el comando de menú **Insertar > Software S7 > Fuente S7-SCL**.

4.4.2 Abrir una fuente S7-SCL

Puede abrir una fuente S7-SCL para compilarla o editarla.

Proceda de la siguiente forma:

1. Abra el cuadro de diálogo "Abrir":
   - haciendo clic en el botón "Abrir", o bien
   - eligiendo el comando de menú **Archivo > Abrir**.
2. En el cuadro de diálogo seleccione:
   - el proyecto deseado,
   - el programa S7 deseado y
   - la carpeta de fuentes correspondiente.
3. Seleccione la fuente S7-SCL.
4. Haga clic en el botón de comando "Aceptar".

Nota

También puede abrir fuentes en el Administrador SIMATIC haciendo doble clic en su icono o eligiendo el comando de menú **Edición > Abrir objeto** después de haber seleccionado el objeto deseado.

4.4.3 Cerrar una fuente S7-SCL

Proceda del siguiente modo:

- elija el comando de menú **Archivo > Cerrar**, o bien
- haga clic en el botón "Cerrar" de la barra de título de la ventana.

Nota

Si ha modificado la fuente, se le preguntará si desea guardar las modificaciones antes de cerrar. Si no desea guardarlas, las modificaciones se perderán.

4.4.4 Abrir bloques

No es posible abrir bloques con la aplicación S7-SCL. Solamente se puede abrir la fuente correspondiente. Sin embargo, los bloques creados con S7-SCL se pueden abrir con el editor KOP/FUP/AWL y visualizarlos y editarlos en el lenguaje de programación AWL. No modifique el bloque en la representación AWL ya que

- los comandos MC7 mostrados no representan necesariamente un bloque AWL válido,
- una compilación correcta con el compilador AWL requiere una serie de modificaciones para las que se requieren profundos conocimientos de AWL y S7-SCL,
- el bloque compilado con AWL no reconoce el lenguaje AWL y S7-SCL,
- la fuente S7-SCL y el código MC7 dejan de ser coeherentes.

Encontrará más información en la ayuda en pantalla de STEP 7.

Nota

Una forma más sencilla de manejar los programas de CPU es realizando las modificaciones necesarias en las fuentes S7-SCL y compilándolas después .

4.4.5 Definir las propiedades de un objeto

Las propiedades de un objeto se pueden definir asignando atributos a los bloques. Las propiedades de la fuente S7-SCL (p.ej., el autor) se definen en el cuadro de diálogo "Propiedades".

Proceda de la siguiente forma:

1. Elija el comando de menú **Archivo > Propiedades**
2. Introduzca las opciones en el cuadro de diálogo "Propiedades".
3. Confirme con "Aceptar".

4.4.6 Crear fuentes S7-SCL con un editor estándar

También es posible utilizar un editor estándar ASCII para editar la fuente S7-SCL, pero en tal caso no dispondrá de las valiosas funciones de edición ni de la ayuda en pantalla integrada en S7-SCL.

Después de crear y guardar la fuente debe importarla a la carpeta "Fuentes" de un programa S7 con el Administrador SIMATIC (consulte la documentación de STEP 7). A continuación puede abrir la fuente en S7-SCL para seguir editándola o para compilarla.

4.4.7 Proteger bloques

Es posible activar una protección de bloque indicando en la fuente el atributo KNOW_HOW_PROTECT al programar el bloque.

La protección de bloque tiene las siguientes consecuencias:

- Si posteriormente abre un bloque compilado con el editor AWL incremental, no se podrá observar el área de instrucciones del bloque.
- En el área de declaración del bloque sólo se indicarán las variables de los tipos de declaración VAR_IN, VAR_OUT y VAR_IN_OUT. Las variables de los bloques de declaración VAR y VAR_TEMP permanecerán ocultas.

Cómo introducir la protección de bloque:

- La palabra clave es KNOW_HOW_PROTECT. Se introduce delante de los demás atributos del bloque.
- De esta forma pueden protegerse OB, FB, FC y DB.

4.5 Reglas para las fuentes S7-SCL

4.5.1 Reglas generales para las fuentes S7-SCL

Las fuentes S7-SCL deben cumplir las siguientes reglas:

- En una fuente S7-SCL se puede editar cualquier cantidad de bloques lógicos (FB, FC, OB), bloques de datos (DB) y tipos de datos de usuario (UDT).
- Cada tipo de bloque tiene una estructura típica.
- Cada instrucción y cada declaración de variables termina con un punto y coma (;).
- No se distingue entre mayúsculas y minúsculas.
- Los comentarios sirven únicamente para documentar el programa. No influyen en la ejecución del programa.
- Los bloques de datos de instancia se generan automáticamente al llamar a un bloque de función. No es necesario editarlos.
- El DB 0 está predefinido. Por lo tanto, no se puede generar ningún DB con dicho nombre.

4.5.2 Criterios para ordenar los bloques en el programa

Los bloques deben figurar en el programa en un debido orden conforme a los criterios mencionados a continuación:

- Los bloques llamados preceden a los bloques invocantes.
- Los tipos de datos de usuario (UDT) preceden a los bloques en los que se usan aquellos.
- Los bloques de datos que tienen asignado un tipo de datos de usuario (UDT) siguen al UDT.
- Los bloques de datos preceden a todos los bloques que acceden a ellos.

4.5.3 Uso de direcciones simbólicas

En un programa S7-SCL se emplean operandos tales como señales de entrada/salida, marcas, contadores, temporizadores y bloques. En el programa se pueden asignar direcciones absolutas a estos operandos (p.ej., E1.1, M2.0, FB11), pero la legibilidad de las fuentes S7-SCL aumenta considerablemente si se utilizan símbolos (p.ej., Motor_ON). Después, en el programa de usuario podrá direccionar estos operandos con el símbolo correspondiente.

Símbolos locales y globales

- Utilice símbolos globales para áreas de memoria de la CPU y para nombres de bloques. Los símbolos globales se conocen en todo el programa de usuario, por lo que deben definirse de forma unívoca. La tabla de símbolos se crea con STEP 7.
- Los símbolos locales sólo se conocen en el bloque en cuya área de declaración hayan sido definidos. Es posible asignar nombres a variables, parámetros, constantes y metas de salto, por lo que se pueden utilizar los mismos nombres en bloques distintos para fines diferentes.

Nota

Asegúrese de que los nombes simbólicos sean unívocos y no sean idénticos a palabras clave.

4.6 Editar en fuentes S7-SCL

4.6.1 Deshacer la última acción

Con el comando de menú **Edición >Deshacer** se puede deshacer una o varias acciones.

No todas las operaciones pueden deshacerse. Así p. ej. el comando de menú **Archivo > Guardar** no puede deshacerse.

4.6.2 Restablecer una acción

Después de deshacer una o varias acciones, puede restablecerlas con el comando de menú **Edición > Restablecer** .

4.6.3 Buscar y reemplazar texto

Si desea revisar o modificar una fuente S7-SCL, podrá ahorrar un tiempo muy valioso buscando un texto determinado y sustituyéndolo cuando sea preciso. Se pueden buscar, p.ej., palabras clave, identificadores absolutos, identificadores simbólicos, etc.

Proceda de la siguiente forma:

1. Elija el comando de menú Edición **> Buscar/reemplazar....**
2. Introduzca las opciones en el cuadro de diálogo "Buscar/reemplazar".
3. Inicie el proceso de búsqueda:
   - haciendo clic en el botón de comando "Buscar" para buscar un texto y seleccionarlo, o
   - haciendo clic en el botón de comando "Reemplazar" o "Reemplazar todo" para buscar el texto y sustituirlo por el texto indicado en el cuadro de texto "Reemplazar por".

4.6.4 Seleccionar texto

Se puede seleccionar un texto determinado manteniendo pulsado la tecla del ratón y arrastrando el puntero del ratón por el área de texto deseada.

Además, se puede:

- seleccionar el texto completo de una fuente eligiendo el comando de menú **Edición > Seleccionar todo**,
- seleccionar una sola palabra, haciendo doble clic en la misma,
- seleccionar una línea completa haciendo clic en el margen izquierdo de dicha línea.

Con el comando de menú **Edición > Cancelar selección** puede anular la selección efectuada.

4.6.5 Copiar texto

Con esta función puede copiar partes de un programa o programas completos. El texto copiado se guardará en el portapapeles, pudiéndose insertar en otro lugar cuantas veces se desee.

Proceda de la siguiente forma:

1. Seleccione el texto que va a copiar.
2. Copie el texto:
   - haciendo clic en el botón "Copiar" de la barra de herramientas, o bien
   - eligiendo el comando de menú **Edición > Copiar**.
3. Mueva el punto de inserción hasta el lugar (de la misma aplicación o de otra distinta) en el que vaya a insertar el texto.
4. Inserte el texto:
   - haciendo clic en el botón "Insertar" de la barra de herramientas, o bien
   - eligiendo el comando de menú **Edición > Pegar**.

Nota

El texto no puede copiarse si el comando de menú **Edición > Copiar** no está activado (fondo gris).

Al ejecutar de nuevo los comandos de menú **Edición > Cortar** o **Edición > Copiar** se reemplaza el contenido del portapapeles.

4.6.6 Cortar texto

Con esta función se mueve el texto seleccionado al portapapeles. Normalmente este comando de menú se utiliza conjuntamente con el comando de menú **Edición > Pegar**, que inserta el contenido del portapapeles en la posición actual del punto de inserción.

Proceda de la siguiente forma:

1. Seleccione el texto que va a cortar.
2. Corte el texto:
   - haciendo clic en el botón "Cortar" de la barra de herramientas, o bien
   - eligiendo el comando de menú **Edición > Cortar**.

Nota

- El texto seleccionado no se puede cortar si no está activado (fondo gris) el comando de menú **Edición > Cortar**.
- Con el comando de menú **Edición > Pegar** se inserta el texto cortado en otro lugar (del mismo programa o de otro programa diferente).
- El contenido del portapapeles se conserva hasta que se ejecuten otra vez los comandos de menú **Edición > Cortar** o **Edición > Copiar**.

4.6.7 Borrar texto

Puede suprimir del texto fuente el texto seleccionado.

Proceda de la siguiente forma:

1. Seleccione el texto que va a borrarse.
2. Elija el comando de menú **Edición > Borrar**.

El texto borrado no se inserta en el portapapeles. El texto borrado puede insertarse de nuevo con el comando de menú **Edición > Deshacer** o **Edición > Restablecer**.

4.6.8 Posicionar el cursor en una línea determinada

Las siguientes funciones permiten posicionar el punto de inserción en el lugar deseado.

Posicionamiento en un número de línea determinado:

El cursor se puede posicionar al comienzo de una línea determinada:

1. Elija el comando de menú **Edición > Ir a > Línea**. Se abre el cuadro de diálogo "Ir a"
2. Introduzca el número de línea en el cuadro de diálogo "Ir a".
3. Confirme con "Aceptar".

Posicionamiento en el marcador siguiente/marcador precedente

En caso de que se hayan posicionado marcadores en la fuente, podrá navegar entre los diferentes marcadores:

- Elija el comando de menú **Edición > Ir a > Marcador siguiente / Marcador precedente**.

Posicionamiento en el error siguiente/error precedente del código del programa

Después de compilar se muestran en la ventana "Resultados" todos los errores sintácticos con él número de fila y columna.

S7-SCL ofrece la posibilidad de navegar entre los distintos errores del programa, para así poder eliminar, uno tras otro, todos los errores de la última fase de compilación:

1. Posicione el cursor en una posición cualquiera del texto fuente.
2. Elija el comando de menú **Edición > Ir a > Error precedente / Error siguiente**.

4.6.9 Sangrado de líneas acorde a la sintaxis

Las siguientes funciones permiten estructurar las fuentes S7-SCL alineando las líneas de la fuente:

- **Sangrar automáticamente**
  Cuando la función está activa, cada vez que se cambia de línea, la nueva línea se alinea bajo la línea precedente, respetando el mismo margen.
- **Sangrar palabras clave**
  Cuando la función está activa, se sangran las líneas en las distintas secciones de declaración y dentro de las estructuras de control IF, CASE, FOR, WHILE y REPEAT.

Proceda de la siguiente forma:

1. Elija el comando de menú **Herramientas > Preferencias**.
2. Elija la ficha "Formato" en el siguiente cuadro de diálogo.
3. Asegúrese de que esté activada la opción "Utilizar los siguientes formatos".
4. Active la opción "Sangrar automáticamente" o "Sangrar palabras clave".

4.6.10 Ajuste del estilo y color de la letra

Utilizando distintos estilos y colores de letra para cada uno de los elementos de lenguaje, mejora la legibilidad de la fuente S7-SCL y adquiere un aspecto profesional. Para formatear el texto fuente se dispone de las siguientes funciones:

- **Palabras clave en mayúscula**:
  Cuando la función está activa, las palabras clave definidas como FOR, WHILE, FUNCTION_BLOCK, VAR o END_VAR se escriben en mayúscula.
- **Definición del estilo y color de la letra**:
  Por defecto, están definidos distintos estilos y colores de letra para los distintos elementos del lenguaje tales como operaciones, comentarios o constantes. El usuario puede modificar estos ajustes.
  Colores de letra predeterminados:

Color de letra	Elemento del lenguaje	Ejemplo
Azul	Palabras clave	ORGANIZATION_BLOCK
	Tipos de datos predeterminados	INT
	Identificadores predeterminados	ENO
	Funciones estándar	BOOL_TO_WORD
Ocre	Operaciones	NOT
Rosa	Constantes	TRUE
Verde azulado	Comentario	//... ó (*...*)
Violeta	Símbolos globales (tabla de símbolos) que aparecen entre comillas	"Motor"
Negro	Texto normal	Variables

Proceda de la siguiente forma:

1. Elija el comando de menú **Herramientas > Preferencias**.
2. Seleccione la ficha "Formato" del cuadro de diálogo que aparece a continuación.
3. Asegúrese de que la opción "Utilizar los formatos siguientes al imprimir" esté activa.
4. Realice los ajustes requeridos. Si selecciona el botón de comando "Ayuda" mientras está abierto el cuadro de diálogo, obtendrá instrucciones precisas sobre dicho cuadro de diálogo.

4.6.11 Posicionar marcadores en el texto fuente

Los marcadores ofrecen la posibilidad de navegar rápidamente por una fuente. Los marcadores resultan muy útiles para realizar modificaciones que repercuten en distintos lugares de una fuente. Los marcadores se pueden insertar en cualquier posición de la fuente. En caso de insertar varios marcadores, se puede navegar atrás y adelante entre los diferentes marcadores.

Validez

Los marcadores permanecen activados mientras esté abierta la fuente. No se almacenan con la fuente.

Insertar marcadores

1. Posicione el cursor en la línea que desea marcar.
2. Elija el comando de menú **Edición > Marcadores**.

Navegar entre distintos marcadores

Elija el comando de menú **Edición > Ir a > Marcador siguiente/Marcador precedente**.

Borrar marcadores

Elija el comando de menú **Edición > Borrar todos los marcadores.**

Nota

Las funciones necesarias para trabajar con marcadores están disponibles en la barra de marcadores. Esta barra se visualiza con el comando de menú **Ver > Barra de marcadores**.

4.6.12 Insertar plantillas

4.6.12.1 Insertar plantillas de bloque

Una de las funciones de edición de S7-SCL consiste en insertar plantillas de bloques para OB, FB, FC, DB, DB de instancia, DB procedentes de UDT y UDT. Estas plantillas facilitan la edición y evitan los errores de sintaxis.

Proceda de la siguiente forma:

1. Posicione el cursor detrás de la posición en la que insertará la plantilla.
2. Elija el comando de menú correspondiente: **Insertar > Plantilla de bloque > OB/FB/FC/DB/DB de instancia/DB de UDT/UDT**.

4.6.12.2 Insertar llamadas de bloque

S7-SCL le ofrece soporte a la hora de programar llamadas a bloques. Se pueden llamar los siguientes bloques:

- bloques de función de sistema (SFB) y funciones de sistema (SFC) de las librerías,
- bloques de función y funciones creadas con S7-SCL,
- bloques de función y funciones creados con otros lenguajes de STEP 7.

Proceda de la siguiente forma:

1. Elija el comando de menú Insertar > Llamada de bloque.
2. Elija en el siguiente cuadro de diálogo el bloque SFC, SFB, FC o FB deseado y confirme con "Aceptar".
   S7-SCL copia automáticamente el bloque llamado en el programa S7 e inserta la llamada del bloque así como los parámetros formales del bloque con una sintaxis correcta en la fuente.
3. En caso de haber insertado una llamada de un bloque de función, indique además el DB de instancia.
4. Introduzca ahora los parámetros actuales que deban suministrar valores al bloque. Para facilitarle la selección de un parámetro actual, S7-SCL indica a modo de comentario el tipo de datos requerido.

4.6.12.3 Insertar plantillas para comentarios

Una de las funciones de edición de S7-SCL consiste en insertar plantillas para comentarios. Estas plantillas facilitan la edición y evitan errores de sintaxis.

Proceda de la siguiente forma:

1. Posicione el cursor detrás del encabezamiento del bloque deseado.
2. Elija el comando de menú correspondiente: **Insertar > Plantilla de bloque > Comentario**.

4.6.12.4 Insertar plantillas de parámetros

Una de las funciones de edición de S7-SCL consiste en insertar plantillas para los bloques de declaración de los parámetros. Estas plantillas facilitan la edición y evitan los errores de sintaxis. Los parámetros se pueden declarar en bloques de función y en funciones.

Proceda de la siguiente forma:

1. Posicione el punto de inserción en la tabla de delcaración de un FB o de una FC.
2. Elija el comando de menú **Insertar > Plantilla de bloque > Parámetro**.

4.6.12.5 Insertar estructuras de control

Una de las funciones de edición de S7-SCL es insertar plantillas de estructuras de control para bloques lógicos. Estas plantillas facilitan la notación y el mantenimiento de la sintaxis.

Proceda de la siguiente forma:

1. Posicione el cursor detrás de la posición en la que insertará la plantilla.
2. Elija el comando de menú correspondiente: **Insertar > Estructura de control > IF/CASE/FOR/WHILE/REPEAT.**

4.7 Compilar un programa S7-SCL

4.7.1 Información importante sobre la compilación

Antes de ejecutar el programa o de testearlo es necesario compilarlo. Al iniciar el proceso de compilación se activa el compilador, que tiene las características siguientes:

- Es posible compilar toda una fuente S7-SCL o compilar sólo determinados bloques de forma selectiva.
- Todos los errores sintácticos que se hayan detectado durante la compilación se visualizarán a continuación en una ventana.
- Cada vez que se llama a un bloque de función se genera el correspondiente bloque de datos de instancia, salvo que ya existiera.
- También es posible compilar varias fuentes S7-SCL de una sola pasada creando un archivo de compilación S7-SCL.
- Para ajustar opciones para el compilador elija el comando de menú **Herramientas > Preferencias**.

Una vez que haya creado y compilado sin errores un programa de usuario puede dar por supuesto que el programa es correcto. No obstante, pueden surgir problemas al ejecutar el programa en el PLC. Utilice las funciones de test de S7-SCL para localizar tales errores.

4.7.2 Ajustar el compilador

También es posible adaptar el proceso de compilación a las propias exigencias.

Proceda de la siguiente forma:

1. Elija el comando de menú **Herramientas > Preferencias** para abrir el cuadro de diálogo "Preferencias".
2. Seleccione la ficha "Compilador" o la ficha "Crear bloque".
3. Introduzca las opciones en la ficha.

Opciones de la ficha "Compilador"

Crear object code	Con esta opción puede determinar si desea crear un código ejecutable o no. El proceso de compilación sin esta opción sirve simplemente para una prueba de sintaxis.
Optimizar object code	Si selecciona esta opción, se optimizan la necesidad de memoria y el tiempo de ejecución en AS de los bloques creados. Se recomienda tener esta opción siempre activada dado que la optimización no implica ninguna desventaja que reduzca la funcionalidad del bloque.
Vigilar límites de arrays	Si selecciona esta opción, durante el tiempo de ejecución del programa S7 se comprueba si los índices de array se encuentran dentro del margen admisible - de acuerdo con la declaración de dicho array. Si un índice de array supera el margen admisible, la marca OK (OK flag) cambia a FALSE.
Crear debug information	Esta opción permite realizar una pasada del test con el depurador después de haber compilado el programa y haberlo cargado en la CPU. La memoria necesaria del programa y los tiempos de ejecución en el PLC se prolongan con esta opción.
Activar OK flag	Esta opción permite consultar la OK Flag de sus texto fuente S7-SCL.
Posibilidad de anidar comentarios	Seleccione esta opción si desea anidar varios comentarios en su fuente S7-SCL.
Número máx. de caracteres	Aquí puede reducir la longitud estándar del tipo de datos STRING. En el ajuste predeterminado ésta ocupa 254 caracteres. El ajuste se aplica a todos los parámetros de salida y de entrada/salida así como a los valores de respuesta de las funciones. Tenga en cuenta que el valor ajustado no debe ser menor que las variables STRING utilizadas en el programa.

Opciones de la ficha "Crear bloque"

Sobrescribir bloques	Sobreescribe los bloques ya existentes en la carpeta "Bloques" de un programa S7 si durante el proceso de compilación se crean bloques con el mismo nombre. También se sobreescriben los bloques del mismo nombre que ya estén disponibles en el sistema de destino durante el proceso de carga. Si no ha seleccionado esta opción, deberá confirmar un mensaje para sobreescribir el bloque.
Mostrar advertencias	Determina si se van a visualizar advertencias adicionales sobre los errores detectados tras la compilación.
Errores antes que advertencias	Determina si los errores se visualizarán antes que las advertencias en la ventana de resultados.
Crear datos de referencia	Seleccione esta opción si desea que se generen de forma automática datos de referencia al crear bloques. El comando de menú **Herramientas > Datos de referencia** ofrece la posibilidad de crear los datos de referencia posteriormente o de actualizarlos.
Considerar atributo de sistema 'S7_server'	Seleccione esta opción si desea considerar el atributo de sistema para parámetros "S7-server" al crear bloques. Este atributo se asigna cuando el parámetro es relevante para la configuración de enlaces o mensajes. Incluye el número de enlace o de mensaje. Esta opción prolonga el proceso de compilación.

4.7.3 Compilar el programa

Antes de poder comprobar o ejecutar un programa es necesario compilarlo. Para asegurarse de que está compilando la versión más reciente de su fuente S7-SCL es aconsejable seleccionar el comando de menú **Herramientas > Preferencias** y, antes de compilar, hacer clic en la opción "Guardar" de la ficha "Editor". El comando de menú **Archivo > Compilar** guarda la fuente S7-SCL de forma implícita.

Proceda de la siguiente forma:

1. Guarde la fuente S7-SCL que vaya a compilar.
2. Para generar un programa ejecutable es imprescindible elegir la opción "Crear object code" de la ficha "Compilador" del cuadro de diálogo "Preferencias".
3. Si es necesario, cambie otros ajustes del compilador.
4. Compruebe si la tabla de símbolos correspondiente se encuentra en el mismo directorio del programa.
5. Para iniciar la compilación dispone de las siguientes posibilidades:
   - El comando de menú **Archivo > Compilar** compila la fuente en su totalidad.
   - El comando de menú **Archivo > Compilación parcial** abre un cuadro de diálogo en el que puede seleccionar bloques individuales para la compilación.
6. En el cuadro de diálogo "Resultados" se visualizan todos los errores sintácticos y advertencias que se hayan producido al compilar el programa. Después del proceso de compilación corrija los errores detectados y repita el procedimiento descrito más arriba.

4.7.4 Crear un archivo de compilación

Los archivos de compilación permiten compilar varias fuentes S7-SCL de una carpeta de fuentes de una sola pasada. Introduzca en el archivo de compilación los nombres de las fuentes S7-SCL en el mismo orden en que deben compilarse.

Proceda de la siguiente forma:

1. Abra el cuadro de diálogo "Nuevo" seleccionando el comando de menú **Archivo > Nuevo**.
2. En el cuadro de diálogo "Nuevo":
   - seleccione una carpeta de fuentes de un programa S7 y
   - desactive el tipo de objeto "Archivo de compilación S7-SCL".
3. En el cuadro de entrada de texto (máximo: 24 caracteres) introduzca el nombre del archivo de control, y confirme con "Aceptar".
4. El archivo se creará y se visualizará en una ventana de trabajo para su posterior procesamiento.
   En la ventana de trabajo introduzca en el orden deseado el nombre de las fuentes S7-SCL que va a compilar y guarde el archivo.
5. Inicie el proceso de compilación con el comando de menú **Archivo > Compilar**.

4.7.5 Eliminar errores después de compilar

Todos los errores sintácticos y advertencias que aparecen al compilar se visualizan posteriormente en la ventana "Resultados". La aparición de un error impide que se genere el bloque correspondiente. Por el contrario, cuando aparece una advertencia sí puede generarse un bloque ejecutable, aunque su posterior ejecución en el PLC puede resultar defectuosa.

Para eliminar un error proceda de la siguiente forma:

1. Marque el error y pulse la tecla F1 para obtener una descripción del error y su solución.
2. En caso de que en el mensaje se indique un número de línea (Línxx) y de columna (Colxx), puede localizar el error del texto fuente:
   - haciendo clic en el cuadro de diálogo "Resultados" en el mensaje de error, y abriendo el menú emergente con la tecla derecha del ratón y eligiendo el comando de menú **Mostrar error**,
   - haciendo doble clic en el mensaje de error para posicionar el punto de inserción junto al lugar indicado (línea, columna),
3. Infórmese sobre la sintaxis correcta en la descripción del lenguaje S7-SCL.
4. Corrija el texto fuente.
5. Guardela fuente.
6. Compile de nuevo la fuente.

4.8 Guardar e imprimir una fuente S7-SCL

4.8.1 Guardar una fuente S7-SCL

En S7-SCL se entiende por "guardar" la memorización de archivos fuente. En S7-SCL, los bloques se generan al compilar el archivo fuente y se depositan automáticamente en el directorio correspondiente del programa. También es posible guardar una fuente S7-SCL en su estado actual, sin compilar el objeto. Los errores sintácticos no se guardan.

Proceda de la siguiente forma:

- Seleccione el comando de menú **Archivo > Guardar** o haga clic en el botón "Guardar" de la barra de herramientas.
  La fuente S7-SCL se actualiza.
- Si desea crear una copia de la fuente S7-SCL actual, seleccione el comando de menú **Archivo > Guardar como**. Aparece el cuadro de diálogo "Guardar como", en el que puede indicar el nombre y la ruta en la que debe guardarse el duplicado.

4.8.2 Ajustar el diseño de página

El aspecto de un impreso se puede modificar de la siguiente manera:

- Con el comando de menú **Archivo > Preparar página** se define el formato de página de la impresión.
- Los encabezados y pies de página para los documentos que se van a imprimir se pueden introducir eligiendo el comando de menú **Archivo > Preparar página** y seleccionando en el cuadro de diálogo que aparece a continuación la ficha "Encabezado/Pie de página".
- Con el comando de menú **Archivo > Instalar impresora** podrá realizar otros ajustes de impresión.

Atención

La orientación de las páginas se ajusta en el cuadro de diálogo "Preparar página". Los ajustes del cuadro de diálogo "Instalar impresora" no son relevantes para la impresión de fuentes S7-SCL.

- Con el comando de menú **Archivo > Presentación preliminar** se comprueban los ajustes efectuados antes de enviar el documento a la impresora.

4.8.3 Imprimir una fuente S7-SCL

Normalmente se imprime la fuente S7-SCL de la ventana de trabajo activa; es decir, para poder imprimir una fuente S7-SCL dicha fuente debe estar abierta.

Proceda de la siguiente forma:

1. Active la ventana de trabajo de la fuente S7-SCL cuyo contenido desea imprimir.
2. Abra el cuadro de diálogo "Imprimir":
   - haciendo clic en el botón "Imprimir", o bien
   - eligiendo el comando de menú **Archivo > Imprimir**.
3. En el cuadro de diálogo "Imprimir" seleccione las opciones de impresión (p.ej., el área de impresión o el número de copias).
4. Confirme con "Aceptar".

4.8.4 Ajuste de las opciones de impresión

Para formatear la impresión puede utilizar las siguientes funciones:

- **Cambiar página al principio/al final del bloque**
  Cuando está activada esta función, cada bloque comenzará a imprimirse en una página nueva o finalizará con un cambio de página.
- **Imprimir con números de línea**
  Cuando está activada esta función, se imprimen números de línea al principio de cada línea.
- **Imprimir en color in Farbe**
  Cuando está activada esta función, al imprimir se imprime en los colores utilizados en la fuente.
- **Imprimir en letra**
  Por defecto está definido el tipo de letra Courier, tamaño 8 para todo el texto. Este tipo de letra ofrece los mejores resultados al imprimir.
- **Estilo**
  Puede definir distintos estilos de letra para los elementos de lenguaje individuales. Se pueden seleccionar individualmente los siguientes elementos:

Elemento del lenguaje	**Ejemplo**
Texto normal	
Palabras clave	ORGANIZATION_BLOCK
Identificador de tipos de datos predeterminados	INT
Identificador predeterminado	ENO
Identificador de funciones estándar	BOOL_TO_WORD
Operaciones	NOT
Constantes	TRUE
Bloque de comentario	(* *)
Línea de comentario	//...
Símbolos globales (tabla de símbolos) que aparecen entre comillas	"Motor"

Proceda de la siguiente forma:

1. Elija el comando de menú **Herramientas > Preferencias**.
2. Seleccione la ficha "Imprimir" del cuadro de diálogo que aparece a continuación.
3. Asegúrese de que la opción "Utilizar los formatos siguientes al imprimir" esté activa.
4. Realice los ajustes que desee. Si selecciona el botón de comando "Ayuda" mientras está abierto el cuadro de diálogo, obtendrá instrucciones precisas sobre dicho cuadro de diálogo.

4.9 Cargar los programas creados

4.9.1 Borrado total de la memoria de la CPU

Con la función Borrado total se puede borrar online todo el programa de usuario de una CPU.

Proceda de la siguiente forma:

1. Elija el comando de menú **Sistema de destino > Estado operativo** y active el estado operativo STOP de la CPU.
2. Elija el comando de menú **Sistema de destino > Borrado total**.
3. Confirme la acción en el cuadro de diálogo que aparece a continuación.

Advertencia

La CPU se resetea.

Se borran todos los datos de usuario.

La CPU deshace todas las conexiones existentes.

Si hay insertada una Memory Card después del borrado total la CPU copia el contenido de la Memory Card en la memoria de carga interna.

4.9.2 Cargar programas de usuario en la CPU

Requisitos

Al compilar una fuente S7-SCL, se generan bloques a partir de la fuente que se guardan en la carpeta "Bloques" del programa S7.

Primero se buscan los bloques que se llaman en primera línea desde los bloques S7-SCL, luego se copian automáticamente en la carpeta "Bloques" y a continuación se registran en la lista de carga.

Los demás bloques del programa de usuario se cargan desde el Administrador SIMATIC de la unidad de programación a la CPU.

Para poder realizar la carga tiene que existir una conexión entre la PG y la CPU. El módulo de la CPU debe tener asignado un programa de usuario online en el Administrador SIMATIC.

Procedimiento

Una vez que haya compilado la fuente puede comenzar el proceso de carga mediante cualquiera de las siguientes opciones:

- El comando de menú **Archivo > Cargar** carga todos los bloques que contiene la fuente, así como todos los bloques llamados en el primer nivel.
- El comando de menú **Archivo > Carga parcial** abre un cuadro de diálogo en el que puede seleccionar bloques individuales para la carga.

Los bloques se transfieren a la CPU. Caso que uno de los bloques ya exista en la RAM de la CPU, indique si el bloque debe sobreescribirse o no.

Nota

Conviene cargar el programa de usuario en el estado operativo STOP, puesto que al sobreescribir un programa antiguo en el estado operativo RUN pueden generarse errores.

4.10 Testear los programas creados

4.10.1 Funciones de test de S7-SCL

Las funciones de test de S7-SCL ofrecen la posibilidad de controlar la ejecución de un programa en la CPU y detectar posibles errores. Los errores de sintaxis se visualizan durante la compilación. Los errores de tiempo de ejecución que aparecen al ejecutar el programa también se visualizan mediante alarmas del sistema; los errores lógicos de programación se detectan con las funciones de test.

Funciones de test de S7-SCL

S7-SCL ofrece las siguiente funciones de test:

- Observar
  Esta función permite indicar nombres y valores actuales de variables en la fuente S7-SCL. Al ejecutar el test se visualizan los valores de las variables y los parámetros de dicha área en orden cronológico se actualizan de manera cíclica.
- Test con puntos de parada/paso a paso
  Esta función permite posicionar puntos de parada y realizar un test paso a paso. Ello permite ejecutar el algoritmo del programa instrucción a instrucción y visualizar cómo varían los valores de las variables procesadas.

Cuidado

Un test con la instalación en marcha puede causar graves daños materiales y personales en caso de que se produzcan averías de funcionamiento o errores en el programa.

Antes de ejecutar las funciones de test asegúrese de que no pueden producirse estados peligrosos.

Requisitos para la ejecución del test

- El programa tiene que haberse compilado con las opciones "Crear object code" y "Crear debug info".
  Las opciones se seleccionan en la ficha "Compilador" del cuadro de diálogo "Preferencias". Para abrir dicho cuadro de diálogo, seleccione el comando de menú **Herramientas > Preferencias**.
- Se dispone de datos de referencia actuales.
  Los datos de referencia se crean automáticamente al compilar si está activada la opción "Crear debug info".
- Entre la PG / el PC y la CPU debe existir una conexión online.
- El programa ha sido cargado en la CPU.
  Para ello se debe ejecutar el comando de menú **Sistema de destino > Cargar**.

4.10.2 Información importante sobre la función de test "Observar"

Al observar un programa se puede comprobar un grupo de instrucciones, el cual se denomina área de observación. Este grupo de instrucciones se denomina "área de observación". Durante la ejecución del test se visualizan los valores de las variables y parámetros de dicha área en orden cronológico y se actualizan de manera cíclica. Si el área de observación se encuentra en una sección del programa que se recorre en cada uno de los ciclos, generalmente los valores de las variables no pueden registrarse a partir de ciclos consecutivos.

Los valores modificados durante la ejecución actual se pueden diferenciar con colores de los que no se han modificado.

Área de observación

El volumen de instrucciones que pueden observarse está limitado por la capacidad de la CPU que esté conectada. Como las instrucciones S7-SCL del código fuente se corresponden con una cantidad indefinida de instrucciones en código máquina, la longitud del área de observación es variable, y será calculada y seleccionada por el depurador S7-SCL cuando se seleccione la primera instrucción del área de observación deseada.

Además existe la posibilidad de definir un área de observación determinada. Marque püara ello las instrucciones que desea observar en la fuente.

Modos de test

La mayoría de las veces esto prolonga los tiempos de ciclo. Para poder actuar sobre esta prolongación S7-SCL ofrece dos modos diferentes:

Modo de funcionamiento	**Explicación**
Modo Test	En el modo "Test" el área de observación sólo está limitada por las características de la CPU que está conectada. Se pueden utilizar todas las funciones de test sin ningún tipo de restricción. El tiempo de ciclo de la CPU se puede prolongar considerablemente ya que, p.ej, se averigua el estado de las instrucciones en bucles programados durante cada ejecución.
Modo Proceso	En el modo "Proceso" el depurador S7-SCL restringe el tamaño máximo del área de observación de forma que durante el test los tiempos de ciclo no superen o superen mínimamente los tiempos de ejecución reales del proceso.

Limitaciones

Tenga en cuenta las siguientes limitaciones de la función "Observar":

- Las variables de un tipo de datos superior no se muestran en su totalidad. Los elementos de dichas variables se pueden observar siempre que sean de un tipo de datos simple.
- Las variables del tipo DATE_AND_TIME y STRING así como los parámetros del tipo BLOCK_FB, BLOCK_FC, BLOCK_DB, BLOCK_SDB, TIMER y COUNTER no se muestran.
- Los accesos a bloques de datos del tipo <simbolo>.<direccion_absoluta> no se muestran (p.ej. datos.DW4).

4.10.3 Información importante sobre "Test con puntos de parada/modo Paso a paso"

"Test con puntos de parada" hace que el test se ejecute paso a paso. Ello permite ejecutar el algoritmo del programa instrucción a instrucción y visualizar cómo varían los valores de las variables procesadas.

Una vez insertados los puntos de parada, es posible ejecutar el programa hasta un punto de parada y comenzar a observar el programa paso a paso desde dicho punto

Funciones de ejecución paso a paso:

Una vez activada la función "Test con puntos de parada" se pueden ejecutar los siguientes pasos:

- Ejecutar instrucción siguiente
  Se ejecuta la instrucción actualmente seleccionada.
- Continuar
  Continúa hasta el siguiente punto de parada que esté activo.
- Ejecutar hasta selección
  Continúa hasta el punto de la fuente que usted defina.
- Ejecutar llamada
  Salto a un bloque subyacente S7-SCL o llamada a un bloque subyacente.

Puntos de parada:

Los puntos de parada se pueden definir en cualquier lugar del área de instrucciones del texto fuente.

Para que se transfieran los puntos de parada al sistema de automatización y se activen es preciso ejecutar el comando de menú **Test > Activar puntos de parada**.

El número máximo de puntos de parada a activar depende de la CPU:

Requisitos:

No se puede modificar la fuente abierta con el editor.

4.10.4 Pasos para observar

Después de cargar el programa compilado en el sistema de destino, puede testear el programa en el modo "Observar".

Nota

El volumen de instrucciones que se puede testear (el área de observación mayor posible) depende de la capacidad de la CPU que esté conectada.

Proceda de la siguiente forma:

1. Asegúrese de que se cumplan los requisitos para la ejecución del test y de que la CPU se encuentre en el estado operativo RUN o RUN-P.
2. Seleccione la ventana que contiene la fuente del programa que va a someterse a test.
3. Si desea cambiar el modo de funcionamiento preajustado (Proceso), seleccione el comando de menú **Test > Modo de funcionamiento > Test**.
4. Defina el área de observación. Para ello dispone de dos posibilidades:
   - Posicione el cursor en aquella línea del texto fuente que contiene la primera instrucción del área que va a someterse a test. S7-SCL seleccionará entonces un área de observación lo mayor posible a partir de la posición del cursor.
   - Marque en el texto fuente las instrucciones que desea observar.
5. Asegúrese de que no pueden producirse estados peligrosos.
6. Elija el comando de menú **Test > Observar**.
7. Desactive el comando de menú **Test > Observar** para interrumpir la ejecución del test.
8. Elija el comando de menú **Test >Finalizar test** para finalizar el test.

Resultado

Se determina el área de observación máxima posible, la cual se representa mediante una barra gris en el borde izquierdo de la ventana. La ventana se divide en dos partes, y a la derecha aparecen los nombres y valores actuales de las variables situadas en el área de observación.

Personalizar la función de observación

Existen las siguientes posibilidades para adaptar la función de observación a sus necesidades:

- Defina un entorno de llamada para el bloque que desea observar.
- Seleccione el comando de menú **Ver > Representación simbólica** para visualizar u ocultar los nombres simbólicos que aparecen en la tabla de símbolos del programa utilizado.
- Seleccione el comando de menú **Herramientas > Preferencias** y realice los ajustes que desee en la ficha "Formato" con respecto al color en el que se van a representar los valores.

4.10.4.1 Definir un entorno de llamada para los bloques

Entorno de llamada

Para definir el área de observación con mayor exactitud, es posible definir un entorno de llamada para los bloques que se van a observar. Consiste en definir que un bloque sea observado solamente si se cumple una de las siguientes condiciones:

Condición	**Opción posible**
El bloque es llamado desde un determinado bloque de orden superior.	Ruta de llamada
El bloque es llamado junto con un determinado bloque de datos.	Bloque de datos global y / o bloque de datos instancia

Proceda del siguiente modo para definir una ruta de llamada

1. Elija el comando de menú **Test > Entorno de llamada de bloques**.

   En el cuadro de diálogo que aparece a continuación se muestra una lista de los bloques existentes.
2. Elija un bloque de la lista.
3. Active la opción "Activar ruta de llamada".

   En la ventana inferior se representan gráficamente las rutas de llamada posibles.
4. Elija la ruta de llamada deseada.

Proceda del siguiente modo para definir un bloque de datos

1. Elija el comando de menú **Test > Entorno de llamada de bloques**.

   En el cuadro de diálogo que aparece a continuación se muestra una lista de los bloques existentes.
2. Elija un bloque de la lista.
3. Active la opción "Bloques de datos abiertos".
4. Introduzca el número del DB deseado.

Nota

Una vez definido el entorno de llamada, proceda de la manera siguiente para iniciar la observación:

1. Posicione el cursor en el bloque que desea observar, es decir, el bloque para el que ha definido el entorno de llamada.
2. Elija el comando de menú **Test > Observar**.

   Se inicia la función de observación. El bloque es observado cuando se cumplen todas las condiciones de llamada que ha definido.

4.10.5 Pasos para realizar el test con puntos de parada

4.10.5.1 Definir puntos de parada

Posicione y defina los puntos de parada de la siguiente forma:

1. Abra la fuente que vaya a testear.
2. Muestre la barra de puntos de parada con el comando de menú **Ver > Barra de puntos de parada**.
3. Posicione el cursor en la posición deseada y elija el comando de menú **Test > Posicionar punto de parada** o el botón correspondiente de la barra de puntos de parada. Los puntos de parada se representan en el borde izquierdo de la ventana en forma de círculo rojo.
4. Si lo desea, elija el comando de menú **Test > Editar puntos de parada** y defina un entorno de llamada. El entorno de llamada determina si un punto de parada estará activo solamente cuando el bloque en el que se encuentra
   - es llamado por un determinado bloque superior y/o
   - es llamado junto con un bloque de datos/bloque de datos instancia determinado.

4.10.5.2 Iniciar el test con puntos de parada

Después de cargar el programa compilado en el sistema de destino y de haber posicionado los puntos de parada puede testear el programa en el modo "Test con puntos de parada".

Proceda de la siguiente forma:

1. Abra la fuente S7-SCL del programa que desee testear.
2. Asegúrese de que no se puedan producir estados peligrosos y de que la CPU se encuentre en el estado operativo RUN-P.
3. Seleccione el comando de menú **Test > Activar puntos de parada** y a continuación el comando de menú **Test > Observar**.
   Resultado: la ventana se divide en dos partes en vertical. El programa se ejecuta hasta el siguiente punto de parada. Cuando se llega a este punto, la CPU cambia al estado operativo PARADA y el punto de parada rojo se marca con una flecha amarilla.
4. Seguidamente
   - elija el comando de menú **Test > Continuar** o haga clic en el botón "Continuar" de la barra de herramientas.
     La CPU pasará al estado operativo RUN. Cuando se alcance el siguiente punto de parada activo se parará visualizando el punto de parada en la parte izquierda de la ventana.
   - elija el comando de menú **Test > Ejecutar instrucción siguiente** o haga clic en el botón "Ejecutar instrucción siguiente" de la barra de herramientas.
     La CPU pasa a estado operativo RUN. Después de ejecutar la instrucción seleccionada se vuelve a parar, y en la mitad derecha de la ventana se visualiza el contenido de las variables recién procesadas.

- elija el comando de menú **Test > Ejecutar hasta selección** o active el botón "Ejecutar hasta selección" de la barra de herramientas.
  La CPU pasará al estado operativo RUN. Al alcanzarse el punto seleccionado del programa se vuelve a parar.
- cuando el programa se detenga en una línea que contenga una llamada a un bloque, elija el comando de menú **Test > Ejecutar llamada**.
  Si el bloque subyacente se creó con S7-SCL, se abrirá y procesará en el modo Test. Después de procesarlo, el programa retornará a la posición de llamada.
  Si el bloque se creó en otro lenguaje, no se tendrá en cuenta la llamada y se marcará la siguiente línea del programa.

Nota

Los comandos de menú **Test > Ejecutar instrucción siguiente** o **Test > Ejecutar hasta selección** posicionan y activan automáticamente un punto de parada. Cuando seleccione dichas funciones, observe que no se haya alcanzado el número máximo de puntos de parada activos que admite la CPU en cuestión.

4.10.5.3 Finalizar el test con puntos de parada

Para retoronar a la ejecución normal del programa proceda del siguiente modo:

- Desactive el comando de menú **Test > Activar puntos de parada**, para interrumpir el test; o
- Elija el comando de menú **Test > Finalizar test**, para terminar el test.

4.10.5.4 Activar, desactivar y anular puntos de parada

Puede activar / desactivar y anular por separado los puntos de parada establecidos:

1. Elija el comando de menú Test > Editar puntos de parada.
2. En el siguiente cuadro de diálogo puede
   - Activar o desactivar los puntos de parada que usted elija marcándolos con una marca de verificación.
   - Anular selectivamente puntos de parada.

Para anular todos los puntos de parada, seleccione el comando de menú **Test > Borrar todos los puntos de parada**.

4.10.5.5 Definir un entorno de llamada para los puntos de parada

Entorno de llamada

Al definir un entorno de llamada se establece que un punto de parada será válido solamente si se cumple una de las siguientes condiciones:

Condición	Opción posible
El bloque en el que se encuentra el punto de parada es llamado por un determinado bloque de orden superior.	Ruta de llamada
El bloque en el que se encuentra el punto de parada se llama junto con un determinado bloque de datos.	Bloque de datos global y / o bloque de datos instancia

Proceda del siguiente modo para definir una ruta de llamada

1. Elija el comando de menú **Test > Editar puntos de parada**.

   En el cuadro de diálogo que aparece a continuación se muestra una lista de los puntos de parada existentes.

2. Elija un punto de parada de la lista.
3. Active la opción "Activar ruta de llamada".

   En la ventana inferior se representan gráficamente las rutas de llamada posibles.

4. Elija la ruta de llamada deseada.

Proceda del siguiente modo para definir un bloque de datos

1. Elija el comando de menú **Test > Editar puntos de parada**.
   En el cuadro de diálogo que aparece a continuación se muestra una lista de los puntos de parada disponibles.
2. Elija un punto de parada de la lista.
3. Active la opción "Bloques de datos abiertos".
4. Introduzca el número del DB deseado.

4.10.5.6 Test en modo Paso a Paso

Proceda de la siguiente forma:

1. Posicione un punto de parada delante de la línea de instrucciones a partir de la que desee comprobar el programa en modo Paso a paso.
2. Seleccione el comando de menú **Test > Activar puntos de parada**.
3. Ejecute el programa hasta llegar a este punto de parada.
4. Para ejecutar la siguiente instrucción, seleccione el comando de menú **Test > Ejecutar siguiente instrucción**.

- Si se trata de una llamada de bloque, se ejecutará la llamada y se pasará a la primera instrucción que aparezca después de la llamada de bloque.
- Con el comando de menú **Test > Ejecutar llamada** se pasa al bloque. Desde allí puede continuar con el test paso a paso o posicionar puntos de parada. Al final del bloque se retorna a la siguiente instrucción que aparezca después de la llamada de bloque.

4.10.6 Uso de las funciones de test de STEP 7

4.10.6.1 Crear y/o mostrar los datos de referencia

Para facilitar el test y las modificaciones del programa de usuario puede crear y evaluar datos de referencia.

Pueden mostrarse los siguientes datos de referencia:

- la estructura del programa de usuario
- la lista de referencias cruzadas
- el plano de ocupación
- la lista de operandos no utilizados
- la lista de operandos sin símbolo

Creación de datos de referencia

Para generar los datos dispone de las siguientes posibilidades:

- Con el comando de menú **Herramientas > Mostrar datos de referencia** puede generar o actualizar los datos que precise, y visualizarlos.
- Puede limitar la cantidad de datos de referencia mostrados por medio de filtros y acelerar considerablemente de esta forma la creación y visualización de los datos. Para ello, seleccione el comando de menú **Herramientas > Datos de referencia > Filtrar**.
- Con el comando de menú **Herramientas > Preferencias** puede establecer que los datos de referencia se creen automáticamente al compilar una fuente. Para ello, en la ficha "Crear bloque" seleccione la entrada "Crear datos de referencia".
  No obstante, la creación automática de datos de referencia prolonga el proceso de compilación.

4.10.6.2 Observar y forzar variables

Al efectuar un test con la función "Observar/forzar variables" puede:

- visualizar (observar) los valores actuales de datos globales de su programa de usuario;
- asignar valores fijos (forzar) a las variables del programa de usuario.

Proceda de la siguiente forma:

- Seleccione el comando de menú **Sistema de destino > Observar/forzar variables**.
- Cree la tabla de variables (VAT) en la ventana que se muestra a continuación.
  Si desea forzar variables, indique además los valores deseados.
- Determine los puntos de disparo y las condiciones de disparo.

Cuidado

Si modifica valores de las variables con la instalación en marcha pueden causarse graves daños materiales o personales en caso de que existan averías de funcionamiento o errores en el programa. Antes de ejecutar esta función asegúrese de que no puedan producirse estados peligrosos.

4.10.6.3 Comprobar la coherencia de los bloques

Nota

Esta función está disponible a partir de la versión 5.3 SP2 de STEP 7.

En caso de modificar una fuente S7-SCL, es posible que deban adaptarse los bloques que a los que se hace referencia en la misma. De lo contrario podría aparecer incoherencias en el programa. Aunque las indicaciones de fecha y hora de fuentes y bloques no coincidan, pueden aparecer incoherencias.

Mediante la función de STEP 7 "Comprobar coherencia de bloque" se pueden localizar dichas incoherencias y así eliminar más rápidamente los errores.

Después de modificar el programa, se inicia la comprobación de coherencia para todas las fuentes S7-SCL del proyecto. En caso de que se detecten incoherencias que no se puedan eliminar automáticamente, la función le conducirá a aquellas posiciones de la fuente que deben ser modificadas. Ahí se realizan las modificaciones o correcciones necesarias. Paso a paso se van eliminando todas las incoherencias.

Requisitos

- Su equipo tiene que tener instalado STEP 7 V5.3 SP2 o superior.
- Una fuente a la que se deba aplicar la función "Comprobar coherencia de bloque" debe haber sido compilada con S7-SCL V5.3 SP1 o superior.
- S7-SCL debe estar instalado en el equipo en el que se va a comprobar la coherencia.

Procedimiento

1. Abra el Administrador SIMATIC.
2. Seleccione la carpeta "Bloques".
3. Elija el comando de menú **Edición > Comprobar coherencia de bloques**.
4. Elija el comando de menú **Ver > Mostrar referencias de fuentes S7-SCL**.

Resultado

STEP 7 comprueba las indicaciones de fecha y hora y los interfaces de todos los bloques y de las fuentes correspondientes en la carpeta actual y notifica los siguientes conflictos:

- Conflictos de fecha y hora entre bloques y fuentes S7-SCL.
- Referencias entre los distintos bloques y los conflictos resultantes de interfaces.

Cuando se detecta una incoherencia, se reinicia la compilación de la fuente correspondiente. Si la fuente contiene varias fuentes de bloques, se vuelven a generar todos los bloques creados a partir de esta fuente. En la compilación se aplican las opciones de compilación que estén ajustadas en ese momento.

Nota

Para más información sobre esta función, consulte la Ayuda de STEP 7.

4.11 Visualizar y cambiar las propiedades de la CPU

4.11.1 Visualizar y cambiar el estado operativo de la CPU

Esta función permite consultar y modificar el estado operativo actual de la CPU siempre y cuando ésta esté conectada.

Proceda de la siguiente forma

1. Elija el comando de menú **Sistema de destino > Estado operativo**.
2. En el cuadro de diálogo siguiente elija uno de los siguientes estados operativos:

- Rearranque completo (rearranque en caliente)
- Rearranque en frío
- Arranque en caliente
- STOP

Advertencia

Modificar el estado operativo con la instalación en marcha puede causar graves daños materiales y personales en caso de que se produzcan averías de funcionamiento o errores en el programa.

Antes de ejecutar esta función asegúrese de que no puedan producirse estados peligrosos.

4.11.2 Visualizar y ajustar la fecha y la hora de la CPU

Esta función permite consultar y modificar la hora de la CPU, siempre y cuando ésta esté conectada.

Proceda de la siguiente forma

1. Elija el comando de menú **Sistema de destino > Ajustar la hora**.
2. En el siguiente cuadro de diálogo, introduzca la fecha y la hora para el reloj de tiempo real de la CPU.

Si la CPU no dispone de reloj de tiempo real, en el cuadro de diálogo se indicará "00:00:00" para la hora y "00.00.00" para la fecha, y no podrá realizar modificaciones.

4.11.3 Visualizar los datos de la CPU

Esta función permite visualizar los siguientes datos de la CPU, siempre y cuando ésta esté conectada:

- la familia de sistemas, el modelo de CPU, el número de referencia y la versión de la CPU,
- el tamaño de la memoria de trabajo y de la memoria de carga y configuración máxima posible de la memoria de carga,
- el área de direccionamiento y el número de entradas y salidas, temporizadores, contadores y marcas,
- el número de datos locales con los que puede trabajar la CPU,
- un mensaje que indica si la CPU es capaz de trabajar en modo multiprocesador o no (depende de la CPU en cuestión).

Proceda de la siguiente forma:

1. Elija el comando de menú **Sistema de destino > Información del módulo**.
2. En el siguiente cuadro de diálogo elija la ficha "General".

4.11.4 Leer el búfer de diagnóstico de la CPU

El búfer de diagnóstico permite determinar la causa del estado operativo STOP o reconstruir la aparición de los distintos eventos de diagnóstico.

Es necesario que exista una conexión con la CPU.

Proceda de la siguiente forma

1. Elija el comando de menú **Sistema de destino > Información del módulo**.
2. En el cuadro de diálogo siguiente elija la ficha "Búfer de diagnóstico".

4.11.5 Visualizar y comprimir la memoria de usuario de la CPU

Esta función permite visualizar el grado de utilización de la memoria de la CPU y reorganizarla, en caso necesario. Es necesaria cuando el tamaño de la mayor área de memoria contigua libre no es suficiente para cargar en la CPU un nuevo objeto desde la unidad de programación (PG).

Es necesario que exista una conexión con la CPU.

Proceda de la siguiente forma:

1. Elija el comando de menú **Sistema de destino > Información del módulo**.
2. En el siguiente cuadro de diálogo elija la ficha "Memoria".

4.11.6 Visualizar el tiempo de ciclo de la CPU

Dentro de los dos valores límite parametrizados se representan los siguientes tiempos:

- Duración del ciclo más largo desde el último cambio de STOP a RUN
- Duración del ciclo más corto desde el último cambio de STOP a RUN
- Duración del último ciclo

Si la duración del último ciclo se aproxima mucho al tiempo de vigilancia, es posible que lo sobrepase y que la CPU cambie al estado operativo STOP. El tiempo de ciclo se puede prolongar por ejemplo cuando se comprueban bloques en el estado de programa. Es necesario que exista una conexión con la CPU para visualizar los tiempos de ciclo del programa.

Proceda de la siguiente forma:

1. Elija el comando de menú **Sistema de destino > Información del módulo**.
2. En el cuadro de diálogo siguiente elija la ficha "Tiempo de ciclo".

4.11.7 Visualizar el sistema de reloj de la CPU

El sistema de tiempo de la unidad central de procesamiento (CPU) ncluye datos acerca del reloj interno y de la sincronización de la hora entre varias CPU.

Es necesario que exista una conexión con la CPU.

Proceda de la siguiente forma:

1. Elija el comando de menú **Sistema de destino > Información del módulo**.
2. En el cuadro de diálogo siguiente elija la ficha "Sistema de reloj".

4.11.8 Visualizar los bloques de la CPU

Es posible visualizar online los bloques disponibles para la CPU.

Es necesario que exista una conexión con la CPU.

Proceda de la siguiente forma:

1. Elija el comando de menú **Sistema de destino > Información del módulo**.
2. En el siguiente cuadro de diálogo seleccione la ficha "Datos característicos/Bloques".

4.11.9 Visualizar los datos de comunicación de la CPU

Es posible hacerse mostrar online datos sobre la velocidad de transferencia ajustada y la velocidad de transferencia máxima de cada CPU, así como sobre las conexiones y el grado de comunicación.

Es necesario que exista una conexión con la CPU.

Proceda de la siguiente forma:

1. Elija el comando de menú **Sistema de destino > Información del módulo**.
2. En el cuadro de diálogo siguiente elija la ficha "Comunicación".

4.11.10 Visualizar las pilas de la CPU

Esta ficha visualiza online datos sobre el contenido de las pilas de cualquier unidad central de procesamiento (CPU). Para ello, la CPU tiene que estar en estado operativo STOP o tiene que haber alcanzado un punto de parada.

Esta función resulta muy útil para encontrar errores (p.ej., cuando esté testeando los bloques). Si la CPU ha cambiado al estado operativo STOP, en la pila de interrupción (USTACK) puede visualizar el punto de interrupción con las indicaciones actuales y los contenidos de los ACU para detectar la causa del error (p.ej., del error de programación).

Es necesario que exista una conexión con la CPU.

Proceda de la siguiente forma:

1. Elija el comando de menú **Sistema de destino > Información del módulo**.
2. En el siguiente cuadro de diálogo elija la ficha "Pilas".

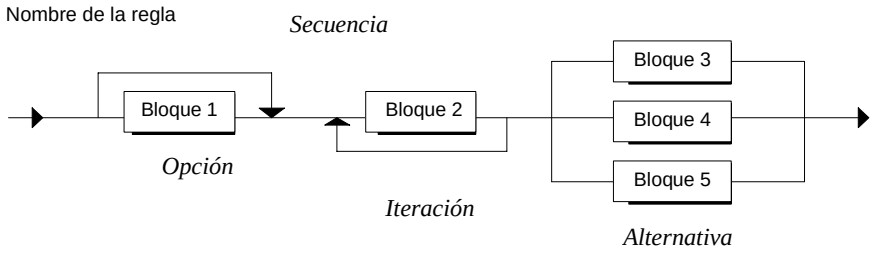
5 Conceptos generales de S7-SCL

5.1 Interpretación de los diagramas sintácticos

La descripción del lenguaje se realiza en cada capítulo mediante diagramas sintácticos. Estos diagramas proporcionan una visión general de la estructura sintáctica de S7-SCL. En el capítulo "Descripción del lenguaje" encontrará una relación completa de los diagramas con sus respectivos elementos.

¿Qué es un diagrama sintáctico?

El diagrama sintáctico es una representación gráfica de la estructura del lenguaje. La estructura se describe mediante una secuencia de reglas. Algunas reglas pueden basarse en reglas que hayan sido introducidas anteriormente.

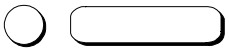


El diagrama sintáctico se lee de izquierda a derecha. Deben tenerse en cuenta las siguientes estructuras:

- Secuencia: Secuencia de bloques
- Opción : Rama que se puede saltar
- Iteración: Repetición de ramas
- Alternativa: Ramificación

¿Qué tipos de bloques hay?

Un bloque es un elemento básico o un elemento que, a su vez, puede componerse de bloques. La figura siguiente muestra los tipos de símbolos que corresponden a los bloques:



Elemento básico, que no precisa mayor explicación.
En este caso se trata de caracteres im primibles y caracteres especiales, pala bras clave e identificadores predefinidos. Los datos referidos a estos bloques de ben aceptarse sin cambios.

Elemento compuesto, descrito por otros diagramas sintácticos.

¿Qué significa libertad de formato?

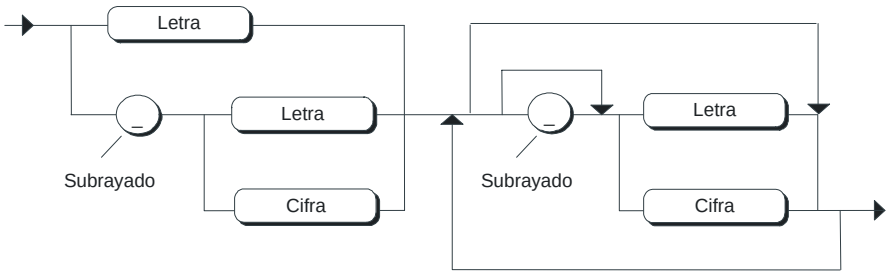
Al introducir textos fuente deben respetarse tanto las **reglas sintácticas** como las **reglas léxicas**.

Las reglas tanto sintácticas como léxicas están descritas detalladamente en el capítulo "Descripción del lenguaje". Libertad de formato significa que entre los bloques de reglas se pueden introducir caracteres de formateo como espacios en blanco, tabuladores, cambios de página y comentarios.

En las reglas léxicas no existe libertad de formato. Si utiliza una regla léxica debe aceptar las indicaciones sin cambios.

Regla léxica

IDENTIFICADOR



Algunos ejemplos válidos de la regla expuesta serían:

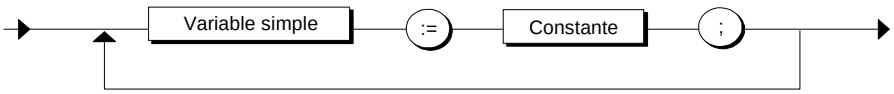
```
R_REGULADOR
_A_ARRAY
_100_3_3_10
```

Algunos ejemplos no válidos debido a los motivos expuestos serían:

```
1_1AB
RR__20
*#AB
```

Regla sintáctica

En las reglas sintácticas existe libertad de formato.



Algunos ejemplos válidos de la regla expuesta serían:

```
VARIABLE_1 := 100;      PULSADOR:=FALSE;
VARIABLE_2 := 3.2;
```

5.2 Juego de caracteres

Letras y cifras

S7-SCL utiliza los siguientes caracteres del juego de caracteres ASCII:

- las letras (mayúsculas y minúsculas) de la A a la Z,
- la numeración arábiga, de 0 a 9,
- los espacios en blanco: el espacio en blanco propiamente dicho (valor 32 de ASCII) y todos los caracteres de control (valores 0-31 de ASCII), incluido el carácter de final de línea (valor 13 de ASCII).

Otros caracteres

Los siguientes caracteres tienen un significado fijo en S7-SCL:

+	-	*	/	=	<	>	[	]	(	)
:	;	$	#	"	'	{	}	%	.	,

Nota

En el capítulo "Descripción del lenguaje" encontrará una lista detallada de todos los caracteres que se pueden utilizar así como los datos para interpretar estos caracteres en S7-SCL.

5.3 Palabras reservadas (palabras clave)

Las palabras reservadas son palabras clave que sólo se pueden utilizar con su significado predefinido. No se distingue entre mayúsculas y minúsculas.

Palabras clave en S7-SCL

AND	END_CASE	ORGANIZATION_BLOCK
ANY	END_CONST	POINTER
ARRAY	END_DATA_BLOCK	PROGRAM
AT	END_FOR	REAL
BEGIN	END_FUNCTION	REPEAT
BLOCK_DB	END_FUNCTION_BLOCK	RETURN
BLOCK_FB	END_IF	S5TIME
BLOCK_FC	END_LABEL	STRING
BLOCK_SDB	END_TYPE	STRUCT
BLOCK_SFB	END_ORGANIZATION_BLOCK	THEN
BLOCK_SFC	END_REPEAT	TIME
BOOL	END_STRUCT	TIMER
BY	END_VAR	TIME_OF_DAY
BYTE	END_WHILE	TO
CASE	ENO	TOD
CHAR	EXIT	TRUE
CONST	FALSE	TYPE
CONTINUE	FOR	VAR
COUNTER	FUNCTION	VAR_TEMP
DATA_BLOCK	FUNCTION_BLOCK	UNTIL
DATE	GOTO	VAR_INPUT
DATE_AND_TIME	IF	VAR_IN_OUT
DINT	INT	VAR_OUTPUT
DIV	LABEL	VOID
DO	MOD	WHILE
DT	NIL	WORD
DWORD	NOT	XOR
ELSE	OF	Nombres de las funciones estándar
ELSIF	OK	
EN	OR	

5.4 Identificadores

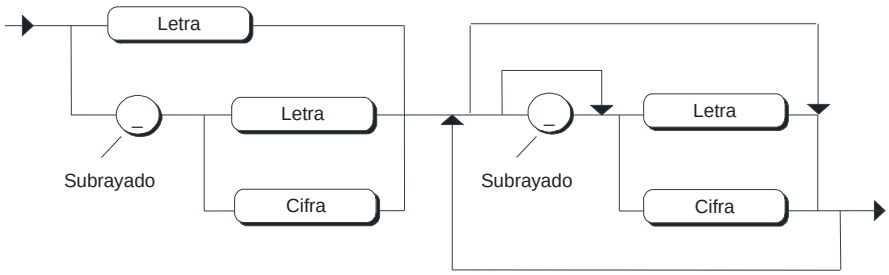
Definición

Un identificador es un nombre que se puede asignar a un objeto del lenguaje S7-SCL, es decir, a una constante, a una variable o a un bloque.

Reglas

Los identificadores se pueden componer de un máximo de 24 letras (sin diéresis) o de cifras dispuestas en cualquier orden con la única limitación de que el primer caracter tiene que ser una letra o un carácter de subrayado. Puede escribirse tanto en mayúsculas como en minúsculas, pues no hay distinción entre mayúsculas y minúsculas (por ejemplo, AnNa y AnnA son idénticos).

IDENTIFICADOR



Ejemplos

Los nombres siguientes son ejemplos de identificadores válidos.

`X`	`y12`
`Suma`	`Temperatura`
`Nombre`	`Superficie`
`Regulador`	`Tabla`

Los nombres siguientes no son identificadores válidos por las razones indicadas.

`4ter`	`//El primer caracter tiene que ser una letra o` `//un carácter de subrayado`
`Array`	`//ARRAY es una palabra clave`
`Valor S`	`//No se admiten los espacios en blanco (tenga en cuenta` `//que un espacio en blanco no es ningún caracter).`

Nota

- Recuerde que el nombre no se asigna por medio de palabras clave o identificadores estándar.
- Lo mejor es que elija nombres unívocos y autoexplicativos que contribuyan a la comprensión de la fuente.

5.5 Identificadores estándar

S7-SCL ofrece una serie de identificadores predefinidos, llamados identificadores estándar. Estos identificadores son:

- los identificadores de bloques,
- los identificadores de operandos para referirse a las áreas de memoria de la CPU,
- los identificadores de temporizadores y
- los identificadores de contadores.

5.6 Identificadores de bloques

Definición

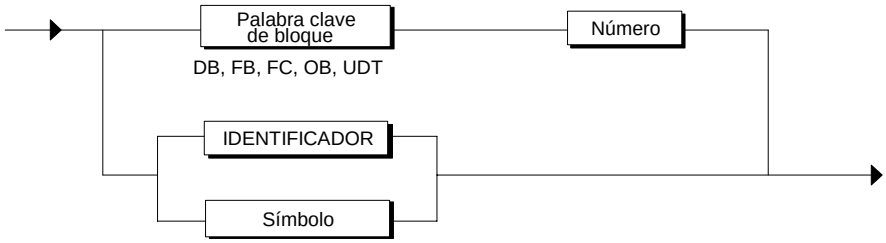
Estos identificadores estándar se utilizan para el direccionamiento absoluto de bloques.

Reglas

La tabla está clasificada según la nemotécnica española (SIMATIC); además se indica la nemotécnica internacional (IEC). La letra x es un comodín que puede ser un número comprendido entre 0 y 65533 o entre 0 y 65535 en el caso de los temporizadores y contadores.

Nemotécnica (SIMATIC)	Nemotécnica (IEC)	Significado
DBx	DBx	Bloque de datos (Data-Block). El nombre de bloque DB0 está reservada para S7-SCL.
FBx	FBx	Bloque de función (Function-Block)
FCx	FCx	Función (Function)
OBx	OBx	Bloque de organización (Organization-Block)
SDBx	SDBx	Bloque de datos del sistema (System-Data-Block)
SFCx	SFCx	Función del sistema (System-Function)
SFBx	SFBx	Bloque de función del sistema (System-Function-Block)
Tx	Tx	Temporizador (Timer)
UDTx	UDTx	Tipo de datos globales o tipo de datos de usuario (Userdefined Data-Type)
Zx	Cx	Contador (Counter)

S7-SCL ofrece varias posibilidades para indicar el nombre de los bloques. Como número del bloque se puede introducir un número decimal entero.



Ejemplo

Los nombres siguientes son válidos:

```
FB10
DB100
T141
```

5.7 Identificadores de operandos

Definición

Indicando el identificador del operando es posible acceder a las áreas de memoria de una CPU desde cualquier posición del programa.

Reglas

La tabla está clasificada según la nemotécnica española (SIMATIC); además se indica la nemotécnica internacional (IEC). Los identificadores de operandos para bloques de datos sólo son válidos junto con la indicación del bloque de datos en cuestión.

Nemotécnica (SIMATIC)	Nemotécnica (IEC)	direccionado	Tipo de dato
Ax.y	Qx.y	Salida (a través de una imagen del proceso)	Bit
ABx	QBx	Salida (a través de una imagen del proceso)	Byte
ADx	QDx	Salida (a través de una imagen del proceso)	Doble palabra
AWx	QWx	Salida (a través de una imagen del proceso)	Palabra
AXx.y	QXx.y	Salida (a través de una imagen del proceso)	Bit
Dx.y	Dx.y	Bloque de datos	Bit
DBx	DBx	Bloque de datos	Byte
DDx	DDx	Bloque de datos	Doble palabra
DWx	DWx	Bloque de datos	Palabra
DXx.y	DXx.y	Bloque de datos	Bit
Ex.y	Ix.y	Entrada (a través de una imagen del proceso)	Bit
EBx	IBx	Entrada (a través de la imagen del proceso)	Byte
EDx	IDx	Entrada (a través de la imagen del proceso)	Doble palabra
EWx	IWx	Entrada (a través de una imagen del proceso)	Palabra
EXx.y	IXx.y	Entrada (a través de una imagen del proceso)	Bit
Mx.y	Mx.y	Marcas	Bit
MBx.y	MBx.y	Marcas	Byte
MDx	MDx	Marcas	Doble palabra
MWx.y	MWx	Marcas	Palabra
MXx	MXx	Marcas	Bit
PABx	PQBx	Salida (periferia directa)	Byte
PADx	PQDx	Salida (periferia directa)	Doble palabra
PAWx	PQWx	Salida (periferia directa)	Palabra
PEBx	PIBx	Entrada (periferia directa)	Byte
PEDx	PIDx	Entrada (periferia directa)	Doble palabra
PEWx	PIWx	Entrada (periferia directa)	Palabra

x = número entre 0 y 65535 (dirección absoluta)

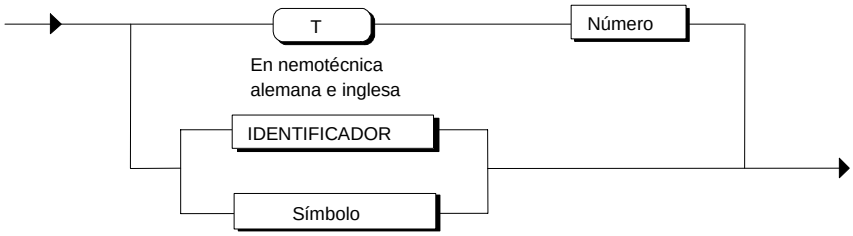
Ejemplo

E1.0 MW10 PAW5 DB20.DW3

5.8 Identificadores de temporizadores

Reglas

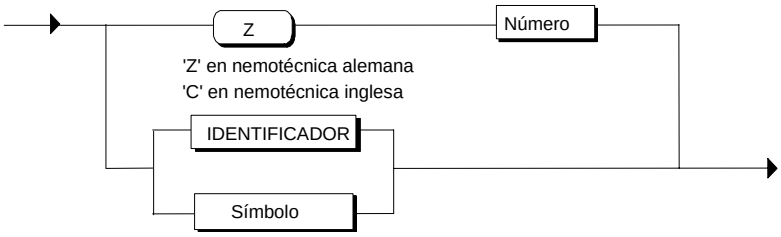
S7-SCL ofrece varias posibilidades para indicar un temporizador. Como número del temporizador puede introducir un número decimal entero.



5.9 Identificadores de contadores

Reglas

S7-SCL ofrece varias posibilidades para indicar un contador. Como número de contador se puede introducir un número decimal entero.



5.10 Números

S7-SCL permite utilizar distintas notaciones para los números. Las reglas siguientes rigen para todos los números:

- Un número puede incluir un signo, un punto decimal o un exponente.
- Un número no debe contener ni comas ni espacios en blanco.
- Para separar visualmente dos cifras puede utilizarse el carácter de subrayado (_).
- El número puede ir precedido de un signo más (+) o de un signo menos (-). Si el número no va precedido de ningún signo se considera positivo.
- Los números no deben sobrepasar ni por exceso ni por defecto determinados valores máximos y mínimos.

Números enteros

Un número entero no tiene ni punto decimal ni exponente. Por lo tanto, un número entero es tan sólo una secuencia de cifras que puede ir precedida de un signo. En S7-SCL hay 2 tipos de números enteros, con rangos numéricos distintos, `INT` y `DINT`.

Veamos algunos números enteros válidos:

0	1	+1	-1
743	-5280	600_00	-32_211

Los siguientes números enteros son **erróneos** por las razones indicadas:

123,456	La coma no está permitida.
36.	Un número entero no debe contener ningún punto decimal.
10 20 30	No está permitido el uso de espacios en blanco.

S7-SCL permite representar números enteros en distintos sistemas de numeración. Esto se hace anteponiendo la palabra clave que corresponde al sistema de numeración. 2# representa el sistema binario; 8# representa el sistema en base 8 y 16# representa el sistema hexadecimal.

Números enteros válidos para el decimal 15:

```
2#1111     8#17  16#F
```

Número real

Un número real debe contener o una coma decimal o un exponente (o ambos). La coma decimal debe indicarse entre dos dígitos. Por lo tanto, el número real no debe comenzar ni terminar con una coma decimal.

Números reales válidos:

0.0	1.0	-0.2	827.602
50000.0	-0.000743	12.3	-315.0066

Los siguientes números reales son **erróneos** por las razones que se indican:

1.	Tiene que haber un dígito a ambos lados de la coma decimal.
1,000.0	No se permite el uso de la coma.
.3333	A ambos lados de la coma decimal debe haber un dígito.

Es posible incluir un exponente para determinar la posición de la coma decimal. Si no se indica ninguna coma decimal, se presupone que ésta figura a la derecha del dígito. El exponente tiene que ser un número entero positivo o negativo. La base 10 se sustituye por la letra E.

El tamaño 3 x 10 exponente10 se puede representar en S7-SCL con los siguientes números reales:

3.0E+10	3.0E10	3e+10	3E10
0.3E+11	0.3e11	30.0E+9	30e9

Los siguientes números reales son **erróneos** por las razones que se indican:

3.E+10	A ambos lados de la coma decimal debe haber un dígito .
8e2.3	El exponente tiene que ser un número entero.
.333e-3	Tiene que haber un dígito a ambos lados de la coma decimal.
30 E10	No está permitido el uso de espacios en blancos.

5.11 Cadenas de caracteres

Definición

Una cadena de caracteres es una secuencia de caracteres (es decir, de letras, cifras o caracteres especiales) encerrados entre comillas.

Algunas cadenas de caracteres válidas son:

```
'ROJO'     '76181  Karlsruhe'      '270-32-3456'
'19,95 DM' 'La respuesta correcta es:'
```

Reglas

Los caracteres de formateo , las comillas (') y el carácter $ se pueden introducir con el símbolo $.

Texto fuente	Después de compilar
'SIGN$'RED$''	SEÑAL 'ROJO'
'50.0$$'	50.0$
'VAL$P'	VALOR Salto de página
'REG-$L'	REG-*Cambio de línea*
'REGL$R	REGLA *Retorno de carro*
'STEP$T'	ETAPA *Tabulador*

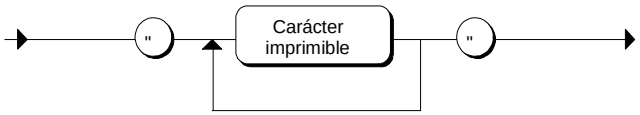
Para los caracteres no imprimibles, introduzca su equivalente en código hexadecimal mediante $*hh*, siendo *hh* el valor del carácter ASCII expresado en hexadecimal.

Para interrumpir una cadena de caracteres (para comentarios que no tienen que imprimirse o visualizarse) el usuario dispone de los caracteres $> y $< .

5.12 Símbolos

En S7-SCL los símbolos se asignan con la siguiente sintaxis: Las comillas sólo se deben indicar en caso de que el símbolo no cumpla la regla IDENTIFICADOR.

Sintaxis:



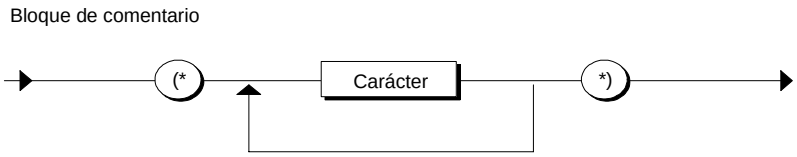
5.13 Bloque de comentario

Reglas

- El bloque de comentario puede abarcar varias líneas y comienza con "(*" y finaliza con "*)" para indicar que se trata de un bloque de comentario.
- De forma estándar está permitido anidar bloques de comentario. Este ajuste se puede cambiar en las preferencias, prohibiendo así el anidamiento de bloques de comentario.
- Un comentario no debe interrumpir ni un nombre simbólico ni una constante. Sin embargo, está permitida la interrupción de cadenas (strings).

Sintaxis

El bloque de comentario se representa formalmente por medio del siguiente diagrama sintáctico:



Ejemplo

```
(* Este es un ejemplo de un bloque de comentario
que puede abarcar varias líneas.*)

PULSADOR := 3 ;
END_FUNCTION_BLOCK
```

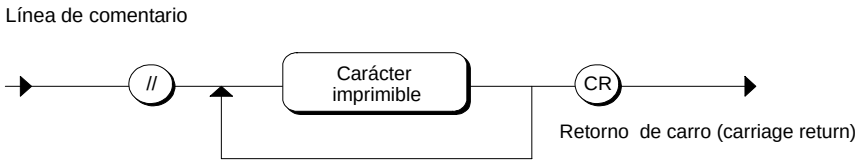
5.14 Línea de comentario

Reglas

- La línea de comentario se inicia con "//" y se extiende hasta el final de la línea.
- La longitud del comentario está restringida a un máximo de 254 caracteres incluyendo los caracteres de inicio '//'.
- Un comentario no debe interrumpir ni un nombre simbólico ni una constante. Sin embargo, está permitida la interrupción de cadenas (strings).

Sintaxis

La línea de comentario se representa formalmente con el siguiente diagrama sintáctico:



Ejemplo

```
VAR
         INTERRUPTOR : INT ; // Línea de comentario
END_VAR
```

Nota

- Los comentarios incluidos en el área de declaración que comienzan con // se adoptan en la interfaz del bloque y se pueden visualizar en el editor KOP/AWL/FUP.
- En el capítulo "Descripción del lenguaje" encontrará información sobre los caracteres imprimibles.
- En las líneas de comentario las parejas de caracteres "(*" y "*)" carecen de significado.

5.15 Variables

Un identificador cuyo valor se puede modificar durante la ejecución del programa se denomina "variable". Cada variable debe ser explicada por separado (es decir, declarada) antes de poder utilizarla en un bloque lógico o en un bloque de datos. La declaración de variables determina que un identificador es una variable (y no una constante etc.) y especifica el tipo de variable asignándole un tipo de datos.

Dependiendo de la validez de las variables se distingue entre:

- datos locales,
- datos globales de usuario, y
- variables predefinidas permitidas (áreas de memoria de una CPU).

Datos locales,

Los datos locales son datos que se declaran en un bloque lógico (FC, FB, OB) y que sólo valen en dicho bloque. En particular, son:

Variable	Significado
Variables estáticas	Una variable estática es una variable local cuyo valor se mantiene constante durante todas las ejecuciones del bloque (memoria del bloque). Sirve para guardar valores de un bloque de función.
Variables temporales	Las variables temporales pertenecen localmente a un bloque lógico y no ocupan ningún área de memoria estática. Su valor se conserva solamente durante una ejecución del bloque. A las variables temporales no se puede acceder fuera del bloque en el que han sido declaradas.
Parámetros del bloque	Los parámetros del bloque son parámetros formales de un bloque de función o de una función. Son variables locales que sirven para transferir los parámetros actuales indicados en la llamada.

Datos globales de usuario

Los datos globales de usuario son datos o áreas de datos que el usuario puede utilizar desde cualquier posición del programa. Para hacerlo debe crear bloques de datos (DB).

Al crear un DB, se define su estructura en la declaración de la estructura. En lugar de la declaración también se puede utilizar un tipo de datos de usuario (UDT). El orden en el que se introduzcan los componentes de la estructura determinará el orden de los datos en el DB.

Áreas de memoria de la CPU

A las áreas de memoria de la CPU se puede acceder directamente desde cualquier posición del programa mediante los identificadores de operandos, sin necesidad de declarar estas variables.

También es posible acceder a estas áreas de datos mediante nombres simbólicos. En este caso la asignación simbólica se realiza globalmente en la tabla de símbolos de STEP 7.

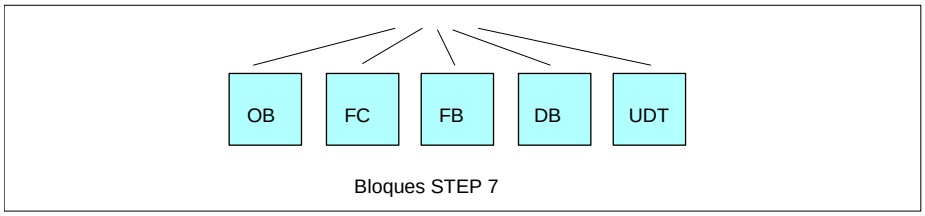
6 Estructura de programas S7-SCL

6.1 Bloques en fuentes S7-SCL

En un archivo fuente S7-SCL se pueden programar entre 1 y n bloques. Los bloques de STEP 7 son componentes del programa de usuario limitados por su función, su estructura o su finalidad.

Tipos de bloques

Bloques disponibles:



Bloques preconfeccionados

No es necesario programar cada una de las funciones, pudiendo recurrir a los bloques preconfeccionados. Estos bloques, que residen en el sistema operativo de las CPUs o en las librerías *(S7lib)* del paquete básico de STEP 7, se pueden utilizar, p.ej. para programar funciones de comunicación.

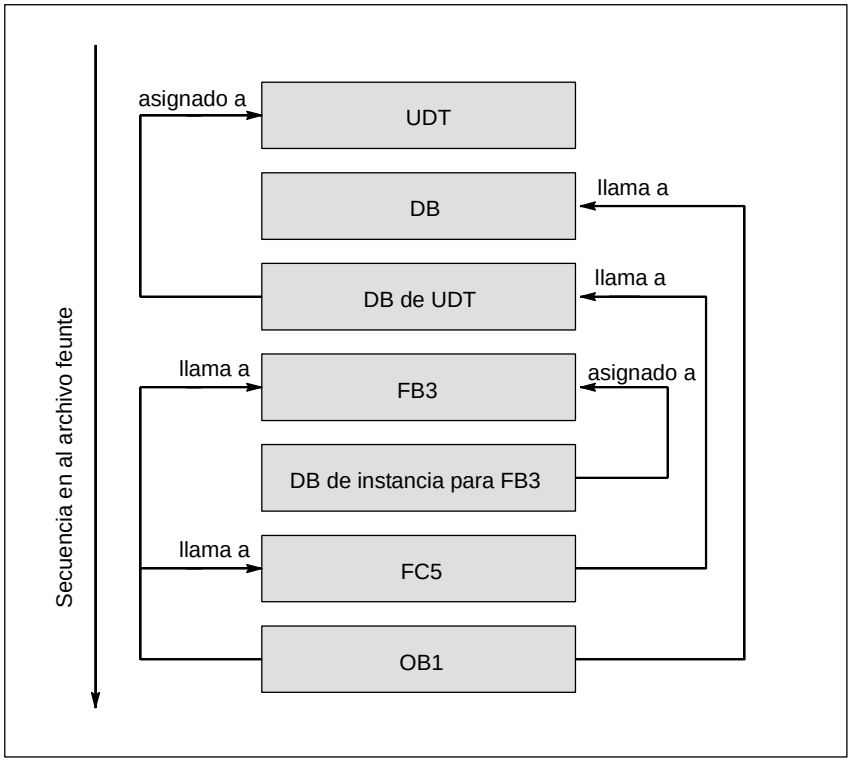
6.2 Criterios para ordenar los bloques en el programa

En principio rige la regla:

Los bloques llamados preceden a los bloques invocantes.

Esto significa que:

- los tipos de datos de usuario (UDT) preceden a los bloques en los que se utilizan.
- Los bloques de datos que tienen asignado un tipo de datos de usuario (UDT) siguen al UDT.
- Los bloques de datos a los que acceden todos los bloques lógicos preceden a los bloques que los llaman.
- Los bloques de datos que tienen asignado un bloque de función tienen que figurar detrás del bloque de función.
- El OB1 que llama a otros bloques figura en último lugar. A su vez, los bloques llamados por los bloques llamados desde el OB1 preceden a los anteriores.
- Los bloques que se llaman desde el archivo fuente pero que no están programados en el mismo archivo fuente deben figurar en el programa de usuario correspondiente al compilar el archivo.
- Además de los bloques, las fuentes fuentes de S7-SCL también pueden contener indicaciones sobre los ajustes del compilador con los que deban compilarse los diferentes bloques. Las opciones del compilador se encuentran fuera de los límites de los bloques.



6.3 Estructura básica de un bloque

Un bloque se compone de las siguientes áreas:

- Principio del bloque, formado por una palabra clave y un número de bloque o un nombre simbólico, p.ej."ORGANIZATION_BLOCK OB1" en el caso de un bloque de organización. En el caso de las funciones, se indica adicionalmente el tipo de función, el cual determina el tipo de datos del valor de respuesta. Si no se programa ningún valor de respuesta, debe indicarse la palabra clave VOID.
- Título del bloque (opcional). El título debe ir precedido de la palabra clave "TITLE ="
- Comentario del bloque (opcional). El comentario del bloque puede abarcar varias líneas, y cada una de ellas comienza por "//".
- Atributos del bloque (opcional)
- Atributos de sistema para bloques (opcional)
- Área de declaración (varía según el tipo de bloque)
- Área de instrucciones en el caso de los bloques lógicos o
  asignación de valores actuales en el caso de los bloques de datos (opcional)
- En los bloques lógicos: Instrucciones
- Fin de bloque, formado por END_ORGANIZATION_BLOCK, END_FUNCTION_BLOCK o END_FUNCTION

6.4 Principio y fin de bloque

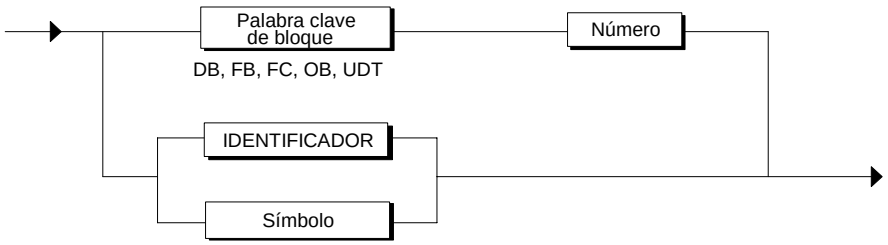
Dependiendo del bloque del que se trate, el texto fuente del bloque empieza por un identificador estándar que señala el principio del bloque así como por el nombre del bloque. El bloque se termina con el identificador de fin de bloque.

La tabla siguiente muestra la sintaxis de los diferentes tipos de bloques:

Nombre	**Tipo de bloque**	**Sintaxis**
Bloque de función	FB	FUNCTION_BLOCK fb_name . . . END_FUNCTION_BLOCK
Función	FC	FUNCTION fc_name : tipo de función . . . END_FUNCTION
Bloque de organización	OB	ORGANIZATION_BLOCK ob_name . . . END_ORGANIZATION_BLOCK
Bloque de datos	DB	DATA_BLOCK db_name . . . END_DATA_BLOCK
Tipo de datos global	UDT	TYPE udt_name . . . END_TYPE

Nombre del bloque

En la tabla, *xx_nombre* es el nombre del bloque de acuerdo con la siguiente sintaxis:



El número del bloque puede ser un valor comprendido entre 0 y 65533. Sin embargo el nombre del bloque de datos DB0 está reservado.

Recuerde que el nombre o símbolo se debe definir en la tabla de símbolos de STEP 7.

Ejemplo

```
FUNCTION_BLOCK FB10
FUNCTION_BLOCK Bloque regulador
FUNCTION_BLOCK "Regulador.B1&U2"
```

6.5 Atributos de bloques

Definición

Un atributo de bloque es una propiedad del bloque que se puede utilizar, por ejemplo, para indicar el título del bloque, la versión, la protección del bloque o el autor. Al seleccionar bloques para crear la aplicación concreta puede hacerse mostrar las propiedades en el diálogo de propiedades de STEP 7.

Se pueden asignar los siguientes atributos:

Palabra clave / Atributo	Significado	Ejemplos
TITLE='caracter_imprimible'	Título del bloque	TITLE='CLASIFICAR'
VERSION : 'secuencia_de_numeros_decimales. secuencia_de_numeros_decimales'	Número de la versión del bloque (0..15) Nota: el atributo VERSION de los bloques de datos (DB) no se indica entre comillas.	VERSION : '3-1' //Con DB: VERSION: 3.1
KNOW_HOW_PROTECT	Protección del bloque; un bloque que haya sido compilado con esta opción no se puede abrir con STEP 7.	KNOW_HOW_PROTECT
AUTHOR :	Nombre del autor: nombre de la empresa, del departamento u otros (IDENTIFICADOR y 'caracteres imprimibles)	AUTHOR : Siemens AUTHOR : 'A&D AS'
NAME:	Nombre del bloque (IDENTIFICADOR y 'caracteres imprimibles')	NAME : PID NAME : 'A&D AS'
FAMILY :	Nombre de la familia de bloques: p.ej., motores. Guarda el bloque en un grupo de bloques para poder encontrarlo rápidamente (IDENTIFICADOR y 'caracteres imprimibles).	FAMILY : Ejemplo FAMILY : 'A&D AS'

Reglas

- Los atributos del bloque se declaran mediante palabras clave directamente detrás de la instrucción de inicio del bloque.
- El identificador puede contener 8 caracteres como máximo.

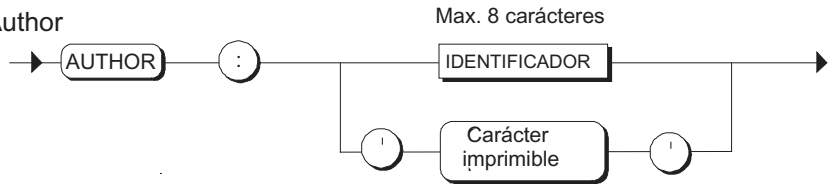
Los atributos del bloque se introducen con la siguiente sintaxis:

Título

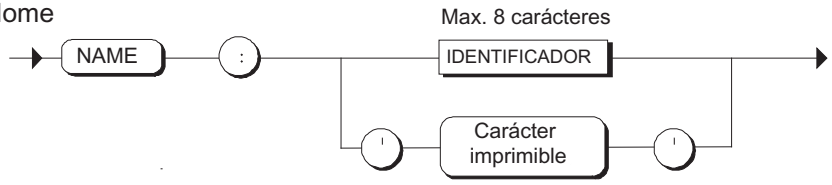
Versión

Protección de bloque

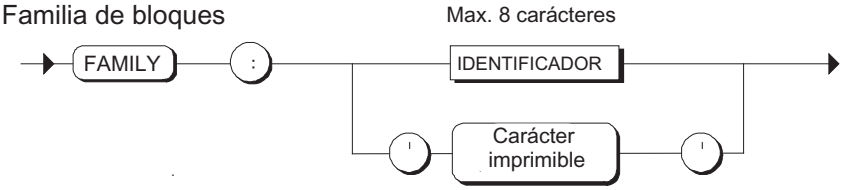
Author



Nome



Familia de bloques



Ejemplos

```
FUNCTION_BLOCK FB10
TITLE = 'Promedio'
VERSION : '2.1'
KNOW_HOW_PROTECT
AUTHOR : AUT_1
```

6.6 Comentario del bloque

Los comentarios relativos a todo el bloque se introducen en el encabezado del bloque, detrás de la línea "TITLE:". Utilice para ello la notación de la línea de comentario. Si el comentario abarca varias líneas, comience cada una de ellas con //.

El comentario del bloque aparece p.ej. en el diálogo Propiedades del bloque del Administrador SIMATIC o en el editor KOP/AWL/FUP.

Ejemplo

```
FUNCTION_BLOCK FB15
TITLE=MI_BLOQUE
//Esto es un comentario de bloque.
//Se indica en forma de secuencia de líneas de comentario
//y se puede visualizar, p.ej., en el Administrador SIMATIC.
AUTHOR...
FAMILY...
```

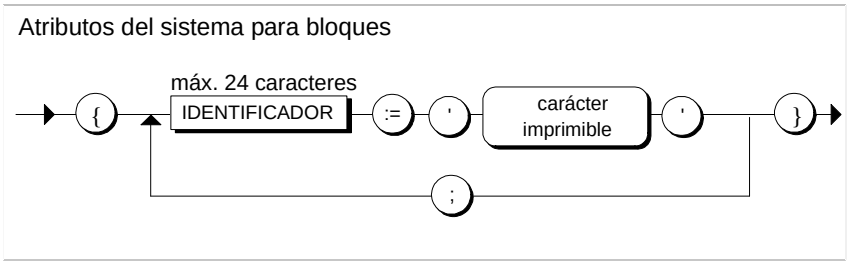
6.7 Atributos de sistema para bloques

Definición

Los atributos de sistema son atributos básicos que sirven para todo el sistema de control de procesos. Los atributos de sistema para bloques rigen todo el bloque.

Reglas

- Los atributos de sistema se declaran inmediatamente después de la instrucción de inicio del bloque.
- Los atributos se introducen con la siguiente sintaxis:



Ejemplos

```
FUNCTION_BLOCK FB10
{S7_m_c := 'true' ;
S7_blockview := 'big'}
```

6.8 Área de declaración

Definición

El área de declaración permite declarar variables locales, parámetros, constantes y metas de salto.

- Variables, parámetros, constantes y metas de salto locales, es decir, que sólo tienen validez dentro de un mismo bloque, se definen en el área de declaración del bloque lógico.
- Las áreas de datos a las que acceden todos los bloques lógicos se definen en el área de declaración de los bloques de datos.
- En el área de declaración de un UDT se define el tipo de datos de usuario.

Estructura

El área de declaración se divide en diferentes bloques de declaración, cada uno de los cuales se caracteriza por una pareja de palabras clave. Cada bloque incluye una lista de declaración para datos semejantes. La secuencia de estos bloques es arbitraria. La tabla siguiente muestra los posibles bloques de declaración:

Datos	Sintaxis	FB	FC	OB	DB	UDT

Constantes:	`CONST;`
	`lista de declaración`
	`END_CONST`
		X	X	X		

Metas de salto	`LABEL;`
	`lista de declaración`
	`END_LABEL`
		X	X	X		

Variables temporales	`VAR_TEMP; lista de declaración` `END_VAR`	X	X	X		

Variables estáticas	`VAR;`
	`lista de declaración`
	`END_VAR`
		X	X *)		X **)	X **)

Parámetros de entrada	`VAR_INPUT;`
	`lista de declaración`
	`END_VAR`
		X	X			

Parámetros de salida	`VAR_OUTPUT;`
	`lista de declaración`
	`END_VAR`
		X	X			

Parámetros de entrada/salida	`VAR_IN_OUT;`
	`lista de declaración`
	`END_VAR`
		X	X			

*) En las funciones se admite la declaración de variables entre las dos palabras clave VAR y END_VAR, sin embargo, las declaraciones se transfieren al área temporal al compilar la fuente.

**) En los DBs y UDTs, las palabras clave VAR y END_VAR se sustituyen por STRUCT y END_STRUCT.

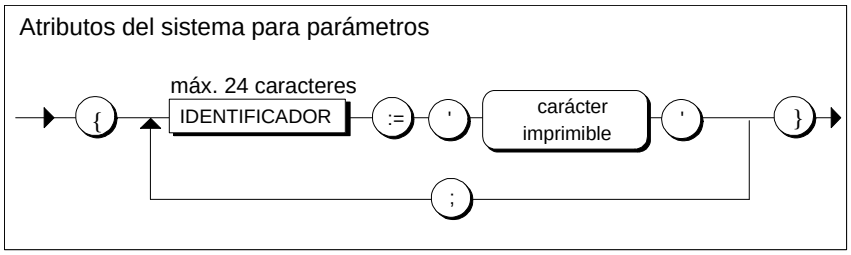
6.9 Atributos de sistema para parámetros

Definición

Los atributos del sistema son atributos básicos que se utilizan en todo el sistema de control de procesos. Sirven, p.ej., para configurar mensajes o enlaces de comunicación. Los atributos de sistema para parámetros solamente son válidos para los parámetros configurados. Los atributos de sistema se pueden asignar a parámetros de entrada, salida y entrada/salida.

Reglas

- Los atributos del sistema para parámetros se asignan en los bloques de declaración 'Parámetros de entrada', 'Parámetros de salida' o 'Parámetros de entrada/salida'.
- Un identificador puede tener 24 caracteres como máximo.
- Los atributos se introducen con la siguiente sintaxis:



Ejemplo

```
VAR_INPUT
        in1      {S7_server:='alarm_archiv';
                  S7_a_type:='ar_send'}: DWORD ;
END_VAR
```

A la Ayuda de los atributos de sistema se accede desde la ayuda en pantalla de SCL. Para ello, seleccione el tema "Llamada de la ayuda de referencia".

6.10 Área de instrucciones

Definición

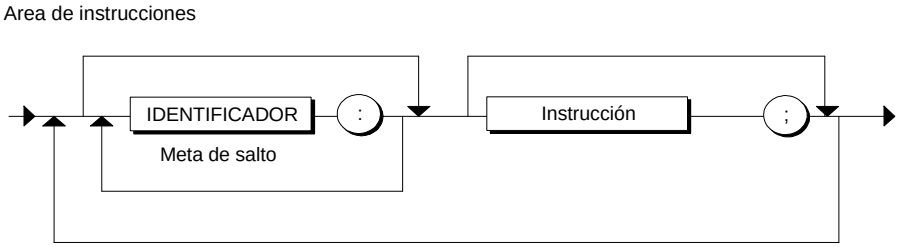
El área de instrucciones contiene instrucciones que se ejecutan una vez llamado el bloque lógico. Estas instrucciones permiten procesar datos o direcciones.

El área de instrucciones de los bloques de datos incluye instrucciones para inicializar sus variables.

Reglas

- Si lo desea, puede comenzar el área de instrucciones con la palabra clave BEGIN. Si se trata de bloques de datos, BEGIN es obligatoria. El área de instrucciones termina con la palabra clave de fin de bloque.
- Cada instrucción termina con un punto y coma.
- Todos los identificadores utilizados en el área de instrucciones deben declararse previamente.
- Opcionalmente, las instrucciones pueden ir precedidas de metas de salto.

Las áreas de instrucciones se introducen con la siguiente sintaxis:



Ejemplo

```
BEGIN
    VALOR_INICIAL:=0;
    VALOR_FINAL  :=200;
.
.
GUARDAR:   RESULTADO   :=VALOR _TEORICO;
.
.
END_FUNCTION_BLOCK
```

6.11 Instrucciones

Definición

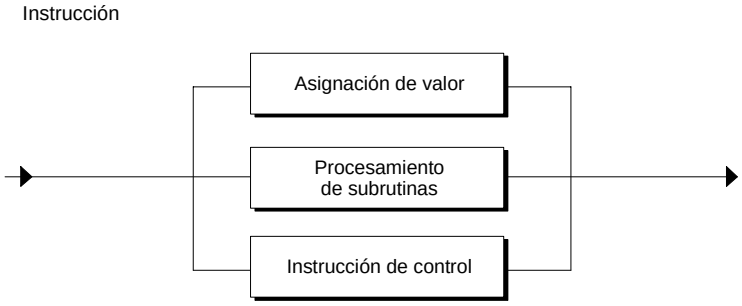
Una instrucción es la unidad autónoma más pequeña del programa de usuario. Constituye una norma de trabajo para el procesador.

S7-SCL admite los siguientes tipos de instrucciones:

- Asignaciones de valor, que sirven para asignar a una variable un valor, el resultado de una expresión o el valor de otra variable.
- Instrucciones de control, que sirven para repetir instrucciones o grupos de instrucciones o para derivar a otra parte del programa.
- Procesamiento de subrutinas que sirven para llamar a funciones y bloques de función.

Reglas

Las instrucciones se introducen conforme a la siguiente sintaxis:



Ejemplo

Los siguientes ejemplos ilustran las distintas variantes de instrucciones:

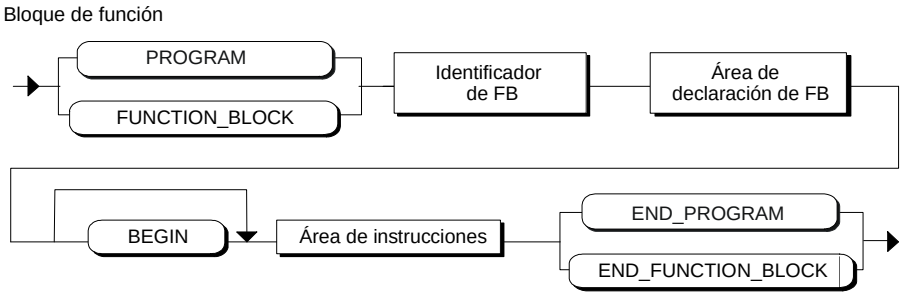
```
// Ejemplo de una asignación de valores
VALOR_MEDIDO:= 0 ;

// Ejemplo del procesamiento de un subprograma
FB1.DB11 (TRANSMISION:= 10) ;

// Ejemplo de una instrucción de control
WHILE CONTADOR < 10 DO..
.
.
END_WHILE;
```

6.12 Estructura de un bloque de función (FB)

Definición

Un bloque de función (FB) es un bloque lógico que contiene una sección del programa y que tiene asignada un área de memoria. Cada vez que se llama a un FB hay que asignarle un DB de instancia. La estructura de este bloque de datos de instancia se determina al definir el área de declaración del FB.

Sintaxis



Nombre de un FB

Introduzca la palabra clave FB como nombre del FB detrás de la palabra clave FUNCTION_BLOCK o PROGRAM y a continuación, el número del bloque o el nombre simbólico del FB. El número del bloque puede ser un valor comprendido entre 0 y 65533.

Ejemplos:

```
FUNCTION_BLOCK FB10
FUNCTION_BLOCK MOTOR1
```

Área de declaración de FB

El área de declaración del FB sirve para definir los datos específicos del bloque. Los bloques de declaración permitidos figuran en el capítulo "Área de declaración". Tenga en cuenta que el área de declaración también determina la estructura del DB de instancia asignado.

Ejemplo

El ejemplo siguiente muestra el código fuente de un bloque de función. Los parámetros de entrada y de salida (en este caso V1, V2) tienen valores iniciales.

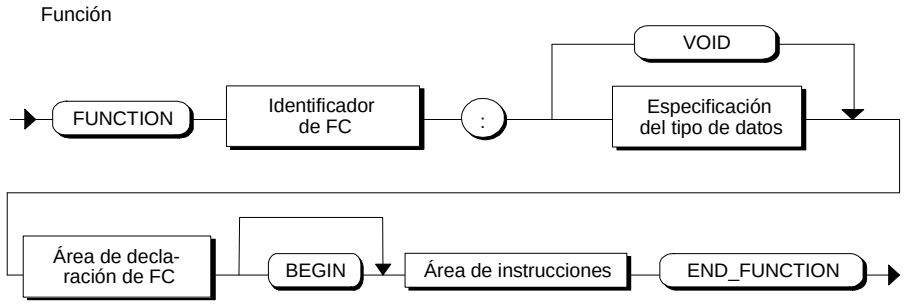
```
FUNCTION_BLOCK FB11
VAR_INPUT
        V1 : INT := 7 ;
END_VAR
VAR_OUTPUT
        V2 : REAL  ;
END_VAR
VAR
        FX1, FX2, FY1, FY2  :  REAL ;

END_VAR
BEGIN
        IF V1 = 7 THEN
           FX1 := 1.5 ;
           FX2 := 2.3 ;
           FY1 := 3.1 ;
           FY2 := 5.4 ;
           //Llamada de la función FC11 con tranferencia
           //de parámetros
           //por medio de variables estáticas.
           V2 := FC11 (X1:= FX1, X2 := FX2, Y1 := FY1, Y2 := FY2) ;
        END_IF ;
END_FUNCTION_BLOCK
```

6.13 Estructura de una función (FC)

Definición

Una función (FC) es un bloque lógico que no tiene asignada ningún área de memoria propia. No necesita bloque de datos de instancia. A diferencia de un FB, una función puede retornar el resultado de la función (valor de respuesta) al punto de llamada. Por consiguiente, la función se puede utilizar igual que una variable en una expresión. Las funciones del tipo VOID no tienen valor de respuesta.

Sintaxis



Nombre de la FC

Introduzca la palabra clave FC como nombre de la FC detrás de la palabra clave "FUNCTION" y a continuación, el luego el número del bloque o el nombre simbólico de la FC. El número del bloque puede ser un valor comprendido entre 0 y 65533.

Ejemplo:

```
FUNCTION FC17 : REAL
FUNCTION FC17 : VOID
```

Especificación del tipo de datos

La especificación del tipo de datos determina el tipo de datos del valor de respuesta. Se admiten todos los tipos de datos excepto STRUCT y ARRAY. Si se renuncia al valor de respuesta con VOID no será necesario indicar ningún tipo de datos.

Área de declaración de la FC

El área de declaración de la FC sirve para declarar datos locales (variable temporal, parámetro de entrada, parámetro de salida, parámetro entrada/salida, constantes o metas de salto).

Área de instrucciones de la FC

En el área de instrucciones hay que asignar el resultado de la función al nombre de la función. Esta asignación no es necesaria en las funciones de tipo VOID. Una instrucción válida dentro de una función con el nombre FC31 sería, por ejemplo:

```
FC31:=VALOR;
```

Ejemplo

```
FUNCTION FC11: REAL
VAR_INPUT
        x1: REAL ;
        x2: REAL ;
        x3: REAL ;
        x4: REAL ;
END_VAR
VAR_OUTPUT
     Q2: REAL ;
END_VAR
BEGIN
        // Retorno del valor de la función
        FC11:= SQRT( (x2 - x1)**2 + (x4 - x3) **2 ) ;
        Q2:= x1 ;
END_FUNCTION
```

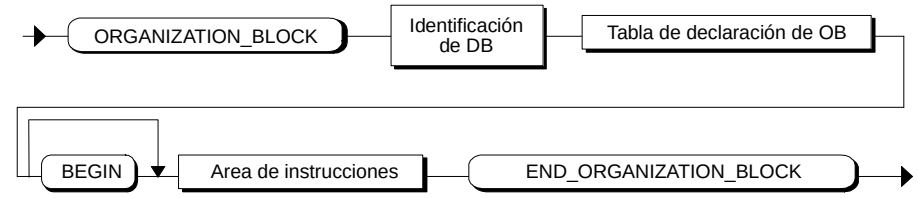
6.14 Estructura de un bloque de organización (OB)

Definición

Al igual que un FB o una FC, el bloque de organización es una parte del programa de usuario que el sistema operativo llama cíclicamente o cuando se producen determinados eventos. Constituye el interface entre el programa de usuario y el sistema operativo.

Sintaxis

Bloque de organización



Nombre del OB

Indique la palabra clave OB como nombre del OB detrás de la palabra clave "ORGANIZATION_BLOCK" y a continuación, el número del bloque o el nombre simbólico del OB. El número del bloque puede ser un número comprendido entre 1 y 65533.

Ejemplos:

```
ORGANIZATION_BLOCK OB1
ORGANIZATION_BLOCK ALARM
```

Área de declaración del OB

El área de declaración del OB sirve para declarar los datos locales (variables temporales, constantes, metas del salto).

Para que se pueda ejecutar, cada OB necesita básicamente 20 bytes de datos locales para el sistema operativo. Por ello se debe declarar un array con un identificador cualquiera. Si inserta la plantilla de bloque OB, este array ya estará incluido en la declaración.

Ejemplo

```
ORGANIZATION_BLOCK OB1
VAR_TEMP
        HEADER : ARRAY [1..20] OF BYTE ; //20 Byte para sist. op.
END_VAR
BEGIN
        FB17.DB10 (V1 := 7) ;
END_ORGANIZATION_BLOCK
```

6.15 Estructura de un bloque de datos (DB)

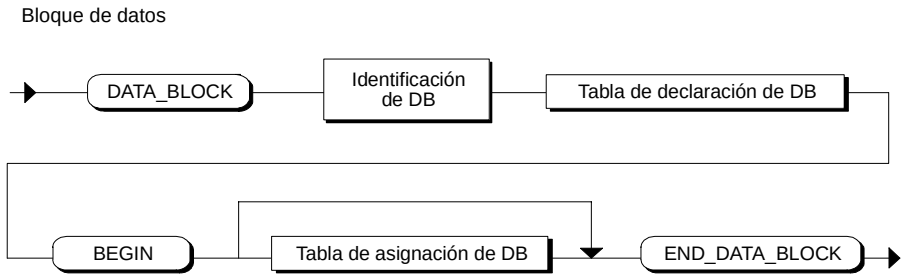
Definición

Los datos globales de usuario a los que acceden todos los bloques de un programa se depositan en bloques de datos. Cada FB, FC u OB puede leer o escribir estos bloques de datos.

Existen dos tipos diferentes de bloques de datos:

- **Bloque de datos**
  Bloques de datos a los que pueden acceder todos los bloques lógicos del programa S7. Todos los FBs, FCs y OBs pueden leer o escribir los datos contenidos en estos bloques de datos.
- **Bloque de datos asociado a un FB (DB de instancia)**
  Los bloques de datos de instancia son bloques de datos asignados a un determinado bloque de función (FB). Incluyen datos locales para el bloque de función asignado. Estos bloques de datos son generados automáticamente por el compilador S7-SCL en cuanto se llama al FB en el programa de usuario.

Sintaxis



Nombre del DB

Indique la palabra clave DB como nombre del DB destrás de la palabra clave "DATA_BLOCK" y a continuación, el número del bloque o el nombre simbólico del DB. El número del bloque puede ser un valor comprendido entre 1 y 65533.

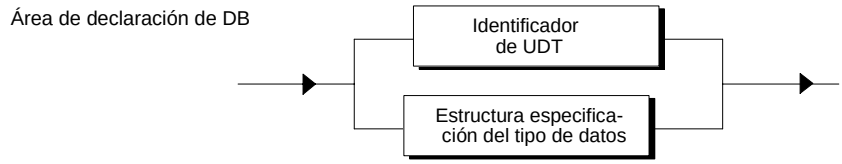
Ejemplos:

```
DATA_BLOCK DB20
DATA_BLOCK MARGEN_DE_MEDIDA
```

Área de declaración del DB

En el área de declaración del bloque de datos se define la estructura de los datos del DB. Dispone de dos posibilidades:

- **Asignando un tipo de datos de usuario**
  Aquí se puede introducir un tipo de datos de usuario previamente definido en el programa. En tal caso, el bloque de datos adoptará la estructura de este UDT. Los valores iniciales de las variables se pueden asignar en la tabla de asignación del DB.
- **Definiendo un tipo de datos STRUCT**
  Al especificar el tipo de datos STRUCT se define el tipo de datos de cada una de las variables que va a guardarse en el DB y, dado el caso, su valor inicial.



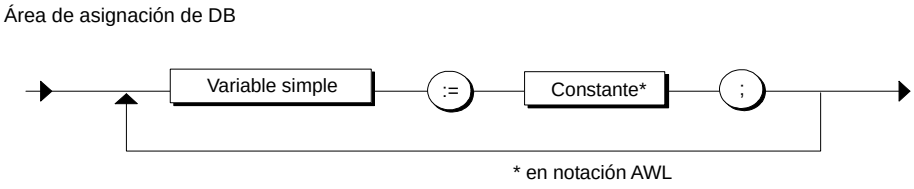
Ejemplo:

```
DATA_BLOCK DB20
        STRUCT   // Área de declaración
                VALOR:ARRAY  [1..100] OF INT;
        END_STRUCT

BEGIN            // Principio del área de declaración
:
END_DATA_BLOCK   // Fin del bloque de datos
```

Área de asignación del DB

En el área de asignación se pueden adaptar los datos declarados en el área de declaración a la aplicación concreta mediante valores específicos del DB.

El área de asignación empieza por la palabra clave BEGIN y se compone de una sucesión de asignaciones de valores.



Al asignar los valores iniciales (de inicialización), al especificar atributos y al indicar comentarios rige la sintaxis de AWL. En la ayuda en pantalla de AWL y en la documentación de STEP 7 puede consultar la distinta notación de las constantes, los atributos y los comentarios.

Ejemplo

```
// Bloque de datos que tiene asignado un tipo de datos STRUCT
DATA_BLOCK DB10
        STRUCT // Declaración de datos con preasignación
                VALOR  :     ARRAY [1..100] OF INT := 100 (1) ;
                PULSADOR     :     BOOL        := TRUE ;
                S_PALABRA    :     WORD  := W#16#FFAA ;
                S_BYTE       :     BYTE  := B#16#FF ;
                S_TIME       :     S5TIME      := S5T#1h30m10s ;
        END_STRUCT

BEGIN // Área de asignación
      // Asignación de valores para determinados elementos del array
      VALOR [1] := 5;
      VALOR [5] := -1 ;

END_DATA_BLOCK

// Bloque de datos que tiene asignado un tipo de datos de usuario
DATA_BLOCK DB11
        UDT 51
BEGIN
END_DATA_BLOCK
```

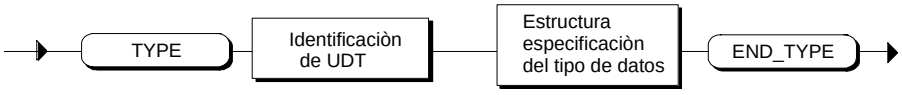
6.16 Estructura de un tipo de datos de usuario

Los tipos de datos de usuario UDT son estructuras especiales creadas por el usuario. Dado que los tipos de datos de usuario tienen un nombre, pueden reutilizarse. Por definición puede utilizarse en la totalidad del programa de usuario, por lo que son tipos de datos globales. Por consiguiente, estos tipos de datos se pueden:

- utilizar en bloques como tipos de datos simples o compuestos, o
- utilizar como plantilla para crear bloques de datos de idéntica estructura.

Al introducir tipos de datos de usuario, tenga en cuenta que en la fuente S7-SCL figuran delante de los bloques en los que son utilizados.

Tipo de datos de usuario



Nombre UDT

Introduzca la palabra clave `UDT` detrás de la palabra clave `TYPE` y a continuación, un número o simplemente el nombre simbólico del UDT. El número del bloque puede ser un valor comprendido entre 0 y 65533.

Ejemplos:

```
TYPE UDT10
TYPE BLOQUE_DE_TRANSFERENCIA_DE_PARAMETROS
```

Especificación del tipo de datos

La especificación del tipo de datos se realiza mediante una **especificación STRUCT**. El tipo de datos UDT puede utilizarse en las áreas de declaración de bloques lógicos o en bloques de datos, o bien asignarse a DBs.

Ejemplo de definición de un UDT

```
TYPE VALORES_MEDIDOS
STRUCT
// Definición UDT con nombre simbólico
        BIPOL_1  : INT := 5;
        BIPOL_2  : WORD := W#16#FFAA ;
        BIPOL_3  : BYTE  := B#16#F1 ;
        BIPOL_4  : WORD := B#(25,25) ;
        MEDICION : STRUCT
                BIPOLAR_10V : REAL ;
                UNIPOLAR_4_20MA : REAL ;
        END_STRUCT ;
END_STRUCT ;
END_TYPE

//Utilización de un UDT en un FB
FUNCTION_BLOCK FB10
VAR
        MARGEN_MEDIDA: VALORES_MEDIDOS;
END_VAR
BEGIN
        // . . .
        MARGEN_MEDIDA.BIPOL_1 := -4 ;
        MARGEN_MEDIDA.MEDICION.UNIPOLAR_4_20MA  :=  2.7 ;
        // . . .
END_FUNCTION_BLOCK
```

6.17 Opciones del compilador en fuentes de S7-SCL

Definición

Además de los bloques, las fuentes de S7-SCL también pueden contener indicaciones sobre los ajustes del compilador con los que se deban compilar los diferentes bloques.

Las opciones del compilador controlan la compilación de distintos bloques o de toda la fuente, independientemente de los ajustes de la ficha "Compilador (Ajustes)".

Las opciones del compilador se pueden utilizar en fuentes de S7-SCL o en archivos de control de compilación.

Validez

Las opciones se aplican solamente en la compilación de la fuente para la que han sido definidas. La validez de una opción de compilación comienza con su denominación y termina al final de la fuente o del archivo de control de compilación. Si hubiera varias opciones idénticas, se aplicará la última.

Si se han definido opciones del compilador para un bloque, dichas opciones tendrán una prioridad mayor que los ajustes de la ficha "Compilador (Ajustes)". Sin embargo, los ajustes de la ficha se seguirán aplicando de forma global.

Reglas

Para las opciones del compilador se aplican las siguientes reglas:

- Las opciones figuran en la fuente fuera de los límites de bloques.
- Las opciones figuran en una línea propia.
- No se distingue entre mayúsculas y minúsculas.

Opciones disponibles

La tabla muestra las opciones disponibles:

Opción	Valor	Significado
[Scl_]ResetOptions	No es posible introducir valores	Preselección de los ajustes de compilación (ajustes del cuadro de diálogo)
[Scl_]OverwriteBlocks	**'y**[es]**'** o **'n**[o]**'**	Sobrescribir bloques
[Scl_]GenerateReferenceData	**'y**[es]**'** o **'n**[o]**'**	Crear datos de referencia
[Scl_]S7ServerActive	**'y**[es]**'** o **'n**[o]**'**	Considerar atributo de sistema "S7_server"
[Scl_]CreateObjectCode	**'y**[es]**'** o **'n**[o]**'**	Crear object code
[Scl_]OptimizeObjectCode	**'y**[es]**'** o **'n**[o]**'**	Optimizar object code
[Scl_]MonitorArrayLimits	**'y**[es]**'** o **'n**[o]**'**	Vigilar límites de arrays
[Scl_]CreateDebugInfo	**'y**[es]**'** o **'n**[o]**'**	Crear debug info
[Scl_]SetOKFlag	**'y**[es]**'** o **'n**[o]**'**	Activar OK flag
[Scl_]SetMaximumStringLength	**'1** .. **254'**	Longitud máxima de string

Ejemplo

```
{SCL_OverwriteBlocks := 'y'   ; SCL_CreateDebugInfo    := 'y'}
{SetOKFlag := 'y'   ; OptimizeObjectCode    := 'y'}
```

7 Tipos de datos

7.1 Sinopsis de los tipos de datos en S7-SCL

Los tipos de datos determinan:

- el tipo y el significado de los elementos de datos,
- los márgenes permitidos para los elementos de datos,
- la cantidad admisible de operaciones que se pueden ejecutar con un operando de un tipo de datos,
- la notación de las constantes del tipo de datos en cuestión.

Tipos de datos simples

Los tipos de datos simples definen la estructura de aquellos datos que no se pueden descomponer en unidades más pequeñas. Corresponden a la definición de la norma DIN EN 1131-3. Un tipo de datos simple describe un área de memoria de longitud fija y significa bit, entero, real, duración, hora y tamaños de caracteres. Los tipos de datos siguientes están predefinidos en S7-SCL.

Grupo	Tipos de datos	Significado
Tipo de datos Bit	BOOL BYTE WORD DWORD	Los datos de este tipo ocupan 1bit, 8 bits, 16 bits o 32 bits
Tipo de datos Carácter	CHAR	Los datos de este tipo ocupan exactamente 1 carácter del juego de caracteres ASCII
Datos numéricos	INT DINT REAL	Los datos de este tipo están a disposición para el procesamiento de valores numéricos
Tiempos	TIME DATE TIME_OF_DAY S5TIME	Los datos de este tipo representan los distintos valores de fecha y hora en STEP 7.

Tipos de datos compuestos

S7-SCL asiste los siguientes tipos de datos compuestos:

Tipo de datos	Significado
DATE_AND_TIME DT	Define un área de 64 bits (8 bytes). Este tipo de datos memoriza (en formato decimal codificado en binario) la fecha y hora del día, y está predefinido en S7-SCL.
STRING	Define un área para una secuencia de caracteres de 254 caracteres como máximo (tipo de datos CHAR).
ARRAY	Define un array de elementos de un mismo tipo de datos (simple o compuesto).
STRUCT	Define un grupo de tipos de datos combinados a voluntad. Es posible definir un array de estructuras o bien una estructura formada a su vez por estructuras y arrays.

Tipos de datos de usuario

Los tipos de datos de usuario se pueden crear a voluntad en la declaración del tipo de datos. Estos datos poseen un nombre propio, por lo que se pueden reutilizar. Ello permite utilizar un tipo de datos de usuario para crear varios bloques de datos de estructura idéntica.

Tipo de datos Parámetros

Los tipos de parámetros son tipos de datos especiales para temporizadores, contadores y bloques que se pueden utilizar como parámetros formales.

Tipo de datos	Significado
TIMER	Sirve para declarar funciones de temporización como parámetros
COUNTER	Sirve para declarar funciones de contaje como parámetros
BLOQUE_xx	Sirve para declarar FC, FB, DB y SDB como parámetros
ANY	Sirve para admitir un operando de cualquier tipo de datos como parámetro
POINTER	Sirve para admitir un área de memoria como parámetro

Tipo de datos ANY

En S7-SCL se pueden utilizar variables del tipo de datos ANY como parámetros formales de un bloque. Además, también es posible crear variables temporales de este tipo y utilizarlas en asignaciones de valores.

7.2 Tipos de datos simples

7.2.1 Tipo de datos Bit

Los datos de este tipo son combinaciones de bits que pueden ocupar 1 bit (tipo de datos BOOL), 8 bits, 16 bits o 32 bits. Para los tipos de datos Byte, Palabra y Palabra doble no se puede indicar un margen numérico. Se trata de combinaciones de bits con las cuales sólo se pueden crear expresiones booleanas.

Tipo	Palabra clave	Ancho en bits	Disposición	Margen permitido
Bit	BOOL	bit 1	comienza en el bit menos significativo del byte	0, 1 o FALSE, TRUE
Byte	BYTE	8 bits	comienza en el byte menos significativo de la palabra	-
Palabra	WORD	16 bits	comienza en un límite de palabra	-
Doble palabra	DWORD	32	comienza en un límite de palabra	-

7.2.2 Tipo de datos Carácter

Los datos de este tipo ocupan exactamente 1 carácter del juego de caracteres ASCII.

Tipo	Palabra clave	Ancho en bits	Margen permitido
Carácter	CHAR	8	Juego de caracteres ASCII ampliado

7.2.3 Tipos de datos numéricos

Los tipos de datos numéricos permiten procesar valores numéricos (p.ej., para calcular expresiones aritméticas).

Tipo	Palabra clave	Ancho en bits	Disposición	Margen permitido
Entero (número entero)	INT	16	comienza en el límite de una palabra	-32_768 hasta 32_767
Doble entero	DINT	32	comienza en el límite de una palabra	-2_147_483_648 hasta 2_147_483_647
Número en coma flotante (Número en coma flotante IEEE)	REAL	32	comienza en un el límite de una palabra	-3.402822E+38 hasta -1.175495E-38 +/- 0 1.175495E-38 hasta 3.402822E+38

7.2.4 Tipo de datos Tiempo

Dentro de STEP 7 los datos del tipo Tiempo representan los diferentes formatos para indicar la fecha y la hora (p.ej., para ajustar la fecha o para introducir un valor de temporización).

Tipo	Palabra clave	Ancho en bits	Disposición	Margen permitido
Tiempo S5	S5TIME S5T	16	Comienza en un límite de palabra	T#0H_0M_0S_10MS hasta T#2H_46M_30S_0MS
Duración: Tiempo IEC en intervalos de 1 ms.	TIME T	32	Comienza en un límite de palabra	T#24D_20H_31M_23S_647MS hasta T#24D_20H_31M_23S_647MS
Fecha: Fecha IEC en intervalos de 1 día.	DATE D	16	Comienza en un límite de palabra	D#1990-01-01 hasta D#2168-12-31
Hora del día: Hora en intervalos de 1 ms.	TIME_OF_DAY TOD	32	Comienza en un límite de palabra	TOD#0:0:0.0 hasta TOD#23:59:59.999

En las variables del tipo de datos S5TIME (tiempo S5) la resolución está limitada; es decir, sólo se dispone de las bases de tiempo 0.01 s, 0.1 s, 1 s y 10 s. El compilador redondea los valores como corresponda. Si el valor ajustado es mayor de lo que permite el margen de valores se utilizará el valor límite superior.

7.3 Tipos de datos compuestos

7.3.1 Tipo de datos DATE_AND_TIME

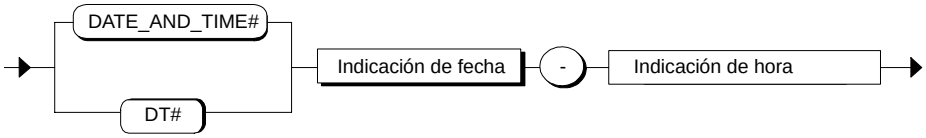
Definición

Este tipo de datos define un área de 64 bits (8 bytes) para indicar la fecha y la hora. El área de datos guarda las siguientes informaciones:

año, mes, día, horas, minutos, segundos, milisegundos

Sintaxis

DATE_AND_TIME



La sintaxis exacta para indicar la fecha y la hora está descrita en el apartado "Declaración de constantes".

Margen permitido

Tipo	Palabra clave	Ancho en bits	Disposición	Margen
Fecha y hora	DATE_AND_TIME DT	64	Comienza y acaba en el límite de una palabra	DT#1990-01-01:0:0:0.0 hasta DT#2089-12-31:23:59:59.999

El tipo de datos Date_And_Time se guarda en formato BCD:

Bytes	Contenido	Margen
0	Año	1990 ... 2089
1	Mes	01 ... 12
2	Día	1 ... 31
3	Hora	0 ... 23
4	Minuto	0 ... 59
5	Segundo	0 ... 59
6	2 MSD (most significant decade) de ms	00 ... 99
7 (4 MSB)	LSD (least significant decade) de ms	0 ... 9
7 (4 LSB)	Día de la semana	1 ... 7 (1 = domingo)

Ejemplo

Una definición válida para las 12 horas, 20 minutos, 30 segundos y 10 milisegundos del día 20.10.1995 sería:

```
DATE_AND_TIME#1995-10-20-12:20:30.10
DT#1995-10-20-12:20:30.10
```

Nota

Para poder acceder directamente a los componentes DATE o TIME se dispone de funciones estándar (en la librería de STEP 7).

7.3.2 Tipo de datos STRING

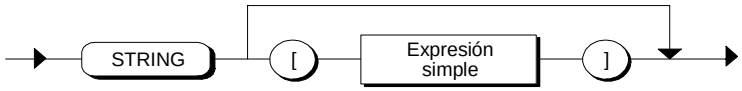
Definición

Un tipo de datos STRING define una cadena de caracteres de 254 caracteres como máximo. El área estándar reservada para una cadena de caracteres se compone de 256 bytes, que es el área de memoria necesaria para guardar 254 caracteres y un encabezado de 2 bytes.

El espacio de memoria reservado para una cadena de caracteres se puede reducir definiendo el número máximo de caracteres que deben guardarse en la cadena. Una cadena cero (es decir, una cadena sin contenido) representa el valor más pequeño posible.

Sintaxis

Especificación de tipo de datos STRING



Dimensión de la cadena

La expresión simple representa el número máximo de caracteres del STRING. En una cadena de caracteres se pueden utilizar todos los caracteres del código ASCII. La cadena también puede contener caracteres especiales, p. ej., caracteres de control, y caracteres no imprimibles. Estos se pueden introducir con la sintaxis $hh, en la cual hh sustituye al valor el carácter ASCII expresado en hexadecimal (ejemplo: '$0D$0AText')

Al declarar el espacio de memoria de las cadenas de caracteres se puede definir el número máximo de caracteres que se deben guardar en la cadena. Si aquí no se indica la longitud máxima, se creará una cadena de caracteres de una longitud de 254.

Ejemplo:

```
VAR
        Text1        : Cadena de caracteres [123];
        Text2        : Cadena de caracteres;
END_VAR
```

La constante "123" en la declaración de la variable "Text1" indica el número máximo de caracteres de la cadena. En la variable "Text2" se reserva una longitud de 254 caracteres.

Nota

En los parámetros de salida y en los de entrada/salida, así como en los valores de retorno de funciones, es posible reducir el margen estándar reservado de 254 caracteres para poder aprovechar mejor los recursos de la CPU. Para reducir este margen elija el comando de menú **Herramientas > Preferencias** y, en el cuadro de diálogo que aparece a continuación, la ficha "Compilador".

A continuación indique el número deseado de caracteres en la opción "Número máx. de caracteres". Tenga en cuenta, que el ajuste afectará a todas las variables STRING de la fuente. Por esta razón, el valor ajustado no debe ser menor que las variables STRING utilizadas en el programa.

Inicialización de cadenas de caracteres

Al igual que otras variables, las variables STRING se pueden inicializar con cadenas de caracteres constantes en la declaración de parámetros de bloques de función (FBs). En los parámetros de funciones (FCs) no se pueden inicializar.

Si la cadena de caracteres asignada por defecto es más corta que la longitud máxima declarada no se ocuparán las posiciones restantes. Al continuar procesando las variables sólo se tendran en cuenta las posiciones actualmente asignadas.

Ejemplo:

x : STRING[7]:='Dirección';

En caso de que se requieran variables temporales del tipo STRING para, por ejemplo, memorizar resultados de forma intermedia, entonces, antes de utilizarlos por primera vez, es imprescindible definirlos con una constante STRING en la declaración de variables o en una posterior asignación con un valor de inicialización.

Nota

Si una función de la librería estándar suministra un valor de retorno del tipo de datos STRING, y si este valor debe ser asignado a una variable temporal, entonces se deberá inicializar la variable en primer lugar.

Ejemplo:

```
FUNCTION Test : STRING[45]
VAR_TEMP
    x : STRING[45];
END_VAR
x := 'a';
x := concat (in1 := x, in2 := x);
Test := x;
END_FUNCTION
```

Sin la inicialización **x := 'a';** la función suministraría un resultado erróneo.

Disposición

Las variables del tipo STRING empiezan y acaban en un límite de palabra.

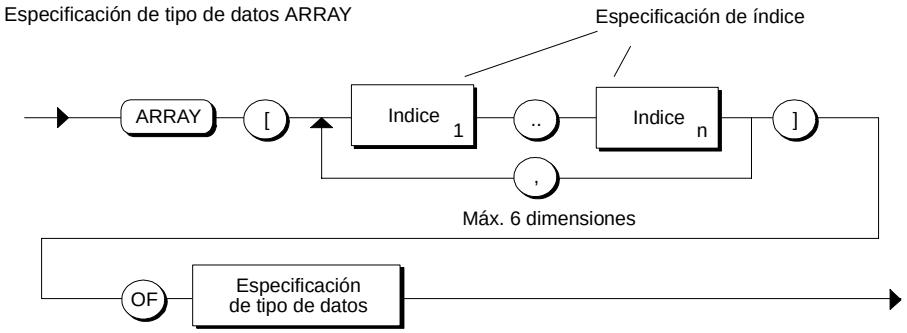
7.3.3 Tipo de datos ARRAY

Definición

Los arrays tienen un número fijo de componentes de un único tipo de datos. S7-SCL distingue los siguientes:

- el tipo de array unidimensional: una lista de elementos de datos dispuestos en orden ascendente;
- el tipo de array bidimensional: una tabla de datos compuesta por filas y columnas. La primera dimensión se refiere al número de línea, y la segunda al número de columna;
- el tipo de array de dimensión superior: una ampliación del tipo de array bidimensional, que adopta más dimensiones. Como máximo se admiten 6 dimensiones

Sintaxis



Especificación de índice

Describe las dimensiones del tipo de datos ARRAY:

- con el índice mínimo y máximo posibles (rango de índice) para cada dimensión. El índice puede ser un número entero cualquiera (de -32768 hasta 32767).
- Los límites deben indicarse separados por dos puntos. Los rangos de índice se separan entre sí por comas.
- La especificación completa del índice se debe indicar entre corchetes.

Especificación del tipo de datos

Al especificar el tipo de datos se declara el tipo de datos de los componentes. Para la especificación están permitidos todos los tipos de datos. Los datos de un array (matriz) también puede ser del tipo STRUCT. Los tipos de parámetros no se deben utilizar como tipo de elemento de un array.

Ejemplo

```
VAR
        Reguladorl :
        ARRAY[1..3,1..4] OF INT:=  -54,  736,  -83,  77,

-1289,     10362,      385,  2,

60,  -37,  -7,   103 ;
        Regulador2 : ARRAY[1..10] OF REAL ;
END_VAR
```

Disposición

Las variables del tipo ARRAY se crean línea por línea. Cada dimensión de una variable del tipo BOOL, BYTE o CHAR acaban en un límite de BYTE, todas las demás en un límite de WORD.

7.3.4 Tipo de datos STRUCT

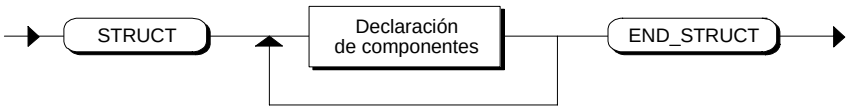
Definición

El tipo de datos STRUCT describe un margen que consiste en un número fijo de componentes, que pueden variar en cuanto al tipo de datos. Estos elementos se indican en la declaración de componentes inmediatamente después de la palabra clave STRUCT.

En particular, un elemento del tipo de datos STRUCT puede ser, a su vez, un tipo de datos compuesto. Por lo tanto, está permitido anidar los tipos de datos STRUCT.

Sintaxis

Especificación de tipo de datos STRUCT

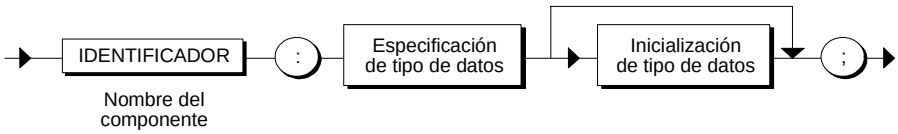


Declaración de componentes

La declaración de componentes es una lista de los distintos componentes del tipo de datos STRUCT. Está compuesta por:

- De 1 a n identificadores con el tipo de datos que tienen asignado y
- una inicialización opcional con valores iniciales.

Declaración de componentes



El identificador es el nombre de un elemento de estructura al que se aplica la siguiente especificación del tipo de datos.

En la especificación del tipo de datos se admiten todos los tipos de datos, excepto los tipos de parámetros.

Después de la especificación del tipo de datos se puede asignar opcionalmente un valor inicial a un elemento de la estructura. Esta correspondencia se establece por medio de una asignación de valores.

Ejemplo

```
VAR
        MOTOR : STRUCT
                DATOS : STRUCT
                          CORRIENTE_DE_CARGA  : REAL ;
                          TENSION : INT  := 5 ;
                END_STRUCT ;
        END_STRUCT ;
END_VAR
```

Disposición

Las variables del tipo STRUCT empiezan y terminan en un límite de WORD.

Atención

En caso de definir una estructura que no termine en un límite WORD, S7-SCL rellenará automáticamente los bytes que faltan, adaptando así el tamaño a la estructura.

La adaptación del tamaño de la estructura puede causar conflictos al acceder a tipos de datos de una longitud de bytes impar.

7.4 Tipos de datos de usuario

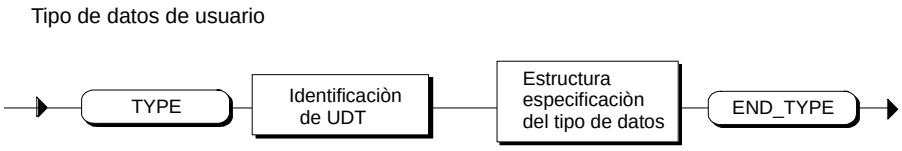
7.4.1 Tipos de datos de usuario (UDT)

Definición

El tipo de datos de usuario (UDT) se define como bloque. Después de definirlo se puede utilizar en todo el programa de usuario y, por lo tanto, se trata de un tipo de datos global. Este tipo de datos se puede utilizar con su identificador UDTx (donde x es un número) o con un nombre simbólico asignado, en el área de declaración de un bloque o de un bloque de datos.

El tipo de datos de usuario se puede utilizar para declarar variables, parámetros, bloques de datos y otros tipos de datos de usuario. Mediante el tipo de datos de usuario también se pueden declarar componentes de arrays o estructuras.

Sintaxis



Identificador UDT

Una declaración del tipo de datos de usuario se inicia mediante la palabra clave TYPE, seguida del nombre del tipo de datos de usuario (identificador UDT). El nombre del tipo de datos de usaurio se puede indicar de forma absoluta, es decir mediante el nombre estándar UDTx (donde x es un número), o bien mediante un nombre simbólico.

Ejemplos:

```
TYPE UDT10
TYPE VALORES_MEDIDOS
```

Especificación del tipo de datos

El identificador del UDT va seguido de la especificación del tipo de datos. Aquí sólo está permitida la especificación del tipo de datos STRUCT:

```
STRUCT
:
END_STRUCT
```

Nota

Dentro del tipo de datos de usuario rige la sintaxis de AWL. Esto vale, por ejemplo, para la notación de constantes y para la asignación de valores iniciales (inicialización). Para más información sobre la notación de constantes, consulte la Ayuda en pantalla de AWL.

Ejemplo

```
// Definición UDT con nombre simbólico
TYPE
VALORES_MEDIDOS: STRUCT
                        BIPOL_1 : INT := 5;
                        BIPOL_2 : WORD := W#16#FFAA ;
                        BIPOL_3 : BYTE  := B#16#F1 ;
                        BIPOL_4 : WORD := W#16#1919 ;
                        MEDICION: STRUCT
                                BIPOLAR_10V : REAL ;
                                UNIPOLAR_4_20MA : REAL ;
                            END_STRUCT;
                    END_STRUCT;
END_TYPE

//Uso de UDT en un FB
FUNCTION_BLOCK FB10
VAR
     MARGEN_MEDICION: VALORES DE MEDICION;
END_VAR
BEGIN
        // . . .
        MARGEN_MEDICION.BIPOL_1 := -4 ;
        MARGEN_MEDICION.MEDICION.UNIPOLAR_4_20MA  :=  2.7 ;
        // . . .
END_FUNCTION_BLOCK
```

7.5 Tipos de datos para parámetros

Para definir los parámetros formales de bloques FB y FC se pueden utilizar los tipos de parámetros, aparte de los tipos de datos que ya hemos comentado.

Parámetro	Tamaño	Descripción
TIMER	2 bytes	Indica un determinado temporizador que será utilizado por el programa en el bloque lógico llamado. Parámetro actual: p.ej. T1
COUNTER	2 bytes	Indica un determinado contador que será utilizado por el programa en el bloque lógico llamado. Parámetro actual: p.ej. Z10
BLOCK_FB BLOCK_FC BLOCK_DB BLOCK_SDB	2 bytes	Indica un determinado bloque que deberá ser utilizado por un bloque de código AWL llamado. Parámetro actual: p.ej. FC101, DB42 **Notas:** Al tipo de datos BLOCK_DB se puede acceder de forma absoluta (`myDB.dw10`). También puede seguir procesándose con BLOCK_DB_TO_WORD(). Los tipos de datos BLOCK_SDB, BLOCK_FB y BLOCK_FC no pueden ser interpretados por programas de S7-SCL. Por consiguiente sólo se pueden utilizar para transferir valores a parámetros de este tipo en las llamadas de bloques AWL.
ANY	10 bytes	Se utiliza cuando como tipo de datos del parámetro actual debe permitirse cualquier tipo de datos, a excepción de ANY.
POINTER	6 bytes	Identifica una determinada área de memoria que será utilizada por el programa. Parámetro actual: p.ej. M50.0

7.5.1 Tipos de datos TIMER y COUNTER

Estos tipos de datos definen un temporizador o un contador determinado que se utilizará al procesar el bloque. Los tipos de datos TIMER y COUNTER sólo se deben utilizar para parámetros de entrada (VAR_INPUT).

7.5.2 Tipo de datos BLOCK

Los tipos de datos BLOCK definen un bloque determinado que se utilizará como parámetro de entrada. La declaración del parámetro de entrada define el tipo de bloque (FB, FC, DB). Al asignar parámetros, es preciso indicar el identificador del bloque, sea de forma absoluta o de forma simbólica.

Al tipo de datos BLOCK_DB se puede acceder mediante direccionamiento absoluto (`myDB.dw10`). Para los demás tipos de datos BLOCK, S7-SCL no ofrece ningún tipo de operación. En las llamadas de bloques solamente se pueden transferir valores a parámetros de este tipo. En las funciones no es posible transferir un parámetro de entrada.

En S7-SCL se puede asignar operandos de los siguientes tipos de datos como parámetros actuales:

- Bloques de función sin parámetros formales.
- Funciones sin parámetros formales ni valor de respuesta (funciones VOID).
- Bloques de datos y bloques de datos de sistema.

7.5.3 Tipo de datos POINTER

Al tipo de datos POINTER se le pueden asignar variables declaradas como parámetros formales de un bloque. Al llamar a un bloque de este tipo se pueden asignar a estos parámetros operandos de cualquier tipo de datos (excepto ANY).

Sin embargo, S7-SCL sólo ofrece una instrucción para procesar el tipo de datos POINTER: transferirlo a los bloques subordinados.

Tipos de operandos que se pueden asignar como parámetros actuales:

- Direcciones absolutas
- Nombres simbólicos
- Operandos del tipo de datos POINTER:
  Sólo es posible cuando el operando es un parámetro formal de tipo compatible.
- Constante NIL:
  indique un puntero cero.

Restricciones

- El tipo de datos POINTER se puede utilizar para parámetros formales de entrada, para parámetros de entrada/salida de FB y FC y para parámetros de salida de FCs. Las constantes no se pueden utilizar como parámetros actuales (excepto la constante NIL).
- Si al llamar un FB o una FC asigna una variable temporal a un parámetro formal del tipo POINTER no se podrá transferir este parámetro a otro bloque. Las variables temporales pierden su validez al pasarlas a otro bloque.
- Al llamar una FC o un FB, las entradas del proceso (%PEW) sólo pueden ser asignadas a parámetros formales del tipo Pointer, siempre y cuando el parámetro formal haya sido declarado como parámetro de entrada.
- Al llamar un FB, las salidas del proceso (%PAW) sólo pueden ser asignadas a parámetros formales del tipo Pointer, siempre y cuando el parámetro formal haya sido declarado como parámetro de salida.

Ejemplo

```
FUNCTION FC100 : VOID
VAR_IN_OUT
        N_out : INT;
        out      : POINTER;
END_VAR
VAR_TEMP
        ret       : INT;
END_VAR
BEGIN
        // ...
        ret := SFC79(N := N_out, SA := out);
        // ...
END_FUNCTION

FUNCTION_BLOCK FB100
VAR
        ii   : INT;
        aa : ARRAY[1..1000] OF REAL;
END_VAR

BEGIN
        // ...
        FC100( N_out := ii, out := aa);
        // ...
END_FUNCTION_BLOCK
```

7.6 Tipo de datos ANY

En S7-SCL, las variables del tipo de datos ANY se declaran de la siguiente forma:

- como parámetros formales de un bloque, pudiendo éstos recibir los parámetros actuales (=reales) de cualquier tipo de datos en la llamada del bloque.
- como variables temporales, pudiendo asignar a estas variables valores de cualquier tipo de datos.

A continuación se indican los datos que se pueden utilizar como valores actuales (=reales) o que se pueden introducir a la derecha de una asignación:

- Variables locales y globales
- Variables del DB (con dirección absoluta o simbólica)
- Variables de la instancia local (con dirección absoluta o simbólica)
- Constante NIL:
  Indique un puntero cero.
- Tipo de datos ANY
- Temporizadores, contadores y bloques:
  indique el identificador (p.ej., T1, Z20 o FB6).

Restricciones

- El tipo de datos ANY se puede utilizar para parámetros formales de entrada, para parámetros de entrada/salida de FBs y FCs y para parámetros de salida de FCs. Las constantes no se pueden utilizar como parámetros actuales, por lo que no pueden figurar a la derecha de una asignación (a excepción de la constante NIL).
- Si al llamar a un FB o FC asigna una variable temporal a un parámetro formal del tipo ANY, no podrá pasar dicho parámetro a otro bloque. Las variables temporales pierden su validez al pasarlas a otro bloque.
- Las variables de este tipo no se deben utilizar ni como componente de una estructura ni como elemento de un array.
- Al llamar una FC o un FB, las entradas del proceso (%PEW) sólo pueden ser asignadas a parámetros formales del tipo ANY, siempre y cuando el parámetro formal haya sido declarado como parámetro de entrada.
- Al llamar un FB, las salidas del proceso (%PAW) sólo pueden ser asignadas a parámetros formales del tipo ANY, siempre y cuando el parámetro formal haya sido declarado como parámetro de salida.

7.6.1 Ejemplo del tipo de datos ANY

```
VAR_INPUT
        iANY : ANY;
END_VAR

VAR_TEMP
        pANY : ANY;
END_VAR


CASE ii OF
1:
   pANY := MW4;         // pANY contiene la dirección de MW4

3..5:
   pANY:= aINT[ii];     // pANY contiene la dirección del ii-avo
                        // elemento del array aINT;
100:
   pANY := iANY;        // pANY contiene el valor de la variable
                        // de entrada iANY
ELSE
   pANY := NIL;         // pANY contiene el valor del puntero NIL
END_CASE;

SFCxxx(IN := pANY);
```

8 Declaración de variables y parámetros locales

8.1 Variables locales y parámetros de bloque

Categorías de variables

La siguiente tabla muestra en qué categorías se pueden dividir las variables locales:

Variable	Significado
Variables estáticas	Las variables estáticas son variables locales cuyo valor se conservaa lo largo de todos los recorridos del bloque (memoria de bloque). Sirven para guardar los valores de un bloque de función y se almacenan en el bloque de datos de instancia.
Variables temporales	Las variables temporales se corresponden localmente con un bloque lógico y no ocupan área de memoria estática, puesto que se depositan en la pila de la CPU. Su valor sólo se conserva durante un recorrido del bloque. A las variables temporales no se puede acceder fuera del bloque en el que se han declarado las variables.

Categorías de parámetros de bloque

Los parámetros de bloque son comodines que se fijan con el uso concreto (llamada) del bloque. Los comodines existentes en el bloque se denominan parámetros formales, y los valores asignados cuando se llama al bloque se denominan parámetros actuales. Los parámetros formales de un bloque pueden considerarse como variables locales.

Los parámetros de bloque se pueden dividir en las siguientes categorías:

Parámetros de bloques	Significado
Parámetro de entrada	Los parámetros de entrada asumen los valores de entrada actuales cuando se llama al bloque. Sólo admiten lectura.
Parámetros de salida	Los parámetros de salida transmiten los valores de salida actuales al bloque invocante. Pueden escribirse y leerse.
Parámetros de entrada/salida	Los parámetros de entrada/salida adquieren valores de entrada actuales al llamar un bloque. Tras el procesamiento del valor, reciben el resultado y lo transfieren al bloque invocante.

OK flag (marca OK)

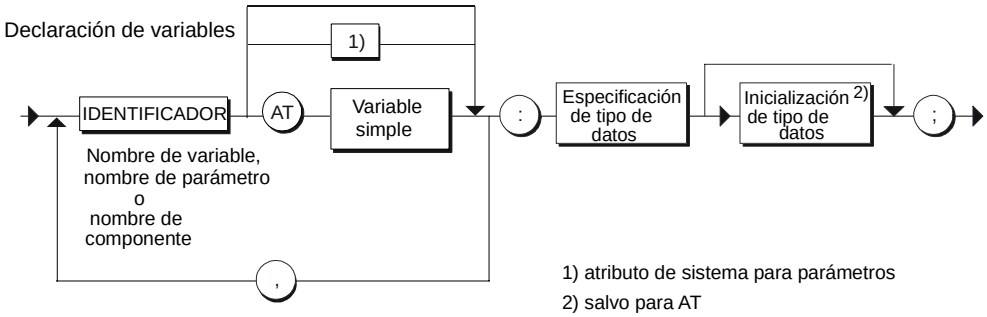
El compilador S7-SCL dispone de una marca que sirve para identificar errores durante la ejecución de programas en la CPU. Es una variable local del tipo BOOL con el nombre predefinido "OK".

8.2 Sintaxis general de una declaración de variables o de parámetros

Las variables y los parámetros de bloques se deben declarar por separado antes de poder utilizarlos en un bloque lógico o en un bloque de datos. En la declaración se determina que se utilizará un identificador como parámetro del bloque o como variable y además se le asigna un tipo de datos.

Una declaración de variable o de parámetro se compone de un identificador discrecional y de un tipo de datos. La forma general se muestra en el diagrama sintáctico.

Sintaxis de la declaración de variables o de parámetros



Ejemplos

```
VALOR1      :      REAL;
si hubiera más variables del mismo tipo:

VALOR2, VALOR3,VALOR4,....: INT;
ARRAY       :      ARRAY[1..100, 1..10] OF REAL;
FRASE       :      STRUCT
                                ARRAY_DE_MEDICION:ARRAY[1..20] OF
REAL;
                                PULSADOR:BOOL;
                           END_STRUCT
```

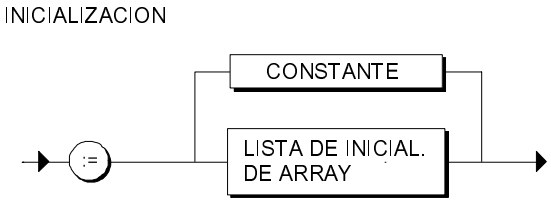
Nota

Para utilizar palabras reservadas como identificadores hay que anteponerles el caracter "#" (p.ej. #FOR).

8.3 Inicialización

Las variables estáticas así como los parámetros de entrada y de salida de un FB se pueden inicializar con un valor en la declaración. También es posible inicializar parámetros de entrada/salida, siempre y cuando sean del tipo de datos simple. En las variables simples esta inicialización se realiza por asignación (:=) de una constante después de la indicación del tipo de datos.

Sintaxis



Ejemplo

```
VALOR       :REAL :=
20.25;
```

Nota

No es posible inicializar una lista de variables (A1, A2, A3,...: INT:=...). En este caso es preciso inicializar las variables una a una.

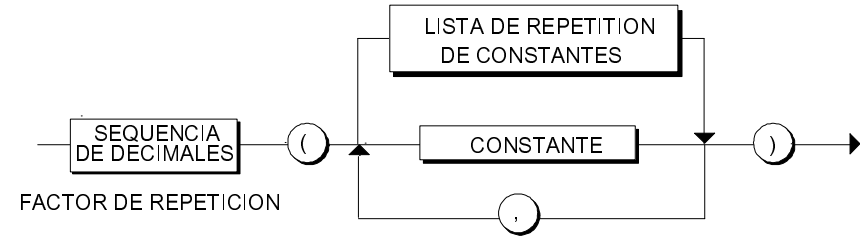
Inicialización de arrays

En la inicialización de ARRAYs se puede indicar un valor separado por coma para cada componente del array o bien inicializar varios componentes con el mismo valor anteponiendo un factor de repetición (integer).

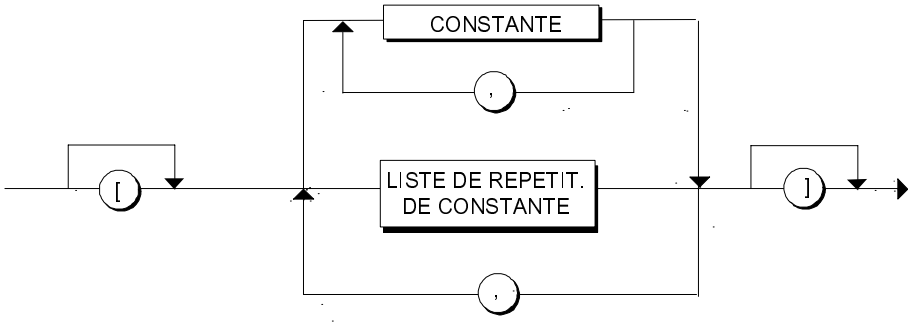
Opcionalmente, los valores iniciales pueden ir encerrados en un paréntesis. También en los arrays (matrices) multidimensionales se indica solamente un par de paréntesis.

Sintaxis de la inicialización de arrays

LISTA DE REPETITION DE CONSTANTES



LISTA DE INICIALIZACION DE ARRAY



Ejemplos

```
VAR
// Inicialización de variables estáticas:
           INDEX1 : INT := 3 ;
//Inicialización de arrays:
   REGULADOR1 : ARRAY [1..2, 1..2] OF INT := -54, 736, -83, 77;
   REGULADOR2 : ARRAY[1..10] OF REAL := 10(2.5);
   REGULADOR1 : ARRAY [1..2, 1..2] OF INT := [-54, 736, -83, 77];
   REGULADOR2 : ARRAY[1..10] OF REAL := [10(2.5)];
//Inicialización de estructura:
     GENERADOR:  STRUCT
       DAT1 :    REAL               := 100.5;
       A1 :      INT                := 10 ;
       A2 :      STRING[6]          := 'FACTOR';
       A3 :      ARRAY[1..12] OF REAL:= 0.0, 10(100.0), 1.0;
            END_STRUCT ;
END_VAR
```

8.4 Declarar vistas sobre áreas de variables

Para acceder a una variable declarada con otro tipo de datos, puede definir vistas sobre la variable o sobre áreas de la variable mediante la palabra clave "AT". Una vista sólo se puede ver localmente en el bloque, ya que no se integra en el interface. La vista se usa del mismo modo que cualquier otra variable del bloque, heredando todas las propiedades de la variable a la que señala, pero no su tipo de datos.

Ejemplo

En el siguiente ejemplo se crean varias vistas sobre un parámetro de entrada:

```
VAR_INPUT
        Buffer : ARRAY[0..255] OF BYTE;
        Telegram1 AT Buffer : UDT100 ;
        Telegram2 AT Buffer : UDT200 ;
END_VAR
```

El bloque invocante transfiere valores al parámetro Buffer y no ve los nombres Telegram1 y Telegram2. El bloque invocante dispone de tres posibilidades para interpretar los datos: como array con el nombre Buffer o con distinta estructura con el nombre Telegram1 o Telegram2.

Reglas

- La declaración de otra vista sobre una variable debe realizarse en el mismo bloque de declaración después de la declaración de la variable a la que señala.
- No se puede inicializar.
- El tipo de datos de la vista debe ser compatible con el tipo de datos de la variable. La variable determina el tamaño del área de memoria. La memoria que requiere la vista puede ser igual o menor. Además, rigen las siguientes reglas de combinación:

		Tipo de datos de la vista:	**Tipo de datos de la variable:**		
			simple	**compuesto**	**ANY/POINTER**
FB	Declaración de una vista en `VAR, VAR_TEMP, VAR_IN` o `VAR_OUT`	**simple** **compuesto** **ANY/POINTER**	x x	x x x (1)	x (1)
	Declaración de una vista en `VAR_IN_OUT`	**simple** **compuesto** **ANY/POINTER**	**x**	**x**	
FC	Declaración de una vista en `VAR` o `VAR_TEMP`	**simple** **compuesto** **ANY/POINTER**	**x** **x**	**x** **x** **x**	**x**
	Declaración de una vista en `VAR_IN, VAR_OUT` o `VAR_IN_OUT`	**simple** **compuesto** **ANY/POINTER**	**x**	**x**	

(1) En VAR_OUT no está permitido el Any_Pointer .

Simple = BOOL, BYTE, WORD, DWORD, INT, DINT, DATE, TIME, S5TIME, CHAR
Compuesto = ARRAY, STRUCT, DATE_AND_TIME, STRING

8.5 Uso de multiinstancias

Según los datos característicos que tengan las CPUs S7 utilizadas (p.ej. la capacidad de memoria) puede ocurrir que no quiera o no pueda crear más que un número limitado de bloques de datos para datos de instancia. Si en su programa de usuario, se llaman desde un FB otros bloques de función ya disponibles (jerarquía de llamada de FB), entonces puede llamar a estos otros bloques de función sin su DB de instancia (es decir, sin uno adicional).

Solución posible:

- Incluya en la declaración de variables del FB invocante los FBs a llamar como variables estáticas.
- Desde este bloque de función llame otros bloques de función sin su DB de instancia.
- De esta forma podrá concentrar los datos de instancia en un bloque de datos de instancia, es decir, podrá aprovechar mejor la cantidad de DBs disponible.

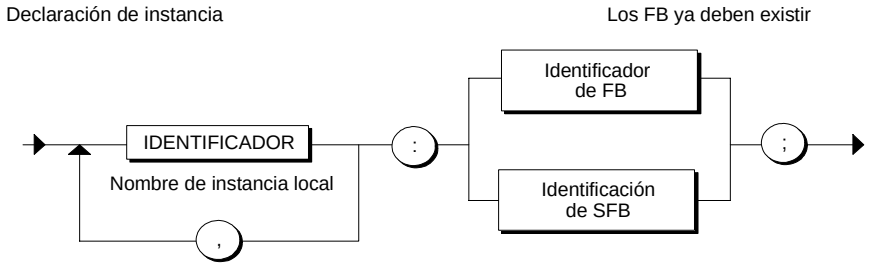
8.6 Declaración de instancias

En el bloque de declaración de variables estáticas (VAR; END_VAR) de los bloques de función es posible declarar variables del tipo FB o SFB además de las variables de tipos de datos simples, compuestos o de usuario . Este tipo de variables se denominan instancias locales del FB o del SFB.

Los datos locales de instancia se guardan en el bloque de datos de instancia del FB invocante. No es posible realizar una inicialización local específica de la instancia.

Los bloques que se llaman como instancias locales no pueden ser de longitud 0. En un bloque de tales características tiene que declararse por lo menos una variable estática o un parámetro.

Sintaxis



Ejemplo

```
Asignacion1      : FB10;
Asignacion2,Asignacion3,Asignacion4      : FB100;
Motor1     : Motor ;
```

En este caso, `Motor` es un símbolo registrado en la tabla de símbolos que representa a un FB.

8.7 Flags (OK flag)

La marca OK indica si el bloque ha sido ejecutado correctamente. Se trata de una variable local de tipo BOOL con el nombre predefinido "OK".

Al iniciar la ejecución del programa, la marca OK tiene el valor TRUE. En cualquier punto del bloque se puede consultar mediante instrucciones S7-SCL si se debe cambiar a TRUE / FALSE. Si durante la ejecución de una operación (p.ej. una división entre cero) se produce un error, OK-Flag cambia a FALSE. Al abandonar el bloque, el valor de la marca OK se guarda en el parámetro de salida ENO, por lo que el bloque invocante puede evaluarlo.

Declaración

OK-Flag es una variable declarada por el sistema, por lo que no es necesario declararla. No obstante, debe seleccionar la opción del compilador "Activar OK flag" antes de compilar, si desea utilizar esta marca en su programa de usuario.

Ejemplo

```
// Cambiar OK flag a TRUE
// para poder comprobar
// si la acción se desarrolla correctamente.
OK:= TRUE;
Division:= 1 / IN;
IF OK THEN
           // La división se desarrolla correctamente.

           // :
           // :

ELSE       // La división es errónea.

           // :

END_IF;
```

8.8 Secciones de declaración

8.8.1 Sinopsis de las secciones de declaración

Cada categoría de variables locales o de parámetros tiene asignado un bloque de declaración propio, que se distingue por una pareja de palabras clave. Cada bloque contiene las declaraciones que admite dicho bloque de declaración. La secuencia de estos bloques es arbitraria.

La tabla siguiente muestra los tipos de variable o parámetro que se pueden declarar en los distintos bloques lógicos:

Datos	**Sintaxis**	**FB**	**FC**	**OB**
Variables como: variable estática	VAR . . . END_VAR	X	X *)	
Variable temporal	VAR_TEMP . . . END_VAR	X	X	X
Parámetros de bloque como: parámetro de entrada	VAR_INPUT . . . END_VAR	X	X	
parámetro de salida	VAR_OUTPUT . . . END_VAR	X	X	
parámetro de entrada/salida	VAR_IN_OUT . . . END_VAR	X	X	

*) En las funciones se admite la declaración de variables dentro del par de palabras clave VAR y END_VAR; sin embargo, las declaraciones se transfieren al área temporal al compilar la fuente.

8.8.2 Variables estáticas

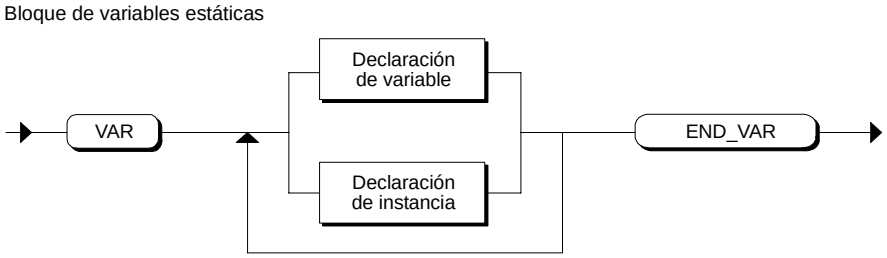
Las variables estáticas son variables locales que mantienen su valor después de recorrer todos los bloques (memoria de bloque). Sirven para guardar valores de un bloque de función y se guardan en el bloque de datos de instancia correspondiente.

Sintaxis

Las variables estáticas se declaran en la sección de declaración VAR / END_VAR. Este bloque de declaración forma parte del área de declaración del FB. Después de compilarlo, este bloque determina, junto con los bloques de parámetros de bloque, la estructura del bloque de datos de instancia asignado.

En este bloque se pueden

- crear variables, asignarles tipos de datos e inicializarlas.
- declarar un FB invocante como variable estática para poderlo llamar desde el FB actual como instancia local.



Ejemplo

```
VAR
EJECUCION         :INT;
ARRAY_DE_MEDICION      :ARRAY [1..10] OF REAL;
PULSADOR          :BOOL;
MTOR_1,MOTOR_2    :FB100;     //Declaración de instancia

END_VAR
```

Acceso

El acceso a las variables se realiza en el área de instrucciones:

- **Acceso desde el interior del bloque**: a la variable se accede desde el área de instrucciones del bloque de función, en cuya área de declaración se ha declarado una variable. El procedimiento completo se explica en el capítulo "Asignación de valores".
- **Acceso desde el exterior a través del DB de instancia**: a la variable se accede desde otros bloques mediante un acceso indexado, p.ej. *DBx.variable.*

8.8.3 Variables temporales

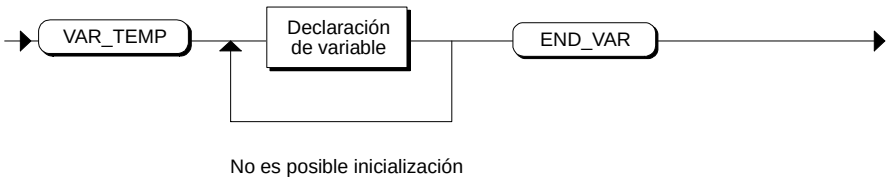
Las variables temporales pertenecen localmente a un bloque lógico y no ocupan ningún área de memoria estática. Se depositan en la pila de la CPU y su valor se conserva solamente durante un recorrido del bloque. A las variables temporales no se puede acceder fuera del bloque en el en el que se han declarado. Al iniciar la ejecución de un OB, FB o FC, el valor de los datos temporales no está definido. Las variables temporales no se pueden inicializar.

Declare como datos temporales aquellos datos que necesite solamente para guardar resultados intermedios durante el procesamiento de su OB, FB o FC.

Sintaxis

La declaración de las variables temporales se realiza en la sección de declaración VAR_TEMP / END_VAR. Este bloque de declaración forma parte de un FB, FC, o OB. En la declaración de variables se indican los nombres de las variables y sus tipos de datos.

Bloque de variables temporales



No es posible inicialización

Ejemplo

```
VAR_TEMP
        BUFFER 1 : ARRAY [1..10] OF INT ;
        AYUD1, AYUD2   : REAL;
END_VAR
```

Acceso

El acceso a las variables se realiza siempre en el área de instrucciones del bloque lógico, en cuyo área de se haya declarado la variable (acceso desde el interior), consulte el capítulo "Asignación de valores".

8.8.4 Parámetros de bloques

Los parámetros son comodines que se definen sólo al utilizar (llamada) el bloque. Los comodines (declarados) que residen en el bloque se denominan parámetros formales; los valores asignados cuando se llama al bloque, parámetros actuales. Por consiguiente, los parámetros constituyen un mecanismo de intercambio de información entre los bloques.

Tipos de parámetros de bloque

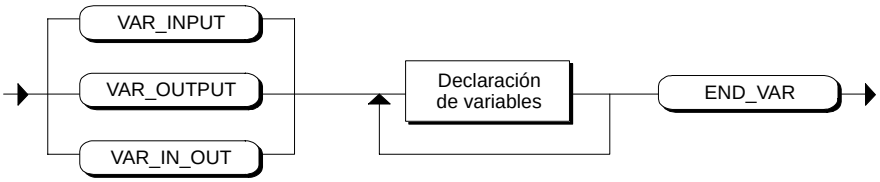
- Los parámetros formales de entrada adquieren los valores de entrada actuales (flujo de datos del exterior al interior).
- Los parámetros formales de salida sirven para tranferir valores de salida (flujo de datos del interior al exterior).
- Los parámetros formales de entrada/salida desempeñan tanto la función de un parámetro de entrada como de un parámetro de salida.

Sintaxis

La declaración de los parámetros formales se realiza en el área de declaración de un bloque de función o de una función, por separado para cada tipo de parámetro en los tres bloques de declaración de parámetros. En la declaración de variables se puede definir el nombre del parámetro y su tipo de datos. Su inicialización sólo es posible en los parámetros de entrada y en los parámetros de salida de un FB.

Al declarar parámetros formales se pueden utilizar los tipos de datos para parámetros además de los tipos de datos simples, compuestos y los de usuario.

Bloque de parámetros



La inicialización sólo es posible para VAR_INPUT y VAR_OUTPUT.

Ejemplo

```
VAR_INPUT        // parámetro de entrada
        MI_DB           : BLOCK_DB ;
        REGULADOR       : DWORD ;
        HORA            : TIME_OF_DAY ;
END_VAR

VAR_OUTPUT       // parámetro de salida
        VALORES_TEORICOS: ARRAY [1..10] of INT ;
END_VAR

VAR_IN_OUT       // parámetro de entrada/salida
        AJUSTE : INT ;
END_VAR
```

Acceso

El acceso a los parámetros de bloque se realiza en el área de instrucciones de un bloque lógico:

- **Acceso desde el interior:** es decir, en el área de instrucciones del bloque en cuya área de declaración se han declarado los parámetros. Este procedimiento se explica en los capítulos "Asignación de valores" y "Expresiones, operaciones y operandos".
- **Acceso desde el exterior a través de un DB de instancia**: A los parámetros de bloque de los bloques de función se accede a través de los DB de instancia asignados.

9 Declaración de constantes y saltos

9.1 Constantes

Las constantes son datos que poseen un valor fijo que no se puede modificar durante el tiempo de ejecución del programa.

S7-SCL distingue los siguientes grupos de constantes.

- Constantes de bits
- Constantes numéricas
  - Constantes de enteros
  - Constantes de números reales
- Constantes de caracteres
  - Constantes CHAR
  - Constantes STRING
- Indicación de tiempos
  - Constantes de fecha
  - Constantes de tiempo
  - Constantes de hora
  - Constantes de fecha y hora

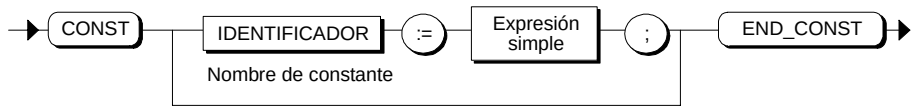
9.1.1 Declaración de nombres simbólicos para constantes

Las constantes no necesitan ser declaradas. Sin embargo, es posible asignar nombres simbólicos a las constantes mediante en el área de declaración.

Los nombres simbólicos para constantes se declaran en el área de declaración del bloque lógico con la instrucción CONST. Es recomendable hacerlo para todas las constantes del bloque, pues ello mejora la legibilidad y simplifica la gestión del bloque en caso de modificar los valores constantes.

Sintaxis

Bloque de constantes



En las expresiones simples sólo se admiten las operaciones aritméticas básicas (*, /, +, -, DIV, MOD).

Ejemplo

```
CONST
    Numero := 10 ;
    HORA   := TIME#1D_1H_10M_22S_2MS ;
    NOMBRE := 'SIEMENS' ;
    NUMERO2      := 2 * 5 + 10 * 4 ;
    NUMERO3      := 3 + NUMERO2 ;
END_CONST
```

9.1.2 Tipos de datos de las constantes

La asignación de tipos de datos a constantes se distingue del procedimiento utilizado en AWL:

Una constante recibe su tipo de datos sólo en la combinación aritmética o lógica en la que se va a utilizar, p.ej.:

```
Int1:=Int2 + 12345          //"12345" recibe el tipo de datos INT
Real1:=Real2 + 12345        //"12345" recibe el tipo de datos REAL
```

De esta forma, a la constante se le asigna el tipo de datos cuyo margen sea suficiente para abarcar la constante sin que ésta experimente ninguna pérdida de valor. Por ejemplo, la constante "12345" no tiene siempre el tipo de datos INT como en AWL, sino que tiene la clase de tipo de datos ANY_NUM, es decir, dependiendo de su uso es INT, DINT, o REAL.

Constantes tipificadas

La notación de constantes tipificada permite asignar explícitamente un tipo de datos con los siguientes tipos de datos numéricos.

Ejemplos:

Tipo de datos	**Notación tipificada**
BOOL	BOOL#1 bool#0 Bool#false BOOL#TRUE
BYTE	BYTE#0 B#2#101 Byte#'ä' b#16#f
WORD	WORD#32768 word#16#f W#2#1001_0100 WORD#8#177777
DWORD	DWORD#16#f000_0000 dword#32768 DW#2#1111_0000_1111_0000 DWord#8#37777777777
INT	INT#16#3f_ff int#-32768 Int#2#1111_0000 inT#8#77777
DINT	DINT#16#3fff_ffff dint#-65000 DInt#2#1111_0000 dinT#8#17777777777
REAL	REAL#1 real#1.5 real#2e4 real#3.1
CHAR	CHAR#A CHAR#49

9.1.3 Notación de constantes

Dependiendo del tipo y del formato de datos, existe una notación (formato) determinada para el valor de una constante. El tipo y el valor de la constante resultan directamente de la notación, por lo que no es necesario declararlos.

Ejemplos:

15	VALOR 15	como constante entera en representación decimal
2#1111	VALOR 15	como constante entera en representación binaria
16#F	VALOR 15	como constante entera en representación hexadecimal

Resumen de las notaciones posibles

Tipo de datos	Descripción	Ejemplos en S7-SCL	Ejemplos en AWL, si difieren
BOOL	1 bit	FALSE TRUE BOOL#0 BOOL#1 BOOL#FALSE BOOL#TRUE	
BYTE	Número hexadecimal de 8 bits	B#16#00 B#16#FF BYTE#0 B#2#101 Byte#'ä' b#16#f	
CHAR	8 bits (1 carácter ASCII)	'A' CHAR#49	
STRING	254 caracteres ASCII como máximo	'dirección'	
WORD	Número hexadecimal de 16 bits Número octal de 16 bits Número binario de 16 bits	W#16#0000 W#16#FFFF word#16#f WORD#8#177777 8#177777 W#2#1001_0100 WORD#32768	

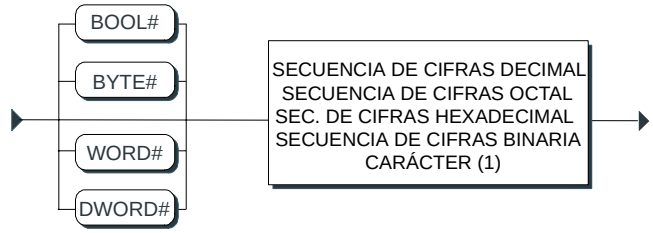
Tipo de datos	Descripción	Ejemplos en S7-SCL	Ejemplos en AWL, si difieren
DWORD	Número hexadecimal de 32 bits Número octal de 32 bits Número binario de 32 bits	DW#16#0000_0000 DW#16#FFFF_FFFF Dword#8#37777777777 8#37777777777 DW#2#1111_0000_1111_0000 dword#32768	
INT	16 bits Número en coma fija	-32768 +32767 INT#16#3f_ff int#-32768 Int#2#1111_0000 inT#8#77777	
DINT	32 bits Número en coma fija	-2147483648 +2147483647 DINT#16#3fff_ffff dint#-65000 Dint#2#1111_0000 dinT#8#17777777777	L#-2147483648 L#+2147483647
REAL	Número en coma flotante de 32 bits	Representación decimal 123.4567 REAL#1 real#1.5 Representación exponencial real#2e4 +1.234567E+02	
S5TIME	16 bits Valor de tiempo en formato SIMATIC	T#0ms TIME#2h46m30s T#0.0s TIME#24.855134d	S5T#0ms S5TIME#2h46m30s
TIME	32 bits Valor de tiempo en formato IEC	-T#24d20h31m23s647ms TIME#24d20h31m23s647ms T#0.0s TIME#24.855134d	
DATE	De16 bits Valor de la fecha	D#1990-01-01 DATE#2168-12-31	
TIME_OF_DAY	De 32 bits Hora	TOD#00:00:00 TIME_OF_DAY#23:59:59.999	
DATE_AND_TIME	Valor de fecha y hora	DT#1995-01-01-12:12:12.2	

9.1.3.1 Constantes de bits

Las constantes de bits contienen valores de una longitud de 1 bit, 8 bits, 16 bits o 32 bits. Estas constantes se pueden asignar a las variables del tipo de datos BOOL, BYTE, WORD o DWORD (dependiendo de su longitud) en el programa S7-SCL.

Sintaxis

CONSTANTE DE BITS



(1) sólo en el tipo de datos BYTE

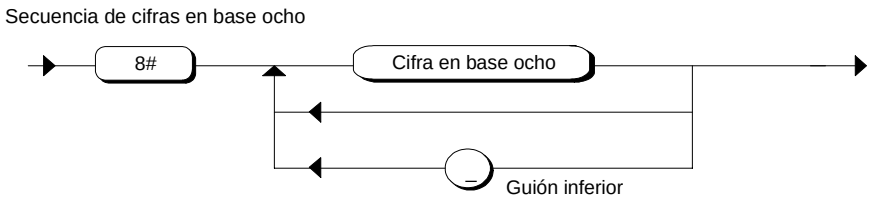
Secuencia de dígitos decimales

La secuencia de dígitos decimales en constantes se puede separar mediante caracteres de subrayado. Estos caracteres ayudan a mejorar la legibilidad cuando los números son grandes. Los siguientes ejemplos muestran notaciones admisibles para una secuencia de dígitos decimales en una constante:

```
DW#2#1111_0000_1111_0000
dword#32768
```

Valores binarios, octales y hexadecimales

La indicación de una constante entera en otro sistema numérico que no sea el decimal se realiza anteponiendo los prefijos **2#, 8# o 16#**, seguidos del número representado en el sistema que corresponda. Esto se explica en la siguiente figura, en la que aparece como ejemplo una cadena de dígitos de un número octal:



Ejemplo

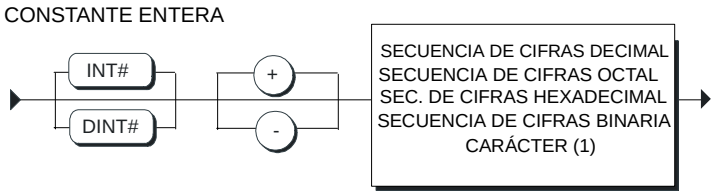
Los siguientes ejemplos constituyen notaciones posibles para constantes de bits:

```
Bool#false
8#177777
DW#16#0000_0000
```

9.1.3.2 Constante entera

Las constantes enteras contienen valores enteros de una longitud de 16 ó 32 bits. Estas constantes pueden asignar a las variables del tipo de datos INT o DINT (dependiendo de su longitud) en el programa S7-SCL.

Sintaxis



(1) sólo en el tipo de datos INT

Secuencia de cifras decimales

La secuencia de cifras decimales dentro de las constantes se puede separar por medio de guiones bajos. Los guiones bajos ayudan a mejorar la legibilidad cuando los números son grandes. Los siguientes ejemplos constituyen notaciones admisibles para una secuencia de cifras decimales dentro de una constante:

```
1000
1_120_200
666_999_400_311
```

Valores binarios, octales y hexadecimales

La indicación de una constante entera en otro sistema de numeración que no sea el decimal se produce mediante la anteposición de los prefijos **2#, 8# o 16#**, seguido del número en la representación del sistema correspondiente.

Ejemplo

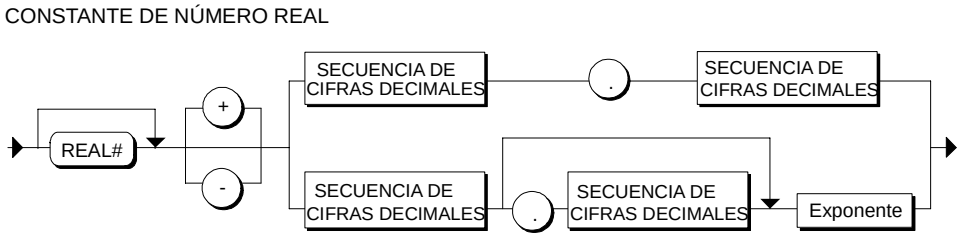
Los siguientes ejemplos constituyen notaciones posibles para constantes enteras:

```
Valor_2:=2#0101;            // Número binario, valor decimal 5
Valor_3:=8#17;              // Número octal, valor decimal 14
Valor_4:=16#F;              // Número hexadecimal, valor decimal 15
Valor_5:=INT#16#3f_ff       // Número hexadecimal,
                            // notación tipificada
```

9.1.3.3 Constante real

Las constantes de números reales son valores con dígitos a la derecha de la coma. Estas constantes se pueden asignar a variables del tipo de datos REAL.

Sintaxis



La indicación del signo es opcional. Si no se indica ningún signo, el número se considera positivo.

La secuencia de dígitos decimales en constantes se puede separar mediante caracteres de subrayado.Estos caracteres ayudan a mejorar la legibilidad cuando los números son grandes. Los siguientes ejemplos muestran notaciones admisibles para una secuencia de dígitos decimales dentro de una constante:

```
1000
1_120_200
666_999_400_311
```

Exponente

En la notación exponencial se puede utilizar un exponente para especificar números en coma flotante. El exponente se especifica anteponiendo la letra "E" o "e" al valor entero.

La magnitud $3x10^{10}$ se puede representar en S7-SCL mediante los siguientes números reales:

```
3.0E+10     3.0E10      3e+10       3E10
0.3E+11     0.3e11      30.0E+9     30e9
```

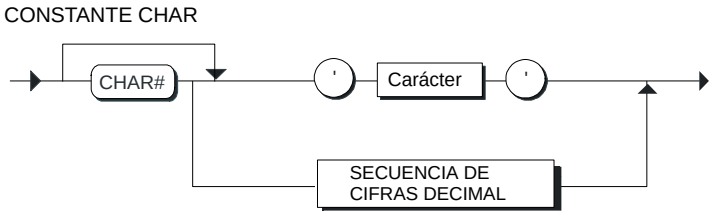
Ejemplos

```
NUMERO4:= -3.4 ;
NUMERO5:= 4e2 ;
NUMERO6:= real#1.5;
```

9.1.3.4 Constante CHAR (un solo carácter)

La constante CHAR contiene un solo carácter. El carácter está encerrado entre "comillas" ('). Las constantes CHAR no se pueden utilizar en expresiones.

Sintaxis



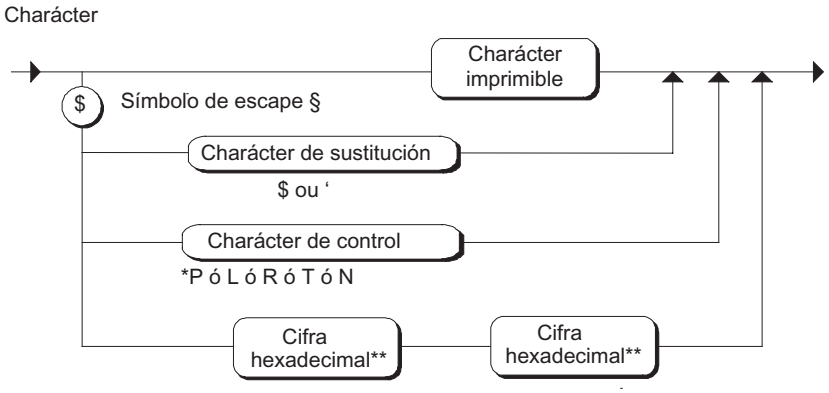
Ejemplo

```
Caracter_1 := 'B';
Caracter_2 := char#43;
Caracter_3 := char#'B';
```

Sintaxis del carácter

El carácter puede proceder del juego de caracteres ASCII ampliado y completo. Los caracteres especiales de formateo, las comillas (') o el carácter $ se pueden introducir con ayuda del símbolo $.

También es posible utilizar los caracteres no imprimibles del juego de caracteres ASCII ampliado y completo. Para ello es necesario introducirlos en código hexadecimal.



Representación de reserve en código hexadecimal

* P=Salto de página
L=Salto de línea
R=Retorno de carro
T=Tabulador
N=Línea nueva

** $00 no permitido

Ejemplo de especificación de un carácter en código hexadecimal:

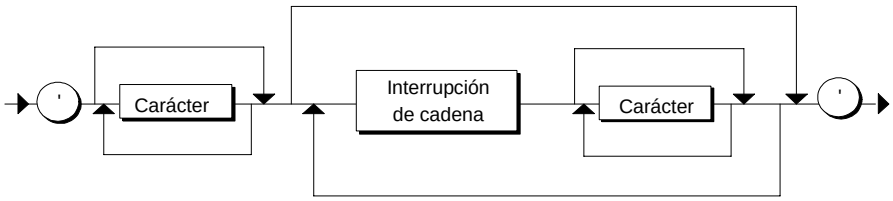
```
CARACTER   := '$41' ;  //corresponde al carácter 'A'
Espacio    :='$20';    //corresponde al carácter
␣
```

9.1.3.5 Constante STRING

Una constante STRING es una cadena de caracteres de 254 caracteres como máximo. Los caracteres se encierran entre comillas. Las constantes STRING no se pueden utilizar en expresiones.

Sintaxis

CONSTANTE STRING

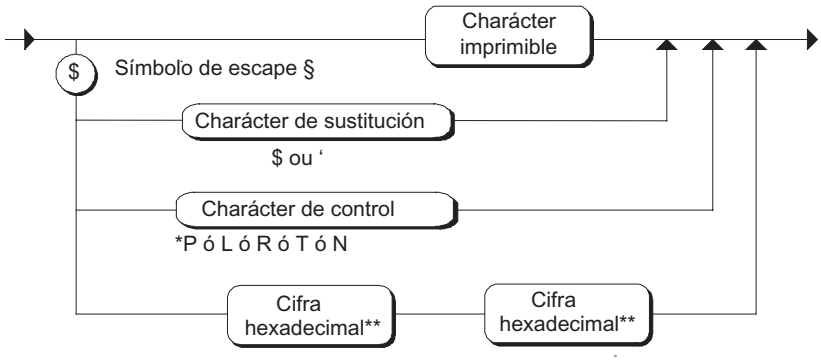


Sintaxis del carácter

El carácter puede proceder del juego de caracteres ASCII ampliado y completo. Los caracteres especiales de formateo, la comilla simple (') o el signo $ se pueden introducir con ayuda del símbolo $.

También es posible utilizar los caracteres no imprimibles del juego de caracteres ASCII ampliado y completo. Para ello es necesario introducirlos en código hexadecimal.

Charácter



Representación de reserve en código hexadecimal

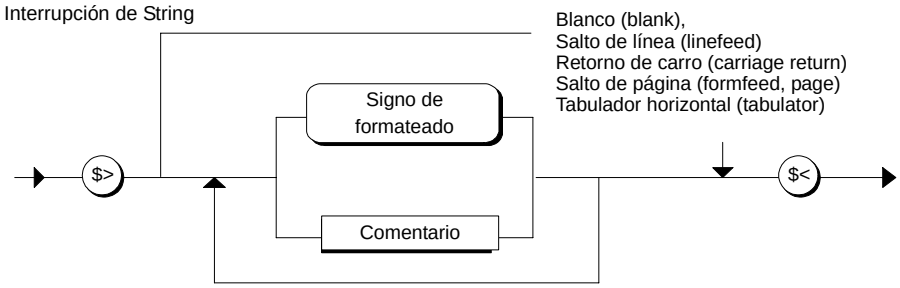
* P=Salto de página
L=Salto de línea
R=Retorno de carro
T=Tabulador
N=Línea nueva

** $00 no permitido

Interrupción de strings

Una constante STRING se puede interrrumpir y reanudar varias veces.

Un string puede hallarse en una línea de un bloque S7-SCL o estar repartido entre varias líneas medianteidentificaciones especiales. Para interrumpir una constante STRING se utiliza el identificador $>, para reanudarla en un renglón de continuación se utiliza el identificador $<. El espacio entre el indicador de interrupción y el de reanudación puede abarcar varias líneas e incluir solamente un comentario junto a los espacios en blanco.



Ejemplos

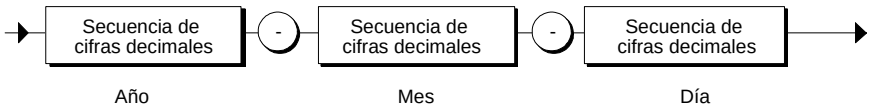
```
// Constante STRING:
NOMBRE:= 'SIEMENS';
//Interrupción de una constante STRING
MENSAJE1:= 'Control $>
$< del MOTOR';
// STRING en representación hexadecimal:
MENSAJE1:= '$41$4E' (*Secuencia de caracteres AN*);
```

9.1.3.6 Constante de fecha

La fecha se inicia con los prefijos DATE# ó D#. La fecha se indica con números enteros para el año (4 cifras), el mes y el día, que se separan por medio de guiones.

Sintaxis

Indicación de fecha



Ejemplo

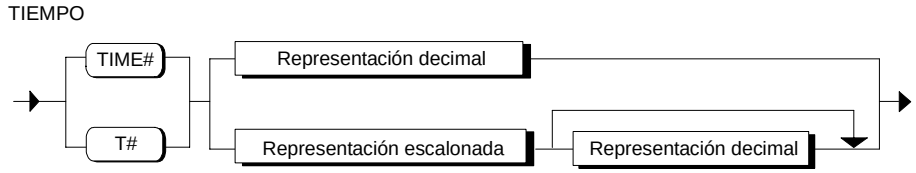
```
VARIABLE_TEMPORAL1:= DATE#1995-11-11 ;
VARIABLE_TEMPORAL2:= D#1995-05-05 ;
```

9.1.3.7 Constante de tiempo

El tiempo se inicia con el prefijo TIME# o T#. El tiempo se puede indicar de dos maneras distintas:

- Representación decimal
- Representación escalonada.

Sintaxis



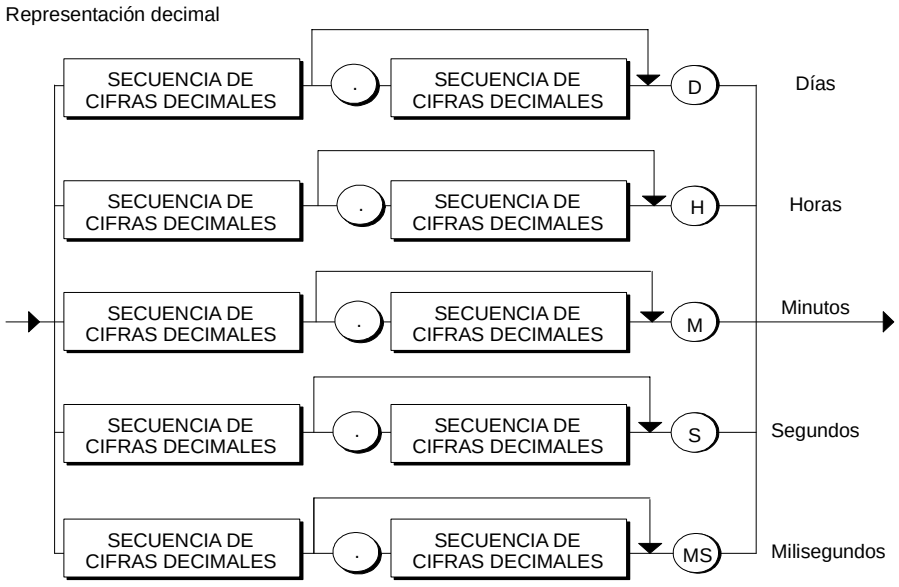
- Cada unidad de tiempo (p.ej.: horas, minutos) sólo puede indicarse una vez.
- Debe respetarse la secuencia: días, horas, minutos, segundos, milisegundos.

Para cambiar de la representación escalonada a la representación decimal es necesario hacerlo antes de especificar las unidades de tiempo.

Después de los prefijos T# o TIME# se debe especificar al menos una unidad de tiempo.

Representación decimal

La representación decimal se utiliza sólo para especificar como número decimal un componente de tiempo (p.ej., horas o minutos).

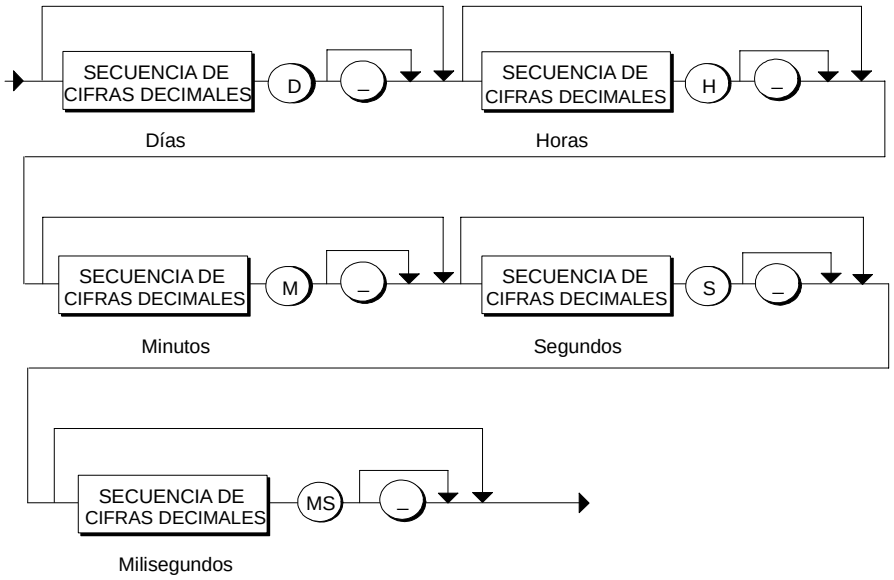


El acceso a la representación decimal sólo es posible con unidades de tiempo no definidas todavía.

Representación escalonada

La representación escalonada es una secuencia de distintos componentes de tiempo. Primero se especifican los días, luego las horas, y así sucesivamente, separados por caracteres de subrayado. Hay que indicar por lo menos un componente.

Representación escalonada



Ejemplo

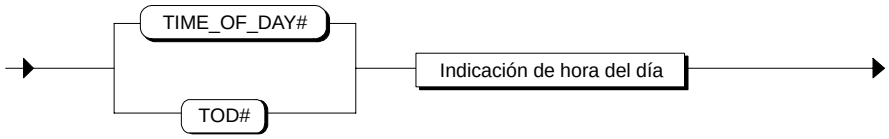
```
// Representación decimal
Intervalo1:= TIME#10.5S ;

// Representación escalonada
Intervalo2:= T#3D_2S_3MS ;

// Representación escalonada y representación decimal
Intervalo3 := T#2D_2.3s ;
```

9.1.3.8 Constantes de hora

La hora del día se inicia con los prefijos TIME_OF_DAY# ó TOD#.

Sintaxis

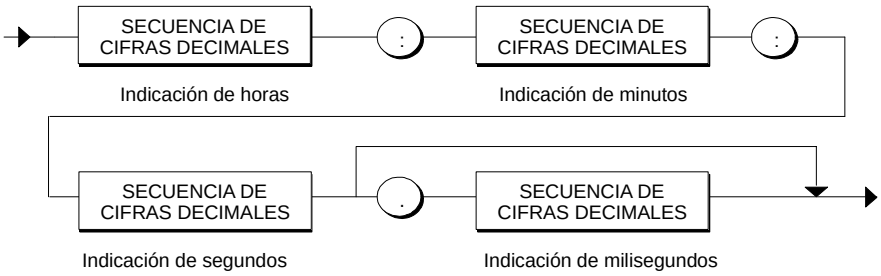
HORA DEL DIA



Para la indicación de la hora del día se declaran las horas, minutos y segundos separados por dos puntos. La indicación de los milisegundos es opcional. En caso de utilizarse, se separa de los demás datos mediante un punto.

Indicación de la hora del día

Indicación de hora del día



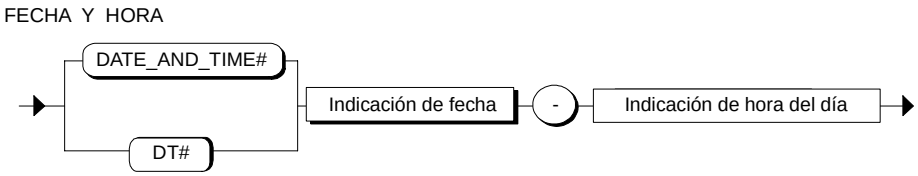
Ejemplo

```
HORA1:= TIME_OF_DAY#12:12:12.2 ;
HORA2:= TOD#11:11:11 ;
```

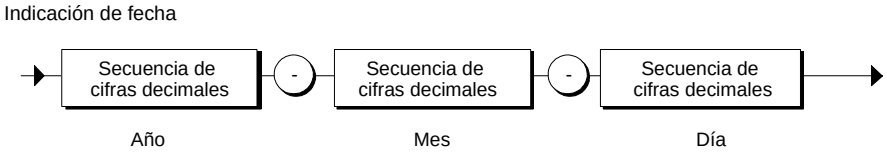
9.1.3.9 Constante de fecha y hora

La indicación de la fecha y hora se inicia con los prefijos DATE_AND_TIME# ó DT#. Se trata de una constante compuesta por la indicación de la fecha y la indicación de la hora.

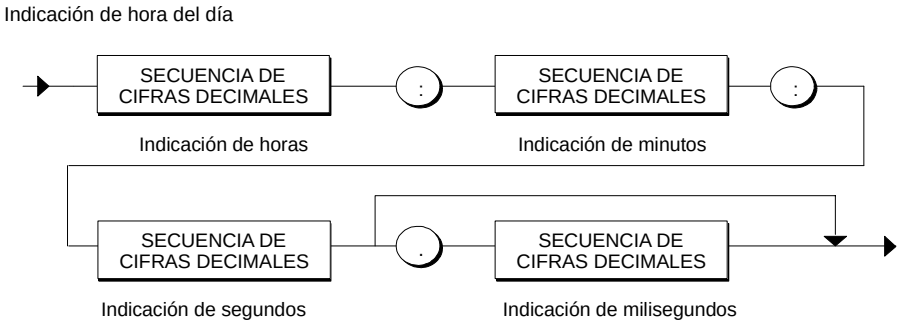
Sintaxis



Indicación de la fecha



Indicación de la hora



Ejemplo

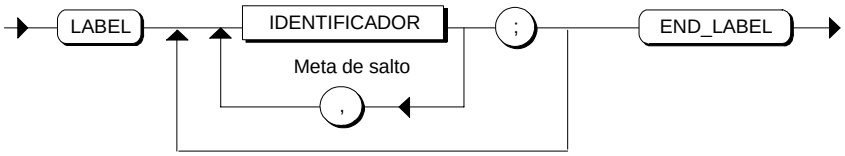
```
HORA1:= DATE_AND_TIME#1995-01-01-12:12:12.2 ;
HORA2:= DT#1995-02-02-11:11:11;
```

9.2 Declaración de metas de salto

Las metas de salto (labels) permiten definir la meta de una instrucción GOTO. Se declaran en el área de declaración del bloque lógico con su nombre simbólico.

Sintaxis

Bloque de metas de salto



Ejemplo

```
LABEL
    META1, META2, META3;
END_LABEL
```

10 Datos globales

10.1 Sinopsis de los datos globales

En S7-SCL tiene la posibilidad de acceder a datos globales. Existen dos tipos de datos globales:

- **Áreas de memoria de la CPU**

  Estas áreas de memoria son datos declarados por el sistema: p. ej., entradas, salidas y marcas. El número de áreas de memoria disponibles de la CPU está determinado por la CPU correspondiente.

- **Datos globales de usuario como bloques de datos cargables:**

  Estas áreas de datos se encuentran en los bloques de datos. Para poder utilizarlas, es necesario crear previamente los bloques de datos y declarar en ellos los datos. En el caso de los bloques de datos de instancia, se derivan de los bloques de función y se crean automáticamente.

Acceso a datos globales

Existen varias formas de acceder a los datos globales:

- **acceso absoluto:** mediante el identificador del operando y la dirección absoluta.
- **acceso simbólico:** mediante el símbolo que se haya definido en la tabla de símbolos.
- **acceso indizado:** mediante el identificador de operando y el índice de array.
- **acceso estructurado:** mediante una variable.

Tipo de acceso	Áreas de memoria de la CPU	Datos de usuario globales
absoluto	Sí	Sí
simbólico	Sí	Sí
indizado	Sí	Sí
estructurado	No	Sí

10.2 Áreas de memoria de la CPU

10.2.1 Sinopsis de las áreas de memoria de la CPU

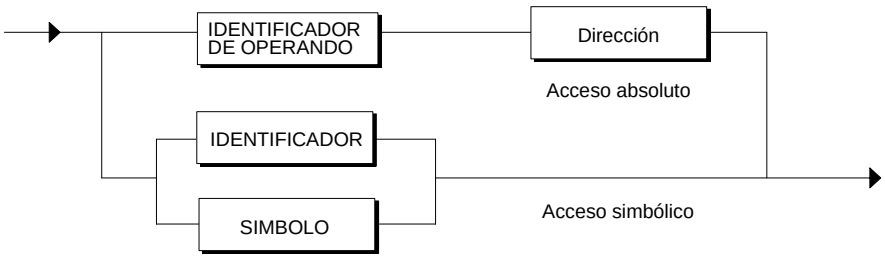
Las áreas de memoria de una CPU son áreas declaradas por el sistema, por lo que no es necesario declararlas en el bloque lógico. Cada CPU ofrece las siguientes áreas de memoria con un área de direccionamiento propia:

- Entradas/salidas en la imagen del proceso (p.ej. A1.0)
- Entradas/salidas de periferia (p.ej., PA1.0)
- Marcas (p.ej., M1.0)
- Temporizadores, contadores (C1)

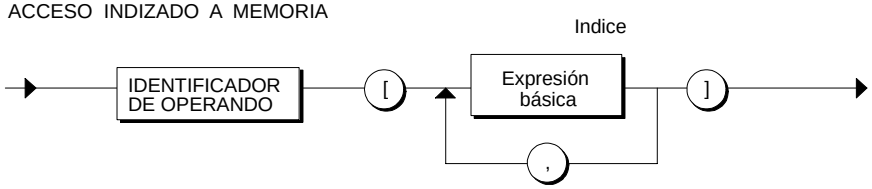
Sintaxis para el acceso

- El acceso a un área de memoria de la CPU se realiza mediante una asignación de valor en el área de instrucciones de un bloque lógico: mediante un acceso simple, que podrá indicar como absoluto o como simbólico, o
- mediante un acceso indizado.

ACCESO SIMPLE A MEMORIA



ACCESO INDIZADO A MEMORIA



10.2.2 Acceso absoluto a áreas de memoria de la CPU

El acceso absoluto a un área de memoria de la CPU se realiza, por ejemplo, mediante asignación de un identificador absoluto a una variable del mismo tipo.

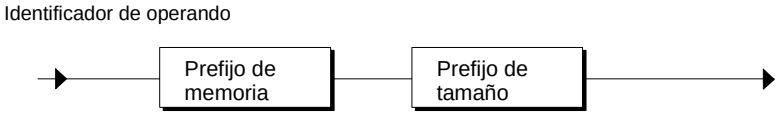
```
ESTADO_2:= EB10;
```

Acceso absoluto

Variable del mismo tipo

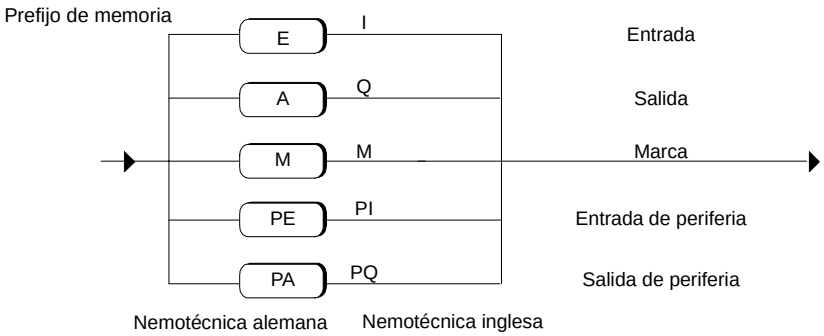
El identificador absoluto remite a un área de memoria de la CPU. Esta área se especifica indicando el identificador de operando (en este caso EB) seguido de la dirección (en este caso `10)`.

Sintaxis del identificador absoluto



Prefijo de memoria

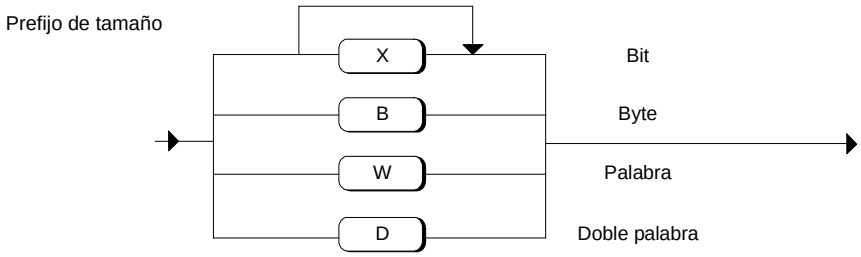
Con el prefijo de memoria se puede definir el tipo de área de memoria que deben direccionarse.



Dependiendo de la nemotécnica que se haya ajustado (inglesa o alemana), los identificadores de operandos varían.

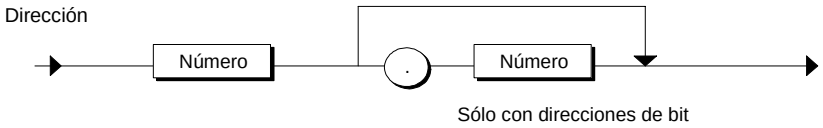
Prefijo de tamaño

Indicando el prefijo de tamaño se especifica la longitud del área de memoria que debe leerse desde la periferia. Por ejemplo, se puede leer un byte o una palabra. La especificación del prefijo de tamaño es opcional si sólo desea especificar un bit.



Dirección

Al especificar la dirección se indica primero la dirección absoluta del byte y, a continuación, separada por un punto, la dirección de bit del byte en cuestión. Especificar la dirección del bit es opcional.



Ejemplos

```
STATUSBYTE :=EB10;
STATUS_3   :=E1.1;
VALOR_MEDIDO     :=EW20;
```

10.2.3 Acceso simbólico a áreas de memoria de la CPU

El direccionamiento simbólico permite utilizar nombres simbólicos para direccionar las áreas de memoria de la CPU, en lugar de emplear un identificador absoluto.

Asigne un nombre simbólico a cada uno de los operandos de su programa en la tabla de símbolos. La tabla de símbolos puede abrirse en cualquier momento desde S7-SCL con el comando de menú **Herramientas > Tabla de símbolos** para añadir más símbolos.

Pueden especificarse todos los tipos de datos simples siempre que admitan el ancho del acceso. La siguiente tabla representa una tabla de símbolos a modo de ejemplo

Símbolo	**Dirección absoluta**	**Tipo de datos**	**Comentario**
Contacto_motor_1	E 1.7	BOOL	Contacto 1 para motor A
Entrada1	EW10	INT	Palabra de estado

Acceso

El acceso se realiza asignando un valor a una variable del mismo tipo usando los símbolos declarados.

Ejemplo

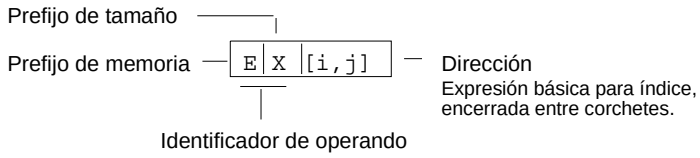
```
VALOR_MEDIDO_1 := Contacto del motor_1;
Estado_Motor1 := Entrada1 ;
```

10.2.4 Acceso indizado a áreas de memoria de la CPU

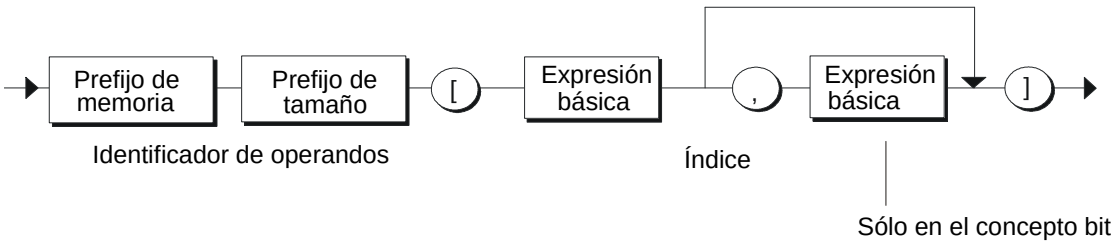
También existe la posibilidad de acceder a áreas de memoria de la CPU mediante un índice. Frente al direccionamiento absoluto ofrece la ventaja de poder direccionar dinámicamente los accesos mediante índices variables. Así por ejemplo, es posible utilizar como dirección la variable de un bucle FOR.

El acceso indizado a un área de memoria se realiza de forma similar al acceso absoluto. Sólo se distingue en la forma de especificar la dirección. En lugar de la dirección se especifica un índice que puede ser una constante, una variable o una expresión aritmética.

En el acceso indizado, el identificador absoluto se compone del identificador del operando (prefijo de memoria, del prefijo de tamaño) y de una expresión base para la indización.



Sintaxis del identificador absoluto



La indización (expresión base) debe ajustarse a la regla siguiente:

- Cada índice debe ser una expresión aritmética del tipo de datos INT
- En un acceso del tipo de datos BYTE, WORD o DWORD, se debe utilizar exactamente un índice. El índice se interpreta como la dirección del byte. El ancho de acceso viene dado por el prefijo de tamaño.
- En un acceso del tipo de datos BOOL se deben utilizar dos índices. El primer índice especifica la dirección del byte, y el segundo la posición del bit dentro del byte.

Ejemplo

```
VALOR_MEDIDO_1    :=EW[CONTADOR];
MARCA_DS          :=E[BYTE, BIT];
```

10.3 Bloques datos

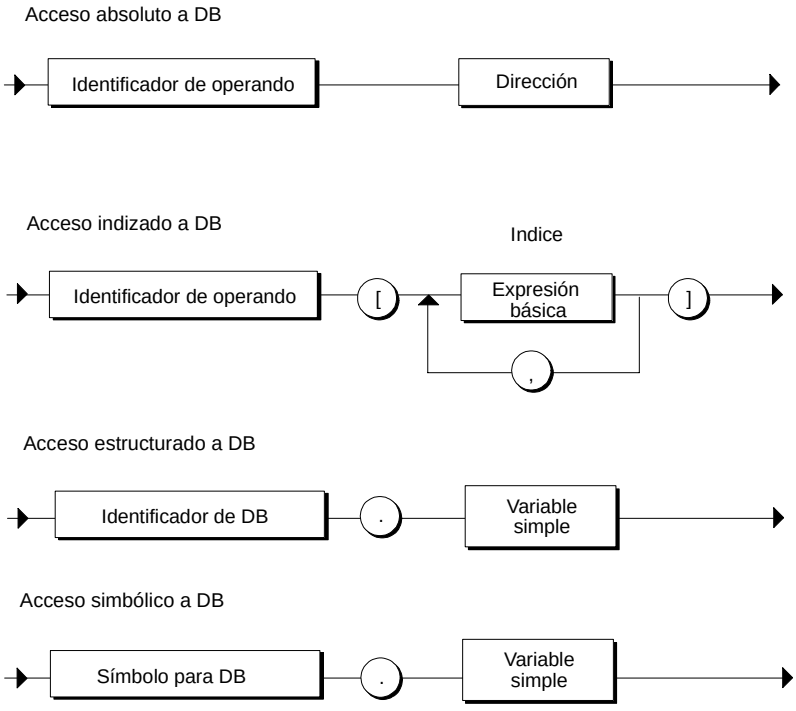
10.3.1 Sinopsis de los bloques de datos

En los bloques de datos se pueden guardar y procesar todos los datos que valgan para todo el programa o proyecto. Todos los bloques lógicos tienen acceso de lectura y escritura a los datos globales de usuario.

Acceso

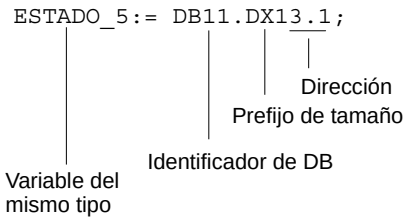
El acceso a los datos de un bloque de datos global puede ser:

- absoluto o simple
- configurado
- indizado



10.3.2 Acceso absoluto a bloques de datos

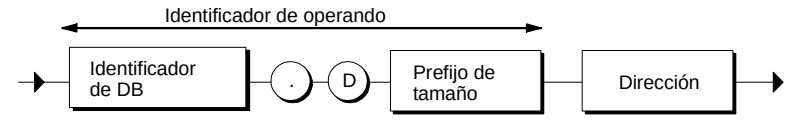
El acceso absoluto a un bloque de datos se efectúa, al igual que en el caso de un acceso absoluto a las áreas de la CPU, asignando un valor a una variable del mismo tipo. El identificador DB va seguido de la palabra clave "D", el prefijo de tamaño (p. ej., X para bit) y la dirección de byte (p. ej., 13.1).



Sintaxis

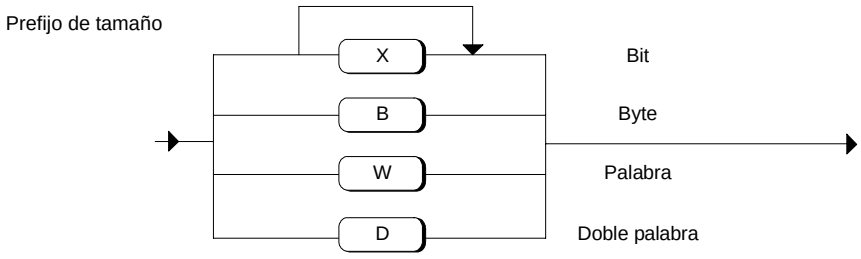
El acceso se especifica mediante el identificador DB, el prefijo de tamañp y dirección

Acceso absoluto a DB



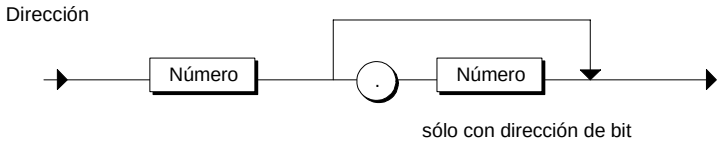
Prefijo de tamaño

El prefijo de tamaño indica la longitud del área de memoria del bloque de datos a que puede llamarse. Por ejemplo, puede leer un byte o una palabra del DB. La especificación del prefijo de tamaño es opcional si sólo desea especificar un bit.



Dirección

Al especificar la dirección se indica primero la dirección absoluta del byte y, a continuación, separada por un punto, la dirección del bit correspondiente (sólo en caso de acceso por bits).



Ejemplo

Veamos algunos ejemplos del acceso absoluto a un bloque de datos. El mismo bloque de datos se especifica de modo absoluto en la primera parte, y simbólicamente en la segunda:

```
STATUSBYTE :=DB101.DB10;
STATUS_3   :=DB30.D1.1;
VALOR_MEDIDO:=DB25.DW20;

STATUSBYTE :=Datos de estado.DB10;
STATUS_3   :="Nuevos datos".D1.1;
VALOR_MEDIDO:=Datos medidos.DW20.DW20;

STATUS_1   :=WORD_TO_BLOCK_DB (INDEX).DW10;
```

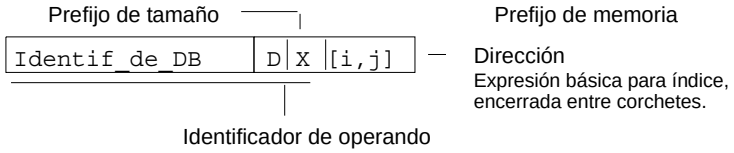
10.3.3 Acceso indizado a bloques de datos

También existe la posibilidad de acceder a bloques de datos mediante un índice. Frente al direccionamiento absoluto esto ofrece la ventaja de poder direccionar operandos cuya dirección se determine durante el tiempo de ejecución. Por ejemplo, podrá utilizar como dirección la variable en curso de un blucle FOR.

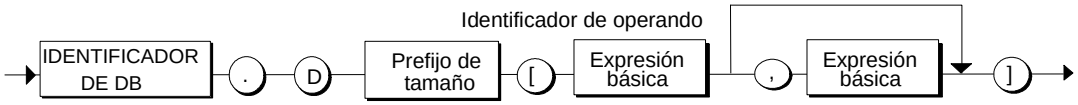
El acceso indizado a un bloque de datos se realiza de forma similar al acceso absoluto. Sólo se distigue en la forma de especificar la dirección.

En lugar de la dirección se especifica un índice que puede ser una constante, una variable o una expresión aritmética.

El acceso indizado se compone del nombre del DB, el identificador del operando (palabra clave "D" y prefijo de tamaño) y una expresión base para el indizado.



Sintaxis



La indización debe cumplir las reglas siguientes:

- En un acceso con un tipo de datos BYTE, WORD o DWORD se debe utilizar exactamente un índice. El índice se interpreta como la dirección del byte. El ancho de acceso viene definido por el prefijo de tamaño.
- En un acceso con un tipo de datos BOOL se debe utilizar dos índices. El primer índice especifica la dirección del byte, y el segundo la posición del bit dentro del byte.
- Cada índice debe ser una expresión aritmética del tipo de datos INT (0 - 32767)

Ejemplo

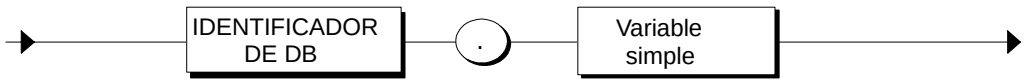
```
STATUS_1:= DB11.DW[CONTADOR];
STATUS_2:= DB12.DX[W, BIT];
STATUS_1:= Base_de_datos.DW[CONTADOR];
STATUS_2:= Base_de_datos2.DX[W, BIT];
STATUS_1:= WORD_TO_BLOCK_DB(INDEX).DW[CONTADOR];
```

10.3.4 Acceso estructurado a bloques de datos

El acceso estructurado se realiza mediante el identificador de las variables declaradas en el bloque de datos. La variable se puede asignar a cualquier variable del mismo tipo.

La variable del bloque de datos se direcciona especificando el nombre del DB y el nombre de la variable simple, separados por un punto.

Sintaxis



En este caso la variable simple es una variable a la que se ha asignado un tipo de datos simple o compuesto al declarar el bloque.

Si se utiliza un parámetro del tipo BLOCK_DB o el resultado de la función de conversión WORD_TO_BLOCk_DB para iniciar un acceso a un bloque de datos, no es posible realizar un acceso estructurado, sino sólo un acceso absoluto o un acceso indizado.

Ejemplo

```
En el área de declaración del FB10:
VAR
Resultado:    STRUCT RESULT1 : INT;
RESULT2 : WORD;
END_STRUCT
END_VAR

Tipo de datos definido por el usuario UDT1
TYPE UDT1   STRUCT RESULT1 : INT;
RESULT2 : WORD;
END_STRUCT

DB20 con tipo de datos de usuario:
DB20
UDT1
BEGIN ...

DB30 sin tipo de datos de usuario:
DB30       STRUCT RESULT1 : INT;
RESULT2 : WORD;
END_STRUCT
BEGIN ...

Bloque de función con los accesos:
..
FB10.DB10();
PALABRA_RESULT_A   :=   DB10.Resultado.RESULT2;
PALABRA_RESULT_B   :=   DB20.RESULT2;
PALABRA_RESULT_C   :=   DB30.RESULT2;
```

11 Expresiones, operaciones y operandos

11.1 Sinopsis de las expresiones, operaciones y operandos

Una expresión representa un valor calculado durante la compilación o el tiempo de ejecución del programa y se compone de operandos (p.ej. constantes, variables o llamadas de función) y operaciones (p.ej. *, /, + o -).

Los tipos de datos de los operandos y las operaciones aplicadas determinan el tipo de expresión. S7-SCL distingue los siguientes:

- expresiones aritméticas
- expresiones de comparación
- expresiones lógicas

La expresión se evalúa en una secuencia concreta determinada por:

- la prioridad de las operaciones realizadas y
- la secuencia izquierda-derecha o
- el uso de los paréntesis en operaciones de la misma prioridad.

El resultado de una expresión puede:

- asignarlo a una variable,
- utilizarlo como condición para una instrucción de control,
- utilizarlo como parámetro para llamar una función o un bloque de función

11.2 Operaciones

Las expresiones están formadas por operaciones y operandos. La mayoría de las operaciones de S7-SCL combinan dos operandos, por lo que se denominan *binarias*. Otras operaciones se realizan con un solo operando, por lo que se denominan *unarias*.

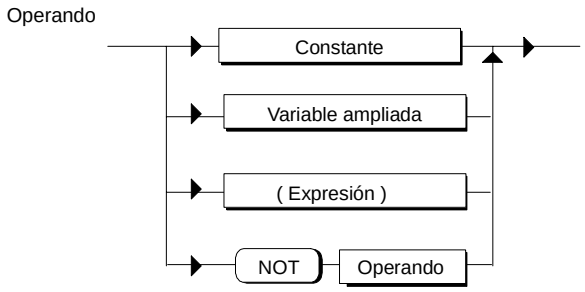
Las operaciones binarias se escriben entre los operandos (p. ej. A + B). Una operación unaria está situada siempre delante de su operando (p. ej. -B).

La prioridad de operaciones que aparece en la siguiente tabla determina la secuencia del cálculo. En la tabla la cifra "1" corresponde a la máxima prioridad.

Clase	Operación	Representación	Prioridad
Operación de asignación:	Asignación	: =	11
Operaciones aritméticas:	Potencia	**	2
	Operaciones unarias		
	signo más unitario	+	3
	Menos unario	-	3
	Operaciones aritméticas básicas		
	Multiplicación	*	4
	División	/	4
	Función módulo	MOD	4
	División por núm. enteros	DIV	4
	Adición	+	5
	Sustracción	-	5
Operaciones de comparación:	Menor que	<	6
	Mayor que	>	6
	Menor o igual que	<=	6
	Mayor o igual que	>=	6
	Igualdad	=	7
	Desigualdad	<>	7
Operaciones lógicas:	Negación	NOT	3
	Operaciones lógicas básicas		
	Y	AND o &	8
	O exclusivo	XOR	9
	O	OR	10
Caracteres de anidado:	Paréntesis	( )	1

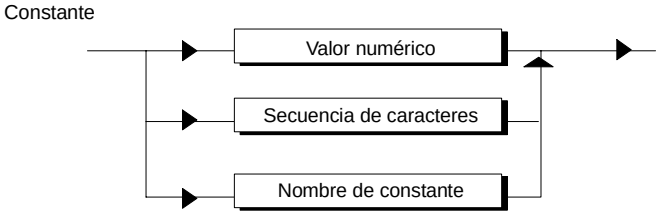
11.3 Operandos

Los operandos son objetos con los que se puede formar una expresión. Para crear operandos se pueden utilizar los siguientes elementos:



Constantes:

Se pueden utilizar constantes con su valor numérico o con un nombre simbólico o cadenas de caracteres.

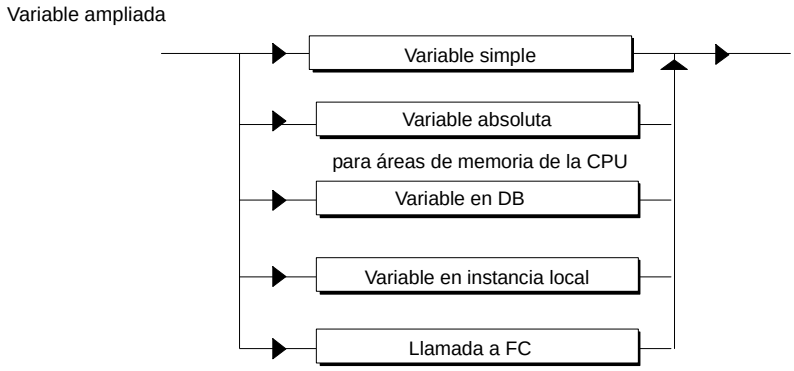


Los siguientes ejemplos son ejemplos de constantes válidas:

```
4_711
4711
30.0
'CARACTER'
FACTOR
```

Variable ampliada

La variable ampliada es un término genérico para una serie de variables. Para más información al respecto, consulte el capítulo "Asignación de valores".



Los siguientes ejemplos son ejemplos de variables válidas:

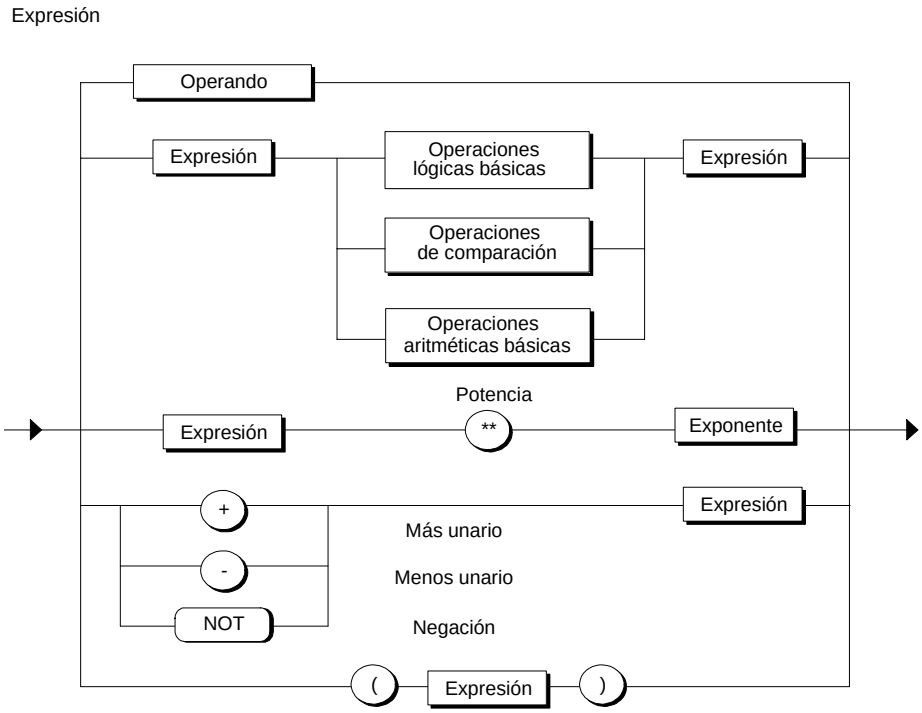
```
VALOR _TEORICO   variable simple
EW10 variable absoluta
E100.5     variable absoluta
DB100.DW [INDICE]      variable del DB
MOTOR.VELOCIDAD  variable de instancia local
SQR (20)   función estándar
FC192 (VALOR_TEORICO)  Llamada a función
```

Nota

En una llamada a función el resultado obtenido (valor de respuesta) se inserta en la expresión en la posición del nombre de la función. Por este motivo, no se permite el uso de funciones VOID que no tengan ningún valor de retorno como operandos de una expresión.

11.4 Sintaxis de una expresión

Sintaxis



Resultado de una expresión

El resultado de una expresión se puede:

- asignar a una variable,
- utilizar como condición para una instrucción de control,
- utilizar como parámetro para llamar a una función o a un bloque de función.

Secuencia de evaluación

La secuencia de evaluación de una expresión depende de:

- la prioridad de las operaciones realizadas
- la secuencia izquierda-derecha
- el uso de los paréntesis (en operaciones de la misma prioridad).

Reglas

Las expresiones se procesan según las siguientes reglas:

- Un operando situado entre dos operaciones de distinta prioridad está siempre ligado a la de mayor rango.
- Las operaciones se ejecutan de acuerdo con el orden jerárquico.
- Las operaciones de la misma prioridad se ejecutan de izquierda a derecha.
- Anteponer un signo menos a un identificador equivale a multiplicar por -1.
- Las operaciones aritméticas no pueden aparecer de forma consecutiva. Por lo tanto, la expresión a * - b no es válida, pero sí está permitida la expresión a*(-b).
- El uso de paréntesis puede alterar la prioridad de la operación, ya que los paréntesis tienen prioridad máxima.
- Las expresiones entre paréntesis se consideran como un único operando y se evalúan siempre en primer lugar.
- El número de paréntesis de apertura debe coincidir con el número de paréntesis de cierre.
- No se pueden utilizar operaciones aritméticas en combinación con caracteres o datos lógicos. Por lo tanto, expresiones como 'A' + 'B' y (n<=0) + (m>0) no son válidas.

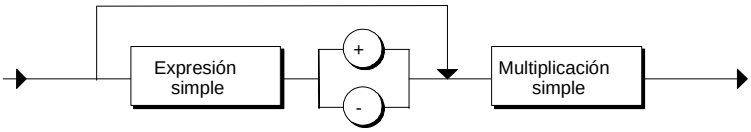
Ejemplos de expresiones

```
EB10              //operando
A1 AND (A2)       //expresión lógica
(A3) < (A4)       //expresión de comparación
3+3*4/2           //expresión aritmética
```

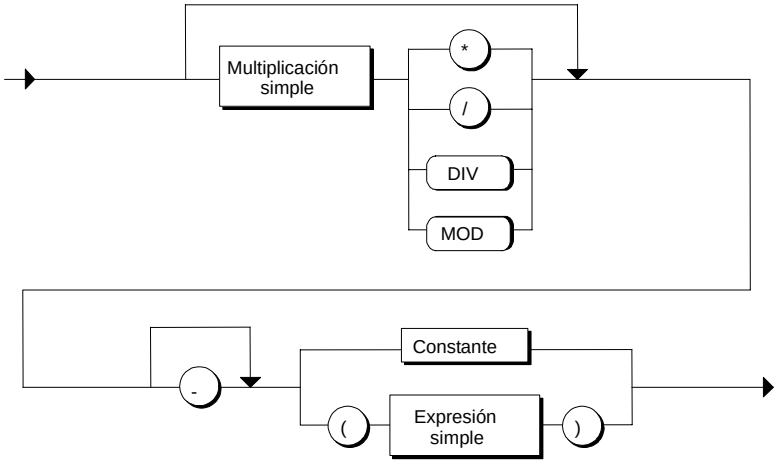
11.5 Expresión simple

En S7-SCL se entiende por expresión simple las expresiones aritméticas simples. Puede multiplicar o dividir por parejas valores contantes y combinar dichas parejas mediante adición o sustracción.

Sintaxis de la expresión simple:



Sintaxis de la multiplicación simple:



Ejemplo:

```
EXPRESION_SIMPLE= A * B + D / C - 3 * VALOR1;
```

11.6 Expresiones aritméticas

Una expresión aritmética es una expresión formada por operaciones aritméticas. Estas expresiones le permiten procesar datos del tipo numérico.

La siguiente tabla muestra las operaciones posibles e indica a qué tipo hay que asignar el resultado en función de los operandos. Se han utilizado las siguientes abreviaturas:

ANY_INT	para los tipos de datos	INT, DINT
ANY_NUM	para los tipos de datos	ANY_INT y REAL

Operación	**Identificador**	**1er operando**	**2º operando**	**Resultado**	**Prioridad**
Potencia	**	ANY_NUM	ANY_NUM	REAL	2
Más unario	+	ANY_NUM	-	ANY_NUM	3
		TIME	-	TIME	3
Menos unario	-	ANY_NUM	-	ANY_NUM	3
		TIME	-	TIME	3
Multiplicación	*	ANY_NUM	ANY_NUM	ANY_NUM	4
		TIME	ANY_INT	TIME	4
División	/	ANY_NUM	ANY_NUM	ANY_NUM	4
		TIME	ANY_INT	TIME	4
División por números enteros	DIV	ANY_INT	ANY_INT	ANY_INT	4
		TIME	ANY_INT	TIME	4
División de módulo	MOD	ANY_INT	ANY_INT	ANY_INT	4
Adición	+	ANY_NUM	ANY_NUM	ANY_NUM	5
		TIME	TIME	TIME	5
		TOD	TIME	TOD	5
		DT	TIME	DT	5
Sustracción	-	ANY_NUM	ANY_NUM	ANY_NUM	5
		TIME	TIME	TIME	5
		TOD	TIME	TOD	5
		DATE	DATE	TIME	5
		TOD	TOD	TIME	5
		DT	TIME	DT	5
		DT	DT	TIME	5

Nota

El tipo de datos del resultado de una división (/) viene dado por el tipo de datos del operando más significativo.
Si por ejemplo se dividen dos operandos del tipo INT, el resultado también será del tipo INT (es decir, 10/3=3, mientras 10.0/3=3.33).

Reglas

Las operaciones de las expresiones aritméticas se ejecutan según su prioridad.

- Para facilitar la lectura de los paréntesis se recomienda poner los signos de los números negativos incluso donde no sea necesario.
- En las divisiones con dos operandos enteros del tipo INT, las operaciones "DIV" y "/" dan el mismo resultado (vea el siguiente ejemplo).
- Las operaciones de división "/", "MOD" y "DIV" requieren que el segundo operando sea distinto a cero.
- Cuando un operando es del tipo INT (entero) y otro del tipo REAL (número de coma flotante), el resultado siempre será del tipo REAL.
- En la resta de datos del tipo DATE_AND_TIME y TIME, el operando del tipo TIME siempre debe figurar a la derecha del operador "-".

Ejemplos

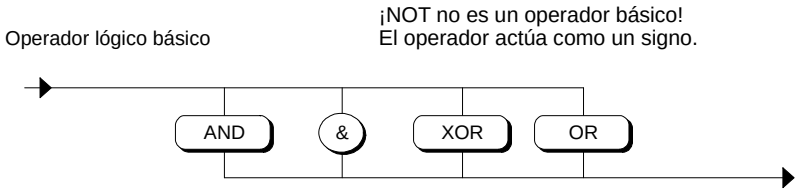
```
// El resultado (11) de la expresión aritmética se asigna
// a la variable "VALOR"
VALOR1 := 3 + 3 * 4 / 2 - (7+3) / (-5) ;
// El VALOR de la siguiente expresión es 1
VALOR2 := 9 MOD 2 ;
```

11.7 Expresiones lógicas

Una expresión lógica es una expresión formada por operaciones lógicas.

Operaciones lógicas básicas

Con las operaciones AND, &, XOR y OR se pueden combinar operandos lógicos (tipo BOOL) o variables del tipo de datos BYTE, WORD o DWORD para crear expresiones lógicas. Para invertir un operando lógico se utiliza la operación NOT.



Operaciones lógicas

El resultado de la expresión puede ser TRUE o FALSE cuando se combinan operandos booleanos, o un patrón de bits en caso de que se hayan combinado los dos operandos bit a bit.

La tabla siguiente indica las expresiones lógicas y los tipos de datos disponibles para el resultado y los operandos. Se han utilizado las siguientes abreviaturas:

ANY_BIT	para los tipos de datos	BOOL, BYTE, WORD, DWORD

Operación	Identificador	1er operando	2º operando	Resultado	Prioridad
Negación	NOT	ANY_BIT	-	ANY_BIT	3
Conjunción	AND	ANY_BIT	ANY_BIT	ANY_BIT	8
Disyunción exclusiva	XOR	ANY_BIT	ANY_BIT	ANY_BIT	9
Disyunción	OR	ANY_BIT	ANY_BIT	ANY_BIT	10

Rsultado

El resultado de una expresión lógica es

- 1 (true) o = 0 (false) al combinar lógicamente operaciones booleanas, o
- un patrón de bits, después de combinar bit a bit los dos operandos.

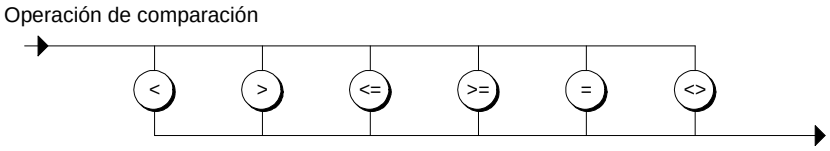
Ejemplos

```
// Se niega el resultado de la expresión de comparación.
           IF NOT (CONTADOR > 5) THEN . . . ;
// Se niega el resultado de la primera expresión de comparación y
// se combina con el resultado de la segunda
           A := NOT (CONTADOR1 = 4) AND (CONTADOR2 = 10) ;
// Disyunción de dos expresiones de comparación
           WHILE  (A >= 9) OR (CONSULTAR <> "n") DO.... ;
// Enmascaramiento de un byte de entrada (combinación binaria)
           Resultado := EB10 AND 2#11110000 ;
```

11.8 Expresiones de comparación

Con las operaciones de comparación se comparan los valores de dos operandos y se obtiene un valor booleano. El resultado es TRUE si la comparación se cumple, y FALSE si no se cumple.

Sintaxis



Reglas

La tabla siguiente muestra los tipos de datos comparables y las reglas que se aplican a la comparación:

Tipo de datos	**=**	**<>**	**>0**	**<0**	**>**	**<**	**Reglas**
INT	✓	✓	✓	✓	✓	✓	En la clase de los tipos de datos numéricos se admiten todos los operadores de comparación. El operador con el tipo más potente determina el tipo de la operación.
DINT	✓	✓	✓	✓	✓	✓	
REAL	✓	✓	✓	✓	✓	✓	
BOOL	✓	✓					En la clase de los tipos de datos bit sólo se admiten como operadores de comparación IGUAL y DIFERENTE. El operador con el tipo más potente determina el tipo de la operación.
BYTE	✓	✓					
WORD	✓	✓					
DWORD	✓	✓					

Tipo de datos	**=**	**<>**	**>0**	**<0**	**>**	**<**	**Reglas**
CHAR	✓	✓	✓	✓	✓	✓	En el caso de los caracteres y las cadenas de caracteres se utiliza la longitud de la variables y el valor numérico de cada carácter ASCII para la comparación. La comparación en el caso de STRING se realiza internamente mediante las funciones EQ_STRNG, GE_STRNG, LE_STRNG, GT_STRNG y LT_STRNG de la librería IEC. Los siguientes operandos no están permitidos para estas funciones: • Parámetro de una FC. • Parámetro IN_OUT de un FB del tipo STRUCT o ARRAY.
STRING	✓	✓	✓	✓	✓	✓	
DATE	✓	✓	✓	✓	✓	✓	
TIME	✓	✓	✓	✓	✓	✓	
DT	✓	✓	✓	✓	✓	✓	La comparación en el caso de DT se realiza internamente mediante las funciones EQ_DT, GE_ DT, LE_ DT, GT_STRNG y LT_ DT de la librería IEC. Los siguientes operandos no están permitidos para estas funciones: • Parámetro de una FC. • Parámetro IN_OUT de un FB del tipo STRUCT o ARRAY.
TOD	✓	✓	✓	✓	✓	✓	
S5-TIME							Las variables S5-TIME no se admiten como operandos de comparación. Es necesario realizar una conversión explícita de S5TIME a TIME con las funciones IEC.

Combinación de comparaciones

- Las expresiones de comparación se pueden combinar según las leyes de la lógica booleana. De este modo se pueden realizar consultas como "si a < b y b < c entonces ...".
  (Ejemplo: Valor_A > 20 AND Valor_B < 20)
  Las operaciones individuales se ejecutan según su prioridad. La prioridad se puede cambiar mediante el uso de paréntesis.

Ejemplos

```
// Comparación 3 MENOR IGUAL 4. El resultado
// es "TRUE"
            A := 3 <= 4
// Comparación 7 DIFERENTE 7. El resultado
// es "FALSE"
            7 <> 7
// Evaluación de una expresión de comparación en
// una ramificación IF
            IF CONTADOR < 5 THEN ....
// Combinación de dos expresiones de comparación
Valor_A > 20 AND Valor_B < 20
// Combinación de dos expresiones de comparación con paréntesis
A<>(B AND C)
```

12 Instrucciones

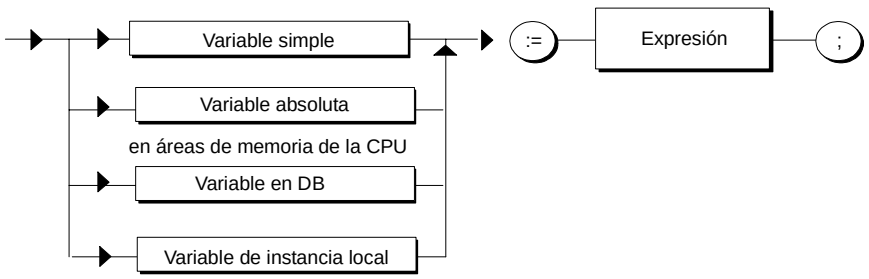
12.1 Asignaciones de valor

Las asignaciones de valor sustituyen el valor actual de una variable por otro valor que se indica mediante una expresión. Dicha expresión también puede incluir identificadores de funciones que se activan y devuelven valores (valor de respuesta).

Según se muestra en la figura siguiente, la expresión se evalúa a la derecha de la asignación, y el resultado se memoriza junto a las variables cuyo nombre figura a la izquierda del identificador de la asignación. En la figura aparecen las variables permitidas.

Sintaxis de la asignación de valores:

Asignación de valor



El tipo de una expresión de asignación es el mismo que el del operando izquierdo. Una variable simple puede ser una variable del tipo de datos simple o compuesto.

12.1.1 Asignación con variables de un tipo de datos simple

Toda expresión y toda variable de un tipo de datos simple puede ser asignada a otra variable del mismo tipo.

```
Identificador := expresión ;
Identificador := variable ;
```

Ejemplo

```
FUNCTION_BLOCK FB12
VAR
    PULSADOR_1   : INT ;
    PULSADOR_2   : INT ;
    VALOR_TEORICO_1    : REAL ;
    VALOR_TEORICO_2    : REAL ;
    CONSULTAR_1  : BOOL ;
    TIEMPO_1           : S5TIME ;
    TIEMPO_2           : TIME ;
    FECHA_1            : DATE ;
    HORA_1 : TIME_OF_DAY ;
END_VAR
BEGIN

// Asignación de una constante a una variable
    PULSADOR_1            := -17 ;
    VALOR_TEORICO_1  := 100.1 ;
    CONSULTAR_1      := TRUE ;
    TIEMPO_1         := T#1H_20M_10S_30MS ;
    TIEMPO_2         := T#2D_1H_20M_10S_30MS ;
    FECHA_1          := D#1996-01-10 ;

// Asignación de una variable a una variable
    VALOR_TEORICO_1  := VALOR_TEORICO_2 ;
    PULSADOR_2  := PULSADOR_1 ;
// Asignación de una expresión a una variable

    PULSADOR_2  := PULSADOR_1 * 3 ;
END_FUNCTION_BLOCK
```

12.1.2 Asignación con variables del tipo STRUCT y UDT

Las variables del tipo STRUCT y UDT son variables estructuradas que representan una estructura completa o un componente de dicha estructura.

Especificaciones válidas de una variable estructurada:

```
Imagen              //identificador de una estructura
Imagen.elemento     //identificador de un componente de estructura
Imagen.array        //identificador de un array simple
                    //dentro de una estructura
Imagen.array[2,5]   //identificador de un componente de un array
                    //en una estructura
```

Asignación de una estructura completa

Una estructura completa sólo es asignable a otra estructura cuando los componentes de ambas coinciden tanto en sus tipos de datos como en sus nombres.

Asignaciones válidas :

```
structname_1 := structname_2 ;
```

Asignación de componentes de una estructura

A todos los componentes de estructuras se les puede asignar una variable de tipo compatible, una expresión de tipo compatible u otro componente de estructura.

Para direccionar un componente de una estructura se indica el identificador de la estructura y el identificador del componentes de la estructura separado por un punto.

Asignaciones válidas:

```
nombrestruct_1.elemento1           := Valor ;
nombrestruct_1.elemento1           := 20.0 ;
nombrestruct_1.elemento1           := nombrestruct_2.elemento1 ;
nombrestruct_1.nombrearray1        := nombrestruct_2.nombrearray2 ;
nombrestruct_1.nombrearray[10]     := 100 ;
```

Ejemplo

```
FUNCTION_BLOCK FB3
VAR
    VARAYUD : REAL ;
    VALOR_MEDIDO : STRUCT    //Estructura meta
                                TENSION:REAL ;
                                RESISTENCIA:REAL ;
                                ARRAY_SIMPLE : ARRAY
[1..2, 1..2] OF INT ;
                            END_STRUCT ;
    VALOR REAL  : STRUCT    //Estructura fuente
                                TENSION: REAL ;
                                RESISTENCIA : REAL ;
                                ARRAY_SIMPLE: ARRAY [1..2,
1..2] OF INT ;
                            END_STRUCT ;
END_VAR

BEGIN
//Asignación de una estructura completa a una estructura completa
    VALOR_MEDIDO := VALOR REAL ;
//Asignación de un componente de estructura a un componente de
estructura
    VALOR_MEDIDO.TENSION:= VALOR REAL.TENSION ;
//Asignación de un componente de estructura a una variable del mismo
tipo
    VARAYUD := VALOR REAL.RESISTENCIA ;
//Asignación de una constante a un componente de estructura
    VALOR_MEDIDO.RESISTENCIA := 4.5;
//Asignación de una constante a un elemento de array simple
    VALOR_MEDIDO.ARRAY_SIMPLE[1,2] := 4;
END_FUNCTION_BLOCK
```

12.1.3 Asignación con variables del tipo ARRAY

Un array tiene de 1 a 6 dimensiones como máximo y contiene elementos, todos del mismo tipo. Hay dos variantes de acceso para la asignación de arrays a una variable: Es posible direccionar tanto arrays completos como partes de arrays.

Asignación de un array completo

Un array completo puede asignarse a otro array cuando coincidan tanto los tipos de datos de los componentes como los límites de los arrays (índices de array mínimo y máximo). En caso de que coincidan, especifique el identificador del array después del identificador de asignación.

Asignaciones válidas:

```
nom_array_1:=nom_array_2;
```

Nota

Tenga en cuenta que no está permitido asignar constantes a arrays enteros.

Asignación de un componente de array

Para direccionar un componente de array se indica el nombre del array seguido de valores de índice encerrados entre corchetes. Para cada dimensión se dispone de un índice, separado por comas y encerrado asimismo entre corchetes. Un índice debe ser una expresión aritmética del tipo de datos INT.

Para realizar una asignación para un componente de array hay que eliminar los índices entre los corchetes que figuran detrás del nombre del array comenzando por la derecha. De esta forma se direcciona una parte cuyo número de dimensiones es igual al número de índices omitidos. Asignaciones válidas:

```
nombrearray_1[ i ] := nombrearray_2[ j ] ;
nombrearray_1[ i ] := expresion ;
identificador_1 := nombrearray_1[ i ] ;
```

Ejemplo

```
FUNCTION_BLOCK FB3
VAR
   VALORES_TEORICOS    :ARRAY [0..127] OF INT ;
   VALORES_REALES      :ARRAY [0..127] OF INT ;
// Declaración de una matriz (=array bidimensional)
// de 3 líneas y 4 columnas
   REGULADOR : ARRAY [1..3, 1..4] OF INT ;
// Declaración de un vector (=array unidimensional) de 4 componentes
   REGULADOR_1 : ARRAY [1..4] OF INT ;
END_VAR

BEGIN
// Asignación de un array completo a un array
   VALORES_TEORICOS := VALORES_REALES ;
// Asignación de un vector a la segunda línea del array REGULADOR
   REGULADOR[2] := REGULADOR_1 ;
// Asignación de un componente de array a un componente
// del array REGULADOR
   REGULADOR [1,4] := REGULADOR_1 [4] ;
END_FUNCTION_BLOCK
```

12.1.4 Asignación con variables del tipo STRING

Una variable del tipo de datos STRING contiene una cadena de caracteres de 254 caracteres como máximo. A todas las variables del tipo de datos STRING se les puede asignar otra variable del mismo tipo.

Asignaciones válidas:

```
variablestring_1 := constante STRING;
variablestring_1 := variablestring_2 ;
```

Ejemplo

```
FUNCTION_BLOCK FB3
VAR
    INDICADOR_1  : STRING[50] ;
    ESTRUCTURA1  : STRUCT
         INDICADOR_2   : STRING[100] ;
         INDICADOR_3   : STRING[50] ;
    END_STRUCT ;
END_VAR

BEGIN
// Asignación de una constante a una variable STRING
         INDICADOR_1 := 'error en el módulo 1' ;
// Asignación de un componente de estructura a una variable STRING.
         INDICADOR_1 := ESTRUCTURA1.INDICADOR_3 ;
// Asignación de una variable STRING a una variable STRING
         If INDICADOR_1 <> ESTRUCTURA1.INDICADOR_3 THEN
            INDICADOR_1 := ESTRUCTURA1.INDICADOR_3 ;
            END_IF;
END_FUNCTION_BLOCK
```

12.1.5 Asignación con variables del tipo DATE_AND_TIME

El tipo de datos DATE_AND_TIME define un área de 64 bits (8 bytes) para especificar la fecha y la hora. A todas las variables del tipo de datos DATE_AND_TIME se les puede asignar otra variable o constante del mismo tipo.

Asignaciones válidas:

```
variabledt_1 := Constante de fecha y hora;
variabledt_1 := variabledt_2 ;
```

Ejemplo

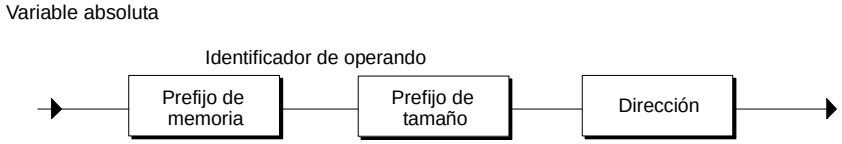
```
FUNCTION_BLOCK FB3
VAR
    TIEMPO_1         : DATE_AND_TIME ;
    ESTRUCTURA1      : STRUCT
            TIEMPO_2 : DATE_AND_TIME ;
            TIEMPO_3 : DATE_AND_TIME ;
    END_STRUCT ;
END_VAR

BEGIN
// Asignación de una constante a una variable DATE_AND_TIME
        TIEMPO_1 := DATE_AND_TIME#1995-01-01-12:12:12.2 ;
        ESTRUCTURA1.TIEMPO_3 := DT#1995-02-02-11:11:11 ;
// Asignación de un componente de estructura a una variable
DATE_AND_TIME.
        TIEMPO_1 := ESTRUCTURA1.TIEMPO_2 ;
// Asignación de una variable DATE_AND_TIME a una variable
DATE_AND_TIME
        If TIEMPO_1 < ESTRUCTURA1.TIEMPO_3 THEN
            TIEMPO_1 := ESTRUCTURA1.TIEMPO_3 ;
        END_IF ;
END_FUNCTION_BLOCK
```

12.1.6 Asignación con variables absolutas para áreas de memoria

La variable absoluta referencia las áreas de memoria globales válidas de una CPU. Existen tres posibilidades para acceder a estas áreas:

- acceso absoluto
- acceso indizado
- acceso simbólico

Sintaxis de las variables absolutas



Ejemplo

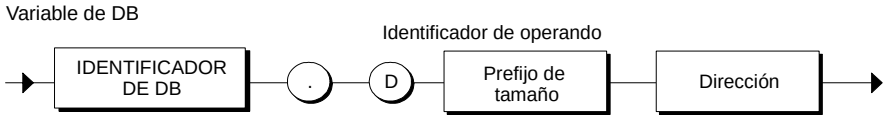
```
FUNCTION_BLOCK FB3
VAR
    PALABRA_DE_ESTADO1 : WORD ;
    PALABRA_DE_ESTADO2 : BOOL ;
    PALABRA_DE_ESTADO3 : BYTE ;
    PALABRA_DE_ESTADO4 : BOOL ;
    DIRECCIÓN          : INT ;
END_VAR
BEGIN
        DIRECCIÓN := 10 ;
// Asignación de una palabra de entrada a una variable
// (acceso sencillo)
        PALABRA_DE_ESTADO1 := EW4 ;
// Asignación de una variable a un bit de salida (acceso sencillo)
        a1.1 := PALABRA_DE_ESTADO2 ;
// Asignación de un byte de entrada a una variable (acceso indexado)
        PALABRA_DE_ESTADO3 := EB[DIRECCIÓN] ;
// Asignación de un bit de entrada a una variable (acceso indexado)
        DIRECCION_FOR := 0 TO 7 BY 1 DO
            PALABRA_DE_ESTADO4 := e[1, DIRECCIÓN] ;
        END_FOR ;
END_FUNCTION_BLOCK
```

12.1.7 Asignación con variables globales

El acceso a datos de bloques de datos globales también se realiza mediante asignación a variables del mismo tipo, o viceversa. A todas las variables globales se les puede asignar una variable o una expresión del mismo tipo. Dispone de varias posibilidades de acceder a estos datos:

- acceso estructurado
- acceso absoluto
- acceso indizado

Sintaxis de la variable DB



Ejemplo

```
FUNCTION_BLOCK FB3
VAR
    REGULADOR_1  : ARRAY [1..4] OF INT ;
    PALABRA_DE_ESTADO1 : WORD ;
    PALABRA_DE_ESTADO2 : ARRAY [0..10] OF WORD ;
    PALABRA_DE_ESTADO3 : INT ;
    PALABRA_DE_ESTADO4 : WORD ;
     DIRECCION   : INT ;
END_VAR
VAR_INPUT
    PALABRA _DE_DIRECCION    : WORD ;
END_VAR
BEGIN
// Asignación de la palabra 1 del DB11
//a una variable (acceso simple)
        PALABRA_DE_ESTADO1 := DB11.DW1 ;
// Al componente del array de la 1ª línea y
// de la 1ª columna de la matriz se le asigna
// el valor de la variable "NUMERO" (acceso configurado):
        REGULADOR_1[1] := DB11.NUMERO ;
// Asignación del componente de estructura "NUMERO2"
// de la estructura "NUMERO1" a la variable palabra de estado3
        PALABRA_DE_ESTADO3  := DB11.NUMERO1.NUMERO2 ;
// Asignación de una palabra con dirección de índice
// del DB11 a una variable (acceso indexado)
        FOR
          DIRECCION := 1 TO 10 BY 1 DO
          PALABRA_DE_ESTADO2[DIRECCION] := DB11.DW[DIRECCION] ;
          // Aquí se utilizan el parámetro de entrada
          // PALABRA_DE_DIRECCION como
          // número del DB y el índice DIRECCIÓN para indicar
          // la dirección de
          // palabra en el DB.
             PALABRA_DE_ESTADO4 :=
WORD_TO_BLOCK_DB(PALABRA_DE_DIRECCION).DW[DIRECCION] ;
     END_FOR ;
END_FUNCTION_BLOCK
```

12.2 Instrucciones de control

12.2.1 Sinopsis de las instrucciones de control

Instrucciones de selección

Una instrucción de selección permite derivar el flujo del programa a distintas secuencias de instrucciones en función de determinadas condiciones.

Tipo de ramificación	Función
Instrucción IF	Con la instrucción IF se deriva el flujo del programa a una de dos alternativas en función de una condición que puede ser TRUE o FALSE.
Instrucción CASE	Con la instrucción CASE se puede controlar el flujo del programa a una ramificación 1:n, haciendo que una variable adopte un valor de entre n valores posibles.

Edición de bucles

El procesamiento de bucles se puede controlar mediante instrucciones de repetición. Una instrucción de repetición indica las partes del programa que deben repetirse en determinadas condiciones.

Tipo de ramificación	Función
Instrucción FOR	Sirve para repetir una secuencia de instrucciones hasta que la variable en ejecución se encuentre dentro del margen indicado.
Instrucción WHILE	Sirve para repetir una secuencia de instrucciones hasta que se cumpla una condición de ejecución.
Instrucción REPEAT	Sirve para repetir una secuencia de instrucciones hasta que se cumpla una condición de interrupción.

Salto del programa

Un salto del programa hace saltar el programa inmediatamente a una meta especificada y, por lo tanto, a otra instrucción del mismo bloque.

Tipo de ramificación	Función
Instrucción CONTINUE	Sirve para interrumpir la ejecución del bucle que se está ejecutando en este momento.
Instrucción EXIT	Sirve para abandonar un bucle en un punto cualquiera e independientemente de que se cumpla o no la condición de interrupción.
Instrucción GOTO	Hace saltar el programa a una meta de salto especificada.
Instrucción RETURN	Abandona el bloque que se está procesando actualmente.

12.2.2 Condiciones

La condición puede ser una expresión de comparación, una expresión lógica o una expresión aritmética. Es del tipo BOOL y puede adoptar los valores TRUE o FALSE. Las expresiones aritméticas son TRUE, si adoptan un valor no igual a cero y FALSE, si resultan cero. La siguiente tabla muestra ejemplos para condiciones:

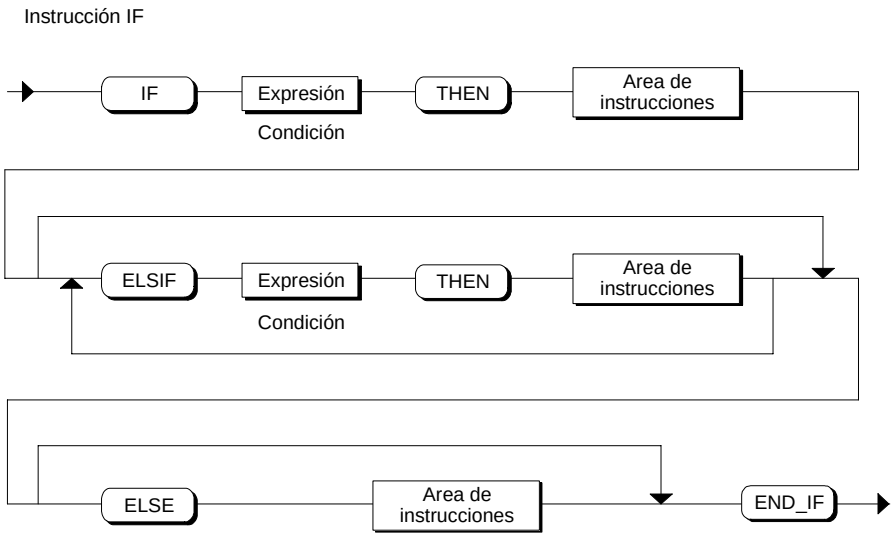
Tipo	**Ejemplo**
Expresión de comparación	TEMP > 50 CONTADOR<= 100 CHAR1 < 'S'
Expresión de comparación y lógica	(ALPHA <> 12) AND NOT BETA
Operando booleano	E1.1
Expresión aritmética	ALPHA = (5 + BETA)

12.2.3 Instrucción IF

La instrucción IF es una instrucción condicional. Ofrece una o varias opciones y selecciona una de sus área de instrucciones (en algunos casos ninguna); la opción seleccionada se ejecutará.

Al ejecutar la instrucción condicional se evalúan las expresiones lógicas especificadas. Si el valor de una expresión es TRUE, la condición se cumple; si el valor es FALSE, la condición no se cumple.

Sintaxis



La instrucción IF se procesa según las reglas siguientes:

- Se ejecuta la primera secuencia de instrucciones cuya expresión lógica = TRUE. Las restantes secuencias de instrucciones no se ejecutan.
- Si no hay ninguna expresión booleana = TRUE, se ejecuta la secuencia de instrucciones de ELSE (o ninguna, si no existe rama ELSE).
- Puede existir cualquier número de instrucciones ELSIF.

Nota

Frente a una cadena de instrucciones-IF, el uso de una o varias ramas ELSIF ofrece la ventaja de que las expresiones lógicas que siguen a una expresión válida ya no se evalúan. Ello permite reducir el tiempo de ejecución de un programa.

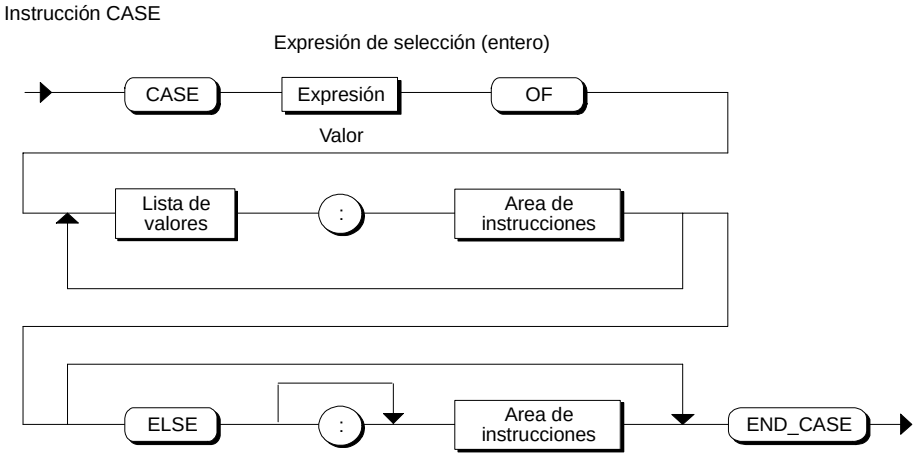
Ejemplo

```
IF  E1.1 THEN
        N := 0 ;
        SUM := 0 ;
        OK := FALSE ;  // Poner OK-Flag a FALSE
ELSIF  START = TRUE THEN
        N := N + 1 ;
        SUM := SUM + N ;
ELSE
        OK := FALSE ;
END_IF ;
```

12.2.4 Instrucción CASE

La instrucción CASE sirve para elegir entre 1-n secciones alternativas del programa. Esta opción se basa en el valor de una expresión de selección.

Sintaxis

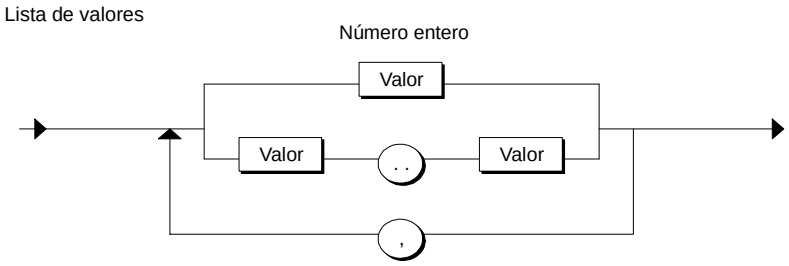


La instrucción CASE se procesa según las reglas siguientes:

- La expresión de selección deberá suministrar un valor del tipo ENTERO.
- Al procesar la instrucción CASE se comprueba si el valor de la expresión de selección está incluida en una lista de valores especificada. En caso de coincidencia se ejecuta el área de instrucciones asignada de la lista.
- Si el proceso de comparación no localiza ninguna coincidencia, se ejecuta el área de instrucciones que sigue a ELSE, o no se ejecuta ninguna instrucción si no existe rama ELSE.

Lista de valores

La lista de valores contiene los valores permitidos para la expresión de selección.

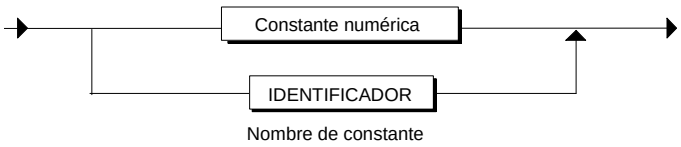


Se aplican las siguientes reglas:

- Todas las listas de valores empiezan por una constante, una lista de constantes o un área de constantes.
- Los valores que se encuentran en la lista de valores deben ser del tipo ENTERO.
- Cada valor debe figurar una sola vez.

Valor

El valor sigue la sintaxis que figura más abajo:



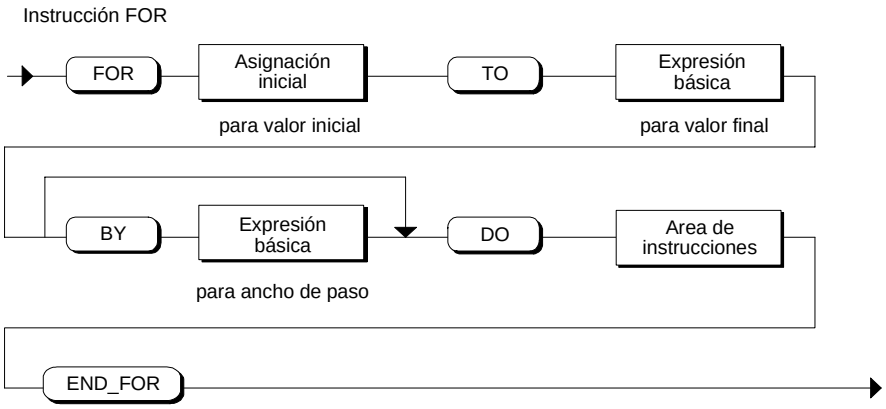
Ejemplo

```
CASE TW OF
        1 :             DISPLAY:= OVEN_TEMP;
        2 :             DISPLAY:= MOTOR_SPEED;
        3 :             DISPLAY:= GROSS_TARE;
                        AW4:= 16#0003;
        4..10:          DISPLAY:= INT_TO_DINT (TW);
                        AW4:= 16#0004;
        11,13,19:       DISPLAY:= 99;
                        AW4:= 16#0005;
ELSE:
        DISPLAY:= 0;
        TW_ERROR:= 1;
END_CASE ;
```

12.2.5 Instrucción FOR

Una instrucción FOR sirve para repetir una secuencia de instrucciones mientras haya una variable de control dentro del rango especificado. La variable actual debe ser el identificador de una variable local del tipo INT o DINT. La definición de un bucle con FOR incluye la definición de un valor inicial y de un valor final. Ambos valores deben ser del mismo tipo que las variables en ejecución.

Sintaxis



La instrucción FOR se procesa de la siguiente forma:

- Al iniciar el bucle la variable de control toma el valor inicial (asignación inicial) y, cada vez que recorre el bucle, se produce el incremento (incremento positivo) o el decremento especificado (incremento negativo) hasta que se alcanza el valor final.
- Cada vez que se recorre el bucle se comprueba si se ha cumplido la condición (valor final alcanzado). En caso afirmativo se ejecuta la secuencia de instrucciones; en caso contrario se salta el bucle y, por tanto, la secuencia de instrucciones.

Reglas

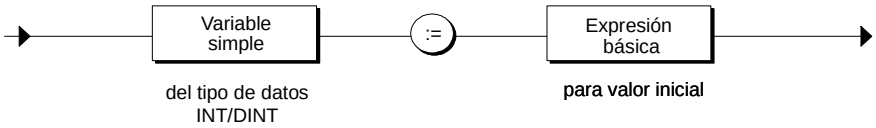
Al formular instrucciones FOR deben aplicarse las siguientes reglas:

- La variable de control sólo puede ser del tipo INT o DINT.
- La especificación de BY [paso] puede omitirse. Si falta la indicación, el paso es de +1.

Asignación del valor inicial

La creación del valor inicial de la variable actual debe corresponder a la siguiente sintaxis. La variable simple en el lado izquierdo de la asignación debe ser del tipo de datos INT o DINT.

Asignación inicial

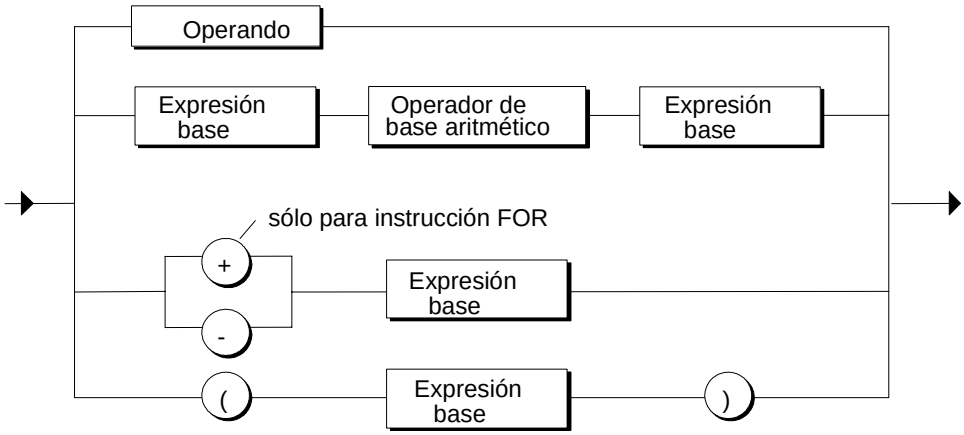


Ejemplos de asignaciones iniciales:

```
FOR I := 1 TO 20
FOR I := 1 TO (INICIO + J)
```

Valor final e incremento

Para la creación del valor final y del incremento deseado se puede crear respectivamente una expresión básica. La creación de una expresión básica debe corresponder a la siguiente sintaxis:



- La especificación de BY [paso] puede omitirse. Si falta la indicación, el incremento es de `+1`.
- El valor inicial, el valor final y el incremento son expresiones (consulte el capítulo "Expresiones, operaciones, operandos). La evaluación tiene lugar al comienzo de la ejecución de la instrucción FOR.
- No está permitido modificar los dos valores (del valor final y del incremento) durante la ejecución.

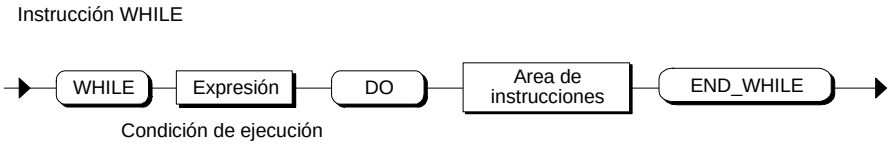
Ejemplo

```
FUNCTION_BLOCK FOR_BSP
VAR
    INDEX: INT ;
    PALABRA_CLAVE: ARRAY [1..50] OF STRING;
END_VAR
BEGIN
FOR INDEX := 1 TO 50 BY 2 DO
    PALABRA_CLAVE IF [INDEX] = 'KEY' THEN
           EXIT;
    END_IF;
END_FOR;

END_FUNCTION_BLOCK
```

12.2.6 Instrucción WHILE

La instrucción WHILE permite la ejecución repetida de una secuencia de instrucciones bajo el control de una condición de ejecución. La condición de ejecución se crea según la regulación de una expresión lógica.

Sintaxis



La instrucción WHILE se ejecuta de acuerdo con las siguientes reglas:

- Antes de cada ejecución del área de instrucciones se valora la condición de ejecución (bucle de rechazo).
- El área de instrucciones que sigue a DO se repite tantas veces como la condición de ejecución arroje el valor TRUE.
- Si se produce el valor FALSE, se salta el bucle y se ejecuta la instrucción que le sigue.

Ejemplo

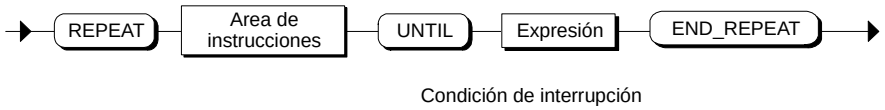
```
FUNCTION_BLOCK WHILE_BSP
VAR
    INDEX: INT ;
    PALABRA_CLAVE: ARRAY [1..50] OF STRING ;
END_VAR
BEGIN
INDEX := 1 ;
WHILE INDEX <= 50 AND PALABRA_CLAVE [INDEX] <> 'KEY' DO
    INDEX := INDEX + 2;
END_WHILE ;
END_FUNCTION_BLOCK
```

12.2.7 Instrucción REPEAT

La instrucción REPEAT permite la ejecución repetida de una secuencia de instrucciones que se encuentre entre REPEAT y UNTIL hasta que se cumpla una condición de interrupción. La condición de interrupción se forma según las reglas de una expresión lógica.

Sintaxis

Instrucción REPEAT



La condición se comprueba después de ejecutar el cuerpo de la instrucción. Esto significa que el cuerpo se ejecutará como mínimo una vez, aunque la condición de interrupción se cumpla desde el principio.

Ejemplo

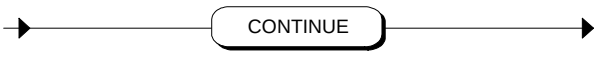
```
FUNCTION_BLOCK REPEAT_EXAMP
VAR
    INDEX: INT ;
    PALABRA_CLAVE: ARRAY [1..50] OF STRING ;
END_VAR

BEGIN
INDEX := 0 ;
REPEAT
    INDEX := INDEX + 2 ;
UNTIL INDEX > 50 PALABRA CLAVE OR[INDEX] = 'KEY'
END_REPEAT ;

END_FUNCTION_BLOCK
```

12.2.8 Instrucción CONTINUE

Una instrucción CONTINUE sirve para cancelar la ejecución del bucle momentáneo de una instrucción de repetición (FOR, WHILE o REPEAT).

Sintaxis

Instrucción CONTINUE



La instrucción CONTINUE se ejecuta según las reglas siguientes:

- Esta instrucción interrumpe de forma inmediata la ejecución de la secuencia de instrucciones.
- Dependiendo de si se cumple o no la condición de repetición del bucle, se ejecuta de nuevo el cuerpo o se abandona la instrucción de repetición y se ejecuta la instrucción inmediatamente posterior.
- En una instrucción FOR inmediatamente después de una instrucción CONTINUE, la variable de control aumenta el incremento especificado.

Ejemplo

```
FUNCTION_BLOCK CONTINUE_EXAMP
VAR
    INDEX  :INT ;
    FELD   :ARRAY[1..100] OF INT ;
END_VAR

BEGIN
INDEX := 0 ;
WHILE INDEX <= 100 DO
    INDEX := INDEX + 1 ;
    // Si ARRAY [INDEX] es igual a INDEX,
    // no se modificará ARRAY [INDEX] :
            ARRAY IF[INDEX] = INDEX THEN
                  CONTINUE ;

            END_IF ;
        ARRAY[INDEX] := 0 ;
    // Otras instrucciones
END_WHILE ;
END_FUNCTION_BLOCK
```

12.2.9 Instrucción EXIT

La instrucción EXIT permite abandonar un bucle (FOR, WHILE o REPEAT) en cualquier punto del programa, independientemente de si se cumple o no la condición de interrupción.

Sintaxis

Instrucción EXIT

La instrucción EXIT se ejecuta según las reglas siguientes:

- Con esta instrucción se abandona de inmediato la instrucción de repetición en la que se encuentra la instrucción EXIT.
- El programa prosigue la ejecución después de la instrucción de fin del bucle de repetición (p.ej., después de END_FOR).

Ejemplo

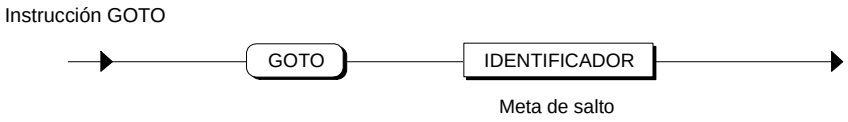
```
FUNCTION_BLOCK EXIT_EXAMP
VAR
    INDEX_1      : INT ;
    INDEX_2      : INT ;
    INDEX_BUSCADO      : INT ;
    PALABRA CLAVE      : ARRAY[1..51] OF STRING ;
END_VAR

BEGIN
INDEX_2    := 0 ;
FOR INDEX_1      := 1 TO 51 BY 2 DO
    // salir del bucle FOR si la
    // PALABRA CLAVE[INDEX_1] es igual a 'KEY':
    PALABRA CLAVE IF[INDEX_1] = 'KEY' THEN
        INDEX_2 := INDEX_1 ;
        EXIT ;
    END_IF ;
END_FOR ;
// La siguiente asignación de valor se
// realizará después de ejecutar
// EXIT o después del final regular del bucle FOR:
INDEX_BUSCADO:= INDEX_2 ;
END_FUNCTION_BLOCK
```

12.2.10 Instrucción GOTO

La instrucción GOTO permite realizar un salto dentro del programa. Cuando se ejecuta esta instrucción se provoca un salto inmediato a una meta especificada y, por lo tanto, a otra instrucción del mismo bloque.

Las instrucciones GOTO sólo se deberían utilizar en casos especiales, p. ej., cuando se procesan errores. Según las reglas de programación estructurada no se debe utilizar la instrucción GOTO.

Sintaxis



La meta del salto designa una marca del bloque de declaración LABEL/END_LABEL. Esta marca precede a la instrucción que se ejecuta después de GOTO.

Al utilizar la instrucción GOTO deben observarse las siguientes reglas:

- El destino de una instrucción de salto debe encontrarse dentro del mismo bloque.
- El destino del salto debe ser inequívoco.
- No se puede saltar a un bucle, pero sí desde un bucle.

Ejemplo

```
FUNCTION_BLOCK GOTO_BSP
VAR
    INDEX  : INT ;
    A      : INT ;
    B      : INT ;
    C      : INT ;
    PALABRA_CLAVE      : ARRAY[1..51] OF STRING ;
END_VAR
LABEL
    META1, META2, META3 ;
END_LABEL

BEGIN
IF A > B THEN
    GOTO META1 ;
ELSIF A > C THEN
    GOTO ETIQUETA2 ;
END_IF ;
// . . .
META1: INDEX := 1 ;
                GOTO META3 ;
META2: INDEX := 2 ;
// . . .
META3:
// . . .
```

12.2.11 Instrucción RETURN

Con una instrucción RETURN se abandona el bloque actual (OB, FB, FC) en ejecución y se retorna al módulo invocante, o bien al sistema operativo en caso de que se abandone un bloque de organización (OB).

Sintaxis

Instrucción RETURN

Nota

Una instrucción RETURN situada al final del área de instrucciones de un bloque lógico o al final del área de declaración de un bloque de datos es redundante, puesto que se ejecuta automáticamente.

12.3 Llamada a funciones y bloques de función

12.3.1 Llamada y transferencia de parámetros

Llamada a FC y FB

Para garantizar la legibilidad y facilitar el mantenimiento de los programas de usuario, toda la funcionalidad del programa se subdivide en tareas parciales que luego ejecutan los bloques de función (FB) y las funciones (FC). Desde un bloque S7-SCL se pueden realizar llamadas a otros FC o FB. Puede llamarse a los siguientes bloques:

- bloques de función y funciones creados en S7-SCL;
- bloques de función y funciones creados en otros lenguajes de S7 (KOP, FUP, AWL);
- funciones del sistema (SFC) y bloques de función del sistema (SFB) disponibles en el sistema operativo de la CPU.

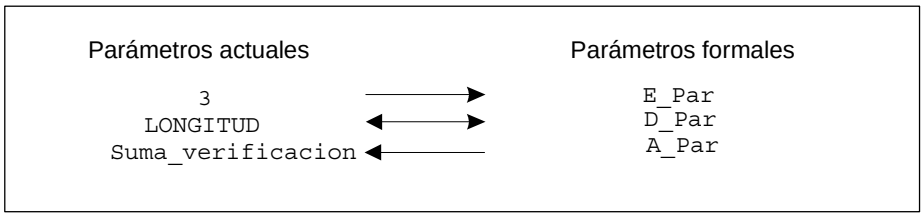
Principio de la transferencia de parámetros

Al llamar funciones o bloques de función se produce un intercambio de datos entre el bloque invocante y el bloque llamado. En el interface del bloque llamado hay definidos unos parámetros con los que opera el bloque. Estos parámetros se denominan parámetros formales. Son simplemente "comodines" para los parámetros que se transferirán al bloque cuando realice una llamada. Los parámetros transferidos al realizar la llamda reciben el nombre de parámetros actuales.

Sintaxis de la transferencia de parámetros

Los parámetros que deben transferirse deben especificarse en la llamada como lista de parámetros. Los parámetros se escriben entre paréntesis. Si son varios parámetros se separan mediante comas.

En el siguiente ejemplo de llamada a función se indican un parámetro de entrada, uno de salida y uno de entrada/salida.



La indicación de parámetros presenta la forma de una asignación de valor. Mediante esta asignación de valor se asigna un valor determinado (parámetro actual) a los parámetros que ha definido en la tabla de declaración del bloque llamado (parámetros formales).

Llamada absoluta:

```
FB10.DB20 (X1:=5,X2:=78,......);
```

Especificación de parámetros

Llamada simbólica:

```
ACCIONAMENTO.INTRO (X1:=5,X2:=78,......);
```

12.3.2 Llamada a bloques de función

12.3.2.1 Llamada a bloques de función (FB o SFB)

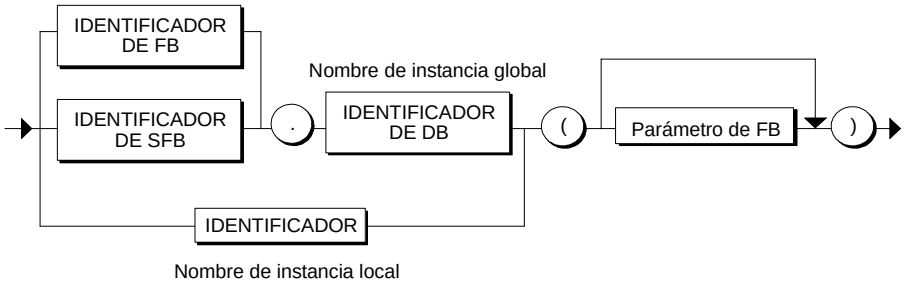
Para llamar a un bloque de función puede utilizar bloques de datos de instancia globales y áreas del bloque de datos de instancia actual.

La llamada a un FB como instancia local se diferencia de la llamada como instancia global en la memorización de los datos. En este último caso los datos no se depositan en un DB aparte, sino en el bloque de datos de instancia del FB invocante.

Sintaxis

Llamada a FB

FB: bloque de función
SFB: bloque de función del sistema



Llamada como instancia global

La llamada se efectúa en una instrucción de llamada especificando:

- el nombre del bloque de función o del bloque de función del sistema (nombre del FB o SFB),
- el bloque de datos de instancia (nombre del DB),
- y la asignación de parámetros (de FB).

Una llamada a una instancia global se puede definir de forma tanto absoluta como simbólica.

Llamada absoluta:

```
FB10.DB20 (X1:=5,X2:=78,......);
```

Especificación de parámetros

Llamada simbólica:

```
ACCIONAMENTO.INTRO (X1:=5,X2:=78,......);
```

Llamada como instancia local#

La llamada se efectúa en una instrucción de llamada especificando:

- el nombre de instancia local (IDENTIFICADOR),
- y la asignación de parámetros (parámetros de FB).

La llamada a una instancia local siempre es simbólica. Los nombres simbólicos deben especificarse en el área de declaración del bloque invocante.

```
MOTOR (X1:=5,X2:=78,......);
                        ↑
                        Especificación de parámetros
```

12.3.2.2 Asignación de parámetros FB

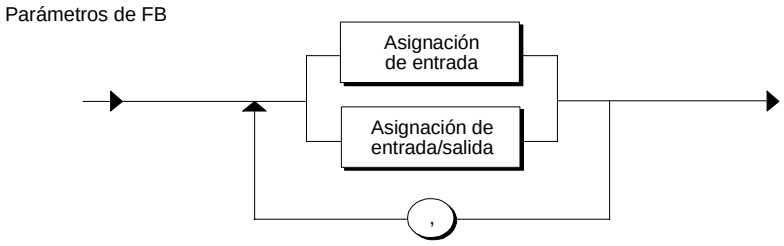
Al llamar a un bloque de función - como instancia global o local - se deben parametrizar los siguientes parámetros de la lista de parámetros:

- parámetros de entrada
- parámetros de entrada/salida

Los parámetros de salida no se especifican al llamar a un FB, sino que se asignan después de la llamada.

Sintaxis de una asignación de valor para definir los parámetros de FB:

La sintaxis de las indicaciones de parámetros de FB es igual al llamar instancias globales y locales.



Para la asignación de parámetros se aplican las siguientes reglas:

- Las asignaciones pueden estar en cualquier orden.
- El tipo de datos de los parámetros formales y de los parámetros actuales debe coincidir.
- Cada una de las asignaciones se separan por comas.
- No es posible realizar asignaciones de salida en llamadas a FB. El valor de un parámetro de salida declarado está memorizado en los datos de instancia, donde están accesibles para todos los FB. Para poder leerlo es necesario definir un acceso desde un FB.
- Tenga en cuenta las particularidades de los parámetros del tipo de datos ANY y del tipo de datos POINTER.

Resultado tras ejecutar el bloque

Tras ejecutar el bloque:

- los parámetros de entrada actuales transferidos permanecen inalterados.
- los valores transferidos pero modificados de los parámetros de entrada/salida están actualizados. Una excepción la constituyen los parámetros de entrada/salida de un tipo de datos simple.
- se pueden leer los parámetros de salida del bloque invocante desde bloque de datos de instancia global o el área de instancia local.

Ejemplo

Una llamada con una asignación de un parámetro de entrada y un parámetro de entrada/salida podría presentar el siguiente aspecto, por ejemplo:

```
FB31.DB77(E_Par:=3, D_Par:=LONGITUD);
```

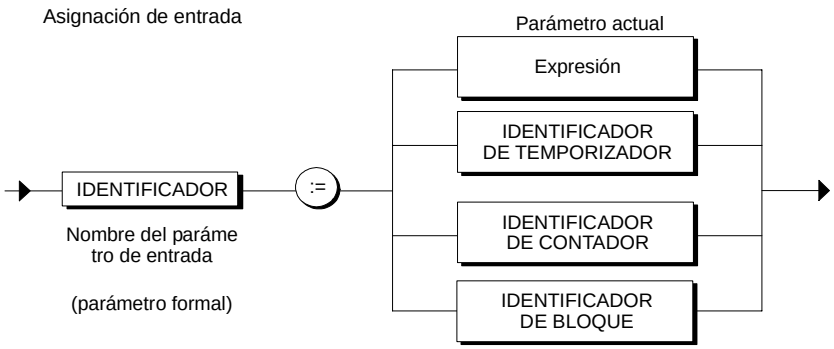
12.3.2.3 Asignación de entrada (FB)

Mediante las asignaciones de entrada se asignan parámetros actuales a los parámetros formales de entrada. El FB no puede cambiar estos parámetros actuales. La asignación de parámetros de entrada actuales es opcional. Si no se especifica ningún parámetro actual, se mantienen los valores de la última llamada.

Parámetros actuales posibles:

Parámetro actual	**Explicación**
THEN	Expresión aritmética, lógica o de comparación Constante Variable ampliada
Identificador de TEMPORIZADOR/CONTADOR	Define un temporizador o un contador determinado a utilizar en la ejecución de un bloque
Identificador de BLOQUE	Define un determinado bloque a utilizar como parámetro de entrada. El tipo de bloque (FB, FC, DB) se determina en la declaración del parámetro de entrada. Al especificar el parámetro indique el número del bloque, ya sea el absoluto o el simbólico.

Sintaxis



12.3.2.4 Asignación de entrada/salida (FB)

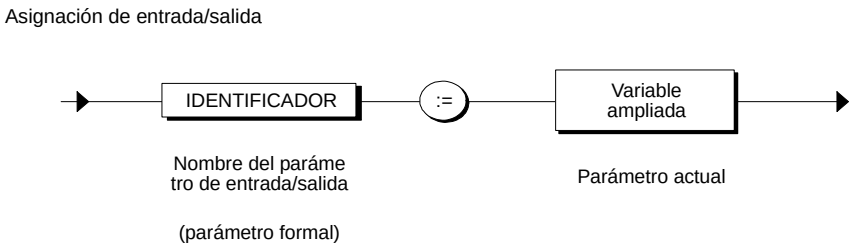
Las asignaciones de entrada/salida sirven para asignar parámetros actuales a los parámetros de entrada/salida formales del FB llamado. El FB llamado puede modificar los parámetros de entrada/salida. El nuevo valor del parámetro que se genera al procesar el FB se reescribe en el parámetro actual, escribiéndose sobre el valor original.

Si en el FB llamado hay declarados parámetros de entrada/salida, deben recibir valores en la primera llamada. En las restantes, la especificación de parámetros actuales es opcional. Con los parámetros de entrada/salida de un tipo de datos simple no se actualiza el parámetro actual si los parámetros formales no reciben valores en la llamada.

Dado que es posible modificar el parámetro actual asignado durante la ejecución del FB y convertirlo en un parámetro de entrada/salida, el parámetro actual debe ser una variable:

Parámetro actual	Explicación
Variable ampliada	Se dispone de los siguientes tipos de variables ampliadas: Variables simples y parámetros Acceso a variables absolutas Acceso a bloques de datos Llamadas a funciones

Sintaxis



Nota

Con tipos de datos ANY y POINTER la asignación se rige por reglas especiales.

En los parámetros de entrada/salida no se permite utilizar como parámetros actuales tipos de datos que no sean simples :

- Parámetros de entrada/salida de FB
- Parámetros de FC

12.3.2.5 Leer valores de salida (llamada a FB)

Después de ejecutar el bloque llamado, los parámetros de salida se pueden leer desde el bloque de instancia global o desde el área de instancia local mediante una asignación de valor.

Ejemplo

```
RESULTADO := DB10.CONTROL ;
```

12.3.2.6 Ejemplo de llamada como instancia global

Un bloque de función con un bucle FOR puede tener un aspecto similar al del siguiente ejemplo. En los ejemplos se presupone que se ha declarado el símbolo `TEST` para el `FB17`.

Bloque de función

```
FUNCTION_BLOCK TEST

VAR_INPUT
    VALOR_FINAL: INT;  //parámetro de entrada
END_VAR
VAR_IN_OUT
    IQ1    :     REAL; //parámetro entrada/salida
END_VAR
VAR_OUTPUT
    CONTROL:     BOOL;//parámetro de salida
END_VAR
VAR
  INDEX:   INT;
END_VAR

BEGIN
CONTROL    :=FALSE;
FOR INDEX  := 1 TO VALOR_FINAL DO
    IQ1    :=IQ1*2;
        IF IQ1 > 10000 THEN
            CONTROL    := TRUE;
        END_IF;
END_FOR;
END_FUNCTION_BLOCK
```

Llamada

Para llamar a este FB puede seleccionar una de las siguientes variantes. Es necesario que la VARIABLE1 del bloque invocante esté registrada como variable REAL.

```
//Llamada absoluta, instancia global:
FB17.DB10 (VALOR_FINAL:=10, IQ1:=VARIABLE1);

//llamada simbólica, instancia global:
TEST.TEST_1 (VALOR_FINAL:= 10, IQ1:= VARIABLE1) ;
```

Resultado

Después de ejecutar el bloque, se dispone del valor hallado para el parámetro de entrada/salida IQ1 en la VARIABLE1.

Leer valores de salida

Con estos dos ejemplos se explican las variantes posibles para la lectura del parámetro de salida CONTROL:

```
// El acceso al parámetro de salida
//se efectúa mediante:
        RESULTADO:= DB10.CONTROL;

//También se puede acceder al parámetro de salida
//con otra llamada a FB
//para la asignación de un parámetro de entrada:
        FB17.DB12 (EIN_1:= DB10.CONTROL);
```

12.3.2.7 Ejemplo de llamada como instancia local

Es posible programar un bloque de función con un bucle FOR sencillo como en el ejemplo "Llamada como instancia global", suponiendo que en la tabla de símbolos esté registrado el símbolo `TEST` para el `FB17` .

Este FB se puede llamar de la siguiente forma siempre que la VARIABLE1 del bloque invocante esté declarada como variable REAL:

Llamada

```
FUNCTION_BLOCK CALL
VAR
// declaración de instancia local
        TEST_L : TEST ;
        VARIABLE1 : REAL ;
        RESULTADO : BOOL ;
END_VAR
BEGIN
. . .

// llamada, instancia local:
TEST_L (VALOR_FINAL:= 10, IQ1:= VARIABLE1) ;
```

Leer valores de salida

El parámetro de salida CONTROL se puede leer de la siguiente forma:
//El acceso al parámetro de salida
//se efectúa mediante:
RESULTADO := TEST_L.CONTROL ;
END_FUNCTION_BLOCK

12.3.3 Llamada a funciones

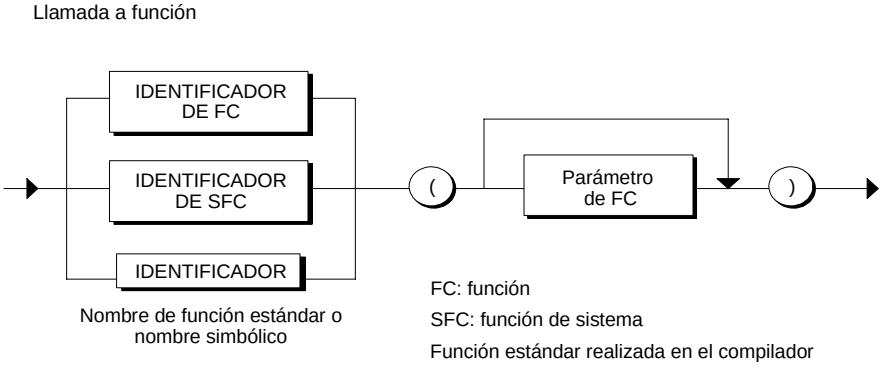
12.3.3.1 Llamada a funciones (FC)

Se llama a una función especificando el nombre de la función (NOMBRE de la FC, de la SFC o IDENTIFICADOR) y la lista de parámetros. El nombre de la función que identifica el valor de respuesta puede especificarse absoluta o simbólicamente:

```
FC31 (X1:=5, Q1:=Suma de verificacion)  // Absoluto
DISTANCIA (X1:=5, Q1=:Suma de verificacion)  // Simbólico
```

Después de la llamada se dispone de los resultados de la función en el valor de respuesta, o de parámetros de salida y de entrada/salida (parámetros actuales).

Sintaxis



Nota

Si en S7-SCL se llama a una función cuyo valor de respuesta no haya sido asignado, puede ejecutarse erróneamente el programa de usuario:

- En una función programada en S7-SCL puede suceder cuando el valor de respuesta ha sido asignado, pero no se ha ejecutado la instrucción correspondiente.
- En una función programada en AWL/KOP/FUP puede suceder cuando la función se ha programado sin asignar el valor de respuesta o cuando no se ha ejecutado la instrucción correspondiente.

12.3.3.2 Valor de respuesta (FC)

A diferencia de los bloques de función, las funciones dan como resultado el valor de respuesta. Por este motivo, las funciones pueden tratarse como si fueran operandos (a excepción de las funciones del tipo VOID).

La función calcula el valor de respuesta que tiene el mismo nombre que la función, y lo retorna al bloque que efectúa la llamada. Alí este valor sustituye a la llamada de la función.

En la siguiente asignación de valor se llama, p.ej., la función `DISTANCIA` y se asigna el resultado de la variable `LONGITUD`:

```
LONGITUD:= DISTANCIA (X1:=-3, Y1:=2);
```

El valor de respuesta se puede utilizar en los siguientes elementos de la FC o del FB:

- en una asignación de valor,
- en una expresión lógica, aritmética o de comparación, o
- como parámetro para llamar a otra función o a otro bloque de función.

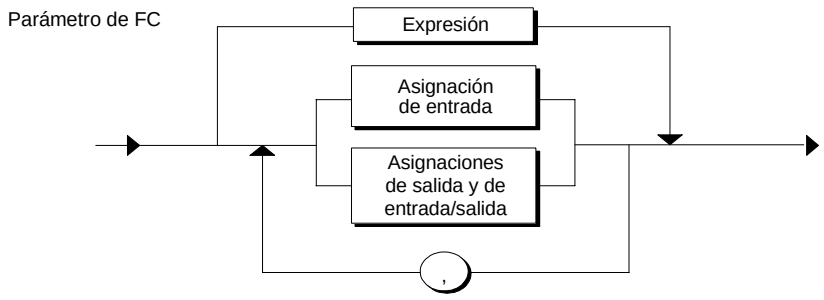
Nota

- En las funciones con valor de retorno ANY tiene que haber por lo menos un parámetro de entrada o de entrada/salida del tipo ANY. Si ha definido varios parámetros ANY, debe asignarles parámetros actuales del mismo tipo (p.ej., INT, DINT y REAL). Entonces el valor de respuesta será automáticamente del tipo de datos de mayor tamaño que se ha utilizado en esta clase.
- La longitud máxima del tipo de datos STRING es ahora de 254 caracteres, pero puede reducirse al número deseado de caracteres.

12.3.3.3 Parámetros de FC

A diferencia de los bloques de función, las funciones no pueden memorizar los valores de los parámetros. Al ejecutarse la función, los datos locales sólo se guardan temporalmente. Por esta razón, al llamar una función es necesario asignar parámetros actuales a todos los parámetros formales de entrada, de entrada/salida y de salida que estén definidos en el área de declaración de la función.

Sintaxis



Reglas

Para la asignación de parámetros se aplican las siguientes reglas:

- Las asignaciones pueden aparecer en cualquier orden.
- El tipo de datos de los parámetros formales y de los parámetros actuales debe coincidir.
- Cada una de las asignaciones se separan por comas.

Ejemplo

Una llamada con tal asignación de un parámetro de entrada, salida y entrada/salida podría presentar este aspecto, por ejemplo:

```
FC32 (E_Param1:=5,D_Param1:=LONGITUD,
     A_Param1:=suma de control)
```

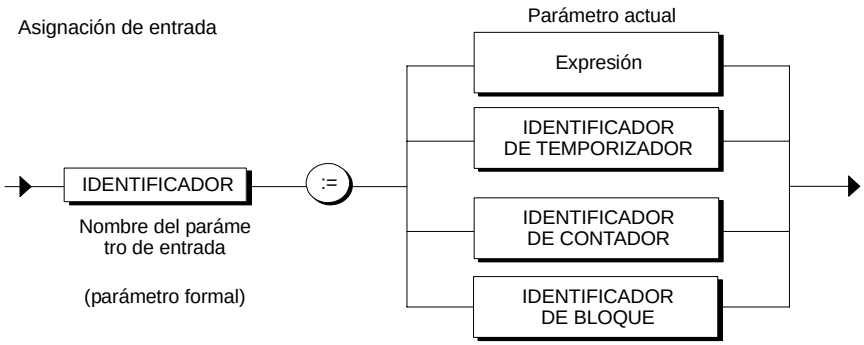
12.3.3.4 Asignación de entrada (FC)

Mediante asignaciones de entrada se asignan valores (parámetros actuales) a los parámetros de entrada formales de la FC llamada. La FC puede operar con dichos parámetros actuales, pero no puede modificarlos. Al contrario que en las llamadas a FB, esta asignación no es opcional en las llamadas a FC.

En las asiganciones de entrada se pueden asignar los siguientes parámetros actuales:

Parámetro actual	Explicación
THEN	Una expresión representa un valor y está formada por operandos y operaciones. Se dispone de los siguientes tipos de expresiones: • Expresión aritmética, lógica o de comparación • Constante • Variable ampliada
Identificador de TEMPORIZADOR/CONTADOR	Define un temporizador o un contador determinado que debe utilizarse en la ejecución de un bloque
Identificador de BLOQUE	Define un determinado bloque que debe utilizarse como parámetro de entrada. El tipo de bloque (FB, FC, DB) se define en la declaración del parámetro de entrada. Al especificar el parámetro indique el número del bloque. El número puede ser tanto absoluto como simbólico

Sintaxis



Nota

En los parámetros de entrada formales que no sean de un tipo de datos simple no se permite el uso de parámetros de entrada/salida FB y parámetros FC como parámetros actuales. Tenga en cuenta las particularidades del tipo de datos ANY y del tipo de datos POINTER.

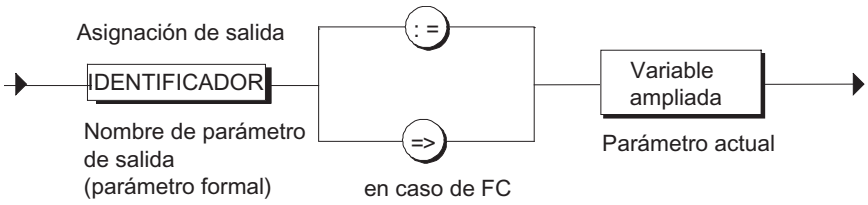
12.3.3.5 Asignación de salida y de entrada/salida (FC)

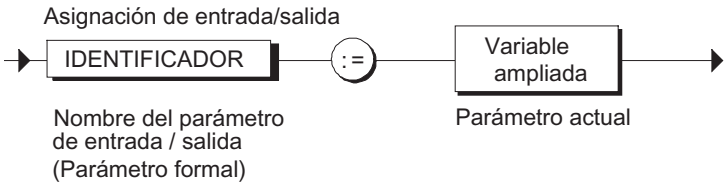
En una asignación de salida se determina en qué variable del bloque invocante se van a escribir los valores de salida que se originan al ejecutarse la función. Con una asignación de entrada/salida se asigna un valor actual a un parámetro de entrada/salida.

Los parámetros actuales de las asignaciones de salida y entrada/salida deben ser una variable, puesto que la función debe escribir valores en los parámetros. Por este motivo no se puede asignar un parámetro de entrada en una asignación de entrada/salida (el valor no podría escribirse). Por consiguiente, en las asignaciones de salida y entrada/salida sólo se puede asignar la variable ampliada:

Parámetro actual	**Explicación**
Variable ampliada	Se dispone de los siguientes tipos de variables ampliadas: • Variables simples y parámetros • Acceso a variables absolutas • Acceso a bloques de datos • Llamadas a funciones

Sintaxis





Nota

En parámetros formales de salida o de entrada/salida no está permitido el uso de los siguientes parámetros actuales:

- Parámetros de entrada de FC/FB
- Parámetros de entrada/salida que no sean del tipo de datos simple
- Parámetros de entrada/salida de FC y parámetros de salida de FC que no sean del tipo de datos simple.
- Tenga en cuenta las particularidades del tipo de datos ANY y del tipo de datos POINTER.
- La longitud máxima del tipo de datos STRING es ahora de 254 caracteres, pero puede reducirse al número desesado de caracteres.

12.3.3.6 Ejemplo de llamada a una función

Función a llamar

Una función DISTANCIA para calcular la distancia entre dos puntos (X1,Y1) y (X2,Y2) del nivel utilizando coordenadas cartesianas podría tener el siguiente aspecto (En los ejemplos se presupone siempre que en una tabla de símbolos para el FC37 está declarado el símbolo DISTANCIA.):

```
FUNCTION DISTANCIA: REAL // simbólico
VAR_INPUT
        X1: REAL;
        X2: REAL;
        Y1: REAL;
        Y2: REAL;
END_VAR
VAR_OUTPUT
        Q2: REAL;
END_VAR
BEGIN
DISTANCIA:= SQRT( (X2-X1)**2 + (Y2-Y1)**2 );
Q2:= X1+X2+Y1+Y2;
END_FUNCTION
```

Bloque invocante

```
Para utilizar posteriormente un valor de la función dispone de las
siguientes posibilidades:
FUNCTION_BLOCK CALL
VAR
        LONGITUD : REAL ;
        SUMA_DE_VERIFICACION : REAL ;
        RADIO : REAL;
        Y : REAL;
END_VAR
BEGIN
. . .
// Llamada en una asignación de valores:
        LONGITUD := DISTANCIA (X1:=3, Y1:=2, X2:=8.9, Y2:= 7.4,
Q2:=SUMA DE VERIFICACION) ;
//Llamada en una expresión aritmética o lógica, p.ej.
        Y := RADIO + DISTANCIA (X1:=-3, Y1:=2, X2:=8.9, Y2:=7.4,
Q2:=suma de control)
//Utilización en la asignación de parámetros de otro bloque llamado
  FB32.DB32 (ESPACIO:= DISTANCIA (X1:=-3, Y1:=2, X2:=8.9,
  Y2:=7.4), Q2:=suma de verificción)
        . . .
  END_FUNCTION_BLOCK
```

12.3.4 Parámetros definidos implícitamente

12.3.4.1 Parámetro de entrada EN

Todos los bloques de función y todas las funciones tienen el parámetro de entrada EN definido implícitamente. EN es del tipo de datos BOOL y se almacena en el área de los datos locales temporales. Cuando EN es igual a TRUE, el bloque llamado se ejecuta; en caso contrario, no se ejecuta. La especificación del parámetro EN es opcional. Hay que tenr en cuenta que no se puede declarar en el área de declaración de un bloque o de una función.

Como EN es un parámetro de entrada, no se puede modificar dentro del bloque.

Nota

El valor de respuesta de una función no está definido si no se ha llamado a la función (EN : FALSE).

Ejemplo

```
FUNCTION_BLOCK FB57
VAR
    MI_ENABLE: BOOL ;
    resultado : REAL;
END_VAR
// . . .
BEGIN
// . . .
MI_ENABLE:= FALSE ;

// Llamada de una función, en la que se asigna el parámetro EN:
Resultado := FC85 (EN:= MI_ENABLE, PAR_1:= 27) ;
// no se ha ejecutado FC85 porque MI_ENABLE se ha establecido como
FALSE
arriba....

END_FUNCTION_BLOCK
```

12.3.4.2 Parámetro de salida ENO

Todos los bloques de función y todas las funciones poseen el parámetro de salida ENO definido implícitamente. Es un parámetro del tipo de datos BOOL. Se almacena en el área de los datos locales temporales. Al terminar la ejecución de un bloque, el valor actual de la marca OK se deposita en ENO.

Inmediatamente después de llamar a un bloque, puede utilizar el valor de ENO para comprobar si en el bloque se han ejecutado correctamente todas las operaciones, o si se han producido errores.

Ejemplo

```
// Llamada a un bloque de función:
FB30.DB30 ([asignación de parámetros]);

// comprobación de que todo se ha desarrollado correctamente en el
bloque llamado:
IF ENO THEN
// todo correcto
// . . .
ELSE
// se ha producido un error por lo que es necesario un tratamiento
de errores
// . . .
END_IF;
```

13 Contadores y temporizadores

13.1 Contadores

13.1.1 Funciones de contaje

STEP 7 ofrece una serie de funciones estándar de contaje. Los contadores se pueden utilizar en el programa S7-SCL sin necesidad de declararlos previamente. Basta especificar los parámetros necesarios. Funciones de contaje de STEP 7:

Función de contaje	Significado
S_CU	Incrementar contador (Counter Up)
S_CD	Decrementar contador (Counter Down)
S_CUD	Incrementar/decrementar contador (Counter Up Down)

13.1.2 Llamada a funciones de contaje

Las funciones de contaje se llaman igual que las funciones. El identificador de función se puede situar en lugar de un operando en cualquier parte de una expresión mientras el tipo del valor de la función sea compatible con el del operando que lo reemplazará.

El valor de la función (valor de retorno) que se envía a la posición de llamada es el valor de contaje actual (formato BCD) con el tipo de datos WORD.

Llamada absoluta o dinámica

En la llamada se puede introducir un valor absoluto como número de contador, (p. ej., C_NO:=Z10) . No obstante, un valor así ya no se podrá modificar durante el tiempo de ejecución.

En lugar del número absoluto también se puede indicar una variable o una constante del tipo de datos INT. Esto tiene la ventaja, que permite crear una llamada dinámica asignando a la variable o a la constante otro número en cada llamada.

Otra posibilidad de la llamada dinámica es la indicación de una variable del tipo de datos COUNTER.

Ejemplos

```
//Ejemplo de una llamada absoluta:
S_CUD (C_NO:=Z12,
    CD:=E0.0,
    CU:=E0.1,
    S:=E0.2 & E0.3,
    PV:=120,
    R:=FALSE,
    CV:=binVal,
    Q:=actFlag);

//Ejemplo de una llamada dinámica: En cada ejecución de un bucle
//FOR se llamará otro contador:
FUNCTION_BLOCK CONTAD
VAR_INPUT
    Contad: ARRAY [1..4] of STRUCT
                            C_NO: INT;
                            PV  : WORD;
                          END_STRUCT;
.
.
END_VAR
.
.
FOR I:= 1 TO 4 DO
    S_CD(C_NO:=Contad[I].C_NO, S:=true, PV:= Contad[I].PV);
END_FOR;

//Ejemplo de una llamada dinámica utilizando una variable del
//tipo de datos COUNTER:
FUNCTION_BLOCK CONTADOR
VAR_INPUT
    MiContador:COUNTER;
END_VAR
.
.
CurrVal:=S_CD (C_NO:=MiContador,.....);
```

Nota

Los nombres de las funciones y parámetros son idénticos en la nemotécnica alemana y en la inglesa. Sólo el identificador cambia (alemán: Z, inglés: C).

13.1.3 Asignación de parámetros en funciones de contaje

La tabla siguiente muestra una lista de los parámetros disponibles para funciones de contaje:

Parámetro	Tipo de datos	Descripción
C_NO	COUNTER INT	Identificador del contador ; el margen depende de la CPU.
CD	BOOL	Entrada CD: decrementar contador
CU	BOOL	Entrada CU: incrementar contador
S	BOOL	Entrada para preselección del contador
PV	WORD	Valor entre 0 y 999 para ajustar el valor del contador (introducido como 16#<Valor>, estando el valor en formato BCD)
R	BOOL	Entrada de inicialización
Q	BOOL	Salida: Estado del contador
CV	WORD	Salida: estado del contador, binario
RET_VAL	WORD	Resultado en formato BCD

Reglas

Debido a que los valores de los parámetros, p. ej., CD:=E0.0) se han guardado globalmente, su indicación es opcional en algunos casos. Al especificar parámetros deben observarse las siguientes reglas:

- El parámetro para la designación del contador C_NO se debe suministrar en la llamada. En la llamada también es posible indicar en lugar del número absoluto del contador (p. ej., Z12) una variable o una constante con el tipo de datos INT o un parámetro de entrada del tipo de datos COUNTER.
- Se deberá suministrar el parámetro CU (incrementar contador) o el parámetro CD (decrementar contador).
- Pueden omitirse por parejas las especificaciones del parámetro PV (valor de preselección) y S (ajuste).
- El resultado en formato BCD es siempre el valor de la función.

Ejemplo

```
FUNCTION_BLOQUE FB1
VAR
    CurrVal, binVal: word;
    actFlag: bool;
END_VAR

BEGIN
CurrVal    :=S_CD (C_NO:= Z10, CD:=TRUE, S:=TRUE, PV:=100, R:=FALSE,
                   CV:=binVal,Q:=actFlag);
CurrVal    :=S_CU (C_NO:= Z11, CU:=M0.0, S:=M0.1,
                   PV:=16#110, R:=M0.2,
                   CV:=binVal,Q:=actFlag);
CurrVal    :=S_CUD(C_NO:= Z12, CD:=E0.0, CU:=E0.1,
                   S:=E0.2 &E0.3, PV:=120, R:=FALSE,
                   CV:=binVal,Q:=actFlag);
CurrVal    :=S_CD (C_NO:= Z10, CD:=FALSE, S:=FALSE, PV:=100,
                   R:=TRUE,
                   CV:=binVal,Q:=actFlag);
END_FUNCTION_BLOQUE
```

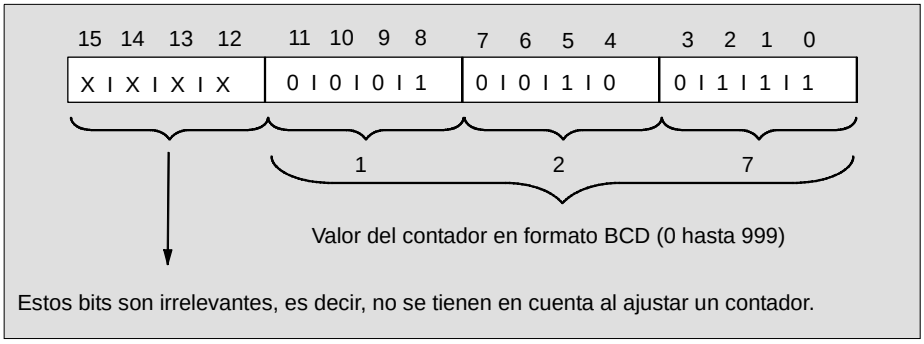
13.1.4 Entrada y evaluación del valor de contaje

Para introducir el valor de inicialización o para evaluar el resultado de la función se requiere la representación interna del valor de contaje. El valor de contaje es del tipo de datos WORD, por lo que los bits 0-11 contienen el valor de contaje en código BCD. Los bits 12-15 se ignoran.

Al ajustar el contador, el valor seleccionado se escribe en el contador. El margen de valores permitido se encuentra entre 0 y 999.El valor de contaje se puede modificar dentro de este margen indicando las operaciones Incrementar/Decrementar contador (S_CUD), Incrementar contador (S_CU) y Decrementar contador (S_CD).

Formato

La siguiente figura explica la configuración binaria del valor de contaje:



Entrada

- decimal, como valor entero: p.ej., 295, siempre que dicho valor corresponda a un formato BCD válido,
- en formato BCD (introducir como constante hexadecimal): p.ej.: 16#127

Evaluación

- como resultado de función (tipo WORD): en formato BCD,
- como parámetro de salida CV (tipo WORD): binario

13.1.5 Incrementar contador (S_CU)

El contador 'Counter Up (S_CU)' sólo ejecuta operaciones de contaje incremental. La tabla ejemplifica el funcionamiento del contador:

Operación	**Funcionamiento**
Incrementar contador	El valor del contador avanza "1" cuando el estado de la señal en la entrada CU cambia de "0" a "1" y el valor del contador es menor que 999.
Preseleccionar contador	Cuando el estado de señal en la entrada **S** cambia de "0" a "1", el contador se ajusta al valor de la entrada **PV**. Este cambio de señal siempre es necesario para ajustar el contador.
Inicializar	El contador se inicializa cuando la entrada **R** = 1. Al inicializar el contador el valor del contador se pone a "0".
Consultar contador	Al consultar el estado de señal en la salida Q se obtiene "1" cuando el valor del contador es mayor que "0". La consulta arroja el resultado "0" cuando el valor del contador es igual "0".

13.1.6 Decrementar contador (S_CD)

El contador 'Counter Down (S_CD)' ejecuta solamente la operación "Decrementar contador". La tabla ejemplifica el funcionamiento del contador:

Operación	**Funcionamiento**
Decrementar contador	El valor del contador decrementa en "1" si el valor de señal en la entrada **CD** cambia de "0" a "1" (flanco ascendente) y el valor del contador es mayor que "0".
Preseleccionar contador	Cuando el estado de señal a la entrada **S** cambia de "0" a "1", el contador se ajusta al valor de la entrada **PV**. Este cambio de la señal siempre es necesario para ajustar el contador.
Inicializar	El contador se inicializa cuando la entrada **R** = 1. Al inicializar el contador el valor del contador se pone a "0".
Consultar contador	La consulta del estado de señal en la salida **Q** da el resultado "1" si el valor de contaje es mayor que "0", La consulta arroja el resultado "0" cuando el valor del contador es igual "0".

13.1.7 Incrementar/decrementar contador (S_CUD)

Con el contador 'Counter Up Down (S_CUD)' se ejecutan las operaciones "Incrementar contador" y "Decrementar contador". Si los impulsos de contaje incremental y decremental son simultáneos, se procesan ambas operaciones. El valor del contador no cambia. La tabla ejemplifica el funcionamiento del contador:

Operación	**Funcionamiento**
Incrementar contador	El valor del contador aumenta "1" cuando el estado de señal en la entrada **CU** cambia de "0" a "1" y el valor del contador es menor que 999.
Decrementar contador	El valor del contador disminuye "1" cuando el estado de señal en la entrada **CD** cambia de "0" a "1" y el valor del contador es mayor que "0".
Preseleccionar contador	Cuando el estado de señal en la entrada **S** cambia de "0" a "1", el contador se ajusta con el valor de la entrada **PV**. Este cambio de la señal es siempre necesario para ajustar el contador.
Inicializar	El contador se inicializa cuando la entrada **R** = 1. Al inicializar el contador el valor del contador se coloca en "0".
Consultar contador	Al consultar el estado de la señal en la salida Q se obtiene "1" cuando el valor del contador es mayor que "0". La consulta arroja el resultado "0" cuando el valor del contador es igual "0".

13.1.8 Ejemplo de funciones de contaje

Asignación de parámetros

La tabla muestra la asignación de parámetros de la función de ejemplo S_CD.

Parámetro	**Descripción**
C_NO	`MICONTADOR`
CD	`ENTRADA E0.0`
S	`AJUSTAR`
PV	`PRESELECCION 16#0089`
R	`INICIALIZAR`
Q	`A0.7`
CV	`VALOR BINARIO`

Ejemplo

```
FUNCTION_BLOQUE CONTAR
VAR_INPUT
    MICONTADOR   : COUNTER ;
END_VAR
VAR_OUTPUT
    RESULTADO       : INT ;
END_VAR
VAR
    SET             : BOOL ;
    RESET  : BOOL ;
    VALOR_BCD       : WORD ; // Valor de contaje en código BCD
    VALOR_BIN       : WORD ; // valor de contaje binario
    PRESELECCION: WORD ;
END_VAR
BEGIN
        A0.0              := 1 ;
        SET               := E0.2 ;
        RESET             := E0.3 ;
        PREDETERMINADO    := 16#0089 ;
//decrementar contador
    BCD_VALOR  := S_CD (C_NO  := MICONTADOR,
        CD:= E0.0 ,
        S  := SET,
        PV:= PRESELECCION,
        R  := RESET,
        CV:= VALOR_BIN ,
        Q  := A0.7) ;
//Continuación del procesamiento como parámetro de salida
        RESULTADO            := WORD_TO_INT (VALOR_BIN) ;
        AW4                  := VALOR_BCD_ ;
END_FUNCTION_BLOCK
```

13.2 Temporizadores

13.2.1 Funciones de temporización

Los temporizadores son elementos del programa con carácter de función que ejecutan y vigilan procesos controlados por tiempo. STEP 7 dispone de una serie de funciones de temporización estándar, a las que se puede acceder con S7-SCL:

Función de temporización	Significado
S_PULSE	Arrancar temporizador como impulso
S_PEXT	Arrancar temporizador como impulso prolongado
S_ODT	Arrancar temporizador como retardo a la conexión
S_ODTS	Arrancar temporizador como retardo a la conexión con memoria
S_OFFDT	Arrancar temporizador como retardo a la desconexión

13.2.2 Llamada a funciones de temporización

Las funciones de temporización se llaman igual que las funciones. El nombre de la función puede utilizarse en cualquier punto del programa en sustitución de un operando dentro de una expresión, siempre que el tipo del resultado de la función sea compatible con el del operando sustituido.

El valor de la función (valor de retorno) que se envía al punto de llamada es un valor de temporización del tipo de datos S5TIME.

Llamada absoluta o dinámica

En la llamada se puede introducir un valor absoluto (p. ej., T_NO:=T10) del tipo de datos TEMPORIZADOR como número de la función de temporización. Un valor de este tipo ya no se podrá modificar durante el tiempo de ejecución.

En lugar del número absoluto también se puede indicar una variable o una constante del tipo de datos INT. Esto tiene la ventaja que permite diseñar la llamada de forma dinámica asignando otro número a la variable o a la constante en cada llamada.

Otra manera de realizar una llamada dinámica es la indicación de una variable del tipo de datos TIMER.

Ejemplos

```
//Ejemplo de una llamada absoluta:
CurrTime:=S_ODT (T_NO:=T10,
                 S:=TRUE,
                 TV:=T#1s,
                 R:=FALSE,
                 BI:=biVal,
                 Q:=actFlag);

//Ejemplo de una llamada absoluta: En cada ejecución de un
//bucle FOR se llamará otra función de temporizador:
FUNCTION_BLOCK TEMPORIZADOR
VAR_INPUT
  MI_TEMPORIZADOR: ARRAY [1..4] of STRUCT
                              T_NO: INT;
                              TV  : WORD;
                            END_STRUCT;
.
.
END_VAR
.
.
FOR I:= 1 TO 4 DO
CurrTime:=  S_ODT(T_NO:=MI_TEMPORIZADOR[I].T_NO, S:=true,
TV:= MI_TEMPORIZADOR[I].TV);
END_FOR;

//Ejemplo de una llamada dinámica utilizando una variable del
//tipo de datos TIMER:
FUNCTION_BLOCK TEMPORIZADOR
VAR_INPUT
  mitemporizador:TIMER;
END_VAR
.
.
CurrTime:=S_ODT (T_NO:=mitemporizador,.....);
```

Nota

Los nombres de las funciones son idénticos en la nemotécnica alemana y en la inglesa.

13.2.3 Asignación de parámetros en funciones de temporización

La tabla siguiente muestra un resumen de los parámetros para funciones de temporización:

Parámetro	Tipo de datos	Descripción
T_NO	TIMER ENTERO	Identificador del temporizador; el margen depende de la CPU
S	BOOL	Entrada de arranque
TV	S5TIME	Preselección del valor de temporización (formato BCD)
R	BOOL	Entrada de inicialización
Q	BOOL	Estado del temporizador
BI	WORD	Valor residual de temporización (binario)
RET_VAL	S5TIME	Valor de temporizador

Reglas

Debido a que los valores de los parámetros se memorizan globalmente, su indicación es en algunos casos opcional. Al especificar parámetros deben observarse las siguientes reglas generales:

- El parámetro para la designación del temporizador T_NO se debe suministrar en la llamada. En lugar del número absoluto del temporizador (p. ej., T10) también puede indicar en la llamada una variable del tipo de datos INT o un parámetro de entrada del tipo de datos TIMER.
- Pueden omitirse por parejas las especificaciones del parámetro PV (valor predefinido) y S (ajuste).
- El resultado en el formato S5TIME es siempre un valor de función.

Ejemplo

```
FUNCTION_BLOCK FB2
VAR
    CurrTime     : S5time;
    BiVal        : word;
    ActFlag      : bool;
END_VAR

BEGIN
CurrTime   :=S_ODT   (T_NO:= T10, S:=TRUE, TV:=T#1s, R:=FALSE,
           BI:=biVal,Q:=actFlag);
CurrTime   :=S_ODTS (T_NO:= T11, S:=M0.0, TV:= T#1s, R:=M0.1,
           BI:=biVal,Q:=actFlag);
CurrTime   :=S_OFFDT(T_NO:= T12, S:=E0.1 & actFlag, TV:= T#1s,
           R:=FALSE, BI:=biVal,Q:=actFlag);
CurrTime   :=S_PEXT (T_NO:= T13, S:=TRUE, TV:= T#1s, R:=E0.0,
           BI:=biVal,Q:=actFlag);
CurrTime   :=S_PULSE(T_NO:= T14, S:=TRUE, TV:= T#1s, R:=FALSE,
           BI:=biVal,Q:=actFlag);
END_FUNCTION_BLOCK
```

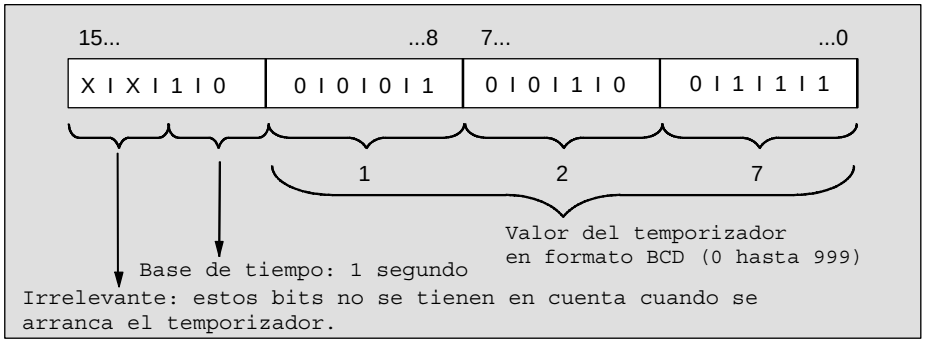
13.2.4 Entrada y evaluación del valor de temporización

Para introducir el valor de inicialización o para evaluar el resultado de la función en código BCD se requiere la representación interna del valor de temporización. El valor de temporización es del tipo de datos WORD; los bits 0-11 contienen el valor de temporización en formato BCD, y los bits 12 y 13 la base de tiempo. Los bits 14 y 15 se ignoran.

Al actualizar el temporizador el valor de temporización decrementa en una unidad, a intervalos que vienen definidos por la base de tiempo. El valor de temporización se reduce hasta que se iguala a "0". El margen de tiempo abarca de 0 a 9990 segundos.

Formato

La siguiente figura explica la configuración de bit del valor de tiempo:



Entrada

Es valor de temporización preseleccionado se puede cargar con los siguientes formatos:

- en representación escalonada
- en representación decimal

En ambos casos la base de tiempo se selecciona automáticamente, y el valor del número inmediatamente inferior se redondea con esta base de tiempo.

Evaluación

El resultado se puede evaluar en dos formatos diferentes:

- como resultado de función (Tipo S5TIME): en formato BCD,
- como parámetro de salida (valor de temporización sin base de tiempo del tipo WORD): binario

Base de tiempo para valores de temporización

Para introducir y evaluar el valor de temporización se requiere la base de tiempo (bits 12 y 13 de la palabra del temporizador). La base de tiempo define el intervalo en el que el valor de temporización se decrementa una unidad (ver tabla). La base de tiempo más pequeña es 10 ms; la mayor, 10 s.

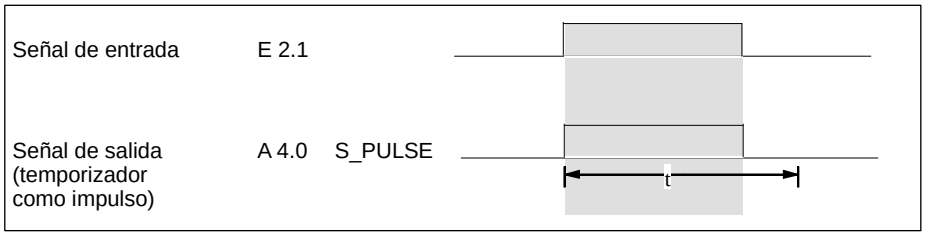
Base de tiempo	Código binario para base de tiempo
10 ms	00
100 ms	01
1 s	10
10 s	11

Nota

Como los valores de temporización sólo se guardan en un intervalo de tiempo, los valores que no son un múltiplo exacto del intervalo de tiempo se truncan. Los valores cuya resolución es demasiado grande para el margen deseado se redondean, de forma que se alcanza el margen deseado pero no la resolución deseada.

13.2.5 Arrancar temporizador como impulso (S_PULSE)

El tiempo máximo en el cual la señal de salida permanece a "1" es idéntico al valor de temporización programado. Si durante el tiempo de ejecución del temporizador aparece en la entrada el estado de señal 0, el temporizador se pondrá a "0". (es decir, el temporizador se detiene).



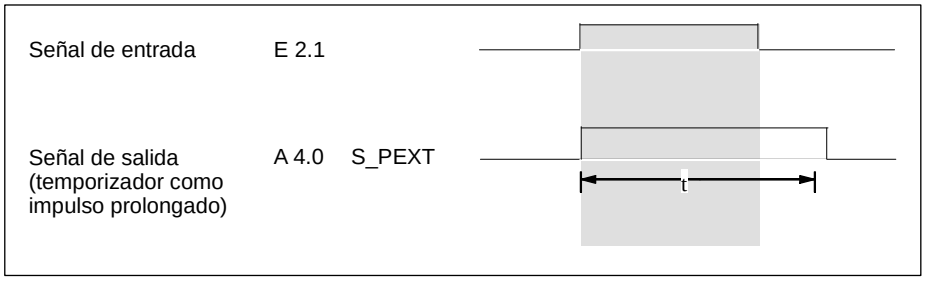
Funcionamiento

La tabla muestra el funcionamiento de la operación "arrancar el temporizador como impulso":

Operación	**Funcionamiento**
Arrancar temporizador	La operación "arrancar temporizador como impulso" arranca un temporizador determinado cuando el estado de señal en la salida de arranque (**S**) cambia de "0" a "1". Para habilitar el temporizador siempre se necesita un cambio de señal.
Definir tiempo de funcionamiento	El temporizador empieza a contar a partir del valor indicado en la entrada **TV** hasta que se termina el tiempo programado y la entrada **S** = 1.
Concluir antes del tiempo programado	Si la entrada **S** cambia de "1" a "0" antes de que se agote el valor de temporización, el temporizador se detiene.
Inicializar	El temporizador se inicializa cuando la entrada de inicialización (**R**) cambia de "0" a "1" mientras funciona el temporizador. Este cambio también pone a "0" el valor de temporización y la base de tiempo. El estado de señal "1" en la entrada **R** no tiene relevancia si no funciona el temporizador.
Consultar estado de señal	Mientras el temporizador esté funcionando, el estado de señal en la salida **Q** dará el resultado "1". Si el tiempo de funcionamiento termina anticipadamente, al consultar el estado de señal en la salida **Q** se obtendrá el resultado "0".
Consultar valor de temporización actual	El valor de temporización actual puede consultarse en la salida BI y mediante el valor de la función S_PULSE.

13.2.6 Arrancar temporizador como impulso prolongado (S_PEXT)

La señal de salida permanece a "1" durante el tiempo programado (t), independientemente del tiempo que permanezca a "1" la señal de entrada. Si se activa de nuevo el impulso de arranque se ejecutará de nuevo la temporización, de forma que el impulso de salida se prolonga temporalmente (redisparo).



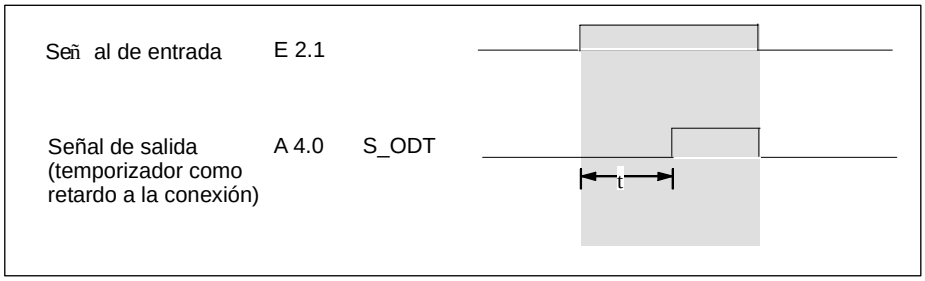
Funcionamiento

La tabla muestra el funcionamiento de la operación "arrancar temporizador como impulso prolongado".

Operación	Funcionamiento
Arrancar temporizador	La operación "arrancar temporizador como impulso prolongado" (S_PEXT) arranca un temporizador determinado si el estado de señal en la salida de arranque (**S**) cambia de "0" a "1". Para habilitar el temporizador siempre se necesita un cambio de señal.
Rearranque del tiempo de funcionamiento	Si el estado de señal en la entrada **S** cambia de nuevo a "1" durante el tiempo de funcionamiento, el temporizador arranca de nuevo con el valor de temporización indicado.
Preselección del tiempo de funcionamiento	El temporizador continúa funcionando con el valor indicado en la entrada **TV** hasta que se agota el tiempo programado.
Inicializar	El temporizador se inicializa cuando la entrada de inicialización (R) cambia de "0" a "1"con el temporizador en funcionamiento. Este cambio también pone a "0" el valor de temporización y la base de tiempo. El estado de señal "1" en la entrada **R** no tiene relevancia si no funciona el temporizador.
Consultar estado de señal	Mientras el temporizador está en funcionamiento el estado de señal en la salida **Q** siempre arrojará el resultado "1", independientemente de la longitud de la señal de entrada.
Consultar valor de temporización actual	El valor de temporización actual puede consultarse en la salida **BI** o mediante el valor de la función S_PEXT.

13.2.7 Arrancar temporizador como retardo a la conexión (S_ODT)

La señal de salida sólo cambia de "0" a "1" si ha transcurrido el tiempo programado y la señal de entrada continúa a "1". Es decir, la salida se conecta con retardo. Las señales de entrada cuyo intervalo es menor que el tiempo programado no aparecen en la salida.



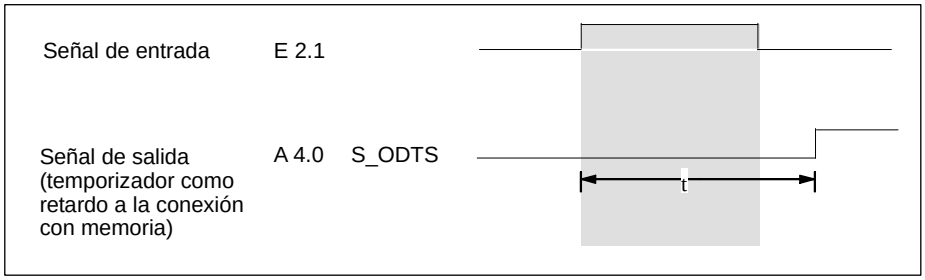
Funcionamiento

La tabla muestra el funcionamiento de la operación "arrancar temporizador como retardo a la conexión":

Operación	**Funcionamiento**
Arrancar temporizador	La operación "Arrancar temporizador como retardo a la entrada" arranca un temporizador determinado si el estado de señal en la entrada de arranque (**S**) cambia de "0" a "1". Para habilitar el temporizador siempre se necesita un cambio de señal.
Conservar temporizador	Si el estado de señal en la entrada **S** cambia de "1" a "0" mientras está en funcionamiento el temporizador, éste se detiene.
Preseleccionar el tiempo de funcionamiento	El temporizador continúa funcionando con el valor indicado en la entrada **TV** mientras el estado de señal en la entrada S = 1.
Inicializar	El temporizador se inicializa cuando la entrada de inicialización (**R**) cambia de "0" a "1" mientras funciona el temporizador. Este cambio también pone a "0" el valor de temporización y la base de tiempo. El temporizador también se inicializa si **R** = 1 mientras no funciona el temporizador.
Consultar estado de señal	Al consultar el estado de señal en la salida **Q** siempre se obtiene "1"si el tiempo ha transcurrido sin errores y la entrada **S** continúa siendo "1". Si el temporizador se ha detenido, el resultado de consultar el estado de señal siempre será "0". Al consultar el estado de señal en la salida **Q** también se obtendrá "0" si el temporizador no está en funcionamiento y el estado de señal en la entrada **S** continúa siendo "1".
Consultar valor de temporización actual	Puede consultarse el valor de temporización actual en la salida **BI** y mediante el valor de la función S_ODT.

13.2.8 Arrancar temporizador como retardo a la conexión con memoria (S_ODTS)

La señal de salida sólo cambia de "0" a "1" cuando ha transcurrido el tiempo programado, independientemente del tiempo que permanezca a "1" la señal de entrada.



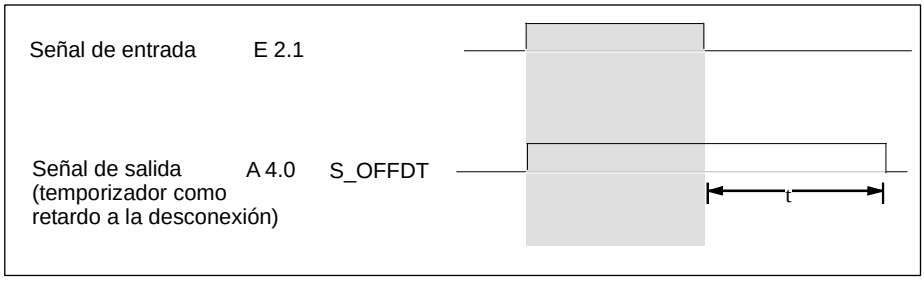
Funcionamiento

La tabla muestra el funcionamiento de la operación "arrancar temporizador con retardo a la conexión con memoria".

Operación	Funcionamiento
Arrancar temporizador	La operación "Arrancar temporizador con retardo a la conexión con memoria" arranca un temporizador determinado cuando el estado de señal en la entrada de arranque (**S**) cambia de "0" a "1". Para habilitar el temporizador siempre se necesita un cambio de señal.
Rearrancar temporizador	El temporizador rearranca con el valor especificado cuando la entrada **S** cambia de "0" a "1"mientras el temporizador está en funcionamiento.
Preseleccionar el tiempo de funcionamiento	El temporizador continúa funcionando con el valor indicado en la entrada **TV** aunque el estado de señal en la entrada S cambie a " 0" antes de haberse agotado el tiempo.
Inicializar	Si cambia la entrada de puesta a cero (**R**) de "0" a "1" se reinicializará el temporizador independientemente del estado de la señal en la entrada **S** .
Consultar estado de señal	Una consulta del estado de señal en la salida **Q** arrojará el resultado "1" después de haberse agotado el tiempo, independientemente del estado de señal en la entrada **S**.
Consultar valor de temporización actual	El valor actual de temporización puede consultarse en la salida **BI** y mediante el valor de la función S_ODTS.

13.2.9 Arrancar temporizador como retardo a la desconexión (S_OFFDT)

Cuando se produce un cambio del estado de señal de "0" a "1" en la entrada de arranque S, en la salida Q aparece el estado "1". Cuando el estado de la entrada de arranque pasa de "1" a "0", arranca el temporizador. La salida sólo cambia al estado de señal "0" después de transcurrido el intervalo. De esta forma la salida se desconecta con retardo.



Funcionamiento

La tabla muestra el funcionamiento de la operación "arrancar temporizador como retardo a la desconexión":

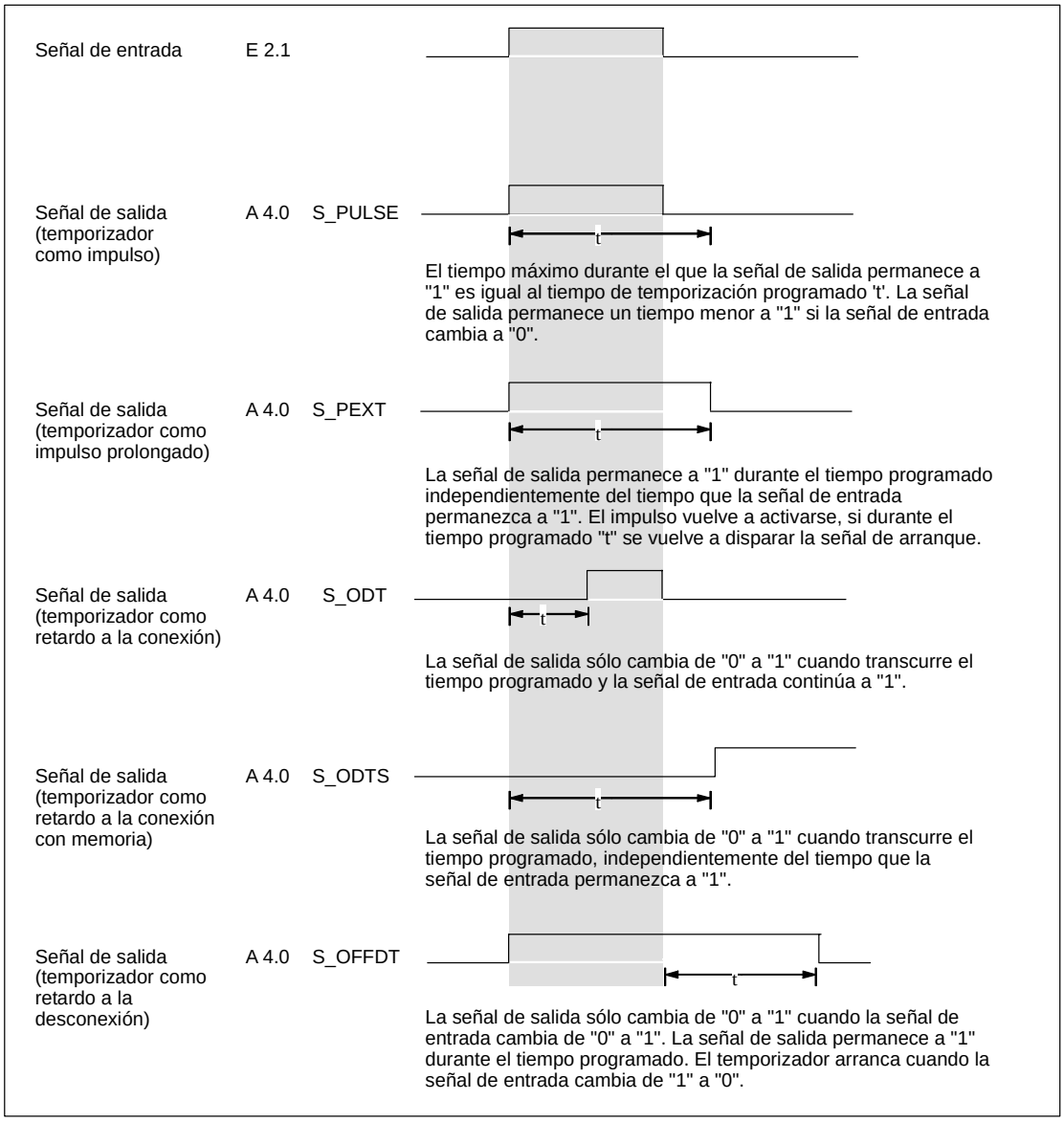
Operación	**Funcionamiento**
Arrancar temporizador	La operación "Arrancar temporizador como retardo a la desconexión" arranca el temporizador determinado cuando el estado de señal en la entrada de arranque (**S**) cambia de "1" a "0". Para desbloquear el temporizador siempre se necesita un cambio de señal.
Rearrancar temporizador	El temporizador se arrancará de nuevo si el estado de señal en la entrada **S** cambia de nuevo de "1" a "0" (p. ej., después de reinicializar).
Definir tiempo de funcionamiento	El temporizador funciona con el valor indicado en la entrada **TV**.
Inicializar	Si la entrada de inicialización (**R**) cambia de "0" a "1"mientras el temporizador está funcionando, el temporizador se inicializa.
Consultar estado de señal	Una consulta del estado de señal en la salida **Q** dará como resultado "1" si el estado de señal en la entrada **S** = 1 o si el temporizador está en funcionamiento.
Consultar valor de temporización actual	El valor de temporización actual puede consultarse en la salida **BI** y mediante el valor de la función S_OFFDT.

13.2.10 Ejemplo de funciones de temporización

```
FUNCTION_BLOCK RELOJ
VAR_INPUT
    mitemporizador      : TIMER ;
END_VAR
VAR_OUTPUT
    resultado           : S5TIME ;
END_VAR
VAR
    introducir          : BOOL ;
    poner a cero        : BOOL ;
    Valor_bcd           : S5TIME ;//Base de tiempo
                                  //y valor restante en código BCD
    Valor_binario       : WORD ;  //Valor de temporización binario
    Predeterminado      : S5TIME ;
END_VAR
BEGIN
    A0.0                := 1;
    introducir          := E0.0 ;
    poner a cero        := E0.1;
    Predeterminado      := T#25S ;
    Valor_bcd           := S_PEXT (T_NO  := mitemporizador ,
                        S    := introducir,
                        TV   := predeterminado,
                        R    := poner a cero,
                        BI   := valor binario ,
                        Q    := A0.7) ;
//Continuar procesando como parámetro de salida
    resultado:= Valor_bcd ;
//A salida para visualización
    AW4 := valor_binario ;
END_FUNCTION_BLOQUE
```

13.2.11 Selección del temporizador correcto

La siguiente figura ofrece un resumen de los cinco temporizadores distintos que se han descrito en este apartado. Este resumen pretende ayudarle a elegir el temporizador más adecuado para sus exigencias.



14 Funciones estándar de S7-SCL

14.1 Funciones de conversión del tipo de datos

14.1.1 Conversión del tipo de datos

Al combinar dos operandos en una operación hay que tener en cuenta la compatibilidad de sus tipos de datos. Si los operandos no son del mismo tipo de datos se debe realizar una conversión. S7-SCL conoce los siguientes tipos de conversión del tipo de datos:

- Conversión implícita de tipos de datos

  Los tipos de datos se dividen en clases. Dentro de las clases, S7-SCL realiza una conversión implícita del tipo de datos. Las funciones que utiliza el compilador se encuentran resumidas en "Funciones de conversión clase A".

- Conversión explícita de tipo de datos

  En operandos que no pertenecen a la misma clase es preciso llamar una función de conversión. S7-SCL pone a disposición numerosas funciones estándar para la conversión explícita del tipo de datos que se pueden dividir en las siguientes clases:

  - Funciones de conversión de clase B
  - Funciones de redondeo y truncado

14.1.2 Conversión implícita del tipo de datos

El compilador realiza una conversión implícita del tipo de datos dentro de las clases de tipos de datos descritas en la tabla, en el orden indicado. El formato común de dos operandos siempre es el tipo de datos mayor de ambos. Así por ejemplo, el formato común de BYTE y WORD - WORD.

Tenga en cuenta que en una conversión del tipo de datos dentro de la clase ANY_BIT los bits de referencia se ajustarán a cero.

Clases	Secuencia de conversión
ANY_BIT	BOOL > BYTE > WORD > DWORD
ANY_NUM	INT > DINT > REAL

Ejemplo de conversión implícita del tipo de datos

```
VAR
    PID_REGULADOR_1 : BYTE ;
    PID_REGULADOR_2 : WORD ;
END_VAR
BEGIN
IF (PID_REGULADOR_1 <> PID_REGULADOR_2) THEN ...
(* En la instrucción IF superior se convierte PID_REGULADOR_1
implícitamente de BYTE a WORD. *)
```

14.1.2.1 Funciones de conversión de clase A

La tabla representa las funciones de conversión del tipo de datos de clase A. El compilador envía estas funciones implícitamente, pero también se pueden indicar explícitamente. El resultado siempre está definido.

Nombre de la función	Regla de conversión
BOOL_TO_BYTE	Completar con ceros a la izquierda.
BOOL_TO_DWORD	Completar con ceros a la izquierda.
BOOL_TO_WORD	Completar con ceros a la izquierda.
BYTE_TO_DWORD	Completar con ceros a la izquierda.
BYTE_TO_WORD	Completar con ceros a la izquierda.
CHAR_TO_STRING	Convertir en una cadena (de longitud 1) que contiene el mismo carácter.
DINT_TO_REAL	Convertir en REAL según la norma IEEE. El valor puede cambiar debido a la diferente precisión en REAL.
INT_TO_DINT	La palabra de mayor rango del valor de la función se completa con 16#FFFF en caso de tratarse de un parámetro de entrada negativo, y con ceros en caso contrario. El valor no varía.
INT_TO_REAL	Convertir en REAL según la norma IEEE. El valor no varía.
WORD_TO_DWORD	Completar con ceros a la izquierda.

14.1.3 Funciones estándar para la conversión explícita del tipo de datos

La llamada de funciones se describe en el apartado "Llamada de funciones".

Al llamar a funciones de conversión hay que tener en cuenta lo siguiente:

- Parámetros de entrada:
  Cada función de conversión del tipo de datos tiene un sólo parámetro de entrada, cuyo nombre es IN. Debido a que se trata de una función con un solo parámetro, no es preciso que se indique.
- Valor de la función
  El resultado es siempre el valor de la función.
- Especificación del nombre
  Como los tipos de datos del parámetro de entrada y del valor de la función se derivan de sus nombres respectivos, no se indican por separado en las sinopsis (clase A y clase B): por ejemplo, en la función BOOL_TO_BYTE el tipo de datos del parámetro de entrada es BOOL, y el tipo de datos del valor de la función, BYTE.

14.1.3.1 Funciones de conversión de clase B

La tabla muestra las funciones de conversión del tipo de datos de la clase B. Estas funciones se deben especificar explícitamente. El resultado también puede ser indeterminado, si el tamaño del tipo de datos de destino es insuficiente.

Esto se puede comprobar programando una comprobación de límites o haciendo que el sistema los compruebe, para lo cual debe seleccionar la opción "OK flag" antes de efectuar la compilación. En los casos, en los que no se haya definido el resultado, el sistema ajustará OK-Flag a FALSE.

Nombre de la función	Regla de conversión	OK
BOOL_TO_INT	WORD_TO_INT(BOOL_TO_WORD(x))	N
BOOL_TO_DINT	DWORD_TO_DINT(BOOL_TO_DWORD(x))	N
BYTE_TO_BOOL	Copiar el bit menos significativo	S
BYTE_TO_CHAR	Aceptar la cadena de bits.	N
BYTE_TO_INT	WORD_TO_INT(BYTE_TO_WORD(x))	N
BYTE_TO_DINT	DWORD_TO_DINT(BYTE_TO_DWORD(x))	N
CHAR_TO_BYTE	Aceptar la cadena de bits.	N
CHAR_TO_INT	La cadena de bits que se encuentra en el parámetro de entrada se introducirá en el byte menos significativo del valor de la función. El byte de mayor valor se completa con ceros.	N
DATE_TO_DINT	Aceptar la cadena de bits.	N
DINT_TO_DATE	Aceptar la cadena de bits.	S
DINT_TO_DWORD	Aceptar la cadena de bits.	N
DINT_TO_INT	Copiar el bit para el signo. el dato existente en el parámetro de entrada se interpreta como tipo de dato s INT. Si el valor es menor que -32_768 o mayor que 32_767, la variable OK cambia a FALSE.	S
DINT_TO_TIME	Aceptar la cadena de bits.	N
DINT_TO_TOD	Aceptar la cadena de bits.	S

Nombre de la función	Regla de conversión	OK
DINT_TO_BOOL	DWORD_TO_BOOL(DINT_TO_DWORD(x))	S
DINT_TO_BYTE	DWORD_TO_BYTE(DINT_TO_DWORD(x))	S
DINT_TO_STRING	DI_STRNG	N
DINT_TO_WORD	DWORD_TO_WORD(DINT_TO_DWORD(x))	S
DWORD_TO_BOOL	Copiar los bits menos significativos	S
DWORD_TO_BYTE	Copiar los 8 bits menos significativos	S
DWORD_TO_DINT	Aceptar la cadena de bits	N
DWORD_TO_REAL	Aceptar la cadena de bits.	N
DWORD_TO_WORD	Copiar los 16 bits menos significativos	S
DWORD_TO_INT	DINT_TO_INT (DWORD_TO_ DINT(x))	S
INT_TO_CHAR	Aceptar la cadena de bits.	S
INT_TO_WORD	Aceptar la cadena de bits.	N
INT_TO_BOOL	WORD_TO_BOOL(INT_TO_WORD(x))	J
INT_TO_BYTE	WORD_TO_BYTE(INT_TO_WORD(x))	S
INT_TO_DWORD	WORD_TO_DWORD(INT_TO_WORD(x))	N
INT_TO_STRING	I_STRNG(x)	N
REAL_TO_DINT	Redondear el valor IEEE-REAL hasta DINT. Si el valor es menor que -2_147_483_648 o mayor que 2_147_483_647, la variable OK cambia a FALSE.	S
REAL_TO_DWORD	Aceptar la cadena de bits.	N
REAL_TO_INT	Redondear el valor IEEE-REAL hasta INT. Si el valor es menor que -32_768 o mayor que 32_767, la variable OK cambia a FALSE.	S
REAL_TO_STRING	R_STRNG(x)	N
STRING_TO_CHAR	Copiar el primer carácter de la cadena. Si STRING no tiene longitud 1, la variable OK cambia a FALSE.	S
STRING_TO_INT	STRNG_I(x)	N
STRING_TO_DINT	STRNG_DI(x)	N
STRING_TO_REAL	STRNG_R(x)	N
TIME_TO_DINT	Aceptar la cadena de bits.	N
TOD_TO_DINT	Aceptar la cadena de bits.	N
WORD_TO_BOOL	Copiar el bit menos significativo	S
WORD_TO_BYTE	Copiar los 8 bits menos significativos	S
WORD_TO_INT	Aceptar la cadena de bits.	N
WORD_TO_DINT	INT_TO_DINT(WORD_TO_INT(x))	N
WORD_TO_BLOCK_DB	La configuración binaria de WORD se interpreta como número del bloque de datos.	N
BLOCK_DB_TO_WORD	El número del bloque de datos se interpreta como configuración binaria de WORD.	N
BCD_TO_INT(x) WORD_BCD_TO_INT(x)	La expresión x es del tipo WORD y se acepta como valor codificado en BCD entre -999 y +999. El resultado que se obtiene después de la conversión es un número entero (en formato binario) del tipo INT. Si se produce un error durante la conversión, el sistema de automatización cambia a STOP. La causa del error se puede evaluar en el OB121..	N

Nombre de la función	Regla de conversión	OK
INT_TO_BCD(x) INT_TO_BCD_WORD(x)	La expresión x es del tipo INT y se acepta como entero con un valor comprendido entre -999 y +999. El resultado que se obtiene después de la conversión es un número codificado en BCD del tipo WORD. Fuera del rango de valores el resultado está sin definir. En caso de haber seleccionado la opción "Activar OK flag" el OK flag adoptará el valor false.	S
BCD_TO_DINT(x) DWORD_BCD_TO_DINT(x)	La expresión x es del tipo DWORD y se acepta como valor codificado en BCD entre -9999999 y +9999999. El resultado que se obtiene después de la conversión es un número entero (formato binario) del tipo DINT. Si se produce un error durante la conversión, el sistema de automatización cambia a STOP. La causa del error puede evaluarse en el OB121.	N
DINT_TO_BCD(x) DINT_TO_BCD_DWORD(x)	La expresión x es del tipo DINT y se acepta como entero con un valor comprendido entre –9999999 y +9999999. El resultado que se obtiene después de la conversión es un número codificado en BCD del tipo DWORD. Fuera del rango de valores el resultado está sin definir. En caso de haber seleccionado la opción "Activar OK flag" el OK flag adoptará el valor false.	S

Atención

En caso de convertir una constante de un tipo de datos más significativo en un tipo de datos menos significativo, al compilar aparecerá un mensaje de error, si la constante se encuentra fuera del rango del tipo de datos menos significativo.

Ejemplos:

```
M0.0 :=WORD_TO_BOOL(W#16#FFFF);
MW0  :=DINT_TO_INT(35000);
```

Nota

También es posible utilizar otras funciones IEC para la conversión del tipo de datos. Para más información sobre las funciones disponibles, consulte el manual de referencia de STEP 7 "Funciones de sistema y funciones estándar para S7-300/400".

14.1.3.2 Funciones de redondeo y truncado

Algunas de las posibles conversiones de los tipos de datos son las funciones de redondeo y truncado de números. La tabla muestra los nombres, tipos de datos (para el parámetro de entrada y para el valor de función) y las tareas de dichas funciones:

Nombre de la función	Tipo de datos Parámetro de entrada	Tipo de datos Valor de la función	Tarea
ROUND	REAL	DINT	Redondear (Crear un número DINT). Según DIN EN 61131-3 se redondea siempre hasta el siguiente valor par entero, es decir, 1.5 se redondea a 2; 2.5 también se redondea a 2.
TRUNC	REAL	DINT	Truncado (forman un número DINT)

Nota

También es posible utilizar otras funciones IEC para la conversión del tipo de datos. Para más información sobre las funciones disponibles, consulte el manual de referencia de STEP 7 "Funciones de sistema y funciones estándar para S7-300/400".

Ejemplo

```
// Se redondea por defecto (Resultado: 3)
        ROUND (3.14) ;

// Se redondea por exceso (Resultado: 4)
        ROUND (3.56) ;

// Se trunca (Resultado: 3)
        TRUNC (3.14) ;

// Se trunca (Resultado: 3)
        TRUNC (3.56) ;
```

14.1.3.3 Ejemplos de conversión con funciones estándar

En el siguiente ejemplo se requiere una conversión explícita, dado que el tipo de datos de destino es menos poderoso que el tipo de datos de la fuente.

```
FUNCTION_BLOCK FB10
VAR
    INTERR  : INT;
    REGULAD : DINT;
END_VAR

(* INT es menos poderoso que DINT *)
INTERRUPTOR:= DINT_TO_INT (REGULADOR) ;
// . . .
END_FUNCTION_BLOCK
```

En el siguiente ejemplo se requiere una conversión explícita del tipo de datos dado que el tipo de datos REAL no está permitido para una expresión aritmética con la operación MOD:

```
FUNCTION_BLOCK FB20
VAR
    INTERRUPTOR  : REAL
    VALORINT     : INT := 17;
    CONV2  : INT ;
END_VAR

(* MOD sólo se debe aplicar a datos del tipo INT o DINT *)
CONV2 := VALORINT MOD REAL_TO_INT (INTERRUPTOR);
// . . .
END_FUNCTION_BLOCK
```

En el siguiente ejemplo se requiere una conversión dado que no existe el tipo de datos correcto para una operación lógica. La operación NOT sólo se debe aplicar a datos del tipo BOOL, BYTE, WORD o DWORD.

```
FUNCTION_BLOCK FB30
VAR
    VALORINT : INT := 17;
    CONV1     : WORD ;
END_VAR

(* NOT no se debe aplicar a datos del tipo INT *)
KONV1 := NOT INT_TO_WORD(VALORINT);
// . . .
END_FUNCTION_BLOCK
```

El siguiente ejemplo muestra la conversión en entradas y salidas de la periferia:

```
FUNCTION_BLOCK FB40
VAR
    Radio_Con    : WORD ;
    Radio            : INT;
END_VAR

    Radio_con    := %EB0;
    Radio            := WORD_TO_INT (radio_con);
(* Conversión al cambiar a otra clase de tipo. El valor procede de
la entrada y se convierte para continuar su cálculo.*)

    Radio    := Radio (superficie:= datos del círculo.superficie)
    %AB0        :=WORD_TO_BYTE (INT_TO_WORD(RADIO));
(*El radio se calcula deduciendo de la superficie y se encuentra en
enteros. Para su salida se convierte el valor en primer lugar en
otra clase de tipo (INT_TO_WORD) y, a continuación, en un tipo menos
poderoso (WORD_TO_BYTE).*)
// . . .
END_FUNCTION_BLOCK
```

14.2 Funciones estándar numéricas

14.2.1 Funciones estándar aritméticas básicas

Estas funciones permiten calcular el valor absoluto, el cuadrado o la raíz cuadrada de una magnitud.

El tipo de datos ANY_NUM representa INT, DINT o REAL. Tenga en cuenta que los parámetros de entrada del tipo INT o DINT se convertirán internamente en variables REAL si el valor de función es del tipo REAL.

Nombre de la función	Tipo de datos Parámetro de entrada	Tipo de datos Valor de la función	Descripción
ABS	ANY_NUM	ANY_NUM	Valor absoluto
SQR	ANY_NUM	REAL	Cuadrado
SQRT	ANY_NUM	REAL	Raíz

Nota

También es posible utilizar otras funciones IEC para la conversión del tipo de datos. Para más información sobre las funciones disponibles, consulte el manual de referencia de STEP 7 "Funciones de sistema y funciones estándar para S7-300/400".

14.2.2 Funciones logarítmicas

Se trata de funciones que permiten calcular un valor exponencial o un logaritmo de una magnitud.

El tipo de datos ANY_NUM representa INT, DINT o REAL. Tenga en cuenta, que los parámetros de entrada del tipo ANY_NUM se transforman internamente en variables REAL.

Nombre de la función	Tipo de datos Parámetro de entrada	Tipo de datos Valor de la función	Descripción
EXP	ANY_NUM	REAL	e elevado a IN
EXPD	ANY_NUM	REAL	10 elevado a IN
LN	ANY_NUM	REAL	logaritmo natural
LOG	ANY_NUM	REAL	logaritmo decimal

Nota

También es posible utilizar otras funciones IEC para la conversión del tipo de datos. Para más información sobre las funciones disponibles, consulte el manual de referencia de STEP 7 "Funciones de sistema y funciones estándar para S7-300/400".

14.2.3 Funciones trigonométricas

Las funciones trigonométricas representadas en la tabla sirven para estimar y calcular magnitudes de ángulos y arcos.

El tipo de datos ANY_NUM representa INT, DINT o REAL. Tenga en cuenta que los parámetros de entrada del tipo ANY_NUM se convierten internamente en variables REAL.

Nombre de la función	Tipo de datos Parámetro de entrada	Tipo de datos Valor de la función	Descripción
ACOS	ANY_NUM	REAL	Arco coseno
ASIN	ANY_NUM	REAL	Arco seno
ATAN	ANY_NUM	REAL	Arco tangente
COS	ANY_NUM	REAL	Coseno
SIN	ANY_NUM	REAL	Seno
TAN	ANY_NUM	REAL	Tangente

Nota

También es posible utilizar otras funciones IEC para la conversión del tipo de datos. Para más información sobre las funciones disponibles, consulte el manual de referencia de STEP 7 "Funciones de sistema y funciones estándar para S7-300/400".

14.2.4 Ejemplos de funciones estándar numéricas

Llamada	RESULTADO
RESULTADO := ABS (-5) ;	// 5
RESULTADO := SQRT (81.0) ;	// 9
RESULTADO := SQR (23) ;	// 529
RESULTADO := EXP (4.1) ;	// 60.340 ...
RESULTADO := EXPD (3) ;	// 1_000
RESULTADO := LN (2.718 281) ;	// 1
RESULTADO := LOG (245) ;	// 2.389_166 ...
PI := 3.141592; RESULTADO := SIN (PI / 6) ;	// 0.5
RESULTADO := ACOS (0.5) ;	// 1.047_197 (=PI / 3)

14.3 Funciones estándar de cadena de bits

Cada función estándar de cadena de bits dispone de dos parámetros de entrada que se designan por medio de IN o N. El resultado es siempre el valor de la función. La siguiente tabla muestra los nombres de las funciones y los tipos de datos de los dos parámetros de entrada y del valor de función. Significan lo siguiente:

- Parámetro de entrada IN: búfer en el que se ejecutan las operaciones de desplazamiento de bits. El tipo de datos de este parámetro de entrada determina el tipo de datos del valor de función.
- Parámetro de entrada N: número de rotaciones en las funciones ROL y ROR del búfer de circulación, o el número de posiciones que deben desplazarse en SHL y SHR.

La tabla muestra las posibles funciones estándar de cadena de bits:

Nombre de la función	**Tipo de datos Parámetro de entrada IN**	**Tipo de datos Parámetro de entrada N**	**Tipo de datos Valor de función**	**Tarea**
ROL	BOOL BYTE WORD DWORD	INT INT INT INT	BOOL BYTE WORD DWORD	El valor que se encuentra en el parámetro IN se desplaza tantas posiciones de bits hacia la izquierda como indique el contenido del parámetro N.
ROR	BOOL BYTE WORD DWORD	INT INT INT INT	BOOL BYTE WORD DWORD	El valor que se encuentra en el parámetro IN se desplaza tantas posiciones de bits hacia la derecha como indique el contenido del parámetro N.
SHL	BOOL BYTE WORD DWORD	INT INT INT INT	BOOL BYTE WORD DWORD	En el valor existente en el parámetro IN se desplazan hacia la izquierda tantos bits y se sustituye por 0 el mismo número de posiciones en el lado derecho como indique el contenido del parámetro N .
SHR	BOOL BYTE WORD DWORD	INT INT INT INT	BOOL BYTE WORD DWORD	En el valor existente en el parámetro IN se desplazan hacia la derecha tantos bits y se reemplazan por 0 el mismo número de posiciones en el lado izquierdo como indique el contenido del parámetro N .

Nota

También es posible utilizar otras funciones IEC para la conversión del tipo de datos. Para más información sobre las funciones disponibles, consulte el manual de referencia de STEP 7 "Funciones de sistema y funciones estándar para S7-300/400".

14.3.1 Ejemplos de funciones estándar de cadena de bits

Llamada	Resultado
RESULTADO:= ROL (IN:=BYTE#2#1101_0011, N:=5);	// 2#0111_1010 // (= 122 decimal)
RESULTADO:= ROR (IN:=BYTE#2#1101_0011, N:=2);	// 2#1111_0100 // (= 244 decimal)
RESULTADO:= SHL (IN:=BYTE#2#1101_0011, N:=3);	// 2#1001_1000 // (= 152 decimal)
RESULTADO:= SHR (IN:=BYTE#2#1101_0011, N:=2);	// 2#0011_0100 // (= 52 decimal)

14.4 Funciones para procesar cadenas de caracteres

14.4.1 Funciones para manipular cadenas

LEN

La función LEN (FC 21) emite la longitud actual de una cadena de caracteres (número de los caracteres válidos) como valor de retorno. Una cadena de caracteres vacía ('') tiene la longitud cero. La función no comunica ningún error.

```
Ejemplo: LEN (S:= 'XYZ' )
```

Parámetro	Declaración	Tipo de datos	Área de memoria	Descripción
S	INPUT	STRING	D, L	Variable de entrada en formato STRING
Valor de respuesta		INT	E, A, M, D, L	Número de caracteres actuales

CONCAT

La función CONCAT agrupa como máximo 32 variables STRING en una cadena de caracteres. Si la cadena de caracteres del resultado es de mayor longitud que la variable establecida en el parámetro de salida se limitará la cadena de caracteres del resultado a la longitud establecida. En caso de utilizar la función S7-SCL CONCAT, automáticamente se llama la FC2 de la librería "Funciones IEC".

```
Ejemplo: CONCAT (IN1:= 'Valvula', IN2:= 'abierta')
```

Parámetro	Declaración	Tipo de datos	Área de memoria	Descripción
IN1	INPUT	STRING CHAR	D, L	Variable de entrada en formato STRING o CHAR
IN2	INPUT	STRING CHAR	D, L	Variable de entrada en formato STRING o CHAR
INn	INPUT	STRING CHAR	D, L	Variable de entrada en formato STRING o CHAR
Valor de respuesta		STRING CHAR	D, L	Cadena de caracteres reagrupada

LEFT o RIGHT

Las funciones LEFT o RIGHT (FC 20 ó FC 32) aportan los primeros o últimos caracteres L de una cadena de caracteres. Si L es mayor que la longitud actual de la variable STRING se devolverá la cadena completa. En L = 0 se devolverá una cadena vacía. Si L es negativa se emitirá una cadena vacía.

```
Ejemplo: LEFT (IN:= 'Valvula', L:= 4)
```

Parámetro	Declaración	Tipo de datos	Área de memoria	Descripción
IN	INPUT	STRING	D, L	Variable de entrada en formato STRING
L	INPUT	INT	E, A, M, D, L, Konst.	Longitud de la cadena de caracteres izquierda
Valor de respuesta		STRING	D, L	Variable de salida en formato STRING

MID

La función MID (FC 26) aporta una parte de una cadena de caracteres. L es la longitud de la cadena de caracteres que se debe leer, P es la posición del primer carácter que se debe leer.
Si la suma de L y (P-1) supera la longitud de la variable STRING se aportará una cadena de caracteres a partir del carácter P. hasta el final del valor de entrada. En todos los demás casos (P se encuentra fuera de la longitud actual, P y/o L es igual a cero o negativa) se emitirá una cadena vacía.

```
Ejemplo: MID (IN:= 'Temperatura', L:= 2, P:= 3)
```

Parámetro	Declaración	Tipo de datos	Área de memoria	Descripción
IN	INPUT	STRING	D, L	Variable de entrada en formato STRING
L	INPUT	INT	E, A, M, D, L, Konst.	Longitud de la cadena central de caracteres
P	INPUT	INT	E, A, M, D, L, Konst.	Posición del primer carácter
Valor de respuesta		STRING	D, L	Variable de salida en formato STRING

INSERT

La función INSERT (FC 17) inserta la cadena de caracteres del parámetro IN2 detrás del carácter P. en la cadena de caracteres del parámetros IN1.Si P es igual a cero, se inserta la segunda cadena de caracteres delante de la primera cadena. Si P es mayor que la longitud actual de la primera cadena de caracteres se insertará la segunda cadena de caracteres detrás de la primera. Si P es negativa se emitirá una cadena vacía. Si la cadena de caracteres del resultado es de mayor longitud que la variable indicada en el parámetro de salida, se límitará la cadena de caracteres del resultado a la longitud máxima ajustada.

```
Ejemplo: INSERT (IN1:= 'abonado_hallado', IN2:='Garcia':= 2, P:= 11)
```

Parámetro	Declaración	Tipo de datos	Área de memoria	Descripción
IN1	INPUT	STRING	D, L	Variable STRING en la cual se insertará
IN2	INPUT	STRING	D, L	Variable STRING que se debe insertar
P	INPUT	INT	E, A, M, D, L, const.	Posición de inserción
Valor de respuesta		STRING	D, L	Cadena de caracteres del resultado

DELETE

La función DELETE (FC 4) borra de una cadena de caracteres los caracteres L a partir del carácter P. (inclusive). Si L y/o P es igual a cero o si P es mayor que la longitud actual de la cadena de caracteres de entrada se devolverá la cadena de caracteres de entrada. Si la suma de L y P es mayor que la cadena de caracteres de entrada se borrará hasta el final de la cadena de caracteres. Si L y/o P es negativa, se emitirá una cadena vacía.

Ejemplo: DELETE (IN:= 'Temperatura ok', L:= 6, P:= 5)

REPLACE

La función REPLACE (FC 31) sustituye caracteres L de la primera cadena de caracteres (IN1) a partir del caracter P. (inclusive) por la segunda cadena de caracteres (IN2). Si L es igual a cero, se devolverá la primera cadena. Si P es igual a cero o uno, se sustituye a partir del primer caracter (inclusive). Si P se encuentra fuera de la primera cadena, se insertará la segunda cadena detrás de la primera. Si L y/o P es negativa, se emitirá una cadena vacía. Si la cadena de caracteres del resultado es de mayor longitud que la variable indicada en el parámetro de salida, se limitará la cadena del resultado a la longitud máxima establecida.

```
Ejemplo: REPLACE (IN1:= 'Temperatura', IN2:= ' hoch' L:= 6, P:= 5)
```

Parámetro	Declaración	Tipo de datos	Área de memoria	Descripción
IN1	INPUT	STRING	D, L	Variable STRING en la cual se insertará
IN2	INPUT	STRING	D, L	Variable STRING que se debe insertar
L	INPUT	INT	E, A, M, D, L, Const.	Número de caracteres que se deben sustituir
P	INPUT	INT	E, A, M, D, L, Const.	Posición del primer carácter sustituido
Valor de respuesta		STRING	D, L	Cadena de caracteres del resultado

FIND

La función FIND (FC 11) aporta la posición de la segunda cadena de caracteres (IN2) dentro de la primera cadena (IN1). La búsqueda comienza a la izquierda; se notifica la primera aparición de la cadena. Si la segunda cadena no existe en la primera, se notificará de vuelta cero. La función no notifica ningún error.

```
Ejemplo: FIND (IN1:= 'Equipo de procesamiento', IN2:='station')
```

Parámetro	Declaración	Tipo de datos	Área de memoria	Descripción
IN1	INPUT	STRING	D, L	Variable STRING, en la cual se busca
IN2	INPUT	STRING	D, L	Variable STRING que se debe buscar
Valor de respuesta		INT	E, A, M, D, L	Posición de la cadena de caracteres encontrada

14.4.2 Funciones para comparar cadenas de caracteres

Las comparaciones con cadenas de caracteres como operandos son posibles con las operaciones S7-SCL =, <>, <, >, <= o bien >=. El compilador insertará la correspondiente llamada de función automáticamente. Las siguientes funciones se mencionan por razones de totalidad.

EQ_STRNG y NE_STRNG

La función EQ_STRNG (FC 10) o (FC 29) compara el contenido de dos variables en formato STRING respecto a igual (FC10) o distinta (FC29) y emite el resultado de la comparación como valor de retorno. El valor de retorno muestra el estado de señal "1" si la cadena de caracteres del parámetro S1 es igual (distinta) que la cadena en el parámetro S2. La función no notifica ningún error.

Parámetro	Declaración	Tipo de datos	Área de memoria	Descripción
S1	INPUT	STRING	D, L	Variable de entrada en formato STRING
S2	INPUT	STRING	D, L	Variable de entrada en formato STRING
Valor de respuesta		BOOL	E, A, M, D, L	Resultado de comparación

GE_STRNG y LE_STRNG

La función GE_STRNG (FC 13) o (FC 19) compara el contenido de dos variables en formato STRING respecto a mayor (menor) o igual y emite el resultado de comparación como valor de retorno. El valor de retorno muestra el estado de señal "1" si la cadena de caracteres del parámetro S1 es mayor (menor) o igual que la cadena de caracteres del parámetro S2. Los caracteres se comparan, empezando por la izquierda, a través de su codificación ASCII (p. ej., es 'a' mayor que 'A'). El primer carácter distinto decide sobre el resultado de la comparación. Si la parte izquierda de la cadena de caracteres más larga es idéntica con la cadena de caracteres más corta, la cadena más larga será válida como la mayor. La función no notifica ningún error.

Parámetro	Declaración	Tipo de datos	Área de memoria	Descripción
S1	INPUT	STRING	D, L	Variable de entrada en formato STRING
S2	INPUT	STRING	D, L	Variable de entrada en formato STRING
Valor de respuesta		BOOL	E, A, M, D, L	Resultado de comparación

GT_STRNG y LT_STRNG

La función GT_STRNG (FC 15) o (FC 24) compara el contenido de dos variables en formato STRING respecto mayor (menor) y emite el resultado de comparación como valor de retorno. El valor de retorno muestra el estado de señal "1" si la cadena de caracteres en el parámetro S1 es mayor (menor) que la cadena de caracteres en el parámetro S2. Los caracteres se comparan a través de la codificación ASCII, empezando por la izquierda (p. ej., es 'a' mayor que 'A'). El primer carácter distinto decide sobre el resultado de la comparación. Si la parte izquierda de la cadena de caracteres más larga es idéntica a la cadena de caracteres más corta, la cadena más larga será válida como mayor. La función no muestra ningún error.

Parámetro	Declaración	Tipo de datos	Área de memoria	Descripción
S1	INPUT	STRING	D, L	Variable de entrada en formato STRING
S2	INPUT	STRING	D, L	Variable de entrada en formato STRING
Valor de respuesta		BOOL	E, A, M, D, L	Resultado de comparación

14.4.3 Funciones para convertir el formato de datos

INT_TO_STRING y STRING_TO_INT

Las funciones INT_TO_STRING y STRING_TO_INT convierten una variable del formato INT en una cadena de caracteres, o una cadena de caracteres en una variable INT. Implícitamente se utilizan las funciones I_STRNG (FC16) o STRNG_I (FC38) de la librería "Funciones IEC". La cadena de caracteres se representará precedida por un signo. Si la variable indicada en el parámetro de retorno demasiado corta, no tendrá lugar la conversión.

Parámetro	Declaración	Tipo de datos	Área de memoria	Descripción
INT_TO_STRING				
I	INPUT	INT	E, A, M, D, L, Const.	Valor de entrada
Valor de respuesta		STRING	D, L	Cadena de caracteres como resultado
STRING_TO_INT				
S	INPUT	STRING	D, L	Cadena de caracteres de entrada
Valor de respuesta		INT	E, A, M, D, L	Resultado

DINT_TO_STRING y STRING_TO_DINT

Las funciones DINT_TO_STRING y STRING_TO_DINT convierten una variable en formato DINT en una cadena de caracteres, o una cadena de caracteres en una variable DINT. Implícitamente se utilizan las funciones DI_STRNG (FC5) o STRNG_DI (FC37) de la librería suministrada "Funciones IEC". La cadena de caracteres se representará precedida por un signo. Si la variable indicada en el parámetro de retorno es demasiado corta, no tendrá lugar ninguna conversión.

Parámetro	Declaración	Tipo de datos	Área de memoria	Descripción
DINT_TO_STRING				
I	INPUT	DINT	E, A, M, D, L, Const.	Valor de entrada
Valor de respuesta		STRING	D, L	Cadena de caracteres de resultado
STRING_TO_DINT				
S	INPUT	STRING	D, L	Cadena de caracteres de entrada
Valor de respuesta		DINT	E, A, M, D, L	Resultado

REAL_TO_STRING y STRING_TO_REAL

Las funciones REAL_TO_STRING y STRING_TO_REAL convierten una variable en formato REAL en una cadena de caracteres, o una cadena de caracteres en una variable REAL. Implícitamente se utilizan las funciones R_STRNG (FC30) o STRNG_R (FC39) de la librería "Funciones IEC" suministrada. La cadena de caracteres debe tener el siguiente formato: ±v.nnnnnnnE±xx (± = signo, v = unidad, n = decimales, x = dígitos exponenciales)

Si la longitud de la cadena de caracteres es menor que 14 o si su estructura se diferencia del formato indicado arriba, no se efectúa la conversión

Si la variable indicada en el parámetro de retorno es demasiado corta, o si en el parámetro IN no aparece ningún número en coma flotante válido, no tendrá lugar ninguna.

Parámetro	**Declaración**	**Tipo de datos**	**Área de memoria**	**Descripción**
REAL_TO_STRING				
IN	INPUT	REAL	E, A, M, D, L, Const.	Valor de entrada
Valor de respuesta		STRING	D, L	Cadena de caracteres de resultado
STRING_TO_REAL				
S	INPUT	STRING	D, L	Cadena de caracteres de entrada
Valor de respuesta		REAL	E, A, M, D, L	resultado

14.4.4 Ejemplo de procesamiento de cadenas de caracteres

Componer textos de mensaje

```
//Componer y almacenar textos de mensajes con control del proceso

////////////////////////////////////////////////////////////////////
//El bloque contiene los textos de mensaje necesarios y los       //
//últimos 20 mensajes generados                                   //
////////////////////////////////////////////////////////////////////

DATA_BLOCK Textos de mensajes

    STRUCT
        Index            : int;
        bufer_de_texto  : array [0..19] of string[34];
        HW                  : array [1..5] of string[16]; //5
distintos aparatos
        stati            : array [1..5] of string[12]; // 5 estados
diferentes
    END_STRUCT
BEGIN
    Index  :=0;
    HW[1]  := 'Motor ';
    HW[2]  := 'Valvula ';
    HW[3]  := 'Prensa ';
    HW[4]  := 'Equipo_de_soldadura ';
    HW[5]  := 'Soplete ';
    Stati[1]  := ' averiado';
    Stati[2]  := ' iniciado';
    Stati[3]  := ' Temperatura';
    Stati[4]  := ' reparado';
    Stati[5]  := ' mantenimiento';
END_DATA_BLOCK

////////////////////////////////////////////////////////////////////
//La función compone textos y los escribe en el DB. Los textos se //
//depositan //en un búfer en anillo. El siguiente índice libre    //
//del búfer de textos //también se deposita en el DB Textos de    //
//mensajes y es actualizado por la //función.                     //
////////////////////////////////////////////////////////////////////

FUNCTION Generador_de_textos : bool
VAR_INPUT
    unit      : int;   //Indice del texto del aparato
    nr          : int; //N° ID del aparato
    status  : int;
    valor      : int;
END_VAR
VAR_TEMP
    texto  : string[34];
    i        : int;
END_VAR
```

```
//inicialización de las variables temporales
   texto := '';
   generador_de_textos := true;
   Case unit of
    1..5 : case status of
       1..5 : text := concat( in1 := textos_mensajes.HW[unit],
                              in2 := right(l:=2,in:=I_STRNG(nr)));
          text := concat( in1 := text,
                          in2 := textos_mensajes.stati[status]);
          valor if <> 0 then
          text := concat( in1 := texto,
                          in2 := I_STRNG(valor));
          end_if;
        else generador_de_textos:= false;
    end_case;
else generador_de_textos:= false;
end_case;
i := textos_mensajes.index;
    textos_mensajes.buferdetexto[i] := texto;
    textos_mensajes.index := (i+1) mod 20;
END_FUNCTION

//////////////////////////////////////////////////////////////////
//La función se llamará en el programa cíclico cuando se cambie   //
//de flanco en %M10.0 y registrará un mensaje una sola vez,       //
//si cambia un parámetro.                                         //
//////////////////////////////////////////////////////////////////

Organization_block Ciclo
Var_temp
    Besy_ifx : array [0..20] of byte;
    error: BOOL;
End_var;

//////////////////////////////////////////////////////////////////
//La siguiente llamada causa la entrada "Motor 12 arrancado" en   //
//el búfer de textos del DB Textos de mensaje, teniendo que       //
//transmitirse a //través de %MW0 un 1, a través de %EW2 el 12    //
//y a través de %MW2 se debe transmitir un 2. *)                  //
//////////////////////////////////////////////////////////////////

if %M10.0 <> %M10.1 then
    error:= generador_de_textos (unit   := word_to_int(%MW0),
                                 nr     := word_to_int(%EW2),
                                 status := word_to_int(%MW2),
                                 valor  := 0);
    %M10.1:=M10.0;
end_if;
end_organization_block
```

14.5 Funciones para la selección de valores

14.5.1 Funciones para la selección de valores

Las siguientes funciones para la selección de valores están disponibles en forma de funciones internas de S7-SCL. Cumplen la norma IEC 61131-3.

Nota

Algunas de las funciones también están incluidas en la librería estándar de STEP 7. Sin embargo, las funciones de la librería no cumplen todas las exigencias de la IEC.

SEL

La función SEL selecciona uno de dos valores de entrada.

Como valores de entrada se admiten todos los tipos de datos, a excepción de los tipos ARRAY, STRUCT y Parámetros. Todas las variables parametrizadas deben ser del tipo de datos de la misma clase.

Ejemplo:

```
A:= SEL (G:= SELECT, IN0:= X, IN1:= Y);
```

Parámetro	Declaración	Tipo de datos	Área de memoria	Descripción
G	INPUT	BOOL	E, A, M, D, L	Criterio de selección
IN0	INPUT	Todos los tipos de datos excepto ARRAY y STRUCT	E, A, M, D, L	Primer valor de entrada
IN1	INPUT	Todos los tipos de datos excepto ARRAY y STRUCT	E, A, M, D, L	Segundo valor de entrada
Valor de retorno	OUTPUT	Todos los tipos de datos excepto ARRAY y STRUCT	E, A, M, D, L	Valor de entrada seleccionado (opcional)

MAX

La función MAX serlecciona el mayor de un número de valores de variables.

Como valores de entrada se admiten tipos de datos numéricos y de tiempo. Todas las variables parametrizadas deben ser del tipo de datos de la misma clase. La expresión adopta el tipo de datos más significativo.

```
Ejemplo: A:= MAX (IN1:=a, IN2:=b, IN3:=c, IN4:=d);
```

Parámetror	Declaración	Tipo de datos	Área de memoria	Descripción
IN1	INPUT	ANY_NUM Tipos de datos de tiempo excepto S5TIME	E, A, M, D, L	Primer valor de entrada
IN2	INPUT	ANY_NUM Tipos de datos de tiempo excepto S5TIME	E, A, M, D, L	Segundo valor de entrada
Inn (n=3...32)	INPUT	ANY_NUM Tipos de datos de tiempo excepto S5TIME	E, A, M, D, L	Último valor de entrada (opcional)
Valor de retorno	OUTPUT	ANY_NUM Tipos de datos de tiempo excepto S5TIME	E, A, M, D, L	El mayor de los valores de entrada (opcional)

MIN

La función MIN selecciona el menor de un número de valores de variables. Como valores de entrada se admiten tipos de datos numéricos y de tiempo. Todas las variables parametrizadas deben ser del mismo tipo de datos. La expresión adopta el tipo de datos más significativo.

```
Ejemplo: A:= MIN (IN1:=a, IN2:=b, IN3:=c, IN4:=d);
```

Parámetror	Declaración	Tipo de datos	Área de memoria	Descripción
IN1	INPUT	ANY_NUM Tipos de datos de tiempo excepto S5TIME	E, A, M, D, L	Primer valor de entrada
IN2	INPUT	ANY_NUM Tipos de datos de tiempo excepto S5TIME	E, A, M, D, L	Segundo valor de entrada
Inn (n=3...32)	INPUT	ANY_NUM Tipos de datos de tiempo excepto S5TIME	E, A, M, D, L	Último valor de entrada (opcional)
Valor de retorno	OUTPUT	ANY_NUM Tipos de datos de tiempo excepto S5TIME	E, A, M, D, L	El menor de los valores de entrada (opcional)

LIMIT

La función LIMIT limita el valor numérico de un avariable a valores límite parametrizables. Como valores de entrada se admiten todos los tipos de datos numéricos y de tiempo. Todos los parámetros deben ser del mismo tipo de datos. La expresión adopta el tipo de datos más significativo. El valor límite inferior (MN) no puede ser superior al valor límite superior (MX).

```
Ejemplo: A:= LIMIT (MN:=5, IN:= pasos de ejecución, MX:= 10);
```

Parámetro	Declaración	Tipo de datos	Área de memoria	Descripción
MN	INPUT	ANY_NUM Tipos de datos de tiempo excepto S5TIME	E, A, M, D, L	Límite inferior
IN	INPUT	ANY_NUM Tipos de datos de tiempo excepto S5TIME	E, A, M, D, L	Variable de entrada
MX	INPUT	ANY_NUM Tipos de datos de tiempo excepto S5TIME	E, A, M, D, L	Límite superior
Valor de retorno	OUTPUT	ANY_NUM Tipos de datos de tiempo excepto S5TIME	E, A, M, D, L	Variable de salida limitada (opcional)

MUX

La función MUX selecciona un valor de entrada de un número de valores de entrada. La selección se lleva a cabo mediante el parámetro de entrada K. Como valores de entrada se admiten todos los tipos de datos. La expresión adopta el tipo de datos más significativo.

Ejemplo:

```
A:= MUX (K:=SELECT, IN0:= Steps, IN1:=Number, IN2:=Total);
```

Parámetro	Declaración	Tipo de datos	Área de memoria	Descripción
K	INPUT	INT	E, A, M, D, L	Criterio de selección
IN0	INPUT	Todos los tipos de datos excepto ARRAY y STRUCT	E, A, M, D, L	Primer valor de entrada
IN1	INPUT	Todos los tipos de datos excepto ARRAY y STRUCT	E, A, M, D, L	Segundo valor de entrada
Inn (n=2…31)	INPUT	Todos los tipos de datos excepto ARRAY y STRUCT	E, A, M, D, L	Último valor de entrada (opcional)

Parámetro	Declaración	Tipo de datos	Área de memoria	Descripción
INELSE	INPUT	Todos los tipos de datos excepto ARRAY y STRUCT	E, A, M, D, L	Valor de entrada alternativo (opcional) Si K se encuentra fuera de 0...n se utiliza el valor actual de INELSE. Si INELSE no está ocupado, se utilizará el valor actual de IN0.
Valor de retorno	OUTPUT	Todos los tipos de datos excepto ARRAY y STRUCT	E, A, M, D, L	Valor seleccionado (opcional)

14.6 SFCs, SFBs y librerías estándar

14.6.1 Funciones de sistema, bloques de función de sistema y librería estándar

Las CPU S7 incluyen, integradas en el sistema operativo, funciones del sistema y funciones estándar que puede utilizar al programar en S7-SCL. En particular son:

- bloques de organización (OB)
- funciones del sistema (SFC)
- bloques de función del sistema (SFB)

Interface de llamada (SFC/SFB)

Puede direccionar bloques de forma absoluta o simbólica. Para ello necesita el nombre simbólico que se debe declarar en la tabla de símbolos o el número para la designación absoluta del bloque.

Al efectuar la llamada, se deben asignar a los parámetros formales (cuyos nombres y tipos de datos fueron determinados al crear el bloque parametrizable) los parámetros actuales, con cuyos valores trabaja el bloque durante el tiempo de ejecución del programa.

S7-SCL explora los siguientes directorios y librerías buscando el bloque que se debe llamar:

- la carpeta "Programas"
- las librerías estándar Simatic
- la librería estándar IEC
- Si S7-SCL encuentra un bloque copia este bloque en el programa de usuario. Excepto los bloques, que debido a su nombre se deben llamar con la notación (" ... "), y los bloques de llamada absoluta. Estos nombres sólo se buscan en la tabla de símbolos del programa de usuario. Estas funciones se deben copiar en el administrador SIMATIC al programa de usuario.

Llamada condicional (SFB/SFC)

Para la llamada condicional se tiene que asignar previamente 0 al parámetro de entrada EN predefinido (p.ej., en la entrada E0.3); en tal caso no se llamará al bloque. Si se asigna 1 a EN, sí se llamará a la función. En tal caso el parámetro de salida ENO también será "1" (en caso contrario, será "0"), siempre y cuando no se produzca ningún error al procesar el bloque.

No es conveniente efectuar una llamada condicional a una SFC porque la variable que debe admitir el valor de retorno de la función no está definida si no se llama la función.

Nota

Si en su programa utiliza las siguientes operaciones para los tipos de datos TIME, DATE_AND_TIME y STRING, S7-SCL llamará implícitamente a los bloques estándar.

Por esta razón están reservados los símbolos y los números de bloque de estos bloques estándar y no se pueden utilizar para otros bloques. S7-SCL no comprueba en todos los casos posibles incumplimientos de esta nota, por lo que pueden aparecer errores de compilación.

La tabla siguiente muestra esquemáticamente las funciones estándar IEC utilizadas implícitamente por S7-SCL.

Operación	**DATE_AND_TIME**	**STRING**
==	EQ_DT (FC9)	EQ_STRING (FC10)
<>	NE_DT (FC28)	NE_STRING (FC29)
>	GT_DT (FC14)	GT_STRING (FC15)
>=	GE_DT (FC12)	GE_STRING (FC13)
<=	LE_DT (FC18)	LE_STRING (FC19)
<	LT_DT (FC23)	LT_STRING (FC24)
DATE_AND_TIME + TIME	AD_DT_TM (FC1)	
DATE_AND_TIME + TIME	SB_DT_TM (FC35)	
DATE_AND_TIME + DATE_AND_TIME	SB_DT_DT (FC34)	
TIME_TO_S5TIME(TIME)	TIM_S5TI (FC40)	
S5TIME_TO_TIME(S5TIME)	S5TI_TIM (FC33)	

En el manual de referencia de STEP 7 "Funciones de sistema y funciones estándar para S7-300/400" encontrará más información sobre los SFB, SFC y OBs disponibles así como una descripción detallada de los interfaces.

14.6.2 Interface de transferencia al OB

Bloques de organización

Los bloques de organización forman el interface entre el sistema operativo de la CPU y el programa de usuario. Los OB permiten ejecutar secciones del programa de forma puntualizada:

- al arrancar la CPU,
- en ejecución cíclica o por ciclos de tiempo,
- a determinadas horas del día o en determinados días,
- después de que transcurra un tiempo predefinido,
- cuando aparezcan errores,
- cuando aparezcan alarmas de proceso y de comunicación

Los bloques de organización se ejecutan siguiendo el orden de prioridad asignado por usted.

OB disponible

No todas las CPUs pueden procesar todas las OB disponibles en S7. En las hojas de datos de su CPU podrá consultar los OB de que dispone.

15 Descripción del lenguaje

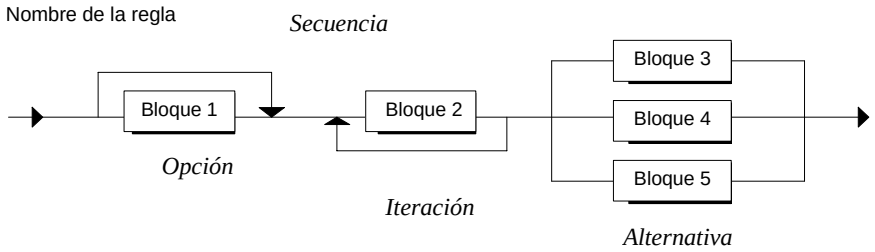
15.1 Descripción formal

15.1.1 Sinopsis de los diagramas sintácticos

La descripción del lenguaje en cada uno de los capítulos se basa en los diagramas sintácticos. Éstos proporcionan una visión general de la estructura sintáctica de S7-SCL. En los capítulos "Regulación léxica" y "Regulación sintáctica" encontrará un resumen completo de todos los diagramas con los elementos de lenguaje correspondientes .

¿Qué es un diagrama sintáctico?

El diagrama sintáctico es una representación gráfica de la estructura del lenguaje. La estructura se describe mediante una secuencia de reglas. Algunas reglas pueden basarse en reglas anteriormente expuestas.

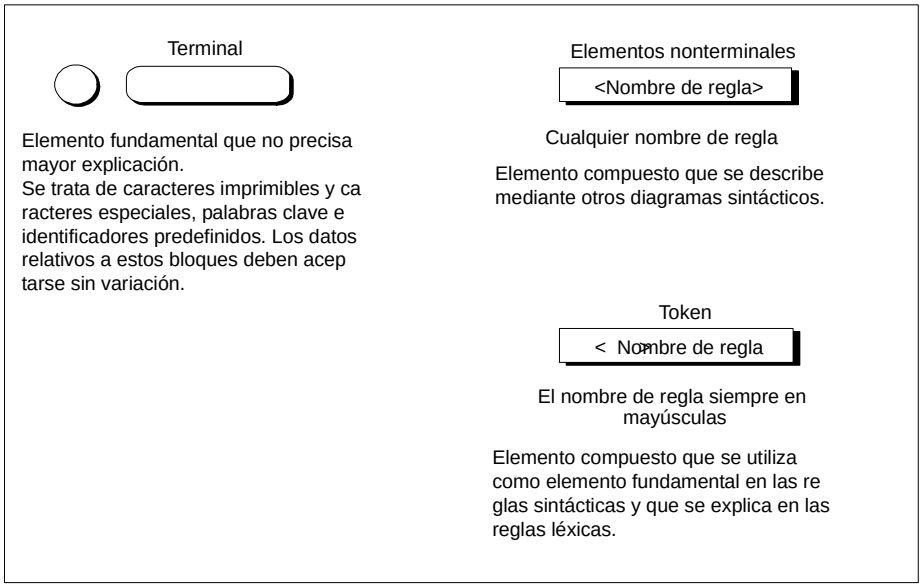


El diagrama sintáctico se lee de izquierda a derecha. Deben tenerse en cuenta las siguientes estructuras:

- Secuencia: secuencia de bloques
- Opción : rama que puede saltarse
- Iteración: repetición de ramas
- Alternativa: ramificación

¿Qué tipos de bloques hay?

Un bloque es un elemento básico o un elemento que a su vez se compone de otros bloques. La figura siguiente indica los tipos de símbolo que corresponden a los bloques:



15.1.2 Reglas

Las reglas que se pueden utilizar para estructurar el programa S7-SCL se dividen en las categorías **léxica** y **sintáctica**.

Reglas léxicas

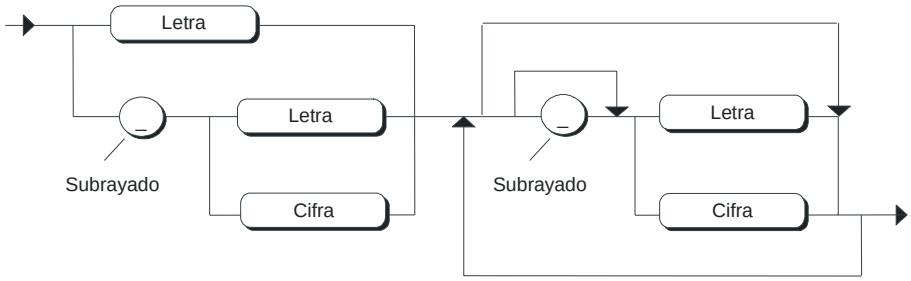
Las reglas léxicas determina la estructura de los elementos (token) que se van a procesar durante el análisis léxico del compilador. Por ello, en la escritura no hay libertad de formato y deben respetarse estrictamente las reglas. En particular, esto significa que:

- no está permitida la inserción de signos de formateado,
- no está permitido introducir comentarios de línea ni de bloque,
- no pueden insertarse atributos para los identificadores.

El ejemplo muestra la regla léxica IDENTIFICADOR, que determina la estructura de un identificador (nombre), p. ej.:

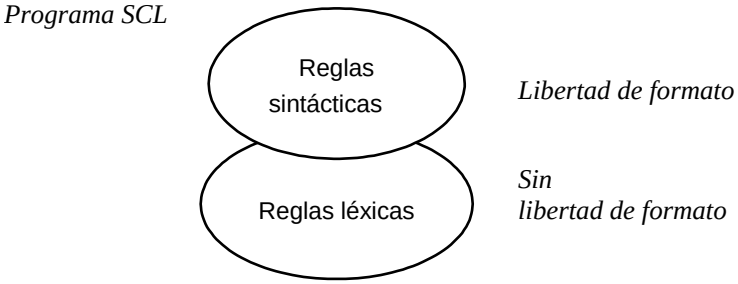
```
ARRAY_MED_12
VALOR_DE_PRESELECCION_B_1
```

IDENTIFICADOR



Reglas sintácticas

Partiendo de las reglas léxicas, en las reglas sintácticas se describe la estructura de S7-SCL. En el margen de estas reglas, se puede crear el programa S7-SCL sin formatear:



Aspectos formales

Cada regla va precedida de un nombre de regla. Cuando la regla se utiliza en una regla de orden superior, su nombre aparece encerrado en un recuadro.

Si el nombre de la regla está escrito en mayúsculas se trata de un token, que se describe en las reglas léxicas.

Aspectos semánticos

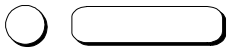
En las reglas sólo se puede representar la estructura formal del lenguaje, de la cual no siempre puede deducirse su significado, es decir, su semántica. Por ello, en puntos importantes se escriben informaciones adicionales junto a las reglas. Ejemplos:

- en los elementos similares de distinto significado se asigna un nombre adicional: p. ej. en la regla Indicación de fecha, en la SECUENCIA DE CIFRAS DECIMALES año, mes o día. Del nombre se deriva el uso que se le dará.
- Las restricciones importantes se hacen constar junto a las reglas: p. ej. junto a la regla Símbolo aparece la nota de que es necesario definir un símbolo en la tabla de símbolos.

15.1.3 Terminales de las reglas léxicas

Definición

Un terminal es un elemento básico que no se explica por medio de otra regla, sino de manera verbal. En los diagramas sintácticos aparecerán con el siguiente símbolo:



El elemento terminal se representa mediante un óvalo o un círculo. Se indicará directamente el con tenido o el nombre (en letras mayúsculas).
Define la cantidad de caracteres ASCII que pueden utilizarse.

En las tablas siguientes se especifican los terminales por medio de la indicación del número de elementos del juego de caracteres ASCII.

Letras y cifras

Las letras y las cifras son los caracteres que más se utilizan. El IDENTIFICADOR está formado p. ej. por letras, cifras y caracteres de subrayado.

Caracteres	**Subgrupo**	**Elementos del juego de caracteres**
Letra	Mayúscula minúscula	A.. Z a.. z
Cifra	Cifra decimal	0.. 9
Cifra octal	Cifra octal	0.. 7
Cifra hexadecimal	Cifra hexadecimal	0.. 9, A.. F, a.. f
Bit	Cifra binaria	0, 1

Caracteres imprimibles y caracteres especiales

El juego de caracteres ASCII ampliado y completo se puede utilizar en cadenas, comentarios y símbolos.

Caracteres	**Subgrupo**	**Elementos del juego de caracteres**
Carácter imprimble	depende del código de caracteres utilizado. En el código ASCII p. ej. a partir del equivalente decimal 31, sin DEL y sin los siguientes caracteres sustitutivos:	todos los caracteres imprimbles
Caracteres sustitutivos	Carácter dólar Apóstrofe	$ '
Signos de control	$P o $p $L o $l $R o $r $T o $t $N o $n	Cambio de página (formfeed, page) Cambio de línea (linefeed) Retorno de carro (carriage return) Tabulador Línea nueva
Representación complementaria en código hexadecimal	$hh	cualquier carácter, expresable en código hexadecimal (hh)

15.1.4 Caracteres de formateo y de separación y operaciones

En las reglas léxicas

En la siguiente tabla encontrará ejemplos del uso de caracteres del juego de caracteres ASCII como caracteres de formateo y de separación en el ámbito de las reglas léxicas.

Caracteres	Descripción
:	Carácter de separación entre horas, minutos y segundos Atributos
.	Caracteres de separación para el direccionamiento absoluto de la representación de intervalos de tiempo y números reales
' '	Caracteres y cadena de caracteres
" "	Signo de inicio de símbolo según las reglas de la tabla de símbolos
_subrayado	En los IDENTIFICADORES puede haber un carácter de separación para valores numéricos en constantes.
$	Símbolo de escape para indicar signos de control o caracteres complementarios
$> $<	Interrupción de la cadena en caso de que la cadena no coincida en una línea o se instroduzcan comentarios.

Para constantes

En la siguiente tabla encontrará ejemplos del uso de caracteres individuales y cadenas de caracteres para constantes en el ámbito de las reglas léxicas. La tabla rige para la nemotécnica alemana y la inglesa.

Prefijo	Identificador de	Regla léxica
BOOL#	Constante tipificada del tipo BOOL	Constante de bit
BYTE#	Constante tipificada del tipo BYTE	Constante de bit
WORD#	Constante tipificada del tipo WORD	Constante de bit
DWORD#	Constante tipificada del tipo DWORD	Constante de bit
INT#	Constante tipificada del tipo INT	Constante entera
DINT#	Constante tipificada del tipo DINT	Constante entera
REAL#	Constante tipificada del tipo REAL	Constante de número real
CHAR#	Constante tipificada del tipo CHAR	Constante CHAR
2#	Constante numérica	Secuencia de cifras binarias
8#	Constante numérica	Secuencia de cifras octales
16#	Constante numérica	Secuencia de cifras hexadecimales
D#	Indicación de tiempo	FECHA
DATE#	Indicación de tiempo	FECHA
DATE_AND_TIME#	Indicación de tiempo	FECHA Y HORA
DT#	Indicación de tiempo	FECHA Y HORA
E	Caracteres de separación para constantes de número real	Exponente
e	Caracteres de separación para constantes de número real	Exponente
D	Carácter de separación para intervalo de tiempo (día)	Días (regla: representación escalonada)

Prefijo	Identificador de	Regla léxica
H	Carácter de separación para intervalo de tiempo (hora)	Horas: (regla: representación escalonada)
M	Carácter de separación para intervalo de tiempo (minutos)	Minutos (regla: representación escalonada)
MS	Carácter de separación para intervalo de tiempo (milisegundos)	Milisegundos (regla: representación escalonada)
S	Carácter de separación para intervalo de tiempo (segundos)	Segundos (regla: representación escalonada)
T#	Indicación de tiempo	INTERVALO
TIME#	Indicación de tiempo	INTERVALO
TIME_OF_DAY#	Indicación de tiempo	HORA
TOD#	Indicación de tiempo	HORA

En las reglas sintácticas

En la siguiente tabla encontrará ejemplos del uso de caracteres individuales como caracteres de formateo y de separación en el ámbito de las reglas sintácticas así como en comentarios y atributos.

Caracteres	Descripción	Regla sintáctica, comentario o atributo
:	Carácter de separación para indicación de tipo, en instrucción después de meta del salto	Declaración de variable, declaración de instancia, función, área de instrucciones, instrucción CASE
;	Terminación de una declaración o de una instrucción	Declaración de variable, área de instrucciones, área de asignación DB, bloque de constantes, bloque de metas de salto, declaración de componentes
,	Carácter de separación para listas y bloques de metas de salto	Declaración de variable, especificación de tipo de datos ARRAY, lista de inicialización de arrays, parámetros FB, parámetros FC, lista de valores, declaración de instancia
..	Indicación de rango	Especificación de tipo de datos ARRAY, lista de valores
.	Carácter de separación para nombres FB y DB, direccionamiento absoluto	Llamada a FB, variable estructurada
( )	Llamada a función y bloque de función Anidamiento en expresiones	Llamada de función, llamada a FB, expresión, Lista de inicialización de arrays, multiplicación simple, expresión de potencia
[ ]	Declaración ARRAY, Variable estructurada, array parcial, indizado en variables globales y strings	Especificación de tipo de datos ARRAY, especificación de tipo de datos STRING
(* *)	Bloque de comentario	consulte "Reglas léxicas"
//	Línea de comentario	consulte "Reglas léxicas"
{ }	Bloque de atributos	Especificación de atributos
%	Prefijo para identificadores directos	Para programar conforme a IEC se puede utilizar %M4.0 en lugar de M4.0.
#	Prefijo para una no palabra clave	Designa un identificador como no palabra clave, p.ej. #FOR.

Operaciones

En la siguiente tabla están representadas todas las operaciones S7-SCL, las palabras clave, p. ej. AND y las operaciones habituales aparecen como caracteres sencillos. La tabla rige para la nemotécnica alemana y la inglesa.

Operación	**Descripción**	**Regla sintáctica**
:=	Operación de asignación, asignación de inicio, inicialización del tipo de datos	Asignación de valores, área de asignación DB, bloque de constantes, asignación de salida o entrada/salida, asignación de entrada, asignación de entrada/salida
+, -	Operaciones aritméticas: operaciones unarias, signo	Expresión, expresión simple, expresión de potencia
+, -, *, / MOD; DIV	Operaciones aritméticas básicas	Operaciones aritméticas básicas, multiplicación simple
**	Operaciones aritméticas: operación de potencia	Expresión
NOT	Operaciones lógicas: negación	Expresión
AND, &, OR; XOR,	Operaciones lógicas básicas	Operación lógica básica
<,>,<=,>=,=,<>	Operación de comparación	Operación de comparación

15.1.5 Palabras clave e identificadores predefinidos

En la siguiente tabla encontrará una lista en orden alfabético de las palabras clave de S7-SCL y los identificadores predefinidos. Además, se incluye una descripción y la regla sintáctica en la que se utilizan como terminales. En general, las palabras clave no dependen de la nemotécnica utilizada.

Palabras clave	Descripción	Regla sintáctica
AND	Operación lógica	Operación lógica básica
ANY	Identificador del tipo de datos ANY	Especificación del tipo de datos de parámetro
ARRAY	Inicio de especificación de un array, después sigue la lista de índice encerrada entre "[" y "]"	Especificación de tipo de datos ARRAY
AT	Declara una vista sobre una varaible	Declaración de variables
BEGIN	Inicio del área de instrucciones de bloques lógicos o del área de inicialización de bloques de datos	Bloque de organización, función, bloque de función, bloque de datos
BLOCK_DB	Identificador de tipo de datos BLOCK_DB	Especificación del tipo de datos de parámetro
BLOCK_FB	Identificador de tipo de datos BLOCK_FB	Especificación del tipo de datos de parámetro
BLOCK_FC	Identificador de tipo de datos BLOCK_FC	Especificación del tipo de datos de parámetro
BLOCK_SDB	Identificador de tipo de datos BLOCK_SDB	Especificación del tipo de datos de parámetro
BOOL	Tipo de datos simple para datos binarios	Tipo de datos de bits
BY	Inicio de salto	Instrucción FOR
BYTE	Tipo de datos simple	Tipo de datos de bits
CASE	Inicio de instrucción de control para selección	Instrucción CASE
CHAR	Tipo de datos simple	Tipo de caracteres
CONST	Inicio de definición de constantes	Bloque de constantes
CONTINUE	Intrucción de control para los bucles FOR, WHILE y REPEAT	Instrucción CONTINUE
COUNTER	Tipo de datos para contador, sólo utilizable en bloque de parámetros	Especificación de tipo de datos de parámetro
DATA_BLOCK	Inicio del bloque de datos	Bloque de datos
DATE	Tipo de datos simple para fecha	Tipo de temporizador
DATE_AND_TIME	Tipo de datos compuesto para fecha y hora del día	DATE_AND_TIME
DINT	Tipo de datos simple para número (entero) precisión doble	Tipo de datos numérico
DIV	Operación de división	Operaciones aritméticas básicas, multiplicación simple
DO	Introducción del área de instrucciones en la instrucción FOR	Intrucción FOR, instrucción WHILE
DT	Tipo de datos simple para fecha y hora del día	DATE_AND_TIME
DWORD	Tipo de datos simple Doble palabra	Tipo de datos de bits
ELSE	Inicio de la condición cuando no se ha cumplido ninguna condición	Intrucción IF, instrucción CASE
ELSIF	Introducción de una condición no combinable	Instrucción IF
EN	OK-flag para confirmación de bloque	
ENO	Marca de error del bloque	

Palabras clave	Descripción	Regla sintáctica
END_CASE	Terminación de la instrucción CASE	Instrucción CASE
END_CONST	Terminación de la definición de constantes	Bloque de constantes
END_DATA_BLOCK	Terminación del bloque de datos	Bloque de datos
END_FOR	Terminación de la instrucción FOR	Instrucción FOR
END_FUNCTION	Terminación de la función	Función
END_FUNCTION_BLOCK	Terminación del bloque de función	Bloque de función
END_IF	Terminación de la instrucción IF	Instrucción IF
END_LABEL	Terminación de declaración de un bloque de metas de salto	Bloque de metas de salto
END_TYPE	Terminación del UDT	Tipo de datos de usuario
END_ORGANIZATION_BLOCK	Terminación del bloque de organización	Bloque de organización
END_REPEAT	Terminación de la instrucción REPEAT	instrucción REPEAT
END_STRUCT	Terminación de la especificación de una estructura	Especificación de tipo de datos STRUCT
END_VAR	Terminación de un bloque de declaración	Bloque de variables temporales, bloque de variables estáticas, bloque de parámetros
END_WHILE	Terminación de la instrucción WHILE	instrucción WHILE
EXIT	Interrupción directa del procesamiento de un bucle	EXIT
FALSE	Constante booleana predefinida: condición lógica no cumplida, valor igual a 0	
FOR	Inicio de instrucción de control para el procesamiento de bucles	Instrucción FOR
FUNCTION	Inicio de función	Función
FUNCTION_BLOCK	Inicio de bloque de función	Bloque de función
GOTO	Instrucción para ejecutar un salto hasta una meta de salto	Salto en el programa
IF	Inicio de instrucción de control para selección	Instrucción IF
INT	Tipo de datos simple para número (entero), precisión simple	Tipo de datos numérico
LABEL	Inicio de declaración de un bloque de metas de salto	Bloque de metas de salto
MOD	Operación aritmética de resto de división	Operación aritmética básica, multiplicación simple
NIL	Puntero cero	
NOT	Operación lógica, pertenece a las operaciones unarias	Expresión
OF	Inicio de especificación del tipo de datos	Especificación del tipo de datos ARRAY, instrucción CASE
OK	OK flag que expresa si se han ejecutado sin error las instrucciones de un bloque	
OR	Operación lógica	Operación lógica básica
ORGANIZATION_BLOCK	Inicio del bloque de organización	Bloque de organización
POINTER	Tipo de datos POINTER (puntero), sólo está permitido en la declaración de parámetros del bloque de parámetros, no se ejecuta en S7-SCL	Véase el capítulo "Datos globales".

Palabras clave	Descripción	Regla sintáctica
PROGRAM	Introducción del área de instrucciones de un FB (sinónimo de FUNCTION_BLOCK)	Bloque de función
REAL	Tipo de datos simple	Tipo de datos numérico
REPEAT	Inicio de instrucción de control para el procesamiento de bucles	Instrucción REPEAT
RET_VAL	Valor de respuesta de una función	Función
RETURN	Instrucción de control para retorno desde subrutina	Instrucción RETURN
S5TIME	Tipo de datos simple para indicaciones de tiempo, formato especial S5	Tipo de temporizador
STRING	Tipo de datos para cadena de caracteres	Especificación de tipo de datos STRING
STRUCT	Inicio de la especificación de una estructura, seguido de la lista de componentes	Especificación de tipo de datos STRUCT
THEN	Inicio de acciones consecutivas cuando se cumple una condición	Instrucción IF
TIME	Tipo de datos simple para indicaciones de tiempo	Tipo de tiempo
TIMER	Tipo de datos para temporizador, sólo utilizable en bloque de parámetros	Especificación de tipo de datos de parámetro
TIME_OF_DAY	Tipo de datos simple para hora del día	Tipo de tiempo
TO	Inicio del valor final	Instrucción FOR
TOD	Tipo de datos simple para hora del día	Tipo de temporizador
TRUE	Constante booleana predefinida; condición lógica cumplida, valor distinto de 0	
TYPE	Inicio del UDT	Tipo de datos de usuario
VAR	Inicio de una tabla de declaración	Bloque de variables estáticas
VAR_TEMP	Inicio de una tabla de declaración	Bloque de variables temporales
UNTIL	Introducción de la condición de interrupción para la instrucción REPEAT	Instrucción REPEAT
VAR_INPUT	Inicio de una tabla de declaración	Bloque de parámetros
VAR_IN_OUT	Inicio de una tabla de declaración	Bloque de parámetros
VAR_OUTPUT	Inicio de una tabla de declaración	Bloque de parámetros
WHILE	Inicio de instrucción de control para procesamiento de bucles	Instrucción WHILE
WORD	Tipo de datos simple (PALABRA)	Tipo de datos de bits
VOID	Sin valor de respuesta en una llamada a función	Función
XOR	Operación lógica	Operación lógica básica

15.1.6 Identificadores de operando y palabras clave de bloques

Datos globales de sistema

En la siguiente tabla encontrará una lista en orden alfabético de la nemotécnica alemana de los identificadores de operandos S7-SCL con una descripción:

- Indicación del identificador de operando:

  Prefijo de memoria (A, E, M, PA, PE) o bloque de datos (D)

- Indicación del tamaño del elemento de datos:

  Prefijo de tamaño (opcional o B, D, W, X)

La nemotécnica representa una combinación entre el identificador del operando (prefijo de memoria o D de bloque de datos) y el prefijo de tamaño. Ambas son reglas léxicas. La tabla está clasificada según la nemotécnica alemana; además, se indica la nemotécnica inglesa.

Nemotécnica alemana	Nemotécnica inglesa	Prefijo de memoria o bloque de datos	Prefijo de tamaño
A	Q	Salida (a través de una imagen del proceso)	Bit
AB	QB	Salida (a través de una imagen del proceso)	Byte
AD	QD	Salida (a través de una imagen del proceso)	Doble palabra
AN	QW	Salida (a través de una imagen del proceso)	Palabra
AX	QX	Salida (a través de una imagen del proceso)	Bit
D	D	Bloque de datos	Bit
DB	DB	Bloque de datos	Byte
DD	DD	Bloque de datos	Doble palabra
DW	DW	Bloque de datos	Palabra
DX	DX	Bloque de datos	Bit
E	I	Entrada (a través de una imagen del proceso)	Bit
EB	IB	Entrada (a través de una imagen del proceso)	Byte
ED	ID	Entrada (a través de una imagen del proceso)	Doble palabra
EW	IW	Entrada (a través de una imagen del proceso)	Palabra
EX	IX	Entrada (a través de una imagen del proceso)	Bit
M	M	Marcas	Bit
MB	MB	Marcas	Byte
MD	MD	Marcas	Doble palabra
MW	MW	Marcas	Palabra
MX	MX	Marcas	Bit
PAB	PQB	Salida (periferia directa)	Byte
PAD	PQD	Salida (periferia directa)	Doble palabra
PAW	PQW	Salida (periferia directa)	Palabra
PEB	PIB	Entrada (periferia directa)	Byte
PED	PID	Entrada (periferia directa)	Doble palabra
PEW	PIW	Entrada (periferia directa)	Palabra

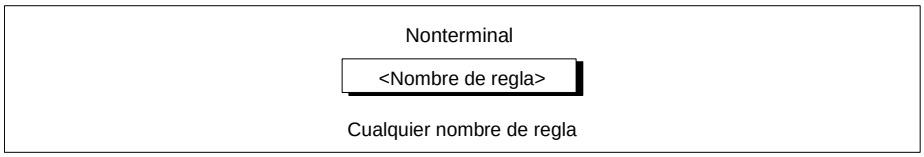
Palabras clave de bloque

Se utilizan para el direccionamiento absoluto de bloques. La tabla está clasificada según la nemotécnica alemana; además, se indica la nemotécnica inglesa.

Nemotécnica alemana	Nemotécnica inglesa	Prefijo de memoria o bloque de datos
DB	DB	Bloque de datos (Data Block)
FB	FB	Bloque de función (Function Block)
FC	FC	Función (Function)
OB	OB	Bloque de organización (Organization Block)
SDB	SDB	Bloque de datos del sistema (System Data Block)
SFC	SFC	Función del sistema (System Function)
SFB	SFB	Bloque de función del sistema (System Function Block)
T	T	Temporizador (Timer)
UDT	UDT	Tipo de datos globales o de usuario (Userdefined Data Type)
Z	C	Contador (Counter)

15.1.7 Sinopsis de no terminales

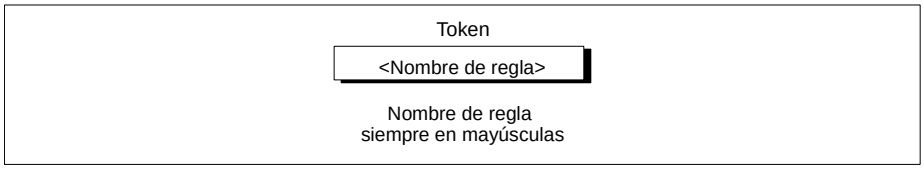
Non-terminal es un elemento compuesto descrito por medio de otra regla léxica o sintáctica. Un non-terminal se prepresenta por medio de un cajetín. El nombre del recuadro corresponde al nombre de la regla que se aplica a continuación.



El elemento aparece en las reglas léxicas y en las reglas sintácticas.

15.1.8 Sinopsis de token

Un token es un elemento compuesto que se utiliza en reglas sintáticas como elemento básico y que aparece explicado en las reglas léxicas. El token se representa mediante un recuadro. El NOMBRE (en mayúsculas) corresponde al nombre de la regla léxica que se aplica a continuación (sin recuadro)



Los token definidos representan identificadores que han sido obtenidos como resultado de la regla léxica. Estos token describen:

- IDENTIFICADORES
- la asignación de nombres en S7-SCL
- las constantes predefinidas y los OK flags

15.1.9 Identificadores

Identificadores

Los identificadores permiten acceder a los objetos del lenguaje S7-SCL. La siguiente tabla explica el significado de las distintas clases de identificadores.

Tipo de identificador	**Observaciones, ejemplos**
Palabras clave	p. ej. instrucciones de control `BEGIN,DO,WHILE`
Nombres predefinidos	Nombres de Tipos de datos estándar (p. ej. BOOL, BYTE, INT) Funciones estándar predefinidas, p. ej. ABS Constantes estándar TRUE y FALSE
Identificadores de operandos en identificadores absolutos	sistema y bloques de datos: p. ej. `E1.2, MW10, FC20, T5, DB30, DB10.D4.5`
Nombres de libre definición según la regla IDENTIFICADOR	Nombres de variables declaradas componentes de estructura parámetros constantes declaradas metas de salto
Símbolos de la tabla de símbolos	Cumplen la regla léxica IDENTIFICADOR o la regla léxica Símbolo, es decir, aparecen entre comillas, p. ej. "xyz"

Mayúsculas y minúsculas

En las palabras clave es irrelevante el uso de mayúsculas o de minúsculas, ya que la versión 4.0 de S7-SCL tampoco distingue entre mayúsculas y minúsculas cuando se trata de nombres predefinidos o nombres de libre elección, p. ej. para variables, así como de los símbolos de la tabla de símbolos. La siguiente tabla ofrece un resumen.

Tipo de identificador	**¿Distingue entre mayúsculas y minúsculas?**
Palabras clave	no
Nombres predefinidos en tipos de datos estándar	no
Nombres en funciones estándar predefinidas	no
Nombres predefinidos en constantes estándar	no
Identificadores de operandos en identificadores absolutos	no
Nombres de libre elección	no
Símbolos de la tabla de símbolos	no

Los nombres de las funciones estándar, p. ej. BYTE_TO_WORD y ABS, también se pueden escribir en minúsculas. Lo mismo ocurre con los parámetros de las funciones de temporización y contaje, p. ej. SV, se o ZV.

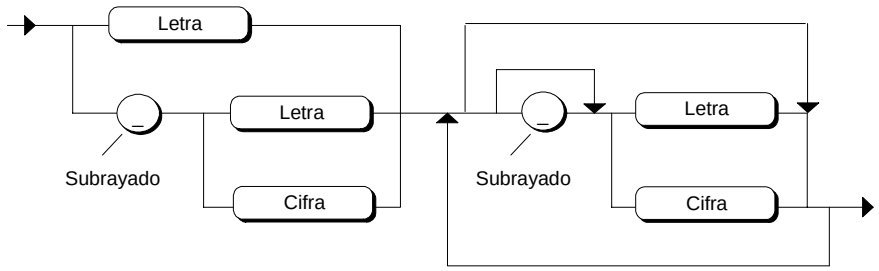
15.1.10 Asignación de nombres en S7-SCL

Asignación de nombres de libre elección

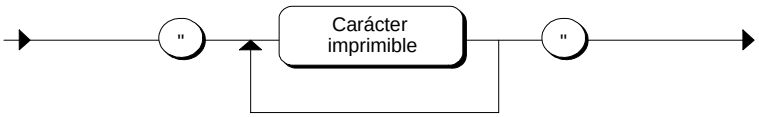
En general, existen dos posibilidades para asignar nombres:

- Es posible asignar nombres personalizados en S7-SCL. Estos nombres deben cumplir la regla IDENTIFICADOR. La regla IDENTIFICADOR se puede utilizar para cualquier nombre dentro de S7-SCL.
- También es posible introducir nombres con ayuda de la tabla de símbolos de STEP 7. En este caso, la regla es también IDENTIFICADOR, existiendo además la posibilidad del símbolo. Utilizando apóstrofos, el símbolo puede escribirse con caracteres imprimibles (p.ej., con espacios en blanco).

IDENTIFICADOR



SIMBOLO



Los símbolos se definen en la tabla de símbolos.

Reglas para la asignación de nombres

Tenga en cuenta que:

- al adjudicar nombres es mejor que seleccione nombres unívocos y autoexplicativos que contribuyan a la inteligibilidad del programa.
- asegúrese de que el nombre no haya sido asignado por el sistema, p. ej. mediante identificadores para tipos de datos o funciones estándar.
- rango de validez: en los nombres que tienen validez global, el rango de validez se extiende al programa completo. Los nombres válidos localmente sólo se aplican dentro de un bloque. Así es posible utilizar un mismo nombre en diferentes bloques. La siguiente tabla contiene información sobre las posibilidades disponibles.

Restricciones

Durante la asignación de nombres debe tener en cuenta una serie de restricciones.

Los nombres deben ser unívocos dentro de su rango de validez; es decir, los nombres que ya se han adjudicado dentro de un bloque no pueden utilizarse otra vez en el mismo bloque. Asimismo, tampoco pueden utilizarse los siguientes nombres ya asignados por el sistema:

- Nombres de palabras clave: p. ej. CONST, END_CONST, BEGIN
- Nombres de operaciones: p. ej. AND, XOR
- Nombres de identificadores predefinidos: p. ej. nombres de tipos de datos como BOOL, STRING, INT
- Nombres de las constantes predefinidas TRUE y FALSE
- Nombres de funciones estándar: p. ej. ABS, ACOS, ASIN, COS, LN
- Nombres de identificadores absolutos o de operandos para datos del sistema globales: p. ej. EB, EW, ED, AB, AW, AD MB, MD

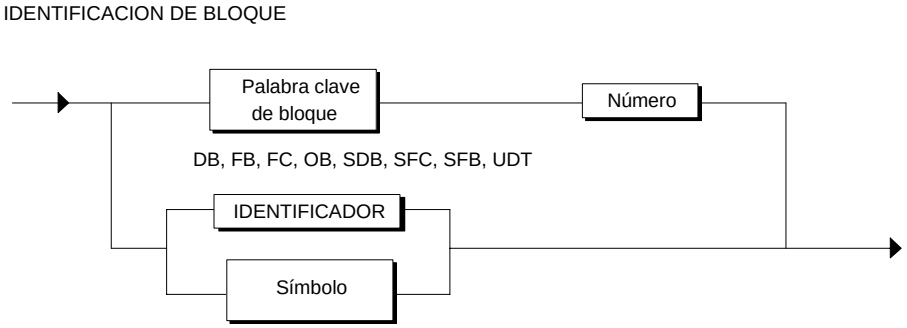
Utilización de IDENTIFICADOR

En la siguiente tabla se muestra en qué casos puede asignar nombres que cumplan la regla IDENTIFICADOR.

IDENTIFICADOR	Descripción	Regla
Nombre de bloque	Nombre simbólico para bloque	DESIGNACIÓN DE BLOQUE, llamada de función
Nombre de temporizador y contador	Nombre simbólico para temporizador y contador	DESIGNACIÓN DE TEMPORIZADOR, DESIGNACIÓN DE CONTADOR
Nombre de atributo	Nombre para un atributo	Asignación de atributo
Secuencia de caracteres	Declaración de constantes simbólicas, utilización	Bloque de constantes, Constante
Meta de salto	Declaración de meta de salto, utilización de meta de salto	Área de instrucciones del bloque de metas de salto, instrucción GOTO
Nombre de variable	Declaración de variables temporales o estáticas	Declaración de variables, variable simple, Variable estructurada
Declaración de instancia	Instancia de declaración local	Declaración de instancia, nombre de llamada FB

IDENTIFICADOR DE BLOQUE

En la regla IDENTIFICADOR DE BLOQUE puede introducir IDENTIFICADOR y Símbolo de forma alternativa:



Las reglas IDENTIFICADOR DE TEMPORIZADOR E IDENTIFICADOR DE CONTADOR funcionan de forma análoga a la regla IDENTIFICADOR DE BLOQUE.

15.1.11 Constantes predefinidas y OK flags

Ambas tablas son válidas tanto para la nemotécnica alemana como para la inglesa.

Constantes:

Nemotécnica	Descripción
FALSE	Constante booleana predefinida (constante estándar) con el valor 0. Tiene el significado lógico de que no se ha cumplido una condición.
TRUE	Constante booleana predefinida (constante estándar) con el valor 1. Tiene el significado lógico de que se ha cumplido una condición.

Marcas

Nemotécnica	Descripción
EN	Flag para la habilitación del bloque
ENO	Marca de error del bloque
OK	Marca que se pone a FALSE si una instrucción se ha ejecutado con errores

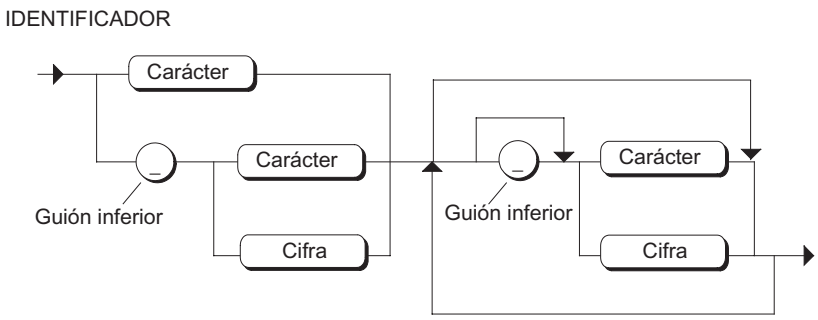
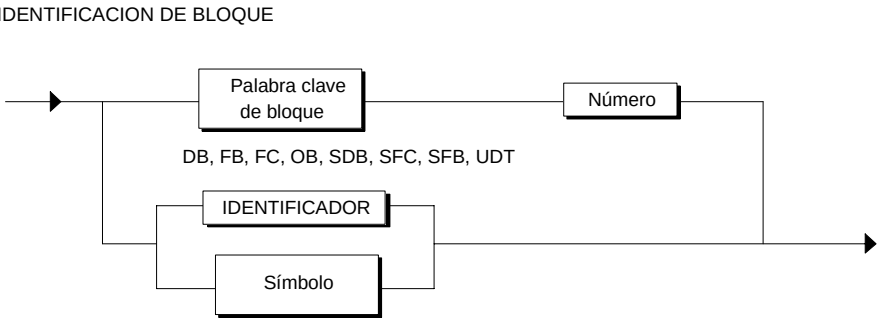
15.2 Reglas léxicas

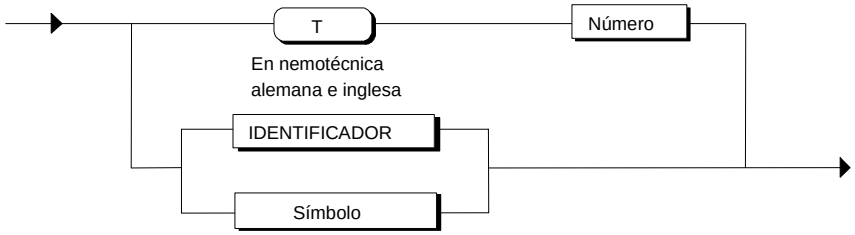
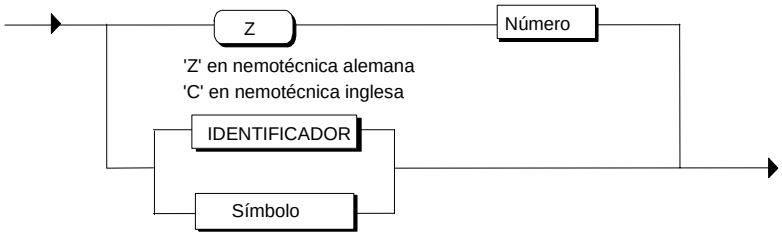
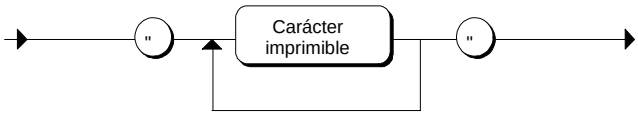
15.2.1 Sinopsis de las reglas léxicas

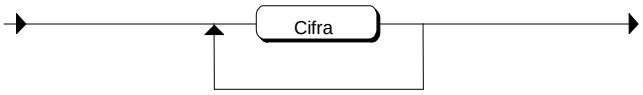
Las reglas léxicas determinan la estructura de los elementos (token) que se van a ejecutar durante el análisis léxico del compilador. Por ello, en la escritura no hay libertad de formato y deben respetarse estrictamente las reglas. En particular esto significa que:

- no está permitida la inserción de signos de formateado;
- no se pueden introducir líneas ni bloques de comentarios,
- no pueden insertarse atributos para los identificadores.

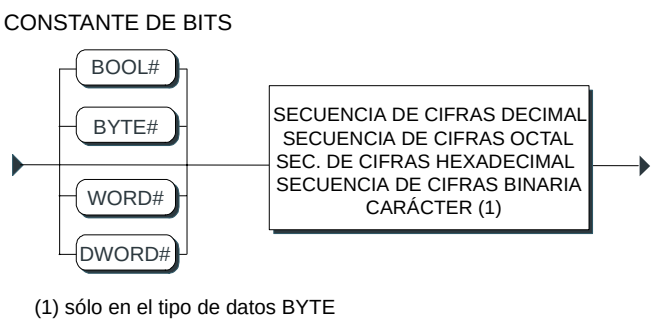
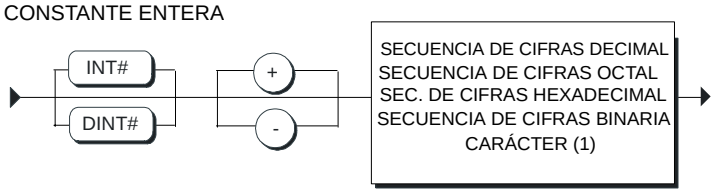
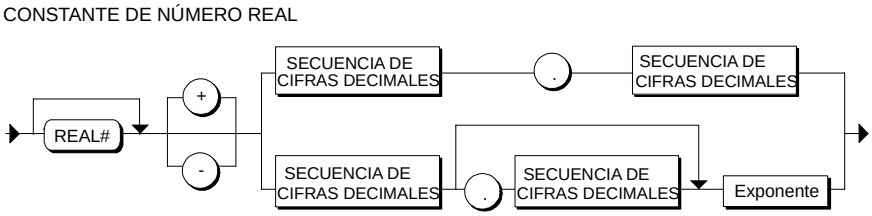
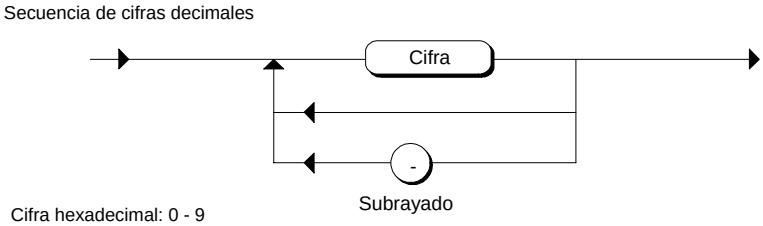
15.2.2 Identificadores

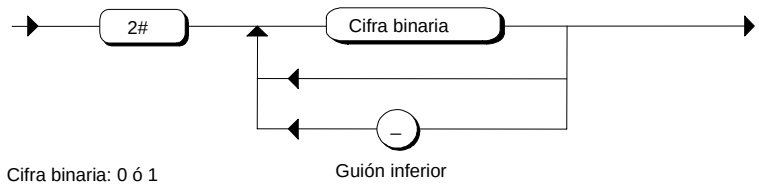
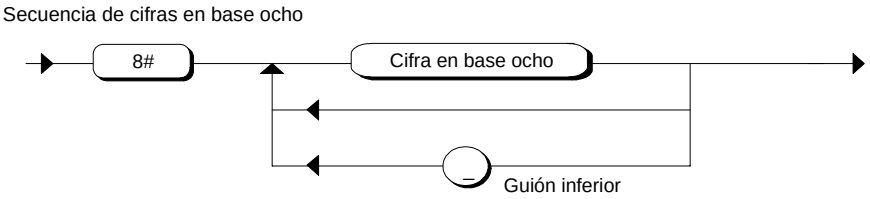
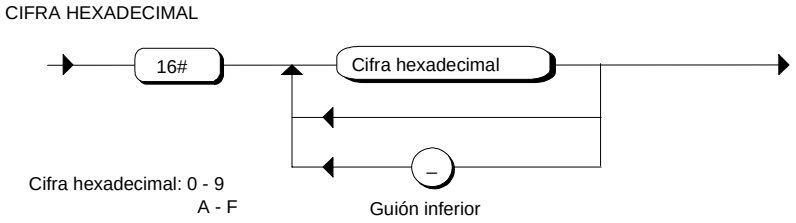
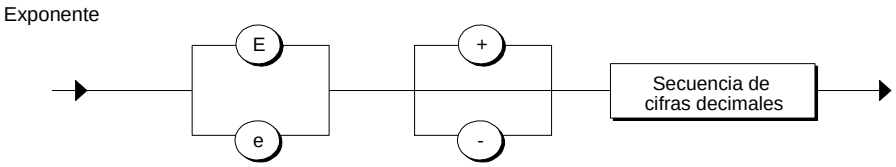
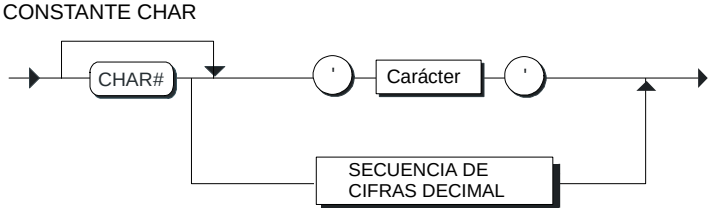
Regla	**Diagrama sintáctico**
Identificadores	
Identificador de BLOQUE	

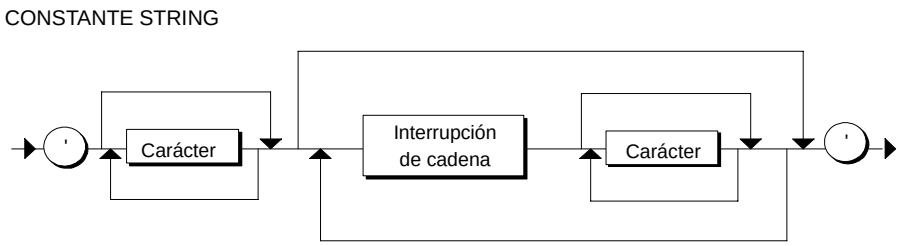
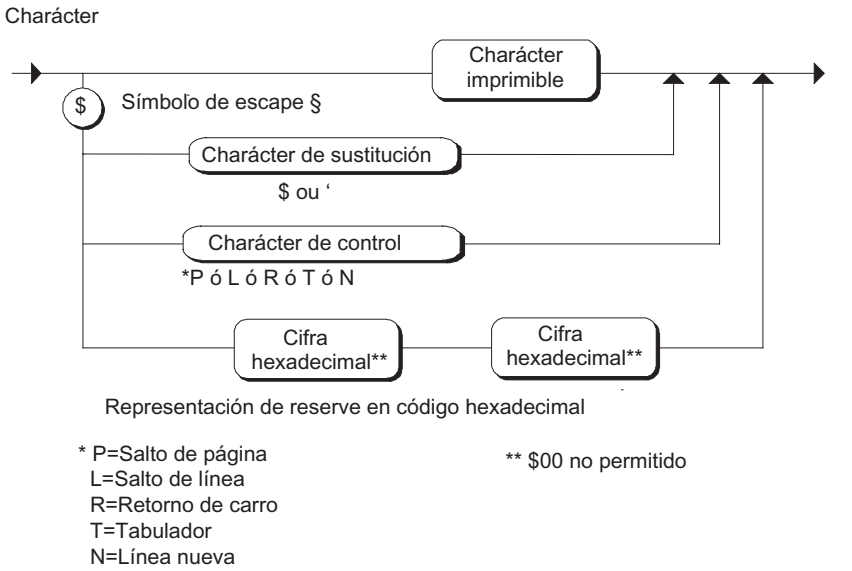
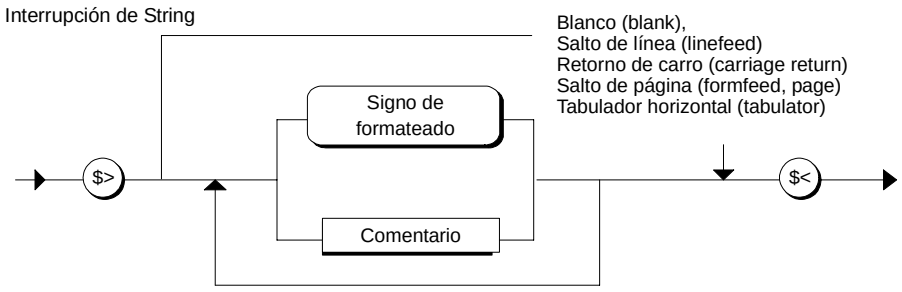
Regla	**Diagrama sintáctico**
Identificador de temporizador	
Identificador de contador	
Palabra clave Bloque	
Símbolo	

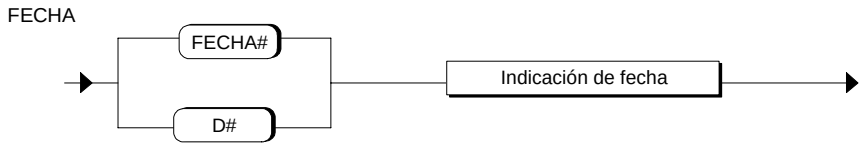
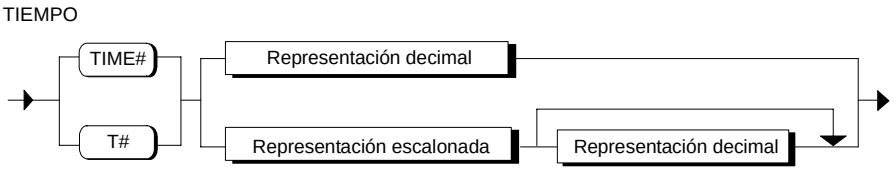
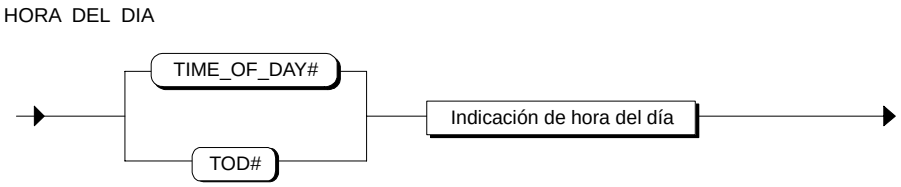
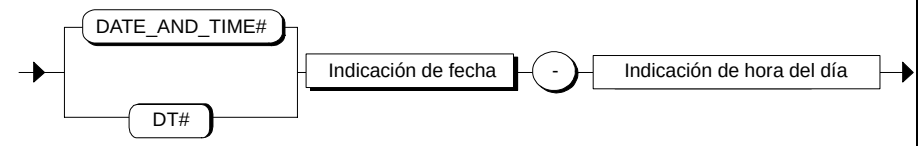
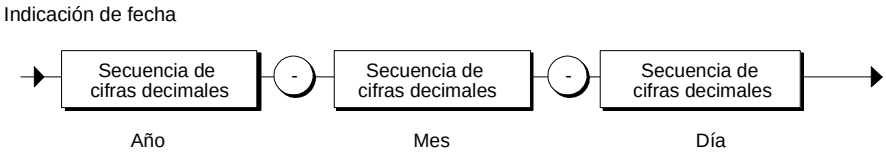
Regla	**Diagrama sintáctico**
Número	

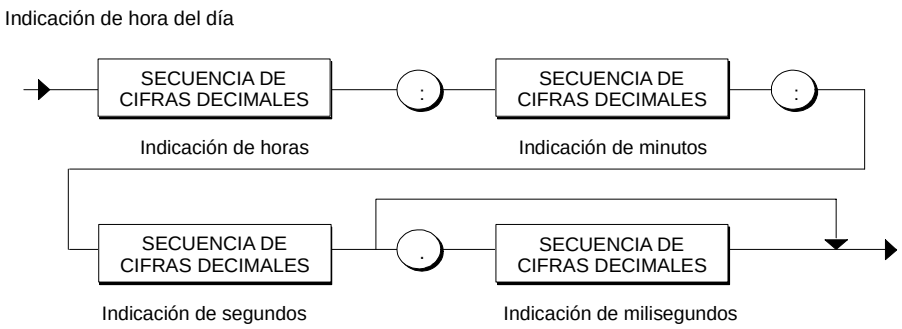
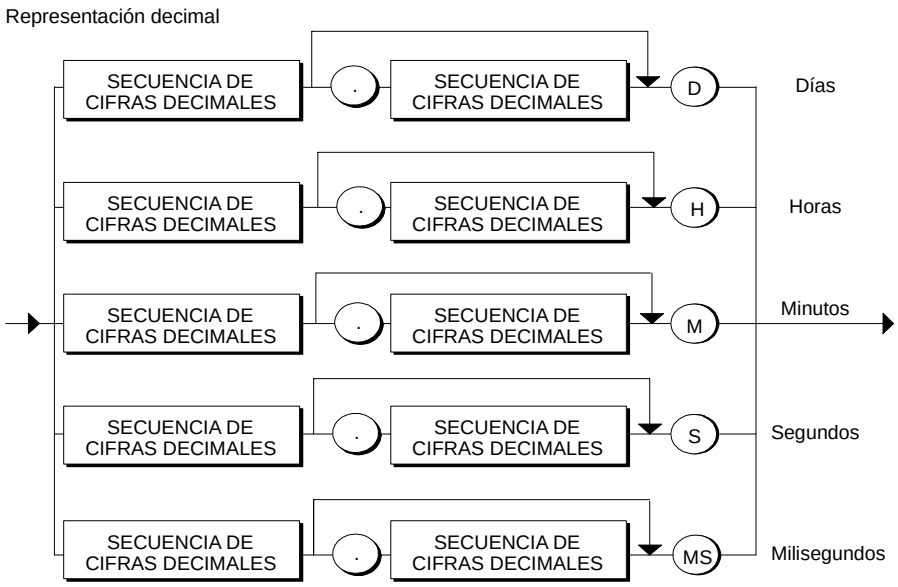
15.2.3 Constantes

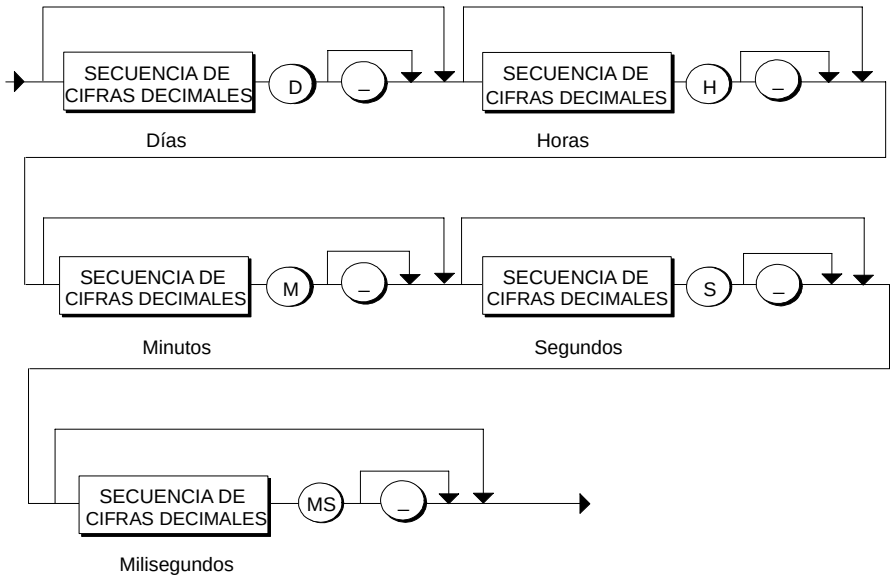
Regla	Diagrama sintáctico
Constante de bits	 (1) sólo en el tipo de datos BYTE
Constante entera	 (1) sólo en el tipo de datos INT
Constante de número real	
Secuencia de cifras decimales	 Cifra hexadecimal: 0 - 9

Regla	**Diagrama sintáctico**
Secuencia de cifras binarias	
Secuencia de cifras octales	
Secuencia de cifras hexadecimales	
Exponente	
Constante Character	

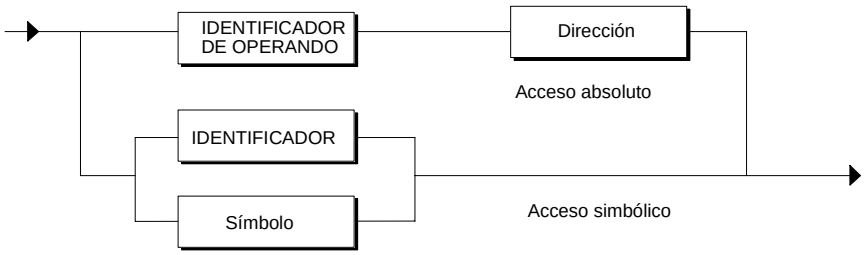
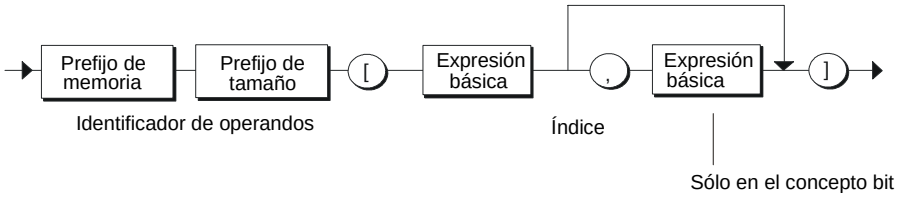
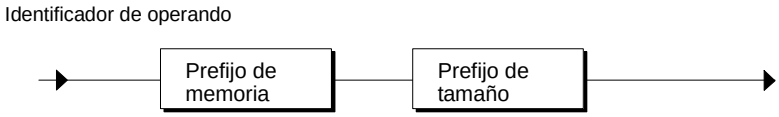
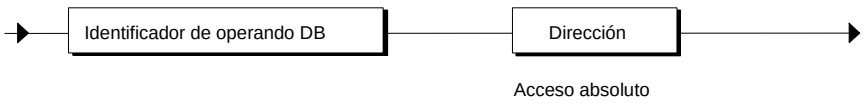
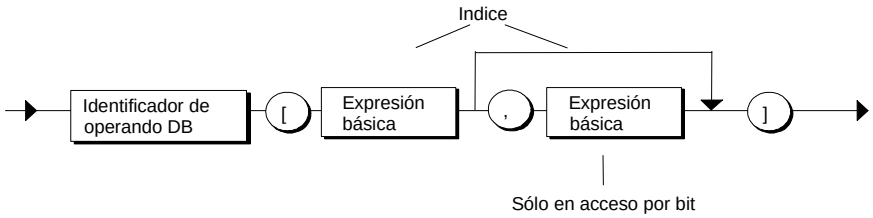
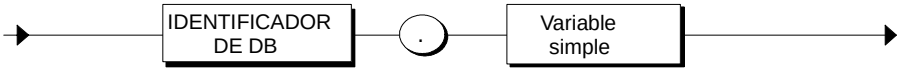
Regla	**Diagrama sintáctico**
Constante STRING	
Carácter	
Interrupción de String	

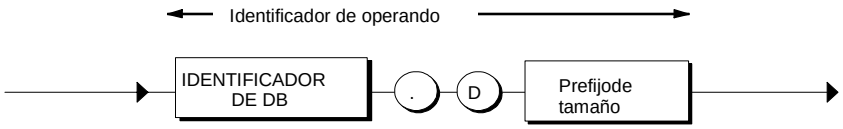
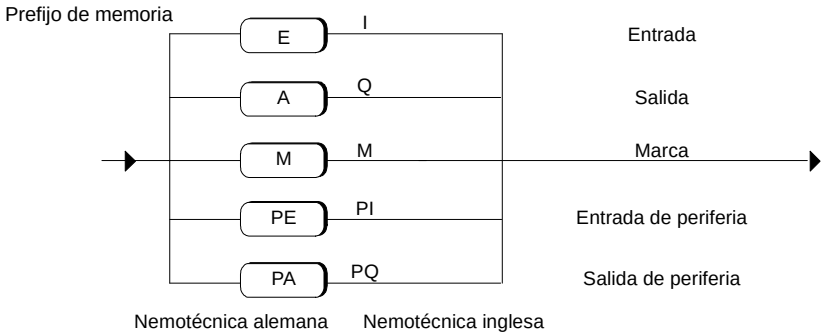
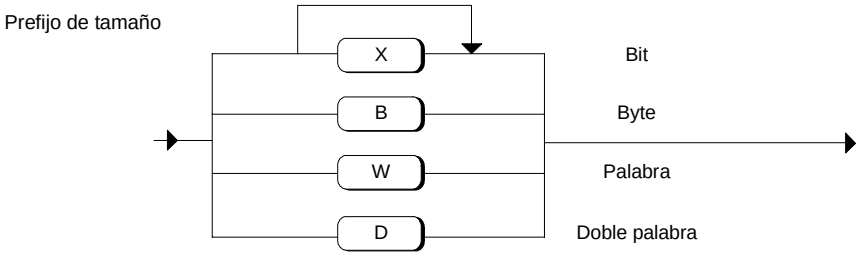
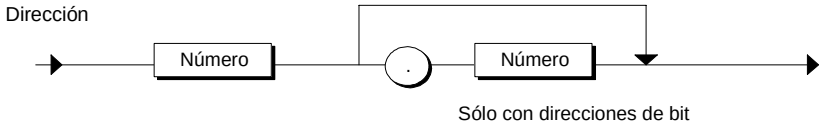
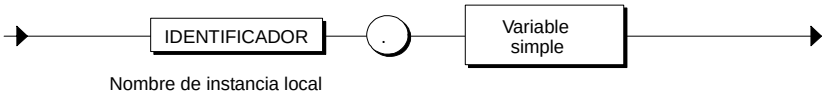
Regla	Diagrama sintáctico
Fecha	
Intervalo	 - Cada unidad de tiempo (p.ej.: horas, minutos) sólo puede indicarse una vez. - Debe respetarse la secuencia: días, horas, minutos, segundos, milisegundos.
Hora del día	
Fecha y hora	
Indicación de la fecha	

Regla	Diagrama sintáctico
Indicación de la hora	
Representación decimal	 El acceso a la representación decimal sólo es posible con unidades de tiempo no definidas todavía.

Regla	**Diagrama sintáctico**
Representación escalonada	Representación escalonada 

15.2.4 Direccionamiento absoluto

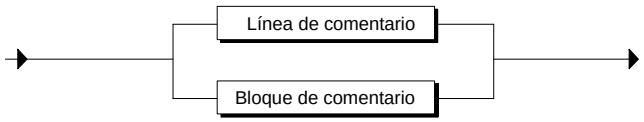
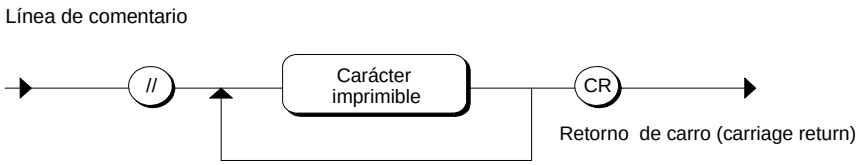
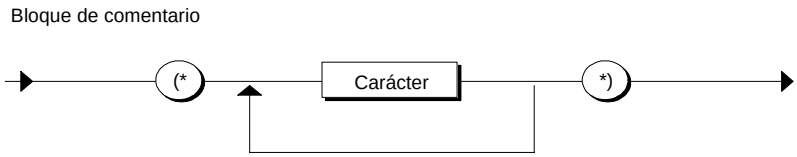
Regla	Diagrama sintáctico
Acceso a memoria simple	
Acceso a memoria indexado	
Identificador de operandos para memoria	
Acceso a DB absoluto	
Acceso a DB indexado	
Acceso a DB configurado	

Regla	Diagrama sintáctico
Identificador de operando DB	
Prefijo de memoria	
Prefijo de tamaño para memoria y DB	
Dirección para memoria y DB	
Acceso a instancia local	

15.2.5 Comentarios

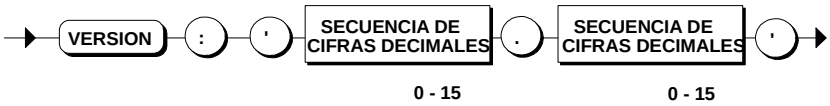
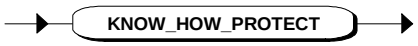
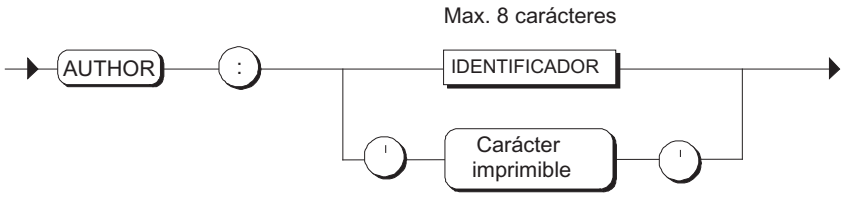
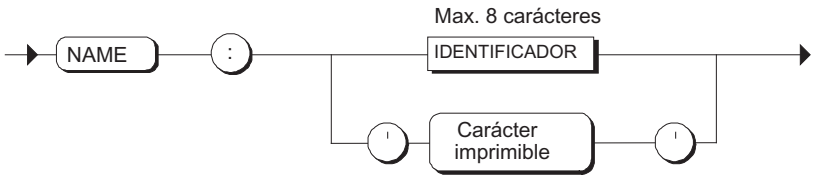
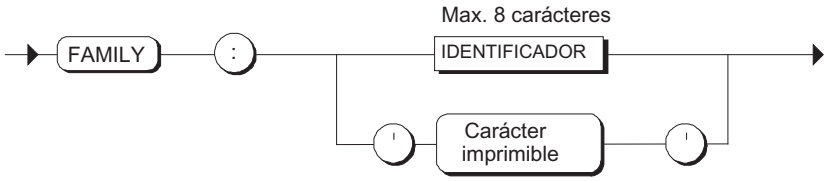
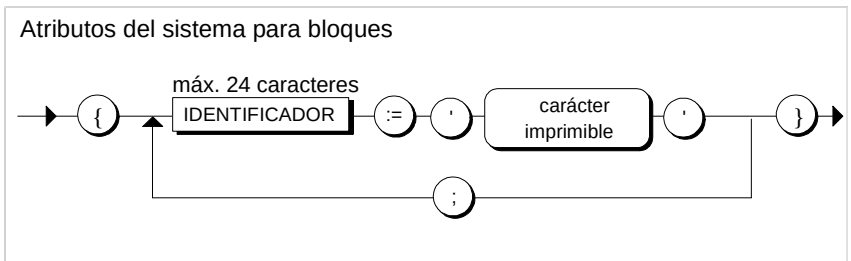
Los factores más importantes que hay que tener en cuenta para la elaboración de comentarios son:

- Se admite la anidado de comentarios cuando está activada la opción "Admitir comentarios anidados".
- Es posible insertarlos en cualquier punto de las reglas sintácticas, pero no en las reglas léxicas.

Regla	**Diagrama sintáctico**
Comentario	
Línea de comentario	
Bloque de comentario	

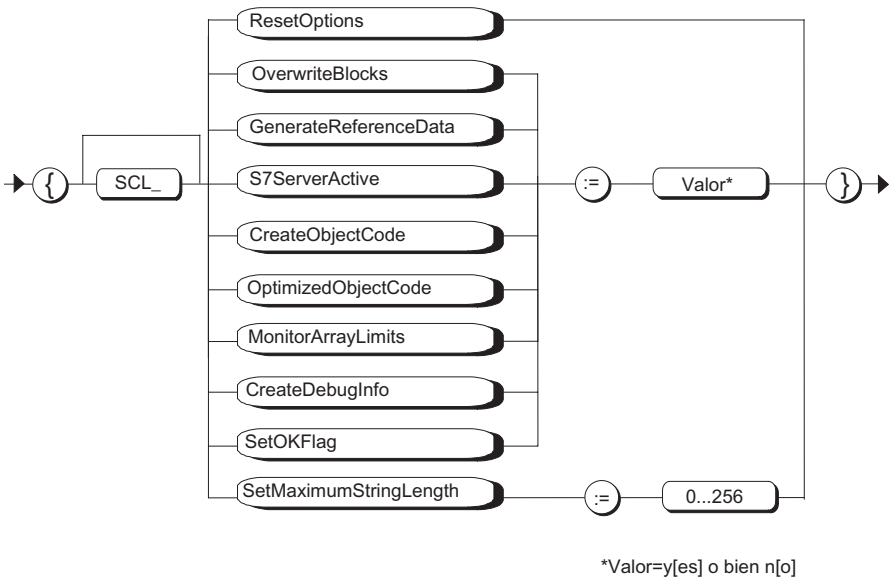
15.2.6 Atributos de bloques

Los atributos de bloque pueden aparecer con la siguiente sintaxis detrás del IDENTIFICADOR DE BLOQUE y delante de la declaración del primer bloque de variables o parámetros.

Regla	Diagrama sintáctico
Título	
Versión	
Protección de bloque	
Autor	
Nombre del parámetro	
Familia de bloques	
Atributos de sistema para bloques	

15.2.7 Opciones de compilador

Las opciones de compilador figuran en una fila propia de la fuente, fuera de los límites de bloques. No se distingue entre mayúsculas y minúsculas.

Regla	Diagrama de sintaxis
Opción de compilador	

15.3 Reglas sintácticas

15.3.1 Sinopsis de las reglas sintácticas

Partiendo de las reglas léxicas, en las reglas sintácticas se describe la estructura de S7-SCL. En el contexto de estas reglas, puede crear su programa S7-SCL sin formatear:

Cada regla va precedido de un nombre de regla. Cuando la regla se utiliza en una regla superior, su nombre aparece encerrado en un recuadro.

Si el nombre del recuadro está escrito con mayúsculas, se trata de un token, descrito en las reglas léxicas.

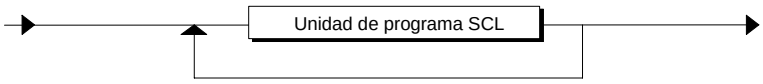
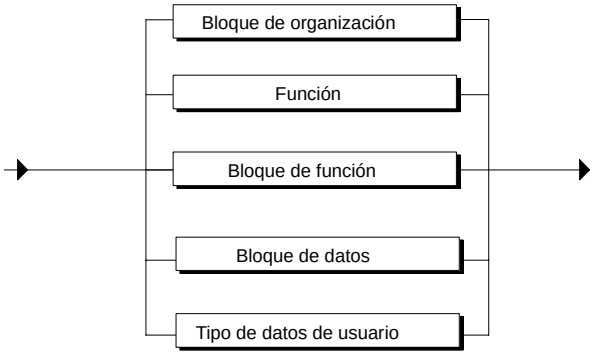
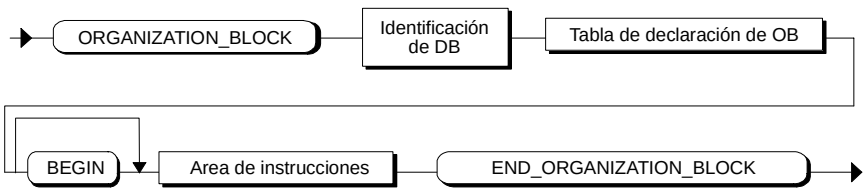
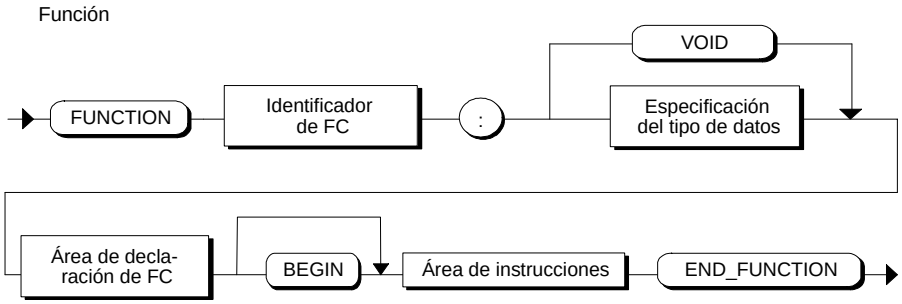
Para más información sobre los nombres de reglas que aparecen en recuadros o círculos consulte el capítulo "Descripción del lenguaje formal".

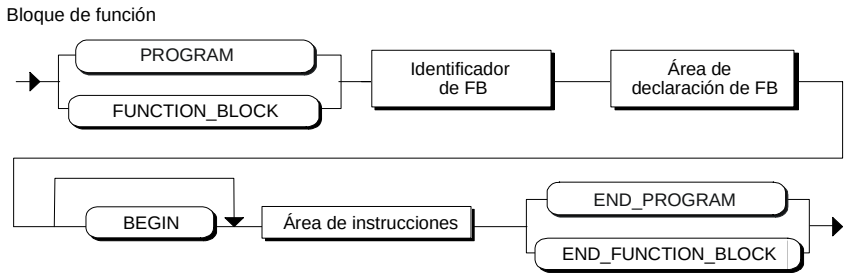
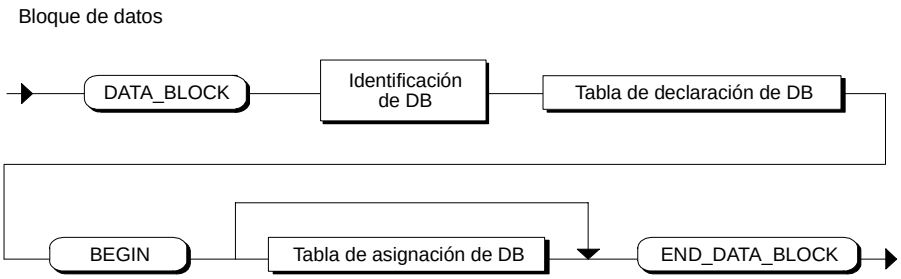
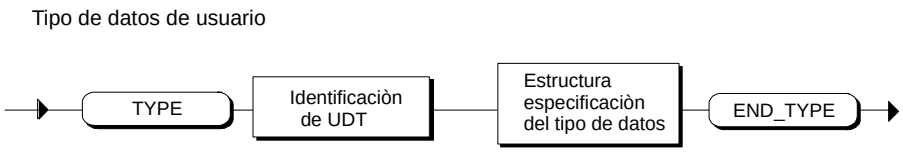
Libertad de formato

Libertad de formato quiere decir que:

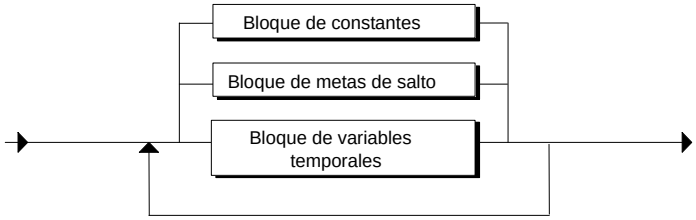
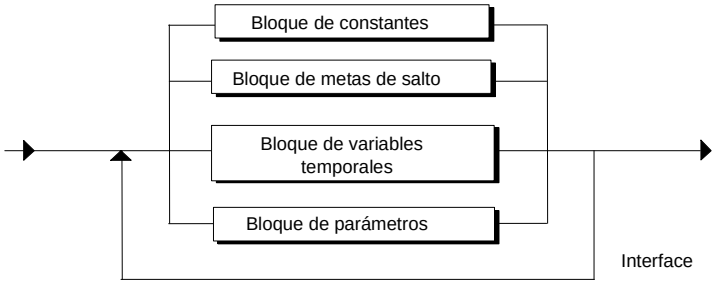
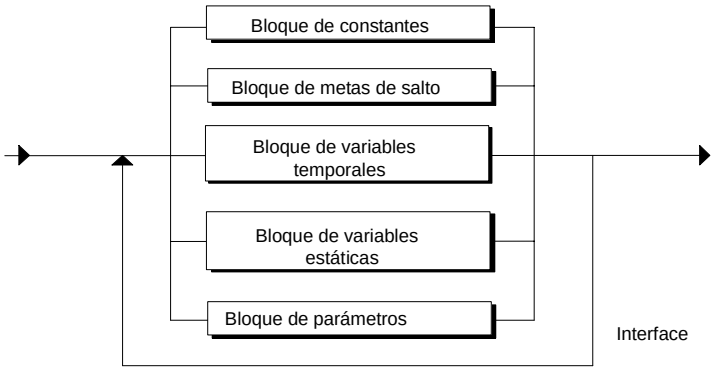
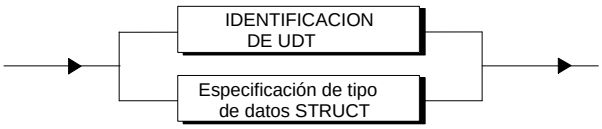
- es posible insertar caracteres de formateo en cualquier punto;
- se pueden insertar comentarios de línea y bloques de comentarios

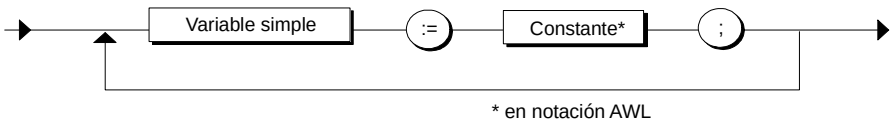
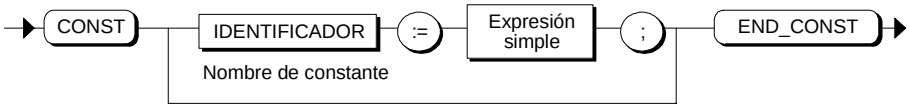
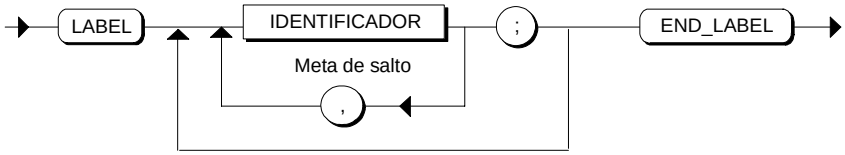
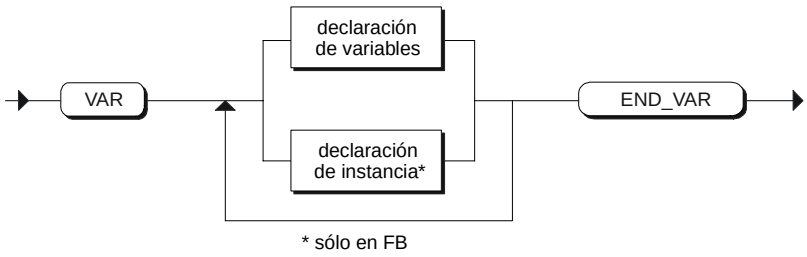
15.3.2 Segmentación de fuentes S7-SCL

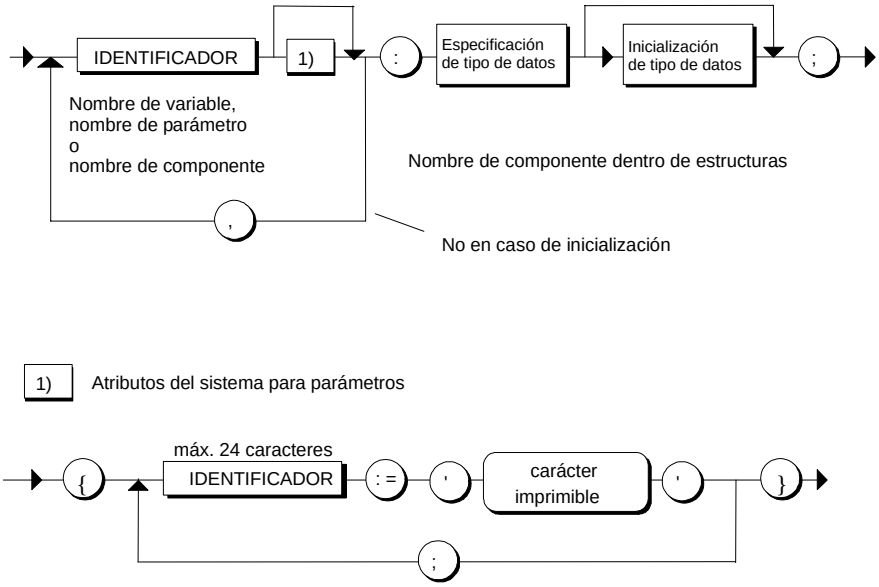
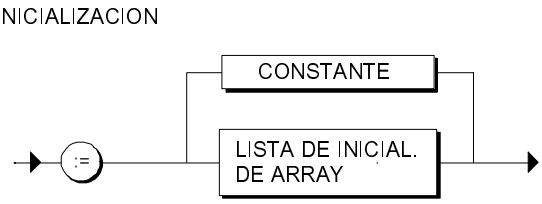
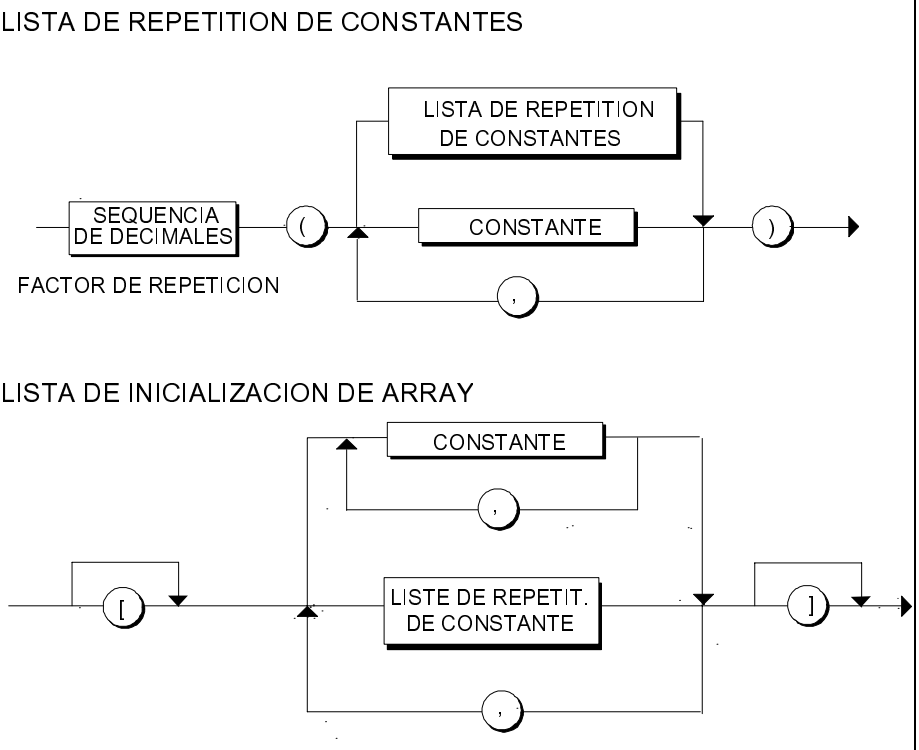
Regla	Diagrama sintáctico
Programa S7-SCL	
Unidad de programa S7-SCL	
Bloque de organización	Bloque de organización 
Función Tenga en cuenta que en las funciones sin VOID en el área de instrucciones hay que asignar el valor de retorno al nombre de la función	Función 

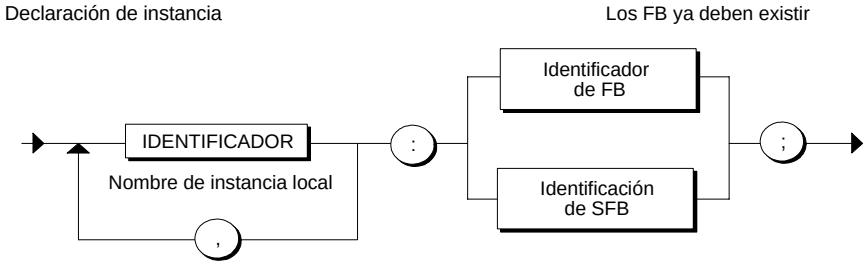
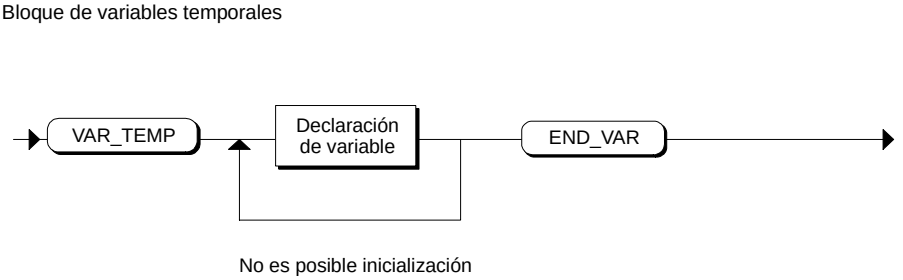
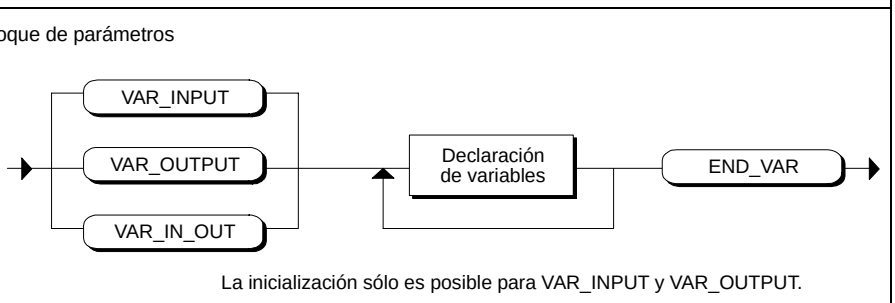
Regla	**Diagrama sintáctico**
Bloque de función	
Bloque de datos	
Tipo de datos de usuario	

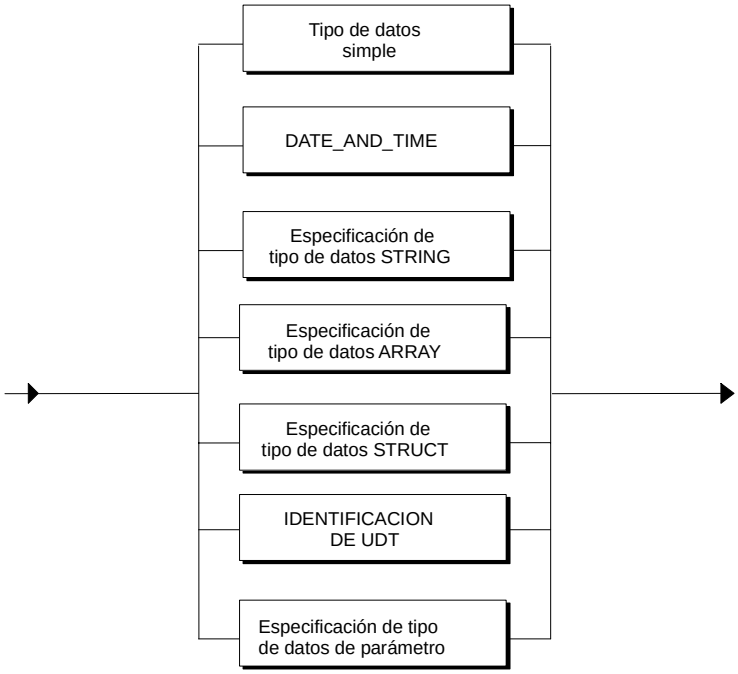
15.3.3 Estructura de las áreas de declaración

Regla	Diagrama sintáctico
Área de declaración de OB	
Área de declaración de una FC	
Area de declaración de FB	
Área de declaración de un DB	

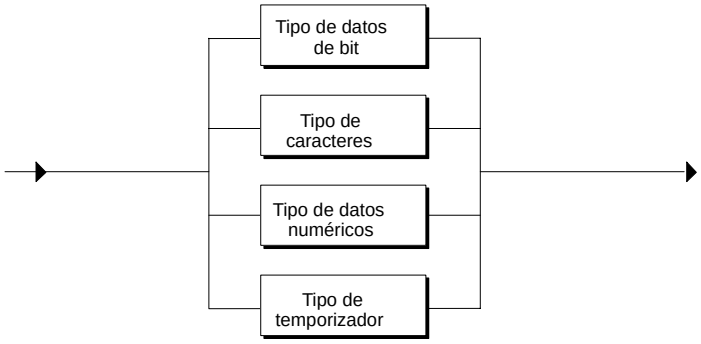
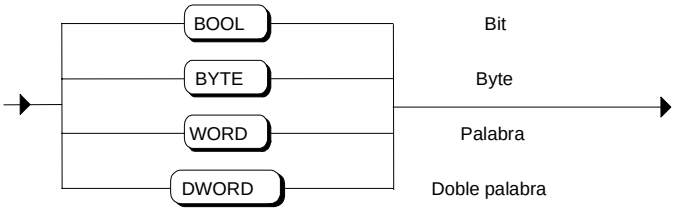
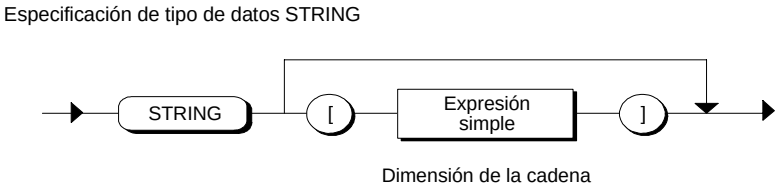
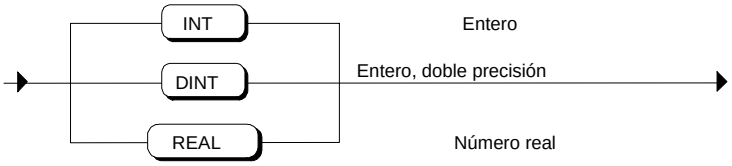
Regla	**Diagrama sintáctico**
Área de asignación de DB	Área de asignación de DB 
Bloque de constantes	Bloque de constantes 
Bloque de metas de salto	Bloque de metas de salto 
Bloque de variables estáticas	Bloque de variables estáticas 

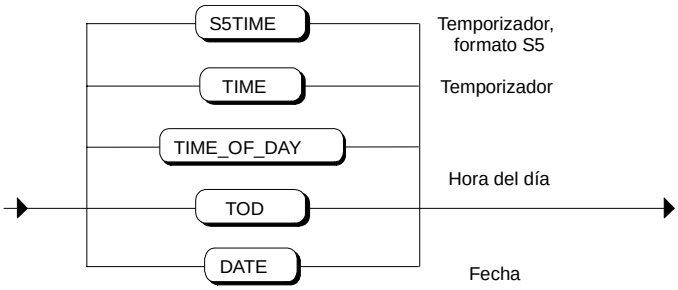
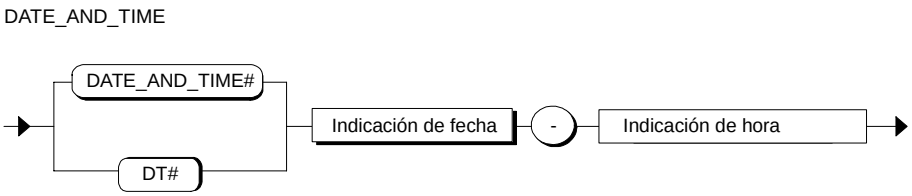
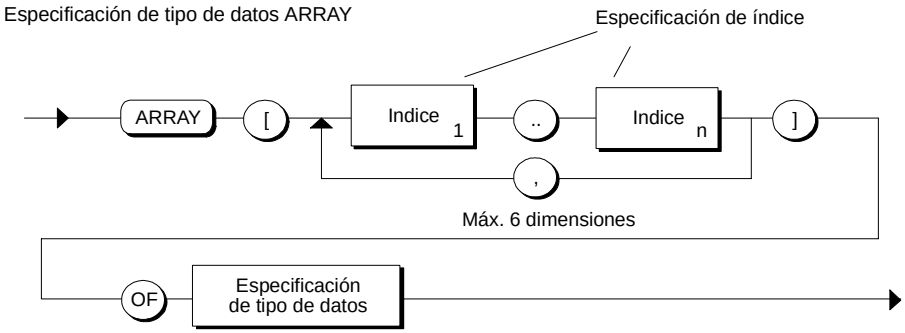
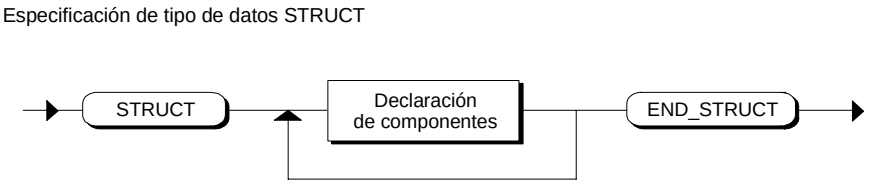
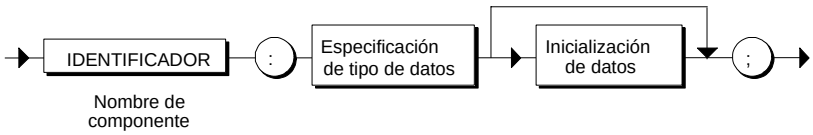
Regla	Diagrama sintáctico
Declaración de variables	
Inicialización del tipo de datos	
Lista de inicialización de arrays	

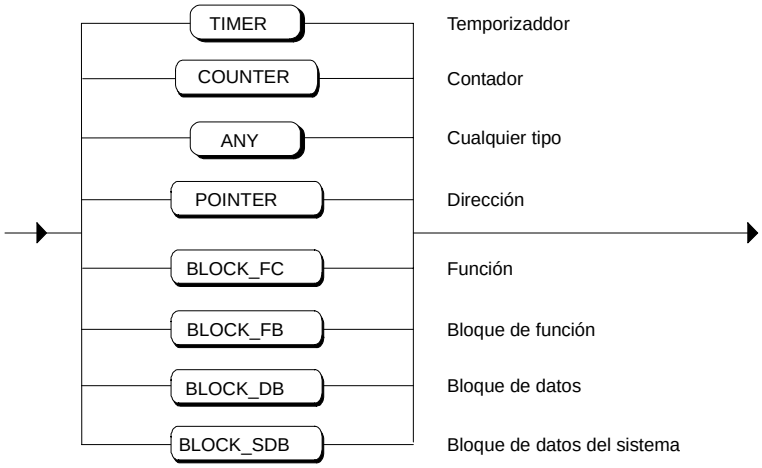
Regla	**Diagrama sintáctico**
Declaración de instancia (sólo es posible en el apartado VAR de un FB)	
Bloque de variables temporales	
Bloque de parámetros	

Regla	**Diagrama sintáctico**
Especificación del tipo de datos	

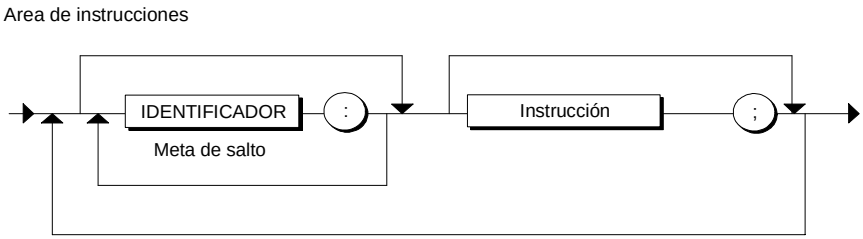
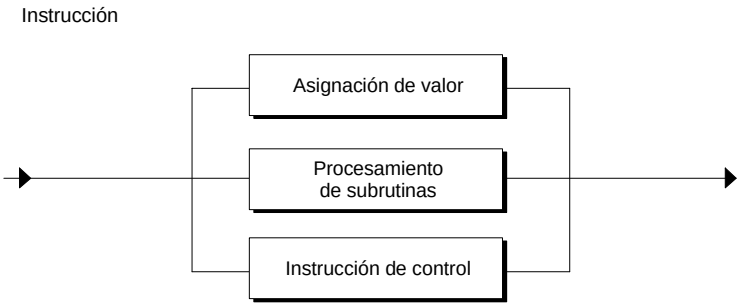
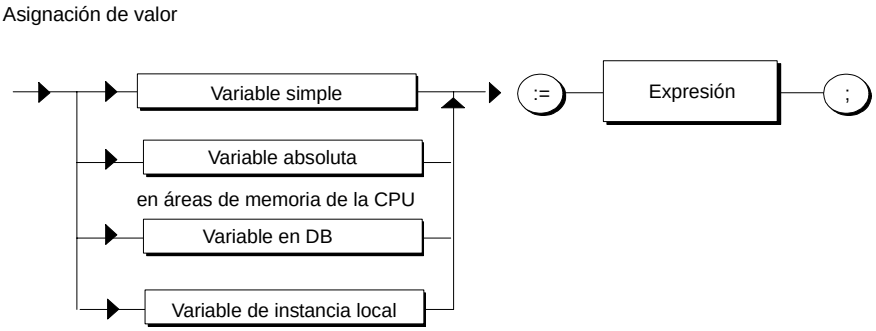
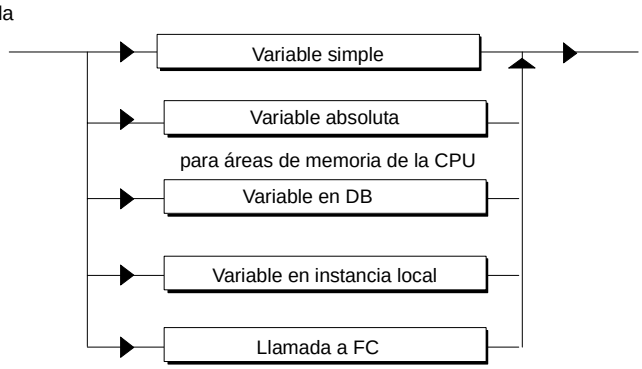
15.3.4 Tipos de datos de S7-SCL

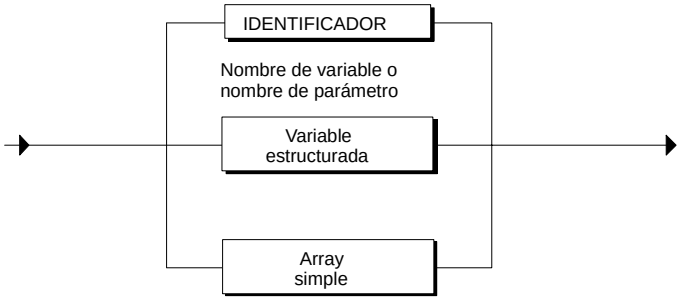
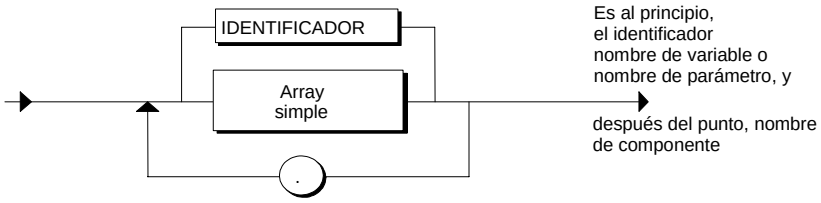
Regla	Diagrama sintáctico
Tipo de datos simple	
Tipo de datos de bits	
Tipo de datos Carácter	
Especificación del tipo de datos STRING	
Tipo de datos numérico	

Regla	**Diagrama sintáctico**
Tipo de tiempo	
DATE_AND_TIME	
Especificación del tipo de datos ARRAY	
Especificación del tipo de datos STRUCT No olvide escribir un punto y coma detrás de la palabra clave END_STRUCT.	
Declaración de componentes	

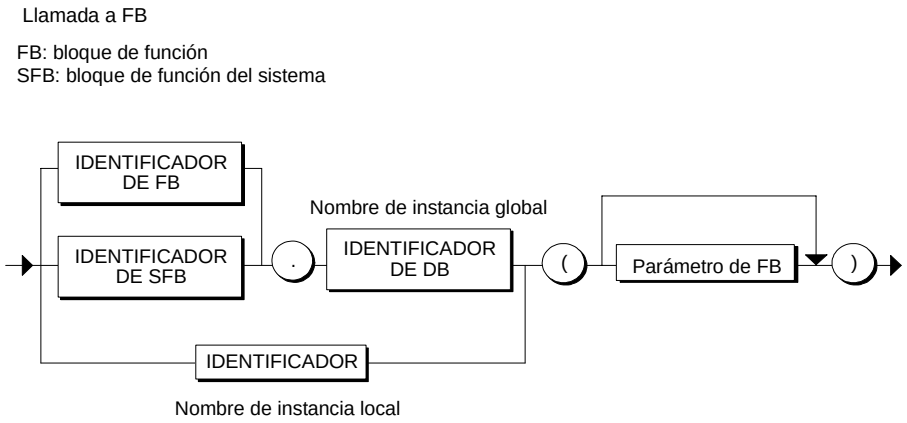
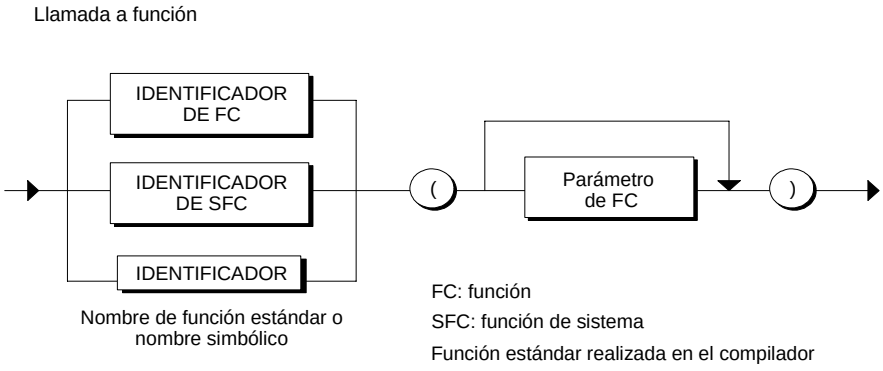
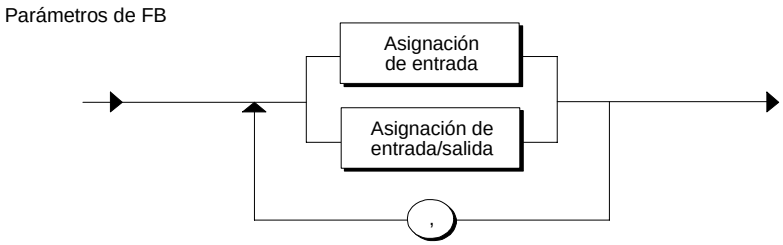
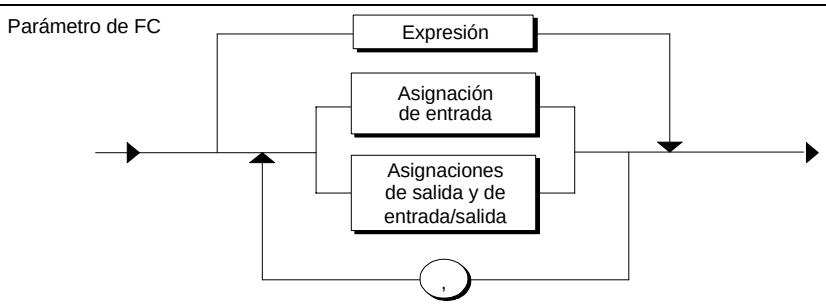
Regla	**Diagrama sintáctico**
Especificación del tipo de parámetro	

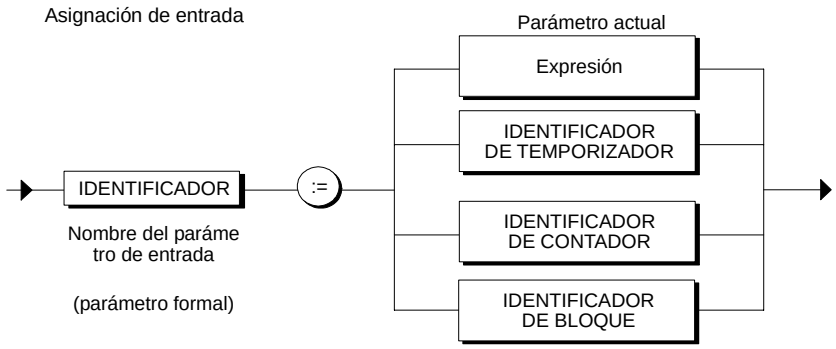
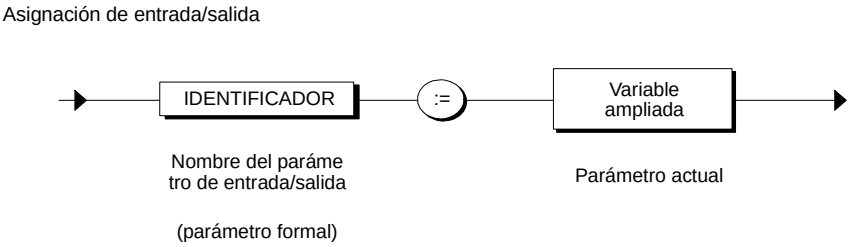
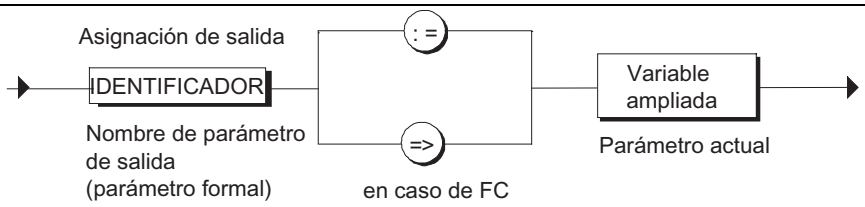
15.3.5 Área de instrucciones

Regla	Diagrama sintáctico
Área de instrucciones	
Instrucción	
Asignación de valor	
Variable ampliada	

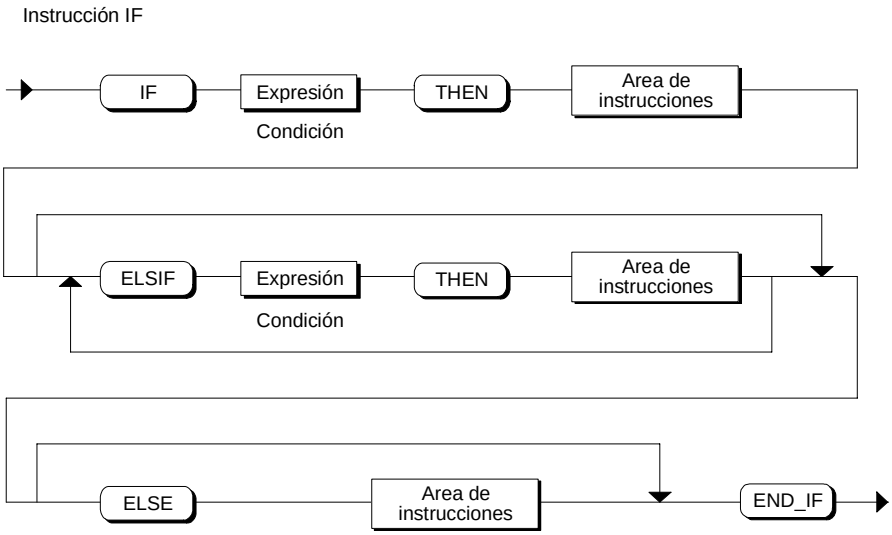
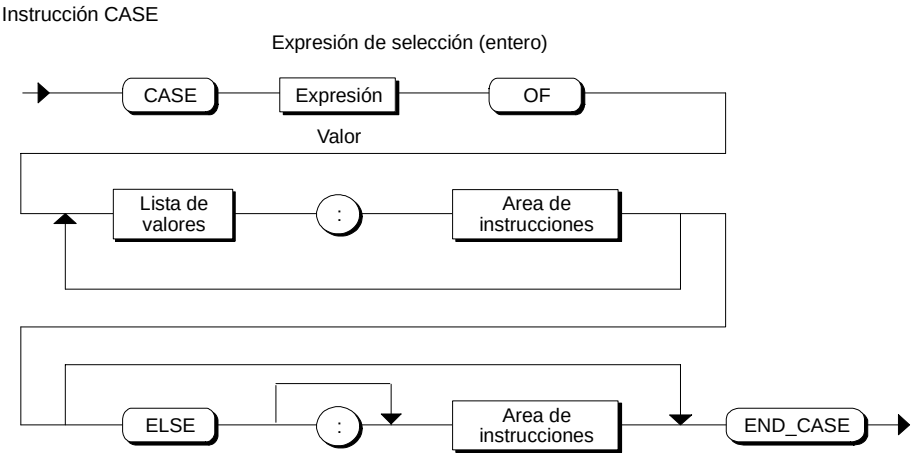
Regla	**Diagrama sintáctico**
Variable simple	
Variable estructurada	

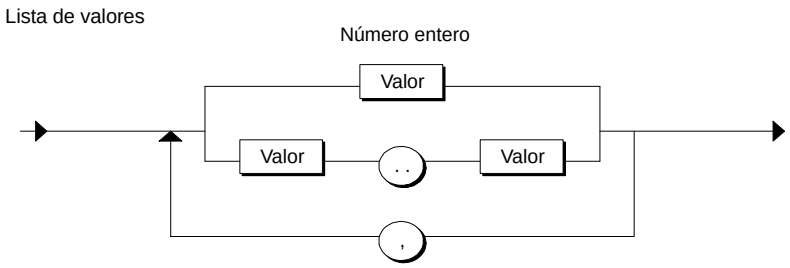
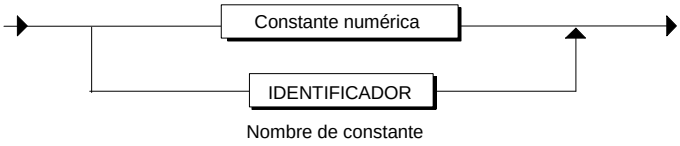
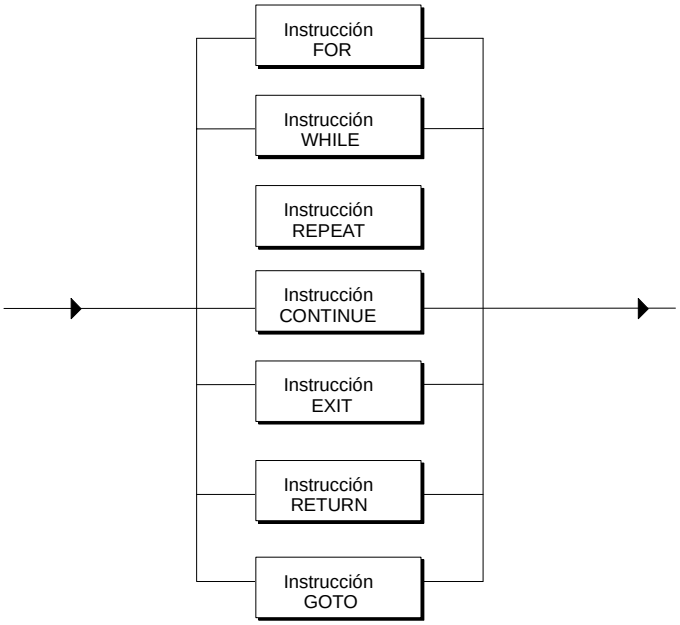
15.3.6 Llamada a funciones y bloques de función

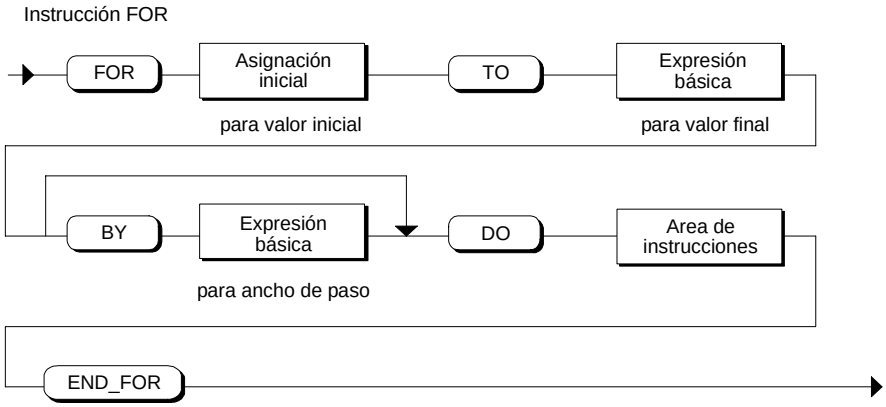
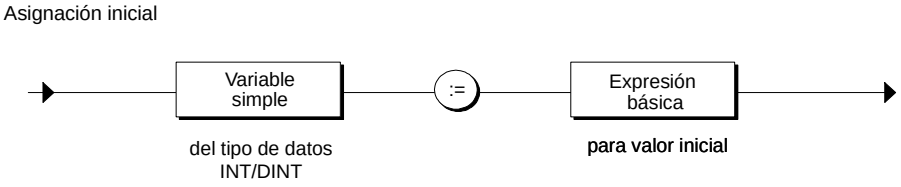
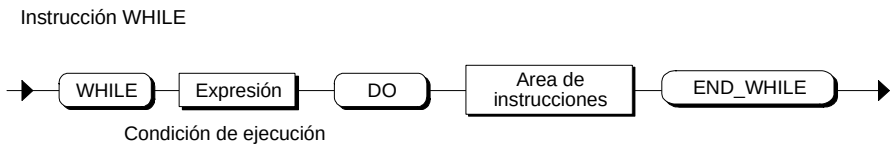
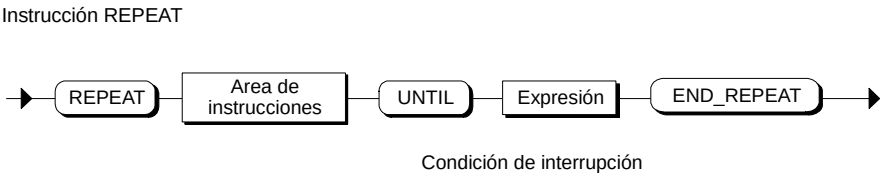
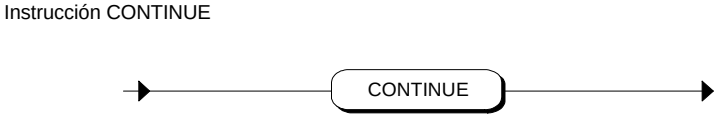
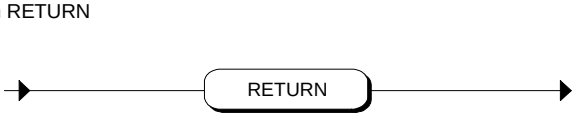
Regla	Diagrama sintáctico
Llamada a FB	
Llamada a función	
Parámetros de FB	
Parámetros de FC	

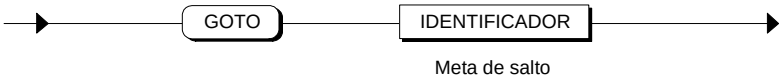
Regla	**Diagrama sintáctico**
Asignación de entrada	
Asignación de de entrada/salida	
Asignación de salida	

15.3.7 Instrucciones de control

Regla	Diagrama sintáctico
Instrucción IF No olvide escribir un punto y coma detrás de la palabra clave END_IF.	
Instrucción CASE No olvide escribir un punto y coma detrás de la palabra clave END_CASE.	

Regla	**Diagrama sintáctico**
Lista de valores	
Valor	
Intrucciones de repetición de de salto	

Regla	**Diagrama sintáctico**
Instrucción FOR No olvide escribir un punto y coma detrás de la palabra clave END_FOR.	
Asignación inicial	
Instrucción WHILE No olvide escribir un punto y coma detrás de la palabra clave END_WHILE.	
Instrucción REPEAT No olvide escribir un punto y coma detrás de la palabra clave END_REPEAT.	
Instrucción CONTINUE	
Instrucción RETURN	

Regla	**Diagrama sintáctico**
Instrucción EXIT	Instrucción EXIT 
Salto en el programa	Instrucción GOTO  Meta de salto

16 Consejos y trucos

División de dos valores enteros con resultado en formato REAL

Programe la siguiente instrucción en SCL:

```
Fraction:=Dividend/Divisor;
```

donde fraction es un valor real y el `dividendo` y el `divisor` son valores enteros.

Tenga en cuenta que, con operaciones de este tipo, el compilador S7-SCL ejecuta una conversión del tipo de datos implícita y, de esta forma, compila la instrucción anterior de este modo:

```
Fraction:=INT_TO_REAL(Dividend/Divisor);
```

De esto se deduce que la división siempre arroja como resultado un valor redondeado, p.ej. 1/3 = 0 o 3/2 = 1.

Código con tiempo de ejecución optimizado para el acceso a estructuras de bloques de datos

Si tiene que acceder a una estructura de un bloque de datos en más de una ocasión, es recomendable que utilice el siguiente procedimiento:

1. Cree una variable local con el tipo de la configuración.
2. Asigne a la variable la estructura del bloque de datos.
3. A partir de entonces podrá utilizar la variable en el código tantas veces como desee sin tener que volver a acceder al DB.

Ejemplo:

```
DB100.array[i].value :=
DB100.array[i].valor1  * DB100.array[i].valor2 /
DB100.array[i].valor3 ;
```

Este ejemplo requiere menos memoria y menos tiempo de ejecución si se programa de la siguiente forma:

```
VAR_TEMP
   tmp : STRUCT
                value  : REAL;
                valor1 : REAL;
                valor2 : REAL;
                valor3 : REAL;
         END_STRUCT;
END_VAR
tmp := DB100.feld[i];
DB100.feld[i].value := tmp.valor1 * tmp.valor2 / tmp.valor3;
```

Nota

Con VAR_TEMP se almacenan las variables en la pila de la CPU. En las CPUs pequeñas, esto puede provocar un desbordamiento de la pila. Por consiguiente, se recomienda no abusar de variables temporales.

Problemas de asignación de memoria de la pila de datos locales en CPUs pequeñas

Los problemas de asignación de memoria de la pila de datos locales se deben al reducido tamaño de la pila de las CPU pequeñas. En la mayoría de los casos, evitará muchos problemas si sigue estas indicaciones:

- No abuse de las variables temporales (apartado VAR_TEMP o VAR).
- No declare ninguna variable de un tipo de datos superior y reduzca el número de variables de un tipo de datos simple a un mínimo absoluto.
- Utilice variables estáticas:
  - cuando programe un FB puede utilizar la sección VAR en lugar de VAR_TEMP.
  - cuando programe un FB o un OB tiene que evitar los bloques de datos globales y las marcas.
- Evite las expresiones largas. Al ejecutar las expresiones largas, el compilador guarda resultados intermedios en la pila. Dependiendo del tipo y de la cantidad de resultados intermedios se puede exceder el tamaño de pila disponible.
  Remedio:
  Fraccione la expresión en partes más pequeñas y asigne los resultados intermedios a variables explícitas.

Emisión de números reales al observar

Al emitir números reales no representables, la función de test "Observar" puede suministrar los siguientes patrones:

Valor	Representación
+ infinity	1.#INFrandom-digits
- infinity	-1.#INFrandom-digits
Indefinite	digit.#INDrandom-digits
NaN digit.	#NANrandom-digits

Visualización de programas S7-SCL en AWL

Con el editor AWL/KOP/FUP se puede abrir un bloque S7-SCL para visualizar los comandos MC7 generados. No realice ningún cambio en AWL, porque:

- los comandos MC7 mostrados pueden no representar siempre un bloque AWL válido.
- una compilación correcta con el compilador AWL requiere por lo general una serie de modificaciones para las que son necesarias conocimientos profundos de AWL y SCL.
- El bloque compilado con AWL ya no reconoce el lenguaje AWL y SCL.
- La fuente S7-SCL y el código MC7 dejan de ser coherentes.

Tratamiento de la indicación de fecha y hora en la fuente, el interface y el código

Fecha y hora de la fuente

Una fuente S7-SCL siempre se indica con la fecha y hora de la última modificación.

Fecha y hora del código del bloque

Los bloques (FB, FC y OB) siempre se indican con la fecha y hora del momento en que fueron compilados.

Fecha y hora del interface del bloque

La fecha y hora de un interface solamente cambia cuando se modifica la estructura del interface, es decir,

- la indicación de fecha y hora de un interface se mantiene igual cuando se realizan modificaciones en el área de intrucciones, en los atributos, en el comentario, en la sección VAR_TEMP (en FC también VAR) o en la notación del nombre de los parámetros o las variables. Esto también es aplicable a los interfaces subordinados.
- La indicación de fecha y hora de un interface se actualiza cuando cambia el tipo de datos o una inicialización disponible de un parámetro o de una variable, o también cuando se añaden o suprimen parámetros o variables y cuando se modifica el nombre del FB en multiinstancias. Esto también es aplicable a los interfaces subordinados.

Valor de retorno de las funciones estándar y las funciones de sistema de STEP 7

Muchas funciones estándar y de sistema de STEP 7 tienen un valor de función del tipo INT, que contiene el código de error. En el manual de referencia de estas funciones se indican los posibles códigos de error como constantes WORD del tipo "W#16#8093".

S7-SCL es un lenguaje que comprueba severamente la igualdad de tipos, por lo tanto no se pueden mezclar INT y WORD. En consecuencia, no se obtiene el resultado deseado con la siguiente consulta.

```
IF SFCxx(..) = 16#8093 THEN ...
```

No obstante, puede notificar al compilador S7-SCL que considere una constante WORD como INT

- mediante la tipificación de la constante: en este caso, la consulta anterior toma la siguiente forma:
  ```
  IF SFCxx(..) = INT#16#8093 THEN ...
  ```
- mediante la función de conversión WORD_TO_INT() : tendrá que formular la consulta anterior de este modo:
  IF SFCxx(..) = WORD_TO_INT(16#8093) THEN ...

Recableado de bloques

En los bloques S7-SCL no es posible recablear la llamada de bloques con la función del administrador SIMATIC **Herramientas > Recablear**. El recableado se efectúa manualmente modificando la llamada del bloque correpondiente en la fuente SCL.

Recomendaciones:

- Defina en la tabla de símbolos nombres simbólicos para el bloque y llame al bloque de forma simbólica.
- Defina en la tabla de símbolos nombres simbólicos para las direcciones absolutas (E, M, A etc.) y utilice estos nombres simbólicos en su programa.

Cuando desee recablear un bloque sólo deberá modificar la correspondencia en la tabla de símbolos y no necesitará realizar ninguna modificación en la fuente SCL.

Asignación de estructuras con una longitud de bytes impar

La longitud de una estructura se rellena siempre hasta el límite de una palabra. Para disponer una estructura sobre un número impar de bytes, S7-SCL ofrece la construcción AT:

Ejemplo:

```
VAR
theStruct : STRUCT
                      twoBytes : ARRAY [0..1] OF BYTE;
                      oneInt : INT
                      oneByte : BYTE;
                  END_STRUCT;
fiveBytes AT theStruct : ARRAY[0..4] OF BYTE;
END_VAR
```

Ahí donde se exigen 5 BYTEs, utilice el identificador fiveBytes. Con el identificador theStruct se podrá acceder entonces de forma estructurada.

Valores límite para instrucciones FOR

Para programar instrucciones FOR "seguras" que no se ejecuten sin fin, tenga en cuenta la regla y los valores límite siguientes:

Regla

```
FOR ii := anfang TO ende BY schritt DO
```

Si...	...entonces	Observación
Principio < fin	Fin < (PMAX - paso)	La variable ii se desplaza en dirección positiva.
Principio > fin AND paso < 0	Fin > (NMAX - paso)	La variable ii se desplaza en dirección negativa.

Valores límite

Para ambos tipos de datos posibles se aplican valores límite diferentes:

Tipo de datos	PMAX	NMAX
ii del tipo INT	32_767	-32_768
ii del tipo DINT	2_147_483_647	-2_147_483_648

Glosario

Archivo fuente

Parte de un programa que se crea con un editor gráfico o textual y a partir del que, por compilación, se crea un programa de usuario ejecutable .

Área de declaración

En caso de crear el programa con un editor de textos, los datos locales de un bloque lógico se declaran en el área de declaración.

Área de memoria

Un módulo central (CPU) de SIMATIC S7 dispone de tres distintas áreas de memoria: la memoria de carga, la memoria de trabajo y la memoria del sistema.

Array

Un array es un tipo de datos compuesto que contiene elementos de datos del mismo tipo. A su vez estos elementos de datos pueden ser simples o compuestos.

Asignación

Mecanismo utilizado para asignar un valor a una variable.

Atributo

Un atributo es una propiedad que, por ejemplo, puede vincularse al nombre de un bloque o de una variable. En S7-SCL existen, por ejemplo, atributos para las siguientes tareas: título del bloque, versión, protección del bloque, autor, nombre del bloque, familia del bloque.

Ayuda en pantalla

STEP 7 ofrece la posibilidad de visualizar en la pantalla textos de ayuda contextual al trabajar con el software de programación.

BCD

Decimal codificado en binario. En STEP 7, los temporizadores y contadores se indican internamente en la CPU en formato BCD.

Bloque

Los bloques son componentes del programa de usuario delimitados por su función, por su estructura o por el uso para el que están destinados. En STEP 7 hay bloques lógicos (FB, FC, OB, SFC; SFB), bloques de datos (DB, SDB) y tipos de datos de usuario (UDT).

Bloque de datos (DB)

Los bloque de datos son bloques que contienen los datos y parámetros con los que opera el programa de usuario. A diferencia de los demás bloques, no contienen instrucciones.

Bloque de datos de instancia (DB de instancia)

Un bloque de datos de instancia almacena los parámetros formales y los datos locales estáticos de los bloques de función. El bloque de datos de instancia puede estar asignado a una llamada FB o a una jerarquía de llamada de bloques de función.

Bloque de datos del sistema (SDB)

Los bloques de datos del sistema son áreas de datos de la CPU S7 que contienen las preferencias del sistema y los parámetros del módulo. Los bloques de datos del sistema se crean y modifican con la herramienta Configuración S7.

Bloque de función (FB)

Según IEC 1131-3, un bloque de función (FB) es un bloque lógico con datos estáticos (datos estáticos). Dado que un FB dispone de memoria (bloque de datos de instancia), se puede acceder a sus parámetros (p. ej. de salida) en cualquier momento y desde cualquier punto del programa de usuario.

Bloque de función de sistema (SFB)

Un bloque de función de sistema (SFB) es un bloque de función integrado en el sistema operativo de la CPU que se puede llamar en caso necesario desde el programa de usuario STEP 7.

Bloque lógico

En SIMATIC S7 un bloque lógico es un bloque que contiene una sección del programa de usuario STEP 7. Por el contrario, un bloque de datos sólo contiene datos. Existen los siguientes bloques lógicos: bloques de organización (OB), bloques de función (FB), funciones (FC), bloques de función del sistema (SFB) y funciones del sistema (SFC).

Bloque de organización (OB)

Los bloques de organización constituyen la interfaz entre el sistema operativo de la CPU y el programa de usuario. En los bloques de organización se determina la secuencia de ejecución que sigue el programa de usuario.

Cargar en el sistema de destino

Transferir objetos cargables (p.ej. bloques lógicos) desde la unidad de programación a la memoria de carga de una CPU.

Clase de bloque

Los bloques se subdividen en dos clases, dependiendo de su contenido: bloques lógicos y bloques de datos.

Comentario

Estructura de lenguaje para la integración de textos explicativos en un programa que no tiene influencia alguna sobre la ejecución del programa.

Comentario del bloque

Información complementaria sobre un bloque (p.ej. aclaraciones sobre el proceso automatizado) que no se carga en la memoria de trabajo de los sistemas de automatización SIMATIC S7.

Compilación orientada a la fuente

En la edición orientada a la fuente, la compilación a un programa de usuario ejecutable no tiene lugar hasta que no se hayan terminado de introducir todas las instrucciones. Durante la compilación se comprueba la presencia de posibles errores.

Compilador S7-SCL

El compilador S7-SCL es un compilador por lotes que permite compilar el programa previamente editado (fuente S7-SCL) al código máquina MC7. Los bloqes así generados se almacenan en la carpeta "Bloques" del programa S7.

Compilar

Crear un programa de usuario ejecutable a partir de un archivo fuente.

Constante

Comodín que se utiliza en bloques lógicos para valores constantes. Las constantes se utilizan para incrementar la legibilidad del programa. Ejemplo: en lugar de indicar un valor directamente (p.ej. 10), se indica el comodín "recorridos_bucle_max". Al llamarlo, el comodín se sustituye por el valor de la constante (p.ej. 10).

Constante (simbólica)

Las constantes con nombres simbólicos son comodines para los valores constantes de los bloques lógicos. Se utilizan para mejorar la legibilidad de un programa.

Contadores

Los contadores son parte de la memoria de sistema de la CPU. El sistema operativo actualiza el contenido de estos contadores de forma asíncrona con respecto al programa de usuario. Con las instrucciones de STEP 7 se determina la función exacta de la célula de contaje (p. ej. contador incremental) y se activa su ejecución (inicio).

Conversión de tipos de datos

La conversión del tipo de datos es necesaria para combinar lógicamente dos operandos de diferentes tipos de datos.

Datos estáticos

Los datos estáticos son datos locales de un bloque de función que se almacenan en el bloque de datos de instancia y, por tanto, conservan su valor hasta la próxima ejecución del bloque de función.

Datos globales

Los datos globales son datos a los que se puede acceder desde cualquier bloque lógico (FC, FB, OB). En concreto, se trata de las marcas (M), entradas (E), salidas (A), temporizadores (T), contadores (Z) y elementos de los bloques de datos DB. A los datos globales se puede acceder tanto con direccionamiento simbólico como con direccionamiento absoluto.

Datos locales

Los datos locales son los datos que están asignados a un bloque lógico y que se declaran en el área de declaración del bloque. Dependiendo del bloque en cuestión, incluyen parámetros formales, datos estáticos y datos temporales.

Datos temporales

Los datos temporales son datos locales de un bloque que se almacenan en la pila de datos locales durante el procesamiento de un bloque y que dejan de estar disponibles al terminarse de procesar.

Datos útiles

Los datos útiles se intercambian entre una módulo central y un módulo de señales, entre un módulo de función y procesadores de comunicación a través de la imagen del proceso o a través de accesos directos. Datos útiles pueden ser señales de entrada/salida digitales y analógicas de módulos de señales así como informaciones de estado y de forzado de los módulos de función.

Declaración

Mecanismo para definir un elemento del lenguaje. Una declaración contiene la conexión de un identificador con un elemento de lenguaje así como la asignación de atributos a tipos de datos.

Declaración de variables

La declaración de variables contiene la indicación de un nombre simbólico, un tipo de datos y, eventualmente, un valor inicial y un comentario.

Depurador S7-SCL

El depurador S7-SCL es un depurador para lenguajes de alto nivel que permite localizar errores lógicos en los programas de usuario creados con S7-SCL.

Direccionamiento

Asignación de una dirección en el programa de usuario. Las direcciones se pueden asignar a determinados operandos o áreas de operandos (ejemplos: entrada 12.1, palabra de marcas MW25)

Direccionamiento absoluto

En el direccionamiento absoluto se indica explícitamente la dirección del operando a procesar. Ejemplo: la dirección A 4.0 designa al bit 0 del byte 4 de la imagen del proceso de las salidas.

Direccionamiento simbólico

En el direccionamiento simbólico, el operando a procesar se indica mediante un nombre simbólico. La correspondencia entre símbolos y direcciones se establece en la tabla de símbolos o en un archivo de símbolos.

Editor S7-SCL

El editor S7-SCL es un editor específico para S7-SCL que permite crear fuentes en el lenguaje S7-SCL.

Enable (EN)

En STEP 7, cada bloque de función y cada función contiene el parámetro de entrada "Enable" (EN), definido implícitamente, que se puede activar al ejecutar el bloque. Cuando EN es TRUE, el bloque llamado se ejecuta, en caso contrario no.

Enable out (ENO)

En STEP 7 cada bloque tiene una salida "Enable Output" (ENO). Al terminar la ejecución de un bloque, el valor actual de la marca OK se deposita en ENO. Inmediatamente después de llamar a un bloque se puede comprobar, mediante el valor de ENO, si todas las operaciones del bloque se han ejecutado correctamente o si se han producido errores.

Entero (INT)

Entero (INT) es uno de los tipos de datos simples. Se representa por medio de un número entero de 16 bits.

Estructura (STRUCT)

Tipo de datos que se compone de elementos de distinto tipos de datos. Los tipos de datos que contiene una estructura pueden ser simples o compuestos.

Expresión

En S7-SCL las expresiones sirven para procesar datos. Se distingue entre expresiones aritméticas, lógicas y de comparación.

Fuente S7-SCL

La fuente S7-SCL es el archivo en el que se crea el programa S7-SCL. Una vez creado, el archivo fuente se compila con el compilador S7-SCL.

Función (FC)

Según IEC 1131-3, una función (FC) es un bloque lógico sin datos estáticos. La función ofrece la posibilidad de transferir parámetros dentro del programa de usuario. Por consiguiente, las funciones resultan idóneas para programar funciones complejas que se repiten con frecuencia; p.ej., los cálculos.

Función de sistema (SFC)

Una función de sistema (SFC) es una función integrada en el sistema operativo de la CPU que puede ser llamada en caso necesario desde el programa de usuario STEP 7.

Identificador del operando

El identificador del operando es aquella parte del operando de una operación que contiene información como, por ejemplo, el área de memoria en la que la operación encuentra un valor (objeto de datos) con el que ejecutar una función lógica, o bien el tamaño de un valor (objeto de datos) con el que ejecuta una función lógica. En la instrucción "Valor:= EB10", "EB" es el identificador del operando ("E" representa el área de memoria de las entradas, "B" representa un byte dentro de este área).

Identificadores

Combinación de letras, cifras y guiones bajos mediante la cual se designa un elemento del lenguaje.

Imagen del proceso

Los estados de señal de los módulos de salida y de entrada digitales se almacenan en la CPU en una imagen del proceso. Se disitingue entre la imagen del proceso de las entradas (PAE) y la de las salidas (PAA).

Imagen del proceso de las entradas (PAE)

Antes de iniciar la ejecución del programa de usuario, el sistema operativo lee la imagen del proceso de las entradas que reside en los módulos de entrada.

Imagen del proceso de las salidas (PAA)

La imagen del proceso de las salidas es transferida por el sistema operativo a los módulos de salida al final del programa de usuario.

Instancia

El término "instancia" sirve para designar la llamada de un bloque de función que tiene asignado un bloque de datos de instancia o una instancia local. Cuando en el programa de usuario STEP 7 se llama n veces a un bloque de función, y cada vez con un parámetro y con un nombre de bloque de datos de instancia diferentes, existen n instancias.

Instrucción

Una instrucción es la unidad independiente más pequeña de un programa de usuario elaborado con un lenguaje textual. Representa una orden de trabajo para el procesador.

Instrucción CASE

Esta instrucción es una instrucción de ramificación. Dependiendo del valor de una expresión de selección, sirve para seleccionar una sección del programa dentro de 1-n opciones posibles.

Instrucción CONTINUE

En S7-SCL, una instrucción CONTINUE sirve para cancelar la ejecución actual del bucle dentro de una instrucción de repetición (FOR, WHILE o REPEAT).

Instrucción EXIT

Conjunto de instrucciones del lenguaje dentro de un programa que permite interrumpir un bucle en cualquier momento e independientemente de las condiciones (interrupción incondicional).

Instrucción FOR

Estructura de un programa. La instrucción FOR ejecuta una secuencia de instrucciones dentro de un bucle, en el que se van asignando valores consecutivos a una variable.

Instrucción GOTO

Estructura de un programa. Una instrucción GOTO tiene como consecuencia el salto inmediato a una meta de salto dada y, por tanto, a otra instrucción del mismo bloque.

Instrucción REPEAT

Estructura de un programa que sirve para repetir una secuencia de instrucciones hasta que se se cumpla una condición de interrupción.

Instrucción RETURN

Estructura de un programa que provoca la salida del bloque que se está ejecutando.

Interface de llamada

El interface de llamada queda definido por los parámetros de entrada, de salida y de entrada/salida (parámetros formales) de un bloque del programa de usuario. Al llamar al bloque se sustituyen por los parámetros actuales.

Jerarquía de llamada

Todos los bloques deben ser llamados antes de ser procesados. El orden y el anidamiento de estas llamadas se denomina jerarquía de llamada.

Literal

Notación formal que determina el valor y el tipo de una constante.

Llamada de bloque

Inicio de un bloque en el programa de usuario de STEP 7: los bloques de organización son llamados básicamente por el sistema operativo; todos los demás bloques son llamados por el programa de usuario de STEP 7.

Marcadores

Los marcadores son marcadores de texto temporales que marcan una posición cualquiera de la fuente y facilitan la navegación por la fuente.

Marcas (M)

Área de la memoria de sistema de una CPU SIMATIC S7. A ella se puede acceder leyendo o escribiendo (por bit, por byte, por palabra o por palabra doble). El usuario puede utilizar el área de marcas para guardar resultados intermedios.

Memoria de sistema (área del sistema)

La memoria de sistema está integrada en la CPU S7 y está realizada como memoria RAM. En la memoria de sistema residen las áreas de operandos (p. ej. temporizadores, contadores, marcas) así como las áreas de datos que el sistema operativo necesita internamente (p. ej. el búfer para comunicaciones).

Multiinstancia

Cuando se utilizan multiinstancias, el bloque de datos de instancia contiene datos para distintos bloques de función de una misma jerarquía de llamada.

Nemotécnica

La nemotécnica es una forma de representación abreviada de los operandos y de las operaciones del programa. STEP 7 asiste tanto la representación inglesa (p.ej."I" para entrada) como la representación alemana (p.ej. "E" para entrada).

Nonterminal

Nonterminal es un elemento compuesto de una descripción sintáctica que se describe mediante otra regla léxica o sintáctica.

Número real

Un número real es una cifra positiva o negativa que representa un valor decimal, por ejemplo 0,339 o –11,1.

Observar

Al observar un programa se controla su ejecución en la CPU. Durante este proceso se visualizan, por orden cronológico, y se actualizan cíclicamente los valores actuales de las variables y de los parámetros.

Offline

Offline designa aquel estado operativo en el que la unidad de programación no está conectada al sistema de automatización (ni de forma física, ni de forma lógica).

OK flag

OK flag sirve para indicar la ejecución correcta o incorrecta de una secuencia de instrucciones de un bloque. Es un tipo de datos BOOL global.

Online

Online designa aquel estado operativo en el que la unidad de programación está conectada al sistema de automatización (de forma física o lógica).

Operación

Una operación es parte de una instrucción e indica qué tiene que hacer el procesador.

Operando

Un operando es parte de una instrucción e indica, con qué tiene que operar el procesador. El operando se puede direccionar tanto absoluta como simbólicamente.

Palabra clave

Unidad léxica que caracteriza a un elemento de lenguaje, p.ej. "IF".

Las palabras clave se utilizan en S7-SCL para señalar el principio de un bloque, para marcar secciones del área de declaración, para identificar instrucciones así como para comentarios y atributos.

Palabra de estado

La palabra de estado forma parte de los registros del módulo central (CPU). En ella se memorizan informaciones sobre estados y errores que se registran durante la ejecución de los comandos de STEP 7. Los bits de estado se pueden leer y escribir, mientras que los bits de error sólo permiten accesos de lectura.

PARADA

El estado operativo PARADA se alcanza desde el estado operativo RUN tras una petición de la unidad de programación. En este estado operativo es posible efectuar funciones de test especiales.

Parámetros actuales

Los parámetros actuales sustituyen a los parámetros formales al llamar a un bloque de función (FB) o a una función (FC).

Ejemplo: el parámetro formal "Start" se sustituye por el parámetro actual "E 3.6".

Parámetros de entrada

Los parámetros de entrada sólo se encuentran en funciones y bloques de función. Mediante los parámetros de entrada se transfieren datos al bloque llamado para el procesamiento de los mismos.

Parámetro de entrada/salida

Los parámetros de entrada/salida se encuentran en funciones y bloques de función. Con estos parámetros se transfieren datos al bloque llamado; allí se procesan y los resultados se vuelven a depositar a continuación en la misma variable.

Parámetros de salida

Los parámetros de salida de un bloque del programa de usuario permiten transferir resultados al bloque invocante.

Parámetros formales

Un parámetro formal es un comodín para el parámetro real (parámetro actual) que se emplea en los bloques lógicos parametrizables. En el caso de los bloques de función (FB) y de las funciones (FC), es el usuario quien define los parámetros formales, mientras que en el caso de los bloques de función de sistema (SFB) y de las funciones de sistema (SFC), los parámetros formales ya están definidos. Al llamar a un bloque se asigna un parámetro actual al parámetro formal para que el bloque llamado trabaje con el valor actual. Los parámetros formales pertenecen a los datos locales del bloque y se dividen a su vez en parámetros de entrada, salida y de entrada/salida.

Paso a paso (por etapas)

Una etapa individual es una función de test por pasos del depurador S7-SCL. Permite ejecutar el programa instrucción por instrucción, y observarlo en la ventana de resultados.

Programa de usuario

El programa de usuario contiene todas las instrucciones y declaraciones necesarias para procesar las señales que permiten controlar una instalación o un proceso. Está asignado a un módulo programable (p. ej. CPU, FM) y puede estructurarse en unidades (bloques) más pequeñas.

Programa de usuario S7

Una carpeta que contiene bloques que se cargan en un módulo S7 programable (p. ej. CPU, FM) donde son ejecutados para controlar una instalación o un proceso.

Programación estructurada

Para soluciones de automatización complejas, el programa de usuario se subdivide en secciones del programa individuales e independientes (bloques). El programa de usuario se organiza siguiendo la estructura funcional o tecnológica de las instalaciones.

Programación simbólica

El lenguaje de programación S7-SCL permite utilizar cadenas de caracteres simbólicas en lugar de operandos. Así p.ej., es posible sustituir el operando A1.1 por la cadena "válvula_17". La tabla de símbolos establece la conexión entre el operando y la cadena de caracteres simbólica asignada éste.

Protección del bloque

Se denomina protección del bloque al medio que permite proteger un bloque contra una posible descompilación en aquellos casos en los que la fuente del bloque haya sido compilada con la palabra clave "KNOW_HOW_PROTECTED".

Proyecto

Una carpeta que contiene todos los objetos de una solución de automatización independientemente del número de equipos y módulos y de su conexión a la red.

Punto de parada

Esta función permite conmutar el módulo central (CPU) al estado operativo PARADA en determinadas posiciones del programa. Cuando se alcanza un punto de parada se puede proceder a realizar funciones de test como p.ej., la ejecución paso a paso de las instrucciones o la observación o el forzado de una variable de estado.

Regla sintáctica

El máximo nivel regulador de la descripción formal del lenguaje S7-SCL está constituido por reglas sintáticas. Al utilizar estas reglas no existe libertad de formato, es decir, no está permitido utilizar espacios en blanco ni caracteres de control.

RUN

En el estado operativo RUN se procesa el programa de usuario y se actualiza la imagen del proceso de forma cíclica. Todas las salidas digitales están habilitadas.

RUN-P

El estado operativo RUN-P equivale al estado operativo RUN, con la diferencia de que en aquél se permiten todas las funciones de la unidad de programación sin restricción alguna.

S7-SCL

Lenguaje de alto nivel similar a PASCAL que cumple la norma DIN EN-61131-3 (int. IEC 1131-3) y que ha sido diseñado para la programación de tareas complejas en un PLC, p. ej. algoritmos o tareas de procesamiento de datos. S7-SCL es la abreviatura de "Structured Control Language".

Sección de declaración

La declaración de variables de un bloque se divide en diversas secciones en las que se declaran los distintos parámetros del bloque. Por ejemplo, la sección de declaración IN contiene la declaración de los parámetros de entrada, y la sección de declaración OUT la declaración de los parámetros de salida.

Semántica

Relación entre los distintos elementos simbólicos de un lenguaje de programación y su significado, diseño y uso.

Símbolo

Un símbolo es un nombre definido por el usuario respetando determinadas reglas sintácticas. Una vez determinado lo que simboliza este nombre (p.ej., variable, tipo de datos, bloque), se puede utilizar en la programación y en el manejo y visualización. Ejemplo: operando: E 5.0; tipo de datos: BOOL; símbolo: pulsador_paro_de_emergencia.

Tabla de símbolos

Tabla de asignación de símbolos (=nombre) a direcciones para datos globales y bloques. Ejemplos: paro de emergencia (símbolo); E1.7 (dirección), o regulador (símbolo) SFB 24 (bloque).

Tabla de variables

En la tabla de variables se reunen las variables a observar o forzar, incluido su formato.

Temporizadores

Los temporizadores son parte de la memoria del sistema de la CPU. El sistema operativo actualiza el contenido de estos temporizadores de forma asíncrona con respecto al programa de usuario. Con las instrucciones de STEP 7 se determina la función exacta de la célula de temporización (p. ej. retardo a la conexión) y se activa su ejecución (inicio).

Terminal

Un terminal es un elemento básico de una regla léxica o sintáctica que no se explica por medio de otra regla, sino de manera verbal. Un terminal puede ser, p.ej,. una palabra clave o un carácter aislado.

THEN

En S7-SCL, una expresión sirve para el procesamiento de datos. Se distinguen expresiones aritméticas, expresiones lógicas y expresiones de comparación.

Tiempo de ciclo

El tiempo de ciclo es el tiempo que necesita la CPU para ejecutar una vez el programa de usuario.

Tiempo de vigilancia del ciclo

Si el tiempo de ejecución del programa de usuario excede el tiempo establecido de vigilancia del, el sistema operativo genera un mensaje de error y la CPU pasa al estado STOP.

Tipo de bloque

La arquitectura orientada a bloques de STEP 7 distingue los siguientes tipos de bloques: bloques de organización, bloques de función, funciones, bloques de datos, bloques de función de sistema, funciones de sistema, bloques de datos de sistema y tipos de datos personalizados.

Tipo de datos

El tipo de datos determina

1. el tipo y el significado de los elementos de datos
2. las áreas de memoria admisibles y valores de los elementos de datos
3. la cantidad admisible de operaciones que pueden ejecutarse con un operando de un determinado tipo de datos.
4. la notación de los elementos de datos

Tipo de datos compuesto

Se trata de tipos de datos complejos que se componen de datos del tipo simple. Se distingue entre estructuras y arrays. También pertenecen a este grupo los tipos de datos STRING y DATE_AND_TIME.

Tipo de datos de usuario

Los tipos de datos de usuario (UDT) los define el usuario al declarar el tipo de datos. Estos datos tienen su propio nombre distintivo, por lo que son de uso múltiple. Por ejemplo, un tipo de datos de usuario se puede aprovechar para crear varios bloques de datos que tengan la misma estructura (p.ej., reguladores).

Tipo de datos simple

Los tipos de datos simples son tipos predefinidos según IEC 1131-3. Ejemplos: El tipo de datos "BOOL" define una variable binaria ("bit"); el tipo de datos "INT" define una variable en coma fija de 16 bits.

Tipo de datos 'Parámetro'

Un tipo de parámetro es un tipo de datos especial para temporizadores, contadores y bloques. Puede utilizarse en parámetros de entrada de bloques de función y de funciones, y en parámetros de entrada/salida de bloques de función para transferir temporizadores, contadores y bloques al bloque llamado.

UDT

Véase: Tipo de datos de usuario.

Variable

Una variable define un dato de contenido variable que se puede utilizar en el programa de usuario STEP 7. Una variable se compone de un operando (p. ej. M 3.1) y un tipo de datos (p. ej. BOOL) y se puede declarar con un símbolo (p. ej. BAND_EIN). La variable se declara en el área de declaración.

Valor inicial

Valor que se asigna a una variable al arrancar el sistema.

Valor de respuesta (RET_VAL)

A diferencia de los bloques de función, las funciones suministran un valor de retorno.

Vista

Para acceder con otro tipo de datos a una variable declarada, es posible definir vistas sobre la variable o sobre áreas de la variable. La vista se utiliza como cualquier otra variable del bloque, heredando todas las propriedades de la variable a la que señala, pero con un tipo de datos distinto.

Índice alfabético

B

C

D

E

F

G

I

J

L

M

N

O

P

R

S

W

X

Y

Z