SoMachine
- Guía de programación

04/2014

EIO0000000071.11

www.schneider-electric.com

Schneider Electric

La información que se ofrece en esta documentación contiene descripciones de carácter general y/o características técnicas sobre el rendimiento de los productos incluidos en ella. La presente documentación no tiene como objeto sustituir dichos productos para aplicaciones de usuario específicas, ni debe emplearse para determinar su idoneidad o fiabilidad. Los usuarios o integradores tienen la responsabilidad de llevar a cabo un análisis de riesgos adecuado y completo, así como la evaluación y las pruebas de los productos en relación con la aplicación o el uso de dichos productos en cuestión. Ni Schneider Electric ni ninguna de sus filiales o asociados asumirán responsabilidad alguna por el uso inapropiado de la información contenida en este documento. Si tiene sugerencias de mejoras o modificaciones o ha hallado errores en esta publicación, le rogamos que nos lo notifique.

Al instalar y utilizar este producto es necesario tener en cuenta todas las regulaciones sobre seguridad correspondientes, ya sean regionales, locales o estatales. Por razones de seguridad y para garantizar que se siguen los consejos de la documentación del sistema, las reparaciones solo podrá realizarlas el fabricante.

Cuando se utilicen dispositivos para aplicaciones con requisitos técnicos de seguridad, siga las instrucciones pertinentes.

Si con nuestros productos de hardware no se utiliza el software de Schneider Electric u otro software aprobado, pueden producirse lesiones, daños o un funcionamiento incorrecto del equipo.

Si no se tiene en cuenta esta información, se pueden causar daños personales o en el equipo.

Tabla de materias

Información de seguridad

Información importante

AVISO

Lea atentamente estas instrucciones y observe el equipo para familiarizarse con el dispositivo antes de instalarlo, utilizarlo o realizar su mantenimiento. Los mensajes especiales que se ofrecen a continuación pueden aparecer a lo largo de la documentación o en el equipo para advertir de peligros potenciales o para ofrecer información que aclara o simplifica los distintos procedimientos.

La inclusión de este icono en una etiqueta “Peligro” o “Advertencia” indica que existe un riesgo de descarga eléctrica, que puede provocar lesiones si no se siguen las instrucciones.

Éste es el icono de alerta de seguridad. Se utiliza para advertir de posibles riesgos de lesiones. Observe todos los mensajes que siguen a este icono para evitar posibles lesiones o incluso la muerte.

⚠ PELIGRO

PELIGRO indica una situación de peligro que, si no se evita, **provocará** lesiones graves o incluso la muerte.

⚠ ADVERTENCIA

ADVERTENCIA indica una situación de peligro que, si no se evita, **podría provocar** lesiones graves o incluso la muerte.

⚠ ATENCIÓN

ATENCIÓN indica una situación peligrosa que, si no se evita, **podría provocar** lesiones leves o moderadas.

AVISO

AVISO indica una situación potencialmente peligrosa que, si no se evita, **puede provocar** daños en el equipo.

TENGA EN CUENTA

La instalación, manejo, puesta en servicio y mantenimiento de equipos eléctricos deberán ser realizados sólo por personal cualificado. Schneider Electric no se hace responsable de ninguna de las consecuencias del uso de este material.

Una persona cualificada es aquella que cuenta con capacidad y conocimientos relativos a la construcción, el funcionamiento y la instalación de equipos eléctricos y que ha sido formada en materia de seguridad para reconocer y evitar los riesgos que conllevan tales equipos.

Acerca de este libro

Presentación

Objeto

En este documento se describe la interfaz gráfica de usuario del software SoMachine y las funciones que ofrece. Para obtener más información, consulte los documentos independientes que se ofrecen en la ayuda online de SoMachine.

Campo de aplicación

Este documento se ha actualizado con la publicación de SoMachine V4.1.

Documentos relacionados

Título de la documentación	Número de referencia
Introducción a SoMachine	EIO0000000734 (ING); EIO0000000787 (FRA); EIO0000000788 (ALE); EIO0000000790 (ESP); EIO0000000789 (ITA); EIO0000000791 (CHI)
SoMachine - Comandos de menú - Ayuda online	EIO0000001854 (ING); EIO0000001855 (FRA); EIO0000001856 (ALE); EIO0000001858 (ESP); EIO0000001857 (ITA); EIO0000001859 (CHI)
SoMachine Central - Guía del usuario	EIO0000001659 (ING); EIO0000001660 (FRA); EIO0000001661 (ALE); EIO0000001663 (ESP); EIO0000001662 (ITA); EIO0000001664 (CHI)
SoMachine - Compatibilidad y migración - Guía del usuario	EIO0000001684 (ING); EIO0000001685 (FRA); EIO0000001686 (ALE); EIO0000001688 (ESP); EIO0000001687 (ITA); EIO0000001689 (CHI)

Título de la documentación	Número de referencia
SoMachine Funciones y bibliotecas - Guía del usuario	EIO0000000735 (ING); EIO0000000792 (FRA); EIO0000000793 (ALE); EIO0000000795 (ESP); EIO0000000794 (ITA); EIO0000000796 (CHI)
Guía del usuario del Asistente del controlador de SoMachine	EIO0000001671 (ING); EIO0000001672 (FRA); EIO0000001673 (ALE); EIO0000001675 (ESP); EIO0000001674 (ITA); EIO0000001676 (CHI)
Modicon M238 Logic Controller - Guía de programación	EIO0000000384 (ING); EIO0000000385 (FRA); EIO0000000386 (ALE); EIO0000000388 (ESP); EIO0000000387 (ITA); EIO0000000389 (CHI)
SoMachine - Device Type Manager (DTM) - Guía de programación	EIO0000000673 (ING); EIO0000000674 (FRA); EIO0000000675 (ALE); EIO0000000676 (ESP); EIO0000000677 (ITA); EIO0000000678 (CHI)
TwidoEmulationSupport - Guía de la biblioteca	EIO0000001692 (ING); EIO0000001693 (FRA); EIO0000001694 (ALE); EIO0000001696 (ESP); EIO0000001695 (ITA); EIO0000001697 (CHI)
SoMachine - Configuración de variables de red - Guía de la biblioteca SE_NetVarUdp	EIO0000001151 (ING); EIO0000001152 (FRA); EIO0000001153 (ALE); EIO0000001155 (ESP); EIO0000001154 (ITA); EIO0000001156 (CHI)

Puede descargar estas publicaciones técnicas y otra información técnica de nuestro sitio web www.schneider-electric.com.

Información relativa al producto

⚠ ADVERTENCIA

PÉRDIDA DE CONTROL

- El diseñador del esquema de control debe tener en cuenta las posibles modalidades de fallo de rutas de control y, para ciertas funciones de control críticas, proporcionar los medios para lograr un estado seguro durante y después de un fallo de ruta. Funciones de control críticas son, por ejemplo, una parada de emergencia y una parada de sobrerrecorrido, un corte de alimentación y un reinicio.
- Para las funciones de control críticas deben proporcionarse rutas de control separadas o redundantes.
- Las rutas de control del sistema pueden incluir enlaces de comunicación. Deben tenerse en cuenta las implicaciones de los retrasos de transmisión no esperados o los fallos en el enlace.
- Tenga en cuenta todas las reglamentaciones para la prevención de accidentes y las directrices de seguridad locales.[1]
- Cada implementación de este equipo debe probarse de forma individual y exhaustiva antes de entrar en servicio.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

[1] Para obtener información adicional, consulte NEMA ICS 1.1 (última edición), "Safety Guidelines for the Application, Installation, and Maintenance of Solid State Control" (Directrices de seguridad para la aplicación, la instalación y el mantenimiento del control de estado estático) y NEMA ICS 7.1 (última edición), "Safety Standards for Construction and Guide for Selection, Installation and Operation of Adjustable-Speed Drive Systems" (Estándares de seguridad para la construcción y guía para la selección, instalación y utilización de sistemas de unidades de velocidad ajustable) o su equivalente aplicable a la ubicación específica.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

- Utilice solo software aprobado por Schneider Electric para este equipo.
- Actualice el programa de aplicación siempre que cambie la configuración de hardware física.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

Parte I
Introducción

Contenido de esta parte

Esta parte contiene los siguientes capítulos:

Capítulo 1
Introducción general a SoMachine Logic Builder

Contenido de este capítulo

Este capítulo contiene los siguiente apartados:

¿Qué es SoMachine Logic Builder?

Descripción general

Logic Builder proporciona el entorno de configuración y programación para los proyectos de SoMachine que cree con SoMachine Central.

Muestra los diferentes elementos de un proyecto en vistas separadas que se pueden organizar en la interfaz de usuario de SoMachine y en el escritorio, en función de las necesidades individuales. Esta estructura de vistas permite añadir elementos de hardware y software al proyecto mediante el método de arrastrar y soltar. Los principales cuadros de diálogo de configuración que permiten crear contenido para el proyecto se proporcionan en el centro de la pantalla de Logic Builder.

Además de una sencilla configuración y programación, Logic Builder también proporciona potentes funciones de diagnóstico y mantenimiento.

Tareas que realiza SoMachine Logic Builder

Configuración y programación de proyectos

Logic Builder permite programar la lógica y añadir dispositivos a los proyectos de SoMachine que cree con SoMachine Central.

Para ayudarle a realizar estas tareas, ofrece las funciones siguientes:

- Las vistas independientes del catálogo de hardware **Controlador, HMI & iPC, Field Devices & Modules y Varios** permiten añadir dispositivos de hardware al proyecto sólo con arrastrar y soltar. También le permite utilizar plantillas de dispositivos y de funciones.
- Las vistas independientes del catálogo de software para **Variables, Activos, Macros, Herramientas, Bibliotecas** permiten añadir diferentes tipos de elementos de software sólo con arrastrar y soltar. La vista **Activos**, por ejemplo, permite crear y gestionar bloques de funciones y POU.

Para visualizar sólo las vistas relevantes para la tarea que se está llevando a cabo, SoMachine ofrece perspectivas *(véase página 47)* individuales para la configuración de hardware y de software y la modalidad online. Podrá adaptar esas perspectivas predeterminadas a sus necesidades específicas y también podrá crear sus propias perspectivas con las vistas que utiliza con más frecuencia.

Compilación de proyectos

Logic Builder ofrece diferentes métodos (como **Compilar**, **Generar todo** o **Limpiar todo**) para compilar un proyecto de SoMachine.

Comunicación con el controlador

Logic Builder ofrece funciones de exploración para detectar los controladores disponibles en la red Ethernet. Admite diferentes protocolos para la comunicación con el controlador.

Una vez establecida la comunicación, las aplicaciones podrán descargarse o cargarse desde el controlador. Las aplicaciones pueden iniciarse y detenerse en el controlador.

Características y supervisión online

Las características de Logic Builder de supervisión online permiten realizar las tareas siguientes:

- Supervisar online los valores del código del programa y de las vistas **Supervisar**
- Realizar cambios online
- Configurar trazas online
- Supervisar trazas online
- Interactuar con la máquina mediante visualizaciones integradas en modalidad online para realizar diagnósticos y pruebas
- Leer el estado de controladores y dispositivos
- Detectar errores potenciales de lógica de programación mediante la función de depuración

Capítulo 2
Interfaz de usuario de SoMachine Logic Builder

Contenido de este capítulo

Este capítulo contiene los siguiente apartados:

Elementos de la pantalla de SoMachine Logic Builder

Descripción general

Logic Builder consta de los siguientes elementos:

- Menús y barras herramientas
- vistas del **Navegador**
- vistas del **Catálogo**
- Panel del editor principal

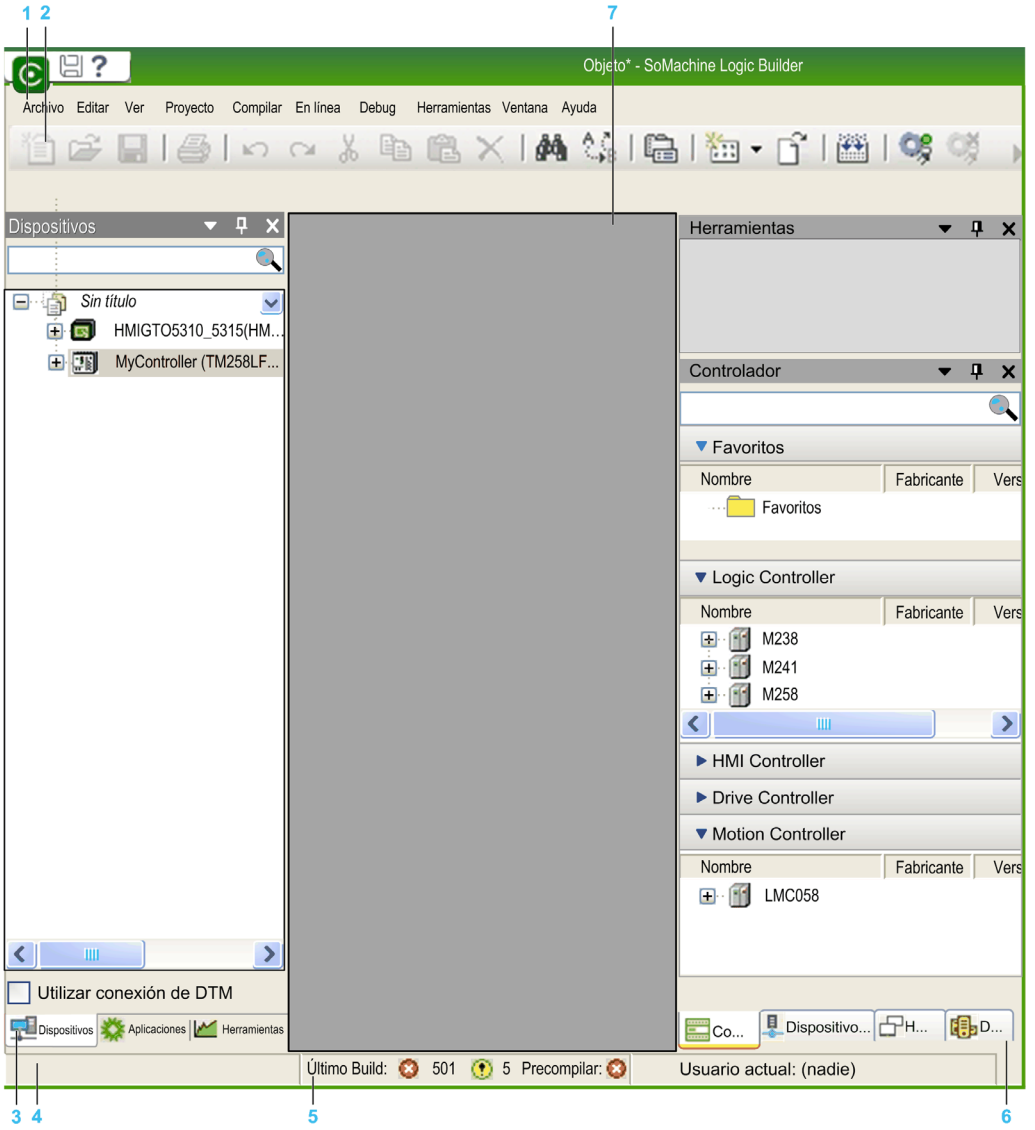
Cuando abre Logic Builder, se ofrece un diseño de pantalla predeterminado. En este documento se describen las posiciones predeterminadas.

Puede adaptar los elementos en función de los requisitos individuales según se describen en el capítulo *Personalización de la interfaz de usuario* *(véase página 44)*. Puede ver y modificar la configuración actual en el cuadro de diálogo **Personalizar**. Está disponible de forma predeterminada en el menú **Herramientas**.

También puede organizar las vistas y las ventanas en cualquier momento desplazándolas, acoplando/desacoplando las vistas, cambiando el tamaño de las ventanas o bien cerrándolas. Las posiciones se guardan con el proyecto. Cuando vuelve a abrir un proyecto, los elementos están colocados en las posiciones en las que se dejaron cuando se guardó el proyecto. Las posiciones de las vistas se guardan por separado en perspectivas *(véase página 47)*.

Pantalla de Logic Builder predeterminada

Posiciones predeterminadas de menús, barras y vistas en la pantalla de Logic Builder



1 Barra de menús
2 Barra de herramientas
3 **Navegadores** de varias fichas: árboles de **Dispositivos, Herramientas, Aplicaciones**
4 Vista **Mensajes**
5 Barra de estado e información

6 Vista de catálogo de varias fichas: catálogo de hardware: **Controlador, HMI & iPC, Devices & Modules, Varios**; catálogo de software: **Variables, Activos, Macros, Herramientas, Bibliotecas**
7 Vista del editor de varias fichas

Componentes estándar

En la pantalla de Logic Builder se incluyen los siguientes componentes que están visibles de forma predeterminada:

Componente	Descripción
Barra de menús	Proporciona menús en los que se incluyen los comandos disponibles según se definen en el cuadro de diálogo **Herramientas** → **Personalizar**.
Barra de herramientas	Incluye botones para ejecutar las herramientas disponibles según se definen en el cuadro de diálogo **Herramientas** →**Personalizar**.
Navegadores de varias fichas	Los siguientes **navegadores** están disponibles como fichas donde los distintos objetos de un proyecto están organizados en forma de estructura de árbol: ● Árbol **Dispositivos** ● Árbol **Aplicaciones** ● Árbol **Herramientas** Para obtener más información, consulte el capítulo *Navegadores de varias fichas (véase página 36)*.
Vista **Mensajes**	Proporciona mensajes en las operaciones de precompilación, compilación y descarga. Para obtener más información, consulte la descripción de los comandos de la vista *(véase SoMachine, Comandos de menú, Ayuda en línea)* **Mensajes**.
Barra de estado e información	Proporciona la información siguiente: ● Información sobre el usuario actual. ● Información sobre la modalidad de edición y la posición actual si hay un editor abierto. Para obtener más información, consulte la sección *Barra de estado e información* en este capítulo.

Componente	Descripción

Vista **Catálogo** de varias fichas	La vista **Catálogo** consta de diferentes fichas en las que se enumeran los objetos de hardware y software disponibles:
	● **Catálogo de hardware**
	  ● **Controlador**
	  ● **HMI & iPC**
	  ● **Devices & Modules**
	  ● **Varios**
	● **Catálogo de software**
	  ● **Variables**
	  ● **Activos**
	  ● **Macros**
	  ● **Herramientas**
	  ● **Bibliotecas**
	Para obtener más información, consulte el capítulo *Vistas de catálogo de varias fichas (véase página 42)*.

Ventana del editor de varias fichas	Se usa para crear el objeto específico en el editor respectivo. En el caso de los editores de lenguaje (por ejemplo, el editor ST o el editor CFC), normalmente la ventana combina el editor de lenguaje en la parte inferior y el editor de declaraciones en la parte superior. En el caso de otros editores, puede proporcionar cuadros de diálogo (como por ejemplo el editor de tarea o el editor de dispositivos). El nombre de la POU o el objeto de recurso se muestran en la barra de título de esta vista. Puede abrir los objetos en la ventana del editor en modalidad offline u online ejecutando el comando **Modificar objeto**.

Barra de estado e información

La barra situada en el borde inferior de la pantalla de Logic Builderofrece 3 tipos de información:

- Información sobre el usuario conectado.
- Si está trabajando en una ventana del editor: la posición del cursor y el estado de la modalidad de edición.
- En modalidad online: el estado actual del programa.

Información sobre el usuario conectado

En cada proyecto se incluye una gestión de usuarios y accesos *(véase página 137)*. El nombre del usuario conectado actualmente se muestra en la barra de estado.

Posiciones de cursor en las ventanas del editor

La posición del cursor se cuenta desde el margen izquierdo o superior de la ventana del editor.

Abreviatura	Descripción
Ln	Línea en la que está situado el cursor.
Col	Columna en la que está situado el cursor. (Una columna incluye exactamente 1 espacio, carácter o dígito).
Ch	Número de caracteres. (En este contexto, un carácter puede ser un solo carácter o dígito, o bien una tabla que incluya, por ejemplo, 4 columnas).

Haga doble clic en uno de los campos para abrir el cuadro de diálogo **Ir a la línea**. Aquí puede introducir una posición distinta en la que se sitúa el cursor.

El estado de la modalidad de edición se indica con las abreviaturas siguientes:

Abreviatura	Descripción
INS	Modalidad de inserción
OVR	Modalidad de sobrescritura

Haga doble clic en este campo para cambiar el ajuste.

Se indica el siguiente estado del programa:

Texto	Descripción
Programa cargado	Programa cargado en el dispositivo.
Programa inalterado	El programa en el dispositivo coincide con el del sistema de programación.
Programa modificado (modificación en línea)	El programa en el dispositivo es distinto al del sistema de programación; se necesita cambio en línea.
Programa modificado (descarga completa)	El programa en el dispositivo es distinto al del sistema de programación; se necesita descarga completa.

Información de la modalidad online

Estado de la aplicación en el dispositivo:

Texto	Color de fondo	Descripción
RUN	Verde	Programa en ejecución.
STOP	Rojo	Programa detenido.
PARAR EN PI	Rojo	Programa detenido en un punto de interrupción.
El siguiente campo de estado sólo está disponible si el controlador, en función de un ajuste en la descripción del dispositivo, admite una supervisión que es independiente del ciclo.		
EN EL CICLO	Blanco	Indica que los valores de las expresiones supervisadas se leen en un ciclo.
FUERA DEL CICLO	Rojo	Indica que la recuperación de los valores de las variables supervisadas no se puede realizar en un ciclo.

Ventanas de supervisión y vistas online de editores

En las ventanas de supervisión y en las vistas online de editor se muestra una vista de supervisión de una POU o una lista definida por el usuario de expresiones de supervisión.

Ventanas, vistas y ventanas de editor

Hay 2 tipos de ventanas distintas en Logic Builder:

- Algunas se pueden acoplar a cualquier margen de la ventana de SoMachine o se pueden situar en la pantalla como ventanas desacopladas separadas de la ventana de SoMachine. Asimismo, se pueden ocultar representándolas como una ficha en el marco de ventana de SoMachine (consulte el capítulo *Personalización de la interfaz de usuario* *(véase página 44)*). Estas ventanas muestran información que no depende de un único objeto del proyecto (por ejemplo la vista **Mensajes** o el árbol **Dispositivos**). Puede acceder a ellas en el menú *(véase SoMachine, Comandos de menú, Ayuda en línea)* **Ver**. En la mayoría de las vistas se incluye una barra de herramientas no configurable con botones para ordenar, ver y buscar en la ventana.
- Otras ventanas se abren cuando ve o edita un objeto de proyecto específico en el editor respectivo. Se muestran en la ventana del editor de varias fichas. No puede ocultarlas ni desacoplarlas de la ventana de SoMachine. Puede acceder a ellas en el menú **Ventana**.

Cambio de ventanas

SoMachine permite cambiar entre vistas y editores abiertos. Para cambiar entre vistas y editores abiertos, pulse las teclas CTRL y TAB simultáneamente. Se abre una vista en las que se enumeran las vistas y los editores que están abiertos. Si se mantiene pulsada la tecla CTRL, la ventana permanece abierta. Utilice la tecla TAB o las teclas de FLECHA simultáneamente para seleccionar una vista o un editor específicos.

Navegadores de varias fichas

Descripción general

Los **navegadores** de varias fichas son componentes estándar de la pantalla de Logic Builder.

De forma predeterminada, están disponibles los navegadores siguientes:

- Árbol **Dispositivos**: Le permite gestionar los dispositivos en los que se debe ejecutar la aplicación.
- Árbol **Aplicaciones**: Le permite gestionar POU específicas del proyecto y globales, además de tareas en una única vista.
- Árbol **Herramientas**: Le permite gestionar bibliotecas específicas del proyecto y globales, así como otros elementos, en una única vista.

Puede acceder a las vistas mediante el menú **Ver**.

Adición de elementos a los navegadores

El nodo raíz de un navegador representa un dispositivo programable. Puede insertar otros elementos debajo de este nodo raíz.

Para añadir elementos a un nodo de un **navegador**, seleccione un dispositivo u objeto en el catálogo de hardware o software de la parte derecha de la pantalla de Logic Builder y arrástrelo al **navegador** (por ejemplo, árbol **Dispositivos**). El nodo o los nodos con los que se corresponda el dispositivo u objeto seleccionado se expandirán automáticamente y se mostrarán en negrita. Los demás nodos en los que no pueda insertarse el dispositivo o el objeto se mostrarán en gris. Suelte el dispositivo u objeto en el nodo adecuado y se insertará automáticamente. Si se requieren elementos adicionales para el dispositivo o el objeto, como administradores de comunicación, se insertarán automáticamente.

Como alternativa, puede seleccionar un nodo del árbol. Es posible añadir un objeto al dispositivo u objeto seleccionado: se muestra un signo más de color verde. Haga clic en ese signo más para abrir un menú que proporciona los elementos disponibles para la inserción.

También es posible añadir un objeto o un dispositivo haciendo clic con el botón derecho en un nodo de un **navegador** y ejecutando el comando **Agregar objeto** o **Agregar dispositivo**. El tipo de dispositivo que puede insertarse depende del objeto seleccionado actualmente en el **navegador**. Por ejemplo, los módulos para un esclavo PROFIBUS DP no pueden insertarse sin haber insertado antes un dispositivo esclavo adecuado. Tenga en cuenta que sólo estarán disponibles para la inserción los dispositivos instalados correctamente en el sistema local y que coincidan con la posición actual en el árbol.

Recolocación de objetos

Para recolocar objetos, utilice los comandos estándar del portapapeles (**Cortar, Copiar, Pegar, Eliminar**) del menú **Editar**. Como alternativa, puede arrastrar el objeto seleccionado pulsando el botón del ratón (más la tecla CTRL para copiarlo). Cuando añade dispositivos utilizando la función de copiar y pegar, el nuevo dispositivo recibe el mismo nombre seguido de un número correlativo.

Actualización de la versión de un dispositivo

Un dispositivo que ya se ha insertado en los **navegadores** puede actualizarse a otra versión o convertirse en otro dispositivo.

Consulte la descripción de los diferentes comandos:

- Comando **Actualizar dispositivo** *(véase página 81)*
- Comando **Convertir dispositivo** *(véase página 83)*

Descripción del árbol Dispositivos

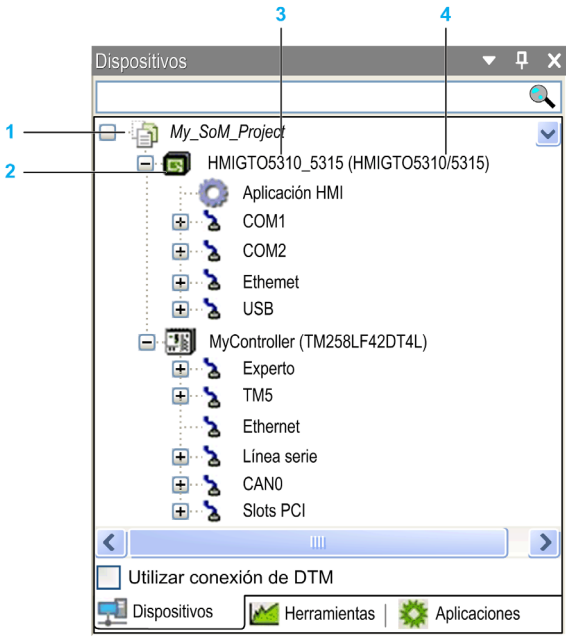
Cada objeto de dispositivo del árbol **Dispositivos** representa un objeto de hardware (de destino) específico.

Ejemplos: controlador, nodo de bus de campo, acoplador de bus, accionamiento, módulo de E/S

Los dispositivos y subdispositivos se gestionan en el árbol **Dispositivos**. Los demás objetos necesarios para ejecutar una aplicación en un controlador se agrupan en los otros **navegadores**.

- El nodo raíz del árbol es una entrada de nodo simbólica:  <nombre de proyecto>
- La configuración del controlador se define mediante la organización topológica de los dispositivos en el árbol **Dispositivos**. La configuración de los parámetros del dispositivo o tarea concretos se realiza en los diálogos del editor correspondiente. Consulte también el capítulo *Configuración de tareas* *(véase página 246)*.
  De este modo, la estructura de hardware se asignará y representará en el árbol **Dispositivos** mediante la correspondiente organización de los objetos de dispositivo, lo que le permite configurar un sistema complejo y heterogéneo de controladores conectados y buses de campo subyacentes.
- Para añadir a su proyecto dispositivos configurados con gestores de tipos de dispositivo (DTM), seleccione la casilla **Utilizar conexión de DTM** en la parte inferior del árbol **Dispositivos**. Esto tendrá como efecto la adición del nodo **FdtConnections** debajo del nodo raíz del árbol. Debajo del nodo **FdtConnections** se insertará automáticamente un nodo de administrador de comunicación. Puede añadir el dispositivo DTM apropiado a este nodo. Para obtener más información, consulte la publicación SoMachine - Device Type Manager (DTM) - Guía del usuario *(véase SoMachine, Device Type Manager (DTM), User Guide)*.
- Tenga en cuenta las recomendaciones para la *adición de elementos a los navegadores* de este capítulo.

Ejemplo de árbol **Dispositivos**:



1 Nodo raíz
2 Dispositivo programable (con aplicaciones)
3 Nombre de dispositivo simbólico
4 Nombre de dispositivo definido en el archivo de descripción de dispositivo

- Cada entrada del árbol **Dispositivos** muestra el símbolo, el nombre simbólico (editable) y el tipo de dispositivo (el nombre del dispositivo según se especifica en la descripción del dispositivo).
- Un dispositivo es programable o configurable. El tipo de dispositivo determina la posible posición en el árbol y también los recursos adicionales que pueden insertarse debajo del dispositivo.
- En un proyecto, puede configurar uno o varios dispositivos programables, con independencia del fabricante o el tipo (multirecurso, multidispositivo, conexión en red).
- Configure la comunicación, los parámetros y la asignación de E/S de un dispositivo en el diálogo de dispositivo (editor de dispositivos). Para abrir el editor de dispositivos, haga doble clic en el nodo de dispositivo del árbol **Dispositivos** (consulte la descripción del editor de dispositivos *(véase página 101)*).
- En modalidad online, el estado de un dispositivo se indica mediante un icono delante de la entrada del dispositivo:

- El controlador está conectado; la aplicación se está ejecutando; el dispositivo está operativo; se intercambian datos. La opción **Actualizar E/S en parada** de la vista **Ajustes PLC** del editor de dispositivos *(véase página 134)* puede activarse o desactivarse.
- El controlador está conectado y detenido (STOP). La opción **Actualizar E/S en parada** de la vista **Ajustes PLC** del editor de dispositivos *(véase página 134)* está desactivada.
- El controlador está conectado; la aplicación activa está en ejecución; hay disponible información de diagnóstico.
- El dispositivo no intercambia datos; no se ha configurado, está en modo de simulación o se ha detectado un error de bus (consulte la descripción del comando **Simulación**).
- El dispositivo se ejecuta en modo de demostración durante 30 minutos. Después de este tiempo finaliza el modo de demostración y el bus de campo deja de intercambiar datos.
- El dispositivo está configurado pero no está completamente operativo. No se intercambian datos. Por ejemplo, los dispositivos CANopen están iniciándose y en estado preoperativo.
- Modo de redundancia activo: El maestro de bus de campo no envía datos actualmente debido a que hay otro maestro en modo activo.
- No se ha encontrado la descripción del dispositivo en el repositorio de dispositivos. Para obtener más información sobre la instalación y desinstalación de dispositivos en el cuadro de diálogo **Repositorio de dispositivos**, consulte la descripción del **repositorio de dispositivos** *(véase SoMachine, Comandos de menú, Ayuda en línea)*.

- Los nombres de todos los dispositivos y aplicaciones conectados actualmente se muestran sombreados en verde.
- Los nombres de los dispositivos que se ejecutan en modo de simulación (consulte la descripción del comando **Simulación**) se muestran en cursiva.
- Se proporciona información adicional de diagnóstico en la vista **Estado** del editor de dispositivos *(véase página 151)*.

También puede ejecutar la aplicación activa en un dispositivo de simulación que esté disponible automáticamente de forma predeterminada en el sistema de programación. Por tanto, no es necesario ningún dispositivo de destino real para probar el comportamiento online de una aplicación (al menos, que no dependa de recursos de hardware para la ejecución). Al pasar al modo de simulación *(véase SoMachine, Comandos de menú, Ayuda en línea)*, la entrada del árbol **Dispositivos** se muestra en cursiva y puede iniciar sesión en la aplicación.

También puede conectarse al controlador en modo de configuración en línea (consulte el capítulo *Modo de configuración en línea (véase SoMachine, Comandos de menú, Ayuda en línea)*) sin tener que cargar primero una aplicación real en el controlador. Esto resulta muy útil para la puesta en marcha inicial de un sistema de E/S porque puede acceder a las E/S y probarlas en la configuración del controlador antes de compilar y cargar un programa de aplicación real.

Para obtener información sobre la conversión de referencias de dispositivo al abrir proyectos, consulte la publicación *SoMachine - Compatibilidad y migración - Guía del usuario*.

Organización y configuración de objetos en el árbol Dispositivos

Adición de dispositivos/objetos:

Para añadir dispositivos u objetos al árbol **Dispositivos**, simplemente seleccione un dispositivo u objeto en el catálogo de hardware, en la parte derecha de la pantalla de Logic Builder, y arrástrelo al árbol **Dispositivos**. El nodo o los nodos con los que se corresponda el dispositivo u objeto seleccionado se expandirán y se mostrarán en negrita. Los demás nodos en los que no pueda insertarse el dispositivo o el objeto se mostrarán en gris. Suelte el dispositivo u objeto en el nodo adecuado y se insertará automáticamente.

Como alternativa, puede seleccionar un nodo del árbol. Es posible añadir un objeto al dispositivo u objeto seleccionado: se muestra un signo más de color verde. Haga clic en el signo más para abrir un menú que proporciona los elementos disponibles para la inserción.

Como alternativa, puede añadir un objeto o un dispositivo haciendo clic con el botón derecho en un nodo del árbol **Dispositivos** y ejecutando el comando **Agregar objeto** o **Agregar dispositivo**. El tipo de dispositivo que puede insertarse depende del objeto seleccionado actualmente en el árbol **Dispositivos**. Por ejemplo, los módulos para un esclavo PROFIBUS DP no pueden insertarse sin haber insertado antes un dispositivo esclavo adecuado. No pueden insertarse aplicaciones debajo de dispositivos no programables.

Tenga en cuenta que sólo estarán disponibles para la inserción los dispositivos instalados correctamente en el sistema local y que coincidan con la posición actual en el árbol.

Recolocación de objetos:

Para recolocar objetos, utilice los comandos estándar del portapapeles (**Cortar, Copiar, Pegar, Eliminar**) del menú **Editar**. Como alternativa, puede arrastrar el objeto seleccionado pulsando el botón del ratón (más la tecla CTRL para copiarlo). Observación para el comando **Pegar**: En caso de que el objeto que se pegará pueda insertarse debajo o encima de la entrada seleccionada actualmente, se abrirá el cuadro de diálogo **Seleccionar la posición de inserción**. Le permitirá definir la posición de inserción. Cuando añade dispositivos utilizando la función de copiar y pegar, el nuevo dispositivo recibe el mismo nombre seguido de un número correlativo.

Actualización de la versión de un dispositivo:

Un dispositivo que ya se ha insertado en el árbol **Dispositivos** puede sustituirse por otra versión del mismo tipo de dispositivo o por un dispositivo de otro tipo (actualización de dispositivo). Al hacerlo, se conservará un árbol de configuración con sangría debajo del dispositivo respectivo el mayor tiempo posible.

Adición de dispositivos al nodo raíz:

Sólo pueden colocarse dispositivos en el nivel directamente inferior al nodo raíz <nombre de proyecto>. Si elige otro tipo de objeto en el cuadro de diálogo **Agregar objeto**, como el objeto **Lista de texto**, se añade al nodo **Global** del árbol **Aplicaciones**.

Subnodos:

Un dispositivo se inserta como un nodo en el árbol. Si se define en el archivo de descripción de dispositivo, los subnodos se insertan automáticamente. Un subnodo puede ser de nuevo un dispositivo programable.

Inserción de dispositivos debajo de un objeto de dispositivo:

Puede insertar otros dispositivos debajo de un objeto de dispositivo. Si se instalan en el sistema local, estarán disponibles en el catálogo de hardware o en el cuadro de diálogo **Agregar objeto** o **Agregar dispositivo**. Los objetos de dispositivo se clasifican en el árbol de arriba abajo. En un determinado nivel del árbol, primero se organizan los dispositivos programables, seguidos de los demás dispositivos, ordenados alfabéticamente.

Descripción del árbol Aplicaciones

Los objetos, la configuración de tareas y los objetos de tareas de la **aplicación** se gestionan en el árbol **Aplicaciones**.

Los objetos necesarios para programar el dispositivo (aplicaciones, listas de texto, etc.) se gestionan en el árbol **Aplicaciones**. Los dispositivos que no son programables (sólo configuración) no pueden asignarse como objetos de programación. Puede editar los valores de los parámetros del dispositivo en el diálogo de parámetros del editor de dispositivos.

Los objetos de programación, como determinadas POU o listas de variables globales, pueden gestionarse de dos maneras en el árbol **Aplicaciones** en función de su declaración:

- Cuando se declaran como subnodo del nodo **Global**, todos los dispositivos pueden acceder a estos objetos.
- Cuando se declaran como subnodo del nodo **Aplicaciones**, sólo los dispositivos correspondientes, declarados en el nodo **Aplicaciones**, pueden acceder a estos objetos.

Un objeto de **Aplicación** sólo puede insertarse en el árbol **Aplicaciones**.

Debajo de cada aplicación, puede insertar objetos de programación adicionales, como **DUT, GVL** u objetos de visualización. Inserte una configuración de tareas debajo de una aplicación. En esta configuración de tareas, deben definirse las llamadas de programa respectivas (instancias de POU del nodo **Global** del árbol **Aplicaciones** o POU específicas de dispositivo). Tenga en cuenta que la aplicación se define en la vista **Asignación E/S** del editor de dispositivos respectivo *(véase página 154)*.

Descripción del árbol Herramientas

Las bibliotecas se gestionan en el árbol **Herramientas**. No pueden asignarse dispositivos configurables puros como objetos de programación. Puede editar los valores de los parámetros del dispositivo en el diálogo de parámetros del editor de dispositivos.

Los objetos de programación, como el **Administrador de bibliotecas**, pueden gestionarse de dos formas diferentes en el árbol **Herramientas**, en función de su declaración:

- Cuando se declaran como subnodo del nodo **Global**, todos los dispositivos pueden acceder a estos objetos.
- Cuando se declaran como subnodo del nodo **Aplicaciones**, sólo los dispositivos correspondientes, declarados en el nodo **Aplicaciones**, pueden acceder a estos objetos.

La vista de varias fichas Catálogo

Descripción general

La vista de varias fichas **Catálogo de hardware** es un componente estándar de la pantalla de Logic Builder.

Contiene las fichas siguientes:

- **Controlador**: Contiene los controladores **Logic**, **HMI**, **Drive** y **Motion** que pueden insertarse en el proyecto de SoMachine.
- **Devices & Modules**: Contiene los **módulos de E/S** y los dispositivos de **comunicación**, **control de motores**, **seguridad** y **gestión de energía** que pueden insertarse en su proyecto de SoMachine. También permite insertar dispositivos mediante una plantilla de dispositivos.
- **HMI & iPC**: Contiene los dispositivos **HMI** e **iPC** que pueden insertarse en su proyecto de SoMachine.
- **Varios**: Contiene dispositivos de terceros que pueden insertarse en su proyecto de SoMachine.

El contenido de las diferentes fichas depende del proyecto. Si los controladores integrados en el proyecto de SoMachine no admiten, por ejemplo, CANopen, los dispositivos CANopen no se mostrarán en los catálogos.

Puede ampliar esta vista mediante las fichas de **Catálogo de software** (**Variables**, **Activos**, **Macros**, **Herramientas**, **Bibliotecas**) a través del menú **Visualizar** →**Catálogo de software**.

Los botones **Catálogo de hardware**  y **Catálogo de software**  de la barra de herramientas le permiten mostrar u ocultar las vistas de catálogo.

Puede añadir los elementos de los catálogos al proyecto simplemente arrastrándolos y soltándolos, como se describe en el capítulo *Adición de dispositivos mediante el método de arrastrar y soltar (véase página 68)*.

Búsqueda en catálogos

Cada ficha de la lista de catálogo contiene un cuadro de búsqueda. Se busca en las sublistas de la ficha la cadena que especifica en el cuadro de búsqueda. En las sublistas abiertas, las entradas encontradas se muestran en amarillo. Cualquier otro elemento de la lista que no corresponda a la cadena de búsqueda quedará oculto. El número de elementos encontrados en sublistas cerradas se muestra en negrita en la barra de título de cada sublista.

De forma predeterminada, la búsqueda se ejecuta en los nombres de los elementos de las listas. No obstante, SoMachine también da soporte al mecanismo de etiquetado. Esto le permite asignar cadenas de búsqueda de su elección a cualquier elemento incluido en la vista **Catálogo**.

Lista Favoritos

Cada ficha de la lista de catálogo contiene una lista **Favoritos**. Para proporcionar un acceso rápido, puede añadir elementos utilizados con frecuencia a esta lista **Favoritos** mediante el método de arrastrar y soltar.

Adición de dispositivos a partir de plantillas de dispositivos en la ficha Devices & Modules

La ficha **Devices & Modules** contiene la opción **Plantilla de dispositivos** en la parte inferior. Active esta opción para mostrar las plantillas disponibles de dispositivos de campo en las listas de la ficha **Devices & Modules**. Añádalas al árbol **Dispositivos** como se describe en el capítulo *Adición de dispositivos a partir de plantillas (véase página 838)*.

Personalización de la interfaz de usuario

Descripción general

El aspecto de la interfaz de usuario, en cuanto a organización y configuración de los componentes específicos, depende de lo siguiente:

- Preajustes estándar para menús, funciones de teclado y barras de herramientas. Puede sobrescribir la configuración predeterminada de SoMachine en el cuadro de diálogo *(véase SoMachine, Comandos de menú, Ayuda en línea)* **Personalizar** (disponible de forma predeterminada en el menú **Herramientas**). La configuración actual se guarda en el sistema local. Hay una función de restablecimiento para restaurar los valores predeterminados en cualquier momento.
- Las propiedades de un editor según se definen en el cuadro de diálogo *(véase SoMachine, Comandos de menú, Ayuda en línea)* **Herramientas** →**Opciones** respectivo. También puede sobrescribir esta configuración. La configuración actual se guarda en el sistema local.
- La forma en la que organiza las vistas o las ventanas del editor en el proyecto. Las posiciones actuales se guardan con el proyecto (consulte más abajo).
- La perspectiva seleccionada. De forma predeterminada, la **Configuración lógica** está seleccionada. Para obtener más información, consulte el párrafo *Perspectivas* de este capítulo *(véase página 47)*.

Organización de barras de menús y barras de herramientas

La barra de menús se sitúa en la parte superior de la interfaz de usuario, entre la barra del título de ventana y las ventanas de vista. Puede situar una barra de herramientas en la misma área como una barra de menú (fija) o como una ventana independiente en cualquier lugar de la pantalla.

En las ventanas de vista, como por ejemplo el árbol **Dispositivos**, hay disponible una barra de herramientas especial. Proporciona elementos para ordenar, ver y buscar en la ventana. No puede configurar esta barra de herramientas.

Organización de ventanas y vistas

Al cerrar una vista o una ventana del editor: haga clic en el botón de cruz situado en la esquina superior derecha.

Apertura de una vista cerrada: de forma predeterminada puede volver abrir las vistas de componentes estándar en el menú **Ver**. Para abrir una ventana del editor, ejecute el comando **Proyecto** →**Editar objeto** o haga doble clic en la entrada respectiva del árbol **Dispositivos**, **Aplicaciones** o **Herramientas**.

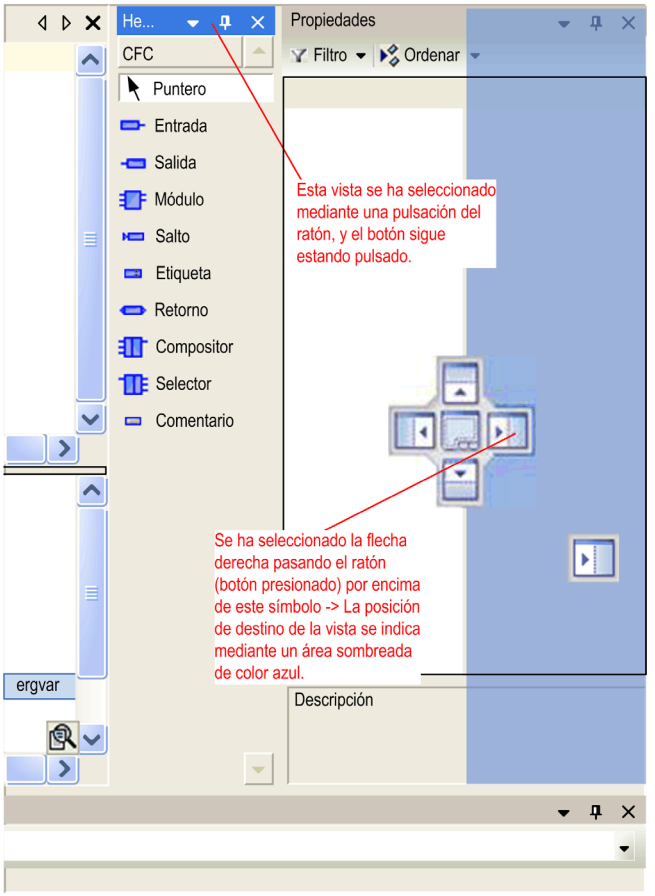
Cambio del tamaño de una vista o ventana en la ventana de marco: mueva las líneas de separador entre las vistas vecinas. Puede cambiar el tamaño de las ventanas de vistas independientes en el escritorio moviendo los bordes de la ventana.

Desplazamiento de una vista a otra posición del escritorio o dentro de la ventana de marco: haga clic en la barra de título o bien, en el caso de las vistas con fichas, en la ficha de la vista, y mueva la vista al lugar que desee. El símbolo de flecha se visualizará con cada posición de destino posible. Mantenga el ratón pulsado y elija la posición deseada moviendo el cursor en el símbolo de flecha respectivo. La posición de destino se indica mediante un área sombreada de color azul.

Símbolos de flecha que indican una posición nueva

Símbolo de flecha	Descripción
	La vista se sitúa encima.
	La vista se sitúa debajo.
	La vista se sitúa a la derecha.
	La vista se sitúa a la izquierda.
	La vista se sitúa aquí: la vista situada actualmente en esta posición y la vista nueva se organizan como iconos.

Ejemplo de navegación con los símbolos de flecha



Cuando suelta el botón del ratón, la vista se sitúa en la nueva posición.

Las vistas con un botón **Ocultar** se pueden situar como ventanas independientes (flotantes) en cualquier lugar de la pantalla; para ello, muévalas sin arrastrarlas en uno de los símbolos de flecha. En este caso, la vista pierde el botón **Ocultar**. Como alternativa, ejecute los comandos **Acoplar** y **Flotante** desde el menú **Ventana**.

Ocultación de vista: puede ocultar vistas con los botones **Ocultar** en el borde de la ventana de SoMachine. Haga clic en el botón **Ocultar** hacia abajo situado en la esquina superior derecha de la vista. La vista se mostrará en forma de ficha en el borde más cercano de la ventana de marco. El contenido de la vista sólo es visible siempre y cuando el cursor se mueva en esta ficha. La ficha muestra el icono y el nombre de la vista. Este estado de la vista se indica con el botón de acoplamiento cambiado a **Ocultar** hacia la izquierda.

Visualización de vistas ocultas: para mostrar una vista oculta, haga clic en el botón **Ocultar** hacia la izquierda.

El comando **Ocultar**, que está disponible de forma predeterminada en el menú **Ventana**, proporciona una manera alternativa de ocultar una vista y volverla a mostrar.

No se puede volver a colocar la barra de estado e información en el borde inferior de la interfaz de usuario *(véase página 31)*.

Perspectivas

Las perspectivas se utilizan para guardar el diseño de las vistas de SoMachine. Guarda si las vistas **Mensajes** y **Supervisar** están abiertas y en qué posición se sitúan las ventanas de vista (acopladas o independientes).

De forma predeterminada, SoMachine proporciona 4 perspectivas para los siguientes casos de uso en el menú **Ventana** →**Cambiar perspectiva** o en la tabla de perspectiva de la barra de herramientas.

Nombre de perspectiva	Caso de uso	Navegadores (en el lado izquierdo)	Vistas de catálogo (en el lado derecho)	Vistas en la parte inferior de la pantalla
Configuración del dispositivo	Para añadir/configurar dispositivos.	• Árbol **Dispositivos** • Árbol **Aplicaciones** • Árbol **Herramientas**	Catálogo de hardware • **Controlador** • **Devices & Modules** • **HMI & iPC** • **Varios**	**Mensajes** (en modalidad **Ocultar**)
Configuración lógica	Para añadir/crear lógica.	• Árbol **Dispositivos** • Árbol **Aplicaciones** • Árbol **Herramientas**	Catálogo de software • **Variables** • **Activos** • **Macros** • **Herramientas** • **Bibliotecas**	**Mensajes** (en modalidad **Ocultar**)
CODESYS Classic	Vistas estándar de CoDeSys	• **Dispositivos** • **POU**	Catálogo de hardware • **Controlador** • **Devices & Modules** • **HMI & iPC** • **Varios**	**Mensajes** (en modalidad **Ocultar**)
En línea	Para modalidad online.	• Árbol **Dispositivos** • Árbol **Aplicaciones** • Árbol **Herramientas**	Catálogo de hardware • **Controlador** • **Devices & Modules** • **HMI & iPC** • **Varios**	• **Mensajes** (en modalidad **Ocultar**) • **Supervisar 1**

La perspectiva **En línea** se selecciona de forma automática cuando la aplicación cambia a modalidad online.

Creación de perspectiva propia:

Además de estas perspectivas estándar, puede crear su propio diseño de vista y guardarlo en perspectivas distintas en función de sus requisitos individuales.

Para crear su propia perspectiva, haga lo siguiente:

Paso	Acción
1	Cambie el tamaño de las vistas, ábralas o ciérrelas en función de sus necesidades individuales.
2	Ejecute el comando **Guarda la disposición de la vista actual como perspectiva** en el menú **Ventana** para guardar las modificaciones en una perspectiva nueva.
3	En el cuadro de diálogo **Guarda la disposición de la vista actual como perspectiva**, introduzca un **Nombre** para la perspectiva. **Resultado**: Se guarda el diseño de la vista actual. La perspectiva nueva está disponible en el menú **Ventana** →**Cambiar perspectiva** y en la tabla de perspectiva de la barra de herramientas.

Restablecimiento de una perspectiva a su estado inicial:

Para restablecer una perspectiva modificada a su estado inicial, ejecute el comando **Restablecer perspectiva actual** en el menú **Ventana**.

Importación/exportación de perspectivas:

Para poder intercambiar perspectivas entre instalaciones de SoMachine diferentes o entre diferentes usuarios, el cuadro de diálogo *(véase SoMachine, Comandos de menú, Ayuda en línea)* **Herramientas** →**Opciones** →**Perspectivas** le permite exportar perspectivas a un archivo XML e importar los archivos XML de perspectiva ya disponibles.

Zoom

Cada ventana del editor proporciona una función de zoom. Haga clic en el botón de zoom  en la esquina inferior derecha para abrir una lista. Permite elegir uno de los niveles de porcentaje de zoom (25, 50, 100, 150, 200 y 400) o introducir el factor de zoom que prefiera. Una impresión siempre hace referencia a una vista del 100%.

La interfaz de usuario se puede personalizar en modalidad offline y online.

Interfaz de usuario en modalidad online

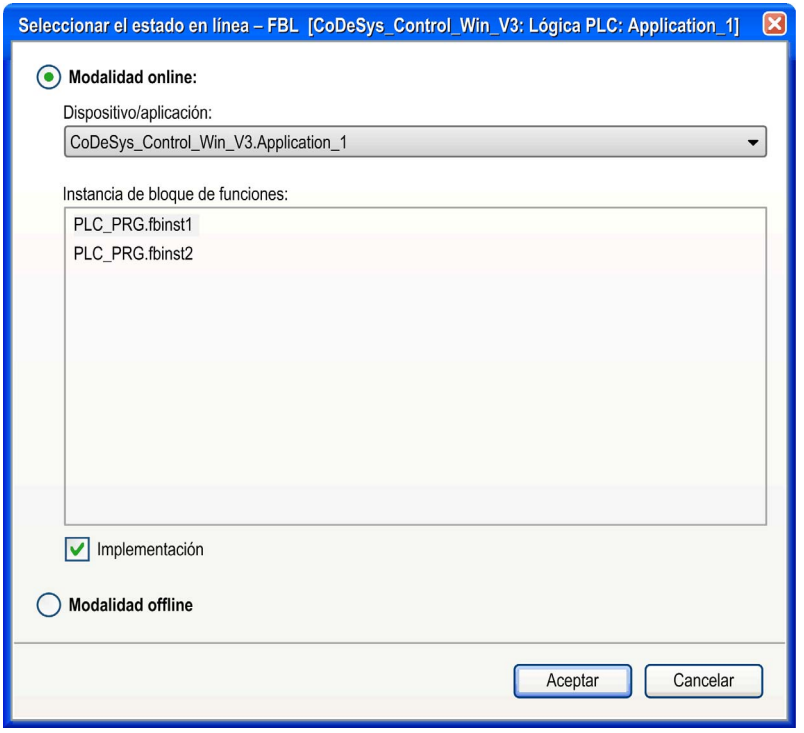
Descripción general

En cuanto inicia sesión con el proyecto, los objetos que ya se han abierto en modalidad offline se muestran automáticamente en modalidad online. La perspectiva cambia automáticamente a la perspectiva *(véase página 47)* **En línea**, lo que significa que la vista **Supervisar** se abre de forma predeterminada.

En modalidad online, para abrir un objeto que todavía no se haya abierto, haga doble clic en el nodo del árbol **Aplicaciones** o ejecute el comando **Proyecto →Modificar objeto**. El objeto se abrirá en modalidad online.

Si hay varias instancias del objeto seleccionado (como bloques de funciones) contenidas en el proyecto, aparecerá un cuadro de diálogo denominado **Seleccionar el estado en línea <nombre de objeto>**. Este permite elegir si debe visualizarse una instancia o la implementación básica del objeto, y si el objeto debe mostrarse en modalidad online u offline.

Cuadro de diálogo **Seleccionar el estado en línea**



El campo **Dispositivo/aplicación** contiene el dispositivo y la aplicación a los que está asociado el objeto correspondiente.

Para abrir la vista online del objeto, active la **modalidad online** y haga clic en **Aceptar**. Para ver la vista offline, active la **modalidad offline**.

Si el objeto es un bloque de funciones, el campo **Instancia de bloque de funciones** contendrá una lista de las instancias utilizadas actualmente en la aplicación.

En tal caso, las opciones disponibles son:

- Seleccione una de las instancias y active la **modalidad online** o la **modalidad offline**.
- O bien seleccione la opción **Implementación**, que, independientemente de la instancia seleccionada, abrirá la vista de implementación básica del bloque de funciones. La opción **Implementación** no tiene ningún efecto en los objetos no instanciados.

Para obtener más información sobre las vistas online de cada editor, consulte la descripción del editor correspondiente.

La barra de estado *(véase página 31)* proporciona información sobre el estado actual de la aplicación.

Menús y comandos estándar

Descripción general

En este capítulo se proporciona una descripción general de la estructura predeterminada de los menús y comandos principales.

Puede haber más comandos disponibles y se pueden insertar comandos adicionales en estos menús o en los menús definidos por el usuario. Por ejemplo, los comandos propios de un editor determinado suelen estar disponibles en el menú correspondiente. Estos comandos están disponibles cuando el editor está abierto.

Ejemplo: Cuando edita un objeto en el editor SFC, el menú **SFC** se añade a la barra de menús.

Si desea reorganizar las estructuras de menú, use el cuadro de diálogo **Herramientas** → **Personalizar**.

En la siguiente figura se muestra la barra de menús estándar:

Menú Archivo

Varios comandos del menú **Archivo** no están disponibles porque se llevan a cabo estas tareas en SoMachine Central. Para obtener más información, consulte SoMachine Central - Guía del usuario *(véase SoMachine Central, Manual del usuario)*.

El menú **Archivo** contiene los comandos siguientes:

Símbolo	Comando	Método abreviado
–	**Importar proyecto de Vijeo-Designer**	–
–	**Importar proyecto de Vijeo-Designer**	–
–	**Cargar el código de origen desde el control...**	–
–	**Descarga de origen...**	–
	**Imprimir...**	–
	**Configuración de página...**	–

Menú Editar

El menú **Editar** contiene comandos para trabajar en los editores (editores de lenguajes, editor de declaraciones).

Símbolo	Comando		Método abreviado
	**Deshacer**		CTRL + Z
	**Rehacer**		CTRL + Y
	**Cortar**		MAYÚS + SUPR
	**Copiar**		CTRL + INSERT
	**Pegar**		MAYÚS + INSERT
	**Eliminar**		SUPR
–	**Seleccionar todo**		CTRL + A
–	**Buscar/Reemplazar**		–
		**Buscar**	CTRL + F
		**Reemplazar**	CTRL + H
		**Buscar siguiente**	F3
–		**Buscar siguiente (seleccionado)**	CTRL + F3
		**Buscar anterior**	SHIFT + F3
–		**Buscar anterior (seleccionado)**	MAYÚS + CTRL + F3
–	**Examinar**		–
		**Ir a la definición**	–
		**Emitir referencias cruzadas**	–
–	**Insert File As Text**		–
–	**Avanzado**		–
–		**Modalidad de sobrescritura**	INS

Símbolo	Comando		Método abreviado
–		**Ir a la línea...**	–
–		**Make Uppercase**	MAYÚS + CTRL + C
–		**Make Lowercase**	CTRL + V
–		**Go To Matching Bracket**	–
–		**Select To Matching Bracket**	–
–	**Marcadores**		–
		**Activar/desactivar marcador**	CTRL + F12
		**Marcador siguiente**	F12
		**Marcador anterior**	MAYÚS + F12
		**Borrar marcadores**	–
	**Accesibilidad...**		F2
–	**Buscador FFB...**		–
–	**Declarar variable...**		MAYÚS + F2
–	**Mensaje siguiente**		F4
–	**Mensaje anterior**		MAYÚS + F4
–	**Ir a la posición del código de origen**		–

Menú Ver

El menú **Ver** contiene comandos para activar las vistas estándar particulares. Se mostrarán en una ventana en la interfaz de usuario. Tenga en cuenta también el menú *(véase página 60)* **Ventana**

Símbolo	Comando		Método abreviado
–	**Navegadores**		–
		**Dispositivos**	–
		**Aplicaciones**	–
		**Herramientas**	–
–	**Navegadores clásicos**		–

Símbolo	Comando		Método abreviado
		**Dispositivos**	–
		**POU**	–
–	**Catálogo de hardware**		–
		**Controlador**	–
		**HMI & iPC**	–
		**Devices & Modules**	–
		**Varios**	–
–	**Catálogo de software**		–
		**Variables**	–
		**Activos**	–
		**Macros**	–
		**Herramientas**	–
		**Bibliotecas**	–
	**Vista de información de catálogo**		–
	**Todas las variables**		–
	**Mensajes**		ALT + 2
	**Propiedades del elemento** para elemento SFC para elementos de visualización		–
–	**Supervisar**		–
		**Supervisar 1...4**	–

Símbolo	Comando		Método abreviado
		**Ver todos los forzados**	–
	**Puntos de interrupción**		–
	**Pila de llamadas**		–
	**Lista de referencias cruzadas**		–
	**Propiedades...**		–

Menú Proyecto

El menú **Proyecto** contiene comandos para la gestión de los objetos de proyecto y la información general sobre el proyecto, la copia (fusión) y exportación de proyectos, así como el control de código de origen y la administración de usuarios.

Símbolo	Comando		Método abreviado
	**Agregar objeto**		–
		**Acción...**	–
		**Propiedad...**	–
		**Transición...**	–
–	**Agregar dispositivo...**		–
	**Convertir dispositivo...**		
–	**Buscar dispositivos...**		–
–	**Actualizar dispositivo...**		–
	**Agregar carpeta...**		–
	**Modificar objeto**		–
–	**Modificar el objeto con...**		–

Símbolo	Comando		Método abreviado
–	**Establecer la aplicación activa**		–
	**Información del proyecto...**		–
	**Configuración del proyecto...**		–
	**Documentar...**		–
	**Comparar...**		–
–	**Exportar PLCOpenXML...**		–
–	**Importar PLCOpenXML...**		–
–	**Exportar...**		–
–	**Importar...**		–
–	**Administración de usuarios**		–
		**Inicio de sesión usuario...**	–
		**Cerrar sesión usuario**	–
–		**Permisos...**	–
–	**Almacenamiento masivo (USB o tarjeta SD)...**		–

Menú Compilar

El menú **Compilar** contiene comandos para compilar el proyecto, es decir, para realizar una ejecución previa a la compilación que incluye una comprobación sintáctica. Comandos para eliminar la información de compilación más reciente (limpiar) que cubre la generación de **Cambio en línea** y código offline.

Símbolo	Comando	Método abreviado
–	**Generar todo**	–
	**Compilar**	F11
–	**Compilar de nuevo**	–
–	**Crear código**	–
–	**Crear archivos del sistema en tiempo de ejecución...**	–
–	**Generar la configuración de Post...**	–
–	**Limpiar**	–
–	**Limpiar todo**	–

Menú En línea

El menú **En línea** contiene comandos para iniciar y cerrar sesión en el controlador, cargar el proyecto en el controlador y reiniciar.

Símbolo	Comando	Método abreviado
	**Inicio de sesión**	ALT + F8
	**Cerrar sesión**	CTRL + F8
–	**Crear aplicación de inicio**	–
	**Terminar la sesión del usuario en línea actual**	–
–	**Descarga**	–
–	**Cambio en línea**	–
–	**Cargar el código de origen desde el control...**	–
–	**Escribir el código de origen en el control conectado**	–
–	**Descarga múltiple...**	–
–	**Reset caliente**	–
–	**Reset frío**	–
–	**Reset origen**	–

Símbolo	Comando		Método abreviado
–	**Simulación**		–
–	**Advanced Configuration**		–
–		**Restaurar datos de archivo CSV...**	–
–		**Guardar datos en archivo CSV...**	–

Menú Debug

El menú **Debug** contiene comandos para controlar la ejecución del programa en el controlador (iniciar, detener) y llevar a cabo acciones de depuración (puntos de interrupción, ejecución paso a paso, escritura y forzado).

Símbolo	Comando		Método abreviado
	**Iniciar**		F5
	**Detener**		MAYÚS + F8
–	**Ciclo individual**		CTRL + F5
	**Nuevo punto de interrupción...**		–
–	**Alternar punto de interrupción...**		F9
	**Paso a paso por función**		F10
	**Paso a paso**		F8
	**Paso a paso para salir**		MAYÚS + F10
	**Ejecutar hasta el cursor**		–
	**Definir la siguiente instrucción**		–
	**Mostrar la siguiente instrucción**		–
–	**Escribir valores**		CTRL + F7
–	**Forzar valores, cuadro de diálogo Preparar valor**		F7

Símbolo	Comando		Método abreviado
–	**Anular forzado para los valores**		ALT + F7
–	**Control de proceso**		–
–	**Modo de visualización**		–
–		**Binario**	–
–		**Decimal**	–
–		**Hexadecimal**	–

Menú Herramientas

El menú **Herramientas** contiene comandos para abrir herramientas que sirven para preparar el entorno para trabajar en un proyecto (como la instalación de bibliotecas y dispositivos, la personalización de la interfaz de usuario, las opciones de los editores, cargar y guardar).

Símbolo	Comando		Método abreviado
–	**Monitor de DTM de SoMachine...**		–
	**Repositorio de bibliotecas...**		–
	**Repositorio de plantillas...**		–
	**Repositorio de dispositivos...**		–
	**Repositorio de estilos de visualización…**		–
–	**Automatización**		–
		**Execute Script File...**	–
		**Enable Script Tracing**	–
–	**OPC [Mi_controlador]...**		–
–	**Personalizar...**		–
–	**Opciones...**		–

Menú Ventana

El menú **Ventana** contiene comandos para organizar las ventanas en la interfaz de usuario (como la posición, abrir, cerrar). Consulte también el menú *(véase página 53)* **Ver**.

Símbolo	Comando		Método abreviado
–	**Siguiente editor**		CTRL + F6
–	**Editor anterior**		CTRL + MAYÚS + F6
	**Cerrar todos los editores**		–
–	**Restablecer diseño de ventanas**		–
	**Cambiar perspectiva**		–
–		**En línea**	–
–		**CODESYS Classic**	–
–		**Configuración lógica**	–
–		**Configuración del dispositivo**	–
–	**Guardar perspectiva**		–
–	**Restablecer perspectiva actual**		–
	**Nuevo grupo horizontal de fichas**		–
	**Nuevo grupo vertical de fichas**		–
–	**Flotante**		–
–	**Acoplar**		–
–	**Ocultar**		–
–	**Panel siguiente**		F6
–	**Panel anterior**		MAYÚS + F6
símbolo del editor	Ventana <n>		–
–	**Ventanas...**		–

Menú Ayuda

El menú **Ayuda** contiene comandos para obtener ayuda online e información sobre el sistema de programación.

Símbolo	Comando	Método abreviado
	**Contenido...**	CTRL + MAYÚS + F1
	**Índice...**	CTRL + MAYÚS + F2
	**Buscar...**	–
–	**Acerca de...**	–

Capítulo 3
Conceptos básicos

Introducción y conceptos básicos

Descripción general

SoMachine es un sistema de programación de controladores independiente del dispositivo.

De acuerdo con la norma IEC 61131-3, admite todos los lenguajes de programación estándar. Además, permite incluir rutinas C. Permite programar múltiples dispositivos de controlador en un mismo proyecto.

Para obtener más información, consulte el capítulo ***Crear archivos del sistema de tiempo de ejecución*** *(véase SoMachine, Comandos de menú, Ayuda en línea)*.

Orientación a objetos

La esencia de la orientación a objetos queda patente no sólo en la disponibilidad de elementos y funciones de programación adecuados, sino también en la gestión de estructuras y versiones de SoMachine y en la organización del proyecto. La utilización multidispositivo de un proyecto de SoMachine se basa en el uso conjunto de unidades de programación instanciadas. Puede clonar aplicaciones, así como combinar dispositivos de controlador configurables y programables en un solo proyecto.

Tratamiento de la versión

Puede realizarse la instalación simultánea de diversas versiones de componentes de SoMachine y trabajar con la combinación de versiones que se desee. Esto también es aplicable al uso de distintas versiones de compilador específicas del dispositivo. Se pueden añadir funciones individuales sin tener que actualizar toda la versión.

Para obtener información más detallada, consulte *SoMachine - Compatibilidad y migración - Guía del usuario*.

Organización del proyecto

La organización del proyecto también se basa en la orientación a objetos. Un proyecto de SoMachine contiene un programa de controlador compuesto por diversos objetos de programación, e incluye definiciones de los recursos necesarios para ejecutar instancias del programa (aplicación) en sistemas de destino (dispositivos, controladores) específicos.

Así, hay dos tipos principales de objetos en un proyecto:

Tipo de objeto	Descripción
Objetos de programación (POU) *(véase página 169)*	Se trata de programas, funciones, bloques de funciones, métodos, interfaces, acciones, tipo de datos, definiciones, etc. Los objetos de programación que se pueden instanciar en todo el proyecto, es decir, para todas las aplicaciones definidas en el proyecto, se deben gestionar en el nodo **Global** del árbol **Aplicaciones**. La instanciación se realiza llamando a una POU de programa mediante una tarea asignada por la aplicación. Los objetos de programación que sólo se gestionan en el árbol **Aplicaciones**, es decir, que se asignan directamente a una aplicación, no se pueden instanciar únicamente mediante otra aplicación insertada debajo.
Objetos de recurso (**Dispositivos**)	Los objetos de dispositivo sólo se gestionan en **Dispositivos**. Cuando inserte objetos en **Dispositivos**, tenga en cuenta las recomendaciones incluidas en la sección *Adición de elementos a los navegadores (véase página 40)*.

Generación de código

La generación de código mediante compiladores integrados y el posterior uso del código de máquina resultante ofrecen un tiempo de ejecución breve.

Transferencia de datos al dispositivo controlador

La transferencia de datos entre SoMachine y el dispositivo se realiza mediante una puerta de enlace (componente) y un sistema de tiempo de ejecución. Se proporciona una funcionalidad online completa para controlar un programa en el dispositivo.

Lenguajes de programación compatibles

Se admiten los lenguajes de programación indicados en el estándar IEC IEC 61131 mediante editores adaptados específicamente:

- FBD/LD/IL editor *(véase página 279)* para diagrama de bloques de funciones (FBD), diagrama de contactos (LD) y lista de instrucciones (IL)
- Editor SFC *(véase página 353)* para diagrama funcional secuencial
- Editor ST *(véase página 389)* para texto estructurado

Además, SoMachine proporciona un editor para programación en CFC que no forma parte del estándar IEC:

- Editor CFC *(véase página 333)* para diagrama de función continua

CFC es una extensión de los lenguajes de programación del estándar IEC.

Parte II
Configuración

Contenido de esta parte

Esta parte contiene los siguientes capítulos:

Capítulo 4
Gestión de dispositivos

Contenido de este capítulo

Este capítulo contiene las siguientes secciones:

Sección 4.1
Adición de dispositivos mediante el método de arrastrar y soltar

Adición de dispositivos mediante el método de arrastrar y soltar

Descripción general

SoMachine V4.0 y sus versiones posteriores proporcionan una vista de catálogo con varias fichas a la derecha de SoMachine Logic Builder.

Están disponibles 2 tipos diferentes de vistas de catálogo:
- el **Catálogo de hardware**
- el **Catálogo de software**

Para añadir un dispositivo al árbol **Dispositivos**, seleccione la entrada respectiva en **Catálogo de hardware**, arrástrela al árbol **Dispositivos** y suéltela en un nodo adecuado. Se añadirá automáticamente a su proyecto.

Adición de controladores mediante el método de arrastrar y soltar

Para añadir un controlador al proyecto, haga lo siguiente:

Paso	Acción
1	Para abrir el **Catálogo de hardware**, haga clic en el botón **Catálogo de hardware**  de la barra de herramientas de SoMachine Logic Builder si todavía no está abierto.
2	Seleccione la ficha **Controlador** en el **Catálogo de hardware**. **Resultado**: Los controladores adecuados para el proyecto de SoMachine se muestran en el **Catálogo de hardware**.
3	Seleccione una entrada de controlador en la ficha **Controlador**, arrástrela al árbol **Dispositivos** y suéltela en un nodo adecuado. Puede soltar el controlador en un espacio vacío dentro del árbol **Dispositivos**. **Resultado:** El controlador se añade al árbol **Dispositivos** como nuevo nodo con diferentes subnodos en función del tipo de controlador.

Adición de dispositivos de ampliación mediante el método de arrastrar y soltar

Para añadir un dispositivo de ampliación a un controlador, haga lo siguiente:

Paso	Acción
1	Para abrir el **Catálogo de hardware**, haga clic en el botón **Catálogo de hardware**  de la barra de herramientas de SoMachine Logic Builder si todavía no está abierto.
2	Seleccione la ficha **Devices & Modules** en el **Catálogo de hardware**. **Resultado**: Los dispositivos de ampliación adecuados para el proyecto de SoMachine se muestran en el **Catálogo de hardware**.
3	Seleccione el dispositivo de ampliación, arrástrelo al árbol **Dispositivos** y suéltelo en el subnodo adecuado de un controlador. **NOTA:** SoMachine expande y resalta los subnodos adecuados. **Resultado:** El dispositivo de ampliación se añade al árbol **Dispositivos**, debajo del subnodo del controlador.
4	Si el dispositivo de ampliación requiere un administrador de comunicación, este nodo se añade automáticamente al árbol **Dispositivos**. Si hay disponibles varios administradores de comunicación para ese dispositivo de ampliación, se muestra un cuadro de diálogo que permite seleccionar el administrador de comunicación adecuado.

Adición de dispositivos y módulos mediante el método de arrastrar y soltar

Para añadir un dispositivo de campo a un controlador, haga lo siguiente:

Paso	Acción
1	Para abrir el **Catálogo de hardware**, haga clic en el botón **Catálogo de hardware**  de la barra de herramientas de SoMachine Logic Builder si todavía no está abierto.
2	Seleccione la ficha **Devices & Modules** en el **Catálogo de hardware**. **Resultado**: Los dispositivos de campo adecuados para el proyecto de SoMachine se muestran en el **Catálogo de hardware**.
3	Seleccione una entrada de dispositivo de campo en la vista de catálogo **Devices & Modules**, arrástrela al árbol **Dispositivos** y suéltela en el subnodo adecuado de un controlador. **NOTA:** SoMachine expande y resalta los subnodos adecuados. **Resultado:** El dispositivo de campo se añade al árbol **Dispositivos**, debajo del subnodo del controlador.
4	Si el dispositivo de campo requiere un administrador de comunicación, este nodo se añade automáticamente al árbol **Dispositivos**. Si hay disponibles varios administradores de comunicación para ese dispositivo de campo, se muestra un cuadro de diálogo que permite seleccionar el administrador de comunicación adecuado.

Adición de dispositivos a partir de plantillas de dispositivos mediante el método de arrastrar y soltar

Para añadir un dispositivo a partir de una plantilla de dispositivos, haga lo siguiente:

Paso	Acción
1	Para abrir el **Catálogo de hardware**, haga clic en el botón **Catálogo de hardware**  de la barra de herramientas de SoMachine Logic Builder si todavía no está abierto.
2	Seleccione la ficha **Devices & Modules** en el **Catálogo de hardware**.
3	Seleccione la opción **Plantilla de dispositivos** en la parte inferior de la ficha **Devices & Modules**. **Resultado**: Las plantillas de dispositivos adecuadas para el proyecto de SoMachine se muestran en la ficha **Devices & Modules**.
4	Añádalas al árbol **Dispositivos** como se describe en el capítulo *Adición de dispositivos a partir de plantillas (véase página 838)*.

Adición de dispositivos a partir de plantillas de funciones mediante el método de arrastrar y soltar

Para añadir un dispositivo a partir de una plantilla de funciones, haga lo siguiente:

Paso	Acción
1	Para abrir el **Catálogo de software**, haga clic en el botón  de la barra de herramientas de SoMachine Logic Buildersi todavía no está abierto.
2	Seleccione la ficha **Macro** en el **Catálogo de software**. **Resultado**: Las plantillas de funciones disponibles en SoMachine se muestran en el **Catálogo de software**.
3	Seleccione una entrada de plantilla de funciones en la vista **Macro**, arrástrela al árbol **Dispositivos** y suéltela en el subnodo adecuado de un controlador. **NOTA:** SoMachine expande y resalta los subnodos adecuados. **Resultado:** El dispositivo basado en la plantilla de funciones se añade al árbol **Dispositivos**.

Sección 4.2
Adición de dispositivos mediante menú contextual o signo más

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Adición de un controlador

Introducción

Como alternativa al método de arrastrar y soltar los dispositivos en el árbol **Dispositivos**, haga clic en el signo más de color verde que se muestra en el nodo pertinente del **árbol**. Otra alternativa es hacer clic con el botón derecho en un nodo del **árbol** para añadir un dispositivo adecuado mediante el menú contextual. Se abrirá el cuadro de diálogo **Agregar dispositivo**, que le permite determinar si el dispositivo se adjuntará, insertará o conectará al nodo seleccionado *(véase SoMachine, Comandos de menú, Ayuda en línea)*.

Cuando se agrega un controlador a su proyecto, automáticamente se agregan muchos nodos al árbol **Dispositivos**. Estos subnodos son específicos de cada controlador, dependiendo de las funciones que proporcione el controlador.

El siguiente párrafo describe el procedimiento general para añadir un controlador. Para obtener más información acerca de un controlador en concreto, consulte el manual de programación de ese controlador.

Adición de un controlador

Para añadir un dispositivo al proyecto de SoMachine, haga lo siguiente:

Paso	Acción
1	Seleccione un nodo de proyecto y haga clic en el signo más de color verde del nodo, o haga clic con el botón derecho en el nodo del proyecto y seleccione el comando **Agregar dispositivo...** en el menú contextual. **Resultado:** Se abre el cuadro de diálogo **Agregar dispositivo**.
2	En el cuadro de diálogo **Agregar dispositivo**, seleccione **Schneider Electric** en el cuadro de lista **Fabricante**.
3	Seleccione el controlador que desea insertar en el proyecto.
4	Para cambiar el nombre del dispositivo, escríbalo en el cuadro de texto **Nombre**. **NOTA:** Elija un nombre que cumpla con el estándar IEC. No use caracteres especiales, cifras antepuestas ni espacios en el nombre. El nombre no debe tener más de 32 caracteres. Si no cambia el nombre del dispositivo, se le asignará uno predeterminado. Asignar un nombre significativo al dispositivo puede facilitar la organización del proyecto.
5	Haga clic en el botón **Agregar dispositivo**. **Resultado:** El controlador seleccionado se añade al proyecto y aparece como un nodo nuevo en el árbol **Dispositivos**. El cuadro de diálogo **Agregar dispositivo** permanece abierto. Puede hacer lo siguiente: ● Para añadir otro controlador, vuelva al paso 3. ● O bien haga clic en el botón **Cerrar** para cerrar el cuadro de diálogo **Agregar dispositivo**.

Adición de dispositivos de ampliación

Dispositivos de ampliación disponibles

Para obtener una lista de los dispositivos de ampliación disponibles para los diferentes controladores, consulte el capítulo sobre *dispositivos admitidos* del documento de introducción a *SoMachine*.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

- Utilice solo software aprobado por Schneider Electric para este equipo.
- Actualice el programa de aplicación siempre que cambie la configuración de hardware física.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

Adición de dispositivos de ampliación

Para añadir dispositivos de ampliación al dispositivo, haga lo siguiente:

Paso	Acción
1	Seleccione un nodo de controlador y haga clic en el signo más de color verde del nodo, o haga clic con el botón derecho del ratón en el nodo de controlador y seleccione el comando **Agregar dispositivo...** en el menú contextual. **Resultado:** Se abre el cuadro de diálogo **Agregar dispositivo**.
2	En el cuadro de diálogo **Agregar dispositivo**, seleccione **Schneider Electric** en la lista **Fabricante**.
3	Elija el dispositivo de ampliación que desee añadir al controlador en la lista **Dispositivo** que aparece a continuación.
4	Para cambiar el nombre del dispositivo de ampliación, escríbalo en el cuadro de texto **Nombre**. **NOTA:** El nombre no puede contener espacios. Si no cambia el nombre del dispositivo de ampliación, se le asignará uno de forma predeterminada. Asignar un nombre significativo al dispositivo de ampliación puede facilitar la organización del proyecto.
5	Haga clic en el botón **Agregar dispositivo**. **Resultado**: El dispositivo de ampliación seleccionado se añade al proyecto y se muestra en el árbol **Dispositivos** como un nuevo subnodo del controlador. El cuadro de diálogo **Agregar dispositivo** permanece abierto. Puede hacer lo siguiente: ● Puede añadir otro dispositivo de ampliación volviendo al paso 3 de esta descripción. ● También puede hacer clic en el botón **Cerrar**.

NOTA: Cuando se añade un objeto de TWDNOI10M3 (AS-Interface Master Module) se insertará automáticamente el correspondiente administrador del bus de campo **Virtual AS interface bus**. Para obtener más información sobre la configuración de AS Interface, consulte el capítulo correspondiente en Modicon M238 Logic Controller - Guía de programación *(véase Modicon M238 Logic Controller, Guía de programación)*.

Configuración de dispositivos de ampliación

Para obtener más información sobre la configuración, consulte la *Guía de programación* del dispositivo de ampliación.

Adición de administradores de comunicación

Descripción general

Los administradores de comunicación son necesarios para activar y configurar cualquier interfaz de bus de hardware, como por ejemplo CANopen o línea serie.

Existen 2 tipos de administradores de comunicación:

- Los administradores de buses de campo, que permiten configurar dispositivos de bus de campo (por ejemplo, esclavos CANopen o esclavos Modbus).
- Los administradores de comunicación generales.

A continuación, se enumeran los administradores de comunicación disponibles en SoMachine:

Nombre	Tipo de interfaz	Descripción
Gestor ASCII	Línea serie	Se utiliza para transmitir o recibir datos con un dispositivo simple.
Administrador de red de SoMachine	• Línea serie (máx. 1) • Ethernet (máx. 3)	Debe utilizarlo si desea conectar un XBTGT, XBTGK, XBTGH o SCU HMI a través del protocolo de SoMachine para ofrecer un intercambio transparente de datos y capacidad de descarga múltiple (descarga de controlador y aplicaciones HMI a través de 1 única conexión del controlador al PC o de HMI al PC). Hay disponibles 4 conexiones como máximo: 1 para SoMachine (incluso si se utiliza una conexión USB) y 3 para Ethernet.
Modbus IOScanner	Línea serie	Administrador de protocolo Modbus RTU o ASCII utilizado para definir intercambios implícitos (exploración de E/S) con dispositivos esclavos de Modbus.
Administrador de Modbus	Línea serie	Se utiliza para el protocolo Modbus RTU o ASCII en modo maestro o esclavo.
CANopen optimizado	CAN	Administrador de CANopen para controladores optimizados (M238, M241, XBTGC, XBTGT, XBTGK, SCU HMI, ATV IMC).
Rendimiento de CANopen	CAN	Administrador de CANopen para controladores de rendimiento (M251, M258, LMC058 y LMC078).
CANmotion	CAN	Administrador de CANmotion únicamente para el puerto CAN1 de Motion Controller LMC058 y LMC078.
Dispositivo esclavo de Modbus TCP	Ethernet	Administrador de Modbus TCP para controladores con puerto Ethernet.
EtherNet/IP	Ethernet	Administrador de EtherNet/IP para controladores con puerto Ethernet (M251, M258, LMC058 y LMC078).

Adición del administrador de comunicación

Los administradores de comunicación se añaden de forma automática con el dispositivo respectivo.

Para añadir un administrador de comunicación aparte, haga lo siguiente:

Paso	Acción
1	En el árbol **Dispositivos**, seleccione la interfaz de bus (**Línea serie**, **CAN0**, **CAN1**, **Ethernet**) y haga clic en el signo más de color verde del nodo o haga clic con el botón derecho en el nodo de interfaz de bus y ejecute el comando **Agregar dispositivo** en el menú contextual. **Resultado**: Se abre el cuadro de diálogo **Agregar dispositivo**.
2	En el cuadro de diálogo **Agregar dispositivo**, seleccione **Schneider Electric** en el cuadro de lista **Fabricante**. **Nota**: Puede ordenar los dispositivos por marca haciendo clic en el cuadro de lista **Fabricante**.
3	Seleccione el **Communication manager** en la lista siguiente.
4	Cambie el nombre del dispositivo escribiéndolo en el cuadro de texto **Nombre**. **Nota**: No utilice espacios en el nombre. Si no cambia el nombre del dispositivo, se le asignará uno predeterminado. Asignar un nombre significativo al dispositivo puede facilitar la organización del proyecto.
5	Haga clic en el botón **Agregar dispositivo**.
6	Haga clic en el botón **Cerrar** para cerrar el cuadro de diálogo **Agregar dispositivo**.
7	Configure el **administrador de comunicación**.

Adición de dispositivos a un administrador de comunicación

Descripción general

Se pueden añadir dispositivos de campo al administrador de comunicación seleccionando el nodo del administrador de dispositivos de campo (por ejemplo, administrador de CANopen o de Modbus) en el árbol de **Dispositivos** y haciendo clic en el signo más de color verde. Como alternativa, puede hacer clic con el botón derecho en el nodo del administrador de dispositivos de campo en el árbol **Dispositivos** y ejecutar el comando **Agregar dispositivo**.

Un requisito previo es que el dispositivo debe estar disponible en el cuadro de diálogo *(véase SoMachine, Comandos de menú, Ayuda en línea)* **Repositorio de dispositivos**.

Adición de dispositivos

Paso	Acción
1	Seleccione el nodo del administrador de dispositivos de campo (administrador de CANopen o de Modbus) en el árbol **Dispositivos** y haga clic en el signo más de color verde, o bien haga clic con el botón derecho en el nodo del administrador de dispositivos de campo y seleccione el comando **Agregar dispositivo...** en el menú contextual. **Resultado**: Se abre el cuadro de diálogo **Agregar dispositivo**.
2	En el cuadro de diálogo **Agregar dispositivo**, seleccione **Schneider Electric** en el cuadro de lista **Fabricante**. **Nota**: Puede ordenar los dispositivos por marca haciendo clic en el cuadro de lista **Fabricante**.
3	Seleccione el dispositivo que desee en la lista siguiente.
4	Cambie el nombre del dispositivo escribiéndolo en el cuadro de texto **Nombre**. **Nota**: No utilice espacios en el nombre. Si no cambia el nombre del dispositivo, se le asignará uno predeterminado. Asignar un nombre significativo al dispositivo puede facilitar la organización del proyecto.
5	Haga clic en el botón **Agregar dispositivo**. **Resultado**: Se añadirá el dispositivo de campo al administrador de dispositivos de campo. **NOTA:** El cuadro de diálogo **Agregar dispositivo** permanece abierto. Puede hacer lo siguiente: ● Para añadir otro dispositivo, vuelva al paso 2. ● Puede hacer clic en el botón **Cerrar**.

Acceso a la información de diagnóstico

Para obtener información de diagnóstico de dispositivos en CANopen utilice CAA_CiA405.library.

Acceso al diagnóstico de configuración (para usuarios avanzados)

Puede usar las opciones **Cancelar en caso de error** y **En caso de error saltar a línea** en la ficha **Service Data Object** del configurador de CANopen para gestionar posibles incoherencias de configuración.

Para optimizar el rendimiento del maestro CAN, los diagnósticos de CAN son externos al maestro CAN del controlador. La estructura de diagnóstico CAN se define en la biblioteca CanConfig Extern, disponible en el **Administrador de bibliotecas**.

La estructura `g_aNetDiagnosis` contiene la información de diagnóstico más reciente de los esclavos. La estructura se actualiza cada vez que se configura un esclavo, por cualquier motivo.

Esta estructura se puede utilizar dentro del programa para hacer lo siguiente:
- Supervisar la respuesta de los esclavos configurados por mensajes SDO.
- Supervisar el maestro para ver si hay mensajes de cancelación de los esclavos antes de permitir un arranque de máquina/aplicación.

Esta estructura debe estar instalada y activa dentro de la aplicación de usuario al realizar pruebas, depurar y poner en marcha la aplicación. Cuando la máquina y su aplicación de control ya se han puesto en marcha y validado, entonces sería posible deshabilitar este código de la ejecución para reducir el tráfico en la red CANopen.

Sin embargo, si durante el ciclo vital de una aplicación (y de la máquina o proceso que ésta controla), se añaden o sustituyen esclavos en el sistema operativo, la estructura de diagnóstico debería continuar activa en la aplicación.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

- Utilice la estructura de datos g_aNetDiagnosis para supervisar las respuestas del esclavo CAN a los comandos de configuración.
- Compruebe que la aplicación no se inicie ni ponga la máquina o el proceso en un estado operativo, en caso de recibir mensajes de cancelación de SDO de cualquiera de los esclavos CAN.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

Tras añadir la biblioteca CanConfig Extern a su aplicación, utilice la definición **Net Diagnostic** de su aplicación para probar los mensajes de cancelación de SDO desde los esclavos CAN.

El ejemplo de código siguiente ilustra el uso de la estructura de datos de diagnóstico de CAN:

```
IF g_aNetDiagnosis[CAN_Net_Number].ctSDOErrorCounter = 0 THEN
    (* No se detecta ningún error en la configuración*)
ELSE
    (* Se ha detectado un error durante la configuración. Obtener la
información más reciente.*)
    // ID de nodo del esclavo que envía el código de anulación
    ReadLastErrorNodeID := g_aNetDiagnosis[CAN_Net_Number].usiNodeID;
    // índice usado en el SDO cancelado
    ReadLastErrorIndex := g_aNetDiagnosis[CAN_Net_Number].wIndex;
    // subíndice usado en el SDO cancelado
    ReadLastErrorSubIndex := g_aNetDiagnosis[CAN_Net_Number].bySubIndex
;
    //Código de cancelación de SDO
    ReadLastErrorSdoAbortCode := g_aNetDiagnosis [CAN_Net_Number].udiAb
ortCode;
    (* No permitir el inicio de otra operación de la máquina o proceso *)
END_IF
```

NOTA: En este ejemplo, `CAN_Net_Number` sería 0 para el puerto CAN0 y, si el controlador está equipado de esa manera, 1 para CAN1port.

Adición de dispositivos a partir de plantillas

Descripción general

También es posible añadir un dispositivo nuevo con una plantilla de dispositivos. Para obtener una descripción de este procedimiento, consulte la sección *Administración de plantillas de dispositivos* *(véase página 838)*.

Sección 4.3
Actualización de dispositivos

Actualización de dispositivos

Introducción

La función de actualizar dispositivos le permite reemplazar un dispositivo seleccionado en el árbol **Dispositivos**

- por otra versión del mismo dispositivo o
- por un tipo de dispositivo diferente.

Actualización de dispositivos

Para reemplazar un dispositivo de su proyecto de SoMachine por otra versión o por un dispositivo diferente, proceda de la siguiente manera:

Paso	Acción
1	Seleccione el dispositivo que desea reemplazar en el árbol **Dispositivos** y ejecute el comando **Actualizar dispositivo...** en el menú **Proyecto**. O Haga clic con el botón derecho en el dispositivo que desea reemplazar en el árbol **Dispositivos** y seleccione el comando **Actualizar dispositivo...** en el menú contextual. **Resultado**: Se abre el cuadro de diálogo **Actualizar dispositivo**. O Haga clic con el botón derecho en el dispositivo que desea reemplazar en el árbol **Dispositivos** y seleccione el comando **Agregar dispositivo...** en el menú contextual. En el cuadro de diálogo **Agregar dispositivo**, seleccione **Acción: Actualizar el dispositivo**. **Resultado**: El cuadro de diálogo **Agregar dispositivo** se convierte en el cuadro de diálogo **Actualizar dispositivo**.
2	Desde la lista **Dispositivo:**, elija el dispositivo que debe reemplazar al dispositivo actual. Para seleccionar una versión específica del dispositivo, seleccione las opciones **Mostrar todas las versiones (sólo para expertos)** y/o **Mostrar versiones antiguas**.
3	Si es necesario distinguir los dispositivos, cambie el nombre de su dispositivo escribiendo el nuevo nombre en el cuadro de texto **Nombre**. De lo contrario, se utilizará el mismo nombre para el dispositivo actualizado. Si el dispositivo se actualiza por un tipo de dispositivo diferente, entonces la descripción del tipo de dispositivo (entre paréntesis después del nombre del dispositivo) se adaptará automáticamente. Asignar un nombre significativo al dispositivo puede facilitar la organización del proyecto.
4	Haga clic en el botón **Actualizar dispositivo**. **Resultado:** El dispositivo que se ha seleccionado en el árbol **Dispositivos** se reemplaza por el nuevo tipo de dispositivo o la nueva versión. El nuevo tipo de dispositivo o la nueva versión se muestra ahora en el nodo seleccionado en el árbol **Dispositivos**.

Efectos después de la actualización de un dispositivo

Los subdispositivos que se encuentran en el árbol **Dispositivos** por debajo del dispositivo actualizado se actualizan también automáticamente.

Los ajustes de configuración del dispositivo no se modifican si el tipo de dispositivo no se ha cambiado.

Si el procedimiento de actualización causa cualquier discrepancia en la configuración existente, esto se detecta en la siguiente ejecución de **Compilación** de la aplicación. Las discrepancias detectadas se indican mediante los mensajes correspondientes. Esto se refiere también a las bibliotecas añadidas de forma implícita, que no se eliminan de forma automática ni de manera apropiada al actualizar un dispositivo.

Sección 4.4
Conversión de dispositivos

Conversión de dispositivos

Introducción

SoMachine 4.0 y versiones posteriores permiten convertir un dispositivo que se ha configurado en su proyecto en un dispositivo distinto pero compatible. SoMachine convierte de forma automática el dispositivo configurado actualmente en el dispositivo seleccionado y muestra los cambios realizados en la vista **Mensajes**.

El comando **Convertir dispositivo** puede añadir o eliminar módulos automáticamente. Estos cambios de hardware influyen, igualmente, en el direccionamiento y las bibliotecas.

Para evitar un comportamiento imprevisto tras la conversión de un dispositivo:
- Asegúrese de que el nuevo dispositivo admita todas las funciones y los puertos de comunicación necesarios en el proyecto.
- Evite el uso de direcciones directas en la aplicación.
- Haga una copia de seguridad del proyecto en el disco duro del PC antes de convertir un dispositivo.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

- Confirme que las direcciones directas utilizadas en la aplicación (por ejemplo, `%IB5`) se hayan convertido correctamente tras la conversión de dispositivo.
- Verifique que el proyecto modificado contenga las configuraciones previstas y proporcione la funcionalidad prevista después de convertir el dispositivo.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

⚠ ATENCIÓN

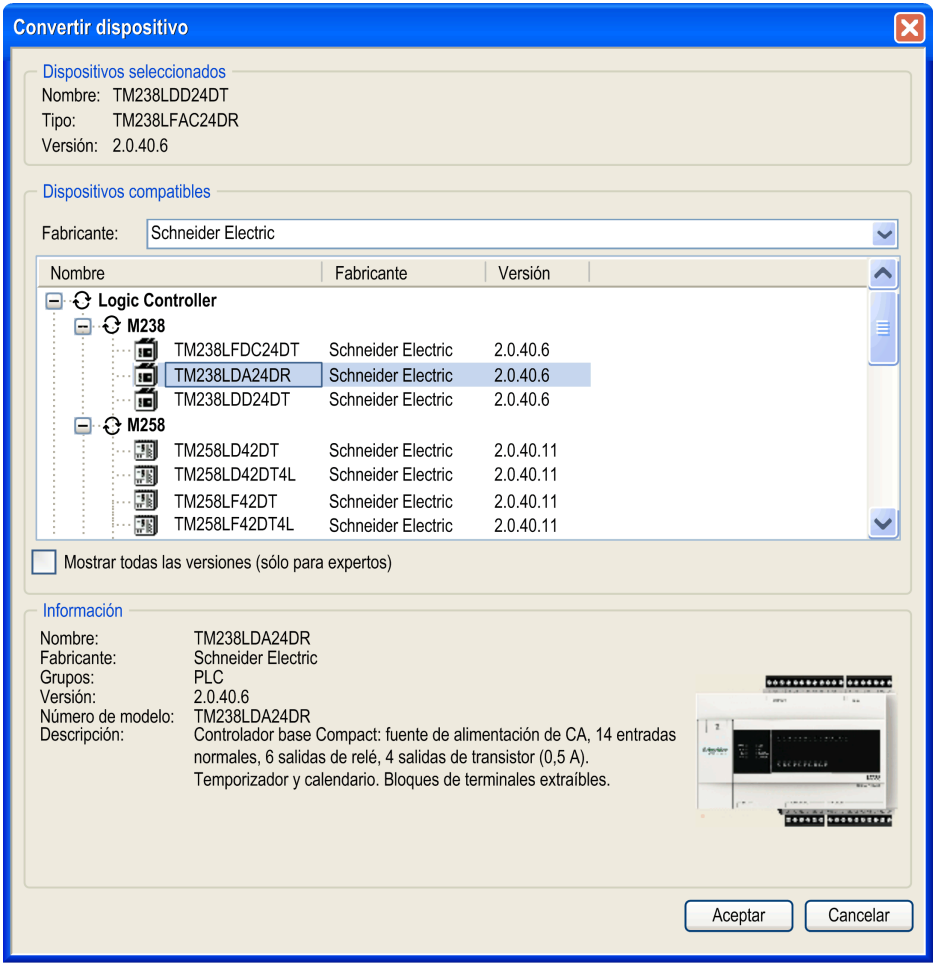
PÉRDIDA DE DATOS

Haga una copia de seguridad del proyecto en el disco duro del PC antes de convertir un dispositivo.

El incumplimiento de estas instrucciones puede causar lesiones o daño al equipo.

Conversión de un dispositivo

Para convertir un dispositivo en un dispositivo compatible, siga estos pasos:

Paso	Acción
1	Haga una copia de seguridad del proyecto en el disco duro del PC ejecutando el comando **Archivo** → **Guardar proyecto como...** antes de convertir un dispositivo.
2	Haga clic con el botón derecho en un dispositivo que desee convertir del árbol **Dispositivos**.
3	Seleccione el comando **Convertir dispositivo** en el menú contextual. **Resultado**: Se muestra el cuadro de diálogo **Convertir dispositivo**. Enumera aquellos dispositivos que son compatibles con el dispositivo que ha seleccionado y ofrece más información sobre el dispositivo seleccionado: 

Paso	Acción
4	Seleccione el dispositivo de la lista al que desee convertir el dispositivo actualmente configurado. Para ver las versiones disponibles de un dispositivo, seleccione la opción **Mostrar todas las versiones (sólo para expertos)**.
5	Si todavía no ha realizado una copia de seguridad del proyecto, haga clic en **Cancelar** para detener el proceso sin los cambios y realice una copia de seguridad antes de reiniciar el proceso. Para iniciar la conversión, haga clic en **Aceptar**. **Resultado**: El dispositivo configurado se convierte en el dispositivo seleccionado en la lista. Se mantiene la información que ha introducido si los módulos relacionados siguen disponibles. Las modificaciones o configuraciones que no se han podido convertir se indican en la vista **Mensajes**.
6	Compruebe si el proyecto convertido aún contiene las configuraciones deseadas y proporciona las funciones previstas. Si no es el caso, adapte la configuración o restaure la copia de seguridad del archivo de proyecto sin modificar.

Información de conversión en la vista Mensajes

En la vista **Mensajes** se muestra la siguiente información del proceso de conversión:

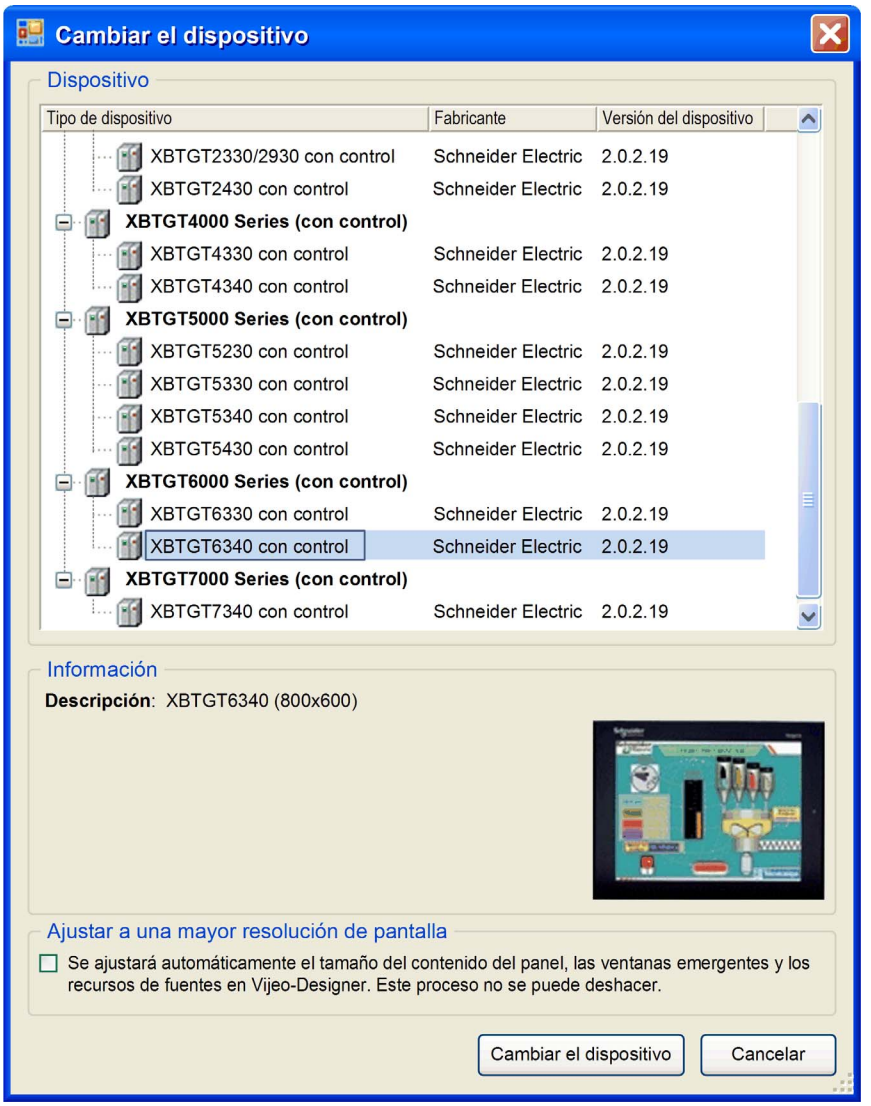
- Dispositivos de origen y dispositivos de destino en los que se han convertido.
- Parámetros que no se han transferido al destino.
- Dispositivos que no se han convertido.

Para guardar la información mostrada en la vista **Mensajes**, puede copiarla en el portapapeles (pulse CTRL + C) y pegarla en un archivo de datos pulse CTRL + V).

Caso especial: Conversión de un dispositivo HMI en otro dispositivo HMI con una resolución de pantalla más alta

Al igual que los demás dispositivos, también puede convertir un dispositivo HMI en otro dispositivo HMI. En este caso, el cuadro de diálogo **Convertir dispositivo** incorpora una opción adicional para dispositivos HMI que permite la adaptación automática a una resolución de pantalla más alta.

Cuadro de diálogo **Convertir dispositivo** para dispositivos HMI



Si el dispositivo HMI nuevo dispone de una pantalla más grande y, por consiguiente, mayor resolución, la opción **Ajustar a una mayor resolución de pantalla** estará habilitada de forma predeterminada. Adapta automáticamente los contenidos de los paneles HMI y las ventanas emergentes, así como las fuentes de los paneles HMI, al aumento de la resolución de pantalla del nuevo dispositivo HMI.

NOTA: Este proceso no se puede deshacer automáticamente. Compruebe y, si es necesario, adapte manualmente los contenidos de los paneles después de la conversión.

Sección 4.5
Conversión de proyectos

Conversión de proyectos de SoMachine Basic y Twido

Introducción

Con SoMachine, puede convertir un proyecto de SoMachine Basic o TwidoSoft/TwidoSuite y el controlador configurado en lógica de SoMachine seleccionable o un controlador HMI *(véase página 981)*. El controlador y la lógica correspondiente se convierten y se integran en el proyecto de SoMachine abierto actualmente en Logic Builder. Para llevar esto a cabo, ejecute el comando **Archivo →Convertir proyecto de SoMachine Basic** o **Archivo →Convertir proyecto de Twido**. Si los comandos no están disponibles, puede insertarlos en un menú de su elección utilizando el cuadro de diálogo **Herramientas →Personalizar** *(véase SoMachine, Comandos de menú, Ayuda en línea)*.

NOTA: Verifique que el proyecto de SoMachine Basic o TwidoSoft/TwidoSuite sea válido antes de convertirlo a SoMachine.

Para ayudar a evitar un comportamiento no deseado después de convertir un proyecto, verifique que el controlador de destino admita todas las funciones y los puertos de comunicación que requiera su proyecto.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

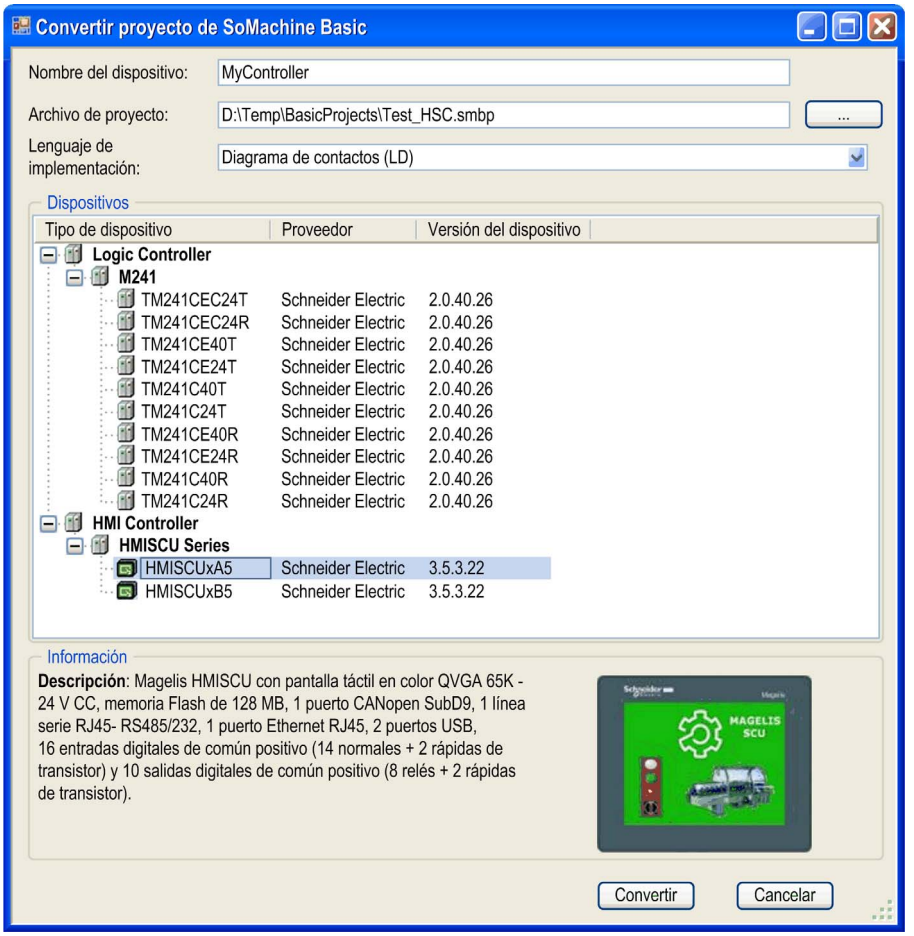
- Verifique que el programa del controlador de destino contenga las configuraciones esperadas y que proporcione las funciones previstas después de convertir el proyecto.
- Depure, verifique y valide por completo la funcionalidad del programa convertido antes de ponerlo en servicio.
- Antes de convertir un programa, verifique que el programa de origen sea válido, es decir, que pueda descargarse en el controlador de origen.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

NOTA: Para obtener más información, consejos e información importante de seguridad sobre la importación de proyectos en SoMachine, consulte *SoMachine - Compatibilidad y migración - Guía del usuario (véase SoMachine - Compatibilidad y migración, Guía del usuario)*.

Conversión de un proyecto de SoMachine Basic o Twido

Para convertir en un proyecto de SoMachine Basic o Twido, proceda del modo siguiente:

Paso	Acción
1	Ejecute el comando **Archivo →Convertir proyecto de SoMachine Basic** o **Archivo →Convertir proyecto de Twido**. **Resultado**: Se muestra el cuadro de diálogo **Convertir proyecto de SoMachine Basic** o **Convertir proyecto de Twido**: 
2	Introduzca un nombre para el controlador en el cuadro **Nombre de dispositivo**.
3	Especifique la ruta de acceso al archivo del proyecto de SoMachine Basic o Twido en el cuadro **Archivo de proyecto**, o haga clic en el botón **...** para buscar el archivo.

Paso	Acción
4	Seleccione el lenguaje de programación al que se convertirá la lógica en la lista **Lenguaje de implementación**. Se admiten los siguientes lenguajes de programación: ● Diagrama de contactos (LD) ● Diagrama de bloques de funciones (FBD) ● Lista de instrucciones (IL) ● Diagrama de función continua (CFC)
5	En la lista **Dispositivos** seleccione el controlador de destino al que desea convertir su controlador de SoMachine Basic o Twido. Se mostrará información adicional sobre el dispositivo seleccionado en el área **Información** del cuadro de diálogo.
6	Haga clic en **Convertir** para iniciar la conversión. **Resultado**: El proyecto de SoMachine Basic o Twido se convierte y se integra en el proyecto de SoMachine abierto. Las modificaciones o configuraciones que no se han podido convertir se indican en la vista *(véase SoMachine, Comandos de menú, Ayuda en línea)* **Mensajes**.
7	Consulte la categoría **Conversión de proyecto** de la vista **Mensajes** y verifique los errores y alertas detectados y enumerados.
8	Compruebe si el proyecto convertido aún contiene las configuraciones deseadas y proporciona las funciones previstas. Si no es así, adapte la configuración.

Biblioteca de TwidoEmulationSupport

La biblioteca *(véase Biblioteca Twido Emulation Support, Guía de la biblioteca)* de TwidoEmulationSupport contiene funciones y bloques de funciones que proporcionan funcionalidad de SoMachine Basic y TwidoSoft/TwidoSuite en una aplicación de SoMachine. La biblioteca de TwidoEmulationSupport se integra automáticamente en el proyecto de SoMachine con el controlador convertido.

Conversión del programa de aplicación

El programa de aplicación de origen se convierte en un programa de controlador único denominado **Principal** en el proyecto de SoMachine. Por cada POU de SoMachine Basic libre o subrutina de Twido, se crea un programa aparte en SoMachine (subprograma). El lenguaje de programación que se utiliza para el programa **Principal** y los subprogramas se determina de acuerdo con el **lenguaje de implementación** seleccionado en el cuadro de diálogo **Convertir proyecto de SoMachine Basic**/**Convertir proyecto de Twido**.

Por cada objeto de lenguaje (como objetos de memoria o bloques de funciones) que utilice el programa de aplicación, se creará una variable global. Se crearán listas de variables globales *(véase página 219)* distintas para cada categoría de objeto (una para bits de memoria, otra para palabras de memoria, etc.).

Se aplican las restricciones siguientes para la conversión de programas de aplicación en relación con la estructura del programa:

- En SoMachine, no es posible saltar a una etiqueta *(véase página 319)* de otro programa, como
  - del programa **Principal** a un subprograma, o
  - de un subprograma a otro subprograma o al programa **Principal**.

- No es posible definir pasos Grafcet (GRAphe Fonctionnel de Commande Etapes/Transitions) en un subprograma.
- No es posible activar o desactivar pasos Grafcet (según la instrucción `#` y `D#`) en un subprograma.

Conversión de objetos de memoria

Las áreas proporcionadas para objetos de memoria en SoMachine Basic y Twido difieren respecto a SoMachine.

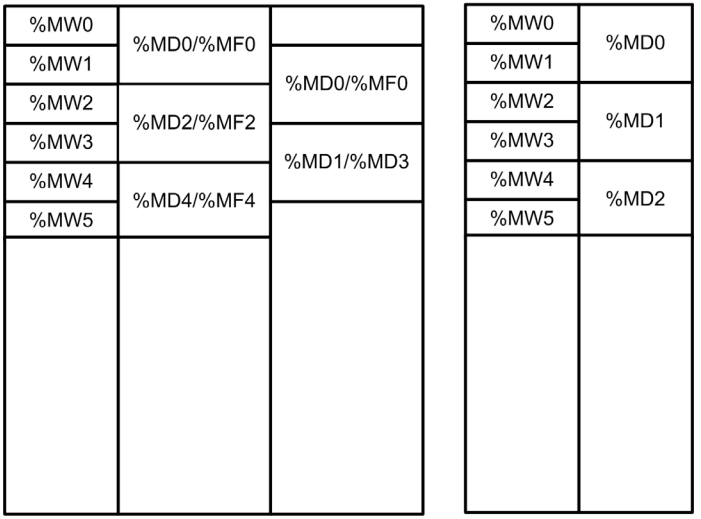
En SoMachine Basic y Twido, hay 3 áreas diferenciadas de objetos de memoria:

Área	Objetos de memoria incluidos
área de bits de memoria	bits de memoria (`%M`)
área de palabras de memoria	• palabras de memoria (`%MW`) • palabras dobles (`%MD`) • valores de coma flotante (`%MF`)
área de constantes	• palabras constantes (`%KW`) • palabras dobles (`%KD`) • valores de coma flotante (`%KF`)

En SoMachine, sólo existe el área de palabras de memoria para los objetos de memoria:

Área	Objetos de memoria incluidos
área de palabras de memoria	• palabras de memoria (`%MW`) • palabras dobles (`%MD`) • valores de coma flotante No existe un formato de direccionamiento específico para valores de coma flotante. Las variables de coma flotante pueden asignarse a una dirección `%MD`.

El gráfico proporciona una descripción general de las diferentes disposiciones de las direcciones %MD y %MF en SoMachine Basic/Twido y SoMachine.



1 Direcciones de memoria en SoMachine Basic/Twido
2 Direcciones de memoria en SoMachine

Los objetos de memoria se convierten del modo siguiente:

Objetos de memoria de origen	Objetos de memoria de destino	Información adicional
%MW	Se asignan a la misma dirección %MW. **Ejemplo** %MW2 se asigna a %MW2.	Por cada objeto %MW, se crea una variable global de tipo INT.
%MD y %MF con direcciones pares	Se asignan de modo que se ubiquen en la misma dirección %MW que antes. **Ejemplo** %MD4/%MF4 se asignan a %MD2.	Por cada objeto %MD, se crea una variable global de tipo DINT. Por cada objeto %MF, se crea una variable global de tipo REAL.
%MD y %MF con direcciones impares	No pueden asignarse, porque una variable INT no puede ubicarse en una dirección de memoria impar.	Se crea una variable para ayudar a garantizar que puede compilarse la aplicación convertida. Sin embargo, debe examinar el efecto que la creación de una variable como esta tiene en la funcionalidad general del programa.
%M	Se asigna como campo de bits empaquetado en una ubicación fija del área %MW.	Por cada objeto %M, se crea una variable global de tipo BOOL.
%KW	Se asigna a direcciones consecutivas del área %MW.	Por cada objeto %KW, se crea una variable global de tipo INT.

La relación entre objetos %KW, %KD y %KF es la misma que para objetos %MW, %MD y %MF. Por ejemplo, %KD4/%KF4 se asignan a la misma ubicación que %KW4. Las direcciones %KD/%KF impares no pueden asignarse.

Acceso remoto

Un dispositivo remoto puede acceder a los objetos de memoria (%MW, %MD, %MF y %M) mediante servicios Modbus:

- Si un dispositivo remoto accede a objetos %MW, %MD o %MF en la aplicación de origen, este acceso seguirá estando disponible en la aplicación SoMachine.
- Si un dispositivo remoto accede a objetos %M en la aplicación de origen, este acceso ya no estará disponible en la aplicación SoMachine.

Conversión de bloques de funciones

Para bloques de funciones en SoMachine Basic/Twido, la biblioteca de TwidoEmulationSupport proporciona bloques de funciones con funciones compatibles:

Bloque de funciones de SoMachine Basic/Twido	Bloque de funciones de la biblioteca de TwidoEmulationSupport
temporizadores %TM	FB_Timer
contadores %C	FB_Counter
registro %R	FB_FiFo/FB_LiFo
tambor %DR	FB_Drum
registro de bits de desplazamiento %SBR	FB_ShiftBitRegister
contador de pasos %SC	FB_StepCounter
programa %SCH	FB_ScheduleBlock
PID	FB_PID
intercambio/mensaje %MSG	FB_EXCH
contador de alta velocidad %HSC/%VFC	–
contador rápido %FC	–
generador de pulsos PLS %PLS	–
generador de pulsos PWM %PWM	–

Para la conversión de bloques de funciones, tenga en cuenta lo siguiente:

- La biblioteca de TwidoEmulationSupport no proporciona bloques de funciones para funciones relacionadas con el hardware, como contadores de alta velocidad, contadores rápidos y generadores de pulsos. Deben controlarse mediante bloques de funciones proporcionados por las bibliotecas HSC y PTO_PWM específicas de la plataforma. Estos bloques de funciones no son compatibles con los bloques de funciones de origen. En resumen, una conversión completa no es posible si el programa de origen contiene funciones basadas en recursos de hardware del controlador. Para obtener más información, consulte la descripción *Conversión de contadores rápidos, contadores de alta velocidad (Twido: contadores muy rápidos) y generadores de pulsos (véase página 96)*.
- En SoMachine Basic/Twido, la función de mensajería se proporciona mediante la instrucción `EXCHx` y el bloque de funciones `%MSGx`. En la aplicación SoMachine, esta función se realiza mediante un bloque de funciones `FB_EXCH` único.
- En SoMachine Basic/Twido, ciertos bloques de funciones pueden configurarse utilizando cuadros de diálogo de configuración especiales. Estos datos de configuración se proporcionan a los bloques de funciones de la biblioteca de TwidoEmulationSupport mediante parámetros dedicados.

Conversión de variables del sistema

Se convierten los bits y palabras del sistema siguientes:

Bit/palabra del sistema	Información adicional
`%S0`	Se establece en 1 en el primer ciclo tras un arranque en frío. **NOTA:** No es posible desencadenar un arranque en frío escribiendo en este bit del sistema.
`%S1`	Se establece en 1 en el primer ciclo tras un arranque en caliente. **NOTA:** No es posible desencadenar un arranque en caliente escribiendo en este bit del sistema.
`%S4`	Pulso con una referencia de tiempo de 10 ms.
`%S5`	Pulso con una referencia de tiempo de 100 ms.
`%S6`	Pulso con una referencia de tiempo de 1 s.
`%S7`	Pulso con una referencia de tiempo de 1 min.
`%S13`	Se establece en 1 en el primer ciclo después de haber iniciado el controlador.
`%S18`	Se establece en 1 si se produce un desborde aritmético. **NOTA:** La biblioteca de TwidoEmulationSupport proporciona este indicador y sólo se establece mediante las funciones proporcionadas por esta biblioteca.
`%SW63...65`	Código de error de los bloques `MSG` 1...3.
`%SW114`	Habilitar indicadores para los fechadores.

La conversión no admite otras variables del sistema. Si el programa de aplicación de origen utiliza una variable del sistema no admitida, se genera un mensaje en la categoría **Conversión de proyecto** de la vista *(véase SoMachine, Comandos de menú, Ayuda en línea)* **Mensajes**.

Conversión de tablas de animación

La gestión de tablas de animación difiere en las aplicaciones de origen y de destino:

- SoMachine Basic/Twido le permiten definir varias listas de animación identificadas por el nombre. Cada lista de animación puede contener varias entradas para objetos que deban animarse.
- En SoMachine hay 4 listas de supervisión *(véase página 464)* predefinidas (**Supervisar 1**...**Supervisar 4**). Cada lista de supervisión puede contener varias variables que deben animarse. Una lista de supervisión puede contener variables de diferentes controladores.

Durante el proceso de conversión, las entradas de las tablas de animación de origen se añaden al final de la lista de supervisión **Supervisar 1**.

Conversión de símbolos

Los símbolos definidos en un proyecto de SoMachine Basic/Twido se transfieren automáticamente al proyecto de SoMachine.

Se aplican las siguientes restricciones a la denominación de símbolos:

Si...	Entonces...
un nombre de símbolo no cumple las reglas de denominación de SoMachine,	se modifica el nombre del símbolo.
un nombre de símbolo coincide con una palabra clave de SoMachine,	se modifica el nombre del símbolo.
no se crea ninguna variable para un objeto de lenguaje,	se descarta el nombre del símbolo.
un símbolo no se utiliza en ningún lugar del programa de aplicación,	puede descartarse el nombre del símbolo.

Para ver la lista completa de modificaciones de símbolos que han sido necesarias, consulte la vista **Mensajes**.

Conversión de contadores rápidos, contadores de alta velocidad (Twido: contadores muy rápidos) y generadores de pulsos

Los bloques de funciones que proporciona SoMachine difieren respecto a los bloques de funciones que proporciona SoMachine Basic/Twido. Sin embargo, la configuración de los contadores rápidos, contadores de alta velocidad y generadores de pulsos se convierte en la medida de lo posible. Por cada bloque de funciones `%FC`, `%HSC`/`%VFC`, `%PLS` y `%PWM` que se utiliza en la aplicación SoMachine Basic/Twido, se crea un programa en SoMachine. Puede mejorar esta implementación básica de acuerdo con las necesidades de su aplicación.

Se aplican las siguientes restricciones:

Restricción	Solución
Las entradas y salidas que utilizan los contadores de alta velocidad y generadores de pulsos convertidos pueden diferir respecto a las entradas y salidas utilizadas de la aplicación de origen.	Tenga esto en cuenta durante el cableado del controlador convertido. La reasignación de entradas y salidas se notifica en la vista **Mensajes** *(véase SoMachine, Comandos de menú, Ayuda en línea)*.
Algunas combinaciones de contadores `%HSC`/`%VFC` y `%FC` no pueden convertirse. Por ejemplo, en SoMachine Basic es posible tener 2 bloques de funciones `%HSC` y 4 bloques de funciones `%FC` en paralelo. Esto no es posible con la función de conversión.	Tiene que adaptar su aplicación manualmente.
El acceso a los parámetros de los bloques de funciones se realiza de forma diferente en SoMachine Basic y SoMachine. En SoMachine Basic, el programa de aplicación puede acceder directamente a los parámetros de un bloque de funciones, por ejemplo, `%VFC.P = 100`. En SoMachine, debe utilizarse un bloque de funciones específico del controlador (por ejemplo, `EXPERTSetParam`) para acceder a un parámetro.	Si la aplicación de origen accede a parámetros del bloque de funciones, debe extender la aplicación convertida consecuentemente.

Conversión de un programa Grafcet

Los lenguajes de programación de SoMachine no admiten la programación con Grafcet.

Por este motivo, una aplicación Grafcet convertida contiene elementos de lenguaje adicionales que implementan la gestión de Grafcet.

Elemento adicional	Descripción
carpeta **Grafcet**	Esta carpeta contiene los elementos de lenguaje siguientes utilizados para la gestión de la máquina de estado Grafcet.
estructura de datos `GRAFCET_STATES`	Esta estructura de datos tiene un elemento de bit por cada estado Grafcet permitido. Si se trata de un estado inicial, el estado se inicializará como TRUE; en caso contrario, será FALSE.
Lista de variables globales **GrafcetVariables**	Esta lista de variables globales contiene las variables siguientes: ● 1 variable `STATES` que contiene 1 bit por cada estado de Grafcet. Cada bit representa el valor actual del estado Grafcet correspondiente (objeto `%Xi`). ● 1 variable `ACTIVATE_STATES` que contiene 1 bit por cada estado de Grafcet. Si el bit es TRUE, el estado Grafcet se activa en el siguiente ciclo. ● 1 variable `DEACTIVATE_STATES` que contiene 1 bit por cada estado de Grafcet. Si el bit es TRUE, el estado Grafcet se desactiva en el siguiente ciclo.
programa **Grafcet**	El programa implementa la máquina de estado Grafcet. Contiene la lógica para la activación y desactivación de los pasos de Grafcet. El programa contiene las siguientes acciones: ● `Init` restablece los pasos de Grafcet a sus estados iniciales. Se ejecuta cuando el programa de aplicación establece el bit de sistema `%S21`. ● `Reset` restablece los pasos de Grafcet a FALSE. Se ejecuta cuando el programa de aplicación establece el bit de sistema `%S22`.

Las instrucciones de Grafcet del programa de aplicación se convierten del modo siguiente:

- El principio de cada paso de Grafcet se marca mediante una etiqueta con el nombre del paso. La primera instrucción del paso de Grafcet comprueba si el paso está activo. Si no es así, salta a la etiqueta del siguiente paso de Grafcet.
- El acceso a `%Xi` se convierte en un acceso a la variable `STATES.Xi`.
- Una instrucción de activación `#i` de Grafcet se convierte en la definición del bit de activación del estado `i` y el bit de desactivación del estado actual.
- Una instrucción de desactivación `#Di` de Grafcet se convierte en la definición del bit de desactivación del estado `i` y el bit de desactivación del estado actual.

Puede extender el programa Grafcet convertido si tiene en cuenta la información proporcionada en esta sección.

Funciones de comunicación de Twido

Las siguientes funciones de comunicación de Twido no se convierten:

- AS Interface
- CANopen
- conexión remota

Si utiliza estas funciones de comunicación en su aplicación Twido, debe adaptar la aplicación SoMachine manualmente.

Durante la conversión, se crea 1 variable para cada objeto de E/S relacionado con el fin de permitir que la aplicación SoMachine se compile correctamente. Estas variables se recopilan en listas de variables globales separadas. Esto le ayuda a identificar las variables que deben sustituirse.

Errores y alertas detectados indicados en la vista Mensajes

Si se detectan errores o alertas durante el proceso de conversión, se muestra un cuadro de mensaje que indica el número de errores y alertas detectados. Para obtener más información, consulte la categoría **Conversión de proyecto** de la vista *(véase SoMachine, Comandos de menú, Ayuda en línea)* **Mensajes**. Verifique detenidamente cada entrada para comprobar si debe adaptar su aplicación.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

- Verifique que el programa del controlador de destino contenga las configuraciones esperadas y que proporcione las funciones previstas después de convertir el proyecto.
- Depure, verifique y valide por completo la funcionalidad del programa convertido antes de ponerlo en servicio.
- Antes de convertir un programa, verifique que el programa de origen sea válido, es decir, que pueda descargarse en el controlador de origen.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

- Un mensaje de **advertencia** indica que el proceso de conversión ha realizado ciertos ajustes que, con toda probabilidad, no tendrán ningún impacto en las funciones de su aplicación.
- Un mensaje de **error** indica que ciertas partes de la aplicación no se han podido convertir por completo. En este caso, debe adaptar la aplicación manualmente para mantener la misma funcionalidad en la aplicación de destino.
- Si el programa de aplicación utiliza una funcionalidad que no se puede convertir por completo, el convertidor crea variables para los objetos de lenguaje no admitidos. Esto le permite compilar correctamente la aplicación. Sin embargo, debe verificar esta funcionalidad no admitida tras la conversión.

Para guardar la información mostrada en la vista **Mensajes**, puede copiarla en el portapapeles (pulse CTRL + C) y pegarla en un archivo de datos (pulse CTRL + V).

Capítulo 5
Cuadros de diálogo del editor de dispositivos común

Contenido de este capítulo

Este capítulo contiene las siguientes secciones:

Sección 5.1
Configuración del dispositivo

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Información general sobre editores de dispositivos

Descripción general

El editor de dispositivos proporciona parámetros para la configuración de un dispositivo, que se gestiona en el árbol **Dispositivos**.

Para abrir el editor de dispositivos para un dispositivo específico, haga lo siguiente:

- Haga doble clic en el nodo del dispositivo en el árbol **Dispositivos** o
- Seleccione el dispositivo en el árbol **Dispositivos** y ejecute el comando **Modificar objeto** a través del menú contextual o del menú **Proyecto**.

El cuadro de diálogo **Herramientas →Opciones →Editor de dispositivo** le permite crear las vistas genéricas de configuración de dispositivos invisibles *(véase página 125)*.

En este capítulo se describen los principales cuadros de diálogo del editor de dispositivos. Los cuadros de diálogo de configuración específicos de bus se describen por separado.

Cuadros de diálogo del editor de dispositivos principal

El título del cuadro de diálogo principal contiene el nombre del dispositivo, por ejemplo, **MYPlc**.

En función del tipo de dispositivo, el editor de dispositivos puede ofrecer las siguientes fichas:

Ficha	Descripción
Selección de controlador *(véase página 103)*	Configuración de la conexión entre el sistema de programación y un dispositivo programable (controlador). Esta es la ficha predeterminada para SoMachine V4.0 y versiones posteriores. SoMachine V3.1 y versiones anteriores utilizan la ficha **Configuración de comunicación** de forma predeterminada.
Configuración de comunicación *(véase página 120)*	Configuración de la conexión entre el sistema de programación y un dispositivo programable (controlador). Esta es la ficha predeterminada para SoMachine V3.1 y versiones anteriores. SoMachine V4.0 y versiones posteriores utilizan la ficha **Selección de controlador** de forma predeterminada.
Configuración *(véase página 125)*	Visualización o configuración de los parámetros del dispositivo.
Aplicaciones *(véase página 127)*	Lista de aplicaciones que están en ejecución en el controlador. Consulte la descripción en el capítulo *(véase página 165)* *Programa*.
Archivos *(véase página 130)*	Configuración de una transferencia de archivos entre el host y el controlador.
Registro *(véase página 132)*	Visualización del archivo de registro del controlador.
Ajustes PLC *(véase página 134)*	Configuración de: ● aplicación asignada para el manejo de E/S ● comportamiento de E/S en estado de detención ● opciones de ciclo de bus
Usuarios y grupos *(véase página 972)*	Administración de usuarios en relación con el acceso al dispositivo en tiempo de ejecución.

Ficha	Descripción
Derechos de acceso *(véase página 976)*	Configuración de los derechos de acceso a los archivos y objetos de ejecución para cada grupo de usuarios.
Distribución de tareas *(véase página 149)*	Visualización de las entradas y salidas asignadas a la tarea definida; se utiliza para la resolución de problemas.
Estado *(véase página 151)*	Estado específico del dispositivo y mensajes de diagnóstico.
Información *(véase página 152)*	Información general sobre el dispositivo (por ejemplo: nombre, fabricante, versión).
Asignación E/S *(véase página 153)*	Asignación de los canales de entrada y salida de un dispositivo de E/S en variables de proyecto (aplicación).

Selección de controlador

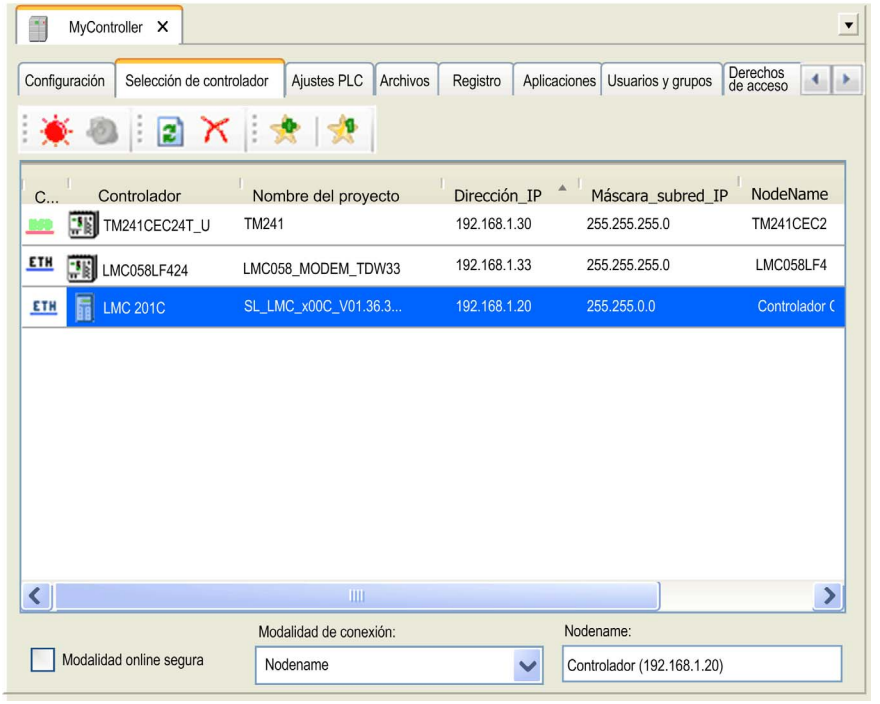
Descripción general

La vista **Selección de controlador** del editor de dispositivos proporciona acceso al servicio Network Device Identification. Esta vista permite buscar en la red Ethernet dispositivos disponibles (como controladores o dispositivos HMI) y visualizarlos en una lista. En esta vista, puede configurar los parámetros para la comunicación entre los dispositivos (denominados *controladores* en este capítulo) y el sistema de programación.

La lista de controladores contiene aquellos controladores de la red que han enviado una respuesta a la petición de SoMachine. Puede que el controlador elegido no esté incluido en esta lista. Esto puede deberse a varios motivos. Para ver las posibles causas y las soluciones correspondientes, consulte el capítulo *Acceso a controladores - Resolución de problemas y preguntas frecuentes* *(véase página 881)*.

La vista **Selección de controlador** sólo se muestra si la comunicación entre el controlador y el sistema de programación se establece mediante la dirección IP. Este es el ajuste predeterminado para SoMachine V4.0 y versiones posteriores. Puede seleccionar entre un establecimiento de comunicación mediante una dirección IP o mediante una ruta activa en el cuadro de diálogo **Configuración del proyecto →Configuración de comunicación**. Si se selecciona la opción **Marque mediante "dirección IP"**, se muestra la vista **Selección de controlador** en el editor de dispositivos. De lo contrario, se muestra la vista **Configuración de comunicación**.

Vista **Selección de controlador** del editor de dispositivos



La vista **Selección de controlador** contiene los siguientes elementos:

- botones de la barra de herramientas
- lista con información sobre los controladores disponibles
- opción, lista y cuadro de texto en la parte inferior de la vista

Descripción de los botones de la barra de herramientas

Los botones siguientes están disponibles en la barra de herramientas:

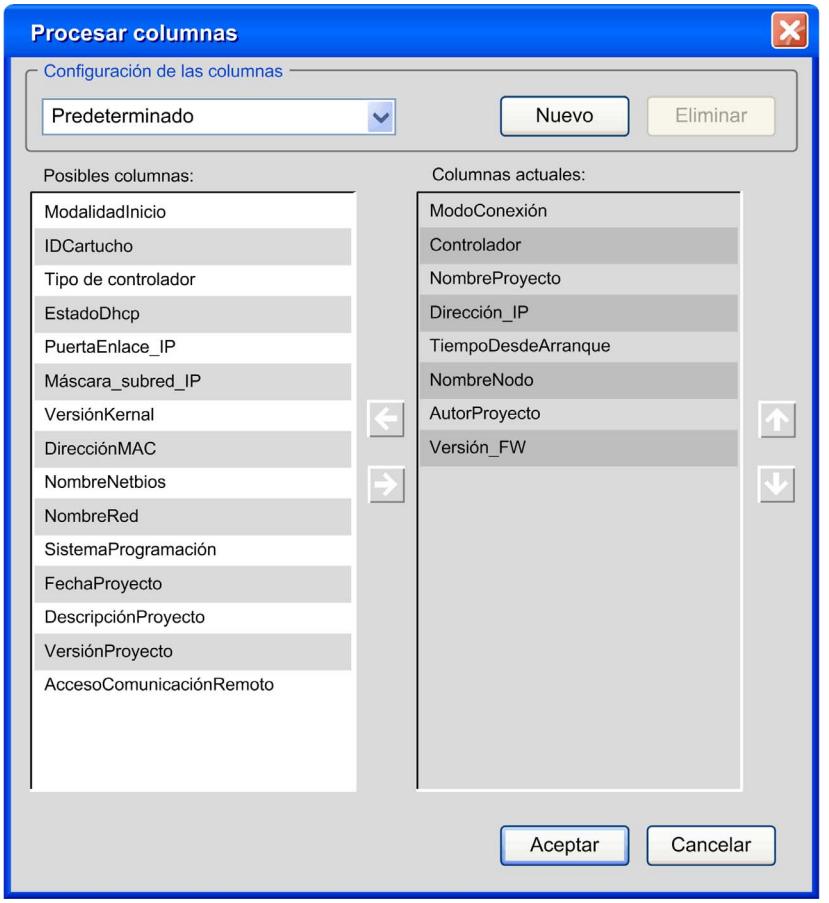
Botón	Descripción
Óptica	Haga clic en este botón para que el controlador seleccionado indique una señal óptica: parpadea rápidamente un LED de control. Esto puede resultar útil para identificar el controlador respectivo si se usan muchos controladores. La función se detiene al hacer clic por segunda vez o de forma automática tras unos 30 segundos. **NOTA:** La señal óptica solo la emiten los controladores que admiten esta función.
Óptica y acústica	Haga clic en este botón para que el controlador seleccionado indique una señal óptica y acústica: empieza haciendo un pitido y parpadea rápidamente un LED de control. Esto puede resultar útil para identificar el controlador respectivo si se usan muchos controladores. La función se detiene al hacer clic por segunda vez o de forma automática tras unos 30 segundos. **NOTA:** Las señales óptica y acústica solo las emiten los controladores que admiten esta función.
Actualización	Haga clic en este botón para actualizar la lista de controladores. Se envía una petición a los controladores de la red. Los controladores que responden a la petición se incluyen en una lista con los valores actuales. Las entradas de controladores existentes se actualizan con cada nueva petición. Los controladores ya incluidos en la lista pero que no responden a una nueva petición no se suprimen. Quedan marcados como inactivos mediante una cruz roja que se añade al icono del controlador. El botón **Actualizar** corresponde al comando **Actualizar lista** que se proporciona en el menú contextual si se hace clic con el botón derecho del ratón en un controlador de la lista. Para actualizar la información de un controlador seleccionado, el menú contextual proporciona el comando **Actualizar este controlador**. Este comando solicita información más detallada del controlador seleccionado. **NOTA:** El comando **Actualizar este controlador** también puede actualizar la información de otros controladores.

Botón	Descripción
Eliminar controladores inactivos de la lista.	Los controladores que no responden a una exploración de red se marcan como inactivos en la lista. Se indica mediante una cruz roja que se añade al icono del controlador. Haga clic en este botón para eliminar de forma simultánea todos los controladores marcados como controladores inactivos de la lista. **NOTA:** Puede que un controlador se marque como inactivo aunque en realidad no lo esté. El menú contextual que se abre al hacer clic con el botón derecho del ratón sobre un controlador de la lista ofrece 2 comandos adicionales para eliminar controladores: ● El comando **Eliminar el controlador seleccionado de la lista** permite eliminar solamente los controladores seleccionados de la lista. ● El comando **Remove all controllers from list** permite eliminar de forma simultánea todos los controladores de la lista.
Nuevo favorito... y **Favorite 0**	Puede utilizar la opción **Favoritos** para ajustar la selección de controladores a sus requisitos personales. Esto puede ayudarle a realizar el seguimiento de muchos controladores de la red. Un **Favorito** describe una colección de controladores reconocidos por un identificador único. Haga clic en un botón de favorito (como **Favorite 0**) para seleccionarlo o cancelar su selección. Si no ha seleccionado ningún favorito, estarán visibles todos los controladores detectados. También es posible acceder a la opción **Favoritos** a través del menú contextual. Para abrirla, basta con hacer clic con el botón derecho del ratón sobre un controlador de la lista. Si se mueve el cursor por encima del botón Favorito de la barra de herramientas, se verán los controladores asociados en el formato de información sobre herramientas.

Lista de controladores

La lista de controladores de la parte central de la vista **Selección de controlador** del editor de dispositivo muestra los controladores que han enviado una respuesta a la exploración de la red. Proporcionar información sobre cada controlador en diversas·columnas. Es posible adaptar las columnas que aparecen en la lista de controladores según sus requisitos personales.

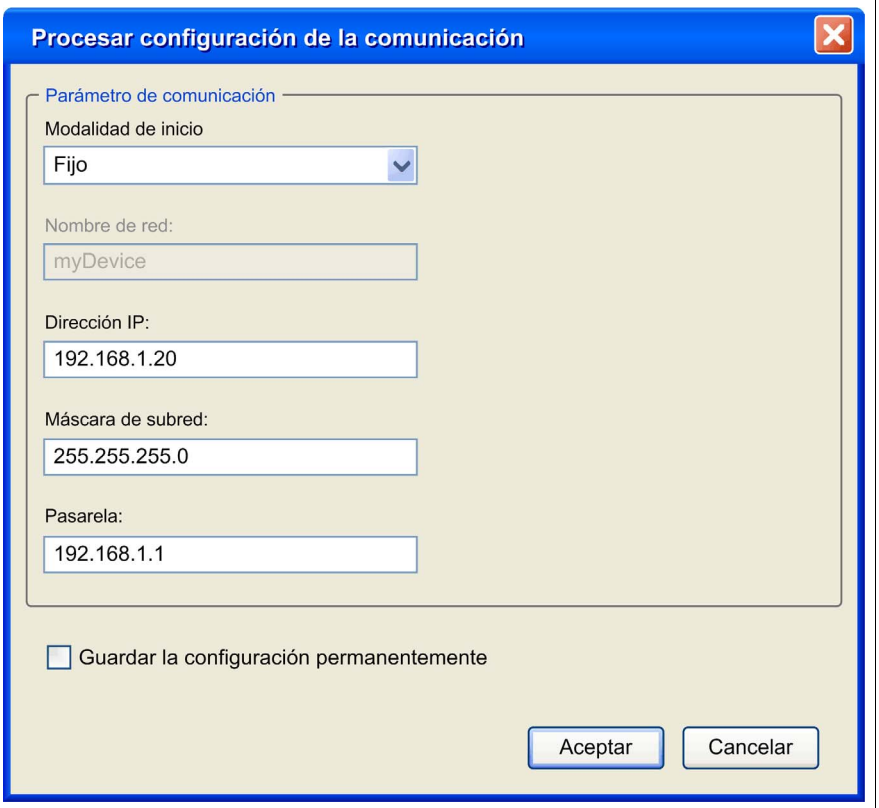
Para ello, haga clic con el botón derecho del ratón en el encabezado de una columna para abrir el cuadro de diálogo **Procesar columnas**.



Puede crear su propio diseño para la tabla. Haga clic en **Nuevo** e introduzca un nombre para el diseño. Para cambiar columnas de la lista de **Posibles columnas** a la lista de **Columnas actuales** y viceversa, haga clic en los botones con flechas horizontales. Para cambiar el orden de las columnas en la lista **Columnas actuales**, haga clic en los botones de flecha arriba y flecha abajo.

Configuración de comunicación

Para establecer los parámetros de comunicación entre el sistema de programación y un controlador, haga lo siguiente:

Paso	Acción
1	Seleccione el controlador de la lista de controladores.
2	Haga clic con el botón derecho en la entrada de controlador y ejecute el comando **Procesar configuración de la comunicación...** en el menú contextual. **Resultado:** Se abre el cuadro de diálogo **Procesar configuración de la comunicación...** con los ajustes actuales del controlador.  **NOTA:** La mayoría de los controladores proporcionan un parámetro (como **RemoteAccess**) que ayuda a evitar que se realicen cambios en los parámetros de comunicación del controlador.

Paso	Acción

3	Configure los parámetros de comunicación:
	● **Modalidad de inicio**
	  ● **FIXED**: Se utiliza una dirección IP fija según los valores especificados más abajo (**Dirección IP**, **Máscara de subred**, **Puerta de enlace**).
	  ● **BOOTP**: La dirección IP se recibe de forma dinámica mediante BOOTP (protocolo Bootstrap). Los valores de más abajo se ignorarán.
	  ● **DHCP**: La dirección IP se recibe de forma dinámica mediante DHCP (protocolo de configuración dinámica de host). Los valores de más abajo se ignorarán.
	NOTA: No todos los dispositivos admiten BOOTP o DHCP.
	● **Dirección IP**
	Al configurar direcciones IP, consulte el mensaje de peligro mostrado más abajo.
	Este cuadro de texto contiene la dirección IP del controlador. Se trata de una dirección exclusiva formada por 4 números comprendidos entre 0 y 255 y separados por puntos. La dirección IP debe ser exclusiva en esta (sub)red.
	● **Máscara de subred**
	La máscara de subred especifica el segmento de red al que pertenece el controlador. Se trata de una dirección formada por 4 números comprendidos entre 0 y 255 y separados por puntos. Generalmente, sólo se utilizan los valores 0 y 255 para los números estándar de máscara de subred. No obstante, pueden utilizarse otros valores numéricos. El valor de la máscara de subred suele ser el mismo para todos los controladores de la red.
	● **Puerta de enlace**
	La dirección de puerta de enlace es la dirección de un enrutador IP local que se encuentra en la misma red que el controlador. El enrutador IP transfiere los datos a destinos fuera de la red local. Se trata de una dirección formada por 4 números comprendidos entre 0 y 255 y separados por puntos. El valor de la puerta de enlace suele ser el mismo para todos los controladores de la red.
	● Para guardar los ajustes de comunicación en el controlador aunque se reinicie dicho controlador, active la opción **Guardar la configuración permanentemente**.

4	Haga clic en **Aceptar** para transferir la configuración al controlador.

Gestione las direcciones IP con cuidado debido a que cada dispositivo de la red necesita una dirección única. Si existen varios dispositivos con la misma dirección IP, puede producirse un funcionamiento impredecible en la red y el equipo asociado.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

- Asegúrese de que todos los dispositivos tengan direcciones exclusivas.
- Solicite su dirección IP al administrador del sistema.
- Confirme que la dirección IP del dispositivo es única antes de poner el sistema en funcionamiento.
- No asigne la misma dirección IP a ningún otro equipo de la red.
- Actualice la dirección IP después de clonar cualquier aplicación que incluya comunicaciones Ethernet a una dirección exclusiva.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

Gestión de Favoritos

Para gestionar los favoritos de la lista de controladores, realice los pasos siguientes:

Paso	Acción
1	Seleccione el controlador de la lista de controladores.

2	Haga clic con el botón derecho del ratón sobre el controlador y seleccione uno de los comandos:
	● **Nuevo favorito** para crear un grupo de favoritos nuevo.
	● **Favorite n** para
	  ● añadir el controlador seleccionado a la lista de favoritos
	  ● eliminar el controlador seleccionado de la lista de favoritos
	  ● eliminar todos los controladores de la lista de favoritos
	  ● seleccionar un favorito
	  ● cambiar el nombre de un favorito
	  ● eliminar un favorito

Opción Modalidad online segura

Con la opción **Modalidad online segura**, SoMachine muestra un mensaje que solicita confirmación cuando se selecciona uno de los siguientes comandos: **Forzar valores, Inicio de sesión, Descarga múltiple, Desactivar la lista de forzado, Ciclo individual, Inicio, Parada, Escribir valores**. Para desactivar la modalidad online segura y, por consiguiente, hacer que no se muestre este mensaje, desmarque esta opción.

Especificación de nombres de dispositivo exclusivos (Nodename)

El término **Nodename** se utiliza como sinónimo del término "nombre de dispositivo". Dado que los nodenames también se utilizan para identificar los controladores tras una exploración de red, debe gestionarlos con precaución como direcciones IP y verificar que cada nodename sea exclusivo en la red. Si hay varios dispositivos con el mismo Nodename, puede producirse un funcionamiento imprevisible de la red y el equipo asociado.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

- Asegúrese de que todos los dispositivos tengan nodenames exclusivos.
- Confirme que el nodename del dispositivo sea exclusivo antes de poner el sistema en funcionamiento.
- No asigne el mismo nodename a ningún otro equipo de la red.
- Actualice el nodename tras clonar cualquier aplicación que incluya comunicaciones Ethernet a un nodename exclusivo.
- Cree un nodename exclusivo para cada dispositivo que no lo cree automáticamente, como los controladores HMI XBT.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

En función del tipo de controlador, el procedimiento de creación automática del **NodeName** (nombre de dispositivo) puede variar. Para crear un nombre exclusivo, algunos controladores integran la dirección IP, y otros utilizan la dirección MAC del adaptador Ethernet. En este caso, no es necesario cambiar el nombre.

Otros dispositivos, como los controladores HMI XBT, no crean automáticamente ningún nombre de dispositivo exclusivo. En tal caso, asigne un nombre de dispositivo exclusivo (**NodeName**) como se indica a continuación:

Paso	Acción
1	Haga clic con el botón derecho en el controlador de la lista y ejecute el comando **Cambiar el nombre del dispositivo...** en el menú contextual. **Resultado**: Se abre el cuadro de diálogo **Cambiar el nombre del dispositivo**.
2	En el cuadro de diálogo **Cambiar el nombre del dispositivo**, especifique un nombre de dispositivo exclusivo en el cuadro de texto **Nuevo**.
3	Haga clic en el botón **Aceptar** para confirmar. **Resultado**: El nombre de dispositivo especificado se asigna al controlador y aparece en la columna **NodeName** de la lista. **NOTA:** Nombre de dispositivo y **Nodename** son sinónimos.

Especificación de la Modalidad de conexión

La lista **Modalidad de conexión** de la parte inferior izquierda de la vista **Selección de controlador** permite seleccionar un formato para la dirección de conexión que debe especificar en el campo **Dirección**.

Se admiten los siguientes formatos:
- Automático *(véase página 112)*
- Nodename *(véase página 113)*
- Dirección IP *(véase página 113)*
- Nodename a través de NAT (TCP remoto) *(véase página 113)* (NAT = traducción de direcciones de red)
- Dirección IP a través de NAT (TCP remoto) *(véase página 114)*
- Nodename a través de pasarela *(véase página 115)*
- Dirección IP a través de pasarela *(véase página 117)*
- Nodename a través de MODEM *(véase página 119)*

NOTA: Después de cambiar la **Modalidad de conexión**, puede que sea necesario realizar dos veces el procedimiento de inicio de sesión para obtener acceso al controlador seleccionado.

Modalidad de conexión: Automático

Si selecciona la opción **Automático** en la lista **Modalidad de conexión**, puede especificar el Nodename, la dirección IP o el URL (localizador uniforme de recursos) de conexión para especificar la **Dirección**.

NOTA: No utilice espacios al principio o al final de la **Dirección**.

Si ha seleccionado otra **Modalidad de conexión** y ha especificado una **Dirección** para esta modalidad, la dirección especificada seguirá estando disponible en el cuadro de texto **Dirección** si cambia a **Modalidad de conexión: →Automático**.

Ejemplo:

Modalidad de conexión →Nodename a través de NAT (TCP remoto) seleccionado y dirección y Nodename especificados

Si cambia a **Modalidad de conexión →Automático**, la información se convierte en un URL, que empieza con el prefijo `enodename3://`

Si se ha especificado una dirección IP para la modalidad de conexión (por ejemplo, cuando se ha seleccionado **Modalidad de conexión →Dirección IP**), la información se convierte en un URL que empieza con el prefijo `etcp3://`. Ejemplo: `etcp3://<IpAddress>`.

Si se ha especificado un Nodename para la modalidad de conexión (por ejemplo, se ha seleccionado **Modalidad de conexión →Nodename**), la información se convierte en un URL que empieza con el prefijo `enodename3://`. Ejemplo: `enodename3://<Nodename>`.

Modalidad de conexión →Nodename

Si selecciona la opción **Nodename** en la lista **Modalidad de conexión**, puede especificar el Nodename de un controlador para indicar la **Dirección**. El cuadro de texto se rellena automáticamente si hace doble clic en un controlador de la lista de controladores.

Ejemplo: **Nodename:** `MyM238 (10.128.158.106)`

Si el controlador seleccionado no proporciona un Nodename, **Modalidad de conexión** cambia automáticamente a **Dirección IP**, y la dirección IP de la lista aparece en el cuadro de texto **Dirección**.

NOTA: No utilice espacios al principio o al final de la **Dirección**.

NOTA: No utilice comas (**,**) en el cuadro de texto **Dirección**.

Modalidad de conexión →Dirección IP

Si selecciona la opción **Dirección IP** en la lista **Modalidad de conexión**, puede especificar la dirección IP de un controlador para indicar la **Dirección**. El cuadro de texto se rellena automáticamente si hace doble clic en un controlador de la lista de controladores.

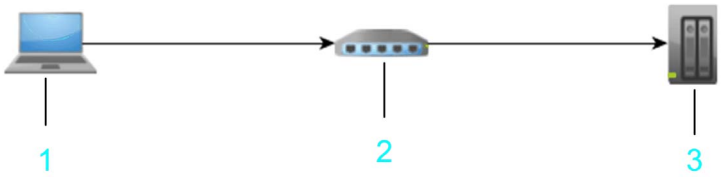
Ejemplo: **Dirección IP:** `190.201.100.100`

Si el controlador seleccionado no proporciona una dirección IP, la **Modalidad de conexión** cambia automáticamente a **Nodename**, y el Nodename de la lista aparece en el cuadro de texto **Dirección**.

NOTA: Especifique la dirección IP con el formato `<Number>.<Number>.<Number>.<Number>`

Modalidad de conexión →Nodename a través de NAT (TCP remoto)

Si selecciona la opción **Nodename a través de NAT (TCP remoto)** en la lista **Modalidad de conexión**, puede especificar la dirección de un controlador que reside después de un enrutador NAT en la red. Introduzca el Nodename del controlador, así como la dirección IP o nombre de host y puerto del enrutador NAT.



1 PC/HMI
2 enrutador NAT
3 dispositivo de destino

Ejemplo: **Dirección/Puerto NAT:** `10.128.158.106/1105` **Nodename de destino:** `MyM238 (10.128.158.106)`

NOTA: Especifique una dirección IP válida (formato `<Number>.<Number>.<Number>.<Number>`) o un nombre de host válido para la **Dirección NAT**.

NOTA: Especifique el puerto del enrutador NAT que se debe utilizar. De lo contrario, se utiliza el puerto predeterminado **1105**.

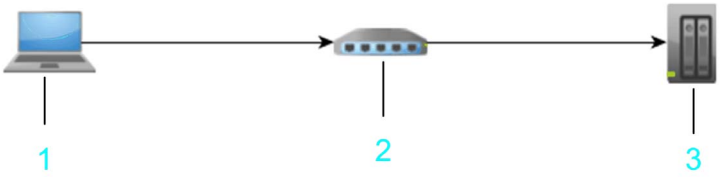
NOTA: No utilice espacios al principio o al final ni utilice comas en el cuadro de texto **Nodename de destino**.

La información que especifique se interpreta como un URL que crea un puente TCP remoto (utilizando un controlador de bloque TCP) y a continuación se conecta buscando un controlador con el Nodename especificado en la puerta de enlace local.

NOTA: El enrutador NAT puede estar ubicado en el propio controlador de destino. Puede utilizarlo para crear un puente TCP a un controlador.

Modalidad de conexión →Dirección IP a través de NAT (TCP remoto)

Si selecciona la opción **Dirección IP a través de NAT (TCP remoto)** (NAT = traducción de direcciones de red) en la lista **Modalidad de conexión**, puede especificar la dirección de un controlador que reside después de un enrutador NAT en la red. Introduzca la dirección IP del controlador, así como la dirección IP o nombre de host y puerto del enrutador NAT.



1 PC/HMI
2 enrutador NAT
3 dispositivo de destino

Ejemplo: **Dirección/Puerto NAT:** `10.128.154.206/1217` **Dirección IP de destino:** `192.168.1.55`

NOTA: Especifique una dirección IP válida (formato `<Number>.<Number>.<Number>.<Number>`) o un nombre de host válido para la **Dirección NAT**.

NOTA: Especifique el puerto del enrutador NAT que se debe utilizar. De lo contrario, se utiliza el puerto de puerta de enlace predeterminado **1217** de SoMachine.

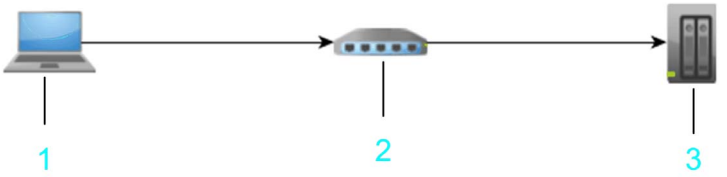
NOTA: Especifique una dirección IP válida (formato `<Number>.<Number>.<Number>.<Number>`) para la **Dirección IP de destino**.

La información que especifique se interpreta como un URL que crea un puente TCP remoto (utilizando un controlador de bloque TCP) y a continuación se conecta buscando un controlador con el Nodename especificado en la puerta de enlace local. La dirección IP se busca en el Nodename (como `MyController (10.128.154.207)`) o llamando a un servicio en cada dispositivo explorado de la puerta de enlace.

NOTA: El enrutador NAT puede estar ubicado en el propio controlador de destino. Puede utilizarlo para crear un puente TCP a un controlador.

Modalidad de conexión →Nodename a través de pasarela

Si selecciona la opción **Nodename a través de pasarela** en la lista **Modalidad de conexión**, puede especificar la dirección de un controlador que reside después o cerca de un enrutador de puerta de enlace SoMachine en la red. Especifique el Nodename del controlador, así como la dirección IP o nombre de host y puerto del enrutador de puerta de enlace SoMachine.



1 PC/HMI
2 PC/HMI/dispositivos con puerta de enlace SoMachine instalada
3 dispositivo de destino

Ejemplo: **Dirección/puerto de pas.:** `10.128.156.28`**/**`1217`**Nodename de destino:** `MyM238`

NOTA: Especifique una dirección IP válida (formato `<Number>.<Number>.<Number>.<Number>`) o un nombre de host válido para **Dirección/puerto de pas.**

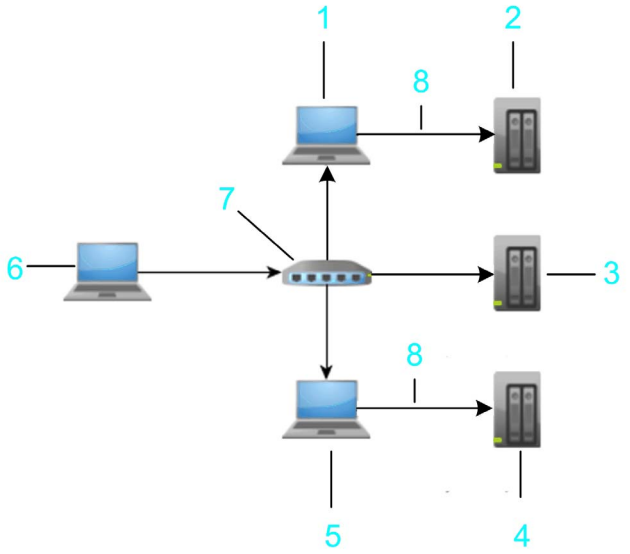
NOTA: Especifique el puerto del enrutador de puerta de enlace que se debe utilizar. De lo contrario, se utiliza el puerto de puerta de enlace predeterminado **1217** de SoMachine.

NOTA: No utilice espacios al principio o al final ni utilice comas en el cuadro de texto **Nodename de destino**.

La información que especifica se interpreta como URL. Se busca en la puerta de enlace un dispositivo con el Nodename especificado que esté conectado directamente a esta puerta de enlace. En la topología de puerta de enlace SoMachine, "conectado directamente" significa que se trata del nodo raíz o de un nodo hijo del nodo raíz.

NOTA: La puerta de enlace SoMachine puede encontrarse en el controlador de destino, el PC de destino o el propio PC local, lo que permite conectarse a un dispositivo que no tiene un nodename exclusivo pero que reside en una subred tras una red SoMachine.

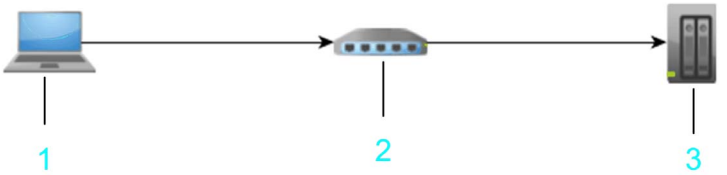
En el gráfico se muestra un ejemplo que permite una conexión desde la PCI/HMI al controlador 3 de destino (elemento 4 del gráfico) mediante la dirección de salto PC2 (elemento 5 del gráfico), que debe tener una puerta de enlace SoMachine instalada.



1 PC 1 de salto
2 controlador de destino 1: MyNotUniqueNodename
3 controlador de destino 2: MyNotUniqueNodename
4 controlador de destino 3: MyNotUniqueNodename
5 PC 2 de salto
6 PC/HMI
7 enrutador
8 Ethernet

Modalidad de conexión →Dirección IP a través de pasarela

Si selecciona la opción **Dirección IP a través de pasarela** en la lista **Modalidad de conexión**, puede especificar la dirección de un controlador que reside después o cerca de un enrutador de puerta de enlace SoMachine en la red. Introduzca la dirección IP del controlador, así como la dirección IP o nombre de host y puerto del enrutador de puerta de enlace SoMachine.



1 PC/HMI
2 PC/HMI/dispositivos con puerta de enlace SoMachine instalada
3 dispositivo de destino

Ejemplo: **Dirección/puerto de pas.:** `10.128.156.28`**/**`1217`**Dirección IP de destino:** `10.128.156.222`

NOTA: Especifique una dirección IP válida (formato `<Number>.<Number>.<Number>.<Number>`) o un nombre de host válido para **Dirección/puerto de pas.**

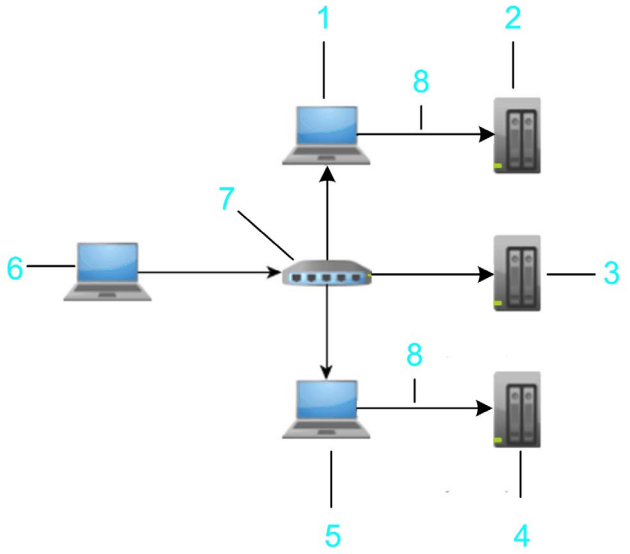
NOTA: Especifique el puerto del enrutador de puerta de enlace que se debe utilizar. De lo contrario, se utiliza el puerto de puerta de enlace predeterminado **1217** de SoMachine.

NOTA: Especifique una dirección IP válida (formato `<Number>.<Number>.<Number>.<Number>`) para la **Dirección IP de destino**.

La información que especifica se interpreta como URL. Se busca en la puerta de enlace un dispositivo con la dirección IP especificada. La dirección IP se busca en el Nodename (como `MyController (10.128.154.207)`) o llamando a un servicio en cada dispositivo explorado de la puerta de enlace.

NOTA: La puerta de enlace SoMachine puede estar ubicada en el controlador de destino, el PC de destino o en el propio PC local. Por tanto, se puede conectar a un dispositivo que no tenga un Nodename exclusivo pero que resida en una subred después de una red SoMachine.

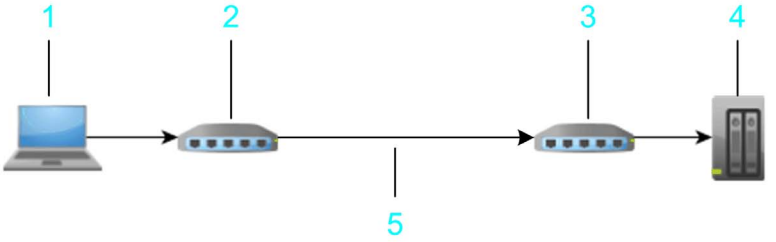
En el gráfico se muestra un ejemplo que permite la conexión desde el PC2 de salto (elemento 5 del gráfico), que debe tener una puerta de enlace SoMachine instalada en el controlador 3 de destino (elemento 4 del gráfico).



1 PC 1 de salto
2 controlador de destino 1: 10.128.156.20
3 controlador de destino 2: 10.128.156.20
4 controlador de destino 3: 10.128.156.20
5 PC 2 de salto
6 PC/HMI
7 enrutador
8 Ethernet

Modalidad de conexión →Nodename a través de MODEM

Si selecciona la opción **Nodename a través de MODEM** en la lista **Modalidad de conexión**, puede especificar un controlador que resida después de una línea de módem.



1 PC/HMI
2 PC/HMI/MODEM
3 módem de destino
4 dispositivo de destino
5 línea telefónica

Para establecer una conexión con el módem, haga clic en el botón **MODEM →Conectar**. En el cuadro de diálogo **Modem Configuration**, especifique el número de teléfono (**Número de teléfono**) del módem de destino y configure los ajustes de comunicación. Haga clic en **Aceptar** para confirmar y establecer una conexión con el módem.

Si se detiene la puerta de enlace SoMachine y se vuelve a iniciar, finalizarán las conexiones de la puerta de enlace local. SoMachine muestra un mensaje que requiere confirmación antes de que empiece el proceso de reinicio.

Una vez establecida correctamente la conexión con el módem, el botón **MODEM** cambia de **Conectar** a **Desconectar**. La lista de controladores se borra y se actualiza explorando la conexión de módem para buscar los controladores conectados. Puede hacer doble clic en un elemento de la lista de controladores o especificar un Nodename en el cuadro de texto **Nodename de destino:** para conectar a un controlador específico.

Haga clic en el botón **MODEM →Desconectar** para finalizar la conexión de módem y detener y reiniciar la puerta de enlace SoMachine. La lista de controladores se borra y se actualiza explorando la red Ethernet.

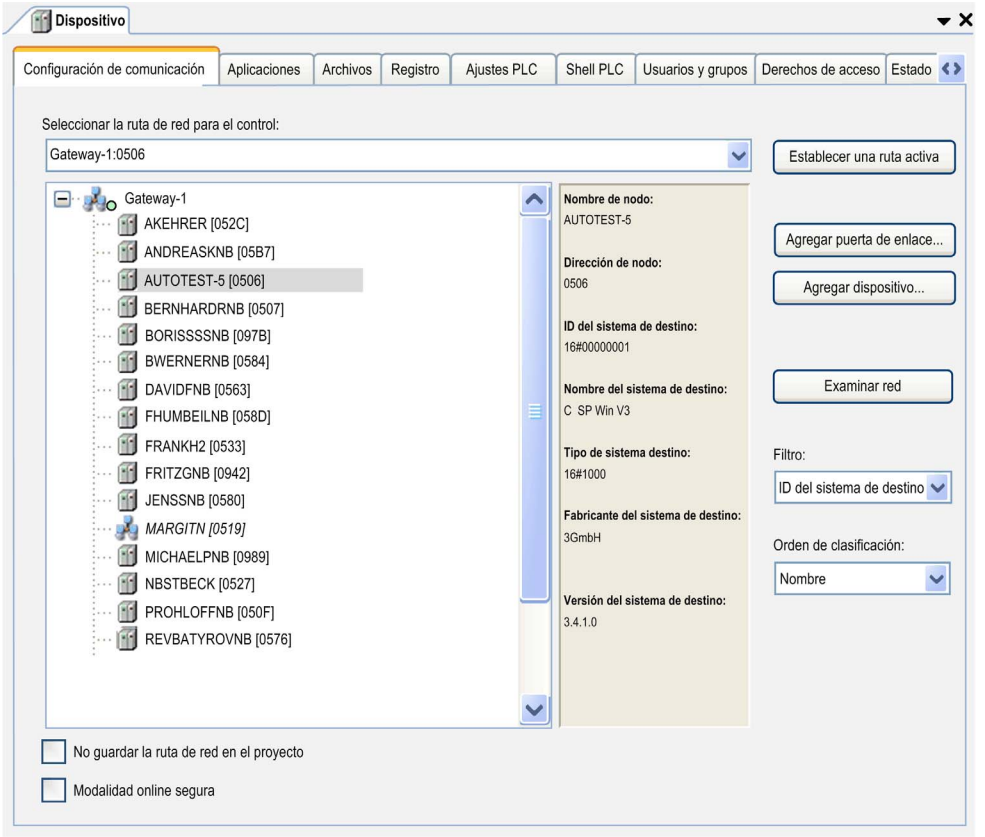
Configuración de comunicación

Descripción general

La vista **Configuración de comunicación** del editor de dispositivos se utiliza para configurar los parámetros para la comunicación entre el dispositivo y el sistema de programación.

Sólo es visible si la comunicación entre el dispositivo y el sistema de programación se establece mediante la ruta activa. Este es el ajuste predeterminado para SoMachine V3.1 y versiones anteriores. Puede seleccionar entre un establecimiento de comunicación mediante una ruta activa y una dirección IP en el cuadro de diálogo **Configuración del proyecto →Configuración de comunicación**. Si la opción **Marque mediante "dirección IP"** está seleccionada, la vista **Selección de controlador** se muestra en el editor de dispositivos en lugar de en la vista **Configuración de comunicación**. Este es el ajuste predeterminado para SoMachine V4.0 y versiones posteriores.

Vista **Configuración de comunicación** del editor de dispositivos



Esta vista está dividida en 2 partes:

- En la parte izquierda se muestran los canales de puerta de enlace configurados actualmente en una estructura de árbol.
- En la parte derecha se muestra la información y los datos correspondientes.

Descripción de la estructura de árbol

Cuando crea el primer proyecto en su sistema local, la **Puerta de enlace** local ya está disponible como nodo en el árbol. Esta puerta de enlace se inicia automáticamente al arrancar el sistema.

La configuración de la puerta de enlace se muestra en la parte derecha de la ventana:

Ejemplo:

Nombre de nodo: Gateway-1

Puerto: 1217

Dirección IP: 127.0.0.1

Controlador: TCP/IP

Cuando la puerta de enlace está en ejecución, se muestra una viñeta verde delante del nodo **Puerta de enlace**; de lo contrario, se muestra una viñeta roja. La viñeta es gris si aún no se ha producido contacto con la puerta de enlace (en función de algunos protocolos de comunicación, no se permite sondear la puerta de enlace, por lo que no se puede mostrar el estado).

Con sangría debajo del nodo **Puerta de enlace** (se abre o cierra por medio del signo **+/-**), verá las entradas de todos los dispositivos a los que se puede obtener acceso mediante esta puerta de enlace. Las entradas de dispositivo vienen precedidas por un símbolo  . Las entradas que cuentan con un ID de destino diferente al del dispositivo actualmente configurado en el proyecto se muestran en una fuente de color gris. Para obtener una lista actualizada de los dispositivos actualmente disponibles, use el botón **Examinar red**.

Los nodos de dispositivo constan de un símbolo seguido de un nombre de nodo y la dirección del nodo. En la parte derecha de la ventana, se muestran las opciones respectivas: **ID del sistema de destino**, **Nombre del sistema de destino**, **Tipo de sistema destino**, **Fabricante del sistema de destino** y **Versión del sistema de destino**.

En el campo **Seleccionar la ruta de red para el control**, se especifica automáticamente el canal de puerta de enlace seleccionando el canal en la estructura del árbol.

Función de filtro y ordenación

Puede filtrar y ordenar los nodos de puerta de enlace y dispositivo mostrados en el árbol por medio de los cuadros de selección situados en la parte derecha de la vista:

- **Filtro**: le permite reducir las entradas de la estructura de árbol a aquellos dispositivos con un **ID de destino** que coincida con el del dispositivo configurado en el proyecto.
- **Orden de clasificación**: le permite ordenar las entradas de la estructura de árbol según los campos **Nombre** o **Dirección de nodo** en orden alfabético o ascendente.

Descripción de los botones/comandos

Para cambiar la configuración de comunicación, existen los siguientes botones o comandos en el menú contextual:

Botón/comando	Descripción
Establecer una ruta activa	Este comando establece el canal de comunicación seleccionado como la ruta activa al controlador. Consulte la descripción del comando **Establecer una ruta activa**. Al hacer clic en el nodo en la estructura de árbol se obtiene el mismo efecto.
Agregar puerta de enlace...	El comando **Puerta de enlace** abre el cuadro de diálogo Puerta de enlace, donde puede definir una puerta de enlace para que se añada a la configuración actual. Consulte la descripción del comando **Agregar puerta de enlace**.
Agregar dispositivo...	Este comando abre el cuadro de diálogo **Agregar dispositivo** en el que puede definir manualmente un dispositivo que se añadirá a la entrada de puerta de enlace seleccionada (tenga en cuenta la funcionalidad **Examinar red**). Consulte la descripción del comando **Agregar dispositivo...**
Modificar puerta de enlace...	Este comando abre el cuadro de diálogo **Puerta de enlace** para editar la configuración de la puerta de enlace seleccionada. Consulte la descripción del comando **Modificar puerta de enlace...**
Eliminar el dispositivo seleccionado	Este comando elimina el dispositivo seleccionado del árbol de configuración. Consulte la descripción del comando **Eliminar el dispositivo seleccionado**.

Botón/comando	Descripción
Búsqueda del dispositivo por la dirección	Este comando busca dispositivos en la red que contengan la dirección especificada aquí en el árbol de configuración. Aquellos que se encuentren se representarán a continuación en la puerta de enlace con la dirección de nodo especificada junto con su nombre. La búsqueda hace referencia a los dispositivos que hay debajo de la puerta de enlace en cuyo árbol hay una entrada seleccionada. Consulte la descripción del comando **Búsqueda del dispositivo por la dirección**.
Búsqueda del dispositivo por nombre	Este comando busca dispositivos en la red que contengan los nombres especificados aquí en el árbol de configuración (la búsqueda distingue entre mayúsculas y minúsculas). Aquellos que se encuentren se representarán a continuación en la puerta de enlace con el nombre especificado junto con su dirección de nodo exclusiva. La búsqueda hace referencia a los dispositivos que hay debajo de la puerta de enlace en cuyo árbol hay una entrada seleccionada. Consulte la descripción del comando **Búsqueda del dispositivo por nombre**.
Búsqueda del dispositivo por la dirección IP	Este comando busca dispositivos en la red que contengan la dirección IP especificada aquí en el árbol de configuración. Aquellos que se encuentren se representarán a continuación en la puerta de enlace con la dirección de nodo especificada junto con su nombre. La búsqueda hace referencia a los dispositivos que hay debajo de la puerta de enlace en cuyo árbol hay una entrada seleccionada. Consulte la descripción del comando **Búsqueda del dispositivo por la dirección IP**.
Conexión a la puerta de enlace local	Este comando abre un cuadro de diálogo para la configuración de una puerta de enlace local y, por tanto, ofrece una alternativa a la edición manual del archivo *Gateway.cfg*. Consulte la descripción del comando **Conexión a la puerta de enlace local…**
Examinar red	Este comando inicia una búsqueda de todos los dispositivos disponibles en la red local. El árbol de configuración de la puerta de enlace afectada se actualizará consecuentemente. Consulte la descripción del comando **Examinar red**.

Descripción de las opciones

Hay 2 opciones disponibles debajo de la estructura de árbol:

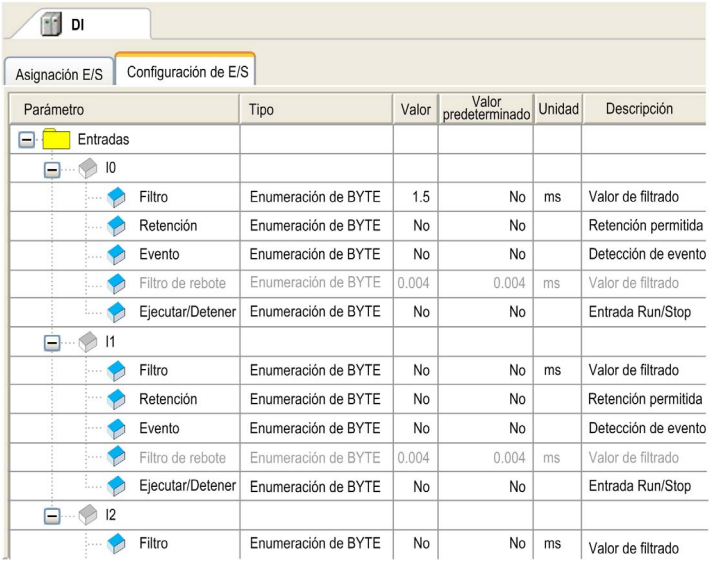
Opción	Descripción
No guardar la ruta de red en el proyecto	Active esta opción si la definición de ruta de red actual no se debe almacenar en el proyecto, sino en la configuración de opciones local del ordenador. Por consiguiente, el ajuste de ruta se restaura si el proyecto se vuelve a abrir en el mismo ordenador. Se deberá volver a definir si el proyecto se usa en otro sistema.
Modalidad online segura	Active esta opción si, por motivos de seguridad, se debe solicitar confirmación al usuario cuando seleccione uno de los comandos online siguientes: **Forzar valores, Descarga múltiple, Desactivar la lista de forzado, Ciclo individual, Inicio, Parada, Escribir valores**.

Configuración

Descripción general

La vista **Configuración** sólo está disponible en el editor de dispositivos si la opción **Mostrar los editores genéricos de configuración** del cuadro de diálogo **Herramientas** →**Opciones** → **Editor de dispositivos** está activada. La vista **Configuración** muestra los parámetros específicos del dispositivo y, si lo permite la descripción del dispositivo, ofrece la posibilidad de editar los valores de los parámetros.

Vista **Configuración** del editor de dispositivos



Parámetro	Tipo	Valor	Valor predeterminado	Unidad	Descripción
Entradas					
I0					
Filtro	Enumeración de BYTE	1.5	No	ms	Valor de filtrado
Retención	Enumeración de BYTE	No	No		Retención permitida
Evento	Enumeración de BYTE	No	No		Detección de evento
Filtro de rebote	Enumeración de BYTE	0.004	0.004	ms	Valor de filtrado
Ejecutar/Detener	Enumeración de BYTE	No	No		Entrada Run/Stop
I1					
Filtro	Enumeración de BYTE	No	No	ms	Valor de filtrado
Retención	Enumeración de BYTE	No	No		Retención permitida
Evento	Enumeración de BYTE	No	No		Detección de evento
Filtro de rebote	Enumeración de BYTE	0.004	0.004	ms	Valor de filtrado
Ejecutar/Detener	Enumeración de BYTE	No	No		Entrada Run/Stop
I2					
Filtro	Enumeración de BYTE	No	No	ms	Valor de filtrado

La vista contiene los siguientes elementos:

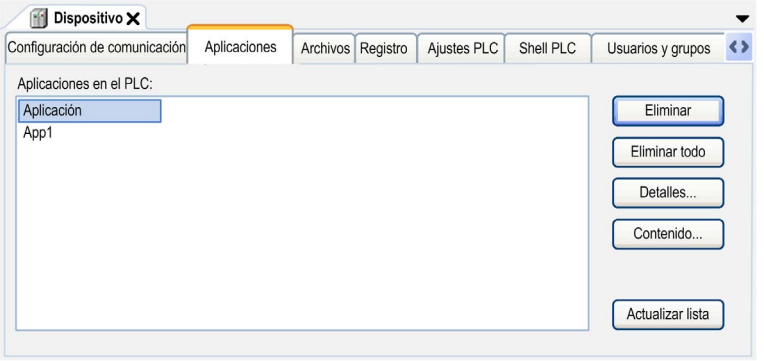
Elemento	Descripción
Parámetro	Nombre del parámetro, no editable.
Tipo	Tipo de datos del parámetro, no editable.
Valor	Principalmente, el valor predeterminado del parámetro se muestra directamente o mediante un nombre simbólico. Si el parámetro se puede modificar (esto depende de la descripción del dispositivo, los parámetros no editables se muestran de color gris), haga clic en la celda de la tabla para abrir un cuadro de edición o una lista de selección para cambiar el valor. Si el valor es una especificación de archivo, el cuadro de diálogo estándar para abrir un archivo se abre haciendo doble clic en la celda. Esto le permite seleccionar otro archivo.
Valor predeterminado	Valor de parámetro predeterminado, no editable.
Unidad	Unidad del valor del parámetro (por ejemplo: **ms** para milisegundos), no editable.
Descripción	Breve descripción del parámetro, no editable.

Aplicaciones

Descripción general

La vista **Aplicaciones** del editor de dispositivos sirve para explorar y eliminar aplicaciones en el controlador. Puede haber disponible información sobre el contenido de la aplicación así como algunos detalles de las propiedades de la aplicación.

Vista **Aplicaciones** del editor de dispositivos



Descripción de los elementos

La vista **Aplicaciones** proporciona los elementos siguientes:

Elemento	Descripción
Aplicaciones en el PLC	En este cuadro de texto se muestran los nombres de las aplicaciones encontradas en el controlador durante la última exploración (haciendo clic en **Actualizar**). Si todavía no se ha ejecutado ninguna exploración o si no se puede llevar a cabo la exploración porque no hay ninguna puerta de enlace configurada *(véase página 120)* para una conexión, se muestra un mensaje.
Eliminar **Eliminar todo**	Haga clic en estos botones para eliminar del controlador la aplicación seleccionada de la lista o todas las aplicaciones.
Detalles	Haga clic en este botón para abrir un cuadro de diálogo que muestra la información que se ha definido en la ficha **Información** del cuadro de diálogo **Propiedades** del objeto de aplicación.
Contenido	Si en **Visualizar** →**Propiedades** →**Opciones de creación de aplicaciones** la opción **Descarga Aplicación Info** está activada para el objeto de aplicación *(véase SoMachine, Comandos de menú, Ayuda en línea)*, se carga información adicional sobre el contenido de la aplicación en el controlador. Haga clic en el botón **Contenido** para ver las distintas POU en una vista de comparación. Tras varias descargas, esta información permite comparar el código de la nueva aplicación con el que ya estaba disponible en el controlador. Esto proporciona una información más detallada para decidir cómo iniciar sesión. Para obtener más información, consulte la descripción del comando **Inicio de sesión**.
Actualizar lista	Haga clic en este botón para ver qué aplicaciones hay en el controlador. La lista se actualizará en consecuencia.

Verificaciones antes de cargar una aplicación en el controlador

Antes de que una aplicación se cargue en el controlador se llevan a cabo las siguientes verificaciones:

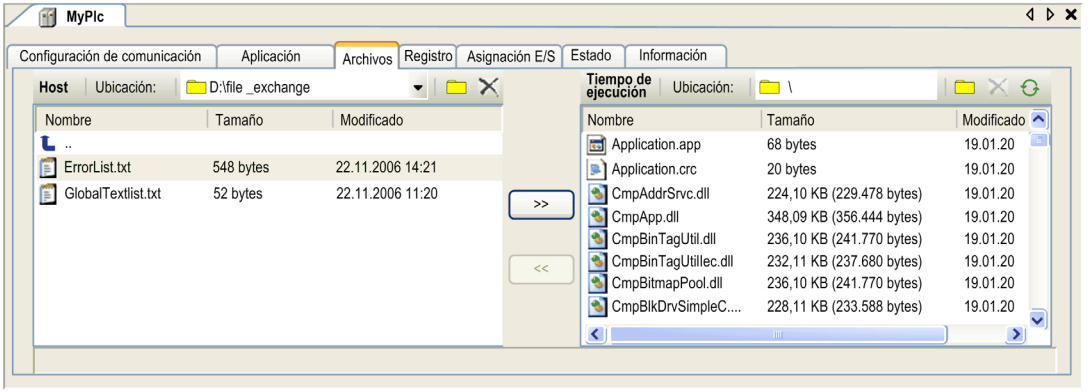
- Se compara la lista de aplicaciones del controlador con las disponibles en el proyecto. Si se detectan discrepancias, se muestran los cuadros de diálogo correspondientes para cargar las aplicaciones todavía no disponibles en el controlador o para eliminar otras aplicaciones del controlador.
- Se verifican las POU implementadas de forma externa en la aplicación que se van a cargar para comprobar si también están disponibles en el controlador. Si no están disponibles en el controlador, se generará el mensaje correspondiente (**Referencia sin resolver**) en un cuadro de mensaje, así como en la vista **Mensajes**, si la opción **Descargar** está seleccionada.
- Se comparan los parámetros (variables) de las POU de la aplicación que se van a cargar con los de las POU con el mismo nombre de la aplicación ya disponibles en el controlador (validación de firmas). Si se detectan discrepancias, se generará el mensaje correspondiente (**Conflicto de firmas**) en un cuadro de mensaje, así como en la vista **Mensajes** si la opción **Descargar** está seleccionada.
- Si en **Visualizar** →**Propiedades** →**Opciones de creación de aplicaciones** la opción **Descarga Aplicación Info** está activada, se cargará información adicional sobre el contenido de la aplicación en el controlador. Consulte la descripción del botón **Contenido** en la tabla anterior.

Archivos

Descripción general

La vista **Archivos** del editor de dispositivos sirve para transferir archivos entre el host y el controlador. Esto significa que puede elegir cualquier archivo de un directorio de la red local para copiarlo en el directorio de archivos del sistema de tiempo de ejecución que esté conectado en ese momento o viceversa.

Vista **Archivos** del editor de dispositivos



Esta vista está dividida en 2 partes:

- En la parte izquierda se muestran los archivos del **Host**.
- En la parte derecha se muestran los archivos del sistema de **Tiempo de ejecución**.

Descripción de los elementos

La vista **Archivos** proporciona los elementos siguientes:

Elemento	Descripción
	Actualiza la lista **Tiempo de ejecución**.
	Crea una nueva carpeta en la que puede copiar los archivos.
	Elimina los archivos o carpetas seleccionados de la lista.
Ubicación	Especifica la carpeta del sistema de archivos respectivo que se puede utilizar para la transferencia de archivos. Selecciona una entrada de la lista o se busca en el árbol del sistema de archivos.
<< >>	Seleccione los archivos que se copiarán en el árbol del sistema de archivos. Puede seleccionar varios archivos simultáneamente o puede seleccionar una carpeta para copiar todos los archivos que contiene. Para copiar los archivos seleccionados en **Host** al directorio **Tiempo de ejecución**, haga clic en **>>**. Para copiar los archivos seleccionados en **Tiempo de ejecución** al directorio **Host**, haga clic en **<<**. Si no hay ningún archivo disponible aún en el directorio de destino, se creará allí. Si ya hay un archivo con el nombre especificado y no está protegido contra escritura, se sobrescribirá. Si está protegido contra escritura, se generará el mensaje oportuno.

Registro

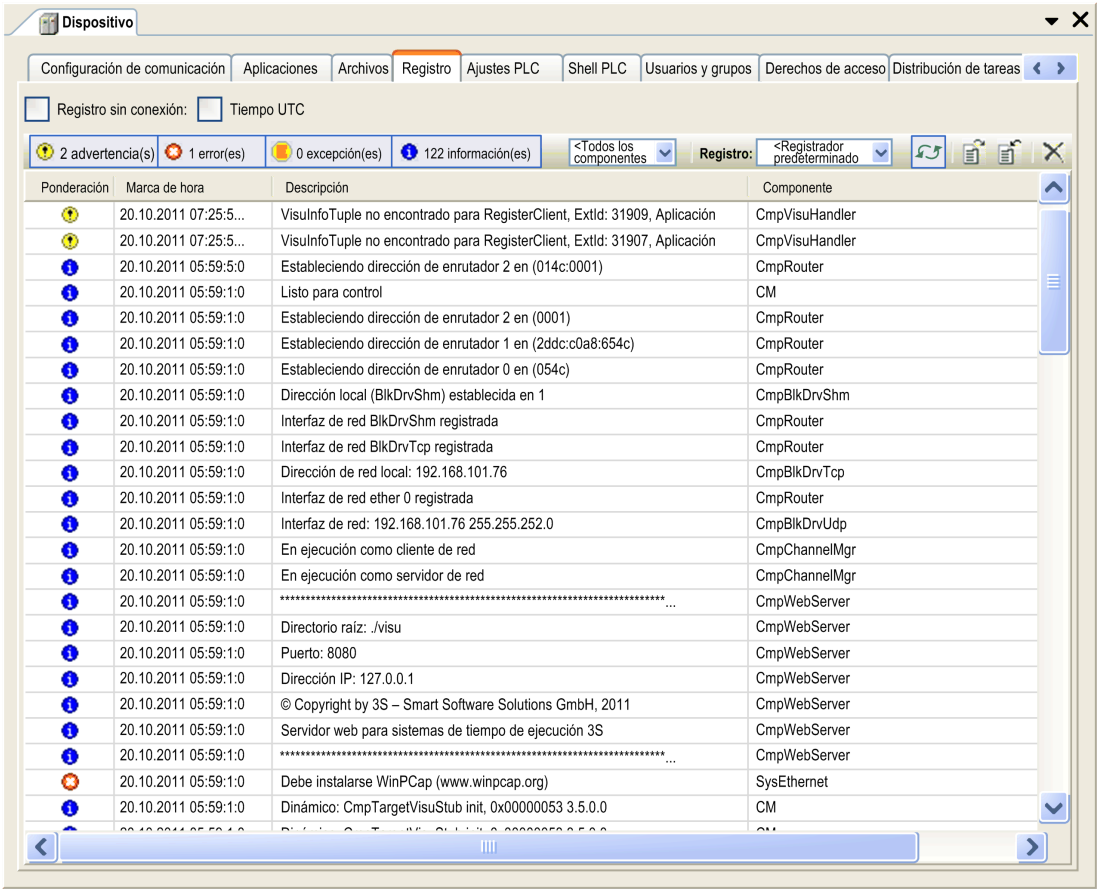
Descripción general

La vista **Registro** del editor de dispositivos sirve para mostrar los eventos que se han registrado en el sistema de tiempo de ejecución del controlador.

Incluye:

- eventos al arrancar o apagar el sistema (componentes cargados y sus versiones)
- descarga de aplicación y descarga de proyecto de arranque
- entradas específicas del cliente
- entradas de registro de controladores de E/S
- entradas de registro del servidor de datos

Vista **Registro** del editor de dispositivos



Descripción de los elementos

La vista **Registro** proporciona los elementos siguientes:

Elemento	Descripción
Ponderación	Los eventos del registro se agrupan en cuatro categorías: ● **advertencia** ● **error** ● **excepción** ● **información** Los botones de la barra encima de la lista indican la cifra actual de registros en la categoría respectiva. Haga clic en los botones para activar o desactivar la visualización de las entradas de cada categoría.
Marca de hora	fecha y hora por ejemplo, **12.01.2007 09:48**
Descripción	descripción del evento por ejemplo, **Error en la función de importación de <CmpFileTransfer>**
Componente	Aquí puede seleccionar un componente concreto para que se muestren sólo las entradas de registro relacionadas con ese componente. La configuración predeterminada es **<Todos los componentes>**.
Registro	La lista de selección ofrece las anotaciones disponibles. La configuración predeterminada es **<Registrador predeterminado>**, definida por el sistema de tiempo de ejecución.
	Actualiza la lista.
	Exporta la lista a un archivo XML. Se abre el cuadro de diálogo estándar para guardar un archivo. El filtro del archivo se establece en **archivos xml (*.xml)**. El archivo del registro se guarda con el nombre de archivo especificado y con la extensión .XML en el directorio seleccionado.
	Muestra las entradas del registro guardadas en un archivo XML que se han podido exportar como se describe más arriba. Se abre el cuadro de diálogo estándar para examinar un archivo. El filtro se establece en **archivos xml (*.xml)**. Seleccione el archivo de registro que desee. Las entradas de este archivo se muestran en una ventana aparte.
	Borra la tabla de registro actual, es decir, elimina todas las entradas que se muestran.
Registro sin conexión	Esta opción no se utiliza en SoMachine.
Tiempo UTC	Active esta opción para ver la marca de hora del sistema de tiempo de ejecución. Si se desactiva, se muestra la marca de la hora local del ordenador (según la zona horaria del sistema operativo).

Ajustes PLC

Descripción general

La vista **Ajustes PLC** del editor de dispositivos sirve para configurar ciertos valores generales del controlador.

Vista **Ajustes PLC** del editor de dispositivos

Descripción de los elementos

La vista **Ajustes PLC** proporciona los elementos siguientes:

Elemento	Descripción
Aplicación para manejo E/S:	Defina aquí la aplicación asignada al dispositivo en el árbol **Dispositivos** que se supervisará para la asignación de E/S. Para SoMachine sólo hay una aplicación disponible.
Área **Ajustes PLC**	
Actualizar E/S en parada	Si esta opción está activada (valor predeterminado), los valores de los canales de entrada y salida también se actualizarán cuando se detenga el controlador. Si se agota el tiempo del watchdog, se asignarán a las salidas los valores predeterminados definidos.
Comportamiento de las salidas en parada	En la lista de selección, elija una de las siguientes opciones para definir cómo se deben gestionar los valores de los canales de salida si se detiene el controlador: ● **Mantener los valores** no se modificarán los valores actuales. ● **Establecer todas las salidas a los valores predeterminados** se asignarán los valores predeterminados resultantes de la asignación. ● **Ejecutar programa** Puede determinar el comportamiento de las salidas mediante un programa disponible en el proyecto. Especifique el nombre de este programa aquí y se ejecutará cuando el controlador se detenga. Haga clic en botón **...** para utilizar **Accesibilidad** para este fin.
Actualizar todas las variables en todos los dispositivos	Si esta opción está activada, se actualizarán las variables de E/S de los dispositivos de la configuración de controlador actual en cada ciclo de la tarea de ciclo de bus. Esto corresponde a la opción **Actualizar siempre las variables**. Puede definirlo por separado para cada dispositivo en la vista **Asignación de E/S** *(véase página 157)*.
Área **Opciones de ciclo de bus**	
Tarea de ciclo de bus	La lista de selección ofrece todas las tareas definidas actualmente en la **Configuración de tareas** de la aplicación activa (por ejemplo, **MAST**). El valor predeterminado **MAST** se especifica automáticamente. **<Sin especificar>** significa que la tarea se selecciona de acuerdo con valores internos del controlador, que dependen del controlador. Esta puede ser la tarea con el tiempo de ciclo más corto, pero también podría ser la que tuviera el tiempo de ciclo más largo.

NOTA: Establecer la tarea de ciclo de bus como **<Sin especificar>** puede provocar un comportamiento imprevisto de la aplicación. Consulte la guía de programación específica de su controlador.

⚠ ATENCIÓN
FUNCIONAMIENTO IMPREVISTO DEL EQUIPO Si no está seguro de los ajustes de tareas de ciclo de bus, no establezca **Tarea de ciclo de bus** en **<<Sin especificar>**, sino que seleccione una tarea dedicada de la lista. **El incumplimiento de estas instrucciones puede causar lesiones o daño al equipo.**

Configuración adicional

El valor **Variables de forzado para la asignación E/S:** sólo está disponible si lo admite el dispositivo. Si la opción está activada, por cada canal de E/S que se asigne a una variable en el cuadro de diálogo **Asignación de E/S**, se crearán 2 variables globales en cuanto se genere la aplicación. Estas variables pueden utilizarse en una visualización de HMI para forzar el valor de E/S. Para obtener más información, consulte el capítulo *Asignación de E/S* *(véase página 153)*.

Usuarios y grupos

Descripción general

La administración de usuarios y grupos de derechos de acceso difiere según el controlador que utilice. Para la mayoría de los dispositivos que admiten la administración de usuarios online (como los controladores M258, M241, M251 y LMC••8), la vista **Usuarios y grupos** descrita en este capítulo sirve para gestionar cuentas de usuario y grupos de acceso de usuarios, así como sus derechos de acceso asociados. Esto le permite controlar el acceso en proyectos y dispositivos de SoMachine en modalidad online.

Para administrar los derechos de usuario, debe iniciar sesión como usuario **Administrator**.

⚠ ATENCIÓN

SIN AUTENTICACIÓN, ACCESO NO AUTORIZADO

- Evite la exposición de los controladores y redes de controladores a las redes públicas e Internet en la medida de lo posible.
- Utilice capas de seguridad adicionales, como VPN para el acceso remoto, e instale mecanismos cortafuegos.
- Permita el acceso únicamente a las personas autorizadas.
- Cambie las contraseñas predeterminadas en el inicio y modifíquelas con frecuencia.
- Valide la efectividad de estas medidas periódica y frecuentemente.

El incumplimiento de estas instrucciones puede causar lesiones o daño al equipo.

NOTA: Puede utilizar los comandos relacionados con la seguridad *(véase SoMachine, Comandos de menú, Ayuda en línea)*, que proporcionan un método para añadir, editar y eliminar a un usuario en la administración de usuarios online del dispositivo de destino en el cual se ha iniciado la sesión.

NOTA: Debe establecer derechos de acceso utilizando el software SoMachine. Si ha *clonado* una aplicación de un controlador a otro, deberá habilitar y establecer derechos de acceso de usuario en el controlador de destino.

NOTA: La única manera de obtener acceso a un controlador que tenga derechos de acceso de usuario habilitados y para los que no tiene las contraseñas es realizando una operación **Actualizar firmware** mediante una tarjeta SD o un lápiz de memoria USB (consulte la *Guía del usuario del Asistente del controlador de SoMachine* para obtener más información), en función del soporte de su controlador en concreto, o ejecutando un script *(véase página 919)*. De este modo se eliminará de forma efectiva la aplicación existente de la memoria del controlador, pero se restaurará la posibilidad de acceder al controlador.

Para controladores **Soft PLC**, en el editor de dispositivos se muestran una vista de **Usuarios y grupos** específica y una vista **Derechos de acceso** aparte. Estas vistas específicas se describen en el capítulo *Administración de usuarios para Soft PLC (véase página 969)*.

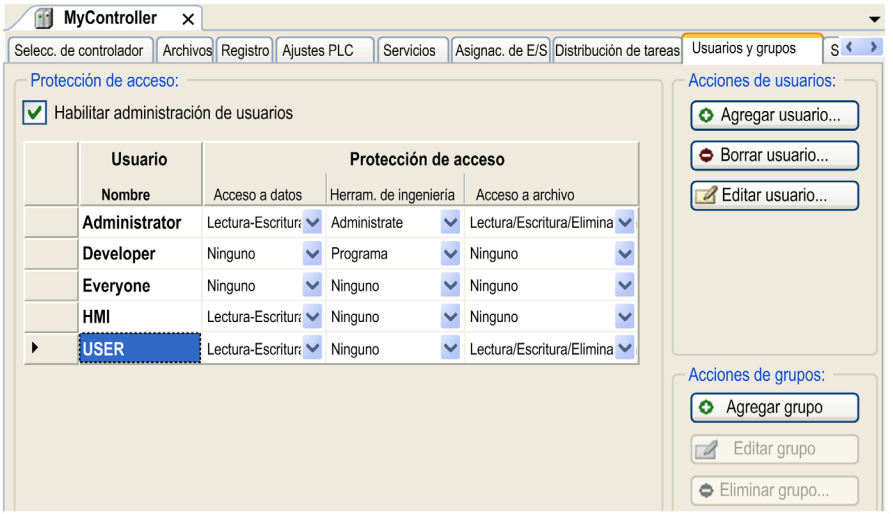
Si desea que determinadas funciones de un controlador sólo las puedan ejecutar usuarios autorizados, la vista **Usuarios y grupos** le permite definir usuarios, asignar derechos de acceso y requerir autenticación de usuario al iniciar sesión.

Para realizar estas acciones, puede crear usuarios y configurar sus derechos de acceso a datos, herramientas de ingeniería y archivos utilizando los botones del área **Acciones del usuario**. Puede crear grupos de derechos de acceso de usuarios y configurar cada permiso individualmente utilizando los botones del área **Acciones de grupos**.

Vista Usuarios y grupos

La vista **Usuarios y grupos** le permite gestionar el acceso de usuarios a proyectos y dispositivos. Los usuarios y los derechos correspondientes son válidos para todos los dispositivos de un proyecto.

Vista **Usuarios y grupos** del editor de dispositivos:



Área Protección de acceso

El área **Protección de acceso** de la vista **Usuarios y grupos** contiene la casilla **Habilitar administración de usuarios**. Esta casilla está inhabilitada de forma predeterminada, lo que significa que la administración de usuarios no está activa. Se proporciona acceso libre a proyectos y dispositivos. Para poder gestionar cuentas de usuario y grupos de derechos de acceso de usuarios y asignar derechos de acceso, active la casilla **Habilitar administración de usuarios**. Para cambiar los valores de administración de usuarios, debe iniciar sesión como **Administrador** o tener derechos de usuario **Administrate** para su inicio de sesión. Durante el primer inicio de sesión como administrador, se le solicitará que cambie la contraseña predeterminada.

Debajo de la casilla **Habilitar administración de usuarios**, se proporciona una lista de usuarios definidos (en la columna **Usuario**) y los grupos de derechos de acceso que se les conceden (en las columnas **Protección de acceso** ordenadas por tipo).

La lista contiene cinco usuario de forma predeterminada. Dispone de acceso limitado en estos usuarios predeterminados según se lista en la tabla:

Usuario predeterminado	Descripción	Puede eliminarse	Puede cambiarse el nombre	Pueden cambiarse los permisos	Contraseña predeterminada (puede cambiarse)
Administrator	El usuario **Administrator** no tiene restricciones de acceso; se utiliza para configurar derechos de usuario.	No	No	No	`Administrator` Debe cambiarse con el primer inicio de sesión.
Everyone	El usuario **Everyone** se utiliza cuando no ha iniciado sesión. Si suprime permisos, debe iniciar sesión con un usuario que tenga permisos.	No	No	Sí	No se asigna contraseña.
USER	El servidor web utiliza el usuario **USER** para acceder al controlador.	No	Sí	Sí	`USER`
HMI	La HMI utiliza el usuario **HMI** para conectarse al controlador.	No	Sí	Sí	`HMI`
Developer	Al usuario **Developer** se le otorgan permisos adecuados para diseñar un proyecto.	Sí	Sí	Sí	`Developer`

Los derechos de acceso de usuarios están contenidos en grupos y se clasifican en tres áreas, o tipos de acceso, bajo **Protección de acceso**:

- **Acceso a datos**: incluye el acceso a aplicaciones que intentan leer/escribir datos del dispositivo, como HMI u OPC.
- **Herramientas de ingeniería**: incluye acceso a herramientas de programación, como SoMachine y Controller Assistant.
- **Acceso a archivo**: incluye el acceso al sistema de archivos interno o externo, como soportes externos, protocolo SoMachine, FTP o web.

Si hace clic en el cuadro desplegable de un usuario definido en alguna de las columnas de tipo de acceso, se presentará una lista de los grupos de derechos de acceso de usuarios que puede elegir para un determinado usuario y tipo de acceso. Para obtener más información sobre grupos de derechos de acceso, consulte *Administración de grupos de derechos de acceso* *(véase página 144)*.

Área Acciones de usuario

Los botones sólo están disponibles si la opción **Habilitar administración de usuarios** está activada.

Al hacer clic en un botón se abre un cuadro de diálogo que le solicita que inicie sesión como usuario **Administrador**, porque es el único usuario que tiene derechos para la administración de usuarios.

Los botones de **Acciones del usuario** se utilizan para realizar funciones de administración estándar en los usuarios:

Botón	Descripción
Agregar usuario...	Haga clic en este botón para añadir un nuevo usuario a la lista. Se abre el cuadro de diálogo **Agregar usuario**. Introduzca un **Nombre de usuario**, un **Nombre** completo, una **Descripción** y una **Contraseña:**. Repita la contraseña en el campo **Confirmar contraseña**. Para que el nuevo usuario esté disponible para su uso, active la opción **Activado**.
Borrar usuario...	Haga clic en este botón para eliminar el usuario de la lista del área **Protección de acceso**.
Editar usuario...	Haga clic en este botón para modificar el usuario seleccionado en la lista del área **Protección de acceso**. Se abre el cuadro de diálogo **Editar usuario**. Corresponde al cuadro de diálogo **Agregar usuario** (véase arriba), que contiene la configuración del usuario definido actualmente. Para que el usuario esté disponible para su uso, active la opción **Activado**.

Cambie las contraseñas predeterminadas para todos los usuarios predeterminados (**USER**, **HMI**, **Developer**). Además, analice detenidamente las implicaciones de otorgar derechos de acceso al usuario predeterminado **Everyone**. Por una parte, otorgar derechos de acceso al usuario **Everyone** le permitirá ejecutar scripts desde el puerto USB o SD (en función de la referencia de su controlador), que, a su vez, concederá a cualquier persona los derechos de acceso que otorgue sin tener que iniciar sesión antes.

⚠ ADVERTENCIA

ACCESO NO AUTORIZADO A DATOS

- Cambie inmediatamente todas las contraseñas predeterminadas por nuevas contraseñas seguras.
- No distribuya las contraseñas a personal no autorizado.
- Limite los derechos de acceso del usuario **Everyone** a únicamente aquellos que sean esenciales para sus necesidades de aplicación.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

NOTA: Una contraseña segura es aquella que no se ha compartido ni distribuido a ninguna persona no autorizada y que no contiene ninguna información personal u obvia. Además, una combinación de mayúsculas, minúsculas y números ofrece mayor seguridad. Debería elegir una contraseña con una longitud de al menos siete caracteres.

Área Acciones de grupos

Los botones sólo están disponibles si la opción **Habilitar administración de usuarios** está activada.

Al hacer clic en un botón se abre un cuadro de diálogo que le solicita que inicie sesión como usuario **Administrador**, porque es el único usuario que tiene derechos para la administración de usuarios.

Los botones se utilizan para realizar funciones de administración de derechos de acceso en grupos:

Botón	Descripción
Agregar grupo	Haga clic en este botón para crear un nuevo grupo. Se abre el cuadro de diálogo **Grupos personalizados**.
Editar grupo	Haga clic en este botón para modificar un grupo. Se abre el cuadro de diálogo **Grupos personalizados**. En la lista **Grupo para editar**, seleccione el grupo que desea editar. La configuración definida para el grupo seleccionado se muestra en las tablas siguientes.
Eliminar grupo...	Haga clic en este botón para eliminar el grupo seleccionado en la lista del área **Protección de acceso**. Nota: Algunos grupos predeterminados no pueden eliminarse *(véase página 137)*.

Grupos de derechos de acceso para tipos de protección de acceso

De forma predeterminada, hay predefinido un número de grupos de derechos de acceso, que están disponibles en las listas desplegables bajo los diferentes tipos de **Protección de acceso** para cada usuario definido.

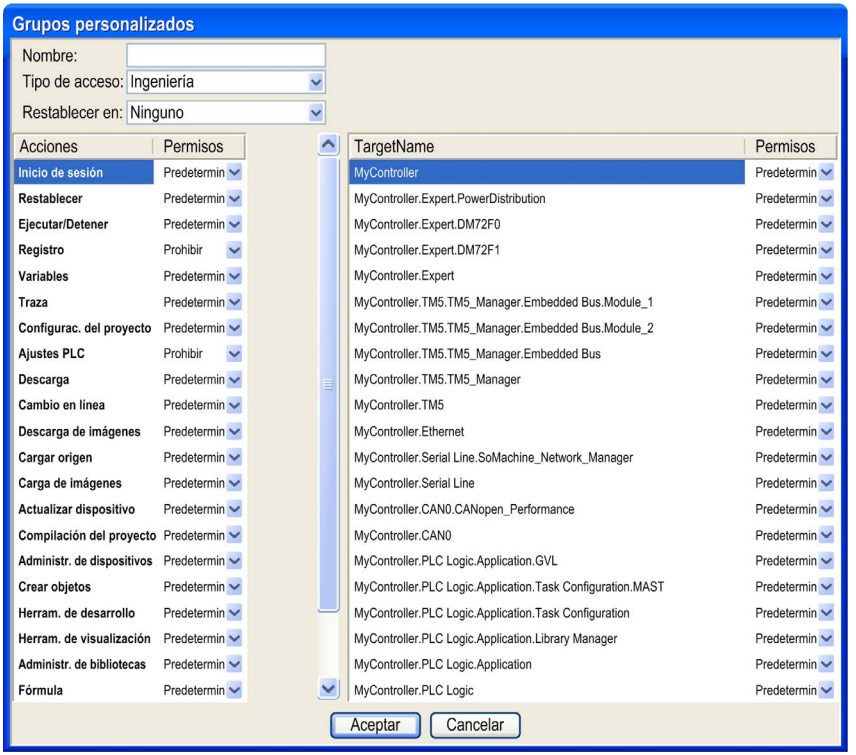
La tabla contiene los grupos de derechos de acceso predeterminados disponibles para cada tipo de **protección de acceso**:

Tipo de protección de acceso	Grupos de derechos de acceso predeterminados	Descripción
Acceso a datos	**Ninguno**	Acceso denegado.
	Lectura	● Acceso a sitio web del dispositivo. ● Lectura de datos no protegidos.
	Lectura-Escritura	● Acceso a sitio web del dispositivo. ● Lectura/escritura de datos no protegidos.

Tipo de protección de acceso	Grupos de derechos de acceso predeterminados	Descripción
Herramientas de ingeniería	**Ninguno**	Inicio de sesión denegado.
	Supervisar	● Inicio de sesión ● Cargar origen ● Variables de sólo lectura
	Operate	● Inicio de sesión ● Cargar origen ● Forzar/escribir variables
	Programa	Sin restricciones, salvo que no se permite la configuración de derechos de usuario.
	Administrate	Sin restricciones, se permite incluso la configuración de derechos de usuario.
Acceso a archivo	**Ninguno**	Acceso denegado.
	Lectura	Leer archivos no protegidos.
	Lectura-Escritura	Leer/escribir archivos no protegidos.
	Lectura/Escritura/Eliminación	Leer/escribir/eliminar archivos no protegidos.

Administración de grupos de derechos de acceso

Además de los grupos de derechos de acceso predeterminados, puede crear sus propios grupos personalizados. Puede configurar derechos de acceso para herramientas o comandos específicos agrupados bajo cada tipo de **protección de acceso** (**Acceso a datos**, **Herramientas de ingeniería**, **Acceso a archivo**). Para llevar esto a cabo, haga clic en **Agregar grupo** o **Editar grupo** en el área **Acciones de grupos**. Se abre el cuadro de diálogo **Grupos personalizados**.



Parámetro	Descripción
Nombre	Escriba un **Nombre** para el grupo utilizando un máximo de 32 caracteres. Se permiten los caracteres a–z, A–Z, 0–9 y el carácter de subrayado.
Tipo de acceso	Seleccione uno de los **tipos de acceso** disponibles para el grupo: ● **Acceso a datos** ● **Herramientas de ingeniería** ● **Acceso a archivo** Las **acciones** y los **permisos** disponibles para el **tipo de acceso** seleccionado se enumeran en la parte izquierda.

Parámetro	Descripción
Restablecer en	Para copiar los derechos de acceso de usuarios de un grupo a otro, seleccione el grupo de la lista **Restablecer en**. De este modo se restablecen las **acciones** y los **permisos** a los valores del grupo que ha seleccionado.
Lista **Acciones** / **Permisos**	Esta lista muestra las **acciones** y los **permisos** disponibles para el **tipo de acceso** seleccionado. Consulte abajo la lista de acciones y permisos para cada tipo de acceso.
Lista **TargetName** / **Permisos**	Los **TargetName** de esta lista son exactamente los que encuentra en el árbol **Dispositivos** de su proyecto para el dispositivo que ha seleccionado. Puede configurar los permisos para cada dispositivo de destino individualmente. Para ello, seleccione el dispositivo en la lista y seleccione la opción adecuada (**Predeterminado**, **Prohibir**, **Sólo lectura**, **Se puede modificar**, **Completo**) en la lista **Permisos**. El valor **Predeterminado** corresponde a los permisos del usuario **Everyone**.

La tabla lista las **acciones** y los **permisos** disponibles que puede asignar para el **tipo de acceso** →**Acceso a datos**:

Acciones disponibles	Permisos
Servidor OPC Permite al usuario conectarse a un dispositivo en el servidor OPC utilizando parámetros de inicio de sesión/contraseña.	● **Prohibir** ● **Lectura** ● **Lectura/Escritura**
HMI Permite al usuario acceder a Vijeo-Designer utilizando los parámetros de inicio de sesión/contraseña.	● **Prohibir** ● **Lectura** ● **Lectura/Escritura**
Servidor web Permite al usuario acceder al servidor web utilizando los parámetros de inicio de sesión/contraseña.	● **Prohibir** El usuario no puede acceder al servidor web. ● **Lectura** El usuario puede acceder al servidor web y leer variables. ● **Lectura/Escritura** El usuario puede acceder al servidor web y leer y escribir variables.

La tabla enumera las **acciones** y los **permisos** disponibles que puede asignar para el **tipo de acceso** →**Herramientas de ingeniería**:

Acciones disponibles	Permisos
Inicio de sesión Permite al usuario conectarse al dispositivo y acceder a la aplicación. Este permiso es un requisito previo para poder otorgar cualquier otro permiso online.	● **Predeterminado** ● **Prohibir** ● **Permitir**
Restablecer	● **Predeterminado** Los mismos permisos que el usuario **Everyone**. ● **Prohibir** El usuario no puede ejecutar un restablecimiento del dispositivo. ● **Frío** El usuario puede ejecutar un reinicio en frío del dispositivo. ● **Caliente/Frío** El usuario puede ejecutar un reinicio en caliente y en frío del dispositivo.
Ejecutar/Detener	● **Predeterminado** Los mismos permisos que el usuario **Everyone**. ● **Prohibir** El usuario no puede ejecutar/detener la aplicación. ● **Permitir** El usuario puede ejecutar/detener la aplicación.
Registro Permite al usuario acceder a la vista **Registro** del editor de dispositivos.	● **Predeterminado** Los mismos permisos que el usuario **Everyone**. ● **Prohibir** ● **Permitir**
Variables	● **Predeterminado** Los mismos permisos que el usuario **Everyone**. ● **Prohibir** El usuario no puede leer/escribir variables. ● **Lectura** El usuario puede leer las variables de la aplicación. ● **Lectura/Escritura** El usuario puede leer, escribir y forzar variables no protegidas.
Traza Permite al usuario acceder al dispositivo de traza.	● **Predeterminado** Los mismos permisos que el usuario **Everyone**. ● **Prohibir** ● **Permitir**

Acciones disponibles	Permisos
Configuración del proyecto Permite al usuario modificar (leer y escribir) la información y los parámetros del proyecto.	● **Predeterminado** Los mismos permisos que el usuario **Everyone**. ● **Prohibir** ● **Permitir**
Ajustes PLC Permite al usuario acceder a la vista **Ajustes PLC** del editor de dispositivos.	● **Predeterminado** Los mismos permisos que el usuario **Everyone**. ● **Prohibir** ● **Permitir**
Descargar Permite al usuario descargar todas las aplicaciones o la aplicación activa actualmente en el dispositivo.	● **Predeterminado** Los mismos permisos que el usuario **Everyone**. ● **Prohibir** ● **Permitir**
Cambio en línea Permite al usuario ejecutar el comando **Cambio en línea**.	● **Predeterminado** Los mismos permisos que el usuario **Everyone**. ● **Prohibir** ● **Permitir**
Descarga de imágenes Permite al usuario descargar imágenes.	● **Predeterminado** Los mismos permisos que el usuario **Everyone**. ● **Prohibir** ● **Permitir**
Cargar origen Permite al usuario ejecutar el comando **Carga de código de origen**.	● **Predeterminado** Los mismos permisos que el usuario **Everyone**. ● **Prohibir** ● **Cambiar**
Carga de imágenes Permite al usuario cargar imágenes.	● **Predeterminado** Los mismos permisos que el usuario **Everyone**. ● **Prohibir** ● **Cambiar**
Actualizar dispositivo Permite al usuario ejecutar el comando **Actualizar dispositivo**.	● **Predeterminado** ● **Prohibir** ● **Cambiar**
Compilación del proyecto Permite al usuario ejecutar el comando **Compilar** y **Generar todo**.	● **Predeterminado** Los mismos permisos que el usuario **Everyone**. ● **Prohibir** ● **Cambiar**
Administración de dispositivos Permite al usuario agregar, editar, eliminar, actualizar y renovar un dispositivo.	● **Predeterminado** Los mismos permisos que el usuario **Everyone**. ● **Prohibir** ● **Cambiar**
Crear objetos Permite al usuario crear objetos en un proyecto.	● **Predeterminado** Los mismos permisos que el usuario **Everyone**. ● **Prohibir** ● **Cambiar**

Acciones disponibles	Permisos
Herramientas de desarrollo Permite al usuario modificar aplicaciones, POU y dispositivos.	● **Predeterminado** Los mismos permisos que el usuario **Everyone**. ● **Prohibir** ● **Cambiar**
Herramientas de visualización Permite al usuario añadir y editar una visualización web y crear visualizaciones en el proyecto actual.	● **Predeterminado** Los mismos permisos que el usuario **Everyone**. ● **Prohibir** ● **Cambiar**
Administración de bibliotecas Permite al usuario realizar la administración de bibliotecas, excepto acceder a la lista de bibliotecas de un proyecto y mostrar sus propiedades.	● **Predeterminado** Los mismos permisos que el usuario **Everyone**. ● **Prohibir** ● **Cambiar**
Fórmula Permite al usuario crear, editar, enviar, cargar y reproducir una fórmula y cargar datos del programa actual en una fórmula.	● **Predeterminado** Los mismos permisos que el usuario **Everyone**. ● **Prohibir** ● **Cambiar**
Herramientas de depuración Permite al usuario ejecutar comandos de depuración, incluidos los puntos de interrupción.	● **Predeterminado** Los mismos permisos que el usuario **Everyone**. ● **Prohibir** ● **Cambiar**
Herramientas SoftMotion Permite al usuario ejecutar comandos CAM y CNC.	● **Predeterminado** Los mismos permisos que el usuario **Everyone**. ● **Prohibir** ● **Cambiar**
Administración de DTM Permite al usuario ejecutar comandos de la herramienta de dispositivo de campo (FDT).	● **Predeterminado** Los mismos permisos que el usuario **Everyone**. ● **Prohibir** ● **Cambiar**

La tabla indica las **acciones** y los **permisos** disponibles que puede asignar para el **tipo de acceso** →**Acceso a archivo**:

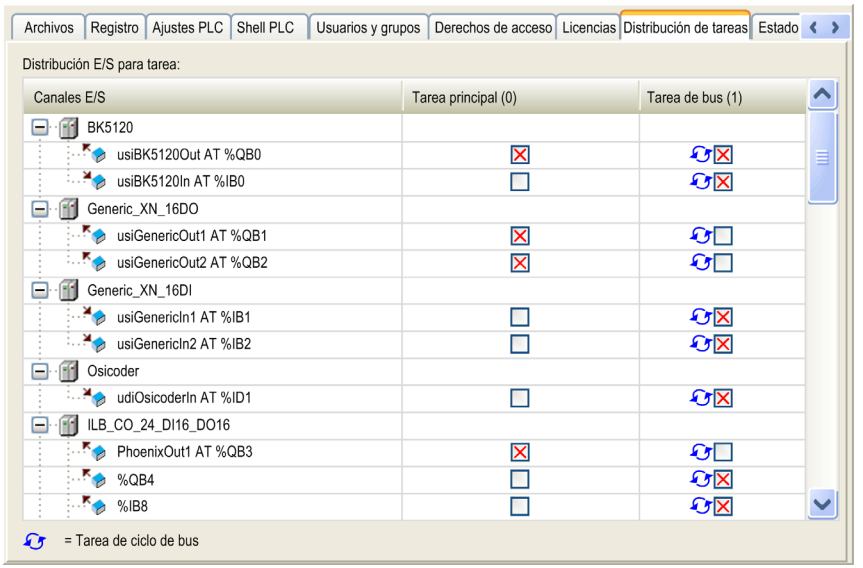
Acciones disponibles	Permisos
Sistema de archivos **FTP** **Removable media** Permite al usuario acceder al archivo mediante FTP, SoMachine o web.	● **Prohibir** ● **Lectura** ● **Lectura/Escritura** ● **Lectura/Escritura/Eliminación**

Distribución de tareas

Descripción general

En la vista **Distribución de tareas** del editor de dispositivos se muestra una tabla con entradas y salidas y su asignación a las tareas definidas. Para que la información se pueda mostrar, el proyecto tiene que estar compilado y el código tiene que estar generado. Esta información resulta de utilidad en la resolución de problemas en el caso de que la misma entrada/salida se actualice en tareas diferentes con prioridades distintas.

Distribución de tareas del editor de dispositivos



En la tabla se muestran las tareas ordenadas por prioridad. Haga clic en los encabezados de columna (nombre de tarea) para mostrar solamente las variables asignadas a esta tarea. Para mostrar todas las variables de nuevo, haga clic en la primera columna (**Canales E/S**).

Para abrir la tabla de asignaciones de E/S de un canal, haga doble clic en la entrada o salida.

Una flecha azul indica la tarea del ciclo de bus.

En el ejemplo anterior, la variable **usiBK5120Out AT %QB0** se utiliza en 2 tareas diferentes. En ese caso, la salida, establecida por una sola tarea, la puede sobrescribir la otra tarea; esto puede provocar que haya un valor no definido. En general, no es recomendable escribir referencias de salida en más de una tarea, ya que provoca que el programa sea difícil de depurar y, con frecuencia, produce resultados imprevistos en el funcionamiento de la máquina o del proceso.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

No escriba en una variable de salida en más de una tarea.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

Estado

Descripción general

La vista **Estado** del editor de dispositivos muestra la información de estado (por ejemplo, **En ejecución**, **Stopped**) y mensajes específicos de diagnóstico del dispositivo correspondiente, así como de la tarjeta usada y del sistema de bus interno.

Información

Descripción general

La vista **Información** del editor de dispositivos muestra información general del dispositivo seleccionado en el árbol **Dispositivos**: **Nombre, Fabricante, Tipo, Número de versión, Número de modelo, Descripción, Imagen**.

Sección 5.2
Asignación de E/S

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Asignación de E/S

Descripción general

La vista **Asignación de E/S** del editor de dispositivos se llama **Asignación de E/S de <tipo de dispositivo>** (por ejemplo, **Asignación de E/S de PROFIBUS DP**). Sirve para configurar una asignación de E/S del controlador. Esto significa que las variables de proyecto utilizadas por la aplicación se asignan a las direcciones de entrada, salida y memoria del controlador.

Defina la aplicación que debe gestionar las E/S en la vista *(véase página 134)* **Ajustes PLC**.

NOTA: Si el dispositivo lo admite, puede utilizar la modalidad de configuración online para acceder a las E/S del hardware sin tener que cargar primero una aplicación. Para obtener más información, consulte la descripción del Modo de configuración en línea *(véase SoMachine, Comandos de menú, Ayuda en línea)*.

Consulte los siguientes capítulos:
- Trabajo con el cuadro de diálogo Asignación de E/S *(véase página 157)*
- Asignación de E/S en modalidad online *(véase página 162)*
- Variables implícitas para forzar E/S *(véase página 163)*

Información general sobre la asignación de E/S en variables

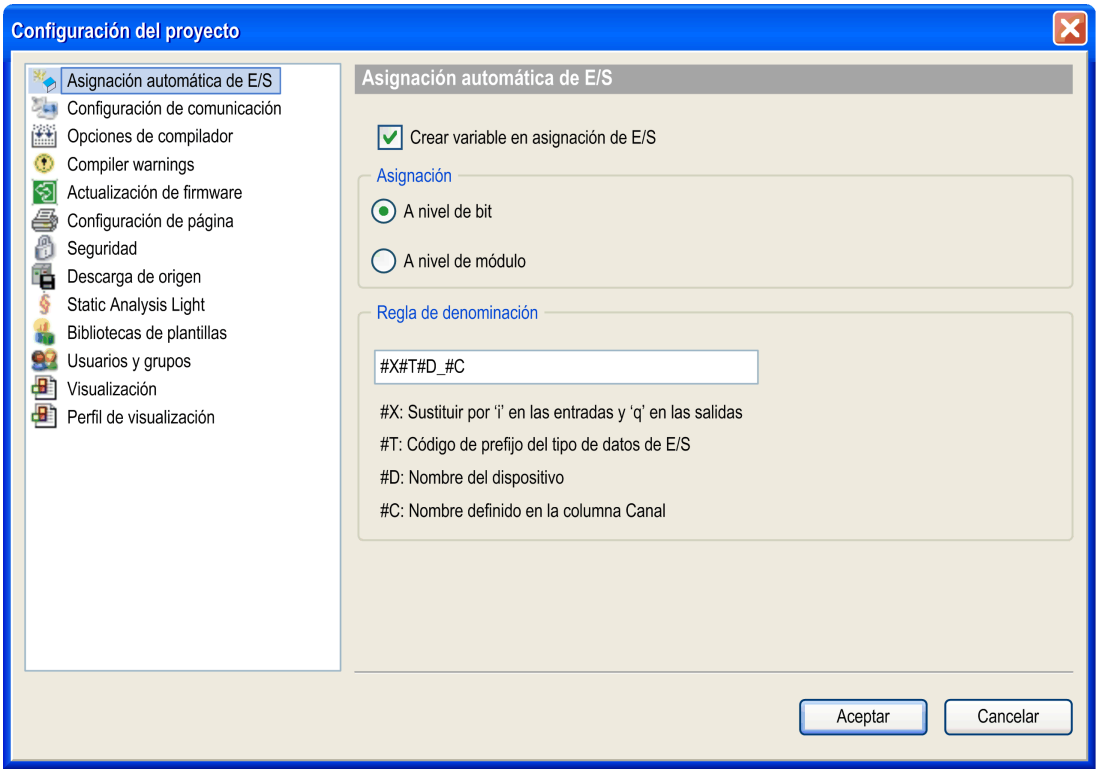
La posibilidad de configurar una asignación de E/S para el dispositivo actual depende del dispositivo. Puede que la vista sólo se utilice para visualizar la instancia de dispositivo creada de forma implícita. Consulte la descripción de los *Objetos IEC (véase página 161)*.

Tenga en cuenta lo siguiente para la asignación de E/S a variables:
- No se puede escribir en las variables que requieren introducción de datos.
- Una variable existente sólo se puede asignar a 1 entrada.
- En lugar de utilizar la vista **Asignación de E/S**, también puede asignar una dirección a una variable mediante la declaración AT *(véase página 586)*.
  No obstante, tenga presente lo siguiente:
  - Puede utilizar declaraciones AT sólo con variables locales o globales, no con variables de entrada y salida de las POU.
  - La posibilidad de generar variables de forzado para E/S (consulte Variables implícitas para forzar E/S *(véase página 163)*) no está disponible para declaraciones AT.
  - Si se utilizan declaraciones AT con miembros de bloque de funciones o estructura, todas las instancias accederán a la misma ubicación de memoria. Esta ubicación de memoria corresponde a variables estáticas en los lenguajes de programación clásicos, como C.
- La disposición en memoria de las estructuras viene determinada por el dispositivo de destino.
- Para cada variable que se asigna a un canal de E/S en la vista **Asignación de E/S**, se pueden crear variables de forzado durante una ejecución de compilación de la aplicación *(véase SoMachine, Comandos de menú, Ayuda en línea)*. Puede utilizarlas para forzar el valor de entrada o salida durante la puesta en marcha de una máquina, por ejemplo, mediante una visualización (HMI). Consulte el capítulo *Variables implícitas para forzar E/S (véase página 163)*.

Asignación automática de E/S

La versión 4.0 y posteriores deSoMachine proporcionan una función de asignación automática de E/S. Esta función crea automáticamente variables IEC en cuanto se añade un dispositivo o módulo con módulos de E/S al árbol **Dispositivos** y las asigna en cada entrada o salida. De forma predeterminada, esta función está activada.

Puede desactivar y configurar la función en el cuadro de diálogo **Proyecto →Configuración del proyecto →Asignación automática de E/S**.



El cuadro de diálogo proporciona los siguientes elementos:

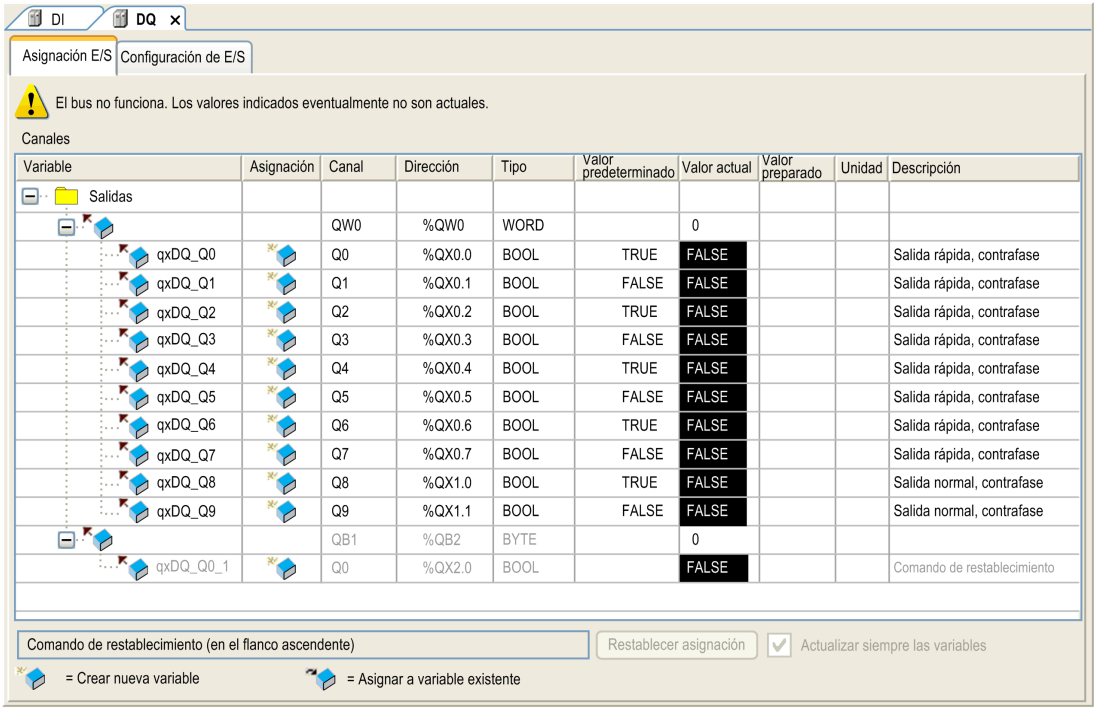
Elemento	Descripción
Crear variable en asignación de E/S	Seleccione o deseleccione esta opción para activar o desactivar la función de asignación automática de E/S.
Área **Asignación**	
A nivel de bit	Seleccione esta opción para crear variables para cada bit.

Elemento	Descripción
A nivel de módulo	Seleccione esta opción para crear una variable para cada módulo, no para los bits a nivel individual.
Área **Regla de denominación**	
Cuadro de texto	Escriba los siguientes caracteres precedidos por un símbolo `#` para especificar de qué partes constará el nombre de variable: ● Escriba `#X` para integrar una `i` para las entradas y una `q` para las salidas en el nombre de variable. ● Escriba `#T` para integrar el código de prefijo para el tipo de datos correspondiente de la variable en el nombre de variable. Los prefijos utilizados para los distintos tipos de datos se indican en el capítulo *(véase página 578) Recomendaciones sobre la denominación de identificadores*. ● Escriba `#D` para integrar el nombre del dispositivo en el nombre de variable. ● Escriba `#C` para integrar el nombre según se ha definido en la columna **Canal** en el nombre de variable.

Trabajo con el cuadro de diálogo Asignación de E/S

Descripción general

A continuación, se muestra una ilustración de la ficha **Asignación E/S** del editor de dispositivos.



Descripción de los elementos del área de canales

La ficha **Asignación E/S** ofrece los siguientes elementos del área de **Canales** si los proporciona el dispositivo:

Elemento	Descripción
Canal	Nombre simbólico del canal de entrada o salida del dispositivo.
Dirección	Dirección del canal, por ejemplo: `%IW0`.
Tipo	Tipo de datos del canal de entrada o salida, por ejemplo: BOOL. Si el tipo de datos no es estándar, sino una estructura o un campo de bits definidos en la descripción del dispositivo, aparecerá en la lista sólo si está incluido en la norma IEC. Se indica como tipo IEC en la descripción del dispositivo. De lo contrario, la entrada de la tabla estará vacía.
Unidad	Unidad del valor del parámetro, por ejemplo: **ms** para milisegundos.
Descripción	Breve descripción del parámetro.
Valor actual	Valor actual del parámetro, se muestra en modalidad online.

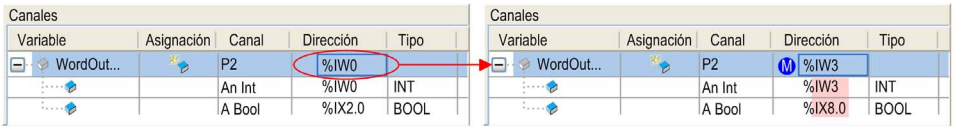
Modificación y fijación de las direcciones

Puede modificar y fijar la dirección que se muestra actualmente de una salida o entrada aquí, en esta ficha. Utilice esta opción para adaptar el direccionamiento a una configuración de hardware determinada o para mantener el valor de la dirección incluso si se cambia el orden de los módulos. De forma predeterminada, esto causaría una adaptación automática de los valores de la dirección.

Tenga en cuenta que en función de la descripción del dispositivo, sólo se puede modificar la dirección de toda la entrada o salida, pero no la de sus subelementos (canales de bits). Por lo tanto, si una entrada o salida está representada en la tabla de asignación con cualquier subárbol, sólo puede editar el campo de dirección de la entrada superior (vea la siguiente figura: sólo se puede abrir el campo de dirección en la primera línea).

Para fijar el valor de la dirección, seleccione la entrada en la columna **Dirección** y pulse la barra espaciadora para abrir el campo de edición. Modifique el valor o déjelo sin modificar y cierre el campo de edición por medio de la tecla RETORNO. El campo de dirección está marcado con un símbolo **M** que indica que se fija el valor actual.

Si el valor se ha modificado, las sucesivas direcciones (hasta la siguiente dirección fija) se adaptarán correspondientemente:



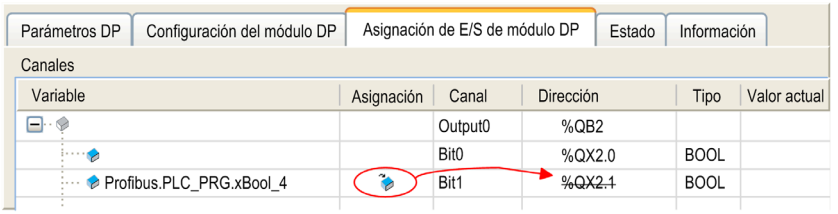
Si desea eliminar la fijación del valor, vuelva a abrir el campo de edición de la dirección, elimine la entrada de dirección y cierre con INTRO. La dirección y las direcciones sucesivas identificadas se establecerán de nuevo a los valores que tenían antes de la modificación manual. El símbolo **M** se eliminará.

Configuración de la asignación de E/S

Realice la asignación de E/S asignando las variables del proyecto apropiadas a los canales de entrada y salida del dispositivo, cada uno en la columna **Variable**.

- El tipo de canal ya está indicado en la columna **Variable** por un símbolo: para la entrada y para la salida. En esta línea, escriba el nombre o la ruta de la variable a la que se debe asignar el canal. Puede asignar una variable de proyecto existente o definir una nueva variable, que luego será automáticamente declarada como una variable global.
- Cuando se asignan variables estructuradas, el editor evitará la introducción de la variable de estructura (por ejemplo, en `%QB0`) conjuntamente con la introducción de los elementos de estructura específicos, en este caso en `%QB0.1` y `QB0.2`).
  Esto significa: cuando hay una entrada principal con un subárbol de entradas de canales de bits en la tabla de asignación (como se muestra en la figura siguiente), entonces se puede introducir una variable en la línea de la entrada principal, o en las de los subelementos (canales de bits), pero nunca en ambas.
- Para la asignación en una variable existente, especifique la ruta completa. Por ejemplo: <nombre de aplicación>.<ruta de POU>.<nombre de variable>';
  Ejemplo: `app1.plc_prg.ivar`
  Para este propósito, puede ser útil abrir el asistente Accesibilidad por medio del botón **...** En la columna **Asignación**, se mostrará el símbolo y el valor de la dirección aparecerá tachado. Esto no significa que esta dirección de memoria ya no exista más. Sin embargo, no se utiliza directamente porque el valor de la variable existente se gestiona en otra ubicación de la memoria y, especialmente en el caso de salidas, ninguna otra variable existente se debe guardar en esta dirección (`%Qxx` en la asignación de E/S) con el fin de evitar ambigüedades durante la escritura de los valores.

Observe en el siguiente ejemplo una asignación de salida de la variable existente `xBool_4`:



- Si desea definir una nueva variable, escriba el nombre de la variable deseada.
  Ejemplo: `bVar1`

  En este caso, el símbolo se insertará en la columna **Asignación** y la variable se declarará internamente como una variable global. A partir de este momento, la variable estará disponible globalmente dentro de la aplicación. El cuadro de diálogo de asignación es otro lugar para la declaración de variables globales.

NOTA: Alternativamente, una dirección también se puede leer o escribir dentro de un código de programa, como en ST (texto estructurado).

- Teniendo en cuenta la posibilidad de realizar cambios en la configuración del dispositivo, se recomienda hacer las asignaciones dentro del cuadro de diálogo de configuración de dispositivo.

NOTA: Si una UNIÓN está representada por canales de E/S en el cuadro de diálogo de asignación, depende del dispositivo el que el elemento raíz sea asignable o no.

Si una variable declarada de un tipo de datos determinado es mayor que aquella a la que se está asignando, el valor de la variable que se está asignando se truncará al tamaño de la variable de destino asignada.

Por ejemplo, si la variable se declara como un tipo de datos WORD y se asigna a un BYTE, sólo 8 bits de la palabra se asignarán al byte.

Esto implica que, para la supervisión del valor en el cuadro de diálogo de asignación, el valor que se muestra en el elemento raíz de la dirección será el valor de la variable declarada, ya que es actualmente válida en el proyecto. En los subelementos debajo de la raíz, se supervisarán los valores de los elementos específicos de la variable asignada. Sin embargo, sólo una parte del valor declarado puede aparecer entre los subelementos.

Existe otra implicación cuando se asigna una variable declarada a las salidas físicas. Igual que en el caso anterior, si asigna un tipo de datos mayor que el tipo de datos de salida, el tipo de datos de salida puede recibir un valor truncado tal que pueda afectar a su aplicación en formas imprevistas.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

Compruebe que el tipo de datos declarado que se está asignando a la E/S física es compatible con el funcionamiento previsto de su máquina.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

Elemento	Descripción
Restablecer asignación	Haga clic en este botón para restablecer los ajustes de asignación a los valores predeterminados definidos por el archivo de descripción del dispositivo.
Actualizar siempre las variables	Si esta opción está activada, las variables se actualizarán en cada ciclo de la tarea de ciclo de bus *(véase página 134)*, sin importar que se utilicen o se asignan a una entrada o salida.

Objetos IEC

Esta parte de la ficha sólo está disponible si se ha creado implícitamente una instancia del objeto de dispositivo, a la que se puede acceder mediante la aplicación (por ejemplo, con el fin de reiniciar un bus o para consultar información). La disponibilidad y utilización de una instancia de este tipo depende del controlador y se describe en su guía de programación.

Opciones de ciclo de bus

Esta opción de configuración estará disponible para dispositivos con llamadas cíclicas antes y después de la lectura de las entradas o salidas. Esto le permite establecer una tarea de ciclo de bus *(véase página 134)* específica del dispositivo.

De forma predeterminada, será válida la configuración del ciclo de bus de orden superior (**Emplear configuración de ciclo del bus de orden superior**). Esto significa que se buscará en el árbol **Dispositivos** la próxima definición válida de la tarea de ciclo de bus.

Para asignar una tarea específica de ciclo de bus, seleccione la que desee en la lista de selección. En la lista se detallan las tareas definidas actualmente en la configuración de la tarea de la aplicación.

Asignación de E/S en modalidad online

Asignación de E/S en modalidad online

Asignación de E/S en modalidad online

Si una variable de estructura se asigna al elemento raíz de la dirección (el que ocupa la posición más alta en el árbol de la dirección respectiva del cuadro de diálogo de asignación), en la modalidad online no se mostrará ningún valor en esta línea. Sin embargo, si, por ejemplo, se asigna una variable DWORD a esta dirección, los valores respectivos se supervisarán en la línea raíz así como en las líneas del canal de bits con sangría debajo. Básicamente, el campo de la línea raíz permanece vacío si el valor está formado por varios subelementos.

Variables implícitas para forzar E/S

Descripción general

Durante la puesta en marcha de una planta o una máquina, puede ser necesario forzar E/S; por ejemplo, mediante una visualización de HMI. Para este fin, puede generar variables de forzado especiales para cada canal de E/S que se asigna en una variable en la ficha **Asignación E/S** del editor de dispositivos.

Como condición previa, el valor **Variables de forzado para la asignación E/S** debe estar activado en la ficha **Ajustes PLC**. A continuación, con cada ejecución compilada de la aplicación se generarán 2 variables por cada canal de E/S asignado de acuerdo con la sintaxis siguiente. Los espacios vacíos en el nombre del canal se sustituirán por caracteres de subrayado.

<nombre de dispositivo>_<nombre de canal>_<dirección IEC>_Force de tipo BOOL, para activar y desactivar el forzado

<nombre de dispositivo>_<nombre de canal>_<dirección IEC>_Value de tipo de datos del canal, para definir el valor que debe forzarse en el canal

Estas variables están disponibles en el asistente Accesibilidad en la categoría **Variables** → **IoConfig_Globals_Force_Variables**. Pueden utilizarse en objetos de programación, en visualizaciones, configuración de símbolos, etc., dentro del sistema de programación.

Un flanco ascendente en la variable de forzado activa el forzado de la E/S respectiva con el valor definido por la variable del valor. Un flanco descendente desactiva el forzado. Es necesario desactivar esto volviendo a establecer la variable de forzado como falsa antes de poder forzar un nuevo valor.

Tenga en cuenta las restricciones siguientes.

Ejemplo

Si la asignación se completa como se muestra en la figura, en la ficha **Asignación E/S** del editor de dispositivos *(véase página 157)*, y se compila (F11) la aplicación, se generarán las siguientes variables, que estarán disponibles en el asistente Accesibilidad:

- `Digitax_ST_Control_word_QW0_Force : BOOL;`
- `Digitax_ST_Control_word_QW0_Value : UINT;`
- `Digitax_ST_Target_position_QD1_Force : BOOL;`
- `Digitax_ST_Target_position_QD1_Value : DINT;`
- `Digitax_ST_Status_word_IW0_Force : BOOL;`
- `Digitax_ST_Status_word_IW0_Value : UINT;`
- `Digitax_ST_Position_actual_value_ID1_Force : BOOL;`
- `Digitax_ST_Position_actual_value ID1_Value : DINT;`

Restricciones

- Sólo los canales asignados en una variable en la ficha **Asignación E/S** (es decir, debe definirse una variable en la columna **Variable** independientemente de si es nueva o existente) pueden forzarse mediante las variables implícitas descritas anteriormente.
- Las entradas/salidas sin usar, así como aquellas asignadas mediante una declaración AT en un programa de aplicación, no pueden forzarse.
- Los canales de E/S respectivos tienen que estar en uso por lo menos en una tarea.
- Las E/S forzadas no se indican en la supervisión (vista de supervisión, cuadro de diálogo Asignación de E/S). El valor sólo se utiliza implícitamente en el controlador de E/S para escribir en el dispositivo.
- Las entradas forzadas se visualizan correctamente mediante el símbolo rojo de forzado (**F**); no así, en cambio, para entradas/salidas forzadas.

Parte III
Programa

Contenido de esta parte

Esta parte contiene los siguientes capítulos:

Capítulo 6
Componentes de programa

Contenido de este capítulo

Este capítulo contiene las siguientes secciones:

Sección 6.1
Unidad de organización de programa (POU)

Contenido de esta sección

Esta sección contiene los siguientes apartados:

POU

Descripción general

El término 'unidad de organización de programa' (POU) se utiliza para todos los objetos de programación (programas, bloques de funciones, funciones, etc.) que se utilizan para crear una aplicación de controlador.

Gestión de POU

Las POU que se gestionan en el nodo **Global** del árbol **Aplicaciones** no son específicas del dispositivo, pero se pueden instanciar para su uso en un dispositivo (aplicación). Para ello, las POU de programa deben invocarse mediante una tarea de la aplicación respectiva.

Las POU que se insertan en el árbol **Aplicaciones** explícitamente debajo de una aplicación sólo las pueden instanciar aplicaciones con sangría debajo de esa aplicación (aplicación secundaria). Para obtener más información, consulte las descripciones del árbol **Dispositivos** *(véase página 37)* y del objeto *(véase página 244)* **Aplicación**.

Pero POU también es el nombre de una subcategoría determinada de esos objetos en el menú **Agregar objeto**. En ese caso, sólo incluye programas, bloques de funciones y funciones.

Por tanto, en general, un objeto POU es una unidad de programación. Es un objeto que se gestiona de forma independiente del dispositivo en el nodo **Global** del árbol **Aplicaciones** o directamente debajo de una aplicación en el árbol **Aplicaciones**. Puede visualizarse y editarse en una vista de editor. Un objeto POU puede ser un programa, una función o un bloque de funciones.

Se pueden configurar determinadas **Propiedades** (como condiciones de compilación, etc.) para cada objeto POU concreto.

Puede consultar una descripción sobre cómo crear una POU en la sección *Adición de POU a una aplicación (véase página 171)*. Las POU que haya creado se añaden a la vista **Activos** del **Catálogo de software**.

Hay dos maneras de añadir una POU disponible en la vista **Activos** al proyecto:
- Seleccione una POU en la vista **Activos** y arrástrela al nodo pertinente del árbol **Aplicaciones**.
- Seleccione una POU en la vista **Activos** y arrástrela a la vista del editor de lógica *(véase página 277)*.

Además de los objetos POU, hay objetos de dispositivo que se utilizan para ejecutar el programa en el sistema de destino (**Recurso**, **Aplicación**, **Configuración de tareas**, etc.). Se gestionan en el árbol **Aplicaciones**.

Adición y llamadas a POU

Introducción

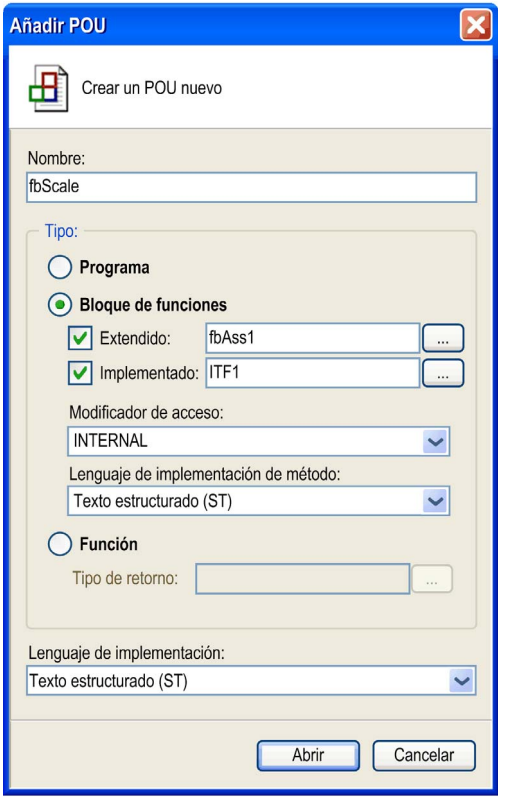
Puede agregar unidades de organización de programa (POU) a su aplicación en **Catálogo de software** →**Activos** o en el árbol **Aplicaciones**.

Estos son los distintos tipos de POU:

- **Programa**: devuelve uno o varios valores durante el funcionamiento. Se conservan todos los valores desde la última vez en que se ejecutó el programa hasta la siguiente. La puede llamar otra POU.
- **Bloque de funciones**: proporciona uno o varios valores durante el procesamiento de un programa. A diferencia de una función, los valores de las variables de salida y las variables internas necesarias pueden persistir de una ejecución del bloque de funciones a la siguiente. Por lo tanto, la invocación de un bloque de funciones con los mismos argumentos (parámetros de entrada) no siempre tiene por qué producir los mismos valores de salida.
- **Función**: produce exactamente un elemento de datos (que puede estar formado por varios elementos, como campos o estructuras) cuando se procesa. La llamada en lenguajes textuales puede representarse como un operador en las expresiones.

Adición de POU a una aplicación

Para añadir una POU a la aplicación del controlador, haga lo siguiente:

Paso	Acción
1	En la sección **Catálogo de software** →**Activos** →**POU**, seleccione un nodo de **Aplicación**, haga clic en el signo más de color verde y ejecute el comando **POU...** Como alternativa, puede hacer clic con el botón derecho en el nodo **Aplicación** del controlador y seleccionar **Agregar objeto** →**POU**. Los 2 métodos también están disponibles en **Aplicaciones** **Resultado:** Se abre el cuadro de diálogo **Añadir POU**. 
2	En el cuadro de diálogo **Añadir POU**, asigne un nombre a su POU escribiendo un nombre en el campo de texto **Nombre**. **NOTA:** El nombre no puede contener espacios. Si no introduce ningún nombre, se le asignará uno predeterminado. Asignar un nombre significativo a una POU puede facilitar la organización del proyecto.

Paso	Acción
3	Seleccione el tipo de POU que desee: • **Programa** • **Bloque de funciones**: **a.** Seleccione **Extendido** y haga clic en el navegador para seleccionar la función de bloque que desee en **Accesibilidad**. **b.** Haga clic en el botón **Aceptar**. **c.** Seleccione **Implementado** y haga clic en el navegador para seleccionar la interfaz que desee en **Accesibilidad**. **d.** Haga clic en el botón **Aceptar**. **e.** En el cuadro de lista **Lenguaje de implementación de método**, seleccione el lenguaje de programación que desee para editar el bloque de funciones. • **Función**: **a.** Haga clic en el botón del navegador para seleccionar el **tipo de retorno** que desee en **Accesibilidad**. **b.** Haga clic en el botón **Aceptar**.
4	En el cuadro de lista **Lenguaje de implementación**, seleccione el lenguaje de programación que desee para editar el programa.
5	Haga clic en el botón **Abrir**.

Las POU que ya están definidas se muestran en la sección **Catálogo de software →Activos → POU**. Puede añadirlas a su aplicación arrastrándolas a **Aplicaciones** y soltándolas en un nodo **Aplicación**. También puede soltar una POU en la vista del editor de lógica.

Asignación de POU a una tarea

Como mínimo se debe asignar 1 POU a una tarea. Para añadir una POU a una tarea, haga lo siguiente:

Paso	Acción
1	En el nodo **Configuración de tareas** del controlador, haga doble clic en la tarea a la que desee añadir la POU. En la ficha **Configuración**, haga clic en **Agregar llamada**. Como alternativa, en **Aplicaciones** , seleccione el nodo de una tarea en la que desee declarar la POU y haga clic en el signo más de color verde. Ejecute el comando **POU...** de la lista. Haga clic en el botón **...**. **Resultado:** Aparecerá el cuadro de diálogo **Accesibilidad**.
2	En la ficha **Categorías** del cuadro de diálogo **Accesibilidad**, seleccione **Programas (proyecto)**.
3	Haga clic para desactivar la casilla **Vista estructurada**.
4	En el panel **Elementos**, seleccione la POU que desee.
5	Haga clic en el botón **Aceptar**.

Llamada a las POU

Las POU pueden llamar a otras POU. Sin embargo, no se permite la recursión (una POU que se llame a sí misma).

Cuando una POU asignada a una aplicación llame a otra POU simplemente por su nombre (sin ningún espacio de nombres *(véase página 795)* añadido), tenga en cuenta el orden siguiente de examen del proyecto para la POU a la que se llamará:

1.	aplicación actual
2.	**Administrador de bibliotecas** de la aplicación actual en **Herramientas**
3.	Nodo **Global** de **Aplicaciones**
4.	**Administrador de bibliotecas** del nodo **Global** de **Herramientas**

Si una POU con el nombre especificado en la llamada está disponible en una biblioteca del **Administrador de bibliotecas** de la aplicación y también en un objeto del nodo **Global** de **Aplicaciones**, no hay ninguna sintaxis para llamar explícitamente a la POU en el nodo **Global** de **Aplicaciones** utilizando únicamente su nombre. En este caso, mueva la biblioteca correspondiente del **Administrador de bibliotecas** de la aplicación al **Administrador de bibliotecas** del nodo **Global** de **Aplicaciones**. Luego, puede llamar a la POU del nodo **Global** de **Aplicaciones** sólo por su nombre (y, si es necesario, a la de la biblioteca añadiéndole delante el espacio de nombres de la biblioteca).

Consulte también el capítulo *POU para comprobaciones implícitas (véase página 195)*.

Programa

Descripción general

Un programa es una POU que devuelve varios valores durante la operación. Se conservan todos los valores desde la última vez en que se ejecutó el programa hasta la siguiente.

Adición de un programa

Para asignar un programa a una aplicación existente, seleccione el nodo de la aplicación en el árbol **Aplicaciones**, haga clic en el signo más de color verde y ejecute el comando **POU...** Como alternativa, haga clic con el botón derecho del ratón en el nodo **Aplicación** y ejecute el comando **Agregar objeto** →**POU** en el menú contextual. Para añadir una POU independiente de la aplicación, seleccione el nodo **Global** del árbol **Aplicaciones** y ejecute los mismos comandos.

En el cuadro de diálogo **Añadir POU**, seleccione la opción **Programa**, especifique un nombre para el programa y seleccione el lenguaje de implementación. Haga clic en **Abrir** para confirmar. La vista de editor del nuevo programa se abrirá y podrá empezar a editar el programa.

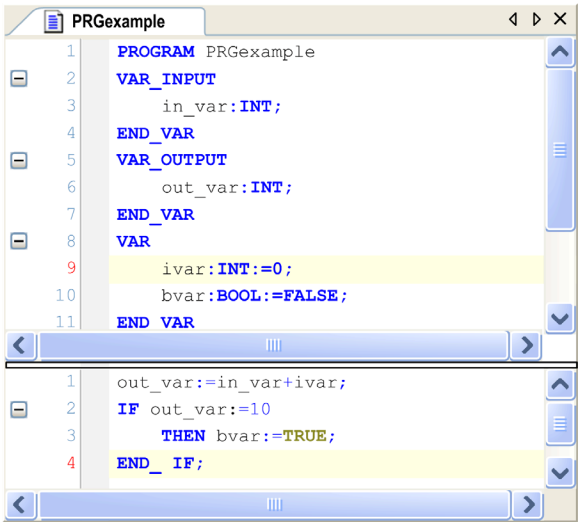
Declaración de un programa

Sintaxis:

PROGRAM <nombre de programa>

Va seguido de las declaraciones de variables de entrada *(véase página 592)*, salida *(véase página 593)* y las variables de programa. Las variables de acceso están también disponibles como opciones.

Ejemplo de un programa



Llamada a un programa

Otra POU puede llamar a un programa. Sin embargo, no se permite una llamada de programa en una función *(véase página 177)*. No hay instancias de programas.

Si una POU ha llamado a un programa y los valores del programa se han modificado, estos cambios se conservarán hasta que se vuelva a llamar a un programa. Esto se aplica aunque se llame desde otra POU. Tenga en cuenta que esto es diferente de la llamada a un bloque de funciones. Cuando se llama a un bloque de funciones, sólo se cambian los valores en la instancia en cuestión del bloque de funciones. Los cambios sólo entran en vigor cuando se vuelve a llamar a la instancia.

Con el fin de establecer parámetros de entrada o salida en el transcurso de una llamada de programa, en los editores de texto de lenguaje (por ejemplo, ST), asigne valores a los parámetros después del nombre de programa entre paréntesis. Para los parámetros de entrada, utilice `:=` para esta asignación, como con la inicialización de variables *(véase página 583)* en la posición de declaración. Para los parámetros de salida, utilice `=>`. Consulte el ejemplo siguiente.

Si el programa se inserta mediante **Accesibilidad** utilizando la opción **Insertar con argumentos** en la vista de implementación de un editor de lenguaje de texto, se visualizará automáticamente según esta sintaxis con todos los parámetros, aunque no es necesario que asigne estos parámetros.

Ejemplo para llamadas de programa

Programa en IL:

```
CAL               PRGexample              (
         in_var:= 33                      )
LD                PRGexample.out_var
ST                erg
```

Ejemplo con asignación de parámetros (**Accesibilidad** utilizando la opción **Insertar con argumentos**):

Programa en IL con argumentos:

```
CAL               PRGexample              (
         in_var:= 33                      ,
        out_var=> erg                     )
```

Ejemplo en ST

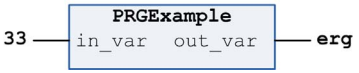
```
PRGexample(in_var:= 33);
erg := PRGexample.out_var;
```

Ejemplo con asignación de parámetros (**Accesibilidad** utilizando la opción **Insertar con argumentos**, como se ha descrito anteriormente):

```
PRGexample (in_var:=33, out_var=>erg );
```

Ejemplo en FBD

Programa en FBD:



Función

Descripción general

Una función es una POU que produce exactamente un elemento de datos (que puede estar formado por varios elementos, como campos o estructuras) cuando se procesa. La llamada en lenguajes textuales puede ocurrir como un operador en expresiones.

Adición de una función

Para asignar la función a una aplicación existente, seleccione el nodo de la aplicación en el árbol **Aplicaciones**, haga clic en el signo más de color verde y ejecute el comando **POU...** Como alternativa, haga clic con el botón derecho del ratón en el nodo **Aplicación** y ejecute el comando **Agregar objeto →POU** en el menú contextual. Para añadir una POU independiente de la aplicación, seleccione el nodo **Global** del árbol **Aplicaciones** y ejecute los mismos comandos.

En el cuadro de diálogo **Añadir POU**, seleccione la opción **Función**. Especifique un **Nombre** (<nombre de la función:>) y un **Tipo de retorno** (<tipo de datos>) para la nueva función y seleccione el lenguaje de implementación deseado. Para elegir el tipo de datos de retorno, haga clic en **...** para abrir el cuadro de diálogo **Accesibilidad**. Haga clic en **Abrir** para confirmar. Se abre la vista de editor para la nueva función y puede empezar la edición.

Declaración de una función

Sintaxis:

FUNCTION <nombre de función> : <tipo de datos>

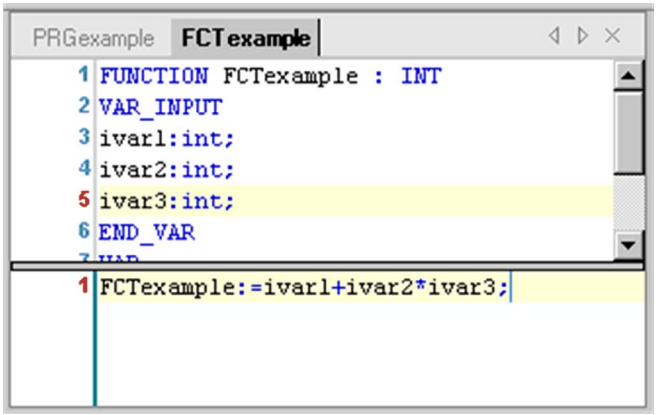
Va seguido de las declaraciones de variables de entrada y las variables de programa.

Asigne un resultado a una función. Esto significa que el nombre de la función se utiliza como variable de salida.

No declare variables locales como `RETAIN` o `PERSISTENT` en una función porque no tendrá efecto.

El compilador genera los mensajes apropiados si se detectan variables locales declaradas como `RETAIN` o `PERSISTENT`.

Ejemplo de una función en ST: esta función toma 3 variables de entrada y devuelve el producto de las 2 últimas añadidas a la primera.



Llamada a una función

La llamada de una función en ST puede aparecer como un operando en expresiones.

En IL, puede colocar una llamada de función sólo dentro de acciones de un paso o en una transición.

Las funciones (a diferencia de un programa o un bloque de funciones) no contienen información interna de estado; es decir, la invocación de una función con los mismos argumentos (parámetros de entrada) siempre produce los mismos valores (salida). Por este motivo, las funciones no pueden contener variables globales y direcciones.

Ejemplo de llamadas a funciones en IL

Llamadas a funciones en IL;

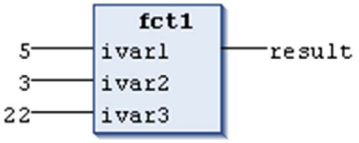
```
LD              5
Fct             3               ,
                22
ST              result
```

Ejemplo de llamadas a funciones en ST

```
result := fct1(5,3,22);
```

Ejemplo de llamadas a funciones en FBD

Llamadas a funciones en FBD:



Ejemplo:

```
fun(formal1 := actual1, actual2); // -> error message
fun(formal2 := actual2, formal1 := actual1); // same semantics as the f
ollowing:
fun(formal1 := actual1, formal2 := actual2);
```

De acuerdo con el estándar IEC 61131-3, las funciones pueden tener salidas adicionales. Deben asignarse en la llamada de la función. En ST, por ejemplo, de acuerdo con la sintaxis siguiente:

out1 => <variable salida 1> | out2 => <variable salida 2> | ...otras variables de salida

Ejemplo

La función `fun` se define con 2 variables de entrada, `in1` e `in2`. El valor de retorno de `fun` se graba en las variables de salida *(véase página 593)* definidas localmente (`VAR_OUTPUT`) `loc1` y `loc2`.

```
fun(in1 := 1, in2 := 2, out1 => loc1, out2 => loc2);
```

Método

Descripción general

La funcionalidad **Método** sólo está disponible si se selecciona en el conjunto de características usado actualmente (**Opciones →Características →Conjunto predefinido de características**).

Puede utilizar métodos para describir una secuencia de instrucciones ya que son compatibles con la programación orientada a objetos. A diferencia de una función, un método no es una POU independiente, sino que se debe asignar a un bloque de funciones *(véase página 196)*. Se puede considerar una función que contiene una instancia del bloque de funciones respectivo. Como tal, tiene un valor de retorno, una parte de declaración propia para variables temporales y parámetros.

También, como medio de programación orientada a objetos, puede utilizar interfaces *(véase página 185)* para organizar los métodos disponibles en un proyecto. Una interfaz es una colección de prototipos de métodos. Esto significa que un método asignado a una interfaz sólo contiene una parte de declaración, pero sin implementación. Además, en la declaración no se permiten variables locales y estáticas, sino variables de entrada, salida y entrada/salida. La implementación del método se debe realizar en el bloque de funciones que implementa la interfaz *(véase página 207)* y utiliza el método.

NOTA: Al copiar o desplazar un método o una propiedad de una POU a una interfaz, las implementaciones contenidas se eliminan de forma automática. Al copiar o desplazar de una interfaz a una POU, se solicita especificar el lenguaje de implementación deseado.

Inserción de un método

Para asignar un método a un bloque de funciones o a una interfaz, seleccione el nodo de bloque de funciones o de interfaz adecuado en **Aplicaciones** , haga clic en el signo más de color verde y ejecute el comando **Método**. Como alternativa, puede hacer clic con el botón derecho del ratón en el nodo del bloque de funciones o de la interfaz y ejecutar el comando **Agregar objeto** → **Método** en el menú contextual.

En el cuadro de diálogo **Añadir método**, introduzca un **Nombre**, el **Tipo de retorno** deseado, el **Lenguaje de implementación:** y el **Modificador de acceso** (véase a continuación). Para elegir el tipo de retorno, haga clic en el botón **...** para abrir el cuadro de diálogo **Accesibilidad...**

Modificador de acceso: por motivos de compatibilidad, los modificadores de acceso son opcionales. El modificador **PUBLIC** está presente como equivalente a no establecer ningún modificador.

Como alternativa, seleccione una de las opciones de la lista de selección:
- **PRIVADO**: el acceso en el método se restringe al bloque de funciones.
- **PROTEGIDO**: el acceso en el método se restringe al bloque de funciones y su derivación.
- **INTERNO**: el acceso en el método se restringe al espacio de nombres (biblioteca) actual.
- **FINAL**: no es posible el acceso de sobrescritura en el método. Permite la generación de código optimizado.

NOTA: Los modificadores de acceso son válidos a partir de la versión del compilador 3.4.4.0 y, por lo tanto, se pueden utilizar como modificadores en versiones anteriores. Para obtener más información, consulte la tabla de asignación de versiones del compilador SoMachine/CoDeSys en SoMachineCompatibilidad y migración - Guía del usuario *(véase SoMachine - Compatibilidad y migración, Guía del usuario)*.

Haga clic en **Abrir** para confirmar. Se abrirá la vista del editor de métodos.

Declaración de un método

Sintaxis:

```
METHOD <modificador acceso> <nombre método> : <tipo datos retorno>VAR_INPUT
 ... END_VAR
```

Para obtener una descripción sobre cómo declarar métodos de manejo de interfaces, consulte el capítulo *Interfaz (véase página 185)*.

Llamada a un método

Las llamadas a métodos también se conocen como llamadas a funciones virtuales. Para obtener información adicional, consulte el capítulo *Invocación de métodos (véase página 209)*.

Para llamar a un método, tenga en cuenta lo siguiente:

- Los datos de un método son temporales y sólo son válidos durante la ejecución del método (variables de pila). Esto significa que las variables y los bloques de funciones declarados en un método se reinicializan en cada llamada al método.
- En el cuerpo de un método, se permite el acceso a las variables de instancia del bloque de funciones.
- Si es necesario, utilice el puntero THIS *(véase página 213)*, que siempre apunta a la instancia actual.
- No se puede acceder a las variables `VAR_IN_OUT` o `VAR_TEMP` del bloque de funciones en un método.
- Sólo se permite que los métodos definidos en una interfaz *(véase página 185)* tengan variables de entrada, salida y entrada/salida, pero no cuerpo (parte de la implementación).
- Los métodos como las funciones pueden tener salidas adicionales. Se deben asignar durante la *invocación al método (véase página 209)*.

Métodos especiales para un bloque de funciones

Método	Descripción
`Init`	Un método denominado `FB_init` se declara implícitamente de forma predeterminada, pero también se puede declarar explícitamente. Contiene código de inicialización para el bloque de funciones tal como se ha declarado en la parte de declaración del bloque de funciones. Consulte el método `FB_init` *(véase página 603)*.
`Reinit`	Si se declara un método denominado `FB_reinit` para una instancia de bloque de funciones, se llama después de que se haya copiado la instancia (como durante el **Cambio en línea**) y se reinicializará el módulo de la nueva instancia. Consulte los métodos `FB_init, FB_reinit` *(véase página 603)*.
`Exit`	Si se desea un método de salida denominado `FB_exit`, se debe declarar explícitamente. No hay declaración implícita. Se llama al método `Exit` para cada instancia del bloque de funciones antes de una nueva descarga, un reinicio o durante el cambio en línea para todas las instancias movidas o eliminadas. Consulte el método `FB_exit` *(véase página 606)*.

Las propiedades *(véase página 183)* y las propiedades de interfaz *(véase página 189)* constan de un método de acceso `Set` o `Get`.

Llamada al método también cuando la aplicación está detenida

En el archivo de descripción del dispositivo, se puede definir que siempre se llame a un determinado método con una tarea cíclica por una determinada instancia de bloque de funciones (de un módulo de biblioteca). Si este método tiene los siguientes parámetros de entrada, también se procesa cuando la aplicación activa no se está ejecutando.

Ejemplo

```
VAR_INPUT
pTaskInfo : POINTER TO DWORD;
pApplicationInfo: POINTER TO _IMPLICIT_APPLICATION_INFO;
END_VAR
```

El programador puede comprobar el estado de la aplicación mediante `pApplicationInfo` y definir qué debe suceder.

```
IF pApplicationInfo^.state = RUNNING THEN <instructions> END_IF
```

Propiedad

Descripción general

La funcionalidad **Propiedad** sólo está disponible si se selecciona en el conjunto de características usado actualmente (**Opciones** →**Características** →**Conjunto predefinido de características**).

Está disponible una propiedad en extensión al estándar IEC 61131-3 como método de programación orientada a objetos. Se compone de un par de métodos de acceso (`Get`, `Set`). Se llaman automáticamente en el acceso de lectura o escritura al bloque de funciones que tenga la propiedad.

Para insertar una propiedad como objeto debajo de un programa *(véase página 174)* o un nodo del bloque de funciones *(véase página 196)*, seleccione el nodo en el árbol **Aplicaciones**, haga clic en el signo más de color verde y ejecute el comando **Propiedad**. Como alternativa, haga clic con el botón derecho en el nodo y ejecute el comando **Agregar objeto** →**Propiedad** en el menú contextual.

En el cuadro de diálogo **Add Property**, especifique el **Nombre**, **Tipo de retorno**, **Lenguaje de implementación** deseado y, opcionalmente, un **Modificador de acceso**.

Están disponibles los mismos modificadores de acceso que para los métodos *(véase página 180)*:

- **PUBLIC**
- **PRIVATE**
- **PROTECTED**
- **INTERNAL**
- **FINAL**

NOTA: Las propiedades también se pueden utilizar dentro de las interfaces.

Descriptores de acceso `Get` y `Set` de una propiedad

Se insertan automáticamente 2 métodos *(véase página 180)* especiales, denominados descriptores de acceso, en el árbol **Aplicaciones**, debajo del objeto de propiedad. Puede eliminar uno de ellos si la propiedad sólo se debe utilizar para la escritura o sólo para la lectura. Un descriptor de acceso, al igual que una propiedad (véase el párrafo anterior), puede tener asignado un modificador de acceso en la parte de declaración, o a través del cuadro de diálogo **Añadir POU**, cuando se añade explícitamente el descriptor de acceso.

- Se llama al descriptor de acceso `Set` cuando se escribe la propiedad, es decir, el nombre de la propiedad se utiliza como entrada.
- Se llama al descriptor de acceso `Get` cuando se lee la propiedad, es decir, el nombre de la propiedad se utiliza como salida.

Ejemplo:

El bloque de funciones `FB1` utiliza una variable local `milli`. Esta variable está determinada por las propiedades `Get` y `Set`:

Ejemplo de `Get`

```
seconds := milli / 1000;
```

Ejemplo de `Set`

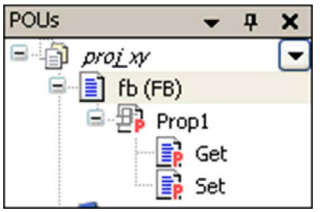
```
milli := seconds * 1000;
```

Puede escribir la propiedad del bloque de funciones (método `Set`); por ejemplo, con `fbinst.seconds := 22;`

(`fbinst` es la instancia de `FB1`).

Puede leer la propiedad del bloque de funciones (método `Get`); por ejemplo, con `testvar := fbinst.seconds;`

En el siguiente ejemplo, la propiedad `Prop1` se asigna al bloque de funciones `fb`:



Una propiedad puede tener variables locales adicionales pero ninguna entrada adicional y, a diferencia de una función *(véase página 177)* o un método *(véase página 180)*, ninguna salida adicional.

NOTA: Al copiar o desplazar un método o una propiedad de una POU a una interfaz, las implementaciones contenidas se eliminan de forma automática. Al copiar o desplazar de una interfaz a una POU, se solicita especificar el lenguaje de implementación deseado.

Supervisión de una propiedad

Una propiedad se puede supervisar en la modalidad online con ayuda de la supervisión en línea *(véase página 392)* o de una lista de supervisión *(véase página 464)*. La condición previa para la supervisión de una propiedad es la adición del pragma `{attribute 'monitoring:=variable'}` (consulte el capítulo Attribute Monitoring *(véase página 637)*) en la parte superior de su definición.

Interfaz

Descripción general

La funcionalidad **Interfaz** sólo está disponible si se selecciona en el conjunto de características usado actualmente (**Opciones** →**Características** →**Conjunto predefinido de características**).

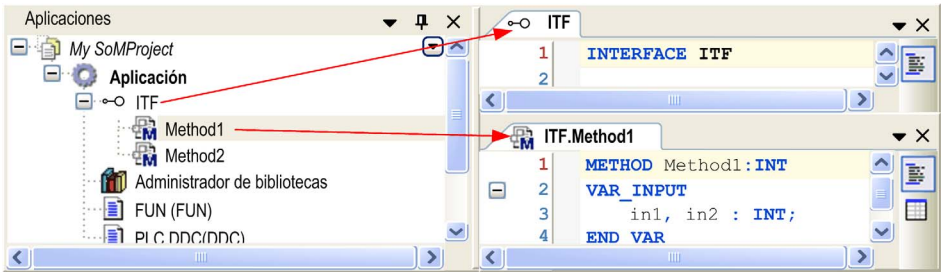
El uso de interfaces es un medio de programación orientada a objetos. Una POU con interfaces describe un conjunto de métodos *(véase página 180)* y prototipos de propiedad *(véase página 183)*. Prototipo significa que sólo contiene declaraciones, pero que no contiene implementaciones. La interfaz podría describirse como la cáscara vacía de un bloque de funciones *(véase página 196)*. Debe implementarse *(véase página 207)* en la declaración del bloque de funciones para que se realice en las instancias del bloque de funciones. Hasta que no forme parte de una definición del bloque de funciones, puede completarse con el código de programación específico del bloque de funciones. Un bloque de funciones puede implementar una o varias interfaces.

Mediante el uso de bloques de funciones diferentes, puede ejecutarse el mismo método con parámetros idénticos pero con código de implementación diferente. Por tanto, se puede utilizar/llamar a una interfaz en cualquier POU sin la necesidad de que la POU identifique el bloque de funciones concreto que está relacionado.

Ejemplo de definición y uso de interfaz en un bloque de funciones

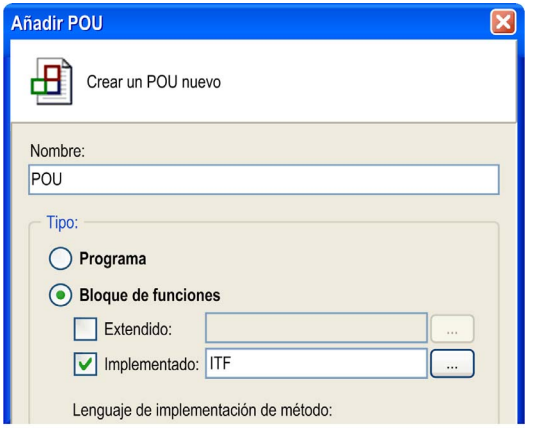
Una interfaz `IFT` se inserta debajo de una aplicación. Contiene 2 métodos: `Method1` y `Method2`. Ni la interfaz ni los métodos contienen ningún código de implementación. Sólo la parte de declaración de los métodos se completará con las declaraciones de variables que se desee:

Interfaz con 2 métodos:



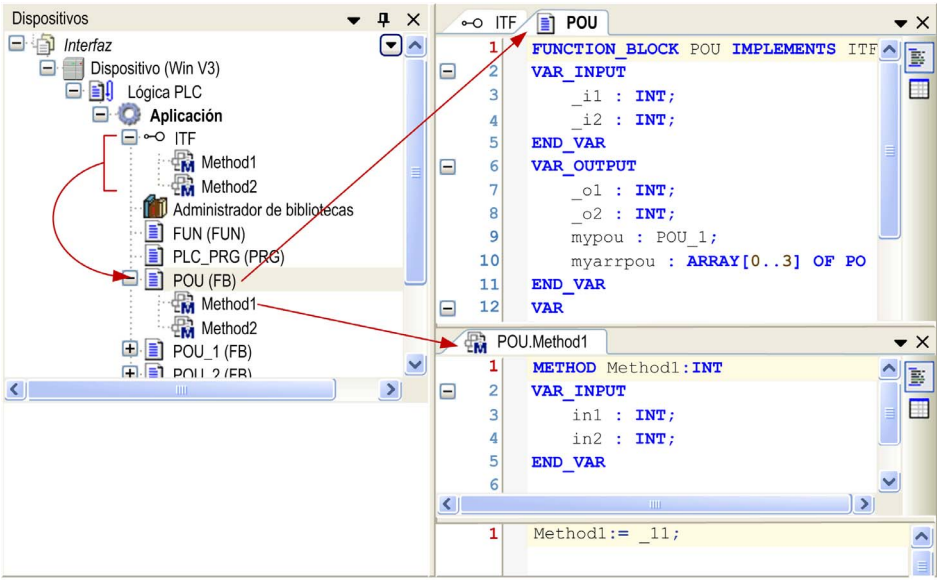
Ahora pueden insertarse 1 o varios bloques de funciones, que implementan la interfaz `ITF` definida más arriba.

Creación de un bloque de funciones que implementa una interfaz



Cuando el bloque de funciones `POU` se añade al árbol **Aplicaciones**, los métodos `Method1` y `Method2` se insertan automáticamente debajo, conforme se ha definido mediante `ITF`. Aquí pueden completarse con código de implementación específico del bloque de funciones.

Uso de la interfaz en la definición del bloque de funciones



Una interfaz puede extender otras interfaces mediante `EXTENDS` (consulte el ejemplo siguiente, *Ejemplo para extender una interfaz (véase página 188)*) en la definición de la interfaz. También se pueden extender los bloques de funciones.

Propiedades de interfaz

Una interfaz también puede definir una propiedad de interfaz, que consta de los métodos de descriptor de acceso `Get` y `Set`. Para obtener más información sobre las propiedades, consulte los capítulos *Propiedad (véase página 183)* y Propiedad de interfaz *(véase página 189)*. Una propiedad en una interfaz, como los métodos que puede incluir, sólo es un prototipo, lo que significa que no contiene código de implementación. Al igual que los métodos, se añade automáticamente al bloque de funciones, que implementa la interfaz. En el bloque de funciones podrá completarse con código de programación específico.

Consideraciones

Tenga en cuenta lo siguiente:

- No se permite declarar variables en la interfaz. La interfaz no tiene cuerpo (parte de implementación) ni acciones. En la interfaz sólo se define una colección de métodos y esos métodos sólo se permite que tengan variables de entrada, variables de salida y variables de entrada/salida.
- Las variables declaradas con el tipo de una interfaz se tratan como referencias.
- Un bloque de funciones que implementa una interfaz debe tener asignados métodos y propiedades con exactamente el mismo nombre que tienen en la interfaz. Deben contener entradas, salidas y entradas/salidas también con el mismo nombre.

NOTA: Al copiar o desplazar un método o una propiedad de una POU a una interfaz, las implementaciones contenidas se eliminan de forma automática. Al copiar o desplazar de una interfaz a una POU, se solicita especificar el lenguaje de implementación deseado.

Inserción de una interfaz

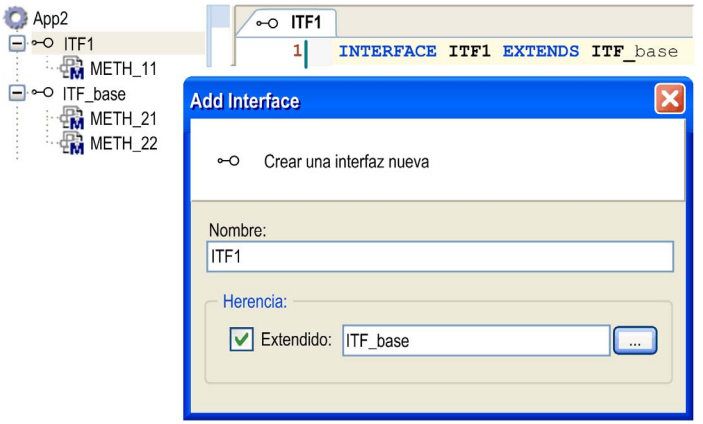
Para añadir una interfaz en una aplicación, seleccione el nodo **Aplicación** en el árbol **Aplicaciones**, o en **Catálogo de software** →**Activos** haga clic en el signo más de color verde y seleccione **Añadir otros objetos...** →**Interfaz**. Como alternativa, ejecute el comando **Agregar objeto** →**Interfaz**. Si selecciona el nodo **Global** antes de ejecutar el comando, la nueva interfaz estará disponible para todas las aplicaciones.

En el cuadro de diálogo **Add Interface**, escriba un nombre para la nueva interfaz (<nombre de interfaz>). Como alternativa, puede activar la opción **Extendido:** si desea que la interfaz actual sea una extensión *(véase página 204)* de otra interfaz.

Ejemplo para extender una interfaz

Si `ITF1` extiende `ITF_base`, todos los métodos que describe `ITF_base` estarán automáticamente disponibles en `ITF1`.

Extensión de una interfaz



Haga clic en **Agregar** para confirmar la configuración. Se abrirá la vista del editor de la nueva interfaz.

Declaración de una interfaz

Sintaxis

INTERFACE <nombre de interfaz>

Para una interfaz que extiende a otra:

INTERFACE <nombre de interfaz> EXTENDS <nombre de interfaz base>

Ejemplo

```
INTERFACE interface1 EXTENDS interface_base
```

Adición de una colección de métodos

Para completar la definición de la interfaz, añada la colección de métodos que desee. Para este fin, seleccione el nodo de la interfaz en el árbol **Aplicaciones** o en **Catálogo de software** → **Activos** y ejecute el comando **Método de interfaces...** Se abrirá el cuadro de diálogo **Add Interface Method** para definir un método que formará parte de la interfaz. Como alternativa, seleccione el nodo de la interfaz en el árbol **Aplicaciones**, haga clic en el signo más de color verde y seleccione **Método de interfaces**. Añada tantos métodos como desee y recuerde que sólo se permite que esos métodos tengan variables de entrada, variables de salida y variables de entrada/salida, pero no cuerpo (parte de implementación).

Propiedad de interfaz

Descripción general

Además de los métodos y programas, también puede utilizarse una propiedad, disponible como un medio de programación orientada a objetos, en la definición de una interfaz *(véase página 185)*. En este caso, se denomina propiedad de interfaz. Para agregarla a la interfaz seleccionada en el árbol **Aplicaciones**, haga clic en el signo más de color verde y ejecute el comando **Propiedad de interfaz...** Como alternativa, haga clic con el botón derecho del ratón en el nodo de interfaz y ejecute el comando **Agregar objeto →Propiedad de interfaz** en el menú contextual.

Para obtener más información sobre una propiedad y sus métodos, consulte *Propiedad* *(véase página 183)*

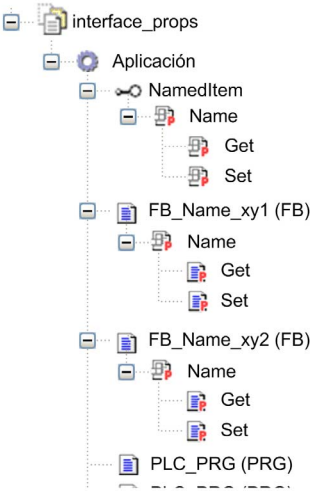
Una propiedad de interfaz amplía la descripción de una interfaz. Al igual que la interfaz, sólo define que los métodos de descriptor de acceso `Get` o `Set` (puede utilizar ambos o sólo uno de ellos) pertenecen a la interfaz; sin embargo, no se proporciona código de implementación para ellos. Cuando se amplía un bloque de funciones con una interfaz que contiene propiedades, estas propiedades y sus descriptores de acceso `Get` o `Set` se insertan automáticamente en el árbol **Dispositivos** debajo del objeto del bloque de funciones. Pueden editarse para añadir el código de implementación deseado.

Ejemplo

En la figura siguiente, la interfaz `NamedItem` tiene una propiedad `Name` con un método de descriptor de acceso `Get` y `Set`. El descriptor de acceso `Get` de este ejemplo está destinado para su uso en la lectura del nombre de cualquier elemento de un bloque de funciones que implementa la interfaz. El descriptor de acceso `Set` puede utilizarse para escribir un nombre en este bloque de funciones. Ambos métodos pueden editarse en la definición de interfaz, pero posteriormente en el bloque de funciones.

El bloque de funciones `FB_Name_xy1` se ha añadido al árbol **Dispositivos**, de forma que se implementa la interfaz (`FUNCTION_BLOCK FB_Name_xy1 IMPLEMENTS NamedItem`). Por tanto, la propiedad `Name` con los métodos `Get` y `Set` se ha insertado automáticamente debajo de `FB_Name_xy1`. Aquí puede editar los métodos de descriptor de acceso; por ejemplo, en el modo en que la variable `name_of_xy1` se lee y, por tanto, se `got` el nombre de un elemento. En otro bloque de funciones que también implementa la misma interfaz, el método `Get` puede rellenarse con otro código. Este código puede proporcionar el nombre de cualquier otro elemento. El método `Set` del ejemplo se utiliza para escribir un nombre, definido por el programa `PLC_PRG ('abc')`, en el bloque de funciones `FB_Name_xy2`.

Interfaz `NamedItem` implementada en 2 bloques de funciones



2 bloques de funciones que implementan la interfaz `NamedItem`

Bloque de funciones `FB_Name_xy1`

```
FUNCTION_BLOCK FB_Name_xy1 IMPLEMENTS NamedItem
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
    name_of_xy1: STRING:='xy1';
END_VAR
```

Bloque de funciones `FB_Name_xy2`

```
FUNCTION_BLOCK FB_Name_xy2 IMPLEMENTS NamedItem
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
    name_of_xy2: STRING:='xy2';
 name_got_from_PLC_PRG: STRING;
END_VAR
```

Implementación de código en los métodos de descriptor de acceso `Get` y `Set` debajo de los 2 bloques de funciones

FB_Name_xy1.Get

```
VAR
END_VAR
name := name_of_xy1;
```

FB_Name_xy2.Get

```
VAR
END_VAR
name := name_of_xy2;
```

FB_Name_xy2.Set

```
VAR
END_VAR
name_got_from_PLC_PRG:=name;
```

Acceso a los bloques de funciones mediante el programa `PLC_PRG`

```
PROGRAM PLC_PRG
VAR
    FBxy1_inst: FB_Name_xy1;
    FBxy2_inst: FB_Name_xy2;
    namexy1: STRING;
    namexy2: STRING;
END_VAR
//get name out of fb
namexy1:=FBxy1_inst.Name;
namexy2:=FBxy2_inst.Name;
//write name to fb
FBxy2_inst.Name:='abc';
```

Acción

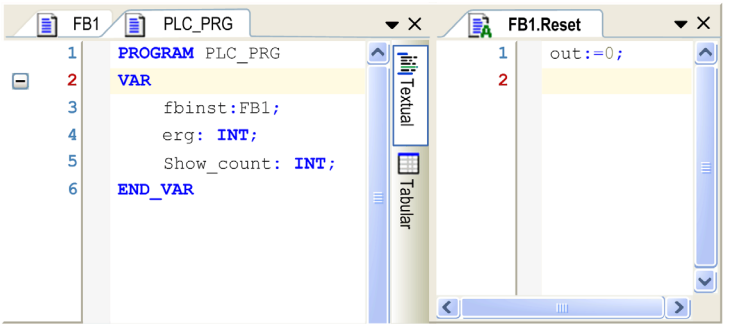
Descripción general

Puede definir acciones y asignarlas a bloques de funciones *(véase página 196)* y programas *(véase página 174)*. Una acción es una implementación adicional. Se puede crear en un lenguaje diferente al de la implementación básica. A cada acción se le da un nombre.

Las acciones trabajan con los datos del bloque de funciones o programa al cual pertenecen. Las acciones utilizan variables de entrada/salida y variables locales definidas y no contienen sus propias declaraciones.

Ejemplo de una acción de un bloque de funciones

En la ilustración siguiente se muestra una acción en FB:



En este ejemplo, cada llamada del bloque de funciones `FB1` incrementa o reduce la variable de salida `out`, en función del valor de la variable de entrada `in`. Si se invoca la acción `Reset` del bloque de funciones, la variable de salida `out` se establece en 0. En ambos casos se escribe la misma variable `out`.

Inserción de una acción

Para añadir una acción, seleccione el nodo correspondiente del programa o bloque de funciones en el árbol **Aplicaciones** o en el nodo **Global** del árbol **Aplicaciones**, haga clic en el signo más de color verde y ejecute el comando **Acción...** Como alternativa, haga clic con el botón derecho en el nodo del programa o bloque de funciones y ejecute el comando **Agregar objeto** →**Acción**. En el cuadro de diálogo **Agregar acción**, defina el **Nombre** de la acción y el **Lenguaje de implementación** deseado.

Llamada a una acción

Sintaxis

```
<Program_name>.<Action_name>
```

o

```
<Instance_name>.<Action_name>
```

Tenga en cuenta la notación en FBD (consulte el ejemplo siguiente).

Si es necesario llamar a la acción desde su propio bloque, que es el programa o el bloque de funciones al cual pertenece, es suficiente utilizar el nombre de la acción.

Ejemplos

En esta sección se ofrecen ejemplos para llamar a la acción que se describe más arriba desde otra POU.

Declaración para todos los ejemplos:

```
PROGRAM PLC_PRG
VAR
    Inst : Counter;
END_VAR
```

Llamada de la acción `Reset` en otra POU, que está programada en IL:

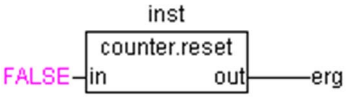
```
CAL Inst.Reset(In := FALSE)
LD Inst.out
ST ERG
```

Llamada de la acción `Reset` en otra POU, que está programada en ST:

```
Inst.Reset(In := FALSE);
Erg := Inst.out;
```

Llamada de la acción `Reset` en otra POU, que está programada en FBD:

Acción en FBD



NOTA: El estándar IEC sólo reconoce las acciones del diagrama funcional secuencial (SFC). Estas acciones son una parte esencial que contiene las instrucciones que se procesarán en pasos concretos del diagrama.

Función, bloque de funciones y método externos

Descripción general

El sistema de programación no generará ningún código para una función, bloque de funciones o método externo.

Realice los siguientes pasos para crear una POU externa:

Paso	Acción
1.	Añada el objeto POU que desee al nodo **Global** del árbol **Aplicaciones** de su proyecto como cualquier objeto interno y defina las respectivas variables de entrada y salida. **NOTA:** Defina las variables locales en los bloques de funciones externos. No las defina en funciones o métodos externos. Las variables `VAR_STAT` no se pueden utilizar en el sistema de tiempo de ejecución.
2.	Defina la POU como externa: Para ello, haga clic con el botón derecho en el objeto POU del nodo **Global** del árbol **Aplicaciones** y ejecute el comando **Propiedades** en el menú contextual. Abra la ficha **Compilar** y active la opción **External Implementation (Late link in the runtime system)**.

En el sistema de tiempo de ejecución, se tiene que implementar una función, bloque de funciones o método equivalente. En una descarga de programa, el equivalente para cada POU externa se busca en el sistema de tiempo de ejecución. Si se encuentra el equivalente, se vincula.

POUs para comprobaciones implícitas

Descripción general

Bajo una aplicación puede añadir POU especiales. Tienen que estar disponibles si durante el tiempo de ejecución se debe utilizar la funcionalidad de comprobación proporcionada implícitamente para los límites de matriz y rango, las divisiones entre cero y los punteros. Puede desactivar esta funcionalidad en el caso de los dispositivos cuyas funciones de comprobación las proporciona una biblioteca implícita especial.

Para esta finalidad, el cuadro de diálogo **Agregar objeto →POUs para comprobaciones implícitas** proporciona las funciones siguientes:

- CheckBounds *(véase página 678)*
- CheckDivInt *(véase página 711)*
- CheckDivLInt *(véase página 711)*
- CheckDivReal *(véase página 711)*
- CheckDivLreal *(véase página 711)*
- CheckRange *(véase página 685)*
- CheckRangeUnsigned *(véase página 685)*
- CheckPointer *(véase página 666)*

Tras haber insertado una POU de comprobación, se abre en el editor correspondiente al lenguaje de implementación seleccionado. En el editor ST hay disponible una implementación predeterminada que puede adaptar a sus necesidades.

Tras haber insertado una POU de comprobación determinada, la opción dejará de estar disponible en el cuadro de diálogo, lo que impedirá una doble inserción. Si ya se han añadido todos los tipos de POU de comprobación bajo la aplicación, en el cuadro de diálogo **Agregar objeto** ya no se proporciona la opción **POUs para comprobaciones implícitas**.

⚠ ATENCIÓN

FUNCIONALIDAD DE COMPROBACIONES IMPLÍCITAS INCORRECTAS

No modifique la parte de declaración de una función de comprobación implícita, a fin de conservar su integridad funcional.

El incumplimiento de estas instrucciones puede causar lesiones o daño al equipo.

NOTA: A partir de SoMachine V4.0, después de eliminar la función de comprobación implícita (como CheckBounds) de la aplicación, no se puede utilizar **Cambio en línea**; solamente se puede realizar una descarga. Aparecerá el mensaje correspondiente.

Sección 6.2
Bloque de funciones

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Información general

Descripción general

Un bloque de funciones es una POU *(véase página 169)* que proporciona 1 o más valores durante el proceso de un programa de controlador. A diferencia de una función, los valores de las variables de salida y las variables internas necesarias pueden persistir de una ejecución del bloque de funciones a la siguiente. Por lo tanto, la invocación de un bloque de funciones con los mismos argumentos (parámetros de entrada) no siempre tiene por qué producir los mismos valores de salida.

Además de la funcionalidad descrita por el estándar IEC11631-3, se admite la programación orientada a objetos, y pueden definirse bloques de funciones como extensiones *(véase página 204)* de otros bloques de funciones. Pueden incluir definiciones de interfaz *(véase página 207)* relativas a la Invocación de métodos *(véase página 209)*. Por consiguiente, puede utilizarse la herencia al programar con bloques de funciones.

Para llamar a un bloque de funciones siempre se utiliza una instancia *(véase página 200)*, que es una reproducción (copia) del bloque de funciones.

Adición de un bloque de funciones

Para añadir un bloque de funciones a una aplicación existente, seleccione el nodo correspondiente en **Catálogo de software** →**Activos** o en el árbol **Aplicaciones**, haga clic en el signo más de color verde y seleccione **POU...** Como alternativa, puede hacer clic con el botón derecho en el nodo y ejecutar el comando **Agregar objeto** →**POU**. Para crear un bloque de funciones independiente de una aplicación, seleccione el nodo **Global** del árbol **Aplicaciones** o de **Activos**.

En el cuadro de diálogo **Agregar objeto**, seleccione la opción **Bloque de funciones**, especifique un **Nombre** (<identificador>) de bloque de funciones y seleccione el **Lenguaje de implementación** que desee.

Además, puede especificar las siguientes opciones:

Opción	Descripción
Extendido	Indique el nombre de otro bloque de funciones disponible en el proyecto que debe ser la base para el actual. Para obtener información detallada, consulte *Ampliación de un bloque de funciones (véase página 204)*.
Implementado	Escriba los nombres de las interfaces *(véase página 185)* disponibles en el proyecto que deben implementarse en el bloque de funciones actual. Puede especificar diversas interfaces separadas por comas. Para obtener información detallada, consulte *Implementación de interfaces (véase página 207)*.

Opción	Descripción
Modificador de acceso	Por motivos de compatibilidad, los modificadores de acceso son opcionales. El modificador **PUBLIC** está disponible como equivalente a no establecer ningún modificador. Como alternativa, seleccione una de las opciones de la lista de selección: ● **INTERNO**: el acceso en el bloque de funciones está restringido al espacio de nombres (biblioteca) actual. ● **FINAL**: no se puede derivar el acceso, es decir, el bloque de funciones no puede ampliarse con otro bloque. Permite la generación de código optimizado. **NOTA:** Los modificadores de acceso son válidos a partir de la versión del compilador 3.4.4.0 y, por lo tanto, se pueden utilizar como modificadores en versiones anteriores. Para obtener más información, consulte la tabla de asignación de versiones del compilador SoMachine/CoDeSys en SoMachineCompatibilidad y migración - Guía del usuario *(véase SoMachine - Compatibilidad y migración, Guía del usuario)*.
Lenguaje de implementación de método	Seleccione el lenguaje de programación que desee para todos los objetos de método creados mediante la implementación de la interfaz, independientemente del establecido para el bloque de funciones en cuestión.

Haga clic en **Agregar** para confirmar la configuración. Se abre la vista de editor para el nuevo bloque de funciones y puede empezar la edición.

Declaración de un bloque de funciones

Sintaxis

FUNCTION_BLOCK <modificador de acceso> <nombre del bloque de funciones> | EXTENDS <nombre del bloque de funciones> | IMPLEMENTS <lista de nombres de interfaz separados por comas>

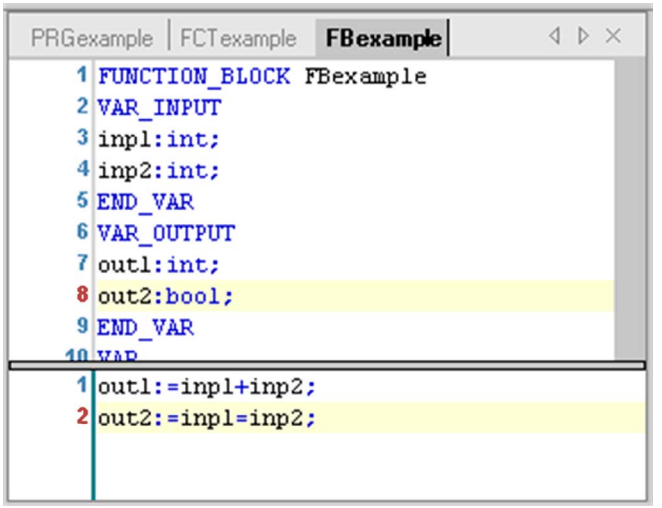
A esto le sigue la declaración de las variables.

Ejemplo

El ejemplo `FBexample` de la siguiente figura tiene 2 variables de entrada y 2 variables de salida, `out1` y `out2`.

`out1` es la suma de las 2 entradas, y `out2` es el resultado de una comparación de igualdad.

Ejemplo de un bloque de funciones en ST



Instancia de bloque de funciones

Descripción general

Los bloques de funciones se llaman *(véase página 201)* mediante una instancia que es una reproducción (copia) de un bloque de funciones *(véase página 197)*.

Cada instancia tiene su propio identificador (el nombre de la instancia) y una estructura de datos que contiene sus entradas, salidas y variables internas.

Las instancias, como las variables, se declaran local o globalmente. El nombre del bloque de funciones se indica como el tipo de datos de un identificador.

Sintaxis para declarar una instancia de bloque de funciones

<identificador>:<nombre del bloque de funciones>;

Ejemplo

Declaración (por ejemplo, en la parte de declaración de un programa) de la instancia `INSTANCE` del bloque de funciones `FUB`:

```
INSTANCE: FUB;
```

Las partes de la declaración de bloques de funciones y programas pueden contener declaraciones de instancias. Sin embargo, las declaraciones de instancias no se permiten en las funciones.

Llamada a un bloque de funciones

Descripción general

Los bloques de funciones *(véase página 197)* se llaman a través de una instancia de bloque de funciones. Por ello, una instancia de bloque de funciones debe declararse local o globalmente. Consulte el capítulo *Instancia de bloque de funciones (véase página 200)* para obtener información sobre cómo realizar la declaración.

A continuación, puede accederse a la variable de bloque de funciones deseada utilizando la sintaxis siguiente.

Sintaxis

<nombre de instancia>.<nombre de variable>

Consideraciones

- Sólo puede accederse a las variables de entrada y salida de un bloque de funciones desde fuera de una instancia de bloque de funciones, pero no a sus variables internas.
- El acceso a una instancia de bloque de funciones está limitado a la POU *(véase página 169)* en la que se ha declarado, a menos que se haya declarado globalmente.
- Al llamar a la instancia, pueden asignarse los valores deseados a los parámetros de bloque de funciones. Consulte el párrafo siguiente, *Asignación de parámetros durante la llamada*.
- Las variables de entrada/salida (`VAR_IN_OUT`) de un bloque de funciones se proporcionan como punteros.
- En SFC, las llamadas de bloque de funciones sólo pueden tener lugar en pasos.
- El nombre de la instancia del bloque de funciones puede utilizarse como parámetro de entrada para una función u otro bloque de funciones.
- Todos los valores de un bloque de funciones se conservan hasta el siguiente procesamiento del bloque de funciones. Por tanto, las llamadas de bloque de funciones no siempre devuelven los mismos valores de salida, aunque se realicen con argumentos idénticos.

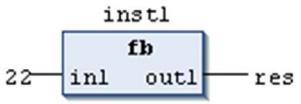
NOTA: Si al menos 1 de las variables del bloque de funciones es una variable remanente, la instancia total se almacena en el área de datos Retain.

Ejemplos de acceso a variables de bloque de funciones

Supongamos que el bloque de funciones `fb` tiene una variable de entrada `in1` de tipo INT. Aquí se puede ver la llamada de esta variable desde dentro del programa `prog`. Observe la declaración e implementación en ST:

```
PROGRAM prog
VAR
inst1:fb;
END_VAR
inst1.in1:=22;   (* fb is called and input variable in1 gets assigned v
alue 22 *)
inst1(); (* fb is called, this is needed for the following access on th
e output variable *)
res:=inst1.outl; (* output variable of fb is read *)
```

Ejemplo de una llamada de bloque de funciones en FBD:



Asignación de parámetros durante la llamada

En los lenguajes de texto IL y ST, puede establecer parámetros de entrada o salida inmediatamente al llamar al bloque de funciones. Los valores pueden asignarse a los parámetros entre paréntesis después del nombre de instancia del bloque de funciones. Para los parámetros de entrada, esta asignación tiene lugar utilizando `:=`, al igual que en la inicialización de variables *(véase página 583)* en la posición de declaración. Para los parámetros de salida, debe utilizarse `=>`.

Ejemplo de llamada con asignaciones

En este ejemplo, se llama a un bloque de funciones de temporizador (instancia `CMD_TMR`) con asignaciones para los parámetros `IN`y `PT`. A continuación, la variable resultante `Q` se asigna a la variable `A`. La variable resultante consta del nombre de la instancia del bloque de funciones, un punto seguido y el nombre de la variable:

```
CMD_TMR(IN := %IX5, PT := 300);
A:=CMD_TMR.Q
```

Ejemplo de inserción con argumentos mediante el asistente Accesibilidad

Si la instancia se inserta mediante **Accesibilidad** con la opción **Insertar con argumentos** en la vista de implementación de una POU ST o IL, se muestra automáticamente de acuerdo con la sintaxis que se muestra en el ejemplo siguiente con todos sus parámetros, aunque no es obligatorio asignar estos parámetros.

En el ejemplo anteriormente mencionado, la llamada se mostraría del modo siguiente.

```
CMD_TMR(in:=, pt:=, q=>)
-> fill in, e.g.:
CMD_TMR(in:=bvar, pt:=t#200ms, q=>bres);
```

Extensión de un bloque de funciones

Descripción general

Al admitir la programación orientada a objetos, un bloque de funciones se puede derivar de otro bloque de funciones. Esto significa que un bloque de funciones puede ser la extensión de otro, con lo que obtiene automáticamente las propiedades del bloque de funciones base además de las suyas propias.

La extensión se efectúa con la palabra clave `EXTENDS` en la declaración de un bloque de funciones. Puede seleccionar la opción `EXTENDS` ya durante la adición de un bloque de funciones al proyecto mediante el cuadro de diálogo **Agregar objeto**.

Sintaxis

FUNCTION_BLOCK <nombre bloque funciones> EXTENDS <nombre bloque funciones>

A esto le sigue la declaración de las variables.

Ejemplo

Definición del bloque de funciones `fbA`

```
FUNCTION_BLOCK fbA
VAR_INPUT
    x:int;
END_VAR
...
```

Definición del bloque de funciones `fbB`

```
FUNCTION_BLOCK fbB EXTENDS fbA
VAR_INPUT
    ivar: INT := 0;
END_VAR
...
```

Extensión mediante `EXTENDS`

La extensión mediante `EXTENDS` significa:

- `fbB` contiene todos los datos y métodos definidos por `fbA`. Ahora se puede utilizar una instancia de `fbB` en cualquier contexto en el que se espere un bloque de funciones de tipo `fbA`.
- `fbB` puede sobrescribir los métodos definidos en `fbA`. Esto significa que `fbB` puede declarar un método con el mismo nombre y la misma entrada y salida que el declarado por `A`.
- `fbB` no puede utilizar variables de bloque de funciones con el mismo nombre que las utilizadas en `fbA`. En este caso, el compilador generará un mensaje de error.
- Se puede acceder a las variables y los métodos de `fbA` directamente desde un ámbito de `fbB` mediante el puntero SUPER *(véase página 211)* (`SUPER^.<method>`).

NOTA: La herencia múltiple no está permitida.

Ejemplo

```
FUNCTION_BLOCK FB_Base
VAR_INPUT
END_VAR
VAR_OUTPUT
    iCnt : INT;
END_VAR
VAR
END_VAR
THIS^.METH_DoIt();
THIS^.METH_DoAlso();

    METHOD METH_DoIt : BOOL
    VAR
    END_VAR
    iCnt := -1;
    METH_DoIt := TRUE;

    METHOD METH_DoAlso : BOOL
    VAR
    END_VAR
    METH_DoAlso := TRUE;
FUNCTION_BLOCK FB_1 EXTENDS FB_Base
VAR_INPUT
END_VAR
VAR_OUTPUT
END_VAR
VAR
END_VAR
// Calls the method defined under FB_1
THIS^.METH_DoIt();
THIS^.METH_DoAlso();
// Calls the method defined under FB_Base
SUPER^.METH_DoIt();
SUPER^.METH_DoAlso();
    METHOD METH_DoIt : BOOL
    VAR
    END_VAR
    iCnt := 1111;
    METH_DoIt := TRUE;
```

```
PROGRAM PLC_PRG
VAR
    Myfb_1: FB_1;
    iFB: INT;
    iBase: INT;
END_VAR
Myfb_1();
iBase := Myfb_1.iCnt_Base;
iFB := Myfb_1.iCnt_THIS;
```

Implementación de interfaces

Descripción general

Para admitir la programación orientada a objetos, un bloque de funciones puede implementar varias interfaces *(véase página 185)* que permiten utilizar métodos *(véase página 180)*.

Sintaxis

FUNCTION_BLOCK <nombre de bloque de funciones> IMPLEMENTS <nombre de interfaz_1>|,<nombre de interfaz_2>, ..., <nombre de interfaz_n>

Un bloque de funciones que implementa una interfaz debe contener todos los métodos y propiedades (propiedad de interfaz *(véase página 189)*) definidas por esa interfaz. Se incluyen el nombre, las entradas y la salida del método o propiedad concretos, que deben ser exactamente los mismos.

Por este motivo, al crear un nuevo bloque de funciones que implemente una interfaz, todos los métodos y propiedades definidos en esa interfaz se insertarán automáticamente debajo del nuevo bloque de funciones en el árbol **Aplicaciones**.

NOTA: Si después se añaden métodos a la definición de interfaz, no se añadirán automáticamente a los bloques de funciones correspondientes. Ejecute el comando Implementar interfaces... *(véase SoMachine, Comandos de menú, Ayuda en línea)* para realizar esta actualización explícitamente.

Ejemplo

`INTERFACE I1` incluye el método `GetName`:

```
METHOD GetName : STRING
```

Cada bloque de funciones `A` y `B` implementa la interfaz `I1`:

```
FUNCTION_BLOCK A IMPLEMENTS I1
FUNCTION_BLOCK B IMPLEMENTS I1
```

Así, en los dos bloques de funciones, el método `GetName` debe estar disponible y se insertará automáticamente debajo de cada uno cuando los bloques de funciones se inserten en el árbol **Aplicaciones**.

Tenga en cuenta esta declaración de una variable de tipo `I1`:

```
FUNCTION DeliverName : STRING
VAR_INPUT
  l_i : I1;
END_VAR
```

Esta entrada puede recibir todos los bloques de funciones que implementan la interfaz `I1`.

Ejemplo de llamadas a funciones:

```
DeliverName(l_i := A_instance); // call with instance of type A
DeliverName(l_i := B_instance); // call with instance of type B
```

NOTA: Para poder llamar a un método, debe asignarse una instancia de un bloque de funciones a una variable de tipo interfaz. Una variable de tipo interfaz siempre es una referencia a la instancia de bloque de funciones asignada.

De este modo, una llamada al método de interfaz produce una llamada a la implementación del bloque de funciones. En cuanto se asigna la referencia, la dirección correspondiente se supervisa en modalidad online. En caso contrario, si todavía no se ha asignado ninguna referencia, se mostrará el valor 0 en la supervisión en modalidad online.

Para este ejemplo, consulte la parte de implementación de la función `DeliverName`:

```
DeliverName := l_i.GetName(); // in this case it depends on the "real"
type of l_i whether A.GetName or B.GetName is called
```

NOTA: Consulte también la posibilidad de extender un bloque de funciones *(véase página 204)* utilizando la palabra clave `EXTENDS` en la declaración.

Invocación de métodos

Descripción general

La programación orientada a objetos con bloques de funciones, aparte de ofrecer extensión *(véase página 204)* mediante `EXTENDS`, admite el posible uso de interfaces *(véase página 207)* y herencias. Esto requiere invocaciones de métodos con resolución dinámica, también denominadas llamadas de función virtuales.

Las llamadas de función virtuales necesitan algo más de tiempo que las llamadas de función normales, y se utilizan en los siguientes casos:

- cuando se realiza una llamada mediante un puntero a un bloque de funciones (`pfub`^.method)
- Cuando se llama a un método de una variable de interfaz (interface1.method).
- Cuando un método llama a otro método del mismo bloque de funciones.
- Cuando se realiza una llamada mediante una referencia a un bloque de funciones.
- Cuando a `VAR_IN_OUT` de un tipo de bloque de funciones básico se le puede asignar una instancia de un tipo de bloque de funciones derivado.

Las llamadas de función virtuales permiten que la misma llamada en un código fuente de programa invoque diferentes métodos durante el tiempo de ejecución.

Para obtener información más detallada, consulte:

- *Método (véase página 180)* para obtener más información sobre métodos.
- *Puntero THIS (véase página 213)* para obtener más información sobre el uso del puntero THIS.
- *Puntero SUPER (véase página 211)* para obtener más información sobre el puntero SUPER.

Métodos de llamada

Según el estándar IEC 61131-3 , los métodos como las funciones *(véase página 177)* normales pueden tener salidas adicionales. Deben asignarse a la llamada de método según la sintaxis siguiente:

<método>(in1:=<valor> |, más asignaciones de entrada, out1 => <variable de salida 1> | out2 => <variable de salida 2> | ...más asignaciones de salida)

Como resultado, la salida del método se escribe en las variables de salida declaradas localmente tal como se especifica en la llamada.

Ejemplo

Supongamos que los bloques de funciones `fub1` y `fub2` extienden (`EXTEND`) el bloque de funciones `fubbase` e implementan (`IMPLEMENT`) `interface1`. Se incluye el método `method1`.

Posible uso de las interfaces y llamadas de método:

```
PROGRAM PLC_PRG
VAR_INPUT
  b : BOOL;
END_VAR
VAR
  pInst : POINTER TO fubbase;
  instBase : fubbase;
  inst1 : fub1;
  inst2 : fub2;
  instRef : REFERENCE to fubbase;
END_VAR
IF b THEN
  instRef REF= inst1;            (* Reference to fub1 *)
  pInst := ADR(instBase);
ELSE
  instRef REF= inst2;            (* Reference to fub2 *)
  pInst := ADR(inst1);
END_IF
pInst^.method1();                 (* If b is true, fubbase.method1 is c
alled, else fub1.method1 is called *)
instRef.method1();                (* If b is true, fub1.method1 is call
ed, else fub2.method1 is called *)
```

Supongamos que `fubbase` del ejemplo anterior contiene 2 métodos, `method1` y `method2`. `fub1` anula `method2`, pero no `method1`.

Se llama a `method1` como se muestra en el ejemplo anterior.

```
pInst^.method1(); (* If b is true fubbase.method1 is called, else fub1.
method1 is called *)
```

Para realizar la llamada mediante el puntero THIS, consulte *Puntero THIS* *(véase página 213)*.

Puntero SUPER

Descripción general

Para cada bloque de funciones está disponible automáticamente un puntero con el nombre SUPER. Apunta a las instancias del bloque de funciones básico, de las que se crea el bloque de funciones con la herencia del bloque de funciones básico.

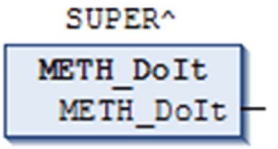
Esto proporciona una solución eficaz para el problema siguiente:

- SUPER ofrece acceso a los métodos de implementación de la clase base. Con la palabra clave SUPER se puede llamar a un método válido en la instancia de clase base (objeto padre). Por lo tanto, no se lleva a cabo ningún enlace de nombre dinámico.

SUPER sólo se puede utilizar en métodos y en la implementación del bloque de funciones asociado.

Debido a que SUPER es un puntero al bloque de funciones básico, tiene que quitarle la referencia para obtener la dirección del bloque de funciones: SUPER^.METH_DoIt.

Llamada de SUPER en distintos lenguajes de implementación

Lenguaje de implementación	Ejemplo
ST	`SUPER^.METH_DoIt();`
FBD/CFC/LD	

NOTA: La funcionalidad de SUPER todavía no está implementada para la lista de instrucciones.

Ejemplo

La variable local `iVarB` sobrecarga la variable de bloque de funciones `iVarB`.

```
FUNCTION_BLOCK FB_Base
VAR_OUTPUT
    iCnt : INT;
END_VAR
    METHOD METH_DoIt : BOOL
    iCnt := -1;

    METHOD METH_DoAlso : BOOL
    METH_DoAlso := TRUE;

FUNCTION_BLOCK FB_1 EXTENDS FB_Base
VAR_OUTPUT
    iBase: INT;
END_VAR
// Calls the method defined under FB_1
THIS^.METH_DoIt();
THIS^.METH_DoAlso();
// Calls the method defined under FB_Base
SUPER^.METH_DoIt();
SUPER^.METH_DoAlso();
iBase := SUPER^.iCnt;

    METHOD METH_DoIt : BOOL
    iCnt := 1111;
    METH_DoIt := TRUE;

END_VAR
ROGRAM PLC_PRG
VAR
    myBase: FB_Base;
    myFB_1: FB_1;
    iTHIS: INT;
    iBase: INT;
END_VAR
myBase();
iBase := myBase.iCnt;
myFB_1();
iTHIS := myFB_1.iCnt;
```

Puntero THIS

Descripción general

Para cada bloque de funciones está disponible automáticamente un puntero con el nombre THIS. Apunta a su propia instancia de bloque de funciones.

Esto proporciona una solución eficaz para los problemas siguientes:

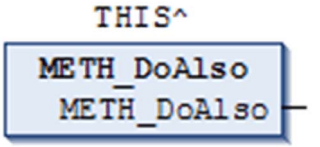
- Si una variable declarada a nivel local en el método oculta una variable de bloque de funciones.
- Si desea referenciar un puntero a su propia instancia de bloque de funciones para el uso en una función.

THIS sólo se puede utilizar en métodos y en la implementación del bloque de funciones asociado.

THIS se debe escribir en mayúsculas. No se aceptan otras redacciones.

Debido a que THIS es un puntero al bloque de funciones que hereda, tiene que quitarle la referencia para obtener la dirección de este bloque de funciones principal: THIS^.METHDoIt.

Llamada de THIS en distintos lenguajes de implementación

Lenguaje de implementación	Ejemplo
ST	`THIS^.METH_DoIt();`
FBD/CFC/LD	

NOTA: La funcionalidad de THIS todavía no está implementada para la lista de instrucciones.

Ejemplo 1

La variable local `iVarB` deja en segundo plano la variable de bloque de funciones `iVarB`.

```
FUNCTION_BLOCK fbA
VAR_INPUT
    iVarA: INT;
END_VAR
iVarA := 1;

FUNCTION_BLOCK fbB EXTENDS fbA
VAR_INPUT
    iVarB: INT := 0;
END_VAR
iVarA := 11;
iVarB := 2;

    METHOD DoIt : BOOL
    VAR_INPUT
    END_VAR
    VAR
        iVarB: INT;
    END_VAR
    iVarB := 22; // Here the local iVarB is set.
    THIS^.iVarB := 222; // Here the function block variable iVarB is se
t, although iVarB is overloaded.

PROGRAM PLC_PRG
VAR
    MyfbB: fbB;
END_VAR

MyfbB(iVarA:=0 , iVarB:= 0);
MyfbB.DoIt();
```

Ejemplo 2

Llamada de función que necesita una referencia a su propia instancia.

```
FUNCTION funA
VAR_INPUT
    pFB: fbA;
END_VAR
...;

FUNCTION_BLOCK fbA
VAR_INPUT
    iVarA: INT;
END_VAR
...;

FUNCTION_BLOCK fbB EXTENDS fbA
VAR_INPUT
    iVarB: INT := 0;
END_VAR
iVarA := 11;
iVarB := 2;

    METHOD DoIt : BOOL
    VAR_INPUT
    END_VAR
    VAR
        iVarB: INT;
    END_VAR
    iVarB := 22;    //Here the local iVarB is set.
    funA(pFB := THIS^);    //Here funA is called with THIS^.

PROGRAM PLC_PRG
VAR
    MyfbB: fbB;
END_VAR
MyfbB(iVarA:=0 , iVarB:= 0);
MyfbB.DoIt();
```

Sección 6.3
Objetos de aplicación

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Tipo de datos (DUT)

Descripción general

Aparte de los tipos de datos estándar, puede definir sus propios tipos de datos. Puede crear estructuras *(véase página 681)*, tipos de enumeración *(véase página 683)* y referencias *(véase página 671)* como tipos de datos (DUT) en un editor DUT *(véase página 423)*.

Para obtener una descripción del estándar concreto y los tipos de datos definidos por el usuario, consulte la descripción de los tipos de datos *(véase página 662)*.

Adición de un objeto DUT

Para añadir un objeto DUT a una aplicación existente, seleccione el nodo de aplicación en **Catálogo de software** →**Activos** o en el árbol **Aplicaciones**, haga clic en el signo más de color verde y seleccione **DUT...** O bien haga clic con el botón derecho en el nodo pertinente y ejecute el comando **Agregar objeto** →**DUT**. Para crear un objeto DUT independiente de aplicación, seleccione el nodo **Global** en **Activos** o el árbol **Aplicaciones**. En el cuadro de diálogo **Add DUT**, introduzca un **Nombre** para el nuevo tipo de datos y elija el tipo deseado de **Estructura**, **Enumeración**, **Alias** o **Unión**.

En el caso del tipo **Estructura**, puede utilizar el principio de herencia, que admite la programación orientada a objetos. Opcionalmente, puede indicar que el DUT sea una extensión de otro DUT que ya esté definido en el proyecto. Esto significa que las definiciones del DUT extendido serán automáticamente válidas dentro del proyecto actual. Para este fin, active la opción **Extendido:** e introduzca el nombre del otro DUT.

Haga clic en **Agregar** para confirmar la configuración. Se abre la vista del editor para el nuevo DUT y puede empezar la edición.

Declaración de un objeto DUT

Sintaxis

TYPE <identificador>: <declaración componente DUT>END_TYPE

La declaración de componente DUT depende del tipo de DUT, como por ejemplo una estructura *(véase página 681)* o una enumeración *(véase página 683)*.

Ejemplo

El ejemplo siguiente contiene 2 DUT, que definen las estructuras `struct1` y `struct2`; `struct2` extiende `struct1`, lo que significa que puede utilizar `struct2.a` en su implementación para acceder a la variable `a`.

```
TYPE struct1 :
    STRUCT
        a:INT;
        b:BOOL;
    END_STRUCT
END_TYPE
TYPE struct2 EXTENDS struct1 :
    STRUCT
  c:DWORD;
  d:STRING;
    END_STRUCT
END_TYPE
```

Lista de variables globales - GVL

Descripción general

La lista de variables globales (GVL) sirve para declarar variables globales *(véase página 594)*. Si se especifica una GVL en el nodo **Global** de **Catálogo de software** →**Activos** →**POU** o del árbol **Aplicaciones**, las variables estarán disponibles para todo el proyecto. Si se asigna una GVL a una aplicación específica, las variables serán válidas en esta aplicación.

Para añadir una GVL a una aplicación existente, seleccione el nodo de aplicación correspondiente en **Catálogo de software** →**Activos** →**POU** o en el árbol **Aplicaciones**, haga clic en el signo más de color verde y seleccione **Lista de variables globales...** Como alternativa, puede hacer clic con el botón derecho en el nodo y ejecutar el comando **Agregar objeto** →**Agregar lista de variables globales...** Si selecciona el nodo **Global** en estas vistas, el nuevo objeto de GVL será independiente de la aplicación.

Utilice el editor GVL *(véase página 425)* para editar una lista de variables globales.

Las variables contenidas en una GVL se pueden definir para que estén disponibles como variables de red *(véase página 895)* para un intercambio de datos de difusión con otros dispositivos de la red. Para ello, configure las propiedades de red adecuadas (de forma predeterminada en el menú **Visualizar** →**Propiedades** →**Variables de red**) para la GVL.

NOTA: El tamaño máximo de una variable de red es de 255 bytes. Puede haber un número ilimitado de variables de red.

NOTA: Las variables declaradas en la GVL se inicializan antes que las variables locales de las POU.

GVL para constantes configurables (lista de parámetros) en bibliotecas

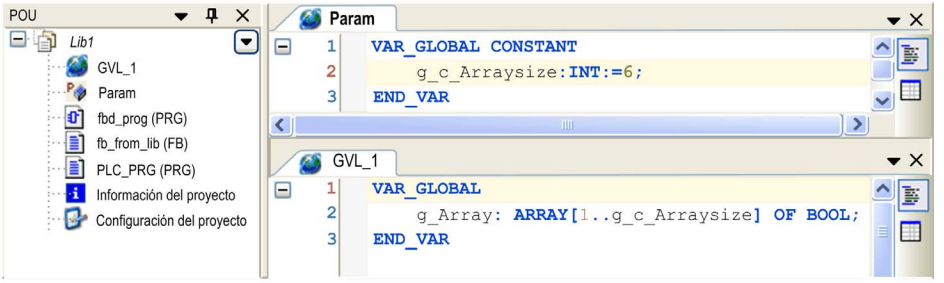
El valor de una constante global proporcionado mediante una biblioteca se puede sustituir por un valor definido por la aplicación. Para ello, la constante se debe declarar en una lista de parámetros de la biblioteca. A continuación, cuando se incluye la biblioteca en la aplicación, su valor se puede editar en la ficha **Lista de parámetros** del **Administrador de bibliotecas** de la aplicación. Consulte el ejemplo siguiente para ver una descripción paso a paso.

Gestión de las listas de parámetros

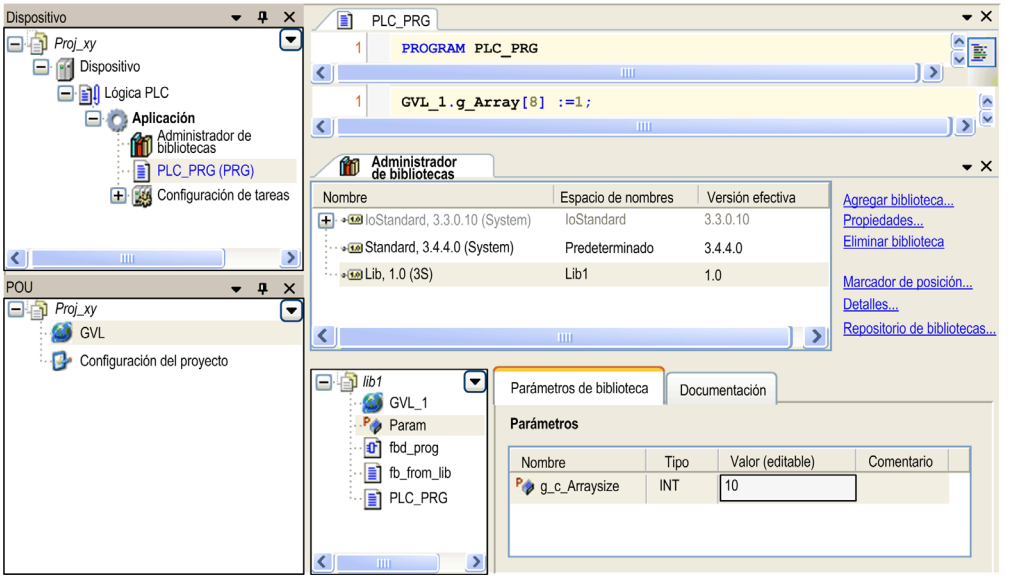
La biblioteca `lib1.library` proporciona una variable de matriz `g_Array`. El tamaño de la variable de matriz se define mediante la constante global `g_c_Arraysize`. La biblioteca está incluida en diversas aplicaciones, y cada una necesita un tamaño de matriz diferente. Por lo tanto, desea sobrescribir la constante global de la biblioteca con un valor específico de la aplicación.

Haga lo siguiente: al crear `lib1.library`, defina la constante global `g_c_Arraysize` en un tipo especial de lista de variables globales (GVL), denominada lista de parámetros. Para ello, ejecute el comando **Agregar objeto** y añada un objeto de lista de parámetros, que en este ejemplo se llama `Param`. En el editor de este objeto, que es igual que el de una GVL estándar, inserte la declaración de la variable `g_c_Arraysize`.

Lista de parámetros `Param` en la biblioteca `Lib1.library`



Editar parámetro `g_c_Arraysize` en el **Administrador de bibliotecas** de un proyecto



Seleccione la biblioteca en la parte superior del **Administrador de bibliotecas** para obtener el árbol de módulos. Seleccione `Param` para abrir la ficha **Parámetros de biblioteca** que muestra las declaraciones. Seleccione la celda de la columna **Valor (editable)** y utilice el espacio vacío para abrir un campo de edición. Introduzca el nuevo valor que desee para `g_c_Arraysize`. Este valor se aplicará al ámbito local actual de la biblioteca tras haber cerrado el campo de edición.

Lista de variables globales de red - GNVL

Descripción general

La funcionalidad **GNVL** sólo está disponible si se selecciona en el conjunto de características usado actualmente (**Opciones →Características →Conjunto predefinido de características**).

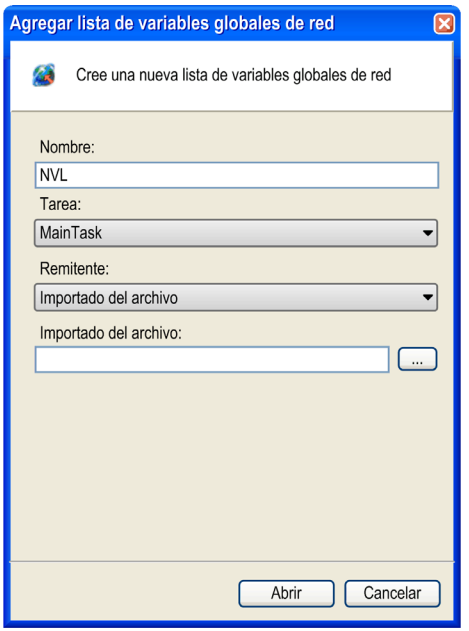
Una lista de variables globales de red (GNVL) se utiliza en la vista **Catálogo de software → Variables →Variables globales** y en el árbol **Aplicaciones** . Esta lista define variables, que se especifican como variables de red en otro dispositivo de la red.

NOTA: El tamaño máximo de una variable de red es de 255 bytes. Puede haber un número ilimitado de variables de red.

Por lo tanto, puede añadir un objeto GNVL a una aplicación si una GVL *(véase página 219)* con propiedades de red especiales (lista de variables de red) está disponible en uno de los otros dispositivos de red. Esto es independiente de si se ha definido en el mismo proyecto o en proyectos diferentes. Si se encuentran varias GVL apropiadas en el proyecto actual para la red actual, seleccione la GVL que desee de una lista de selección **Remitente** al añadir una GNVL por medio del cuadro de diálogo **Agregar objeto →Agregar lista de variables globales de red**. Las GVL de otros proyectos se deben importar tal como se describe en este capítulo.

Esto significa que cada GNVL del dispositivo actual (receptor) corresponde exactamente a una GVL en otro dispositivo (emisor).

Cuadro de diálogo **Agregar lista de variables globales de red**



Descripción de los elementos

Al añadir la GNVL, además de un **Nombre**, defina también una **Tarea**, responsable de la gestión de las variables de red.

Como alternativa a seleccionar directamente una GVL en **Remitente** de otro dispositivo, puede especificar un archivo de exportación de GVL *.GVL con la opción **Importado del archivo**. Este archivo GVL se ha generado anteriormente a partir de esa GVL de **Remitente** por medio del cuadro de diálogo *(véase SoMachine, Comandos de menú, Ayuda en línea)* **Ver** →**Propiedades** →**Vínculo con archivo**. En cualquier caso, esto es necesario si la GVL deseada se define dentro de otro proyecto. Con este fin, seleccione la opción **Importado del archivo** en la lista de selección **Remitente** y especifique la ruta del archivo en el campo de texto **Importado del archivo** (o haga clic en el botón **...** para utilizar el cuadro de diálogo estándar para la navegación por el sistema de archivos).

Puede modificar la configuración más tarde por medio del cuadro de diálogo *(véase SoMachine, Comandos de menú, Ayuda en línea)* **Ver** →**Propiedades** →**Configuración de red**.

El editor NVL *(véase página 428)* muestra una GNVL, pero no se puede modificar. Muestra el contenido actual de la GVL correspondiente. Si cambia la GVL básica, la GNVL se actualizará en consecuencia.

Se añade un comentario automáticamente en la parte superior de la parte de la declaración de una GNVL, que proporciona información sobre el emisor (ruta del dispositivo), el nombre de la GVL y el tipo de protocolo.

Ejemplo de lista de variables globales de red

Lista de variables globales de red

NVL32 [Device_B: Plc Logic: Application]

```
1  //This gobal variable list is received via the network.
2  //Sender: GVL321 [Device_B: Plc Logic: Application]
3  //Protocol: UDP
4
5  VAR_GLOBAL
6      iglobvar321:INT;
7      bglobvar321:BOOL;
8      strglobvar321:STRING;
9  END_VAR
```

NOTA: Sólo se transfieren a la aplicación remota las matrices cuyos límites están definidos por un literal o una constante. En este caso las expresiones constantes no están permitidas para la definición de límites. Ejemplo: `arrVar : ARRAY[0..g_iArraySize-1] OF INT ;` no se transfiere `arrVar : ARRAY[0..10] OF INT ;` se transfiere.

NOTA: Para obtener más información, consulte el capítulo *Comunicación de red (véase página 895)*.

Ejemplo de intercambio de variables de red simples

En el ejemplo siguiente, se establece un intercambio de variables de red simples. Se crea una lista de variables globales (GVL) en el controlador de emisor. Se crea la lista de variables globales de red (GNVL) correspondiente en el controlador de receptor.

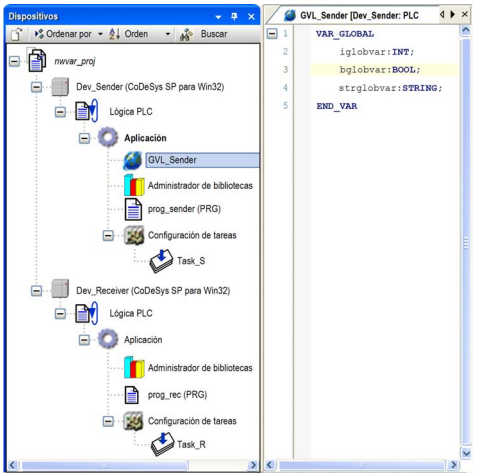
Lleve a cabo las preparaciones siguientes en un proyecto estándar, en el que estén disponibles un controlador de emisor **Dev_Sender** y un controlador de receptor **Dev_Receiver** en el árbol **Dispositivos**:

- Cree una POU (programa) **prog_sender** debajo del nodo **Aplicación** de **Dev_Sender**.
- Debajo del nodo **Configuración de tareas** de esta aplicación, añada la tarea **Task_S** que llama a **prog_sender**.
- Cree una POU (programa) **prog_rec** debajo del nodo **Aplicación** de **Dev_Receiver**.
- Debajo del nodo **Configuración de tareas** de esta aplicación, añada la tarea **Task_R** que llama a **prog_rec**.
  NOTA: Se deben configurar los 2 controladores en la misma subred de la red Ethernet.

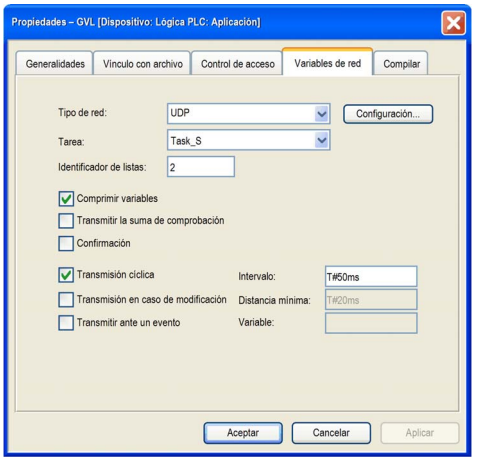
Definición de la GVL del emisor

Paso 1: Defina una lista de variables globales en el controlador de emisor:

Paso	Acción	Comentario
1	En **Catálogo de software** →**Assets View** →**POU**, seleccione el nodo **Aplicación** del controlador **Dev_Sender** y haga clic en el signo más de color verde. Seleccione el comando **Lista de variables globales...**	Aparecerá el cuadro de diálogo **Agregar lista de variables globales**.
2	Introduzca el **Nombre** `GVL_Sender` y haga clic en **Agregar** para crear una nueva lista de variables globales.	El nodo **GVL_Sender** aparece debajo del nodo **Aplicación** en el árbol **Aplicaciones** y el editor se abre en medio de la pantalla de SoMachine.

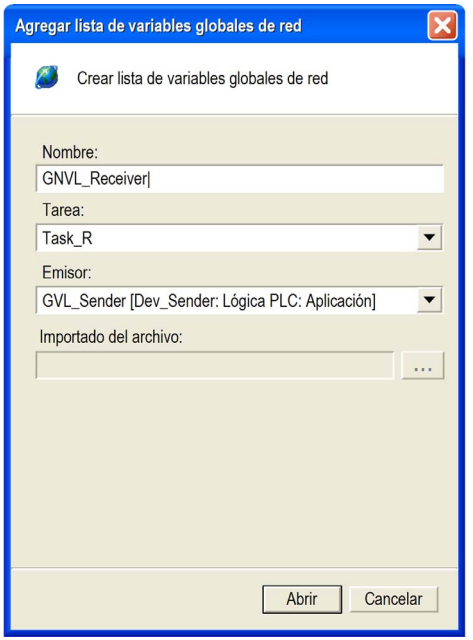
3	En el editor, introduzca las definiciones de variable siguientes:
	`VAR_GLOBAL`
	`iglobvar:INT;`
	`bglobvar:BOOL;`
	`strglobvar:STRING;`
	`END_VAR`
	 
		–

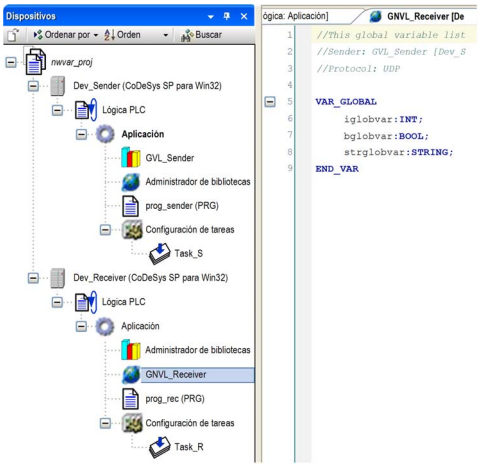
Paso 2: Defina las propiedades de la red de la GVL de emisor:

Paso	Acción	Comentario
1	En el árbol **Aplicaciones** , seleccione el nodo **GVL_Sender**, haga clic en el signo más de color verde y ejecute el comando **Propiedades...**	Aparecerá el cuadro de diálogo **Propiedades - GVL_Sender**.
2	Abra la ficha **Variables de red** y configure los parámetros, tal como se muestran en el gráfico: 	–
3	Haga clic en **Aceptar**.	Se cerrará el cuadro de diálogo y se establecerán las propiedades de la red de la GVL.

Definición de la GNVL del receptor

Paso 1: Defina una lista de variables globales de red en el controlador de receptor:

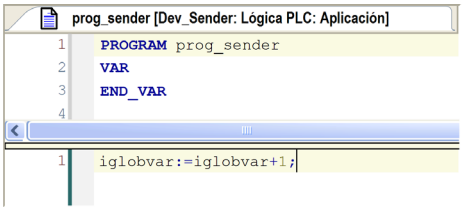
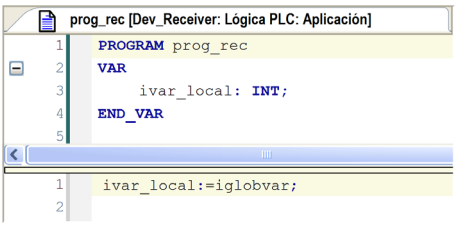
Paso	Acción	Comentario
1	En el árbol **Aplicaciones** , seleccione el nodo **Aplicación** del controlador **Dev_Receiver**, haga clic en el signo más de color verde y ejecute el comando **Lista de variables globales de red...**	Aparecerá el cuadro de diálogo **Agregar lista de variables globales de red**.
2	Configure los parámetros tal como se muestran en el gráfico. 	Esta lista de variables globales de red es la homóloga de la GVL definida para el controlador de emisor.

Paso	Acción	Comentario
3	Haga clic en **Abrir**.	Se cerrará el cuadro de diálogo y aparecerá **GNVL_Receiver** debajo del nodo **Aplicación** del controlador **Dev_Receiver**:  Esta GNVL contiene automáticamente las mismas declaraciones de variables que **GVL_Sender**.

Paso 2: Compruebe o modifique la configuración de red de la GNVL:

Paso	Acción	Comentario
1	En el árbol **Dispositivos**, haga clic con el botón derecho en el nodo **GNVL_Receiver** y seleccione el comando **Propiedades...**	Aparecerá el cuadro de diálogo **Propiedades - GNVL_Receiver**.
2	Abra la ficha **Configuración de red**.	–

Paso 3: Pruebe el intercambio de las variables de red en modalidad online:

Paso	Acción	Comentario
1	Debajo del nodo **Aplicación** del controlador **Dev_Sender**, haga doble clic en la POU **prog_sender**.	Se abrirá el editor para **prog_sender** en la parte de la derecha.
2	Introduzca el código siguiente para la variable `iglobvar`: 	–
3	Debajo del nodo **Aplicación** del controlador **Dev_Receiver**, haga doble clic en la POU **prog_rec**.	Se abrirá el editor para **prog_rec** en la parte de la derecha.
4	Introduzca el código siguiente para la variable `ivar_local`: 	–
5	Inicie sesión con las aplicaciones de emisor y de receptor de la misma red e inícielas.	La variable `ivar_local` del receptor obtiene los valores de `iglobvar` tal como se muestra actualmente en el emisor.

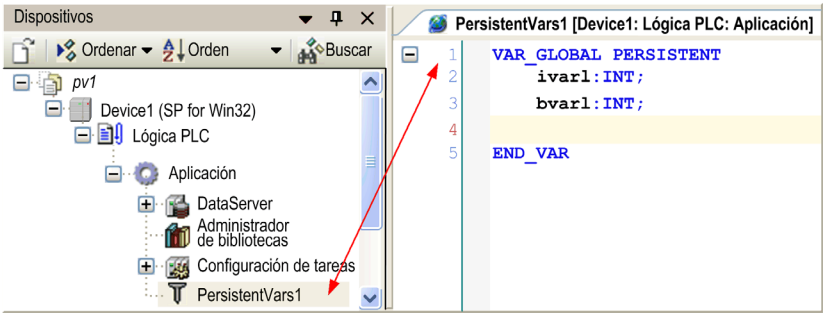
Variables persistentes

Descripción general

Este objeto es una lista de variables globales que sólo contiene las variables persistentes de una aplicación. Por lo tanto, debe asignarse a una aplicación. Para ello, debe insertarse en el árbol **Aplicaciones** seleccionando el nodo correspondiente, haciendo clic en el signo más de color verde y seleccionando **Añadir otros objetos** →**Variables persistentes...**

Edite una lista de variables persistentes en el editor GVL *(véase página 425)*.
`VAR_GLOBAL PERSISTENT RETAIN` ya está predefinido en la primera línea.

Lista de variables persistentes



Las variables persistentes sólo se reinicializan con **Reset (origen) <aplicación>**. Para obtener más información, consulte la descripción de las variables remanentes *(véase página 596)*.

También puede consultar la descripción de los comandos especiales para gestionar variables persistentes *(véase SoMachine, Comandos de menú, Ayuda en línea)*.

Adición y declaración de variables remanentes

Al añadir variables a una aplicación, puede declarar algunas de estas como variables remanentes. Las variables remanentes pueden conservar sus valores en caso de cortes de alimentación, reinicios, restablecimientos y descargas de programas de aplicación. Existen varios tipos de variables remanentes, declaradas individualmente como de tipo "retain" o "persistent", o en combinación como "retain-persistent".

Para obtener información sobre el tamaño de memoria reservado para las variables Retain y Persistent en los diferentes controladores, consulte la *Guía de programación* del controlador que esté utilizando.

Para añadir una lista de variables globales denominada **Variables persistentes** a una aplicación, haga lo siguiente:

Paso	Acción
1	Seleccione el nodo de aplicación correspondiente en el árbol **Aplicaciones**, haga clic en el signo más de color verde y seleccione **Añadir otros objetos →Variables persistentes...** Como alternativa, puede hacer clic con el botón derecho en el nodo de aplicación y ejecutar el comando **Agregar objeto →Variables persistentes...**
2	En el cuadro de diálogo **Agregar variables persistentes**, escriba un nombre para esta lista en el cuadro de texto **Nombre**.
3	Haga clic en **Agregar**. **Resultado:** Se crea un nodo de variables persistentes en el árbol **Aplicaciones**. Para ver un ejemplo, consulte el apartado *Descripción general* en este capítulo.

Archivo externo

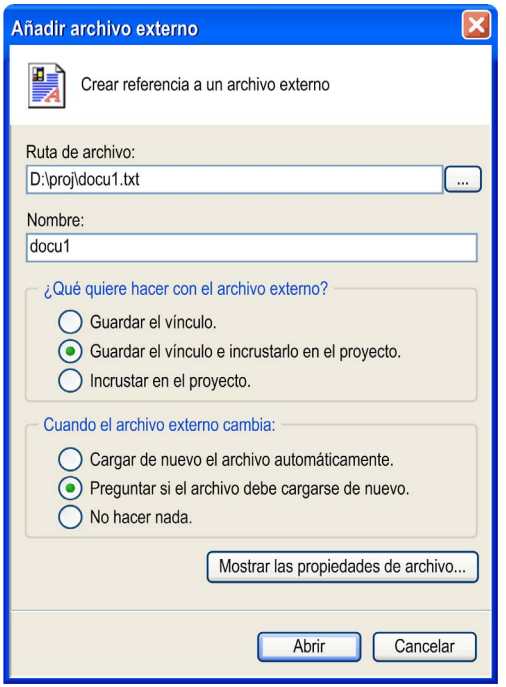
Descripción general

La funcionalidad **Archivo externo** sólo está disponible si se selecciona en el conjunto de características usado actualmente (**Opciones →Características →Conjunto predefinido de características**).

Para agregar un archivo externo al nodo **Global** del árbol **Aplicaciones** o del árbol **Herramientas**, seleccione el nodo **Global**, haga clic en el signo más de color verde y ejecute los comandos **Añadir otros objetos →Archivo externo...**

Haga clic en el botón **...** para abrir el cuadro de diálogo y examinar un archivo. La ruta de este archivo se especifica en el cuadro de texto **Ruta de archivo**. En el cuadro de texto **Nombre**, el nombre del archivo elegido se especifica automáticamente sin extensión. Puede editar este campo para definir otro nombre para el archivo con el cual debe gestionarse en el proyecto.

Cuadro de diálogo **Añadir archivo externo**:



Descripción de la sección ¿Qué desea hacer con el archivo externo? del cuadro de diálogo

Seleccione una de las opciones siguientes:

Opción	Descripción
Guardar el vínculo.	El archivo estará disponible en el proyecto sólo si está disponible en la ruta de enlace definida.
Guardar el vínculo e incrustarlo en el proyecto.	Se guardará una copia del archivo internamente en el proyecto, pero también se recuperará el vínculo al archivo externo. Mientras el archivo externo esté disponible según se ha definido, las opciones de actualización definidas se implementarán en consonancia. En caso contrario, sólo estará disponible la versión del archivo almacenada en el proyecto.
Incrustar en el proyecto.	Sólo se almacenará una copia del archivo en el proyecto. No habrá otras conexiones con el archivo externo.

Descripción de la sección Cuando el archivo externo cambia del cuadro de diálogo

Si el archivo externo está vinculado al proyecto, puede seleccionar además una de las opciones:

Opción	Descripción
Cargar de nuevo el archivo automáticamente.	El archivo se actualiza en el proyecto en cuanto se modifica externamente.
Preguntar si el archivo debe cargarse de nuevo.	Aparece un cuadro de diálogo en cuanto el archivo se modifica externamente. Puede decidir si el archivo se actualizará también en el proyecto.
No hacer nada.	El archivo permanece sin cambios en el proyecto, aunque se modifique externamente.

Descripción de los botones

Botón	Descripción
Mostrar las propiedades de archivo...	Este botón abre el cuadro de diálogo estándar de propiedades de un archivo. Este cuadro de diálogo también aparece al seleccionar el objeto de archivo en el árbol **Aplicaciones** o el árbol **Herramientas** y ejecutar el comando **Propiedades**. En la ficha **Archivo externo** de este cuadro de diálogo, puede ver y modificar las propiedades.
Abrir	Después de completar la configuración, haga clic en el botón **Abrir** para añadir el archivo al nodo **Global** del árbol **Aplicaciones** o del árbol **Herramientas**. Se abre en una herramienta definida de forma predeterminada para el formato del archivo en cuestión.

Lista de texto

Descripción general

La funcionalidad **Lista de texto** sólo está disponible si se selecciona en el conjunto de características usado actualmente (**Opciones →Características →Conjunto predefinido de características**).

Una lista de texto es un objeto gestionado globalmente en el nodo **Global** del árbol **Aplicaciones** o asignado a una aplicación en el árbol **Aplicaciones**.

Sirve para lo siguiente:

- soporte en varios idiomas para textos estáticos *(véase página 235)* y dinámicos *(véase página 236)*, e información sobre herramientas en visualizaciones y en la gestión de alarmas
- intercambio de texto dinámico

Las listas de texto pueden exportarse y (re)importarse *(véase página 239)*. La exportación es necesaria si debe proporcionarse un archivo de idioma en formato XML para una visualización de destino, pero también puede ser útil para traducciones.

Posibles formatos de listas de texto:

- texto
- XML

Puede activar el soporte de Unicode *(véase página 239)*.

Cada lista de texto se define de forma exclusiva mediante su espacio de nombre. Contiene cadenas de texto referenciadas de forma exclusiva en la lista mediante un identificador (ID, compuesto de una secuencia de caracteres) y un identificador de idioma. La lista de texto que debe utilizarse se especifica al configurar el texto de un elemento de visualización.

En función del idioma definido en la visualización, se mostrará en modalidad online la cadena de texto correspondiente. El idioma empleado en una visualización se cambia mediante la entrada **Cambio de idioma**. Esto se lleva a cabo mediante una acción de ratón que ha configurado en el elemento de visualización en cuestión. Cada lista de texto debe contener al menos un idioma predeterminado y, opcionalmente, otros idiomas que desee definir. Si no se encuentra ninguna entrada que coincida con el idioma establecido actualmente en SoMachine, se utilizará el idioma predeterminado de la lista de texto. Cada texto puede contener definiciones de formato *(véase página 239)*.

Estructura básica de una lista de texto

Identificador (Índice)	Predeterminado	<Idioma 1>	<Idioma 2>	.... <Idioma n>
<cadena exclusiva de caracteres>	<texto abc en el idioma predeterminado>	<texto abc en el idioma 1>	<texto abc en el idioma 2>	...
<cadena exclusiva de caracteres>	<texto xyz en el idioma predeterminado>	<texto xyz en el idioma 1>	<texto xyz en el idioma 2>	...

Tipos de lista de texto

Hay dos tipos de texto que se pueden utilizar en elementos de visualización y, por consiguiente, hay dos tipos de listas:

- `GlobalTextList` para textos estáticos
- lista de texto para textos dinámicos

GlobalTextList para textos estáticos

GlobalTextList es una lista de texto especial en la que los identificadores de las diferentes entradas de texto se gestionan implícitamente y no son editables. Asimismo, la lista no puede eliminarse. Sin embargo, puede exportar la lista, editarla externamente y, a continuación, volver a importarla.

El texto estático en una visualización, a diferencia del texto dinámico, no puede intercambiarse por una variable en modalidad online. La única opción para cambiar el idioma de un elemento de visualización es mediante la entrada **Cambio de idioma**. Un texto estático se asigna a un elemento de visualización mediante la propiedad **Texto** o **Información sobre herramientas** en la categoría **Textos**. Cuando se define el primer texto estático en un proyecto, se añade un objeto de lista de texto denominado **GlobalTextList** al nodo **Global** del árbol **Aplicaciones**. Contiene la cadena de texto definida que se encuentra en la columna **Predeterminado**, y se asigna automáticamente un número entero como identificador de texto. Por cada texto estático creado posteriormente, el número de identificador se incrementa y se asigna al elemento de visualización.

Si se inserta un texto estático en un elemento de visualización (por ejemplo, si en un rectángulo de la categoría de propiedad **Textos**, se especifica la cadena **Text Example**), se buscará ese texto en **GlobalTextList**.

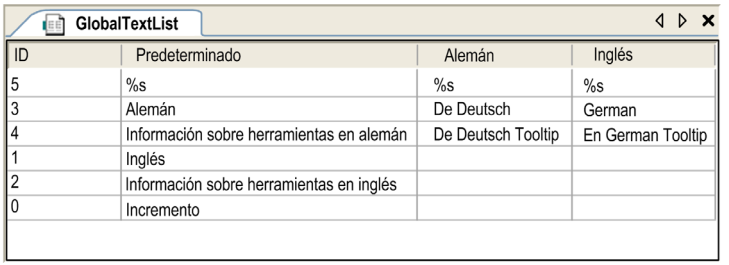
- Si se encuentra el texto (por ejemplo, **ID 4711**, **Text Example**), el valor de elemento **4711** de **TextId** se asignará a una variable interna. Esto establece la relación entre el elemento y la línea correspondiente en **GlobalTextList**.
- Si no se encuentra el texto, se inserta una nueva línea en **GlobalTextList** (por ejemplo, **ID 4712, Text Example**). En el elemento, se asignará el valor **4712** a la variable interna.

NOTA: Si todavía no existe, puede crear una lista global de texto explícitamente mediante el comando **Crear la lista global de texto**.

Si ha exportado, editado y reimportado **GlobalTextList**, se valida en función de si los identificadores todavía coinciden con los que se utilizan en la configuración de los elementos de visualización respectivos. Si es necesario, se realizará una actualización implícita de los identificadores utilizados en la configuración.

Ejemplo de GlobalTextList

Crear la lista global de texto



GlobalTextList

ID	Predeterminado	Alemán	Inglés
5	%s	%s	%s
3	Alemán	De Deutsch	German
4	Información sobre herramientas en alemán	De Deutsch Tooltip	En German Tooltip
1	Inglés		
2	Información sobre herramientas en inglés		
0	Incremento		

Lista de texto para textos dinámicos

El texto dinámico puede cambiarse dinámicamente en modalidad online. El índice de texto (**ID**), que es una cadena de caracteres, debe ser exclusivo en la lista de texto. A diferencia de **GlobalTextLists**, debe definirlo. A diferencia también de **GlobalTextList**, puede crear listas de texto explícitamente para texto dinámico seleccionando el nodo **Global**, haciendo clic en el signo más de color verde y ejecutando el comando **Añadir otros objetos →Lista de texto...**

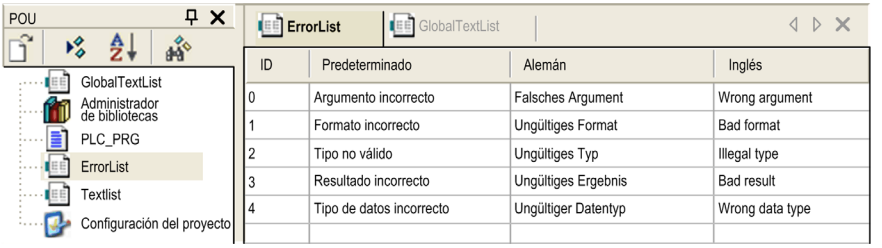
Las listas de texto dinámico disponibles actualmente se ofrecen al configurar un elemento de visualización mediante la propiedad **Dynamic texts**/**Lista de texto**. Si especifica un nombre de lista de texto combinado con el índice de texto (**ID**), que puede especificarse directamente o mediante una variable de proyecto que define la cadena de ID, el texto actual puede modificarse en modalidad online.

Si es necesario, debe exportarse una lista de texto dinámico como archivo de idioma para cambiar de idioma en una visualización de destino. Especifique la ruta del archivo en **Visualization Options**. Al igual que **GlobalTextList**, una lista de texto dinámico también puede exportarse para su edición externa y volver a importarse. A diferencia de **GlobalTextList**, al importar listas de texto dinámico no se produce ninguna comprobación ni actualización automáticas de los identificadores.

AVISO
MODIFICACIÓN ACCIDENTAL DE IDENTIFICADORES No modifique los identificadores al editar la lista exportada. **El incumplimiento de estas instrucciones puede causar daño al equipo.**

Ejemplo de lista de texto dinámico denominada ErrorList

Ejemplo: **ErrorList**



ID	Predeterminado	Alemán	Inglés
0	Argumento incorrecto	Falsches Argument	Wrong argument
1	Formato incorrecto	Ungültiges Format	Bad format
2	Tipo no válido	Ungültiges Typ	Illegal type
3	Resultado incorrecto	Ungültiges Ergebnis	Bad result
4	Tipo de datos incorrecto	Ungültiger Datentyp	Wrong data type

Ejemplo detallado

En este ejemplo se explica cómo configurar un elemento de visualización, que muestra el mensaje correspondiente cuando se detecta un error en una aplicación que procesa eventos de error identificados mediante ID numéricos asignados a una variable entera `ivar_err`.

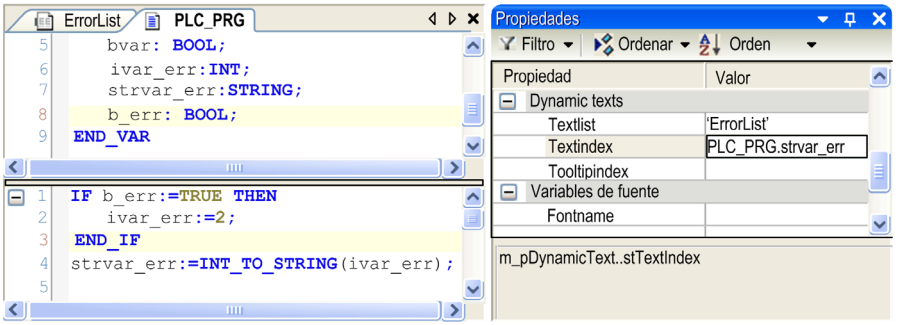
Proporcione una lista de texto dinámico denominada **ErrorList** en la que el texto de los mensajes para los ID 0 a 4 se definan en los idiomas **Alemán**, **Inglés** y **Predeterminado**:

ID	Predeterminado	Alemán	Inglés
0	Error 0. Haga lo siguiente...	Fehler 0. Führen Sie folge...	Error 0. Do the...
1	Error 1. Cierre...	Fehler 1. Schließen Sie...	Error 1. Close the...
2	Error 2. Realice un...	Fehler 2. Führen Sie einen...	Error 2. Perform...
3	Error 3. Intente...	Fehler 3. Versuchen Sie...	
4	Error 4. Inicie...	Fehler 4. Starten Sie...	

Para utilizar los ID de error en la configuración de la visualización, defina una variable STRING, como por ejemplo `strvar_err`. Para asignar el valor entero de `ivar_err` a `strvar_err`, utilice `strvar_err:=INT_TO_STRING(ivar_err);`.

`strvar_err` puede especificarse como parámetro **Textindex** en la configuración de las propiedades **Dynamic texts** de un elemento de visualización. El elemento mostrará el mensaje apropiado en modalidad online.

El ejemplo siguiente es para procesar el ID de error mediante variables de proyecto y la configuración de un elemento de visualización (**Propiedades**), que debe mostrar el mensaje apropiado:



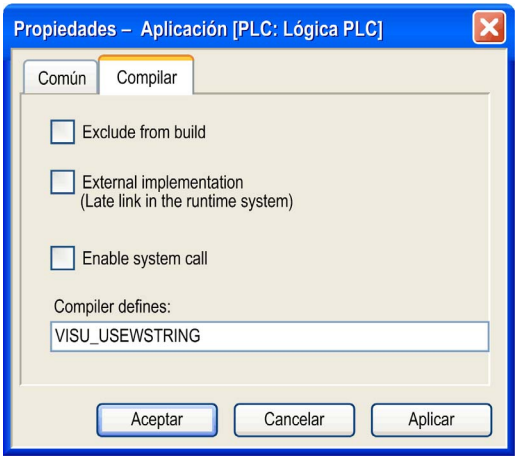
Creación de una lista de texto

- Para crear una lista de texto para textos dinámicos *(véase página 236)*, añada un objeto **Lista de texto** al proyecto en el árbol **Aplicaciones**. Para crear una lista de texto específica de la aplicación, seleccione un nodo de la aplicación. Para crear una lista global de texto, seleccione el nodo **Global**. A continuación, haga clic en el signo más de color verde del nodo seleccionado y ejecute el comando **Añadir otros objetos** →**Lista de texto...** Cuando haya especificado un nombre de lista y haya confirmado el cuadro de diálogo **Add Textlist**, la nueva lista se insertará debajo del nodo seleccionado y se abrirá una vista del editor de listas de texto.
- Para obtener una lista de texto para textos estáticos *(véase página 235)* (**GlobalTextList**), asigne texto a la propiedad **Texto** en la categoría **Textos** de un objeto de visualización para obtener la vista creada automáticamente, o genérela explícitamente mediante el comando **Crear la lista global de texto**.
- Para abrir una lista de texto existente para su edición, seleccione el objeto de lista en el árbol **Aplicaciones** o el nodo **Global** del árbol **Aplicaciones**. Haga clic con el botón derecho en el nodo de lista de texto y ejecute el comando **Modificar objeto**, o haga doble clic en el nodo de lista de texto. Consulte la tabla *Estructura básica de una lista de texto* para saber cómo se estructura una lista de texto.
- Para añadir un nuevo texto predeterminado en una lista de texto, utilice el comando **Insertar texto** o edite el campo respectivo en la línea vacía de la lista. Para editar un campo en una lista de texto, haga clic en el campo para seleccionarlo y, a continuación, vuelva a hacer clic en el campo o pulse la barra espaciadora para obtener un cuadro de edición. Introduzca los caracteres deseados y cierre el cuadro de edición pulsando RETORNO.

Soporte de formato Unicode

Para utilizar el formato Unicode, active la opción correspondiente en el **Gestor de visualización**. Además, establezca una directiva de compilación especial para la aplicación: seleccione la aplicación en el árbol **Dispositivos**, abra el cuadro de diálogo **Propiedades**, ficha **Compilación**. En el campo **Definiciones de compilador**, especifique `VISU_USEWSTRING`.

Cuadro de diálogo con definición de compilador



Exportación e importación de listas de texto

Las listas de texto estático y dinámico pueden exportarse como archivos en formato CSV. Los archivos exportados también pueden utilizarse para añadir texto externamente; por ejemplo, por parte de un traductor externo. Sin embargo, sólo pueden volver a importarse los archivos disponibles en formato de texto (**.csv*).

Consulte la descripción de los respectivos comandos de lista de texto *(véase SoMachine, Comandos de menú, Ayuda en línea)*.

Especifique la carpeta en la que deben guardarse los archivos de exportación en el cuadro de diálogo **Archivo** →**Configuración del proyecto** →**Visualización**.

Formato del texto

El texto puede contener definiciones de formato `%s`, `%d`…), que permiten incluir los valores actuales de las variables en un texto. Para ver las cadenas de formato posibles, consulte el apartado *Visualización* de la ayuda online de SoMachine.

Cuando utilice texto con cadenas de formato, la sustitución se realiza en el orden siguiente:

- Se busca la cadena de texto que debe utilizarse por nombre de lista e ID.
- Si el texto contiene definiciones de formato, se sustituyen por el valor de la variable correspondiente.

Entrega posterior del texto traducido

Al insertar *GlobalTextList.csv* en el directorio utilizado para cargar archivos de texto, se permite la integración posterior de texto traducido. Cuando se inicia el proyecto de arranque, el firmware detecta que hay disponible un archivo adicional. El texto se compara con el de los archivos de listas de texto existentes. Los textos nuevos y modificados se aplican a los archivos de listas de texto. Los archivos de listas de texto actualizados se aplicarán en el siguiente inicio.

Componentes de lista para la introducción de texto

Mediante el cuadro de diálogo **Herramientas** →**Opciones** →**Visualización**, puede especificar un archivo de plantilla de texto. Todos los textos de la columna **Predeterminado** de este archivo se copiarán en una lista, que se utilizará para la funcionalidad **List Components**. Puede utilizarse un archivo de plantilla creado anteriormente mediante el comando **Exportar**.

Operaciones multiusuario

Mediante el uso del control de código de origen, es posible que varios usuarios trabajen simultáneamente en el mismo proyecto. Si más de un usuario modifica texto estático en elementos de visualización, esto provocará modificaciones en **GlobalTextList** (consulte **GlobalTextList** *(véase página 235)*). En este caso, puede que los ID de texto no sean coherentes con los elementos de visualización. Utilice los siguientes métodos de detección y corrección de errores:

- Utilice el comando **Comprobar los identificadores de texto de visualización** para detectar estos errores en las visualizaciones.
- Utilice el comando **Actualizar los identificadores de texto de visualización** para corregir automáticamente estos errores. Tanto las visualizaciones afectadas como **GlobalTextList** deben tener permiso de escritura.

Uso de listas de texto para modificar el idioma en las visualizaciones

Si hay disponible una lista de texto adecuada, es decir, una lista de texto que define versiones en varios idiomas para un texto, el idioma utilizado para el texto de una visualización puede cambiarse en modalidad online mediante una entrada en el elemento de visualización. Las propiedades **Dynamic Texts** del elemento deben especificar la lista de texto que debe utilizarse, y debe configurarse la acción de entrada **OnMouse**, **Cambio de idioma**, para especificar el idioma que debe emplearse después de realizar la acción del ratón.

NOTA: Debe especificarse el idioma exactamente con la cadena que se muestra en el encabezado de columna de la lista de texto respectiva.

Colección de imágenes

Descripción general

La funcionalidad **Colección de imágenes** sólo está disponible si se selecciona en el conjunto de características usado actualmente (**Opciones →Características →Conjunto predefinido de características**).

Las colecciones de imágenes son tablas que definen la ruta de archivo, una vista previa y un ID de cadena para cada imagen. Si se especifica el ID y (para un acceso exclusivo) adicionalmente el nombre de archivo de imagen, se puede hacer referencia a la imagen, por ejemplo, cuando se inserta en una visualización (configuración de las propiedades de un elemento de imagen; consulte *Uso de imágenes gestionadas en colecciones de imágenes* *(véase página 242)*).

NOTA: Se recomienda reducir el tamaño de un archivo de imagen tanto como sea posible antes de añadirlo a una colección de imágenes. De lo contrario, el tamaño del proyecto y la carga y almacenamiento de las aplicaciones de visualización, incluidas las imágenes, pueden resultar muy grandes.

Estructura de una colección de imágenes

Ejemplo de una colección de imágenes

Elemento	Descripción
ID	ID de cadena (por ejemplo: **logo**, **y_icon**, **2**). Para obtener una referencia exclusiva de una imagen, se debe combinar el nombre de lista de imágenes y el ID (por ejemplo: **List1.basic_logo**).
Nombre de archivo	Ruta del archivo de imagen (por ejemplo: *C:\programs\images\logo.bmp*).
Imagen	Vista previa de la imagen.

Creación y edición de una colección de imágenes

Un proyecto puede contener varias colecciones de imágenes.

Si todavía no hay una colección disponible en un proyecto, se creará automáticamente una colección de imágenes con el nombre predeterminado **GlobalImagePool** en cuanto añada el primer elemento de imagen y especifique un ID (ID estático) para la imagen correspondiente en las propiedades de los elementos visuales. Se inserta una entrada para la imagen. **GlobalImagePool** es una colección global en la que se busca en primer lugar cuando se debe utilizar un archivo de imagen. Además de esta colección, se pueden utilizar colecciones adicionales cuyo nombre se ha asignado individualmente.

Para crear colecciones de imágenes manualmente, haga lo siguiente: **GlobalImagePool** se crea con el comando **Image Pool Editor Commands** →**Crear la colección global de imágenes**. Puede insertar los demás objetos de colección debajo de un nodo de aplicación o debajo del nodo **Global** del árbol **Aplicaciones** haciendo clic en el signo más de color verde y ejecutando los comandos **Añadir otros objetos** →**Colección de imágenes...** En el cuadro de diálogo **Add Image Pool**, defina un **Nombre** para la colección.

Para añadir manualmente una imagen a una colección, utilice el comando **Insertar archivo de imagen** o cumplimente la tabla de la colección manualmente. En este último caso, seleccione el campo de ID de la primera línea vacía de la tabla de la colección, pulse la barra espaciadora para abrir un cuadro de edición y especifique un ID (cadena). El ID pasará a ser exclusivo automáticamente. A continuación, coloque el cursor en el campo de nombre **Archivo**, pulse la barra espaciadora y haga clic en el botón **...** para abrir el cuadro de diálogo **Selección de imagen**. Aquí puede especificar la ruta del archivo de imagen que desee.

Uso de imágenes gestionadas en colecciones de imágenes

Si el ID de la imagen que se debe utilizar se especifica en múltiples colecciones de imágenes:

- Orden de búsqueda: Si selecciona una imagen gestionada en **GlobalImagePool**, no es necesario especificar el nombre de colección. El orden de búsqueda de las imágenes corresponde al de las variables globales:
  1. **GlobalImagePool**
  2. colecciones de imágenes asignadas a la aplicación activa
  3. colecciones de imágenes del nodo **Global** del árbol **Aplicaciones** además de **GlobalImagePool**
  4. colecciones de imágenes en bibliotecas
- Acceso exclusivo: Puede llamar directamente a la imagen que desee añadiendo el nombre de colección de imágenes antes del ID, con la siguiente sintaxis: <nombre de colección>.<ID de imagen> (para ver un ejemplo, consulte `imagepool1.drive_icon` en el gráfico anterior).

Uso de una imagen en un elemento de visualización de tipo imagen

Al insertar un elemento de imagen en una visualización, puede definirlo como imagen estática o imagen dinámica. La imagen dinámica se puede cambiar en modalidad online según el valor de una variable de proyecto:

Imágenes estáticas:

En la configuración del elemento (propiedad **ID estático**), especifique el ID de imagen o el nombre de colección de imágenes + ID de imagen. Tenga en cuenta las observaciones sobre el orden de búsqueda y el acceso exclusivo del párrafo anterior.

Imágenes dinámicas:

En la configuración del elemento (propiedad **Variable ID de mapa de bits**), especifique la variable que define el ID; por ejemplo, `PLC_PRG.imagevar`.

Uso de una imagen para el fondo de visualización

En la definición del fondo de una visualización, puede definir que se muestre una imagen como fondo de la visualización. Como se ha descrito anteriormente, el archivo de imagen se puede especificar para un elemento de visualización mediante el nombre de la colección de imágenes y el nombre del archivo de imagen.

Sección 6.4
Aplicación

Aplicación

Descripción general

Una aplicación es un conjunto de objetos que se necesitan para ejecutar una instancia determinada del programa de controlador en un dispositivo de hardware concreto (controlador). Con esta finalidad, los objetos independientes gestionados en el nodo **Global** del árbol **Aplicaciones** se instancian y se asignan a un dispositivo. Esto se corresponde con el concepto de programación orientada a objetos. Sin embargo, también puede utilizar las POU puramente específicas de la aplicación.

Una aplicación se representa mediante un objeto de aplicación en el árbol **Aplicaciones**. Debajo de una entrada de aplicación, inserte los objetos que definen el conjunto de recursos de la aplicación.

Hay una aplicación disponible para cada controlador. No se pueden añadir más aplicaciones.

Una parte de cada aplicación es **Configuración de tareas**, que controla la ejecución de un programa (instancias de POU o POU específicas de la aplicación). Además, puede tener asignados objetos de recurso como listas de variables globales, entre otros. Estos, a diferencia de los gestionados en el nodo **Global** del árbol **Aplicaciones**, solamente los puede utilizar la aplicación en cuestión y sus subobjetos. Para ver las reglas, consulte la descripción de la organización y configuración de objetos en el árbol **Dispositivos** *(véase página 40)*.

Consideraciones

Al iniciar sesión con una aplicación en un dispositivo de destino (controlador o destino de simulación), se efectúan dos comprobaciones: ¿qué aplicación está en el controlador en ese momento? ¿Se corresponden los parámetros de la aplicación del controlador con los de la aplicación de SoMachine? Los correspondientes mensajes indican las discrepancias y ofrecen algunas formas de continuar en este caso. También tiene la posibilidad de suprimir la aplicación del controlador. Consulte la descripción del comando *Inicio de sesión* *(véase página 254)* para obtener más detalles.

Capítulo 7
Configuración de tareas

Contenido de este capítulo

Este capítulo contiene los siguiente apartados:

Configuración de tareas

Descripción general

La **Configuración de tareas** define 1 o varias tareas para controlar el procesamiento de un programa de aplicación.

Es un objeto de recurso para una aplicación *(véase página 244)*. Se tiene que insertar en el árbol **Aplicaciones** bajo un nodo de aplicación. Una tarea puede llamar a una POU de un programa específico de la aplicación que está sólo disponible en el árbol **Aplicaciones**, bajo la aplicación. También puede llamar a un programa que se gestiona en el nodo **Global** del árbol **Aplicaciones**. En este último caso, la aplicación instanciará el programa que está disponible a nivel global.

Puede editar una configuración de tareas en el **editor de tarea** *(véase página 453)*.

En la modalidad online, el **editor de tarea** proporciona una vista de supervisión que ofrece información sobre los ciclos, tiempos de ciclo y estado de tarea.

Como funcionalidad adicional de la configuración de tareas, la vista de supervisión, si es compatible con el dispositivo, permite un análisis dinámico de las POU que están controladas por una tarea. Proporciona información acerca de los tiempos de ciclo, la cantidad de llamadas de bloques de funciones y las líneas de código no utilizadas.

Adición de tareas

Introducción

Puede añadir tareas a la aplicación mediante el árbol **Aplicaciones**.

Procedimiento

Paso	Acción
1	En el árbol **Aplicaciones**, seleccione el nodo **Configuración de tareas**, haga clic en el signo más de color verde y ejecute el comando **Tarea...** Como alternativa, puede hacer clic con el botón derecho en el nodo **Configuración de tareas** y seleccionar **Agregar objeto** →**Tarea...** en el menú contextual. **Resultado:** Se abre el cuadro de diálogo **Agregar tarea**.
2	En el cuadro de diálogo **Agregar tarea**, escriba un nombre en el cuadro de texto **Nombre:**. **Nota:** El nombre no puede contener espacios ni tener más de 32 caracteres.
3	Haga clic en **Agregar**.

Capítulo 8
Gestión de aplicaciones

Contenido de este capítulo

Este capítulo contiene las siguientes secciones:

Sección 8.1
Información general

Introducción

Introducción

Para ejecutar una aplicación, primero es necesario conectar el PC al controlador y descargar la aplicación al controlador.

NOTA: Debido a la limitación de tamaño de la memoria, algunos controladores no pueden almacenar el origen de la aplicación, sino sólo una aplicación compilada que se ejecute. Esto significa que no se podrá cargar el origen de la aplicación del controlador a un PC.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

- Asegúrese de haber introducido la designación o la dirección del dispositivo correctas en el cuadro de diálogo **Configuración de comunicación** al descargar una aplicación.
- Asegúrese de que las protecciones y las etiquetas de la máquina están colocadas con el fin de evitar el funcionamiento imprevisto de la máquina y riesgos de daños personales o materiales.
- Lea y tenga en cuenta toda la documentación para el usuario del software y los dispositivos relacionados, así como la documentación referente al funcionamiento de la máquina o el equipo.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

Condiciones previas

Compruebe que la aplicación reúne las condiciones siguientes antes de descargarla en el controlador:

- La ruta activa está definida para el controlador correcto.
- La aplicación que se desea descargar está activa.
- La aplicación no tiene errores de compilación.

Aplicación de arranque

La aplicación de arranque es la que se ejecuta al iniciar el controlador. Esta aplicación se almacena en la memoria del controlador. Para configurar la descarga de la aplicación de arranque, haga clic con el botón derecho en el nodo **Aplicación** del árbol **Aplicaciones** y seleccione el comando **Propiedades**.

Al final de una descarga correcta de una nueva aplicación, aparecerá un mensaje para preguntarle si desea crear la aplicación de arranque.

Puede crear una aplicación de arranque manualmente de las siguientes maneras:

- En modalidad offline: Haga clic en **En línea** →**Crear aplicación de inicio** para guardar esta aplicación en un archivo.
- En modalidad online, con el controlador en modalidad de detención: Ejecute el comando *(véase SoMachine, Comandos de menú, Ayuda en línea)* **En línea** →**Crear aplicación de inicio** para descargar la aplicación de arranque en el controlador.

Sección 8.2
Compilación y descarga de aplicaciones

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Compilar aplicaciones

Descripción general

SoMachine proporciona diferentes procedimientos de compilación en el menú Compilar *(véase SoMachine, Comandos de menú, Ayuda en línea)*. Estos procedimientos sirven para gestionar las comprobaciones sintácticas, ya sea solamente en los objetos cambiados o en todos los objetos de la aplicación activa.

Puede realizar una generación de código offline para comprobar los errores de compilación antes de descargar el código en el dispositivo. Para que el inicio de sesión sea correcto, la generación de código debe haberse completado sin detectar errores.

Generación de código, información de compilación

El código máquina no se generará hasta que el proyecto Aplicación *(véase página 244)* esté descargado en el dispositivo de destino (controlador, destino de simulación). En cada descarga, la información de compilación, que contiene el código y un ID de referencia de la aplicación cargada, se almacena en el directorio del proyecto en un archivo llamado *<nombre de proyecto>.<nombre de dispositivo>.<ID de aplicación>.compileinfo*. El archivo *compileinfo* se elimina cuando se ejecuta el comando **Limpiar** o **Limpiar todo**.

La generación del código no se lleva a cabo cuando el proyecto se compila mediante los comandos de compilación (de forma predeterminada en el menú **Compilar**). El proceso de compilación comprueba el proyecto para detectar si hay errores de programación. Los errores de programación que se detecten se muestran en la vista **Mensajes** (categoría de mensajes **Generación**).

Durante la generación de código, pueden detectarse y mostrarse errores adicionales. Estos errores solamente los puede detectar el generador de código o están provocados por la asignación de memoria.

Inicio de sesión

Descripción general

El comando **Online** →**Inicio de sesión** conecta la aplicación al dispositivo de destino (controlador o destino de la simulación) y, de este modo, cambia a la modalidad online.

El método abreviado predeterminado es ALT + F8.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

- Asegúrese de haber introducido la designación o dirección del dispositivo correcta en el cuadro de diálogo "Configuración de comunicación" al descargar una aplicación.
- Asegúrese de que las protecciones y las etiquetas de la máquina están colocadas con el fin de evitar el funcionamiento imprevisto de la máquina y riesgos de daños personales o materiales.
- Lea y tenga en cuenta toda la documentación para el usuario del software y los dispositivos relacionados, así como la documentación referente al funcionamiento de la máquina o el equipo.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

Si hay una administración de usuarios online (consulte también el capítulo *Usuarios y grupos (véase página 972)*) establecida en el dispositivo de destino, se le solicitará que introduzca el nombre de usuario y la contraseña adecuados. Para ello, se abre el cuadro de diálogo **Inicio de sesión de usuario en el dispositivo**.

Procedimientos de inicio de sesión

Existen 2 procedimientos de inicio de sesión, según el modo de **Marcar** seleccionado en el cuadro de diálogo **Proyecto** →**Configuración del proyecto** →**Configuración de comunicación**. La configuración predeterminada de este modo de **Marcar** depende de la versión de SoMachine. El editor de dispositivos proporciona cuadros de diálogo individuales para cada tipo de inicio de sesión.

Tipo de inicio de sesión	Modo de Marcar	Valor predeterminado para la versión de SoMachine	Cuadro de diálogo del editor de dispositivos
1	**Dirección IP**	V4.0 y posteriores	**Selección de controlador** *(véase página 103)*
2	**ruta activa**	V3.1 y anteriores	**Configuración de comunicación** *(véase página 120)*

Procedimiento de inicio de sesión con el modo de Marcar Dirección IP

Este es el procedimiento de inicio de sesión predeterminado para SoMachine V4.0 y versiones posteriores. El modo de **Marcar** en el cuadro de diálogo **Proyecto →Configuración del proyecto →Configuración de comunicación** está establecido como **Marque mediante "dirección IP"**.

Para conseguir un inicio de sesión correcto, la generación de código se debe haber completado sin detectar errores (consulte el capítulo *Proceso de compilación antes de iniciar la sesión (véase página 257)*).

Paso	Acción
1	Ejecute el comando **Online →Inicio de sesión** o haga clic en el botón **Inicio de sesión**  en la barra de herramientas o pulse ALT + F8. **Resultado**: Puesto que no se ha establecido ninguna dirección de destino antes, se abre la vista **Selección de controlador** del editor de dispositivos. Se muestra un cuadro de mensaje que indica que no se ha definido una dirección válida.
2	Si SoMachine sólo ha detectado un controlador al explorar la red Ethernet, este controlador se marca en la lista y se utiliza como dispositivo de destino. Si se han detectado varios controladores, haga doble clic en el controlador en el que desee iniciar la sesión.
3	Ejecute el comando **Online →Inicio de sesión** o haga clic en el botón **Inicio de sesión**  en la barra de herramientas o pulse ALT + F8. **Resultado**: Se muestra un cuadro de mensaje para informarle de los riesgos potenciales.
4	Haga clic en **Cancelar** para anular la operación de inicio de sesión o pulse ALT + F para confirmar el mensaje e iniciar sesión en el controlador seleccionado. **Resultado**: Si pulsa ALT + F, se establece la conexión con el controlador y se puede descargar la aplicación *(véase página 258)*.

Procedimiento de inicio de sesión con el modo de Marcar de ruta activa

Este es el procedimiento de inicio de sesión predeterminado para SoMachine V3.1 y versiones anteriores. El modo de **Marcar** en el cuadro de diálogo **Configuración del proyecto** → **Configuración de comunicación** está establecido como **Marque mediante "ruta activa"**.

Para conseguir un inicio de sesión correcto, la generación de código se debe haber completado sin detectar errores (consulte el capítulo *Proceso de compilación antes de iniciar la sesión (véase página 257)*). Además, la configuración de comunicación *(véase página 103)* del dispositivo se debe configurar correctamente.

Paso	Acción
1	Ejecute el comando **Online** →**Inicio de sesión** o haga clic en el botón **Inicio de sesión**  en la barra de herramientas o pulse ALT + F8. **Resultado**: Puesto que no se ha establecido ninguna dirección de destino antes, se abre la vista **Configuración de comunicación** del editor de dispositivos. Se muestra un cuadro de mensaje que indica que la ruta activa no se ha establecido y que se está explorando la red.
2	Si SoMachine sólo ha detectado un controlador al explorar la red Ethernet, este controlador se marca en la lista y se utiliza como dispositivo de destino. Si se han detectado varios controladores, haga doble clic en el controlador en el que desee iniciar la sesión. **NOTA:** Sólo se muestran los controladores que tienen el mismo **ID del sistema de destino** que el controlador seleccionado. Para mostrar todos los controladores de la lista, establezca el criterio **Filtro** como **Ninguno**.
3	Ejecute el comando **Online** →**Inicio de sesión** o haga clic en el botón **Inicio de sesión**  en la barra de herramientas o pulse ALT + F8. **Resultado**: Se muestra un cuadro de mensaje para informarle de los riesgos potenciales.
4	Haga clic en **Cancelar** para anular la operación de inicio de sesión o pulse ALT + F para confirmar el mensaje e iniciar sesión en el controlador seleccionado. **Resultado**: Si pulsa ALT + F, se establece la conexión con el controlador y se puede descargar la aplicación *(véase página 258)*.

Proceso de generación en aplicaciones cambiadas

Proceso de generación antes del inicio de sesión

Antes del **Inicio de sesión**, se compilará el proyecto de aplicación afectado actual si no se ha compilado desde que se abrió o desde la última modificación. Esto significa que el proyecto se generará respecto a la **compilación** correspondiente que se ejecuta en modalidad offline y que se generará el código de compilación para el controlador.

Si se detectan errores durante la compilación, se abrirá un cuadro de mensaje con el texto siguiente: **Había errores de compilación. ¿Quiere iniciar la sesión sin cargar el programa?** Puede elegir entre corregir primero los errores detectados o bien iniciar la sesión directamente. En este último caso, se iniciará sesión en esa versión de la aplicación que probablemente ya esté disponible en el controlador.

Los errores detectados se enumeran en la vista **Mensajes** (categoría **Compilar**).

Descarga de una aplicación

Introducción

Para ejecutar una aplicación, conecte primero el PC al controlador y, a continuación, descargue la aplicación en el controlador.

La descarga de un proyecto permite copiar el proyecto actual de SoMachine a la memoria del controlador.

NOTA: Debido a la limitación de tamaño de la memoria, algunos controladores no pueden almacenar el origen de la aplicación, sino sólo una aplicación compilada que se ejecute. Esto significa que no se podrá cargar el origen de la aplicación del controlador a un PC.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

- Asegúrese de haber introducido la designación o la dirección del dispositivo correctas en el cuadro de diálogo **Configuración de comunicación** al descargar una aplicación.
- Asegúrese de que las protecciones y las etiquetas de la máquina están colocadas con el fin de evitar el funcionamiento imprevisto de la máquina y riesgos de daños personales o materiales.
- Lea y tenga en cuenta toda la documentación para el usuario del software y los dispositivos relacionados, así como la documentación referente al funcionamiento de la máquina o el equipo.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

Condiciones previas

Compruebe que la aplicación reúne las condiciones siguientes antes de descargarla en el controlador:

- La ruta activa está definida para el controlador correcto.
- La aplicación que se desea descargar está activa.
- La aplicación no tiene errores de compilación.

Aplicación de arranque

La aplicación de arranque es la que se ejecuta al iniciar el controlador. Esta aplicación se almacena en la memoria del controlador. Para configurar la descarga de la aplicación de arranque, haga clic con el botón derecho del ratón en el nodo **Aplicación** de la vista **Dispositivos** y seleccione el comando **Propiedades**.

Al final de una descarga correcta de una nueva aplicación, aparecerá un mensaje para preguntarle si desea crear la aplicación de arranque.

Puede crear una aplicación de arranque manualmente de las siguientes maneras:

- En modalidad offline: Haga clic en **En línea** →**Crear aplicación de inicio** para guardar la aplicación de arranque en un archivo.
- En modalidad online, con la aplicación en modalidad STOP: Haga clic en **En línea** →**Crear aplicación de inicio** para descargar la aplicación de arranque en el controlador.

Modalidades de funcionamiento

El método de descarga variará según la relación entre la aplicación cargada y la aplicación que se desee descargar. Los tres casos son los siguientes:

- Caso 1: La aplicación del controlador es la misma que la que se desea cargar. En este caso, no se produce ninguna descarga: sólo se conecta SoMachine al controlador.
- Caso 2: Se han realizado modificaciones en la aplicación que se ha cargado en el controlador en comparación con la aplicación de SoMachine. En este caso, puede especificar si desea descargar toda la aplicación modificada o partes de esta, o bien mantenerla tal cual en el controlador.
- Caso 3: Ya hay disponible una versión diferente o nueva de la aplicación en el controlador. En este caso, se le solicitará si debe sustituirse esa aplicación.
- Caso 4: La aplicación todavía no está disponible en el controlador. En este caso, se le solicitará que confirme la descarga.

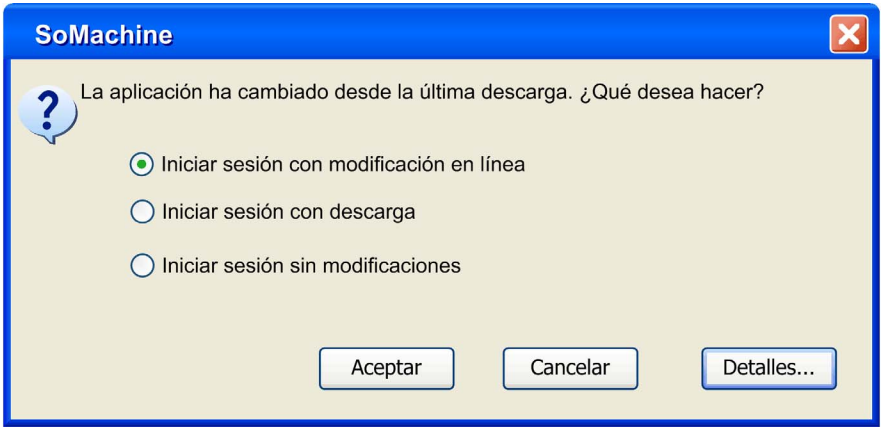
Descarga de la aplicación en el controlador: Caso 1

La aplicación del controlador es la misma que la que se desea cargar. En este caso, no se produce ninguna descarga: sólo se conecta SoMachine al controlador.

Paso	Acción
1	Para conectarse con el controlador, seleccione **En línea** →**Iniciar la sesión en 'Aplicación[Nombre_aplicación; Lógica Plc]'**.
2	Ya está conectado al controlador.

Descarga de la aplicación en el controlador: Caso 2

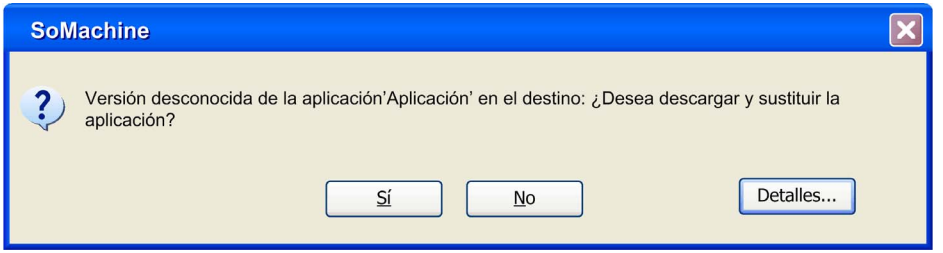
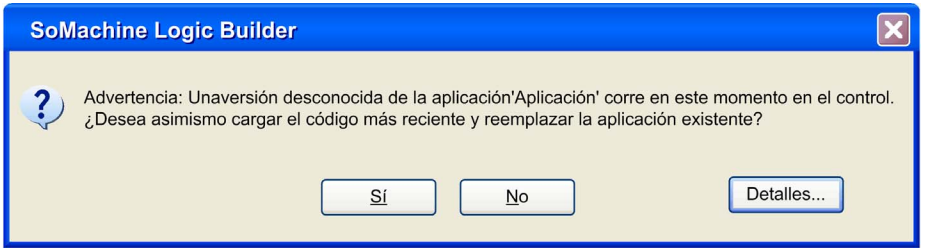
Se han realizado modificaciones en la aplicación que se ha cargado en el controlador en comparación con la aplicación de SoMachine.

Paso	Acción
1	Para conectarse con el controlador, seleccione **En línea →Iniciar la sesión en 'Aplicación[Nombre_aplicación; Lógica Plc]'**.
2	Si ha modificado la aplicación y desea volver a cargarla en el controlador, aparecerá el mensaje siguiente:  **Iniciar sesión con modificación en línea** Sólo se volverán a cargar en el controlador las partes modificadas de un proyecto que ya esté en ejecución. **Iniciar sesión con descarga** La aplicación modificada se vuelve a cargar por completo en el controlador. **Iniciar sesión sin modificaciones** No se cargarán las modificaciones. **NOTA:** Si selecciona la opción **Iniciar sesión sin modificaciones**, los cambios que realice en la aplicación SoMachine no se descargarán en el controlador. En este caso, la barra de estado e información de SoMachine mostrará **EN EJECUCIÓN** como estado operativo e indicará **Programa modificado (modificación en línea)**. Esto difiere de las opciones **Iniciar sesión con modificación en línea** o **Iniciar sesión con descarga**, donde la barra de estado e información indica **Programa inalterado**. **NOTA:** En este caso, la supervisión de variables es posible, pero el flujo lógico puede ser confuso, porque los valores de las salidas del bloque de funciones pueden no coincidir con los valores de las entradas. **Ejemplos** En LD, los estados de los contactos se supervisan en función de las variables afectadas. Esto puede tener como resultado que se muestre un contacto animado en azul seguido de un enlace azul (que indica el valor TRUE, verdadero), aunque la bobina conectada a este contacto muestre que es falso. En el flujo lógico de ST, parece que se ejecute una instrucción IF o un bucle, pero en realidad no se ejecuta porque la expresión de la condición es diferente en el proyecto y en el controlador.
3	Seleccione la opción adecuada y haga clic en **Aceptar**.

NOTA: Consulte la guía de programación del controlador para ver información importante sobre seguridad relativa a la descarga de aplicaciones.

Descarga de la aplicación en el controlador: Caso 3

Ya hay disponible una versión diferente o nueva de la aplicación en el controlador.

Paso	Acción
1	Para conectarse con el controlador, seleccione **En línea** →**Iniciar la sesión en 'Aplicación[Nombre_aplicación; Lógica Plc]'**.
2a	En caso de que el controlador no esté en modalidad de ejecución y desee cargar una aplicación diferente de la que se encuentra actualmente en el controlador, aparece el mensaje siguiente:  Consulte los mensajes de peligro siguientes antes de hacer clic en **Sí** para descargar la nueva aplicación en el controlador, o en **No** para cancelar la operación.
2b	En caso de que el controlador esté en modalidad de ejecución y desee cargar una aplicación diferente de la que se encuentra actualmente en el controlador, aparece el mensaje siguiente:  Consulte los mensajes del compilador siguientes antes de hacer clic en **Sí** para descargar la nueva aplicación en el controlador, o en **No** para cancelar la operación.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

Verifique que tenga la aplicación correcta antes de confirmar la descarga.

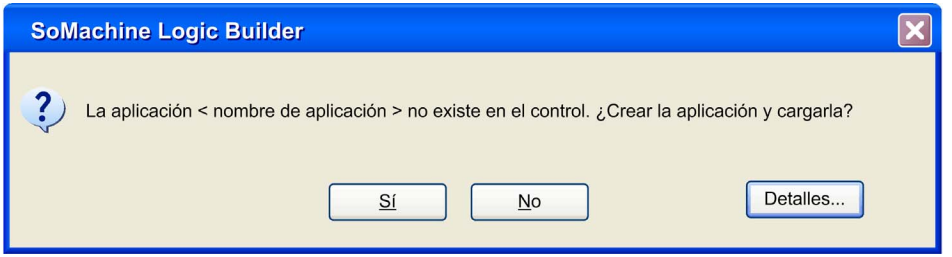
El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

Si hace clic en **Sí** , la aplicación en ejecución en el controlador se sobrescribirá.

Para ayudar a prevenir la pérdida de información, cancele esta operación haciendo clic en **No** y ejecute antes el comando **Carga de código de origen**. La aplicación actualmente disponible en su controlador se cargará en el PC. A continuación, puede compararla con la que tiene previsto descargar.

Descarga de la aplicación en el controlador: Caso 4

La aplicación todavía no está disponible en el controlador.

Paso	Acción
1	Para conectarse con el controlador, seleccione **En línea** →**Iniciar la sesión en 'Aplicación[Nombre_aplicación; Lógica Plc]'**.
2	En caso de que la aplicación todavía no esté disponible en el controlador, se le solicitará que confirme la descarga. Para este fin, se muestra un cuadro de diálogo con el texto siguiente:  Haga clic en **Sí** para descargar la aplicación en el controlador, o en **No** para cancelar la operación.

NOTA: Consulte la guía de programación del controlador para ver información importante sobre seguridad relativa a la descarga de aplicaciones.

Cambio en línea

El comando **Cambio en línea** modifica el programa de aplicación en ejecución y no afecta a un proceso de reinicio:

- El código del programa puede comportarse de forma distinta después de una inicialización completa debido a que la máquina mantiene su estado.
- Las variables de puntero mantienen sus valores desde el último ciclo. Si hay un puntero en una variable que ha cambiado su tamaño debido a un cambio en línea, el valor dejará de ser correcto. Verifique que las variables de puntero se vuelvan a asignar en cada ciclo.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

Pruebe minuciosamente el código de la aplicación para comprobar su correcto funcionamiento antes de poner el sistema en funcionamiento.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

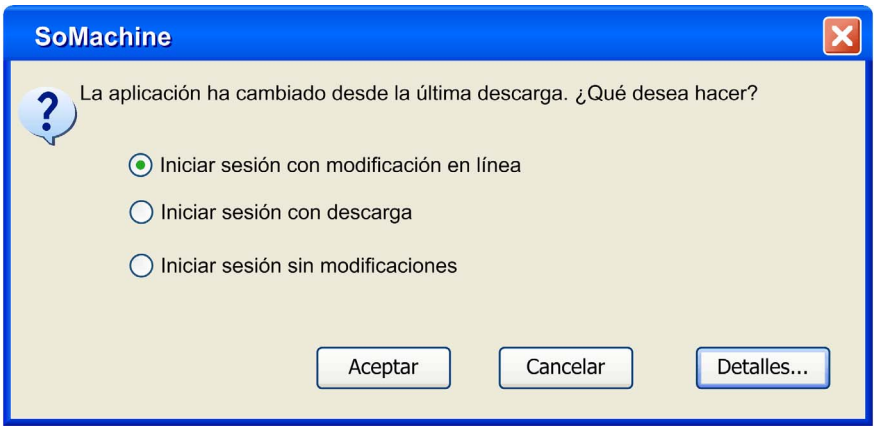
NOTA: Para obtener información específica, consulte el capítulo sobre la ***descripción de los estados del controlador*** en la guía de programación de su controlador.

Si el proyecto de aplicación actualmente en ejecución en el controlador se ha modificado en el sistema de programación desde que se descargó por última vez, sólo se cargarán los objetos modificados del proyecto en el controlador mientras el programa esté en ejecución.

Cambio en línea implícito

Cuando intenta volver a iniciar sesión con una aplicación modificada (comprobada mediante COMPILEINFO, que se ha almacenado en la carpeta del proyecto durante la última descarga), se le solicitará si desea realizar un cambio en línea, una descarga o un inicio de sesión sin realizar cambios.

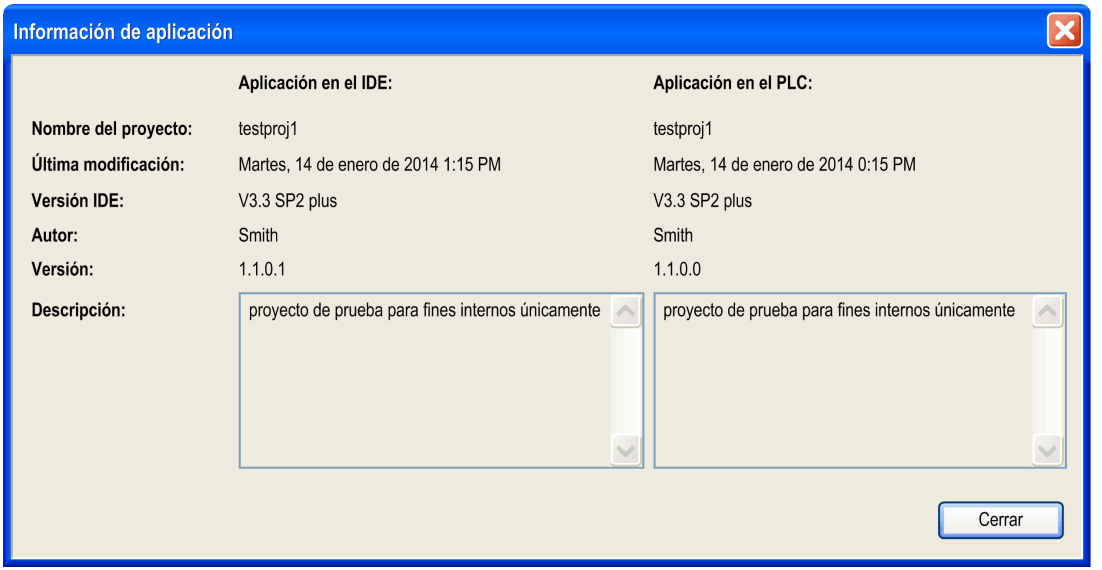
Cuadro de diálogo de inicio de sesión:



Descripción de los elementos:

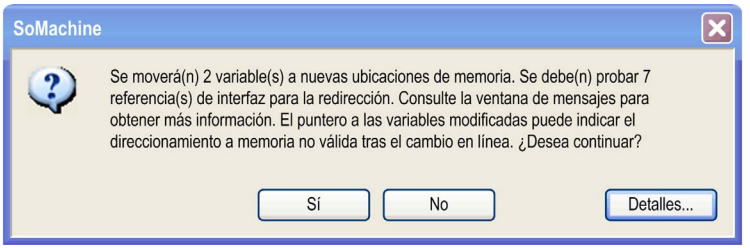
Elemento	Descripción
Iniciar sesión con modificación en línea	Esta opción está seleccionada de forma predeterminada. Si confirma el cuadro de diálogo haciendo clic en **Aceptar**, las modificaciones se cargarán y se mostrarán inmediatamente en una vista online (supervisión) del objeto o los objetos respectivos.
Iniciar sesión con descarga	Active esta opción para cargar e inicializar el proyecto de la aplicación por completo.
Iniciar sesión sin modificaciones	Active esta opción para mantener sin cambios el programa que está en ejecución en el controlador. Posteriormente puede realizarse una descarga explícita, de forma que se carga el proyecto de aplicación completo. También es posible que se le vuelva a solicitar si debe realizarse un cambio en línea en el siguiente inicio de sesión.
Detalles	Haga clic en este botón para obtener el cuadro de diálogo **Información de aplicación** (**nombre del proyecto, última modificación, versión IDE, autor, descripción**) en la aplicación actual en el IDE (entorno de desarrollo integrado, es decir, SoMachine), en comparación con la que hay disponible actualmente en el controlador. Consulte la figura siguiente y el capítulo *Información de aplicación* *(véase página 253)*.

Cuadro de diálogo **Información de aplicación**



Para obtener más información, consulte el capítulo *Inicio de sesión* *(véase página 254)*.

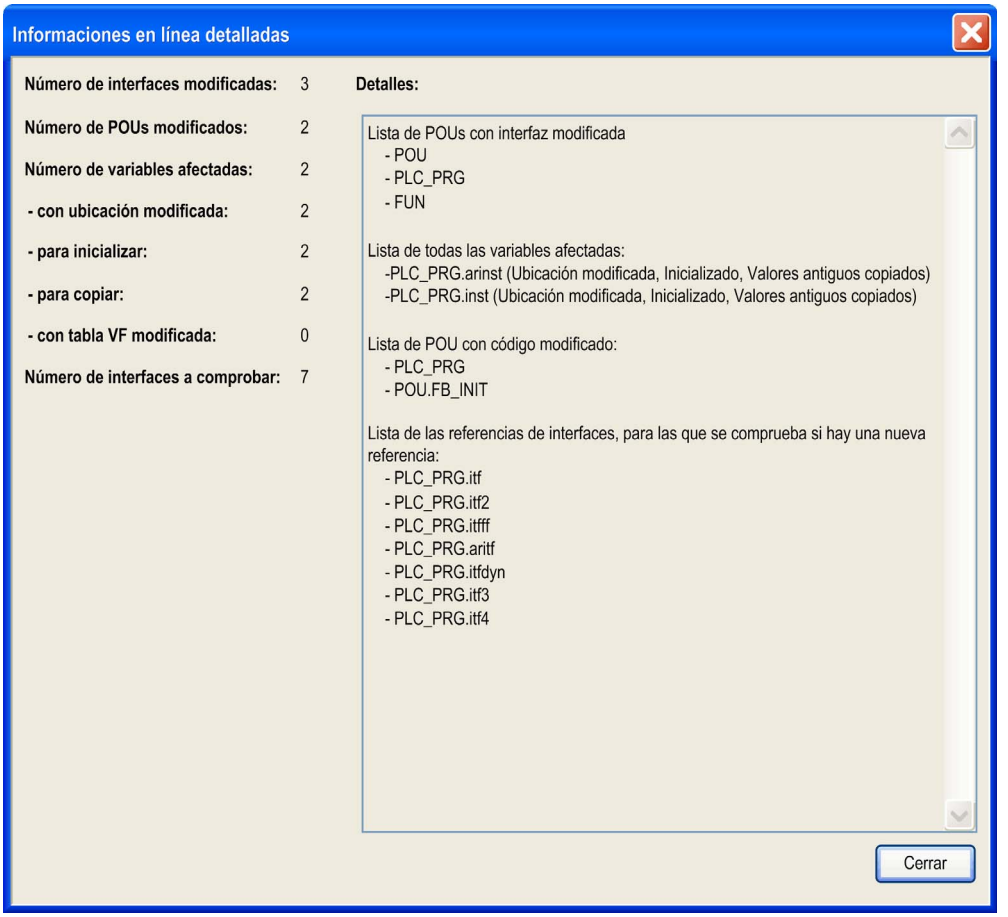
Si el cambio en línea comportará cambios considerables en el código de descarga, como por ejemplo posibles desplazamientos de direcciones del puntero o redirecciones necesarias de referencias de interfaz *(véase página 185)*, se mostrará otro cuadro de mensaje después de confirmar el cuadro de diálogo **Cambio en línea** con **Aceptar**, antes de realizar la descarga. Le informa acerca de los efectos que debe tener en cuenta y proporciona la opción de cancelar la operación de cambio en línea.



NOTA: Con SoMachine V4.0 y posteriores, después de eliminar la función de comprobación implícita (como CheckBounds) de la aplicación, no se puede utilizar **Cambio en línea**; solamente se puede realizar una descarga. Aparecerá el mensaje correspondiente.

Haga clic en el botón **Detalles** en este mensaje para mostrar información detallada, como el número y un listado de las interfaces modificadas, las POU, las variables afectadas, etc.

Cuadro de diálogo **Informaciones en línea detalladas**



Cambio en línea explícito

Ejecute el comando **Cambio en línea** (de forma predeterminada, en el menú **En línea**) para realizar explícitamente una operación de cambio en línea en una determinada aplicación.

Un **cambio en línea** de un proyecto modificado ya no es posible después de una operación **Limpiar** (**Compilar** →**Limpiar todo**, **Compilar** →**Limpiar**). En este caso, se borrará la información sobre los objetos que se han modificado desde la última descarga. Por tanto, sólo puede descargarse el proyecto completo.

NOTA:

Tenga en cuenta lo siguiente antes de ejecutar el comando **Cambio en línea**:
- Compruebe que el código modificado no presente errores de lógica.
- Las variables de puntero mantienen sus valores desde el último ciclo. Si apunta a una variable que ahora se ha desplazado, el valor ya no será correcto. Por este motivo, reasigne las variables de puntero en cada ciclo.

Información sobre el proceso de descarga

Cuando se carga por completo el proyecto en el controlador al **iniciar la sesión** o parcialmente en un **cambio en línea**, la vista **Mensajes** mostrará información sobre el tamaño del código generado, el tamaño de los datos globales, el espacio de memoria necesario en el controlador y, en caso de cambio en línea, también en las POU afectadas.

NOTA: En modalidad online, no es posible modificar la configuración de dispositivos o módulos. Para cambiar los parámetros de los dispositivos, debe finalizar la sesión en la aplicación. En función del sistema de bus, puede permitirse la modificación de ciertos parámetros especiales en modalidad online.

Aplicación de arranque (proyecto de arranque)

Tras cada descarga correcta, la aplicación activa se almacena automáticamente en el archivo *application.app* en la carpeta del sistema del controlador, de forma que está disponible como aplicación de arranque. Una aplicación de arranque es el proyecto que se inicia automáticamente al iniciarse (arrancar) el controlador. Para hacer que la descarga de la aplicación activa sea la aplicación de arranque, debe ejecutar el comando **Crear aplicación de inicio** (disponible en el menú **En línea**).

El comando **Crear aplicación de inicio** copiará el archivo *application.app* en un archivo denominado <*nombre de proyecto*>*.app*, de forma que se convertirá en la aplicación de arranque del controlador. También puede crear la aplicación de arranque en modalidad offline *(véase página 259)*.

Si desea conectarse al mismo controlador desde el sistema de programación de un PC diferente o recuperar la aplicación activa de un PC diferente sin necesidad de un cambio en línea o una descarga, siga los pasos que se describen en el párrafo *Transferencia de proyectos a otros sistemas*.

Transferencia de proyectos a otros sistemas

Para transferir un proyecto a otro sistema, utilice un archivo de proyecto *(véase SoMachine, Comandos de menú, Ayuda en línea)*.

Puede transferir un proyecto que ya se esté ejecutando en un controlador *xy* del sistema de programación en PC1 al de PC2. Para poder volver a conectarse al mismo controlador *xy* desde PC2 sin tener que realizar un cambio en línea o una descarga, compruebe la siguiente configuración del proyecto antes de crear un archivo de proyecto.

Siga los pasos siguientes:

1. Verifique que sólo se incluyan bibliotecas con versiones definitivas, excepto en el caso de bibliotecas de interfaz puras. (Abra el **Administrador de bibliotecas** y compruebe las entradas con un asterisco (*) en lugar de una versión fija *(véase SoMachine, Funciones y bibliotecas - Guía del usuario)*).
2. Asegúrese de que se haya establecido una versión definitiva de compilador en el cuadro de diálogo **Configuración del proyecto** →**Opciones de compilador** *(véase SoMachine, Comandos de menú, Ayuda en línea)*.
3. Asegúrese de definir un perfil de visualización en el cuadro de diálogo **Configuración del proyecto** →**Perfil de visualización** (para obtener más información, consulte el apartado *Visualización* de la ayuda online de SoMachine).
4. Verifique que la aplicación abierta actualmente sea la misma que la que ya está disponible en el controlador. Es decir, el proyecto de arranque (consulte el comando **En línea** →**Crear aplicación de inicio** *(véase SoMachine, Comandos de menú, Ayuda en línea)*) debe ser idéntico al proyecto en el sistema de programación. Si hay un asterisco detrás del título del proyecto en la barra de título de la ventana del sistema de programación, significa que el proyecto se ha modificado pero todavía no se ha guardado. En este caso, puede diferir respecto al proyecto de arranque. Si es necesario, antes de transferir el proyecto a otro PC, cree un (nuevo) proyecto de arranque (para algunos controladores, esto se lleva a cabo automáticamente durante la descarga) y luego descargue e inicie el proyecto en el controlador.
5. Cree el archivo de proyecto mediante SoMachine Central. Seleccione la información siguiente: **archivos de información de descarga**, **Perfil de biblioteca**, **Dispositivos referenciados**, **Bibliotecas referenciadas**, **Perfil de visualización**.
6. Cierre la sesión. Si es necesario, detenga y reinicie el controlador *xy* antes de volver a conectarse desde PC2.
7. Extraiga el archivo de proyecto en PC2 con las mismas opciones de información activadas que las que se enumeran en el paso 5.

Sección 8.3
Ejecución de aplicaciones

Ejecución de aplicaciones

Introducción

En esta parte se muestra cómo iniciar o detener una aplicación.

RUN/STOP con SoMachine

El controlador se puede iniciar y detener mediante la ejecución de SoMachine en un PC conectado al controlador.

Haga clic en **En línea →Inicio 'Aplicación [Nombre de aplicación: Lógica PLC]'** o CTRL + F5, o bien en el botón **Inicio 'Aplicación [Nombre de aplicación: Lógica Plc]'** de la barra de menús para iniciar la aplicación.

Haga clic en **En línea →Parada 'Aplicación [Nombre de aplicación: Lógica PLC]'** o CTRL +MAYÚS + F5, o bien en el botón **Parada 'Aplicación [Nombre de aplicación: Lógica Plc]'** de la barra de menús para detener la aplicación.

Entrada RUN/STOP para controladores

Algunos controladores permiten configurar una entrada RUN/STOP para controlar el inicio y la detención de la aplicación.

Estado	Descripción
0	Detención de la aplicación. El comando EJECUTAR no es posible en SoMachine.
Flanco ascendente	Inicio de la aplicación.
1	Se ejecuta la aplicación. El comando EJECUTAR/DETENER es posible en SoMachine.

Consulte el manual de su controlador para conocer si admite esta función.

Configure y utilice la entrada RUN/STOP si va a utilizar comandos remotos para iniciar y detener el controlador. Es la mejor forma de asegurarse un control local durante la ejecución del controlador; también ayuda a evitar un inicio inadvertido del controlador desde una ubicación remota.

⚠ ADVERTENCIA

INICIO NO DESEADO DE LA MÁQUINA O DEL PROCESO DE ACTIVACIÓN

- Compruebe el estado de seguridad de su máquina o del entorno del proceso antes de conectar la alimentación a la entrada Run/Stop.
- Use la entrada Run/Stop para evitar activaciones no deseadas de ubicaciones remotas.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

Sección 8.4
Mantenimiento de aplicaciones

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Supervisión

Descripción general

En modalidad online, hay varias opciones para mostrar los valores actuales de los objetos del controlador:

- Puede ver los valores de los objetos en una pantalla de editor de programa mientras esté en modalidad online. Para obtener información detallada, consulte la descripción del editor correspondiente.
- Puede ver los valores de los objetos en la vista online del editor de declaraciones. Para obtener información detallada, consulte la descripción del editor de declaraciones *(véase página 413)*.
- Puede ver objetos de forma independiente en listas mediante el comando **Supervisar**. Para obtener información detallada, consulte la descripción de la vista de supervisión/editor de lista de supervisión *(véase página 464)*. Para insertar una variable en una vista de supervisión, selecciónela y ejecute el comando **Add watchlist** en el menú contextual.
- Puede visualizar valores mediante el muestreo de trazas: registro y visualización de valores de variables desde el controlador. Para obtener información detallada, consulte la descripción de la funcionalidad de objetos de traza *(véase página 524)*.
- Puede ver los valores de los objetos contenidos en fórmulas: conjuntos de variables definidos por el usuario para escribir y supervisar las variables en el controlador. Consulte la descripción del gestor de fórmulas *(véase página 503)*.

Para obtener información sobre la supervisión de propiedades insertadas debajo de POU o bloques de funciones, consulte el capítulo Propiedad *(véase página 183)*.

Para obtener información sobre la supervisión de llamadas de función, consulte el capítulo *Attribute Monitoring* *(véase página 637)*.

NOTA: Si un valor no es válido (por ejemplo, el resultado de calcular la raíz cuadrada de un número negativo), el resultado puede mostrarse como `NaN` (no es un número) o `INF` (valor infinito), según la operación, el objeto y la plataforma concreta del controlador. Para obtener más información, consulte la guía de programación de su controlador.

Depuración

Descripción general

Para evaluar errores de programación potenciales, puede utilizar la funcionalidad de depuración en modalidad online. También puede, hasta cierto punto, depurar su aplicación en modo de simulación. Si bien la simulación evita la necesidad de conectarse al hardware físico, existen limitaciones por las que es posible que deba llevar a cabo la depuración en línea.

Puede establecer puntos de interrupción en ciertas posiciones para forzar una interrupción de la ejecución. Por cada punto de interrupción pueden establecerse ciertas condiciones, como qué tareas se identifican y en qué ciclos debe ser efectivo el punto de interrupción (puntos de interrupción condicionales). Consulte la descripción sobre *puntos de interrupción* en este capítulo.

Existen funciones de ejecución paso a paso que permiten ejecutar un programa en pasos controlados. Consulte el párrafo *Ejecución paso a paso* *(véase página 274)*.

En cada interrupción, puede examinar los valores actuales de las variables. Puede visualizar una pila de llamadas *(véase SoMachine, Comandos de menú, Ayuda en línea)* para la posición del paso actual.

Puede activar la función **Control de proceso** para realizar el seguimiento de las partes ejecutadas del programa de aplicación. A diferencia de la función de supervisión estándar, que sólo muestra el valor de una variable entre 2 ciclos de ejecución, **Control de flujo** proporciona el valor en cada paso de ejecución concreto, exactamente en el momento de la ejecución.

Puntos de interrupción

Un punto de interrupción que se defina en un programa de aplicación provocará una interrupción durante la ejecución del programa. Sólo la tarea que llegue al punto de interrupción (en lo sucesivo, denominada tarea de depuración) se detendrá; sin embargo, las demás se seguirán ejecutando. Las posiciones posibles de los puntos de interrupción dependen del editor. En cada caso, hay un punto de interrupción al final de una POU.

NOTA: Las E/S gestionadas por la tarea de depuración no se actualizarán en la detención de un punto de interrupción. Esto se aplica aunque la opción **Actualizar E/S en parada** esté activada en la ficha **Ajustes PLC** del editor de dispositivos *(véase página 134)*.

Consulte el capítulo *Comandos relacionados con los puntos de interrupción* para ver una descripción de los comandos relacionados con los puntos de interrupción. El cuadro de diálogo **Puntos de interrupción** *(véase SoMachine, Comandos de menú, Ayuda en línea)* proporciona una descripción general de todos los puntos de interrupción, lo que le permite añadir, eliminar y modificar puntos de interrupción.

Puntos de interrupción condicionales

Se puede establecer que la parada en un punto de interrupción dependa del número de ciclos de una determinada tarea o de la tarea que esté actualmente activa. Al declarar una tarea de depuración específica, puede evitar que cada tarea que comparte la misma POU se vea afectada salvo el punto de interrupción (consulte el apartado relativo a los *puntos de interrupción y la ejecución por pasos en aplicaciones con múltiples tareas* *(véase página 275)*).

Símbolos de punto de interrupción

Símbolo	Descripción
	Punto de interrupción activado
	Punto de interrupción desactivado
	Parada en el punto de interrupción en modalidad online
	Posición actual del paso Se indica con una flecha amarilla delante de la línea respectiva y una sombra amarilla detrás de la operación en cuestión.

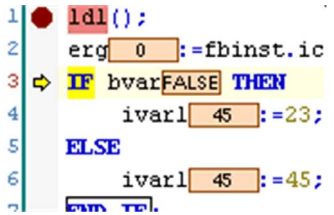
Ejecución paso a paso

La ejecución paso a paso permite la ejecución controlada de un programa de aplicación para fines de depuración. Básicamente, se pasa de una instrucción a la siguiente accediendo a la instrucción (step into), pasando a la siguiente instrucción (step over) o saliendo de la instrucción (step out). Consulte el capítulo *Comandos relacionados con puntos de interrupción (véase SoMachine, Comandos de menú, Ayuda en línea)* para ver una descripción de los comandos de ejecución paso a paso.

Ejemplo de operación paso a paso (step Into)

A partir del punto de interrupción, puede ejecutar cada línea de comandos con el comando de ejecución paso a paso.

Paso a paso, ejemplo



Puntos de interrupción y ejecución paso a paso en aplicaciones con múltiples tareas

Si múltiples tareas pueden acceder a un punto de interrupción porque varias tareas utilizan la POU, sólo se detendrá la tarea que llegue en primer lugar. Tenga esto en cuenta en el caso de una única ejecución paso a paso o si prosigue la depuración después de una parada. Podría suceder que otra tarea se detuviera la próxima vez que lo alcance (posiblemente, el ciclo todavía no ha terminado). Si sólo debe haber 1 determinada tarea implicada (tarea de depuración), puede especificarlo en las propiedades de las condiciones del punto de interrupción (cuadro de diálogo **Puntos de interrupción** →**Nuevo punto de interrupción**, ficha **Condición**).

Parte IV
Editores de lógica

Contenido de esta parte

Esta parte contiene los siguientes capítulos:

Capítulo 9
Editor FBD/LD/IL

Contenido de este capítulo

Este capítulo contiene las siguientes secciones:

Sección 9.1
Información sobre el editor FBD/LD/IL

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Editor FBD/LD/IL

Descripción general

Existe un editor combinado para editar las POU en los lenguajes FBD (diagrama de bloques de funciones *(véase página 282)*), LD (diagrama de contactos *(véase página 283)*) e IL (lista de instrucciones *(véase página 284)*).

Por lo tanto, se utiliza un conjunto común de comandos y elementos y se efectúa una conversión interna automática entre los 3 lenguajes. En modalidad offline, el programador puede pasar de una vista de editor a otra (menú **Ver**).

Tenga en cuenta que hay algunos elementos especiales que no se pueden convertir, por lo que solamente se mostrarán en el lenguaje correspondiente. Además, hay algunos componentes que no se pueden convertir sin ambigüedades de IL a FBD y viceversa y que, por lo tanto, se normalizarán como FBD en la conversión; concretamente, la negación de expresiones y las asignaciones de salida explícita/implícita.

Puede definir el comportamiento, el aspecto y los menús del editor FBD/LD/IL en los cuadros de diálogo **Personalizar** y **Opciones**. También dispone de opciones para definir la visualización de comentarios y direcciones.

El editor se abre en una ventana doble. Cuando se edita un objeto programado en FBD/LD/IL, la parte superior contiene un editor de declaraciones *(véase página 414)* y la parte inferior contiene el área de codificación.

El lenguaje de programación para un objeto nuevo se especifica al crear el objeto.

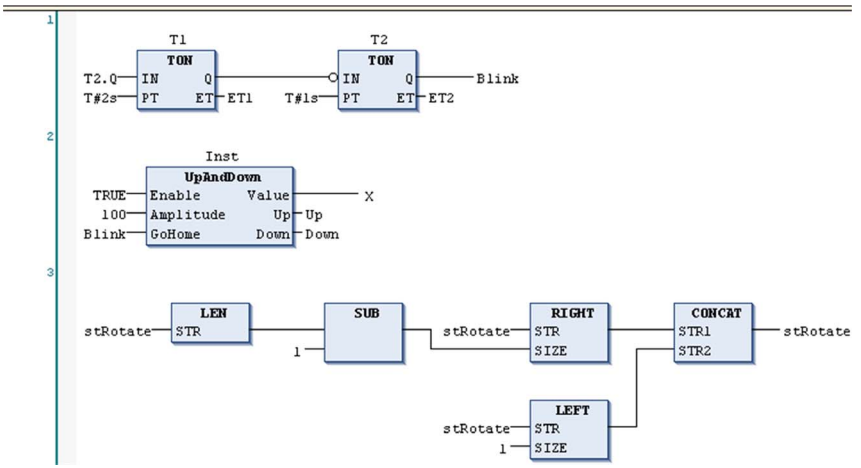
Para obtener más información, consulte:
- *Trabajo en el editor FBD y LD (véase página 289)*
- *Trabajo en la vista del editor IL (véase página 294)*

Lenguaje de diagrama de bloques de funciones (FBD)

Descripción general

El diagrama de bloques de funciones es un lenguaje de programación orientado gráficamente. Funciona con una lista de redes en la que cada red contiene una estructura gráfica de cuadros y líneas de conexión que representa una expresión lógica o aritmética, la llamada de un bloque de funciones, un salto o una instrucción de retorno.

Redes FBD



Lenguaje del diagrama de contactos (LD)

Descripción general

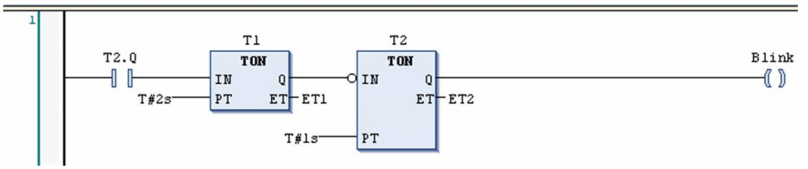
El diagrama de contactos es un lenguaje de programación orientado a gráficos que se asemeja a la estructura de un circuito eléctrico.

Por un lado, el diagrama de contactos es adecuado para la confección de interruptores lógicos y, por otro lado, también permite crear redes como en FBD. Por lo tanto, el LD es útil para controlar la llamada de otras POU.

El diagrama de contactos se compone de una serie de redes, cada una limitada por una línea de corriente vertical (raíl de alimentación) a la izquierda. Una red contiene un diagrama de circuito formado por contactos, bobinas, POU adicionales opcionales (módulos) y líneas de conexión.

En el lado izquierdo, hay 1 contacto o una serie de contactos que transmiten de izquierda a derecha la condición ON u OFF, que corresponde a los valores booleanos TRUE y FALSE. A cada contacto se le asigna una variable booleana. Si esta variable es TRUE, la condición se transmitirá de izquierda a derecha a lo largo de la línea de conexión. De lo contrario, se transmitirá OFF. Por lo tanto, las bobinas colocadas en la parte derecha de la red reciben un ON u OFF proveniente de la parte izquierda. En consecuencia, el valor TRUE o FALSE se escribirá en una variable booleana asignada.

Red de diagrama de contactos



Lenguaje de la lista de instrucciones (IL)

Descripción general

La lista de instrucciones (IL) es un lenguaje de programación conforme al estándar IEC 61131-3, parecido a ensamblador.

Este lenguaje admite la programación basada en un acumulador. Se admiten los operadores IEC 61131-3, así como varias entradas/varias salidas, negaciones, comentarios, establecimiento/restablecimiento de salidas y saltos incondicionales/condicionales.

Cada instrucción se basa principalmente en la carga de los valores en el acumulador mediante el uso del operador LD. Tras ello, la operación se ejecuta con el primer parámetro extraído del acumulador. El resultado de la operación está disponible en el acumulador, desde donde debe almacenarlo con la instrucción ST.

Para programar ejecuciones condicionales o bucles, IL admite tanto los operadores de comparación (EQ, GT, LT, GE, LE, NE) como los saltos. Estos últimos pueden ser incondicionales (JMP) o condicionales (JMPC/JMPCN). Para los saltos condicionales, el valor del acumulador se referencia como TRUE o FALSE.

Sintaxis

Una lista de instrucciones (IL) está formada por una serie de instrucciones. Cada instrucción empieza en una línea nueva y contiene un operador y, según el tipo de operación, 1 o más operandos separados por comas. Puede extender el operador con un modificador.

En la línea anterior a una instrucción, puede haber una marca de identificación (etiqueta) seguida de dos puntos (:) (`m1:` en el ejemplo que se muestra a continuación). Una etiqueta puede ser el destino de una instrucción de salto (`JMPC m1` en el ejemplo que se muestra a continuación).

Incluya un comentario como último elemento de una línea.

Puede insertar líneas vacías entre las instrucciones.

Ejemplo

```
PROGRAM IL
VAR
     inst1: TON;
     dwVar: DWORD;
     dwRes: DWORD;
     t1:    TIME;
     tout1: TIME;
     inst2: TON;
     bVar:  BOOL;
END_VAR
LD       bVar                                variable
ST       inst1.IN          starts timer with risin...
JMPC     ml
CAL      inst1(
         PT:=t1,
         ET:=>tout1)
LD       inst1.Q           is TRUE, PT seconds aft...
ST       inst2.IN          starts timer with risin...
ml:
LD       dwVar
ADD      230
ST       dwRes
```

Para obtener más información, consulte:

- *Trabajo en la vista del editor IL (véase página 294)*
- *Modificadores y operadores en IL (véase página 286)*

Modificadores y operadores en IL

Modificadores

Puede utilizar los siguientes modificadores en Lista de instrucciones *(véase página 284)*.

C	con JMP, CAL, RET:	La instrucción sólo se ejecutará si el resultado de la expresión anterior es TRUE.
N	con JMPC, CALC, RETC:	La instrucción sólo se ejecutará si el resultado de la expresión anterior es FALSE.
N	Otros operadores según la tabla *Operadores* siguiente (*N* en la columna *Modificadores*)	Negación del operando (no del acumulador).

Operadores

En la siguiente tabla se muestran los operadores que se pueden utilizar en combinación con los modificadores especificados.

El acumulador guarda el valor actual, resultante de la operación anterior.

Operador	Modificadores	Significado	Ejemplo
LD	N	Carga el valor (negado) del operando en el acumulador.	`LD iVar`
ST	N	Guarda el contenido (negado) del acumulador en la variable de operando.	`ST iErg`
S	–	Establece el operando (tipo BOOL) en TRUE cuando el contenido del acumulador es TRUE.	`S bVar1`
R	–	Establece el operando (tipo BOOL) en FALSE cuando el contenido del acumulador es TRUE.	`R bVar1`
AND	N,(	AND a nivel de bit del acumulador y el operando (negado).	`AND bVar2`
OR	N,(	OR a nivel de bit del acumulador y el operando (negado).	`OR xVar`
XOR	N,(	OR a nivel de bit exclusivo del acumulador y el operando (negado).	`XOR N, (bVar 1,bVar2)`
NOT	–	Negación a nivel de bit del contenido del acumulador.	–
ADD	(	Suma de acumulador y operando; el resultado se copia en el acumulador.	`ADD (iVar1 ,iVar2)`
SUB	(	Resta de acumulador y operando; el resultado se copia en el acumulador.	`SUB iVar2`

Operador	Modificadores	Significado	Ejemplo
MUL	(	Multiplicación de acumulador y operando; el resultado se copia en el acumulador.	`MUL iVar2`
DIV	(	División de acumulador y operando; el resultado se copia en el acumulador.	`DIV 44`
GT	(	Verifica si el acumulador es mayor o igual que el operando; el resultado (BOOL) se copia en el acumulador; >=	`GT 23`
GE	(	Verifica si el acumulador es mayor o igual que el operando; el resultado (BOOL) se copia en el acumulador; >=	`GE iVar2`
EQ	(	Verifica si el acumulador es igual que el operando; el resultado (BOOL) se copia en el acumulador; =	`EQ iVar2`
NE	(	Verifica si el acumulador no es igual que el operando; el resultado (BOOL) se copia en el acumulador; <>	`NE iVar1`
LE	(	Verifica si el acumulador es menor o igual que el operando; el resultado (BOOL) se copia en el acumulador; <=	`LE 5`
LT	(	Verifica si el acumulador es menor que el operando; el resultado (BOOL) se copia en el acumulador; <	`LT cVar1`
JMP	CN	Salto incondicional (condicional) a la etiqueta	`JMPN next`
CAL	CN	Llamada (condicional) de PROGRAM o FUNCTION_BLOCK (si el acumulador es TRUE).	`CAL prog1`
RET	–	Retorno anticipado de la POU y salto hacia atrás a la POU de llamada	`RET`
RET	C	Condicional (si el acumulador es TRUE), retorno anticipado de la POU y salto hacia atrás a la POU de llamada	`RETC`
RET	CN	Condicional (si el acumulador es FALSE), retorno anticipado de la POU y salto hacia atrás a la POU de llamada	`RETCN`
)	–	Evaluar operación diferida	–

Consulte también Operadores IEC *(véase página 703)* y Trabajar en el editor IL *(véase página 294)* para obtener información sobre cómo utilizar y gestionar múltiples operandos, operandos complejos, llamadas de función/método/bloque de funciones/programa/acción y saltos.

Ejemplo

Programa IL de ejemplo con diversos modificadores:

```
LD      TRUE        load TRUE to accumulator
ANDN    bVar1       execute AND with negative value of bVar1
JMPC    m1         if accum. is TRUE, jump to label "m1"
LDN     bVar2      store negated value of bVar2...
ST      bRes       ... in bRes
m1:
LD      bVar2      store value of bVar2...
ST      bRes      ... in bRes
```

Trabajo en el editor FBD y LD

Descripción general

Las redes son las entidades básicas en la programación en FBD y LD. Cada red contiene una estructura que muestra una expresión lógica o aritmética, una POU (función, programa, llamada de bloque de funciones, etc.), un salto o una instrucción de retorno.

Cuando se crea un nuevo objeto, la ventana del editor contiene de forma automática 1 red vacía.

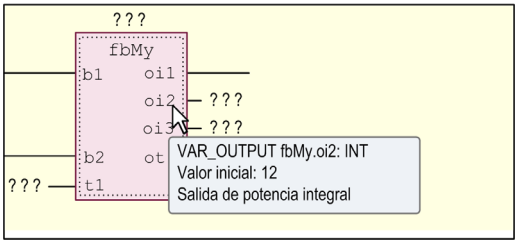
Consulte la configuración general del editor en el cuadro de diálogo **Opciones**, categoría **IL, FBD, LD** para ver las opciones de visualización posibles del editor.

Información sobre herramientas

La información sobre herramientas contiene información sobre las variables o los parámetros del módulo.

Al colocar el cursor sobre el nombre de una variable o de un parámetro del módulo aparecerá la información sobre herramientas. Muestra el tipo correspondiente. En el caso de las instancias de bloques de funciones, se mostrará el ámbito, el nombre, el tipo de datos, el valor inicial y el comentario. En el caso de los operadores IEC `SEL`, `LIMIT` y `MUX`, se mostrará una breve descripción de las entradas. Si se definen, la dirección y el comentario de símbolo se mostrarán además del comentario de operandos (entre comillas en una segunda línea).

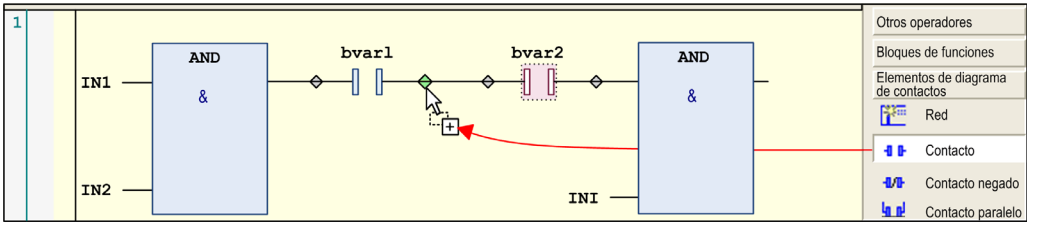
Ejemplo: Información sobre herramientas de una salida de una POU



Inserción y organización de los elementos

- Los comandos para trabajar en el editor están disponibles de forma predeterminada en el menú IL, FBD, LD *(véase página 305)*. Los comandos de uso frecuente también están disponibles en el menú contextual. Los elementos que se puedan insertar a través del comando de menú depende de la posición actual del cursor o de la selección actual (se puede utilizar la multiselección, consulte Selección *(véase página 291)* más adelante).
- También puede arrastrar elementos con el ratón desde las herramientas *(véase página 313)* hasta la ventana del editor o desde una posición dentro del editor hasta otra (arrastrar y soltar). Para ello, seleccione el elemento que se va a arrastrar con un clic del ratón, mantenga pulsado el botón del ratón y arrastre el elemento a la red correspondiente en la vista del editor. En cuanto llegue a la red, todas las posiciones de inserción posibles para el tipo de elemento correspondiente se indicarán mediante marcadores de posición de color gris. Cuando mueva el cursor del ratón a una de estas posiciones, el marcador se volverá de color verde y podrá soltar el botón del ratón para insertar el elemento en esa posición.

Insertar posiciones en el editor LD



- Puede utilizar los comandos para cortar, copiar, pegar y eliminar disponibles en el menú **Editar** para organizar los elementos. También puede copiar un elemento mediante el método de arrastrar y soltar: seleccione el elemento en una red con un clic del ratón, pulse la tecla CTRL y, mientras mantiene pulsados el botón del ratón y la tecla, arrastre el elemento a la posición de destino. En cuanto se llegue a ella (marcador de posición de color verde), se añadirá un signo más al símbolo del cursor. Luego, suelte el botón del ratón para insertar el elemento.
- Para ver las posiciones posibles del cursor, consulte *Posiciones de cursor en FBD, LD e IL (véase página 301)*.
- La inserción de módulos EN/ENO se gestiona de forma distinta en el editor FBD y LD. Consulte la descripción del comando **Insertar llamada de módulo** para obtener más información (la inserción de módulos EN/ENO no se admite en el editor IL).

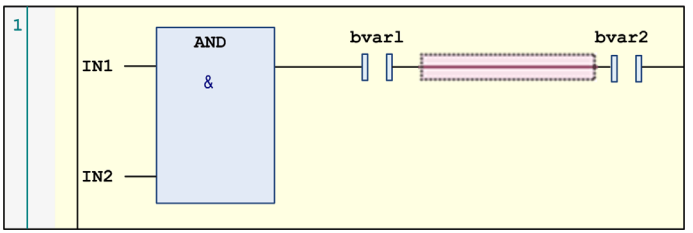
Desplazamiento en el editor

- Puede utilizar las teclas de flecha para ir a la posición del cursor *(véase página 301)* siguiente o anterior. Esto también se puede hacer entre redes. El desplazamiento con las teclas ←y → sigue el flujo de la señal, que suele ir de izquierda a derecha y viceversa. En el caso de los saltos de línea, la posición actual del cursor también se puede dejar bajo la posición que esté marcada en ese momento. Si pulsa la tecla ↑ o ↓, la selección pasa al elemento siguiente que está por encima o por debajo de la posición actual si este elemento está en el mismo grupo lógico (por ejemplo, un pin de un módulo). Si no existe un grupo de este tipo, pasa al elemento contiguo más cercano por encima o por debajo. El desplazamiento por los elementos conectados en paralelo se efectúa en la primera derivación.
- Pulse la tecla Inicio para ir al primer elemento. Pulse la tecla FIN para ir al último elemento de la red.
- Utilice la tecla TAB para ir a la posición del cursor *(véase página 301)* anterior o siguiente dentro de una red.
- Pulse CTRL + INICIO para desplazarse al principio del documento y marcar la primera red.
- Pulse CTRL + FIN para desplazarse al final del documento y marcar la última red.
- Pulse RE PÁG para desplazarse 1 pantalla hacia arriba y marcar el rectángulo que está más arriba.
- Pulse AV PÁG para desplazarse 1 pantalla hacia abajo y marcar el rectángulo que está más arriba.

Selección

- Puede seleccionar un elemento, incluida una red, tomando la posición del cursor correspondiente con un clic del ratón o con las teclas de flecha o TAB. Los elementos seleccionados estarán sombreados en rojo. Consulte también *Posiciones de cursor en FBD, LD e IL (véase página 301)*.
- En el editor LD, también puede seleccionar las líneas entre los elementos para ejecutar comandos; por ejemplo, para insertar otro elemento en esa posición.

Línea seleccionada en el editor LD

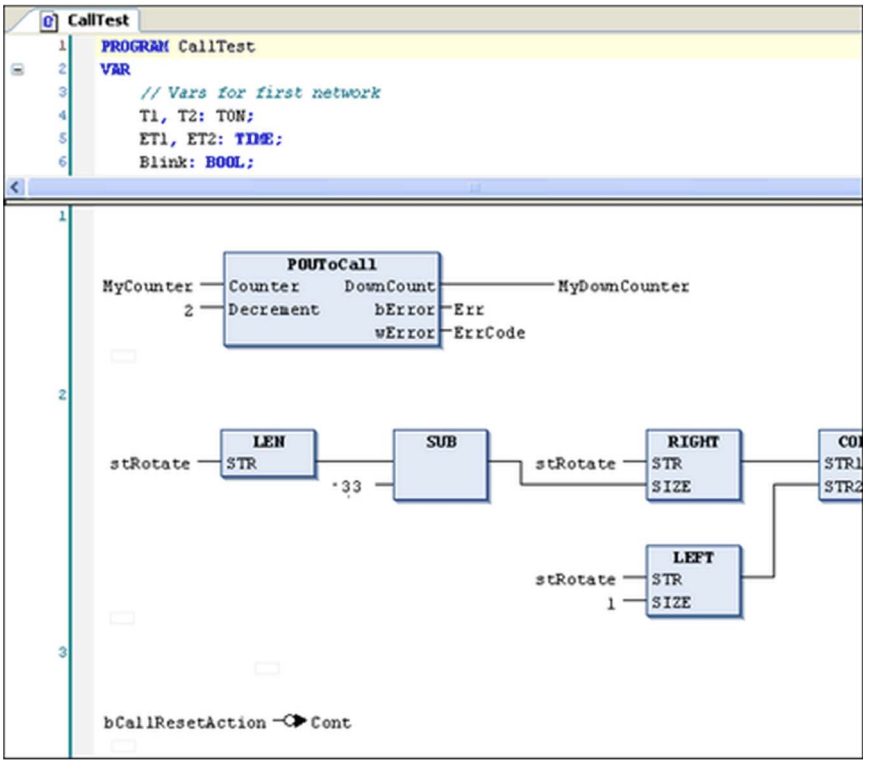


- Se puede efectuar una selección múltiple de elementos o redes; para ello, mantenga pulsada la tecla CTRL mientras selecciona los elementos que desee uno tras otro.

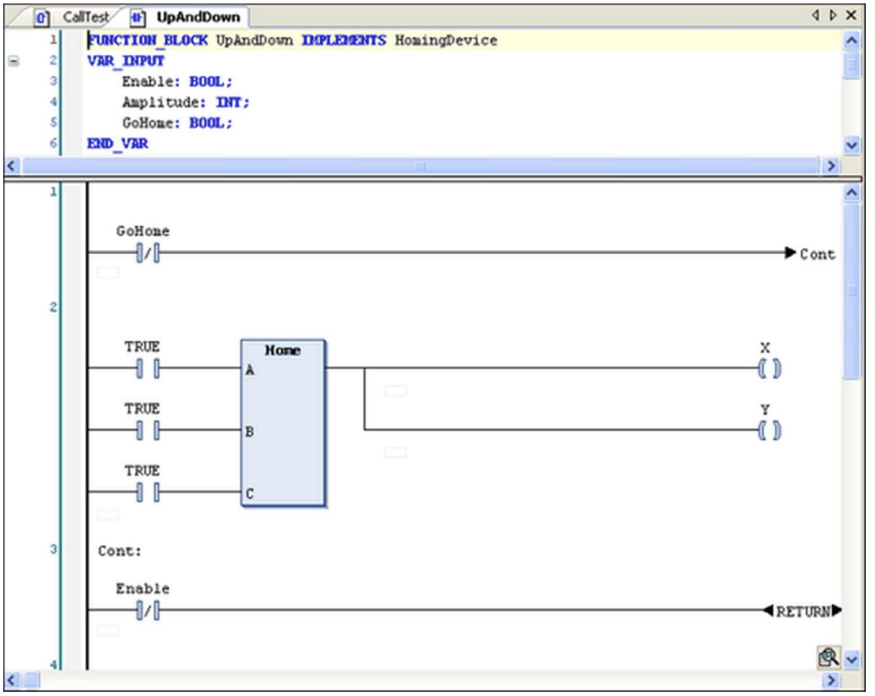
Abrir un bloque de funciones

Si se añade un bloque de funciones al editor, puede abrir este bloque con un doble clic. Como alternativa, utilice el comando **Examinar - Ir a la definición** en el menú contextual.

Editor FBD



Editor LD



Para obtener información sobre los lenguajes, consulte:

- *Diagrama de bloques de funciones (FBD) (véase página 282)*
- *Diagrama de contactos (LD) (véase página 283)*

Trabajo en el editor IL

Descripción general

El editor Lista de instrucciones *(véase página 284)* (IL) es un editor de tablas. La estructura de red de los programas FBD o LD también se representa en un programa IL. Básicamente, una red *(véase página 315)* es suficiente en un programa IL, pero teniendo en cuenta el cambio entre FBD, LD e IL también puede utilizar redes para la estructuración de un programa IL.

Consulte también la configuración general del editor en el cuadro de diálogo **Opciones**, categoría **IL, FBD, LD**.

Información sobre herramientas

La información sobre herramientas contiene información sobre las variables o los parámetros del módulo.

Consulte *Trabajo en el editor FBD y LD (véase página 289)*.

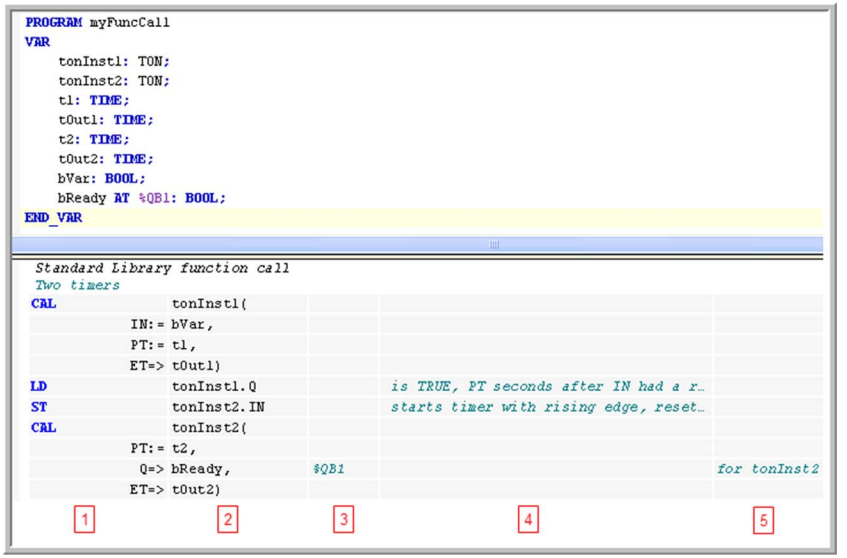
Inserción y organización de los elementos

- Los comandos para trabajar en el editor están disponibles en el menú **IL, FBD, LD**. Los comandos de uso frecuente también están disponibles en el menú contextual.
- Las unidades de programación que sean elementos se insertan cada una en la posición actual del cursor a través de los comandos **Insertar**, disponibles en el menú **IL, FBD, LD**.
- Puede utilizar los comandos cortar, copiar, pegar y eliminar disponibles en el menú **Editar**, para organizar los elementos.
- Consulte también algo de información sobre el lenguaje de programación Lista de instrucciones - IL *(véase página 284)*.
- Los operadores con funcionalidad EN/ENO sólo se pueden insertar dentro del editor FBD y LD.

En este capítulo se describe cómo se estructura el editor de tablas, cómo se puede navegar en el editor y cómo utilizar operandos complejos, llamadas y saltos.

Estructura del editor de tablas IL

Editor de tablas IL



Cada línea de programa está escrita en una fila de la tabla, estructurada en campos por las columnas de la tabla:

Columna	Contiene...	Descripción
1	operador	Este campo contiene el operador IL (LD, ST, CAL, AND, OR, etc.) o un nombre de función. En caso de una llamada de bloque de funciones, los parámetros respectivos también se especifican aquí. Introduzca el campo prefijo := o =>. Para obtener más información, consulte *Modificadores y operadores en IL* *(véase página 286)*.
2	operando	Este campo contiene exactamente un operando o un símbolo de salto. Si se necesita más de un operando (operadores múltiples o extensibles AND A, B, C, o llamadas a funciones con varios parámetros), escríbalos en las siguientes líneas y deje el campo del operador vacío. En este caso, añada una coma de separación de parámetros. En el caso de un bloque de funciones, programa o llamada de acción, añada los paréntesis de apertura o cierre correspondientes.

Columna	Contiene...	Descripción
3	dirección	Este campo contiene la dirección del operando como se define en la parte de declaración. No se puede editar este campo. Utilice la opción **Mostrar dirección de símbolo** para activarlo o desactivarlo.
4	comentario de símbolo	Este campo contiene el comentario definido para el operando en la parte de declaración. No se puede editar este campo. Utilice la opción **Mostrar dirección de símbolo** para activarlo o desactivarlo.
5	comentario de operandos	Este campo contiene el comentario de la línea actual. Es editable y se puede activar o desactivar mediante la opción **Mostrar comentario de operandos**.

Navegación en la tabla

- Teclas de flecha ARRIBA y ABAJO: desplazamiento al campo por encima o por debajo.
- TAB: desplazamiento dentro de una línea al campo de la derecha.
- MAYÚS+TAB: desplazamiento dentro de una línea al campo de la izquierda.
- ESPACIO: abre el campo seleccionado para su edición. Como alternativa, realiza una pulsación del ratón sobre el campo. Si procede, el asistente Accesibilidad estará disponible a través del botón **...** Puede cerrar un campo de edición abierto presionando INTRO, lo que confirma las entradas actuales, o presionando ESC para cancelar las entradas realizadas.
- CTRL+INTRO: introduce una nueva línea debajo de la actual.
- SUPR: borra la línea actual, o sea, donde haya seleccionado actualmente cualquier campo.
- **Cortar**, **Copiar**, **Pegar**: para copiar 1 o varias líneas, seleccione al menos 1 campo de la línea o líneas y ejecute el comando **Copiar**. Para cortar una línea, utilice el comando **Cortar**. **Pegar** insertará las líneas copiadas o cortadas previamente antes de la línea donde actualmente se ha seleccionado un campo. Si no se ha seleccionado ningún campo, se insertarán en el extremo de la red.
- CTRL+INICIO desplaza hasta el principio del documento y marca la primera red.
- CTRL+FIN desplaza hasta el final del documento y marca la última red.
- RE PÁG desplaza hacia arriba 1 pantalla y marca el rectángulo superior.
- AV PÁG desplaza hacia abajo 1 pantalla y marca el rectángulo superior.

Operandos múltiples (operadores extensibles)

Si el mismo operador *(véase página 286)* se utiliza con varios operandos, se pueden emplear 2 formas de programación:

- Los operandos se introducen en las líneas siguientes, separados por comas, por ejemplo:

```
LD      7
ADD     2,
        4,
        7
ST      iVar
```

- La instrucción se repite en las líneas siguientes, por ejemplo:

```
LD      7
ADD     2
ADD     4
ADD     7
ST      iVar
```

Operandos complejos

Si se va a utilizar un operando complejo, escriba antes un paréntesis de apertura y, a continuación, utilice las siguientes líneas para los componentes particulares del operando. Por debajo de ellos, en una línea distinta, introduzca el paréntesis de cierre.

Ejemplo: Rotación de una cadena de 1 carácter en cada ciclo.

Código ST correspondiente:

```
stRotate := CONCAT(RIGHT(stRotate, (LEN(stRotate) -
 1)), (LEFT(stRotate, 1)));
```

```
LD      stRotate
RIGHT(  stRotate
LEN
SUB     1
)
CONCAT( stRotate
LEFT    1
)
ST      stRotate
```

Llamadas a funciones

Introduzca el nombre de la función en el campo del operador. Proporcione el parámetro de entrada (primero) como operando en una operación LD precedente. Si hay más parámetros, proporcione el siguiente en la misma línea de la del nombre de la función. Puede añadir más parámetros en esta línea, separados por comas o en líneas posteriores.

El valor de retorno de la función se almacenará en el acumulador. Se aplica la siguiente restricción relativa a la norma IEC.

NOTA: No es posible una llamada de función con varios valores de retorno. Sólo se puede utilizar 1 valor de retorno para una operación subsiguiente.

Ejemplo: Se llama a la función `GeomAverage`, que tiene 3 parámetros de entrada. El primer parámetro es dado por `X7` en una operación precedente. El segundo, `25`, es dado por el nombre de la función. El tercero es dado por la variable `tvar`, en la misma línea o en la posterior. El valor de retorno se asigna a la variable `Ave`.

Código ST correspondiente:

```
Ave := GeomAverage(X7, 25, tvar);
```

Llamada de función en IL:

```
LD              X7
GeomAverage     25
                tvar
ST              Ave
```

Llamadas de bloque de funciones y llamadas de programa

Utilice el operador *(véase página 286)* CAL- o CALC. Introduzca el nombre de la instancia del bloque de funciones o el nombre del programa en el campo de operando (segunda columna), seguido de un paréntesis de apertura. Introduzca los parámetros de entrada en una de las siguientes líneas:

Campo del operador: nombre del parámetro

Campo de prefijo:
- `:=` para los parámetros de entrada
- `=>` para los parámetros de salida

Campo de operando: parámetro actual

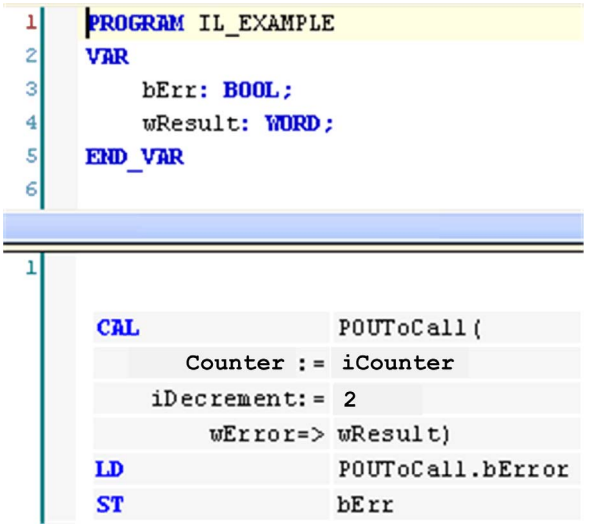
Campo de sufijo:
- `,` si siguen otros parámetros
  `)` después del último parámetro
- `()` en caso de llamadas sin parámetros

Ejemplo: Llamada de `POUToCAll` con 2 parámetros de entrada y 2 parámetros de salida.

Código ST correspondiente:

```
POUToCall(Counter := iCounter, iDecrement:=2, bError=>bErr, wError=>wRe
sult);
```

Llamada de programa en IL con los parámetros de entrada y de salida:



```
PROGRAM IL_EXAMPLE
VAR
    bErr: BOOL;
    wResult: WORD;
END_VAR
```

```
CAL             POUToCall(
    Counter := iCounter
  iDecrement:= 2
      wError=> wResult)
LD              POUToCall.bError
ST              bErr
```

No es necesario utilizar todos los parámetros de un bloque de funciones o de un programa.

NOTA: No se pueden utilizar expresiones complejas. Éstas se deben asignar a la entrada del bloque de funciones o programa antes de la instrucción de llamada.

Llamada de acción

Para realizarse como un bloque de funciones o una llamada de programa, el nombre de la acción se va a añadir al nombre de la instancia o del programa.

Ejemplo: Llamada de acción `ResetAction`.

Código ST correspondiente:

```
Inst.ResetAction();
```

Llamada de acción en IL:

```
CAL      Inst.ResetAction()
```

Llamada de método

Para realizarse como una llamada de función, el nombre de la instancia con el nombre de método anexado se introducirá en la primera columna (operador).

Ejemplo: Llamada de método `Home`.

Código ST correspondiente:

```
Z := IHome.Home(TRUE, TRUE, TRUE);
```

Llamada de método en IL:

```
LD              TRUE
IHome.Home      TRUE
                TRUE
ST              Z
```

Salto

Un salto *(véase página 318)* se programa con `JMP` en la primera columna (operador) y un nombre de etiqueta en la segunda columna (operando). La etiqueta se ha de definir en la red de destino en el campo etiqueta *(véase página 319)*.

La lista de instrucciones que precede al salto incondicional tiene que terminar con uno de los siguientes comandos: ST, STN, S, R, CAL, RET u otro JMP.

Este no es el caso de un salto condicional *(véase página 318)*. La ejecución del salto depende del valor cargado.

Ejemplo: Instrucción de salto condicional; en caso de que `bCallRestAction` sea TRUE, el programa debe saltar a la red marcada con `Cont`.

Instrucción de salto condicional en IL:

```
LDN         bCallResetAction
JMPC        Cont
```

Posiciones de cursor en FBD, LD e IL

Editor IL

Se trata de un editor de texto, estructurado en forma de tabla. Cada celda de la tabla es una posible posición del cursor. Consulte también *Trabajo en la vista del editor IL* *(véase página 294)*.

Editor FBD y LD

Estos son editores gráficos; vea a continuación los ejemplos (1) a (15) que muestran las posibles posiciones del cursor: texto, entrada, salida, módulo, contacto, bobina, retorno, salto y línea entre los elementos y la red.

Acciones tales como cortar, copiar, pegar, borrar y otros comandos específicos del editor se refieren a la posición actual del cursor o elemento seleccionado. Consulte *Trabajo en el editor FBD y LD* *(véase página 289)*.

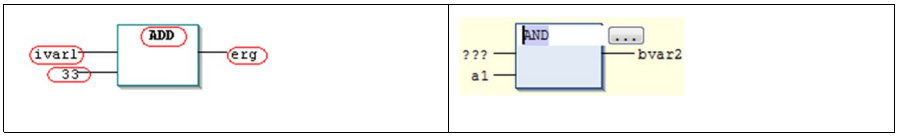
Básicamente, en FBD un rectángulo de puntos alrededor del elemento respectivo indica la posición actual del cursor. Además, los textos y los módulos se vuelven de color azul o rojo sombreado.

En LD, las bobinas y los contactos se vuelven de color rojo sombreado en cuanto el cursor se sitúa sobre ellos.

La posición del cursor determina qué elementos se encuentran disponibles en el menú contextual para ser insertados *(véase página 290)*.

Posibles posiciones del cursor

(1) En cada campo de texto



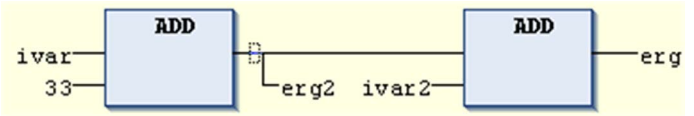
En la imagen de la izquierda, las posibles posiciones del cursor están marcadas por un marco rojo. La imagen de la derecha muestra un módulo con el cursor colocado en el campo `AND`. Tenga en cuenta la posibilidad de introducir direcciones en lugar de nombres de variables si está configurada adecuadamente en el cuadro de diálogo **Opcionesdel editor FBD, LD e IL**.

(2) En cada entrada:

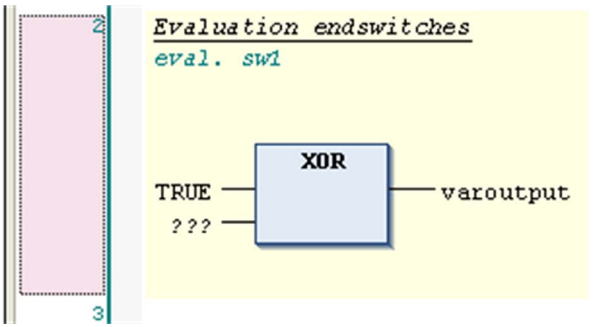
(3) En cada operador, función o bloque de funciones:

(4) En las salidas, si viene posteriormente una asignación o un salto:

(5) Justo antes de una bifurcación sobre una asignación antes de un salto o una instrucción de retorno:

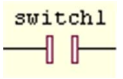


(6) En la posición más a la derecha del cursor en la red o en cualquier parte de la red fuera de las otras posiciones del cursor. Esto seleccionará toda la red:



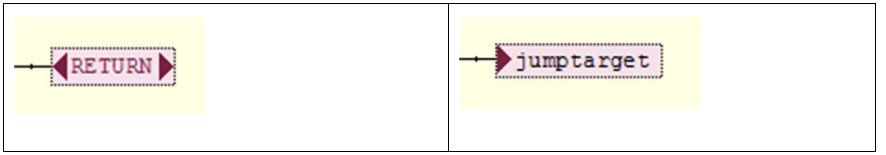
(7) En la bifurcación directamente delante de una asignación:

(8) En cada contacto:

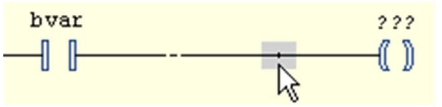


(9) En cada bobina:

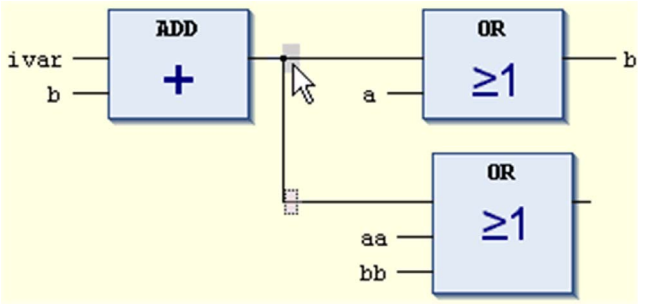
(10) En cada retorno y salto:



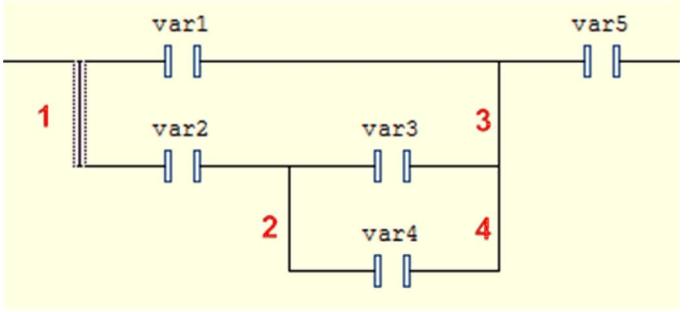
(11) En la línea de conexión entre los contactos y las bobinas:



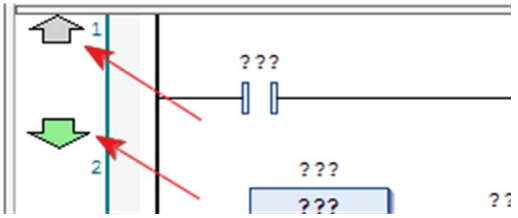
(12) En cada rama o subred dentro de una red:



(13) En la línea de conexión entre los contactos paralelos (pos. 1...4):

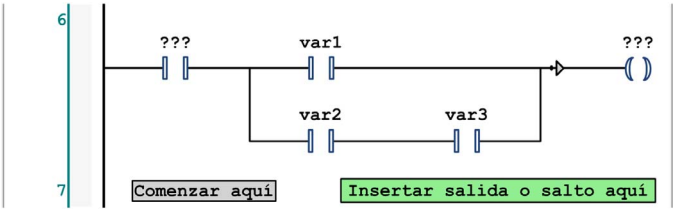


(14) Antes o después de una red:



Puede añadir nuevas redes en el lado más a la izquierda del editor. La inserción de una nueva red antes de una red existente sólo es posible antes de la red 1.

(15) Al principio o al final de una red:



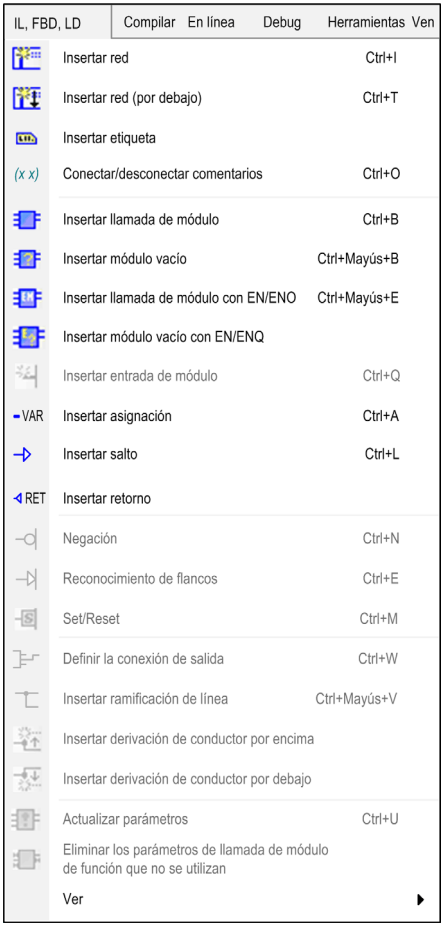
Puede añadir contactos y bloques de funciones al principio de una red en el campo **Comenzar aquí**. Puede añadir los elementos retorno, salto y bobina al final de una red en el campo **Insertar salida o salto aquí**.

Menú IL, FBD, LD

Descripción general

Cuando el cursor se coloca en la ventana del editor FBD/LD/IL *(véase página 281)*, el menú IL, FBD, LD está disponible en la barra de menús, lo que proporciona los comandos para programar en la vista de editor establecida actualmente.

Menú IL, FBD, LD en la vista del editor FBD:



- Para obtener una descripción de los comandos, consulte el capítulo *Comandos del editor FBD/LD/IL*.
- Para obtener la configuración del menú, consulte la descripción del menú **Personalizar**.

Editor FBD/LD/IL en modalidad online

Descripción general

En la modalidad online, el editor FBD/LD/IL proporciona vistas para la supervisión *(véase página 306)* y para la escritura y el forzado de las variables y expresiones en el controlador.

Hay funciones de depuración disponibles (puntos de interrupción, ejecución paso a paso, etc.); consulte *Posiciones de punto de interrupción o de parada (véase página 310)*.

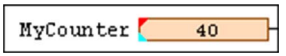
- Para obtener información acerca de la apertura de objetos en modalidad online, consulte el capítulo *Interfaz de usuario en modalidad online (véase página 49)*.
- Tenga en cuenta que la ventana del editor de un objeto FBD, LD o IL también incluye el **Editor de declaraciones** en la parte superior. Consulte también el capítulo *Editor de declaraciones en modalidad online (véase página 419)*.

Supervisión

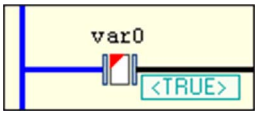
Si la supervisión en línea no se desactiva explícitamente en el cuadro de diálogo **Opciones**, se complementará en el editor FBD o LD con pequeñas ventanas de supervisión detrás de cada variable o mediante una columna de supervisión adicional que muestra los valores reales (supervisión en línea). Esto es así incluso para entradas y salidas de bloques de funciones sin asignar.

La ventana de supervisión en línea de una variable muestra un pequeño triángulo rojo en la esquina superior izquierda si la variable está forzada *(véase página 309)* actualmente y un triángulo azul en la esquina inferior izquierda si está preparada actualmente para la escritura o el forzado. En LD para contactos y bobinas, el valor preparado actualmente (TRUE o FALSE) aparecerá justo debajo del elemento.

Ejemplo de una variable que está actualmente forzada y preparada para suprimir el forzado



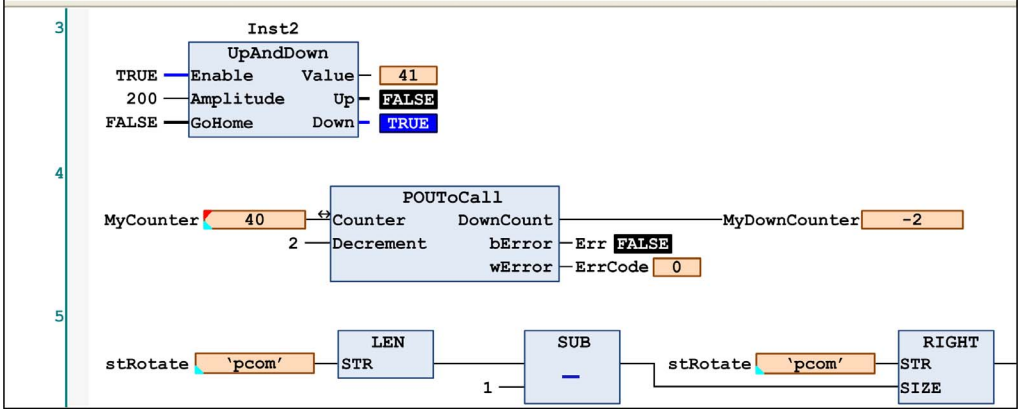
Ejemplo de variable de contacto preparada actualmente para la escritura o el forzado con el valor TRUE



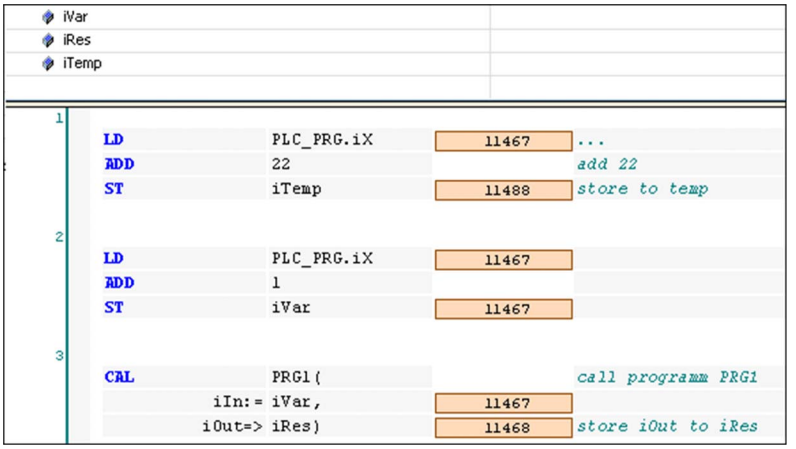
Vista online de un programa FBD

SP_for_Win32.App.CallTest

Expresión	Comentario	Tipo	Valor	Valor preparado
Inst2		UpAndDown		
Enable		BOOL	TRUE	
Amplitude		INT	200	
GoHome		BOOL	FALSE	
Value		INT	41	
Up		BOOL	FALSE	
Down		BOOL	TRUE	
Counter		DINT	2159	
diValue		DINT	41	
X		BOOL	FALSE	
X		INT	0	
X2		INT	0	
Up		BOOL	TRUE	
Down		BOOL	FALSE	
MyCounter	Variables para tercera red	DINT	F 40	<Anular forzado y rehacer>
MyDownCounter		DINT	-2	
Err		BOOL	FALSE	
ErrCode		WORD	0	
stRotate	Variables para quinta red	STRING	'pcom'	'abc'
Ave	Variables para quinta red	DINT	38	



Vista online de un programa IL

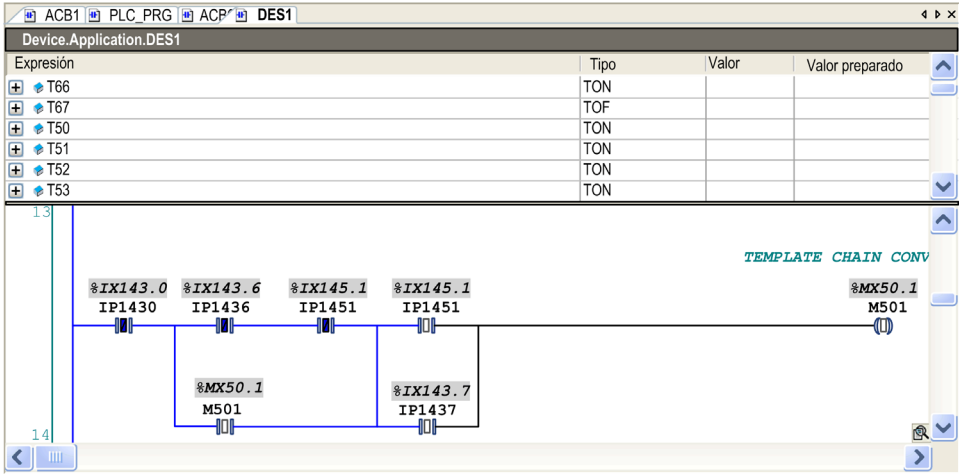


En la vista online, las redes LD tienen conexiones animadas:

- Las conexiones con el valor TRUE se muestran en color azul y en negrita.
- Las conexiones con el valor FALSE se muestran en color negro y en negrita.
- Las conexiones sin valor conocido o con un valor analógico se muestran con un contorno estándar (en color negro y sin negrita).

Los valores de las conexiones se calculan a partir de los valores de supervisión.

Vista online de un programa LD

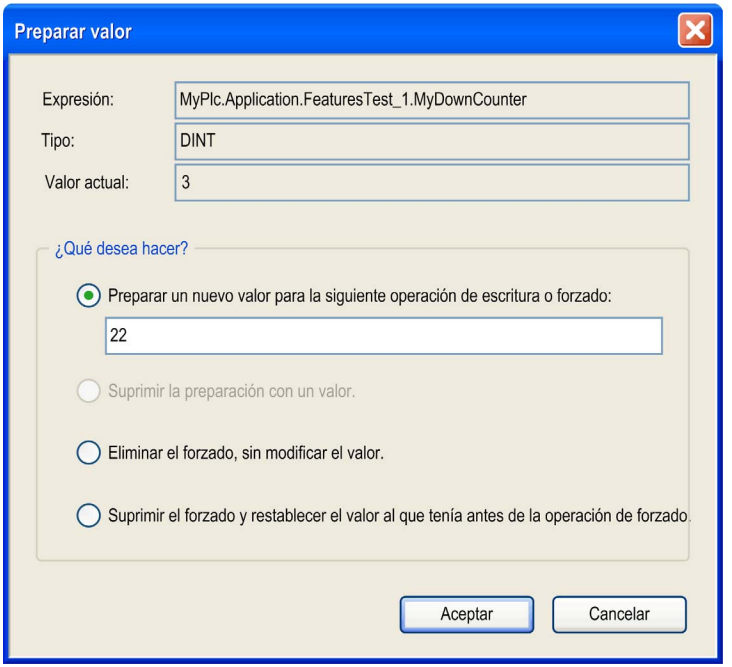


Abra una función haciendo doble clic o ejecute el comando **Examinar - Ir a la definición** en el menú contextual. Consulte la descripción de la *Interfaz de usuario en modalidad online* *(véase página 49)* para obtener más información.

Forzado/escritura de variables

En la modalidad online, puede preparar un valor para forzar o escribir una variable en el editor de declaraciones *(véase página 419)* o dentro del editor. Haga doble clic en una variable en el editor para abrir el cuadro de diálogo siguiente:

Cuadro de diálogo **Preparar valor**



Encontrará el nombre de la variable completada por su ruta en el árbol de dispositivos (**Expresión**), su tipo y su valor actual. Al activar el elemento correspondiente, puede hacer lo siguiente:

- Preparar un valor nuevo que se debe introducir en el campo de edición.
- Eliminar un valor preparado.
- Levantar el forzado de la variable.
- Levantar el forzado de la variable y restablecerla al valor que tenía asignado justo antes del forzado.

La acción seleccionada se realizará al ejecutar el comando de menú **Forzar valores** (en el menú **En línea**) o pulsando F7.

Para obtener información sobre cómo se indica el estado actual de una variable (forzado, valor preparado) en el elemento respectivo en la red, consulte la sección *Supervisión (véase página 306)*.

Posiciones de punto de interrupción o de parada

Las posiciones posibles que se pueden definir para un punto de interrupción (posición de parada) con fines de depuración son las posiciones en las que los valores de las variables pueden cambiar (instrucciones), en los que el flujo del programa se bifurca o en los que se llama a otra POU.

Son las posiciones siguientes:

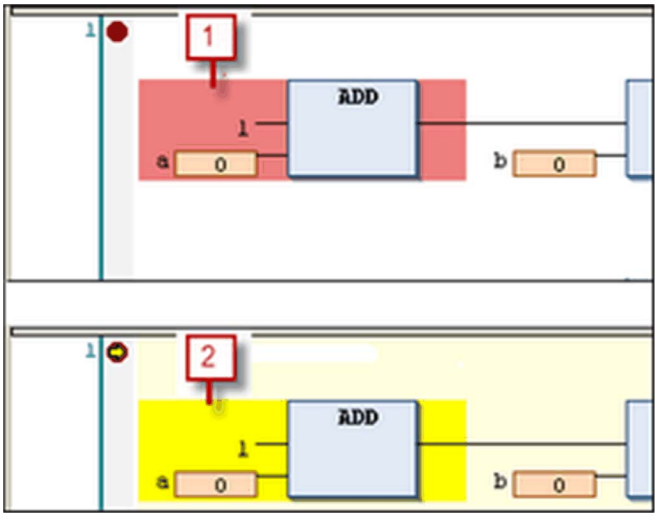
- En la red en su conjunto de modo que el punto de interrupción se aplicará a la primera posición posible en la red.
- En un módulo *(véase página 320)*, si contiene una instrucción. Por lo tanto, no es posible en módulos de operadores como, por ejemplo, `ADD, DIV`. Consulte la nota siguiente.
- En una asignación.
- Al final de una POU en el punto de retorno al llamante; en modalidad online, se mostrará automáticamente una red vacía con este fin. En lugar de un número de red, se identifica mediante `RET`.

NOTA: No se puede establecer un punto de interrupción directamente en el primer módulo de una red. Si, no obstante, se ha establecido un punto de interrupción sobre la red completa, la posición de parada se aplicará automáticamente al primer módulo.

Para conocer las posiciones posibles actualmente, consulte la lista de selección del cuadro de diálogo **Visualizar** →**Puntos de interrupción**.

Una red que contiene cualquier posición de punto de interrupción activa se marca con el símbolo de punto de interrupción (círculo con relleno rojo) a la derecha del número de red y un fondo rectangular sombreado en rojo para la primera posición de punto de interrupción posible en la red. Las posiciones de puntos de interrupción desactivados se indican mediante un círculo rojo sin relleno o un rectángulo rojo circundante sin relleno.

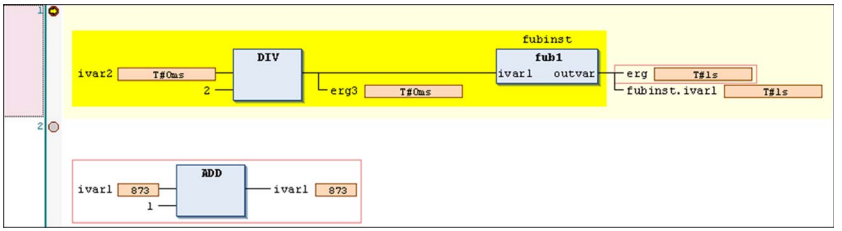
Punto de interrupción establecido y punto de interrupción alcanzado



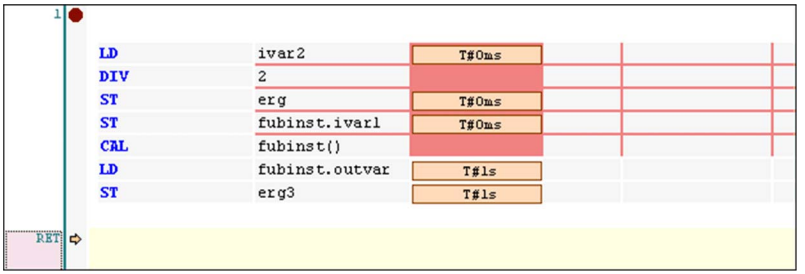
1 punto de interrupción establecido
2 punto de interrupción alcanzado

Cuando se llegue a una posición de punto de interrupción durante la ejecución paso a paso o el procesamiento del programa, se mostrará una flecha amarilla en el símbolo del punto de interrupción y el área sombreada en rojo pasará a ser amarilla.

Posiciones de parada mostradas en FBD



Posición de parada mostrada en IL



NOTA: Se establecerá automáticamente un punto de interrupción en todos los métodos que puedan llamarse. Si se llama un método gestionado por interfaces, los puntos de interrupción se establecerán en todos los métodos de bloques de funciones que implementen esa interfaz y en todos los bloques de funciones derivados que suscriban el método. Si se llama un método mediante un puntero en un bloque de funciones, se establecerán puntos de interrupción en el método del bloque de funciones y en todos los bloques de funciones derivados que suscriban el método.

Sección 9.2
Elementos FBD/LD/IL

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Herramientas FBD/LD/IL

Descripción general

El editor FBD/LD/IL *(véase página 281)* proporciona herramientas que ofrecen elementos de programación para su inserción en la ventana del editor mediante el método de arrastrar y soltar. Abra las herramientas ejecutando el comando **Herramientas** que está en el menú **Visualizar**.

Los elementos disponibles para insertar dependerán de la vista de editor activa actualmente (consulte la descripción correspondiente de los comandos de inserción).

Los elementos se clasifican en categorías: **General** (elementos generales como **Red, Asignación**, etc.), **Operadores lógicos**, **Operadores matemáticos**, **Otros operadores** (por ejemplo, `SEL, MUX, LIMIT` y `MOVE`), **Bloques de funciones** (por ejemplo, `R_TRIG, F_TRIG, RS, SR, TON, TOF, CTD, CTU`), **Elementos de diagrama de contactos** y **POU** (definido por el usuario).

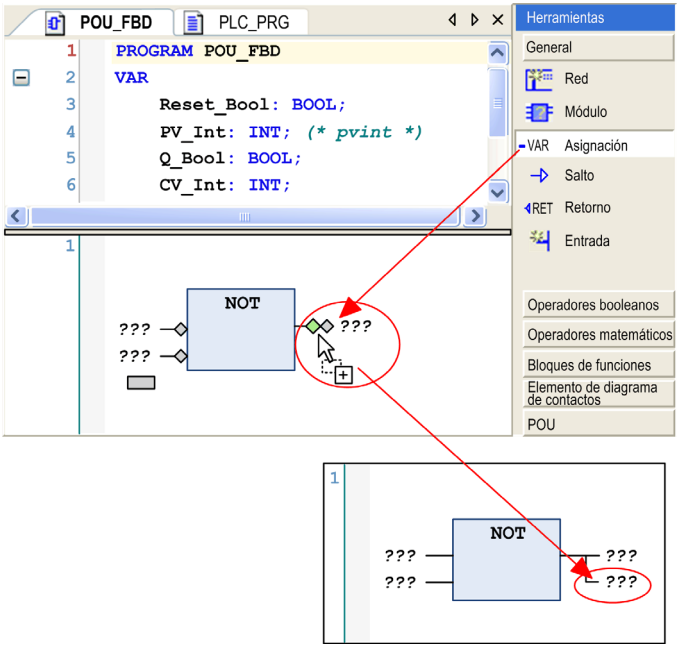
La categoría **POU** muestra todas las POU que se han definido bajo la misma aplicación que el objeto FBD/LD/IL, que está abierto en el editor. Si se ha asignado un mapa de bits a una POU en sus propiedades, se mostrará delante del nombre de POU. En caso contrario, se utilizará el icono estándar para indicar el tipo de POU. La lista se actualizará automáticamente cuando se añadan o eliminen POU de la aplicación.

La categoría **Otros operadores** contiene entre los operadores `SEL, MUX, LIMIT` y `MOVE` un elemento de conversión de marcadores de posición. Puede arrastrar este elemento y colocarlo en la posición apropiada de la red. El tipo de conversión se establece automáticamente en función del tipo necesario de la posición de inserción. Sin embargo, en algunas situaciones no se puede determinar automáticamente el tipo de conversión necesario. En este caso, cambie manualmente el elemento.

Para desplegar las carpetas de categorías, haga clic en el botón que muestra el nombre de categoría respectivo. Como se puede ver en la imagen siguiente, la categoría **General** está desplegada, mientras que las demás están contraídas. En la imagen se muestra un ejemplo de inserción de un elemento **Asignación** mediante el método de arrastrar y soltar desde las herramientas.

Sólo la sección **General** de las herramientas está desplegada:

Insertar desde las herramientas



Para insertar un elemento en el editor, selecciónelo en las herramientas haciendo clic en él con el ratón y, a continuación, arrástrelo y suéltelo en la ventana del editor. Las posibles posiciones de inserción se indicarán mediante marcadores de posición, que aparecen mientras se dibuje el elemento (con el botón del ratón presionado) en la ventana del editor. Las posiciones posibles más cercanas se indicarán en verde. Al soltar el botón del ratón, el elemento se insertará en la posición verde.

Si arrastra un elemento de módulo a un elemento de módulo existente, el nuevo sustituirá al antiguo. Si ya se han asignado entradas y salidas, permanecerán según se han definido, pero no se conectarán al nuevo módulo.

Red en FBD/LD/IL

Descripción general

Una red es la entidad básica de un programa FBD *(véase página 282)* o LD *(véase página 283)*. En el editor FBD/LD, las redes se muestran en una lista vertical. Cada red está indicada en el lado izquierdo con un número de red de serie y tiene una estructura que consta de una expresión lógica o aritmética, un programa, función o llamada de bloque de funciones y, en algunos casos, instrucciones de retorno o salto.

El Editor IL *(véase página 284)*, dada la base de editor común con los editores FBD y LD, también utiliza el elemento de red. Si un objeto inicialmente se programó en FBD o LD y posteriormente se convirtió a IL, las redes seguirán estando presentes en el programa IL. Por el contrario, si empezó programando un objeto en IL, necesitará como mínimo 1 elemento de red que pueda contener todas las instrucciones, pero también puede utilizar redes adicionales para estructurar el programa.

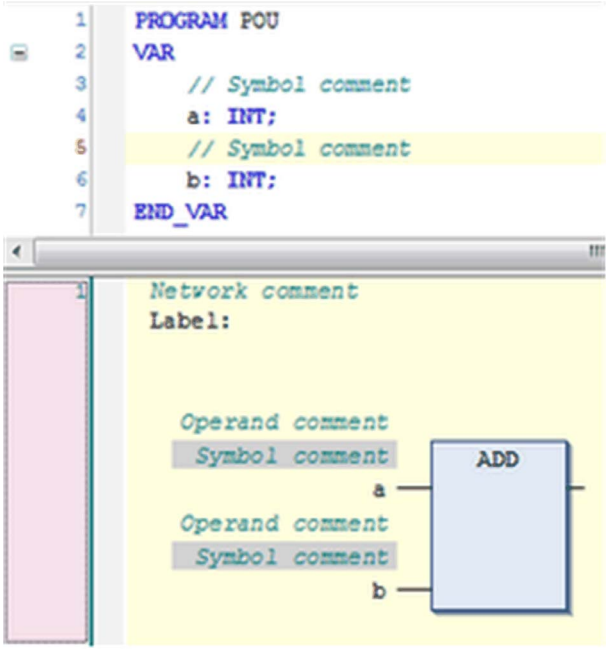
Opcionalmente, se puede asignar un título, un comentario y una etiqueta *(véase página 319)* a una red.

Puede activar o desactivar la disponibilidad de los campos de título y comentario en el cuadro de diálogo de opciones del editor FBD, LD e IL. Si la opción está activada, haga clic en la red justo debajo del borde superior para abrir un campo de edición para el título. Asimismo, para introducir un comentario, abra un campo de edición directamente debajo del campo de título. El comentario puede tener varias líneas. Pulse INTRO para insertar saltos de línea. Pulse CTRL + INTRO para finalizar la introducción del texto del comentario.

La visualización o no de un comentario de red y la forma en que se muestra en el editor se define en el cuadro de diálogo de opciones del editor FBD, LD e IL.

Para añadir una etiqueta *(véase página 319)* y asignarle un salto *(véase página 318)*, utilice el comando **Insertar etiqueta**. Si se define una etiqueta, se mostrará debajo del campo de título y comentario. Y si estos campos no están disponibles, directamente debajo del borde superior de la red.

Comentarios y etiqueta de una red



Puede establecer una red como comentario. Esto indica que la red no se procesa, sino que se muestra y se gestiona como un comentario. Para ello, utilice el comando **Conectar/desconectar comentarios**.

En una red seleccionada actualmente (posición del cursor 6 *(véase página 301)*), puede aplicar los comandos predeterminados para copiar, cortar, insertar y eliminar.

NOTA: Al hacer clic con el botón derecho (posición del cursor 6 *(véase página 301)*) en títulos, comentarios o etiquetas, se seleccionará únicamente esa entrada en lugar de toda la red. De este modo, la ejecución de los comandos predeterminados no afecta a la red.

Para insertar una red, utilice el comando **Insertar red** o arrástrela desde las herramientas *(véase página 313)*. Una red, junto con todos sus elementos, también se puede copiar o mover *(véase página 289)* mediante el método de arrastrar y soltar en el editor.

También puede crear subredes *(véase página 322)* insertando bifurcaciones.

Red RET

En la modalidad online, se mostrará una red vacía adicional debajo de las redes existentes. En lugar de tener asignado un número de red, se identifica mediante RET.

Representa la posición en la que la ejecución regresará a la POU de llamada y proporciona una posición de parada *(véase página 306)* posible.

Asignación en FBD/LD/IL

Descripción general

En función de la posición del cursor *(véase página 301)* seleccionada en FBD o en LD, se insertará una asignación directamente delante de la entrada seleccionada (posición del cursor 2 *(véase página 301)*), directamente después de la salida seleccionada (posición del cursor 4 *(véase página 301)*) o al final de la red (posición del cursor 6 *(véase página 301)*). En una red LD, la asignación se mostrará como una bobina *(véase página 332)*. Como alternativa, puede arrastrar el elemento de asignación desde las herramientas *(véase página 313)* o bien utilizar el método de arrastrar y soltar para copiarlo o moverlo *(véase página 289)* a la vista del editor.

Tras la inserción, la cadena de texto `???` puede sustituirse por el nombre de la variable que se va a asignar. Para ello, utilice el botón **...** para abrir el asistente **Accesibilidad**.

En IL *(véase página 284)*, las asignaciones se programan mediante instrucciones LD y ST. Consulte *Modificadores y operadores en IL (véase página 286)*.

Salto en FBD/LD/IL

Descripción general

En función de la posición del cursor *(véase página 301)* seleccionada en FBD *(véase página 282)* o en LD *(véase página 283)*, se insertará un salto directamente delante de la entrada seleccionada (posición del cursor 2), directamente después de la salida seleccionada (posición del cursor 4) o al final de la red (posición del cursor 6). Como alternativa, puede arrastrar el elemento de salto desde las herramientas *(véase página 313)* o bien utilizar el método de arrastrar y soltar para copiarlo o moverlo *(véase página 289)* al editor.

Tras la inserción, puede sustituir el texto `???` introducido automáticamente por la etiqueta a la cual debe asignarse el salto.

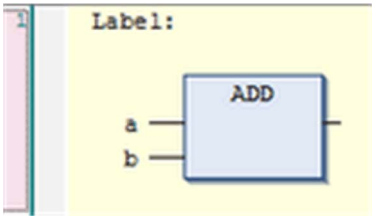
En IL *(véase página 284)*, el salto se inserta mediante una instrucción JMP. En este contexto, consulte la descripción de los modificadores y operadores en IL *(véase página 286)*.

Etiqueta en FBD/LD/IL

Descripción general

Debajo del campo de comentario de red, cada red FBD *(véase página 282)*, LD *(véase página 283)* o IL tiene un campo de entrada de texto para definir una etiqueta. La etiqueta es un identificador opcional de la red y se puede direccionar cuando se define un salto *(véase página 318)*. Puede estar formada por una secuencia de caracteres cualquiera.

Posición de una etiqueta en una red



Consulte el cuadro de diálogo **Herramientas** →**Opciones** →**FBD, LD e IL** para definir la pantalla de comentario y título.

Módulos en FBD/LD/IL

Descripción general

Un módulo, que se puede insertar en una red FBD *(véase página 282)*, LD *(véase página 283)* o IL *(véase página 284)*, es un elemento complejo y puede representar funciones adicionales como temporizadores, contadores, operaciones aritméticas o también programas, funciones IEC y bloques de funciones IEC.

Un módulo puede tener una o varias entradas y salidas; lo puede proporcionar una biblioteca del sistema o se puede programar. Sin embargo, al menos 1 entrada y 1 salida deben asignarse a valores booleanos.

Si se proporciona con el módulo respectivo y si la opción **Mostrar símbolo de módulo** está activada, se mostrará un icono dentro del módulo.

Uso en FBD, LD

Puede colocar un módulo en una red LD o en una red FBD mediante el comando **Insertar llamada de módulo**, **Insertar módulo vacío**. Como alternativa, puede insertarlo desde el cuadro de herramientas *(véase página 313)* o bien copiarlo o moverlo dentro del editor mediante la acción de arrastrar y soltar. Para obtener más información, consulte la descripción del comando **Insertar llamada de módulo**.

Uso en IL

En un programa IL *(véase página 284)*, se insertará una instrucción CAL *(véase página 286)* con parámetros para representar un elemento de módulo.

Puede actualizar los parámetros del módulo (entradas, salidas), en caso de que la interfaz de módulo se haya modificado, con la implementación actual sin tener que volver a insertar el módulo; para ello, ejecute el comando **Actualizar parámetros**.

Instrucción RETURN en FBD/LD/IL

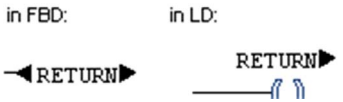
Descripción general

Con una instrucción RETURN, se puede salir de la POU de FBD *(véase página 282)*, LD *(véase página 283)* o IL *(véase página 284)*.

En una red FBD o LD, puede colocarla en paralelo o al final de los elementos previos. Si la entrada de una instrucción RETURN es TRUE, se saldrá inmediatamente del procesamiento de la POU.

Ejecute el comando **Insertar retorno** para insertar una instrucción RETURN. Como alternativa, puede arrastrar el elemento desde las herramientas *(véase página 313)* o bien lo puede copiar o mover *(véase página 289)* desde otra posición en el editor.

Elemento RETURN



En IL, la instrucción RET *(véase página 286)* se usa para el mismo fin.

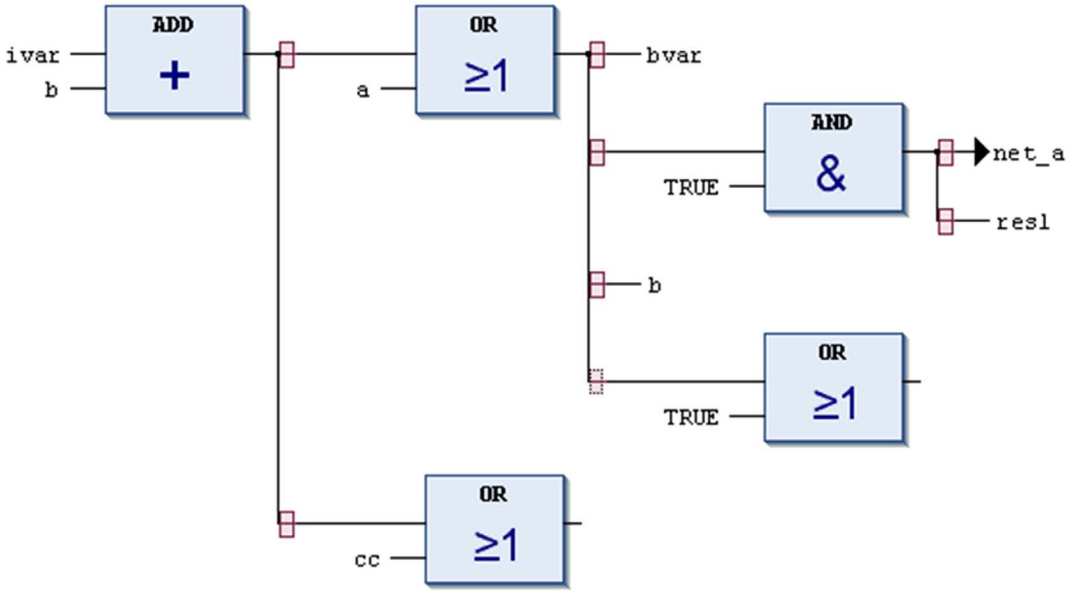
Bifurcación/Bobina en paralelo en FBD/LD/IL

Descripción general

En una red de Diagrama de bloques de funciones *(véase página 282)* o de Diagrama de contactos *(véase página 283)*, una bifurcación o una bobina en paralelo dividirá la línea de procesamiento a partir de la posición actual del cursor. La línea de procesamiento continuará en dos subredes que se ejecutarán una tras la otra de arriba abajo. Cada subred puede tener una bifurcación adicional y, por tanto, habrá varias bifurcaciones en una red.

Cada subred tendrá su propio marcador (el símbolo es un rectángulo vertical). Puede seleccionarlo (posición del cursor 11 *(véase página 301)*) para realizar acciones en esta rama de la bifurcación.

Marcadores de bifurcación

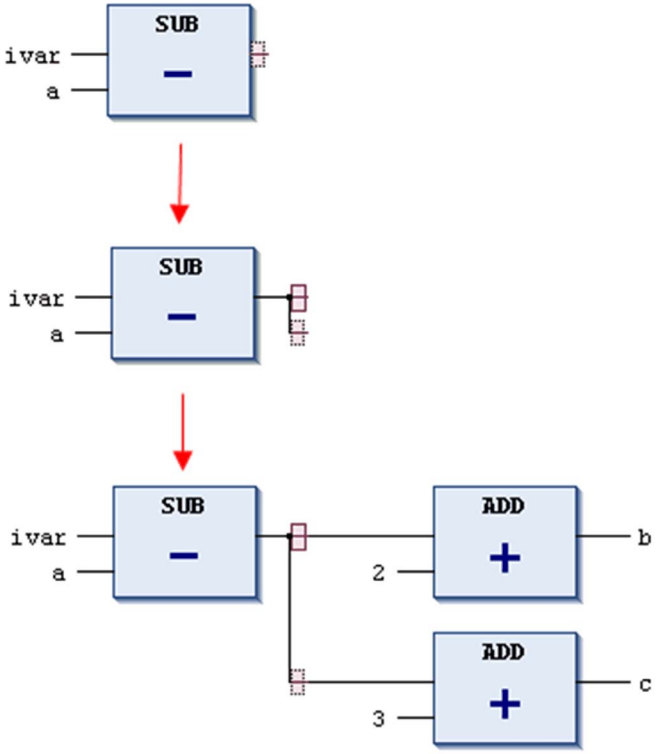


En FBD, inserte una bifurcación mediante el comando **Insertar rama**. Como alternativa, arrastre el elemento desde las herramientas *(véase página 313)*. Sobre las posibles posiciones de inserción, consulte la descripción del comando **Insertar rama**.

NOTA: En las subredes no puede utilizarse el método de cortar y pegar.

En el ejemplo siguiente, se ha insertado una bifurcación en la salida del módulo `SUB`. Esta acción ha creado dos subredes, y cada una se puede seleccionar mediante su marcador de subred. Después, se añadió un módulo `ADD` en cada subred.

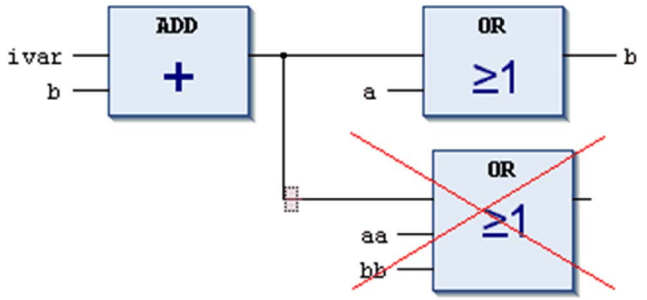
Red en FBD con una bifurcación insertada



Para eliminar una subred, quite primero todos los elementos de la subred, es decir, todos los elementos que están situados a la derecha del marcador de la subred. A continuación, seleccione el marcador y ejecute el comando **Eliminar** estándar o pulse la tecla SUPR.

En la imagen siguiente, el elemento OR de 3 entradas tiene que borrarse primero para que pueda seleccionar y eliminar el marcador de la subred inferior.

Eliminar bifurcación o subred



Ejecución en modalidad online

Cada rama se ejecutará de izquierda a derecha y, después, de arriba abajo.

IL (lista de instrucciones)

La IL *(véase página 284)* no admite redes con bifurcaciones. Se conservarán en su representación original.

Bifurcaciones simultáneas

Para configurar la evaluación de una bifurcación simultánea *(véase página 325)* en redes LD, puede utilizar bifurcaciones simultáneas.

A diferencia de una rama abierta (sin punto de unión), las bifurcaciones simultáneas están cerradas. Tienen puntos de división y de unión comunes.

Bifurcación simultánea

Descripción general

Una bifurcación simultánea le permite implementar una evaluación paralela de los elementos lógicos. Esto se lleva a cabo mediante una metodología descrita como evaluación de cortocircuitos (SCE). SCE le permite eludir la ejecución de un bloque de funciones con una salida booleana si determinadas condiciones paralelas se evalúan como verdaderas. La condición puede representarse en el editor LD mediante una bifurcación paralela a la rama del bloque de funciones. La condición SCE se define mediante 1 o varios contactos con esta bifurcación, conectados en paralelo o en serie.

NOTA: El término rama también se utiliza para otro elemento que divide el flujo de una señal. Esta rama *(véase página 322)*, a diferencia de la bifurcación paralela, no tiene un punto de unión.

La bifurcación paralela funciona como sigue: primero se analizan las ramas que no contienen un bloque de funciones. Si 1 de estas ramas se evalúa como verdadera, no se invocará el bloque de funciones de la bifurcación paralela y el valor de la entrada de la rama del bloque de funciones se pasará a la salida. Si la condición de SCE se evalúa como falsa, se invocará el bloque de funciones y se pasará el resultado booleano de la llamada de ejecución del bloque de funciones.

Si todas las ramas contienen bloques de funciones, se evaluarán de arriba abajo y sus salidas se combinarán con operaciones lógicas OR. Si no hay ramas que contengan una llamada a bloques de funciones, se realizará la operación OR normal.

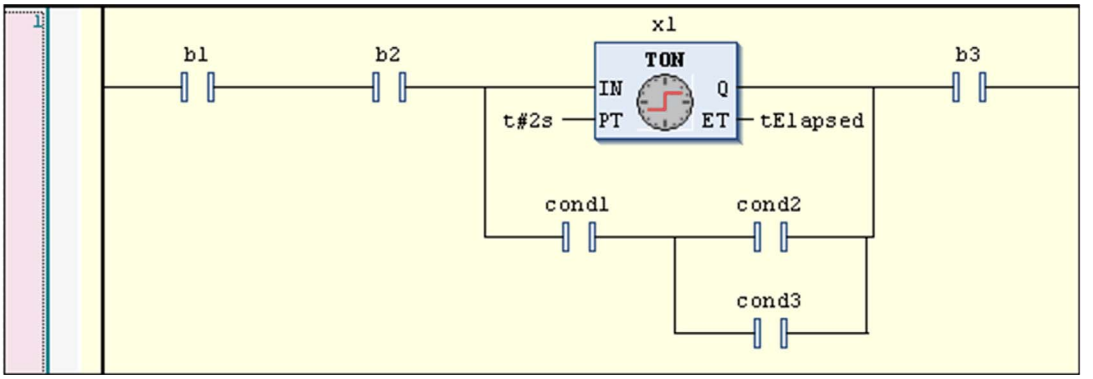
Para insertar una bifurcación paralela con la función SCE, seleccione el módulo del bloque de funciones y ejecute el comando **Insertar contacto paralelo (por encima)** o **Insertar contacto paralelo (por debajo)**. Esto sólo es posible si la primera entrada y la salida principal del bloque de funciones son de tipo BOOL.

A continuación, se muestra un ejemplo del modelo de lenguaje generado para la red en cuestión.

Ejemplo de SCE

La instancia del bloque de funciones `x1 (TON)` tiene una entrada booleana y una salida booleana. Esta ejecución puede omitirse si la condición de la bifurcación paralela se evalúa como verdadera. El valor de esta condición es el resultado de las operaciones `OR` y `AND` que conectan los contactos `cond1`, `cond2` y `cond3`.

Bifurcación paralela para SCE en una red de contactos

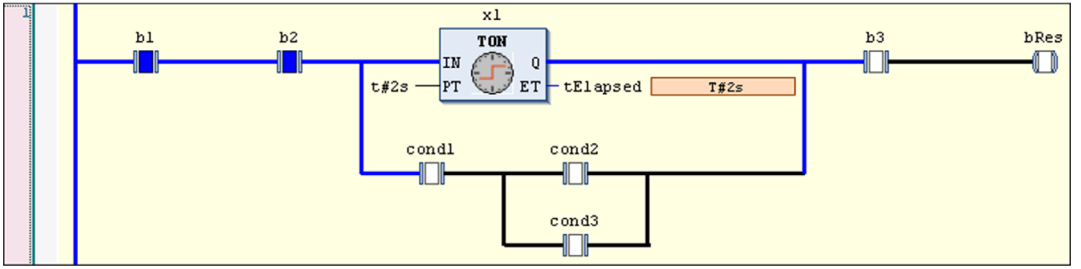


El procesamiento es como se muestra a continuación, donde `P_IN` y `P_OUT` representan el valor booleano en la entrada (punto de división) y la salida (punto de unión) de la bifurcación paralela, respectivamente.

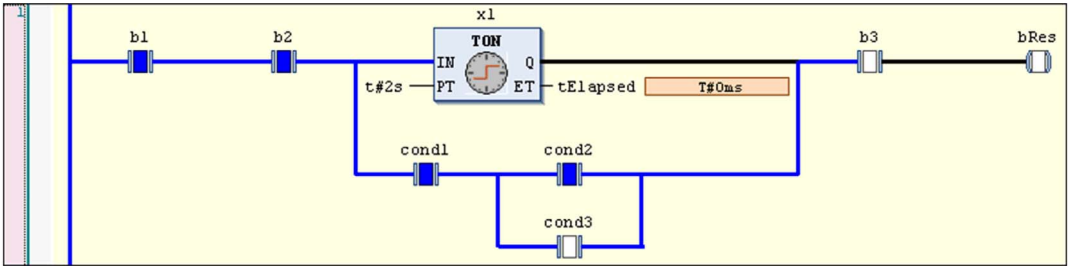
```
P_IN := b1 AND b2;
IF ((P_IN AND cond1) AND (cond2 OR cond3)) THEN
   P_OUT := P_IN;
ELSE
   x1(IN := P_IN, PT := {p 10}t#2s);
   tElapsed := x1.ET;
   P_OUT := x1.Q;
END_IF
bRes := P_OUT AND b3;
```

En las imágenes siguientes se muestra el flujo de datos (en azul) en caso de que el bloque de funciones se ejecute (la condición resultante de `cond1`, `cond2` y `cond3` es falsa) o se omita (condición verdadera).

Si la condición es falsa, el bloque de funciones se ejecuta:



Si la condición es verdadera, el bloque de funciones se omite:



Set/Reset en FBD/LD/IL

FBD y LD

Se puede establecer o restablecer una salida booleana en FBD *(véase página 282)* o, correspondientemente, a una bobina LD *(véase página 283)*. Para cambiar entre los estados establecidos, utilice el comando respectivo **Set/Reset** en el menú contextual cuando se selecciona la salida. La salida o bobina estará marcada por una `S` o una `R`.

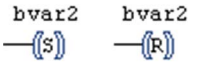
Set	Si el valor TRUE llega a una salida o bobina Set, esta salida o bobina se convertirá en TRUE y permanecerá así. Este valor no se puede sobrescribir en esta posición mientras la aplicación está en ejecución.
Restablecer	Si el valor TRUE llega a una salida o bobina Reset, esta salida o bobina se convertirá en FALSE y permanecerá así. Este valor no se puede sobrescribir en esta posición mientras la aplicación está en ejecución.

Definición de la salida en FBD

En el editor LD, puede insertar bobinas Set y Reset mediante el método de arrastrar y soltar. Para realizar esta acción, utilice **Herramientas**, categoría **Elemento de diagrama de contactos**, o los elementos `S` y `R` de la barra de herramientas.

Ejemplo:

Bobina Set, bobina Reset



Para obtener más información, consulte Bobina Set/Reset *(véase página 329)*.

IL

En una lista de instrucciones, utilice los operadores S y R *(véase página 286)* para establecer o restablecer un operando.

Bobina Set/Reset

Descripción general

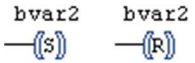
Las bobinas *(véase página 332)* también pueden definirse como bobinas Set o Reset.

Puede reconocer una bobina Set por la `S` que se incluye en el símbolo de la bobina: `(S)`. Una bobina Set no sobrescribirá el valor TRUE de la variable booleana adecuada. Es decir, una vez que la variable se ha establecido en TRUE, permanece en TRUE.

Puede reconocer una bobina Reset por la `R` que se incluye en el símbolo de la bobina: `(R)`. Una bobina Reset no sobrescribirá el valor FALSE de la variable booleana adecuada. Es decir, una vez que la variable se ha establecido en FALSE, permanece en FALSE.

En el editor LD, se pueden insertar bobinas Set y Reset directamente mediante el método de arrastrar y soltar desde **Herramientas**, categoría **Elemento de diagrama de contactos**.

Bobina Set, bobina Reset



Sección 9.3
Elementos LD

Contenido de esta sección

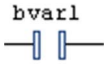
Esta sección contiene los siguientes apartados:

Contacto

Descripción general

Este es un elemento LD.

En la parte izquierda de LD *(véase página 283)*, cada red contiene 1 o varios contactos. Los contactos se representan con 2 líneas paralelas y verticales.

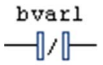


Los contactos transmiten la condición ON (TRUE) u OFF (FALSE) de izquierda a derecha. Se asigna una variable booleana a cada contacto. Si esta variable es TRUE, la condición se transmite de izquierda a derecha y, finalmente, a una bobina en la parte derecha de la red. De lo contrario, la conexión de la derecha recibe el valor FALSE.

Puede conectar varios contactos tanto en serie como en paralelo. Los contactos en paralelo representan una condición lógica "OR", de forma que sólo uno de ellos necesita tener el valor TRUE para que la bifurcación simultánea transmita el valor TRUE. En cambio, los contactos en serie representan una condición lógica "AND", por lo que todos los contactos deben ser TRUE para que el contacto final transmita TRUE.

Por lo tanto, la disposición de los contactos corresponde a un circuito eléctrico en paralelo o en serie.

Un contacto también puede ser negado. Esto se indica mediante la barra inclinada que aparece en el símbolo de contacto.



Un contacto negado transmite la condición entrante (TRUE o FALSE) sólo si la variable booleana asignada es FALSE.

Puede insertar un contacto en una red LD mediante uno de los comandos **Insertar contacto**, **Insertar contacto (a la derecha)**, **Insertar contacto paralelo (por encima)**, **Insertar contacto paralelo (por debajo)**, **Insertar contacto de flanco ascendente** o **Insertar contacto de flanco descendente** que forman parte del menú **LD**. Como alternativa, puede insertar el elemento mediante el método de arrastrar y soltar desde **Herramientas** *(véase página 313)* o desde otra posición del editor (arrastrar y soltar).

FBD e IL

Si está trabajando en la vista FBD *(véase página 282)* o IL *(véase página 284)*, el comando no estará disponible. Sin embargo, los contactos y bobinas insertados en una red LD se representarán con los elementos FBD o instrucciones IL correspondientes.

Bobina

Descripción general

Este es un elemento LD.

En el lado derecho de una red LD, puede haber cualquier número de bobinas que están representadas por paréntesis.

Sólo se pueden disponer en paralelo. Una bobina transmite el valor de las conexiones de izquierda a derecha y lo copia en una variable booleana adecuada. En la línea de entrada, el valor ON (TRUE) o el valor OFF (FALSE) pueden estar presentes.

Las bobinas también pueden ser negadas. Esto se indica mediante la barra inclinada que aparece en el símbolo de la bobina.

En este caso, el valor negado de la señal de entrada se copiará en la variable booleana adecuada.

Puede insertar una bobina en una red mediante uno de los comandos **Insertar bobina**, **Inserta bobina Set**, **Insertar bobina Reset** o **Insertar bobina negada** del menú **LD**. Como alternativa, puede insertar el elemento mediante el método de arrastrar y soltar desde **Herramientas** (**Elemento de diagrama de contactos**) o a través del método de arrastrar y soltar desde otra posición dentro del editor. Asimismo, consulte *Bobina Set/Reset (véase página 329)*.

FBD e IL

Si está trabajando en la vista FBD *(véase página 282)* o IL *(véase página 284)*, el comando no estará disponible. Sin embargo, los contactos y bobinas insertados en una red LD se representarán con los elementos FBD o instrucciones IL correspondientes.

Capítulo 10
Editor de diagrama de función continua (CFC)

Contenido de este capítulo

Este capítulo contiene los siguiente apartados:

Lenguaje de diagrama de función continua (CFC)

Descripción general

El diagrama de función continua (CFC) es una extensión del estándar IEC 61131-3, y es un lenguaje de programación gráfico basado en el lenguaje de diagrama de bloques de funciones (FBD) *(véase página 282)*. Sin embargo, en contraposición al lenguaje de FBD, no hay redes. CFC permite colocar libremente los elementos gráficos, lo que a su vez permite que haya lazos de realimentación.

Para crear objetos de programación CFC en SoMachine, consulte la descripción del editor CFC *(véase página 335)*.

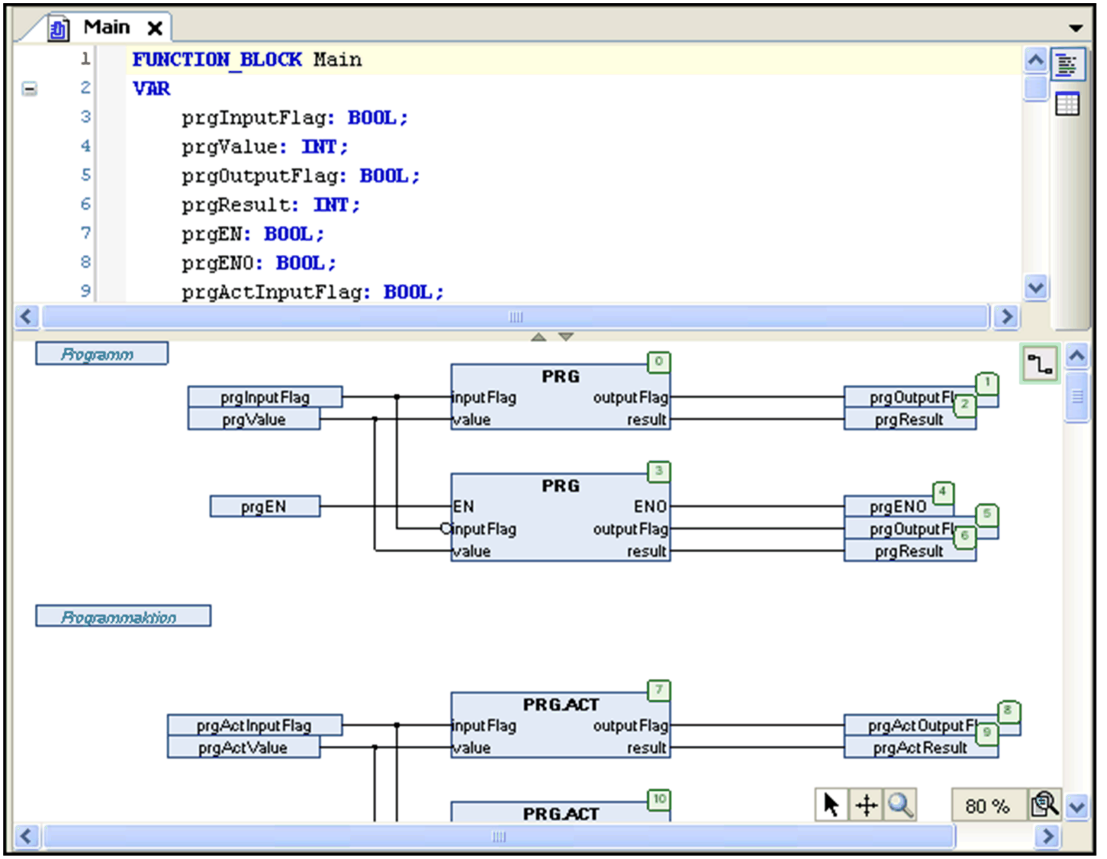
Editor CFC

Descripción general

El editor CFC es un editor gráfico disponible para la programación de objetos en el lenguaje de programación de diagrama de función continua (CFC) *(véase página 334)*, que es una extensión de los lenguajes de programación IEC 61131-3. Elija el lenguaje al añadir un nuevo objeto de unidad organizativa de programa (POU) al proyecto. Para proyectos grandes, plantéese utilizar la versión orientada a páginas *(véase página 351)*.

El editor estará disponible en la parte inferior de la ventana, que se abre al abrir el objeto POU de CFC. Esta ventana también incluye el editor de declaraciones *(véase página 414)* en la parte superior.

Editor CFC



El editor CFC, a diferencia del editor FBD / LD, permite el posicionamiento *(véase página 346)* libre de los elementos, lo que posibilita la inserción directa de rutas de realimentación. La secuencia de procesamiento se determina mediante una lista que contiene todos los elementos insertados actualmente y que pueden modificarse.

Los siguientes elementos están disponibles en un cuadro de herramientas *(véase página 339)* y pueden insertarse mediante el método de arrastrar y soltar:

- módulo (operadores, bloques de funciones y programas)
- entrada
- salida
- salto
- etiqueta
- retorno
- compositor
- selector
- marcas de conexión
- comentarios

Puede conectar los pines de entrada y salida de los elementos dibujando una línea con el ratón. La ruta de la línea de conexión se creará automáticamente y trazará la trayectoria más corta posible. Las líneas de conexión se ajustan automáticamente en cuanto se mueven los elementos. Para obtener más información, consulte la descripción sobre cómo insertar y organizar elementos *(véase página 346)*. Para diagramas complejos, puede utilizar marcas de conexión *(véase página 340)* en lugar de líneas. También puede plantearse la posibilidad de modificar el encaminamiento.

Puede pasar que los elementos se coloquen de tal modo que cubran conexiones ya encaminadas. Estas colisiones se indican mediante líneas de conexión rojas. Si existen colisiones en el diagrama, el botón de la esquina superior derecha de la vista de editor presentará un reborde rojo:

 . Para editar las colisiones paso a paso, haga clic en este botón y ejecute el comando **Mostrar siguiente colisión**. A continuación, se seleccionará la siguiente conexión implicada detectada.

Para diagramas complejos, puede utilizar marcas de conexión *(véase página 340)* en lugar de líneas. También puede utilizar la versión orientada a páginas del editor CFC.

Una función de zoom le permite cambiar la dimensión de la ventana del editor: utilice el botón  de la esquina inferior derecha de la ventana y elija de entre los factores de zoom enumerados. Como alternativa, puede seleccionar la entrada **...** para abrir un cuadro de diálogo donde puede escribir cualquier factor arbitrario.

Puede invocar los comandos para trabajar en el editor CFC desde el menú contextual o desde el menú **CFC** que está disponible en cuanto el editor CFC está activo.

Posiciones de cursor en CFC

Descripción general

Las posiciones de cursor en un programa CFC se indican mediante un fondo gris cuando se pasa el ratón por encima del elemento de programación.

Al hacer clic en una de esas áreas sombreadas, antes de soltar el botón del ratón, el color de fondo cambiará a rojo. En cuanto suelte el botón del ratón, ese punto pasará a ser la posición actual del cursor, con el elemento o el texto respectivo seleccionado y de color rojo.

Hay tres categorías de posiciones de cursor. Consulte las posiciones posibles indicadas por una zona sombreada de color gris que se muestran en las ilustraciones de los párrafos siguientes:

Cursor situado en un texto

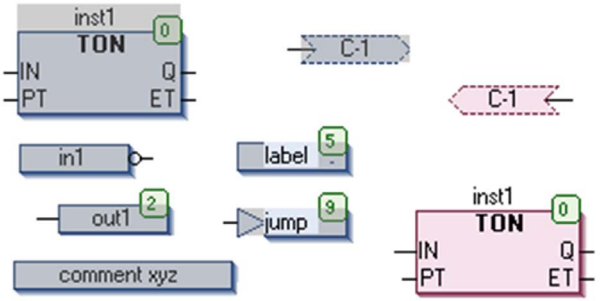
Si el cursor se ha situado en un texto y hace clic en el botón del ratón, se mostrará con un sombreado de color azul y el texto se podrá editar. El botón **...** está disponible para abrir el asistente Accesibilidad. Después de haber insertado un elemento, los caracteres `???` representan el nombre del elemento. Sustituya esos caracteres con un identificador válido. A continuación, se mostrará la información sobre herramientas cuando el cursor se sitúe sobre el nombre de una variable o un parámetro de módulo. La información sobre herramientas contiene el tipo de la variable o parámetro y, si existe, el comentario asociado en una segunda línea.

Posibles posiciones del cursor y ejemplo de texto seleccionado:

Cursor situado sobre el cuerpo de un elemento

Si el cursor se sitúa sobre el cuerpo de un elemento (módulo, entrada, salida, salto, etiqueta, retorno, comentario, marca de conexión), este se mostrará de color rojo y se puede mover al mover el ratón.

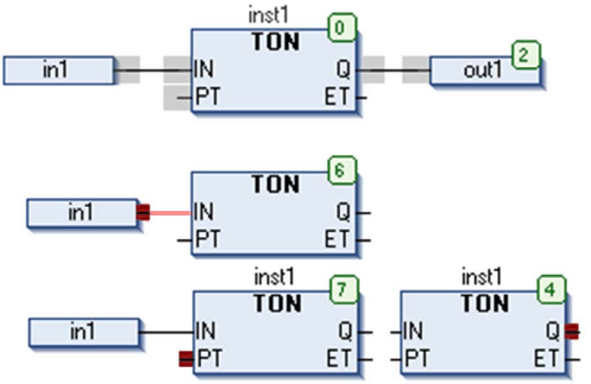
Posibles posiciones del cursor y un ejemplo de un cuerpo seleccionado:



Cursor situado sobre el cuerpo en la conexión de entrada o de salida de un elemento

Si el cursor se sitúa sobre una conexión de entrada o de salida de un elemento, un cuadrado rojo indicará esa posición (punto de conexión). Puede ser negado o Set/Reset.

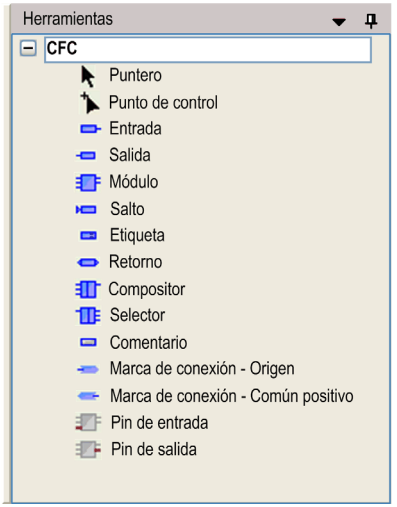
Posibles posiciones del cursor (sombreados de color gris) y ejemplos de posiciones seleccionadas de salida y de entrada (cuadrados rojos):



Elementos CFC/Herramientas

Descripción general

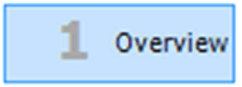
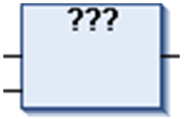
Las herramientas proporcionan los elementos gráficos disponibles para programar en la ventana del editor CFC *(véase página 335)*. Abra las herramientas ejecutando el comando **Herramientas** en el menú **Ver**.

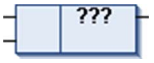


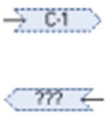
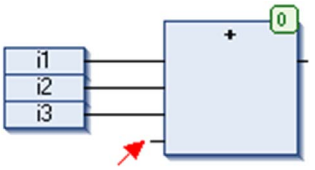
Seleccione el elemento que desee en las herramientas e insértelo *(véase página 346)* en la ventana del editor arrastrándolo y soltándolo.

Además de los elementos de programación, hay una entrada Pointer en la parte superior de la lista de las herramientas. En tanto que esta entrada esté seleccionada, el cursor tendrá la forma de una flecha y podrá seleccionar elementos en la ventana del editor para colocarlos y editarlos.

Elementos CFC

Nombre	Símbolo	Descripción
página		El número de la página se indica automáticamente en función de su posición. Puede introducir el nombre (`Overview` en este ejemplo) en el campo naranja en la parte superior de la página.
punto de control		El punto de control es necesario para fijar un enrutamiento de línea de conexión modificado manualmente, ya que es útil para impedir que el comando **Enrutar todas las conexiones** revierta la modificación. Mediante 2 puntos de control puede marcar un segmento específico de una línea cuyo enrutamiento desea modificar.
entrada		Puede seleccionar el texto ofrecido por `???` y reemplazarlo por una variable o constante. Accesibilidad se usa para seleccionar un identificador válido.
salida		
módulo		Puede usar un módulo para representar operadores, funciones, bloques de funciones y programas. Puede seleccionar el texto que ofrece `???` y reemplazarlo por un operador, función, bloque de funciones o nombre de programa. Accesibilidad se usa para seleccionar uno de los objetos disponibles. Si inserta un bloque de funciones, se visualizará otro `???` encima del cuadro. Reemplace los signos de interrogación por el nombre de la instancia del bloque de funciones. Si reemplaza un módulo existente por otro (modificando el nombre introducido) y el nuevo tiene un número mínimo o máximo de pines de entrada o de salida, los pines se adaptarán según corresponda. Si se eliminan los pines, primero se eliminará el que tenga el valor más bajo.
salto		Utilice el elemento de salto para indicar la posición en la que debe reanudarse la ejecución del programa. Esta posición se define con una etiqueta (consulte más abajo). Por tanto, reemplace el texto que ofrece `???` por el nombre de etiqueta.
etiqueta		Una etiqueta marca la posición hasta la que puede saltar el programa (consulte el salto de elemento). En modalidad online, se inserta una etiqueta de retorno para marcar el final de la POU.
retorno		En modalidad online, se inserta un elemento de retorno en la primera columna y después del último elemento en el editor. En la ejecución paso a paso, se salta automáticamente a él antes de que la ejecución abandone la POU.

Nombre	Símbolo	Descripción
compositor		Utilice un compositor para gestionar una entrada de un módulo que es un tipo de estructura. El compositor visualizará los componentes de estructura y, por tanto, el programador podrá acceder a ellos en el CFC. Para este fin, asigne un nombre al compositor como la estructura respectiva (reemplazando ??? por el nombre) y conéctelo al módulo en lugar de usar un elemento de entrada.
selector		Un selector, en contraposición con el compositor, se utiliza para gestionar una salida de un módulo que es un tipo de estructura. El selector visualizará los componentes de estructura y, por tanto, el programador podrá acceder a ellos en el CFC. Para este fin, asigne un nombre al selector como la estructura respectiva (reemplace ??? por el nombre) y conéctelo al módulo en lugar de usar un elemento de salida.
comentario		Use este elemento para añadir comentarios en un diagrama. Seleccione el texto del marcador de posición y reemplácelo por el texto que desee. Para obtener una nueva línea dentro del comentario, pulse CTRL + INTRO.

Nombre	Símbolo	Descripción
marca de conexión - entrada marca de conexión - salida		Puede usar las marcas de conexión en lugar de una línea de conexión *(véase página 347)* entre los elementos, ya que puede ser útil para borrar los diagramas complejos. Para una conexión válida, asigne un elemento de marca de conexión – entrada en la salida de un elemento y asigne una marca conexión - salida (consulte más abajo) en la entrada de otro elemento. Asigne el mismo nombre a las dos marcas (no hay distinción entre mayúsculas y minúsculas). Nomenclatura: El primer elemento de marca de conexión – entrada que se inserta en un CFC de forma predeterminada se denomina `C-1` y se puede modificar manualmente. En su marca de conexión - salida homóloga, reemplace `???` por la cadena con el mismo nombre que se usa en la marca de entrada. El editor verificará que los nombres de las marcas sean exclusivos. Si se cambia el nombre de una marca de entrada, también se cambiará automáticamente el nombre de la marca de salida conectada. Sin embargo, si se cambia el nombre de una marca de salida, la marca de entrada conservará su nombre antiguo, lo que le permitirá volver a configurar las conexiones. Del mismo modo, si elimina una marca de conexión no se eliminará su homóloga. Para usar una marca de conexión en el diagrama, arrástrela desde las herramientas hasta la ventana del editor y, a continuación, conecte su pin con el pin de salida o el pin de entrada del elemento respectivo. Como alternativa, puede convertir una línea de conexión normal existente mediante el comando **Marca de conexión**. Este comando también le permite revertir las marcas de conexión a sus líneas de conexión normal. Para ver las figuras en las que se muestran algunos ejemplos de marcas de conexión, consulte el capítulo *Marca de conexión*.
pin de entrada		En función del tipo de módulo, puede añadir una entrada adicional. Para este fin, seleccione el elemento de módulo en la red CFC y dibuje el elemento de pin de entrada en el módulo.
pin de salida	–	En función del tipo de módulo, puede añadir una salida adicional. Para este fin, seleccione el elemento de módulo en la red CFC y dibuje el elemento de pin de salida en el módulo.

Ejemplo de un compositor

Un programa CFC `cfc_prog` gestiona una instancia de bloque de funciones `fublo1`, que tiene una variable de entrada `struvar` de tipo de estructura. Utilice el elemento de composición para acceder a los componentes de estructura.

Definición de estructura `stru1`:

```
TYPE stru1 :
STRUCT
  ivar:INT;
  strvar:STRING:='hallo';
END_STRUCT
END_TYPE
```

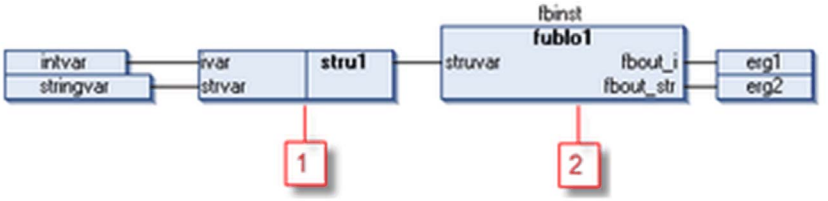
Declaración e implementación de bloque de funciones `fublo1`:

```
FUNCTION_BLOCK fublo1
VAR_INPUT
  struvar:STRU1;
END_VAR
VAR_OUTPUT
  fbout_i:INT;
  fbout_str:STRING;
END_VAR
VAR
  fbvar:STRING:='world';
END_VAR
fbout_i:=struvar.ivar+2;
fbout_str:=CONCAT (struvar.strvar,fbvar);
```

Declaración e implementación de programa `cfc_prog`:

```
PROGRAM cfc_prog
VAR
  intvar: INT;
  stringvar: STRING;
  fbinst: fublo1;
  erg1: INT;
  erg2: STRING;
END_VAR
```

Elemento de composición



1 compositor
2 bloque de funciones con variable de entrada `struvar` de estructura de tipo `stru1`

Ejemplo de un selector

Un programa CFC `cfc_prog` gestiona una instancia de bloque de funciones `fublo2`, que tiene una variable de salida `fbout` de estructura de tipo `stru1`. Utilice el elemento selector para acceder a los componentes de estructura.

Definición de estructura `stru1`:

```
TYPE stru1 :
STRUCT
  ivar:INT;
  strvar:STRING:='hallo';
END_STRUCT
END_TYPE
```

Declaración e implementación del bloque de funciones `fublo1`:

```
FUNCTION_BLOCK fublo2
VAR_INPUT CONSTANT
  fbin1:INT;
  fbin2:DWORD:=24354333;
  fbin3:STRING:='hallo';
END_VAR
VAR_INPUT
  fbin : INT;
END_VAR
VAR_OUTPUT
  fbout : stru1;
  fbout2:DWORD;
END_VAR
VAR
  fbvar:INT;
  fbvar2:STRING;
END_VAR
```

Declaración e implementación del programa `cfc_prog`:

```
VAR
  intvar: INT;
  stringvar: STRING;
  fbinst: fublo1;
  erg1: INT;
  erg2: STRING;
  fbinst2: fublo2;
END_VAR
```

En la ilustración se muestra un elemento selector donde las conexiones no utilizadas se han eliminado con la ejecución del comando **Eliminar las conexiones sin enlace**.

1 bloque de funciones con variable de salida `fbout` de estructura de tipo `stru1`
2 selector

Trabajo en el editor CFC

Descripción general

Los elementos disponibles para la programación en el editor CFC se proporcionan en Herramientas *(véase página 339)*, que de forma predeterminada está disponible en una ventana en cuanto se abre el editor CFC.

En **Herramientas** →**Opciones** →**Editor CFC** se define la configuración general para trabajar en el editor.

Inserción

Para insertar un elemento, selecciónelo en **Herramientas** *(véase página 339)* haciendo clic con el ratón, mantenga pulsado el botón del ratón y arrastre el elemento a la posición que desee en la ventana del editor. Al arrastrar, el cursor se mostrará como una flecha más un rectángulo y un signo más. Al soltar el botón del ratón, se insertará el elemento.

Selección

Para seleccionar un elemento insertado para realizar acciones adicionales como, por ejemplo, editar o reorganizar, haga clic en el cuerpo del elemento para seleccionar el elemento. Se mostrará de forma predeterminada sombreado en rojo. Si pulsa además la tecla MAYÚS, puede hacer clic en más elementos para seleccionarlos. También puede pulsar el botón izquierdo del ratón y dibujar un rectángulo con puntos alrededor de todos los elementos que desee seleccionar. En cuanto suelte el botón, se indicará la selección. Con el comando **Seleccionar todo**, se seleccionan todos los elementos a la vez.

Mediante las teclas de flecha, puede cambiar la marca de selección a la siguiente posición posible del cursor. La secuencia depende del orden de ejecución de los elementos, que se indica mediante números de elemento *(véase página 348)*.

Cuando se seleccione un pin de entrada y se pulse CTRL + FLECHA IZQUIERDA, se seleccionará la salida correspondiente. Cuando se seleccione un pin de entrada y se pulse CTRL + FLECHA IZQUIERDA, se seleccionarán las salidas correspondientes.

Sustitución de módulos

Para sustituir un elemento de módulo existente, sustituya el identificador insertado actualmente por el del elemento nuevo que desee. El número de pines de entrada y salida se adaptará si es necesario debido a la definición de las POU y, de este modo, se pueden eliminar algunas asignaciones existentes.

Desplazamiento

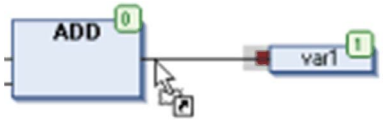
Para mover un elemento, selecciónelo haciendo clic en su cuerpo (vea las posiciones de cursor *(véase página 337)* posibles) y arrástrelo, manteniendo el botón del ratón pulsado, a la posición deseada. Luego, suelte el botón del ratón para colocar el elemento. También puede utilizar los comandos **Cortar** y **Pegar** con el mismo fin.

Conexión

Puede conectar los pines de entrada o salida de dos elementos mediante una línea de conexión o mediante marcas de conexión.

Línea de conexión: Puede seleccionar un punto de conexión válido que sea un pin de entrada o salida de un elemento (consulte *Posiciones de cursor en CFC (véase página 337)*) y luego trazar una línea hasta otro punto de conexión con el ratón. O bien puede seleccionar dos puntos de conexión y ejecutar el comando **Select Connected Pins**. Un punto de conexión posible seleccionado se indica mediante un cuadrado con relleno rojo. Cuando se traza una línea desde tal punto hasta el elemento de destino, se puede identificar el punto de conexión de destino posible. Cuando se coloca el cursor sobre un punto de conexión válido, se añade un símbolo de flecha al cursor al moverlo sobre dicho punto, indicando la posible conexión.

La figura siguiente ofrece un ejemplo: Después de hacer clic con el ratón en un pin de entrada del elemento `var1`, se muestra el rectángulo rojo que muestra que se trata de un punto de conexión seleccionado. Manteniendo pulsado el botón del ratón, mueva el cursor al pin de salida del módulo ADD hasta que el símbolo del cursor aparezca tal como se muestra en la figura. Ahora suelte el botón del ratón para establecer la línea de conexión.



Se creará la conexión más corta posible teniendo en cuenta el resto de los elementos y las conexiones. Si la ruta de las líneas de conexión se solapa con otras líneas de conexión, su color será gris claro

Marcas de conexión: También puede utilizar marcas de conexión en lugar de líneas de conexión para simplificar los diagramas complejos. Consulte la descripción de las marcas de conexión *(véase página 340)*.

Copia

Para copiar un elemento, selecciónelo y utilice los comandos **Copiar** y **Pegar**.

Edición

Después de insertar un elemento, de forma predeterminada la parte de texto se representa mediante `???`. Para sustituir estos signos por el texto deseado, (nombre de POU, nombre de etiqueta, nombre de instancia, comentario, etc.), haga clic en el texto para obtener un campo de edición. También estará disponible el botón **...** para abrir **Accesibilidad**.

Eliminación

Puede eliminar un elemento seleccionado o una línea de conexión ejecutando el comando **Eliminar**, que está disponible en el menú contextual, o pulsando la tecla SUPR.

Apertura de un bloque de funciones

Si se añade un bloque de funciones al editor, puede abrir este bloque con un doble clic. Como alternativa, utilice el comando **Examinar** →**Ir a la definición** en el menú contextual.

Orden de ejecución, Números de elemento

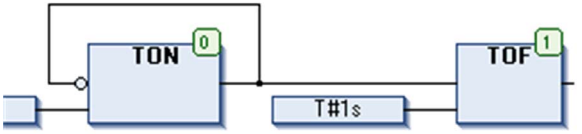
La secuencia en la que se ejecutan los elementos en una red CFC en modalidad online se indica mediante números en la esquina superior derecha de los elementos módulo, salida, salto, retorno y etiqueta. El procesamiento se inicia en el elemento con el número más bajo, que es 0.

Puede modificar el orden de ejecución mediante comandos que están disponibles en el submenú **Orden de ejecución** del menú **CFC**.

Al añadir un elemento, el número se asignará automáticamente según la secuencia topológica (de izquierda a derecha y de arriba abajo). El elemento nuevo recibe el número de su sucesor topológico si la secuencia ya se ha modificado y todos los números más altos se incrementan en 1.

Tenga en cuenta que el número de un elemento permanece constante cuando se mueve.

Tenga en cuenta que la secuencia influye sobre el resultado y se debe cambiar en determinados casos.



Cambio del tamaño de la hoja de trabajo

Para obtener más espacio en torno a un diagrama CFC existente en la ventana del editor, puede cambiar el tamaño del área de trabajo (hoja de trabajo). Para hacerlo, seleccione y arrastre todos los elementos con el ratón o utilice los comandos cortar y pegar (consulte *Desplazamiento (véase página 346)*).

Como alternativa, puede utilizar un cuadro de diálogo de configuración de dimensiones especial. Esto puede ahorrarle tiempo si trabaja con diagramas de gran tamaño. Consulte la descripción del cuadro de diálogo *(véase SoMachine, Comandos de menú, Ayuda en línea)* **Editar la hoja de trabajo**. En el caso de CFC orientado a la página, puede utilizar el comando *(véase SoMachine, Comandos de menú, Ayuda en línea)* **Edit Page Size**.

Editor CFC en modalidad online

Descripción general

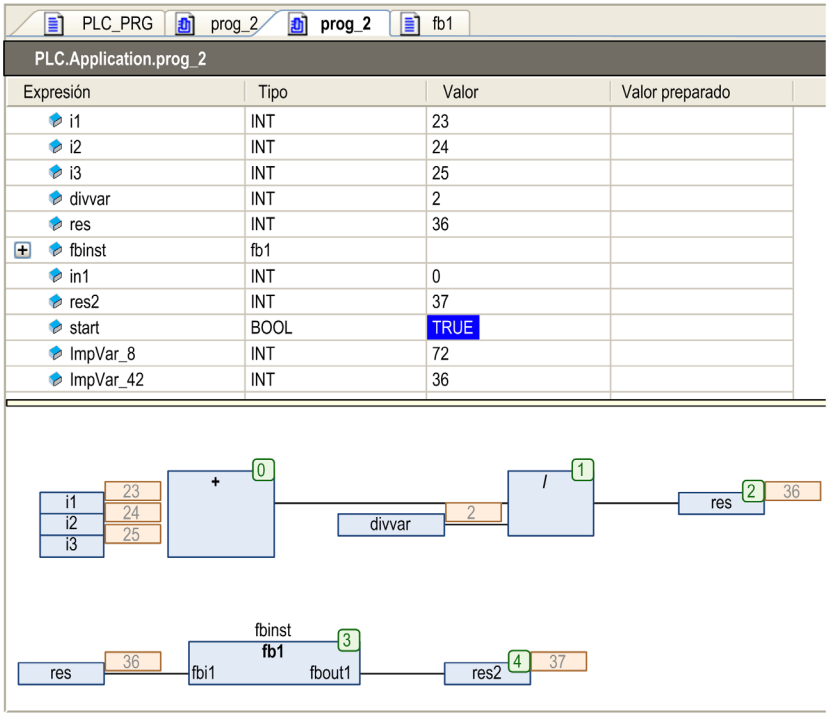
En la modalidad online, el editor CFC proporciona vistas para la supervisión. Las vistas para la escritura y el forzado de las variables y expresiones en el controlador se describen en capítulos separados. La funcionalidad de depuración (puntos de interrupción, ejecución paso a paso, etc.) está disponible como se describe a continuación.

- Para obtener información acerca de la apertura de objetos en modalidad online, consulte la descripción de la interfaz de usuario en modalidad online *(véase página 49)*.
- La ventana del editor de un objeto CFC también incluye el editor de declaraciones en la parte superior. Consulte la descripción del editor de declaraciones en modalidad online *(véase página 419)*.

Supervisión

Los valores reales se muestran en las pequeñas ventanas de supervisión que hay detrás de cada variable (supervisión en línea).

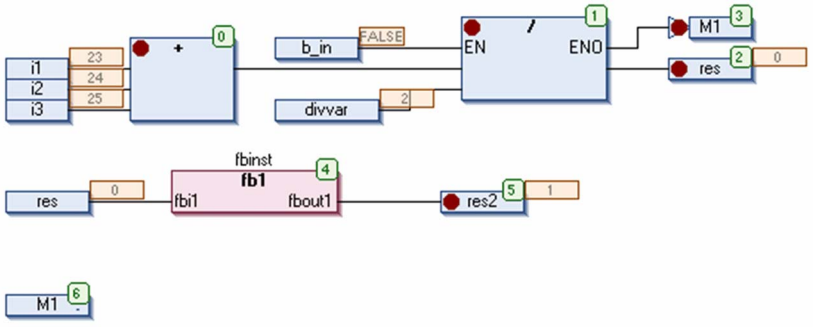
Vista online de un objeto de programa PLC_PRG:



Posiciones de puntos de interrupción en el editor CFC

Las posiciones posibles de los puntos de interrupción son básicamente las posiciones de una POU en las que los valores de las variables pueden cambiar, en las que el flujo del programa se bifurca o en las que se llama a otra POU. Vea las posiciones posibles en la imagen siguiente.

Posiciones de puntos de interrupción en el editor CFC:



NOTA: Se establecerá automáticamente un punto de interrupción en todos los métodos que puedan llamarse. Si se llama un método gestionado por interfaces, los puntos de interrupción se establecerán en todos los métodos de bloques de funciones que implementen esa interfaz y en todos los bloques de funciones derivados que suscriban el método. Si se llama un método mediante un puntero en un bloque de funciones, se establecerán puntos de interrupción en el método del bloque de funciones y en todos los bloques de funciones derivados que suscriban el método.

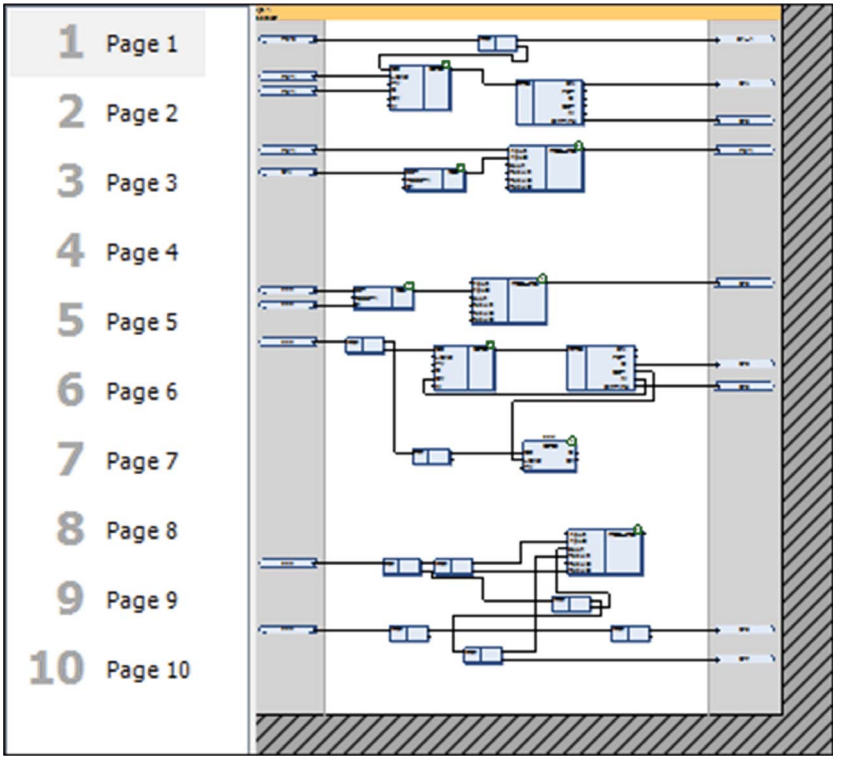
Editor CFC orientado a la página

Descripción general

Además del editor CFC estándar, SoMachine proporciona la paginación del editor CFC. Aparte de las herramientas *(véase página 339)* y comandos del editor CFC estándar, este editor permite organizar los elementos en un número cualquiera de páginas.

NOTA: No puede convertir las POU creadas en lenguaje CFC orientado a la página a CFC normal y viceversa. Puede copiar elementos entre estos 2 editores con los comandos para copiar y pegar (a través del portapapeles) o con la función de arrastrar y soltar.

Paginación de CFC

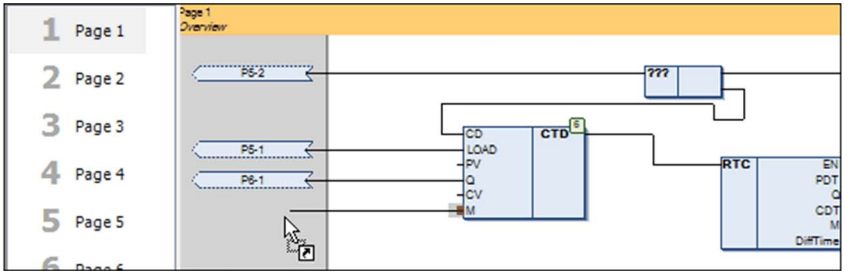


Para cambiar el tamaño de la página, ejecute el comando **Edit Page Size**.

Conexiones entre 2 páginas

Las conexiones entre 2 páginas se realizan con los elementos marca de conexión - entrada y marca de conexión - salida (consulte la descripción de las marcas de conexión). Puede colocar la marca de conexión - entrada mediante el método de arrastrar y soltar en el margen derecho y la marca de conexión - salida en el margen izquierdo. Si dibuja una línea de conexión desde una entrada o una salida de un elemento hasta el margen, la marca de conexión se coloca de forma automática.

Inserción de marcas de conexión



Orden de ejecución

El orden de ejecución de las páginas va de arriba abajo. Dentro de una página, el orden sigue las reglas del editor CFC estándar (consulte la información adicional sobre el orden de ejecución *(véase página 348)*). Puede cambiar el orden de ejecución de los elementos solamente dentro de la página asociada. No puede cambiar el orden de ejecución de los elementos en páginas diferentes.

Capítulo 11
Editor de diagrama funcional secuencial (SFC)

Contenido de este capítulo

Este capítulo contiene los siguiente apartados:

Editor SFC

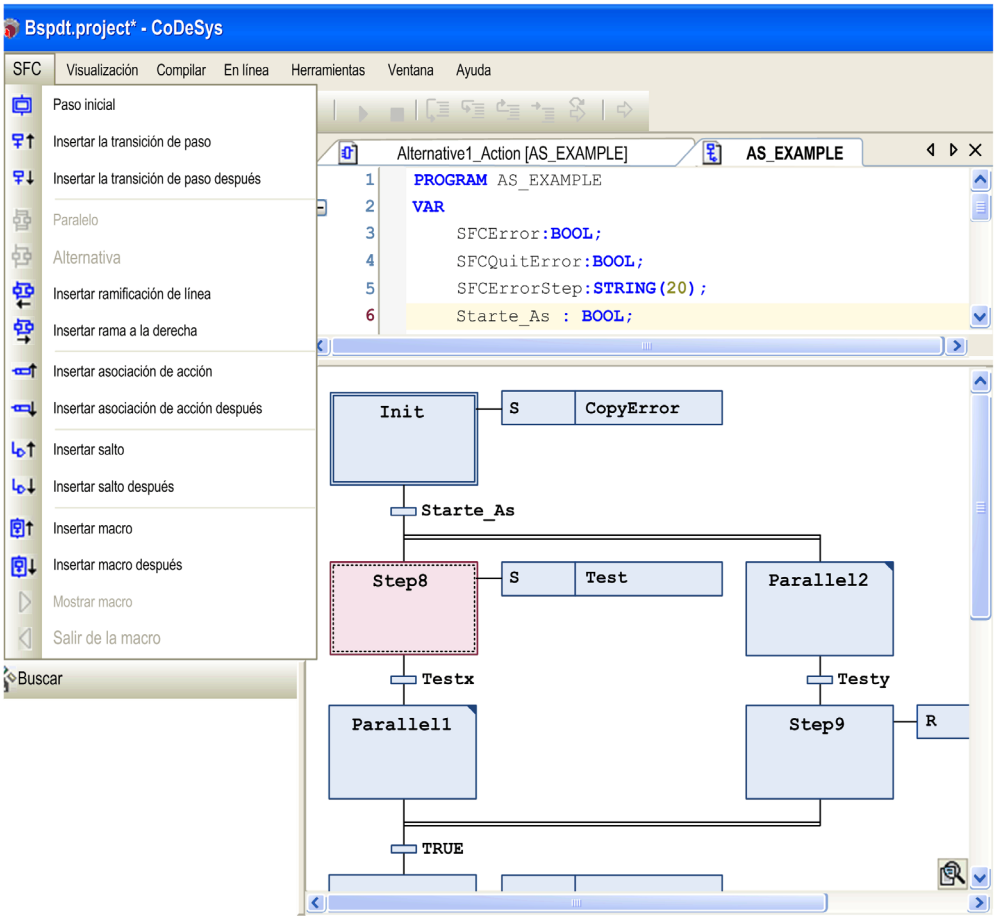
Descripción general

El editor SFC está disponible para la programación de objetos en el lenguaje de programación IEC 61131-3 SFC - Diagrama funcional secuencial *(véase página 356)*. Seleccione el lenguaje cuando añada un nuevo objeto POU al proyecto.

El editor SFC es un editor gráfico. Realice la configuración general relativa al comportamiento y la visualización en el cuadro de diálogo **Opciones** →**Editor SFC**.

El editor SFC está disponible en la parte inferior de la ventana que se abre cuando se edita un objeto POU SFC. Esta ventana también incluye el **editor de declaraciones** *(véase página 414)* en la parte superior.

Editor SFC



Trabajo con el editor SFC

Los elementos *(véase página 363)* utilizados en un diagrama SFC están disponibles en el menú **SFC**. El menú estará disponible tan pronto como el editor SFC esté activo. Dispóngalos en una secuencia o en secuencias paralelas de pasos que están conectados por transiciones. Para obtener más información, consulte *Trabajo en el editor SFC (véase página 359)*.

Puede editar las propiedades de los pasos en una ventana separada de propiedades *(véase página 361)*. Entre otras cosas, se puede definir el tiempo mínimo y máximo de la actividad para cada paso.

Puede acceder a variables implícitas *(véase página 377)* para controlar el procesamiento de un SFC (por ejemplo, estado de los pasos, análisis de timeout, restablecimiento).

SFC - Lenguaje de diagrama funcional secuencial

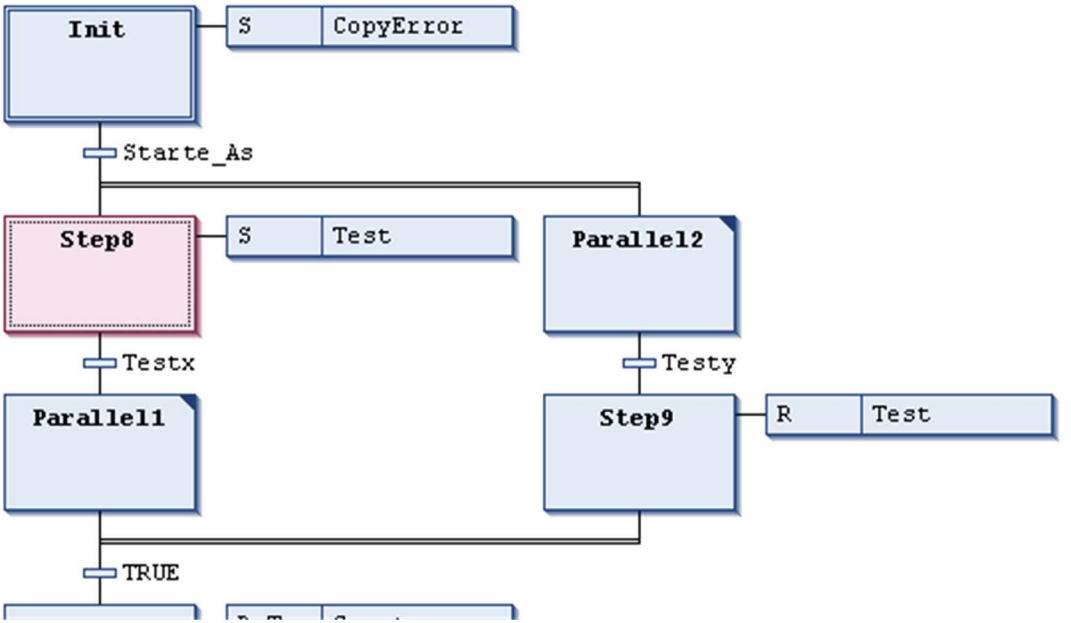
Descripción general

El diagrama funcional secuencial (SFC) es un lenguaje orientado gráficamente que describe el orden cronológico de acciones concretas en un programa. Estas acciones están disponibles como objetos de programación independientes, y están escritas en cualquier lenguaje de programación disponible. En SFC, esas acciones se asignan a elementos de paso y los elementos de transición controlan la secuencia de procesamiento. Para obtener una descripción detallada del modo en el que se procesarán los pasos en modalidad online, consulte *Secuencia de procesamiento en SFC (véase página 383)*.

Para obtener información sobre cómo utilizar el editor SFC en SoMachine, consulte la descripción del *editor SFC (véase página 354)*.

Ejemplo

Ejemplo de una secuencia de pasos en un módulo SFC:



Posiciones de cursor en SFC

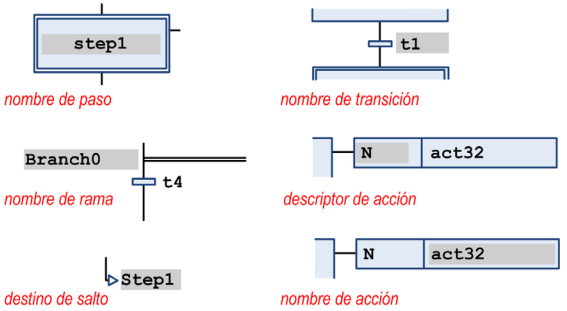
Descripción general

Las posibles posiciones de cursor en un diagrama SFC en el editor SFC *(véase página 354)* se indican mediante una sombra de color gris cuando se pasa el cursor sobre los elementos.

Posiciones de cursor en textos

Hay 2 categorías de posiciones de cursor: textos y cuerpos de elementos. Consulte las posiciones posibles indicadas por una zona sombreada de color gris que se muestran en las ilustraciones siguientes:

Posibles posiciones de cursor en textos:

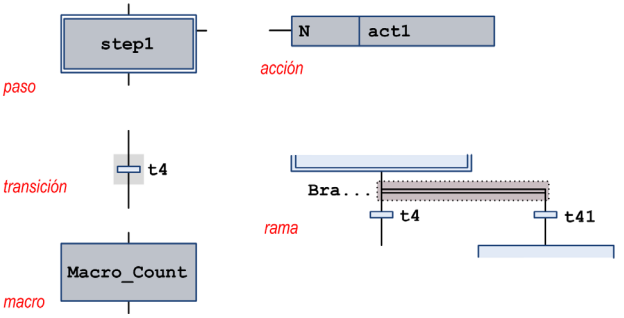


Si se hace clic en una posición de cursor en un texto, la cadena podrá editarse.

Seleccione el nombre de la acción para editarlo:

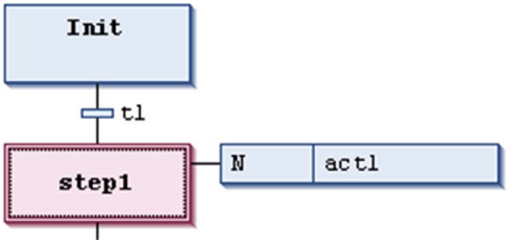
Posiciones de cursor en cuerpos de elementos

Posibles posiciones de cursor en cuerpos de elementos:



Al hacer clic en una zona sombreada, el elemento se selecciona. Se muestra un marco de puntos y aparece un sombreado de color rojo (para realizar una selección múltiple, consulte *Trabajo en el editor SFC* *(véase página 359)*).

Elemento de paso seleccionado



Trabajo en el editor SFC

Descripción general

De forma predeterminada, una nueva POU SFC contiene un paso inicial y una transición posterior. En este capítulo se proporciona información sobre cómo añadir más elementos y cómo organizar y editar los elementos.

Posibles posiciones del cursor

Para obtener más información, consulte el capítulo *Posiciones de cursor en SFC (véase página 357)*.

Desplazamiento

Utilice las teclas de flecha para ir al elemento anterior o siguiente del diagrama.

Inserción de elementos

Para insertar los elementos SFC *(véase página 363)* específicos, ejecute los comandos pertinentes del menú **SFC**. Para obtener más información, consulte la descripción de los comandos del editor SFC *(véase SoMachine, Comandos de menú, Ayuda en línea)*. Haga doble clic en un paso, transición o elemento de acción ya insertado que todavía no haga referencia a un objeto de programación a fin de abrir un cuadro de diálogo para asignar uno.

Selección de elementos

Seleccione un elemento o campo de texto haciendo clic en una posición posible del cursor. También puede asignar la selección a un elemento adyacente mediante las teclas de flecha. El elemento pasará a mostrarse en color rojo. Por ejemplo, consulte el capítulo *Posiciones de cursor en SFC (véase página 357)*.

NOTA: A diferencia de las versiones anteriores de SoMachine, puede seleccionar y mover (cortar, copiar, pegar) o eliminar pasos y transiciones por separado.

Para realizar una selección múltiple, dispone de las siguientes opciones:

- Mantener pulsada la tecla MAYÚS y hacer clic en los elementos específicos que desea seleccionar.
- Pulsar el botón izquierdo del ratón y dibujar un rectángulo (línea discontinua) alrededor de los elementos por seleccionar.
- Ejecutar el comando **Seleccionar todo**, de forma predeterminada desde el menú **Editar**.

Edición de textos

Haga clic en una posición de cursor de texto para abrir el campo de edición, donde podrá editar el texto. Si se ha seleccionado un área de texto mediante las teclas de flecha, abra el campo de edición explícitamente con la barra espaciadora.

Edición de acciones asociadas

Haga doble clic en un paso (entrada, activo o salida) o una asociación de acción de transición para abrir la acción asociada en el editor correspondiente. Por ejemplo, puede hacer doble clic en el elemento de transición o en el triángulo que indica una acción de salida en un elemento de paso.

Cortado, copiado y pegado de elementos

Seleccione los elementos y ejecute el comando **Cortar**, **Copiar** o **Pegar** (en el menú **Editar**) o utilice las teclas correspondientes.

NOTA:

- Al pegar uno o varios elementos cortados o copiados, el contenido del portapapeles se inserta antes de la posición seleccionada actualmente. Si no se seleccionada nada, los elementos se añaden al final del diagrama cargado actualmente.
- Si pega una bifurcación cuando el elemento seleccionado también es una bifurcación, los elementos de bifurcación pegados se insertarán a la izquierda de los elementos existentes.
- Si pega una acción (lista de acciones) en un paso seleccionado actualmente, las acciones se añadirán al principio de la lista de acciones del paso o bien se creará una lista de acciones para el paso.
- Elementos incompatibles al cortar/copiar:
  Si selecciona una acción (lista de acciones) asociada y, además, un elemento que no es el paso al que pertenece la acción (lista de acciones), se muestra este cuadro de mensaje: **La selección actual contiene elementos incompatibles. No se transmitirán datos al portapapeles.** La selección no se almacenará y no la podrá pegar ni copiar.
- Elementos incompatibles al pegar:
  Si intenta pegar una acción (lista de acciones) y el elemento seleccionado no es un paso u otra asociación, se muestra este cuadro de mensaje: **Imposible insertar el contenido del portapapeles en la selección actual.** Si intenta pegar un elemento como un paso, bifurcación o transición y hay seleccionada una acción (lista de acciones) asociada, se muestra el mismo cuadro de mensaje.

Eliminación de elementos

Seleccione los elementos y ejecute el comando **Eliminar** o pulse la tecla SUPR.

Tenga en cuenta lo siguiente:

- Al eliminar un paso también se elimina la lista de acciones asociada.
- Al eliminar el paso inicial, el siguiente paso se establece automáticamente como paso inicial. La opción **Paso inicial** se activará en las propiedades de este paso.
- La eliminación de la línea horizontal anterior a un área bifurcada eliminará todas las bifurcaciones.
- La eliminación de todos los elementos específicos de una bifurcación eliminará la bifurcación.

Propiedades del elemento SFC

Descripción general

Puede visualizar y editar las propiedades de un elemento SFC en el cuadro de diálogo **Propiedades del elemento**. Abra este cuadro de diálogo mediante el comando **Propiedades del elemento**, que forma parte del menú **Visualizar**.

Las propiedades que se visualicen dependerán del elemento seleccionado actualmente. Las propiedades están agrupadas. Puede abrir y cerrar determinadas secciones de grupos utilizando los símbolos Más o Menos.

Puede configurar si deben visualizarse determinados tipos de propiedades junto a un elemento en la gráfica SFC en la ficha **Visualizar** de las opciones del editor SFC.

Configuración general

Propiedad	Descripción
Nombre	Nombre del elemento; de forma predeterminada <elemento><número de ejecución>. Ejemplos: nombre de paso Step0, Step1, nombre de rama branch0, etc.
Comentario	Comentario sobre el elemento en una cadena de texto. Ejemplo: Restablecer contador. Pulse CTRL + INTRO para insertar saltos de línea.
Símbolo	Por cada elemento SFC se crea implícitamente un indicador que se denomina igual que el elemento. Aquí puede especificar si esta variable de indicador debe exportarse a la configuración de símbolos y cómo debe ser accesible el símbolo en el controlador. Haga doble clic en el campo del valor o seleccione el campo del valor y pulse la barra espaciadora para abrir la lista de selección en la que puede elegir una de las siguientes opciones de acceso: Ninguno: El símbolo se exportará a la configuración de símbolos, pero no estará accesible en el controlador. Lectura: El símbolo se exportará a la configuración de símbolos y podrá leerse en el controlador. Escritura: El símbolo se exportará a la configuración de símbolos y podrá escribirse en el controlador. Lectura/escritura: combinación de lectura y escritura. De forma predeterminada, este campo se deja vacío. Es decir, no se exporta ningún símbolo a la configuración de símbolos.

Propiedades específicas

Propiedad específica		Descripción
Paso inicial		La opción se activa en las propiedades del paso inicial *(véase página 363)* actual. De forma predeterminada, se activa para el primer paso de SFC y se desactiva para los demás pasos. Si activa esta opción para otro paso, debe desactivarla en el paso inicial anterior. En caso contrario, se generará un error de compilador.
Tiempos:		Define los tiempos de proceso mínimo y máximo del paso. **NOTA:** Los timeouts en los pasos se indican mediante el indicador SFCError de la variable implícita *(véase página 377)*.
	Mínimo activo	Duración mínima de tiempo que debería requerir el procesamiento de este paso. Valores permitidos: tiempo según la sintaxis IEC (por ejemplo, `t#8s`) o variable TIME; valor predeterminado: `t#0s`.
	Máximo activo	Duración máxima de tiempo que debería requerir el procesamiento de este paso. Valores permitidos: tiempo según la sintaxis IEC (por ejemplo, `t#8s`) o variable TIME; valor predeterminado: `t#0s`.
Acciones:		Define las Acciones *(véase página 368)* que deben realizarse cuando el paso está activo. Consulte la descripción de la sección Secuencia de procesamiento en SFC *(véase página 383)* para obtener más información.
	Paso activado	Esta acción se ejecutará después de que el paso se active
	Paso activo	Esta acción se ejecutará cuando el paso esté activo y las posibles acciones de la entrada ya se hayan procesado.
	Paso desactivado	Esta acción se ejecutará en el ciclo posterior a la desactivación de un paso (acción de salida).

NOTA: Utilice las variables implícitas correspondientes para determinar el estado de las acciones y los timeouts mediante indicadores SFC *(véase página 377)*.

Elementos SFC / Herramientas

Descripción general

Puede insertar los elementos gráficos que se pueden utilizar para la programación en la ventana del editor SFC ejecutando los comandos del menú **SFC**.

Para obtener información sobre cómo trabajar en el editor, consulte la descripción en el capítulo *Trabajo en el editor SFC* *(véase página 359)*

Los elementos siguientes están disponibles y se describen en este capítulo:
- paso *(véase página 363)*
- transición *(véase página 363)*
- acción *(véase página 368)*
- bifurcación (alternativa) *(véase página 371)*
- bifurcación (simultánea) *(véase página 372)*
- salto *(véase página 374)*
- macro *(véase página 374)*

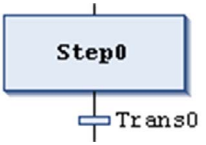
Paso / Transición

Para insertar un solo paso o una sola transición, ejecute el comando **Paso** o **Transición** desde **Herramientas**. También se pueden insertar pasos y transiciones en combinación, mediante el comando **Insertar la transición de paso** ( ) o **Insertar transición de paso después** ( ) desde la barra de herramientas.

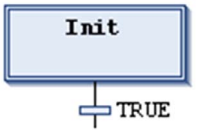
Un paso se representa mediante un módulo que contiene principalmente un nombre de paso generado automáticamente. Se conecta a la transición anterior y posterior mediante una línea. El marco del módulo del primer paso en un SFC, el paso inicial, tiene una línea doble.

La transición se representa mediante un pequeño rectángulo. Después de insertarla, tiene un nombre predeterminado, `Trans<n>`, donde `n` es un número consecutivo.

Ejemplo de paso y transición posterior:



Ejemplo de paso inicial y transición posterior:



Puede editar los nombres de pasos y transiciones en línea.

Los nombres de paso deben ser exclusivos en el ámbito de la POU padre. Téngalo en cuenta especialmente al utilizar acciones programadas en SFC. De lo contrario, se detectará un error durante el proceso de compilación.

Puede transformar cada paso en un paso inicial ejecutando el comando **Paso inicial** o activando la propiedad de paso respectiva. Un paso inicial se ejecutará primero cuando se llama a la POU IL.

Cada paso se define mediante las propiedades *(véase página 361)* del paso.

Después de insertar un paso, asocie las acciones que se deben realizar cuando el paso esté activo (procesado); consulte a continuación la información adicional sobre las acciones *(véase página 368)*.

Recomendaciones sobre las transiciones

Una transición debe proporcionar la condición que hará que el paso posterior se active en cuanto el valor de la condición sea TRUE. Por lo tanto, una condición de transición debe tener el valor TRUE o FALSE.

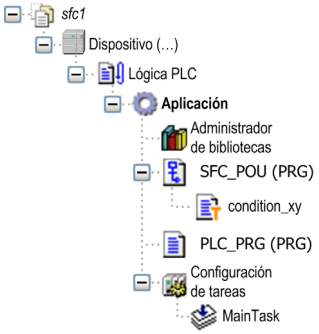
Una condición de transición se puede definir de los 2 modos siguientes:

Tipo de definición	Tipo de condición	Descripción
directa	en línea	Sustituya el nombre de transición predeterminado por uno de los elementos siguientes: ● variable booleana ● dirección booleana ● constante booleana ● instrucción que tiene un resultado booleano (ejemplo: (i<100) AND b). Aquí no se pueden especificar programas, bloques de funciones ni asignaciones.
utilizando un objeto de transición o de propiedad independiente	multiuso	Sustituya el nombre de transición predeterminado por el nombre de un objeto de transición () o de propiedad () disponible en el proyecto. (Esto permite el uso múltiple de transacciones; véase por ejemplo `condition_xy` en las figuras siguientes). El objeto como una transición en línea puede contener los elementos siguientes: ● variable booleana ● dirección ● constante ● instrucción ● varias instrucciones con código arbitrario

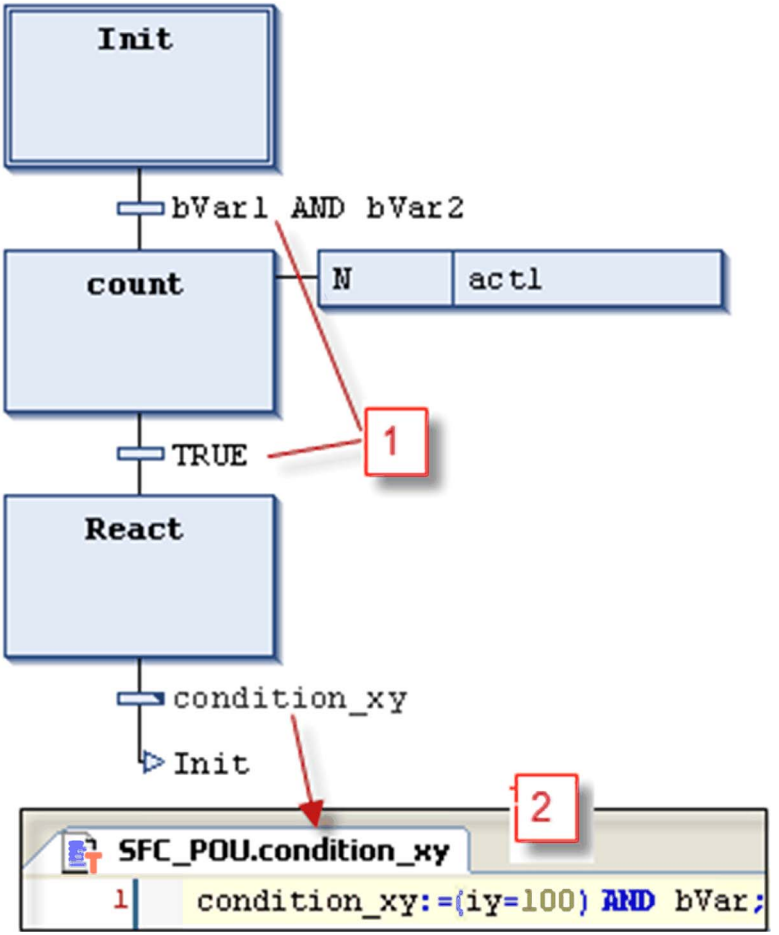
NOTA: Si una transición produce varias instrucciones, asigne la expresión deseada a una variable de transición.

NOTA: Las transiciones que constan de un objeto de transición o de propiedad se indican mediante un pequeño triángulo en la esquina superior derecha del rectángulo.

Objeto de transición (transición de uso múltiple):

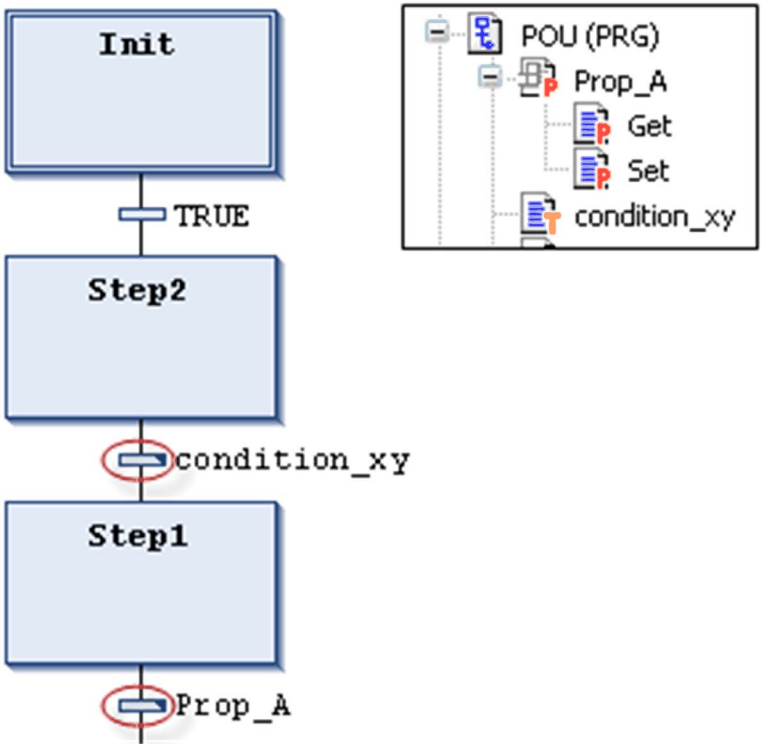


Ejemplos de transiciones:



1 Condiciones de transición introducidas directamente
2 Transición `condition_xy` programada en ST

Las condiciones de uso múltiple (transiciones o propiedades) se indican mediante un triángulo:



A diferencia de versiones anteriores de SoMachine, una llamada a una transición se gestiona como una llamada a un método. Se introducirá de acuerdo con la sintaxis siguiente:

<nombre de transición>:=<condición de transición>;

Ejemplo: `trans1:= (a=100);`

o sólo

<condición de transición>;

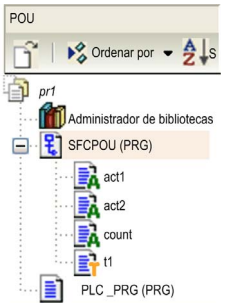
Ejemplo: `a=100;`

Consulte también el ejemplo (`condition_xy`) en la figura *Ejemplos de transiciones*.

Acción

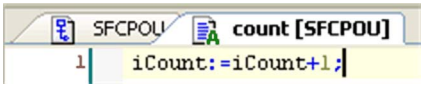
Una acción puede contener una serie de instrucciones escritas en uno de los lenguajes de programación válidos. Se asigna a un paso y, en modalidad online, se procesará de acuerdo con la secuencia de procesamiento *(véase página 383)* definida.

Cada acción que se debe utilizar en pasos SFC debe estar disponible como una POU válida en el SFC POU o el proyecto ( ).



Los nombres de paso deben ser exclusivos en el ámbito de la POU padre. Una acción no puede contener un paso que tenga el mismo nombre que el paso al cual está asignada. De lo contrario, se detectará un error durante el proceso de compilación.

Ejemplo de acción escrita en ST



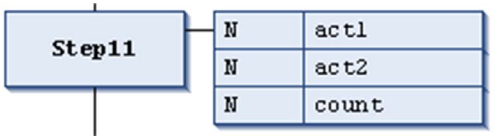
Las acciones de paso conformes a IEC y que extienden IEC se describen en los párrafos siguientes.

Acción de paso conforme a IEC (acción IEC)

Es una acción conforme a la norma IEC61131-3 que se procesará de acuerdo con su descriptor *(véase página 376)* cuando el paso se active, y luego una segunda vez cuando se desactive. En caso de asignar varias acciones a un paso, la lista de acciones se ejecutará de arriba abajo.

- A diferencias de las acciones de paso normales, para las acciones de paso IEC se pueden utilizar descriptores diferentes.
- Una diferencia adicional respecto a las acciones de paso normales es que cada acción de paso IEC está provista de un indicador de control. Esto permite que, aunque otro paso también llame a la acción, la acción se ejecute siempre sólo una vez al mismo tiempo. Esto no es así en el caso de las acciones de paso normales.
- Una acción de paso IEC se representa mediante un módulo doble conectado a la parte derecha de un paso mediante una línea de conexión. En la parte izquierda muestra el descriptor de la acción y en la parte derecha el nombre de la acción. Se pueden editar las dos en línea.
- Las acciones de paso IEC se asocian a un paso por medio del comando **Insertar asociación de acción**. Puede asociar una o varias acciones con un paso. La posición de la nueva acción depende de la posición actual del cursor y del comando. Las acciones deben estar disponibles en el proyecto y se deben insertar con un nombre de acción exclusivo (por ejemplo `plc_prg.a1`).

Lista de acciones de paso conformes a IEC asociadas con un paso:



Cada módulo de acción de la primera columna muestra el descriptor y, en la segunda, se muestra el nombre de la acción.

Acciones de paso que extienden IEC

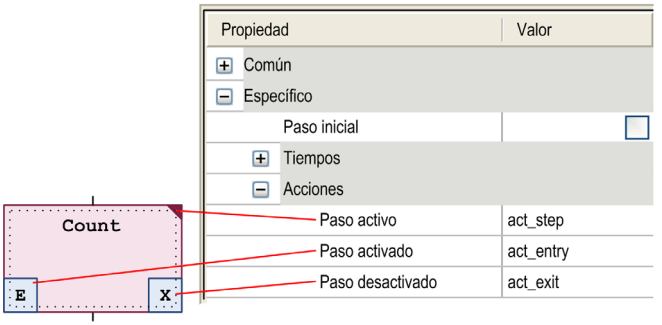
Se trata de acciones que extienden la norma IEC. Deben estar disponibles como objetos debajo del objeto SFC. Seleccione nombres de acción exclusivos. Se definen en las propiedades del paso.

La tabla muestra las acciones de paso que extienden IEC:

Tipo de acción	Procesamiento	Asociación	Representación
acción de entrada del paso (paso activado)	Este tipo de acción de paso se procesará en cuanto el paso se active y antes de la acción activa del paso.	La acción se asocia a un paso por medio de una entrada en el campo **Paso activado** de las propiedades de paso *(véase página 361)*.	Se representa mediante una `E` en la esquina inferior izquierda del módulo de paso respectivo.

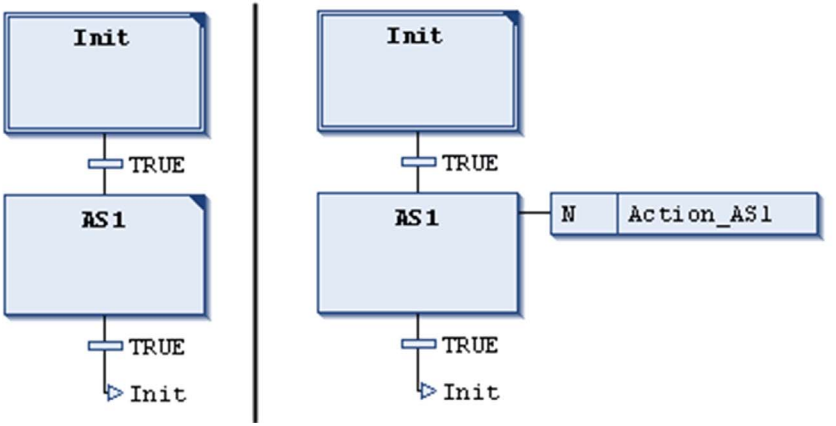
Tipo de acción	Procesamiento	Asociación	Representación
acción activa de paso (acción de paso)	Este tipo de acción de paso se procesará cuando el paso se haya activado y después de que una posible acción de entrada de paso de este paso se haya procesado. No obstante, a diferencia de una acción de paso IEC (véase arriba), no se vuelve a ejecutar cuando se desactiva y no puede obtener descriptores asignados.	La acción se asocia a un paso por medio de una entrada en el campo **Paso activo** de las propiedades de paso *(véase página 361)*.	Se representa mediante un pequeño triángulo en la esquina superior derecha del módulo de paso respectivo.
acción de salida del paso (paso desactivado)	Una acción de salida se ejecutará una vez cuando el paso se desactive. No obstante, esta ejecución no se realizará en el mismo ciclo sino al principio del ciclo posterior.	La acción se asocia a un paso por medio de una entrada en el campo **Paso desactivado** de las propiedades de paso *(véase página 361)*.	Se representa mediante una X en la esquina inferior derecha del módulo de paso respectivo.

Acciones de paso que extienden IEC



Ejemplo: Diferencia entre acciones de paso conformes a / que extienden IEC

La principal diferencia entra las acciones de paso y las acciones IEC con descriptor `N` es que la acción IEC se ejecuta como mínimo dos veces: la primera vez cuando el paso está activo y la segunda vez cuando el paso se desactiva. Consulte el ejemplo siguiente.



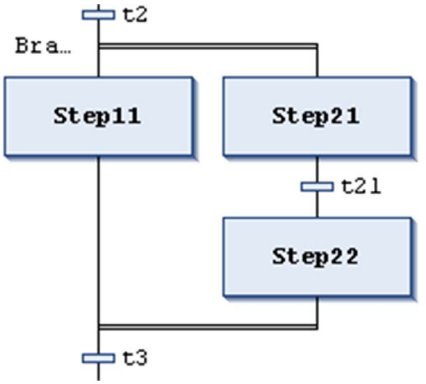
La acción `Action_AS1` está asociada con el paso `AS1` como acción de paso (izquierda), o como acción IEC con el descriptor `N` (derecha). Debido al hecho de que en ambos casos se utilizan 2 transiciones, se deberán realizar 2 ciclos del controlador para cada una antes de que se vuelva a llegar al paso inicial, suponiendo que una variable `iCounter` se incremente en `Action_AS1`. Tras una reactivación del paso `Init`, `iCounter` en el ejemplo de la izquierda tendrá el valor 1. En la derecha, no obstante, tendrá el valor 2 porque la acción IEC, debido a la desactivación de `AS1`, se ha ejecutado dos veces.

Para obtener más información sobre calificadores, consulte la list of available qualifiers *(véase página 376)*.

Bifurcaciones

Un diagrama funcional secuencial (SFC) puede ramificarse; es decir, las líneas de procesamiento se pueden bifurcar en 2 o más líneas adicionales (bifurcaciones). Las bifurcaciones simultáneas *(véase página 372)* se procesarán en paralelo (simultáneamente). En el caso de las bifurcaciones alternativas *(véase página 371)*, sólo se procesará una de acuerdo con la condición de transición anterior. Cada bifurcación de un diagrama va precedida de una línea horizontal doble (paralela) o simple (alternativa) y también finaliza con dicha línea o con un salto *(véase página 374)*.

Bifurcación simultánea



Una bifurcación simultánea debe empezar y finalizar con un paso. Las bifurcaciones simultáneas pueden contener bifurcaciones alternativas u otras bifurcaciones simultáneas.

Las líneas horizontales anteriores y posteriores al área bifurcada son líneas dobles.

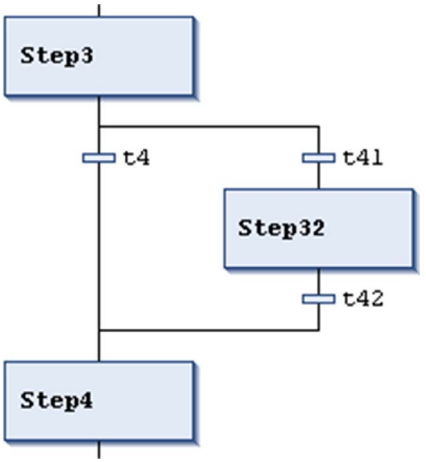
Procesamiento en modalidad online: si la transición anterior (`t2` en el ejemplo que se muestra en la parte izquierda) es verdadera (TRUE), los primeros pasos de todas las bifurcaciones simultáneas se activarán (`Step11` y `Step21`). Las bifurcaciones específicas se procesarán simultáneamente antes de que la transición (`t3`) se reconozca.

Para insertar una bifurcación simultánea, seleccione un paso y ejecute el comando **Insertar rama a la derecha**.

Puede transformar bifurcaciones simultáneas y alternativas entre sí ejecutando los comandos **Paralelo** o **Alternativa**.

Se añadirá automáticamente una etiqueta de bifurcación en la línea horizontal anterior a la bifurcación que lleva por nombre `Branch<n>` donde `n` es un número consecutivo que empieza por 0. Puede especificar esta etiqueta al definir un destino de salto *(véase página 374)*

Bifurcación alternativa



Las líneas horizontales anteriores y posteriores al área bifurcada son líneas simples.

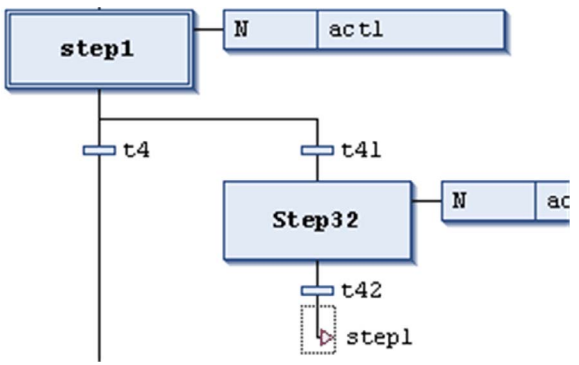
Una bifurcación alternativa debe empezar y finalizar con una transición. Las bifurcaciones alternativas pueden contener bifurcaciones simultáneas y otras bifurcaciones alternativas.

Si el paso que precede a la línea de inicio alternativa está activo, entonces la primera transición de cada bifurcación alternativa se evaluará de izquierda a derecha. La primera transición de la izquierda cuya condición de transición tenga el valor TRUE se abrirá y se activarán los pasos siguientes.

Para insertar bifurcaciones alternativas, seleccione una transición y ejecute el comando **Insertar rama a la derecha**.

Puede transformar bifurcaciones simultáneas y alternativas entre sí ejecutando los comandos **Paralelo** o **Alternativa**.

Salto



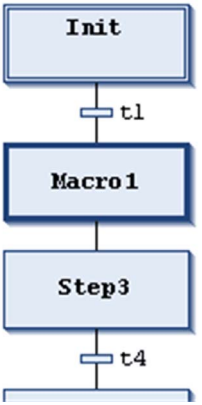
Un salto se representa mediante una línea de conexión vertical más una flecha horizontal y el nombre del destino del salto. Define el siguiente paso que se debe procesar en cuanto la transición anterior tenga el valor TRUE. Se pueden utilizar saltos para evitar que las líneas de procesamiento se crucen o conduzcan hacia arriba.

Además del salto predeterminado al final del diagrama, un salto sólo se puede utilizar al final de una bifurcación. Para insertar un salto, seleccione la última transición de la bifurcación y ejecute el comando **Insertar salto**.

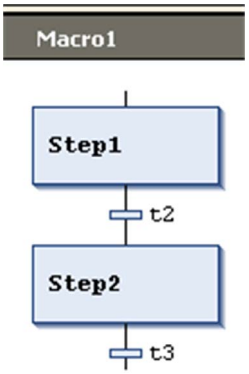
El destino del salto se especifica mediante la cadena de texto asociada, que se puede editar. Puede ser un nombre de paso o la etiqueta de una bifurcación simultánea.

Macro

Vista principal del editor SFC



Vista del editor de macros para `Macro1`



Una macro se representa mediante un cuadro con marco en negrita que contiene el nombre de la macro. Incluye parte del diagrama SFC, que por lo tanto no resulta directamente visible en la vista principal del editor. El uso de macros no afecta al flujo del proceso, sólo es un modo de ocultar algunas partes del programa, por ejemplo, para simplificar la visualización. Para insertar un cuadro de macro, ejecute el comando **Insertar macro(después)**. El nombre de la macro se puede editar.

Para abrir el editor de macros, haga doble clic en el cuadro de macro o ejecute el comando **Mostrar macro**. Puede editarla aquí igual que en la vista principal del editor e introducir la sección deseada del diagrama SFC. Para salir, ejecute el comando **Salir de la macro**.

La línea de título del editor de macros muestra la ruta de la macro en el ejemplo de SFC actual:

Calificador para las acciones en SFC

Descripción general

Para configurar de qué forma las acciones *(véase página 368)* deben asociarse a los pasos IEC, hay disponibles algunos calificadores, que deben insertarse en el campo de calificador de un elemento de acción.

Calificadores disponibles

Calificador	Nombre desarrollado	Descripción
N	no almacenada	La acción se mantiene activa mientras el paso esté activo.
R0	restablecimiento de anulación	La acción se desactiva.
S0	establecida (almacenada)	La acción se iniciará cuando el paso esté activo y se continuará una vez que el paso esté desactivado y hasta que la acción se restablezca.
L	limitada en el tiempo	La acción se iniciará cuando el paso esté activo. Continuará hasta que el paso se desactive o haya pasado un tiempo determinado.
D	retardada	Se iniciará un temporizador de retardo cuando el paso esté activo. Si el paso sigue activo después del retardo, la acción se iniciará y continuará hasta que se desactive.
P	pulso	La acción se iniciará cuando el paso esté activo/inactivo y se ejecutará una sola vez.
SD	almacenada y retardada	La acción se iniciará tras el retardo establecido y continuará hasta que se restablezca.
DS	retardada y almacenada	Si el paso sigue activo después del retardo especificado, la acción se iniciará y continuará hasta que se restablezca.
SL	almacenada y limitada en el tiempo	La acción se iniciará cuando el paso esté activo y continuará durante el tiempo especificado o hasta que se produzca un restablecimiento.

Los calificadores L, D, SD, DS y SL requieren un valor de tiempo en formato constante TIME.

NOTA: Al desactivar una acción IEC, se ejecutará una vez más. Por consiguiente, cada acción se ejecutará al menos dos veces.

Variables implícitas - Indicadores SFC

Descripción general

Cada paso SFC y acción IEC proporciona variables generadas de forma implícita para la observación del estado *(véase página 377)* de los pasos y acciones IEC durante el tiempo de ejecución. También puede definir variables para observar y controlar la ejecución de un SFC (tiempos de espera, restablecimiento, modo de información). Estas variables también las puede generar implícitamente el objeto SFC.

Básicamente, para cada paso y cada acción IEC, se genera una variable implícita. Una instancia de estructura, llamada así por el elemento, por ejemplo, paso 1 para un paso con nombre de paso **step1**. Puede definir en las propiedades del elemento *(véase página 361)* si se debe exportar una definición de símbolo para este indicador a la configuración de símbolos y de qué forma este símbolo debe ser accesible en el controlador.

Los tipos de datos para esas variables implícitas están definidos en la biblioteca IecSFC.library. Esta biblioteca se incluirá automáticamente en el proyecto tan pronto como se añada un objeto SFC.

Estado de paso y acción, y tiempo de paso

Básicamente, para cada paso y acción IEC se crea una variable de estructura implícita de tipo `SFCStepType` o `SFCActionType`. Los componentes de la estructura (indicadores) describen el estado de un paso o una acción, o el tiempo procesado de un paso activo.

La sintaxis para la declaración de variable implícita es:

<nombre de paso>: SFCStepType;

o

_<nombre de acción>:SFCActionType;

NOTA: A diferencia de las versiones anteriores de SoMachine, las variables implícitas de las acciones están precedidas por un guión bajo en las versiones V4.0 y posteriores.

Están disponibles los siguientes indicadores booleanos para los estados de paso o acción:

Indicadores booleanos para **los pasos**:

Indicador booleano	Descripción
<nombre de paso>.x	muestra el estado de activación actual
<nombre de paso>._x	muestra el estado de activación para el siguiente ciclo

Si <nombre de paso>.x = TRUE, el paso se ejecutará en el ciclo actual.

Si <nombre de paso>._x = TRUE y <nombre de paso>.x = FALSE, el paso se ejecutará en el siguiente ciclo. Esto significa que <nombre de paso>._x se copia a <nombre de paso>.x al principio de un ciclo.

Indicadores booleanos para **acciones**:

Indicador booleano	Descripción
_<nombre de acción>.x	es TRUE si se ejecuta la acción
_<nombre de acción>._x	es TRUE si la acción está activa

Generación de símbolos

En las propiedades del elemento *(véase página 361)* de un paso o una acción, se puede definir si se debe añadir una definición de símbolo a una configuración de símbolos posiblemente creada y descargada para el indicador de paso o nombre de acción. Para ello, haga una entrada para el derecho de acceso deseado en la columna **Símbolo** de la vista de propiedades del elemento.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

Si utiliza el indicador booleano <nombre de paso>.x para forzar un determinado valor de estado para un paso (para establecer un paso como activo), tenga en cuenta que esto afectará a todos los estados no controlados en el SFC.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

Tiempo a través de variables TIME:

El indicador `t` ofrece el intervalo de tiempo actual que ha transcurrido desde que se activó el paso. Esto sólo se aplica a los pasos, no importa si existe o no un tiempo mínimo configurado en los atributos de paso *(véase página 361)* (vea más adelante: `SFCError`).

Para **pasos**:

<nombre de paso>.t (<nombre de paso>._t no utilizable para propósitos externos)

Para **acciones**:

No se utilizan variables de tiempo implícitas.

Control de la ejecución SFC (tiempos de espera, restablecimiento, modo de información)

Puede utilizar algunas variables implícitas disponibles, también denominadas indicadores SFC (vea la tabla siguiente) para controlar el funcionamiento de un SFC. Por ejemplo, para indicar los desbordamientos de tiempo o activar el modo de información para cambiar las transiciones.

Para poder acceder a estos indicadores, tiene que declararlos y activarlos. Para ello, utilice el cuadro de diálogo **Configuración SFC**. Éste es un subcuadro de diálogo del cuadro de diálogo **Propiedades**.

La declaración manual, que se necesitaba en SoMachine V3.1, sólo es necesaria para permitir el acceso de escritura desde otra POU (consulte el párrafo *Acceso a los indicadores*).

En este caso, considere lo siguiente:

Si declara el indicador globalmente, debe desactivar la opción **Declarar** en el cuadro de diálogo **Configuración SFC**. De lo contrario, esto da lugar a un indicador local declarado implícitamente, que se emplearía en lugar del global. Tenga en cuenta que la configuración SFC para una POU SFC se determina inicialmente por las definiciones establecidas en el cuadro de diálogo **Opciones →SFC**.

Tenga presente que la declaración de una variable de indicador realizada únicamente a través del cuadro de diálogo **Configuración SFC** sólo será visible en la vista online SFC POU.

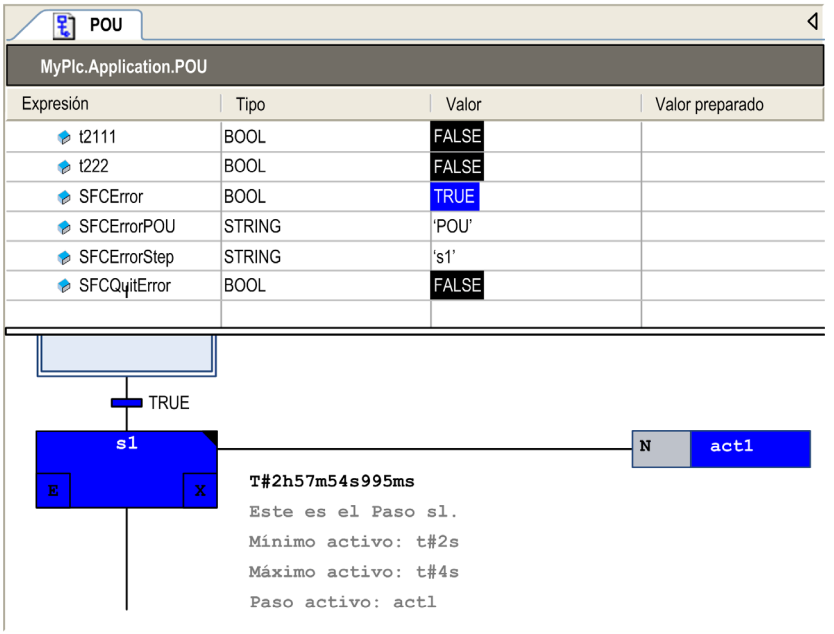
Se pueden utilizar las siguientes variables implícitas (indicadores). Para este fin, tiene que declararlas y activarlas en el cuadro de diálogo **Configuración SFC**.

Variable	Tipo	Descripción
`SFCInit`	BOOL	Si esta variable se convierte en TRUE, el diagrama funcional secuencial se establecerá de nuevo en el Paso inicial *(véase página 363)*. Todos los pasos y acciones y otros indicadores SFC se restablecerán (inicialización). El paso inicial se mantendrá activo, pero no se ejecutará mientras la variable sea TRUE. Vuelva a ajustar `SFCInit` a FALSE para volver al procesamiento normal.
`SFCReset`	BOOL	Esta variable se comporta de forma similar a `SFCInit`. A diferencia de esta última, sin embargo, se realiza un posterior procesamiento después de la inicialización del paso inicial. Por lo tanto, en este caso, se puede realizar un restablecimiento a FALSE del indicador `SFCReset` en el paso inicial.
`SFCError`	BOOL	Tan pronto como se produzca un timeout en 1 de los pasos en el SFC, esta variable se convertirá en TRUE. Condición previa: `SFCEnableLimit` debe ser TRUE. Tenga en cuenta que no se puede registrar un timeout adicional antes de un restablecimiento de `SFCError`. Se debe definir `SFCError` si desea utilizar los otros indicadores de control de tiempo (`SFCErrorStep`, `SFCErrorPOU` y `SFCQuitError`).

Variable	Tipo	Descripción
SFCEnableLimit	BOOL	Puede utilizar esta variable para la activación (TRUE) y desactivación (FALSE) explícitas del control de tiempo en los pasos a través de SFCError. Esto significa que si esta variable se declara y se activa(**Configuración SFC**) entonces se debe establecer en TRUE con el fin de conseguir que SFCError funcione. De lo contrario, no se registrará ningún timeout de los pasos. El uso puede ser razonable en inicios o en funcionamiento manual. Si la variable no está definida, SFCError funcionará de forma automática. Condición previa: se debe definir SFCError.
SFCErrorStep	STRING	Esta variable almacena el nombre de un paso en el que SFCError.timeout registró un timeout. Condición previa: se debe definir SFCError.
SFCErrorPOU	STRING	Esta variable almacena el nombre de la POU SFC en la que se ha producido un timeout. Condición previa: se debe definir SFCError.
SFCQuitError	BOOL	Siempre que esta variable sea TRUE, la ejecución del diagrama SFC se detendrá y la variable SFCError se restablecerá. Tan pronto como la variable se haya restablecido a FALSE, todos los estados de tiempo actuales en los pasos activos se restablecerán. Condición previa: se debe definir SFCError.
SFCPause	BOOL	Siempre que esta variable sea TRUE, la ejecución del diagrama SFC se detendrá.
SFCTrans	BOOL	Esta variable se convierte en TRUE tan pronto como se acciona una transición.
SFCCurrentStep	STRING	Esta variable almacena el nombre del paso activo, independientemente de la supervisión del tiempo. En caso de secuencias simultáneas, se registrará el nombre del paso exterior derecho.
SFCTipSFCTipMode	BOOL	Estas variables permiten utilizar el modo marcha lenta en la gráfica actual. Cuando este modo se ha activado por SFCTipMode=TRUE, sólo puede saltar al siguiente paso mediante el establecimiento de SFCTip=TRUE (flanco ascendente). Mientras SFCTipMode esté establecido en FALSE, se puede saltar por las transiciones.

La imagen siguiente muestra un ejemplo de varios indicadores de errores SFC detectados en modalidad online del editor.

Se ha detectado un timeout en el paso `s1` del objeto SFC **POU** con el indicador `SFCError`.



Acceso a los indicadores

Para permitir el acceso a los indicadores para el control de la ejecución del SFC (timeouts, restablecimiento, modo de información), declare y active las variables de indicador como se describió anteriormente (Control de la ejecución SFC *(véase página 379)*).

Sintaxis para el acceso desde una acción o transición dentro de la POU SFC:

<nombre de paso>.<indicador>

o

_<nombre de acción>.<indicador>

Ejemplos:

```
status:=step1._x;
checkerror:=SFCerror;
```

Sintaxis para acceder desde otra POU:

<POU SFC>.<nombre de paso>.<indicador>

o

<SFC POU>_<nombre de acción>.<indicador>

Ejemplos:

```
status:=SFC_prog.step1._x;
checkerror:=SFC_prog.SFCerror;
```

Tenga en cuenta lo siguiente en caso de acceso de escritura desde otra POU:

- La variable implícita, además, se tiene que declarar explícitamente como una variable VAR_INPUT de la POU SFC, o
- Se tiene que declarar globalmente en una GVL (lista de variables globales).

Ejemplo: Declaración local

```
PROGRAM SFC_prog
VAR_INPUT
  SFCinit:BOOL;
END_VAR
```

Ejemplo: Declaración global en una GVL

```
VAR_GLOBAL
  SFCinit:BOOL;
END_VAR
```

Acceso al indicador en `PLC_PRG`:

```
PROGRAM PLC_PRG
VAR
  setinit: BOOL;
END_VAR
SFC_prog.SFCinit:=setinit;  //Write access to SFCinit in SFC_prog
```

Secuencia de procesamiento en SFC

Descripción general

En modalidad online, los tipos de acción concretos se procesarán conforme a una secuencia definida; consulte la tabla siguiente.

Definición de términos

Se utilizan los términos siguientes:

Término	Descripción
paso activo	Un paso cuya acción de paso se está ejecutando. En modalidad online, los pasos activos se muestran de color azul.
paso inicial	En el primer ciclo después de que se haya llamado a una POU SFC, el paso inicial pasa a estar activo automáticamente y se ejecuta el paso asociado acción *(véase página 368)*.
acciones IEC	Las acciones IEC se ejecutan como mínimo dos veces: ● La primera vez cuando se activan. ● La segunda vez (en el ciclo siguiente) cuando se han desactivado.
bifurcaciones alternativas	Si está activo el paso que precede a la línea de inicio horizontal de bifurcaciones alternativas, la primera transición de cada bifurcación concreta se evaluará de izquierda a derecha. Se buscará la primera transición de la izquierda cuya condición de transición tenga el valor TRUE y se ejecutará la bifurcación correspondiente, es decir, que el paso siguiente en esa bifurcación pasará a estar activo.
bifurcaciones simultáneas	Si está activa la línea doble en la línea del principio de las bifurcaciones simultáneas y la condición de transición anterior tiene el valor TRUE, en todas las bifurcaciones simultáneas pasarán a estar activos todos los primeros pasos. Las bifurcaciones se procesarán en paralelo entre sí. Cuando todos los pasos previos estén activos y la condición de transición después de la línea doble tenga el valor TRUE, pasará a estar activo el paso siguiente a la línea doble al final de la bifurcación.

Orden de procesamiento

Orden de procesamiento de elementos en una secuencia:

Paso	Descripción
1. Restablecimiento de las acciones IEC	Todos los indicadores de control de acción de las acciones *(véase página 368)* IEC se restablecerán (sin embargo, no se restablecerán los indicadores de las acciones IEC que se llaman en las acciones).
2. Acciones de salida del paso (paso desactivado)	Se comprueban todos los pasos en el orden que tienen en el diagrama de secuencia (de arriba abajo y de izquierda a derecha) para determinar si se cumple el requisito para la ejecución de la acción de salida del paso. Si ese es el caso, la acción se ejecutará. Si el paso se va a desactivar *(véase página 363)*, se ejecutará una acción de salida. Esto es así si se han ejecutado las acciones de entrada y de paso (si existen) durante el último ciclo y si la transición para el paso siguiente es TRUE.
3. Acciones de entrada del paso (paso activado)	Se comprueban todos los pasos en el orden que tienen en la secuencia para determinar si se cumple el requisito para la ejecución de la acción de entrada del paso. Si ese es el caso, la acción se ejecutará. La acción de entrada se ejecutará si la condición de transición que precede al paso es TRUE y, por tanto, el paso se ha activado.
4. Comprobación de timeout, acciones de paso activo	Para pasos que no sean IEC, la acción de paso activo correspondiente se ejecuta ahora en el orden en que se han colocado en la secuencia (de arriba abajo y de izquierda a derecha).
5. Acciones IEC	Las acciones *(véase página 368)* IEC que se utilizan en la secuencia se ejecutan en orden alfabético. Esto se lleva a cabo en 2 pasos a través de la lista de acciones. En el primer paso, se ejecutan todas las acciones IEC que están desactivadas en el ciclo actual. En el segundo paso, se ejecutan todas las acciones IEC que están activas en el ciclo actual.
6. Comprobación de transición, activación de los pasos siguientes	Las transiciones *(véase página 363)* se evalúan. Si el paso del ciclo actual estaba activo y la transición siguiente devuelve TRUE (y, si procede, el tiempo activo mínimo ya ha transcurrido), se activa el paso siguiente.

NOTA: Una acción se puede ejecutar varias veces en 1 ciclo porque se llama desde más de una de las demás acciones IEC cuando hay varios pasos activos. Es decir, la misma acción IEC se utiliza simultáneamente en diferentes niveles de un SFC, y este hecho podría producir efectos no deseados.

NOTA: Ejemplo: Un SFC podría tener 2 acciones IEC A y B, ambas implementadas en SFC, y que ambas llaman a la acción IEC C. Por tanto, las acciones IEC A y B pueden estar activas en el mismo ciclo y, además, en ambas acciones puede estar activa la acción IEC C. En consecuencia, se llamará a C dos veces.

⚠ ADVERTENCIA
FUNCIONAMIENTO IMPREVISTO DEL EQUIPO No llame a acciones IEC desde varias acciones IEC en el mismo ciclo. **El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.**

NOTA: Para determinar el estado de los pasos y acciones o la ejecución del diagrama, utilice variables implícitas *(véase página 377)*.

Editor SFC en modalidad online

Descripción general

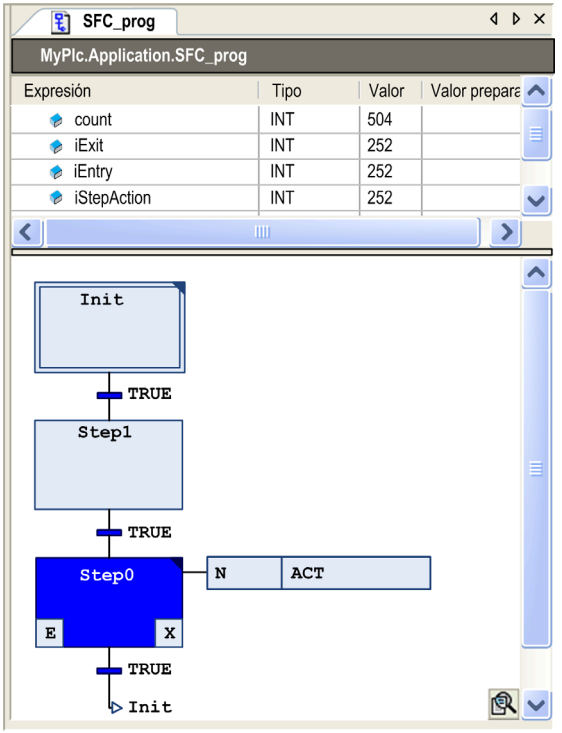
En la modalidad online, el editor SFC proporciona vistas para la supervisión (véase a continuación) y para la escritura y el forzado de las variables y expresiones en el controlador. La funcionalidad de depuración como la de los demás lenguajes IEC (puntos de interrupción, ejecución paso a paso, etc.), no está disponible en SFC. Sin embargo, tenga en cuenta las sugerencias siguientes para depurar SFC:

- Para obtener información acerca de la apertura de objetos en modalidad online, consulte la descripción de la interfaz de usuario en modalidad online *(véase página 49)*.
- La ventana del editor de un objeto SFC también incluye el editor de declaraciones en la parte superior. Para obtener información general, consulte el capítulo *Editor de declaraciones en modalidad online* *(véase página 419)*. Si ha declarado variables implícitas (indicadores SFC) *(véase página 377)* a través del cuadro de diálogo **Configuración** de SFC, se añadirán aquí, pero no se verán en la modalidad offline del editor de declaraciones.
- Tenga en cuenta la secuencia de procesamiento *(véase página 383)* de los elementos de un diagrama funcional secuencial (SFC).
- Consulte las propiedades del objeto o las opciones del editor SFC y los valores predeterminados de SFC para ver la configuración relativa a la compilación o la visualización online de los elementos SFC y sus atributos.
- Contemple la posibilidad de utilizar indicadores *(véase página 377)* para supervisar y controlar el procesamiento de un SFC.

Supervisión

Los pasos activos se muestran de color azul. La visualización de los atributos de los pasos depende de las opciones del editor SFC que se hayan establecido.

Vista online del objeto de programa `SFC_prog`



Capítulo 12
Editor de texto estructurado (ST)

Contenido de este capítulo

Este capítulo contiene las siguientes secciones:

Sección 12.1
Información sobre el editor ST

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Editor ST

Descripción general

El editor ST se utiliza para crear objetos de programación en texto estructurado (ST) del lenguaje de programación IEC o texto estructurado extendido que proporciona algunas extensiones al estándar IEC 61131-3.

El editor ST es un editor de texto. Por lo tanto, utilice la configuración del editor de texto correspondiente en los cuadros de diálogo **Opciones** y **Personalizar** para configurar el comportamiento, la apariencia y los menús. Allí puede definir la configuración predeterminada de la coloración de resalte, números de línea, tabuladores, sangría y muchas otras opciones.

Información adicional

Para seleccionar bloques, pulse la tecla ALT y seleccione el área de texto que desee con el ratón.

El editor estará disponible en la parte inferior de una ventana que también incluye el editor de declaraciones *(véase página 414)* en la parte superior.

Si se detectan errores sintácticos durante la edición, los mensajes correspondientes se mostrarán en la ventana **Precompile Messages**. Esta ventana se actualiza cada vez que se restablece el foco en la ventana del editor (por ejemplo, coloque el cursor en otra ventana y luego de nuevo en la ventana del editor).

Editor ST en modalidad online

Descripción general

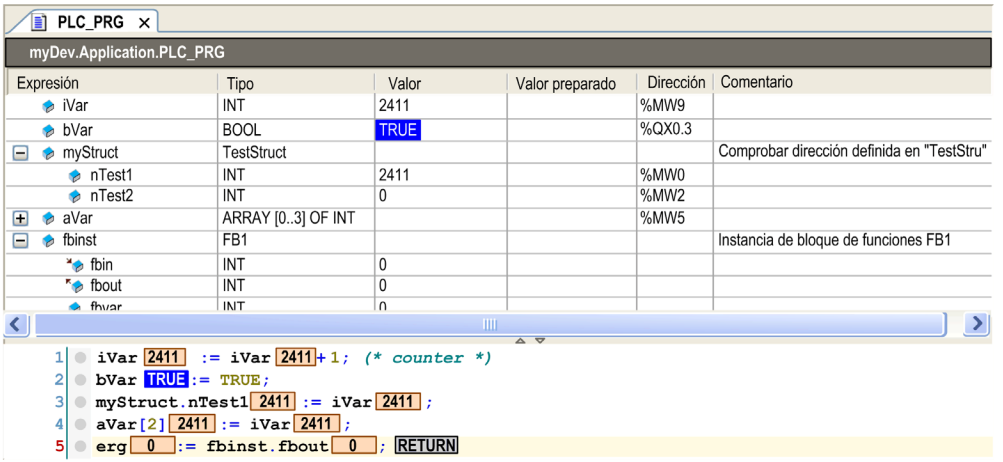
En la modalidad online, el editor de texto estructurado (editor ST) proporciona vistas para supervisar *(véase página 392)*, así como para escribir y forzar las variables y expresiones en el controlador. También hay disponible la depuración (puntos de interrupción, ejecución paso a paso, etc.). Consulte Posiciones de punto de interrupción en el editor ST *(véase página 394)*.

- Para obtener información acerca de la apertura de objetos en modalidad online, consulte la descripción de la interfaz de usuario en modalidad online *(véase página 49)*.
- Para obtener información sobre cómo especificar valores preparados para variables en modalidad online, consulte *Forzado de variables (véase página 393)*.
- La ventana del editor de un objeto ST también incluye el editor de declaraciones en la parte superior. Para obtener información sobre el editor de declaraciones en modalidad online, consulte Editor de declaraciones en modalidad online *(véase página 419)*.

Supervisión

Si la supervisión no se ha desactivado explícitamente en el cuadro de diálogo **Opciones**, se mostrarán pequeños cuadros de supervisión detrás de cada variable con el valor real.

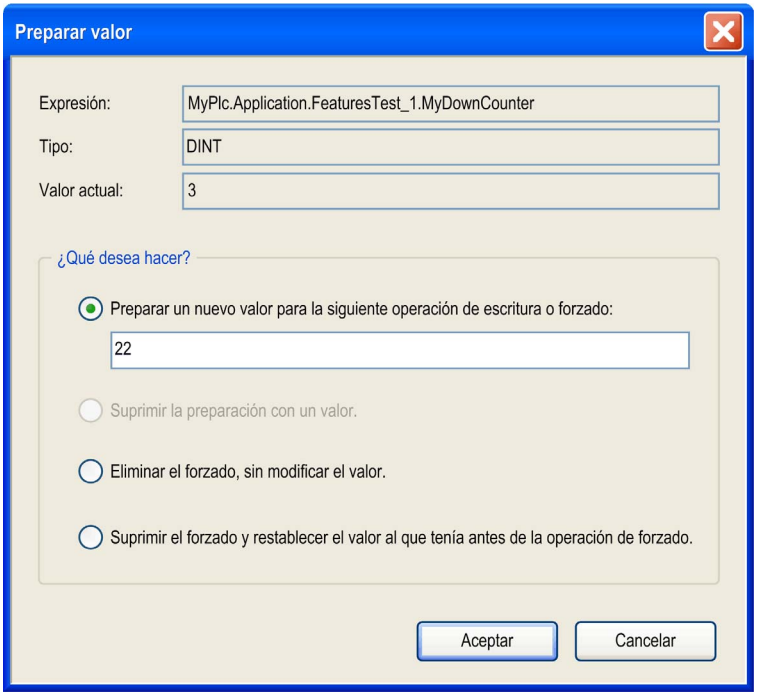
Vista online de un objeto del programa `PLC_PRG` con supervisión:



Forzado de variables

Además de la posibilidad de especificar un valor preparado para una variable en la declaración de cualquier editor, el editor ST permite hacer doble clic en el cuadro de supervisión de una variable en la parte de implementación (en modalidad online). Especifique el valor preparado en el cuadro de diálogo que aparece.

Cuadro de diálogo **Preparar valor**



Se muestra el nombre de la variable completado con su ruta en el árbol **Dispositivos** (**Expresión**), el tipo y el valor actual.

Al activar el elemento correspondiente, puede elegir las siguientes opciones:

- Preparar un valor nuevo que se debe introducir en el campo de edición.
- Eliminar un valor preparado.
- Levantar el forzado de la variable.
- Levantar el forzado de la variable y restablecerla al valor que tenía antes de la operación de forzado.

Para realizar la acción seleccionada, ejecute el comando **Debug** →**Forzar valores** (opción **Online**) o pulse la tecla F7.

Posiciones de punto de interrupción en el editor ST

Puede establecer un punto de interrupción básicamente en las posiciones de una POU en las que los valores de las variables pueden cambiar o el flujo del programa se ramifica o se llama a otra POU. En las descripciones siguientes, `{BP}` indica una posición de punto de interrupción posible.

Asignación:

Al principio de la línea. Tenga en cuenta que las asignaciones como expresiones *(véase página 398)* no definen más posiciones de puntos de interrupción dentro de una línea.

Bucle **FOR**:
1. antes de la inicialización del contador
2. antes de la prueba del contador
3. antes de una instrucción

```
{BP} FOR i := 12 TO {BP} x {BP} BY 1 DO
{BP} [statement1]
...
{BP} [statementn-2]
END_FOR
```

Bucle **WHILE**:
1. antes de comprobar la condición
2. antes de una instrucción

```
{BP} WHILE i < 12 DO
{BP} [statement1]
...
{BP} [statementn-1]
END_WHILE
```

Bucle **REPEAT**:
- antes de comprobar la condición

```
REPEAT
{BP} [statement1]
...
{BP} [statementn-1]
{BP} UNTIL i >= 12
END_REPEAT
```

Llamada de un programa o un bloque de funciones:

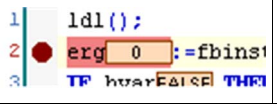
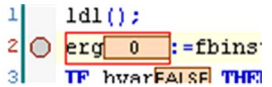
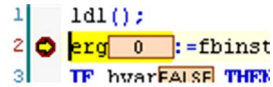
Al principio de la línea.

```
{{BP} POU( );
```

Al final de una POU:

En la ejecución paso a paso, también se alcanzará esta posición después de una instrucción `RETURN`.

Visualización de puntos de interrupción en ST

Punto de interrupción en modalidad online	Punto de interrupción deshabilitado	Parada de programa en punto de interrupción
		

NOTA: Se establecerá automáticamente un punto de interrupción en todos los métodos que puedan llamarse. Si se llama un método gestionado por interfaces, los puntos de interrupción se establecerán en todos los métodos de bloques de funciones que implementen esa interfaz y en todos los bloques de funciones derivados que suscriban el método. Si se llama un método mediante un puntero en un bloque de funciones, se establecerán puntos de interrupción en el método del bloque de funciones y en todos los bloques de funciones derivados que suscriban el método.

Sección 12.2
Lenguaje de texto estructurado (ST)/texto estructurado extendido (ExST)

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Texto estructurado (ST)/texto estructurado extendido (ExST)

Descripción general

El texto estructurado es un lenguaje de programación de alto nivel textual, parecido a PASCAL o C. El código de programa consta de expresiones *(véase página 398)* e instrucciones *(véase página 400)*. En contraposición a IL (Lista de instrucciones), puede usar varias construcciones para programar bucles, lo que permite desarrollar algoritmos complejos.

Ejemplo

```
IF value < 7 THEN
 WHILE value < 8 DO
  value:=value+1;
 END_WHILE;
END_IF;
```

El texto estructurado extendido (ExST) es una extensión específica de SoMachine para el estándar IEC 61131-3 para texto estructurado (ST). Ejemplos: Asignación como expresión, operadores Set/Reset.

Expresiones

Descripción general

Una expresión es una construcción que devuelve un valor tras su evaluación. Este valor se utiliza en las instrucciones.

Las expresiones están compuestas por operadores *(véase página 703)*, operandos *(véase página 797)* o asignaciones. Un operando puede ser una constante, una variable, una llamada de función u otra expresión.

Ejemplos

`33`	(* Constante *)
`ivar`	(* Variable *)
`fct(a,b,c)`	(* Llamada de función*)
`a AND b`	(* Expresión *)
`(x*y) / z`	(* Expresión *)
`real_var2 := int_var;`	(* Asignación, véase a continuación *)

Orden de las operaciones

La evaluación de una expresión se realiza procesando los operadores según unas reglas determinadas. El operador con el orden más alto de operación se procesa primero, seguido del operador con el nivel de operación siguiente y así sucesivamente hasta que se hayan procesado todos los operadores.

A continuación encontrará una tabla de los operadores ST ordenados por nivel de operación ordinal:

Operación	Símbolo	Nivel de operación
entre paréntesis	(expresión)	orden más alto
llamada de función	nombre de la función (lista de parámetros)	.....
exponenciación	EXPT	.....
negación	–	.....
creación de complementos	NOT	.....
multiplicar	*	.....
dividir	/	.....
módulo	MOD	.....
sumar	+	.....
restar	–	.....
comparar V	<,>,<=,>=	.....

Operación	Símbolo	Nivel de operación
igual a	=	....
no es igual a	<>	...
AND booleano	AND	..
XOR booleano	XOR	.
OR booleano	OR	orden más bajo

Asignación como expresión

Como ampliación del estándar IEC 61131-3 (ExST), las asignaciones se pueden utilizar como una expresión.

Ejemplos:

`int_var1 := int_var2 := int_var3 + 9;`	(* `int_var1` e `int_var2` equivalen al valor de `int_var3` + 9*)
`real_var1 := real_var2 := int_var;`	(* las asignaciones correctas, `real_var1` y `real_var2` obtendrán el valor de `int_var` *)
`int_var := real_var1 := int_var;`	(* se mostrará un mensaje debido al conflicto de tipos `real-int` *)

`IF b := (i = 1) THEN`
`i := i + 1;`
`END_IF`
	(*Expresión empleada dentro de una instrucción de condición IF: primero se asignará a `b` TRUE o FALSE, en función de si `i` es 1 o no, tras lo cual se evaluará el valor resultante de `b`.*)

Instrucciones

Descripción general

Las instrucciones describen qué se debe hacer con las expresiones *(véase página 398)* especificadas.

En ST se pueden utilizar las instrucciones siguientes:

Instrucción	Ejemplo
asignación *(véase página 401)*	`A:=B; CV := CV + 1; C:=SIN(X);`
Llamar a un bloque de funciones *(véase página 402)* y utilizar la salida del bloque de funciones	`CMD_TMR(IN := %IX5, PT := 300);` `A:=CMD_TMR.Q`
`RETURN` *(véase página 402)*	`RETURN;`

`IF` *(véase página 403)*	`D:=B*B;`
	`IF D<0.0 THEN`
	`C:=A;`
	`ELSIF D=0.0 THEN`
	`C:=B;`
	`ELSE`
	`C:=D;`
	`END_IF;`

`CASE` *(véase página 404)*	`CASE INT1 OF`
	`1: BOOL1 := TRUE;`
	`2: BOOL2 := TRUE;`
	`ELSE`
	` BOOL1 := FALSE;`
	` BOOL2 := FALSE;`
	`END_CASE;`

`FOR` *(véase página 405)*	`J:=101;`
	`FOR I:=1 TO 100 BY 2 DO`
	`IF ARR[I] = 70 THEN`
	`J:=I;`
	`'EXIT;`
	`END_IF;`
	`END_FOR;`

`WHILE` *(véase página 406)*	`J:=1;`
	`WHILE J<= 100 AND ARR[J] <> 70 D`
	`O`
	`J:=J+2;`
	`END_WHILE;`

`REPEAT` *(véase página 407)*	`J:=-1;`
	`REPEAT`
	`J:=J+2;`
	`UNTIL J= 101 OR ARR[J] = 70`
	`END_REPEAT;`

Instrucción	Ejemplo
EXIT *(véase página 408)*	`EXIT;`
CONTINUE *(véase página 408)*	`CONTINUE;`

JMP *(véase página 408)*	`label1: i:=i+1;`
	`IF i=10 THEN`
	`  JMP label2;`
	`END_IF`
	`JMP label1;`
	`label2:`

instrucción vacía	`;`

Operadores de asignación

Asignación predeterminada

En la parte izquierda de una asignación, hay un operando (variable, dirección) al que se asigna el valor de la expresión de la derecha mediante el operador de asignación `:=`.

Consulte también la descripción del operador *(véase página 716)* `MOVE`, que tiene la misma función.

Ejemplo

```
Var1 := Var2 * 10;
```

Después de completar esta línea, `Var1` tiene el valor de `Var2` multiplicado por diez.

Operador Set `S=`

El valor se establecerá: si se establece una vez como TRUE, seguirá siendo TRUE.

Ejemplo

```
a S= b;
```

`a` obtiene el valor de `b`: si se establece una vez como TRUE, seguirá siendo TRUE, aunque `b` pase a ser FALSE otra vez.

Operador Reset `R=`

El valor se restablecerá: si se establece una vez como FALSE, seguirá siendo FALSE.

Ejemplo

```
a R= b;
```

`a` se establece como FALSE en cuanto `b` = TRUE.

NOTA: En caso de una asignación múltiple, las asignaciones de establecimiento y restablecimiento hacen referencia al último miembro de la asignación.

Ejemplo

```
a S= b R= fun1(par1,par2)
```

En este caso, `b` será el valor de salida restablecido de `fun1`. Pero `a` no obtiene el valor establecido de `b`, sino que obtiene el valor de salida establecido de `fun1`.

Tenga en cuenta que una asignación se puede utilizar como una expresión *(véase página 398)*. Esto es una extensión de la norma IEC 61131-3.

Llamada a bloques de funciones en ST

La llamada a un bloque de funciones (FB, por sus siglas en inglés) se realiza en texto estructurado, de acuerdo con la sintaxis siguiente:

<nombre de instancia de FB>(variable de entrada de FB:=<valor o dirección>|, <variable de entrada de FB adicional:=<valor o dirección>|...variables de entrada de FB adicionales);

Ejemplo

En este ejemplo, se llama a un bloque de funciones de temporizador (`TON`) con asignaciones para los parámetros `IN` y `PT`. A continuación, la variable resultante `Q` se asigna a la variable `A`. El bloque de funciones de temporizador se instancia mediante `TMR:TON;`. La variable resultante, como en IL, se direcciona de acuerdo con la sintaxis <nombre de instancia de FB>.<variable de FB>:

```
TMR(IN := %IX5, PT := 300);
A:=TMR.Q;
```

También hay otra sintaxis disponible para las salidas:

```
fb(in1:=myvar, out1=>myvar2);
```

Instrucción RETURN

Puede utilizar la instrucción `RETURN` para salir de una POU.

Sintaxis

```
RETURN;
```

Ejemplo

```
IF b=TRUE THEN
RETURN;
END_IF;
a:=a+1;
```

Si `b` es TRUE, la instrucción `a:=a+1;` no se ejecutará. Como resultado, se saldrá de la POU inmediatamente.

Instrucción IF

Con la instrucción IF se puede probar una condición y, en función de esta condición, ejecutar instrucciones.

Sintaxis

IF <expresión booleana1> > THEN

<instrucciones IF>

{ELSIF <expresión booleana2> THEN

<instrucciones ELSIF1>

..

..

ELSIF <expresión booleana n> THEN

<instrucciones ELSIF-1>

ELSE

<instrucciones ELSE>}

END_IF;

El segmento entre llaves {} es opcional.

Si la <expresión_booleana1> devuelve el valor TRUE, entonces sólo se ejecutan las <instrucciones_IF> y, como resultado, no se ejecuta ninguna de las otras instrucciones. De lo contrario, las expresiones booleanas que empiezan por <expresión_booleana2> se evalúan una tras otra hasta que 1 de las expresiones devuelve el valor TRUE. Entonces sólo se evalúan las instrucciones posteriores a esta expresión booleana y anteriores a la siguiente expresión ELSE o ELSIF. Si ninguna de las expresiones booleanas da como resultado el valor TRUE, entonces sólo se evalúan las <instrucciones_ELSE>.

Ejemplo

```
IF temp<17
THEN heating_on := TRUE;
ELSE heating_on := FALSE;
END_IF;
```

Aquí, la calefacción se enciende cuando la temperatura baja de los 17 grados. De lo contrario, permanece apagada.

Instrucción CASE

Con la instrucción CASE, puede combinar varias instrucciones condicionadas con la misma variable de condición en una sola construcción.

Sintaxis

CASE <Var1> OF

<valor1>: <instrucción 1>

<valor2>: <instrucción 2>

<valor3, valor4, valor5>: <instrucción 3>

<valor6..valor10>: <instrucción 4>

..

..

<valor n>: <instrucción n>

ELSE <instrucción ELSE>

END_CASE;

Una instrucción CASE se procesa según el modelo siguiente:

- Si la variable de <Var1> tiene el valor <Valor I>, se ejecutará la instrucción <Instrucción I>.
- Si <Var 1> no tiene ninguno de los valores indicados, se ejecutará la <instrucción ELSE>.
- Si la misma instrucción se debe ejecutar para varios valores de las variables, entonces puede escribir estos valores uno después del otro, separados por comas, y de este modo condicionar la ejecución común.
- Si la misma instrucción se debe ejecutar para un rango de valores de una variable, puede escribir el valor inicial y el valor final separados por 2 puntos. Por lo tanto, puede condicionar la condición común.

Ejemplo

```
CASE INT1 OF
1, 5: BOOL1 := TRUE;
      BOOL3 := FALSE;
2: BOOL2 := FALSE;
      BOOL3 := TRUE;
10..20: BOOL1 := TRUE;
      BOOL3:= TRUE;
ELSE
    BOOL1 := NOT BOOL1;
    BOOL2 := BOOL1 OR BOOL2;
END_CASE;
```

Bucle FOR

Con el bucle FOR, puede programar procesos repetitivos.

Sintaxis

INT_Var:INT;

FOR <INT_Var> := <INIT_VALUE> TO <END_VALUE> {BY <tamaño paso>} DO

<instrucciones>

END_FOR;

El segmento entre llaves {} es opcional.

Las <instrucciones> se ejecutan siempre que el contador <INT_Var> no tenga un valor superior a <VALOR_FINAL>. Esto se comprueba antes de ejecutar las <instrucciones> de modo que las <instrucciones> no se ejecuten si <VALOR_INI> es superior a <VALOR_FINAL>.

Cuando se ejecutan <instrucciones>, el valor de <INT_Var> se incrementa con el valor de <tamaño de paso>. El tamaño de paso puede tener cualquier valor entero. Si falta, entonces se establecerá en 1. El bucle finalizará cuando el valor de <INT_Var> sea superior a <VALOR_FINAL>.

Ejemplo

```
FOR Counter:=1 TO 5 BY 1 DO
Var1:=Var1*2;
END_FOR;
Erg:=Var1;
```

Suponiendo que el valor predeterminado de Var1 sea 1. Entonces tendrá el valor 32 después del bucle FOR.

NOTA: Si <VALOR_FINAL> es igual al valor límite para el tipo de datos de <INT_Var> (contador en el ejemplo anterior), se producirá un bucle infinito. Si el contador es de tipo SINT, por ejemplo, y el <VALOR_FINAL> es 127 (el máximo valor positivo para una variable de tipo SINT), entonces el bucle no podrá terminar nunca porque al añadir 1 a este valor máximo la variable resultaría negativa y nunca superaría los límites impuestos por la instrucción FOR.

⚠ ATENCIÓN

BUCLE INFINITO QUE PROVOCA UN FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

Asegúrese de que el tipo de variable utilizado en las instrucciones FOR tenga suficiente capacidad (tenga un límite superior suficientemente elevado) para ajustarse al <VALOR_FINAL> + 1.

El incumplimiento de estas instrucciones puede causar lesiones o daño al equipo.

Puede utilizar la instrucción CONTINUE en un bucle FOR. Esto es una extensión de la norma IEC 61131-3.

Bucle WHILE

Una alternativa al bucle FOR es el bucle WHILE, que ejecuta el bucle si, y mientras, una condición booleana tenga, y siga teniendo, el valor TRUE. Si la condición no tiene inicialmente el valor TRUE, el bucle no se ejecuta. Si la condición que inicialmente tenía el valor TRUE pasa a tener el valor FALSE, el bucle finaliza.

Sintaxis

WHILE <expresión booleana> DO

<instrucciones>

END_WHILE;

Evidentemente, la expresión booleana inicial y en curso debe asumir un valor FALSE en algún momento en las instrucciones del bucle. De lo contario, el bucle no finalizará, lo que provocará una condición de bucle infinito.

⚠ ATENCIÓN

BUCLE INFINITO QUE PROVOCA UN FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

Asegúrese de que el bucle WHILE finalice en las instrucciones del bucle creando una condición FALSE de la expresión booleana.

El incumplimiento de estas instrucciones puede causar lesiones o daño al equipo.

A continuación, se ofrece un ejemplo de instrucciones del bucle que hacen que el bucle finalice:

```
WHILE Counter<>0 DO
 Var1 := Var1*2;
 Counter := Counter-1;
END_WHILE
```

La instrucción REPEAT aún no se ha introducido, por lo que se mueve el párrafo (con modificación) a la parte inferior.

Puede utilizar la instrucción CONTINUE en un bucle WHILE.

Bucle REPEAT

El bucle REPEAT es otra alternativa al bucle FOR, y también al bucle WHILE. El bucle REPEAT se diferencia del bucle WHILE en que la condición de salida sólo se evalúa después de que el bucle se haya ejecutado como mínimo una vez, al final del bucle.

Sintaxis

REPEAT

<instrucciones>

UNTIL <expresión booleana>

END_REPEAT;

Las <instrucciones> se llevan a cabo repetidamente mientras la <expresión booleana> devuelva el valor TRUE. Si la <expresión booleana> ya se produce en la primera evaluación UNTIL, entonces las <instrucciones> sólo se ejecutan una vez. Evidentemente, la <expresión booleana> debe asumir un valor TRUE en algún momento en las instrucciones del bucle. De lo contario, el bucle no finalizará, lo que provocará una condición de bucle infinito.

⚠ ATENCIÓN

BUCLE INFINITO QUE PROVOCA UN FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

Asegúrese de que el bucle REPEAT finalice en las instrucciones del bucle creando una condición TRUE de la expresión booleana.

El incumplimiento de estas instrucciones puede causar lesiones o daño al equipo.

A continuación, se ofrece un ejemplo de instrucciones del bucle que hacen que el bucle finalice:

```
REPEAT
 Var1 := Var1*2;
 Counter := Counter-1;
UNTIL
 Counter=0
END_REPEAT;
```

Puede utilizar la instrucción CONTINUE en un bucle REPEAT. Esto es una extensión de la norma IEC 61131-3.

En cierto sentido, los bucles WHILE y REPEAT son más potentes que el bucle FOR, porque no es necesario saber el número de ciclos antes de ejecutar el bucle. En algunos casos, sólo podrá trabajar con estos dos tipos de bucle. Sin embargo, si la cantidad de bucles está clara, es preferible un bucle FOR porque, en la mayoría de los casos, excluye de forma inherente los bucles infinitos (consulte el mensaje de peligro en el párrafo sobre el bucle FOR *(véase página 405)*).

Instrucción CONTINUE

Como extensión de la norma IEC 61131-3, se admite la instrucción CONTINUE en los bucles FOR, WHILE y REPEAT. CONTINUE hace que la ejecución continúe con el siguiente ciclo del bucle.

Ejemplo

```
FOR Counter:=1 TO 5 BY 1 DO
INT1:=INT1/2;
IF INT1=0 THEN
CONTINUE; (* to avoid division by zero *)
END_IF
Var1:=Var1/INT1; (* only executed, if INT1 is not "0" *)
END_FOR;
Erg:=Var1;
```

Instrucción EXIT

La instrucción EXIT finaliza el bucle FOR, WHILE o REPEAT en el que se encuentra sin tener en cuenta ninguna condición.

Instrucción JMP

Puede utilizar la instrucción JMP para realizar un salto incondicional a una línea de código marcada con una etiqueta de salto.

Sintaxis

JMP <etiqueta>;

La <etiqueta> es un identificador arbitrario pero exclusivo que se coloca al principio de una línea de programa. La instrucción JMP debe ir seguida de la indicación del destino del salto, que tiene que ser igual a una etiqueta predefinida.

⚠ ATENCIÓN

BUCLE INFINITO QUE PROVOCA UN FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

Asegúrese de que el uso de la instrucción JMP sea condicional de modo que tenga como resultado un bucle infinito.

El incumplimiento de estas instrucciones puede causar lesiones o daño al equipo.

A continuación, se ofrece un ejemplo de instrucciones que crean condiciones lógicas que evitan un bucle infinito entre el salto y su destino:

```
aaa:=0;
_label1: aaa:=aaa+1;
(*instructions*)
IF (aaa < 10) THEN
JMP _label1;
END_IF;
```

Mientras la variable `i`, que se inicializa con 0, tenga un valor inferior a 10, la instrucción de salto del ejemplo anterior afectará a un retorno repetido de la línea de programa definida por la etiqueta `_label1`. Por lo tanto, afectará a un procesamiento repetido de las instrucciones comprendido entre la etiqueta y la instrucción `JMP`. Puesto que estas instrucciones también incluyen el incremento de la variable `i`, la condición de salto se infringirá (en la novena comprobación) y el flujo del programa continuará.

También puede conseguir esta funcionalidad utilizando un bucle `WHILE` o `REPEAT` en el ejemplo. Por lo general, se debe evitar el uso de instrucciones de salto porque reducen la legibilidad del código.

Comentarios en ST

Existen 2 posibilidades para escribir comentarios en un objeto de texto estructurado:

- Inicie el comentario con `(*` y ciérrelo con `*)`. Esto permite insertar comentarios que abarcan varias líneas. Ejemplo: (*Esto es un comentario.*)
- Comentarios de una sola línea como extensión a la norma IEC 61131-3: `//` indica el inicio de un comentario que finaliza al final de la línea. Ejemplo: // Esto es un comentario.

Puede colocar los comentarios en cualquier lugar de la parte de declaración o implementación del editor ST.

Comentarios anidados: puede colocar comentarios dentro de otros comentarios.

Ejemplo

```
(*
a:=inst.out; (* to be checked *)
b:=b+1;
*)
```

En este ejemplo, el comentario que empieza con el primer paréntesis no se cierra con el paréntesis a continuación de `checked`, sino con el último paréntesis.

Parte V
Editores de objetos

Contenido de esta parte

Esta parte contiene los siguientes capítulos:

Capítulo 13
Editores de declaraciones

Descripción general

El editor de declaraciones de texto sirve para crear la parte de declaración de un objeto POU. Se puede complementar con una vista tabular. Cualquier modificación realizada en una de las vistas se aplica a la otra de forma inmediata.

En función de la configuración actual de las opciones del editor de declaraciones, estará disponible solamente la vista de texto o solamente la vista tabular. Puede pasar de una a otra mediante los botones (**Textual** / **Tabular**) del borde derecho de la ventana del editor.

Normalmente, el editor de declaraciones se utiliza junto con los editores de los lenguajes de programación. Esto significa que se colocará en la parte superior de la ventana que se abre cuando se va a editar o ver (supervisar) un objeto en modalidad offline u online. En la parte de la declaración se describe el tipo de POU (por ejemplo: PROGRAM, FUNCTION_BLOCK, FUNCTION). Se puede ampliar mediante atributos de pragma globales de POU.

La modalidad online del editor de declaraciones *(véase página 419)* se estructura igual que la de una vista **Supervisar**.

La declaración de variables también se efectúa en **Global Variable Lists** y **Tipos de datos**, que se crean en editores diferentes.

Consulte también el capítulo *Declaración de variables* *(véase página 575)*.

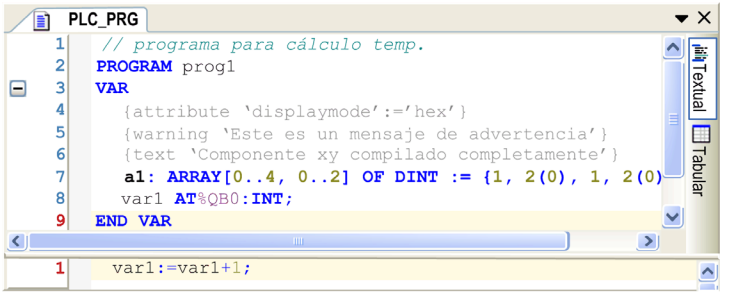
Contenido de este capítulo

Este capítulo contiene los siguiente apartados:

Editor de declaraciones de texto

Descripción general

Vista de texto del editor

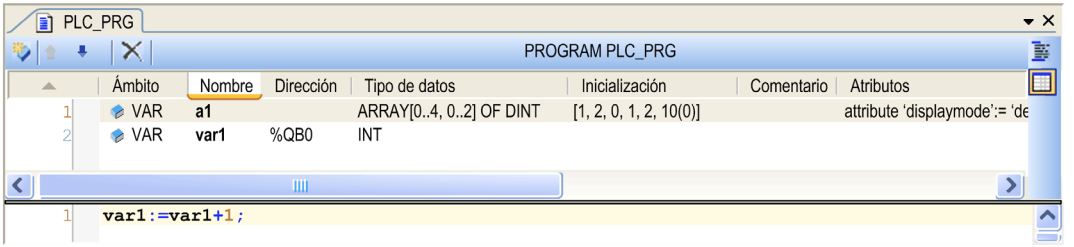


El comportamiento y el aspecto vienen determinados por la configuración del editor de texto actual respectivo en los cuadros de diálogo **Opciones** y **Personalizar**. Allí puede definir la configuración predeterminada de la coloración de resalte, números de línea, tabuladores, sangría y muchas más opciones. Se dispone de las funciones de edición habituales, como copiar y pegar. Se puede utilizar la selección de bloque; para ello, pulse la tecla ALT mientras selecciona con el ratón el área de texto que desea.

Editor de declaraciones tabular

Descripción general

Vista tabular del editor



La vista tabular del editor proporciona columnas para las definiciones habituales para la declaración de variables *(véase página 575)*: **Ámbito**, **Nombre**, **Dirección**, **Tipo de datos**, **Inicialización**, **Comentario** y **Atributos** (pragma). Cada declaración se inserta como una línea numerada.

Para añadir una nueva línea de declaración encima de una ya existente, en primer lugar seleccione esta línea y luego ejecute el comando **Insertar** de la barra de herramientas o el menú contextual.

Para añadir una nueva declaración al final de la tabla, haga clic después de la última línea de declaración existente y utilice también el comando **Insertar**.

La declaración recién insertada utiliza en primer lugar de forma predeterminada el ámbito **VAR** y el tipo de datos recién introducido. El campo de entrada para la variable obligatoria **Nombre** se abre automáticamente. Introduzca un identificador válido y cierre el campo con la tecla INTRO o con un clic en otra parte de la vista.

Haga doble clic en una celda de la tabla para abrir las opciones respectivas para introducir un valor.

Haga doble clic en **Ámbito** para abrir una lista en la que puede elegir el ámbito y la palabra clave de atributo de ámbito (indicador).

Escriba un valor en **Tipo de datos** directamente o haga clic en el botón **>** para utilizar **Accesibilidad** o **Asistente de matriz**.

Escriba un valor en **Inicialización** directamente o haga clic en el botón **...** para abrir el cuadro de diálogo **Valor de inicialización**. Esto es especialmente útil en el caso de las variables estructuradas.

Cada variable se declara en una línea por separado; las líneas están numeradas.

Puede cambiar el orden de las líneas (los números de línea); para ello, seleccione una línea y muévala una línea más arriba o más abajo con el comando **Uno adelante** o **Uno atrás** de la barra de herramientas o del menú contextual.

Puede ordenar la lista de declaraciones según cada una de las columnas si hace clic en el encabezado de columna en cuestión. La columna que determina el orden en un momento dado es la que tiene un símbolo de flecha:

flecha arriba = orden ascendente

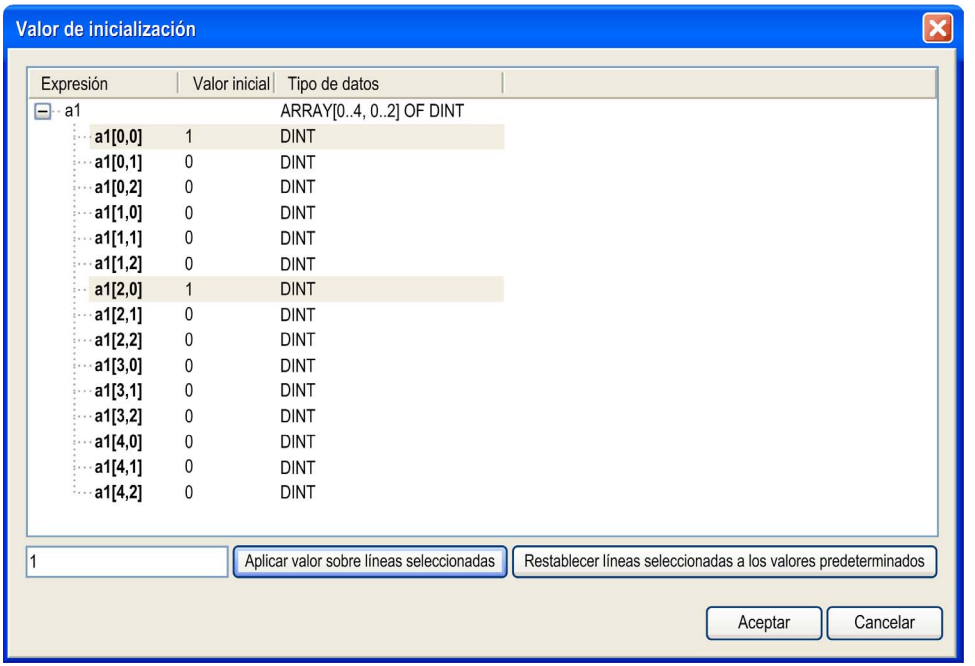
flecha abajo = orden descendente

Con cada clic en el encabezado de columna se alterna entre orden ascendente y descendente.

Para eliminar una o varias declaraciones, seleccione las líneas correspondientes y pulse la tecla SUPR, ejecute el comando **Eliminar** en el menú contextual o haga clic en el botón de la barra de herramientas.

Valor de inicialización

Cuadro de diálogo **Valor de inicialización**



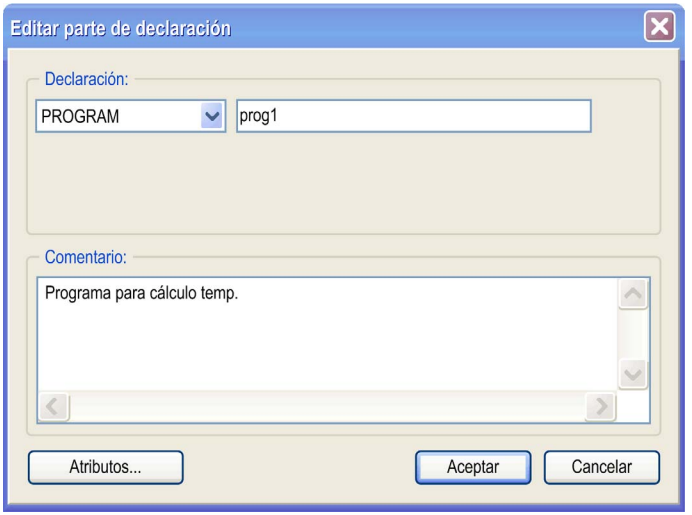
Las **Expresiones** de la variable se muestran con los valores de inicialización actuales. Seleccione las variables que desee y edite el valor de inicialización en el campo que hay bajo la lista. A continuación, haga clic en el botón **Aplicar valor sobre líneas seleccionadas**. Para restaurar las inicializaciones predeterminadas, haga clic en el botón **Restablecer líneas seleccionadas a los valores predeterminados**.

Pulse CTRL + INTRO para insertar saltos de línea en la entrada de **Comentario**.

Editar parte de declaración

Puede editar la parte de declaración en el cuadro de diálogo **Editar parte de declaración**. Para abrirla, haga clic en la barra de encabezado del editor (PROGRAM PLC_PRG en la figura anterior) o utilice el comando **Editar parte de declaración**.

Cuadro de diálogo **Editar parte de declaración**



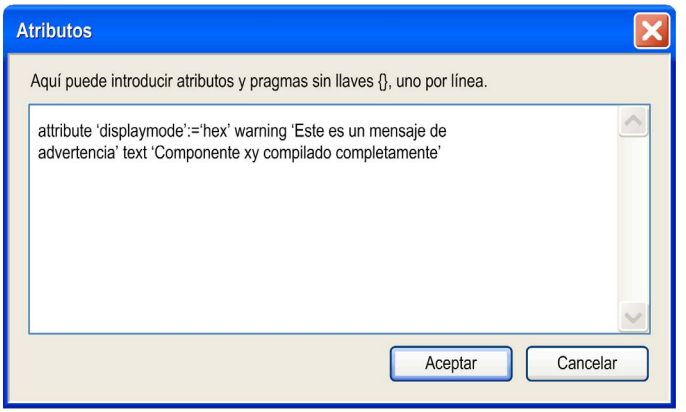
El cuadro de diálogo **Editar parte de declaración** proporciona los elementos siguientes:

Elemento	Descripción
Declaración	Inserte el tipo (de la lista de selección) y el nombre del objeto POU.
Comentario	Inserte un comentario. Pulse CTRL + INTRO para insertar saltos de línea.
Atributos	Abre el cuadro de diálogo **Atributos** (véase más adelante en este capítulo) para insertar los pragmas y los atributos.

Atributos

En el cuadro de diálogo **Editar parte de declaración**, haga clic en el botón **Atributos...** para abrir el cuadro de diálogo **Atributos**. Le permite introducir varios atributos y pragmas en formato de texto. Insértelos sin encerrarlos entre llaves {}; utilice una línea por separado para cada uno. Puede ver la vista de texto correspondiente al ejemplo de la figura siguiente en el gráfico de la vista del editor de texto *(véase página 414)* mostrado anteriormente.

Cuadro de diálogo **Atributos**



Editor de declaraciones en modalidad online

Descripción general

Tras iniciar sesión en el controlador, cada objeto que se haya abierto en una ventana en modalidad offline se mostrará automáticamente en la vista online.

La vista online del editor de declaraciones presenta una tabla similar a la que se utiliza en las vistas de supervisión *(véase página 467)*. En la línea de encabezado se indica la ruta de objeto real <nombre de dispositivo>.<nombre de aplicación>.<nombre de objeto>. La tabla para cada expresión de supervisión indica el tipo y el valor actual, así como un valor preparado (si está definido) para forzar o escribir. Si están disponibles, en las demás columnas se muestran la **Dirección** IEC asignada directamente y el **Comentario**.

Para establecer un valor preparado para una variable, utilice el cuadro de diálogo **Preparar valor** o haga clic en el campo asignado de la columna **Valor preparado** y escriba directamente el valor que desee. En caso de enumeraciones, se abrirá una lista que muestra los valores de enumeración para que pueda seleccionar uno. Si se trata de una variable booleana, el método todavía es más fácil.

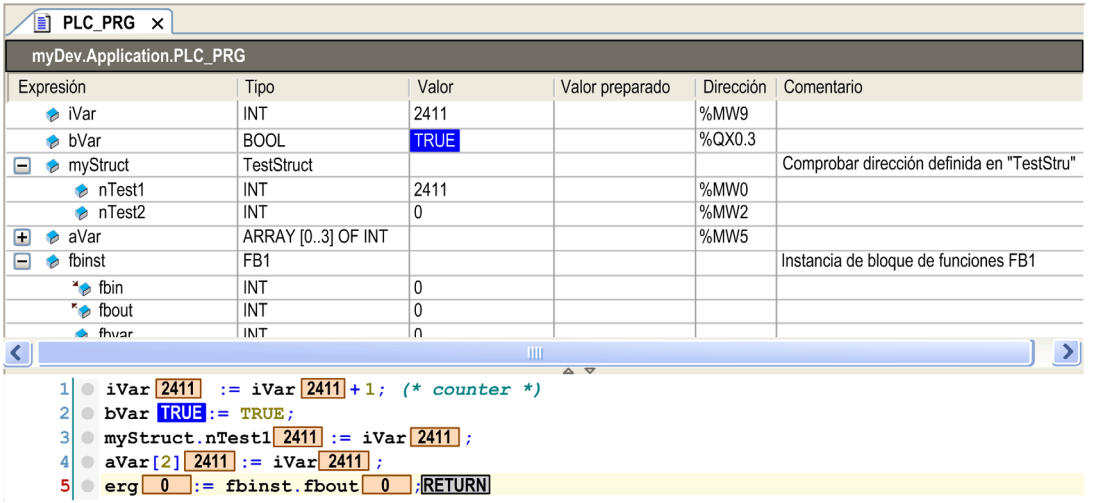
Para alternar los valores de preparación booleanos, utilice la tecla RETORNO o la de espacio en el orden siguiente:

- Si el valor es TRUE, los pasos de preparación son FALSE -> TRUE -> nada.
- Si el valor es FALSE, los pasos de preparación son TRUE -> FALSE -> nada.

Si una expresión de supervisión (variable) es un tipo estructurado, por ejemplo, una instancia de bloque de funciones o una variable de matriz, un signo de más o de menos precederá a la expresión. Con un clic del ratón en ese signo, se mostrarán también los elementos concretos del objeto instanciado (véase `fbinst` en la imagen siguiente) o bien se ocultarán (véase `aVar`). Los iconos indican si la variable es una entrada  , una salida  o una variable normal  .

Si apunta con el cursor a una variable en la parte de implementación, se mostrará la información de la herramienta con la declaración y el comentario de la variable. En la imagen siguiente se muestra el editor de declaraciones en la parte superior del objeto de programa `PLC_PRG` en la vista online:

Vista online del editor de declaraciones



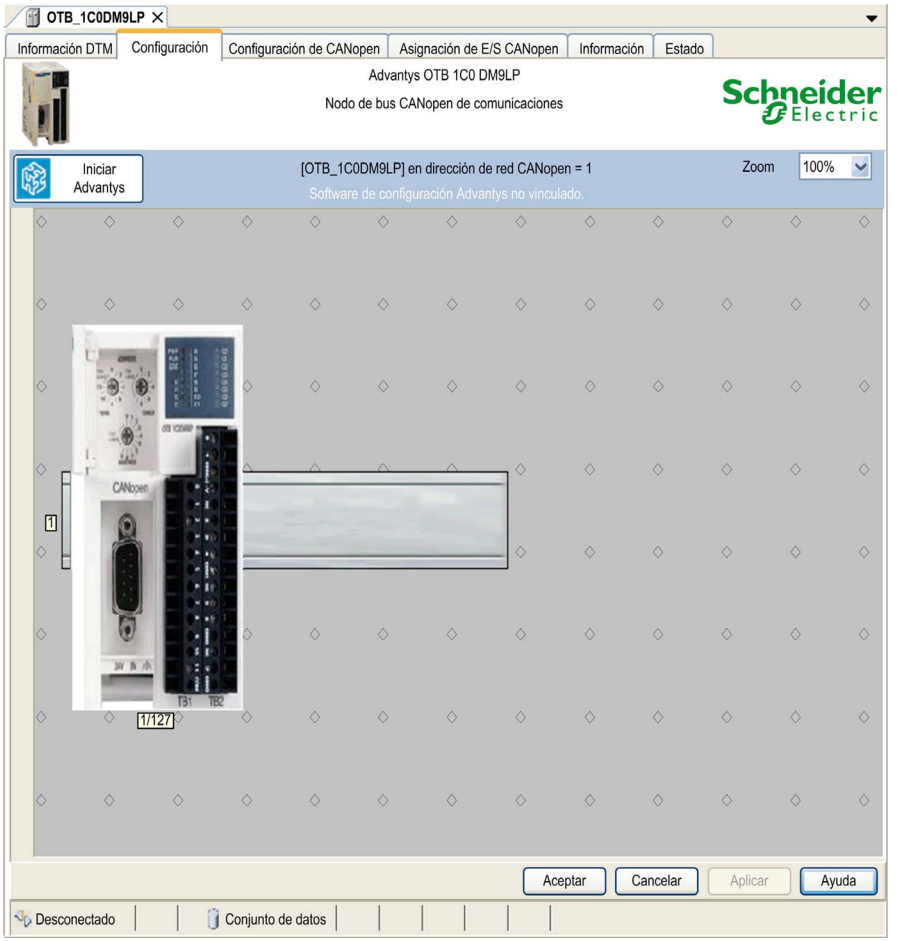
Capítulo 14
Editor de gestor de tipos de dispositivo (DTM)

Editor DTM

Descripción general

La vista del editor DTM depende del gestor de tipos de dispositivo.

El gráfico ofrece un ejemplo de editor DTM.



Para obtener más información acerca de DTM, consulte el documento *SoMachine Device Type Configuration (DTM) - Guía del usuario (véase SoMachine, Device Type Manager (DTM), User Guide)*.

Para obtener una lista de las versiones de DTM que actualmente admite SoMachine, consulte las Notas de la versión de la instalación de SoMachine.

Capítulo 15
Editor de tipo de datos (DUT)

Editor de tipo de datos

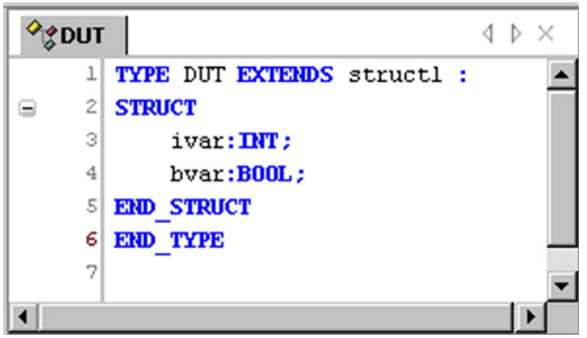
Descripción general

En el editor de tipo de datos (DUT) se pueden crear tipos de datos *(véase página 662)*. Es un editor de texto y funciona conforme a las opciones actuales que se hayan configurado para el editor de texto.

El editor DUT se abrirá automáticamente en una ventana cuando se añada un objeto DUT en el cuadro de diálogo **Agregar objeto**. En ese caso, ofrece de forma predeterminada la sintaxis de una declaración de estructura extendida. Puede utilizarse para especificar una declaración sencilla de estructura o para especificar la declaración de otro tipo de datos, por ejemplo una enumeración.

El editor también se abre cuando se abre un objeto DUT existente seleccionado actualmente en la vista de las POU.

Ventana del editor DUT



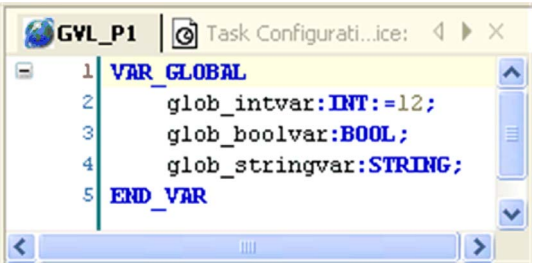
Capítulo 16
Editor de lista de variables globales (GVL)

Editor GVL

Descripción general

El editor GVL es un **editor de declaraciones** para editar **listas de variables globales**. El editor GVL funciona igual que el editor de declaraciones y se corresponde con las opciones, tanto online como offline, establecidas para el editor de texto. La declaración empieza con `VAR_GLOBAL` y acaba con `END_VAR`. Estas palabras clave se proporcionan automáticamente. Introduzca declaraciones válidas de variables globales entre ellas.

Editor GVL



Capítulo 17
Editor de lista de variables de red (NVL)

Contenido de este capítulo

Este capítulo contiene las siguientes secciones:

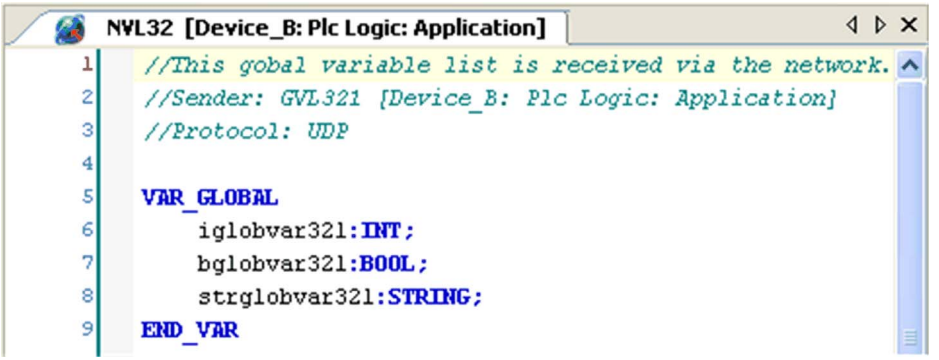
Sección 17.1
Información sobre el editor NVL

Editor de lista de variables de red

Descripción general

El editor NVL es un **editor de declaraciones** para editar **listas de variables de red**. El editor NVL funciona igual que el editor de declaraciones y se corresponde con las opciones, tanto online como offline, establecidas para el editor de texto. La declaración empieza con `VAR_GLOBAL` y acaba con `END_VAR`. Estas palabras clave se proporcionan automáticamente. Introduzca declaraciones de variables globales *(véase página 573)* válidas entre ellas.

Editor NVL



Sección 17.2
Información general sobre las variables de red

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Introducción a lista de variables de red (NVL)

Descripción general

La característica de lista de variables de red (NVL) consiste en una lista fija de variables que se pueden enviar o recibir a través de una red de comunicación. Esto permite intercambiar datos dentro de una red a través de variables de red, si el controlador lo admite (sistema de destino).

La lista se debe definir en los controladores de emisor y de receptor (y puede gestionarse en un único proyecto o en varios). Sus valores se transmiten mediante difusión a través de los datagramas del protocolo de datagramas de usuario (UDP). El UDP es un protocolo de comunicaciones de Internet sin conexión definido por IETF RFC 768. Este protocolo facilita la transmisión directa de datagramas en redes de protocolo de Internet (Internet Protocol, IP). Los mensajes UDP/IP no necesitan una respuesta y, por lo tanto, son perfectos para aplicaciones en las que los paquetes cerrados no requieren retransmisión (como redes y vídeos que necesitan rendimiento en tiempo real).

La funcionalidad NVL es una potente característica de SoMachine. Le permite compartir y monitorizar datos entre los controladores y sus aplicaciones. No obstante, no existen restricciones respecto a la finalidad de los datos intercambiados entre los controladores, incluidos, pero sin limitación a, el intento de enclavamiento de máquinas o procesos o incluso cambios de estado del controlador.

Únicamente usted, el diseñador o el programador de la aplicación, conoce todas las condiciones y factores presentes durante el funcionamiento de la máquina o del proceso y, por tanto, solo usted puede determinar las estrategias de comunicación adecuadas, los enclavamientos y las medidas de seguridad necesarias para el intercambio de datos entre controladores mediante esta característica. Se debe prestar especial atención al monitorizar este tipo de características de comunicación; además, se debe garantizar que el diseño de la máquina o del proceso no presenta riesgos de seguridad para las personas ni los bienes inmuebles.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DE LA MÁQUINA DEBIDO A UNA COMUNICACIÓN INCORRECTA DE LA MISMA

- Debe tener en cuenta las modalidades de fallo potenciales de rutas de control y proporcionar, para ciertas funciones críticas de control, los medios para lograr un estado seguro durante y después de un fallo de ruta, incluidos los cortes de alimentación y los reinicios del sistema.
- Para las funciones de control críticas deben proporcionarse rutas de control separadas o redundantes.
- Debe tener en cuenta las implicaciones de los retrasos de transmisión no esperados o los fallos de la conexión.
- Tenga en cuenta todas las reglamentaciones para la prevención de accidentes y las normativas de seguridad locales.
- Antes de entrar en servicio debe probarse, de forma individual y exhaustiva, cada implementación del equipo que utiliza esta característica.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

Puede utilizar los bloques de funciones Diagnóstico *(véase SoMachine, Configuración de variables de red, Guía de la biblioteca SE_NetVarUdp)* y Gestión de errores *(véase SoMachine, Configuración de variables de red, Guía de la biblioteca SE_NetVarUdp)*, así como los parámetros de propiedades de red para monitorizar el funcionamiento, el estado y la integridad de las comunicaciones que utilizan esta característica. Esta característica ha sido diseñada para compartir y monitorizar datos y no puede utilizarse para funciones de control críticas.

Lista de variables de red (NVL)

Las variables de red que se intercambiarán se definen en los 2 tipos de listas siguientes:

- Listas de variables globales (GVL) en un controlador de transmisión (emisor).
- Listas de variables globales de red (GNVL) en un controlador de recepción (receptor).

Las GVL y GNVL correspondientes contienen las mismas declaraciones de variables. Una vez que haya hecho doble clic en el nodo de la GVL o GNVL en el panel **Dispositivos**, podrá visualizar el contenido de la lista en el editor correspondiente.

Una GVL contiene las variables de red de un emisor. En la opción **Variables de red** del emisor se definen los parámetros de protocolo y transmisión. Según esta configuración, los valores de las variables se difunden dentro de la red. Todos los controladores que dispongan de una GNVL correspondiente pueden recibirlos.

NOTA: Para intercambiar variables de red es necesario instalar las bibliotecas de red que correspondan. Este procedimiento se lleva a cabo de manera automática para el tipo de red estándar de UDP tan pronto como se establezcan las propiedades de red de una GVL.

Las variables de red se transmiten de la GVL (emisor) a una o más GNVL (receptores). Para cada controlador, puede definir GVL, así como GNVL. Por tanto, cada uno de los controladores puede actuar tanto de emisor como de receptor.

El mismo u otro proyecto pueden proporcionar una GVL de emisor. Así que, al crear una GNVL, puede seleccionarse la GVL de emisor en una lista de selección en la que aparezcan todas las GVL disponibles dentro de la red, o puede leerse de un archivo de exportación generado previamente (por ejemplo, utilizando el cuadro de diálogo **Vínculo con archivo**) en la GVL.

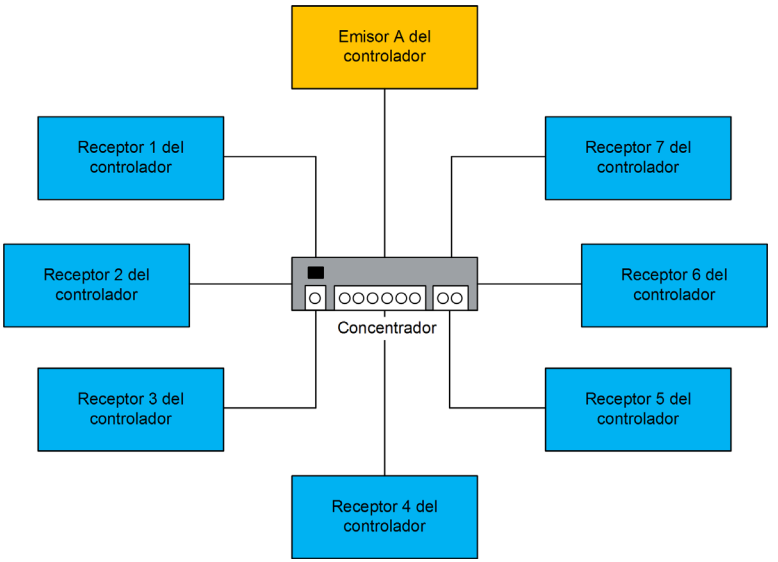
NOTA: Si la GVL de emisor que se utilizará se define en otro proyecto, se necesitará un archivo de exportación.

Consideraciones de NVL

En la tabla siguiente se muestra la lista de controladores que admiten la funcionalidad lista de variables de red (NVL).

Nombre de la función	M238	M241	M251	M258 LMC058	XBTGC XBT GK XBT GT	ATV IMC
Lista de variables de red	No	Sí	Sí	Sí	Sí	Sí*
*ATV IMC admite NVL solo en tareas de ejecución libre.						

La figura muestra una red formada por 1 emisor y los 7 receptores recomendados como máximo:



Emisor A del controlador: emisor con la lista de variables globales (GVL) y controlador de receptor con las listas de variables globales de red (GNVL).

Receptor del controlador del 1 al 7: receptores (con GNVL) de A y controlador de emisor (GVL) solo para A.

Configuración del intercambio de variables de red

Descripción general

Para intercambiar variables de red entre un emisor y un receptor, deben estar disponibles un controlador de emisor y un controlador de receptor en **Dispositivos.** de SoMachine. A estos controladores se les han asignado las propiedades de red descritas a continuación.

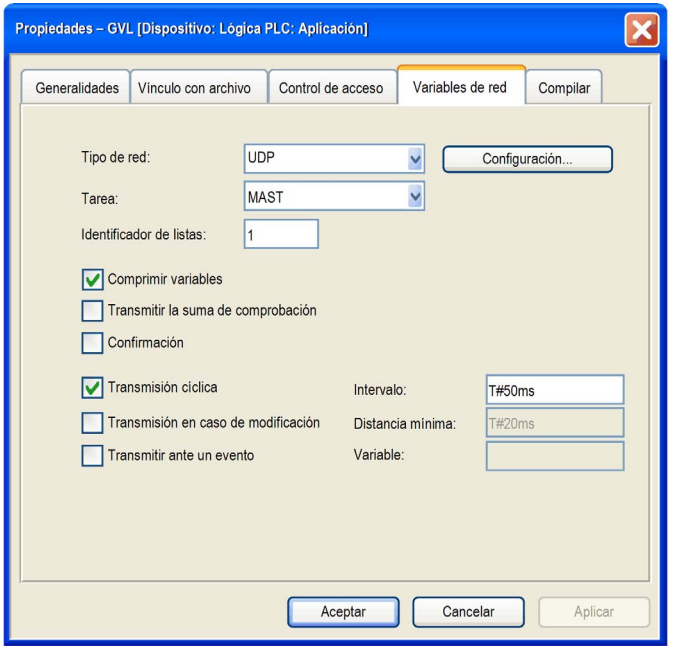
Siga los pasos siguientes para configurar la lista de variables de red:

Paso	Acción
1	Cree un controlador de emisor y otro de receptor en el árbol **Dispositivos**.
2	Cree un programa (POU) para el controlador de emisor y el de receptor.
3	Añada una tarea para el controlador de emisor y el de receptor. **NOTA:** Con el fin de mantener la transparencia del rendimiento, deberá definir la prioridad de tareas de la tarea NVL dedicada a un valor superior a 25 y regular las comunicaciones para evitar la saturación de la red innecesariamente.
4	Defina la lista de variables globales (GVL) del emisor.
5	Defina la lista de variables globales de red (GNVL) del receptor.

En el Anexo *(véase página 442)* se proporciona un ejemplo más detallado.

Lista de variables globales

Para crear la GVL del emisor, defina las propiedades de red siguientes en el cuadro de diálogo **GVL →Propiedades →Variables de red**:



Descripción de los parámetros

Parámetro	Valor predeterminado	Descripción
Tipo de red	**UDP**	Solo está disponible el tipo de red estándar **UDP**. Para cambiar los valores de **Broadcast Adr.** y **Puerto**, haga clic en el botón **Configuración...**.
Tarea	**MAST**	Seleccione la tarea que ha configurado anteriormente en la opción **Configuración de tareas** para ejecutar el código de la NVL. Para ayudar a mantener la transparencia del rendimiento, le recomendamos configurar un **Intervalo** de tiempo de ciclo de ≥50 ms para esta tarea. **NOTA:** Con el fin de mantener la transparencia del rendimiento, deberá definir la prioridad de tareas de la tarea NVL dedicada a un valor superior a 25 y regular las comunicaciones para evitar la saturación de la red innecesariamente.
Identificador de listas	**1**	Introduzca un número exclusivo para cada GVL de la red. Los receptores lo utilizan para identificar las listas de variables *(véase página 440)*.

Parámetro	Valor predeterminado	Descripción
Comprimir variables	activado	Si se activa esta opción, las variables se comprimen en paquetes (datagramas) para su transmisión. Si se desactiva esta opción, se transmite un paquete por variable.
Transmitir la suma de comprobación	desactivado	Active esta opción para añadir una suma de comprobación a cada paquete de variables durante la transmisión. Posteriormente, los receptores verificarán la suma de comprobación de cada paquete que reciban y rechazarán aquellos que no tengan una suma de comprobación coincidente. Se enviará una notificación con el `parámetro` NetVarError_CHECKSUM *(véase SoMachine, Configuración de variables de red, Guía de la biblioteca SE_NetVarUdp)*.
Confirmación	desactivado	Active esta opción para que el receptor envíe un mensaje de confirmación por cada paquete de datos que reciba. Se enviará una notificación con el `parámetro` NetVarError_ACKNOWLEDGE *(véase SoMachine, Configuración de variables de red, Guía de la biblioteca SE_NetVarUdp)* si el emisor no recibe este mensaje de confirmación por parte del receptor antes de enviar el próximo paquete de datos.
Transmisión cíclica ● **Intervalo**	activado	Seleccione esta opción para establecer una transmisión de datos cíclica en el **Intervalo** definido. Este **Intervalo** debe ser un múltiplo del tiempo de ciclo que haya definido en la tarea para ejecutar el código de la NVL a fin de lograr un tiempo de transmisión preciso de las variables de red.
Transmisión en caso de modificación ● **Distancia mínima**	desactivado ● **T#20ms**	Seleccione esta opción para transmitir variables cuando sus valores hayan cambiado. **NOTA:** Tras la primera descarga o el uso del comando **Reset frío** o **Reset caliente** en la modalidad Online, los controladores del receptor no se actualizan y conservan su último valor, mientras que el valor del controlador del emisor pasa a ser 0 (cero). El parámetro **Distancia mínima** define el intervalo de tiempo mínimo que debe transcurrir entre la transferencia de datos.
Transmisión controlada por eventos ● **Variable**	desactivado ● –	Seleccione esta opción para transmitir variables siempre que la **Variable** especificada sea TRUE. La variable se compara con todos los ciclos de la tarea para ejecutar el código de la NVL.

Descripción del botón **Configuración...**

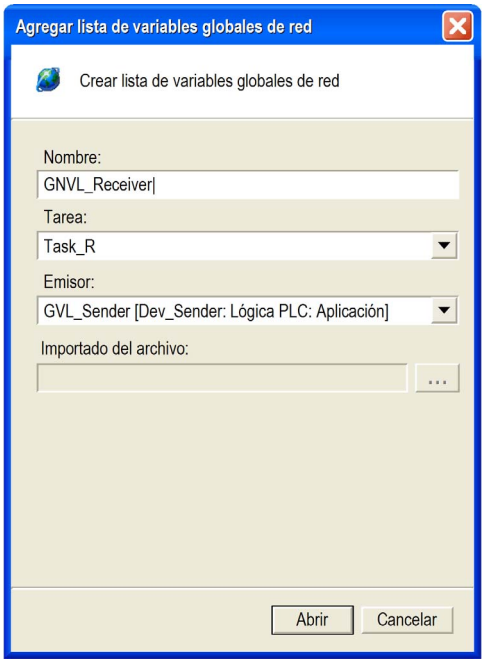
Parámetro	Valor predeterminado	Descripción
Puerto	1202	Introduzca un número de puerto exclusivo (≥ 1202) para cada emisor de la GVL.
Broadcast Adr.	255.255.255.255	Introduzca una dirección IP de difusión específica para su aplicación.

Lista de variables globales de red (GNVL)

Solo se puede añadir una lista de variables globales de red en el árbol **Dispositivos**. Esta lista define variables, que se especifican como variables de red en otro controlador dentro de la red.

Por ello, solo se puede añadir un objeto de GNVL a una aplicación si ya se ha creado en uno de los otros controladores de red una lista de variables globales (GVL) con propiedades de red (lista de variables de red). Estos controladores pueden estar en el mismo o en distintos proyectos.

Para crear la GNVL, defina los parámetros siguientes en el cuadro de diálogo **Añadir objeto** → **Lista de variables globales de red**:



Descripción de los parámetros

Parámetro	Valor predeterminado	Descripción
Nombre	**NVL**	Introduzca un nombre para la GNVL.
Tarea	Tarea definida en el nodo **Configuración de tareas** de esta **Aplicación**.	En la lista de tareas, seleccione la tarea que recibirá las tramas del emisor disponibles en el nodo **Configuración de tareas** del controlador de receptor.
Emisor	1 de las GVL actualmente disponibles en el proyecto.	Seleccione la GVL de emisor en la lista de todas las GVL de emisor con propiedades de red actualmente disponibles en el proyecto. En la lista, seleccione la entrada **Importado del archivo** para utilizar una GVL de otro proyecto. Esta acción activa el parámetro **Importado del archivo:** siguiente.
Importado del archivo:	–	Este parámetro solo está disponible si se selecciona la opción **Importado del archivo** para el parámetro **Emisor**. **...** abrirá una ventana estándar del Explorador de Windows en la que podrá buscar el archivo de exportación **.gvl* que haya creado a partir de una GVL en otro proyecto. Para obtener más información, consulte el párrafo *Cómo añadir una GNVL de un proyecto diferente*.

Cómo añadir una GNVL en el mismo proyecto

Al añadir una GNVL mediante el cuadro de diálogo **Añadir objeto**, se proporcionan todas las GVL adecuadas que se encuentran en el proyecto actual de la red actual para que se seleccionen en el cuadro de lista **Emisor**. Se deben importar las GVL de otros proyectos (consulte el párrafo *Cómo añadir una GNVL de un proyecto diferente*).

Con esta selección, cada GNVL del controlador actual (emisor) se vincula a una GVL específica de otro controlador (receptor).

Además, debe definir un nombre y una tarea, que será la responsable de gestionar las variables de red cuando se añada la GNVL.

Cómo añadir una GNVL de un proyecto diferente

En lugar de elegir directamente una GVL de emisor de otro controlador, también puede especificar un archivo de exportación de GVL que haya generado previamente a partir de la GVL mediante la opción **Vínculo con archivo**. Esto le permite utilizar una GVL definida en otro proyecto.

Para ello, seleccione la opción **Importado del archivo** para el parámetro **Emisor:** y especifique la ruta en el parámetro **Importado del archivo:**.

Más adelante, puede modificar la configuración en el cuadro de diálogo **Propiedades - GVL**.

Propiedades de la GNVL

Al hacer doble clic en una **GNVL** del árbol **Dispositivos**, se mostrará el contenido de la misma en un editor que aparecerá en la parte de la derecha. Sin embargo, no se puede editar el contenido de la GNVL, ya que es solo una referencia al contenido de la GVL correspondiente. En la parte superior del panel del editor se indica el nombre exacto y la ruta del emisor que contiene la GVL correspondiente junto con el tipo de protocolo de red utilizado. Si se modifica la GVL correspondiente, el contenido de la GNVL se actualiza en consecuencia.

Normas de la lista de variables de red (NVL)

Normas sobre la cantidad de datos

Debido a que existen algunas limitaciones de rendimiento, respete las normas siguientes:

Número	Norma
1	La transmisión de datos de una GVL (emisor) a una GNVL (receptor) no debe superar los 200 bytes.
2	El intercambio de datos entre diversas GVL (emisores) de un controlador y sus GNVL (receptores) asociadas no debe superar los 1.000 bytes de variables.

Normas sobre el número de datagramas

Para limitar el tiempo de ciclo máximo de tareas de NVL, respete las recomendaciones siguientes:

Número	Norma	Descripción
1	Limite el número de datagramas recibidos por ciclo a 20.	Cuando se supere el límite, el resto de datagramas se tratarán en el próximo ciclo. Aparecerá la notificación **Desborde de recibidos** en los datos de diagnóstico *(véase SoMachine, Configuración de variables de red, Guía de la biblioteca SE_NetVarUdp)*. Un datagrama puede contener hasta 256 bytes. Eso significa que no debe superar el límite de 5.120 bytes de datos recibidos por un receptor.
2	Limite el número de datagramas transmitidos por ciclo a 20.	Cuando se supere el límite, el resto de datagramas se tratarán en el próximo ciclo. Aparecerá la notificación **Desborde de transmitidos** en los datos de diagnóstico *(véase SoMachine, Configuración de variables de red, Guía de la biblioteca SE_NetVarUdp)*. Un datagrama puede contener hasta 256 bytes. Eso significa que no debe superar el límite de 5.120 bytes de datos transmitidos por un controlador de emisor.

Si el número de datagramas recibidos/transmitidos por ciclo supera el límite varias veces, puede suceder lo siguiente:

- Pérdida de datagramas de UDP (protocolo de datagramas de usuario)
- Intercambio de variables incoherente o inconsistente

Adapte los parámetros siguientes según sus necesidades:

- tiempo de ciclo del controlador de emisor
- tiempo de ciclo del controlador de receptor
- número de emisores en la red

AVISO

PÉRDIDA DE DATOS

Antes de poner el sistema en funcionamiento, compruebe minuciosamente que la transmisión y la recepción de datagramas de UDP de su aplicación sean correctas.

El incumplimiento de estas instrucciones puede causar daño al equipo.

Número máximo de GVL (emisores)

Defina un máximo de 7 GVL por controlador (emisor) para ayudar a mantener la transparencia del rendimiento.

Normas sobre los tiempos de ciclo de la tarea de las GVL (emisores) y de las GNVL (receptores)

Para evitar un desborde en la recepción, debe definir un tiempo de ciclo para la tarea que gestiona la transmisión de GVL que sea al menos dos veces superior al tiempo de ciclo de la tarea que gestiona la recepción de GNVL.

Normas sobre la protección del identificador de listas

La función NVL incluye una comprobación del identificador de listas:

El identificador de listas ayuda a evitar que una GVL (emisor) de dos controladores individuales con el mismo identificador de listas (consulte el cuadro de diálogo **GVL →Propiedades → Identificador de listas:**) envíe datagramas a la misma lista de variables globales de red (GNVL) de cualquier controlador. Si el **Identificador de listas** no es exclusivo, puede ocasionar una interrupción en el intercambio de variables.

AVISO
PÉRDIDA DE COMUNICACIÓN
Asegúrese de que solo una dirección IP utiliza el identificador de listas de la red.
El incumplimiento de estas instrucciones puede causar daño al equipo.

La función de comprobación del identificador de listas se implementa en el controlador de receptor.

Si una GNVL (receptor) detecta que dos direcciones IP distintas utilizan el mismo identificador de listas, el receptor deja inmediatamente de recibir datagramas.

Además, aparecerá una notificación en el bloque de funciones `NETVARGETDIAGINFO`. Se proporcionan las direcciones IP de los dos emisores en los parámetros de salida `dwDuplicateListIdIp1` y `dwDuplicateListIdIp2` de este bloque de funciones *(véase SoMachine, Configuración de variables de red, Guía de la biblioteca SE_NetVarUdp)*.

Con el bloque de funciones `NETVARRESETERROR` se restablecerán los errores que se detecten en la NVL y se volverá a iniciar la comunicación.

Estado de funcionamiento del emisor y del receptor

Estado de funcionamiento

Estado de funcionamiento del...		Comportamiento de las variables de red
Emisor	**Receptor**	
RUN	RUN	Las variables de red se intercambian entre el emisor y el receptor.
STOP	RUN	El emisor deja de enviar variables al receptor. No se intercambian variables de red entre el emisor y los receptores.
RUN	STOP	El receptor no está procesando variables de red del emisor. Cuando el receptor vuelve al estado RUN, este vuelve a procesar las variables de red.
STOP	STOP	No se intercambian variables.

NOTA: Se detectan varios errores de inicialización de la comunicación (`NetVarError_INITCOMM`), cuando cambia el estado de funcionamiento del emisor de STOP a RUN.

Eventos de la tarea que gestiona la NVL

Si se dieran los eventos siguientes en la tarea que gestiona la NVL, el comportamiento que se espera que tenga la NVL es el mismo que si el controlador estuviera en un estado STOP en la matriz anterior:

- Se produce una excepción en la aplicación que da lugar a que la tarea se suspenda.
- Se alcanza un punto de interrupción o se procesa un ciclo individual en la tarea.

Ejemplo de

Descripción general

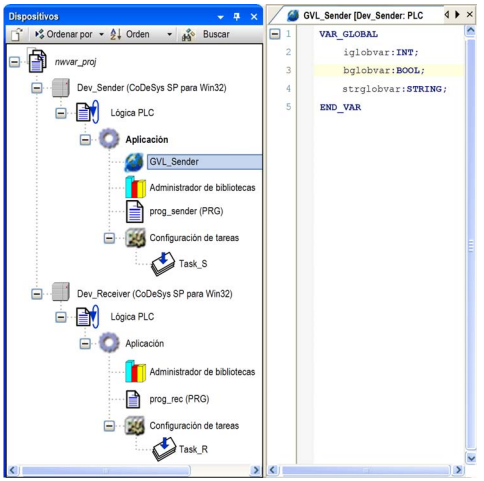
En el ejemplo siguiente, se establece un intercambio simple de variables de red. Se crea una lista de variables globales (GVL) en el controlador de emisor. Se crea la lista de variables globales de red (GNVL) correspondiente en el controlador de receptor.

Lleve a cabo las preparaciones siguientes en un proyecto estándar, en el que estén disponibles un controlador de emisor **Dev_Sender** y un controlador de receptor **Dev_Receiver** en el árbol **Dispositivos**:

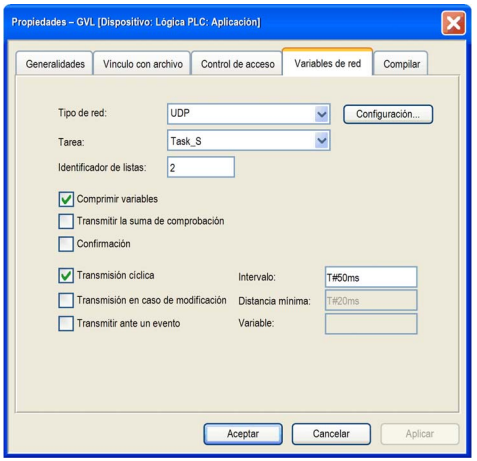
- Cree un POU (programa) **prog_sender** debajo del nodo **Aplicación** de **Dev_Sender**.
- Debajo del nodo **Configuración de tareas** de esta aplicación, añada la tarea **Task_S** que invoca **prog_sender**.
- Cree un POU (programa) **prog_rec** debajo del nodo **Aplicación** de **Dev_Receiver**.
- Debajo del nodo **Configuración de tareas** de esta aplicación, añada la tarea **Task_R** que invoca **prog_rec**.
  NOTA: Se deben configurar los 2 controladores en la misma subred de la red Ethernet.

Definición de la GVL del emisor

Paso 1: Defina una lista de variables globales en el controlador de emisor:

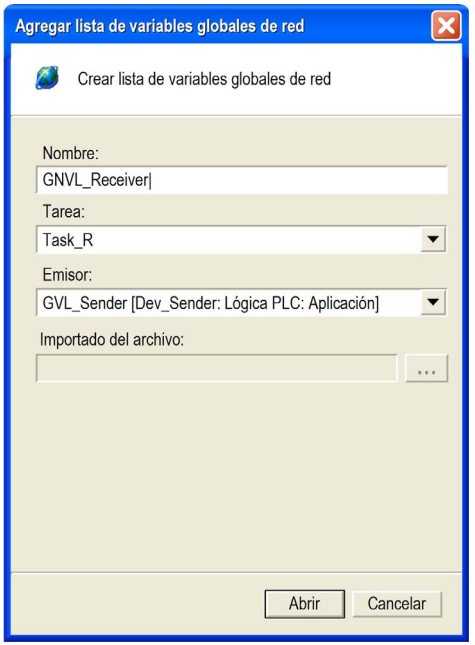
Paso	Acción	Comentario
1	En el panel **Dispositivos**, haga clic con el botón derecho en el nodo **Aplicación** del controlador **Dev_Sender** y seleccione los comandos **Agregar objeto →Lista de variables globales...**.	Aparecerá el cuadro de diálogo **Agregar lista de variables globales**.
2	Introduzca el **Nombre** `GVL_Sender` y haga clic en **Abrir** para crear una nueva lista de variables globales.	El nodo **GVL_Sender** aparecerá debajo del nodo **Aplicación** en el panel **Dispositivos** y el editor se abrirá en la parte derecha.
3	En el editor situado en la parte derecha, introduzca las definiciones de variables siguientes: `VAR_GLOBAL iglobvar:INT; bglobvar:BOOL; strglobvar:STRING; END_VAR` 	–

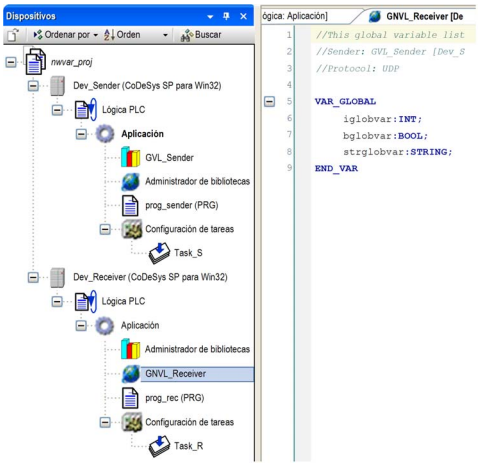
Paso 2: Defina las propiedades de la red de la GVL de emisor:

Paso	Acción	Comentario
1	En el panel **Dispositivos**, haga clic con el botón derecho en el nodo **GVL_Sender** y seleccione el comando **Propiedades...**.	Aparecerá el cuadro de diálogo **Propiedades - GVL_Sender**.
2	Abra la ficha **Variables de red** y configure los parámetros, tal como se muestra en el gráfico: 	–
3	Haga clic en **Aceptar**.	Se cerrará el cuadro de diálogo y se establecerán las propiedades de la red de la GVL.

Definición de la GNVL del receptor

Paso 1: Defina una lista de variables globales de red en el controlador de receptor:

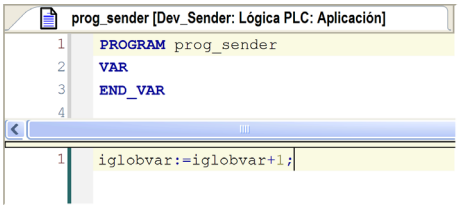
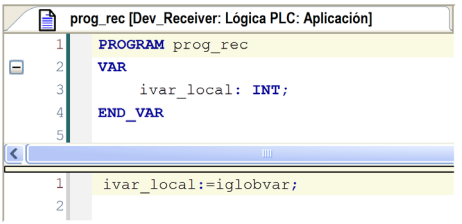
Paso	Acción	Comentario
1	En el panel **Dispositivos**, haga clic con el botón derecho en el nodo **Aplicación** del controlador **Dev_Receiver** y seleccione los comandos **Agregar objeto →Lista de variables globales de red...**.	Aparecerá el cuadro de diálogo **Agregar lista de variables globales de red**.
2	Configure los parámetros tal como se muestra en el gráfico. 	Esta lista de variables globales de red es la homóloga de la GVL definida para el controlador de emisor.

Paso	Acción	Comentario
3	Haga clic en **Abrir**.	Se cerrará el cuadro de diálogo y aparecerá **GNVL_Receiver** debajo del nodo **Aplicación** del controlador **Dev_Receiver**:  Esta GNVL contiene automáticamente las mismas declaraciones de variables que **GVL_Sender**.

Paso 2: Compruebe y/o modifique la configuración de red de la GNVL:

Paso	Acción	Comentario
1	En el panel **Dispositivos**, haga clic con el botón derecho en el nodo **GNVL_Receiver** y seleccione el comando **Propiedades...**.	Aparecerá el cuadro de diálogo **Propiedades - GNVL_Receiver**.
2	Abra la ficha **Configuración de red**.	–

Paso 3: Pruebe el intercambio de las variables de red en la modalidad online:

Paso	Acción	Comentario
1	Debajo del nodo **Aplicación** del controlador **Dev_Sender**, haga doble clic en la POU **prog_sender**.	Se abrirá el editor para **prog_sender** en la parte de la derecha.
2	Introduzca el código siguiente para la variable `iglobvar`: 	–
3	Debajo del nodo **Aplicación** del controlador **Dev_Receiver**, haga doble clic en la POU **prog_rec**.	Se abrirá el editor para **prog_rec** en la parte de la derecha.
4	Introduzca el código siguiente para la variable `ivar_local`: 	–
5	Inicie sesión con las aplicaciones del emisor y del receptor de la misma red e inícielas.	La variable `ivar_local` del receptor obtiene los valores de `iglobvar` tal como se muestra actualmente en el emisor.

Compatibilidad

Introducción

A pesar de que los controladores funcionan con aplicaciones de versiones distintas del sistema de programación (por ejemplo, V2.3 y V3.x), es posible establecer una comunicación mediante las variables de red.

Sin embargo, la diferencia de los formatos de los archivos de exportación entre versiones (*.exp en comparación con *.gvl) hace que sea imposible importar y exportar simplemente estos archivos entre proyectos.

Si se configura una lista de variables globales de red (GNVL) de lectura en la versión más reciente (por ejemplo, V3.x), el emisor de la versión más reciente (por ejemplo, V3.x) debe proporcionar la configuración de los parámetros de red necesarios. Los archivos de exportación **.exp* creados por una versión anterior (por ejemplo, V2.3) a partir de un emisor no contienen esta información.

En los párrafos siguientes se proporciona una solución para intercambiar variables de red entre aplicaciones de distintas versiones de sistemas de programación.

Actualización de la lista de variables globales de red

Para intercambiar variables de red entre aplicaciones de distintas versiones de sistemas de programación (por ejemplo, V2.3 y V3.x), actualice la lista de variables globales de red siguiendo los pasos siguientes:

Paso	Acción	Comentario
1	Vuelva a crear la lista de variables de red (NVL) que ya está disponible en la versión anterior (V2.3) en la versión más reciente (V3.x).	Para ello, añada una lista de variables globales (GVL) con propiedades de red, que contenga las mismas declaraciones de variables que la NVL de la versión anterior (V2.3).
2	Exporte la nueva GVL a un archivo **.exp* mediante la ficha **Vínculo con archivo**.	**NOTA:** Active la opción **Excluir de la compilación** en la ficha **Compilar** para conservar la GVL en el proyecto sin que se incluyan eventos anteriores a la compilación y nombres ambiguos. En caso de que sea necesario modificar la GVL, desactive la opción para volver a crear el archivo **.exp*.
3	Vuelva a importar la lista.	Para ello, cree una lista de variables globales de red (GNVL) nueva mediante el archivo **.exp* previamente generado para crear una lista de receptores configurados de manera apropiada.

En el ejemplo siguiente, se muestran estos pasos.

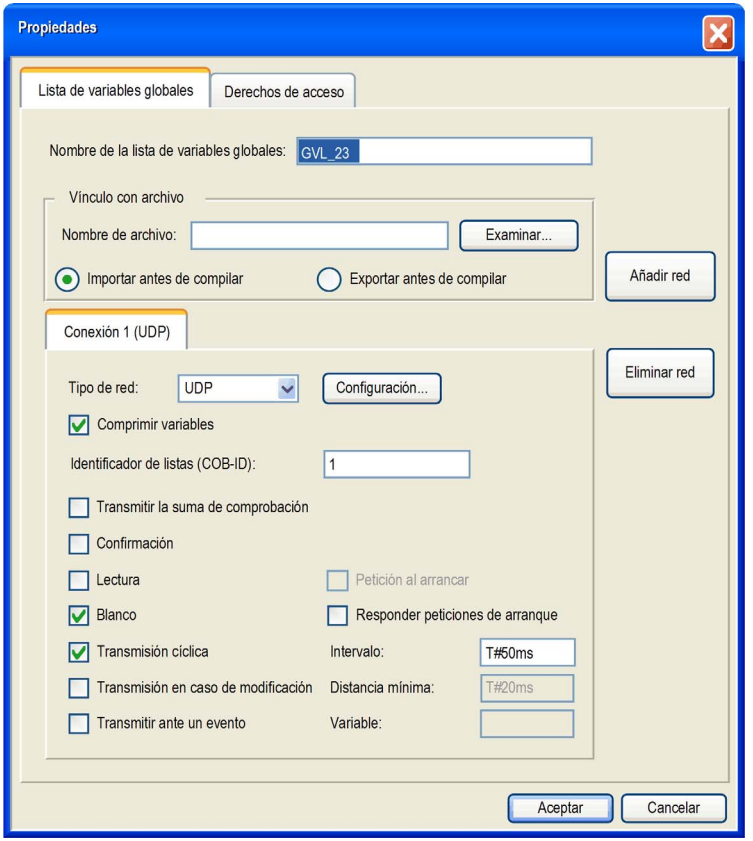
Ejemplo

En este ejemplo, la variable `trans23`, definida en una aplicación V2.3, se pone a disposición de la versión más reciente (V3.x).

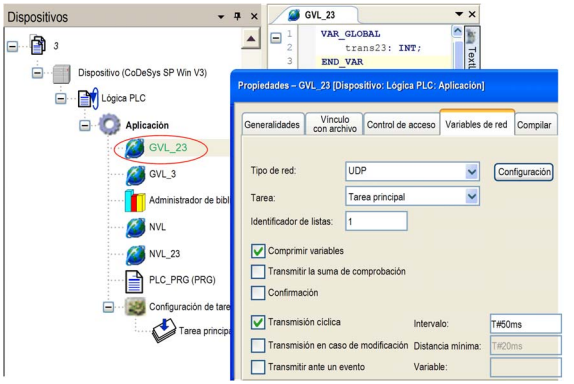
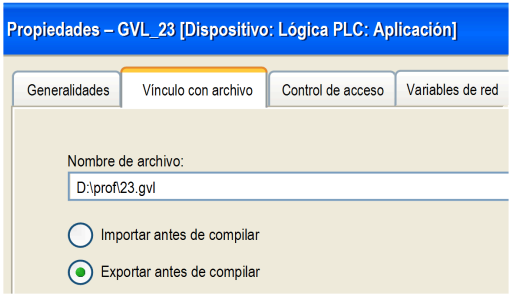
Se definen las condiciones siguientes:

Condición	Descripción

1	En la versión anterior del sistema de programación (V2.3), el proyecto *23.pro* contiene una lista de variables globales **GVL_23** con la declaración siguiente:
	`VAR_GLOBAL`
	`trans23:INT;`
	`END_VAR`

2	Las propiedades de red de **GVL_23** se configuran de la siguiente manera:
	 
	
	NOTA: La exportación de **GVL_23** genera un archivo **.exp* que contiene únicamente la declaración de variables siguiente:
	`VAR_GLOBAL`
	`trans23:INT;`
	`END_VAR`
	El archivo **.exp* no contiene ajustes de configuración.

En la tabla siguiente se muestran los pasos siguientes que tienen que ejecutarse para volver a crear **GVL_23** en la versión más reciente (V3.x):

Paso	Acción	Comentario
1	Añada a una aplicación un objeto GVL llamado **GVL_23**.	–
2	Establezca las propiedades de red, tal como se definen en el proyecto *23.pro*.	
3	En la ficha **Vínculo con archivo**, configure un archivo de exportación de destino *23.gvl*.	
4	En la ficha **Compilar**, establezca la opción **Excluir de la compilación**.	Este ajuste le permite conservar el archivo en el disco para futuras modificaciones. 

Paso	Acción	Comentario
5	Compile el proyecto.	Se generará el archivo *23.gvl*, que contiene ajustes de configuración y de variables: 
6	Añada un objeto GNVL en el proyecto V3.x del archivo de exportación *23.gvl* (con el comando **Importado del archivo:**).	Esto sirve para leer la variable `trans23` del controlador del sistema de programación anterior (V.2.3). Si se ejecutan en la red tanto el proyecto de la versión anterior (V2.3) como la aplicación de la versión más reciente (V3.x), la aplicación de la versión más reciente (V3.x) puede leer la variable `trans23` del proyecto *23.pro*.

Capítulo 18
Editor de tarea

Contenido de este capítulo

Este capítulo contiene los siguiente apartados:

Información sobre la configuración de tareas

Descripción general

La configuración de tareas define una o varias tareas para controlar el procesamiento de un programa de aplicación. Por lo tanto, la configuración de tareas es un recurso esencial para una aplicación y debe estar disponible en el árbol **Aplicaciones**.

Descripción del árbol de configuración de tareas

En la posición más alta de un árbol de configuración de tareas, se encuentra la entrada **Configuración de tareas**  . A continuación se encuentran las tareas definidas, cada una representada por su nombre. Las llamadas a POU de cada tarea específica se muestran en el árbol de configuración de tareas.

Puede editar el árbol de tareas (añadir, copiar, pegar o eliminar tareas) con los comandos apropiados que pueden utilizarse en el árbol **Aplicaciones**. Por ejemplo, para añadir una nueva tarea, seleccione el nodo **Configuración de tareas**, haga clic en el signo más de color verde y ejecute el comando **Tarea...** Como alternativa, puede hacer clic con el botón derecho en el nodo **Configuración de tareas** y ejecutar el comando **Agregar objeto →Tarea...**

Configure las tareas concretas en el editor de tarea *(véase página 458)*, que, además, proporciona una vista de supervisión en la modalidad online. Las opciones disponibles para la configuración de tareas dependen de la plataforma del controlador.

Configuración de tareas en el árbol **Aplicaciones**

Tareas

Una tarea *(véase página 458)* se utiliza para controlar el procesamiento de un programa IEC. Se define por un nombre, una prioridad y por un tipo que determina en qué condición se desencadenará el inicio de la tarea. Puede definir esta condición por un tiempo (cíclico, de ejecución libre) o por un evento interno o externo que desencadenará la tarea; por ejemplo, el flanco ascendente de una variable de proyecto global o un evento de interrupción del controlador.

Para cada tarea, puede especificar una serie de POU de programa que será iniciada por la tarea. Si la tarea se ejecuta en el presente ciclo, estos programas se procesarán durante 1 ciclo.

La combinación de prioridad y condición determinará en qué orden cronológico *(véase página 462)* se ejecutarán las tareas.

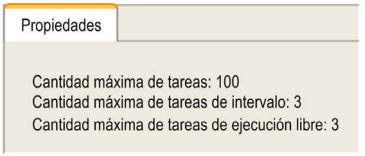
Para cada tarea, puede configurar un control de tiempo (watchdog). La configuración posible depende de la plataforma del controlador en cuestión.

Ficha Propiedades

Descripción general

Cuando se selecciona el nodo **Configuración de tareas** *(véase página 454)*, la ficha **Propiedades** se abre en la vista del editor de tareas.

Configuración de tareas, ficha **Propiedades**, ejemplo



Se visualizará información sobre la configuración de tareas actual según la proporcione el controlador, como por ejemplo el número máximo de tareas permitido por tipo de tarea.

Ficha Supervisión

Descripción general

Si se admite en el sistema de destino, se permite la funcionalidad de supervisión. Se trata de un análisis dinámico del tiempo de ejecución, el número de llamadas y la cobertura de código de las POU, que se controla mediante una tarea. En la modalidad online, se puede supervisar el procesamiento de tareas.

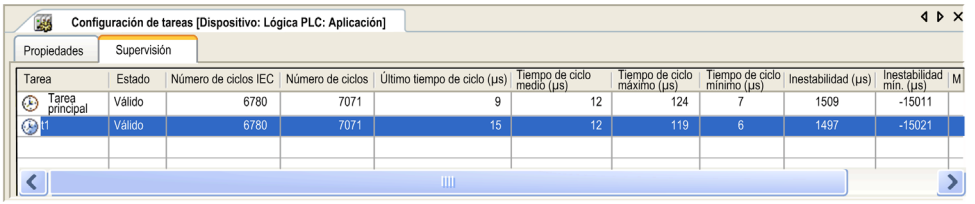
Vista online del editor de tarea

Cuando selecciona el nodo superior del árbol **Configuración de tarea**, además de la ficha **Propiedades** *(véase página 455)*, la ficha **Supervisión** está disponible. En la modalidad online, muestra el estado y algunas estadísticas actuales sobre los ciclos y tiempos de ciclo en una vista de tabla. El intervalo de actualización de los valores es el mismo que el utilizado para la supervisión de los valores del controlador.

Descripción de los elementos

Cuando se selecciona el nodo superior del árbol **Configuración de tareas**, además del cuadro de diálogo **Propiedades** *(véase página 455)* está disponible en otra ficha el cuadro de diálogo **Supervisión**. En la modalidad online, muestra el estado y algunas estadísticas actuales sobre los ciclos y tiempos de ciclo en una vista de tabla. El intervalo de actualización de los valores es el mismo que el utilizado para la supervisión de los valores del controlador.

Configuración de tareas, **Supervisión**



Para cada tarea se muestra la siguiente información en una línea:

Tarea	Nombre de la tarea definido en la **Configuración de tareas**.
Estado	Entradas posibles: ● **No creado**: no se ha iniciado desde la última actualización; especialmente utilizado para tareas de eventos. ● **Creado**: la tarea se conoce en el sistema de tiempo de ejecución, pero aún no se ha configurado para su funcionamiento. ● **Válido**: la tarea funciona de forma normal. ● **Excepción**: la tarea tiene una excepción.
Número de ciclos IEC	Número de ciclos de ejecución desde que se inició la aplicación; 0 si la función no es compatible con el sistema de destino.

Número de ciclos	Número de ciclos que ya están en ejecución (dependiendo del sistema de destino, puede ser igual al número de ciclos IEC, o mayor si los ciclos se cuentan aun cuando la aplicación no esté en ejecución).
Último tiempo de ciclo (µs)	Último tiempo de ejecución medido en µs.
Tiempo de ciclo medio (µs)	Promedio de tiempo de ejecución de todos los ciclos en µs.
Tiempo de ciclo máximo (µs)	Tiempo de ejecución máximo de todos los ciclos medido en µs.
Tiempo de ciclo mínimo (µs)	Tiempo de ejecución mínimo de todos los ciclos medido en µs.
Inestabilidad (µs)	Tiempo de la última inestabilidad* en µs.
Inestabilidad mín. (µs)	Tiempo mínimo de inestabilidad* medido en µs.
Inestabilidad máx. (µs)	Tiempo máximo de inestabilidad* medido en µs.
* inestabilidad: Tiempo que pasa después del inicio de la tarea hasta que el sistema operativo indica que está en ejecución.	

Para restablecer los valores a 0 para una tarea, coloque el cursor en el campo de nombre de tarea y ejecute el comando **Restablecer**, disponible en el menú contextual.

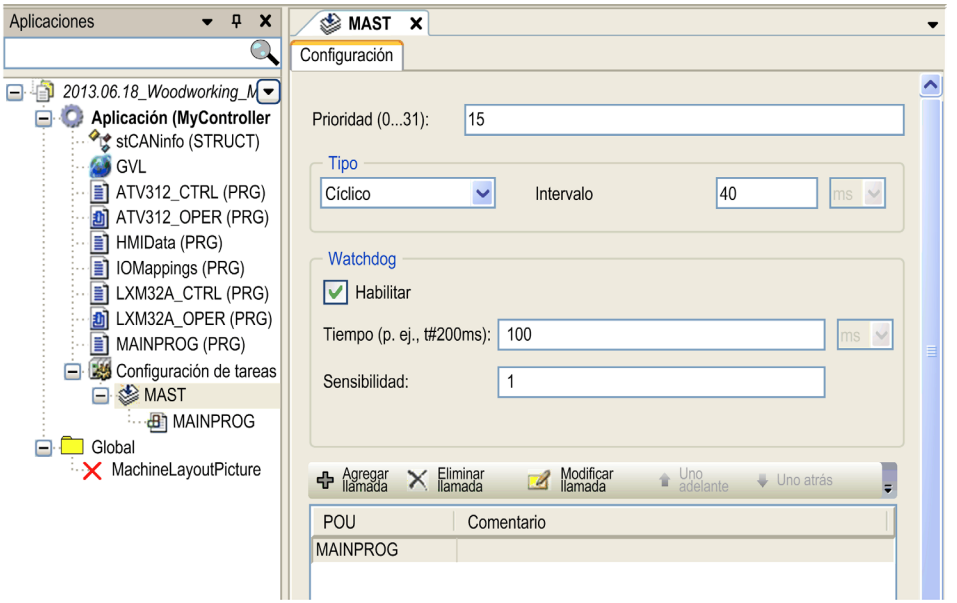
Configuración de una tarea específica

Descripción general

Al insertar una tarea en el nodo **Configuración de tareas** del árbol **Aplicaciones**, la vista del editor de tareas para definir la configuración de tareas se abre con la ficha **Configuración**.

También se abre si hace doble clic en una tarea disponible (por ejemplo, **MAST**) para modificar la configuración de la tarea.

Ficha **Configuración** de una tarea



NOTA: Puede modificar el nombre de la tarea editando la entrada respectiva en el árbol **Aplicaciones**.

Inserte los atributos que desee.

Prioridad	
Prioridad (0...31)	Un número de 0 a 31; 0 es la prioridad más alta y 31 la más baja.

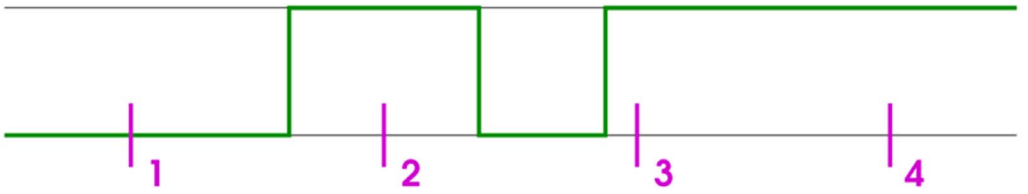
Tipo	
La lista de selección ofrece los tipos de tarea siguientes:	
Cíclico	La tarea se procesará cíclicamente según la definición del tiempo (tiempo de ciclo de tarea) especificada en el campo **Intervalo** (véase a continuación).

Tipo	
Ejecución libre	La tarea se procesará en cuanto se inicie el programa y, al finalizar una ejecución, se reiniciará automáticamente en un bucle continuo. No hay un tiempo de ciclo definido.
Estado	La tarea se iniciará si la variable definida en el campo **Evento** es TRUE. **NOTA:** Esta función no está disponible para todos los controladores admitidos. Para obtener más información, consulte la guía de programación del controlador.
Evento	La tarea se iniciará en cuanto la variable definida en el campo **Evento** obtenga un flanco ascendente.
Evento externo	La tarea se iniciará en cuanto se produzca el evento del sistema, que se define en el campo **Evento**. Los eventos que se admitirán y se ofrecerán en la lista de selección dependen del destino. No se deben mezclar con eventos del sistema.

Diferencia entre estado y evento

Si el evento especificado es TRUE, cumple la condición de inicio de una tarea iniciada por estado, mientras que una tarea iniciada por un evento requiere el cambio del evento de FALSE a TRUE. Si el evento cambia demasiado rápido de TRUE a FALSE y otra vez a TRUE, es posible que no se detecte y que, por tanto, no se inicie la tarea **Evento**.

El ejemplo siguiente ilustra el comportamiento resultante de la tarea en reacción a un evento (línea verde):



En los puntos de muestreo 1...4, tareas de distintos tipos muestran una reacción diferente:

Comportamiento en el punto:	1	2	3	4
Estado	no iniciar	iniciar	iniciar	iniciar
Evento	no iniciar	iniciar	No iniciar porque el evento ha cambiado demasiado rápido de TRUE a FALSE y otra vez a TRUE	no iniciar

Entradas obligatorias según la elección de la tarea

Entrada	Descripción
Intervalo Tiempo de ciclo de tarea	Obligatorio para el tipo **Cíclico**. Tiempo (en milisegundos [ms]) tras el que debe reiniciarse la tarea. **NOTA:** Al configurar el tiempo de ciclo de tarea, tenga en cuenta el sistema de bus utilizado en la actualidad. Por ejemplo, en un bus CAN, puede establecer la **Tarea de ciclo de bus** en la ficha **Asignación de E/S CANopen**. Debe corresponderse con la velocidad de transmisión establecida actualmente y la cantidad de tramas utilizadas en el bus. Además, los tiempos establecidos para heartbeat, nodeguarding y sync deben ser siempre múltiplos del tiempo de ciclo de tarea. De lo contrario, pueden perderse tramas CAN. Para obtener más información, consulte la sección sobre el *editor de dispositivos* de la ayuda online SoMachine.
Evento	Obligatorio para el tipo **Evento** o activado por un **Evento externo**. Una variable booleana global que activará el inicio de la tarea en cuanto se detecte un flanco ascendente. Utilice el botón **...** o el **asistente Accesibilidad** para obtener una lista de todas las variables de eventos globales disponibles. **NOTA:** Si el evento que inicia una tarea se origina en una entrada, debe haber como mínimo una tarea que no se inicie por eventos. De lo contrario, las E/S nunca se actualizarán y la tarea nunca se iniciará.

Configuración de watchdogs

Para cada tarea, puede configurar un control de tiempo (watchdog).

La configuración predeterminada del watchdog depende del controlador.

Cuando la opción **Habilitar** está activada (tiene la marca de verificación), el watchdog está habilitado. Cuando el watchdog de tareas está habilitado, se detecta un error de **Excepción** si el tiempo de ejecución de la tarea supera el límite de tiempo de tarea definido (**Tiempo**) relativo a la **Sensibilidad** definida. Si la opción **Actualizar E/S en parada** está habilitada en el cuadro de diálogo de configuración del controlador, las salidas se configurarán con los valores predeterminados predefinidos según la plataforma de controlador específica.

Tiempo (por ejemplo t#200 ms)	Define el tiempo de ejecución máximo permitido para una tarea. Cuando una tarea tarda más tiempo del permitido, el controlador notificará una excepción de watchdog de tareas.
Sensibilidad	Define el número de excepciones de watchdog de tareas que debe producirse antes de que el controlador detecte un error de aplicación.

Para obtener más información, consulte el capítulo *System and Task Watchdog* de la guía de programación de su controlador.

POU

Las POU controladas por la tarea se enumeran aquí en una tabla con el nombre de la POU y un **Comentario** opcional. Encima de la tabla hay comandos para la edición:

- Para definir una POU nueva, abra el cuadro de diálogo **Accesibilidad** mediante el comando **Agregar llamada**. Seleccione uno de los programas disponibles en el proyecto. También puede añadir POU de tipo programa a la lista mediante el método de arrastrar y soltar desde el árbol **Aplicaciones**.
- Para sustituir una llamada de programa por otra, seleccione la entrada en la tabla, abra **Accesibilidad** mediante el comando **Modificar llamada** y seleccione otro programa.
- Para eliminar una llamada, selecciónela en la tabla y utilice el comando **Eliminar llamada**.
- El comando **Abrir POU** abre el programa seleccionado actualmente en el editor correspondiente.

La secuencia de las llamadas POU enumeradas de arriba abajo determina la secuencia de ejecución en modalidad online. Puede desplazar la entrada seleccionada dentro de la lista por medio de los comandos **Uno adelante** y **Uno atrás**.

Funciones de cadena

Las funciones de cadena de la biblioteca estándar (consulte la sección *Bibliotecas CoDeSys / Biblioteca estándar* de la ayuda online de SoMachine) no están programadas para ser reentrantes.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

No utilice la misma función de cadena en varias tareas, ya que puede causar fallos de programa al sobrescribir.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

Para ayudar a superar esta limitación, puede utilizar la *sincronización de tareas IEC* de CAA_Types.library (consulte la sección *Bibliotecas CoDeSys / CAA_Types.library* de la ayuda online de SoMachine).

Procesamiento de tareas en modalidad online

¿Qué tarea se está procesando?

Para la ejecución de las tareas definidas en la **Configuración de tareas**, se aplican las siguientes reglas:

- Se ejecuta esa tarea, cuya condición se ha cumplido. Esto es, si ha expirado el tiempo establecido, o después de que su variable de condición (evento) muestre un flanco ascendente.
- Si varias tareas tienen un requisito válido, entonces se ejecutará la tarea con la máxima **prioridad**.
- Si varias tareas tienen condiciones válidas y prioridades equivalentes, se ejecutará primero la tarea con el tiempo de espera más largo.
- El procesamiento de llamadas de la POU (de tipo programa) se llevará a cabo según su orden (descendente) en el **Editor de tarea**. Si se llama a una POU que está disponible con el mismo nombre, tanto en el árbol **Aplicaciones** asignado a la aplicación como en una biblioteca o de forma global en el proyecto en el nodo **Global** del árbol **Aplicaciones**, se ejecutará la que esté declarada justo debajo en el árbol **Aplicaciones**.

Capítulo 19
Editor de lista de supervisión

Contenido de este capítulo

Este capítulo contiene los siguiente apartados:

Editor de vista de supervisión/lista de supervisión

Descripción general

Una lista de supervisión es un conjunto de variables de proyecto definido por el usuario. Se muestran en la vista de supervisión para supervisar *(véase página 467)* los valores de una tabla. Además, en la vista de supervisión se pueden escribir y forzar las variables.

Para abrir una vista de supervisión, utilice el submenú del comando **Supervisar** (de forma predeterminada, se encuentra en el menú **Ver**). Proporciona un editor para crear listas de supervisión *(véase página 465)*.

De forma predeterminada, se pueden configurar hasta 4 listas de supervisión individuales en las vistas de supervisión **Supervisar 1**, **Supervisar 2**, **Supervisar 3** y **Supervisar 4**. La vista **Ver todos los forzados** en modalidad online se llena *(véase página 468)* automáticamente con todos los valores actualmente forzados de la aplicación activa.

Creación de una lista de supervisión

Descripción general

Para configurar una lista de supervisión **Supervisar<n>** en la vista **Supervisar**, haga clic en un campo de la columna **Expresión** y pulse la barra espaciadora para editar la columna **Expresión**. Introduzca la ruta completa para la expresión de supervisión deseada. El **asistente Accesibilidad** se encuentra disponible a través del botón **...**

Sintaxis para la expresión de supervisión

<nombre de dispositivo>.<nombre de aplicación>.<nombre de objeto>.<nombre de variable>

Ejemplo

Ejemplo:

```
Dev1.App1.PLC_PRG.ivar
```

El tipo de la variable se indica mediante un icono antes del nombre de variable:

Icono	Variable
	Entrada
	Salida
	Normal

Después de especificar la variable en la columna **Expresión**, el tipo de datos correspondiente se añade automáticamente en la columna **Tipo**. Si se ha añadido un comentario a la declaración de una variable, se añade en la columna **Comentario**.

La columna **Valor** muestra el valor actual de la expresión en la modalidad online *(véase página 467)*.

Para preparar un valor de una variable, haga clic en el campo asignado de la columna **Valor preparado** y escriba directamente el valor deseado.

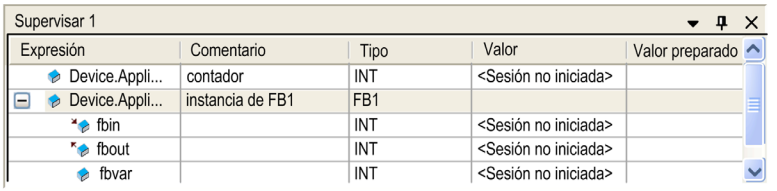
En caso de una variable booleana, el método es aún más fácil: puede cambiar los valores de preparación booleanos utilizando RETORNO o ESPACIO según el siguiente orden:

Si el valor es TRUE, los pasos de preparación son FALSE -> TRUE -> ninguna entrada; si no, si el valor es FALSE, los pasos de preparación son TRUE-> FALSE -> ninguna entrada.

Haga lo mismo con otras expresiones o variables deseadas en otras líneas. Vea un ejemplo en la siguiente imagen, que muestra la vista de supervisión en la modalidad offline: contiene expresiones de objetos `PLC_PRG` y `Prog_St`.

Tenga en cuenta que en caso de una variable estructurada, como la instancia de bloque de funciones, los componentes de la instancia específica se añaden automáticamente cuando se introduce el nombre de la instancia (véalo en el ejemplo: `Dev1.App1.PLC_PRG.fbinst`). Haga clic en el signo más/menos para mostrarlos u ocultarlos en una estructura.

Ejemplo, vista de supervisión en la modalidad offline



En la modalidad online *(véase página 467)*, puede utilizar la lista para la supervisión de los valores de las variables.

NOTA: En la modalidad online puede añadir expresiones a la lista de supervisión mediante el uso del comando **Agregar a la lista de supervisión**.

Lista de supervisión en modalidad online

Supervisión

Una lista de supervisión *(véase página 464)* (**Supervisar<n>**) en modalidad online muestra el valor actual de una variable en la columna **Valor**. Este es el valor que tiene la variable entre 2 ciclos de tarea.

También se muestra una dirección IEC directa posiblemente asignada o comentarios. Los componentes de la vista corresponden a los de la vista online del editor de declaraciones *(véase página 419)*.

Consulte el capítulo *Creación de una lista de supervisión (véase página 465)*, donde se explica cómo configurar una lista de supervisión y cómo manejar la estructura en caso de variables estructuradas.

Vista **Supervisar** en modalidad online

Supervisar 1

Expresión	Tipo	Valor	Valor preparado	Dirección	Comentario
myDev.Application.PLC_PRG.myStruct	TestStruct				Comprobar dirección definida en "TestStruct"
nTest1	INT	11876		%MW0	
nTest2	INT	0		%MW2	
myDev.Application.PLC_PRG.bVar	BOOL	TRUE		%QX0.3	
myDev.Application.PLC_PRG.fbinst	FB1				Instancia de bloque de funciones FB1
fbin	INT	0			
fbout	INT	0			
fbvar	INT	0			

Mensajes – Total 0 error(es), 0 advertencia(s), 0 mensaje(s) | Supervisar 1

NOTA: En modalidad online, puede añadir expresiones a la lista de supervisión mediante el comando **Agregar a la lista de supervisión**.

Escribir valores y Forzar valores

En la columna **Valor preparado**, puede introducir el valor deseado que se escribirá o forzará en la expresión correspondiente en el controlador mediante el comando **Escribir valores** o **Forzar valores**. Consulte las descripciones de los comandos **Escribir** y **Forzar**, que también se pueden utilizar en otras vistas de supervisión (por ejemplo, el editor de declaraciones).

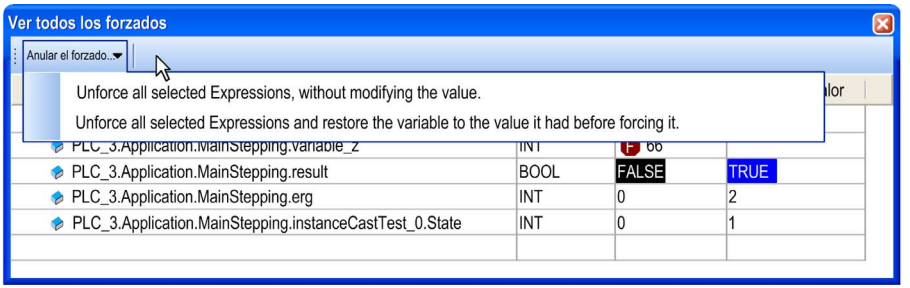
Ver todos los forzados

Esta es una vista de lista de supervisión especial, que en modalidad online se rellena automáticamente con todos los valores actuales forzados de la aplicación activa. Se mostrará cada **Expresión**, **Tipo**, **Valor** y **Valor preparado**, como en la vista online de una lista **Supervisar<n>**.

Puede anular el forzado de los valores mediante uno de los siguientes comandos, disponibles con el botón **Anular el forzado...**:

- **Unforce all selected Expressions, without modifying the value.**
- **Unforce all selected Expressions and restore the variable to the value it had before forcing it.**

Cuadro de diálogo **Ver todos los forzados**



Capítulo 20
Herramientas en los editores de lógica

Contenido de este capítulo

Este capítulo contiene los siguiente apartados:

Buscador de funciones y bloques de funciones

Descripción general

SoMachine proporciona el buscador FFB (funciones y bloques de funciones) que le ayuda a encontrar una función o un bloque de funciones específico, incluso si no sabe su nombre exacto.

Puede usar el buscador de funciones y bloques de funciones en los siguientes lenguajes de programación que permiten insertar bloques de funciones:

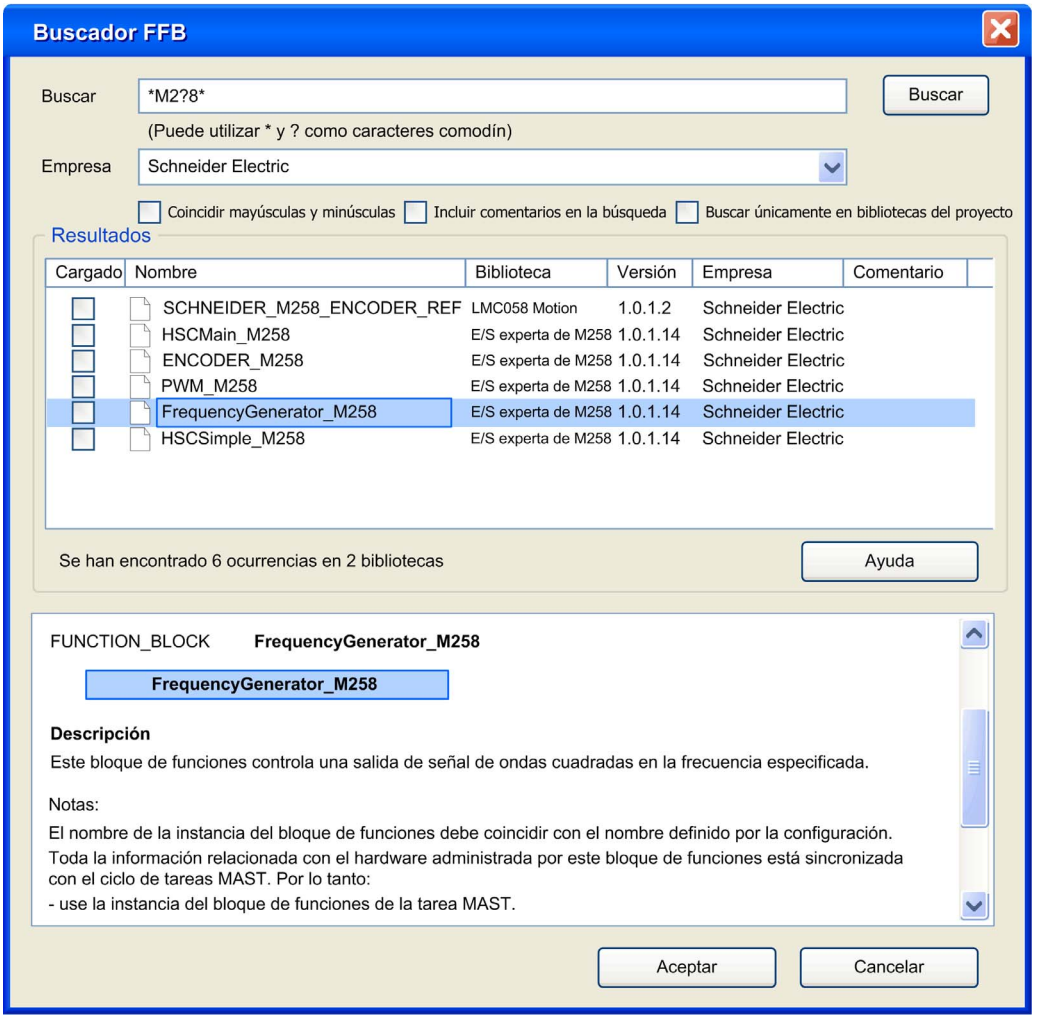
- CFC
- LD
- IL
- FBD
- ST

Cómo encontrar una función o un bloque de funciones con el buscador FFB

Cuando vaya a crear un código de programación en SoMachine Logic Builder, vaya al lugar donde desee insertar el bloque de funciones y abra el buscador FFB como se indica a continuación:

- Seleccione el menú **Editar** →**Buscador FFB**.
  O bien
- Haga clic con el botón derecho en el lugar correspondiente del editor y seleccione el comando **Buscador FFB...** en el menú contextual.

Aparece el cuadro de diálogo **Buscador FFB**:



El cuadro de diálogo **Buscador FFB** contiene los siguientes elementos para encontrar una función o un bloque de funciones:

Elemento	Descripción
Buscar	En el cuadro de texto **Buscar**, introduzca el nombre de la función o el bloque de funciones que desea insertar en el código de programación. Como comodín puede usar un signo de cierre de interrogación (?), que reemplaza exactamente un carácter, o bien un asterisco (*), que puede reemplazar varios caracteres o ninguno.
Empresa	Si sabe la empresa que creó la biblioteca que incluye el bloque de funciones que está buscando, puede seleccionar las siguientes empresas en la lista **Empresa**: ● **3S - Smart Software Solutions GmbH** ● **Grupo de trabajo técnico CAA** ● **Schneider Electric** ● **Sistema** Este parámetro está establecido de forma predeterminada en **Todas las empresas**.
Mayúsculas y minúsculas	Marque la casilla **Mayúsculas y minúsculas** para realizar una búsqueda que distinga entre mayúsculas y minúsculas. De forma predeterminada, esta casilla no está seleccionada.
Incluir comentarios en la búsqueda	Marque la casilla **Incluir comentarios en la búsqueda** para buscar la cadena especificada no sólo en los nombres de funciones y bloques de funciones, sino también en los comentarios que se guardan con ellos. De forma predeterminada, esta casilla no está seleccionada.
Buscar únicamente en bibliotecas del proyecto	Marque la casilla **Buscar únicamente en bibliotecas del proyecto** para limitar la búsqueda a las bibliotecas que se usan en la aplicación actual. De forma predeterminada, esta casilla no está seleccionada, y la operación de búsqueda incluye todas las bibliotecas instaladas en el PC con SoMachine.
Buscar	Haga clic en el botón **Buscar** o pulse la tecla INTRO para empezar a buscar la función o el bloque de funciones.

Resultados devueltos por el buscador FFB

Las funciones o bloques de funciones que coincidan con los criterios de búsqueda especificados se indicarán en la lista **Resultados**, con la siguiente información:

- **Nombre** de la función o bloque de funciones.
- **Biblioteca** en la que está almacenada la función o bloque de funciones.
- **Versión** de la biblioteca.
- **Empresa** que ha creado la biblioteca.
- Si está disponible, se mostrará un comentario en la columna situada en la parte derecha.
- La columna **Cargado** en la parte izquierda indica si la biblioteca, la función o el bloque de funciones ya está en uso en el proyecto actual.

Para mostrar más información acerca de una de las funciones o bloques de funciones, selecciónelo en la lista. En el siguiente campo, se mostrará un gráfico de la función/bloque de funciones con sus entradas y salidas, así como una descripción o cualquier información adicional, si está disponible.

Integración de una función/bloque de funciones en el código de programación

Para integrar una función/bloque de funciones encontrado por el buscador FFB en el código de programación, selecciónelo en la lista **Resultados** y:

- Haga doble clic en la entrada seleccionada de la lista **Resultados**.
- O bien, haga clic en el botón **Aceptar**.

La función/bloque de funciones seleccionado se insertará en el lugar donde esté colocado el cursor en el código de programación y la biblioteca respectiva se cargará automáticamente.

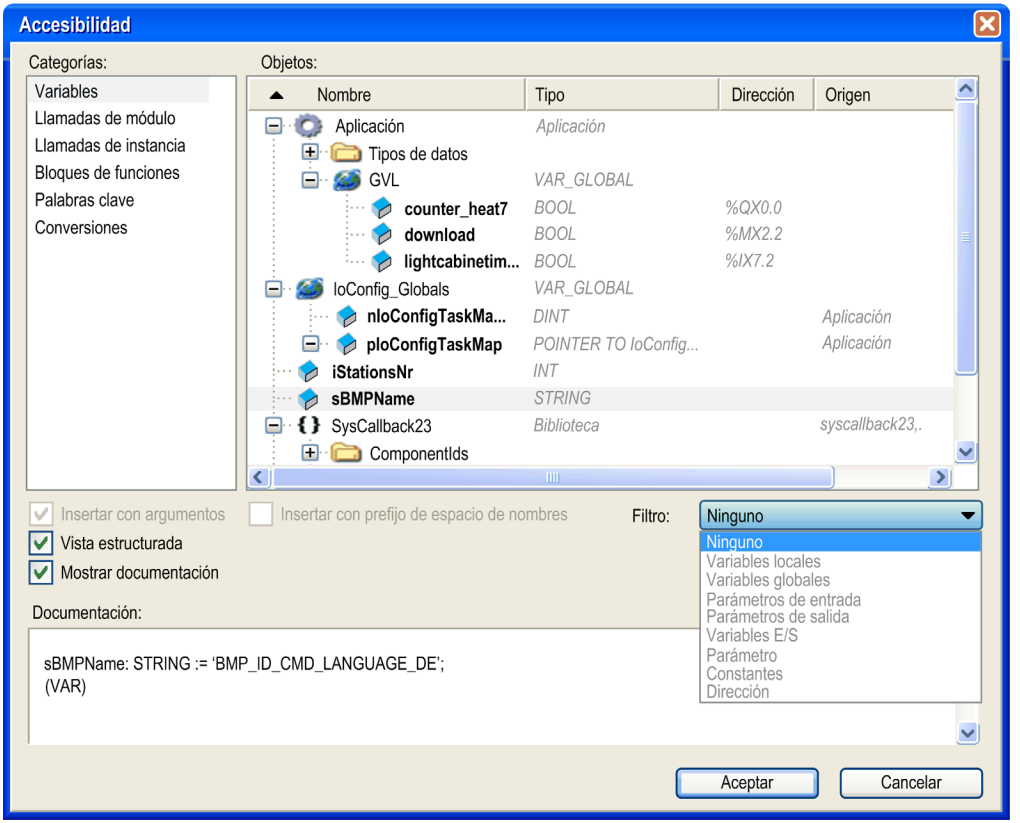
Repita esta operación cuando necesite ayuda para encontrar una función/bloque de funciones específico.

Accesibilidad

Descripción general

El cuadro de diálogo **Accesibilidad** y el comando correspondiente **Accesibilidad** (de forma predeterminada, en el menú **Editar** →**Codificación inteligente**) sólo está disponibles si el cursor se coloca en una ventana del editor de texto. El cuadro de diálogo ofrece los elementos del proyecto disponibles para su inserción en la posición actual del cursor.

Cuadro de diálogo **Accesibilidad**



Descripción de los elementos

El cuadro de diálogo **Accesibilidad** proporciona los elementos siguientes:

Elemento	Descripción
Categorías	En esta área, los elementos del proyecto se clasifican por **categorías**.
Filtro	Puede definir un **filtro** para la categoría **Variables**. Para mostrar un determinado tipo de variable, seleccione una entrada de la lista, como **Variables locales**, **Variables globales** o **Constantes**.
Área **Objetos**	
Nombre:, **Tipo**, **Dirección**, **Origen**	El área **Objetos** muestra los elementos disponibles y, en función de la categoría, también el **Tipo**, **Dirección** y **Origen** de los datos para la categoría seleccionada en el área **Categorías**: **Origen** se muestra para variables de E/S (ruta en el árbol **Dispositivos**) y variables definidas por biblioteca (nombre de biblioteca y categoría). Puede ordenar los elementos por **Nombre**, **Tipo**, **Dirección** u **Origen**, en orden alfabético ascendente o descendente. Para ello, haga clic en el encabezado de columna respectivo (símbolo de flecha hacia arriba o hacia abajo). Para mostrar u ocultar las columnas **Tipo**, **Dirección** u **Origen**, haga clic con el botón derecho del ratón en el encabezado de columna respectivo.
Vista estructurada	Si la opción **Vista estructurada** está seleccionada, los elementos del proyecto se muestran en un árbol de estructura complementado con iconos. Si la opción no está seleccionada, los elementos del proyecto se organizan sin jerarquías. Cada elemento del proyecto se muestra con la POU a la que pertenece (ejemplo: **GVL1.gvar1**).

NOTA: Si hay objetos disponibles con el mismo nombre en el nodo **Global** del árbol **Aplicaciones** así como debajo de una aplicación (árbol **Aplicaciones**), sólo se ofrece 1 entrada en **Accesibilidad**, porque el uso del objeto viene determinado por las prioridades de llamada habituales (primero el objeto asignado por la aplicación, luego el global).

Elemento	Descripción
Mostrar documentación	Si la opción **Mostrar documentación** esta seleccionada, el cuadro de diálogo **Accesibilidad** se amplía con el campo **Documentación**. Si el elemento seleccionado es una variable y se asigna una dirección a esta variable o se añade un comentario a su declaración, se mostrarán aquí.
Insertar con argumentos	Si se selecciona esta opción, los elementos que incluyen argumentos, como por ejemplo las funciones, se insertan con estos argumentos. Ejemplo: Si el bloque de funciones `FB1`, que contiene una variable de entrada `fb1_in` y una variable de salida `fb1_out`, se inserta con argumentos, se escribirá lo siguiente en el editor: `fb1(fb1_in:= , fb1_out=> )`
Insertar con prefijo de espacio de nombres	Si se selecciona esta opción, el elemento se inserta con el espacio de nombres prefijado. Actualmente, esta opción sólo está disponible para variables globales.

Parte VI
Herramientas

Contenido de esta parte

Esta parte contiene los siguientes capítulos:

Capítulo 21
Gestión de alarmas

Contenido de este capítulo

Este capítulo contiene los siguiente apartados:

Introducción

Administración de alarmas

Puede utilizar la administración de alarmas como parte de una aplicación para los fines siguientes:

- para ayudar a indicar los posibles errores de la máquina o del proceso
- para registrarlos
- para visualizarlos con la ayuda de elementos de visualización adecuados

A través de la aplicación o entradas del usuario en la visualización, las alarmas se pueden advertir y gestionar adecuadamente a través de acuses de recibo y acciones posteriores. Para este propósito hay disponibles elementos de visualización especiales, así como una biblioteca.

Para obtener más información, consulte el capítulo *Definición de alarma* *(véase página 483)*.

Establecimiento de un árbol Configuración de alarmas

Establezca un árbol **Configuración de alarmas** adecuado en un proyecto situado bajo la aplicación correspondiente, como se muestra en la siguiente figura.

- Primero inserte un objeto **Configuración de alarmas** en el árbol **Aplicaciones** bajo la aplicación haciendo clic en el signo más de color verde y ejecute el comando **Añadir otros objetos →Configuración de alarmas...** La biblioteca respectiva AlarmManager.library se añade al **Administrador de bibliotecas** automáticamente.
- Inserte los siguientes objetos bajo el nodo **Configuración de alarmas** haciendo clic en el signo más de color verde y seleccionando las opciones:
  - **Clase de alarma...**
  - **Grupo de alarmas...**
- Defina los grupos de alarmas que desee *(véase página 492)*.
- Configure las clases de alarmas requeridas *(véase página 486)*.
- Configure el almacenamiento de alarmas *(véase página 498)*.
- Programe la gestión de alarmas como desee en las POU de la aplicación.
- Cree una visualización utilizando el elemento de tabla de alarmas o banner de alarmas.

Configuración de alarmas

Descripción general

Para configurar la gestión de alarmas en SoMachine, añada un objeto **Configuración de alarmas** debajo del nodo **Aplicación** en el árbol **Aplicaciones**. Este objeto no muestra ninguna vista de editor. Sólo sirve como nodo raíz debajo del cual deberá insertar los objetos de configuración concretos:

- **Clase de alarma**: Una clase de alarma sirve para establecer el tipo de alarma, que significa que define un conjunto básico de parámetros determinado para una alarma. La clase es un ajuste obligatorio que debe realizarse cuando se crea un grupo de alarmas. Por ejemplo, define cómo acusar recibo de la alarma y si debe registrarse y visualizarse o escribirse en un archivo. Para obtener más información sobre cómo realizar la configuración, consulte *Editor de clases de alarmas (véase página 486)*.
- **Grupo de alarmas**: Un grupo de alarmas sirve para realizar la configuración concreta de una o varias alarmas (a las que se asigna una clase determinada y otros parámetros) para usarlas en la aplicación. Para obtener más información sobre cómo realizar la configuración, consulte *Editor de grupos de alarmas (véase página 492)*.
- Archivo de **Guardado de alarmas**: Este archivo lo proporciona implícitamente una base de datos y sirve para guardar los registros de alarmas. Para obtener más información sobre cómo realizar la configuración, consulte *Editor de almacenamiento de alarmas (véase página 498)*.

Puede definir más configuraciones en los elementos respectivos de visualización de alarmas, la tabla de alarmas y el banner de alarmas.

Para poder usar esta configuración de alarmas en la gestión de las alarmas de las POU de la aplicación, la biblioteca AlarmManager.library ofrece funciones e interfaces especiales.

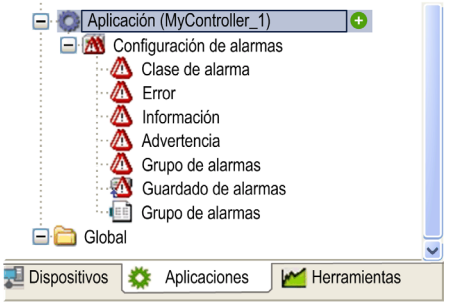
Establecimiento de la configuración de alarmas para una aplicación

Primero inserte un objeto **Configuración de alarmas** en el árbol **Aplicaciones** por debajo de la aplicación haciendo clic en el signo más de color verde y ejecute el comando **Añadir otros objetos →Configuración de alarmas...** La biblioteca respectiva AlarmManager.library se añade al **Administrador de bibliotecas** automáticamente.

Configure el sistema de alarmas en los editores de los tres tipos de objeto siguientes:

- **Clase de alarma**
- **Grupo de alarmas**
- **Guardado de alarmas**

Árbol de **Configuración de alarmas**



Definición de alarma

Descripción general

En SoMachine, una definición de alarma se compone de los siguientes elementos básicos:

- descripción general (como el ID y el texto del mensaje)
- descripción de la causa de la alarma (por ejemplo, la expresión a supervisar, los límites o el tiempo de espera mínimo)
- descripción de los efectos de la alarma (por ejemplo, acciones, propiedades de visualización o método de acuse de recibo)

Consulte las siguientes descripciones y definiciones universales para la gestión de alarmas en SoMachine:

Alarma: Generalmente una alarma se considera una condición especial (valor de la expresión).

Prioridad: La prioridad, también llamada gravedad, de una alarma describe la importancia de la condición de alarma. La prioridad más alta es 0 y el valor válido de prioridad más baja es 255.

Estado de alarma: Una expresión o variable de configuración para el control de la alarma puede tener los siguientes estados.

Método de acuse de recibo: Están disponibles los siguientes acuses de recibo. Para obtener información más detallada, consulte la descripción del editor de clases de alarmas *(véase página 486)*.

Estados de alarma y métodos de acuse de recibo	Descripción
REP (2#010)	alarma inactiva después de haber eliminado la causa
ACK (2#001)	alarma inactiva después del acuse de recibo método de acuse de recibo para eventos
REP_ACK (2#011)	alarma inactiva después de reparación (singular) y acuse de recibo
ACK_REP (2#110)	alarma inactiva después de acuse de recibo y reparación
ACK_REP_ACK (2#111)	alarma inactiva después del acuse de recibo opcional de haberla recibido, reparación
NO_ACK	sin acuse de recibo

Acuse de recibo de alarmas: El objetivo principal de las alarmas es informar al usuario sobre las situaciones de alarma. Para ello, suele ser necesario verificar que el usuario haya recibido esta información (consulte las posibles acciones asignadas a una alarma en la configuración de la clase de alarma). El usuario debe acusar recibo de la alarma para que se elimine de la lista de alarmas. Los tipos concretos de acuse de recibo se pueden describir como transiciones de estado en los diagramas de estado.

Evento de alarma: No confunda un evento de alarma con una condición de alarma. Mientras que una condición de alarma puede ser válida para un tiempo más largo, un evento de alarma describe la ocurrencia momentánea de un cambio, o sea, un cambio del estado normal al estado de alarma. En la configuración de alarma de SoMachine se utilizan los mismos nombres (Active, ACK, Deactive) para los 3 tipos de eventos y los estados de alarma correspondientes.

SoMachine admite las siguientes funciones:

- desactivación de la generación de alarma para las alarmas individuales, así como para grupos de alarmas
- selección de las alarmas que se mostrarán mediante la definición de grupos y prioridades de alarma
- almacenamiento de los eventos de alarma en una tabla de alarmas
- visualización de los elementos **tabla de alarmas** y **banner de alarmas** en SoMachine

Editor de configuración de alarmas

Descripción general

El editor para configurar la gestión de alarmas en SoMachine consta de las partes siguientes, cada una de ellas asignada a su propio objeto bajo el nodo **Configuración de alarmas** en el árbol **Aplicaciones**:

- editor de clases de alarmas *(véase página 486)*
- editor de grupos de alarmas *(véase página 492)*
- editor de almacenamiento de alarmas *(véase página 498)*

Al hacer doble clic en un objeto se abre el cuadro de diálogo del editor respectivo.

Editor de clases de alarmas

Descripción general

Una clase de alarma describe características generales de una alarma:

- gestión de acuses de recibo (confirmación de una alarma por parte del usuario)
- acciones que se deben ejecutar automáticamente en cuanto se haya detectado un estado de alarma concreto
- colores y mapas de bits que se deben utilizar para una visualización de una tabla de alarmas

Las clases de alarma se añaden al árbol de configuración de alarmas de una aplicación como objetos individuales. Cada clase de alarma se puede configurar.

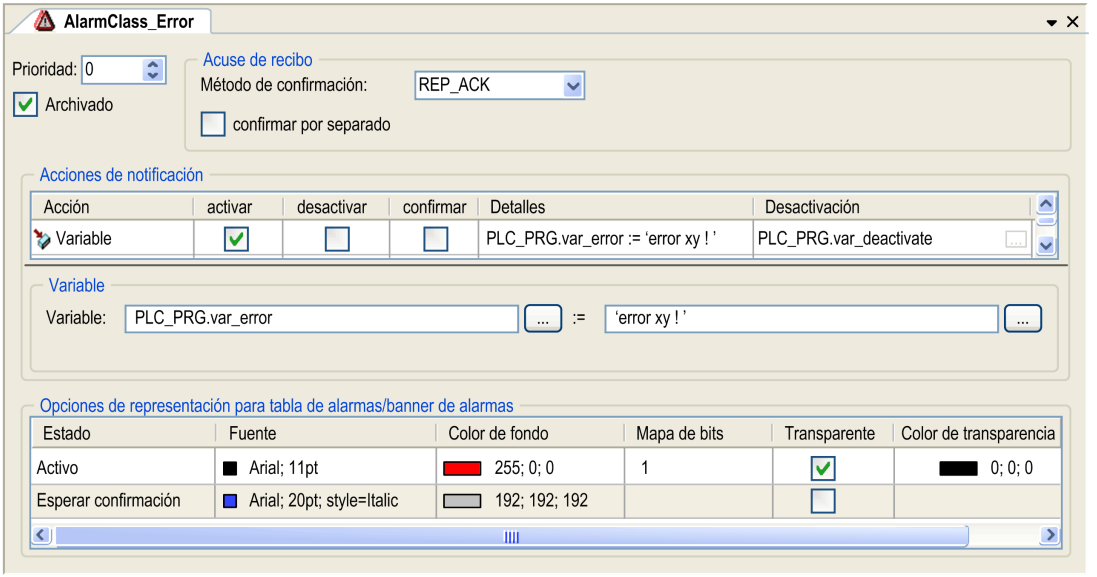
Las clases de alarmas configuradas están disponibles en una lista de selección en el cuadro de diálogo de configuración de un grupo de alarmas *(véase página 492)* para definir la configuración básica para cada alarma perteneciente a este grupo específico.

Por ejemplo, podría definir una clase de alarma para alarmas que son de alta ponderación o prioridad y una para alarmas que son de prioridad o ponderación inferior, en las que sólo se deberían ofrecer advertencias. De forma invariable, estos tipos de situaciones de alarma diferirán por lo que respecta a la prioridad y los requisitos de acuse de recibo.

Creación de una clase de alarma

Inserte un objeto **Clase de alarma** debajo del nodo **Configuración de alarmas** en el árbol **Aplicaciones**. La vista del editor de configuración se abre automáticamente, titulada con el nombre de la clase:

Vista del editor de clases de alarmas



Parámetro	Descripción
Prioridad	Especifique la prioridad de las alarmas *(véase página 483)* de esta clase (0...255). La prioridad, también conocida como ponderación, describe la importancia (ponderación) de las condiciones de la alarma, con 0 como la prioridad más alta y 255 como la más baja.
Archivado	Habilite esta opción si desea que se registren las alarmas de esta clase respecto a la configuración de almacenamiento de alarmas definida actualmente.

Descripción del área Acuse de recibo

Suele ser necesario verificar que el usuario se ha dado cuenta de la alarma (consulte las acciones posibles asignadas a una alarma en la configuración de la clase de alarma). El usuario debe acusar recibo de la alarma para que se elimine de la lista de alarmas.

Método de acuse de recibo:

Están disponibles los siguientes acuses de recibo:

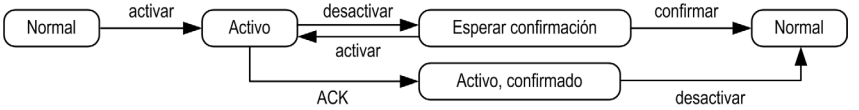
Acuse de recibo	Descripción
REP	alarma inactiva después de haber eliminado la causa
ACK	alarma inactiva después del acuse de recibo método de acuse de recibo para eventos
REP_ACK	alarma inactiva después de reparación y acuse de recibo únicos
ACK_REP	alarma inactiva después de acuse de recibo y reparación
ACK_REP_ACK	alarma inactiva después del acuse de recibo opcional de haber recibido la alarma, reparación y acuse de recibo de fin de la situación de alarma

Vea a continuación para cada método de acuse de recibo un diagrama que muestra la secuencia de estados y estados de transición. Estos diagramas se muestran en una información sobre herramientas cuando se coloca el cursor del ratón sobre el cuadro de selección en el editor de configuración.

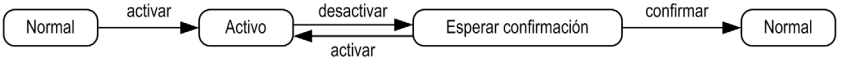
Transiciones de estado para el método de acuse de recibo ACK

Transiciones de estado para el método de acuse de recibo REP

Transiciones de estado para el método de acuse de recibo ACK_REP



Transiciones de estado para el método de acuse de recibo REP_ACK



Transiciones de estado para el método de acuse de recibo ACK_REP_ACK

Estados posibles

Estado	Descripción
Normal	Sin alarma (estado normal).
Activo	Situación de alarma detectada (alarma existente).
Esperar confirmación	La condición que generó la alarma ya no se produce, la confirmación aún no se ha realizado.
Activo, confirmado	La condición de alarma aún se produce, acuse de recibo realizado, esperando confirmación.

Estados de transición posibles

Estado	Descripción
activar	condición detectada que provoca el estado de alarma
desactivar	la condición que provoca la alarma ya no se detecta
confirmar	confirmación de un estado de alarma preexistente
ACK	acuse de recibo de una alarma activa

Confirmar por separado: Si esta opción está activada, no se puede acusar recibo de la alarma junto con otras alarmas.

Descripción del área Acciones de notificación

A cada clase de alarma se le puede asignar una lista de acciones que se deben realizar en cuanto se produzca una transición de estado en la alarma.

Para añadir una **Acción**, haga doble clic en la columna **Acción** de la última línea vacía de la tabla.

Se abre una lista de selección en la que se puede seleccionar una de las opciones siguientes:
- **Llamada (de un bloque de funciones)**
- **Ejecutar (un archivo)**
- **Variable (asignación)**

La configuración correspondiente a **activar, desactivar, confirmar** depende del método de acuse de recibo establecido actualmente para la clase de alarma y representa las transiciones de estado. Puede modificarla directamente en la tabla.

Desactivación: Aquí, se puede introducir una variable por medio de la cual la aplicación puede desactivar la acción.

Según el tipo de acción, se deben definir los parámetros adicionales siguientes en los campos de edición en el área que hay bajo la tabla:

Acción	Parámetros
Variable	En el campo de la izquierda, especifique la **Variable** (ejemplo: `PLC_PRG.altitude`). En la derecha, introduzca otra variable o una expresión para definir la asignación que debe afectar a una alarma. Haga clic en el botón **...** para abrir el asistente Accesibilidad. Opcionalmente, puede introducir los marcadores de posición siguientes definiendo un texto de mensaje: ● ALARM, en caso de variables booleanas ● STATE, en caso de variables enteras ● cualquier literal conforme a IEC (por ejemplo, "demasiado elevado"), en caso de variables de cadena
Llamada	Especifique el nombre de la **Instancia de bloque de funciones** a la que se debe llamar. El bloque de funciones debe implementar la interfaz `IAlarmNotifiable` (AlarmManager.library) para proporcionar información sobre la alarma. Si el bloque de funciones necesita una estructura de parámetros adicional, los componentes de la estructura se proporcionarán en la tabla **Estructura de parámetros adicional** y se pueden rellenar allí. Ejemplo: FB tiene el atributo `{attribute AlarmManagerAdditionalData := StructEmailParams}`. La estructura utilizada debe ser plana y constar de componentes escalares o de componentes de tipo `POINTER TO`. El tipo `ARRAY` o los tipos definidos por el usuario no se admiten. Sugerencia: Puede marcar los componentes de la estructura como parámetros obligatorios utilizando el atributo `{attribute AlarmManagerMandatoryParameter}`.
Ejecutar	Especifique el nombre del **Archivo ejecutable** que se debe ejecutar en cuanto se produzca el evento de alarma. En el campo **Parámetro**, puede escribir directamente cualquier parámetro que se tenga que añadir a la llamada. Separe varios parámetros con un espacio vacío.

Por motivos de claridad, las propiedades específicas de la acción se muestran en la columna **Detalles**.

Opciones de representación para tabla de alarmas/banner de alarmas

Aquí puede definir cómo se mostrarán los estados específicos de una alarma de esta clase en el elemento de visualización de la alarma. Los estados que se enumeran actualmente aquí en la columna **Estado** de la tabla dependen del tipo de acuse de recibo establecido anteriormente. Para cada estado, puede definir las propiedades siguientes después de hacer doble clic en la celda de la tabla respectiva:

Fuente, Color de fondo, Mapa de bits, Transparente, Color de transparencia

Haga clic en el botón **...** para abrir los cuadros de diálogo estándar para seleccionar una fuente o un color.

Puede añadir un mapa de bits introduciendo un ID de mapa de bits. Si dicho ID aún no se ha asignado a un archivo, se abrirá el cuadro de diálogo estándar para buscar un archivo. Después de seleccionar uno, la definición del mapa de bits se añadirá a la colección de imágenes **GlobalImagePool**.

Editor de grupos de alarmas

Descripción general

Los grupos de alarmas se utilizan para organizar las alarmas específicas en la gestión de alarmas de una aplicación. Cada alarma está asignada de forma definitiva a un grupo de alarmas y este grupo la gestiona. A todas las alarmas de un grupo se les puede asignar una variable de desactivación común. Tenga en cuenta que incluso una sola alarma debe estar configurada dentro de un grupo de alarmas. Puede definir una estructura jerárquica de grupos de alarmas en el árbol **Aplicaciones** utilizando los elementos de la carpeta.

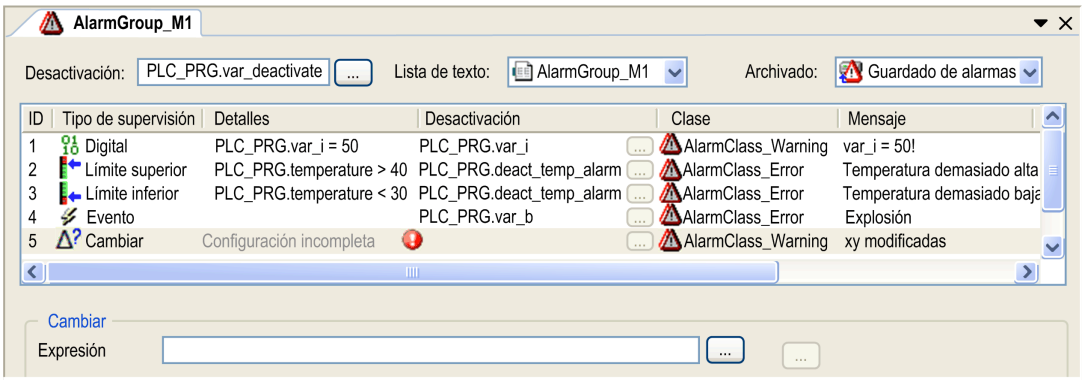
El cuadro de diálogo del editor de **grupos de alarmas** contiene una tabla que enumera las alarmas del grupo en cuestión. La configuración básica para el grupo, como por ejemplo la de desactivación, archivado y la lista de texto asignada, así como las propiedades de cada alarma específica, se muestra y se puede modificar.

La tabla se puede ordenar de forma ascendente y descendente según las columnas específicas haciendo clic en el encabezado de la columna respectiva.

Introduzca una alarma nueva haciendo doble clic en la columna de la tabla **Tipo de supervisión** en la celda de debajo de la última entrada de la tabla o en la primera línea de la tabla si aún no hay ninguna entrada. Después de haber seleccionado el tipo de supervisión deseado para la nueva alarma, se introducirá un ID automáticamente y se mostrarán sugerencias relacionadas sobre qué se debe hacer en los campos restantes. Los campos que aún no contienen una entrada correcta se marcarán con un icono rojo que contiene un signo de exclamación.

Vea la descripción siguiente sobre los elementos específicos del cuadro de diálogo. Donde esté disponible, puede hacer clic en el botón **...** para abrir el asistente Accesibilidad para realizar la entrada respectiva.

En la figura siguiente se muestra un ejemplo de la configuración de un grupo de alarmas **AlarmGroup_M1** en funcionamiento:



Descripción de los parámetros

Parámetro	Descripción
Desactivación	Aquí puede introducir una variable booleana que habilite (en un flanco ascendente) o deshabilite (en un flanco descendente) la activación de las alarmas pertenecientes a este grupo.
Lista de texto	Este campo sirve para visualizar el nombre de la lista de texto *(véase página 234)* que se genera automáticamente al crear un grupo de alarmas y que sirve para almacenar los textos de mensaje de alarma.
Archivado	Este ajuste muestra si el almacenamiento de alarmas ha finalizado o no para las alarmas del grupo actual. En cuanto se introduzca una clase de alarma para la que se ha habilitado el almacenamiento de alarmas en la tabla de definiciones de alarmas, esta entrada se modificará de **(ninguno)** a **Guardado de alarmas**. **Guardado de alarmas** corresponde al nombre del objeto de almacenamiento de alarmas definido actualmente en la configuración de alarmas. Para obtener más información, consulte la descripción del editor de almacenamiento de alarmas *(véase página 498)*.

Descripción de la tabla de definiciones de alarmas

Las propiedades de las alarmas específicas se deben definir en las columnas de las tablas respectivas. Según el tipo de supervisión seleccionado, es posible que tenga que establecer propiedades adicionales en campos de edición en el área situada bajo la tabla.

Las propiedades de alarma posibles se describen en la tabla:

Elemento de definición de alarma	Descripción
ID	Al definir una alarma nueva, se asigna automáticamente un ID exclusivo en forma de número consecutivo. Este ID corresponde al utilizado en la lista de texto asignada. Puede modificar el ID posteriormente en el campo **ID** en la tabla. Pero tenga en cuenta que el ID debe seguir siendo exclusivo en el grupo de alarmas. Una modificación del ID en la tabla de alarmas también actualiza directamente la lista de texto y viceversa.
[1)] Para este tipo de supervisión, puede supervisar la expresión especificada de forma absoluta o relativa. Si se hace de forma absoluta, el límite se define mediante un valor fijo o una variable que proporcione un valor fijo. Si se hace de forma relativa, el límite se basa en una determinada expresión, por ejemplo: **Límite superior**: `Variable x >= 0.9 * y`. De forma opcional, se puede especificar una histéresis. **Histéresis en %**: Si se especifica una histéresis, la situación de alarma será verdadera hasta que se alcance una cierta desviación respecto al valor de límite especificado. El tamaño de la desviación se define como un porcentaje [%] del valor de límite. Ejemplo: **Límite superior**: `i_temp >= 30 Hysteresis: 10%`. En cuanto la variable `i_temp` alcanza o supera el valor 30, se produce la situación de alarma. La situación de alarma no termina hasta que el valor queda por debajo de 27.	

Elemento de definición de alarma		Descripción
Tipo de supervisión	**Digital**	**Expresión**: En la parte izquierda, introduzca la expresión que se debe supervisar y, en la parte derecha, la expresión *(véase página 398)* respecto a la cual desea comprobarla (límite); en la parte del medio, seleccione el operador de comparación que desee (= o <>).
	Límite superior	**Expresión**: Tal como se ha descrito anteriormente para **Digital**, pero con las opciones de comparación > o >= y la definición **Histéresis en %** opcional.
	Límite inferior[1)]	**Expresión**: Tal como se ha descrito anteriormente para **Digital**, pero con las opciones de comparación < o <= y la definición **Histéresis en %** opcional.
	Dentro de rango[1)]	**Expresión**: Introduzca la expresión que se debe supervisar. **Área**: Se producirá una situación de alarma en cuanto la expresión que se debe supervisar quede dentro del rango de valores definido. En la parte izquierda, introduzca la expresión que define el límite inferior del área; en la parte derecha, el límite superior. La expresión que se debe supervisar se muestra en el campo no editable en la parte del medio. Establezca los operadores de comparación del modo apropiado y, opcionalmente, defina una **Histéresis en %**.
	Fuera de rango[1)]	**Expresión**: Introduzca la expresión que se debe supervisar. **Área**: Se producirá una situación de alarma en cuanto la expresión que se debe supervisar quede fuera del rango de valores definido. En la parte izquierda, introduzca la expresión que define el límite inferior del área; en la parte derecha, el límite superior. La expresión que se debe supervisar se muestra en el campo no editable en la parte del medio. Establezca los operadores de comparación del modo apropiado y, opcionalmente, defina una **Histéresis en %**.
	Modificación	**Expresión**: Introduzca la expresión que se debe supervisar. Se producirá una situación de alarma en cuanto su valor cambie.
	Evento	En este caso, se activa una situación de alarma por medio de la aplicación, utilizando funciones de AlarmManager.library. Ejemplo para un grupo de alarmas: **AlarmGroup_M1** (coincide con el ejemplo que se muestra en la figura anterior): `AlarmManager.AlarmGlobals.g_AlarmHandler.RaiseEvent(AlarmGroup_M1', '4';)`

[1)] Para este tipo de supervisión, puede supervisar la expresión especificada de forma absoluta o relativa. Si se hace de forma absoluta, el límite se define mediante un valor fijo o una variable que proporcione un valor fijo. Si se hace de forma relativa, el límite se basa en una determinada expresión, por ejemplo: **Límite superior**: `Variable x >= 0.9 * y`.
De forma opcional, se puede especificar una histéresis.
Histéresis en %: Si se especifica una histéresis, la situación de alarma será verdadera hasta que se alcance una cierta desviación respecto al valor de límite especificado. El tamaño de la desviación se define como un porcentaje [%] del valor de límite. Ejemplo: **Límite superior**: `i_temp >= 30 Hysteresis: 10%`. En cuanto la variable `i_temp` alcanza o supera el valor 30, se produce la situación de alarma. La situación de alarma no termina hasta que el valor queda por debajo de 27.

Elemento de definición de alarma	Descripción
Detalles	Este campo no es editable. Muestra la configuración definida para la alarma actual por medio de los campos adicionales que hay debajo de la tabla.
Desactivación	Se puede especificar de forma opcional una variable mediante la cual la alarma se puede desactivar.
Clase	Entrada obligatoria de una de las clases de alarma disponibles proporcionadas en la lista de selección. Consulte también Editor de clases de alarmas *(véase página 486)*.
Mensaje	Entrada obligatoria del texto de un mensaje que se debe visualizar en el elemento de la **Tabla de alarmas** en cuanto se produzca la alarma. El texto se escribirá automáticamente en la lista de texto especificada para el grupo. Para insertar saltos de línea, pulse CTRL + INTRO. Los marcadores de posición admitidos se describen en el párrafo *(véase página 496)* siguiente. Estos marcadores de posición se sustituirán en tiempo de ejecución por el valor actual.
Tiempo de espera mín.	Si la alarma no se debe activar inmediatamente cuando se produce el evento de alarma, entonces se puede definir un tiempo de espera aquí. Introduzca el retardo deseado de acuerdo con la sintaxis *(véase página 800)* de IEC 6-1131, por ejemplo `t#2ms`.
Latch var 1, Latch var 2	Puede utilizar las variables latch para definir 2 valores adicionales para registrarlos cuando la alarma se active. Ejemplo: Mediante un método `RaiseEvent` en `IAlarmHandler`, partes de la aplicación pueden afectar a alarma de eventos. Con este objetivo, visualice el nombre del grupo de alarmas y el ID de la alarma. Donde de detecte la alarma, se mostrarán avisos registrados adicionales utilizando el mecanismo de variables latch.

[1)] Para este tipo de supervisión, puede supervisar la expresión especificada de forma absoluta o relativa. Si se hace de forma absoluta, el límite se define mediante un valor fijo o una variable que proporcione un valor fijo. Si se hace de forma relativa, el límite se basa en una determinada expresión, por ejemplo: **Límite superior**: `Variable x >= 0.9 * y`.
De forma opcional, se puede especificar una histéresis.
Histéresis en %: Si se especifica una histéresis, la situación de alarma será verdadera hasta que se alcance una cierta desviación respecto al valor de límite especificado. El tamaño de la desviación se define como un porcentaje [%] del valor de límite. Ejemplo: **Límite superior**: `i_temp >= 30 Hysteresis: 10%`. En cuanto la variable `i_temp` alcanza o supera el valor 30, se produce la situación de alarma. La situación de alarma no termina hasta que el valor queda por debajo de 27.

Elemento de definición de alarma	Descripción
Alarma de prioridad superior	Aquí, puede especificar una alarma definida de prioridad *(véase página 483)* superior, que acusará recibo automáticamente de la actual si ambas se producen a la vez. Esto sirve para obtener una gestión de alarmas de dos etapas, ya que la alarma con la prioridad inferior queda solapada por la de prioridad superior. Por ejemplo, en caso de un sistema de control de temperatura: Está configurando únicamente una alarma de prioridad inferior (por ejemplo, prioridad=10) para generar una advertencia en caso de que la temperatura supere los 30° C. Ya ha configurado otra alarma de prioridad superior (por ejemplo, prioridad=1) que debe generar una alerta en cuanto la temperatura supere los 50° C. Ahora. esta alarma de alerta se puede introducir aquí en la configuración de la alarma de advertencia como una alarma de prioridad superior. Esto ayuda a garantizar que se acuse recibo automáticamente de una alarma de advertencia existente en cuanto se produzca la alarma de alerta.
[1)] Para este tipo de supervisión, puede supervisar la expresión especificada de forma absoluta o relativa. Si se hace de forma absoluta, el límite se define mediante un valor fijo o una variable que proporcione un valor fijo. Si se hace de forma relativa, el límite se basa en una determinada expresión, por ejemplo: **Límite superior**: `Variable x >= 0.9 * y`. De forma opcional, se puede especificar una histéresis. **Histéresis en %**: Si se especifica una histéresis, la situación de alarma será verdadera hasta que se alcance una cierta desviación respecto al valor de límite especificado. El tamaño de la desviación se define como un porcentaje [%] del valor de límite. Ejemplo: **Límite superior**: `i_temp >= 30 Hysteresis: 10%`. En cuanto la variable `i_temp` alcanza o supera el valor 30, se produce la situación de alarma. La situación de alarma no termina hasta que el valor queda por debajo de 27.	

Marcadores de posición para la definición de mensajes

Marcador de posición	Descripción
DATE	Fecha de modificación al estado actual.
TIME	Hora de la última modificación de estado.
EXPRESSION	Expresión (definida en la alarma) que ha activado la alarma.
PRIORITY	Prioridad de la alarma (definida en la clase de alarma).
TRIGGERVALUE *	Valor que ha provocado que la condición de alarma esté activa.
ALARMID	ID de alarma tal como se muestra en la primera columna de la tabla de alarmas.
CLASS	Nombre de la clase de alarma (definida en la alarma).
ALLDEFAULT	Se volcará toda la información sobre la alarma.
CURRENTVALUE *	Valor actual de la variable supervisada.
* Para TRIGGERVALUE, CURRENTVALUE, LATCH1 y LATCH2 también puede utilizar cadenas de formato válidas para la función C `printf`; por ejemplo: `<CURRENTVALUE %d>`.	

Marcador de posición	Descripción
LATCH1 *	Valor de la primera variable latch. Una variable latch debe ser de tipo escalar, excepto las de cadena (8 bytes). A variable latch o una expresión se registra cuando la alarma está activa (como por ejemplo la variable de desencadenador) y permite proporcionar información adicional sobre la alarma.
LATCH2 *	Valor de la segunda variable latch.
ALARM	TRUE en caso de que el estado de la alarma sea **Activa**. FALSE en caso de que el estado de la alarma sea cualquier otro.
STATE	Estado de alarma: 0 = normal, 1 = activa, 2 = esperando confirmación, 3 = activa, confirmada.
* Para TRIGGERVALUE, CURRENTVALUE, LATCH1 y LATCH2 también puede utilizar cadenas de formato válidas para la función C `printf`; por ejemplo: `<CURRENTVALUE %d>`.	

Editor de almacenamiento de alarmas

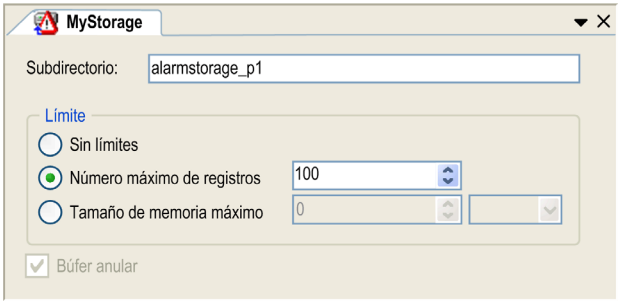
Descripción general

El objeto **Guardado de alarmas** forma parte de la administración de alarmas de una **Aplicación**. Permite definir determinados ajustes para un archivo de base de datos, que está disponible automáticamente para almacenar los registros de alarmas.

Hay exactamente 1 archivo de almacenamiento que se mantiene en una base de datos y que opcionalmente se puede guardar en un directorio del controlador. El nombre de este archivo no se puede modificar. Se deriva del nombre de la aplicación según esta convención: <nombre de aplicación>.alarmstorage.sqlite. Las clases de alarma *(véase página 486)* y los grupos de alarmas *(véase página 492)* determinan si utilizan el archivo de almacenamiento o no.

Las entradas de registro almacenadas en la base de datos pueden visualizarse en el elemento de visualización de alarma correspondiente.

Inserte el objeto **Guardado de alarmas** debajo del nodo **Configuración de alarmas** de la aplicación. Haga doble clic en ese objeto para abrir el siguiente editor de configuración.



Parámetro		Descripción
Subdirectorio		Opcionalmente, puede introducir el nombre de un subdirectorio en el controlador, donde se guardará el archivo de almacenamiento.
Límite		Active una de las siguientes opciones para poder definir cómo debe limitarse el tamaño de archivo del almacenamiento de alarmas.
	Sin límites	–
	Número máximo de registros	Especifique el número de entradas de registro que se deben almacenar. Al alcanzar ese número, la entrada más antigua se eliminará en cuanto entre una nueva entrada de registro (búfer anular).

Parámetro		Descripción
	Tamaño de memoria máximo	Especifique el tamaño máximo del archivo de almacenamiento introduciendo un número y una unidad. Este tamaño se convertirá de forma implícita en el número máximo de entradas de registro correspondiente que se almacenará en el archivo (búfer anular).
Búfer anular		Se habilita de forma automática si se ha especificado un límite; no es editable, pero indica que se utiliza un búfer anular para el almacenamiento.

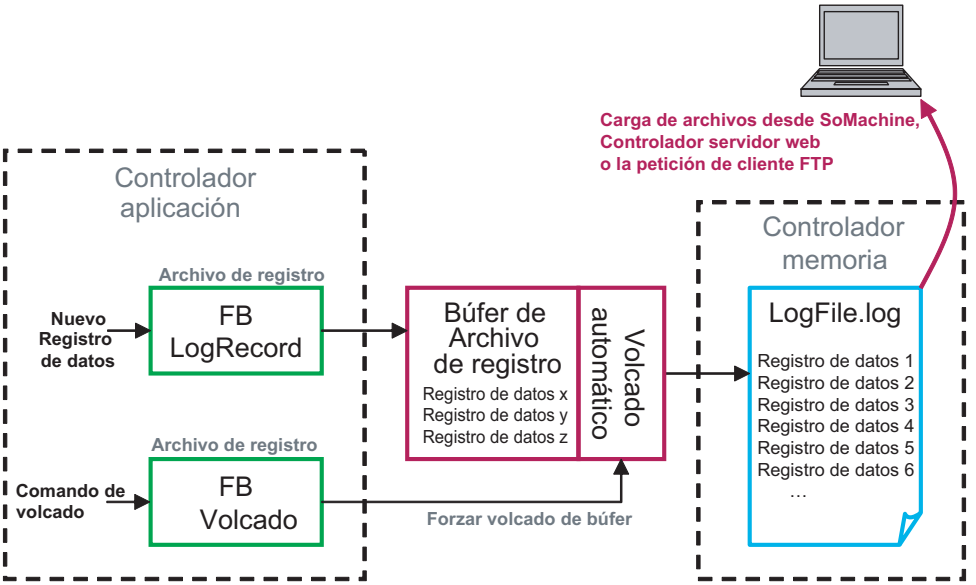
NOTA: El archivo de almacenamiento se eliminará al realizar un restablecimiento de origen.

Capítulo 22
Registro de datos

Introducción al registro de datos

Descripción general

Puede supervisar y analizar los datos de la aplicación examinando el archivo de registro de datos (.log).



En la ilustración se muestra una aplicación que incluye los 2 bloques de funciones, `LogRecord` y `Dump`. El bloque de funciones `LogRecord` escribe datos en el búfer, que se vacían en el archivo de registro de datos (.log) ubicado en la memoria del controlador. El volcado de búfer es automático cuando está al 80%, o se puede forzar con la función `Dump`. Como cliente FTP estándar, un PC puede acceder a este archivo de registro de datos cuando el controlador actúa como servidor FTP. También es posible cargar el archivo con SoMachine o el servidor web del controlador.

NOTA: Sólo los controladores con la función de gestión de archivos son compatibles con el registro de datos. Consulte el manual de programación del controlador para ver si admite la gestión de archivos. El software por sí mismo no comprueba la compatibilidad del controlador con las actividades de registro de datos.

Archivo de registro de datos de ejemplo (.log)

```
Entries in File: 8; Last Entry: 8;
18/06/2009;14:12:33;cycle: 1182;
18/06/2009;14:12:35;cycle: 1292;
18/06/2009;14:12:38;cycle: 1450;
18/06/2009;14:12:40;cycle: 1514;
18/06/2009;14:12:41;cycle: 1585;
18/06/2009;14:12:43;cycle: 1656;
18/06/2009;14:14:20;cycle: 6346;
18/06/2009;14:14:26;cycle: 6636;
```

Procedimiento de implementación

Antes de empezar a escribir el programa, primero debe declarar y configurar los archivos de registro de datos en la aplicación.

Capítulo 23
Gestor de fórmulas

Contenido de este capítulo

Este capítulo contiene los siguiente apartados:

Gestor de fórmulas

Descripción general

La funcionalidad **Gestor de fórmulas** sólo está disponible si se selecciona en el conjunto de características usado actualmente (**Opciones →Características →Conjunto predefinido de características**).

El gestor de fórmulas proporciona la funcionalidad para gestionar listas de variables de proyecto definidas por el usuario, definiciones de fórmulas con nombre y conjuntos de valores definidos para estas variables dentro de una definición de fórmula, denominados fórmulas.

Puede utilizar estas fórmulas para configurar y supervisar parámetros de control en el controlador. También se pueden cargar de archivos y guardar en archivos. Estas interacciones son posibles utilizando elementos de visualización que debe configurar debidamente (comando de ejecución de configuración de entrada). También puede utilizar determinados comandos de fórmula en la aplicación *(véase página 513)*.

Cuando haya seleccionado una fórmula, confirme si dicha fórmula se ajusta al proceso que se controlará.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

- Realizar un análisis de seguridad para la aplicación y el equipo instalado.
- Verificar que la fórmula sea apropiada para el proceso y el equipo o la función en la instalación.
- Proporcionar los parámetros apropiados, especialmente para los límites y otros elementos relacionados con la seguridad.
- Verificar que todos los sensores y accionadores sean compatibles con la fórmula seleccionada.
- Probar minuciosamente todas las funciones durante la verificación y la puesta en marcha.
- Proporcionar rutas independientes para funciones de control críticas (parada de emergencia, condiciones de superación de los límites, etc.) según los análisis de seguridad y las regulaciones y códigos aplicables.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

De forma predeterminada, el gestor de fórmulas se carga en el controlador durante la descarga. Gestiona la escritura y la lectura de las fórmulas cuando la aplicación se está ejecutando en el controlador. Sin embargo, no es necesario cargar el gestor de fórmulas en el controlador para utilizar fórmulas sólo para intercambiar parámetros durante el inicio del sistema (es decir, cuando SoMachine todavía está conectado al controlador). Puede desactivar su descarga con este fin utilizando la opción **Recipe management in the plc**. La escritura y la lectura de valores de fórmulas la gestionarán entonces los comandos y servicios online estándar. Si el gestor de fórmulas debe ejecutarse en el controlador porque el programa de aplicación lo necesita durante el tiempo de ejecución, entonces el bloque de funciones `RecipeCommands` se encarga de gestionar los comandos de fórmula.

Para ver una descripción del comportamiento de las fórmulas en las distintas modalidades online, consulte el capítulo *Definición de fórmula (véase página 508)*.

Si el gestor de fórmulas se encuentra en un controlador que no sea la aplicación afectada por las fórmulas, se utilizará el servidor de datos para leer/escribir las variables contenidas en las fórmulas. La lectura y escritura de las variables se realiza de forma síncrona. Puede llamar a `g_RecipeManager.LastError` después de leer/escribir para verificar si la transmisión se ha realizado correctamente (`g_RecipeManager.LastError=0` en este caso).

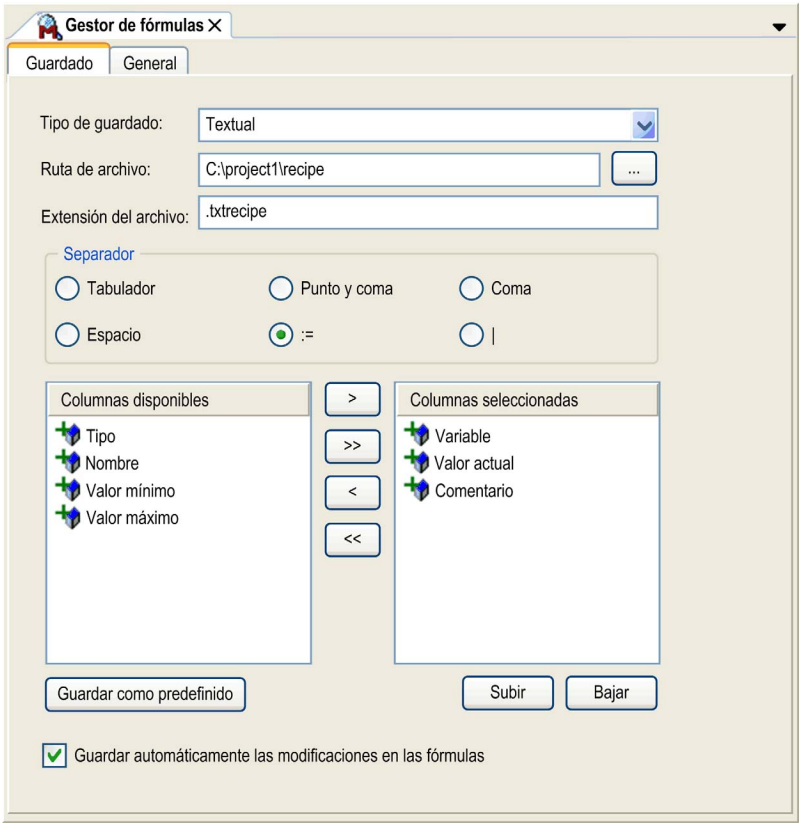
Objetos de gestión de fórmulas en el árbol Herramientas

Para añadir un objeto de **Gestor de fórmulas** en el árbol **Herramientas**, seleccione el nodo **Aplicación**, haga clic en el signo más de color verde y ejecute el comando **Añadir otros objetos... →Gestor de fórmulas...** Confirme el cuadro de diálogo **Add Recipe Manager** haciendo clic en **Agregar** y el nodo **Gestor de fórmulas** se insertará debajo del nodo **Aplicación**.

Puede añadir uno o varios objetos **Definición de fórmula** a un nodo **Gestor de fórmulas**. Para ello, haga clic en el signo más de color verde del nodo **Gestor de fórmulas** y ejecute el comando **Definición de fórmula...** Especifique un **Nombre** en el cuadro de diálogo **Add Recipe Definition** y haga clic en **Agregar**. Haga doble clic en el nodo para ver y editar definiciones de fórmula, incluidas las fórmulas específicas, en una ventana de editor independiente. Para ver una descripción del comportamiento de las fórmulas en las distintas modalidades online, consulte el capítulo *Definición de fórmula (véase página 508)*.

Editor de gestor de fórmulas, ficha Guardado

De forma predeterminada, las fórmulas se almacenarán automáticamente en archivos de acuerdo con la configuración de la ficha **Guardado** del editor de **Gestor de fórmulas**:



Parámetro	Descripción
Tipo de guardado	Seleccione el tipo de guardado **Textual** o **Binario**.
Ruta de archivo	Especifique la ubicación en la que se almacenará la fórmula.
Extensión del archivo	Especifique la extensión del archivo de la fórmula.

NOTA: También se puede definir un archivo de almacenamiento mediante la entrada en un elemento de visualización (configuración de entrada - ejecutar comando - guardar/cargar una fórmula desde un archivo). No obstante, al definir el nombre de este archivo en la configuración de visualización, no sobrescriba el archivo ** .txtrecipe* definido aquí en el gestor de fórmulas.

Parámetro	Descripción
Separador	En el caso del almacenamiento textual, las columnas seleccionadas para el almacenamiento se separarán mediante un separador. Seleccione una de las seis opciones propuestas.
Columnas disponibles	Todas las columnas de la definición de fórmula, representadas por el respectivo encabezado.
Columnas seleccionadas	Columnas seleccionadas de la definición de fórmula, es decir, las columnas que se deben almacenar. En esta parte se incluye, como mínimo, la columna que contiene el **Valor actual**. No se puede deseleccionar.
Botones de flecha	Las demás columnas se pueden desplazar a la derecha o a la izquierda seleccionando la entrada respectiva y haciendo clic en los botones de flecha. También puede pasar todas las entradas de un lado al otro a la vez utilizando los botones con la flecha doble.
Botones **Subir** y **Bajar**	Haga clic en estos botones para ajustar el orden de las columnas seleccionadas, que representa el orden de las columnas en el archivo de almacenamiento. Para cada fórmula, se creará un archivo <nombre de fórmula>.<definición de fórmula>.<extensión del archivo> en la carpeta especificada. Este archivo se volverá a cargar en el gestor de fórmulas en cada reinicio de la aplicación. Para conocer la configuración de actualización de los archivos de almacenamiento de fórmulas, consulte la descripción de la ficha *(véase página 507)* **General**.
Guardar como predefinido	Haga clic en el botón **Guardar como predefinido** para utilizar la configuración realizada en este cuadro de diálogo como configuración predeterminada para cada gestor de fórmulas adicional insertado.
Guardar automáticamente las modificaciones en las fórmulas	Seleccione esta opción para actualizar inmediatamente los archivos de almacenamiento después de cualquier modificación de una fórmula durante la ejecución.

Editor de gestor de fórmulas, ficha General

Parámetro	Descripción
Recipe management in the plc	Si el gestor de fórmulas no se necesita en el controlador porque no se tienen que gestionar fórmulas durante el tiempo de ejecución de la aplicación, se puede desactivar esta opción. De este modo se evita la descarga del gestor. La actualización automática del archivo de fórmulas sólo se puede llevar a cabo después de que se haya realizado la descarga. Para descargar el gestor de fórmulas en el controlador, seleccione esta opción.

Definición de fórmula

Descripción general

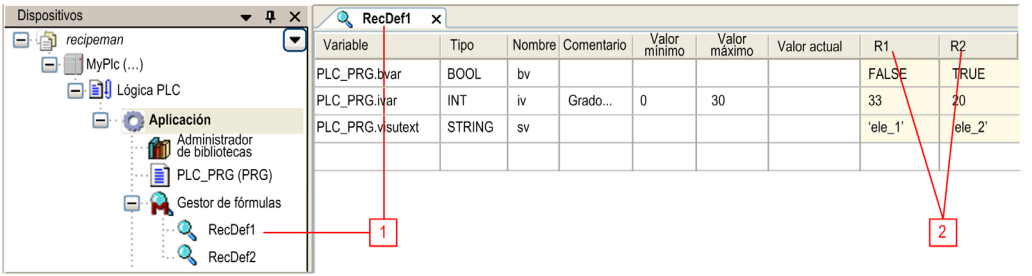
El gestor de fórmulas *(véase página 504)* gestiona una o varias definiciones de fórmulas. Una definición de fórmula contiene una lista de variables y una o varias fórmulas (conjuntos de valores) para estas variables. Utilizando distintas fórmulas, puede asignar otro conjunto de valores a un conjunto de variables en el controlador de una sola vez. No existe ninguna limitación sobre el número de definiciones de fórmulas, fórmulas y variables por fórmula.

Definición de fórmula

Puede añadir uno o varios objetos **Definición de fórmula** a un nodo **Gestor de fórmulas** en el árbol **Herramientas**. Para hacerlo, haga clic en el signo más de color verde del nodo **Gestor de fórmulas** y ejecute el comando **Definición de fórmula...**

Haga doble clic en el nodo para ver y editar definiciones de fórmula, incluidas las fórmulas específicas, en una vista de editor independiente.

Vista del editor de definiciones de fórmulas



1 Nombre de la definición de fórmula
2 Nombres de fórmulas

La ventana del editor llevará como título el nombre de la definición de fórmula.

Parámetro	Descripción
Variable	En una tabla, puede introducir varias variables de proyecto para las que desee definir una o varias fórmulas. Con este objetivo, puede utilizar el comando **Insertar variable** cuando el cursor esté en cualquier campo de cualquier línea. Como alternativa, puede hacer doble clic en un campo **Variable** o puede seleccionarlo y pulsar la barra espaciadora para entrar en la modalidad de editor. Introduzca el nombre válido de una variable de proyecto, por ejemplo `plc_prg.ivar`. Haga clic en el botón **...** para abrir el asistente Accesibilidad.
Tipo	El campo **Tipo** se rellena automáticamente. Opcionalmente, se puede definir un **Nombre** simbólico.
Nombre	Puede definir un **Nombre** simbólico.

Parámetro	Descripción
Comentario	Introduzca información adicional, como la unidad del valor registrada en la variable.
Valor mínimo y **Valor máximo**	Opcionalmente, puede especificar estos valores que deben ser admisibles para escribirse en esta variable.
Valor actual	Este valor se supervisa en modalidad online.
Guardar automáticamente las modificaciones en las fórmulas	Se recomienda activar esta opción, porque afecta al comportamiento habitual de la gestión de fórmulas: los archivos de almacenamiento se actualizarán inmediatamente cuando se produzca cualquier modificación de una fórmula durante el tiempo de ejecución. Tenga en cuenta que la opción sólo puede ser efectiva mientras el gestor de fórmulas esté disponible en el controlador.

Puede eliminar una variable (línea) de la tabla pulsando la tecla SUPR cuando una de sus celdas esté seleccionada. Puede seleccionar varias líneas manteniendo pulsada la tecla CTRL mientras selecciona celdas. Puede copiar las líneas seleccionadas mediante copiar y pegar. El comando Pegar inserta las líneas copiadas encima de la línea seleccionada actualmente. Al hacerlo, los valores de las fórmulas se insertarán en la columna de la fórmula coincidente, si está disponible.

Para añadir una fórmula a la definición de fórmula, ejecute el comando *(véase SoMachine, Comandos de menú, Ayuda en línea)* **Insertar fórmula** cuando el foco esté en la vista del editor. Para cada fórmula, se creará una columna propia, que llevará como título el nombre de la fórmula (ejemplo: **R1** y **R2** en la figura anterior).

En modalidad online, una fórmula se puede modificar mediante un elemento de visualización configurado adecuadamente (comando de ejecución de configuración de entrada) o utilizando los métodos apropiados del bloque de funciones `RecipeManCommands` de Recipe_Management.library.

Son compatibles los métodos siguientes:
- `ReadRecipe`: Los valores de la variable actual se adoptan en la fórmula.
- `WriteRecipe`: La fórmula se escribe en las variables.
- `SaveRecipe`: La fórmula se almacena en un archivo de fórmula estándar.
- `LoadRecipe`: La fórmula se carga de un archivo de fórmula estándar.
- `CreateRecipe`: Se crea una fórmula nueva en la definición de la fórmula.
- `DeleteRecipe`: Se elimina una fórmula existente de una definición de fórmula.

Consulte en los párrafos siguientes cómo se comportan las fórmulas en los estados online específicos. Se recomienda establecer la opción **Guardar automáticamente las modificaciones en las fórmulas** (para obtener el comportamiento habitual de una gestión de fórmulas).

Fórmula

Puede añadir o eliminar una fórmula offline u online. Si está en modalidad offline, utilice los comandos **Insertar fórmula** *(véase SoMachine, Comandos de menú, Ayuda en línea)* y **Quitar fórmula** *(véase SoMachine, Comandos de menú, Ayuda en línea)* en el editor de gestor de fórmulas. En modalidad online, configure una entrada en un elemento de visualización debidamente configurado o utilice los métodos apropiados del bloque de funciones `RecipeManCommands` de Recipe_Management.library.

Al añadir una fórmula, se añadirá una columna adicional detrás de la columna situada más a la derecha, que llevará como título el nombre de la fórmula (consulte la figura de la vista del editor de fórmulas). Los campos de una columna de fórmula se pueden rellenar con valores apropiados. De este modo, para el mismo conjunto de variables se pueden preparar diferentes conjuntos de valores en las fórmulas específicas.

Uso de fórmulas en modalidad online

Las fórmulas se pueden gestionar (crear, leer, escribir, guardar, cargar, eliminar) utilizando los métodos del bloque de funciones `RecipeManCommands`, proporcionado por la biblioteca Recipe_Management.libray, en el código de la aplicación, o por medio de entradas en elementos de visualización.

Gestión de fórmulas en modalidad online si la opción **Guardar automáticamente las modificaciones en las fórmulas** está activada:

Acciones	Fórmulas definidas en el proyecto	Fórmulas creadas durante el tiempo de ejecución
Online Reset Warm **Online Reset Cold** **Descarga**	Las fórmulas de todas las definiciones de fórmulas se establecen con los valores del proyecto actual.	Las fórmulas creadas dinámicamente permanecen inalteradas.
Online Reset Origin	La aplicación se eliminará del controlador. Si se realiza una descarga nueva posteriormente, las fórmulas se restaurarán como en un **Online Reset Warm**.	
Apague y reinicie el controlador	Después del reinicio, las fórmulas se vuelven a cargar de los archivos creados automáticamente. Por lo tanto, se restaurará el estado anterior al apagado.	
Cambio en línea	Los valores de la fórmula permanecen inalterados. Durante el tiempo de ejecución, una fórmula sólo puede ser modificada por los comandos del bloque de funciones `RecipeManCommands`.	
Detención	Al detenerse/iniciarse el controlador, las fórmulas permanecen inalteradas.	

Gestión de fórmulas en modalidad online si la opción **Guardar automáticamente las modificaciones en las fórmulas** NO está activada:

Acciones	Fórmulas definidas en el proyecto	Fórmulas creadas durante el tiempo de ejecución
Online Reset Warm **Online Reset Cold** **Descargar**	Las fórmulas de todas las definiciones de fórmulas se establecen con los valores del proyecto actual. No obstante, sólo se establecen en la memoria. Para almacenar la fórmula en un archivo, el comando Guardar se debe utilizar explícitamente.	Las fórmulas creadas dinámicamente se pierden.
Online Reset Origin	La aplicación se eliminará del controlador. Si se realiza una nueva descarga posteriormente, las fórmulas se restaurarán.	Las fórmulas creadas dinámicamente se pierden.
Apague y reinicie el controlador	Después del reinicio, las fórmulas se vuelven a cargar de los valores iniciales que se habían creado al descargar los valores del proyecto. Por lo tanto, el estado anterior al apagado no se restaurará.	
Cambio en línea	Los valores de la fórmula permanecen inalterados. Durante el tiempo de ejecución, una fórmula sólo puede ser modificada por los comandos del bloque de funciones `RecipeManCommands`.	
Detención	Al detenerse/iniciarse el controlador, las fórmulas permanecen inalteradas.	

Información adicional:

- Por lo que respecta al almacenamiento de fórmulas en archivos, que se vuelven a cargar al reiniciarse la aplicación, consulte la descripción de *Editor de gestor de fórmulas, ficha Guardado (véase página 506)*.
- Para obtener una descripción de los métodos *(véase página 513)* `RecipeManCommands` específicos, consulte la documentación en la biblioteca.
- Para conocer la configuración de entrada apropiada de un elemento de visualización, consulte su página de ayuda (categoría **Entrada** →**Ejecutar comando**).

Se pueden realizar las acciones siguientes sobre las fórmulas:

Acciones	Descripción
Crear fórmula (= **Insertar fórmula**)	Se creará una nueva fórmula en la definición de fórmula especificada.
Leer fórmula	Los valores actuales de las variables de la definición de fórmula especificada se leerán del controlador y se escribirán en la fórmula especificada. Esto significa que los valores se almacenarán implícitamente (en un archivo en el controlador). También se supervisarán inmediatamente en la tabla de definición de fórmulas en el **Gestor de fórmulas**. En otras palabras, la fórmula gestionada en el **Gestor de fórmulas** se actualiza con los valores reales del controlador.
Escribir fórmula	Los valores de la fórmula en cuestión, tal como aparecen en el gestor de fórmulas, se escribirán en las variables en el controlador.
Guardar fórmula	Los valores de la fórmula especificada se escribirán en un archivo con la extensión **.txtrecipe*, el nombre del cual deberá definir. Con este propósito, se abrirá el cuadro de diálogo estándar para guardar un archivo en el sistema de archivos local. **NOTA:** Es posible que los archivos de fórmula utilizados implícitamente, necesarios como búfer para la lectura y escritura de los valores de fórmula, no se sobrescriban. Esto significa que el nombre del nuevo archivo de fórmula deberá ser diferente de <nombre de la fórmula>.<nombre de la definición de fórmula>.txtrecipe.
Cargar fórmula	La fórmula que se ha almacenado en un archivo (consulte la descripción de **Guardar fórmula**) se puede volver a cargar desde este archivo. El cuadro de diálogo estándar para seleccionar un archivo se abrirá con este fin. El filtro se establece automáticamente en la extensión **.txtrecipe*. Después de volver a cargar el archivo, los valores de la fórmula se actualizarán según corresponda en el gestor de fórmulas.
Borrar la fórmula (= **Quitar fórmula**)	La fórmula especificada se eliminará de la definición de fórmula.
Cambiar fórmula	El valor de las variables del proyecto se puede cambiar. Con una acción de escribir fórmula, las variables del proyecto apropiadas se escriben con los nuevos valores.

Comandos RecipeMan

Descripción general

Al llamar a un comando de fórmula, se realizará un acceso a datos internos. Según el tipo de dispositivo, esta acción tardará unos pocos milisegundos. Verifique que estas llamadas no las realice una tarea con un watchdog configurado o una tarea en tiempo real. Tenga en cuenta que la opción **Guardar automáticamente las modificaciones en las fórmulas** también realizará un acceso a archivos con cada cambio de la fórmula. Desactive esta opción si la aplicación activa el almacenamiento de la fórmula.

Valores de retorno

Estos son los valores de retorno posibles para comandos de fórmulas:

Valor de retorno	Descripción
ERR_NO_RECIPE_MANAGER_SET	No hay gestores de fórmulas disponibles en el controlador.
ERR_RECIPE_DEFINITION_NOT_FOUND	La definición de fórmula no existe.
ERR_RECIPE_ALREADY_EXIST	La fórmula ya existe en la definición de fórmula.
ERR_RECIPE_NOT_FOUND	La fórmula no existe en la definición de fórmula.
ERR_RECIPE_FILE_NOT_FOUND	El archivo de fórmula no existe.
ERR_RECIPE_MISMATCH	El contenido del archivo de fórmula no coincide con la fórmula actual. **NOTA:** Este valor de retorno sólo se genera cuando el tipo de almacenamiento es textual y cuando un nombre de variable del archivo no coincide con el nombre de variable en la definición de la fórmula. El archivo de fórmula no se carga.
ERR_RECIPE_SAVE_ERR	El archivo de fórmula no se ha podido abrir con acceso de escritura.
ERR_FAILED	La operación no se ha realizado correctamente.
ERR_OK	La operación se ha realizado correctamente.

CreateRecipe

Este método crea una fórmula nueva en la definición de fórmula especificada y, posteriormente, lee los valores del controlador actuales en la nueva fórmula. Al final, la nueva fórmula se almacena en el archivo predeterminado.

Parámetro	Descripción
`RecipeDefinitionName`:	Nombre de la definición de fórmula
`RecipeName`:	Nombre de la fórmula

Valores de retorno *(véase página 513)*:
`ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_ALREADY_EXIST, ERR_FAILED, ERR_OK`

CreateRecipeNoSave

Este método crea una fórmula nueva en la definición de fórmula especificada y, posteriormente, lee los valores del controlador actuales en la nueva fórmula.

Parámetro	Descripción
`RecipeDefinitionName`:	Nombre de la definición de fórmula
`RecipeName`:	Nombre de la fórmula

Valores de retorno *(véase página 513)*:
`ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_NOT_FOUND, ERR_FAILED, ERR_OK`

DeleteRecipe

Este método elimina una fórmula de la definición de fórmula.

Parámetro	Descripción
`RecipeDefinitionName`:	Nombre de la definición de fórmula
`RecipeName`:	Nombre de la fórmula

Valores de retorno *(véase página 513)*:
`ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_NOT_FOUND, ERR_FAILED, ERR_OK`

DeleteRecipeFile

Este método elimina el archivo de fórmula estándar de una fórmula.

Parámetro	Descripción
RecipeDefinitionName:	Nombre de la definición de fórmula
RecipeName:	Nombre de la fórmula

Valores de retorno:
*(véase página 513)*ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_NOT_FOUND, ERR_RECIPE_FILE_NOT_FOUND, ERR_OK

LoadAndWriteRecipe

Este método carga una fórmula del archivo de fórmula estándar y, después, escribe la fórmula en las variables del controlador.

Parámetro	Descripción
RecipeDefinitionName:	Nombre de la definición de fórmula
RecipeName:	Nombre de la fórmula

Valores de retorno *(véase página 513)*:
ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_NOT_FOUND, ERR_RECIPE_FILE_NOT_FOUND, ERR_RECIPE_MISMATCH, ERR_FAILED, ERR_OK

LoadFromAndWriteRecipe

Este método carga una fórmula del archivo de fórmula especificado y, después, escribe la fórmula en las variables del controlador.

Parámetro	Descripción
RecipeDefinitionName:	Nombre de la definición de fórmula
RecipeName:	Nombre de la fórmula
FileName:	Nombre del archivo

Valores de retorno *(véase página 513)*:
ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_NOT_FOUND, ERR_RECIPE_FILE_NOT_FOUND, ERR_RECIPE_MISMATCH, ERR_FAILED, ERR_OK

LoadRecipe

Este método carga una fórmula del archivo de fórmula estándar. El nombre del archivo de fórmula estándar es <fórmula>.<definición de fórmula>.<extensión de fórmula>.

Parámetro	Descripción
RecipeDefinitionName:	Nombre de la definición de fórmula
RecipeName:	Nombre de la fórmula

Valores de retorno *(véase página 513)*:
ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_NOT_FOUND, ERR_RECIPE_FILE_NOT_FOUND, ERR_RECIPE_MISMATCH, ERR_FAILED, ERR_OK

ReadAndSaveRecipe

Este método lee los valores del controlador actuales en la fórmula y, posteriormente, almacena la fórmula en el archivo de fórmula estándar.

Parámetro	Descripción
RecipeDefinitionName:	Nombre de la definición de fórmula
RecipeName:	Nombre de la fórmula

Valores de retorno *(véase página 513)*:
ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_NOT_FOUND, ERR_RECIPE_SAVE_ERR, ERR_FAILED, ERR_OK

ReadAndSaveRecipeAs

Este método lee los valores del controlador actuales en la fórmula y, posteriormente, almacena la fórmula en el archivo de fórmula especificado. El contenido de un archivo existente se sobrescribiría.

Parámetro	Descripción
RecipeDefinitionName:	Nombre de la definición de fórmula
RecipeName:	Nombre de la fórmula
FileName:	Nombre del archivo

Valores de retorno *(véase página 513)*:
ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_NOT_FOUND, ERR_RECIPE_SAVE_ERR, ERR_FAILED, ERR_OK

SaveRecipe

Este método almacena la fórmula en el archivo de fórmula estándar. El contenido de un archivo existente se sobrescribiría. El nombre del archivo de fórmula estándar es <fórmula>.<definición de fórmula>.<extensión de fórmula>.

Parámetro	Descripción
RecipeDefinitionName:	Nombre de la definición de fórmula
RecipeName:	Nombre de la fórmula

Valores de retorno *(véase página 513)*:
ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_NOT_FOUND, ERR_RECIPE_SAVE_ERR, ERR_FAILED, ERR_OK

ReadRecipe

Este método lee los valores del controlador actuales en la fórmula.

Parámetro	Descripción
RecipeDefinitionName:	Nombre de la definición de fórmula
RecipeName:	Nombre de la fórmula

Valores de retorno *(véase página 513)*:
ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_NOT_FOUND, ERR_FAILED, ERR_OK

WriteRecipe

Este método escribe la fórmula en las variables del controlador.

Parámetro	Descripción
RecipeDefinitionName:	Nombre de la definición de fórmula
RecipeName:	Nombre de la fórmula

Valores de retorno *(véase página 513)*:
ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_NOT_FOUND, ERR_FAILED, ERR_OK

ReloadRecipes

Este método vuelve a cargar la lista de fórmulas desde el sistema de archivos.

Parámetro	Descripción
`RecipeDefinitionName`:	Nombre de la definición de fórmula

Valores de retorno *(véase página 513)*:
`ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_FAILED, ERR_OK`

GetRecipeCount

Este método devuelve la cantidad de fórmulas de la definición de fórmula correspondiente.

Parámetro	Descripción
`RecipeDefinitionName`:	Nombre de la definición de fórmula

Valores de retorno: `-1` : Si no se encuentra la definición de fórmula.

GetRecipeNames

Este método devuelve los nombres de las fórmulas de la definición de fórmula correspondiente.

Parámetro	Descripción
`RecipeDefinitionName`:	Nombre de la definición de fórmula
`pStrings` :	Las cadenas en las que deben almacenarse los valores de las fórmulas
`iSize` :	El tamaño de una matriz de cadena
`iStartIndex` :	El índice de inicio Se puede utilizar para una función de desplazamiento

Valores de retorno *(véase página 513)*:
`ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_FAILED, ERR_OK`

Ejemplo:

Hay 50 fórmulas. Para crear una tabla que muestre 10 nombres de fórmulas a la vez, defina una matriz de cadena:

```
strArr: ARRAY[0..9] OF STRING;
```

Correspondiente a `iStartIndex`, los nombres de las fórmulas se pueden leer de un área específica.

```
iStartIndex := 0;
```

Se devuelven los nombres 0...9.

```
iStartIndex := 20;
```

Se devuelven los nombres 20...29. En este ejemplo:

```
iSize := 10;
```

GetRecipeValues

Este método devuelve los valores de las variables de fórmula de la fórmula correspondiente.

Parámetro	Descripción
`RecipeDefinitionName`:	Nombre de la definición de fórmula
`RecipeName`	Nombre de la fórmula
`pStrings` :	Las cadenas en las que deben almacenarse los valores de las fórmulas
`iSize` :	El tamaño de una matriz de cadena
`iStartIndex` :	El índice de inicio Se puede utilizar para una función de desplazamiento
`iStringLength` :	La longitud de la cadena de la matriz

Valores de retorno *(véase página 513)*:
`ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_NOT_FOUND, ERR_FAILED, ERR_OK`

Ejemplo:

Hay 50 fórmulas. Para crear una tabla que muestre 10 nombres de fórmulas a la vez, defina una matriz de cadena:

```
strArr: ARRAY[0..9] OF STRING;
```

Correspondiente a `iStartIndex`, los nombres de las fórmulas se pueden leer de un área específica.

```
iStartIndex := 0;
```

Se devuelven los valores 0...9.

```
iStartIndex := 20;
```

Se devuelven los valores 20...29. En este ejemplo:

```
iStringLength := 80;
iSize := 10;
```

GetRecipeVariableNames

Este método devuelve el nombre de la variable de la fórmula correspondiente.

Parámetro	Descripción
`RecipeDefinitionName`:	Nombre de la definición de fórmula
`RecipeName`	Nombre de la fórmula
`pStrings` :	Las cadenas en las que deben almacenarse los valores de las fórmulas
`iSize` :	El tamaño de una matriz de cadena
`iStartIndex` :	El índice de inicio Se puede utilizar para una función de desplazamiento

Valores de retorno *(véase página 513)*:
`ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_NOT_FOUND, ERR_FAILED, ERR_OK`

Ejemplo:

Hay 50 fórmulas. Para crear una tabla que muestre 10 nombres de fórmulas a la vez, defina una matriz de cadena:

```
strArr: ARRAY[0..9] OF STRING;
```

Correspondiente a `iStartIndex`, los nombres de las fórmulas se pueden leer de un área específica.

```
iStartIndex := 0;
```

Se devuelven los nombres 0...9.

```
iStartIndex := 20;
```

Se devuelven los nombres 20...29. En este ejemplo:

```
iSize := 10;
```

SetRecipeValues

Este método establece los valores de la fórmula en la fórmula correspondiente.

Parámetro	Descripción
`RecipeDefinitionName`:	Nombre de la definición de fórmula
`RecipeName`	Nombre de la fórmula
`pStrings` :	Las cadenas en las que deben almacenarse los valores de las fórmulas
`iSize` :	El tamaño de una matriz de cadena
`iStartIndex` :	El índice de inicio Se puede utilizar para una función de desplazamiento

Valores de retorno *(véase página 513)*:
`ERR_NO_RECIPE_MANAGER_SET, ERR_RECIPE_DEFINITION_NOT_FOUND, ERR_RECIPE_NOT_FOUND, ERR_FAILED, ERR_OK`

Ejemplo:

Hay 50 fórmulas. Para crear una tabla que muestre 10 nombres de fórmulas a la vez, defina una matriz de cadena:

```
strArr: ARRAY[0..9] OF STRING;
```

Correspondiente a `iStartIndex`, los nombres de las fórmulas se pueden leer de un área específica.

```
iStartIndex := 0;
```

Se establecen los valores 0...9.

```
iStartIndex := 20;
```

Se establecen los valores 20...29. En este ejemplo:

```
iStringLength := 80;
iSize := 10;
```

GetLastError

Este método devuelve el último error detectado de las operaciones anteriores.

Valores de retorno: *(véase página 513)*`ERR_NO_RECIPE_MANAGER_SET, ERR_OK`

ResetLastError

Este método restablece el último error detectado.

Valores de retorno: *(véase página 513)*`ERR_NO_RECIPE_MANAGER_SET, ERR_OK`

Capítulo 24
Editor de traza

Contenido de este capítulo

Este capítulo contiene las siguientes secciones:

Sección 24.1
Objeto de traza

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Información básica sobre trazas

Funcionalidad de traza

La funcionalidad de traza le permite capturar la progresión de los valores de las variables en el controlador en un determinado tiempo, de forma similar al osciloscopio de muestras digitales. Asimismo, puede definir un desencadenador para controlar la captura de datos con señales de entrada (desencadenadoras). Los valores de las variables de traza se escriben constantemente en un SoMachinebúfer de un tamaño especificado. Pueden observarse en forma de gráfico bidimensional representado como una función de tiempo.

Método para trazar datos

La traza de datos en el control se realiza de 2 formas diferentes:

- Desde el código IEC generado por el objeto de traza y descargado en el controlador mediante una aplicación de traza secundaria.
- O bien, en el componente `CmpTraceMgr` (también llamado **administrador de trazas**).

La captura de datos que se realice dependerá de una entrada en los valores de destino (**traza** → **administrador de trazas**).

El administrador de trazas tiene una funcionalidad avanzada. Le permite:

- Configurar y trazar parámetros del sistema de control, como la curva de temperatura de la CPU o de la batería. Para obtener más información, consulte Configuración de variable *(véase página 531)* y Configuración de registro (desencadenador) *(véase página 535)*.
- Leer trazas de dispositivos, como la traza de la corriente eléctrica de un variador. Para obtener más información, consulte la descripción del comando **Cargar traza**. *(véase SoMachine, Comandos de menú, Ayuda en línea)*
- Trazar variables del sistema de otros componentes del sistema.

Además, está disponible el comando *(véase SoMachine, Comandos de menú, Ayuda en línea)* adicional **Lista en línea**.

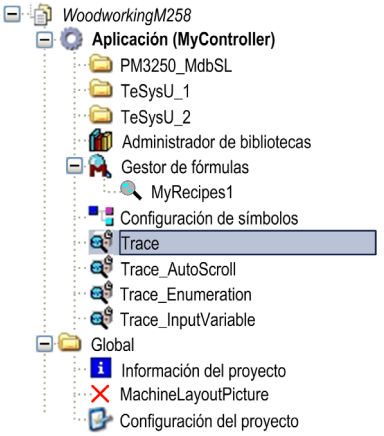
NOTA:

- Si se utiliza una traza en la visualización, los parámetros de dispositivo no pueden trazarse ni utilizarse para el desencadenador.
- El nivel de desencadenador no puede establecerse en una expresión IEC, sólo se admiten literales y constantes.
- La condición de registro no puede establecerse en una expresión IEC de tipo BOOL; sólo se admiten variables.
- Si se traza o utiliza una propiedad para el desencadenador, debe anotarse con attribute monitoring *(véase página 637)* en la declaración IEC.

Configuración

Configure los datos de traza así como los valores de visualización de los datos de traza en **Configuración**. Proporciona comandos para acceder a los cuadros de diálogo de configuración. Varias variables pueden trazarse y mostrarse al mismo tiempo y en diferentes vistas como el modo multicanal. Registre trazas de variables con diferentes configuraciones de desencadenador en su propio objeto de traza. Puede crear cualquier número de objetos de traza.

Árbol **Herramientas** con varios objetos de traza



Los comandos para modificar los valores de la visualización se describen en el párrafo *Características* *(véase página 528)*. Hay disponibles funcionalidades de zoom y un cursor, así como comandos para ejecutar la traza para que el gráfico se pueda comprimir o ampliar.

Para integrar la lectura de una traza en una visualización, utilice el elemento de visualización **Traza**.

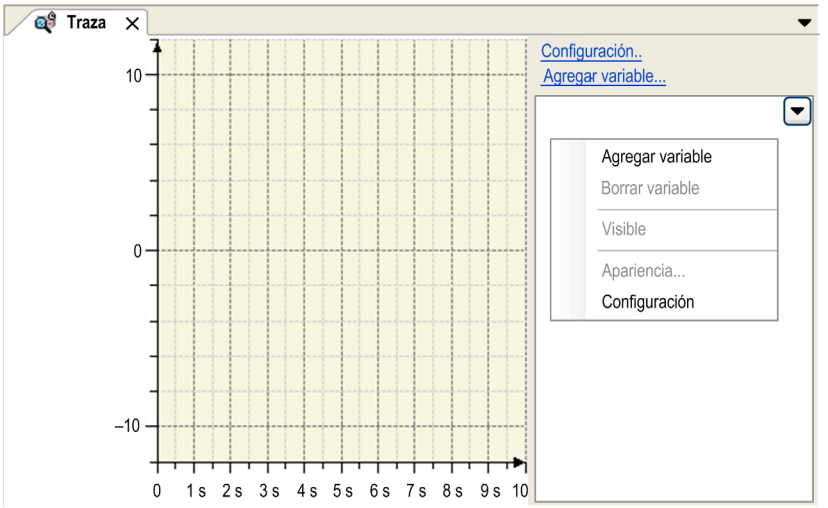
Creación de un objeto de traza

Descripción general

Para insertar un objeto de traza en el árbol **Herramientas**, seleccione el nodo **Aplicación**, haga clic en el signo más de color verde y ejecute el comando **Añadir otros objetos →Traza...** Haga doble clic en el nodo **Traza** en el árbol **Herramientas** para abrir el editor de trazas.

Configuración

Traza recién creada con menú contextual



Una traza contiene al menos una variable que está muestreada.

En el área del árbol de traza, en la parte derecha de la ventana, se muestran las variables de traza configuradas. De forma predeterminada, las variables de traza se muestran con su ruta de instancia completa.

Seleccione la casilla **Ocultar rutas de instancia** para ocultar la ruta de instancia. Para mostrar esta casilla, haga clic en el botón de flecha de la esquina superior derecha del área del árbol de traza.

Para configurar o cambiar la configuración de traza, utilice los comandos del menú contextual en el área del árbol de traza:

- **Agregar variable...**: abre el cuadro de diálogo **Configuración de traza** con **Configuración de variable** *(véase página 531)*.
- **Borrar variable**: borra la variable seleccionada. Sólo está disponible si existe al menos una variable de traza.
- **Visible**: este comando hace que sea visible la variable seleccionada. Sólo está disponible si existe al menos una variable de traza.

- **Apariencia...**: se abre el cuadro de diálogo **Modificar configuración de pantalla** *(véase página 541)*. Le permite configurar la apariencia de la gráfica y el sistema de coordenadas. Este comando está atenuado hasta que se carga una configuración.
- **Configuración...**: abre el cuadro de diálogo **Configuración de traza** con **Configuración de registro** *(véase página 535)*.

Características

Para ejecutar la traza, utilice los comandos siguientes:
- **Agregar variable** *(véase SoMachine, Comandos de menú, Ayuda en línea)*
- **Descarga de traza** *(véase SoMachine, Comandos de menú, Ayuda en línea)*
- **Iniciar/Detener traza** *(véase SoMachine, Comandos de menú, Ayuda en línea)*
- **Restablecer desencadenador** *(véase SoMachine, Comandos de menú, Ayuda en línea)*

Para personalizar la vista de las gráficas, utilice los comandos siguientes:
- **Cursor** *(véase SoMachine, Comandos de menú, Ayuda en línea)*
- **Zoom** *(véase SoMachine, Comandos de menú, Ayuda en línea)*
- **Restablecer vista** *(véase SoMachine, Comandos de menú, Ayuda en línea)*
- **Escalar automáticamente** *(véase SoMachine, Comandos de menú, Ayuda en línea)*
- **Comprimir** *(véase SoMachine, Comandos de menú, Ayuda en línea)*
- **Estirar** *(véase SoMachine, Comandos de menú, Ayuda en línea)*
- **Multicanal** *(véase SoMachine, Comandos de menú, Ayuda en línea)*
- Para obtener más información, consulte el capítulo *(véase página 549) Operaciones de teclado para diagramas de traza*.

Para acceder a trazas almacenadas en el sistema de tiempo de ejecución, utilice los comandos siguientes:
- **Lista en línea** *(véase SoMachine, Comandos de menú, Ayuda en línea)*
- **Cargar traza** *(véase SoMachine, Comandos de menú, Ayuda en línea)*

Para acceder a trazas almacenadas en el disco, utilice los comandos siguientes:
- **Guardar la traza...** *(véase SoMachine, Comandos de menú, Ayuda en línea)*
- **Cargar la traza...** *(véase SoMachine, Comandos de menú, Ayuda en línea)*
- **Exportar configuración de traza simbólica** *(véase SoMachine, Comandos de menú, Ayuda en línea)*

Pasos iniciales

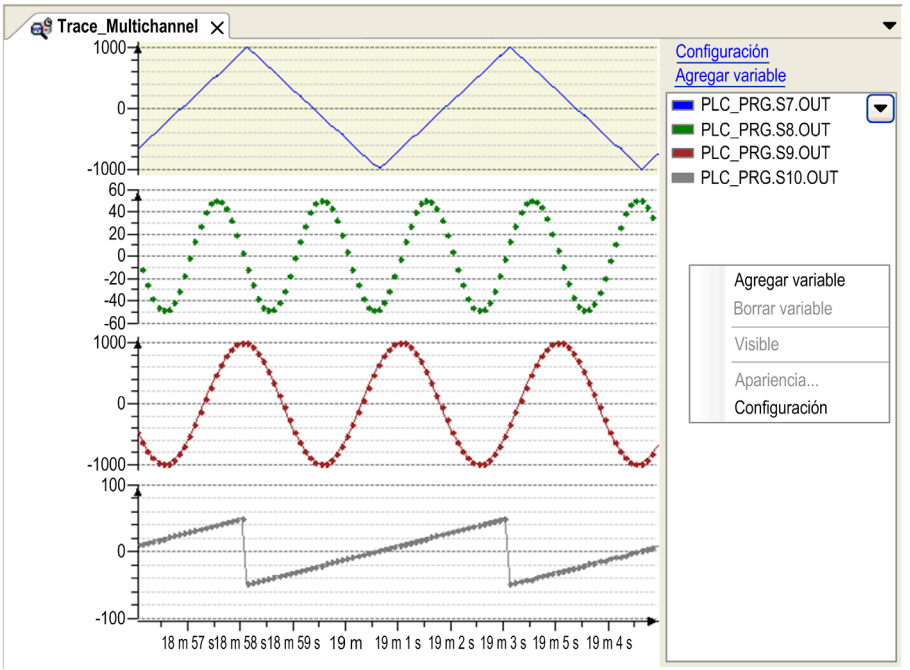
Para iniciar la traza en modalidad online, descargue la configuración de traza en el controlador ejecutando el comando **Descarga de traza**. Las gráficas de las variables de traza se mostrarán en la ventana del editor de trazas, donde puede almacenarlas en un archivo externo. Este archivo puede volver a cargarse en el editor. Consulte también el capítulo *Editor de traza en modalidad online* *(véase página 548)*.

Paso	Acción
1	Iniciar sesión y ejecutar la aplicación asociada. **Resultado:** La aplicación se ejecuta en el controlador.
2	Descarga de traza **Resultado:** Las gráficas de traza se muestran inmediatamente en función de la configuración de traza.
3	Organizar las gráficas de traza, almacenar los datos de traza, iniciar/detener la traza.

Ejemplo

El editor de trazas muestra un ejemplo de traza en modalidad online. Se han seleccionado cuatro variables para su visualización en el árbol de variables, en la parte derecha del cuadro de diálogo.

Traza en modalidad online



Sección 24.2
Configuración traza

Contenido de esta sección

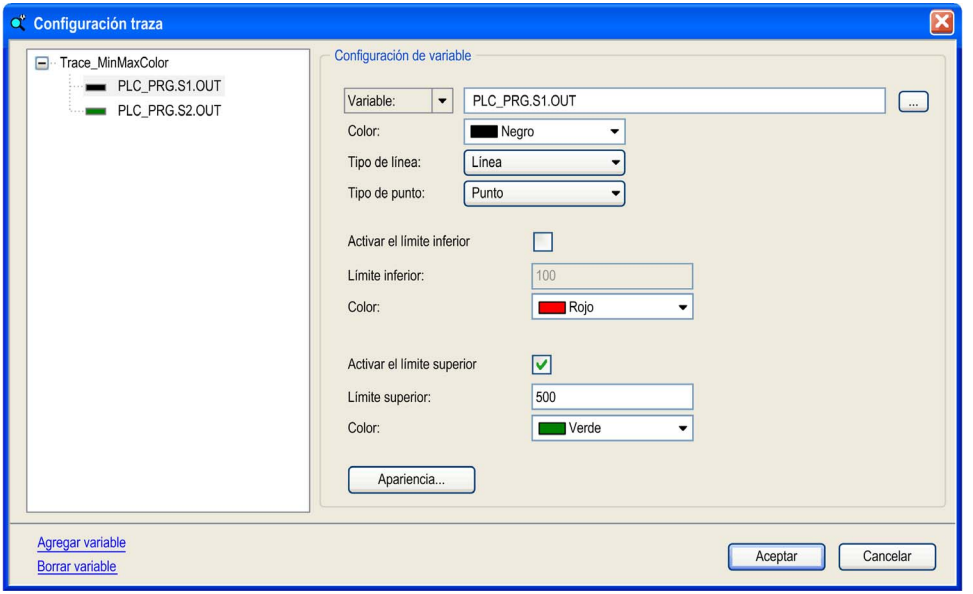
Esta sección contiene los siguientes apartados:

Configuración de variables

Descripción general

Se abre el cuadro de diálogo **Configuración traza** con **Configuración de variable** al seleccionar una variable de traza en el árbol de trazas. Permite configurar qué variables se deben trazar y cómo se muestran.

Cuadro de diálogo **Configuración traza** con **Configuración de variable**



Las variables de traza se muestran en la parte izquierda de la ventana en una estructura de árbol. El nodo superior lleva como título el nombre de la traza.

Adición y eliminación de una variable de traza

Para añadir una variable al árbol de trazas o eliminar una, utilice los comandos que hay debajo del árbol de trazas:

Comando	Descripción
Añadir variable	crea una entrada anónima en el árbol de trazas. En la parte derecha del cuadro de diálogo, los valores de la nueva variable están listos para la configuración.
Eliminar variable	elimina la variable seleccionada con la configuración asociada.

Definición y modificación de la configuración de la variable

Para seleccionar la configuración de la variable, seleccione la variable deseada en el árbol de trazas. Se mostrará la configuración actual en la parte derecha de la ventana de configuración de la traza. Para modificar la configuración de la variable más tarde, seleccionar la entrada de la variable en el árbol de trazas y vuelva a utilizar el cuadro de diálogo **Configuración de variable**.

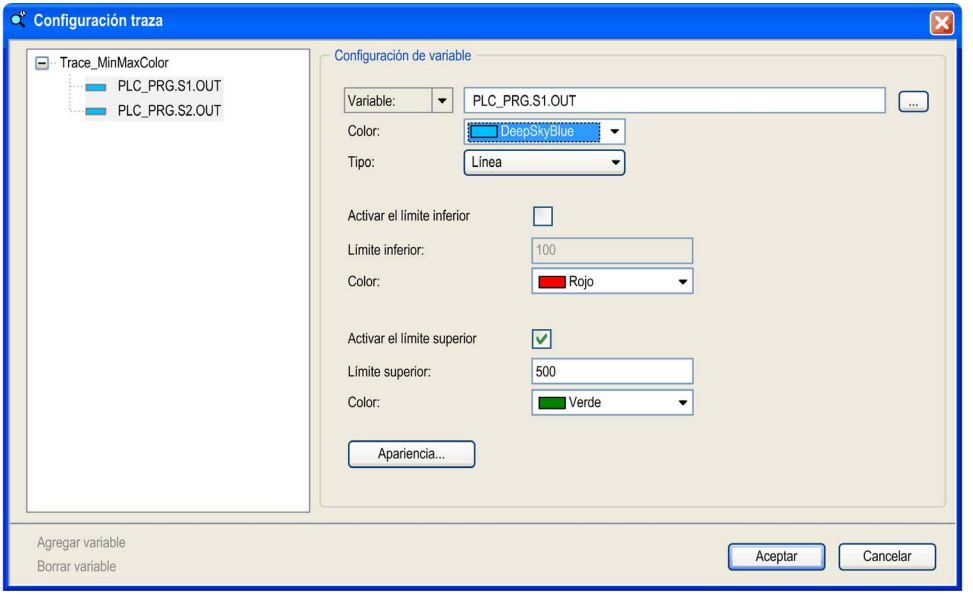
Parámetro		Descripción
Variable		Introduzca el nombre (ruta) de la señal para especificar la señal que se trazará. Una señal válida es una variable IEC, una propiedad, una referencia, el contenido de un puntero o un elemento de matriz de la aplicación. Los tipos permitidos son todos los tipos básicos de IEC, excepto STRING, WSTRING o ARRAY. También se permiten enumeraciones cuyo tipo básico no sea STRING, WSTRING ni ARRAY. Haga clic en el botón **...** para abrir el asistente Accesibilidad, que le permite obtener una entrada válida. Los controladores que admiten el trazado de parámetros proporcionan una lista si se hace clic en el parámetro **Variable:**. Si desea trazar un parámetro de dispositivo, seleccione el elemento **Parámetro** en esta lista. Entonces podrá encontrar uno con la ayuda del asistente Accesibilidad. Edite o verifique la configuración actual de la variable. Los parámetros del dispositivo sólo se admiten si se utiliza el componente `CmpTraceMgr`. Si se utilizan parámetros de dispositivo para variables de traza (o de desencadenador), no se puede activar la opción **Crear el módulo traza para visualización**. **NOTA:** Si se utiliza `CmpTraceMgr` para el trazado, una Propiedad *(véase página 183)* que se utiliza como variable de traza (o de desencadenador) debe obtener el atributo del compilador Attribute Monitoring *(véase página 637)*.
Color		Seleccione un color de la lista de selección de colores en el que aparecerá la curva de traza para la variable.
Tipo de línea		Especifique cómo se conectarán las muestras en el gráfico. Utilice **Línea** para grandes volúmenes de datos. También es el valor predeterminado.
	Línea	Las muestran están conectadas a una línea (valor predeterminado).
	Paso	Las muestras se conectan en forma de escalera. De este modo, hay una línea horizontal hacia la marca de hora de la siguiente muestra seguida de una línea vertical al valor de la siguiente muestra.
	Ninguno	Las muestras no están conectadas.
Tipo de punto		Especifique cómo se dibujarán los propios valores en el gráfico.
	Punto	Las muestras se dibujan como puntos (valor predeterminado).
	Cruz	Las muestras se dibujan como cruces.
	Ninguno	Las muestras no se muestran.

Parámetro	Descripción
Activar el límite inferior	Si esta opción está activada, el gráfico de traza se mostrará en el color definido en **Color** en cuanto la variable supere el valor definido en **Límite inferior**.
Límite inferior	Si el valor de la variable introducido aquí ha caído por debajo del límite y la opción **Activar el límite inferior** está activa, los valores de la curva cambian en el color especificado siguiente.
Color	Valor del color para el límite inferior activado.
Activar el límite superior	Si esta opción está activada, el gráfico de traza se mostrará en el color definido en **Color** en cuanto la variable supere el valor definido en **Límite superior**.
Límite superior	Si el valor de la variable introducido aquí se supera y la opción **Activar el límite superior** está activa, los valores de la curva cambian en el color especificado siguiente.
Color	Valor del color para el límite superior activado.
Apariencia...	Abre el cuadro de diálogo **Representación eje Y**. Permite configurar la visualización de la ventana de traza para el eje Y configurado actualmente (colores y comportamiento del desplazamiento) para cada variable en su propio estilo. Esta configuración se utiliza cuando el diagrama de traza se muestra en la vista multicanal.

Selección múltiple de variables

Utilizando los métodos abreviados de teclado MAYÚS + clic con el ratón o CTRL + clic con el ratón, puede seleccionar varias variables para editarlas. Luego, los cambios en el cuadro de diálogo **Configuración de variable** se aplican a todas las variables seleccionadas. Se puede hacer lo mismo con MAYÚS + FLECHA ARRIBA/ABAJO o CTRL + FLECHA ARRIBA/ABAJO.

Selección múltiple en el cuadro de diálogo **Configuración traza**



Configuración de registro

Descripción general

El cuadro de diálogo **Configuración traza** con **Configuración de registro** se abre si se ejecuta el comando **Configuración...** o si se hace doble clic en el nombre de la traza en la parte superior del árbol de trazas. El comando de configuración también está disponible en el menú contextual del árbol de trazas en la parte derecha de la ventana del editor de trazas principal.

NOTA: La configuración especificada en el cuadro de diálogo **Configuración traza** con **Configuración de registro** es válida para todas las variables del gráfico de traza.

Información básica sobre desencadenadores

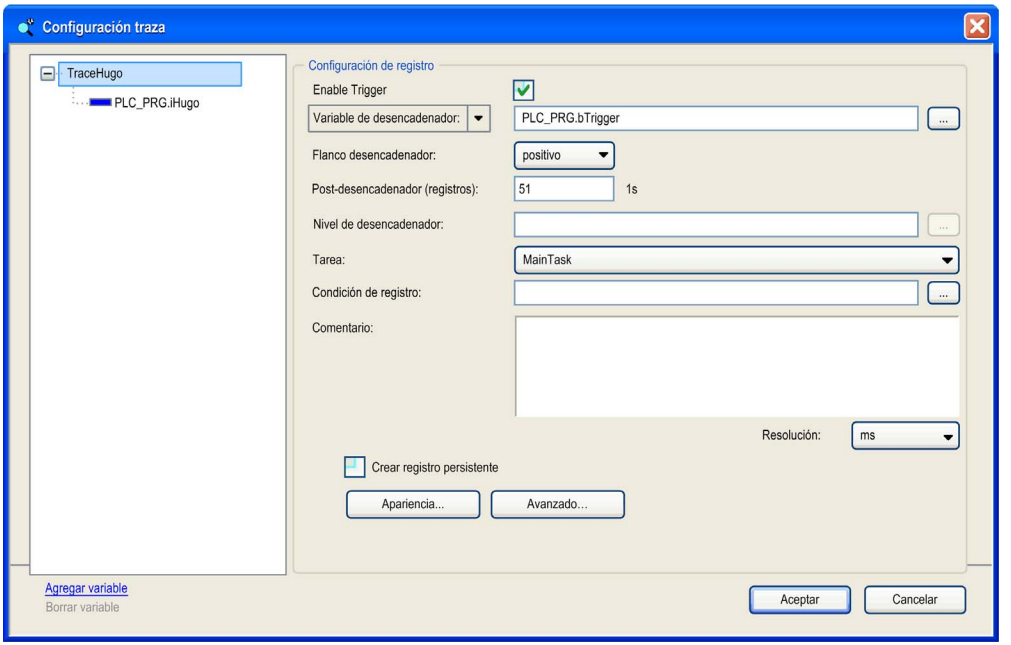
En la mayoría de los casos, no es deseable que el trazado y la visualización de las señales de entrada se inicie en momentos aleatorios, como por ejemplo inmediatamente después de la medición anterior o cuando el usuario pulsa el botón de inicio. La mayor parte del tiempo es preferible que el trazado se realice cuando se active un desencadenador para el número de registros configurado (postdesencadenador). Este se conoce como activación y se debe definir aquí.

Los modos siguientes se utilizan para activar señales de entrada:

- configuración de una variable de desencadenador
- configuración de una condición de registro
- o ambos.

Realización y modificación de la configuración de registro (desencadenador)

Cuadro de diálogo **Configuración traza** con **Configuración de registro**



Parámetro	Descripción
Enable Trigger	Seleccione la casilla para habilitar el sistema desencadenador. Se puede activar o desactivar independientemente de los ajustes más bajos. Si el sistema desencadenador está deshabilitado, la traza es de funcionamiento libre.

Parámetro		Descripción
Variable de desencadenador		Asigne una variable. Especifique qué señal se utilizará como desencadenador introduciendo el nombre (y la ruta) de la señal. Una señal de desencadenador válida es una variable IEC, una propiedad, una referencia, un puntero, un elemento de matriz de la aplicación o una expresión. Los tipos permitidos son todos los tipos básicos de IEC, excepto STRING, WSTRING o ARRAY. También se permiten enumeraciones cuyo tipo básico no sea STRING, WSTRING ni ARRAY. El contenido de un puntero no es una señal válida. Haga clic en el botón **...** para abrir el asistente Accesibilidad, que le permite obtener una entrada válida. Los controladores que admiten el uso de parámetros de dispositivo como desencadenadores proporcionan una lista si se hace clic en el parámetro **Variable de desencadenador:**. Si desea utilizar un parámetro de dispositivo como desencadenador, seleccione el elemento **Parámetro desencadenador** en esta lista. Abra **Accesibilidad** con el botón **...** y seleccione parámetros trazables. En **Elementos**, se enumeran los parámetros disponibles en el sistema. También se pueden escribir los nombres de los parámetros directamente o mediante el método de copiar y pegar (desde otra configuración) en el campo de texto. Los parámetros del dispositivo sólo se admiten si se utiliza el administrador de trazas. **NOTA:** Si se utiliza `CmpTraceMgr` para el trazado, una Propiedad *(véase página 183)* que se utiliza como variable de traza (o de desencadenador) debe obtener el atributo del compilador Attribute Monitoring *(véase página 637)*.
Flanco desencadenador		–
	positivo	Evento desencadenador en el flanco ascendente de la variable de desencadenador booleana. O en cuanto una ejecución ascendente alcance el valor definido por **Nivel de desencadenador** para una variable de desencadenador analógico.
	negativo	Evento desencadenador en el flanco descendente de la variable de desencadenador booleana. O en cuanto una ejecución descendente alcance el valor definido por **Nivel de desencadenador** para una variable de desencadenador analógico.
	ambos	Evento desencadenador en las condiciones descritas para **positivo** y **negativo**.
Post-desencadenador		Introduzca un número de registros por señal de traza que se registran después de que se active el desencadenador. Valor predeterminado: 50. Rango: 0...(2^{32}-1).

Parámetro	Descripción
Nivel de desencadenador	Introduzca un valor en el que se active el desencadenador. Con **Flanco desencadenador**, puede especificar si se activa en el flanco ascendente o descendente de la variable de desencadenador. Se debe establecer si y sólo si una variable analógica (variable con tipo numérico, como por ejemplo LREAL o INT) se utiliza como variable de desencadenador. Introduzca directamente un valor. Se permite una constante GVL o un valor ENUM si su tipo se puede convertir al de la variable de desencadenador. Si se utiliza código IEC, entonces también se puede introducir una expresión IEC arbitraria de un tipo que se pueda convertir al de la variable de desencadenador. Valor predeterminado: – (vacío).
Tarea	En la lista de tareas disponibles, seleccione la tarea donde se realiza la captura de las señales de entrada.
Condición de registro	Si desea iniciar el registro mediante una condición, introduzca aquí una variable. Si la traza se inicia antes, por ejemplo pulsando el botón de inicio, y la variable asignada aquí pasa a ser TRUE, la captura de datos se inicia y se mostrará el gráfico trazado. Si se utiliza `CmpTraceMgr`, la condición de registro debe ser una variable de tipo BOOL o de acceso de bit. El contenido de un puntero no es una entrada válida. Las propiedades también se admiten. Si se utiliza el código IEC, también se puede introducir una expresión IEC arbitraria de tipo BOOL.
Comentario	Introduzca un texto de comentario referente el registro actual.
Resolución	Introduzca una resolución de la marca de hora de la traza en ms o µs. Para cada señal capturada, se almacenan pares de valor y marca de hora y se transmiten al sistema de programación. Las marcas de hora transmitidas son relativas y hacen referencia al inicio del trazado. Si la tarea de traza tiene un tiempo de ciclo de 1 ms o menos, se recomienda una marca de hora con resolución en µs. Esta opción sólo es posible cuando hay un administrador de trazas disponible en el controlador.
Crear registro persistente	Establezca esta opción si la configuración de traza y el último contenido del búfer de traza del RTS se deben almacenar permanentemente en el dispositivo de destino. Esta opción sólo es posible cuando un administrador de trazas está trazando en el controlador.
Apariencia...	Abre el cuadro de diálogo **Representación eje Y** *(véase página 546)*. Permite configurar la visualización de la ventana de traza para el registro configurado actualmente, como ejes, colores y comportamiento del desplazamiento.
Avanzado...	Haga clic en este botón para abrir el cuadro de diálogo *(véase página 539)* **Configuraciones de traza avanzadas**. Permite configurar ajustes adicionales para el desencadenador de traza.

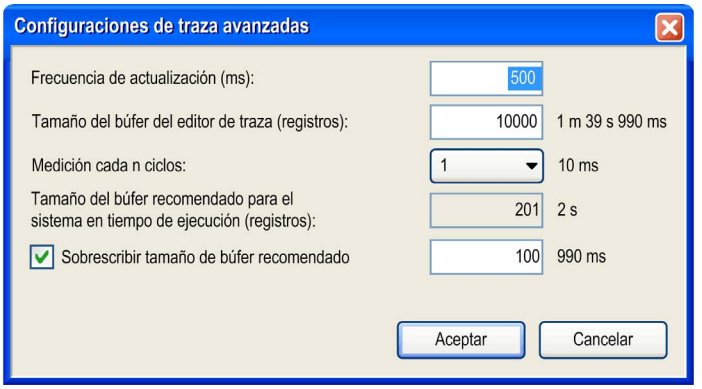
NOTA: Si desea capturar y visualizar una señal de traza con una base de tiempo diferente, debe realizar una configuración de registro en un objeto de traza separado.

Configuraciones de traza avanzadas

Descripción general

El cuadro de diálogo **Configuraciones de traza avanzadas** se abre al hacer clic en el botón **Avanzado…** del cuadro de diálogo **Configuración traza** con **Configuración de registro**.

Cuadro de diálogo **Configuraciones de traza avanzadas**



Para cada valor que figura en los registros o ciclos, el intervalo de tiempo asociado se muestra a su lado (por ejemplo, **1m39s990ms**). El intervalo de tiempo para los búferes comprende todo el búfer. Si la tarea no está establecida, y no es cíclica ni una tarea de sistema, entonces se desconoce el tiempo de ciclo de la tarea. En este caso, el periodo no se puede calcular y no se mostrará en la parte derecha.

Descripción de los parámetros

Parámetro	Descripción
Frecuencia de actualización (ms)	Con este intervalo de tiempo, los pares de datos capturados (valor con marca de hora) del trazado se almacenan en el búfer del editor de traza. Rango: 150 ms...10000 ms Valor predeterminado: 500 ms Si se utiliza el administrador de trazas, los pares de datos se transfieren con este intervalo de tiempo del sistema de tiempo de ejecución al sistema de programación. Si no se utiliza el administrador de trazas, entonces los datos se transmiten cada 200 ms al sistema de programación.
Tamaño del búfer del editor de traza (registros)	Introduzca el tamaño de búfer de la traza en las muestras (registros). Este búfer tiene que ser mayor o igual a dos veces el tamaño de lo que pueda ser el búfer para el sistema de tiempo de ejecución. Rango: $1...10^7$
Medición cada n ciclos	Seleccione en la lista un intervalo de tiempo (en ciclos de tarea) para la captura de la señal de entrada. Valor predeterminado y mínimo: 1 (equivale a una medición en cada ciclo)
Tamaño del búfer recomendado para el sistema en tiempo de ejecución (registros)	Se muestra el número de muestras recomendado del búfer del sistema de tiempo de ejecución para cada señal de traza. Este valor se calcula en función del tiempo de ciclo de tarea, el tiempo de actualización y el valor de **Medición cada n ciclos**. Esto significa que se asigna un búfer para cada variable de traza. **NOTA:** El tamaño del búfer se proporciona en las muestras y se crea un búfer para cada variable de traza.
Sobrescribir tamaño de búfer recomendado	Si se selecciona esta opción, se utiliza el valor introducido aquí, en lugar del valor predeterminado para el tamaño del búfer de tiempo de ejecución. Ejemplo: Rango: 10...el tamaño de búfer del editor de traza

Modificar configuración de pantalla

Descripción general

El cuadro de diálogo **Modificar configuración de pantalla** se abre al hacer clic en el botón **Apariencia...** del cuadro de diálogo **Configuración traza** con **Configuración de registro**.

Los siguientes ajustes definen la apariencia del sistema de coordenadas y los ejes X/Y. Los ajustes del eje Y se utilizan cuando se visualiza el diagrama de traza en la vista de un solo canal. En la vista multicanal, se utilizan los ajustes especificados en el cuadro de diálogo **Representación eje Y**.

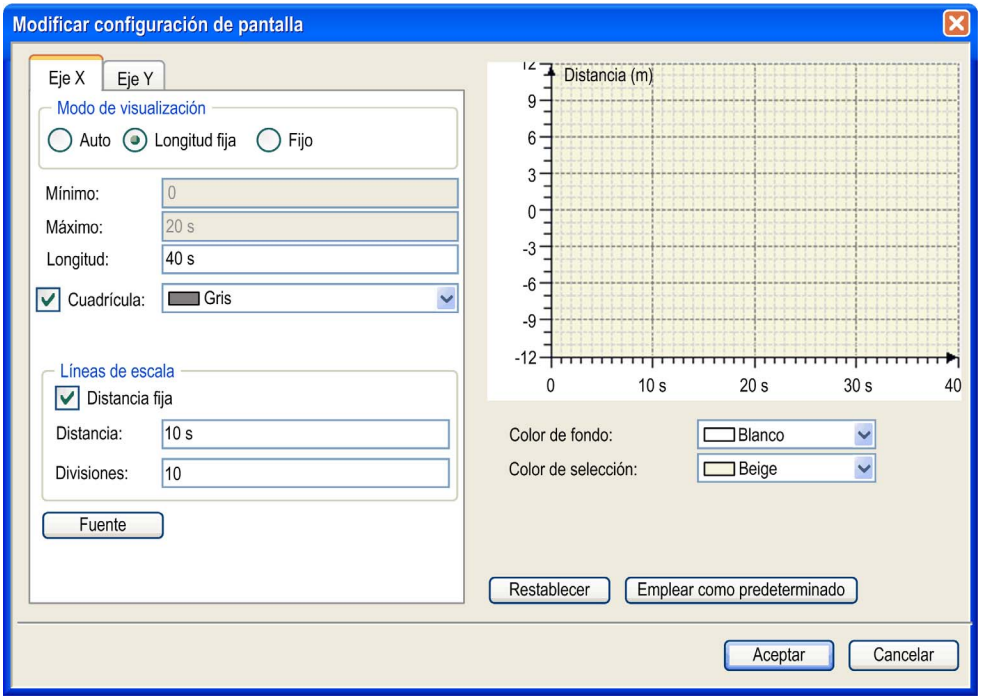
Los ajustes para los ejes X/Y (especificados en la parte de la izquierda) y el sistema de coordenadas se aplican de inmediato en el sistema de coordenadas de la derecha.

Puede gestionar los ajustes con los siguientes botones:

Botón	Descripción
Restablecer	Con este comando, se restablece el valor predeterminado de la apariencia.
Emplear como predeterminado	Con este comando, la apariencia actual se establece como apariencia predeterminada. Se utilizará al configurar una nueva traza o variable.

Ficha Eje X

Ficha **Eje X** del cuadro de diálogo **Modificar configuración de pantalla**



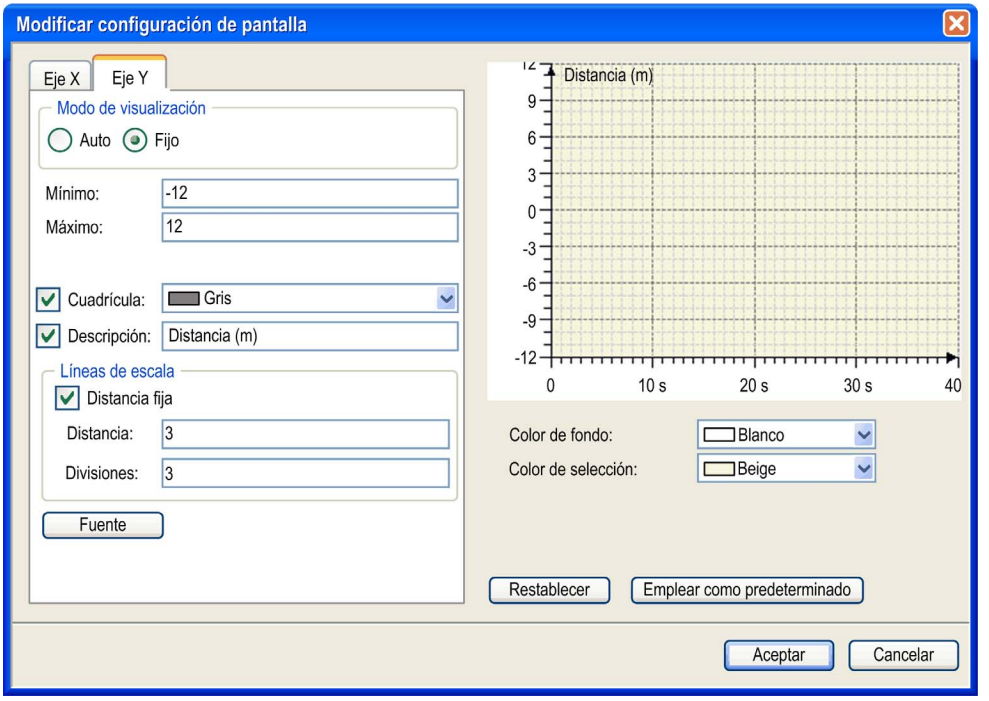
Parámetro		Descripción
Modo de visualización		Seleccione el modo de visualización.
	Auto	Si esta opción está activada, el eje de tiempo se escala automáticamente según el contenido del búfer del editor de traza *(véase página 539)*. El contenido actual del búfer de traza se visualiza en el diagrama. No tienen que establecerse más entradas.
	Longitud fija	Si esta opción está activada, el intervalo mostrado del eje de tiempo tiene una longitud fija. Defina esta longitud con el parámetro **Longitud**. La escala también se ajusta a la longitud. La gráfica se desplaza automáticamente a un rango visible. Por consiguiente, en el diagrama se muestra un intervalo de tiempo con la longitud configurada y los datos asociados más recientes. Pero sólo contiene tantos valores como valores registrados. En cuanto se registran nuevos datos, la pantalla se desplaza.
	Fijo	Si esta opción está activada, el intervalo mostrado del eje X se define con el valor mínimo y el valor máximo.
Mínimo		Este valor define el valor mínimo mostrado del eje de tiempo.[1]

Parámetro		Descripción
Máximo		Este valor define el valor máximo mostrado del eje de tiempo.[1]
Longitud		Este valor define la longitud del intervalo mostrado del eje de tiempo.[1]
Cuadrícula		Si esta opción está activada, se muestra una cuadrícula. Seleccione en la lista el color de las líneas de la cuadrícula.
Líneas de escala		–
	Distancia fija	Seleccione esta casilla para escalar el eje a determinadas distancias.
	Distancia	Introduzca una distancia positiva. El eje X es un eje de tiempo cuyo valor predeterminado es 1s.
	Divisiones	Introduzca un número razonable de divisiones para cada distancia.
Fuente		Abre el cuadro de diálogo estándar para la definición de la fuente de la pantalla de traza.

[1] Las entradas de tiempo no requieren el prefijo # como sucede en el código IEC. Algunas entradas de tiempo posibles son, por ejemplo, **2s**, **1ms** o **1m20s14ms**. Utilice us para los microsegundos. Por ejemplo: **122ms500us**. Los valores de utilidad también dependen de la resolución del eje de tiempo.

Ficha Eje Y

Ficha **Eje Y** del cuadro de diálogo **Modificar configuración de pantalla**



Parámetro		Descripción
Modo de visualización		Seleccione el modo de visualización.
	Automático	Si esta opción está activada, el eje Y se escala automáticamente según los valores capturados. No tienen que establecerse más entradas.
	Fija	Si esta opción está activada, la sección mostrada del eje Y se define con el valor mínimo y el valor máximo.
Mínimo		Este valor define el valor mínimo mostrado del eje Y.
Máximo		Este valor define el valor máximo mostrado del eje Y.
Cuadrícula		Si esta opción está activada, se muestra una cuadrícula. Seleccione en la lista el color de las líneas de la cuadrícula.
Descripción		Si esta opción está activada, el eje Y se etiqueta con el texto introducido en el campo que tiene al lado.

Parámetro		Descripción
Líneas de escala		–
	Distancia fija	Seleccione esta casilla para escalar el eje a determinadas distancias.
	Distance	Introduzca una distancia positiva. Valor predeterminado: 1
	Divisiones	Escriba un número, de 1 a 10, de subdivisiones para cada distancia.
Fuente		Haga clic en este botón para abrir el cuadro de diálogo estándar para la definición de la fuente de la pantalla de traza.

Parámetros del sistema de coordenadas

Parámetro	Descripción
Color de fondo	Seleccione en la lista el color de fondo del sistema de coordenadas. Este color se utiliza mientras el diagrama no esté seleccionado en la ventana de traza.
Color de selección	Seleccione en la lista el color de fondo del sistema de coordenadas. Este color se utiliza mientras el diagrama esté seleccionado en la ventana de traza.

Representación eje Y

Descripción general

El cuadro de diálogo **Representación eje Y** se abre al hacer clic en el botón **Apariencia...** del cuadro de diálogo **Configuración traza** con **Configuración de variable**.

La configuración siguiente define la apariencia del eje Y. Se utiliza cuando el diagrama de traza se muestra en la vista multicanal.

Cuadro de diálogo **Representación eje Y**

Parámetro		Descripción
Modo de visualización		Seleccione el modo de visualización.
	Automático	Si esta opción está activada, el eje Y se escala automáticamente según los valores capturados. No tienen que establecerse más entradas.
	Fijo	Si esta opción está activada, la sección mostrada del eje Y tiene que estar definida por el valor mínimo y el valor máximo.
Mínimo		Este valor define el valor mínimo mostrado del eje Y.
Máximo		Este valor define el valor máximo mostrado del eje Y.
Cuadrícula		Si esta opción está activada, se muestra una cuadrícula. Seleccione en la lista el color de las líneas de la cuadrícula.
Descripción		Si esta opción está activada, el eje Y se etiqueta con el texto introducido en el campo que tiene al lado.
Líneas de escala		–
	Distancia fija	Seleccione esta casilla para escalar el eje a determinadas distancias.

Parámetro		Descripción
	Distance	Introduzca una distancia positiva. Valor predeterminado: 1
	Divisiones	Introduzca un número de subdivisiones para cada distancia.
Fuente		Abre el cuadro de diálogo estándar para la definición de la fuente de la pantalla de traza.

Sección 24.3
Editor de traza en modalidad online

Editor de traza en modalidad online

Descripción general

Si en el dispositivo se ejecuta una traza, esto se indica en el cuadro de diálogo de trazas **Lista en línea** *(véase SoMachine, Comandos de menú, Ayuda en línea)*.

Descarga de traza

Para iniciar la traza en modalidad online, descargue de forma explícita la traza en controlador con el **Traza** →**Descarga de traza** menu command *(véase SoMachine, Comandos de menú, Ayuda en línea)* mientras la aplicación esté conectada. Los gráficos de las señales de traza se mostrarán en la ventana del editor de traza.

Al realizar inicios y cierres de sesión en la misma aplicación, las trazas se ejecutan sin una nueva descarga.

Si se cambia el código de aplicación, entonces la modalidad de inicio de sesión determinará lo que sucede con las trazas:

- **Iniciar sesión con modificación en línea** o **Iniciar sesión sin modificaciones**: Las trazas seguirán ejecutándose.
- **Iniciar sesión con descarga**: Las trazas del controlador se eliminan, por lo que será necesario volver a descargarlas.

Cambio en línea de la configuración de gráfico de traza

El cuadro de diálogo **Configuración traza** con **Configuración de registro** y el cuadro de diálogo **Configuración traza** con **Configuración de variable** están disponibles en modalidad online y pueden realizarse bastantes cambios en la configuración de traza mientras la traza se está ejecutando. Si esto no es posible, por ejemplo cuando se cambia el nombre de la señal de traza, la traza se detiene, por lo que será necesaria una nueva descarga.

Navegación en línea del gráfico de traza

El rango visualizado de los valores de variable de traza capturados no sólo depende de la configuración de traza. También puede reordenarse mediante las funcionalidades de desplazamiento y zoom que están disponibles en el menú **Traza**, la barra de herramientas o mediante el uso de métodos abreviados. Para obtener información sobre cómo navegar en el diagrama de traza, consulte el capítulo *Métodos abreviados de teclado (véase página 549)*.

Sección 24.4
Operaciones de teclado para diagramas de traza

Métodos abreviados de teclado

Descripción general

En la tabla siguiente se describen las acciones de teclado y ratón:

Acciones	Mediante el teclado	Mediante el ratón
Desplazar el gráfico de traza horizontalmente a lo largo del eje de tiempo.	Ningún cursor de traza: ● FLECHA IZQUIERDA/DERECHA ● Con distancias mayores: CTRL + FLECHA IZQUIERDA/DERECHA 1 o 2 cursores de traza: ● ALT + FLECHA IZQUIERDA/DERECHA ● Con distancias mayores: CTRL + ALT + FLECHA IZQUIERDA/DERECHA	Desplace el gráfico mediante el método de arrastrar y soltar. Esto aparece indicado con una vista diferente del cursor del ratón.
Desplazar el gráfico de traza verticalmente a lo largo del eje Y.	FLECHA ARRIBA/ABAJO Con distancias mayores: CTRL + FLECHA ARRIBA/ABAJO	Utilice CTRL + arrastrar y soltar.
Hacer zoom a un rectángulo (ventana) seleccionado con el ratón.	–	Utilice el comando **Zoom** *(véase SoMachine, Comandos de menú, Ayuda en línea)*.
Desplazar el cursor negro de traza.	FLECHA IZQUIERDA/DERECHA Con distancias mayores: CTRL + FLECHA IZQUIERDA/DERECHA	Haga clic en el triángulo negro del cursor de traza, arrástrelo a lo largo del eje X hasta soltarlo en el punto deseado.
Desplazar el cursor gris de traza.	MAYÚS + FLECHA IZQUIERDA/DERECHA Con distancias mayores: CTRL + MAYÚS + FLECHA IZQUIERDA/DERECHA	Haga clic en el triángulo gris del cursor de traza, arrástrelo a lo largo del eje X hasta soltarlo en el punto deseado.
Comprimir el eje de tiempo. En modalidad multicanal, los ejes de tiempo de todos los diagramas están comprimidos.	–	Utilice la rueda del ratón. O bien utilice el comando **Comprimir** *(véase SoMachine, Comandos de menú, Ayuda en línea)*.

Acciones	Mediante el teclado	Mediante el ratón
Estirar el eje de tiempo. En modalidad multicanal, los ejes de tiempo de todos los diagramas están estirados.	+	Utilice la rueda del ratón. O bien utilice el comando **Estirar** *(véase SoMachine, Comandos de menú, Ayuda en línea)*.
Comprimir el eje Y. En modalidad multicanal, el eje Y de los diagramas seleccionados aparece comprimido.	CTRL + –	CTRL + rueda del ratón
Estirar el eje Y. En modalidad multicanal, el eje Y de los diagramas seleccionados aparece estirado.	CTRL + +	CTRL + rueda del ratón
Seleccionar el diagrama siguiente en modalidad multicanal.	TABULADOR	Haga clic en un diagrama que no esté seleccionado para seleccionarlo.

Capítulo 25
Editor de configuración de símbolos

Contenido de este capítulo

Este capítulo contiene los siguiente apartados:

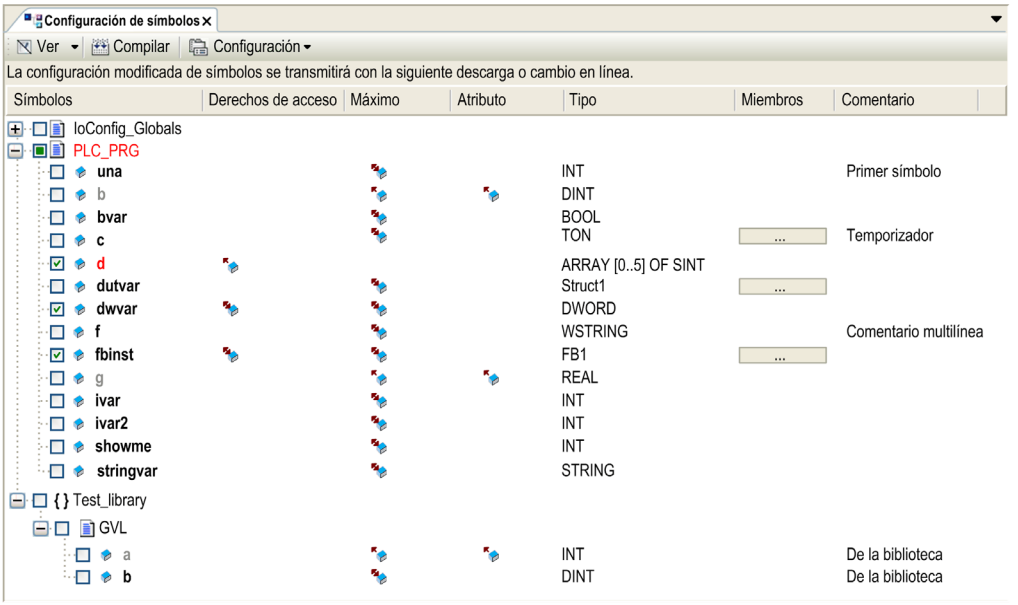
Editor de configuración de símbolos

Descripción general

La funcionalidad de configuración de símbolos permite crear descripciones de símbolo. Puede accederse a los símbolos y las variables que representan mediante aplicaciones externas, comoVijeo-Designer o un servidor OPC.

Para configurar símbolos para una aplicación, haga doble clic en el nodo **Configuración de símbolos** del árbol **Herramientas**. Se abre la vista del editor de configuración de símbolos.

Editor de configuración de símbolos



El editor contiene una tabla. En función del filtro establecido, muestra las variables disponibles o sólo las ya seleccionadas para la configuración de símbolos. Para ello, se incluyen las POU o bibliotecas correspondientes en la columna **Símbolos**. Puede expandirlas para ver las variables específicas.

Descripción de los elementos

El botón **Ver** disponible en la barra de herramientas sobre la tabla permite establecer los siguientes filtros para reducir el número de variables visualizadas:

Filtro	Descripción
No configurados de proyecto	Se muestran incluso las variables todavía no añadidas a la configuración de símbolos, pero disponibles para ello en el proyecto.
No configurados de bibliotecas	También se muestran las variables de bibliotecas todavía no añadidas a la configuración de símbolos, pero disponibles para ello en el proyecto.
Símbolos exportados vía atributo	Este ajuste surte efecto cuando se visualizan sólo las variables ya configuradas (consulte los dos filtros descritos anteriormente). Tiene como efecto que también se incluyen las variables ya seleccionadas para obtener símbolos mediante {`attribute 'symbol' := 'read'`} en la declaración. Estos símbolos se muestran atenuados. La columna **Atributo** muestra el derecho de acceso establecido actualmente para la variable por parte del pragma. Consulte la siguiente descripción de derecho de acceso.

Para modificar los derechos de acceso de un elemento seleccionado, haga clic en el símbolo de la columna **Derechos de acceso**. Con cada clic del ratón se alternará el símbolo de las siguientes definiciones: lectura+escritura  , sólo escritura  , sólo lectura  .

La columna **Máximo** muestra qué derecho puede establecerse como máximo.

La columna **Comentario** muestra los comentarios que se hayan añadido en la declaración de la variable.

Con la propiedad de POU **Ligar siempre**, un objeto no compilado se puede reinterpretar y descargar en el controlador. Así, todas las variables declaradas en este objeto estarán disponibles. Además, puede utilizar el pragma {attribute linkalways} *(véase página 635)* para que las variables no compiladas estén disponibles en la configuración de símbolos.

Las variables configuradas para exportar pero que actualmente no son válidas en la aplicación (por ejemplo, porque su declaración se ha eliminado) se mostrarán en color rojo. Esto también se aplica a la POU o nombre de biblioteca pertinente.

Para obtener símbolos para una variable de un tipo de datos estructurado, al igual que para otras variables, primero debe activar el elemento en la columna **Símbolos**. Con ello, se exportarán los símbolos al archivo de símbolos para todos los miembros de la estructura. Esto puede generar un gran número de entradas en el archivo de símbolos, aunque no todas sean necesarias. También puede seleccionar determinadas variables de miembro. Puede hacerlo en el cuadro de diálogo **Configuración de símbolos para tipo de datos**. Haga clic en el botón **...** de la columna **Miembros** para abrir este cuadro de diálogo. En caso de tipos anidados, este cuadro de diálogo tendrá un botón para abrir otro cuadro de diálogo de configuración de símbolos de tipo de datos.

La vista de editor se actualiza automáticamente cuando se ejecuta una compilación. La barra de herramientas proporciona el botón **Compilar** para un acceso rápido.

El botón **Configuración** de la barra de herramientas permite activar la opción **Incluir comentarios en XML**. Con ello, los comentarios asignados a variables también se exportarán al archivo de símbolos.

De forma predeterminada, un archivo de símbolos se crea con una ejecución de generación de código. Este archivo se transfiere al dispositivo con la siguiente descarga. Si desea crear el archivo sin realizar una descarga, utilice el comando **Crear código**, que de forma predeterminada está disponible en el menú **Compilar**.

NOTA: Las variables de una lista de variables globales (GVL) sólo estarán disponibles en la configuración de símbolos si como mínimo una de ellas se utiliza en el código de programación.

Si se utiliza un dispositivo que admita un archivo de aplicación independiente para la configuración de símbolos (consulte también el capítulo Configuración de símbolos *(véase página 555)*), en la barra de herramientas se mostrará el botón **Descargar**. Puede utilizarlo para iniciar de inmediato una nueva descarga del archivo *<nombre de aplicación>._Symbols* si la configuración de símbolos se ha modificado en modalidad online.

Configuración de símbolos

Descripción general

La configuración de símbolos se utiliza para crear símbolos con derechos de acceso específicos. Permiten acceder a las variables del proyecto externamente, por ejemplo mediante Vijeo-Designer. La descripción de los símbolos estará disponible en un archivo XML (archivo de símbolos) en el directorio del proyecto. Se descargará al controlador junto con la aplicación.

Información sobre los símbolos

Los símbolos definidos para una aplicación se exportan a un archivo XML en el directorio del proyecto (archivo de símbolos) cuando la aplicación se descarga al controlador. El nombre de este archivo sigue esta sintaxis:

<nombre de proyecto>.<nombre de dispositivo>.<nombre de aplicación>.xml

Ejemplo: *proj_xy.PLC1.application.xml*

NOTA: Si la descarga al controlador no es posible, puede crear el archivo de configuración de símbolos mediante la ejecución del comando **Crear código**.

Después, la información sobre los símbolos se descarga al controlador con la aplicación. En función de la descripción del dispositivo, se incluirá en la aplicación o se generará una aplicación secundaria por separado. También aparecerá en la lista con el nombre <*nombre de aplicación*>*._Symbols* en la vista Aplicaciones del editor de dispositivos *(véase página 127)*.

Si la configuración de símbolos se ha modificado en modalidad online, puede volver a cargarla en el controlador haciendo clic en el botón **Descargar** en la ventana del editor *(véase página 553)*.

Por ejemplo, en lo que se refiere al número máximo de aplicaciones en un controlador, la aplicación de símbolos debe gestionarse como una aplicación normal.

Adición de una configuración de símbolos

Requisitos previos

Las variables que se intercambiarán entre el controlador y (varios) dispositivos HMI mediante el protocolo *(véase SoMachine, Introduction)* transparente de SoMachine se deben publicar en el controlador mediante la **Configuración de símbolos**. A continuación, estarán disponibles como variables de SoMachine en Vijeo-Designer.

Definición de una configuración de símbolos

Para que la funcionalidad de configuración de símbolos esté disponible, añada el objeto de configuración de símbolos a la aplicación en el árbol **Herramientas**, como se describe en el párrafo *Apertura de la configuración de símbolos*. Esta acción incluirá de forma automática la biblioteca IECVarAccess.library en el **Administrador de bibliotecas**.

En el editor de configuración de símbolos *(véase página 552)* o mediante pragmas (attribute symbol *(véase página 655)*) puede definir las variables para que se exporten como símbolos, que se añadirán a la declaración de las variables.

NOTA: Las variables de una lista de variables globales (GVL) sólo estarán disponibles en la configuración de símbolos si como mínimo una de ellas se utiliza en el código de programación.

Otra posibilidad es utilizar el editor SFC: puede definir indicadores de elemento creados de forma implícita en las propiedades del elemento *(véase página 361)* para exportarlos a la configuración de símbolos.

El nombre de un símbolo creado por la configuración de símbolos está compuesto conforme a la sintaxis siguiente:

<nombre de aplicación>.<nombre de POU>.<nombre de variable>

Ejemplos:

MyApplication.PLC_PRG.a

MyApplication.GVL.a

Para acceder a la variable, defina completamente el nombre del símbolo.

Apertura de la configuración de símbolos

Para abrir **Configuración de símbolos**, haga lo siguiente:

Paso	Acción
1	Seleccione el nodo **Aplicación** en el árbol **Herramientas**, haga clic en el signo más de color verde y seleccione el comando **Añadir otros objetos** → **Configuración de símbolos...** **Resultado**: Aparecerá el cuadro de diálogo **Agregar configuración de símbolos**.
2	En el cuadro de diálogo **Agregar configuración de símbolos**, escriba un **Nombre** para la configuración de símbolos en el cuadro de texto.
3	Haga clic en el botón **Agregar**. **Resultado**: Se creará un nodo **Configuración de símbolos** bajo el nodo **Aplicación** del árbol **Herramientas**. La **Configuración de símbolos** aparecerá en el lado derecho.

NOTA: Sólo se puede crear un nodo de configuración de símbolos por dispositivo.

Para obtener más información sobre el intercambio de variables entre el controlador y la parte HMI, consulte el capítulo Intercambio de datos del controlador SoMachine y HMI *(véase página 559)*.

Capítulo 26
Intercambio de datos entre el controlador SoMachine y HMI

Contenido de este capítulo

Este capítulo contiene los siguiente apartados:

Definición de variable simple de SoMachine

Descripción general

Al publicar las variables en SoMachine, estas están disponibles automáticamente para su uso en la aplicación HMI Vijeo-Designer.

Para intercambiar variables con el protocolo SoMachine, siga estos pasos:

- Cree variables en la parte del controlador.
- Publique las variables definiéndolas como **símbolos** en la parte del controlador. A continuación, estarán disponibles en la parte HMI como variables de SoMachine.
- Configure la conexión física (configurada automáticamente por SoMachine).

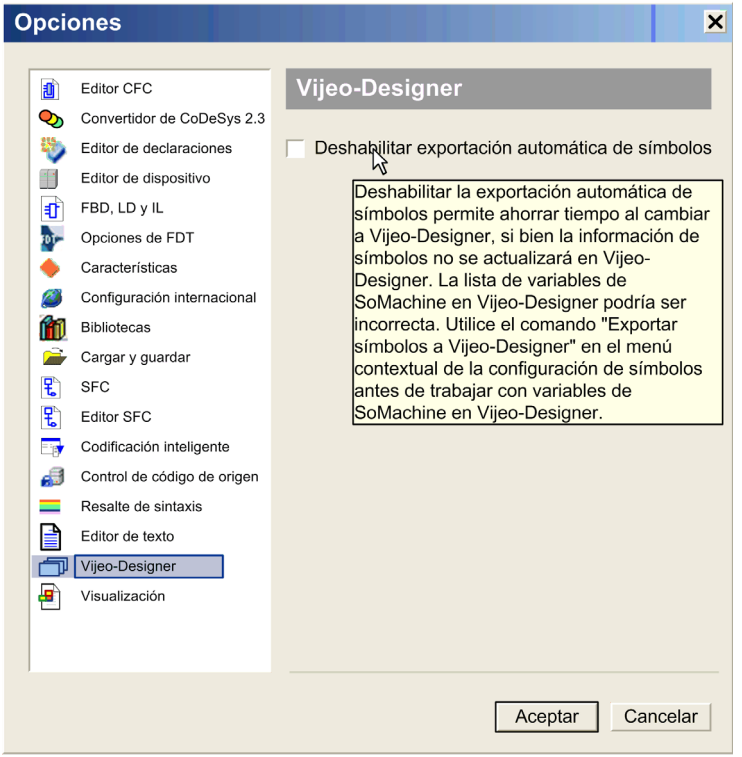
NOTA: El último paso no es necesario para los controladores XBTGC, ya que pueden comunicarse con sus propias variables de control.

Desactivación de la exportación de símbolos automática a Vijeo-Designer

De forma predeterminada, SoMachine exporta de forma automática esas variables definidas como **símbolos** a la aplicación HMI Vijeo-Designer.

Cuando los símbolos se han transferido a Vijeo-Designer, normalmente no es necesario hacer la transferencia cada vez que realice una llamada a Vijeo-Designer. Si más adelante añade o modifica símbolos en la aplicación de SoMachine después de haber transferido inicialmente los símbolos, puede transferir los símbolos a Vijeo-Designer manualmente según desee. Para ahorrar tiempo al abrir Vijeo-Designer puede desactivar la transferencia automática de símbolos de la manera siguiente:

Paso	Acción
1	Seleccione el comando **Opciones...** del menú **Herramientas**. **Resultado**: Aparecerá el cuadro de diálogo **Opciones**.
2	Seleccione la entrada **Vijeo-Designer** en la lista del lado izquierdo.

Paso	Acción
3	En el lado derecho, seleccione la casilla de verificación **Deshabilitar exportación automática de símbolos**. 
4	Haga clic en **Aceptar** para cerrar el cuadro de diálogo.

NOTA: Activar la función **Deshabilitar exportación automática de símbolos** evita la exportación automática de variables de SoMachine definidas como **símbolos** a Vijeo-Designer. Para realizar esta transferencia de forma manual, haga clic con el botón derecho en el nodo **Configuración de símbolos** de la ventana **Dispositivos** y ejecute el comando **Exportar símbolos a Vijeo-Designer**. Si no realiza esta transferencia manual, puede que Vijeo-Designer no muestre los símbolos correctos, lo cual, a su vez, puede ocasionar la detección de errores en el proyecto.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

Ejecute el comando Exportar símbolos a Vijeo-Designer si ha activado Deshabilitar exportación automática de símbolos antes de empezar a trabajar en Vijeo-Designer.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

Tipos de variables para el intercambio de datos del controlador SoMachine y HMI

En la siguiente tabla aparecen los tipos de variables para el intercambio de datos del controlador SoMachine y HMI:

Tipo de variable de SoMachine	Tipo de variable de Vijeo-Designer	Comentario
BOOL	BOOL	--
BYTE	Entero	--
WORD	UINT	--
DWORD	UDINT	--
SINT	Entero	--
INT	INT	--
DINT	DINT	--
USINT	Entero	--
UINT	UINT	--
UDINT	UDINT	--
REAL	REAL	--
STRING	STRING	--
WSTRING	STRING	WSTRING se admite en Vijeo-Designer como un tipo de STRING general. Esto significa que puede intercambiar sólo STRING o sólo WSTRING con HMI. No se permite mezclar estos dos tipos de variables. Si utiliza WSTRING, todas sus cadenas deben ser WSTRING. Indique al controlador de Vijeo-Designer que todas las cadenas deben administrarse como UNICODE WSTRING de la siguiente manera: Seleccione el nodo **SoMachineNetwork** o **SoMachineCombo** en el árbol **Navegador** de Vijeo-Designer y establezca el parámetro **Codificación de la cadena** con el valor **Unicode**.

Tipo de variable de SoMachine	Tipo de variable de Vijeo-Designer	Comentario
Matriz	–	En Vijeo-Designer, sólo se puede hacer referencia a elementos de una matriz, no a la matriz entera. **Ejemplo**: Su matriz está formada por SINT llamados `myValues`. En Vijeo-Designer, puede hacer referencia a `myValues[0]` o `myValues[5]` y fijarlo como una variable del HMI controller. Las matrices no pueden contener más de 2.048 elementos. Si intenta utilizar matrices con más de 2.048 elementos en Vijeo-Designer, aparecerá un mensaje.
DUT	–	En Vijeo-Designer sólo se puede hacer referencia a los elementos de un DUT, no al DUT entero. En este caso, los DUT se comportan de modo similar a las matrices.

Tipos de variables no admitidas

Los siguientes tipos de variables para el intercambio de datos entre SoMachine y HMI no se admiten:

- Todos los formatos de enteros de 64 bits.
- LREAL.
- Todos los formatos de fecha y hora.
- Matrices no nulas: no se puede importar una matriz definida; por ejemplo, `myArray[1..100]`.
- Matrices de matrices: no puede importar una matriz que tenga una matriz como tipo de elemento; por ejemplo, `ARRAY [0..9] OF ARRAY [0..9] OF INT`. Sin embargo, puede utilizar matrices multidimensionales, como `ARRAY [0..9, 0..9] OF INT`.

Para obtener más información acerca de los tipos de variables para el intercambio de datos entre SoMachine y HMI, consulte la ayuda online de Vijeo-Designer.

Longitud del identificador

En Vijeo-Designer, la longitud máxima del nombre del símbolo está limitada a 32 caracteres.

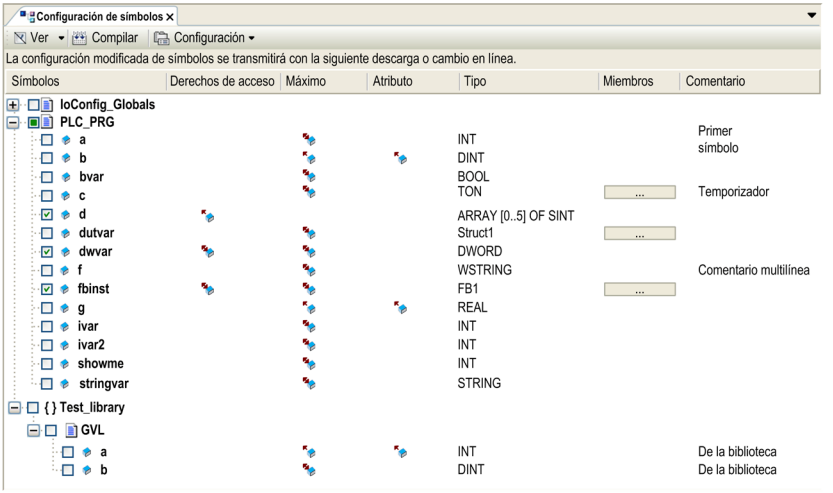
Publicación de variables en la parte del controlador

Descripción general

Publique variables en la parte de controlador de la aplicación SoMachine en el editor de **Configuración de símbolos** o en la vista **Variables** del catálogo de software *(véase página 32)* de una POU.

Publicación de variables en el editor de Configuración de símbolos

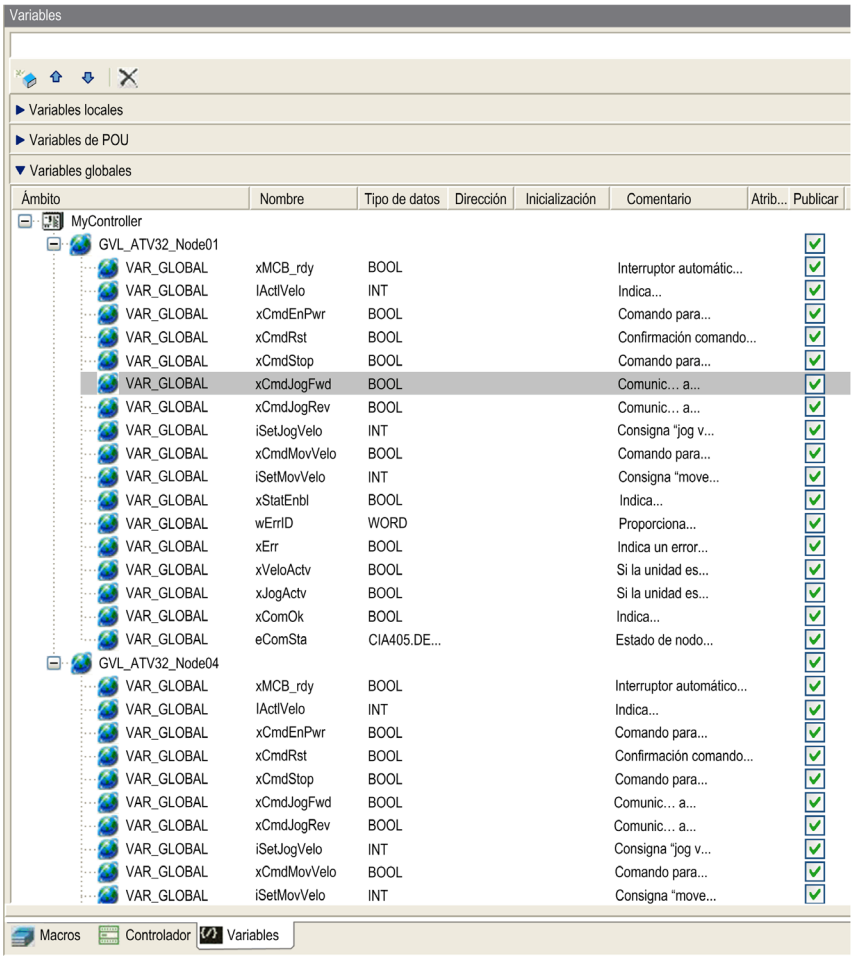
Para publicar variables en el editor de **Configuración de símbolos**, haga lo siguiente:

Paso	Acción
1	Cree un nodo **Configuración de símbolos** bajo el nodo **Aplicación** del árbol **Herramientas** tal como se describe en el capítulo *(véase página 556) Adición de una configuración de símbolos*.
2	Haga doble clic en el nodo **Configuración de símbolos** para abrir el editor de **Configuración de símbolos**.
3	En el editor de **Configuración de símbolos**, seleccione las variables elementales que desee publicar para la comunicación con uno o varios terminales HMI. Para ello, marque o desmarque la casilla en la columna **Símbolos**:  También puede asignar individualmente derechos de acceso de lectura/escritura a cada variable en la columna **Derechos de acceso**. Para obtener más información, consulte la descripción del editor de Configuración de símbolos *(véase página 552)*. **Nota**: Las variables de tipos de datos elementales serán las únicas disponibles para el intercambio con terminales HMI.
4	Para validar las opciones, haga clic en el enlace **Descargar** del editor de **Configuración de símbolos**.

NOTA: El mecanismo de publicación consume una carga del sistema de unos 50 kilobytes en el controlador. Cada variable publicada consume 11 bytes en la aplicación de controlador.

Publicación de variables en la vista Variables del catálogo de software

Para publicar variables en la vista **Variables** del catálogo de software *(véase página 32)*, haga lo siguiente:

Paso	Acción
1	Abra la vista **Variables** del catálogo de software *(véase página 32)*.
2	Para publicar una variable, seleccione la casilla respectiva en la columna **Publicar**. 

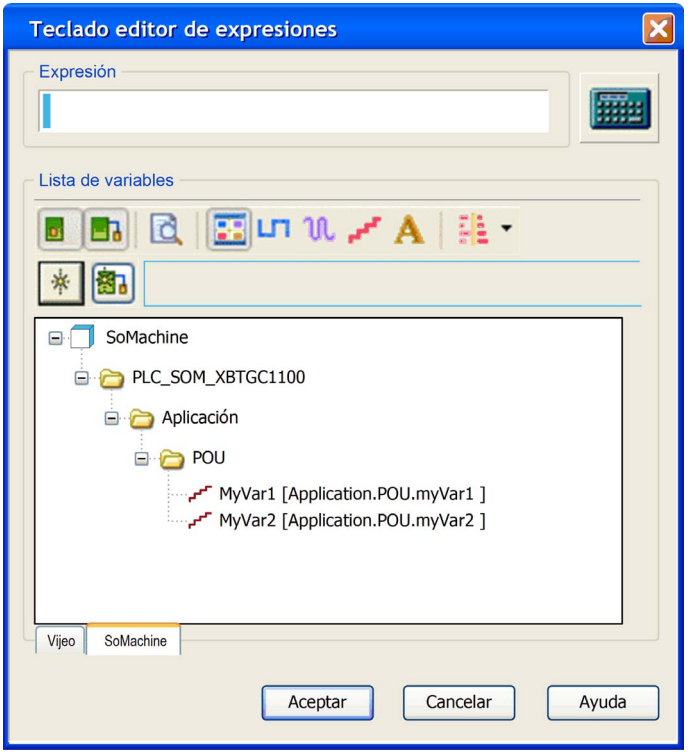
NOTA: Verifique que se llame en una tarea a la POU de las variables seleccionadas. De lo contrario, las variables seleccionadas no se publicarán.

Selección de variables en la parte HMI

Selección de variables

Las variables que se han publicado en la parte del controlador están disponibles directamente en la parte de HMI.

En el **Teclado editor de expresiones** de Vijeo-Designer, seleccione la ficha **SoMachine** para tener acceso directo a las variables publicadas en SoMachine.



Para obtener más información, consulte la ayuda online de Vijeo-Designer.

Publicación de variables en la parte HMI

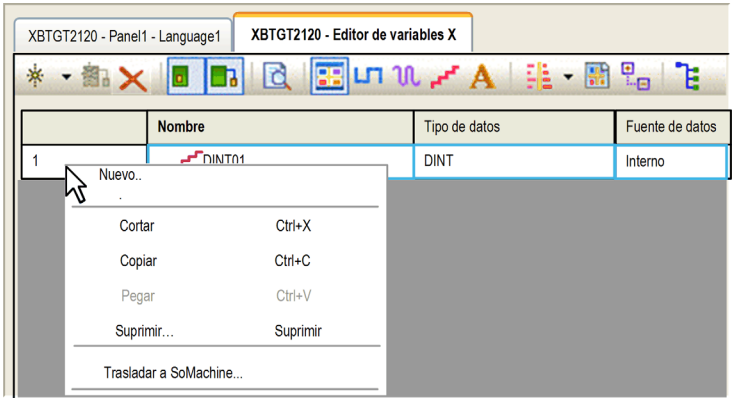
Tipos de variables admitidas

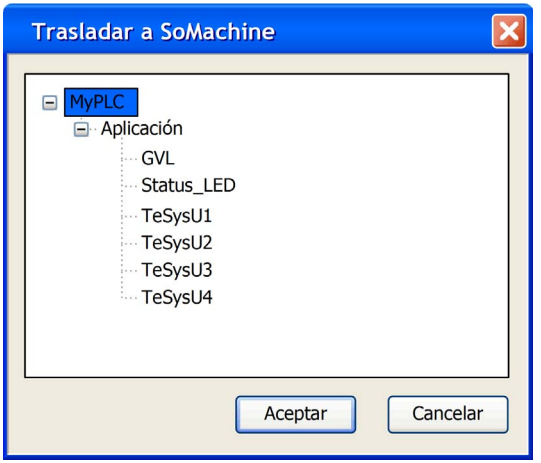
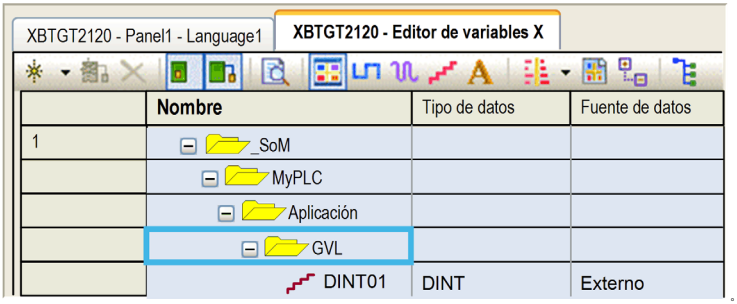
Los siguientes tipos de variables pueden publicarse en Vijeo-Designer para ponerlas a disposición de todo el proyecto de SoMachine:

- BOOL
- DINT
- INT
- UINT
- UDINT
- Entero
- REAL
- STRING

Procedimiento

Para publicar los tipos de variables arriba mencionados, proceda de este modo:

Paso	Acción
1	En el Vijeo-Designer **Editor de variables**, seleccione las variables que desee publicar.
2	Haga clic con el botón derecho del ratón en las variables seleccionadas y ejecute el comando **Trasladar a SoMachine** del menú contextual.  **Resultado**: aparecerá el cuadro de diálogo **Trasladar a SoMachine**.
3	En el cuadro de diálogo **Trasladar a SoMachine**, abra las subcarpetas de los dispositivos definidos en SoMachine para ver los niveles en los que están definidas las variables (POU o GVL).

Paso	Acción
4	Seleccione la POU o GVL a la que desee añadir las Vijeo-Designer variables seleccionadas y haga clic en **Aceptar**.  **Resultado**: se han movido las variables seleccionadas a la POU o GVL de SoMachine seleccionada y están disponibles en todo el proyecto de SoMachine. 

Parametrización de los medios mecánicos

Descripción general

El intercambio de datos en tiempo de ejecución entre el controlador y HMI se ejecuta en medios distintos, según el hardware seleccionado.

Ejemplo de configuración

La configuración predeterminada que se expone a continuación es válida para las comunicaciones entre M238 y un panel HMI mediante la línea serie RS485 con un cable XBTZ9008 (línea serie SubD-RJ45).

Configuración de M238 con panel HMI:

Configuración de línea serie del controlador M238

Parámetro	Valor
Medio físico	RS485
Velocidad en baudios	115200
Paridad	ninguna
Bits de datos	8
Bits de parada	1

Configuración del administrador de E/S del panel HMI utilizando un controlador: SoMachine - red con al menos un grupo de exploración (para obtener más información, consulte la ayuda online de Vijeo-Designer).

Parámetro	Valor
Medio físico	RS485
Velocidad en baudios	115200
Paridad	ninguna
Bits de datos	8
Bits de parada	1
Nombre del equipo	Nombre del dispositivo de controlador (disponible en el cuadro de diálogo de configuración de comunicación)

Configuración de XBTGC:

Dispositivo	Configuración
Controlador XBTGC	no requiere ninguna configuración
Subventana HMI	Administrador de E/S
Controlador	SoMachine - combinación con al menos un grupo de exploración

Parte VII
Referencia de programación

Contenido de esta parte

Esta parte contiene los siguientes capítulos:

Capítulo 27
Declaración de variables

Contenido de este capítulo

Este capítulo contiene las siguientes secciones:

Sección 27.1
Declaración

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Información general

Descripción general

Puede declarar variables:

- en la vista **Variables** del **Catálogo de software** *(véase página 31)*
- en el **Editor de declaraciones** de un POU *(véase página 414)*
- a través del cuadro de diálogo **Declarar variable** *(véase página 584)*
- en un editor DUT
- en un editor GVL

El tipo (en el editor de declaraciones tabular se denomina **Ámbito**) de las variables que se va a declarar se especifica mediante las palabras claves que rodean la declaración de uno o varias variables. En el editor de declaraciones textual *(véase página 414)*, a la declaración de variables común se incluye `VAR` y `END_VAR`.

Para obtener más información sobre los ámbitos de declaración de variables, consulte:

- `VAR_INPUT`
- `VAR_OUTPUT`
- `VAR_IN_OUT`
- `VAR_GLOBAL`
- `VAR_TEMP`
- `VAR_STAT`
- `VAR_EXTERNAL`
- `VAR_CONFIG`

Las palabras clave del tipo de variable pueden complementarse con palabras clave de atributos *(véase página 596)*.

Ejemplo: `RETAIN (VAR_INPUT RETAIN)`

Sintaxis

Sintaxis de la declaración de variables:

<Identificador> {`AT` <dirección>}:<tipo de datos> {:=<inicialización>};

Las partes que aparecen dentro de llaves {} son opcionales.

Identificador

El identificador es el nombre de una variable.

Tenga en cuenta lo siguiente cuando defina un identificador.

- No se permiten espacios o caracteres especiales.
- No hay distinción entre mayúsculas y minúsculas: `VAR1`, `Var1` y `var1` son la misma variable.
- Reconocimiento del guión bajo: `A_BCD` y `AB_CD` se consideran dos identificadores distintos. No utilice más de un guión bajo en una misma fila.
- Longitud ilimitada.
- Recomendaciones sobre el uso múltiple (ver párrafo siguiente).

Además, tenga en cuenta las recomendaciones del capítulo Recomendaciones sobre la nomenclatura de identificadores *(véase página 578)*.

Uso múltiple de identificadores (espacios de nombres)

A continuación se especifican las normas relacionadas con el uso múltiple de identificadores:

- No crear un identificador que sea idéntico a una palabra clave.
- No se permite el uso duplicado de identificadores localmente.
- El uso múltiple de un identificador se permite globalmente: una variable local puede tener el mismo nombre que una global. En este caso, tendrá prioridad la variable local dentro del POU.
- Una variable definida en una lista de variables globales (GVL) puede tener el mismo nombre que una variable definida en otra lista de variables globales (GVL). En este contexto, considere las funciones de extensión de la siguiente IEC 61131-3:
  - Operador de ámbito global: una ruta de instancia que empieza con un punto (`.`) abre un ámbito global. De este modo, si existe una variable local, por ejemplo `ivar`, con el mismo nombre que una variable global, `.ivar` hará referencia a la variable global.
  - Puede utilizar el nombre de una lista de variables globales (GVL) como espacio de nombres de las variables incluidas. Puede declarar variables con el mismo nombre en distintas listas de variables globales (GVL). Se puede acceder específicamente a estas variables poniendo el nombre de la lista delante del nombre de la variable.
    Ejemplo
    ```
    globlist1.ivar := globlist2.ivar;
    (* ivar de globlist2 en la biblioteca lib1 se copia a ivar en GVL globlist1 *)
    ```
  - Se puede acceder a las variables definidas en una lista de variables globales de una biblioteca incorporada según la sintaxis <espacio de nombres de la biblioteca>.<nombre del GVL>.<variable>.
    Ejemplo:
    ```
    globlist1.ivar := lib1.globlist1.ivar
    (* ivar de globlist1 en la biblioteca lib1 se copia a ivar en GVL globlist1 *)
    ```
- Asimismo, para una biblioteca, se define un espacio de nombres cuando se incorpora a través del **Administrador de bibliotecas**. Así que podrá acceder al módulo de una biblioteca o una variable mediante <espacio de nombres de la biblioteca>.<modulename|variablename>. Tenga en cuenta que, si se trata de bibliotecas anidadas, los nombres de espacios de todas las bibliotecas afectadas deben describirse sucesivamente.

Ejemplo: si `Lib1` hace referencia a `Lib0`, al módulo `fun` que forma parte de `Lib1` se accede a través de `Lib0.Lib1.fun`:
`ivar := Lib0.Lib1.fun(4, 5);` (* el valor de retorno de `fun` se copia a la variable `ivar` del proyecto *)
NOTA: Cuando la casilla de verificación **Tornar visibles todos los símbolos IEC en el proyecto, como si esta referencia aquí se hubiese incorporado directamente.** se ha activado dentro del cuadro de diálogo **Propiedades** de la biblioteca referenciada `Lib`, también se puede acceder al módulo `fun` directamente a través de `Lib0.fun`.

AT <dirección>

Puede vincular la variable directamente a una dirección definida *(véase página 586)* con la palabra clave `AT`.

En los bloques de funciones, también puede especificar variables con instrucciones de direcciones incompletas. Para que dicha variable se pueda usar en una instancia local, tiene que existir una entrada para ello en la configuración de variables.

Tipo

Tipo de datos *(véase página 662)* válido, ampliado de manera opcional con una :=< inicialización> *(véase página 583)*.

Instrucciones Pragma

De manera opcional, puede añadir instrucciones pragma *(véase página 607)* en la parte de la declaración de un objeto para la generación de códigos con distintos objetivos.

Sugerencias

También es posible la declaración automática *(véase página 584)* de variables.

Para introducir declaraciones más rápidamente, use la modalidad de acceso directo *(véase página 585)*.

Recomendaciones sobre la nomenclatura de identificadores

Descripción general

Los identificadores se definen:

- en la declaración de variables (nombre de la variable)
- en la declaración de tipos de datos definidos por el usuario
- en la creación de POU (funciones, bloques de funciones, programas)

Además de los elementos generales que se deben tener en cuenta cuando se define un identificador (consulte el capítulo *Información general* sobre la declaración de variables *(véase página 575)*), tenga en cuenta las recomendaciones siguientes para que la nomenclatura sea lo más exclusiva posible:

- Nombres de variables *(véase página 578)*
- Nombres de variables en bibliotecas *(véase página 580)*
- Tipos de datos definidos por el usuario (DUT) en bibliotecas *(véase página 581)*
- Funciones, bloques de funciones, programas (POU), acciones *(véase página 581)*
- POU en bibliotecas *(véase página 582)*
- Nombres de visualización *(véase página 582)*

Nombres de variables

Para dar nombre a las variables de las aplicaciones y las bibliotecas, siga la notación húngara en la medida de lo posible.

Busque una descripción breve y representativa para cada variable. Esta descripción se utilizará como nombre de base. Ponga en mayúscula la primera letra de cada palabra del nombre de base. Escriba el resto de la palabra en minúsculas (ejemplo: `FileSize`).

Tipo de datos	Límite inferior	Límite superior	Contenido de información	Prefijo	Comentario
BOOL	FALSE	TRUE	1 bits	`x`*	–
				`b`	Reservado
BYTE	–	–	8 bits	`by`	cadena de bits, no para operaciones aritméticas
WORD	–	–	16 bits	`w`	cadena de bits, no para operaciones aritméticas
DWORD	–	–	32 bits	`dw`	cadena de bits, no para operaciones aritméticas
LWORD	–	–	64 bits	`lw`	no para operaciones aritméticas
SINT	–128	127	8 bits	`si`	–
USINT	0	255	8 bits	`usi`	–

* intencionalmente para variables booleanas x se elige como prefijo para diferenciar del BYTE y también para acomodar la percepción de un programador IEC (consulte el direccionamiento de `%IX0.0`).

Tipo de datos	Límite inferior	Límite superior	Contenido de información	Prefijo	Comentario
INT	–32,768	32,767	16 bits	i	–
UINT	0	65,535	16 bits	ui	–
DINT	–2,147,483,648	2,147,483,647	32 bits	di	–
UDINT	0	4,294,967,295	32 bits	udi	–
LINT	-2^{63}	$2^{63}-1$	64 bits	li	–
ULINT	0	$2^{64}-1$	64 bits	uli	–
REAL	–	–	32 bits	r	–
LREAL	–	–	64 bits	lr	–
STRING	–	–	–	s	–
TIME	–	–	–	tim	–
TIME_OF_DAY	–	–	–	tod	–
DATE_AND_TIME	–	–	–	dt	–
DATE	–	–	–	date	–
ENUM	–	–	16 bits	e	–
POINTER	–	–	–	p	–
ARRAY	–	–	–	a	–
* intencionalmente para variables booleanas x se elige como prefijo para diferenciar del BYTE y también para acomodar la percepción de un programador IEC (consulte el direccionamiento de %IX0.0).					

Declaración simple

Ejemplos de declaraciones simples:

```
bySubIndex: BYTE;
sFileName: STRING;
udiCounter: UDINT;
```

Declaración anidada

Ejemplo de una declaración anidada en la que los prefijos se conectan entre sí en el orden de las declaraciones:

```
pabyTelegramData: POINTER TO ARRAY [0..7] OF BYTE;
```

Variables e instancias del bloque de funciones de tipos de datos definidos por el usuario

Las variables e instancias del bloque de funciones de los tipos de datos definidos por el usuario obtienen un acceso directo para el nombre del tipo de datos o el bloque de funciones como prefijo (por ejemplo: sdo).

Ejemplo

```
cansdoReceivedTelegram: CAN_SDOTelegram;
```

```
TYPE CAN_SDOTelegram :     (* prefix: sdo *)
STRUCT
  wIndex:WORD;
  bySubIndex:BYTE;
  byLen:BYTE;
  aby: ARRAY [0..3] OF BYTE;
END_STRUCT
END_TYPE
```

Constantes locales

Las constantes locales (c) empiezan con el prefijo `c` y un guión bajo pegado, seguido del prefijo de tipo y el nombre de la variable.

Ejemplo

```
VAR CONSTANT
  c_uiSyncID: UINT := 16#80;
END_VAR
```

Variables globales y constantes globales

Las variables globales van precedidas de `g_` y las constantes globales, de `gc_`.

Ejemplo

```
VAR_GLOBAL
  g_iTest: INT;
END_VAR
VAR_GLOBAL CONSTANT
  gc_dwExample: DWORD;
END_VAR
```

Nombres de variables en bibliotecas

Estructura

Por lo general, consulte la descripción anterior para los nombres de variables. Use el espacio de nombres de la biblioteca como prefijo cuando acceda a una variable en su código de aplicación.

Ejemplo

```
g_iTest: INT; (declaration)
CAN.g_iTest (implementation, call in an application program
```

Tipos de datos definidos por el usuario (DUT) en bibliotecas

Estructura

El nombre del tipo de datos de cada estructura consta de una descripción breve y representativa (por ejemplo, `SDOTelegram`) de la estructura.

Ejemplo (en una biblioteca con espacio de nombres `CAL`):

```
TYPE Day :(
MONDAY,
TUESDAY,
WEDNESDAY,
THURSDAY,
FRIDAY,
SATURDAY,
SUNDAY);
```

Declaración:

```
eToday: CAL.Day;
```

Uso en aplicación:

```
IF eToday = CAL.Day.MONDAY THEN
```

NOTA: Tenga en cuenta el uso del espacio de nombres al utilizar DUT o enumeraciones declaradas en bibliotecas.

Funciones, bloques de funciones, programas (POU), acciones

Los nombres de funciones, bloques de funciones y programas van precedidos de un nombre corto representativo del POU (por ejemplo, `SendTelegram`). Al igual que ocurre con las variables, la primera letra de una palabra del nombre del POU debe ir siempre en mayúscula, mientras que el resto de letras irán en minúscula. Se recomienda que el nombre del POU se componga de un verbo y un sustantivo.

Ejemplo

```
FUNCTION_BLOCK SendTelegram (* prefix: canst *)
```

En la parte de declaración, escriba una breve descripción del POU en forma de comentario. Además, las entradas y salidas deben estar provistas de comentarios. En caso de bloques de funciones, inserte el prefijo asociado para instancias de configuración directamente después del nombre.

Acciones

Las acciones no llevan prefijo. Únicamente aquellas acciones que se llamarán solo internamente, es decir, por el mismo POU, empiezan por `prv_`.

Se supone que cada función, por motivos de compatibilidad con versiones de software anteriores, tiene al menos un parámetro. No utilice estructuras como valores de retorno en funciones externas.

POU en bibliotecas

Estructura

Para crear nombres de métodos, se aplican las mismas reglas que para las acciones. Escriba comentarios en inglés para las posibles entradas de un método. Añada una breve descripción de un método a su declaración. Comience los nombres de la interfaz con la letra `I`; por ejemplo, `ICANDevice`.

NOTA: Tenga en cuenta el uso del espacio de nombres al utilizar POU declaradas en bibliotecas.

Nombres de visualización

Evite nombrar una visualización de forma similar a otro objeto del proyecto, pues esto podría ocasionar anomalías si se producen cambios en la visualización.

Inicialización de variables

Valor de inicialización predeterminado

El valor de inicialización predeterminado es 0 para todas las declaraciones, pero puede añadir valores de inicialización definidos por el usuario en la declaración de cada variable y tipo de datos.

Valores de inicialización definidos por el usuario

La inicialización definida por el usuario se debe al operador de asignaciones := y puede ser cualquier expresión ST válida. Por consiguiente, los valores de constante, así como otras variables o funciones, se pueden utilizar para definir el valor de inicialización. Compruebe que ya se ha iniciado la propia variable usada para la inicialización u otra variable.

Ejemplo de inicializaciones de variables válidas:

```
VAR
var1:INT := 12;               * Integer variable with initial value of
12. *
x : INT := 13 + 8;            * Integer value defined an expression wit
h literal values.*
y : INT := x + fun(4);        * Integer value defined by an expression
containing a function call. NOTE: Be sure that any variables used in
the variable initialization have already been defined. *
z : POINTER TO INT := ADR(y); * POINTER is not described by the IEC6113
1-3:
Integer value defined by an address function; NOTE: The pointer will
not be initialized if the declaration is modified online. *
END_VAR
```

Información adicional

Para obtener más información, consulte las siguientes descripciones:

- inicialización de matrices *(véase página 678)*
- inicialización de estructuras *(véase página 681)*
- inicialización de una variable con un tipo de subrango *(véase página 685)*

NOTA: Las variables de las listas de variables globales (GVL) se inicializan antes que las variables locales de un POU.

Declaración

Tipos de declaración

Puede declarar las variables manualmente con el editor de declaraciones *(véase página 414)* textual o tabular o automáticamente como se explica en este capítulo.

Autodeclaración automática

Puede definir en el cuadro de diálogo **Opciones**, categoría **Editor de texto** →**Editar**, que el cuadro de diálogo **Declarar variable** se deberá abrir en cuanto se introduzca una cadena todavía no declarada en la parte de la implementación de un editor y se pulse la tecla INTRO. Este cuadro de diálogo admite la declaración de la variable *(véase página 575)*.

Declaración de variables manualmente

Para abrir el cuadro de diálogo **Declarar variable** manualmente:

- ejecute el comando **Declarar variable**, que está disponible de forma predeterminada en el menú **Editar**; o
- pulse las teclas SHIFT+F2.

Si elige una variable ya declarada antes de abrir el cuadro de diálogo **Declarar variable**, podrá editar la declaración de esta variable.

Modalidad de acceso directo

Descripción general

El editor de declaraciones *(véase página 414)* y el resto de editores de textos en los que se ejecutan las declaraciones admiten la modalidad de acceso directo.

Puede activar esta modalidad pulsando CTRL+INTRO al terminar una línea de la declaración.

Le permite utilizar accesos directos en vez de escribir completamente la declaración.

Accesos directos compatibles

Son compatibles los siguientes accesos directos:

- Todos los identificadores hasta el último identificador de una línea pasarán a ser identificadores de variables de la declaración.
- El tipo de declaración se determina mediante el último identificador de la línea.

En este contexto, se ejecutan los siguientes reemplazos:

`B` o `BOOL`	se reemplaza por	BOOL
`I` o `INT`		INT
`R` o `REAL`		REAL
`S` o `string`		STRING

- Si no se ha establecido ningún tipo mediante estas normas, BOOL será automáticamente el tipo y el último identificador no se utilizará como tal (consulte el ejemplo 1).
- Todas las constantes, en función del tipo de declaración, se convertirán en una inicialización o cadena (consulte los ejemplos 2 y 3).
- Una dirección (como en `%MD12`) se extiende mediante la palabra clave `AT` (consulte el ejemplo 4).
- El texto después de un punto y coma (;) pasa a ser un comentario (consulte el ejemplo 4).
- El resto de caracteres de la línea se ignoran (consulte, por ejemplo, el signo de exclamación del ejemplo 5).

Ejemplos

N.º de ejemplo	Método abreviado	Declaración resultante
1	`A`	`A: BOOL;`
2	`A B I 2`	`A, B: INT := 2;`
3	`ST S 2; A string`	`ST:STRING(2); (* A string *)`
4	`X %MD12 R 5 Real Number`	`X AT %MD12: REAL := 5.0;(* Real Number *)`
5	`B !`	`B: BOOL;`

Declaración AT

Descripción general

Para enlazar una variable de proyecto con una dirección definida puede asignar variables a una dirección en la vista de **asignación E/S** de un dispositivo en la configuración del controlador(editor de dispositivo). También puede introducir esta dirección directamente en la declaración de la variable.

Sintaxis

<identificador> AT <dirección> : <tipo de datos>;

Una dirección válida debe seguir a la palabra clave AT. Para obtener más información, consulte la descripción de la *dirección (véase página 813)*. Tenga en cuenta la posibilidad de solapamientos en caso de modalidad de direccionamiento de bytes

Esta declaración permite asignar un nombre significativo a una dirección. Cualquier modificación relativa a una señal entrante o saliente solamente podrá realizarse en un único sitio (por ejemplo, en la declaración).

Tenga en cuenta los siguientes puntos al escoger una variable y asignarla a una dirección:

- No se puede acceder mediante escritura a las variables que requieren una entrada. El compilador lo intercepta detectando un error.
- Las declaraciones AT solamente pueden utilizarse con variables locales o globales. No pueden utilizarse con variables de entrada y salida de POU.
- Las declaraciones AT no están permitidas en las listas de variables persistentes.
- Si se utilizan declaraciones AT con miembros de un bloque de funciones o estructura, todas las instancias accederán a la misma ubicación de memoria de dicha estructura o bloque de funciones. Esto corresponde a las variables estáticas en los lenguajes de programación clásicos como C.
- La disposición de memoria de las estructuras está también determinada por el destino.

Ejemplos

```
xCounterHeat7 AT %QX0.0: BOOL;
xLightCabinetImpulse AT %IX7.2: BOOL;
xDownload AT %MX2.2: BOOL;
```

Nota

Si se asignan variables booleanas a una dirección de BYTE, WORD o DWORD ocuparán un byte con VERDADERO o FALSO, no solo el primer bit tras el offset.

Explicación: Las booleanas tienen 8 bits al declararse y cuando se escriben en otros tipos de variables, todos los 8 bits van juntos.

Palabras clave

Descripción general

Escriba las palabras clave en letras mayúsculas en los editores.

Las siguientes cadenas se reservan como palabras clave. No se pueden utilizar como identificadores para variables o POU:

- ABS
- ACOS
- ACTION (solo utilizado en el formato de exportación)
- ADD
- ADR
- AND
- ANDN
- ARRAY
- ASIN
- AT
- ATAN
- BITADR
- BOOL
- BY
- BYTE
- CAL
- CALC
- CALCN
- CASE
- CONSTANT
- COS
- DATE
- DINT
- DIV
- DO
- DT
- DWORD
- ELSE
- ELSIF
- END_ACTION (solo utilizado en el formato de exportación)
- END_CASE
- END_FOR
- END_FUNCTION (solo utilizado en el formato de exportación)
- END_FUNCTION_BLOCK (solo utilizado en el formato de exportación)
- END_IF
- END_PROGRAM (solo utilizado en el formato de exportación)
- END_REPEAT
- END_STRUCT

- END_TYPE
- END_VAR
- END_WHILE
- EQ
- EXIT
- EXP
- EXPT
- FALSE
- FOR
- FUNCTION
- FUNCTION_BLOCK
- GE
- GT
- IF
- INDEXOF
- INT
- JMP
- JMPC
- JMPCN
- LD
- LDN
- LE
- LINT
- LN
- LOG
- LREAL
- LT
- LTIME
- LWORD
- MAX
- METHOD
- MIN
- MOD
- MOVE
- MUL
- MUX
- NE
- NOT
- OF
- OR
- ORN
- PARAMS
- PERSISTENT
- POINTER
- PROGRAM

- R
- READ_ONLY
- READ_WRITE
- REAL
- REFERENCE
- REPEAT
- RET
- RETAIN
- RETC
- RETCN
- RETURN
- ROL
- ROR
- S
- SEL
- SHL
- SHR
- SIN
- SINT
- SIZEOF
- SUPER
- SQRT
- ST
- STN
- STRING
- STRUCT
- SUPER
- SUB
- TAN
- THEN
- THIS
- TIME
- TO
- TOD
- TRUE
- TRUNC
- TYPE
- UDINT
- UINT
- ULINT
- UNTIL
- USINT
- VAR
- VAR_ACCESS (solo utilizado de forma específica, en función del hardware)
- VAR_CONFIG

- VAR_EXTERNAL
- VAR_GLOBAL
- VAR_IN_OUT
- VAR_INPUT
- VAR_OUTPUT
- VAR_STAT
- VAR_TEMP
- WHILE
- WORD
- WSTRING
- XOR
- XORN

Además, los operadores de conversión como aparecen en **Accesibilidad** se consideran palabras clave.

Sección 27.2
Tipos de variables

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Tipos de variables

Descripción general

En este capítulo se proporciona información sobre los tipos de variables siguientes:

- VAR variables locales *(véase página 592)*
- VAR_INPUT variables de entrada *(véase página 592)*
- VAR_OUTPUT variables de salida *(véase página 593)*
- VAR_IN_OUT variables de entrada y salida *(véase página 593)*
- VAR_GLOBAL variables globales *(véase página 594)*
- VAR_TEMP variables temporales *(véase página 594)*
- VAR_STAT variables estáticas *(véase página 594)*
- VAR_EXTERNAL variables externas *(véase página 595)*

Variables locales: VAR

Entre las palabras clave VAR y END_VAR, se declaran *(véase página 575)* todas las variables locales de un POU. No tienen conexión externa; esto es, no pueden escribirse desde el exterior.

Considere la posibilidad de añadir un atributo *(véase página 596)* a VAR.

Ejemplo

```
VAR
 iLoc1:INT; (* 1. Local Variable*)
END_VAR
```

Variables de entrada: VAR_INPUT

Entre las palabras clave VAR_INPUT y END_VAR, se declaran *(véase página 575)* todas las variables que sirven como variables de entrada para un POU. Es decir, en la posición de llamada, el valor de las variables se puede proporcionar con una llamada.

Considere la posibilidad de añadir un atributo *(véase página 596)*.

Ejemplo

```
VAR_INPUT
 iIn1:INT (* 1. Inputvariable*)
END_VAR
```

Variables de salida: VAR_OUTPUT

Entre las palabras clave VAR_OUTPUT y END_VAR, se declaran todas las variables que sirven como variables de salida de un POU. Es decir, estos valores se devuelven al POU que hace la llamada.

Considere la posibilidad de añadir un atributo *(véase página 596)* a VAR_OUTPUT.

Ejemplo

```
VAR_OUTPUT
 iOut1:INT; (* 1. Outputvariable*)
END_VAR
```

Variables de salida en funciones y métodos:

Conforme al borrador 2 de la IEC 61131-3, las funciones (y los métodos) pueden tener salidas adicionales. Puede asignarlas en la llamada de la función como se muestra en el ejemplo siguiente.

Ejemplo

```
fun(iIn1 := 1, iIn2 := 2, iOut1 => iLoc1, iOut2 => iLoc2);
```

Variables de entrada y salida: VAR_IN_OUT

Entre las palabras clave VAR_IN_OUT y END_VAR, se declaran *(véase página 575)* todas las variables que sirven como variables de entrada y salida para un POU.

NOTA: Con variables de tipo IN_OUT, se cambia el valor de la variable transferida (transferida como puntero, llamar-por-referencia). Esto es, el valor de entrada para esas variables no puede ser una constante. Por este motivo, incluso las variables VAR_IN_OUT de un bloque de funciones no se pueden leer o escribir directamente desde el exterior a través de <FBinstance>.<InOutVariable>.

NOTA: No asigne símbolos de tipo bit (tales como variables %MXaa.b o BOOL que están situadas en una dirección de dicho tipo bit) a parámetros VAR_IN_OUT de tipo BOOL de bloques de funciones. Si se detecta cualquiera de esas asignaciones, se comunica como un error de **compilación** detectado en la vista *(véase SoMachine, Comandos de menú, Ayuda en línea)* **Mensajes**.

Ejemplo

```
VAR_IN_OUT
 iInOut1:INT; (* 1. Inputoutputvariable *)
END_VAR
```

Variables globales: VAR_GLOBAL

Puede declarar variables normales, constantes, externas o variables remanentes que se conozcan a través del proyecto como variables globales. Para declarar variables globales, utilice las listas de variables globales (GVL). Puede añadir un GVL ejecutando el comando **Añadir objeto** (de forma predeterminada en el menú **Proyecto**).

Declare las variables localmente entre las palabras clave `VAR_GLOBAL` y `END_VAR`.

Considere la posibilidad de añadir un atributo *(véase página 596)* a `VAR_GLOBAL`.

Se reconoce una variable como variable global porque va precedida de un punto; por ejemplo, `.iGlobVar1`.

Para obtener más información sobre el uso múltiple de nombres de variables, el operador de ámbito global `dot (.)` y los espacios de nombres, consulte el capítulo *Operador de ámbito global* *(véase página 794)*.

Las variables globales solo se pueden declarar en las listas de variables globales (GVL). Sirven para gestionar variables globales dentro de un proyecto. Puede añadir un GVL ejecutando el comando **Añadir objeto** (de forma predeterminada en el menú **Proyecto**).

NOTA: Una variable definida localmente en un POU con el mismo nombre que una variable global tendrá prioridad dentro del POU.

NOTA: Las variables globales se inicializan antes de las variables locales de los POU.

Variables temporales: VAR_TEMP

Esta función es una extensión de la norma IEC 61131-3.

Las variables temporales se (re)inicializan en cada llamada del POU. Las declaraciones `VAR_TEMP` solo son posibles dentro de programas y bloques de funciones. Estas variables son también solo accesibles dentro del cuerpo del POU del programa o el bloque de funciones.

Declare las variables localmente entre las palabras clave `VAR_TEMP` y `END_VAR`.

NOTA: Puede utilizar `VAR_TEMP` en lugar de `VAR` para reducir el espacio en memoria que requiere un POU (por ejemplo, dentro de un bloque de funciones si la variable solo se utiliza temporalmente).

Variables estáticas: VAR_STAT

Esta función es una extensión de la norma IEC 61131-3.

Las variables estáticas se pueden usar en bloques de funciones, métodos y funciones. Declárelas localmente entre las palabras clave `VAR_STAT` y `END_VAR`. Se inicializan en la primera llamada del POU correspondiente.

Al igual que las variables globales, las variables estáticas no pierden su valor tras dejar el POU en que se declaran. Se comparten entre los POU en que se declaran (por ejemplo, varios métodos, funciones e instancias de los bloques de funciones comparten la misma variable estática). Por ejemplo, se pueden utilizar en una función como contador del número de llamadas de función.

Considere la posibilidad de añadir un atributo *(véase página 596)* a `VAR_STAT`.

Variables externas: VAR_EXTERNAL

Son variables globales que se importan al POU.

Declárelas localmente entre las palabras clave VAR_EXTERNAL y END_VAR y en la lista de variables globales (GVL). La declaración y la declaración global deben ser idénticas. Si la variable global no existe, se mostrará un mensaje.

NOTA: No es necesario definir las variables como externas. Estas palabras clave se proporcionan para mantener la compatibilidad con IEC 61131-3.

Ejemplo

```
VAR_EXTERNAL
 iVarExt1:INT; (* 1st external variable *)
END_VAR
```

Palabras clave de atributo para los tipos de variables

Descripción general

Puede añadir las palabras clave de atributo siguientes a la declaración *(véase página 575)* del tipo de variable para especificar el ámbito:

- `RETAIN`: consulte *Variables Retain (véase página 596)*
- `PERSISTENT`: consulte *Variables persistentes (véase página 597)*
- `CONSTANT`: consulte *Constantes - CONSTANT (véase página 598)*, *Literales con tipo (véase página 598)*

Variables remanentes - `RETAIN, PERSISTENT`

Las variables remanentes pueden conservar sus valores durante el periodo de ejecución del programa habitual. Declárelas como variables Retain o, para una mayor exigencia, como variables persistentes. En cada caso se emplea un área de memoria propia.

La declaración determina el grado de resistencia de una variable remanente en caso de restablecimientos, descargas o un reinicio del controlador. En las aplicaciones se utiliza principalmente la combinación de ambos indicadores remanentes (consulte *Variables persistentes (véase página 597)*).

NOTA: Utilice el comando *(véase SoMachine, Comandos de menú, Ayuda en línea)* **Agregar todas las rutas de instancia** para tomar las variables declaradas como persistentes en el objeto **Persistent list**.

NOTA: No utilice la declaración `AT` junto con `VAR RETAIN` ni `VAR PERSISTENT`.

Variables Retain

Las variables declaradas como Retain se almacenan en un área de memoria no volátil. Para declarar este tipo de variable, utilice la palabra clave `RETAIN` en la parte de declaración de una POU o en una lista de variables globales.

Ejemplo

```
VAR RETAIN
 iRem1 : INT; (* 1. Retain variable*)
END_VAR
```

Las variables Retain conservan su valor incluso después de un apagado imprevisto del controlador o después de una operación de apagado y encendido normal del controlador (o cuando se ejecuta el comando **En línea Reset caliente**). Al reiniciar el programa, los valores conservados se seguirán procesando. Las otras variables (que no son Retain) se inicializan de nuevo, ya sea con sus valores de inicialización o con sus valores de inicialización predeterminados (en el caso de que no se haya declarado un valor de inicialización).

Por ejemplo, puede utilizar un valor conservado cuando una operación (por ejemplo, el conteo de piezas en una máquina de producción) debe continuar tras un corte en el suministro eléctrico.

Las variables Retain, sin embargo, se reinicializan al ejecutar el comando **En línea Reset origen** y, a diferencia de las variables persistentes, al ejecutar el comando **En línea Reset frío** o durante la descarga de una aplicación.

NOTA: Solamente las variables definidas como `VAR RETAIN` se almacenan en la memoria no volátil. Sin embargo, las variables locales definidas como `VAR RETAIN` en las funciones NO se almacenan en la memoria volátil. La definición de `VAR RETAIN` localmente en las funciones no tiene efecto alguno.

Variables persistentes

Las variables persistentes se identifican mediante la palabra clave `PERSISTENT` (`VAR_GLOBAL PERSISTENT`). Solamente se reinicializan al ejecutar el comando **En línea Reset origen**. A diferencia de las variables Retain, conservan sus valores después de una descarga.

Ejemplo de aplicación:

Un contador de horas de funcionamiento, que debe continuar contando incluso después de un corte en el suministro eléctrico o una descarga. Consulte la tabla sinóptica referente al comportamiento de las variables remanentes *(véase página 598)*.

Las variables persistentes solamente se pueden declarar en una lista de variables globales especial de variables persistentes de tipo de objeto, que se asigna a una aplicación. Solamente se puede añadir una única lista de este tipo a una aplicación.

NOTA: Una declaración con `VAR_GLOBAL PERSISTENT` tiene el mismo efecto que una declaración con `VAR_GLOBAL PERSISTENT RETAIN` o `VAR_GLOBAL RETAIN PERSISTENT`.

Al igual que las variables Retain, las variables persistentes se almacenan en un área de la memoria por separado.

Ejemplo

```
VAR_GLOBAL PERSISTENT RETAIN
  iVarPers1 : DINT; (* 1. Persistent+Retain Variable App1 *)
  bVarPers : BOOL;  (* 2. Persistent+Retain Variable App1 *)
END_VAR
```

NOTA: Las variables persistentes solamente se pueden declarar dentro del objeto **Persistent list**. Si se declaran en otro lugar, se comportarán como variables Retain y se notificarán como un error de **compilación** detectado en la vista **Mensajes**. (Las variables Retain se pueden declarar en las listas de variables globales o en las POU).

Cada vez que la aplicación se vuelva a cargar, la lista de variables persistentes del controlador se comparará con la del proyecto. La lista del controlador se identifica mediante el nombre de la aplicación. En caso de discrepancia, se le pedirá que reinicialice todas las variables persistentes de la aplicación. La discrepancia puede ser el resultado de cambiar el nombre de las declaraciones existentes en la lista, de eliminar dichas declaraciones o de realizar otro tipo de modificaciones en ellas.

NOTA: Reflexione detenidamente sobre cualquier modificación en la parte de declaración de la lista de variables persistentes y el efecto de los resultados en lo que respecta a la reinicialización.

Puede añadir declaraciones nuevas únicamente al final de la lista. Durante una descarga, se detectarán como nuevas y no exigirán una reinicialización de la lista completa.

Comportamiento de las variables remanentes

Para obtener más información sobre el comportamiento de las variables remanentes del controlador, consulte la *guía de programación* correspondiente al controlador que utilice.

Constantes - CONSTANT

Las constantes se identifican mediante la palabra clave `CONSTANT`. Puede declararlas local o globalmente.

Sintaxis

`VAR CONSTANT`<identificador>:<tipo> := <inicialización>;`END_VAR`

Ejemplo

```
VAR CONSTANT
 c_iCon1:INT:=12; (* 1. Constant*)
END_VAR
```

Consulte el capítulo *Operandos (véase página 797)* para ver una lista de las constantes posibles.

Literales con tipo

Al utilizar constantes IEC, básicamente se utilizará el tipo de datos más pequeño posible. Si tiene que utilizarse otro tipo de datos, se puede hacer con la ayuda de literales con tipo sin necesidad de declarar explícitamente las constantes. Para ello, la constante se proporcionará con un prefijo que indica el tipo.

Sintaxis

<tipo>#<literal>;

<tipo>	Especifica el tipo de datos que se desea. Entradas posibles: BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, LREAL Escriba el tipo con letras mayúsculas.
<literal>	Especifica la constante. Introduzca datos que se ajusten al tipo de datos especificado en <tipo>.

Ejemplo

```
iVar1:=DINT#34;
```

Si la constante no se puede convertir al tipo de destino sin sufrir una pérdida de datos, aparece un mensaje.

Puede utilizar los literales con tipo allí donde se pueda utilizar una constante normal.

Constantes en modalidad online

Siempre que la configuración predeterminada **Replace constants** (**Archivo** →**Configuración del proyecto** →**Opciones de compilador**) esté activada, las constantes en modalidad online tendrán un símbolo  delante del valor en la columna **Valor** de la vista de supervisión o declaración. En este caso, no se puede acceder a ellas mediante forzado o escritura, por ejemplo.

Configuración de variables - VAR_CONFIG

Descripción general

Puede utilizar la configuración de variables para asignar variables de bloques de funciones a la imagen del proceso en las E/S del dispositivo. Así se evita la necesidad de especificar la dirección ya definida en la declaración de la variable del bloque de funciones. La asignación de la dirección *(véase página 813)* definida en este caso está centralizada para todas las instancias de bloques de funciones en una lista de VAR_CONFIG global.

Con este propósito, puede asignar direcciones incompletas a las variables del bloque de funciones declaradas entre las palabras clave VAR y END_VAR. Utilice un asterisco para identificar estas direcciones.

Sintaxis del identificador

<identificador> AT %<I|Q>* : <tipo de datos>

Ejemplo de utilización de direcciones no definidas por completo:

```
FUNCTION_BLOCK locio
VAR
 xLocIn AT %I*: BOOL := TRUE;
 xLocOut AT %Q*: BOOL;
END_VAR
```

En este ejemplo se definen dos variables locales de E/S: una entrada local (%I*) y una variable de salida local (%Q*).

Defina las direcciones en la configuración de variables en una lista de variables globales (GVL) de la siguiente manera:

Paso	Acción
1	Ejecute el comando **Agregar objeto**.
2	Agregue un objeto de **Lista de variables globales** (GVL) al **Dispositivos**.
3	Introduzca las declaraciones de las variables de instancia con la dirección definida entre las palabras clave VAR_CONFIG y END_VAR.

Al definir las direcciones tenga en cuenta lo siguiente:

- Especifique las variables de instancia con la ruta completa de la instancia y separe los POU individuales y los nombres de instancia por períodos.
- Introduzca una dirección cuya clase (entrada/salida) se corresponda con la de la dirección incompleta especificada %I*, %Q*) en el bloque de funciones.
- Verifique que el tipo de datos coincida con la declaración en el bloque de funciones.

Sintaxis de ruta de variables de instancias

<ruta de variable de instancia> `AT` %<I|Q><ubicación> : <tipo de datos>;

Las variables de configuración cuya ruta de instancia no sea válida porque la instancia no existe se identifican como errores detectados. También se detectará un error si no existe una configuración de dirección definida para una variable de instancia asignada a una dirección incompleta.

Ejemplo de configuración de variable

Supongamos que en un programa se da la siguiente definición para el bloque de funciones `locio` (véase el ejemplo anterior):

```
PROGRAM PLC_PRG
VAR
locioVar1: locio;
locioVar2: locio;
END_VAR
```

Entonces, una configuración de variable corregida sería:

```
VAR_CONFIG
PLC_PRG.locioVar1.xLocIn AT %IX1.0 : BOOL;
PLC_PRG.locioVar1.xLocOut AT %QX0.0 : BOOL;
PLC_PRG.locioVar2.xLocIn AT %IX1.0 : BOOL;
PLC_PRG.locioVar2.xLocOut AT %QX0.3 : BOOL;
END_VAR
```

NOTA: Las modificaciones en las E/S directamente asignadas se muestran inmediatamente en la imagen del proceso, mientras que las modificaciones realizadas en las variables asignadas mediante `VAR_CONFIG` no se muestran hasta el final de la tarea responsable.

Sección 27.3
Tipos de métodos

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Métodos FB_init y FB_reinit

FB_Init

El método FB_init se utiliza para reinicializar un bloque de funciones o estructura. FB_init puede declararse de forma explícita para bloques de funciones, pero en cualquier caso siempre está implícitamente disponible.

El método FB_init contiene un código de inicialización para el bloque de funciones o estructura tal y como se declara en la parte de declaraciones del objeto correspondiente. Si el método init se declara explícitamente, el código de inicialización implícito se insertará dentro de este método. El programador puede ampliar el código de inicialización.

NOTA: Cuando la ejecución alcanza el código de inicialización definido por el usuario, el bloque de funciones, o la estructura, está ya completamente inicializado mediante el código de inicialización implícito.

El método init es llamado después de la descarga para cada una de las instancias de bloque de funciones declaradas y cada variable de un tipo de estructura.

NOTA: La ejecución del comando **Cambio en línea** sobrescribirá los valores de inicialización sustituyéndolos por valores anteriores.

Para obtener más información acerca de la secuencia de llamadas en caso de herencia, consulte el apartado correspondiente del capítulo del *método FB_exit (véase página 606)*.

Para obtener más información acerca de la posibilidad de definir un método de instancia de bloque de funciones que sea llamado automáticamente después de la inicialización mediante FB_init consulte el capítulo *Attribute call_after_init (véase página 624)*.

Interfaz del método init

```
METHOD fb_init : BOOL
VAR_INPUT
  bInitRetains : BOOL; // if TRUE, the retain variables are initialized
 (warm start / cold start)
  bInCopyCode : BOOL; // if TRUE, the instance afterwards
gets moved into the copy code (online change)
END_VAR
```

El valor de retorno no es secuencial.

NOTA: Considere también la posibilidad de utilizar un método de salida y la orden de ejecución resultante (consulte el capítulo del *método FB_exit (véase página 606)*).

Entrada definida por el usuario

Puede declarar entradas de bloques de funciones adicionales en un método init Asígnelos a la declaración de una instancia de bloque de funciones.

Ejemplo de método `init` para un bloque de funciones `serialdevice`:

```
METHOD PUBLIC FB_init : bool
VAR_INPUT
    bInitRetains : BOOL; // Initialization of the retain variables
    bInCopyCode : BOOL; // Instance moved into copy code
    nCOMnum : INT; // Input: number of the COM interface to listen at
END_VAR
```

Declaración del bloque de funciones `serialdevice`:

```
    com1: serialdevice (nCOMnum:=1);
    com0: serialdevice (nCOMnum:=0);
```

Ejemplo de utilización de `FB_init` para una estructura `DUTxy`:

Estructura `DUTxy`

```
TYPE DUTxy :
STRUCT
    a: INT := 10;
    b: INT := 11;
    c: INT := 12;
END_STRUCT
END_TYPE
```

Llamada a `fb_init` para reinicialización:

```
PROGRAM PLC_PRG
VAR
    dutTest: DUTxy;
    xInit: BOOL:=FALSE;
END_VAR
IF xinit THEN // if xinit is set TRUE, then the
reinitialization via fb_init to the values as defined in DUTxy will be
done
    dutTest.FB_Init(TRUE,TRUE);
ELSE
    dutTest.a := 1;
    dutTest.b := 2;
    dutTest.c := 3;
END_IF
```

FB_reinit

Si un método con el nombre `FB_reinit` es declarado para una instancia de bloque de funciones, este método será llamado una vez que se copie la instancia. Este es el caso de un cambio en línea después de una modificación de la declaración del bloque de funciones. El método provocará una reinicialización del nuevo módulo de instancia creado por la copia del código. Puede ser conveniente llevar a cabo una reinicialización debido a que los datos de la instancia original serán escritos en una nueva instancia después del copiado, aún cuando quiera obtener los valores de inicialización originales. El método `FB_reinit` no tiene entradas. Tenga en cuenta que, al contrario de lo que ocurre con `FB_init` , el método debe ser declarado explícitamente. Si se desea una reinicialización de la implementación del bloque de funciones básico, `FB_reinit` debe ser llamado explícitamente para ese POU.

El método `FB_reinit` no tiene entradas.

Para obtener más información acerca de la secuencia de llamadas en caso de herencia, consulte el apartado correspondiente del capítulo del *método FB_exit (véase página 606)*.

Método FB_exit

Descripción general

El método FB_exit es un método especial para un bloque de funciones. Ha de declararse de forma explícita, ya que no hay una declaración implícita. El método exit, si lo hubiese, se solicita para todas las instancias declaradas del bloque de funciones antes de una nueva descarga, en un inicio o durante un cambio online de todas las instancias movidas o eliminadas.

Interfaz del método FB_exit

Solo hay un parámetro obligatorio:

```
METHOD fb_exit : BOOL
VAR_INPUT
 bInCopyCode : BOOL; // if TRUE, the exit method is called
for exiting an instance that is copied afterwards (online change).
END_VAR
```

Asimismo, tenga en cuenta el *método FB_init (véase página 603)* y la siguiente orden de ejecución:

1	Método exit: salir de una instancia anterior	`old_inst.fb_exit(bInCopyCode := TRUE);`
2	Método init: inicializar una nueva instancia	`new_inst.fb_init(bInitRetains := FALSE, bInCopyCode := TRUE);`
3	Copiar valores del bloque de funciones (código de copia)	`copy_fub(&old_inst, &new_inst);`

Herencia

Además, en caso de herencia, la secuencia de llamada siguiente es TRUE (suponiendo lo siguiente para los POU usados para este listado: SubFB EXTENDS MainFB y SubSubFB EXTENDS SubFB):

```
fbSubSubFb.FB_Exit(...);
fbSubFb.FB_Exit(...);
fbMainFb.FB_Exit(...);
fbMainFb.FB_Init(...);
fbSubFb.FB_Init(...);
fbSubSubFb.FB_Init(...);
```

Para FB_reinit, se aplica lo siguiente:

```
fbMainFb.FB_reinit(...);
fbSubFb.FB_reinit(...);
fbSubSubFb.FB_Init(...);
```

Sección 27.4
Instrucciones Pragma

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Instrucciones Pragma

Descripción general

Las instrucciones pragma se utilizan para afectar a las propiedades de una o varias variables respecto al proceso de compilación o precompilación (preprocesador). Es decir, un pragma afecta a la generación del código.

NOTA: Tenga en cuenta que los pragmas disponibles no son implementaciones 1:1 de las directivas del preprocesador de C. Se gestionan como una instrucción normal por lo que solamente pueden utilizarse en las posiciones de instrucciones.

Un pragma puede determinar si una variable será inicializada, monitoreada, añadida a una lista de parámetros, añadida a la lista de símbolos *(véase página 655)*, o si es invisible en el **administrador de bibliotecas**. Puede forzar salidas de mensajes durante el proceso de compilación. Puede utilizar pragmas condicionales para definir la manera en que la variable debe ser tratada en función de ciertas condiciones. También puede introducir estos pragmas como definiciones en las propiedades de compilación de un objeto determinado.

Puede utilizar un pragma en una línea separada o con texto complementario en una implementación o línea del editor de declaraciones. Ejecute el comando **Insertar etiqueta** dentro del editor de IL/FBD/LD y reemplace el texto por defecto `Label:` en el campo de texto que aparece por el pragma. En caso de que desee establecer también una etiqueta además de un pragma, introduzca primero el pragma y después la etiqueta.

La instrucción del pragma aparece entre llaves.

Sintaxis

{ <texto de instrucción> }

La llave de apertura puede aparecer inmediatamente después de un nombre de variable. Las llaves de apertura y cierre deben estar en la misma línea.

Información adicional

En función del tipo y contenidos de un pragma, el pragma opera en la siguiente instrucción, y en todas las posteriores, hasta que se cumpla una de las siguientes condiciones:

- Es finalizado por un pragma apropiado.
- El mismo pragma es ejecutado con diferentes parámetros.
- Se alcanza el final del código.

En este contexto el término código se refiere a una parte de declaración, una parte de implementación, una lista de variables globales o una declaración de tipo.

NOTA: Las instrucciones de Pragma distinguen entre mayúsculas y minúsculas.

Si el compilador no puede interpretar de manera significativa el texto de la instrucción, todo el pragma se gestiona como un comentario y se omite.

Consulte los siguientes tipos de pragma:

- *Pragmasde mensaje (véase página 610)*
- *Atributo obsoleto (véase página 650)*
- *Pragmasde atributo (véase página 621)*
- *Pragmascondicionales (véase página 612)*
- *Símbolo de atributo (véase página 621)*

Pragmas del mensaje

Descripción general

Puede utilizar los pragmas del mensaje para forzar la salida de mensajes en la vista **Mensajes** (de manera predeterminada en el menú **Editar**) durante la compilación del proyecto.

Puede insertar la instrucción pragma en una línea existente o en una línea separada en el editor de textos de un POU. Los pragmas del mensaje situados dentro de las secciones no definidas actualmente del código de implementación no se considerarán cuando se compile el proyecto. Para obtener más información, consulte el ejemplo proporcionado con la descripción del (identificador) definido en el capítulo *Pragmas condicionales* *(véase página 612)*.

Tipos de Pragmas del mensaje

Existen cuatro tipos de pragmas del mensaje:

Pragma	Icono	Tipo de mensajes
{texto 'cadena de texto'}	–	tipo de texto Se mostrará la cadena de texto especificada.
{información 'cadena de texto'}		información Se mostrará la cadena de texto especificada.
{dígito de advertencia 'cadena de texto'}		tipo de alerta Se mostrará la cadena de texto especificada. A diferencia del pragma obsoleto *(véase página 650)* global, esta alerta se define de manera explícita en la posición local.
{error 'cadena de texto'}		tipo de error Se mostrará la cadena de texto especificada.

NOTA: En los mensajes de información de tipos, alertas y errores detectados, puede alcanzar la posición del código de origen del mensaje (es decir, cuando el pragma se reemplaza en un POU) ejecutando el comando **Mensaje siguiente**. Esto no es posible en el tipo de texto.

Ejemplo de declaración e implementación en el editor ST

```
VAR
ivar : INT; {info 'TODO: should get another name'}
bvar : BOOL;
arrTest : ARRAY [0..10] OF INT;
i:INT;
END_VAR
arrTest[i] := arrTest[i]+1;
ivar:=ivar+1;
{warning 'This is an alert'}
{text 'Part xy has been compiled completely'}
```

Salida en la vista **Mensajes**:

Mensajes

Compilar

Descripción	Proyecto	Objeto	Posición
------ Generación iniciada: Aplicación: Res.App2 -------			
Tipificar código...			
Tiempo de compilación antes de la tipificación: 0 ms			
Tiempo de compilación tras la tipificación: 15 ms			
PENDIENTE: debe obtenerse otro nombre.	TS pragma	NewPOU	Línea 3 (Decl)
Esto es una alerta	TS pragma	NewPOU	Línea 7 (Impl)
Componente xy compilado completamente	TS pragma	NewPOU	Línea 10 (Impl)
Compilación completada -- 1 error, 2 advertencias			

Pragmas condicionales

Descripción general

El idioma ExST (ST extendido) admite diversas *instrucciones Pragma* *(véase página 608)*, condicionales que afectan la generación de códigos en el proceso de precompilación o compilación.

El código de implementación que se utiliza para la compilación puede depender de las condiciones siguientes:

- ¿Se trata de determinado tipo de datos o variables declarado?
- ¿Un tipo o una variable tienen un atributo determinado?
- ¿Tiene una variable un tipo de datos determinado?
- ¿Es una POU determinada, una tarea disponible o forma parte de un árbol de llamadas?

NOTA: Un POU o GVL declaradas en el **Árbol de POU** no puede utilizar un `{define...}` declarado en una aplicación. Las definiciones en las aplicaciones solo afectarán a interfaces insertadas bajo la aplicación correspondiente.

`{define identifier string}`	Durante el procesamiento previo, todas las instancias del identificador que se ejecuten a partir de ese momento se reemplazarán con la secuencia de tokens dada si la cadena del token no está vacía (está permitida y bien definida). El identificador permanece definido y dentro del ámbito hasta el final del objeto o hasta que esté sin definir en una directiva `{undefine}` . Se emplea para compilaciones condicionales *(véase página 613)*.
`{undefine identifier}`	Se eliminará la definición del preprocesador del `identifier` (por `{define}`, consulte la primera fila de esta tabla) y, por tanto, el identificador estará sin definir. Si el identificador especificado no está actualmente definido, se ignorará este pragma.

`{IF expr}`
...
`{ELSIF expr}`
...
`{ELSE}`
...
`{END_IF}`
	Estos son pragmas para compilaciones condicionales. Las expresiones especificadas `exprs` son necesarias para ser constante en el tiempo de compilación; se evalúan en el orden en el que aparecen hasta que una de las expresiones se evalúa como un valor diferente de cero. El texto asociado con la directiva correcta se preprocesa y compila con normalidad y el resto se ignora. Se determina el orden de las secciones, no obstante, las secciones `elsif` y `else` son opcionales, y las secciones `elsif` pueden aparecer de forma arbitraria más a menudo.
	Dentro de la constante `expr`, se pueden utilizar diversos operadores de compilación condicionales *(véase página 613)*.

Operadores de compilación condicional

Dentro de la expresión de la constante `expr` de una compilación condicional pragma (`{if}` o `{elsif}`) (consulte la tabla anterior), se pueden utilizar diversos operadores. Es posible que no se puedan no definir ni redefinir estos operadores mediante `{undefine}` o `{define}`, respectivamente. Teniendo en cuenta que también puede utilizar estas expresiones, así como la definición completada por `{define}` en el campo de texto **Definiciones de compilador:** del cuadro de diálogo **Propiedades** de un objeto (**Ver** →**Propiedades** →**Compilar**).

Son compatibles los siguientes operadores:

- defined (identifier) *(véase página 613)*
- defined (variable:variable) *(véase página 614)*
- defined (type:identifier) *(véase página 614)*
- defined (pou:pou-name) *(véase página 614)*
- hasattribute (pou: pou-name, attribute) *(véase página 615)*
- hasattribute (variable: variable, attribute) *(véase página 616)*
- hastype (variable:variable, type-spec) *(véase página 616)*
- hasvalue (define-ident, char-string) *(véase página 618)*
- NOT operator *(véase página 618)*
- operator AND operator *(véase página 618)*
- operator OR operator *(véase página 619)*
- operator *(véase página 619)*

`defined (identifier)`

Este operador hace que la expresión adopte el valor TRUE, en cuanto se haya definido el `identifier` con una instrucción `{define}` y no haya sido no definido posteriormente por una instrucción `{undefine}` . En caso contrario su valor será FALSE.

Ejemplo en `defined (identifier)`:

Precondición: hay 2 aplicaciones `App1` y `App2`. La variable `pdef1` está declarada en la `App2`, pero no en la `App1`.

```
{IF defined pdef1)}
(* this code is processed in App1 *)
{info 'pdef1 defined'}
hugo := hugo + SINT#1;
{ELSE}
(* the following code is only processed in application App2 *)
{info 'pdef1 not defined'}
hugo := hugo - SINT#1;
```

Además, se incluye un ejemplo para un mensaje pragma *(véase página 610)*:

Solo se mostrará la información de `pdef1 defined` en la vista **Mensajes** cuando se compila la aplicación porque `pdef1` está definido. El mensaje **pdef1 no definido** se mostrará cuando `pdef1` no esté definido.

defined (variable:variable)

Cuando se aplica a una variable, su valor es TRUE si esta variable en particular está declarada dentro del ámbito actual. De lo contrario, es FALSE.

Ejemplo en `defined (variable:variable)`:

Precondición: hay 2 aplicaciones `App1` y `App2`. La variable `g_bTest` está declarada en la `App2`, pero no en la `App1`.

```
{IF defined (variable:g_bTest)}
(* the following code is only processed in application App2 *)
g bTest := x > 300;
{END_IF}
```

defined (type:identifier)

Cuando se aplica a un identificador de tipo, su valor es TRUE si existe un tipo declarado con ese nombre en particular. De lo contrario, es FALSE.

Ejemplo en `defined (type:identifier)` :

Precondición: hay 2 aplicaciones `App1` y `App2`. El tipo de datos DUT está definido en la `App2` pero no en la `App1`.

```
{IF defined (type:DUT)}
(* the following code is only processed in application App1 *)
bDutDefined := TRUE;
{END_IF}
```

defined (pou:pou-name)

Cuando se aplica a un nombre de POU, su valor es TRUE si se define un POU o una acción con ese nombre de POU determinado. De lo contrario, es FALSE.

Ejemplo en `defined (pou: pou-name)`:

Precondición: hay 2 aplicaciones `App1` y `App2`. Los `CheckBounds` del POU están definidos en la `App2`, pero no en la `App1`.

```
{IF defined (pou:CheckBounds)}
(* the following code is only processed in application App1 *)
arrTest[CheckBounds(0,i,10)] := arrTest[CheckBounds(0,i,10)] + 1;
{ELSE}
(* the following code is only processed in application App2 *)
arrTest[i] := arrTest[i]+1;
{END_IF}
```

hasattribute (pou: pou-name, attribute)

Cuando se aplica a un POU, su valor es TRUE si este `attribute` determinado se especifica en la primer línea de la sección de declaración del POU.

Ejemplo en `hasattribute (pou: pou-name, attribute)`:

Precondición: hay 2 aplicaciones `App1` y `App2`. La función `fun1` está definida en la `App1` y la `App2`, pero en la `App1` tiene un atributo `vision`:

Definición de `fun1` en la `App1`:

```
{attribute 'vision'}
FUNCTION fun1 : INT
VAR_INPUT
i : INT;
END_VAR
VAR
END_VAR
```

Definición de `fun1` en la `App2`:

```
FUNCTION fun1 : INT
VAR_INPUT
i : INT;
END_VAR
VAR
END_VAR
```

Instrucción Pragma

```
{IF hasattribute (pou: fun1, 'vision')}
(* the following code is only processed in application App1 *)
ergvar := fun1 ivar);
{END_IF}
```

hasattribute (variable: variable, attribute)

Cuando se aplica a una `variable`, su valor es TRUE si este atributo determinado se especifica mediante la instrucción `{attribute}`, una línea antes de la declaración de la variable.

Ejemplo en `hasattribute (variable: variable, attribute)`:

Precondición: hay 2 aplicaciones `App1` y `App2`. La variable `g_globalInt` se utiliza en la `App1` y la `App2`, pero en la `App1` tiene un atributo `DoCount` :

Declaración de `g_globalInt` en la `App1`

```
VAR_GLOBAL
{attribute 'DoCount'}
g_globalInt : INT;
g_multiType : STRING;
END_VAR
```

Declaración de `g_globalInt` en la `App2`

```
VAR_GLOBAL
g_globalInt : INT;
g_multiType : STRING;
END_VAR
```

Instrucción Pragma

```
{IF hasattribute (variable: g_globalInt, 'DoCount')}
(* the following code line will only be processed in App1, because ther
e variable g_globalInt has got the attribute 'DoCount' *)
g_globalInt := g_globalInt + 1;
{END_IF}
```

hastype (variable:variable, type-spec)

Cuando se aplica a una `variable`, su valor es TRUE si esta variable determinada tiene `type-spec` especificado. De lo contrario, es FALSE.

Tipos de datos disponibles en `type-spec`

- ANY
- ANY_DERIVED
- ANY_ELEMENTARY
- ANY_MAGNITUDE
- ANY_BIT
- ANY_STRING
- ANY_DATE
- ANY_NUM
- ANY_REAL
- ANY_INT
- LREAL
- REAL
- LINT

- DINT
- INT
- SINT
- ULINT
- UDINT
- UINT
- USINT
- TIME
- LWORD
- DWORD
- WORD
- BYTE
- BOOL
- STRING
- WSTRING
- DATE_AND_TIME
- DATE
- TIME_OF_DAY

Ejemplo en el operador `hastype (variable: variable, type-spec)`:

Precondición: hay 2 aplicaciones `App1` y `App2`. La variable `g_multitype` está declarada en la `App1` con el tipo LREAL y en la aplicación `App2` con el tipo STRING:

```
{IF (hastype (variable: g_multitype, LREAL))}
(* the following code line will be processed only in App1 *)
g_multitype := (0.9 + g_multitype) * 1.1;
{ELSIF (hastype (variable: g_multitype, STRING))}
(* the following code line will be processed only in App2 *)
g_multitype := 'this is a multitalent';
{END_IF}
```

hasvalue (define-ident, char-string)

Si el definidor (define-ident) está definido y tiene el valor especificado (char-string), entonces su valor es TRUE. De lo contrario, es FALSE.

Ejemplo en hasvalue (define-ident, char-string):

Precondición: la test de variables se utiliza en las aplicaciones App1 y App2. Obtiene el valor 1 en la App1 y el valor 2 en la App2:

```
{IF hasvalue(test,'1')}
(* the following code line will be processed in App1, because there var
iable test has value 1 *)
x := x + 1;
{ELSIF hasvalue(test,'2')}
(* the following code line will be processed in App1, because there var
iable test has value 2 *)
x := x + 2;
{END_IF}
```

NOT operator

La expresión obtiene el valor TRUE cuando el valor invertido del operator es TRUE. El operator puede ser uno de los operadores descritos en este capítulo.

Ejemplo en el NOT operator:

Precondición: hay 2 aplicaciones App1 y App2. El POU PLC_PRG1 se utiliza en la App1 y la App2. Los CheckBounds del POU solo están disponibles en la App1:

```
{IF defined (pou: PLC_PRG1) AND NOT (defined (pou: CheckBounds))}
(* the following code line is only executed in App2 *)
bANDNotTest := TRUE;
{END_IF}
```

AND operator

La expresión obtiene el valor TRUE si ambos operadores son TRUE. El operator puede ser uno de los operadores que aparecen en esta tabla.

Ejemplo en AND operator:

Precondición: hay 2 aplicaciones App1 y App2. El POU PLC_PRG1 se utiliza en las aplicaciones App1 y App2. Los CheckBounds del POU solo están disponibles en la App1:

```
{IF defined (pou: PLC_PRG1) AND (defined (pou: CheckBounds))}
(* the following code line will be processed only in applications App1,
 because only there "PLC_PRG1" and "CheckBounds" are defined *)
bORTest := TRUE;
{END_IF}
```

OR operator

La expresión es TRUE si uno de los operadores es TRUE. El `operator` puede ser uno de los operadores descritos en este capítulo.

Ejemplo en `OR operator`:

Precondición: el POU `PLC_PRG1` se utiliza en las aplicaciones `App1` y `App2`. Los `CheckBounds` del POU solo están disponibles en la `App1`:

```
{IF defined (pou: PLC_PRG1) OR (defined (pou: CheckBounds))}
(* the following code line will be processed in applications App1 and A
pp2, because both contain at least one of the POUs "PLC_PRG1" and "Chec
kBounds" *)
bORTest := TRUE;
{END_IF}
```

(operator)

`(operator)` operador entre llaves.

Sección 27.5
Atributo Pragmas

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Pragmas de atributos

Descripción general

Puede asignar pragmas de atributos *(véase página 608)* a una firma para influir en la compilación o precompilación que es la generación de código.

Hay atributos definidos por el usuario *(véase página 622)* que puede usar en combinación con pragmas condicionales *(véase página 612)*.

También están los siguientes pragmas de atributos estándar predefinidos:

- attribute displaymode *(véase página 625)*
- attribute ExpandFully *(véase página 626)*
- attribute global_init_slot *(véase página 628)*
- attribute hide *(véase página 629)*
- attribute hide_all_locals *(véase página 630)*
- attribute initialize_on_call *(véase página 631)*
- attribute init_namespace *(véase página 632)*
- attribute init_On_Onlchange *(véase página 633)*
- attribute instance-path *(véase página 634)*
- attribute linkalways *(véase página 635)*
- attribute monitoring *(véase página 637)*
- attribute namespace *(véase página 641)*
- attribute no_check *(véase página 643)*
- attribute no_copy *(véase página 644)*
- attribute no-exit *(véase página 645)*
- attribute noinit *(véase página 646)*
- attribute no_virtual_actions *(véase página 647)*
- attribute obsolete *(véase página 650)*
- attribute pack_mode *(véase página 651)*
- attribute qualified_only *(véase página 652)*
- attribute reflection *(véase página 653)*
- attribute subsequent *(véase página 654)*
- attribute symbol *(véase página 655)*
- attribute warning disable *(véase página 657)*

Atributos definidos por el usuario

Descripción general

Puede asignar pragmas de atributos arbitrarios definidos por el usuario o por la aplicación a POU, declaraciones de tipo o variables. Este atributo se puede consultar antes de la compilación mediante pragmas condicionales *(véase página 612)*.

Sintaxis

{atributo 'atributo'}

Esta instrucción pragma es válida para la declaración de variable o declaración del POU siguiente.

Puede asignar un atributo definido por el usuario a:
- un POU o acción
- una variable
- un tipo de datos

Ejemplo de POU y acciones

Atributo `vision` para función `fun1`:

```
{attribute 'vision'}
FUNCTION fun1 : INT
VAR_INPUT
i : INT;
END_VAR
VAR
END_VAR
```

Ejemplo de variables

Atributo `DoCount` para variable `ivar`:

```
PROGRAM PLC_PRG
VAR
{attribute 'DoCount'};
ivar:INT;
bvar:BOOL;
END_VAR
```

Ejemplo de tipos

Atributo `aType` para tipo de datos DUT_1:

```
{attribute 'aType'}
TYPE DUT_1 :
STRUCT
a:INT;
b:BOOL;
END_STRUCT
END_TYPE
```

Para el uso de pragmas condicionales, consulte el capítulo *Pragmas condicionales* *(véase página 612)*.

Attribute call_after_init

Descripción general

Use el pragma `{attribute call_after_init}` para definir un método al que se invoca de forma implícita tras la inicialización de una instancia de bloque de funciones. Para ello, adjunte el atributo tanto al propio bloque de funciones como al método de instancia al que se invoca (por motivos de rendimiento). El método tiene que invocarse después de FB_Init *(véase página 603)* y después de haber aplicado los valores de las variables de una expresión de inicialización en la declaración de instancia.

Esta funcionalidad se admite desde SoMachine V3.0.

Sintaxis

{attribute 'attribute call_after_init'}

Ejemplo

Con la siguiente definición:

```
{attribute 'call_after_init'}
FUNCTION_BLOCK FB
... <functionblock definition>
{attribute 'call_after_init'}
METHOD FB_AfterInit
... <method definition>
```

... declaración como la siguiente:

```
inst : FB := (in1 := 99);
```

... producirá la siguiente orden de procesamiento de código:

```
inst.FB_Init();
inst.in1 := 99;
inst.FB_AfterInit();
```

Así, en `FB_Afterinit`, puede reaccionar ante la inicialización definida por el usuario.

Attribute displaymode

Descripción general

Utilice el pragma `{attribute displaymode}` para definir la modalidad de visualización de una variable simple. Esta configuración sobrescribirá la configuración global de la modalidad de visualización de todas las variables de supervisión realizadas a través de los comandos del submenú **Modalidad de visualización** (de forma predeterminada en el menú **Online**).

Sitúe el pragma en la línea anterior a la línea que contiene la declaración de variable.

Sintaxis

{attribute 'displaymode':=<displaymode>}

Se aceptan las siguientes definiciones:

- para visualizar en formato binario

```
{attribute 'displaymode':='bin'}
{attribute 'displaymode':='binary'}
```

- para visualizar en formato decimal

```
{attribute 'displaymode':='dec'}
{attribute 'displaymode':='decimal'}
```

- para visualizar en formato hexadecimal

```
{attribute 'displaymode':='hex'}
{attribute 'displaymode':='hexadecimal'}
```

Ejemplo

```
VAR
   {attribute 'displaymode':='hex'}
   dwVar1: DWORD;
END_VAR
```

Attribute ExpandFully

Descripción general

Utilice el pragma `{attribute 'ExpandFully'}` para que todos los miembros de las matrices se usen como variables de entradas en las visualizaciones referenciadas a las que se puede acceder dentro del cuadro de diálogo **Propiedades de visualización**.

Sintaxis

{attribute 'ExpandFully'}

Ejemplo

La visualización `visu` está pensada para insertarse en un marco dentro de la visualización `visu_main`.

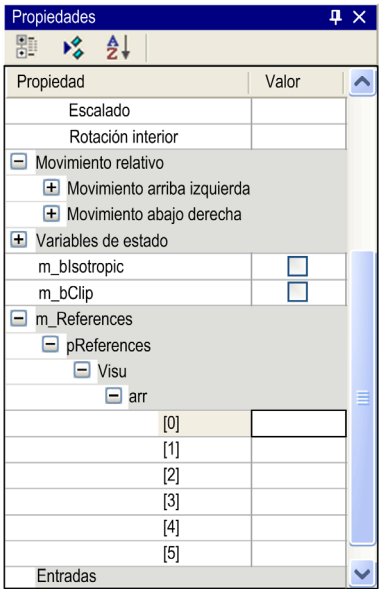
Al igual que la variable de entrada, `arr` se define en el editor de interfaces de `visu` y estará disponible más adelante para las asignaciones en el cuadro de diálogo **Propiedades** del marco de `visu_main`.

Para que los componentes particulares de la matriz estén disponibles en el cuadro de diálogo **Propiedades**, inserte el atributo `ExpandFully` en el editor de interfaces de `visu` justo delante de `arr`.

Declaración en el editor de interfaces de `visu`:

```
VAR_INPUT
{attribute 'ExpandFully'}
arr : ARRAY[0..5] OF INT;
END_VAR
```

El cuadro de diálogo **Propiedades** resultante del marco en `visu_main`:



Attribute global_init_slot

Descripción general

Puede aplicar el pragma `{attribute 'global_init_slot'}` solo en firmas. De forma predeterminada, el orden de inicialización de las variables de una lista de variables globales es arbitrario. Sin embargo, en algunos casos, la prescripción de un orden es necesaria; por ejemplo, si las variables de una lista global hacen referencia a las variables de otra lista. En este caso, puede utilizar el pragma para ordenar la inicialización global.

Sintaxis

{attribute 'global_init_slot' := '<valor>'}

Sustituya la plantilla <valor> por un valor entero que describa el orden de inicialización de la firma. El valor predeterminado es 50,000. Un valor inferior provoca una inicialización precoz. Cuando se trata de firmas que transmiten el mismo valor para el atributo `global_init_slot`, la secuencia de su inicialización permanece indefinida.

Ejemplo

Asuma el proyecto incluidas las dos listas de variables globales `GVL_1` y `GVL_2`.

La variable global `A` es miembro de la lista de variables globales de ejemplo `GVL_1`:

```
{attribute 'global_init_slot' := '300'}
VAR_GLOBAL
A : INT:=1000;
END_VAR
```

Los valores iniciales de las variables `B` y `C` de `GVL_2` dependen de la variable `A`.

```
{attribute'global_init_slot' := '350'}
VAR_GLOBAL
B : INT:=A+1;
C : INT:=A-1;
END_VAR
```

La configuración del atributo `'global_init_slot'` de `GVL_1` en 300, esto es, el valor mínimo del orden de inicialización de ejemplo, sirve para asegurarse de que la expresión `A+1` se ha definido correctamente al inicializar `B`.

Attribute hide

Descripción general

El pragma `{attribute hide}` sirve para evitar que variables o incluso firmas completas se muestren dentro de la funcionalidad de los componentes del listado *(véase página 658)* o la parte de la declaración o la accesibilidad en la modalidad online. Solamente se ocultará la variable posterior al pragma.

Sintaxis

{attribute 'hide'}

Para ocultar todas las variables de una firma, utilice attribute hide_all_locals *(véase página 630)*.

Ejemplo

El bloque de funciones `myPOU` se implementa mediante el atributo:

```
FUNCTION_BLOCK myPOU
VAR_INPUT
a:INT;
{attribute 'hide'}
a_invisible: BOOL;
a_visible: BOOL;
END_VAR
VAR_OUTPUT
b:INT;
END_VAR
VAR
END_VAR
```

Se definen en las dos instancias principales del programa del bloque de funciones `myPOU`:

```
PROGRAM PLC_PRG
VAR
POU1, POU2: myPOU;
END_VAR
```

Al asignar un valor de entrada a `POU1`, la funcionalidad de los componentes del listado *(véase página 658)* que funciona introduciendo `POU1` en la parte de implementación de `PLC_PRG` mostrará las variables `a` y `b`. Las variables locales ocultas `c` o `d` no se mostrarán.

Attribute hide_all_locals

Descripción general

El pragma `{attribute 'hide_all_locals'}` le ayuda a impedir que todas las variables locales de una firma se muestren en la funcionalidad de componentes de listado *(véase página 658)* o el asistente de entrada. Este atributo es idéntico a asignar el atributo hide *(véase página 629)* a cada una de las variables locales.

Sintaxis

{attribute 'hide_all_locals'}

Ejemplo

El bloque de funciones `myPOU` se implementa usando el atributo:

```
{attribute 'hide_all_locals'}
FUNCTION_BLOCK myPOU
VAR_INPUT
a:INT;
END_VAR
VAR_OUTPUT
b:BOOL;
END_VAR
VAR
c,d:INT;
END_VAR
```

En el programa principal se definen dos instancias del bloque de funciones `myPOU`:

```
PROGRAM PLC_PRG
VAR
POU1, POU2: myPOU;
END_VAR
```

Al asignar un valor de entrada a `POU1`, la funcionalidad de componentes de listado *(véase página 658)* que funciona al escribir `POU1` en la parte de implementación de `PLC_PRG` mostrará las variables `a` y `b`. Las variables locales ocultas `c` o `d` no se mostrarán.

Attribute initialize_on_call

Descripción general

Puede añadir el pragma `{attribute initialize_on_call}` a las variables de entrada. Se inicializará una entrada de un bloque de funciones con este atributo en cualquier llamada del bloque de funciones. Si una entrada espera un puntero y este puntero se ha eliminado por un cambio online, la entrada se definirá en NULL.

Sintaxis

{attribute 'initialize_on_call'}

Attribute init_namespace

Descripción general

Una variable de tipo STRING o WSTRING, que está declarada con el pragma `{attribute init_namespace}` en una biblioteca, se inicializará con el espacio de nombres actual de esa biblioteca. Para obtener más información, consulte la descripción de la administración de bibliotecas *(véase SoMachine, Funciones y bibliotecas - Guía del usuario)*.

Sintaxis

{attribute 'init_namespace'}

Ejemplo

El `POU` del bloque de funciones se proporciona con todos los atributos necesarios:

```
FUNCTION_BLOCK POU
VAR_OUTPUT
{attribute 'init_namespace'}
myStr: STRING;
END_VAR
```

En el programa principal `PLC_PRG` se define una instancia `fb` del `POU` del bloque de funciones:

```
PROGRAM PLC_PRG
VAR
fb:POU;
newString: STRING;
END_VAR
newString:=fb.myStr;
```

La variable `myStr` se inicializará con el espacio de nombres actual, por ejemplo `MyLib.XY`. Este valor se asignará a `newString` en el programa principal.

Attribute init_On_Onlchange

Descripción general

Coloque pragma `{attribute 'init_on_onlchange'}` en una variable para inicializar esta variable con cada cambio online.

Sintaxis

{attribute 'init_on_onlchange' }

Attribute instance-path

Descripción general

Puede añadir pragma `{attribute instance-path}` a una variable de cadena local. Esta variable de cadena local se inicializará con la ruta **Árbol de aplicaciones** del POU a la cual pertenece esta variable de cadena. Al aplicar este pragma se asume el uso de attribute reflection *(véase página 653)* para el POU correspondiente y el atributo noinit *(véase página 646)* adicional para la variable de cadena.

Sintaxis

{attribute 'instance-path'}

Ejemplo

Suponiendo que el siguiente bloque de funciones `POU` está equipado con el atributo `'reflection'`:

```
{attribute 'reflection'}
FUNCTION_BLOCK POU
VAR
{attribute 'instance-path'}
{attribute 'noinit'}
str: STRING;
END_VAR
```

En el programa principal `PLC_PRG`, se llama a una instancia `myPOU` del bloque de funciones `POU`:

```
PROGRAM PLC_PRG
VAR
myPOU:POU;
myString: STRING;
END_VAR
myPOU();
myString:=myPOU.str;
```

Tras la inicialización de la instancia `myPOU`, la variable de cadena `str` se asigna a la ruta de la instancia `myPOU`, por ejemplo: `PLCWinNT.Application.PLC_PRG.myPOU`. Esta ruta se asignará a la variable `myString` dentro del programa principal.

NOTA: La longitud de una variable de cadena se puede definir de forma arbitraria (par >255). Sin embargo, la cadena se recortará (desde su extremo trasero) si se asigna a una variable de cadena de una longitud más corta.

Attribute linkalways

Descripción general

Use el pragma `{attribute 'linkalways'}` para marcar POU en el compilador de forma que siempre se incluyan en la información de compilación. Como resultado, los objetos con esta opción siempre se compilarán y descargarán en el controlador. Esta opción solo afecta a los POU y a las listas de variables globales (GVL) que se sitúen por debajo de una aplicación o en bibliotecas que se inserten por debajo de una aplicación. La opción del compilador **Ligar siempre** influye del mismo modo.

Sintaxis

{attribute 'linkalways'}

Cuando utiliza el editor de configuración de símbolos, los POU marcados se emplean como base de las variables seleccionables para la configuración de símbolos.

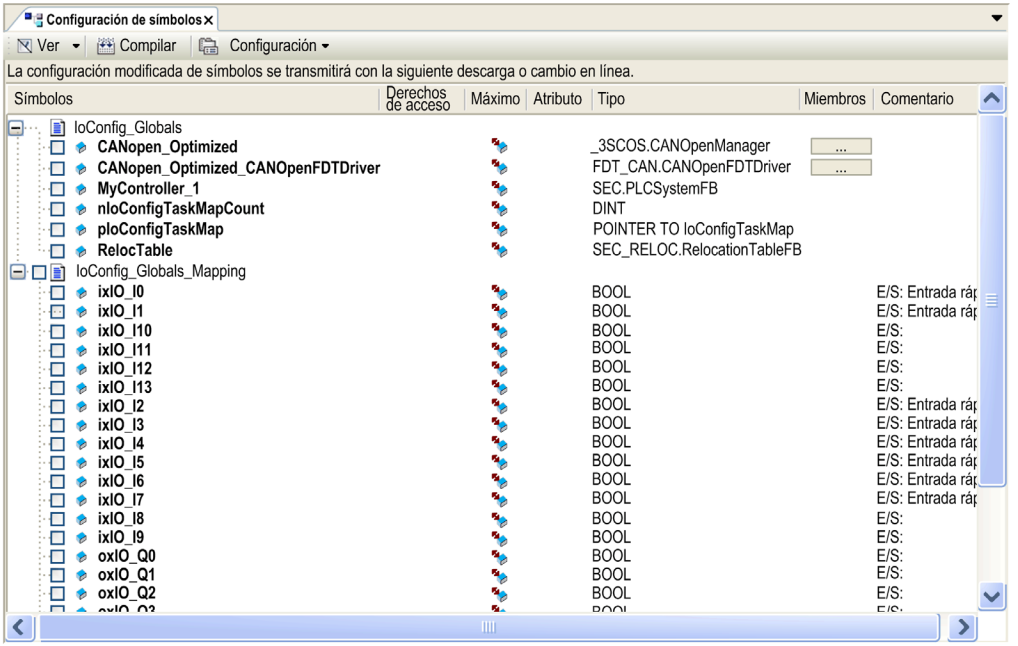
Ejemplo

La lista de variables globales `GVLMoreSymbols` se implementa a través del atributo `'linkalways'`:

```
{attribute 'linkalways'}
VAR_GLOBAS
g_iVar1: INT;
g_iVar2: INT;
END_VAR
```

Con este código, los símbolos de `GVLMoreSymbols` pasan a ser seleccionables en la **Configuración de símbolos**.

Editor de la **configuración de símbolos**



Attribute monitoring

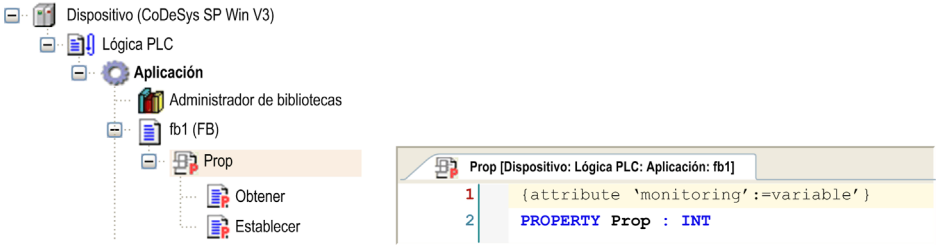
Descripción general

Este pragma de atributo permite supervisar los resultados de propiedades y llamadas de funciones en la vista en línea del editor IEC o en una ventana de supervisión.

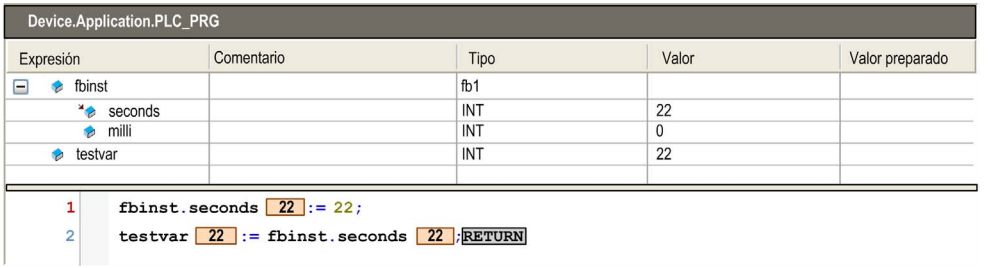
Supervisar propiedades

Añada el pragma en la línea situada por encima de la línea de definición de una propiedad. Se mostrará el nombre, tipo y valor de las variables de la propiedad en la vista en línea del POU que utiliza la propiedad o en una ventana de supervisión. Una vez aquí, podrá introducir valores preparados para las variables de forzado pertenecientes a la propiedad.

Ejemplo de propiedad preparada para la supervisión de variables



Ejemplo de vista de supervisión



Supervisar el valor actual de las variables de la propiedad

Existen dos formas diferentes de supervisar el valor actual de las variables de propiedad. Para un caso específico, tenga en cuenta qué atributo es el adecuado para obtener el valor deseado. Esto dependerá de si las operaciones de las variables son implementadas en la propiedad

1. Pragma {attribute 'monitoring':=}'variable'

Se crea una variable implícita para la propiedad, que obtendrá el valor actual de la propiedad en el momento en que la aplicación haga una llamada a los métodos "get" o "set". El último valor almacenado en esta variable implícita será supervisado.

Sintaxis

{attribute 'monitoring':='variable'}

2. Pragma {attribute 'monitoring':='call'}

Solo podrá utilizar este atributo para propiedades que retornen tipos de datos simples o punteros, no para tipos estructurados.

El valor que va a ser supervisado es recuperado por una llamada directa de propiedad: el servicio de supervisión del sistema en tiempo de ejecución hace una llamada al método `get` y la función de la propiedad se ejecutará.

NOTA: Al elegir este tipo de supervisión en lugar de utilizar una variable intermedia (consulte *1. Pragma*), tenga en cuenta los posibles efectos inesperados que pueden tener lugar si alguna de las operaciones de la variable está implementada en la propiedad.

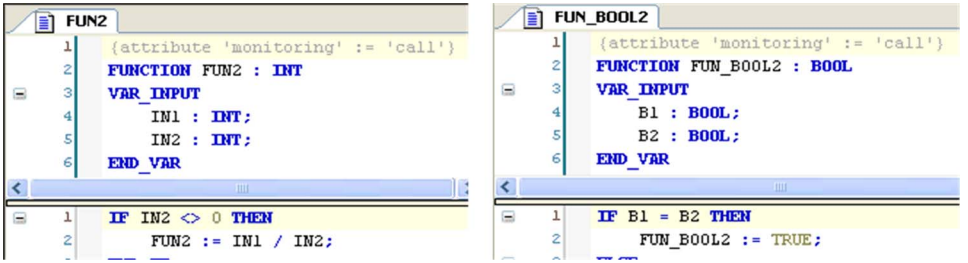
Sintaxis

{attribute 'monitoring':='call'}

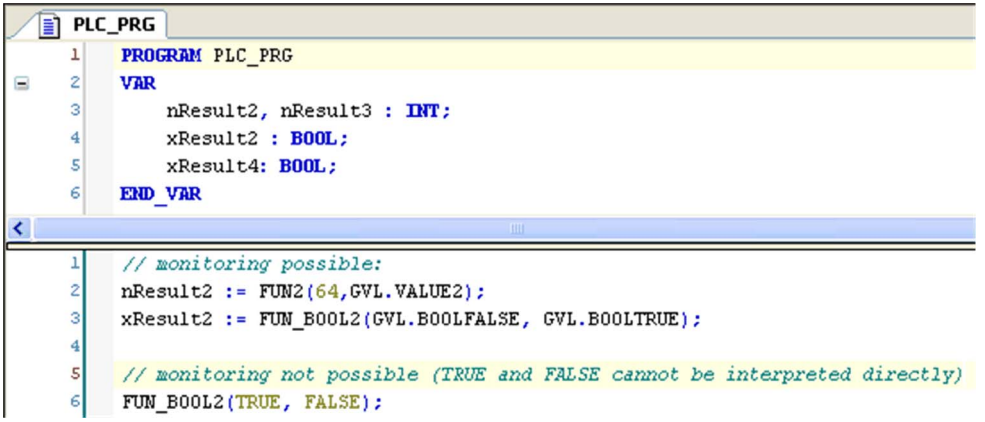
Supervisión de los resultados de llamada de función

Puede utilizar la supervisión de llamada de función para cualquier valor constante que pueda interpretarse como un valor numérico de 4 bytes (por ejemplo, INT, SHORT, LONG). Para los otros parámetros de entrada (por ejemplo, BOOL), utilice una variable en lugar de un parámetro de constante. Añada el pragma `{attribute 'monitoring':='call'}` a la línea sobre la declaración de función. Podrá supervisar esta variable en la vista del editor de texto en la vista en línea del POU en el que a una variable se le asigna el resultado de una llamada de función. También puede agregar la variable a una ventana de supervisión con el mismo propósito. Para añadir inmediatamente la variable a la ventana de supervisión, ejecute el comando **Add watchlist**.

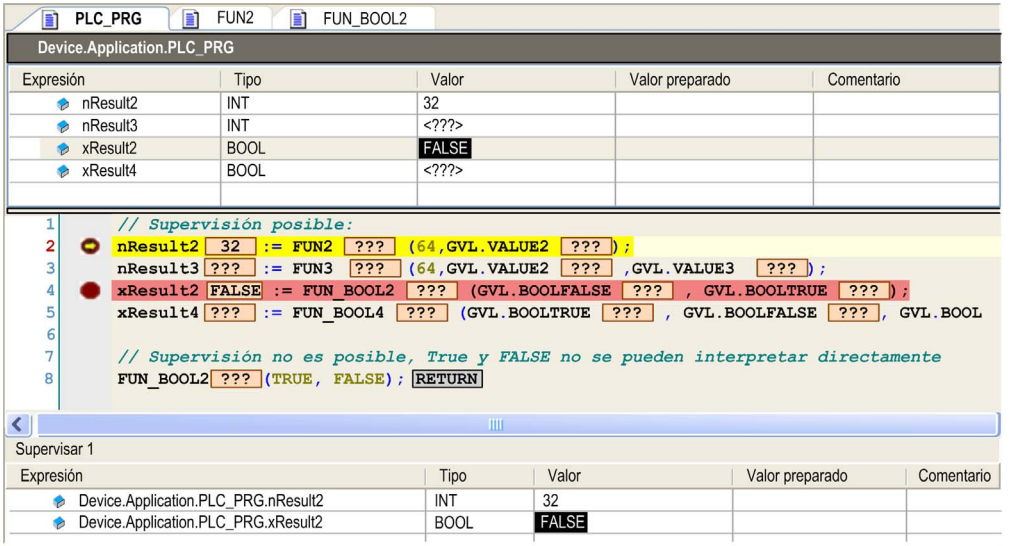
Ejemplo 1: Funciones `FUN2` y `FUN_BOOL2` con el atributo `'monitoring'`



Ejemplo 2: Llamada de funciones `FUN2` y `FUN_BOOL2` en un programa POU



Ejemplo 3: Llamadas de funciones en modalidad en línea:



Supervisión de variables con una llamada implícita de una función externa

Para supervisar variables con una llamada implícita de una función externa deben cumplirse las siguientes condiciones:

- La función está marcada con `{attribute 'monitoring' := 'call'}`.
- La función está marcada como **Ligar siempre**.
- La variable está marcada con `{attribute 'monitoring_instead' := 'MyExternalFunction(a,b,c)'}`.
- Los valores `a,b,c` son valores enteros y se corresponden con los parámetros de entrada de la función que debe llamarse.

NOTA: No compatible con la escritura o forzado de funciones. Puede implementar de manera implícita el forzado añadiendo un parámetro de entrada adicional para la función específica que funciona como indicador de forzado interno.

NOTA: No es posible supervisar la función en el sistema de tiempo en ejecución compacto.

Attribute namespace

Descripción general

En combinación con el attribute symbol *(véase página 655)*, el pragma `{attribute namespace}` sirve para redefinir el espacio de nombres de las variables del proyecto. Puede aplicarlo a los POU completos, como GVL o programas, pero no a variables particulares. Las variables afectadas se exportarán con la nueva definición del espacio de nombres a un archivo de símbolos cuando se pueda descargar este archivo en el controlador.

Esto le permitirá acceder a las variables de los POU o las visualizaciones que tuviesen originalmente espacios de nombres diferentes. Por ejemplo, también permite ejecutar una visualización previa de SoMachine en un entorno de SoMachine posterior.

Para obtener más información, consulte la descripción de la configuración de símbolos. Se creará un nuevo archivo de símbolos en una descarga o cambio en línea del proyecto. Se descarga en el controlador junto a la aplicación.

Sintaxis

{attribute 'namespace' := '<espacio de nombres>'}

Ejemplo de un reemplazo de espacios de nombres para las variables de un programa

```
{attribute 'namespace':='prog'}
PROGRAM PLC_PRG
VAR
  {attribute 'symbol' := 'readwrite'}
  iVar:INT;
  bVar:BOOL;
END_VAR
```

Si antes se accedía a `iVar`, por ejemplo, mediante `App1.PLC_PRG.ivar`, ahora se accede a través de `prog.ivar`.

Más ejemplos de reemplazo

Espacios de nombres originales	Variable	Reemplazo de espacios de nombres	Acceso a la variable dentro del proyecto actual
App1.Lib2.GVL2	Var07	{attribute 'namespace':=''}	.Var07
App1.GVL2	Var02	{attribute 'namespace':='Ext''}	Ext.Var02
App1.GVL2.FB1	Var02	{attribute 'namespace':='App1.GVL2' '}	App1.GVL2.Var02

Los reemplazos que aparecen en la tabla se convierten en las siguientes entradas del archivo de símbolos:

```
<NodeList>
 <Node name="">
     <Node name="Var07" type="T_INT" access="ReadWrite">
 </Node>
</NodeList>
<NodeList>
 <Node name="Ext">
  <Node name="Var02 " type="T_INT" access="ReadWrite"></Node>
 </Node>
</NodeList>
<NodeList>
 <Node name="App1">
  <Node name="GVL2">
   <Node name="Var02 " type="T_INT" access="ReadWrite"></Node>
     </Node>
 </Node>
</NodeList>
```

Attribute no_check

Descripción general

pragma `{attribute no_check}` se ha añadido a un POU para suprimir la llamada de cualquier POU para comprobaciones implícitas. Como las funciones de comprobación pueden influir en el rendimiento, aplique este atributo a los POU que se llama frecuentemente o a los que ya se han aprobado.

Sintaxis

{attribute ’no_check’}

Attribute no_copy

Descripción general

Por lo general, un cambio en línea requerirá una reasignación de instancias, por ejemplo de los POU. Se copiará el valor de las variables dentro de esta instancia.

Sin embargo, si el pragma `{attribute no_copy}` se añade a una variable, no se ejecutará la copia de un cambio en línea de esta variable, sino que se inicializará esta variable. Esto puede ser razonable si se trata de una variable de puntero local, que señala a una variable desplazada, en realidad, por un cambio en línea (y, por consiguiente, con la dirección modificada).

Sintaxis

{attribute ’no_copy’}

Attribute no-exit

Descripción general

Si un bloque de funciones proporciona un método exit *(véase página 606)*, puede suprimir su llamada para una instancia especial con la ayuda de asignar el pragma `{attribute no-exit}` a la instancia del bloque de funciones.

Sintaxis

{attribute 'symbol'= 'no-exit'}

Ejemplo

Suponga que el parámetro `FB_Exit` del método `exit` se añade a un POU con nombre del bloque de funciones:

En el programa principal `PLC_PRG`, se instancian dos variables de tipo POU:

```
PROGRAM PLC_PRG
VAR
POU1 : POU;
{attribute 'symbol' := 'no-exit'}
POU2 : POU;
END_VAR
```

Cuando la variable `bInCopyCode` toma el valor TRUE en `POU1`, se invoca el método de salida `FB_Exit` al salir de una instancia que se copiará después (cambio en línea), aunque el valor de la variable `bInCopyCode` no influirá en `POU2`.

Attribute noinit

Descripción general

Las variables proporcionadas con pragma `{attribute no_init}` no se inicializarán de forma implícita. pragma pertenece a la variable declarada posteriormente.

Sintaxis

```
{attribute 'no_init'}
```

también es posible

```
{attribute 'no-init'}
{attribute 'noinit'}
```

Ejemplo

```
PROGRAM PLC_PRG
VAR
A : INT;
{attribute 'no_init'}
B : INT;
END_VAR
```

Si se lleva a cabo un reseteo en la aplicación asociada, la variable de entero `A` se volverá a inicializar de forma implícita con 0, mientras que la variable `B` mantendrá el valor que tenga asignado actualmente.

Attribute no_virtual_actions

Descripción general

Este atributo es válido para bloques de funciones derivados de un bloque de funciones base implementado en SFC y que utilizan el flujo de trabajo del SFC principal de la base. Las acciones llamadas al mismo muestran el mismo comportamiento virtual que los métodos. Esto significa que las acciones de base se pueden reemplazar por implementaciones específicas relacionadas con las clases derivadas.

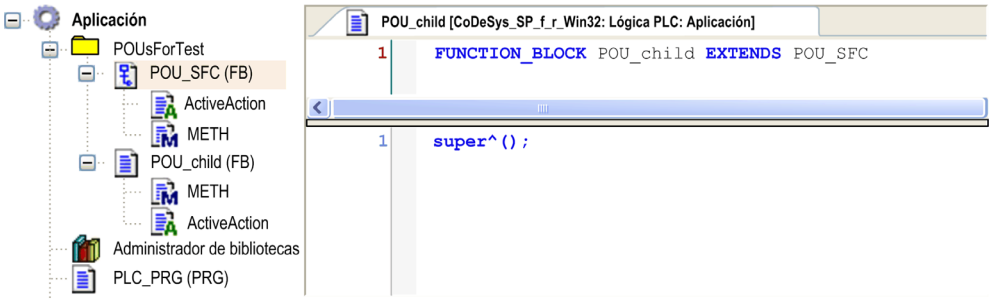
Para evitar que se reemplace la acción de la base, puede asignar el pragma `{attribute 'no_virtual_actions'}` a la base.

Sintaxis

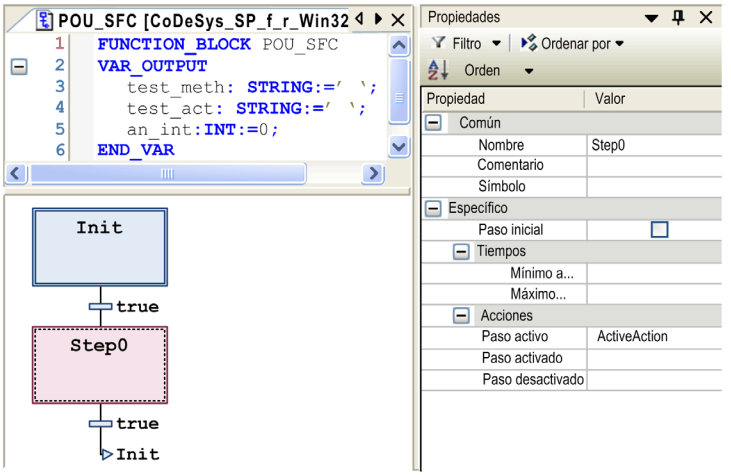
{attribute 'no_virtual_actions'}

Ejemplo

En el siguiente ejemplo, el bloque de funciones `POU_SFC` provoca la ampliación de la base por el bloque de funciones `POU_child`.



Utilizando la palabra clave `SUPER`, la clase derivada `POU_child` llama al flujo de trabajo de la base implementada en el SFC.



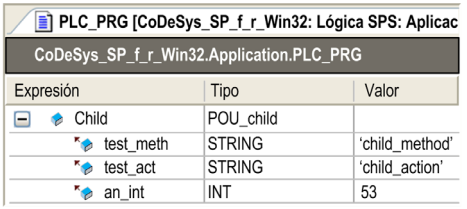
El ejemplo de la implementación de este flujo de trabajo está restringido al paso inicial. Esto es seguido por un paso único con el paso de acción asociado `ActiveAction`, que se ocupa de asignar las variables de salida:

```
an_int:=an_int+1;           // counting the action calls
test_act:='father_action'; // writing string variable test_act
METH();                    // Calling method METH for writing string var
iable test_meth
```

En el caso de la clase derivada `POU_child`, el paso de acción se reemplazará por una implementación específica de `ActiveAction`. Difiere del original al asignar la cadena `'child_action'` en lugar de `'father_action'` a la variable `test_act`.

Asimismo, el método `METH`, que asigna la cadena `'father_method'` a la variable `test_meth` dentro de la base, se reemplazará de manera que `test_meth` se asignará en lugar de `'child_method'`.

El programa principal `PLC_PRG` ejecutará de manera reiterada llamadas a `Child` (un ejemplo de `POU_child`). Según lo esperado, el valor actual de la cadena de salida informa acerca de la llamada a la acción y del método de la clase derivada:


PLC_PRG [CoDeSys_SP_f_r_Win32: Lógica SPS: Aplicac

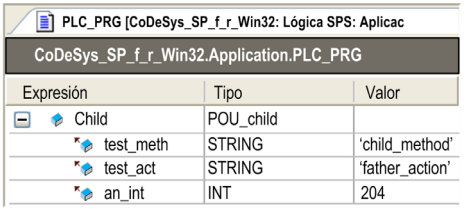
CoDeSys_SP_f_r_Win32.Application.PLC_PRG

Expresión	Tipo	Valor
Child	POU_child	
test_meth	STRING	'child_method'
test_act	STRING	'child_action'
an_int	INT	53

Puede observar un comportamiento diferente si la base es precedida por el atributo `'no_virtual_actions'`.

```
{attribute 'no_virtual_actions'}
FUNCTION_BLOCK POU_SFC...
```

Si bien el método `METH` se reemplazará por su implementación dentro de la clase derivada, una llamada del paso de acción resultará en una llamada a la acción `ActiveAction` de la base. Por lo tanto, `test_act` se asignará a la cadena `'father_action'`.


PLC_PRG [CoDeSys_SP_f_r_Win32: Lógica SPS: Aplicac

CoDeSys_SP_f_r_Win32.Application.PLC_PRG

Expresión	Tipo	Valor
Child	POU_child	
test_meth	STRING	'child_method'
test_act	STRING	'father_action'
an_int	INT	204

Attribute obsolete

Descripción general

Puede añadir un obsolete pragma a una definición de tipo datos para provocar una alerta definida por el usuario durante una compilación si se usa el tipo de datos correspondiente (estructura, bloque de funciones, etc.) dentro del proyecto. Por consiguiente, puede anunciar que el tipo de datos ya no se usa.

A diferencia de un mensaje pragma *(véase página 610)* usado localmente, esta alerta se define dentro de la definición y, por consiguiente, de manera global en todas las instancias del tipo de datos.

Esta instrucción pragma es válida en la línea actual o (si se encuentra en una línea separada) en la línea siguiente.

Sintaxis

{attribute 'obsolete' := 'user-defined text'}

Ejemplo

El pragma obsoleto se introduce en la definición del bloque de funciones `fb1`:

```
{attribute 'obsolete' := 'datatype fb1 not valid!'}
FUNCTION_BLOCK fb1
VAR_INPUT
i:INT;
END_VAR
...
```

Si se utiliza `fb1` como tipo de datos en una declaración, por ejemplo `fbinst: fb1;`, se visualizará la siguiente alerta cuando se genere el proyecto:

```
'datatype fb1 not valid'
```

Attribute pack_mode

Descripción general

pragma `{attribute 'pack_mode'}` define la modalidad en que se agrupa una estructura de datos mientras se asigna. Defina el atributo en la parte superior de una estructura de datos. Influye en la agrupación de toda la estructura.

Sintaxis

{attribute 'pack_mode' := '<valor>'}

Sustituya la plantilla <valor> incluida en las comillas simples por uno de los valores siguientes disponibles:

Valor	Modalidad de agrupación asignada
0	alineada (no habrá zonas de memoria vacías)
1	alineada de 1 byte (idéntico a alineado)
2	alineada de 2 bytes (el tamaño máximo de una zona de memoria vacía es 1 byte)
4	alineada de 4 bytes (el tamaño máximo de una zona de memoria vacía es 3 bytes)
8	alineada de 8 bytes (el tamaño máximo de una zona de memoria vacía es 7 bytes)

Ejemplo

```
{attribute 'pack_mode' := '1'}
TYPE myStruct:
STRUCT
 Enable: BOOL;
 Counter: INT;
 MaxSize: BOOL;
 MaxSizeReached: BOOL;
END_STRUCT
END_TYPE
```

Una variable del tipo de datos `myStruct` creará instancia alineada.

Si la dirección de su componente `Enable` es `0x0100`, el componente `Counter` seguirá una dirección `0x0101`, `MaxSize` en `0x0103` y `MaxSizeReached` en `0x0104`.

Con `pack_mode=2`, `Counter` se encontrará en `0x0102`, `MaxSize` en `0x0104` y `MaxSizeReached` en `0x0105`.

NOTA: Asimismo, puede aplicar el atributo a los POU. Use esta aplicación con precaución debido a los punteros internos finales existentes del POU.

Attribute qualified_only

Descripción general

Cuando el pragma `{attribute 'qualified_only'}` se asigna en la parte superior de una lista de variables globales, a las variables de esta lista solo se puede acceder usando el nombre de la variable global, por ejemplo `gvl.g_var`. Esto vale incluso para variables de tipo de enumeración. Puede ser útil para evitar discrepancias de nombres con las variables locales.

Sintaxis

{attribute 'qualified_only'}

Ejemplo

Suponga que la siguiente lista de variables globales (GVL) se proporciona con el atributo `'qualified_only'`:

```
{attribute 'qualified_only'}
VAR_GLOBAL
iVar:INT;
END_VAR
```

En el POU `PLC_PRG`, a la variable global se la tiene que invocar con el prefijo `GVL`, según se muestra en este ejemplo:

```
GVL.iVar:=5;
```

La siguiente llamada incompleta de la variable se detectará como un error:

```
iVar:=5;
```

Attribute reflection

Descripción general

pragma `{attribute 'reflection'}` está vinculado a firmas. Por motivos de rendimiento, es un atributo obligatorio para los POU con el atributo instance-path *(véase página 634)*.

Sintaxis

{attribute 'reflection'}

Ejemplo

Consulte el ejemplo *(véase página 634)* de `attribute instance-path`.

Attribute subsequent

Descripción general

El pragma `{attribute 'subsequent'}` obliga a que las variables se asignen en una fila en una ubicación de la memoria. Si la lista se modifica, toda la lista se asignará a una nueva ubicación. Este pragma se emplea en programas y listas de variables globales (GVL).

Sintaxis

{attribute 'subsequent'}

NOTA: Si alguna variable de la lista es `RETAIN`, toda la lista se colocará en la memoria retentiva.

NOTA: `VAR_TEMP` en un programa con el atributo `subsequent` se detectará como error de compilación.

Attribute symbol

Descripción general

El pragma `{attribute 'symbol'}` define qué variables se van a gestionar en la configuración de símbolos.

Las siguientes operaciones de importación se aplican a las variables:
- Las variables se exportan como símbolos dentro de una lista de símbolos.
- Las variables se exportan como un archivo XML en el directorio de proyectos.
- Las variables se exportan a un archivo no visible y disponible en el sistema de destino para acceso externo, mediante un servidor OPC, por ejemplo.

Las variables con ese atributo serán descargadas al controlador incluso si no se han configurado o no son visibles en el editor de configuración de símbolos.

NOTA: La configuración de símbolos debe estar disponible como un objeto bajo la aplicación correspondiente en el **árbol de herramientas**.

Sintaxis

{attribute 'symbol' := 'none' | 'read' | 'write' | 'readwrite'}

Solo se permite el acceso a los símbolos que provengan de programas o de la lista de variables globales. Para acceder a un símbolo especifique su nombre completo.

Puede asignar la definición de pragma a una variable específica o a un conjunto de todas las variables declaradas en un programa.
- Para que esto sea válido para una única variable coloque el pragma en la línea anterior a la declaración de variable.
- Para que sea válido para todas las variables contenidas en la parte de declaraciones de un programa, coloque el pragma en la primera línea del editor de declaraciones. En este caso, también puede modificar la configuración para variables específicas añadiendo explícitamente un pragma.

La posibilidad de acceso a un símbolo está definida por los siguientes parámetros de pragma:
- `'none'`
- `'read'`
- `'write'`
- `'readwrite'`

Si no hay ningún parámetro predefinido se validará el parámetro por defecto `'readwrite'`.

Ejemplo

Con la siguiente configuración, las variables `A` y `B` se exportarán con acceso de lectura y escritura. La variable `D` se exportará con acceso de lectura.

```
{attribute 'symbol' := 'readwrite'}
PROGRAM PLC_PRG
VAR
A : INT;
B : INT;
{attribute 'symbol' := 'none'}
C : INT;
{attribute 'symbol' := 'read'}
D : INT;
END_VAR
```

Attribute warning disable

Descripción general

Puede utilizar el pragma `warning disable` para suprimir alertas. Para habilitar la visualización de la alerta, use el pragma `warning restore`.

Sintaxis

{warning disable <ID del compilador>}

Todas las alertas y los errores detectados por el compilador poseen un ID único que se muestra al principio de la descripción.

Ejemplo de mensajes del compilador

```
------ Build started: Application: Device.Application ------
typify code ...
C0196: Implicit conversion from unsigned Type 'UINT' to signed Type 'IN
T' : possible change of sign
Compile complete -- 0 errors
```

Ejemplo

```
VAR
    {warning disable C0195}
    test1 : UINT := -1;
    {warning restore C0195}
    test2 : UINT := -1;
END_VAR
```

En este ejemplo, se detectará una alerta para `test2`. Pero no se detectará ninguna alerta para `test1`.

Sección 27.6
La funcionalidad Intelli-sense

Intelli-sense

Descripción general

Siempre que los identificadores (por ejemplo, las variables o instancias de bloques de funciones) se puedan introducir (por ejemplo en el interior de los editores de lenguajes de IEC 61131-3 o dentro de las ventanas de **Supervisar, Traza, Visualización**), será compatible con la funcionalidad de intelli-sense. Puede personalizar esta función (activada o desactivada) en la sección **Codificación inteligente** del cuadro de diálogo **Herramientas** →**Opciones**.

Soporte en inserción del identificador

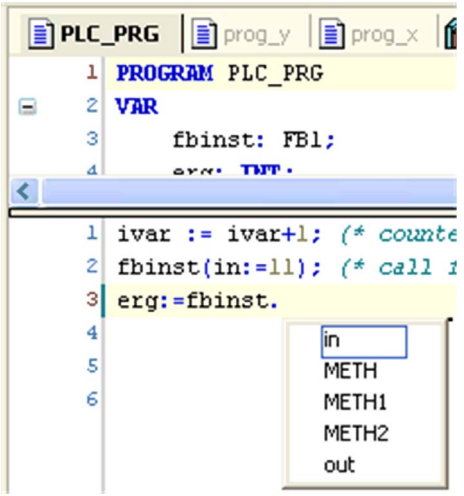
La funcionalidad intelli-sense permite insertar un identificador adecuado:

- Si introduce un punto (.) en cualquier posición en la que se permita insertar un identificador global en vez del identificador, se mostrará un cuadro de selección. Enumera las variables globales actualmente disponibles. Puede elegir uno de estos elementos y pulsar la tecla RETORNO para introducirlo detrás del punto. También puede insertar el elemento haciendo doble clic en la entrada de la lista.
- Si introduce una instancia del bloque de funciones o una variable de estructura seguida de un punto (.), aparecerá un cuadro de selección. Enumera las variables de entrada y salida del bloque de funciones correspondiente o los componentes de estructura. Puede elegir el elemento que desee pulsando la tecla RETORNO o haciendo doble clic en la entrada de la lista para insertarla.
- En el editor ST, si introduce cualquier cadena y pulsa CTRL+ESPACIO, aparecerá un cuadro de selección. Enumera los POU y las variables globales disponibles en el proyecto. Se seleccionará la primera entrada de la lista, que comienza con la cadena proporcionada. Pulse la tecla RETORNO para insertarla en el programa.

Ejemplos

La funcionalidad intelli-sense ofrece componentes de estructura:

La funcionalidad intelli-sense ofrece componentes de un bloque de funciones:



Capítulo 28
Tipos de datos

Contenido de este capítulo

Este capítulo contiene las siguientes secciones:

Sección 28.1
Información general

Tipos de datos

Descripción general

Puede utilizar tipos de datos estándar *(véase página 663)*, tipos de datos definidos por el usuario *(véase página 676)* o instancias de bloques de funciones al programar en SoMachine. Cada identificador está asignado a un tipo de dato. Este tipo de dato indica el espacio de la memoria que se reservará y qué tipo de valores almacena.

Sección 28.2
Tipos de datos estándar

Tipos de datos estándar

Descripción general

SoMachine admite todos los tipos de datos *(véase página 662)* descritos en la norma IEC61131-3.

Los siguientes tipos de datos se describen en este capítulo:
- BOOL *(véase página 663)*
- Entero *(véase página 663)*
- REAL / LREAL *(véase página 664)*
- STRING *(véase página 665)*
- Tipos de datos de tiempo *(véase página 665)*

Asimismo, se permiten ciertos tipos de datos de extensión/estándar *(véase página 666)* y el usuario puede definir sus propios tipos de datos definidos por el usuario *(véase página 676)*.

BOOL

Las variables de tipo BOOL pueden tener los valores TRUE (1) y FALSE (0). Se reservan 8 bits de espacio en memoria.

Para obtener más información, consulte el capítulo *constantes BOOL (véase página 799)*.

NOTA: Puede usar las comprobaciones implícitas para validar la conversión de los tipos de variable (consulte el capítulo *POU para comprobaciones implícitas (véase página 195)*).

Entero

En esta tabla se enumeran los tipos de datos enteros disponibles. Cada uno de estos tipos abarca un rango distinto de valores. Se aplican las limitaciones de rango siguientes.

Tipo de datos	Límite inferior	Límite superior	Espacio en memoria
BYTE	0	255	8 bits
WORD	0	65,535	16 bits
DWORD	0	4,294,967,295	32 bits
LWORD	0	2^{64}-1	64 bits
SINT	–128	127	8 bits
USINT	0	255	8 bits
INT	–32,768	32,767	16 bits
UINT	0	65,535	16 bits

Tipo de datos	Límite inferior	Límite superior	Espacio en memoria
DINT	–2,147,483,648	2,147,483,647	32 bits
UDINT	0	4,294,967,295	32 bits
LINT	-2^{63}	$2^{63}-1$	64 bits
ULINT	0	$2^{64}-1$	64 bits

NOTA: Las conversiones desde tipos más grandes hasta tipos más pequeños pueden provocar una pérdida de información.

Para obtener más información, consulte la descripción de constantes de números *(véase página 805)*.

NOTA: Puede usar las comprobaciones implícitas para validar la conversión de los tipos de variable (consulte el capítulo *POU para comprobaciones implícitas (véase página 195)*).

REAL / LREAL

Los tipos de datos REAL y LREAL se denominan tipos de coma flotante. Representan números racionales. Se reservan 32 bits de espacio en memoria para REAL y 64 bits para LREAL.

Rango de valores para REAL:

1.401e-45...3.403e+38

Rango de valores para LREAL:

2.2250738585072014e-308...1.7976931348623158e+308

NOTA: La compatibilidad del tipo de datos LREAL depende del dispositivo de destino. Consulte la documentación correspondiente para ver si el tipo de 64 bits LREAL se convierte a REAL durante la compilación (posiblemente con una pérdida de información) o se mantiene.

NOTA: Si un REAL o LREAL se convierte en SINT, USINT, INT, UINT, DINT, UDINT, LINT o ULINT y el valor del número real está fuera del rango de valores de ese entero, el resultado no estará indefinido y dependerá del sistema de destino. En este caso, incluso una excepción sería posible. Para obtener un código de destino/independiente, gestione cualquier exceso del rango mediante la aplicación. Si el número REAL/LREAL está dentro del rango de valores de entero, la conversión funcionará del mismo modo en todos los sistemas.

Al asignar `i1 := r1;`, se ha detectado un error. Por consiguiente, la nota anterior se aplica cuando se utilizan operadores de conversión *(véase página 749)* como el siguiente:

```
i1 := REAL_TO_INT(r1);
```

Para obtener más información, consulte Constantes REAL/LREAL (operandos) *(véase página 806)*.

NOTA: Puede usar las comprobaciones implícitas para validar la conversión de los tipos de variable (consulte el capítulo *POU para comprobaciones implícitas (véase página 195)*).

STRING

Una variable de tipo de datos STRING puede contener cualquier cadena de caracteres. La entrada del tamaño en la declaración determina el espacio en memoria que se reserva para la variable. Hace referencia al número de caracteres de la cadena y se puede poner entre paréntesis o corchetes. Si no se especifica nada con respecto al tamaño, se usará el tamaño predefinido de 80 caracteres.

En general, la longitud de una cadena no está limitada. Sin embargo, las funciones de la cadena solo pueden procesar cadenas con una longitud de entre 1 y 255 caracteres. Si se inicializa una variable con una cadena demasiado larga para el tipo de datos de la variable, la cadena se acortará como corresponda de derecha a izquierda.

NOTA: El espacio en memoria necesario para una variable de tipo STRING es de 1 byte por carácter más 1 byte adicional. Es decir, la declaración `"STRING[80]"` requiere 81 bytes.

Ejemplo de una declaración de cadena con 35 caracteres:

```
str:STRING(35):='This is a String';
```

Para obtener más información, consulte WSTRING *(véase página 669)* y Constantes de STRING (operandos) *(véase página 807)*.

NOTA: Puede usar las comprobaciones implícitas para validar la conversión de los tipos de variable (consulte el capítulo *POU para comprobaciones implícitas* *(véase página 195)*).

Tipos de datos de tiempo

Los tipos de datos TIME, TIME_OF_DAY (abreviado TOD), DATE y DATE_AND_TIME (abreviado DT) se gestionan internamente como DWORD. El tiempo se indica en milisegundos en TIME y TOD. El tiempo en TOD comienza a las 12 a. m. El tiempo se indica en segundos en DATE y DT desde el 1 de enero de 1970 a las 12 a. m.

Para obtener más información, consulte las siguientes descripciones:

- Tipos de datos *(véase página 662)*
- LTIME *(véase página 668)*: extensión de la norma IEC 61131-3, disponible como un tipo de datos de tiempo de 64 bits.
- Constantes TIME *(véase página 800)*
- Constantes DATE *(véase página 802)*
- Constantes DATE_AND_TIME *(véase página 803)*
- Constantes TIME_OF_DAY *(véase página 804)*

NOTA: Puede usar las comprobaciones implícitas para validar la conversión de los tipos de variable (consulte el capítulo *POU para comprobaciones implícitas* *(véase página 195)*).

Sección 28.3
Ampliaciones a IEC estándar

Descripción general

En este capítulo se enumeran los tipos de datos compatibles con SoMachine además de la norma IEC 61131-3.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

UNION

Descripción general

Como ampliación a la norma IEC 61131-3 , puede declarar uniones en los tipos definidos por el usuario.

Los componentes de una unión tienen el mismo offset. Esto significa que ocupan la misma ubicación de almacenamiento. Por tanto, si partimos de la definición de unión que se muestra en el ejemplo siguiente, una asignación a `name.a` también manipula `name.b`.

Ejemplo

```
TYPE name: UNION
a : LREAL;
b : LINT;
END_UNION
END_TYPE
```

LTIME

Descripción general

Como ampliación a IEC 61131-3, LTIME es compatible como base de tiempo para temporizadores de alta resolución. LTIME tiene un tamaño de 64 bits y una resolución en nanosegundos.

Sintaxis

LTIME#<declaración de tiempo>

La declaración de tiempo puede incluir las unidades de tiempo como se utilizan con la constante TIME y como:

- us : microsegundos
- ns : nanosegundos

Ejemplo

```
LTIME1 := LTIME#1000d15h23m12s34ms2us44ns
```

Comparar con TIME de 32 bits y resolución en milisegundos *(véase página 665)*.

WSTRING

Descripción general

Este tipo de datos de cadena es una ampliación a la norma IEC 61131-3.

Difiere del tipo de STRING estándar (ASCII) en la interpretación del formato Unicode y la necesidad de 2 bytes para cada carácter y 2 bytes de espacio adicional de la memoria (cada uno solo 1 en caso de una STRING).

Ejemplo

```
wstr:WSTRING:="This is a WString";
```

Para obtener más información, consulte las siguientes descripciones:

- STRING *(véase página 665)*
- ConstantesSTRING *(véase página 807)* (operandos)

BIT

Descripción general

Puede utilizar el tipo de datos BIT solo para variables particulares dentro de Estructuras *(véase página 681)*. Los valores posibles son TRUE (1) y FALSE (0).

Un elemento BIT consume 1 bit de espacio en memoria y sirve para direccionar bits únicos de una estructura por nombre (para obtener más información, consulte el párrafo *Acceso a bits en estructuras (véase página 682)*). Los elementos de bits que hayan declarado uno detrás de otro se combinarán en bytes. A diferencia de los tipos BOOL *(véase página 663)*, cuando se reservan 8 bits, el uso de espacio en memoria se puede optimizar. En cambio, el acceso a los bits llevará indudablemente más tiempo. Por ello, use el tipo de datos BIT si desea almacenar diversos tipos de información booleana en un formato compacto.

Referencias

Descripción general

Este tipo de datos está disponible en la ampliación a la norma IEC 61131-3.

Una referencia funciona como un alias para un objeto. El alias se puede escribir o leer a través de identificadores. La diferencia con un puntero es que el valor al que apunta se ve directamente afectado y que la asignación de la referencia y el valor es fija. Defina la dirección de la referencia a través de una operación de asignación independiente. Puede utilizar el operador `__ISVALIDREF` para comprobar si una referencia apunta a un valor válido (que no es igual a 0). Para obtener más información, consulte el párrafo *Búsqueda de referencias válidas* más adelante en este capítulo.

Sintaxis

<identificador> : REFERENCE TO <tipo de datos>

Declaración de ejemplo

```
ref_int : REFERENCE TO INT;
a : INT;
b : INT;
```

`ref_int` ya se encuentra disponible para usarse como alias en variables de tipo INT.

Ejemplo de uso

```
ref_int REF= a;   (* ref_int now points to a *)
ref_int := 12;    (* a now has value 12 *)
b := ref_int * 2; (* b now has value 24 *)
ref_int REF= b;   (* ref_int now points to b *)
ref_int := a / 2; (* b now has value 6 *)
ref_int REF= 0;    (* explicit initialization of the reference *)
```

NOTA: No es posible declarar referencias como REFERENCE TO REFERENCE, ARRAY OF REFERENCE o POINTER TO REFERENCE.

Búsqueda de referencias válidas

Puede utilizar el operador __ISVALIDREF para comprobar si una referencia apunta a un valor válido que es un valor distinto a 0.

Sintaxis

```
<boolean variable> := __ISVALIDREF(identifier, declared with type <REFE
RENCE TO <datatype>);
<boolean variable> will be TRUE, if the reference points to a valid val
ue, FALSE if not.
```

Ejemplo

Declaración

```
ivar : INT;
ref_int : REFERENCE TO INT;
ref_int0: REFERENCE TO INT;
testref: BOOL := FALSE;
```

Implementación

```
ivar := ivar +1;
ref_int REF= hugo;
ref_int0 REF= 0;
testref := __ISVALIDREF(ref_int);   (* will be TRUE, because ref_int po
ints to ivar, which is unequal 0 *)
testref0 := __ISVALIDREF(ref_int0); (* will be FALSE, because ref_int0
is set to 0 *)
```

Punteros

Descripción general

Como extensión a la norma IEC 61131-3, puede utilizar punteros.

Los punteros almacenan las direcciones de variables, programas, bloques de funciones, métodos y funciones mientras se ejecuta un programa de aplicación. Un puntero puede apuntar a cualquiera de esos objetos y a cualquier tipo de datos *(véase página 662)*, incluso a los tipos de datos definidos por el usuario *(véase página 677)*. La posibilidad de utilizar una función de control del puntero implícito se describe más adelante en el párrafo *Función de CheckPointer (véase página 675)*.

Sintaxis de una declaración de puntero

<identificador>: POINTER TO <tipo de datos | bloque de funciones | programa | método | función>;

La desreferenciación de un puntero equivale a obtener el valor almacenado actualmente en la dirección que indica. Puede desreferenciar un puntero añadiendo el operador de contenido ^ (caret ASCII o símbolo circunflejo) *(véase página 746)* después del identificador de puntero. Observe `pt^` en el ejemplo siguiente.

Puede utilizar el `ADR` address operator *(véase página 745)* para asignar la dirección de una variable a un puntero.

Ejemplo

```
VAR
  pt:POINTER TO INT;  (* of pointer pt *)
var_int1:INT := 5;  (* declaration of variables var_int1 and var_int2 *
)
  var_int2:INT;
END_VAR
pt := ADR(var_int1); (* address of var_int1 is assigned to pointer pt *
)
var_int2:= pt^;      (* value 5 of var_int1 gets assigned to var_int2 v
ia dereferencing of pointer pt; *)
```

Función Punteros

SoMachine también admite punteros de función. Estos punteros se pueden pasar a bibliotecas externas, pero no se puede llamar a un puntero de función dentro de una aplicación del sistema de programación. La función de tiempo de ejecución del registro de funciones de devolución de llamada (función de la biblioteca del sistema) espera el puntero de la función y, en función de la devolución de llamada para la que se haya solicitado el registro, el sistema en tiempo de ejecución llamará de forma implícita a la función correspondiente (por ejemplo, en STOP). Para habilitar esa llamada del sistema (sistema en tiempo de ejecución), defina las propiedades correspondientes (de forma predeterminada en **Ver** →**Propiedades...** →**Crear**) del objeto de la función.

Puede utilizar el operador *(véase página 745)* `ADR` en los nombres de funciones, nombres de programas, nombres de bloques de funciones y nombres de métodos. Como las funciones pueden moverse después del cambio en línea, el resultado no será la dirección de la función, sino la dirección de un puntero en la función. Esta dirección será válida siempre que la función exista en el destino.

La ejecución del comando **Cambio online** puede cambiar el contenido de las direcciones.

⚠ ATENCIÓN

PUNTERO NO VÁLIDO

Cuando utilice punteros en las direcciones y ejecute el comando **Cambio online**, verifique siempre el contenido de los punteros.

El incumplimiento de estas instrucciones puede causar lesiones o daño al equipo.

Acceso por índice a punteros

Como extensión a la norma IEC 61131-3, se permite el acceso por índice `[]` a variables de tipo POINTER, STRING *(véase página 665)* y WSTRING *(véase página 669)*.

- `pint[i]` devuelve el tipo de datos base.
- El acceso por índice en los punteros es aritmético:
  Si el acceso por índice se emplea en una variable de tipo puntero, el offset `pint[i]` es igual a `(pint + i * SIZEOF(base type))^`. El acceso por índice también ejecuta una desreferenciación implícita en el puntero. El tipo de resultado es el tipo de base puntero. Tenga en cuenta que `pint[7] does not equate to (pint + 7)^`.
- Si el acceso por índice se emplea en una variable de tipo STRING, el resultado es el carácter en offset `index-expr`. El resultado es de tipo BYTE. `str[i]` devolverá el carácter i-th de la cadena como SINT (ASCII).
- Si el acceso por índice se emplea en una variable de tipo WSTRING, el resultado es el carácter en offset `index-expr`. El resultado es de tipo WORD. `wstr[i]` devolverá el carácter i-th de la cadena como INT (Unicode).

NOTA: También puede utilizar Referencias *(véase página 671)*. A diferencia de un puntero, las referencias afectan directamente a un valor.

Función CheckPointer

Para comprobar el acceso al puntero durante el tiempo de ejecución, puede utilizar la función de comprobación disponible de forma implícita `CheckPointer`. Se llama antes de cada acceso a la dirección de un puntero. Para conseguirlo, añada el objeto **POU para comprobaciones implícitas** a la aplicación. Para ello, marque la casilla de verificación relacionada con el tipo **CheckPointer**, seleccione un lenguaje de implementación y confirme su configuración haciendo clic en **Abrir**. De este modo se abre la función de comprobación en el editor correspondiente al lenguaje de implementación seleccionado. Independientemente de esta elección, la parte de declaraciones está preestablecida. No puede modificarla, salvo que añada más variables locales. Sin embargo, a diferencia de otras funciones de comprobación, no hay una implementación predeterminada de `CheckPointer` disponible.

NOTA: No existe una llamada implícita de la función de comprobación para el puntero `THIS`.

Plantilla:

Parte de la declaración:

```
// Implicitly generated code : DO NOT EDIT
FUNCTION CheckPointer : POINTER TO BYTE
VAR_INPUT
ptToTest : POINTER TO BYTE;
iSize : DINT;
iGran : DINT;
bWrite: BOOL;
END_VAR
```

Parte de la implementación (incompleta):

```
// No standard way of implementation. Fill your own code here
CheckPointer := ptToTest;
```

Cuando se llama, se incluyen los siguientes parámetros de entrada en la función:

- `ptToTest`: Dirección de destino del puntero.
- `iSize`: Tamaño de la variable referenciada; el tipo de datos de `iSize` debe ser compatible con enteros y abarcar el tamaño máximo de datos posibles almacenados en la dirección del puntero.
- `iGran`: Granularidad del acceso que es el tipo de datos no estructurados más grande usado en la variable referenciada; el tipo de datos de `iGran` ha de ser compatible con enteros.
- `bWrite`: Tipo de acceso (TRUE= acceso de escritura, FALSE= acceso de lectura); el tipo de datos de `bWrite` debe ser BOOL.

Si se obtiene un resultado positivo de la comprobación, se devolverá el puntero de entrada no modificado (`ptToTest`).

Sección 28.4
Tipos de datos definidos por el usuario

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Tipos de datos definidos

Descripción general

Además, en los tipos de datos estándar, puede definir tipos de datos especiales dentro de un proyecto.

Puede definirlos creando objetos de tipo de unidades de datos (DUT) en el **Árbol de POU** o el **Árbol de dispositivos** o dentro de una parte de la declaración de un POU.

Consulte las recomendaciones sobre los nombres de identificadores *(véase página 578)* para que sean lo más exclusivas posible.

Consulte los siguientes tipos de datos definidos por el usuario:

- matrices *(véase página 678)*
- estructuras *(véase página 681)*
- enumeraciones *(véase página 683)*
- tipos de subárea *(véase página 685)*
- referencias *(véase página 671)*
- punteros *(véase página 673)*

Matrices

Descripción general

Los campos de una, dos y tres dimensiones (matrices) se admiten como tipos de datos elementales. Puede definir matrices tanto en la parte de declaraciones de un POU como en las listas de variables globales. También puede utilizar comprobaciones de límites implícitas *(véase página 679)*.

Sintaxis

<Nombre_de_matriz>:ARRAY [<ll1>..<ul1>,<ll2>..<ul2>,<ll3>..<ul3>] OF <tipo elem.>

`ll1, ll2, ll3` identifican el límite más bajo del rango de campo.

`ul1, ul2` y `ul3` identifican el límite superior del rango de campo.

Los valores del rango deben ser de tipo entero.

Ejemplo

```
Card_game: ARRAY [1..13, 1..4] OF INT;
```

Inicializando matrices

Ejemplo para la inicialización completa de una matriz.

```
arr1 : ARRAY [1..5] OF INT := [1,2,3,4,5];
arr2 : ARRAY [1..2,3..4] OF INT := [1,3(7)]; (* short for 1,7,7,7 *)
arr3 : ARRAY [1..2,2..3,3..4] OF INT := [2(0),4(4),2,3];
      (* short for 0,0,4,4,4,4,2,3 *)
```

Ejemplo de la inicialización de la matriz de una estructura.

Definición de la estructura

```
TYPE STRUCT1
STRUCT
 p1:int;
 p2:int;
 p3:dword;
END_STRUCT
END_TYPE
```

Inicialización de la matriz

```
ARRAY[1..3] OF STRUCT1:= [(p1:=1,p2:=10,p3:=4723),(p1:=2,p2:=0,p3:=299)
,(p1:=14,p2:=5,p3:=112)];
```

Ejemplo de la inicialización parcial de una matriz.

```
arr1 : ARRAY [1..10] OF INT := [1,2];
```

Los elementos en los que no hay ningún valor preasignado se inicializan con el valor inicial predeterminado del tipo básico. En consecuencia, los elementos `arr1[6]...arr1[10]` del ejemplo anterior son inicializados con 0.

Acceso a los componentes de la matriz

En una matriz de dos dimensiones, acceda a los componentes de la siguiente manera:

<Nombre_de_matriz>[Index1,Index2]

Ejemplo:

```
Card_game [9,2]
```

Funciones de verificación para límites de matriz

Para acceder a un elemento de la matriz de una manera apropiada durante la ejecución, la función `CheckBounds` debe estar disponible para la aplicación. Por lo tanto, añada el objeto **POU para comprobaciones implícitas** a la aplicación utilizando **Agregar objeto →POU para comprobaciones implícitas**. Seleccione la casilla de verificación relacionada con el tipo **CheckBounds**. Seleccione un lenguaje de implementación. Confirme su configuración con **Abrir**. La función `CheckBound` se abrirá en el editor correspondiente al lenguaje de implementación seleccionado. Independientemente de esta elección, la parte de declaraciones está preestablecida. No puede modificarla, salvo añadiendo más variables locales. El editor ST propondrá una implementación predeterminada de la función que usted puede modificar.

Esta función de verificación debe tratar las violaciones de límite con un método apropiado (por ejemplo, estableciendo un indicador de error detectado o ajustando el índice). La función se llamará implícitamente tan pronto como se asigne una variable de tipo ARRAY.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

No modifique la parte de declaraciones de una función de verificación implícita.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

Ejemplo de uso de `CheckBounds` de función

La implementación predeterminada de la función de verificación es la siguiente:

Parte de declaraciones:

```
// Implicitly generated code : DO NOT EDIT
FUNCTION CheckBounds : DINT
VAR_INPUT
index, lower, upper: DINT;
END_VAR
```

Parte de implementación

```
// Implicitly generated code : Only an Implementation suggestion
IF  index < lower THEN
CheckBounds := lower;
ELSIF  index > upper THEN
CheckBounds := upper;
ELSE
CheckBounds := index;
END_IF
```

Una vez llamada, la función obtiene los siguientes parámetros de entrada:

- `index`: índice de elementos de campo.
- `lower`: el límite inferior del rango de campo.
- `upper`: el límite superior del rango de campo.

Mientras el índice permanezca dentro del rango, el valor de retorno será el propio índice. De lo contrario, en correspondencia con la infracción de rango, el límite superior o inferior del rango de campo será devuelto.

Sobrepasar el límite superior de la matriz `a`

El límite superior de la matriz `a` es sobrepasado en el ejemplo a continuación:

```
PROGRAM PLC_PRG
VAR a: ARRAY[0..7] OF BOOL;
b: INT:=10;
END_VAR
a[b]:=TRUE;
```

En este caso, la llamada implícita a la función `CheckBounds` que precede a la asignación modifica el valor del índice, que pasa de `10` a un límite superior de `7`. Así pues, el valor TRUE se asignará al elemento `a[7]` de la matriz. De esta manera podrá corregir intentos de acceso fuera del rango de campo mediante la función `CheckBounds`.

Estructuras

Descripción general

Cree estructuras en un proyecto como objetos DUT (unidad de tipos de datos) a través del cuadro de diálogo **Agregar objeto**.

Empiezan por las palabras clave `TYPE` y `STRUCT` y terminan con `END_STRUCT` y `END_TYPE`.

Sintaxis

```
TYPE <structurename>:
STRUCT
    <declaration of variables 1>
    ...
    <declaration of variables n>
END_STRUCT
END_TYPE
```

<nombre de estructura> es un tipo que se reconoce mediante el proyecto y se puede utilizar como un tipo de datos estándar.

Se permiten las estructuras anidadas. La única restricción es que puede que las variables no se asignen a direcciones (no se permite la declaración `AT`).

Ejemplo

Ejemplo de una definición de estructura denominada `Polygonline`:

```
TYPE Polygonline:
STRUCT
    Start:ARRAY [1..2] OF INT;
    Point1:ARRAY [1..2] OF INT;
    Point2:ARRAY [1..2] OF INT;
    Point3:ARRAY [1..2] OF INT;
    Point4:ARRAY [1..2] OF INT;
    End:ARRAY [1..2] OF INT;
END_STRUCT
END_TYPE
```

Inicialización de estructuras

Ejemplo:

```
Poly_1:polygonline := ( Start:=[3,3], Point1:=[5,2], Point2:=[7,3], Poi
nt3:=[8,5], Point4:=[5,7], End:= [3,5]);
```

Las inicializaciones con variables no son posibles. Para obtener un ejemplo de la inicialización de una matriz de una estructura, consulte *Matrices* *(véase página 678)*.

Acceso a componentes de estructura

Puede obtener acceso a los componentes de estructura con la sintaxis siguiente:

<nombre de estructura>.<nombre de componente>

Para el ejemplo anterior de la estructura `Polygonline`, puede acceder al componente `Start` mediante `Poly_1.Start`.

Acceso a bits en estructuras

El tipo de datos BIT *(véase página 670)* es un tipo de datos especial que solo se puede definir en estructuras. Consume 1 bit de espacio en memoria y permite direccionar bits únicos de una estructura por nombre.

```
TYPE <structurename>:
STRUCT
    <bitname bit1> : BIT;
    <bitname bit2> : BIT;
    <bitname bit3> : BIT;
    ...
    <bitname bitn> : BIT;
END_STRUCT
END_TYPE
```

Puede obtener acceso al componente de estructura `BIT` con la sintaxis siguiente:

<nombre de estructura>.<nombre de bit>

NOTA: No es posible utilizar referencias y el puntero en variables `BIT`. Además, las variables `BIT` no se permiten en matrices.

Enumeraciones

Descripción general

Una enumeración es un tipo definido por el usuario que consta de una serie de constantes de cadena. Estas constantes se denominan valores de enumeración.

Los valores de enumeración se reconocen globalmente en todas las áreas del proyecto, incluso si se declaran dentro de un POU.

Una enumeración se crea en un proyecto como objeto DUT mediante el cuadro de diálogo **Agregar objeto**.

NOTA: La declaración de enumeración local solo es posible dentro de TYPE.

Sintaxis

TYPE <identificador> (<enum_0>,<enum_1>, ...,<enum_n>) |<tipo de datos base>;END_TYPE

Una variable de tipo <identificador> puede adoptar uno de los valores de enumeración `<enum_..>` y se inicializará con la primera. Estos valores son compatibles con números enteros, lo que significa que puede realizar operaciones con ellos al igual que con las variables enteras. Puede asignar un número x a la variable. Si los valores de enumeración no se inicializan con valores específicos dentro de la declaración, el recuento comenzará por 0. Al inicializar, asegúrese de que los valores iniciales van en aumento dentro de la fila de componentes. La validez del número se verifica cuando se ejecuta.

Ejemplo

```
TYPE TRAFFIC_SIGNAL: (red, yellow, green:=10);  (* The initial value fo
r each of the colors is red 0, yellow 1, green 10 *)
END_TYPE
TRAFFIC_SIGNAL1 : TRAFFIC_SIGNAL;
TRAFFIC_SIGNAL1:=0; (* The value of the traffic signal is "red" *)
FOR i:= red TO green DO
 i := i + 1;
END_FOR;
```

Primera extensión a la norma IEC 61131-3

Puede utilizar la denominación de tipo de las enumeraciones (como operador de ámbito *(véase página 795)*) para evitar ambigüedad en el acceso a una constante de enumeración.

De este modo, es posible utilizar la misma constante en enumeraciones diferentes.

Ejemplo

Definición de dos enumeraciones

```
TYPE COLORS_1: (red, blue);
END_TYPE
TYPE COLORS_2: (green, blue, yellow);
END_TYPE
```

Uso del valor de enumeración azul en un POU

Declaración

```
colorvar1 : COLORS_1;
colorvar2 : COLORS_2;
```

Implementación

```
(* possible: *)
colorvar1 := colors_1.blue;
colorvar2 := colors_2.blue;
(* not possible: *)
colorvar1 := blue;
colorvar2 := blue;
```

Segunda extensión a la norma IEC 61131-3

Puede especificar explícitamente el tipo de datos base de la enumeración, que de forma predeterminada es INT.

Ejemplo

El tipo de dato base de la enumeración `BigEnum` debe ser DINT:

```
TYPE BigEnum : (yellow, blue, green:=16#8000) DINT;
END_TYPE
```

Tipos de subárea

Descripción general

Un tipo de subárea es un tipo definido por el usuario *(véase página 677)* cuyo rango de valores es solo un subconjunto del tipo de rangos básico. También puede utilizar comprobaciones de límites implícitas *(véase página 686)*.

Puede realizar la declaración en un objeto DUT pero también puede declarar la variable directamente con un tipo de subárea.

Sintaxis

Sintaxis para la declaración como un objeto DUT:

TYPE <name>: <Inttype> (<ug>..<og>) END_TYPE;

<name>	un identificador IEC válido
<inttype>	uno de los tipos de datos SINT, USINT, INT, UINT, DINT, UDINT, BYTE, WORD, DWORD (LINT, ULINT, LWORD)
<ug>	una constante compatible con el tipo básico, estableciendo el límite inferior de los tipos de rango El propio límite inferior está incluido en este rango.
<og>	una constante compatible con el tipo básico, estableciendo el límite superior de los tipos de rango El propio límite inferior está incluido en este tipo básico.

Ejemplo

```
TYPE
 SubInt : INT (-4095..4095);
END_TYPE
```

Declaración directa de una variable con un tipo de subárea

```
VAR
 i : INT (-4095..4095);
 ui : UINT (0..10000);
END_VAR
```

Si un valor es asignado a un tipo de subárea (en la declaración o en la implementación) pero no se corresponde con este rango (por ejemplo, `i:=5000` en el ejemplo de declaración mostrado arriba), se mostrará un mensaje.

Funciones de verificación para límites de rango

Para verificar los límites de rango durante el tiempo de ejecución, las funciones `CheckRangeSigned` o `CheckRangeUnsigned` deben estar disponibles para la aplicación. Puede añadir el objeto **POU para comprobaciones implícitas** a la aplicación utilizando el cuadro de diálogo **Agregar objeto**. Marque la casilla de verificación relacionada con **CheckRangeSigned** o **CheckRangeUnsigned**. Seleccione un lenguaje de implementación. Confirme su configuración con **Abrir**. La función seleccionada se abrirá en el editor correspondiente al lenguaje de implementación seleccionado. Independientemente de esta elección, la parte de declaraciones está preestablecida. No puede modificarla, salvo añadiendo más variables locales. El editor ST propondrá una implementación predeterminada de la función que usted puede modificar.

El propósito de esta función de verificación es el tratamiento correcto de las infracciones de la subárea (por ejemplo, estableciendo un indicador de error o modificando el valor). Se llama implícitamente a la función en cuanto se asigna una variable de tipo de subárea.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

No modifique la parte de declaraciones de una función de verificación implícita.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

Ejemplo

La asignación de una variable perteneciente a un tipo de subárea con signo implica una llamada implícita a `CheckRangeSigned`. La implementación por defecto de esa función reduciendo un valor al rango permitido es la siguiente:

Parte de la declaración:

```
// Implicitly generated code : DO NOT EDIT
FUNCTION CheckRangeSigned : DINT
VAR_INPUT
value, lower, upper: DINT;
END_VAR
```

Parte de implementación:

```
// Implicitly generated code : Only an Implementation suggestion
IF (value < lower) THEN
CheckRangeSigned := lower;
ELSIF(value > upper) THEN
CheckRangeSigned := upper;
ELSE
CheckRangeSigned := value;
END_IF
```

Una vez llamada, la función obtiene los siguientes parámetros de entrada:
- `value`: el valor que se va a asignar al tipo de rango
- `lower`: el límite inferior del rango
- `upper`: el límite superior del rango

Mientras el valor asignado permanezca dentro del rango válido, se utilizará como valor de retorno de la función. De lo contrario, como consecuencia de la infracción de rango, se devolverá el límite superior o inferior del rango de campo.

La asignación `i:=10*y` ahora se reemplazará implícitamente por

```
i := CheckRangeSigned(10*y, -4095, 4095);
```

Si `y`, por ejemplo, tiene el valor `1000`, la variable `i` no se asignará a 10*1000=10000 (como en la implementación original), sino al límite superior del rango, que es 4095.

Lo mismo se aplica a la función `CheckRangeUnsigned`.

NOTA: Si ninguna de las funciones `CheckRangeSigned` o `CheckRangeUnsigned` está presente, no tendrá lugar ninguna verificación de tipo de subárea durante el tiempo de ejecución. En este caso, la variable `i` podría adoptar cualquier valor entre –32768 y 32767 en cualquier momento.

Capítulo 29
Directrices de programación

Contenido de este capítulo

Este capítulo contiene las siguientes secciones:

Sección 29.1
Convención sobre nombres

Información general

Creación de nombres de identificación

Elija una descripción breve y relevante en inglés para cada identificador: el nombre base. El nombre base debería ser autoexplicativo. Ponga en mayúscula la primera letra de cada palabra del nombre base. Escriba el resto de la palabra en minúsculas (ejemplo: `FileSize`). Este nombre base recibe los prefijos para indicar el ámbito y las propiedades.

Cuando sea posible, el identificador no debe contener más de 20 caracteres. Este valor es orientativo. Puede ajustar el número hacia arriba o hacia abajo en caso necesario.

Si se emplean abreviaturas de términos estándar (TP, JK-FlipFlop, etc.), los nombres no deben contener más de 3 letras mayúsculas seguidas.

Mayúsculas/minúsculas

Tenga en cuenta el uso de mayúsculas/minúsculas, especialmente en el caso de los prefijos, para mejorar la lectura cuando se emplean identificadores en el programa IEC.

NOTA: El compilador no se escribe en mayúsculas.

Caracteres válidos

Utilice solamente las siguientes letras, números y caracteres especiales en los identificadores:

del 0 al 9 y de la A a la Z en mayúscula y en minúscula

Con el fin de visualizar los prefijos de forma clara, se utiliza un guión bajo como separador. La sintaxis se explica en la sección correspondiente de prefijos.

No utilice guiones bajos en el nombre base.

Ejemplos

Identificador recomendado	Identificadores no recomendados
diState	diSTATE
xInit	x_Init
diCycleCounter	diCyclecounter
lrRefVelocity	lrRef_Velocity
c_lrMaxPosition	clrMaxPosition
FC_PidController	FC_PIDController

Sección 29.2
Prefijos

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Partes de prefijo

Descripción general

Los prefijos se usan para asignar nombres por función.

Las siguientes partes de prefijos están disponibles:

Parte de prefijo	Uso	Sintaxis	Ejemplo
Prefijo de ámbito *(véase página 696)*	Ámbito de variables y constantes	[ámbito]_[designador]	`G_diFirstUserFault`
Prefijo de tipo de datos *(véase página 697)*	Identificación del tipo de datos de variables y constantes	[tipo][designador]	`xEnable`
Prefijo de propiedad *(véase página 699)*	Identificación de las propiedades de variables y constantes	[propiedad]_[designador]	`c_iNumberOfAxes`
Prefijo de POU *(véase página 701)*	Identificación de la implementación de la POU como función, bloque de funciones o programa	[POU]_[designador]	`FB_VisuController`
Prefijo de espacio de nombres *(véase página 702)*	Para POU, tipos de datos, variables y constantes declaradas en una biblioteca	[espacio de nombres].[identificador]	`TPL.G_dwErrorCode`

Orden de los prefijos

Descripción general

Los indicadores incluyen el prefijo de ámbito y el prefijo de tipo. Use el prefijo de propiedad según la propiedad de las variables (por ejemplo, para las constantes). Se utiliza un prefijo de espacio de nombres adicional para las bibliotecas.

Orden obligatoria

La siguiente orden es obligatoria:

ámbito][propiedad][_][tipo][identificador]

Los prefijos de ámbito y los prefijos de propiedad se separan de los prefijos de tipo con un guión bajo (_).

Ejemplo

```
Gc_dwErrorCode    : DWORD;
diCycleCounter    : DINT;
```

Se utiliza el prefijo de espacio de nombres adicional para las bibliotecas:

[espacio de nombres].[ámbito][propiedad][_][tipo][identificador]

Ejemplo

```
ExampleLibrary.Gc_dwErrorCode
```

Unidades de organización de programa (POU) independientes

Introduzca un guión bajo para separar los prefijos de las unidades de organización del programa (funciones, bloques de funciones y programas) de los identificadores:

[POU][_][identificador]

Ejemplo

```
FB_MotionCorrection
```

Use el prefijo de espacio de nombres adicional para bibliotecas:

[espacio de nombres].[POU][_][identificador]

Los prefijos de espacios de nombres se separan de los prefijos de POU con un punto (.).

Ejemplo

```
ExampleLibrary.FC_SetError()
```

Unidades de organización de programa (POU) dependientes

Los métodos, las acciones y las propiedades se consideran POU dependientes. Se emplean en un nivel inferior al POU independiente.

Los métodos y las acciones no tienen prefijos.

Las propiedades reciben el prefijo de tipo de su valor de retorno.

Ejemplo

```
PROPERTY lrVelocity : LREAL
```

Prefijo de ámbito

Descripción general

El prefijo de ámbito indica el ámbito de variables y constantes. Indica si es una variable local o global, o una constante.

Las variables globales se indican mediante una `G_` mayúscula y se añade un prefijo de propiedad `c` a las constantes globales (seguido de un subrayado en cada caso).

NOTA: Además, identifica las variables y constantes globales de bibliotecas con el espacio de nombres de la biblioteca.

Prefijo de ámbito	Tipo	Uso	Ejemplo
sin prefijo	`VAR`	variable local	`xEnable`
`G_`	`VAR_GLOBAL`	variable global	`G_diFirstUserFault`
`Gc_`	`VAR_GLOBAL CONSTANT`	constante global	`Gc_dwErrorCode`

Ejemplo

```
VAR_GLOBAL CONSTANT
    Gc_dwExample : DWORD := 16#0000001A;
END_VAR
```

Acceso a la variable global de una biblioteca con el espacio de nombres `INF`:

```
INF.G_dwExample := 16#0000001A;
```

Prefijo de tipo de datos

Tipos de datos estándar

El prefijo del tipo de datos identifica el tipo de datos de las variables y las constantes.

NOTA: El prefijo del tipo de datos también puede ser compuesto; por ejemplo, en el caso de punteros, referencias y matrices. El puntero o la matriz aparece en primer lugar, seguido del prefijo del tipo de puntero o del tipo de matriz.

En la tabla figuran los prefijos de tipo de datos del estándar IEC 61131-3, así como los prefijos de las ampliaciones del estándar.

Prefijo de tipo de datos	Tipo	Uso (ubicación de memoria)	Ejemplo
x	BOOL	booleano (8 bits)	xName
by	BYTE	secuencia de bits (8 bits)	byName
w	WORD	secuencia de bits (16 bits)	wName
dw	DWORD	secuencia de bits (32 bits)	dwName
lw	LWORD	secuencia de bits (64 bits)	lwName
si	SINT	entero corto (8 bits)	siName
i	INT	entero (16 bits)	iName
di	DINT	entero doble (32 bits)	diName
li	LINT	entero largo (64 bits)	liName
uli	ULINT	entero largo (64 bits)	uliName
usi	USINT	entero corto (8 bits)	usiName
ui	UINT	entero (16 bits)	uiName
udi	UDINT	entero doble (32 bits)	udiName
r	REAL	número de coma flotante (32 bits)	rName
lr	LREAL	número de coma flotante doble (64 bits)	lrName
dat	DATE	fecha (32 bits)	datName
t	TOD	hora (32 bits)	tName
dt	DT	fecha y hora (32 bits)	dtName
tim	TIME	duración (32 bits)	timName
ltim	LTIME	duración (64 bits)	ltimName
s	STRING	cadena de caracteres ASCII	sName
ws	WSTRING	cadena de caracteres Unicode	wsName
p	pointers	puntero	pxName
r	reference	referencia	rxName

Prefijo de tipo de datos	Tipo	Uso (ubicación de memoria)	Ejemplo
a	array	campo	axName
e	enumeration	tipo de lista	eName
st	struct	estructura	stName
if	interface	interfaz	ifMotion
ut	union	unión	uName
fb	function block	bloque de funciones	fbName

Ejemplos

```
piCounter: POINTER TO INT;
aiCounters: ARRAY [1..22] OF INT;
paiRefCounter: POINTER TO ARRAY [1..22] OF INT;
apstTest   : ARRAY[1..2] OF POINTER TO ST_MotionStructure;
rdiCounter : REFERENCE TO DINT;
ifMotion   : IF_Motion;
```

Prefijo de propiedad

Descripción general

El prefijo de propiedad identifica las propiedades de las variables y constantes.

Tipo de prefijo	Uso	Sintaxis	Ejemplo
c_	VAR CONSTANT	constante local	c_xName
r_	VAR RETAIN	variable remanente de tipo retención	r_xName
p_	VAR PERSISTENT	variable remanente de tipo persistente	p_xName
rp_	VAR PERSISTENT	variable remanente de tipo retención persistente	rp_xName
i_	VAR_INPUT	parámetro de entrada de un POU	i_xName
q_	VAR_OUTPUT	parámetro de salida de un POU	q_xName
iq_	VAR_IN_OUT	parámetro de entrada/salida de un POU	iq_xName
ati_	AT %IX x.y AT %IB z AT %IW k	variable de entrada que debería escribir en el área de entrada de IEC	ati_x0_0MasterEncoderInitOK
atq_	AT %QX x.y AT %QB z AT %QW k	variable de salida que debería escribir en el área de salida de IEC	atq_w18AxisNotDone
atm_	AT %MX x.y AT %MB z AT %MW k	variable de marcador que debería escribir en el área de marcador de IEC	atm_w19ModuleNotReady

NOTA:

- No declare constantes como RETAIN o PERSISTENT.
- No declare ninguna variable RETAIN dentro de los POU. Esto administra todo el POU en el área de memoria retentiva.

Ejemplo de AT - Variables declaradas

El nombre de la variable declarada AT también contiene el tipo de variable de destino. Se utiliza del mismo modo que el prefijo de tipo.

```
ati_xEncoderInit AT %IX0.0 : BOOL;
atq_wAxisNotDone AT %QW18 : WORD;
atm_wModuleNotReady AT %MW19 : WORD;
```

NOTA: Una variable también se puede asignar a una dirección en el cuadro de diálogo de asignaciones de un dispositivo en la configuración del controlador (editor de dispositivos). En la documentación del dispositivo se describe si este ofrece este cuadro de diálogo.

Prefijo de POU

Descripción general

Las unidades de organización de programa (POU) siguientes se definen en el estándar IEC 61131-3:

- función
- bloque de funciones
- programa
- estructura de datos
- tipo de lista
- unión
- interfaz

La designación está formada por un prefijo de POU y un nombre lo más corto posible (por ejemplo, `FB_GetResult`). Al igual que en el caso de una variable, ponga la primera letra de cada palabra en mayúsculas en el nombre base. Escriba el resto en letras minúsculas. Forme un nombre de POU compuesto a partir de un verbo y un nombre.

El prefijo se escribe con un carácter de subrayado antes del nombre e identifica el tipo de POU según la tabla:

Prefijo de POU	Tipo	Uso	Ejemplo
`SR_`	`PROGRAM`	programa	`SR_FlowPackerMachine`
`FB_`	`FUNCTION_BLOCK`	bloques de funciones	`FB_VisuController`
`FC_`	`FUNCTION`	funciones	`FC_SetUserFault`
`ST_`	`STRUCT`	estructura de datos	`ST_StandardModuleInterface`
`ET_`	`Enumeration`	tipo de lista	`ST_StandardModuleInterface`
`UT_`	`UNION`	unión	`UT_Values`
`IF_`	`INTERFACE`	interfaz	`IF_CamProfile`

Prefijo de espacio de nombres

Descripción general

Puede ver el espacio de nombres de una biblioteca en el **Administrador de bibliotecas**. Use un acrónimo breve (PacDriveLib -> `PDL`) como espacio de nombres. No cambie el espacio de nombres predeterminado de una biblioteca.

Para reservar un espacio de nombres no ambiguo para sus propias bibliotecas, desarrolladas por usted mismo, póngase en contacto con el responsable de Schneider Electric.

Ejemplo

Una función `FC_DoSomething()` se encuentra en la biblioteca TestlibraryA (espacio de nombres `TLA`) así como en TestlibraryB (espacio de nombres `TLB`). A la función respectiva se accede añadiendo un prefijo al espacio de nombres.

Si ambas bibliotecas se encuentran dentro de un proyecto, la siguiente llamada produce un error detectado durante la compilación:

```
FC_DoSomething();
```

En este caso, es necesario definir claramente a qué POU se tiene que llamar.

```
TLA.FC_DoSomething();
TLB.FC_DoSomething();
```

Capítulo 30
Operadores

Descripción general

SoMachine admite todos los operadores IEC. A diferencia de las funciones estándar, estos operadores se reconocen implícitamente a través del proyecto.

Además de los operadores IEC, se admiten los siguientes operadores, los cuales no se especifican en la norma:

- `ANDN`
- `ORN`
- `XORN`
- `SIZEOF` (consulte operadores aritméticos *(véase página 704)*)
- `ADR`
- `BITADR`
- Operador de contenido (consulte operadores de dirección *(véase página 744)*)
- Algunos operadores de ámbito *(véase página 794)*

Contenido de este capítulo

Este capítulo contiene las siguientes secciones:

Sección 30.1
Operadores aritméticos

Descripción general

Los siguientes operadores, prescritos por la norma IEC1131-3, están disponibles:

- ADD
- MUL
- SUB
- DIV
- MOD
- MOVE

Además, existe el siguiente operador estándar/de extensión:

- SIZEOF

Tenga en cuenta los posibles desbordes de las operaciones aritméticas si el valor resultante supera el rango de tipo de datos usado para la variable de resultado. Esto podría provocar que los valores que se escriben en la máquina sean los bajos en lugar de los altos o viceversa.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

Certifique siempre los operandos y resultados usados en operaciones matemáticas para evitar el desborde aritmético.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

ADD

Descripción general

Operador IEC para la adición de variables

Tipos permitidos

- BYTE
- WORD
- DWORD
- LWORD
- SINT
- USINT
- INT
- UINT
- DINT
- UDINT
- LINT
- ULINT
- REAL
- LREAL
- TIME
- TIME_OF_DAY(TOD)
- DATE
- DATE_AND_TIME(DT)

Para los tipos de datos de tiempo, se pueden realizar las siguientes combinaciones:

- TIME+TIME=TIME
- TOD+TIME=TOD
- DT+TIME=DT

En el editor FBD/LD, el operador `ADD` es un cuadro ampliable. Esto significa que se puede usar un cuadro con varias entradas en lugar de una serie de cuadros `ADD` concatenados. Use el comando **Insertar entrada de módulo** para añadir más entradas. El número es ilimitado.

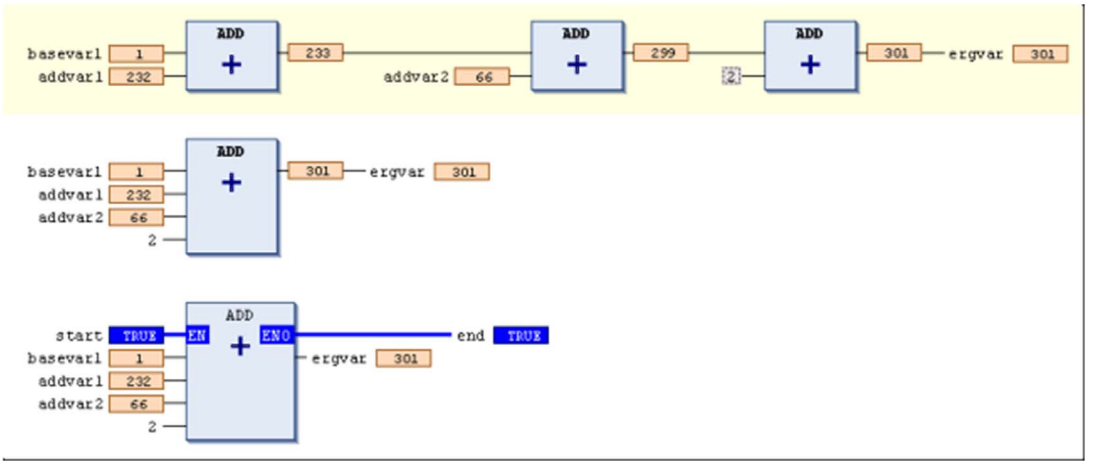
Ejemplo en IL

```
LD      7
ADD     2
ADD     4
ADD     7
ST      iVar
```

Ejemplo en ST

```
var1 := 7+2+4+7;
```

Ejemplos en FBD



1. serie de cuadros `ADD`
2. cuadro `ADD` ampliado
3. cuadro `ADD` con parámetros `EN`/`ENO`

MUL

Descripción general

Operador IEC para multiplicación de variables

Tipos permitidos

- BYTE
- WORD
- DWORD
- LWORD
- SINT
- USINT
- INT
- UINT
- DINT
- UDINT
- LINT
- ULINT
- REAL
- LREAL
- TIME

Las variables TIME se pueden multiplicar por variables enteras.

En el editor FBD/LD, el operador `MUL` es un cuadro ampliable. Esto significa que se puede usar un cuadro con varias entradas en lugar de una serie de cuadros `MUL` concatenados. Use el comando **Insertar entrada de módulo** para añadir más entradas. El número es ilimitado.

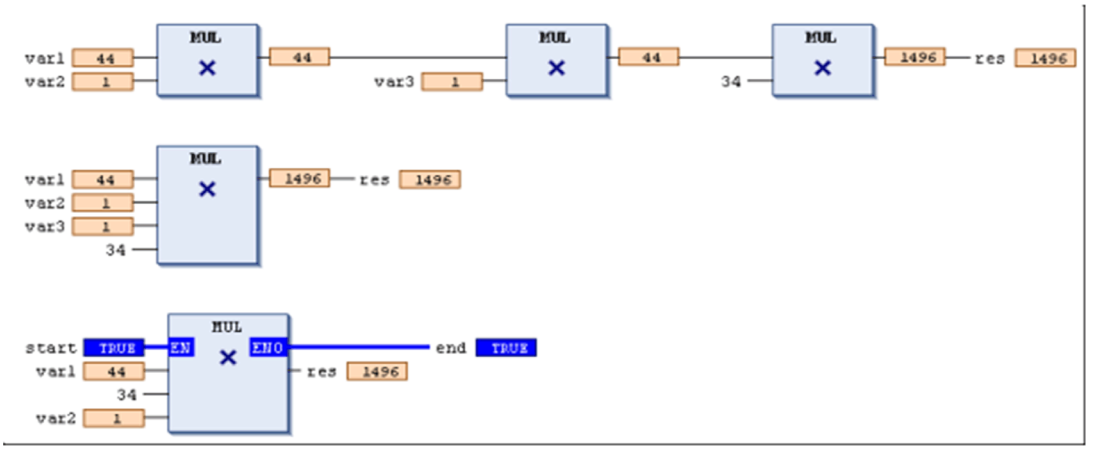
Ejemplo en IL

```
LD      7
MUL     2       ,
        4       ,
        7
ST      Var1
```

Ejemplo en ST

```
var1 := 7*2*4*7;
```

Ejemplos en FBD



1. serie de cuadros `MUL`
2. cuadro `MUL` ampliado
3. cuadro `MUL` con parámetros `EN/ENO`

SUB

Descripción general

Operador IEC para restar una variable de otra.

Tipos permitidos:

- BYTE
- WORD
- DWORD
- LWORD
- SINT
- USINT
- INT
- UINT
- DINT
- UDINT
- LINT
- ULINT
- REAL
- LREAL
- TIME
- TIME_OF_DAY(TOD)
- DATE
- DATE_AND_TIME(DT)

Para los tipos de datos de tiempo, se pueden realizar las siguientes combinaciones:

- TIME–TIME=TIME
- DATE–DATE=TIME
- TOD–TIME=TOD
- TOD–TOD=TIME
- DT-TIME=DT
- DT-DT=TIME

Tenga en cuenta que los valores TIME negativos no están definidos.

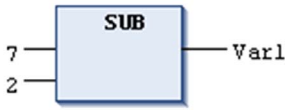
Ejemplo en IL

```
LD     7
SUB    2
ST     Var1
```

Ejemplo en ST

```
var1 := 7-2;
```

Ejemplo en FBD



DIV

Descripción general

Operador IEC para la división de una variable entre otra.

Tipos permitidos:

- BYTE
- WORD
- DWORD
- LWORD
- SINT
- USINT
- INT
- UINT
- DINT
- UDINT
- LINT
- ULINT
- REAL
- LREAL
- TIME

Las variables TIME pueden dividirse entre variables enteras.

Ejemplo en IL

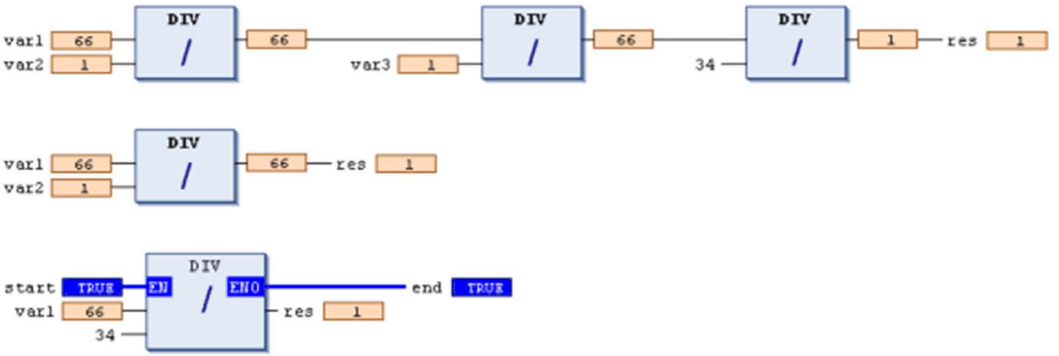
(El resultado en `Var1` es 4).

```
LD      8
DIV     2
ST      Var1
```

Ejemplo en ST

```
var1 := 8/2;
```

Ejemplos en FBD



1. series de módulos `DIV`
2. módulo`DIV` único
3. módulo `DIV` con parámetros `EN/ENO`

Sistemas de destino distintos pueden comportarse de manera diferente respecto a un error de división entre cero. Puede dar lugar a una parada (HALT) del controlador o no llegarse a detectar.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

Use las funciones de comprobación descritas en este documento o escriba sus propias comprobaciones para evitar la división entre cero en el código de programación.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

NOTA: Para obtener más información acerca de las funciones de verificación implícitas consulte el capítulo *POU para comprobaciones implícitas (véase página 195)*.

Funciones de verificación

Puede utilizar las siguientes funciones de verificación para comprobar el valor del divisor y evitar una división entre cero.

- `CheckDivInt`
- `CheckDivLint`
- `CheckDivReal`
- `CheckDivLReal`

Cada una de las divisiones que tiene lugar en el código relacionado provocará una llamada anterior a estas funciones.

Utilice el comando **Agregar objeto** para insertarlas en la aplicación. Allí, elija el objeto **POU para comprobaciones implícitas**. Marque la casilla de verificación de una función de verificación correspondiente. Seleccione un lenguaje de implementación. Confirme su elección con **Open**. La función seleccionada se abrirá en el editor correspondiente a la función de implementación elegida. Independientemente de esta elección, la parte de declaraciones de las funciones está preestablecida. No puede modificarla, salvo añadiendo variables locales. Una implementación predeterminada de las funciones que puede modificar está disponible en ST.

Implementación predeterminada de la función `CheckDivReal`

Parte de declaraciones:

```
// Implicitly generated code : DO NOT EDIT
FUNCTION CheckDivReal : REAL
VAR_INPUT
 divisor:REAL;
END_VAR
```

Parte de implementación:

```
// Implicitly generated code : only an suggestion for implementation
IF divisor = 0 THEN
 CheckDivReal:=1;
ELSE
 CheckDivReal:=divisor;
END_IF;
```

El operador `DIV` utiliza la salida de función`CheckDivReal` como un divisor. En el siguiente ejemplo se prohíbe una división entre cero y el valor inicializado 0 del divisor `d` es cambiado a 1 por `CheckDivReal` antes de la ejecución de la división. Por lo tanto, el resultado de la división es 799.

```
PROGRAM PLC_PRG
VAR
 erg:REAL;
 v1:REAL:=799;
 d:REAL;
END_VAR
erg:= v1 / d;
```

MOD

Descripción general

Operador IEC para la división de módulo de una variable entre otra.

Tipos permitidos:
- BYTE
- WORD
- DWORD
- LWORD
- SINT
- USINT
- INT
- UINT
- DINT
- UDINT
- LINT
- ULINT

El resultado de esta función es el resto entero de la división.

Sistemas de destino distintos pueden comportarse de manera diferente respecto a un error de división entre cero. Puede dar lugar a una parada (HALT) del controlador o no llegarse a detectar.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

Use las funciones de comprobación descritas en este documento o escriba sus propias comprobaciones para evitar la división entre cero en el código de programación.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

NOTA: Para obtener más información sobre las funciones de comprobación implícitas, consulte el capítulo *POU para comprobaciones implícitas (véase página 195)*.

Ejemplo en IL

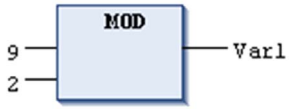
El resultado en `Var1` es 1.

```
LD      9
MOD     2
ST      Var1
```

Ejemplo en ST

```
var1 := 9 MOD 2;
```

Ejemplos en FBD



MOVE

Descripción general

Operador IEC para la asignación de una variable a otra variable de un tipo de datos apropiado.

El operador `MOVE` se puede utilizar para todos los tipos de datos.

Como `MOVE` está disponible como un cuadro en los editores gráficos FBD, LD, CFC, la funcionalidad `EN/ENO` (desbloqueo) también se puede aplicar a una asignación de variable.

Ejemplo en CFC en conjunción con la función EN/ENO

Solo si `en_i` es TRUE, `var1` se asignará a `var2`.

Ejemplo en IL

Resultado: `var2` obtiene el valor de `var1`

```
LD      var1
MOVE
ST      var2
```

Obtiene el mismo resultado con

```
LD      var1
ST      var2
```

Ejemplo en ST

```
ivar2 := MOVE(ivar1);
```

Obtiene el mismo resultado con

```
ivar2 := ivar1;
```

SIZEOF

Descripción general

Este operador aritmético no se especifica en la norma IEC 61131-3.

Puede utilizarlo para determinar el número de bytes requeridos por la variable `x` proporcionada.

El operador `SIZEOF` devuelve un valor no asignado. El tipo del valor de retorno se adaptará al tamaño encontrado de la variable `x`.

Valor de retorno de SIZEOF(x)	Tipo de datos de la constante usada implícitamente para el tamaño encontrado
0 ≤tamaño de x < 256	USINT
256 ≤tamaño de x < 65.536	UINT
65.536 ≤tamaño de x < 4.294.967.296	UDINT
4.294.967.296 ≤tamaño de x	ULINT

Ejemplo en ST

```
var1 := SIZEOF(arr1);  (* d.h.: var1:=USINT#10; *)
```

Ejemplo en IL

El resultado es 10

```
arr1:ARRAY[0..4] OF INT;
Var1:INT;

LD     arr1
SIZEOF
ST     Var1
```

Sección 30.2
Operadores de cadenas de bits

Descripción general

Los siguientes operadores de cadenas de bits están disponibles y cumplen con la norma IEC1131-3:

- AND *(véase página 719)*
- OR *(véase página 720)*
- XOR *(véase página 721)*
- NOT *(véase página 722)*

Los siguientes operadores no están especificados en la norma y no están disponibles:

- `ANDN`
- `ORN`
- `XORN`

Los operadores de cadenas de bits comparan los respectivos bits de 2 o diversos operandos.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

AND

Descripción general

El operador de cadenas de bits IEC para el AND a nivel de bit de los operandos de bit.

Si cada bit de entrada es 1, el bit que resulta será 1; de lo contrario, será 0.

Tipos permitidos:

- BOOL
- BYTE
- WORD
- DWORD
- LWORD

Ejemplo en IL

El resultado en `Var1` es 2#1000_0010.

```
Var1:BYTE;
```

```
LD      2#1001_0011
AND     2#1000_1010
ST      var1
```

Ejemplo en ST

```
var1 := 2#1001_0011 AND 2#1000_1010
```

Ejemplo en FBD

OR

Descripción general

El operador de cadenas de bits IEC para el OR a nivel de bit de los operandos de bit.

Si al menos 1 de los bits de entrada es 1, el bit que resulta será 1; de lo contrario, será 0.

Tipos permitidos:

- BOOL
- BYTE
- WORD
- DWORD
- LWORD

Ejemplo en IL

El resultado en `var1` es 2#1001_1011.

```
var1:BYTE;
```

```
LD      2#1001_0011
OR      2#1000_1010
ST      Var1
```

Ejemplo en ST

```
Var1 := 2#1001_0011 OR 2#1000_1010
```

Ejemplo en FBD

XOR

Descripción general

Operador IEC de cadena de bit para XOR a nivel de bit de operandos de bit.

Si solo uno de los bits de entrada es 1, entonces el bit resultante será 1; si ambos o ninguno son 1, el bit resultante será 0.

Tipos permitidos:
- BOOL
- BYTE
- WORD
- DWORD
- LWORD

NOTA: `XOR` permite añadir entradas adicionales. Si hay más de dos entradas disponibles, entonces se realiza una operación XOR sobre las dos primeras entradas. El resultado, a su vez, será XOR combinado con la entrada 3, etc. Esto tiene el efecto de que un número de entradas impar producirá un bit resultante = 1.

Ejemplo en IL

El resultado es 2#0001_1001.

```
Var1:BYTE;

LD      2#1001_0011
XOR     2#1000_1010
ST      var1
```

Ejemplo en ST

```
Var1 := 2#1001_0011 XOR 2#1000_1010
```

Ejemplo en FBD

NOT

Descripción general

El operador de cadenas de bits IEC para la operación NOT a nivel de bit de un operando de bit.

El bit que resulta será 1 si el bit de entrada correspondiente es 0 y viceversa.

Tipos permitidos

- BOOL
- BYTE
- WORD
- DWORD
- LWORD

Ejemplo en IL

El resultado en `Var1` es 2#0110_1100.

```
Var1:BYTE;

LD      2#1001_0011
NOT
ST      var1
```

Ejemplo en ST

```
Var1 := NOT 2#1001_0011
```

Ejemplo en FBD

Sección 30.3
Operadores de desplazamiento de bits

Contenido de esta sección

Esta sección contiene los siguientes apartados:

SHL

Descripción general

El operador IEC para el desplazamiento a la izquierda de bits de un operando.

```
erg:= SHL (in, n)
```

`in`: operando que hay que desplazar a la izquierda

`n`: número de bits, por los que `in` se desplaza a la izquierda

NOTA: Si `n` supera el ancho del tipo de datos, dependerá del sistema de destino cuántos operandos BYTE, WORD, DWORD y LWORD se rellenarán. Algunos causan que se rellenen con ceros (`0`) y otros con `n MOD <register width>`.

NOTA: La cantidad de bits que se tiene en cuenta para la operación aritmética depende del tipo de datos de la variable de entrada. Si la variable de entrada es una constante, se tendrá en cuenta el tipo de datos mínimo posible. El tipo de datos de la variable de salida no tiene ningún efecto en la operación aritmética.

Ejemplos

Observe en el ejemplo siguiente en notación hexadecimal los distintos resultados para `erg_byte` y `erg_word`. El resultado depende del tipo de datos de la variable de entrada (BYTE o WORD), aunque los valores de las variables de entrada `in_byte` y `in_word` son las mismas.

Ejemplo en ST

```
PROGRAM shl_st
VAR
 in_byte : BYTE:=16#45; (* 2#01000101 )
 in_word : WORD:=16#0045; (* 2#0000000001000101 )
 erg_byte : BYTE;
 erg_word : WORD;
 n: BYTE :=2;
END_VAR
erg_byte:=SHL(in_byte,n); (* Result is 16#14, 2#00010100 *)
erg_word:=SHL(in_word,n); (* Result is 16#0114, 2#0000000100010100 *)
```

Ejemplo en FBD

Ejemplo en IL

```
LD      in_byte
SHL     2
ST      erg_byte
```

SHR

Descripción general

Operador IEC para desplazamiento a la derecha a nivel de bit de un operando.

```
erg:= SHR (in, n)
```

`in`: operando que se debe desplazar a la derecha

`n`: número de bits, por el que `in` se desplaza a la derecha

NOTA: Si `n` sobrepasa el ancho del tipo de datos, dependerá del sistema de destino cómo se rellenarán los operandos BYTE, WORD, DWORD y LWORD. Algunas causas se llenan con ceros (`0`), otras con `n MOD <register width>`.

Ejemplos

El ejemplo siguiente en notación hexadecimal muestra los resultados de la operación aritmética que depende del tipo de la variable de entrada (BYTE o WORD).

Ejemplo en ST

```
PROGRAM shr_st
VAR
 in_byte : BYTE:=16#45; (* 2#01000101 )
 in_word : WORD:=16#0045; (* 2#0000000001000101 )
 erg_byte : BYTE;
 erg_word : WORD;
 n: BYTE :=2;
END_VAR
erg_byte:=SHR(in_byte,n); (* Result is 16#11, 2#00010001 *)
erg_word:=SHR(in_word,n); (* Result is 16#0011, 2#0000000000010001 *)
```

Ejemplo en FBD

Ejemplo en IL

```
LD      in_byte
SHR     2
ST      erg_byte
```

ROL

Descripción general

Operador IEC para rotación a nivel de bit de un operando a la izquierda.

```
erg:= ROL (in, n)
```

Tipos de datos permitidos

- BYTE
- WORD
- DWORD
- LWORD

`in` se desplazará la posición de 1 bit a la izquierda `n` veces mientras que el bit que está situado más a la izquierda se volverá a insertar desde la derecha.

NOTA: La cantidad de bits que se tiene en cuenta para la operación aritmética depende del tipo de datos de la variable de entrada. Si la variable de entrada es una constante, se tendrá en cuenta el tipo de datos mínimo posible. El tipo de datos de la variable de salida no tiene ningún efecto en la operación aritmética.

Ejemplos

Observe en el ejemplo siguiente en notación hexadecimal los distintos resultados para `erg_byte` y `erg_word`. El resultado depende del tipo de datos de la variable de entrada (BYTE o WORD), aunque los valores de las variables de entrada `in_byte` y `in_word` son las mismas.

Ejemplo en ST

```
PROGRAM rol_st
VAR
in_byte : BYTE:=16#45;
in_word : WORD:=16#45;
erg_byte : BYTE;
erg_word : WORD;
n: BYTE :=2;
END_VAR
erg_byte:=ROL(in_byte,n); (* Result is 16#15 *)
erg_word:=ROL(in_word,n); (* Result is 16#0114 *)
```

Ejemplo en FBD

Ejemplo en IL

```
LD      in_byte
ROL     n
ST      erg_byte
```

ROR

Descripción general

Operador IEC para rotación a nivel de bit de un operando a la derecha.

```
erg:= ROR (in, n)
```

Tipos de datos permitidos

- BYTE
- WORD
- DWORD
- LWORD

`in` se desplazará la posición de 1 bit a la derecha `n` veces mientras que el bit que está situado más a la izquierda se volverá a insertar desde la izquierda.

NOTA: La cantidad de bits que se consideran para la operación aritmética depende del tipo de datos de la variable de entrada. Si la variable de entrada es una constante, se considerará el tipo de datos mínimo posible. El tipo de datos de la variable de salida no tiene ningún efecto en la operación aritmética.

Ejemplos

Observe en el ejemplo siguiente en notación hexadecimal los distintos resultados para `erg_byte` y `erg_word`. El resultado depende del tipo de datos de la variable de entrada (BYTE o WORD), aunque los valores de las variables de entrada `in_byte` y `in_word` son las mismas.

Ejemplo en ST

```
PROGRAM ror_st
VAR
in_byte : BYTE:=16#45;
in_word : WORD:=16#45;
erg_byte : BYTE;
erg_word : WORD;
n: BYTE :=2;
END_VAR
erg_byte:=ROR(in_byte,n); (* Result is 16#51 *)
erg_word:=ROR(in_word,n); (* Result is 16#4011 *)
```

Ejemplo en FBD

Ejemplo en IL

```
LD      in_byte
ROR     n
ST      erg_byte
```

Sección 30.4
Operadores de selección

Descripción general

Las operaciones de selección también se pueden realizar con variables.

Con el objetivo de facilitar la comprensión, en este documento solo se ofrecen los siguiente ejemplos que emplean constantes como operadores:

- SEL *(véase página 732)*
- MAX *(véase página 733)*
- MIN *(véase página 734)*
- LIMIT *(véase página 735)*
- MUX *(véase página 736)*

Contenido de esta sección

Esta sección contiene los siguientes apartados:

SEL

Descripción general

Operador de selección IEC para selecciones binarias.

G determina si se asigna IN0 o IN1 a OUT.

```
OUT := SEL(G, IN0, IN1)
```
significa:

```
OUT := IN0;
```
if G=FALSE

```
OUT := IN1;
```
if G=TRUE

Tipos de datos permitidos:

IN0, IN1, OUT): cualquier tipo

G: BOOL

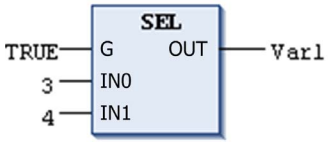
Ejemplo en IL

```
LD  TRUE
SEL 3,4   (* IN0 = 3, IN1 =4 *)
ST  Var1  (* result is 4 *)
LD  FALSE
SEL 3,4
ST  Var1  (* result is 3 *)
```

Ejemplo en ST

```
Var1:=SEL(TRUE,3,4); (* result is 4 *)
```

Ejemplo en FBD



Nota

NOTA: Una expresión que se produce delante de IN1 no se procesará si IN0 es TRUE. Una expresión que se produce delante de IN2 no se procesará si IN0 es FALSE.

MAX

Descripción general

Operador de selección IEC realizando una función máxima.

El operador `MAX` devuelve el mayor de 2 valores.

```
OUT := MAX(IN0, IN1)
```

`IN0`, `IN1` y `OUT` pueden ser cualquier tipo de variable.

Ejemplo en IL

El resultado es 90

```
LD      90
MAX     30
MAX     40
MAX     77
ST      Var1
```

Ejemplo en ST

```
Var1:=MAX(30,40); (* Result is 40 *)
Var1:=MAX(40,MAX(90,30)); (* Result is 90 *)
```

Ejemplo en FBD

MIN

Descripción general

Operador de selección IEC realizando una función mínima.

El operador `MIN` devuelve el menor de 2 valores.

```
OUT := MIN(IN0, IN1)
```

`IN0`, `IN1` y `OUT` pueden ser cualquier tipo de variable.

Ejemplo en IL

El resultado es 30

```
LD      90
MIN     30
MIN     40
MIN     77
ST      Var1
```

Ejemplo en ST

```
Var1:=MIN(90,30); (* Result is 30 *);
Var1:=MIN(MIN(90,30),40); (* Result is 30 *);
```

Ejemplo en FBD

LIMIT

Descripción general

Operador de selección IEC que ejecuta una función de limitación.

```
OUT := LIMIT(Min, IN, Max) means:
OUT := MIN (MAX (IN, Min), Max)
```

`Max` es el límite superior y `Min` el inferior del resultado. Si el valor `IN` sobrepasa el límite superior `Max`, `LIMIT` devolverá `Max`. Si `IN` cae por debajo de `Min`, el resultado será `Min`.

`IN` y `OUT` pueden ser cualquier tipo de variable.

Ejemplo en IL

El resultado es 80

```
LD     90
LIMIT  30     ,
       80
ST     Var1
```

Ejemplo en ST

```
Var1:=LIMIT(30,90,80); (* Result is 80 *);
```

MUX

Descripción general

Operador de selección IEC para operaciones de multiplexación.

`OUT := MUX(K, IN0,...,INn))` significa:

`OUT := INk`

`IN0, ...,INn` y `OUT` pueden ser cualquier tipo de variable.

`K` tiene que ser BYTE, WORD, DWORD, LWORD, SINT, USINT, INT, UINT, DINT, LINT, ULINT o UDINT.

`MUX` selecciona el valor `K`° entre un grupo de valores.

Ejemplo en IL

El resultado es 30

```
LD      0
MUX     30      ,
        40      ,
        50      ,
        60      ,
        70      ,
        80
ST      Var1
```

Ejemplo en ST

```
Var1:=MUX(0,30,40,50,60,70,80); (* Result is 30 *);
```

NOTA: No se procesará una expresión que se produzca delante de una entrada distinta de `INk` para ahorrar tiempo de ejecución. Únicamente en la modalidad de simulación se ejecutarán todas las expresiones.

Sección 30.5
Operadores de comparación

Descripción general

Los siguientes operadores que cumplen la norma IEC1131-3 están disponibles:

- GT *(véase página 738)*
- LT *(véase página 739)*
- LE *(véase página 740)*
- GE *(véase página 741)*
- EQ *(véase página 742)*
- NE *(véase página 743)*

Contenido de esta sección

Esta sección contiene los siguientes apartados:

GT

Descripción general

Operador de comparación que realiza la función Mayor que.

El operador `GT` es un operador booleano que devuelve el valor TRUE cuando el valor del primer operando es mayor que el del segundo.

Los operandos pueden ser de cualquier tipo de datos básicos.

Ejemplo en IL

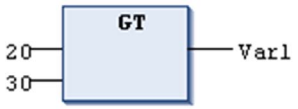
El resultado es FALSE

```
LD      20
GT      30
ST      Var1
```

Ejemplo en ST

```
VAR1 := 20 > 30;
```

Ejemplo en FBD



LT

Descripción general

Operador de comparación que realiza la función Menor que.

El operador `LT` es un operador booleano que devuelve el valor TRUE cuando el valor del primer operando es menor que el del segundo.

Los operandos pueden ser de cualquier tipo de datos básicos.

Ejemplo en IL

El resultado es TRUE

```
LD      20
LT      30
ST      Var1
```

Ejemplo en ST

```
VAR1 := 20 < 30;
```

Ejemplo en FBD

LE

Descripción general

Operador de comparación que ejecuta una función Menor o igual que.

El operador `LE` es un operador booleano que devuelve el valor TRUE cuando el valor del primer operando es menor o igual que el del segundo.

Los operandos pueden ser de cualquier tipo de datos básicos.

Ejemplo en IL

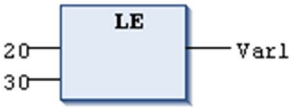
El resultado es TRUE

```
LD      20
LE      30
ST      Var1
```

Ejemplo en ST

```
VAR1 := 20 <= 30;
```

Ejemplo en FBD



GE

Descripción general

Operador de comparación que ejecuta una función Mayor o igual que.

El operador `GE` es un operador booleano que devuelve el valor TRUE cuando el valor del primer operando es mayor o igual que el del segundo.

Los operandos pueden ser de cualquier tipo de datos básicos.

Ejemplo en IL

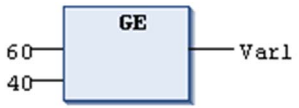
El resultado es TRUE

```
LD      60
GE      40
ST      Var1
```

Ejemplo en ST

```
VAR1 := 60 >= 40;
```

Ejemplo en FBD



EQ

Descripción general

Operador de comparación que realiza la función Igual a.

El operador `EQ` es un operador booleano que devuelve el valor TRUE cuando los operandos son iguales.

Los operandos pueden ser de cualquier tipo de datos básicos.

Ejemplo en IL

El resultado es TRUE

```
LD      40
EQ      40
ST      Var1
```

Ejemplo en ST

```
VAR1 := 40 = 40;
```

Ejemplo en FBD

NE

Descripción general

Operador de comparación que realiza la función Distinto a.

El operador `NE` es un operador booleano que devuelve el valor TRUE cuando los operandos no son iguales.

Los operandos pueden ser de cualquier tipo de datos básicos.

Ejemplo en IL

```
LD      40
NE      40
ST      Var1
```

Ejemplo en ST

```
VAR1 := 40 <> 40;
```

Ejemplo en FBD

Sección 30.6
Operadores de dirección

Contenido de esta sección

Esta sección contiene los siguientes apartados:

ADR

Descripción general

Este operador de dirección no se especifica en la norma IEC 61131-3.

`ADR` devuelve la dirección *(véase página 813)* de este argumento en un DWORD. Esta dirección se puede asignar a un puntero *(véase página 673)* dentro del proyecto.

NOTA: SoMachine permite utilizar el operador `ADR` con nombres de funciones, nombres de programas, nombres de bloque de funciones y nombres de métodos.

Consulte el capítulo *Punteros (véase página 673)*, y tenga en cuenta que los punteros de función se pueden transferir a bibliotecas externas. Sin embargo, no se puede llamar a un puntero de función dentro de SoMachine. A fin de habilitar una llamada de sistema (sistema en tiempo de ejecución), defina la propiedad de objeto correspondiente (en el menú **Vista** →**Propiedades...** → **Compilar**) del objeto de función.

Ejemplo en ST

```
dwVar:=ADR(bVAR);
```

Ejemplo en IL

```
LD      bVar
ADR
ST      dwVar
```

Consideraciones para cambios en línea

Al ejecutar el comando **Cambio en línea**, se pueden mover las variables a otra ubicación de la memoria. Durante el cambio online, se indicará si se requiere una copia.

El desplazamiento de variables puede hacer que las variables `POINTER` apunten a una memoria no válida. Por tanto, asegúrese de que no se conserve un puntero entre ciclos, sino que se reasigne en cada ciclo.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

Asigne el valor de cualquier variable de tipo `POINTER TO` antes de usarlo por primera vez dentro de un POU y en todos los ciclos siguientes.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

NOTA: Las variables `POINTER TO` de funciones y métodos no deben devolverse al emisor de llamadas de esta función ni pasarse a variables globales.

Operador de contenido

Descripción general

Este operador de dirección no se especifica en la norma IEC 61131-3. Puede desreferenciar un puntero añadiendo el operador de contenido ^ (caret ASCII o símbolo circunflejo) detrás del identificador del puntero.

Ejemplo en ST

```
pt:POINTER TO INT;
var_int1:INT;
var_int2:INT;
pt := ADR(var_int1);
var_int2:=pt^;
```

Consideraciones para cambios en línea

La ejecución del comando **Cambio online** puede cambiar el contenido de las direcciones.

⚠ ATENCIÓN

PUNTERO NO VÁLIDO

Cuando utilice punteros en las direcciones y ejecute el comando **Cambio online**, verifique siempre el contenido de los punteros.

El incumplimiento de estas instrucciones puede causar lesiones o daño al equipo.

BITADR

Descripción general

Este operador de dirección no se especifica en la norma IEC 61131-3.

BITADR devuelve el offset de bit dentro del segmento en un DWORD. El valor del offset depende de si la opción **Direccionamiento de bytes** de la configuración de destino está activada o no.

El cuarteto más alto de esa DWORD indica el área de memoria:

Memoria: 16x40000000

Entrada: 16x80000000

Salida: 16xC0000000

Ejemplo en ST

```
VAR
        var1 AT %IX2.3:BOOL;
        bitoffset: DWORD;
END_VAR
bitoffset:=BITADR(var1); (* Result if byte addressing=TRUE: 16x80000013
, if byte addressing=FALSE: 16x80000023 *)
```

Ejemplo en IL

```
LD     Var1
BITADR
ST     bitoffset
```

Consideraciones para cambios en línea

La ejecución del comando **Cambio online** puede cambiar el contenido de las direcciones.

⚠ ATENCIÓN

PUNTERO NO VÁLIDO

Cuando utilice punteros en las direcciones y ejecute el comando **Cambio online**, verifique siempre el contenido de los punteros.

El incumplimiento de estas instrucciones puede causar lesiones o daño al equipo.

Sección 30.7
Operador de llamada

CAL

Descripción general

Operador IEC para llamar a un bloque de funciones o un programa.

Use `CAL` en IL para llamar a una instancia del bloque de funciones. Coloque las variables que servirán como variables de entrada entre paréntesis a continuación del nombre de la instancia del bloque de funciones.

Ejemplo

Llamada a la instancia `Inst` de un bloque de funciones en el que las variables de entrada `Par1` y `Par2` son 0 y TRUE, respectivamente.

```
CAL INST(PAR1 := 0, PAR2 := TRUE)
```

Sección 30.8
Operadores de conversión de tipo

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Funciones de conversiones de tipo

Descripción general

No se permite convertir de forma implícita de un tipo mayor a uno menor (por ejemplo, de INT a BYTE o de DINT a WORD). Para ello, debe realizar conversiones de tipo especial. Puede convertir básicamente de cualquier tipo elemental a cualquier otro tipo elemental.

Sintaxis

<elem.type1>_TO_<elem.type2>

NOTA: En las conversiones ...TO_STRING, la cadena se genera alineada a la izquierda. Si se define demasiado corta, se cortada a partir de la derecha.

Se admiten las conversiones del tipo siguiente:

- Conversiones BOOL_TO *(véase página 751)*
- Conversiones TO_BOOL *(véase página 754)*
- Conversiones entre tipos de números integrales *(véase página 756)*
- Conversiones REAL_TO-/ LREAL_TO *(véase página 757)*
- Conversiones TIME_TO/TIME_OF_DAY *(véase página 759)*
- Conversiones DATE_TO/DT_TO *(véase página 761)*
- Conversiones STRING_TO *(véase página 763)*
- TRUNC *(véase página 765)* (conversión a DINT)
- TRUNC_INT *(véase página 766)*
- ANY_NUM_TO_<tipo de datos numéricos>
- Conversiones ANY_..._TO *(véase página 767)*

Conversiones BOOL_TO

Definición

Operador IEC para conversiones del tipo BOOL a cualquier otro tipo.

Sintaxis

BOOL_TO_<tipo de datos>

Resultados de la conversión

Los resultados de la conversión para tipos de números y tipos de cadena dependen del estado del operando:

Estado del operando	Resultado de tipos de números	Resultado de tipos de cadena
TRUE	1	TRUE
FALSE	0	FALSE

Ejemplos en ST

Ejemplos en ST con resultados de la conversión:

Ejemplo	Resultado
`i:=BOOL_TO_INT(TRUE);`	`1`
`str:=BOOL_TO_STRING(TRUE);`	`TRUE`
`t:=BOOL_TO_TIME(TRUE);`	`T#1ms`
`tof:=BOOL_TO_TOD(TRUE);`	`TOD#00:00:00.001`
`dat:=BOOL_TO_DATE(FALSE);`	`D#1970`
`dandt:=BOOL_TO_DT(TRUE);`	`DT#1970-01-01-00:00:01`

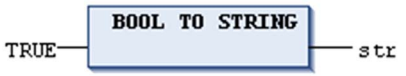
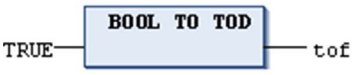
Ejemplos en IL

Ejemplos en IL con resultados de la conversión:

Ejemplo	Resultado
LD TRUE BOOL_TO_INT ST i	1
LD TRUE BOOL_TO_STRI... ST str	TRUE
LD TRUE BOOL_TO_TIME ST t	T#1ms
LD TRUE BOOL_TO_TOD ST tof	TOD#00:00:00.001
LD FALSE BOOL_TO_DATE ST dandt	D#1970-01-01
LD TRUE BOOL_TO_DT ST dandt	DT#1970-01-01-00:00:01

Ejemplos en FBD

Ejemplos en FBD con resultados de conversión:

Ejemplo	Resultado
	1
	TRUE
	T#1ms
	TOD#00:00:00.001

Ejemplo	Resultado
	D#1970-01-01
	DT#1970-01-01-00:00:01

Conversiones TO_BOOL

Definición

Operador IEC para conversiones de otro tipo de variable a BOOL.

Sintaxis

<data type>_TO_BOOL

Resultados de la conversión

El resultado es TRUE cuando el operando no es igual a 0. El resultado es FALSE cuando el operando es igual a 0.

El resultado es TRUE para variables de tipo STRING cuando el operando es TRUE. De lo contrario, el resultado es FALSE.

Ejemplos en ST

Ejemplos en ST con resultados de la conversión:

Ejemplo	Resultado
`b := BYTE_TO_BOOL(2#11010101);`	TRUE
`b := INT_TO_BOOL(0);`	FALSE
`b := TIME_TO_BOOL(T#5ms);`	TRUE
`b := STRING_TO_BOOL('TRUE');`	TRUE

Ejemplos en IL

Ejemplos en IL con resultados de la conversión:

Ejemplo	Resultado

`LD 213`
`BYTE_TO_BOOL`
`ST b`
	TRUE

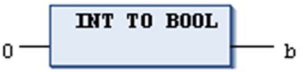
`LD 0`
`INT_TO_BOOL`
`ST b`
	FALSE

`LD T#5ms`
`TIME_TO_BOOL`
`ST b`
	TRUE

`LD TRUE`
`STRING_TO_BOOL`
`ST b`
	TRUE

Ejemplos en FBD

Ejemplos en FBD con resultados de conversión:

Ejemplo	Resultado
	TRUE
	FALSE
	TRUE
	TRUE

Conversión entre tipos de números integrales

Definición

Conversión de un tipo de número integral a otro tipo de número.

Sintaxis

<INT data type>_TO_<INT data type>

Para obtener información sobre el tipo de datos enteros, consulte el capítulo *Tipos de datos estándar (véase página 663)*.

Resultados de la conversión

Si el número que va a convertir sobrepasa el límite del rango, se ignorarán los primeros bytes del número.

Ejemplo en ST

```
si := INT_TO_SINT(4223); (* Result is 127 *)
```

Si guarda el entero 4223 (16#107f representado como hexadecimal) como una variable SINT, aparecerá como 127 (16#7f representando como hexadecimal).

Ejemplo en IL

```
LD                  4223
INT_TO_SINT
ST                  si
```

Ejemplo en FBD

Conversiones REAL_TO / LREAL_TO

Definición

El operador IEC para conversiones desde el tipo de variable REAL o LREAL hasta otro tipo de variable.

El valor se redondeará al alza o a la baja al número entero más próximo y se convertirá al nuevo tipo de variable.

Los siguientes tipos de variables son excepciones a esta norma:

- STRING
- BOOL
- REAL
- LREAL

Resultados de la conversión

Si un REAL o LREAL se convierte en SINT, USINT, INT, UINT, DINT, UDINT, LINT o ULINT y el valor del número real no se encuentra dentro del rango de valores de ese entero, el resultado será indefinido y puede conducir a una excepción del controlador.

NOTA: Valide cualquier desborde del rango mediante la aplicación y compruebe que el valor del REAL o LREAL se encuentra dentro de los límites del entero de destino antes de realizar la conversión.

Al realizar la conversión al tipo STRING, tenga en cuenta que el número total de dígitos se limita a 16. Si el número (L)REAL tiene más dígitos, se redondeará el decimosexto. Si la longitud de la STRING se define como demasiado corta, se cortará desde el extremo derecho.

Ejemplo en ST

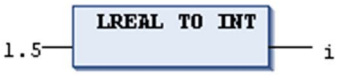
Ejemplos en ST con resultados de la conversión:

Ejemplo	Resultado
`i := REAL_TO_INT(1.5);`	2
`j := REAL_TO_INT(1.4);`	1
`i := REAL_TO_INT(-1.5);`	-2
`j := REAL_TO_INT(-1.4);`	-1

Ejemplo en IL

```
LD                  2.75
REAL_TO_INT
ST                  i
```

Ejemplo en FBD



Conversiones TIME_TO/TIME_OF_DAY

Definición

Operador IEC para conversiones del tipo de variable TIME o TIME_OF_DAY a un tipo diferente.

Sintaxis

TIME_TO_<tipo de datos>

TOD_TO_<tipo de datos>

Resultados de la conversión

La hora se almacenará internamente en un DWORD en milisegundos (empezando por las 12:00 A.M. para la variable TIME_OF_DAY). Este valor se convertirá.

En caso de tipo STRING el resultado es una constante de tiempo.

Ejemplos en ST

Ejemplos en ST con resultados de la conversión:

Ejemplo	Resultado
`str := TIME_TO_STRING(T#12ms);`	`T#12ms`
`dw := TIME_TO_DWORD(T#5m);`	`300000`
`si := TOD_TO_SINT(TOD#00:00:00.012);`	`12`

Ejemplos en IL

Ejemplos en IL con resultados de la conversión:

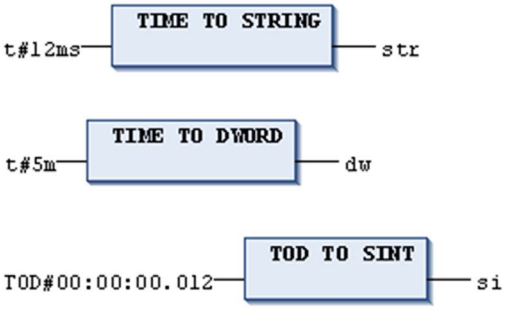
Ejemplo	Resultado

`LD T#12ms`
`TIME_TO_STRI...`
`ST str`
	`T#12ms`

`LD T#300000ms`
`TIME_TO_DWORD`
`ST dw`
	`300000`

`LD TOD#00:00:00.012`
`TIME_TO_SINT`
`ST si`
	`12`

Ejemplos en FBD



Conversiones DATE_TO/DT_TO

Definición

Operador IEC para conversiones del tipo de variable DATE o DATE_AND_TIME a otro tipo.

Sintaxis

DATE_TO_<tipo de fecha>

DT_TO_<tipo de fecha>

Resultados de la conversión

La fecha se almacenará internamente en una DWORD en segundos desde el 1 de enero de 1970. Este valor se convertirá.

Para variables de tipo STRING, el resultado es la constante de fecha.

Ejemplos en ST

Ejemplos en ST con resultados de la conversión:

Ejemplo	Resultado
`b := DATE_TO_BOOL(D#1970-01-01);`	`FALSE`
`i := DATE_TO_INT(D#1970-01-15);`	`29952`
`byt := DT_TO_BYTE(DT#1970-01-15-05:05:05);`	`129`
`str := DT_TO_STRING(DT#1998-02-13-14:20);`	`DT#1998-02-13-14:20`

Ejemplos en IL

Ejemplos en IL con resultados de la conversión:

Ejemplo	Resultado

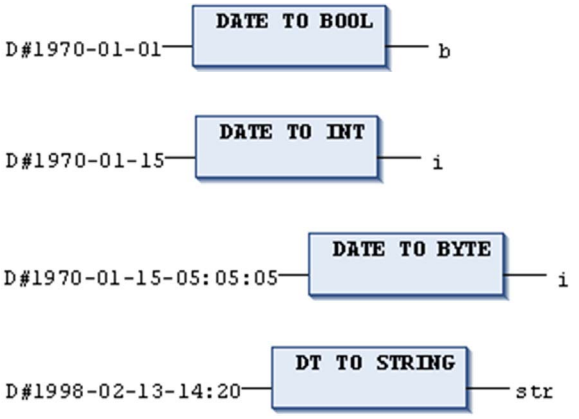
`LD D#1970-01-01`
`DATE_TO_BOOL`
`ST b`
	`FALSE`

`LD D#1970-01-01`
`DATE_TO_INT`
`ST i`
	`29952`

`LD D#1970-01-15-05:05:`
`DATE_TO_BYTE`
`ST byt`
	`129`

`LD D#1998-02-13-14:20`
`DATE_TO_STRI...`
`ST str`
	`'DT#1998-02-13-14:20'`

Ejemplos en FBD



Conversiones STRING_TO

Definición

Operador IEC para conversiones del tipo de variable STRING a otro tipo distinto.

La conversión funciona conforme al mecanismo de compilación C estándar:

STRING a INT y INT a BYTE. El byte más alto se cortará; por tanto, solo habrá valores entre 0 y 255.

Sintaxis

STRING_TO_<tipo de datos>

Resultados de la conversión

El operando de la variable de tipo STRING debe contener un valor válido en el tipo de la variable de destino. En caso contrario, el resultado será 0.

Ejemplos en ST

Ejemplos en ST con resultados de conversión:

Ejemplo	Resultado
`b := STRING_TO_BOOL('TRUE');`	`TRUE`
`w := STRING_TO_WORD('abc34');`	`0`
`t := STRING_TO_TIME('T#127ms');`	`T#127ms`
`bv := STRING_TO_BYTE('500');`	`244` Explicación: 500 = 0x1F4; 0xF4=244

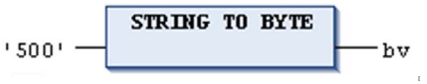
Ejemplos en IL

Ejemplos en IL con resultados de la conversión:

Ejemplo	Resultado
LD 'TRUE' STRING_TO_BOOL ST b	TRUE
LD 'abc34' STRING_TO_WORD ST w	0
LD 't#117ms' STRING_TO_TIME ST t	T#117ms
LD '500' STRING_TO_BYTE... ST bv	244

Ejemplos en FBD

Ejemplos en FBD con resultados de conversión:

Ejemplo	Resultado
	TRUE
	0
	T#127ms
	244

TRUNC

Definición

Operador IEC para conversiones de REAL a DINT. Se utilizará toda la parte del número del valor.

Ejemplos en ST

Ejemplos en ST con resultados de la conversión:

Ejemplo	Resultado
`diVar:=TRUNC(1.9);`	`1`
`diVar:=TRUNC(-1.4);`	`-1`

Ejemplo en IL

```
LD                  1.9
TRUNC
ST                  diVar
```

TRUNC_INT

Definición

Operador IEC para conversiones de REAL a INT. Se utilizará toda la parte del número del valor.

Ejemplos en ST

Ejemplos en ST con resultados de la conversión:

Ejemplo	Resultado
`iVar:=TRUNC_INT(1.9);`	`1`
`iVar:=TRUNC_INT(-1.4);`	`-1`

Ejemplo en IL

```
LD                  1.9
TRUNC_INT
ST                  iVar
```

Conversiones ANY_..._TO

Definición

Conversión a partir de cualquier tipo de datos o, más concretamente, de cualquier tipo de datos numéricos a otro tipo de datos. Al igual que con cualquier tipo de conversión, se debe tener en cuenta el tamaño de los operandos para que la conversión sea correcta.

Sintaxis

ANY_NUM_TO_<tipo de datos numéricos>

ANY_TO_<cualquier tipo de datos>

Ejemplo

Conversión de una variable de tipo de datos REAL a INT:

```
re : REAL := 1.234;
i : INT := ANY_TO_INT(re)
```

Sección 30.9
Funciones numéricas

Descripción general

En este capítulo se describen los operadores IEC numéricos disponibles especificados en la norma IEC 61131-3.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

ABS

Definición

Operador IEC numérico para devolver el valor absoluto de un número.

Entrada y salida pueden ser cualquier tipo de dato básico numérico.

Ejemplo en IL

El resultado en `i` es 2.

```
LD                  -2
ABS
ST                  i
```

Ejemplo en ST

```
i:=ABS(-2);
```

Ejemplo en FBD

SQRT

Definición

Operador IEC numérico para obtener la raíz cuadrada de un número.

La variable de entrada puede ser cualquier tipo de dato básico numérico, la variable salida debe ser el tipo REAL o LREAL.

Ejemplo en IL

El resultado en `q` es 4.

```
LD                  16
SQRT
ST                  q
```

Ejemplo en ST

```
q:=SQRT(16);
```

Ejemplo en FBD

LN

Definición

Operador IEC numérico para obtener el logaritmo natural de un número.

La variable de entrada puede ser cualquier tipo de dato básico numérico, la variable de salida debe ser el tipo REAL o LREAL.

Ejemplo en IL

El resultado en `q` es 3,80666.

```
LD                  45
LN
ST                  q
```

Ejemplo en ST

```
q:=LN(45);
```

Ejemplo en FBD

LOG

Definición

Operador IEC numérico para obtener el logaritmo de un número de base 10.

La variable de entrada puede ser cualquier tipo de dato básico numérico, la variable de salida debe ser el tipo REAL o LREAL.

Ejemplo en IL

El resultado en `q` es 2,49762.

```
LD                  314.5
LOG
ST                  q
```

Ejemplo en ST

```
q:=LOG(314.5);
```

Ejemplo en FBD

EXP

Definición

Operador IEC numérico para devolver la función exponencial.

La variable de entrada puede ser cualquier tipo de dato básico numérico, la variable de salida debe ser el tipo REAL o LREAL.

Ejemplo en IL

El resultado en `q` es 7,389056099.

```
LD                  2
EXP
ST                  q
```

Ejemplo en ST

```
q:=EXP(2);
```

Ejemplo en FBD

SIN

Definición

Operador IEC numérico para obtener el seno de un ángulo.

La entrada que define el ángulo en minutos de arco puede ser de cualquier tipo de dato básico numérico, la variable de salida debe ser de tipo REAL o LREAL.

Ejemplo en IL

El resultado en `q` es 0,479426.

```
LD                  0.5
SIN
ST                  q
```

Ejemplo en ST

```
q:=SIN(0.5);
```

Ejemplo en FBD

COS

Definición

Operador IEC numérico para obtener el coseno de un ángulo.

La entrada que define el ángulo en minutos de arco puede ser de cualquier tipo de dato básico numérico cuando la variable de salida debe ser de tipo REAL o LREAL.

Ejemplo en IL

El resultado en `q` es 0,877583.

```
LD                  0.5
COS
ST                  q
```

Ejemplo en ST

```
q:=COS(0.5);
```

Ejemplo en FBD

TAN

Definición

Operador IEC numérico para obtener la tangente de un número. El valor se calcula en minutos de arco.

La variable de entrada puede ser cualquier tipo de dato básico numérico cuando la variable de salida debe ser el tipo REAL o LREAL.

Ejemplo en IL

El resultado en `q` es 0,546302.

```
LD                  0.5
TAN
ST                  q
```

Ejemplo en ST

```
q:=TAN(0.5);
```

Ejemplo en FBD

ASIN

Definición

Operador IEC numérico para obtener el arcoseno (función inversa del seno) de un número. El valor se calcula en minutos de arco.

La variable de entrada puede ser cualquier tipo de dato básico numérico cuando la variable de salida debe ser el tipo REAL o LREAL.

Ejemplo en IL

El resultado en `q` es 0,523599.

```
LD                  0.5
ASIN
ST                  q
```

Ejemplo en ST

```
q:=ASIN(0.5);
```

Ejemplo en FBD

ACOS

Definición

Operador IEC numérico para obtener el arcoseno (función inversa del coseno) de un número. El valor se calcula en minutos de arco.

La variable de entrada puede ser cualquier tipo de dato básico numérico cuando la variable de salida debe ser el tipo REAL o LREAL.

Ejemplo en IL

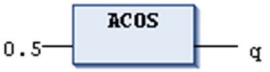
El resultado en `q` es 1,0472.

```
LD                  0.5
ACOS
ST                  q
```

Ejemplo en ST

```
q:=ACOS(0.5);
```

Ejemplo en FBD



ATAN

Definición

Operador IEC numérico para obtener la arcotangente (función inversa de la tangente) de un número. El valor se calcula en minutos de arco.

La variable de entrada puede ser cualquier tipo de dato básico numérico cuando la variable de salida debe ser el tipo REAL o LREAL.

Ejemplo en IL

El resultado en `q` es 0,463648.

```
LD                  0.5
ATAN
ST                  q
```

Ejemplo en ST

```
q:=ATAN(0.5);
```

Ejemplo en FBD

EXPT

Definición

Operador IEC numérico para la exponenciación de una variable con otra variable:

`OUT` = `IN1` a la `IN2`

La variable de entrada puede ser cualquier tipo de dato básico numérico cuando la variable de salida debe ser el tipo REAL o LREAL.

Ejemplo en IL

El resultado es 49.

```
LD              7
EXPT            2
ST              Var1
```

Ejemplo en ST

```
var1:=EXPT(7,2);
```

Ejemplo en FBD

Sección 30.10
Operadores de ampliación de IEC

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Operadores de ampliación de IEC

Descripción general

Además de los operadores de IEC, SoMachine admite los siguientes operadores de ampliación de IEC:

- ADR *(véase página 745)*
- BITADR *(véase página 747)*
- SIZEOF *(véase página 717)*
- __DELETE *(véase página 783)*
- __ISVALIDREF *(véase página 785)*
- __NEW *(véase página 786)*
- __QUERYINTERFACE *(véase página 790)*
- __QUERYPOINTER *(véase página 792)*
- operadores de ámbito *(véase página 794)*

__DELETE

Definición

Este operador no se especifica en la norma IEC 61131-3.

NOTA: Por motivos de compatibilidad, la versión del compilador debe ser ≥3.3.2.0.

Para obtener más información, consulte la tabla de asignaciones de la versión del compilador de SoMachine/CoDeSys en la Compatibilidad y migración - Guía del usuario *(véase SoMachine - Compatibilidad y migración, Guía del usuario)*.

El operador `__DELETE` anula la memoria de objetos asignados antes a través del operador *(véase página 786)* `__NEW`.

`__DELETE` no tiene un valor de retorno y su operando se establecerá en 0 tras la operación.

Active la opción **Utilizar asignación dinámica de memoria** en la vista **Opciones de creación de aplicaciones** (**Ver** →**Propiedades...** →**Opciones de creación de aplicaciones**).

Sintaxis

__DELETE (<puntero>)

Si el puntero apunta a un bloque de funciones, se llamará al método especializado `FB_Exit` ante de que el puntero se defina en `NULL`.

Ejemplo

En el ejemplo siguiente, el bloque de funciones `FBDynamic` se asigna de forma dinámica a través de `_NEW` desde el POU `PLC_PRG`. De este modo, se llamará al método `FB_Init`, en el cual se asigna un tipo de `DUT`. Cuando se llama a `__DELETE`, el puntero del bloque de funciones de `PLC_PRG`, se llamará a `FB_Exit`, que a su vez libera el tipo interno asignado.

```
FUNCTION_BLOCK FBDynamic
VAR_INPUT
in1, in2 : INT;
END_VAR
VAR_OUTPUT
out : INT;
END_VAR
VAR
test1 : INT := 1234;
_inc : INT := 0;
_dut : POINTER TO DUT;
neu : BOOL;
END_VAR
out := in1 + in2;
```

```
METHOD FB_Exit : BOOL
VAR_INPUT
bInCopyCode : BOOL;
END_VAR
__Delete(_dut);


METHOD FB_Init : BOOL
VAR_INPUT
bInitRetains : BOOL;
bInCopyCode : BOOL;
END_VAR
_dut := __NEW(DUT);


METHOD INC : INT
VAR_INPUT
END_VAR
_inc := _inc + 1;
INC := _inc;


PLC_PRG(PRG)
VAR
pFB : POINTER TO FBDynamic;
bInit: BOOL := TRUE;
bDelete: BOOL;
loc : INT;
END_VAR
IF (bInit) THEN
pFB := __NEW(FBDynamic);
bInit := FALSE;
END_IF
IF (pFB <> 0) THEN
pFB^(in1 := 1, in2 := loc, out => loc);
pFB^.INC();
END_IF
IF (bDelete) THEN
__DELETE(pFB);
END_IF
```

__ISVALIDREF

Definición

Este operador no se especifica en la norma IEC 61131-3.

Permite comprobar si una referencia apunta a un valor válido.

Para obtener información sobre cómo utilizar y ver un ejemplo, consulte la descripción de los tipos de datos de referencia *(véase página 671)*.

__NEW

Definición

Este operador no está preestablecido por el estándar IEC 61131-3.

NOTA: Por motivos de compatibilidad, la versión del compilador debe ser ≥3.3.2.0.

Para obtener más información, consulte la tabla de asignaciones de la versión del compilador SoMachine/CoDeSys en la guía Compatibilidad y migración - Guía del usuario *(véase SoMachine - Compatibilidad y migración, Guía del usuario)*.

El operador `__NEW` asigna memoria para las instancias de los bloques de funciones o para matrices de tipos de datos estándar. El operador devuelve un puntero adecuadamente escrito al objeto. Si el operador no se utiliza en una asignación aparecerá un mensaje.

Active la opción **Utilizar asignación dinámica de memoria** en la vista **Opciones de creación de aplicaciones** (**Ver** →**Propiedades...** →**Opciones de creación de aplicaciones**) para utilizar el operador `__NEW`.

Si no puede asignar memoria,`__NEW` dará un retorno 0.

Utilice `__DELETE` si quiere anular una asignación.

Sintaxis

__NEW (<type>, [<size>]

El operador crea un nuevo objeto del tipo especificado `<type>` y retornará un puntero a dicho `<type>`. Se llama a la inicialización del objeto después de su creación. Si el retorno es 0 la operación no se habrá completado satisfactoriamente.

Si `<type>` es escalar, el operando opcional `<length>` deberá establecerse adicionalmente y el operador creará una matriz de tipo escalar con esa longitud.

Ejemplo

```
pScalarType := __New(ScalarType, length);
```

NOTA: No es posible una copia del código en el cambio en línea de los objetos creados de forma dinámica.

Por lo tanto, únicamente los bloques de funciones fuera de bibliotecas (debido a que no pueden cambiar) y los bloques de funciones con el atributo `enable_dynamic_creation` están permitidos para el operador `__NEW`. Si un bloque de funciones cambia con este indicador de manera que se necesite hacer una copia del código, aparecerá un mensaje.

NOTA: El código para la asignación de memoria no puede ser reentrante.

Se utiliza un semáforo (`SysSemEnter`) para evitar que dos tareas traten de asignar memoria al mismo tiempo. En consecuencia, un uso excesivo de `__New` puede producir una mayor fluctuación.

Ejemplo con un tipo escalar:

```
TYPE DUT :
STRUCT
a,b,c,d,e,f : INT;
END_STRUCT
END_TYPE
```

```
PROGRAM PLC_PRG
VAR
pDut : POINTER TO DUT;
bInit: BOOL := TRUE;
bDelete: BOOL;
END_VAR
```

```
IF (bInit) THEN
pDut := __NEW(DUT);
bInit := FALSE;
END_IF
IF (bDelete) THEN
__DELETE(pDut);
END_IF
```

Ejemplo con un bloque de funciones:

```
FBDynamic(FP)
{attribute 'enable_dynamic_creation'}
FUNCTION_BLOCK FBDynamic
VAR_INPUT
in1, in2 : INT;
END_VAR
VAR_OUTPUT
out : INT;
END_VAR
VAR
test1 : INT := 1234;
_inc : INT := 0;
_dut : POINTER TO DUT;
neu : BOOL;
END_VAR
out := in1 + in2;
```

```
PLC_PRG(PRG)
VAR
pFB : POINTER TO FBDynamic;
loc : INT;
bInit: BOOL := TRUE;
bDelete: BOOL;
END_VAR
IF (pFB <> 0) THEN
pFB^(in1 := 1, in2 := loc, out => loc);
pFB^.INC();
END_IF
IF (bDelete) THEN
__DELETE(pFB);
END_IF
```

Ejemplo con una matriz:

```
PLC_PRG(PRG)
VAR
bInit: BOOL := TRUE;
bDelete: BOOL;
pArrayBytes : POINTER TO BYTE;
pArrayDuts : POINTER TO BYTE;
test: INT;
parr : POINTER TO BYTE;
END_VAR
IF (bInit) THEN
pArrayBytes := __NEW(BYTE, 25);
```

```
bInit := FALSE;
END_IF
IF (pArrayBytes <> 0) THEN
pArrayBytes[24] := 125;
test := pArrayBytes[24];
END_IF
IF (bDelete) THEN
__DELETE(pArrayBytes);
END_IF
```

__QUERYINTERFACE

Definición

Este operador no se prescribe mediante la norma IEC 61131-3.

Durante el tiempo de ejecución, `__QUERYINTERFACE` habilita una conversión de tipo de una referencia de interfaz a otra. El operador devuelve un resultado con tipo BOOL. TRUE implica que la conversión se ha ejecutado correctamente.

NOTA: Por motivos de compatibilidad, la definición de la referencia que se debe convertir debe ser una extensión de la interfaz base `__SYSTEM.IQueryInterface` y la versión del compilador debe ser ≥ 3.3.0.20.

Para obtener más información, consulte la tabla de asignaciones de la versión del compilador de SoMachine/CoDeSys en la Compatibilidad y migración - Guía del usuario *(véase SoMachine - Compatibilidad y migración, Guía del usuario)*.

Sintaxis

__QUERYINTERFACE(<ITF_Source>, < ITF_Dest>

El operador requiere una referencia de interfaz o una instancia del bloque de funciones del tipo previsto como primer operando y una referencia de interfaz como segundo operando. Tras la ejecución de `__QUERYINTERFACE`, el `ITF_Dest` contiene una referencia para la interfaz prevista si el objeto referenciado del código de origen `ITF` implementa la interfaz. En este caso, la conversión es correcta y el resultado del operador devuelve TRUE. En el resto de casos, el operador devuelve FALSE.

Un requisito para una conversión explícita es que no solo `ITF_Source`, sino también `ITF Dest`, sean una extensión de la interfaz `__System.IQueryInterface`. Esta interfaz se proporciona de forma implícita y no requiere una biblioteca.

Ejemplo

Ejemplo en ST:

```
INTERFACE ItfBase EXTENDS __System.IQueryInterface
METHOD mbase : BOOL
END_METHOD
INTERFACE ItfDerived1 EXTENDS ItfBase
METHOD mderived1 : BOOL
END_METHOD
INTERFACE ItfDerived2 EXTENDS ItfBase
METHOD mderived2 : BOOL
END_METHOD
PROGRAMM POU
VAR
itfderived1 : ItfDerived1;
itfderived2 : ItfDerived2;
bTest1, bTest2, xResult1, xResult2: BOOL;
END_VAR
xResult1 := __QUERYINTERFACE(itfbase, itfderived1); // variablen vom Ty
pe ItfBase bzw ItfDerived1
xResult2 := __QUERYINTERFACE(itfbase, itfderived2); // variablen vom Ty
pe ItfBase bzw ItfDerived2
IF (xResult1 = TRUE) THEN
bTest1 := itfderived1.mderived1();
ELSIF xResult2 THEN
bTest2 := itfderived2.mderived2();
END_IF
```

__QUERYPOINTER

Definición

Este operador no se especifica en la norma IEC 61131-3.

Durante el tiempo de ejecución, `__QUERYPOINTER` se asigna a una referencia de interfaz en un puntero sin tipo. El operador devuelve un resultado con tipo BOOL. TRUE implica que la conversión se ha ejecutado correctamente.

NOTA: Por motivos de compatibilidad, la definición de la referencia de la interfaz prevista debe ser una extensión de la interfaz base `__SYSTEM.IQueryInterface` y la versión del compilador debe ser ≥ 3.3.0.20.

Para obtener más información, consulte la tabla de asignaciones de la versión del compilador de SoMachine/CoDeSys en la Compatibilidad y migración - Guía del usuario *(véase SoMachine - Compatibilidad y migración, Guía del usuario)*.

Sintaxis

__QUERYPOINTER (<ITF_Source>, < Pointer_Dest>

Para el primer operando, el operador requiere una referencia de interfaz o una instancia del bloque de funciones del tipo previsto y para el segundo operando, un puntero sin tipo. Tras la ejecución de `__QUERYPOINTER`, el `Pointer_Dest` contiene la dirección de la referencia en la interfaz prevista. En este caso, la conversión es correcta y el resultado del operador devuelve TRUE. En el resto de casos, el operador devuelve FALSE. `Pointer_Dest` es sin tipo y puede emitirse en cualquier tipo. El programador debe garantizar el tipo real. Por ejemplo, la interfaz podría proporcionar un método que devuelva un código de tipo.

Un requisito para una conversión explícita es que el `ITF_Source` es una extensión de la interfaz `__System.IQueryInterface`. Esta interfaz se proporciona de forma implícita y no requiere una biblioteca.

Ejemplo

```
INTERFACE Itf EXTENDS __System.IQueryInterface
FUNCTION_BLOCK FBVariant IMPLEMENTS ITF
METHOD m1 : BOOL
VAR
END_VAR
m1 := TRUE;
END_METHOD
```

```
PROGRAMM POU
VAR
  itf1 : Itf;
  insV : FBVariant;
  xResult, xTest : BOOL;
  pVar: POINTER TO FBVariant;
END_VAR
itf1 := insV;
xResult := __QUERYPOINTER(itf1, pVar);
IF xResult THEN
  xTest := pVar.m1();
END_IF
```

Operadores de ámbito

Definición

Como extensión a los operadores de IEC, existen varias posibilidades para evitar la ambigüedad al acceder a las variables o módulos si el nombre de la variable o módulo se utiliza múltiples veces dentro del ámbito de un proyecto.

Los siguientes operadores de ámbito pueden utilizarse para definir el espacio de nombre correspondiente:

- operador de ámbito global
- nombre de lista de variables globales
- nombre de enumeración
- espacio de nombres de la biblioteca

Operador de ámbito global

Una ruta de instancia que comienza con un punto (.) abre un ámbito global (espacio de nombre) Así pues, si existe una variable local con el mismo nombre `<varname>` que una variable global `.<varname>` se refiere a la variable global.

Nombre de lista de variables globales

Puede utilizar el nombre de una lista de variables globales como espacio de nombre para las variables comprendidas en dicha lista. Así, es posible declarar variables con nombres idénticos en diferentes listas de variables globales y, precediendo el nombre de la variable por `<global variable list name>.`, es posible acceder a la deseada.

Sintaxis

<global variable list name>.<variable>

Ejemplo

La lista de variables globales `globlist1` y `globlist2` contienen, cada una, una variable nombrada `varx`. En la línea siguiente, `varx` de `globlist2` es copiada a `varx` en `globlist1`:

```
globlist1.varx := globlist2.varx;
```

Si un nombre de variable declarado en más de una lista de variables global es referenciado sin que el nombre de lista de variables globales preceda al operador, aparecerá un mensaje.

Espacio de nombres de la biblioteca

Puede utilizar el espacio de nombres de la biblioteca para acceder a los componentes de la biblioteca de manera explícita.

Ejemplo

Si una biblioteca incluida en un proyecto contiene un módulo `fun1` y hay también un POU `fun1` definido localmente en el proyecto, puede añadir el `namespace` de la biblioteca al nombre del módulo para hacer el acceso único.

Sintaxis

`<namespace>.<module name>`, por ejemplo `lib1.fun1`.

Por defecto, el `namespace` de una biblioteca es idéntico al nombre de la biblioteca. Sin embargo, puede definir cualquier otro tanto en **Información del proyecto** al crear un proyecto de biblioteca en **Información del proyecto** (por defecto en el menú **Proyecto**), como más tarde en el cuadro de diálogo **Propiedades** de una librería incluida en el **Administrador de bibliotecas**.

Ejemplo

Hay una función `fun1` en la biblioteca `lib`. También hay una función `fun1` declarada en el proyecto. De forma predeterminada, el nombre de espacio de la biblioteca `lib` es nombrado `'lib'`:

```
res1 := fun(in := 12); // call of the project function fun
res2 := lib.fun(in := 12); // call of the library function fun
```

Nombre de enumeración

Puede utilizar la denominación de tipo de una enumeración para evitar la ambigüedad al acceder a una constante de enumeración. De este modo, es posible utilizar la misma constante en enumeraciones diferentes.

El nombre de enumeración debe preceder al nombre de la constante, y estar separado de este por un punto (.).

Sintaxis

<nombre enumeración>.<nombre constante>

Ejemplo

La constante `Blue` es un componente de la enumeración `Colors` al mismo tiempo que de la enumeración `Feelings`.

```
color := Colors.Blue; // Access to enum value Blue in type Colors
feeling := Feelings.Blue; // Access to enum value Blue in type Feelings
```

Sección 30.11
Operador de inicialización

Operador INI

Descripción general

NOTA: El operador `INI` está obsoleto. El método `FB_init` sustituye al operador `INI`. Para obtener más información sobre el método `FB_init`, consulte el capítulo Métodos FB_init, FB_reinit *(véase página 603)*. Sin embargo, el operador aún se puede utilizar para mantener la compatibilidad con los proyectos importados de versiones de SoMachine anteriores.

Puede utilizar el operador `INI` para inicializar las variables de retención que proporciona una instancia del bloque de funciones usada en el POU.

Asigne el operador a una variable booleana.

Sintaxis

<bool-variable> := INI(<FB-instance, TRUE|FALSE)

Si el segundo parámetro del operador se define en TRUE, se inicializarán todas las variables de retención definidas en el bloque de funciones `FB`.

Ejemplo en ST

`fbinst` es la instancia del bloque de funciones `fb`, en la que se define una variable de retención `retvar`.

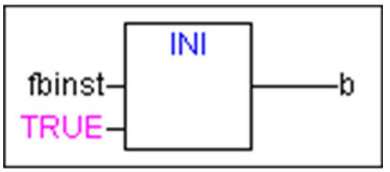
Declaración en POU

```
fbinst:fb;
b:bool;
```

Parte de implementación

```
b := INI(fbinst, TRUE);
ivar:=fbinst.retvar (* => retvar gets initialized *)
```

Ejemplo de llamada del operador en FBD



Capítulo 31
Operandos

Contenido de este capítulo

Este capítulo contiene las siguientes secciones:

Sección 31.1
Constantes

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Constantes BOOL

Descripción general

Las constantes BOOL son los valores lógicos TRUE y FALSE.

Consulte la descripción de los tipos de datos BOOL *(véase página 663)*.

Constantes TIME

Descripción general

Las constantes TIME se emplean para hacer funcionar los módulos del temporizador estándar. La constante de tiempo TIME tiene un tamaño de 32 bits y se ajusta a la norma IEC 61131-3. Además, LTIME se admite como ampliación a la norma como base de tiempo para temporizadores de alta resolución. LTIME tiene un tamaño de 64 bits y una resolución en nanosegundos.

Sintaxis para la constante TIME

t#<declaración de tiempo>

En vez de `t#`, también puede usar los siguientes:

- `T#`
- `time`
- `TIME`

La declaración de tiempo puede incluir las siguientes unidades de tiempo. Deben utilizarse en la siguiente secuencia, pero no es obligatorio usarlas todas.

- d: días
- h: horas
- m: minutos
- s: segundos
- ms: milisegundos

Ejemplos de constantes TIME correctas en una asignación de ST

Ejemplo	Descripción
`TIME1 := T#14ms;`	–
`TIME1 := T#100S12ms;`	(* Es posible que el componente más alto pueda superar su límite *)
`TIME1 := t#12h34m15s;`	–

Ejemplos de uso incorrecto

Ejemplo	Descripción
`TIME1 := t#5m68s;`	(* límite superado en un componente inferior *)
`TIME1 := 15ms;`	(* falta T# *)
`TIME1 := t#4ms13d;`	(* orden de entradas incorrecto *)

Sintaxis para la constante LTIME

LTIME#<declaración de tiempo>

La declaración de tiempo puede incluir las unidades de tiempo tal y como se usan con la constante TIME y además:

- us: microsegundos
- ns: nanosegundos

Ejemplos de constantes LTIME correctas en una asignación de ST:

```
LTIME1 := LTIME#1000d15h23m12s34ms2us44ns
LTIME1 := LTIME#3445343m3424732874823ns
```

Para obtener más información, consulte la descripción de los tipos de datos TIME *(véase página 665)*.

Constantes DATE

Descripción general

Use estas constantes para introducir fechas.

Sintaxis

d#<declaración de fecha>

En vez de `d#`, también puede usar los siguientes:

- `D#`
- `date`
- `DATE`

Introduzca la declaración de fecha en el formato <año-mes-día>.

Los valores DATE se procesan internamente como valores DWORD y contienen el intervalo de tiempos en segundos a partir de 01.01.1970, 00:00.

Ejemplos

```
DATE#1996-05-06
d#1972-03-29
```

Para obtener más información, consulte la descripción de los tipos de datos TIME *(véase página 665)*.

Constantes DATE_AND_TIME

Descripción general

Las constantes DATE y las constantes TIME_OF_DAY también se pueden combinar para formar las denominadas constantes DATE_AND_TIME.

Sintaxis

`dt#`<declaración de fecha y hora>

En lugar de `dt#` puede usar lo siguiente:

- `DT#`
- `date_and_time`
- `DATE_AND_TIME`

Introduzca la declaración de fecha y hora en formato <año-mes-día-hora:minuto:segundo>.

Puede introducir segundos como números reales. Esto permite especificar fracciones de segundo.

Los valores DATE_AND_TIME se gestionan internamente como valores DWORD, que contienen el intervalo de tiempo en segundos desde 01.01.1970, 00:00.

Ejemplos

```
DATE_AND_TIME#1996-05-06-15:36:30
dt#1972-03-29-00:00:00
```

Para obtener más información, consulte la descripción de los tipos de datos TIME *(véase página 665)*.

Constantes TIME_OF_DAY

Descripción general

Use este tipo de constante para almacenar las horas.

Sintaxis

tod#<declaración de tiempo>

En lugar de `tod#`, también puede usar lo siguiente:

- `TOD#`
- `time_of_day#`
- `TIME_OF_DAY#`

Introduzca la declaración de tiempo en el formato <hora:minutos:segundos>.

Puede introducir segundos como números reales. Esto permite especificar fracciones de segundo.

Los valores TIME_OF_DAY se gestionan internamente como valores DWORD, que contienen el intervalo de tiempo en milisegundos desde las 00:00 h.

Ejemplos

```
TIME_OF_DAY#15:36:30.123
tod#00:00:00
```

Para obtener más información, consulte la descripción de los tipos de datos TIME *(véase página 665)*.

Constantes de número

Descripción general

Los valores de número pueden aparecer como números binarios, números octales, números decimales y números hexadecimales. Los valores enteros que no son números decimales se representan con la base seguida de un signo de número (#) delante de la constante de entero. Los valores para los números 10...15 en números hexadecimales se representan con las letras A...F.

Puede incluir el carácter guión bajo dentro del número.

Ejemplos

`14`	(número decimal)
`2#1001_0011`	(número dual)
`8#67`	(número octal)
`16#A`	(número hexadecimal)

Estos valores numéricos pueden ser de tipo:

- BYTE
- WORD
- DWORD
- SINT
- USINT
- INT
- UINT
- DINT
- UDINT
- REAL
- LREAL

No se permiten las conversiones implícitas de tipos de variables mayores o menores. Es decir, una variable DINT no se puede utilizar simplemente como una variable INT. Utilice las funciones de conversiones de tipo *(véase página 750)*.

Constantes REAL/LREAL

Descripción general

Las constantes REAL y LREAL se pueden proporcionar como fracciones decimales y representar de forma exponencial. Para ello, utilice el formato estándar inglés con el punto decimal.

Ejemplos

`7.4`	en lugar de `7,4`
`1.64e+009`	en lugar de `1,64e+009`

Constantes STRING

Descripción general

Una cadena es una secuencia arbitraria de caracteres. Las constantes STRING *(véase página 665)* van precedidas y seguidas de comillas simples. También puede introducir espacios en blanco y caracteres especiales (caracteres especiales de distintos idiomas, como acentos o diéresis). Se tratarán igual que el resto de caracteres.

En las cadenas, la combinación del signo del dólar ($) seguido de dos números hexadecimales se interpretará como una representación hexadecimal del código de carácter de 8-bits.

Además, hay ciertas combinaciones de caracteres que empiezan con un signo del dólar y que se interpretan como sigue:

Combinación introducida	Interpretación
$<dos números hexadecimales>	representación hexadecimal del código de carácter de 8-bits
$$	signo del dólar
$'	comillas simples
$L o $l	avance de línea
$N o $n	nueva línea
$P o $p	avance de página
$R o $r	salto de línea
$T o $t	tabulador

Ejemplos

```
'w1Wüß?'
' Abby and Craig '
':-)'
'costs ($$)'
```

Constantes con tipo / Literales tipados

Descripción general

Fundamentalmente, al utilizar constantes IEC se utilizará el menor tipo de datos posible. Una excepción es una constante REAL/LREAL cuando se usa siempre LREAL. Si se debe utilizar otro tipo de datos, utilice literales tipados (constantes con tipo) sin tener que declarar explícitamente las constantes. Para ello, se proporcionará la constante con un prefijo que determine el tipo.

Sintaxis

```
<Type>#<Literal>
```

`<Type>` es el tipo de datos deseado. Las entradas posibles son:

- BOOL
- SINT
- USINT
- BYTE
- INT
- UINT
- WORD
- DINT
- UDINT
- DWORD
- REAL
- LREAL

Escriba el tipo en mayúsculas.

`<Literal>` especifica la constante. Los datos introducidos deben concordar dentro del tipo de datos especificado en `<Type>`.

Ejemplo

```
var1:=DINT#34;
```

Si la constante no se puede convertir al tipo de destino sin perder información, se generará un mensaje.

Puede usar los literales tipados siempre que se puedan usar constantes normales.

Sección 31.2
Variables

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Variables

Descripción general

Puede declarar las variables localmente en la parte de declaración de un POU, en una lista de variables globales (GVL), en una lista de variables persistentes o en la asignación de E/S de dispositivos.

Consulte el capítulo *Declaración de variables* *(véase página 573)* para obtener más información sobre la declaración de una variable, incluidas las normas relacionadas con el identificador de la variable y el uso múltiple.

El tipo de datos *(véase página 661)* determina cuándo se puede usar una variable.

Puede acceder a las variables disponibles a través de la **Accesibilidad**.

Acceso a variable para matrices, estructuras y POU

En la tabla se enumera la sintaxis correspondiente para acceder a matrices, estructuras y POU:

Sintaxis	Acceso a
<nombre de matriz>[Index1, Index2]	componentes de la matriz *(véase página 678)* de dos dimensiones
<nombre de estructura>.<nombre de variable>	Variables de estructura *(véase página 681)*
<nombre del bloque de funciones>.<nombre de variable>	variables de programa y bloque de funciones

Direccionamiento de bits en variables

Descripción general

En las variables enteras, se puede acceder a bits individuales. Para ello, agregue el índice del bit para direccionarlo a la variable y sepárelo con un punto. Puede proporcionar cualquier constante al índice de bits. La indexación e basa en 0.

Sintaxis

<nombre de variable>.<índice de bit>

Ejemplo

```
a : INT;
b : BOOL;
...
a.2 := b;
```

El tercer bit de la variable `a` se definirá en el valor de la variable `b`, esto significa que la variable `a` será igual a 3.

Si el índice es mayor que el ancho del bit de la variable, se generará el siguiente mensaje:

'Índice '<n>' fuera del rango válido para la variable '<var>'!'

El direccionamiento de bits es posible con las variables de los tipos de datos siguientes:

- SINT
- INT
- DINT
- USINT
- UINT
- UDINT
- BYTE
- WORD
- DWORD

Si el tipo de datos no permite el acceso de bit, se generará el siguiente mensaje:

'Tipo de dato no válido '<type>' para indexación directa'.

No asigne el acceso de bit a una variable `VAR_IN_OUT`.

Acceso de bit a través de una constante global

Si ha declarado una constante global que define el índice de bits, podrá usar esta constante para acceder al bit.

Ejemplo de un acceso de bit a través de una constante global y una variable:

1. Declaración de la constante global en una lista de variables globales

La variable `enable` define el bit al que se accede:

```
VAR_GLOBAL CONSTANT
    enable:int:=2;
END_VAR
```

2. Acceso de bit en una variable entera

Declaración en POU:

```
VAR
    xxx:int;
END_VAR
```

Acceso de bit en tipos de datos BIT

El tipo de datos BIT es un tipo de datos especial que solo se permite en estructuras. Para obtener más información, consulte Acceso de bit en estructuras *(véase página 682)*.

Ejemplo de acceso de bit en tipos de datos BIT

Declaración de estructura

```
TYPE ControllerData :
STRUCT
    Status_OperationEnabled : BIT;
    Status_SwitchOnActive : BIT;
    Status_EnableOperation : BIT;
    Status_Error : BIT;
    Status_VoltageEnabled : BIT;
    Status_QuickStop : BIT;
    Status_SwitchOnLocked : BIT;
    Status_Warning : BIT;
END_STRUCT
END_TYPE
```

Declaración en POU

```
VAR
    ControllerDrive1:ControllerData;
END_VAR
```

Acceso de bit

```
ControllerDrive1.OperationEnabled := TRUE;
```

Sección 31.3
Direcciones

Dirección

Consideraciones para cambios en línea

La ejecución del comando **Cambio online** puede cambiar el contenido de las direcciones.

⚠ ATENCIÓN

PUNTERO NO VÁLIDO

Cuando utilice punteros en las direcciones y ejecute el comando **Cambio online**, verifique siempre el contenido de los punteros.

El incumplimiento de estas instrucciones puede causar lesiones o daño al equipo.

Descripción general

Al especificar una dirección, el tamaño y la ubicación de memoria se indican mediante secuencias de caracteres especiales.

Sintaxis

%<memory area prefix><size prefix><number|.number|.number....>

Los siguientes prefijos de área de memoria son compatibles:

I	entrada (entradas físicas a través de controladores de entradas, sensores)
Q	salida (salidas físicas a través de controladores de salidas, actuadores)
M	Ubicación de memoria

Los siguientes prefijos de tamaño son compatibles:

X	bit único
Ninguna	bit único
B	byte (8 bits)
W	palabra (16 bits)
D	palabra doble (32 bits)

Ejemplos

Dirección de ejemplo	Descripción
%QX7.5	bit de salida 7,5
%Q7.5	
%IW215	palabra de entrada 215
%QB7	byte de salida 7
%MD48	palabra doble en posición de memoria 48 en la ubicación de memoria
%IW2.5.7.1	la interpretación depende de la configuración del controlador actual (véase a continuación)
ivar AT %IW0: WORD;	ejemplo de una declaración de variable que incluye una asignación de dirección Para obtener más información, consulte el capítulo de declaración *(véase página 586)* AT.

Asignar direcciones válidas

Para asignar una dirección válida en una aplicación, especifique:

- la posición adecuada dentro del proceso de imagen que es el área de memoria que se va a usar: I = input (entrada), Q = output (salida) o M = memory area (área de memoria), tal como se indica en la tabla anterior
- el tamaño deseado: X = bit, B = byte, W = word (palabra), D = dword, tal como se indica en la tabla de ejemplo

La configuración actual de dispositivo y ajustes (hardware, estructura, descripción del dispositivo, configuración de E/S) juega un papel decisivo. Considere especialmente las diferencias entre la interpretación de dirección de bit entre los dispositivos que utilizan la modalidad de direccionamiento de byte y aquellos que utilizan la modalidad de direccionamiento IEC orientada a la palabra. Por ejemplo, en un dispositivo de direccionamiento de byte el primer elemento de la dirección de bit %IX5.5 dará como resultado byte 5. En cambio, un dispositivo de direccionamiento de palabra dará como resultado palabra 5. A diferencia de esto, el direccionamiento de una dirección de palabra o byte es independiente del tipo de dispositivo: con %IW5, siempre se obtendrá como resultado palabra 5 y con la dirección de byte %IB5 siempre byte 5.

Dependiendo de la modalidad de direccionamiento y tamaño se pueden direccionar diferentes celdas de memoria con la misma definición de dirección.

Diferencias entre direccionamiento de byte y direccionamiento de palabra orientada a IEC:

Consulte la tabla a continuación para hacer una comparación entre direccionamiento de byte y direccionamiento de palabra orientada a IEC para bits, bytes, words y dwords. Muestra las áreas de memoria superpuestas en el caso de modalidad de direccionamiento de byte (consulte el ejemplo bajo la tabla).

Respecto a la notación, tenga en cuenta que para direcciones de bit, la modalidad de direccionamiento de `IEC` siempre está orientada a la palabra. Esto quiere decir que el lugar antes del punto corresponde al número de palabra y el lugar detrás del punto nombra el número de bit.

Comparación entre direccionamiento de byte y orientado a la palabra para los tamaños de dirección `D, W, B` y `X`:

DWords / Words (palabras)				**Bytes**	**X (Bits)**					
direccionamiento de byte		direccionamiento IEC orientado a la palabra			direccionamiento de byte			direccionamiento IEC orientado a la palabra		
D0	W0	D0	W0	B0	X0.7	...	X0.0	X0.7	...	X0.0
D1	W1	–	–	B1	X1.7	...	X1.0	X0.15	...	X0.8
...	W2	–	W1	B2	...	–	–	X1.7	...	X1.0
–	W3	–	–	B3	–	–	–	X1.15	...	X1.8
–	W4	D1	W2	B4	–	–	–	–	–	–
–	...	–	–	B5	–	–	–	–	–	–
–	–	–	W3	B6	–	–	–	–	–	–
–	–	–	–	B7	–	–	–	–	–	–
–	–	D2	W4	B8	–	–	–	–	–	–
–	–	...	–	...	–	–	–	–	–	–
–	–	...	–	...	–	–	–	–	–	–
–	–	...	–	...	–	–	–	–	–	–
D(n-3)	–	D(n/4)	...	–	–	–	–	–	–	–
–	–	–	–	–	–	–	–	–	–	–
–	W(n-1)	–	W(n/2)	–	–	–	–	–	–	–
–	–	–	–	Bn	Xn.7	...	Xn.0	X(n/2) .15	...	X(n/2) .8
N = número de byte										

Ejemplo de solapamiento de rangos de memoria en caso de modalidad de direccionamiento de byte:

- `D0` contiene `B0...B3`
- `W0` contiene `B0` y `B1`
- `W1` contiene `B1` y `B2`
- `W2` contiene `B2` y `B3`

Para eludir el solapamiento no utilice `W1` o `D1, D2, D3` para el direccionamiento.

NOTA: Si no se especifica ninguna dirección de bit único explícita, la asignación de los valores booleanos dependerá de los bytes. Ejemplo: Un cambio en el valor de `varbool1 AT %QW0` afecta al rango desde `QX0.0...QX0.7`.

Sección 31.4
Funciones

Funciones

Descripción general

En ST, una llamada a función se puede utilizar como un operando.

Ejemplo

```
Result := Fct(7) + 3;
```

Función TIME()

Esta función devuelve el tiempo (basado en milisegundos) que ha pasado desde que se inició el sistema.

El tipo de datos es TIME.

Ejemplo en IL

```
TIME
ST systime  (* Result for example: T#35m11s342ms *)
```

Ejemplo en ST

```
systime:=TIME();
```

Parte VIII
Plantillas de SoMachine

Contenido de esta parte

Esta parte contiene los siguientes capítulos:

Capítulo 32
Información general sobre las plantillas de SoMachine

Sección 32.1
Plantillas de SoMachine

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Información general sobre las plantillas de SoMachine

Descripción general

SoMachine ofrece plantillas para que las funcionalidades dedicadas de control y visualización, desarrolladas en un proyecto de SoMachine, estén disponibles fácilmente en otros proyectos de SoMachine. Estas plantillas ayudan a estandarizar el uso de dispositivos de campo y funciones de aplicaciones en distintos proyectos de SoMachine.

Están disponibles los tipos de plantillas siguientes:

- plantillas de dispositivos asociadas a un único dispositivo de campo o módulo de E/S
- plantillas de funciones asociadas a una función de aplicación de alto nivel

SoMachine proporciona varias plantillas, pero también puede crear las suyas propias para cualquier funcionalidad que desee que esté disponible para otros proyectos.

Creación de sus propias plantillas

Es preciso realizar los pasos siguientes para todas las plantillas de SoMachine:

Paso	Acción
1	Cree su funcionalidad en un proyecto de SoMachine y pruébela con el hardware apropiado o en la simulación.
2	Guarde la funcionalidad en una biblioteca de plantillas.
3	Abra otro proyecto de SoMachine y seleccione la plantilla desde la biblioteca de plantillas para que la funcionalidad esté disponible en este proyecto.

Notas generales

Cuando utilice plantillas de SoMachine, tenga en cuenta lo siguiente:

- Las plantillas no son específicas del controlador, por lo que pueden estar disponibles para cualquier controlador. Verifique que el controlador al que se añade la plantilla sea capaz de ejecutar la funcionalidad contenida en la plantilla.
- Una vez instalada la plantilla, podrá adaptar libremente los objetos creados a sus requisitos individuales.
- La función de plantillas no admite aplicaciones de Vijeo-Designer; las aplicaciones HMI no se incluyen en las plantillas de SoMachine.
- Es posible instalar una plantilla varias veces en el mismo dispositivo controlador. Para evitar conflictos de nomenclatura al crear los mismos objetos varias veces, se les cambia el nombre automáticamente durante la instalación. Para obtener más información, consulte la sección *Nomenclatura de los objetos* del capítulo *Adición de dispositivos a partir de plantillas* *(véase página 840)*.
- Se deben definir bloques de funciones o tipos de datos definidos por el usuario (DUT) en una biblioteca de bloques de funciones si se van a utilizar en plantillas.

- Las plantillas no admiten el uso de representaciones directas de variables (por ejemplo, `%IX2.0`).
  No obstante, se pueden utilizar representaciones directas con una especificación de dirección incompleta (por ejemplo, `%I*`). Para obtener más información, consulte el capítulo *Configuración de variables - VAR_CONFIG (véase página 600)*.

NOTA: Aunque este formulario de marcadores de posición para direcciones directas está disponible, evite el direccionamiento directo en los programas y use direccionamiento simbólico siempre que sea posible.

SoMachine permite programar las instrucciones mediante un método directo o indirecto del uso de parámetros. El método directo, denominado direccionamiento inmediato, implica el uso de la dirección directa de un parámetro como, por ejemplo, %IWx o %QWx. El método indirecto, denominado direccionamiento simbólico, implica en primer lugar la definición de símbolos para los mismos parámetros y, a continuación, el uso de símbolos asociados con las instrucciones del programa.

Los dos métodos son válidos y aceptables; sin embargo, el direccionamiento simbólico ofrece diferentes ventajas, especialmente al hacer modificaciones posteriores en la configuración. Al configurar las E/S y otros dispositivos para la aplicación, SoMachine asignará automáticamente las direcciones inmediatas. A continuación, al añadir o eliminar la E/S u otros dispositivos de la configuración, SoMachine realizará los cambios en la configuración mediante la reasignación de las direcciones inmediatas. Este proceso cambiará en todo caso las asignaciones de su estado anterior desde el punto en el que comienzan los cambios de la configuración.

Si ya ha creado parte o la totalidad del programa utilizando direcciones inmediatas, necesitará realizar este cambio en todas las instrucciones, bloques de funciones, etc. del programa, modificando todas las direcciones inmediatas que se han reasignado. Sin embargo, si en el programa utiliza símbolos en lugar de direcciones inmediatas, esta acción no será necesaria. Los símbolos se actualizarán automáticamente con las nuevas asociaciones de direcciones inmediatas siempre que estén vinculadas a la dirección en el cuadro de diálogo Asignación de E/S del Editor de dispositivos correspondiente, no solo una declaración "AT" del propio programa.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

Revise y modifique, según sea necesario, las direcciones inmediatas que se utilizan en el programa después de modificar la configuración.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

NOTA: Durante la programación, Schneider Electric recomienda encarecidamente el uso sistemático de símbolos para evitar modificar el programa en gran medida y limitar la posibilidad de que surjan anomalías de programación una vez que se haya modificado la configuración del programa al añadir o eliminar E/S u otros dispositivos.

Módulos de E/S admitidos

Las plantillas de SoMachine pueden incluir los siguientes módulos de E/S:

- TM2
- TM3
- TM5

Buses de campo admitidos

Las plantillas de SoMachine pueden incluir dispositivos de campo vinculados a los siguientes buses de campo:

- CANopen
- Modbus línea serie (Modbus IOScanner)
- Modbus TCP IO Scanner
- colección de unidades generales SoftMotion (LMC058)
- CANmotion

Administración de plantillas de SoMachine

Descripción general

En los apartados siguientes se ofrece una descripción general de cómo crear nuevas plantillas de dispositivos o funciones, o bien de cómo cambiar las existentes, y guardarlas como archivos para transferirlas a otros PC.

Bibliotecas de plantillas

Las bibliotecas de plantillas contienen la definición de varias plantillas de dispositivos o funciones.

Protección contra escritura

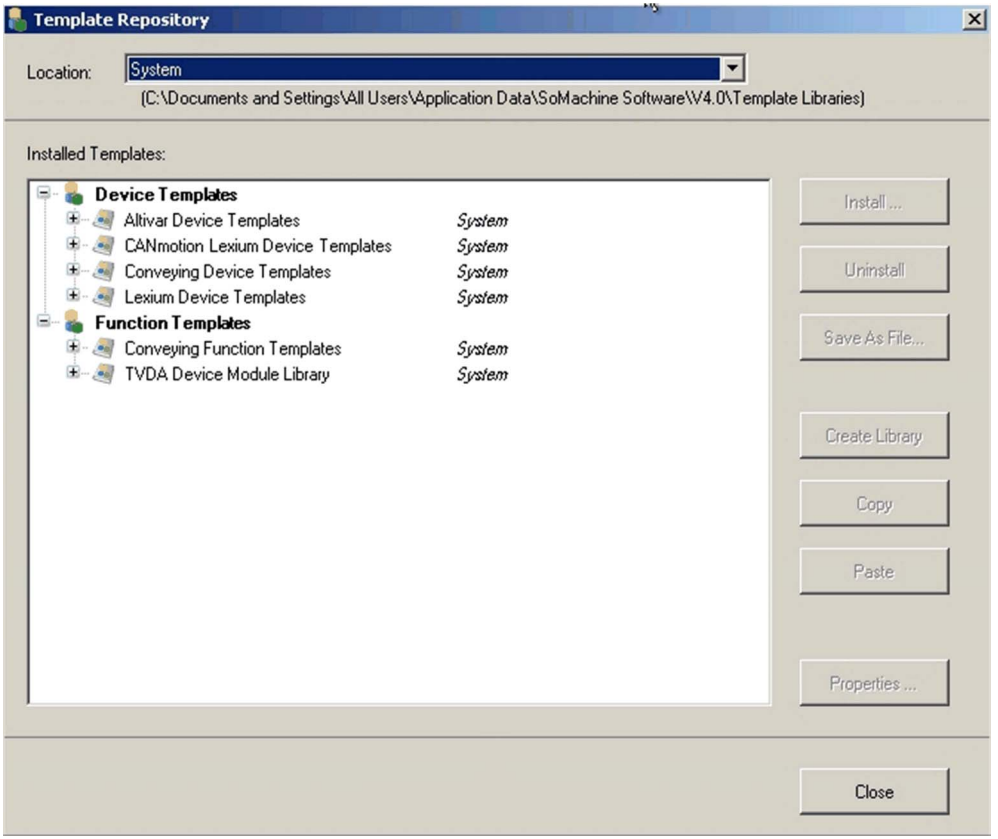
Las bibliotecas de plantillas estándar incluidas en el alcance de suministro de SoMachine están protegidas contra escritura, lo que significa que no se pueden eliminar ni renombrar.

NOTA: No se pueden cambiar las bibliotecas protegidas contra escritura (desinstalar plantillas individuales o cambiar nombres), pero se pueden desinstalar completamente.

Administración de plantillas

Para administrar las plantillas de dispositivos y funciones disponibles en SoMachine, seleccione **Herramientas** →**Repositorio de plantillas** en SoMachine Logic Builder. Para acceder al **Repositorio de plantillas** desde SoMachine Central, abra el cuadro de diálogo **Opciones del sistema** haciendo clic en el botón **Opciones del sistema** en la barra de herramientas o haciendo clic en el botón **Configuración** en la pantalla **Versiones**. En el cuadro de diálogo **Opciones del sistema**, haga clic en el botón *(véase SoMachine Central, Manual del usuario)* **Repositorio de plantillas**.

Se abre el cuadro de diálogo **Repositorio de plantillas**:



En la lista **Ubicación**, seleccione el tipo de plantillas que se deben visualizar en el cuadro **Plantillas instaladas**:

- La opción **<Todas las ubicaciones>** está seleccionada de forma predeterminada. Se muestran todas las plantillas de dispositivos y funciones disponibles.
- **Herencia** muestra las plantillas de dispositivos y funciones de SoMachine V3.1 (si está instalado).
- **Usuario**: Sólo muestra las plantillas de dispositivos y funciones que el usuario ha creado o instalado.
- **Sistema**: Muestra las plantillas de dispositivos y funciones estándar suministradas con SoMachine.

Debajo del campo **Ubicación**, aparece la ruta de acceso del directorio en el que se almacenan las bibliotecas de plantillas.

En el cuadro **Plantillas instaladas** se indican las plantillas instaladas en dos grupos: **Plantillas de dispositivos** y **Plantillas de funciones**. Cada biblioteca de plantillas puede contener plantillas de dispositivos o de funciones.

Instalación de bibliotecas de plantillas adicionales

Para añadir bibliotecas de plantillas adicionales a esta lista, siga estos pasos:

Paso	Acción
1	Haga clic en el botón **Instalar** del cuadro de diálogo **Repositorio de plantillas**. **Resultado**: Se abre el cuadro de diálogo **Abrir archivo**.
2	Busque la carpeta en la que esté guardado el archivo de biblioteca de plantillas que desee instalar.
3	Seleccione el archivo de biblioteca que desee instalar y haga clic en **Aceptar**. **Resultado**: La biblioteca de plantillas seleccionada se instala y se indica en el cuadro de diálogo **Repositorio de plantillas**, incluidas las plantillas de dispositivos o funciones que contenga.

Eliminación de bibliotecas de plantillas

Para eliminar una biblioteca de plantillas, haga lo siguiente:

Paso	Acción
1	En la lista **Plantillas instaladas** del cuadro de diálogo **Repositorio de plantillas**, seleccione la biblioteca de plantillas que desee eliminar.
2	Para eliminar la biblioteca de plantillas seleccionada, haga clic en el botón **Desinstalar**. **Resultado**: La biblioteca de plantillas seleccionada se elimina de la instalación.

Cambio de nombre de bibliotecas de plantillas

Para cambiar el nombre de una biblioteca de plantillas, haga lo siguiente:

Paso	Acción
1	En la lista **Plantillas instaladas** del cuadro de diálogo **Repositorio de plantillas**, seleccione la biblioteca de plantillas cuyo nombre desee cambiar.
2	Haga clic en el nombre de la biblioteca de plantillas que desee cambiar. **Resultado**: Se abre un cuadro.
3	Escriba el nombre nuevo en el cuadro y pulse **Intro** o salga del cuadro. **Resultado**: Se asigna el nuevo nombre a la biblioteca de plantillas.

Creación de una biblioteca de plantillas nueva

Para crear una biblioteca de plantillas nueva, haga lo siguiente:

Paso	Acción
1	Para crear una biblioteca de plantillas nueva, seleccione la opción **Usuario** o **<Todas las ubicaciones>** en la lista **Ubicación**.
2	Para crear una biblioteca de plantillas nueva para plantillas de dispositivos, seleccione el nodo **Plantillas de dispositivos** de la lista **Plantillas instaladas** y haga clic en el botón **Crear biblioteca**. **Resultado**: Se añadirá una biblioteca de plantillas nueva con un nombre predeterminado al final de la sección **Plantillas de dispositivos** de la lista **Plantillas instaladas**. Para crear una biblioteca de plantillas nueva para plantillas de funciones, seleccione el nodo **Plantillas de funciones** de la lista **Plantillas instaladas** y haga clic en el botón **Crear biblioteca**. **Resultado**: Se añadirá una biblioteca de plantillas nueva con un nombre predeterminado al final de la sección **Plantillas de funciones** de la lista **Plantillas instaladas**.
3	Cambie el nombre de la biblioteca de plantillas nueva tal como se indica más arriba y complétela con plantillas de dispositivos o funciones utilizando, por ejemplo, las operaciones de copiado y pegado descritas a continuación.

Almacenamiento de bibliotecas de plantillas como archivo

Las bibliotecas de plantillas que contienen plantillas de dispositivos o funciones son archivos XML específicos de SoMachine.

Para que se puedan utilizar en otros PC, haga lo siguiente:

Paso	Acción
1	Seleccione la biblioteca de plantillas que desee exportar en la lista **Plantillas instaladas**.
2	Haga clic en el botón **Guardar como archivo...**
3	En el cuadro de diálogo **Guardar archivo**, busque la carpeta en la que desee guardar el archivo de biblioteca de plantillas.
4	Transfiera el archivo de biblioteca de plantillas al otro PC e instálelo mediante el **Repositorio de plantillas**.

Operaciones de copiado y pegado para bibliotecas de plantillas

El cuadro de diálogo **Repositorio de plantillas** también admite las operaciones de copiado y pegado para bibliotecas de plantillas.

Para copiar una biblioteca de plantillas con la plantilla de dispositivos o funciones que contenga, seleccione el elemento en cuestión en la lista **Plantillas instaladas** y haga clic en el botón **Copiar**.

Ahora seleccione el nodo **Plantillas de dispositivos** o **Plantillas de funciones** y haga clic en el botón **Pegar** para insertar una copia de esta biblioteca de plantillas con un nombre predeterminado en la lista **Plantillas instaladas**.

Cambie el nombre predeterminado por el que desee.

Operaciones de copiado y pegado para plantillas

El cuadro de diálogo **Repositorio de plantillas** también admite las operaciones de copiado y pegado para plantillas de dispositivos o funciones.

Para copiar una plantilla de dispositivos o funciones, seleccione el elemento en cuestión debajo del nodo de una biblioteca de plantillas en la lista **Plantillas instaladas** y haga clic en el botón **Copiar**.

Ahora puede pegar la plantilla en una biblioteca de plantillas si la biblioteca no está protegida contra escritura.

Una biblioteca sólo se puede pegar en otra biblioteca del mismo tipo.

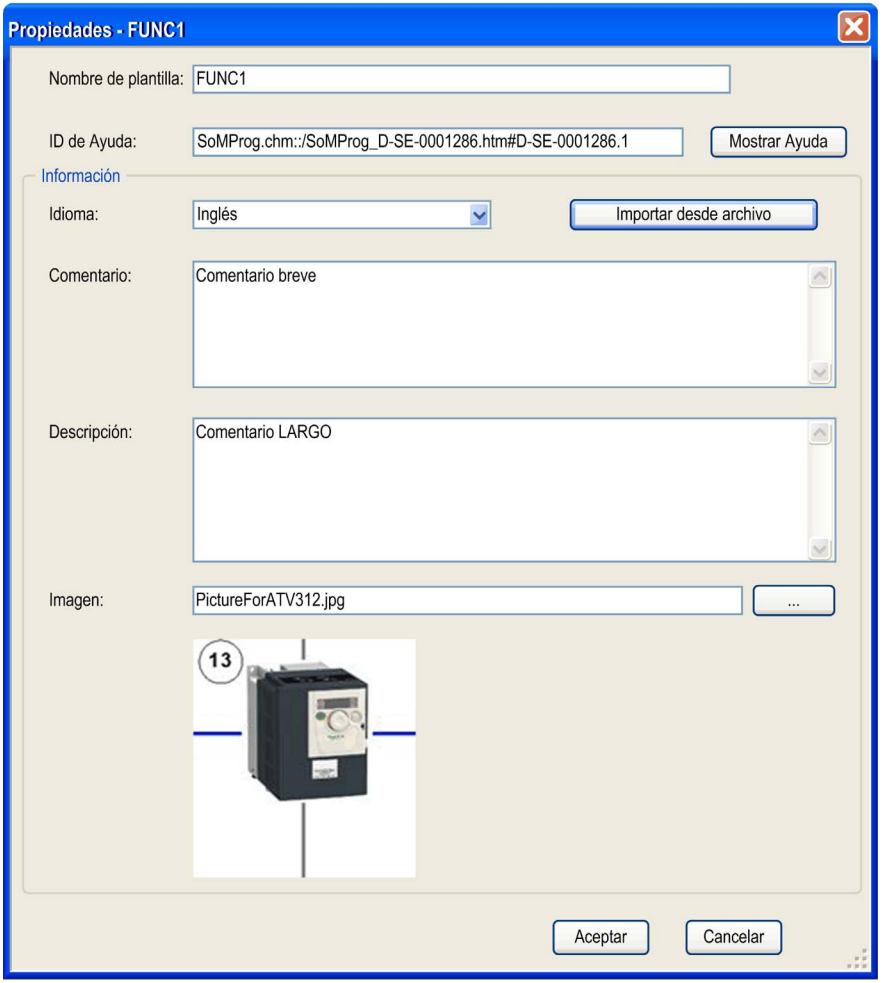
Si lo desea, cambie el nombre predeterminado.

Adición de información para plantillas o bibliotecas de plantillas

El cuadro de diálogo **Repositorio de plantillas** permite introducir más información para las plantillas o las bibliotecas de plantillas.

Para añadir más información, seleccione una biblioteca o una biblioteca de plantillas en la lista **Plantillas instaladas** y haga clic en el botón **Propiedades...**

Se mostrará el cuadro de diálogo **Propiedades** de la biblioteca o biblioteca de plantillas seleccionada.



Si la biblioteca o la biblioteca de plantillas seleccionada no está protegida contra escritura, el cuadro de diálogo **Propiedades** contiene los parámetros siguientes que se pueden editar, junto con sus correspondientes botones:

Elemento	Descripción
Cuadro **Nombre de plantilla**/**Nombre de biblioteca**	Indica el nombre de la biblioteca o biblioteca de plantillas a la que se aplicarán estas propiedades. Para cambiar el nombre, haga clic en este cuadro y adapte el nombre de acuerdo con sus requisitos.

Elemento		Descripción
Cuadro **ID de Ayuda**		Para las plantillas o bibliotecas de plantillas de Schneider Electric, contiene la referencia a la descripción respectiva en la ayuda online. Si hay un documento de ayuda online disponible para sus propias plantillas, puede introducir una referencia completa a su ubicación en la ayuda online o una palabra clave correspondiente a un índice de la ayuda online.
Botón **Mostrar Ayuda**		Abre el documento de ayuda online especificado en el cuadro **ID de Ayuda** o el índice de la ayuda online buscando la palabra clave especificada en el cuadro **ID de Ayuda**.
Sección **Información**		–
	Lista **Idioma**	Contiene los idiomas que están disponibles para la interfaz gráfica de usuario de SoMachine. Si selecciona un idioma, el contenido de los elementos dependientes del idioma **Comentario**, **Descripción** e **Imagen** se muestran en el idioma seleccionado. Si no hay contenido específico del idioma disponible, se muestra el idioma predeterminado, **Inglés**.
	Botón **Importado del archivo**	Muestra un cuadro de diálogo **Abrir** estándar. Permite buscar un archivo XML que contiene el contenido localizado de los elementos dependientes del idioma **Comentario**, **Descripción** e **Imagen**. La estructura del archivo XML debe seguir la estructura indicada en el ejemplo *(véase página 833)*.
	Cuadro **Comentario**	Permite introducir un texto breve (por ejemplo, una descripción general del contenido y el objetivo de la biblioteca o biblioteca de plantillas). Este texto se indica en forma de información sobre herramientas cuando se seleccionan bibliotecas de plantillas en SoMachine.
	Cuadro **Descripción**	Permite escribir un texto largo (por ejemplo, una descripción detallada del contenido y el objetivo de la biblioteca o biblioteca de plantillas).
	Parámetro **Imagen** Botón **...**	Permite introducir una ruta a una imagen específica del idioma. También puede hacer clic en el botón **...** para buscar el archivo gráfico. Formatos gráficos disponibles: ● Mapa de bits: **.bmp* ● JPEG: **.jpg* ● Formato de intercambio de gráficos: **.gif* ● Icono: **.ico* Una vez especificada la imagen, aparecerá en el cuadro de diálogo **Propiedades**. Si hace clic en el botón **Aceptar**, la imagen se integrará en la plantilla.

La casilla **Sólo lectura** sólo está disponible para las bibliotecas de plantillas, con el fin de indicar si la biblioteca de plantilla seleccionada tiene el estado de sólo lectura. No es posible cambiar el estado de la biblioteca de plantillas aquí.

Localización de elementos dependientes del idioma

Puede localizar el contenido de los elementos dependientes del idioma **Comentario**, **Descripción** e **Imagen** importando un archivo XML con la estructura siguiente:

```
<?xml version="1.0" encoding="UTF-8"?>
<TemplateProperties>
<HelpId>SoMProg.chm::/SoMProg_D-SE-0001286.htm#D-SE-0001286.1</HelpId>
<PropertySet languageId = "en">
<Comment>This is a short description</Comment>
<Description>This is a long description</Description>
<ImageFile>PictureEnglish.jpg</ImageFile>
</PropertySet>
<PropertySet languageId = "de">
<Comment>Kurze Beschreibung</Comment>
<Description>Lange Beschreibung</Description>
<ImageFile>PictureGerman.jpg</ImageFile>
</PropertySet>
</TemplateProperties>
```

Capítulo 33
Administración de plantillas de dispositivos

Sección 33.1
Administración de plantillas de dispositivos

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Información acerca de las plantillas de dispositivos

Información general sobre el uso de los términos

La descripción siguiente se aplica a los dispositivos de campo así como a los módulos de E/S, si bien el término dispositivo de campo solamente se utiliza para facilitar la lectura.

Contenido de las plantillas de dispositivos

Las plantillas de dispositivos están relacionadas con un dispositivo de campo o módulo de E/S concreto. Incluyen la información siguiente:

- Configuración del bus de campo
- Lógica de control (programación del controlador) (opcional)
- Elementos de visualización (programación de visualización) (opcional)

Uso de las plantillas de dispositivos

Las plantillas de dispositivos ya disponibles están almacenadas en bibliotecas de plantillas. Cada biblioteca de plantillas contiene la definición de varias plantillas de dispositivos que tienen una base común (por ejemplo, están relacionadas con el control de motores).

Puede seleccionarlas y adaptarlas a los requisitos de sus proyectos de SoMachine individuales para crear nuevos dispositivos de campo previamente configurados y listos para usar.

Creación de plantillas de dispositivos nuevas

Con el fin de que los dispositivos de campo ya configurados se puedan volver a utilizar para cualquier proyecto de SoMachine, guárdelos como plantillas de dispositivos. Esto también incluye la visualización y la programación del controlador relacionadas con este dispositivo de campo.

Versiones de las plantillas de dispositivos

Durante la creación de una plantilla de dispositivos, se verifica si la descripción del dispositivo que se va a crear existe realmente. Si no existe, el dispositivo se actualiza automáticamente a la última versión, si existe una versión superior.

Adición de dispositivos a partir de plantillas

Descripción general

Las plantillas de dispositivos siempre están relacionadas con un dispositivo de bus de campo específico. Incluyen la información siguiente:

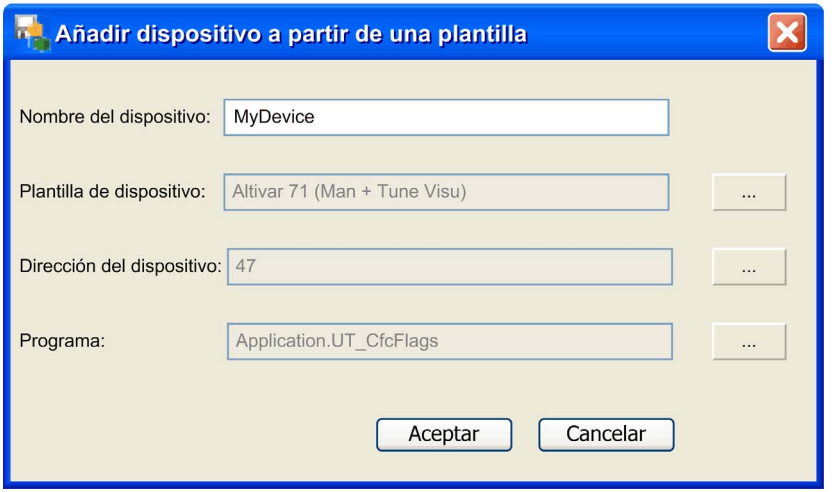
- Configuración del dispositivo de bus de campo
- Lógica de control (programación del controlador) (opcional)
- Elementos de visualización (programación de visualización) (opcional)

Puede crear sus propias plantillas de dispositivos a partir del proyecto. Para obtener más información, consulte el capítulo *Pasos para crear una plantilla de dispositivos* *(véase página 846)*.

Adición de un dispositivo a partir de una plantilla

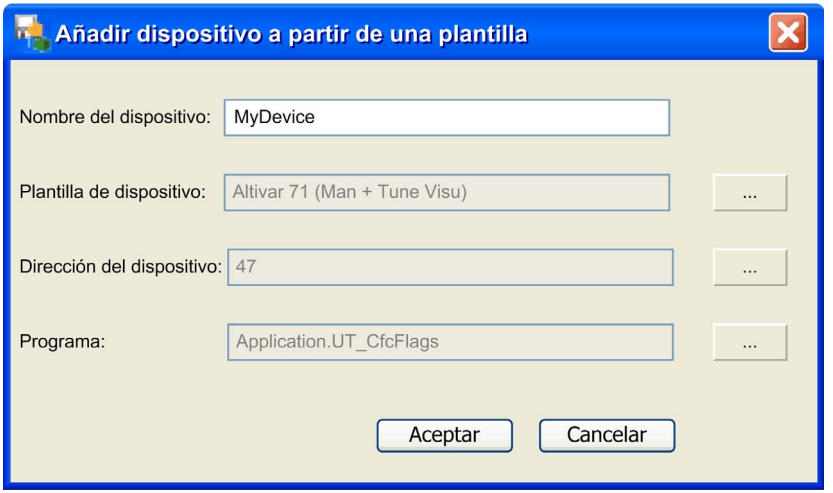
SoMachine ofrece 2 formas de añadir un dispositivo a partir de una plantilla de dispositivos:

- Creación de un dispositivo arrastrando y soltando una plantilla de dispositivos:

Paso	Acción
1	Abra la vista **Dispositivos de campo** del catálogo de hardware.
2	En la parte inferior de la vista **Dispositivos de campo**, active la opción **Plantilla de dispositivos**. **Resultado**: Las plantillas de dispositivos de campo disponibles en SoMachine se muestran en la vista **Dispositivos de campo**.
3	Seleccione una entrada en la vista **Dispositivos de campo**, arrástrela al árbol **Dispositivos** y suéltela en el subnodo adecuado de un controlador. **Observación**: SoMachine resalta los subnodos adecuados. **Resultado**: Aparecerá el cuadro de diálogo **Añadir dispositivo a partir de una plantilla**. 

Paso	Acción
4	En el cuadro de diálogo **Añadir dispositivo a partir de una plantilla**, defina el **Nombre de dispositivo** y la **Dirección del dispositivo** si el bus de campo requiere direcciones numéricas. Si las plantillas de dispositivos incluyen lógica de control, seleccione el programa (POU) en el que se insertará la lógica de control.
5	Haga clic en el botón **Aceptar**. **Resultado**: El dispositivo se crea y configura de acuerdo con la plantilla de dispositivos, lo que incluye las pantallas de visualización opcionales y la lógica de control.

- Creación de un dispositivo utilizando una plantilla de dispositivos mediante el menú contextual:

Paso	Acción
1	Abra el árbol **Dispositivos** .
2	Haga clic con el botón derecho en el administrador de dispositivos de campo y ejecute el comando **Añadir dispositivo a partir de una plantilla** en el menú contextual. **Resultado**: Aparecerá el cuadro de diálogo **Añadir dispositivo a partir de una plantilla**. 
3	En el cuadro de diálogo **Añadir dispositivo a partir de una plantilla**, seleccione la **Plantilla de dispositivos** que se utilizará, el **Nombre de dispositivo** y la **Dirección del dispositivo** si el bus de campo requiere direcciones numéricas. Si las plantillas de dispositivos incluyen lógica de control, seleccione el programa (POU) en el que se insertará la lógica de control.
4	Haga clic en el botón **Aceptar**. **Resultado**: El dispositivo se crea y configura de acuerdo con la plantilla de dispositivos, lo que incluye las pantallas de visualización opcionales y la lógica de control.

NOTA: La función deshacer/rehacer no está disponible para el proceso de creación de dispositivos de campo.

Nomenclatura de los objetos

Para evitar conflictos de nomenclatura si se utiliza la misma plantilla de dispositivos como base para crear distintos dispositivos de campo, se aplican las convenciones sobre nomenclatura siguientes a los dispositivos de campo y los objetos asociados (FB, visualización y variables):

Si el nombre del objeto original...	**Entonces...**
Caso 1:	
incluye el nombre del dispositivo de campo original,	esta parte del objeto se reemplazará por el nombre del nuevo dispositivo de campo que se cree.
Ejemplo:	
La plantilla de dispositivos del dispositivo de campo `ATV1` contiene una variable `Var_ATV1_Input`.	En el caso de un dispositivo nuevo `Axis1` creado con esta plantilla, la variable nueva se denominará correspondientemente `Var_Axis1_Input`.
Caso 2:	
no contiene el nombre del dispositivo original,	se insertarán en el nombre original el nombre del dispositivo nuevo y un carácter de subrayado para formar un nombre nuevo exclusivo.
Ejemplo:	
La plantilla de dispositivos del dispositivo de campo `ATV1` contiene una variable `Var_Input1`.	En el caso de un dispositivo nuevo `Axis1` creado con esta plantilla, la variable nueva se denominará correspondientemente `Axis1_Var_Input1`.

Creación de una plantilla de dispositivos basada en dispositivos de campo o módulos de E/S

Descripción general

Puede crear plantillas de dispositivos basadas en dispositivos de campo o módulos de E/S. La descripción siguiente se aplica a los dispositivos de campo así como a los módulos de E/S, si bien el término dispositivo de campo solamente se utiliza para facilitar la lectura.

En los párrafos siguientes se enumeran:

- Los criterios que se deben cumplir para guardar un dispositivo de campo o módulo de E/S, incluidas la lógica y la visualización, como plantilla de dispositivos.
- La información que se guarda en la plantilla de dispositivos.

Requisitos previos de los dispositivos de campo

Los dispositivos de campo deben cumplir los siguientes criterios para poderse guardar como plantillas de dispositivos:

- Los dispositivos de campo deben estar vinculados a los buses de campo enumerados en la lista Buses de campo admitidos *(véase página 825)*.
- El tipo de dispositivo se debe instalar en el **Repositorio de dispositivos**.

Requisitos previos de los módulos de E/S

Sólo los módulos de E/S admitidos pueden guardarse como plantillas de dispositivos *(véase página 825)*.

Requisitos previos de la aplicación

Sólo se pueden crear plantillas desde aplicaciones correctas. Por correctas se entiende que no se han detectado errores durante el proceso de **compilación**.

Requisitos previos para incluir la lógica de control en una plantilla

Para incluir la lógica de control en una plantilla, es necesario que dicha lógica contenga una o varias secciones de código que intercambien datos con este dispositivo de campo. Esta lógica de control debe ejecutarse (añadirse a una tarea o que otro programa la llame). De lo contrario, no se tendrá en cuenta cuando se ejecute el comando **Compilar**.

Información de dispositivo guardada en plantillas de dispositivos

La información de dispositivos de campo siguiente se guarda en plantillas de dispositivos:

- configuración del dispositivo
- asignación de E/S del dispositivo de campo
- visualizaciones apropiadas para el dispositivo de campo
- lógica de control que intercambia datos con el dispositivo de campo

Visualizaciones apropiadas para la creación de plantillas de dispositivos

Descripción general

Cada plantilla de dispositivos se puede asociar con una o varias visualizaciones de Logic Builder. Los tipos de visualización admitidos se describen a continuación.

Visualizaciones admitidas

SoMachine admite los dos tipos de visualización:

- visualizaciones simples
- visualizaciones modulares mediante marcos

Las visualizaciones mediante marcos ofrecen una mayor flexibilidad y modularidad.

Visualizaciones simples

Las visualizaciones sin marcos se basan en un solo objeto de visualización creado para el dispositivo de E/S.

SoMachine hace referencia a los datos del dispositivo de E/S en las propiedades de los elementos visuales. Cuando se crea un nuevo dispositivo basado en esta plantilla de dispositivos, SoMachine sustituye directamente las variables en las propiedades de los elementos visuales.

Visualizaciones con marcos

Una visualización que usa marcos se genera a partir de una pantalla principal que puede integrarse con otras visualizaciones, usando varias visualizaciones más pequeñas que se combinarán como módulos en áreas predefinidas de la pantalla principal (marcos).

En la pantalla principal se coloca un objeto-marco, como un objeto rectangular, como contenedor. Puede asignar otra visualización a dicho contenedor.

A continuación, la visualización integrada puede usarse con una interfaz para acceder a elementos visuales internamente.

Para obtener más información, consulte el apartado **Programación con SoMachine** → **Visualización** de la ayuda online de SoMachine.

Para usar las visualizaciones integradas para plantillas de dispositivos, defina una interfaz que incluya definiciones de todas las variables relacionadas con la conexión al dispositivo de E/S o bloque de funciones para cada módulo de visualización. Cuando se crea un dispositivo basado en esta plantilla de dispositivos, SoMachine adapta todos los marcadores de posición de las visualizaciones integradas de acuerdo con el nombre del dispositivo de E/S creado.

NOTA: Todas las visualizaciones que usan marcos y los bloques de funciones vinculados al dispositivo de E/S específico deben definirse en una biblioteca para que SoMachine pueda encontrarlos.

Información adicional acerca de la integración de la lógica de control en plantillas de dispositivos

Descripción general

Puede incluir una lógica de control en una plantilla de dispositivos si la lógica contiene una o más secciones de código que intercambian datos con este dispositivo de campo de una de las siguientes formas:

- Una sección de código usa una nueva variable definida en la asignación de E/S del dispositivo de campo.
- Una sección de código y la asignación de E/S del dispositivo de campo usan una variable común que se define en una GVL o un programa del controlador contenido por la aplicación a la que pertenece la sección del código.
  NOTA: Si usa estructuras o matrices, asegúrese de que sólo estén relacionadas con un dispositivo de campo.
- Una sección de código y el dispositivo de campo usan una variable fija específica del dispositivo (por ejemplo, las variables de referencia del eje usadas con las unidades Altivar o Lexium).

Llamadas interconectadas de secciones de código

Las secciones de código están formadas por una secuencia de llamadas interconectadas de bloques de funciones, funciones y operadores.

Si existe una de las siguientes relaciones entre las llamadas individuales, se considera que están conectadas:

- Hay una conexión gráfica entre las llamadas individuales en CFC, FBD y LD.
- Hay una variable conectada a la salida de una llamada y a la entrada de la otra llamada.
- Una llamada usa el parámetro de la otra llamada.

Selección individual de bloques de funciones

Puede seleccionar individualmente los bloques de funciones incluidos en estas secciones de código que intercambian datos con el dispositivo de campo que se incluirá en la plantilla de dispositivos. Esto permite crear otras plantillas de dispositivos que proporcionan distintas funciones para el mismo dispositivo de campo.

NOTA: El tipo de bloque de funciones debe estar definido en una biblioteca.

Inclusión de expresiones en las plantillas de dispositivos

Las expresiones, así como las variables usadas en estas expresiones, que están conectadas a los parámetros de un bloque de funciones, una función o un operador, se guardan automáticamente en la plantilla de dispositivos.

Recomendaciones generales para la creación de la lógica de control

Incluya sólo una lógica de control simple en una plantilla de dispositivos.

De este modo, las secciones de código trabajan de forma idéntica aunque se creen en diferentes lenguajes IEC.

NOTA: Para la lógica de control compleja, es preferible crear una plantilla de funciones.

Recomendaciones para la creación de la lógica de control en FBD/LD

Evite los elementos de detección de flancos, ya que no existen en otros lenguajes IEC.

Si es posible, use los bloques de funciones `R_TRIG` o `F_TRIG`.

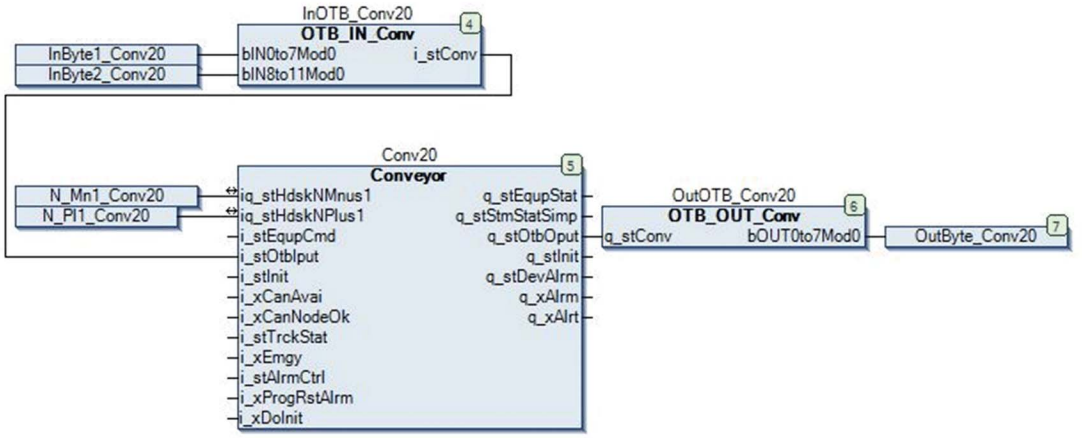
Recomendaciones para la creación de la lógica de control en CFC

Use el comando **Orden de ejecución →Ordenar conforme al flujo de datos** para ordenar los elementos CFC que pertenecen a la misma sección de código, de acuerdo con su posición en el flujo de datos. Esto proporciona una mejor compatibilidad con otros lenguajes IEC.

Deje espacio (en dirección horizontal) entre los elementos CFC individuales, ya que debido al cambio de nombre, los nombres de las variables se ampliarán al crear un dispositivo a partir de una plantilla.

Ejemplo de lógica de control

En la siguiente figura se muestra un ejemplo típico de una sección de código para un dispositivo de E/S distribuidas Advantys OTB en una aplicación de transporte:



La sección de código está formada por los siguientes bloques de funciones:

Nombre	Tipo	Función
`InOTB_Conv20`	Bloque de entrada	Conversión de datos procedentes de OTB al formato necesario para el bloque de control
`Conv20`	Bloque de control	Datos de procesamiento
`OutOTB_Conv20`	Bloque de salida	Conversión de datos procedentes del bloque de control al formato necesario para OTB

Las variables `InByte1_Conv20`, `InByte2_Conv20` y `OutByte_Conv20` están definidas en la asignación de E/S de OTB. Esto significa que la sección de código intercambia datos con el dispositivo OTB. De esta forma, se convierte en parte de la plantilla de dispositivos.

Pasos para crear una plantilla de dispositivos

Descripción general

En los párrafos siguientes se enumeran los pasos que deben llevarse a cabo para guardar dispositivos de campo que cumplan los criterios indicados en *Creación de una plantilla de dispositivos basada en dispositivos de campo o módulos de E/S* *(véase página 841)*.

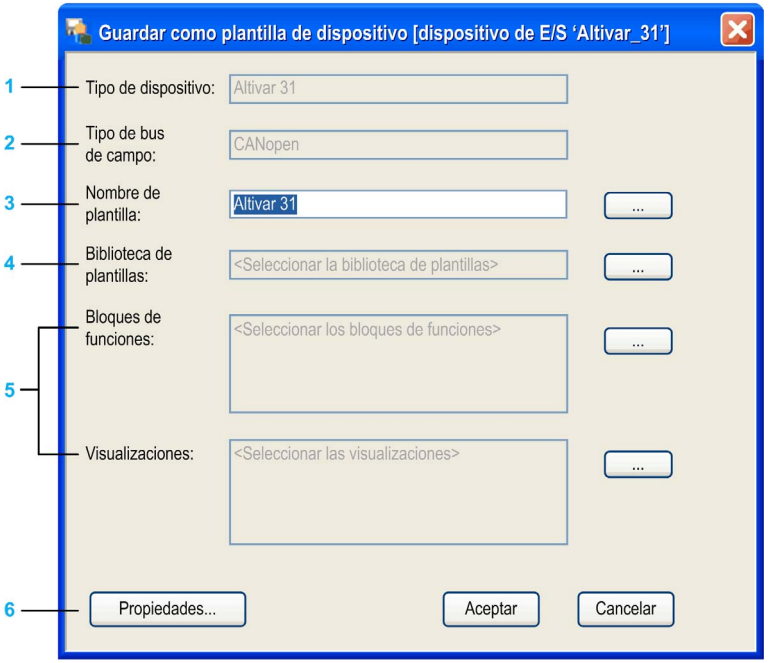
Pasos para guardar un dispositivo de campo como plantilla

Para guardar un dispositivo de campo existente como plantilla de dispositivos, haga lo siguiente:

Paso	Acción
1	Haga clic con el botón derecho en el dispositivo de campo que desea guardar como plantilla de dispositivos en el árbol **Dispositivos**.
2	Seleccione el comando **Guardar como plantilla de dispositivo** en el menú contextual. **Resultado**: SoMachine genera la aplicación automáticamente. Una vez que se haya completado este proceso correctamente, aparecerá el cuadro de diálogo **Guardar como plantilla de dispositivo**.
3	Defina la nueva plantilla de dispositivos en el cuadro de diálogo **Guardar como plantilla de dispositivo** tal como se indica a continuación.
4	Haga clic en **Aceptar** para cerrar el cuadro de diálogo **Guardar como plantilla de dispositivo** y crear la nueva plantilla de dispositivos.

Cuadro de diálogo Guardar como plantilla de dispositivo

El cuadro de diálogo **Guardar como plantilla de dispositivo** contiene los siguientes parámetros:



1 Indica el tipo de dispositivo de campo en el que se basa la plantilla de dispositivos.
2 Indica el tipo de bus de campo del dispositivo de campo.
3 El nombre de la plantilla de dispositivos que se creará (inicialmente, el nombre del dispositivo de campo original).
4 Seleccione la biblioteca de plantillas a la que se añadirá la plantilla de dispositivos.
5 Seleccione bloques de funciones y visualizaciones que deberían guardarse con la plantilla de dispositivos.
6 Botón **Propiedades** para añadir más información a la plantilla de dispositivos.

Definición de un nombre para la plantilla de dispositivos nueva

Utilice el cuadro de texto **Nombre de plantilla** para definir un nombre para la plantilla de dispositivos nueva.

De forma predeterminada, este cuadro de texto incluye el nombre del dispositivo de campo seleccionado.

Puede escribir el nombre que desee directamente en este cuadro de texto o hacer clic en el botón **...** para seleccionar una plantilla de dispositivos existente en las listas si desea sobrescribir esta plantilla de dispositivos.

Selección de la biblioteca de plantillas

Para seleccionar una de las bibliotecas de plantillas previamente instaladas o creadas en la que se almacenará la plantilla de dispositivos, haga lo siguiente:

Paso	Acción
1	En el cuadro de diálogo **Guardar como plantilla de dispositivo**, haga clic en el botón **...** situado a la derecha del cuadro de texto **Biblioteca de plantillas**. **Resultado**: Aparecerá el cuadro de diálogo **Seleccionar la biblioteca de plantillas**.
2	En el cuadro de diálogo **Seleccionar la biblioteca de plantillas** se muestran todas las bibliotecas de plantillas que se han instalado para el proyecto actual o que se han creado. No se muestran las bibliotecas de plantillas protegidas contra escritura. Para añadir la plantilla de dispositivos nueva a una de estas bibliotecas de plantillas, seleccione la entrada apropiada y haga clic en **Aceptar**.

Selección de los bloques de funciones

Para seleccionar las instancias de los bloques de funciones que se incluirán en la plantilla de dispositivos, siga estos pasos:

Paso	Acción
1	En el cuadro de diálogo **Guardar como plantilla de dispositivo**, haga clic en el botón **...** situado a la derecha del cuadro de texto **Bloques de funciones**. **Resultado**: Aparecerá el cuadro de diálogo **Seleccionar los bloques de funciones**. En el cuadro de diálogo **Seleccionar los bloques de funciones** se muestran todas las instancias de los bloques de funciones contenidas en la lógica de control del dispositivo *(véase página 843)* de campo.
2	Seleccione la casilla de un bloque de funciones específico para seleccionarlo para la plantilla de dispositivos. Como alternativa, seleccione la casilla de un nodo raíz para seleccionar todos los elementos que hay bajo dicho nodo.
3	Haga clic en el botón **Aceptar**.

Selección de las visualizaciones

Para seleccionar las visualizaciones que se incluirán en el dispositivo de campo, siga estos pasos:

Paso	Acción
1	En el cuadro de diálogo **Guardar como plantilla de dispositivo**, haga clic en el botón **...** situado a la derecha del cuadro de texto **Visualizaciones**. **Resultado**: Aparecerá el cuadro de diálogo **Seleccionar las visualizaciones**. En el cuadro de diálogo **Seleccionar las visualizaciones** se muestran las visualizaciones vinculadas con el dispositivo de campo o con uno de los bloques de funciones seleccionados.
2	Seleccione la casilla de una visualización específica para seleccionarla para la plantilla de dispositivos. Como alternativa, seleccione la casilla de un nodo raíz para seleccionar todos los elementos que hay bajo dicho nodo.
3	Haga clic en el botón **Aceptar**.

Adición de más información a la nueva plantilla de dispositivos

Para añadir más información a la plantilla de dispositivos nueva, haga clic en el botón **Propiedades...** Se abre el cuadro de diálogo **Propiedades**. Este cuadro permite introducir más información de la plantilla de dispositivos. Puesto que este cuadro de diálogo es idéntico al de las plantillas de dispositivos y las bibliotecas de plantillas, consulte la descripción en el capítulo Adición de información para plantillas o bibliotecas de plantillas *(véase página 830)*.

Capítulo 34
Administración de plantillas de funciones

Sección 34.1
Administración de plantillas de funciones

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Información acerca de las plantillas de funciones

Contenido de las plantillas de funciones

Las plantillas de funciones representan funcionalidades dedicadas de control y visualización que están asociadas con una función de aplicación.

Una plantilla de funciones puede incluir los elementos siguientes:

- uno o varios programas IEC
- uno o varios dispositivos de campo o módulos de E/S que utiliza la función de aplicación
- una o varias visualizaciones que se utilizan para visualizar la función de aplicación
- una o varias listas de variables globales
- una o varias variables globales que se pueden compartir con otras funciones de aplicaciones
- una o varias trazas
- una o varias tablas CAM
- una o varias variables de E/S que se deben asignar a un canal de E/S
- uno o varios parámetros de plantilla

Uso de plantillas de funciones

Las plantillas de funciones ya disponibles están almacenadas en bibliotecas de plantillas. Cada biblioteca de plantillas contiene la definición de varias plantillas de funciones que tienen una base común (por ejemplo, todas están relacionadas con aplicaciones de empaquetado).

Puede seleccionarlas y adaptarlas fácilmente a los requisitos de sus proyectos de SoMachine individuales para crear nuevas funciones de aplicaciones listas para usar.

Creación de plantillas de funciones nuevas

Con el fin de que la función de aplicación ya creada se pueda volver a utilizar para cualquier proyecto de SoMachine, puede guardarla como plantilla de funciones.

Cuando guarde la plantilla de funciones, decida en qué biblioteca de plantillas desea almacenarla.

Versiones de plantillas de funciones

Durante la creación de una plantilla de funciones, se verifica si la descripción del dispositivo que se va a crear existe realmente. Si no existe, el dispositivo se actualiza automáticamente a la última versión, si existe una versión superior.

Adición de funciones a partir de plantillas

Procedimiento

SoMachine ofrece 2 formas de añadir una función a partir de una plantilla de funciones:

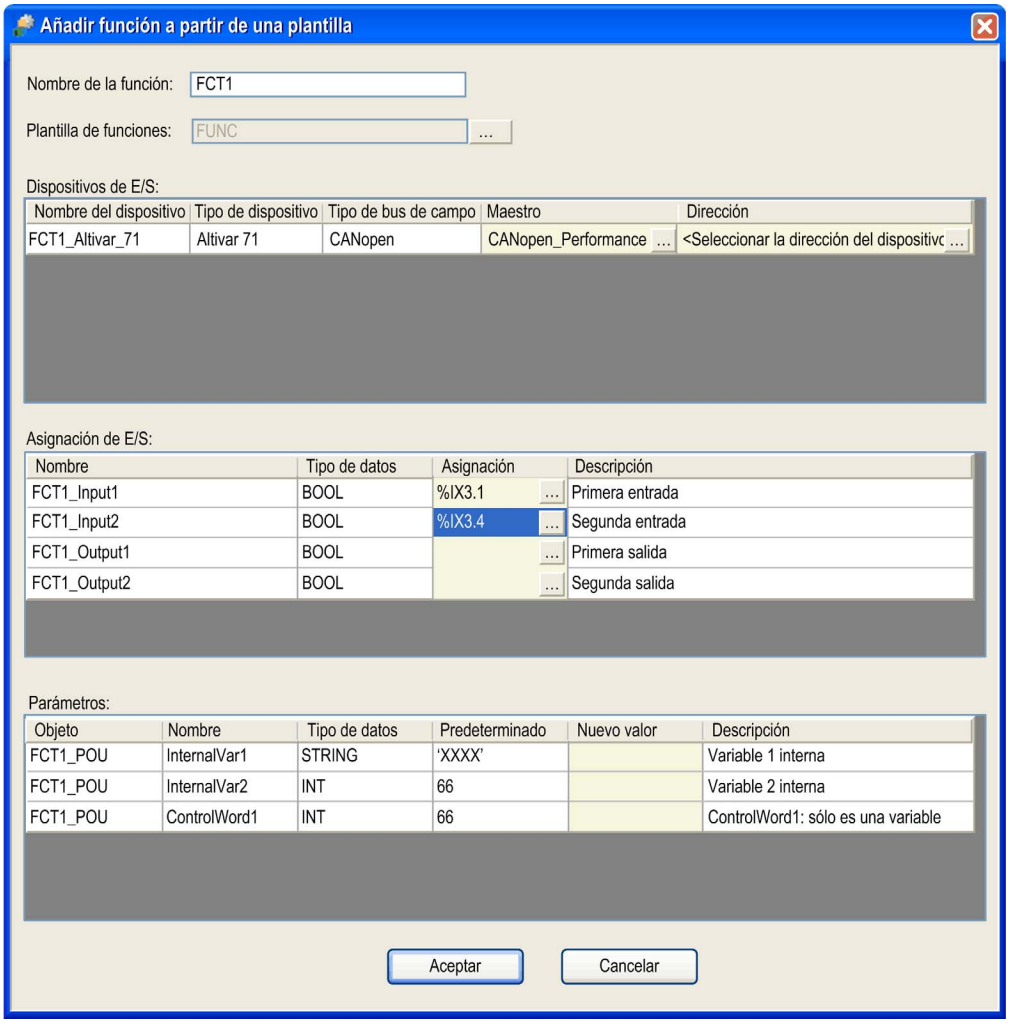
Para añadir una función de aplicación a partir de una plantilla de funciones mediante el método de arrastrar y soltar, haga lo siguiente:

Paso	Acción
1	Abra la vista **Macros** del **Catálogo de software**.
2	Seleccione una plantilla de funciones en la vista **Macros**, arrástrela al árbol **Aplicaciones** y suéltela en un nodo **Aplicación** adecuado o en una carpeta debajo del nodo **Aplicación**. **Observación**: SoMachine resalta los subnodos adecuados. **Resultado**: Aparecerá el cuadro de diálogo **Añadir función a partir de una plantilla**.

Para añadir una función de aplicación a partir de una plantilla de funciones mediante el menú contextual, haga lo siguiente:

Paso	Acción
1	Abra el árbol **Aplicaciones**.
2	Haga clic con el botón derecho del ratón en un nodo **Aplicación** o en una carpeta debajo del nodo **Aplicación** y ejecute el comando **Añadir función a partir de una plantilla** en el menú contextual. **Resultado**: Aparecerá el cuadro de diálogo **Añadir función a partir de una plantilla**.

Cuadro de diálogo Añadir función a partir de una plantilla



El cuadro de diálogo **Añadir función a partir de una plantilla** ofrece los elementos siguientes para configurar la función:

Elemento		Descripción
Cuadro de texto **Nombre de la función**		Escriba un nombre que se utilizará para la nueva carpeta de esta aplicación y para nombrar los elementos que contenga.
Plantilla de funciones		Haga clic en el botón **...** y seleccione una plantilla de funciones en el cuadro de diálogo **Seleccionar la plantilla de funciones**.
Tabla **Dispositivos de E/S**		–
	Nombre del dispositivo	Contiene el nombre del dispositivo de campo futuro. No se puede cambiar este nombre.
	Tipo de dispositivo	Indica el tipo del dispositivo de campo. No se puede editar esta celda.
	Tipo de bus de campo	Indica el tipo de bus de campo del dispositivo de campo. No se puede editar esta celda.
	Maestro	Contiene el maestro de bus de campo al que está conectado el dispositivo de campo. Si hay varios maestros, se puede seleccionar uno en la lista.
	Dirección	Inicialmente vacío. En el caso de dispositivos de campo en buses de campo que requieran direcciones numéricas (línea serie Modbus y CANopen), haga clic en el botón **...** situado junto al campo y asigne la dirección que desee.
Tabla **Asignación de E/S**		Muestra las variables de E/S que forman parte de la plantilla de funciones. Permite asignarlas a los canales de E/S de dispositivos y módulos existentes.
	Nombre	Contiene el nombre de la variable de E/S que se debe asignar en un canal de E/S.
	Tipo de datos	Indica el tipo de datos del canal de E/S al que se asignó originalmente la variable de E/S.
	Asignación	Haga clic en el botón **...** para abrir el cuadro de diálogo **Seleccionar asignación de E/S**. Le permite seleccionar un canal de E/S en el que puede asignar la variable seleccionada. Después de que la variable se haya asignado a un canal de E/S, este campo **Asignación** contiene la dirección de entrada o salida del canal de E/S al que está asignada la variable.
	Descripción	Contiene una descripción de la variable de E/S.
Tabla **Parámetros**		Muestra los parámetros de plantilla incluidos en la plantilla de funciones.
	Objeto	Indica el nombre de la GLV o el programa en el que está definida la variable. No se puede editar este campo.
	Nombre	Contiene el nombre de la variable. No se puede editar esta celda.
	Tipo de datos	Indica el tipo de datos de la variable. No se puede editar esta celda.

Elemento		Descripción
	Predeterminada	Indica el valor predeterminado de la variable. Este es el valor inicial de la variable cuando se creó la plantilla. No se puede editar esta celda.
	Nuevo valor	Edite esta celda si desea asignar un valor nuevo a la variable. Si deja esta celda vacía, se utilizará el valor **Predeterminado** para esta variable. Introduzca un valor que sea válido para el tipo de datos en cuestión.
	Descripción	Contiene una descripción de la variable.
Botón **Aceptar**		Confirme los ajustes haciendo clic en el botón **Aceptar**. **Resultado**: SoMachine Comprueba si los ajustes son correctos e inserta la nueva función de aplicación como un nodo independiente debajo del nodo **Aplicación** o muestra un mensaje de detección de error.

Cuadro de diálogo Seleccionar asignación de E/S

El cuadro de diálogo **Seleccionar asignación de E/S** se utiliza para asignar una variable seleccionada en el cuadro de diálogo **Añadir función a partir de una plantilla** a un canal de E/S.

Muestra los canales de E/S disponibles en una estructura de árbol similar al árbol **Dispositivos** . El nodo raíz es el controlador. Sólo se muestran los canales de E/S cuyo tipo de datos coincide con el tipo de datos de la nueva variable.

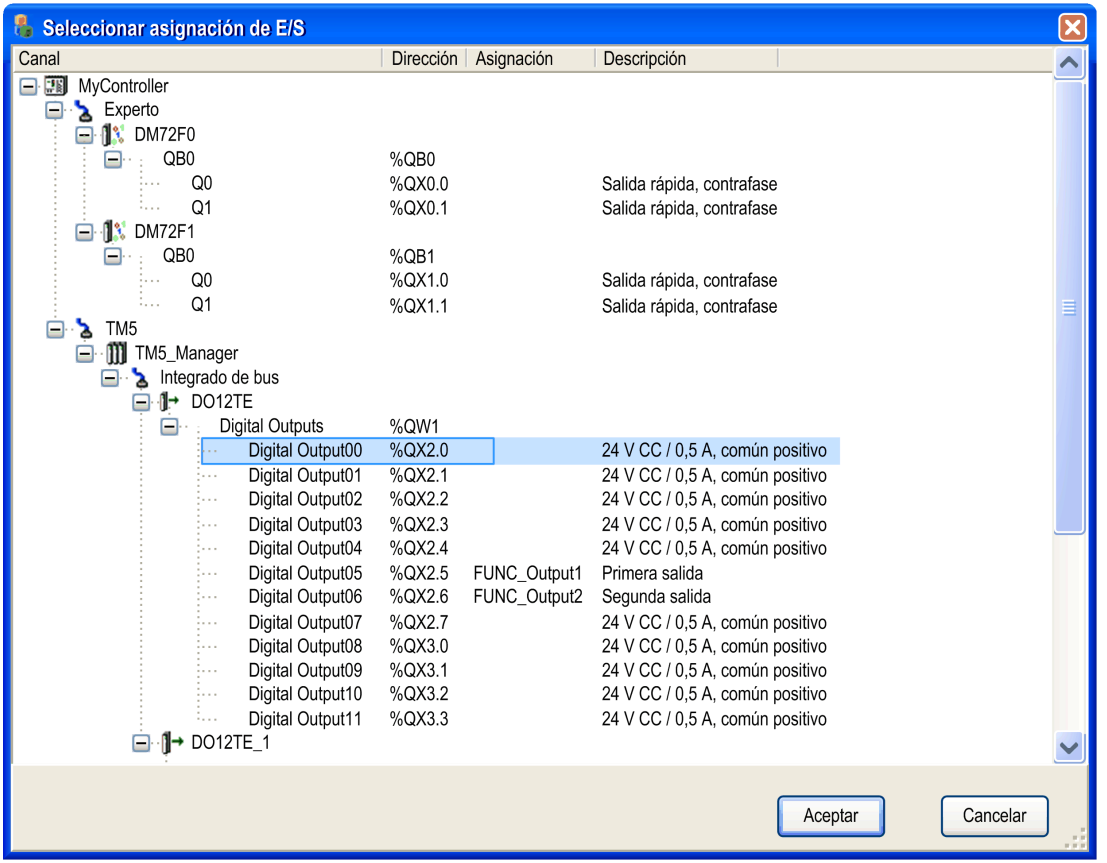
2 tipos de datos son compatibles si tienen nombres de tipo idénticos o si son tipos de datos IEC elementales con el mismo tamaño.

Ejemplo:

UINT --> INT está permitido

UDINT --> INT no está permitido

Visualice los subnodos haciendo clic en el signo más.



El cuadro de diálogo **Seleccionar asignación de E/S** contiene las columnas siguientes:

Columna	Descripción
Canal	Contiene la estructura de árbol. Cada dispositivo se representa mediante el nombre del dispositivo y el icono del dispositivo. Cada canal de E/S se representa por medio del nombre de canal.
Dirección	Contiene la dirección de entrada / salida que corresponde al canal de E/S.
Asignación	Contiene la variable de E/S que está asignada actualmente en el canal de E/S.
Descripción	Contiene la descripción del canal de E/S.

Tenga en cuenta las prácticas siguientes para asignar variables a canales de E/S:

- Asigne todas las variables proporcionadas por la plantilla de funciones a los canales de E/S.
- Puede asignar una variable de E/S de una plantilla de funciones a un canal de E/S que ya tenga una asignación. La asignación existente se sobreescribe.
- No se permiten asignaciones que causen múltiples asignaciones de variables en el mismo canal de E/S.

Objetos creados

La plantilla de funciones crea los objetos siguientes en el proyecto:

Objeto	Descripción
Directorio raíz	Se crea una carpeta nueva bajo el nodo **Aplicación** en la vista **Dispositivos** con el nombre definido en el cuadro de texto **Nombre de la función**.
Dispositivos de campo	Los dispositivos de campo incluidos en la plantilla de funciones se crean con nombres que respetan las reglas de nomenclatura y están conectados al maestro de bus de campo. Si es necesario, la asignación de E/S se ajusta automáticamente.
Visualizaciones	Las visualizaciones incluidas en la plantilla de funciones se crean debajo del directorio raíz con nombres que respetan las reglas de nomenclatura. Las propiedades de la visualización se ajustan automáticamente.
Programas	Los programas incluidos en la plantilla de funciones se crean debajo del directorio raíz con nombres que respetan las reglas de nomenclatura. Los nombres de los objetos del programa que forman parte de la plantilla de funciones se ajustan automáticamente.
Trazas	Las trazas incluidas en la plantilla de funciones se crean debajo del directorio raíz con nombres que respetan las reglas de nomenclatura y se pueden utilizar para realizar el seguimiento de variables pertenecientes a la función de aplicación.
tablas CAM	Las tablas CAM incluidas en la plantilla de funciones se crean debajo del directorio raíz con nombres que respetan las reglas de nomenclatura. Sólo son necesarias si la función de aplicación incluye dispositivos SoftMotion.
Configuración de tareas	La plantilla de funciones ajusta la configuración de tareas cuando es necesario.
Listas de variables globales	Las listas de variables globales incluidas en la plantilla de funciones se crean debajo del directorio raíz con nombres que respetan las reglas de nomenclatura.
Variables externas	Las variables globales cuyas listas de variables globales no pertenecen a la plantilla de funciones se restauran en su lista de variables globales original del modo siguiente: ● Si aún no existe una lista de variables globales con el nombre original debajo de la aplicación, se crea automáticamente. ● Si aún no existe una variable global con el nombre original en esta lista de variables globales, se crea automáticamente. Si el tipo de variable global no es correcto, SoMachine muestra un mensaje de detección de error.

Objeto	Descripción
Variables persistentes	Las variables persistentes se restauran en la lista de variables respectiva de la aplicación del modo siguiente: ● Si aún no existe una lista de variables persistentes debajo de la aplicación, se crea automáticamente con su nombre original. ● Si aún no existe una variable con el nombre original en la lista de variables persistentes, se crea automáticamente. Si el tipo de variable persistente no es correcto, SoMachine emite un mensaje.

Todos los objetos que se creen con la instanciación de la plantilla de funciones aparecerán en la subventana **Mensajes**.

Nomenclatura de los objetos

Para evitar conflictos de nomenclatura, si se instancia la misma plantilla de funciones varias veces en el mismo dispositivo controlador, se aplican las convenciones de nomenclatura siguientes a las funciones de aplicaciones y los objetos asociados:

Si el nombre del objeto original...	Entonces...
Caso 1:	
contiene el nombre de la función de aplicación,	esta parte del objeto se reemplazará por el nombre de la nueva función de aplicación que se cree.
Ejemplo:	
La función de aplicación original de plantilla `Axis` contiene un programa `Axis_Init`.	En el caso de una función de aplicación nueva `Axis1` creada con esta plantilla, el programa nuevo se denominará correspondientemente `Axis1_Init`.
Caso 2:	
no contiene el nombre de la función de aplicación,	el nombre de la función de aplicación nueva y un carácter de subrayado se insertarán en el nombre original para formar un nombre nuevo exclusivo.
Ejemplo:	
La función de aplicación original `Axis` contiene un programa `InitProg`.	En el caso de una función de aplicación nueva `Axis1` creada con esta plantilla de funciones, el programa nuevo se denominará correspondientemente `Axis1_InitProg`.

NOTA: Utilice más bien nombres cortos para las funciones de aplicaciones para que no aparezcan cortados.

Funciones de aplicaciones como base para plantillas de funciones

Descripción general

En los párrafos siguientes se enumeran:

- Los criterios que deben cumplirse para guardar como plantilla de funciones una función de aplicación con sus dispositivos de campo, visualizaciones y módulos de E/S asociados.
- La información que se guarda en la plantilla de funciones.

Definición de funciones de aplicaciones como plantillas de funciones

Para guardar las funciones de aplicaciones como plantillas de funciones, haga clic con el botón derecho en un subnodo del nodo **Aplicación** del árbol **Aplicaciones**. O bien puede crear su propia plantilla en la vista **Macros**, seleccionando los objetos para la plantilla. Estos 2 procedimientos se describen en el capítulo *Pasos para crear una plantilla de funciones* *(véase página 864)*.

Requisitos previos de la aplicación

Sólo se pueden crear plantillas desde aplicaciones correctas. Por correctas se entiende que no se han detectado errores durante el proceso de compilación.

Requisitos previos para guardar una función de aplicación como plantilla de funciones

Para poder guardar una función de aplicación como plantilla de funciones, se deben ejecutar todos los programas de la función de aplicación.

Esto significa que deben cumplir uno de los criterios siguientes:

- Se deben añadir a una tarea.
- Los debe llamar otro programa.

De lo contrario, no se tendrán en cuenta cuando se ejecute el comando **Compilar**.

Variables de E/S en las plantillas de funciones

Una variable de E/S es una variable que se asigna a un canal de E/S de un dispositivo de campo. Se guarda en la plantilla de funciones si se cumplen las condiciones siguientes:

- La variable de E/S se utiliza en los programas o visualizaciones incluidos en la plantilla de funciones.
- En la plantilla de funciones no puede incluirse el dispositivo de campo o el módulo de E/S a los cuales está asignada la variable de E/S.

Si se crea una función de aplicación desde la plantilla de funciones *(véase página 855)*, se puede asignar una variable de E/S que se haya guardado en la plantilla de funciones a un canal de E/S existente.

La variable de E/S incluye una descripción que se visualiza en el cuadro de diálogo **Añadir función a partir de una plantilla**.

Esta descripción se crea del siguiente modo:

- Si la variable de E/S se creó en la ficha **Asignación E/S** del editor de dispositivos *(véase página 157)*, la descripción se toma de la descripción del canal de E/S (esto sólo se aplica si la descripción original ha cambiado).
- Si la variable de E/S es una referencia a una variable existente, la descripción se toma del comentario de esa variable.

Parámetros de plantilla

Un parámetro de plantilla es una variable con un valor inicial ajustable.

Ejemplo: Si el dispositivo se utiliza a través de un bloque de funciones de comunicación, deberá asignar la dirección del dispositivo a ese bloque de funciones como parámetro de entrada. Para poder configurar esta dirección, conecte una variable al bloque de funciones y defina la variable como parámetro de plantilla.

La variable puede convertirse en un parámetro de plantilla si se cumplen las condiciones siguientes:

- La variable está definida en un programa o en una lista de variables globales que se incluyen en la plantilla de funciones.
- La variable tiene un tipo de datos simple (BOOL, cualquier tipo de dato numérico, cualquier cadena STRING, tipos de alias basados en un tipo de datos simple).
- El valor inicial de la variable está definido explícitamente como valor literal.

Todas las variables que cumplen esas condiciones se pueden seleccionar como parámetro de plantilla cuando se guarda la plantilla de funciones *(véase página 868)*.

Si se seleccionó una variable como parámetro de plantilla, el valor inicial de esa variable puede ajustarse cuando se crea una nueva función de aplicación desde la plantilla de funciones *(véase página 855)*.

Objetos guardados en plantillas de funciones

Los objetos siguientes se guardan en plantillas de funciones:

- todos los programas directamente ubicados en la carpeta de funciones de aplicaciones, así como sus subobjetos
- todas las listas de variables globales directamente ubicadas en la carpeta de funciones de aplicaciones
- todas las visualizaciones directamente ubicadas en la carpeta de funciones de aplicaciones
- todas las tablas CAM directamente ubicadas en la carpeta de funciones de aplicaciones
- todas las trazas directamente ubicadas en la carpeta de funciones de aplicaciones
- todos los dispositivos de campo y módulos de E/S utilizados por cualquier programa o visualización incluidos en la plantilla de funciones
- todas las variables globales cuyas listas de variables no formen parte de la plantilla de funciones pero sean utilizadas por cualquier programa o visualización que forme parte de ella
- todas las variables persistentes utilizadas por cualquier programa o visualización que forme parte de la plantilla de funciones

NOTA: No se guardará en la plantilla de funciones ningún otro tipo de objeto (aunque se guarde en la carpeta de funciones de aplicaciones). Utilice únicamente los bloques de funciones y tipos de datos almacenados en una biblioteca.

Pasos para crear una plantilla de funciones

Descripción general

SoMachine proporciona 2 maneras de crear una plantilla de funciones:

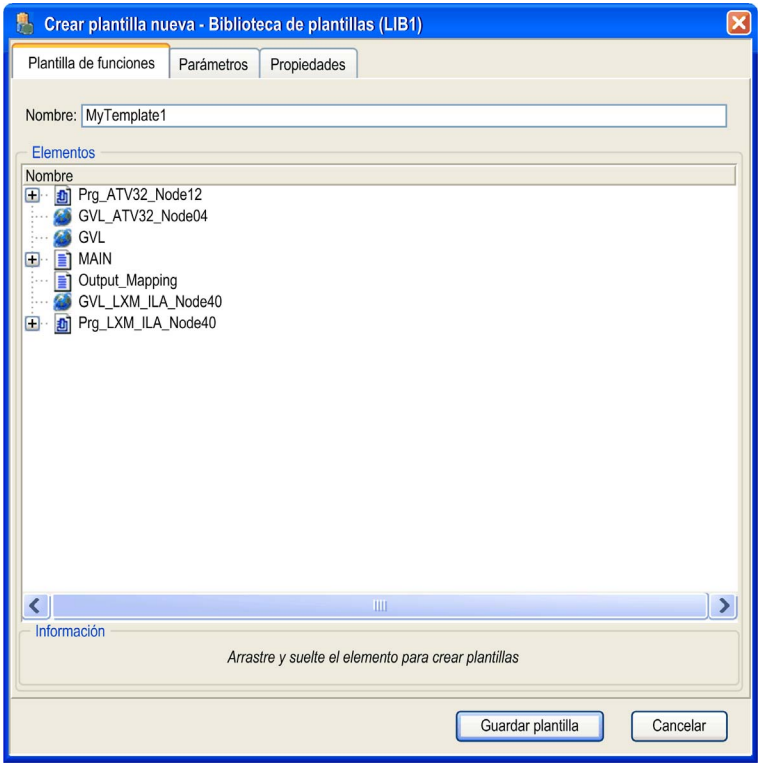
- En la vista **Macros**, con el cuadro de diálogo **Crear plantilla nueva**.
- En el árbol **Aplicaciones**, con el cuadro de diálogo **Guardar como plantilla de funciones**.

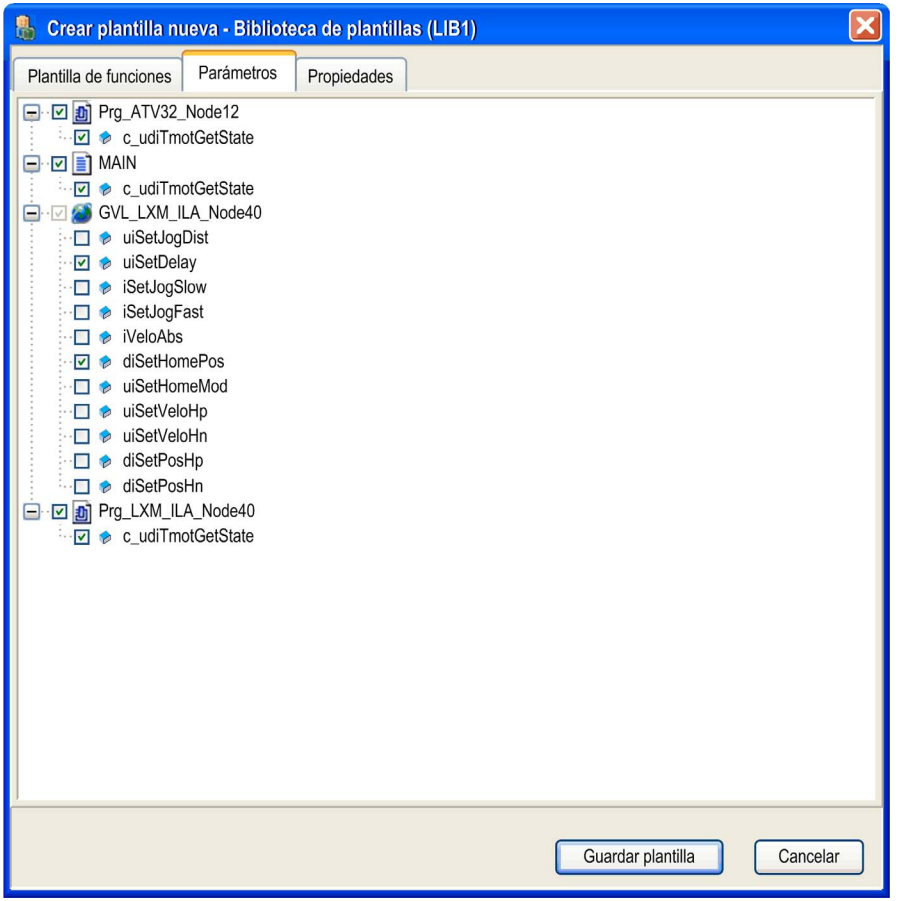
En los párrafos siguientes se enumeran los pasos necesarios para guardar como plantillas de funciones las funciones de aplicaciones ya disponibles que cumplan los criterios expuestos en el apartado *Funciones de aplicaciones como base para plantillas de funciones* *(véase página 861)*.

Procedimiento mediante la vista Macros

El procedimiento mediante la vista **Macros** permite crear su propia plantilla de funciones arrastrando y soltando elementos:

Paso	Acción
1	En la vista **Macros**, expanda la sección **Mi plantilla**.
2	Seleccione el nodo **Mi plantilla** y haga clic en el signo más de color verde. **Resultado:** Se inserta un nodo nuevo con el nombre predeterminado **LIB1** bajo el nodo **Mi plantilla**.

Paso	Acción
3	Seleccione el nodo **LIB1** y haga clic en el signo más de color verde. **Resultado:** Se muestra el cuadro de diálogo **Crear plantilla nueva**. 
4	En la ficha **Plantilla de funciones** del cuadro de diálogo **Crear plantilla nueva**, en **Nombre**, introduzca un nombre para la plantilla de funciones. Arrastre los elementos que desea incluir en la plantilla de funciones desde **Aplicaciones** hasta el cuadro **Elementos** de la ficha **Plantilla de funciones**. Los elementos enumerados en este cuadro se insertan en la plantilla de funciones. **NOTA:** Los elementos deben pertenecer a la misma aplicación.

Paso	Acción
5	La ficha **Parámetros** del cuadro de diálogo **Crear plantilla nueva** muestra las variables que están incluidas en los elementos que ha seleccionado en la ficha **Plantilla de funciones**.  En la lista de variables, seleccione aquellas que desee declarar como parámetros de plantilla; para ello, seleccione la casilla de la variable o de un nodo.
6	La ficha **Propiedades** del cuadro de diálogo **Crear plantilla nueva** permite añadir más información a la plantilla de funciones. Puede insertar un enlace a la ayuda online de esta plantilla de funciones. El cuadro de diálogo le permite añadir texto informativo adicional que se puede traducir; también puede añadir un gráfico a modo de ilustración de esta plantilla de funciones. Para obtener una descripción de estos parámetros, consulte el capítulo *Adición de información para plantillas o bibliotecas de plantillas* *(véase página 830)*.
7	Haga clic en el botón **Guardar plantilla**.

Procedimiento mediante el árbol Aplicaciones

Para guardar una función de aplicación ya disponible como plantilla de funciones, haga lo siguiente:

Paso	Acción
1	Haga clic con el botón derecho del ratón en una subcarpeta del nodo **Aplicación** en el árbol **Aplicaciones**.
2	Seleccione el comando **Guardar como plantilla de funciones** en el menú contextual. **Resultado:** SoMachine genera la aplicación automáticamente. Una vez que se haya completado este proceso correctamente, aparecerá el cuadro de diálogo **Guardar como plantilla de funciones**. 
3	Defina la nueva plantilla de funciones tal como se indica a continuación.
4	Haga clic en **Aceptar** para cerrar el cuadro de diálogo **Guardar como plantilla de funciones** y crear la nueva plantilla de funciones. **Resultado**: SoMachine comprobará que se pueda crear la plantilla de funciones y mostrará un mensaje para indicar que se ha creado correctamente o indicar los errores detectados.

Asignación de un nombre de plantilla

En el cuadro de texto **Nombre de plantilla** del cuadro de diálogo **Guardar como plantilla de funciones**, defina el nombre con el que se guardará la plantilla de funciones en la biblioteca de plantillas. De forma predeterminada, este cuadro de texto contiene el nombre de la carpeta en la que se encuentra la función de aplicación en el árbol **Aplicaciones**, aunque puede adaptar el nombre a sus requisitos individuales.

Selección de la biblioteca de plantillas

Para seleccionar una de las bibliotecas de plantillas previamente instaladas o creadas en la que se almacenará la nueva plantilla de funciones, haga lo siguiente:

Paso	Acción
1	En el cuadro de diálogo **Guardar como plantilla de funciones**, haga clic en el botón **...** situado al lado del cuadro de texto **Biblioteca de plantillas**. **Resultado**: Aparecerá el cuadro de diálogo **Seleccionar la biblioteca de plantillas**.
2	En el cuadro de diálogo **Seleccionar la biblioteca de plantillas** se muestran todas las bibliotecas de plantillas que se han instalado para el proyecto actual o que se han creado. No se muestran las bibliotecas protegidas contra escritura. Para añadir una plantilla de funciones nueva a una de estas bibliotecas de plantillas, seleccione la entrada apropiada y haga clic en **Aceptar**.

Selección de variables como parámetros

Puede definir las variables de la plantilla de funciones como parámetros de la plantilla *(véase página 862)*.

Para definir las variables de la plantilla de funciones como parámetros de la plantilla, haga lo siguiente:

Paso	Acción
1	En el cuadro de diálogo **Guardar como plantilla de funciones**, haga clic en el botón **...** situado a la derecha del cuadro de texto **Parámetros**. **Resultado**: Aparece el cuadro de diálogo **Seleccionar variables como parámetros**. Muestra las variables que están definidas en la aplicación seleccionada.
2	Seleccione la casilla de una variable determinada para seleccionarla como parámetro de la plantilla de funciones. Como alternativa, seleccione la casilla de un nodo raíz para seleccionar todos los elementos que hay bajo dicho nodo.
3	Haga clic en el botón **Aceptar**. **Resultado**: Las variables seleccionadas se muestran en el cuadro de texto **Parámetros** del cuadro de diálogo **Guardar como plantilla de funciones**. Se muestran en la tabla **Parámetros** del cuadro de diálogo **Añadir función a partir de una plantilla**, donde puede asignar valores nuevos para estos parámetros mediante **Nuevo valor**.

Sobreescritura de una plantilla de funciones existente

Para sobrescribir una plantilla de funciones existente con la función de aplicaciones seleccionada, haga lo siguiente:

Paso	Acción
1	En el cuadro de diálogo **Guardar como plantilla de funciones**, haga clic en el botón **...** situado a la derecha del cuadro de texto **Nombre de plantilla**.
2	Busque la plantilla de funciones ya disponible que desee reemplazar.
3	Seleccione la plantilla de funciones que desee reemplazar. **Resultado**: El nombre de esta plantilla de funciones se insertará en el cuadro de texto **Nombre de plantilla** y el nombre de la biblioteca de plantillas en la que se almacene se insertará en el cuadro de texto **Biblioteca de plantillas**.
4	Haga clic en **Aceptar** para cerrar el cuadro de diálogo **Guardar como plantilla de funciones** y reemplazar la plantilla de funciones seleccionada por la función de aplicación nueva.

Adición de más información a la nueva plantilla de funciones

Para añadir más información a la plantilla de funciones nueva, haga clic en el botón **Propiedades...** Se abre el cuadro de diálogo **Propiedades**. Permite introducir más información para la plantilla de funciones. Puesto que este cuadro de diálogo es idéntico al de las plantillas de dispositivos y las bibliotecas de plantillas, consulte la descripción en el capítulo Adición de información para plantillas o bibliotecas de plantillas *(véase página 830)*.

Parte IX
Solución de problemas y FAQ

Contenido de esta parte

Esta parte contiene los siguientes capítulos:

Capítulo 35
Genérico - Solución de problemas y FAQ

Sección 35.1
Preguntas frecuentes

Contenido de esta sección

Esta sección contiene los siguientes apartados:

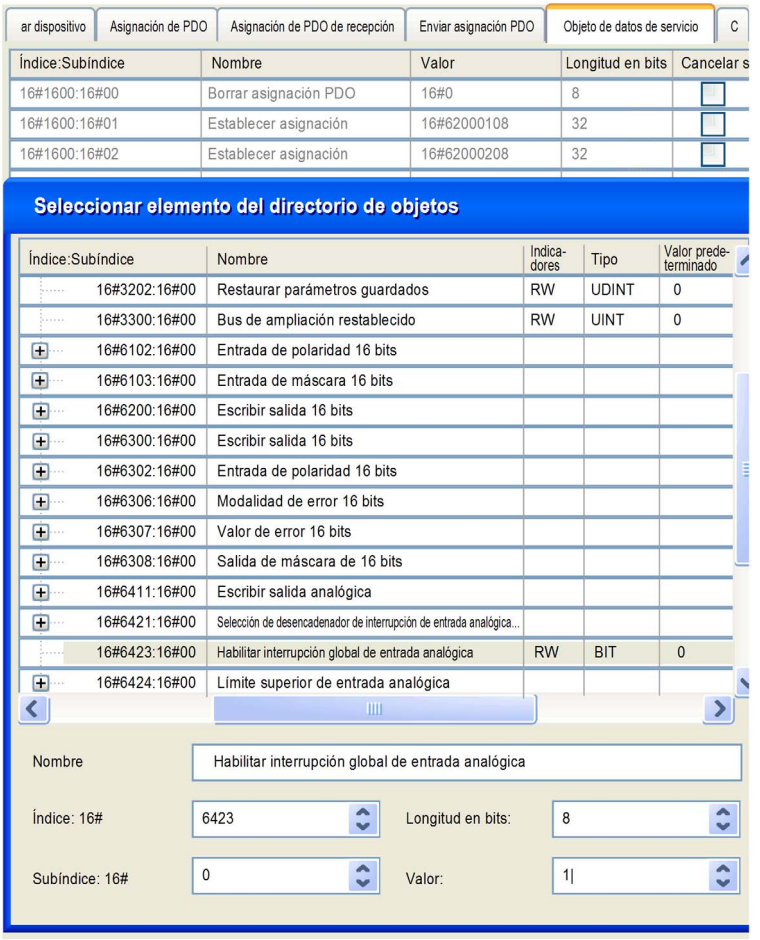
¿Cómo puedo habilitar y configurar las entradas analógicas en CANopen?

Descripción general

En esta sección se proporcionan instrucciones para habilitar entradas analógicas según el estándar CANopen configurando el SDO (Service Data Object) 6423 con el valor `1`.

Procedimiento

Paso	Acción
1	Haga doble clic en el nodo del dispositivo CANopen analógico en el árbol **Dispositivos**.
2	En la ficha **Dispositivo remoto CANopen** del editor, active la opción **Activar configuraciones de experto**. **Resultado**: Aparecen fichas adicionales y se completa la ficha **Service Data Object** con información.

Paso	Acción
3	Abra la ficha **Service Data Object** y haga clic en el botón **Nuevo...** **Resultado**: Aparece el cuadro de diálogo **Seleccione una entrada del directorio de objetos**. 
4	En la lista de objetos, seleccione el objeto **6423**, escriba `1` en **Valor** y haga clic en **Aceptar**. **Resultado**: Se activa la transmisión de las entradas analógicas en el bus CANopen. Ahora podrá configurar parámetros de los valores analógicos tal como se describe en el manual de hardware del dispositivo.

¿Por qué a veces la velocidad de arranque de SoMachine es más lenta?

Descripción general

Además de la configuración del PC, hay otros factores que pueden aumentar el tiempo de arranque de SoMachine:

Fase de arranque	Velocidad de arranque
Primer inicio después de instalar SoMachine	En el primer inicio después de haber instalado SoMachine, el software generará su propio entorno de trabajo en el PC. Esta acción sólo se realiza una vez. Aun así, esta acción afecta considerablemente a la duración del primer arranque.
Primer inicio después del reinicio	Una vez que haya reiniciado el PC, el arranque de SoMachine puede tardar más de lo habitual, ya que Microsoft Windows inicia diversos servicios en segundo plano necesarios para ejecutar SoMachine. Esto puede influir en la duración del arranque y es inevitable.
Inicios posteriores	Los usuarios obtendrán una mayor velocidad en el arranque cuando el sistema se haya iniciado anteriormente en el PC.

¿Cómo puedo restaurar la configuración predeterminada para los métodos abreviados y menús?

Descripción general

Los menús y métodos abreviados del software SoMachine difieren según el estado actual, es decir, la ventana o el editor que esté abierto.

Puede adaptar los métodos abreviados y menús a sus preferencias individuales, o bien puede cargar los métodos abreviados y menús estándar de SoMachine o CoDeSys, como se describe en las siguientes secciones.

Personalización de métodos abreviados y menús

Si desea adaptar los métodos abreviados y menús a sus preferencias individuales, utilice el menú **Herramientas** →**Personalizar**.

Restauración de las teclas de método abreviado y los menús estándar de SoMachine

Para restaurar las teclas de método abreviado y los menús estándar de SoMachine (después de personalizarlos), haga lo siguiente:

Paso	Acción
1	Ejecute el comando **Personalizar** del menú **Herramientas**. **Resultado**: Aparecerá el cuadro de diálogo **Personalizar**.
2	En el cuadro de diálogo **Personalizar**, haga clic en el botón **Cargar...** **Resultado**: Aparecerá el cuadro de diálogo **Cargar menú**.
3	En el cuadro de diálogo **Cargar menú**, busque la carpeta *...\Archivos de programa\Schneider Electric\SoMachine Software\V4.0\LogicBuilder\Settings*, seleccione el archivo *Standard.opt.menu* y haga clic en **Abrir**. **Resultado**: Ahora el cuadro de diálogo **Personalizar** muestra la configuración estándar de SoMachine.
4	Para cargar esta configuración estándar en la interfaz gráfica de usuario de SoMachine, haga clic en **Aceptar**.

Selección de las teclas de método abreviado y los menús estándar de CoDeSys

Para importar las teclas de método abreviado y los menús de CoDeSys a la interfaz gráfica de usuario de SoMachine, haga lo siguiente:

Paso	Acción
1	Ejecute el comando **Personalizar** del menú **Herramientas**. **Resultado**: Aparecerá el cuadro de diálogo **Personalizar**.
2	En el cuadro de diálogo **Personalizar**, haga clic en el botón **Cargar**. **Resultado**: Aparecerá el cuadro de diálogo **Cargar menú**.
3	En el cuadro de diálogo **Cargar menú**, busque la carpeta *...\Archivos de programa\Schneider Electric\SoMachine Software\V4.0\LogicBuilder\Settings\OriginalCoDeSys*, seleccione el archivo *Standard.opt.menu* y haga clic en **Abrir**. **Resultado**: Ahora, en el cuadro de diálogo **Personalizar** aparecerá la configuración de CoDeSys.
4	Para cargar esta configuración de CoDeSys en la interfaz gráfica de usuario de SoMachine, haga clic en **Aceptar**.

NOTA: Los menús y las teclas de método abreviado del software de SoMachine varían según la ventana o el editor que esté abierto actualmente.

Expansión de menús

Los menús principales y contextuales de SoMachine pueden verse expandidos o contraídos. En el modo contraído se ocultan los comandos usados con menor frecuencia o los deshabilitados. Al pulsar la flecha ▼ del final del menú, el menú correspondiente se expande, con lo que se muestran todos sus elementos.

Si desea que se muestren siempre los menús en modo completo, active la opción **Mostrar siempre menús completos** en el cuadro de diálogo **Herramientas** →**Opciones** → **Características**.

Capítulo 36
Acceso a controladores - Resolución de problemas y preguntas frecuentes

Contenido de este capítulo

Este capítulo contiene las siguientes secciones:

Sección 36.1
Resolución de problemas: Acceso a nuevos controladores

Contenido de esta sección

Esta sección contiene los siguientes apartados:

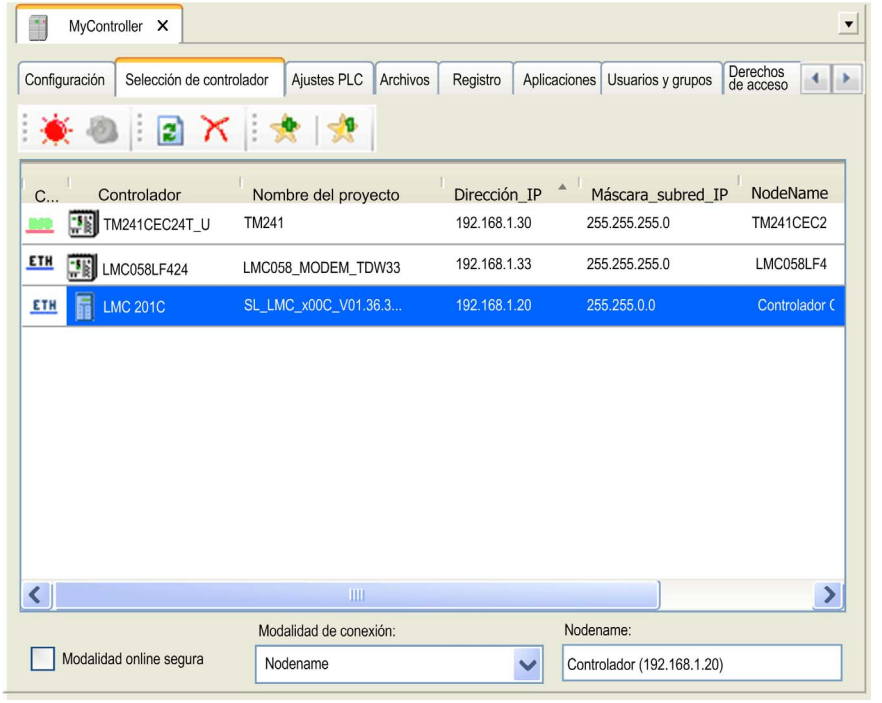
Acceso a nuevos controladores

Descripción general

Para acceder a un nuevo controlador, adapte la configuración de red del controlador a la red del PC de SoMachine. En este capítulo se ofrece un ejemplo paso a paso.

Ejemplo

En este ejemplo se indican los pasos para acceder a un LMC058 con dirección IP 192.168.1.33 desde un PC con la dirección IP 192.168.1.10 que reside en la subred 255.255.255.0.

Paso	Acción
1	Conecte el controlador directamente al PC que ejecuta SoMachine o a la red del PC mediante un cable Ethernet.
2	En SoMachine, abra la vista **Selección de controlador** del editor de dispositivos *(véase página 103)*. **Resultado:** El controlador LMC058 se incluirá en la lista. 

Paso	Acción
3	Para adaptar la configuración de comunicación del controlador, haga clic con el botón derecho en el controlador en la lista **Selección de controlador** y ejecute el comando **Procesar configuración de la comunicación...** en el menú contextual. **Resultado:** Se abrirá el cuadro de diálogo **Procesar configuración de la comunicación**.
4	En el cuadro de diálogo **Procesar configuración de la comunicación**, escriba una **dirección IP** libre que esté disponible en su red. Al configurar direcciones IP, consulte el mensaje de peligro mostrado más abajo.
5	Haga clic en **Aceptar** para confirmar el cuadro de diálogo **Procesar configuración de la comunicación**.
6	En la vista **Selección de controlador**, conecte con el controlador.

Gestione las direcciones IP con cuidado debido a que cada dispositivo de la red necesita una dirección única. Si existen varios dispositivos con la misma dirección IP, puede producirse un funcionamiento impredecible en la red y el equipo asociado.

⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

- Asegúrese de que todos los dispositivos tengan direcciones exclusivas.
- Solicite su dirección IP al administrador del sistema.
- Confirme que la dirección IP del dispositivo es única antes de poner el sistema en funcionamiento.
- No asigne la misma dirección IP a ningún otro equipo de la red.
- Actualice la dirección IP después de clonar cualquier aplicación que incluya comunicaciones Ethernet a una dirección exclusiva.

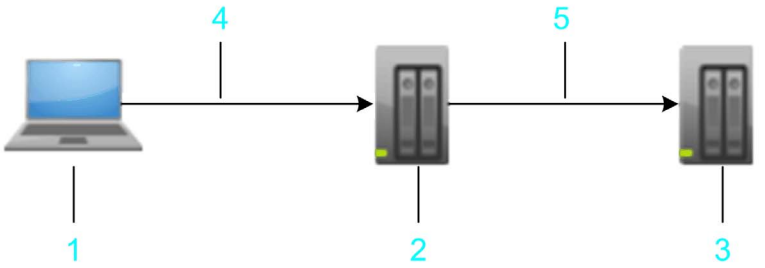
El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

NOTA: Algunos controladores admiten un parámetro que ayuda a evitar que se acceda a ellos remotamente (parámetro **RemoteCommunicationAccess** de los controladores LMC •0•C).

Conexión a través de una dirección IP e información de dirección

Descripción general

El protocolo de comunicación que se utiliza ofrece un mecanismo para conectar con un controlador independientemente del tipo de conexión. Por ejemplo, esta característica permite acceder a un controlador de destino que esté conectado mediante Ethernet a otro controlador de salto que está conectado a través de USB al PC.

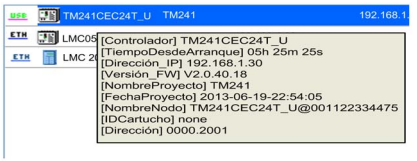


1 PC
2 controlador de salto
3 controlador de destino
4 USB
5 Ethernet

Información de dirección

En este ejemplo, el USB utiliza un protocolo diferente. Por tanto, normalmente no se puede utilizar una dirección IP para direccionar el controlador de destino. En su lugar, se utiliza la información de rutas de acceso, que describe la manera de conectar con el controlador de destino en uno o más saltos.

La información de rutas de acceso se muestra como información sobre herramientas de una entrada de la lista de controladores (en el ejemplo siguiente, **[Address] 0000.2001**):



NOTA: Como esta dirección sólo describe la manera en que se conecta con el controlador, puede cambiar con cada modificación de los PC locales o de la configuración del adaptador de red del controlador de salto. Por ejemplo, al activar o desactivar los adaptadores de red o al iniciar/detener los servicios que utilizan los adaptadores de red. El direccionamiento a un destino específico puede diferir según los diferentes PC de transmisión.

Nodename

Como el **Nodename** del controlador es un identificador estable en el sistema, se utiliza para identificar el destino.

Si se selecciona **Dirección IP** como **Modalidad de conexión**, se prueba para obtener la información del propio **Nodename**. Algunos controladores (como LMC •0•C) crean el **Nodename** automáticamente incluyendo la dirección IP. También puede configurar el **Nodename** por sus propios medios (como se describe en Preguntas frecuentes - ¿Por qué no se lista el controlador en la vista Selección de controlador? *(véase página 889)*) para permitir que el sistema encuentre un controlador por su dirección IP. Si en el Nodename falta la dirección IP, esta se intenta obtener de un controlador. Pero no todos los dispositivos o su versión de firmware actual admiten el servicio. En ese caso, utilice **Modalidad de conexión Nodename** para conectar o configurar un nombre de dispositivo que incluya la dirección IP entre paréntesis. Por ejemplo, **MyDevice (192.168.1.30)**.

Sección 36.2
Preguntas frecuentes: ¿Qué puedo hacer en caso de que haya problemas de conexión con el controlador?

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Preguntas frecuentes - ¿Por qué no se puede establecer una conexión con el controlador?

¿Por qué no se puede establecer una conexión con el controlador aunque la dirección IP parezca la adecuada?

Si ha establecido la dirección IP del controlador como se describe en el capítulo *Acceso a nuevos controladores* *(véase página 883)* y aun así no puede conectar con el controlador, puede que el motivo sea la máscara de subred. Como el protocolo de comunicación utilizado requiere una máscara de subred idéntica en el sitio del emisor y el del receptor, puede que un ping al controlador se ejecute correctamente, pero que no se pueda establecer una conexión.

Para resolver este problema, haga lo siguiente:

Paso	Acción
1	En SoMachine, abra la vista **Selección de controlador** del editor de dispositivos *(véase página 103)*.
2	Para adaptar la configuración de comunicación del controlador, haga clic con el botón derecho en el controlador en la lista **Selección de controlador** y ejecute el comando **Procesar configuración de la comunicación...** en el menú contextual. **Resultado:** Se abrirá el cuadro de diálogo **Procesar configuración de la comunicación**.
3	Adapte la **Máscara de subred** configurada para el controlador para que se ajuste exactamente a la máscara de subred de su PC con SoMachine. **Ejemplo:** Cambie **255.255.0.0** por **255.255.255.0**.

¿Por qué no aparece el controlador en la lista de la vista Configuración de comunicación?

Si establece una conexión entre el controlador y el PC con SoMachine utilizando la ruta activa, la vista **Configuración de comunicación** se muestra en el editor de dispositivos *(véase página 120)*. Este es el ajuste predeterminado para SoMachine V3.1 y versiones anteriores.

Si el controlador elegido no se muestra en la vista **Configuración de comunicación**, puede cambiar temporalmente a establecimiento de conexión mediante la dirección IP como se indica a continuación:

Paso	Acción
1	Abra el cuadro de diálogo **Configuración del proyecto →Configuración de comunicación** desde el menú **Proyecto**.
2	Seleccione la opción **Marque mediante "dirección IP"** y confirme el ajuste haciendo clic en **Aceptar**.
3	Adapte los ajustes de red del controlador a la red de su PC con SoMachine como se describe en el capítulo *Acceso a nuevos controladores (véase página 883)*.
4	Vuelva a definir el establecimiento de la conexión en la ruta activa. Ahora el controlador debe aparecer en la vista **Configuración de comunicación**.

¿Por qué no aparece el controlador en la lista de la vista Selección de controlador?

Si el controlador no aparece en la lista de la vista **Selección de controlador** del editor de dispositivos *(véase página 103)*, puede que haya 2 dispositivos diferentes asignados al mismo **Nodename**. Si hay 2 dispositivos asignados al mismo **Nodename**, sólo 1 de estos dispositivos aparece en la lista **Selección de controlador**.

Debe gestionar con precaución el **Nodename**, porque cada dispositivo de la red requiere un **Nodename** exclusivo. Si hay varios dispositivos con el mismo **Nodename**, puede producirse un funcionamiento imprevisible de la red y los equipos asociados.

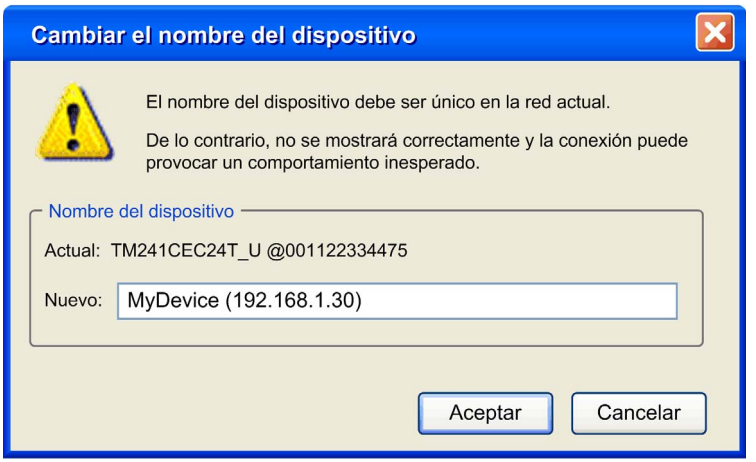
⚠ ADVERTENCIA

FUNCIONAMIENTO IMPREVISTO DEL EQUIPO

- Verifique que todos los dispositivos tengan un **Nodename** exclusivo antes de poner el sistema en funcionamiento.
- Actualice el **nodename** tras clonar cualquier aplicación que incluya comunicaciones Ethernet a un **nodename** exclusivo.

El incumplimiento de estas instrucciones puede causar la muerte, lesiones serias o daño al equipo.

Para cambiar el **Nodename** de un dispositivo, haga lo siguiente:

Paso	Acción
1	Haga clic con el botón derecho en el dispositivo que tiene asignado un **Nodename** duplicado en la lista **Selección de controlador** y ejecute el comando **Cambiar el nombre del dispositivo** en el menú contextual. **Resultado:** Se muestra el cuadro de diálogo **Cambiar el nombre del dispositivo**. 

Paso	Acción
2	En el cuadro de diálogo **Cambiar el nombre del dispositivo**, especifique un **Nodename** exclusivo en el cuadro de texto **Nuevo**.
3	Haga clic en **Aceptar** para confirmar y cerrar el cuadro de diálogo **Cambiar el nombre del dispositivo**.
4	En la vista **Selección de controlador**, haga clic en el botón **Actualizar** para actualizar la lista de dispositivos. **Resultado:** El segundo dispositivo con el mismo **Nodename** del dispositivo que acaba de cambiar aparecerá en la lista.
5	Repita los pasos del 1 al 4 hasta que haya eliminado todos los **Nodename** duplicados.

NOTA: Algunos controladores, como los controladores LMC •0•C, crean un **Nodename** automáticamente a partir del nombre de dispositivo del proyecto tras una descarga de proyecto y la dirección IP (por ejemplo, **MyLMC (192.168.1.30)**). Este nombre automático sobrescribe el **Nodename** asignado si se realizan cambios en el controlador.

Preguntas frecuentes - ¿Por qué se ha interrumpido la comunicación entre el PC y el controlador?

¿Por qué se ha interrumpido la comunicación entre el PC y el controlador?

Puede que sea necesario reiniciar la puerta de enlace de la siguiente forma:

Paso	Acción
1	Haga clic con el botón derecho del ratón en el icono **Gateway Tray Application** en la barra de tareas de Windows  .
2	Ejecute el comando **Restart Gateway** en el menú contextual.

Apéndices

Contenido de este anexo

Este anexo contiene los siguientes capítulos:

Apéndice A
Comunicación de red

Contenido de este capítulo

Este capítulo contiene los siguiente apartados:

Topología de red

Descripción general

La red de control de SoMachine es un sistema programado para configurarse a sí mismo (asignación de dirección) para admitir los medios de comunicación transparentes y enrutar paquetes entre redes diferentes. El mecanismo de enrutamiento es lo suficientemente simple para que cualquier nodo de la red, es decir, incluso los nodos con pocos recursos, puedan enrutar paquetes. Por ello, se evitan las tablas de enrutamiento grandes, los cálculos complejos y las solicitudes durante el tiempo de ejecución.

La red de control se configura jerárquicamente; es decir, cada nodo tiene un nodo padre y un número arbitrario de subobjetos. Un nodo que no tiene un nodo padre se llama nodo de nivel superior. No se permiten los ciclos; es decir, una red de control tiene una estructura de árbol.

Las relaciones objeto padre-subobjeto proceden de la especificación de segmentos de red. Un segmento de red corresponde, por ejemplo, a una Ethernet local o a una conexión punto a punto serie. Diferencia entre la red principal (mainnet) y las subredes (subnet). Cada nodo tiene, como máximo, una red principal, en la que se espera que esté el objeto padre. Para cada nodo se puede configurar un número arbitrario de subredes. El nodo actúa como objeto padre de todos ellos.

Si un segmento de red se ha definido al mismo tiempo que una subred de varios nodos, la red tendrá varios objetos padre. Sin embargo, la configuración resultante no será válida, ya que cada segmento de red solamente puede tener un objeto padre.

Direccionamiento y enrutamiento

Descripción general

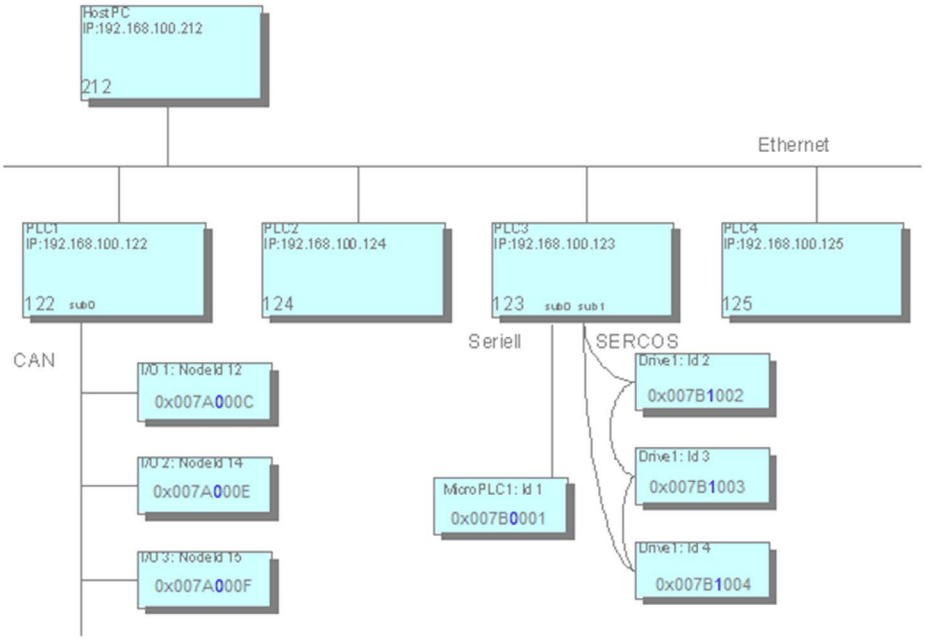
El direccionamiento correlaciona la topología de la red de control con una dirección exclusiva. Una dirección de nodo *(véase página 900)* está estructurada jerárquicamente.

Por cada conexión de red, se asigna una dirección local que identifica el nodo de forma exclusiva en su red local respectiva mediante el controlador de bloque correspondiente. En el caso de la dirección de nodo completa, esta dirección local va precedida por el índice de subred a la que está asignada la red local por parte del objeto padre. Además, debe ir precedida de la dirección de nodo de su objeto padre.

La longitud del índice de subred (en bits) está determinada por el dispositivo, mientras que la longitud de la dirección local está determinada por el tipo de red.

Un nodo sin red principal es un nodo de nivel superior con la dirección 0. Un nodo con una red principal que no contenga un objeto padre también será un nodo de nivel superior, y se asignará a su dirección local en la red principal.

Ejemplo: Red principal y subredes



En el ejemplo, las direcciones de los nodos secundarios se proporcionan en representación hexadecimal. Los 4 primeros dígitos representan la dirección del objeto padre en cuestión en la red principal. Por ejemplo, 0x007A=122 para PLC1. El siguiente byte (mostrado en azul) está reservado para el índice de subred y va seguido de la dirección local, por ejemplo, C=12 para el nodo ID 12.

Debido a la estructuración de la dirección, el algoritmo de enrutamiento puede mantenerse relativamente simple. Por ejemplo, no se necesitan tablas de enrutamiento. Se requiere información localmente, sobre la propia dirección y sobre la dirección del nodo padre.

A partir de entonces, un nodo puede gestionar los paquetes de datos de forma correcta.

- Si la dirección de destino es igual a la dirección del nodo actual, se determina como receptor.
- Si la dirección de destino empieza con la dirección del nodo actual, el paquete va destinado a un objeto secundario o descendente del nodo y debe reenviarse.
- En caso contrario, el receptor no es descendiente del nodo actual. El paquete debe reenviarse a su propio objeto padre.

Direccionamiento relativo

El direccionamiento relativo es una característica especial. Las direcciones relativas *(véase página 901)* no contienen el número de nodo del nodo receptor, sino que describen directamente la ruta del emisor al receptor. El principio es similar a la ruta relativa en el sistema de archivos. La dirección consta del número de pasos que debe desplazarse hacia arriba el paquete, es decir, al siguiente objeto padre respectivo, y la ruta descendente posterior al nodo de destino.

La ventaja del direccionamiento relativo es que 2 nodos del mismo subárbol pueden continuar la comunicación cuando se mueve todo el subárbol a otra posición en la red de control general. Si bien las direcciones absolutas de los nodos cambiarán debido a la reubicación, las direcciones relativas son todavía válidas.

Determinación de direcciones

Un nodo intenta determinar su propia dirección como procedente de su objeto padre o de sí mismo como nodo de nivel superior. Con este fin, un nodo enviará una determinación de la dirección en forma de mensaje de difusión a la red principal durante el arranque. Mientras no se responda a este mensaje, el nodo se considerará un nodo de nivel superior, aunque seguirá intentando detectar un nodo padre. Un nodo padre responderá mediante una notificación de dirección. A continuación, el nodo completará su propia dirección y la proporcionará a las subredes.

La determinación de la dirección puede ejecutarse durante el arranque o a petición del PC de programación.

Estructura de direcciones

Descripción general

A continuación se muestra una descripción detallada sobre la estructura de los siguientes tipos de dirección:

- Direcciones de red *(véase página 899)*
- Direcciones de nodo *(véase página 900)*
- Direcciones absolutas y relativas *(véase página 901)*
- Direcciones de difusión *(véase página 902)*

Direcciones de red

Las direcciones de red representan una asignación de direcciones de un tipo de red (por ejemplo, direcciones IP) a direcciones lógicas dentro de una red de control. Esta asignación se gestiona mediante el controlador de bloque respectivo. En una Ethernet con direcciones IP de clase C, los 3 primeros bytes de la dirección IP son los mismos para todos los dispositivos de la red. Por tanto, los últimos 8 bits de la dirección IP son suficientes como dirección de red puesto que permiten una asignación inequívoca entre las 2 direcciones en el controlador de bloque.

Un nodo cuenta con direcciones de red distintas para cada conexión de red. Las diferentes conexiones de red pueden contar con la misma dirección de red puesto que esta dirección debe ser exclusiva sólo de forma local para cada conexión de red.

Terminología: en general, la dirección de red de un nodo sin una instrucción de la conexión de red hace referencia a la dirección de red de la red principal.

La longitud de una dirección de red se especifica en bits y la puede elegir el controlador de bloque según corresponda. En un segmento de red, se debe usar la misma longitud para todos los nodos.

Una dirección de red se representa en forma de matriz de bytes con la siguiente codificación:

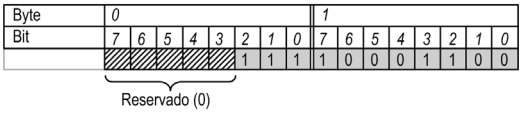
- Longitud de la dirección de red: n bits
- Bytes necesarios: b = (n + 7) DIV 8
- Los (n MOD 8) bits de orden más bajo del primer byte y los (n DIV 8) bytes restantes se utilizan para la dirección de red.

Ejemplo de dirección de red

Longitud: 11 bits

Dirección: 111 1000 1100

Ejemplo de codificación de dirección de red



Direcciones de nodo

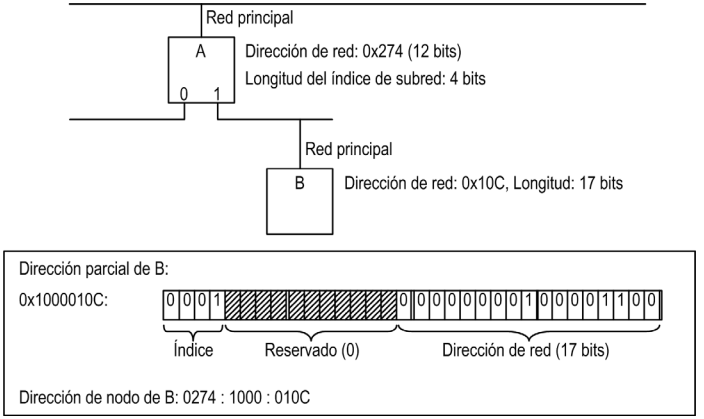
La dirección de nodo indica la dirección absoluta de un nodo en una red de control y, por tanto, es exclusiva en todo el árbol. La dirección consta de hasta 15 componentes de dirección, cada uno de los cuales está formado por 2 bytes. Cuanto más bajo se encuentre un nodo en la jerarquía de red, más larga será su dirección.

La dirección de nodo consta de las direcciones parciales de todos los predecesores del nodo y del mismo nodo. Cada dirección parcial consta de uno o varios componentes de dirección. Por consiguiente, la longitud es siempre un múltiplo de 2. La dirección parcial de un nodo se compone a partir de la dirección de red del nodo en su red principal y del índice de subred de la red principal en el nodo padre. El enrutador del nodo principal determina los bits que se necesitan para el índice de subred. Entre el índice de subred y la dirección de red se insertan bits de relleno para garantizar que la longitud de la dirección parcial sea un múltiplo de 2 bytes.

Casos especiales:

- El nodo tiene una red principal: esto significa que no hay ningún índice de subred ni una dirección de red en la red principal. En este caso, la dirección tiene el valor 0x0000.
- Nodo con red principal pero sin padre: en este caso, se da por sentado un índice de subred con una longitud de 0 bits. La dirección parcial se corresponde con una dirección de red, complementada con los bits de relleno, si es necesario.

Ejemplo de dirección de nodo



La representación de la dirección de nodo es siempre hexadecimal. Los componentes de dirección individuales (2 bytes en cada caso) están separados por dos puntos (**:**). Los bytes en un componente se muestran secuencialmente sin un separador (consulte el ejemplo anterior). Puesto que esto representa una matriz de bytes y no un valor de 16 bits, los componentes no se muestran en formato little-endian. En el caso de las direcciones introducidas manualmente, los dígitos que falten en un componente de dirección se rellenan con ceros no significativos desde la izquierda: 274 = 0274. Para mejorar la legibilidad, la salida siempre debe incluir ceros no significativos.

Direcciones absolutas y relativas

La comunicación entre 2 nodos se puede basar en las direcciones relativas o absolutas. Las direcciones absolutas son idénticas a las direcciones de nodo. Las direcciones relativas especifican una ruta desde el emisor hasta el receptor. Constan de un offset de dirección y una ruta descendente al receptor.

El offset de dirección (negativo) describe el número de componentes de dirección que un paquete debe subir en el árbol de dirección antes de que se pueda bajar otra vez desde un padre común. Puesto que los nodos pueden usar direcciones parciales que consten de más de un componente de dirección, el número de nodos padre que se deben transferir siempre será igual al offset de dirección, lo que significa que la demarcación entre los nodos padres ya no será inequívoca. Por este motivo, la parte inicial común de las direcciones de los interlocutores de comunicación se utiliza como dirección padre. Cada componente de dirección se cuenta como un paso ascendente, con independencia de los nodos padre reales. El nodo padre correspondiente detecta cualquier error introducido a raíz de estos supuestos y lo debe gestionar correctamente.

Al llegar al padre común, la ruta relativa (una matriz de componentes de dirección) se sigue hacia abajo de la forma habitual.

Formal: la dirección de red del receptor se forma eliminando los últimos componentes de offset de dirección de la dirección de nodo del emisor y agregando la ruta relativa a la dirección restante.

Ejemplo

En el ejemplo, una letra representará un componente de dirección, mientras que un punto separará los nodos concretos. Puesto que un nodo puede tener varios componentes de dirección, se permite que pueda tener varias letras en el ejemplo.

Nodo A: a.bc.d.ef.g

Nodo B: a.bc.i.j.kl.m

- Dirección del padre común más bajo: a. bc
- Direcciones relativas de la A a la B: -4/i.j.kl.m (el número -4 se obtiene de los 4 componentes d, e, f y g; por tanto, el paquete se debe subir).

La dirección relativa se debe ajustar con cada paso a través de un nodo intermedio. Basta con ajustar el offset de dirección. El nodo padre siempre se encarga de ello: si un nodo padre recibe un paquete de una de sus subredes, el offset de dirección debe incrementarse en la longitud del componente de dirección de esta subred.

- Si el nuevo offset de dirección es < 0, el paquete debe reenviarse al nodo padre.
- Si el offset de dirección 1 ≥ 0, el paquete debe reenviarse al nodo hijo cuya dirección local se encuentra en la posición descrita por el offset de dirección en la dirección relativa. En primer lugar, el offset de dirección debe incrementarse en la longitud del nodo hijo para garantizar que el nodo vea una dirección correcta.

Surge una situación especial cuando se produce el error descrito anteriormente al determinar el padre común. En este caso, el offset de dirección en el padre común "real" es negativo, pero la magnitud es mayor que la longitud de la dirección parcial de la subred de la que procede el paquete. El nodo debe detectar este caso, calcular la dirección local del siguiente nodo hijo a partir de la dirección del nodo anterior y la diferencia de longitud, y adaptar el offset de dirección, de modo que el siguiente nodo vea una dirección relativa correcta. Además, los componentes de dirección no cambiarán, sólo se modificará el offset de dirección.

Direcciones de difusión

Hay 2 tipos de difusión: global y local. Una difusión global se envía a todos los nodos de una red de control. La dirección de nodo vacía (longitud 0) se reserva para este fin.

Las difusiones locales se envían a todos los dispositivos de un segmento de red. Para este fin, todos los bits de la dirección de red tienen el valor 1. Esto es posible tanto en las direcciones relativas como en las absolutas.

Un controlador de bloque debe poder gestionar las dos direcciones de difusión, es decir, las direcciones de red vacías y las direcciones de red con todos los bits establecidos en 1 se deben interpretar y enviar como una difusión.

Apéndice B
Uso del servidor OPC 3

Descripción general

La descripción proporcionada en este capítulo está dirigida a personas con experiencia en la tecnología del servidor OPC.

El archivo *OPC_V3_how_to_use_E.pdf* , que se instala automáticamente con SoMachine en el directorio *C:\Archivos de programa\Schneider Electric\SoMachine OPCServer*, proporciona una descripción más detallada acerca de cómo configurar el servidor OPC.

Contenido de este capítulo

Este capítulo contiene los siguiente apartados:

Información general

Descripción de OPC

OPC es una interfaz estandarizada para acceder a los datos de proceso. Se basa en el estándar de Microsoft COM/DCOM2 (Component Object Model / Distributed COM) que se ha ampliado debido a los requisitos de acceso a los datos en la automatización, en los que la interfaz se utiliza principalmente para leer datos del controlador o escribir datos en él.

Ejemplos de clientes OPC típicos
- visualizaciones
- programas cuyo objetivo es recopilar datos de funcionamiento

Ejemplos de proveedores típicos de servidores OPC
- sistemas de controlador
- tarjetas de interfaz de bus de campo

Descripción de un servidor OPC

El servidor OPC es un programa ejecutable que se inicia automáticamente durante el establecimiento de una conexión entre el cliente y el controlador. Por tanto, el servidor OPC puede informar al cliente acerca de estados o valores de variables modificados.

El servidor OPC proporciona todas las variables (denominadas **Objetos** en OPC) que están disponibles en el controlador (**Item Pool** o **Address Space**). Estos elementos se administran desde una **caché de datos** que ayuda a garantizar un acceso rápido a sus valores. También es posible el acceso directo, sin memoria caché, a los elementos del controlador.

En el servidor OPC, los elementos pueden organizarse en **Grupos** (**Privado** y **Público**).

Los grupos privados pueden estar compuestos en el cliente arbitrariamente a partir de elementos específicos. Inicialmente no afectan a las agrupaciones en el servidor OPC pero, si es necesario, pueden transformarse en grupos públicos. Por ejemplo, el trabajo con grupos privados es útil para activar o desactivar determinados grupos de variables con un solo comando, dependiendo de si deben estar accesibles o no.

Los datos agrupados deben leerse desde el servidor OPC coherentemente; es decir, todas las variables deben leerse al mismo tiempo. Sin embargo, esto no siempre es posible en el caso de sistemas de destino con búferes de comunicación restringidos.

Debido a las características de COM/DCOM, se puede acceder a un OPC que se ejecuta en otro ordenador. También se permite que más de un cliente acceda al origen de datos al mismo tiempo. Otra ventaja es la posibilidad de aplicación de diferentes lenguajes (C++, Visual Basic, Delphi, Java).

Descripción general del servidor OPC 3 CoDeSys

El servidor OPC CoDeSys se basa en el PLCHandler de 3S - Smart Software Solutions GmbH. Este módulo de comunicaciones permite una comunicación directa para aquellos controladores que pueden programarse con CoDeSys.

El servidor OPC V.3 o posterior admite las siguientes especificaciones OPC:

- Definiciones e interfaces comunes de OPC Versión 1.0
- Interfaz estándar personalizada de acceso a datos Versión 1.0
- Interfaz estándar personalizada de acceso a datos Versión 2.05A
- Interfaz estándar personalizada de acceso a datos Versión 3.0
- Interfaz estándar de automatización de acceso a datos Versión 2.0

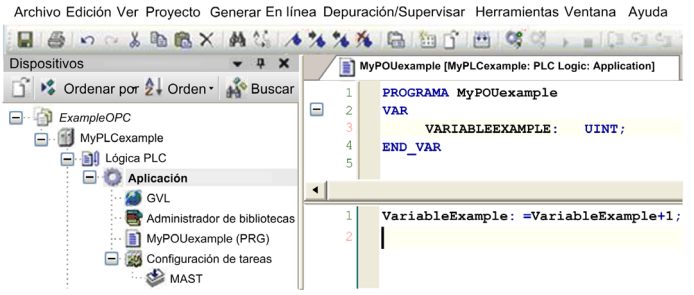
La comunicación entre el servidor OPC y el controlador puede llevarse a cabo mediante la siguiente interfaz:

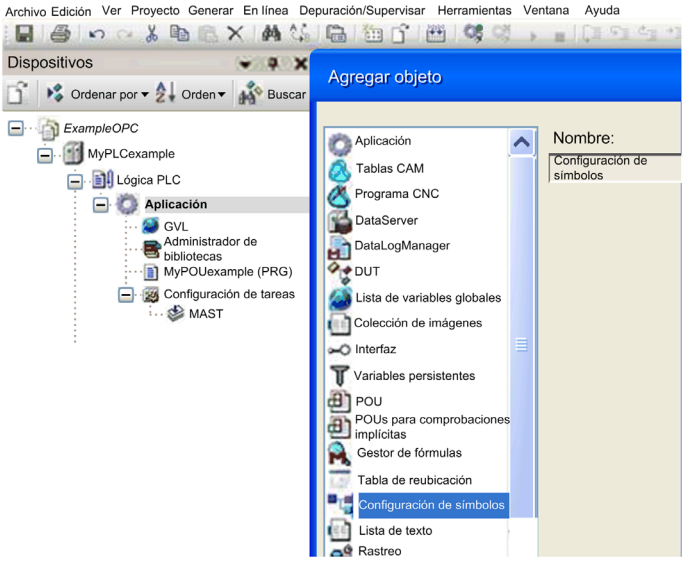
- Puerta de enlace V3

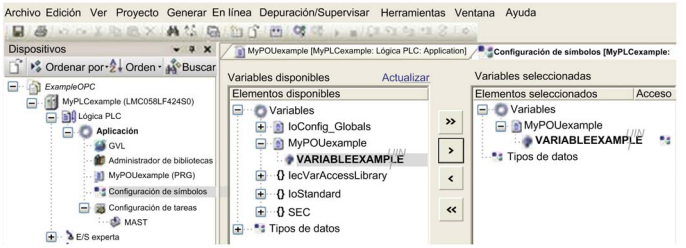
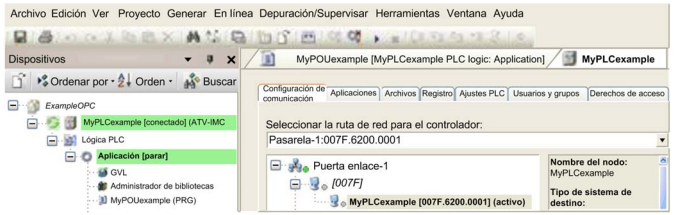
Declaración de una variable que se usará con OPC

Pasos para declarar una variable

Declare una variable que se usará con OPC como se indica a continuación:

Paso	Acción	Ejemplo
1	Cree un proyecto.	**ExampleOPC**
2	Añada y seleccione un controlador mediante el comando **Agregar dispositivo**.	–
3	Configure el nombre del dispositivo haciendo clic dos veces en el nodo del dispositivo para poder modificarlo.	**MyPLCexample**
4	Cree un **PROGRAMA** en su aplicación haciendo clic con el botón derecho en el elemento **Aplicación** y ejecutando el comando **Agregar objeto** → **POU...**.	Ejemplo de programa: Aumente una variable UINT: VARIABLEEXAMPLE
5	Configure el nombre del **PROGRAMA**.	**MyPOUexample**
6	Haga doble clic en una tarea y asocie el **PROGRAMA** a la tarea.	Ejemplo de tarea: **MAST** Ejemplo de **PROGRAMA**: **MyPOUexample**
7	Ejecute el comando **Generar todo** en el menú **Generar** y asegúrese de que no se detecta ningún error durante la ejecución del comando **Generar**. 	–

Paso	Acción	Ejemplo
8	Cree un objeto **Configuración de símbolos** en su aplicación haciendo clic con el botón derecho en el elemento **Aplicación** y ejecutando el comando **Agregar objeto →Configuración de símbolos...**. 	–
9	En el cuadro de diálogo **Agregar configuración de símbolos**, haga clic en **Abrir**.	–
10	Haga clic en el vínculo **Actualizar**.	–
11	Amplíe el elemento **Variables**, en la lista **Variables disponibles**.	–
12	Seleccione la variable que desee compartir con su cliente OPC de su programa.	Ejemplo de variable: VARIABLEEXAMPLE Ejemplo de programa: **MyPOUexample**

Paso	Acción	Ejemplo
13	Haga clic en el botón **>** para enviar la variable a la base de datos compartida para que esté disponible para el cliente OPC. 	–
14	Ejecute el comando **Generar todo** en el menú **Generar** y asegúrese de que no se detecta ningún error durante la ejecución del comando **Generar**.	–
15	Seleccione la ficha **Configuración de símbolos**.	**Comentario**: En el directorio en el que ha almacenado el proyecto encontrará un archivo XML que incluye una lista de variables que están accesibles para el cliente OPC.
16	Conecte el PC al controlador con la ficha **Configuración de comunicación**.	–
17	Descargue la aplicación.	–
18	Inicie la aplicación. 	–

Archivo XML con las variables accesibles para el cliente OPC

En el directorio en el que ha almacenado el proyecto encontrará un archivo XML creado automáticamente en el que se describe la lista de variables que están accesibles para el cliente OPC.

Configuración del servidor OPC

Inicio de la herramienta de configuración de OPC

Configure el servidor OPC y vincúlelo al proyecto que ha creado, como se indica a continuación:

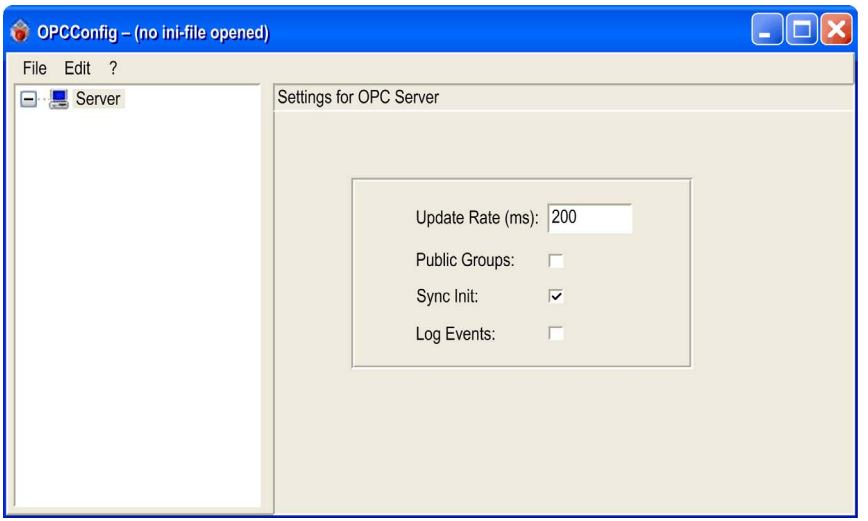
Paso	Acción
1	Vaya al directorio: *C:\Archivos de programa\Schneider Electric\SoMachine OPCServer*
2	Haga doble clic en el archivo *OPCConfig.exe*
3	La herramienta de configuración *OPCconfig.exe* permite generar un archivo INI necesario para inicializar el servidor OPC con los parámetros deseados para la comunicación entre el proyecto de CoDeSys y los controladores.

Herramienta de configuración OPC

La herramienta de configuración contiene los elementos siguientes:

- Una barra de menús
- Una vista de árbol para asignar uno o varios controladores al servidor
- Un cuadro de diálogo de configuración que corresponde a la entrada de árbol seleccionada

Después de haber iniciado la herramienta, aparecerá como sigue, conteniendo la configuración común predeterminada:



Menú Archivo de la herramienta de configuración de OPC

El menú **Archivo** proporciona comandos para cargar y guardar los archivos de configuración a/de la herramienta de configuración:

Comando	Métodos abreviados	Descripción
Abrir	CTRL+O	Para modificar una configuración existente. Se abre el cuadro de diálogo predeterminado para abrir un archivo. Seleccione un archivo INI existente. El filtro se establece automáticamente en *Archivos OPCconfg *.ini.* La configuración descrita en el archivo INI se cargará en la herramienta de configuración.
Nuevo	CTRL+N	Para crear una configuración. Si hay una configuración abierta, se le preguntará si desea guardarla antes de cerrarla. A continuación, la herramienta de configuración mostrará la configuración predeterminada.
Guardar	CTRL+S	Guarda la configuración actual en el archivo INI cargado.
Guardar como	–	Guarda la configuración actual en un archivo con otro nombre que puede especificar en el cuadro de diálogo predeterminado.
<n> archivos INI abiertos recientemente	–	Lista de archivos INI que se han modificado desde la última vez que se inició la herramienta. Puede seleccionar un archivo para volver a cargarlo en la herramienta de configuración.
Salir	–	Termina la herramienta. Si no se ha guardado ningún cambio en la configuración actual, se le pedirá que lo haga.

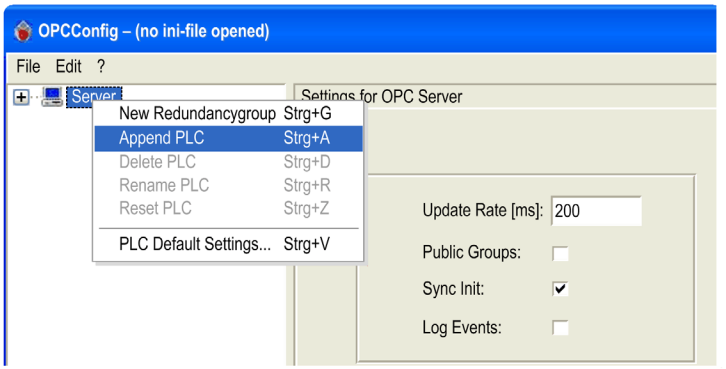
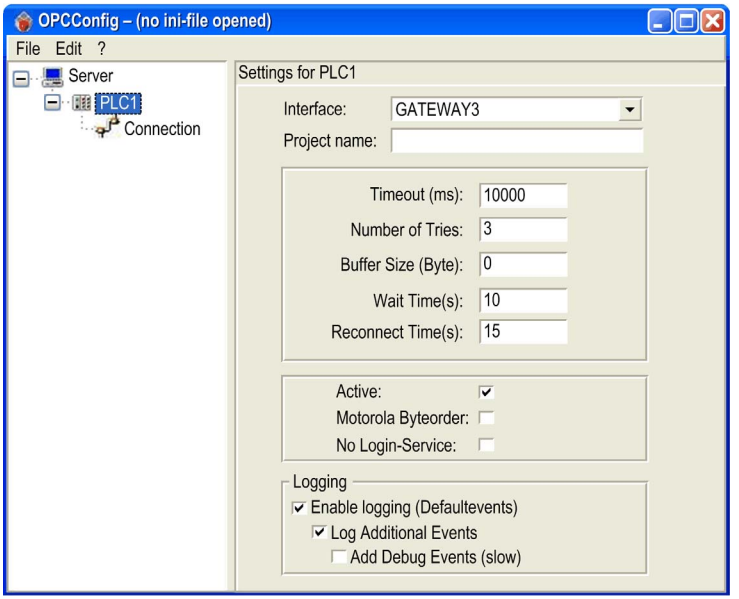
Menú Edición de la herramienta de configuración de OPC

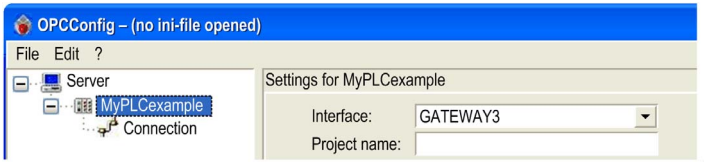
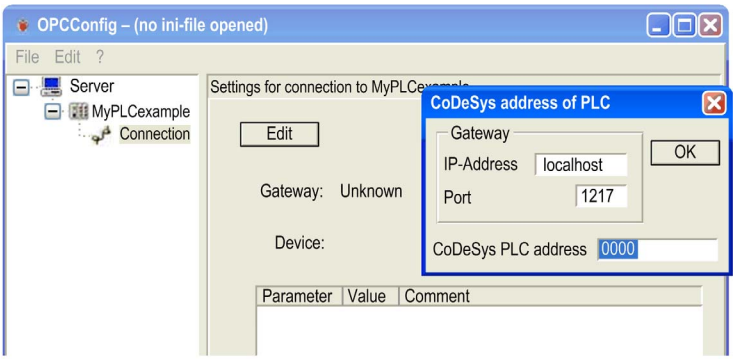
El menú **Edición** proporciona los comandos para editar el árbol de configuración en la parte izquierda del configurador.

Comando	Métodos abreviados	Descripción
Nuevo grupo redundante	CTRL+G	Se agregará una entrada de grupo redundante bajo **Servidor**. Si ya hay un controlador o grupo redundante en el árbol, el nuevo grupo redundante se añadirá al final. De forma predeterminada, una nueva entrada se denomina **Redundant<n>**, donde n es un número consecutivo que empieza en 1. Para cambiar el nombre de la entrada, selecciónela en el árbol y use el comando **Edición →Cambiar nombre de PLC** o haga clic dos veces para hacer que pueda modificarse.
Agregar PLC	CTRL+O	Se agregará una entrada del controlador bajo **Servidor**. Se agregará un nuevo controlador al final del árbol existente. De forma predeterminada, una nueva entrada se denomina **PLC<n>**, donde n es un número consecutivo que empieza en 1. Para cambiar el nombre de la entrada, selecciónela en el árbol y use el comando **Edición →Cambiar nombre de PLC** o haga clic dos veces para hacer que pueda modificarse.
Eliminar PLC	CTRL+D	La entrada del controlador seleccionado se eliminará del árbol de configuración.
Cambiar nombre de PLC	CTRL+R	Se puede cambiar el nombre de la entrada del controlador seleccionado.
Restablecer PLC	CTRL+Z	La configuración de la entrada del controlador seleccionada se restablecerá a los valores predeterminados definidos en **Configuración predeterminada de PLC**.
Configuración predeterminada de PLC...	–	aún no disponible

Configuración del servidor OPC

Configure el servidor OPC como se indica a continuación:

Paso	Acción
1	Haga clic con el botón derecho en el icono **Servidor** y ejecute el comando **Agregar PLC**: 
2	Seleccione **GATEWAY3** en la lista **Interfaz**. 

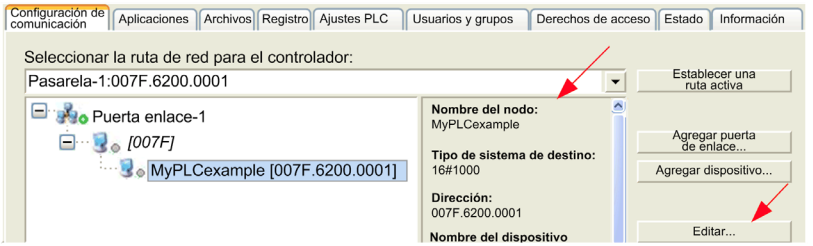
Paso	Acción
3	Haga doble clic en la opción **PLC1** para cambiar su nombre (por ejemplo: **MyPLCexample**). 
4	Haga doble clic en el icono **Conexión** y haga clic en el botón **Edición**. **Resultado**: Aparecerá el cuadro de diálogo **Dirección CoDeSys del PLC**. 
5	Para poder acceder a su variable desde su cliente OPC, introduzca la dirección del controlador (por ejemplo: **MyPLCexample**). La dirección se proporciona en el cuadro de diálogo SoMachine **Configuración de comunicación** del proyecto.

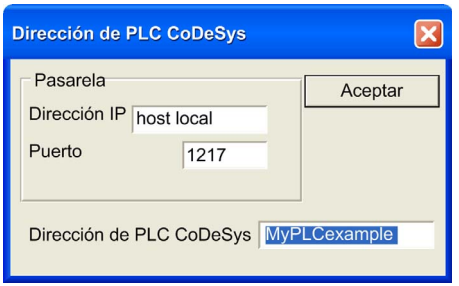
La dirección puede ser física o lógica: Para evitar reconfiguraciones de los valores de las direcciones cuando hay muchos dispositivos en el proyecto, debería usar direcciones lógicas.

Direccionamiento lógico

En nuestro ejemplo, la dirección es: **MyPLCexample**.

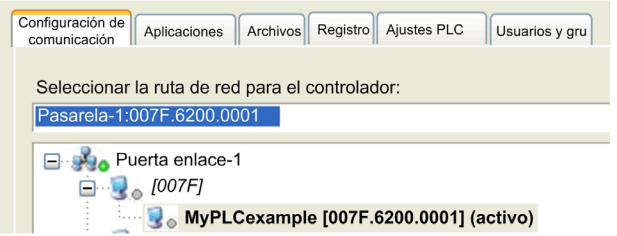
Introduzca directamente el **nombre del nodo** proporcionado en la ficha **Configuración de comunicación** de **MyPLCexample** en su proyecto. Para configurar el **nombre del nodo**, haga clic en el botón **Edición**.



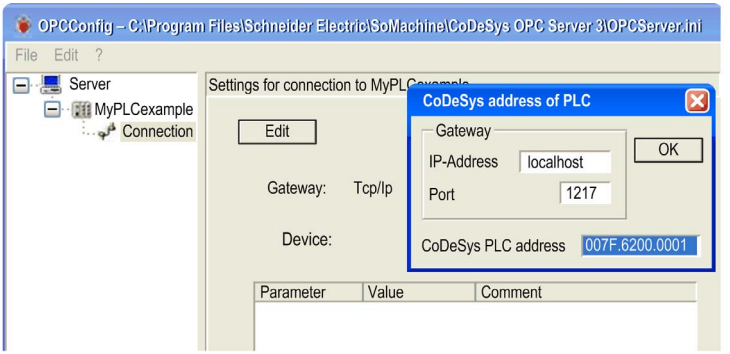
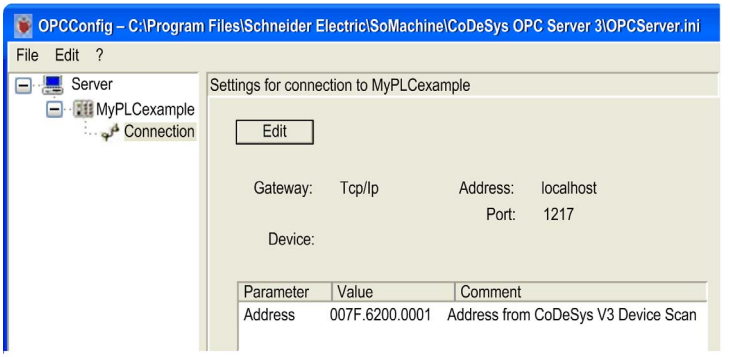


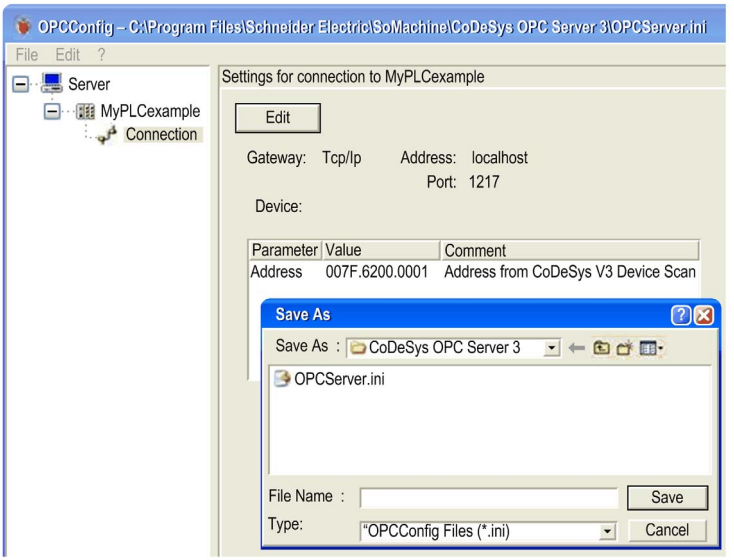
Direccionamiento física

En nuestro ejemplo, la dirección es: 007F.6200.0001



Siga estas instrucciones:

Paso	Acción
1	Tome esta dirección y configure el valor de la **Dirección CoDeSys de PLC**: 
2	Haga clic en el botón **Aceptar**. **Resultado:** Se mostrará el siguiente cuadro de diálogo: 

Paso	Acción
3	Abra el menú **Archivo** y ejecute el comando **Guardar como**. Haga clic en el botón **Aceptar** y se mostrará el siguiente cuadro de diálogo: 
4	Seleccione *OPCServer.ini* y haga clic en **Guardar**. **NOTA:** Observe que el nombre del archivo debe ser *OPCServer.ini*. No utilice un nombre diferente.

Uso del servidor OPC CoDeSys

Descripción general

Después de la instalación del servidor OPC, debería ofrecerse para que el cliente OPC lo seleccione (por ejemplo, visualización). El nombre del servidor OPC es **CoDeSys.OPC.DA**.

Tan pronto como un cliente establece una conexión, el sistema operativo inicia automáticamente el servidor OPC. La finalización del servidor OPC será automática cuando los clientes hayan cerrado sus conexiones con el servidor.

No habrá ningún icono del servidor OPC en la barra de tareas. Sólo aparecerá en el **Administrador de tareas de Windows** como un proceso.

Ejecución de un cliente OPC en un PC en el que no se ejecute SoMachine

Para poder ejecutar el cliente OPC en un PC donde SoMachine no está instalado, continúe como se indica a continuación:

- Consulte el capítulo sobre la *instalación del servidor OPC CoDeSys* en SoMachine *Configuration Manager - Guía del usuario*, y ejecute las siguientes acciones:
  - Instale la puerta de enlace en el PC en el que se esté ejecutando el cliente OPC.
  - Según el cliente OPC, es necesario iniciar el archivo *WinCoDeSysOPC.exe*.
- Copie el archivo *OPCServer.ini* en el directorio en el que está instalado *WinCoDeSysOPC.exe*.

Apéndice C
Lenguaje de script

Contenido de este capítulo

Este capítulo contiene las siguientes secciones:

Sección C.1
Información general

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Introducción

Lenguaje de script SoMachine

El lenguaje de script SoMachine proporciona una potente herramienta para automatizar secuencias. Puede iniciar comandos individuales o secuencias de comandos complejas directamente desde el entorno de programación SoMachine o la línea de comandos de Windows. El lenguaje de script SoMachine es un lenguaje modular basado en IronPython 2.7. El intérprete de IronPython está integrado en el entorno de desarrollo SoMachine. Esta implementación permite utilizar las bibliotecas de la infraestructura de Python. Entre otras cosas, proporcionan acceso a archivos en redes.

Este capítulo no proporciona un ejemplo para cada miembro del Script Engine porque estaría fuera del ámbito de este documento. Los ejemplos proporcionados aquí se han seleccionado debido a una cierta lógica que discurre como un hilo común en toda la API (interfaz de programación de aplicaciones). Si sigue este hilo, encontrará otros miembros en el documento *Automation Platform SDK*, que se proporciona como ayuda online en la página web CoDeSys. A continuación, podrá utilizar estos miembros de la misma manera que se describe en los ejemplos de este capítulo.

Entornos de programación para Python

Hay disponibles los siguientes IDE (entornos de desarrollo integrado) gratuitos:

IDE	Características
Notepad++	• Resalte de sintaxis. • Editor universal.
IDLE (entorno de desarrollo integrado)	• Resalte de sintaxis.
Complemento Visual Studio	• Resalte de sintaxis. • Construcciones de lenguaje IntelliSense para IronPython (y .NET). • Función de depuración integrada. • Fácil de usar. • Requiere Visual Studio.

PyCharm	• Resalte de sintaxis.
	• IntelliSense
	• Función de depuración integrada.
	• Fácil de usar.
	• Producto de software autónomo.
	• La versión Community Edition 3.0 y posteriores admiten numerosas características como:
	  • Codificación y aprendizaje de Python puro.
	  • Editor inteligente con completamiento de código, resalte inmediato de errores, correcciones automáticas, etc.
	  • Posibilidades de reescritura automática de código y navegación con funciones adicionales.
	  • Depurador integrado y soporte para pruebas de unidades.
	  • Integraciones VCS (sistema de control de versiones) nativas.
	  • Interfaz de usuario personalizable y vinculación de claves, con emulación VIM (editor de texto Vi IMproved) disponible.

Notas sobre compatibilidad

Python es un lenguaje dinámico utilizado para una amplia variedad de finalidades que pone el acento sobre código limpio y expresivo. Permite la máxima flexibilidad para el desarrollador a la vez que conserva la legibilidad del código. IronPython aporta Python a .NET y permite el acceso nativo a la infraestructura y las clases .NET. La implementación del intérprete IronPyhon se basa en la versión 2.7 de Python. Hay muchos tutoriales gratuitos y sistemas de ayuda online disponibles en Internet.

NOTA: Incompatibilidad de versiones con Python V3.x. Tenga en cuenta que los usos del lenguaje Python pueden actualizarse, en cuyo caso pueden eliminarse construcciones de lenguaje anteriores en desuso. Por tanto, escriba los scripts de forma compatible con versiones futuras. Esto implica, por ejemplo, utilizar la instrucción
`from __future__ import print_function.`

Para obtener más información, consulte directamente los sitios web:
- *http://wiki.python.org/*
- *http://docs.python.org/*

Ejemplos de nuevas funciones:

```
from __future__ import print_function
from __future__ import division

# New Python print syntax
print('Hello World!')
# Division
# Python 2 return an integer and rounds off
# Python 3 returns a float
print(17/3)
```

Convenciones de codificación

Con el fin de armonizar y facilitar el trabajo de los diferentes programadores en el mismo proyecto de programación, es conveniente acordar un estilo de programación común. Schneider Electric y CoDeSys han acordado aceptar la *Guía de estilo para código Python*. Los nuevos scripts también deben cumplir este estándar.

Para obtener más información, consulte la *Guía de estilo para código Python* en: *http://www.python.org/dev/peps/pep-0008/*.

Enlaces de utilidad

Para obtener más información, consulte los sitios web siguientes:

- Página web oficial de Python, que proporciona un tutorial y documentación de referencia del lenguaje en: *http://docs.python.org/*.
- Blog oficial de Python, en: *http://blog.python.org/*.
- *Beginner's Guide to Python*, en: *http://wiki.python.org/moin/BeginnersGuide*.
- Artículo de la Wikipedia sobre *Python (lenguaje de programación)* en: *http://es.wikipedia.org/wiki/Python*.
- Foro oficial de CoDeSys, que proporciona ejemplos e información práctica, en: *http://forum.codesys.com/*.
- Intérprete IronPython, en: *http://ironpython.codeplex.com/*.
- PyTools (complemento Visual Studio) en: *http://pytools.codeplex.com/*.
- *IronPython Cookbook* en: *http://www.ironpython.info/index.php/Contents*.
- Galileo Openbook, gratuito (sólo disponible en alemán), en: *http://openbook.galileocomputing.de/python/*.

Ejecución de scripts

Descripción general

Puede ejecutar archivos de script (*filename.py*), que contengan una secuencia de comandos de script, desde la interfaz de usuario de SoMachine.

Para obtener más información sobre la ejecución de scripts desde la interfaz de usuario de SoMachine, consulte el capítulo *Comandos relacionados con los scripts (véase SoMachine, Comandos de menú, Ayuda en línea)*.

Archivos por lotes

Comandos de uso frecuente

Comando	Descripción
`- REM o ::`	La línea es un comentario y se ignorará.
`cd`	Cambia a otro directorio.
`echo off`	No se mostrarán los comandos. Con el fin de evitar que se muestren los comandos individuales, inserte un carácter `@` delante del comando.
`echo`	Muestra una cadena o una variable en la consola de programación.
`set`	Declara una variable y le asigna un valor.
`>`	Escribe la salida en un archivo. Si el archivo ya existe, se sobrescribirá.
`>>`	Añade la salida a un archivo. Si el archivo no existe, se creará.

Ejemplo de aplicación:

```
@echo off
REM Go to the directory where SoMachine is installed
cd "<Replace this with the path to the Central.exe, for example,
C:\Program Files (x86)\Schneider Electric\SoMachine Software\>"
REM Run Central.exe with no graphical user interface and the full path
to the script
Central.exe --noui --runscript="<Replace this with the full file path
where the script is stored, for example, D:\MyScripts\TestScript.py>"
pause
```

Aplicación de consola C#

La ejecución del script en una aplicación C# le permite editarlo de forma dinámica antes de que el motor lo ejecute. Además, en la aplicación C# también se pueden realizar algunos pasos previos.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Diagnostics;
namespace ExecuteScriptExample
{
    class Program
    {
        static void Main(string[] args)
        {
            try
            {
                ProcessStartInfo psi = new ProcessStartInfo();

                // Specify the name and the arguments you want to pass
                psi.FileName = @"<Replace this with the path to the
Central.exe, for example,
C:\Program Files (x86)\Schneider Electric\SoMachine Software\>Central.e
xe>";
                psi.Arguments = "--noui --enablescripttracing --
 AdvancedPythonFunctions --runscript=\
"<Replace this with the full file path where the script is stored,
for example, D:\MyScripts\TestScript.py>"";
                // Create new process and set the starting information
                Process p = new Process();
                p.StartInfo = psi;
                // Set this so that you can tell when the process has
completed
                p.EnableRaisingEvents = true;
                p.Start();
                // Wait until the process has completed
                while (!p.HasExited)
                {
                    System.Threading.Thread.Sleep(1000);
                }
```

```
                }
                catch (Exception ex)
                {
                    Console.Write(ex.Message);
                }
                Console.ReadKey();
            }
        }
    }
```

Mejores prácticas

Mejores prácticas

Mejores prácticas

Tenga en cuenta las siguientes prácticas que pueden ayudarle a evitar errores en el código del programa:

- Los bloques de funciones no se crean mediante llaves (`{}`), sino mediante la sangría de los bloques de código.

Ejemplos:

Lenguaje de programación C	Lenguaje de programaciónPython

`int factorial(int x)`
`  {`
`  if (x > 1)`
`  return x* factorial(x - 1);`
`  else`
`  return 1;`
`  }`
	`def factorial(x):`
	`  if x > 1:`
	`  return x * factorial(x - 1)`
	`  else:`
	`  return 1`

- Tenga cuidado al usar los comandos de copiar y pegar.
  Los caracteres de tabulación se sustituyen internamente por 8 espacios. Dado que numerosos editores usan 4 espacios de forma predeterminada, esto puede conducir a errores de código que son difíciles de encontrar. Los bloques de código fuente parecen tener el mismo tamaño que la sangría, pero en realidad el sangrado es diferente.
  Para ayudar a evitar esto, configure el editor de forma que sustituya automáticamente las tabulaciones por espacios.
- Tenga en cuenta la distinción entre mayúsculas y minúsculas (incluso en la línea de comandos).
- Tenga cuidado al introducir los nombres de ruta.
- No se olvide de introducir las comillas de cierre al final de una línea de comandos (`--runscript=""`).
- Tenga cuidado con los caracteres de control que se utilizan con frecuencia en los nombres de ruta. Un carácter de control va precedido de una barra invertida (`\`). Para desactivar este efecto, inserte una `r` delante de las comillas.
  Ejemplo:
  ```
  project_path = r"D:\PythonProjects\SetParameter.project"
  ```
- Asegúrese de que las declaraciones y condiciones de bucle terminan con dos puntos.
  Ejemplo:
  ```
  if len(messages) == 0:
          print("--- Build successful ---")
  ```

Lectura de documentación de API .NET

Lectura de documentación de API .NET para programadores de Python

La versión preliminar actual de la documentación de interfaz de script se genera automáticamente de las fuentes .NET / C# subyacentes. Por tanto, contiene algunas expresiones que no son comunes a los programadores de Python.

La lista ofrece algunas sugerencias sobre cómo convertirlas a la lógica de Python.

- Las interfaces en .NET son un contrato en relación con qué miembros (métodos, propiedades) deben proporcionar las clases que implementan la interfaz. En IronPython, se pueden implementar una o varias interfaces .NET derivándolas como se hace con las clases básicas. Cuando falta un miembro declarado por la interfaz en la declaración, se emite una excepción en tiempo de ejecución. (El ejemplo *DeviceImportFromSvn.py* muestra una clase que implementa la interfaz `ImportReporter`.)
- En .NET, todos los parámetros, propiedades y valores de retorno de funciones se tipifican estáticamente. El tipo permitido se anota delante del nombre del parámetro. En las funciones, el tipo del valor de retorno va delante del nombre de la función. Se permiten instancias de subclases cuando se menciona una clase padre (o interfaz). `void` denota una función sin valor de retorno.
- Los métodos pueden sobrecargarse, una clase puede tener varios métodos con el mismo nombre pero que difieren en el número o los tipos de parámetros. IronPython llamará automáticamente a la variable que coincida.
- El tipo INT puede contener un número entero entre -2.147.483.648 y 2.147.483.647, BOOL es igual al tipo Python BOOL (TRUE y FALSE), el tipo STRING es igual al tipo Python `str` y unicode (que son iguales en IronPython). IDictionary<objeto, objeto> denota un diccionario normal de Python. IronPython realiza automáticamente la conversión entre tipos Python y .NET.
- Si un tipo `T` deriva de `IBaseObject<T>`, ese tipo puede ampliarse con más miembros mediante otros complementos. Los usos reales de este tipo `T` en los parámetros o los valores de retorno se marcarán con `IExtendedObject<T>`.
- La interfaz `IEnumerable<T>` describe cualquier secuencia (listas, matrices, generadores) que producen sólo objetos de tipo `T` (o subclases). Cuando la secuencia produce un objeto incompatible, se emite una excepción en tiempo de ejecución.
- La interfaz `IList<T>` describe una lista que contiene sólo objetos de tipo `T` (o subclases). Cuando se intenta añadir un objeto incompatible, se emite una excepción.
- La sintaxis `params T [] name` es igual a la sintaxis Python `*name` para listas de argumentos variables, pero limita los parámetros al tipo `T` (o subclases).

- Las enumeraciones (ENUM) no existen como construcción del lenguaje en Python. Se utiliza para definir una cantidad fija de valores constantes; por ejemplo, los días de la semana. Puede accederse a los valores de enumeración .NET en IronPython mediante la sintaxis Nombre.Miembro (de forma similar a los miembros de clases estáticas); por ejemplo, `OnlineChangeOption.Try`. Existen varios patrones para emular valores de enumeración en Python; por ejemplo, *http://pypi.python.org/pypi/enum/* o *http://www.ironpython.info/index.php/Enumerations*.
- Las propiedades marcadas con `{ get; set; }` son de lectura y escritura, mientras que las propiedades marcadas sólo con `{ get; }` son de sólo lectura. Son similares al decorador `@property` en Python.

Los siguientes puntos de entrada están disponibles para scripts:

- `system`: funcionalidad básica para la integración en el sistema SoMachine. Este objeto proporciona todas las funciones descritas en la *interfaz ISystem*, como la salida de SoMachine, el acceso a la ventana de mensajes o la consulta si se está ejecutando el modo `--noUI` mediante el uso del comando `ui-present`.
- `projects`: funcionalidad básica para la gestión de proyectos. Este objeto proporciona todas las funciones descritas en la *interfaz IScriptProjects*, como la carga de proyectos y archivos del proyecto. Además, es el punto de entrada a proyectos individuales.
- `online`: funcionalidad básica para la conexión online al dispositivo. Mediante el uso del método `create_online_application` puede crearse el objeto online de un objeto de aplicación. Este objeto online permite iniciar sesión en los controladores, iniciar aplicaciones y recuperar valores de variables.

Puntos de entrada

Información detallada sobre los puntos de entrada

Controlador	Nombre (Tipo)	Descripción
Sistema	system (ISystem)	Función básica para la integración en el sistema de SoMachine
	Severity	ENUM para la prioridad de las noticias
	Guid	Tipo de datos para **G**lobally **u**nique **id**entifier (identificador exclusivo globalmente)
	PromptResult	ENUM para valores de retorno a solicitud del usuario
	MultipleChoiceSelector	Delegación del tipo para la selección de varias solicitudes de elección
	PromptChoiceFilter	
Proyectos	projects (IScriptProjects)	Función básica para la gestión de proyectos
	ExportReporter	Interfaz para la tramitación de eventos durante la exportación
	ImportReporter	Interfaz para la tramitación de eventos durante la importación
	ConflictResolve	ENUM para resolución de conflictos durante la importación
Online	online (IScriptOnline)	Función básica para la conexión online al dispositivo
	OnlineChangeOption	ENUM para los tipos de descarga durante el inicio de sesión en el dispositivo
	ApplicationState	ENUM para el estado de la aplicación
	OperatingState	ENUM para el estado de la operación
	ValuesFailedException	excepción debida a errores detectados en expresiones online
	TimeoutException	Excepción en el timeout durante operaciones online
DeviceObject	DeviceID	Encapsulado del tipo para la identificación de dispositivo

Sección C.2
Ejemplos de Schneider Electric Script Engine

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Parámetros de dispositivo

Descripción general

Para cambiar un parámetro, se necesita el ID de parámetro y `ParameterSet`.

Para encontrar el dispositivo requerido y enumerar los parámetros respectivos, utilice el método de búsqueda que encuentra objetos a partir de un nombre o ruta específico en el proyecto.

Ejemplo de Script Engine

```
# We enable the new python 3 print syntax
from __future__ import print_function
# The path to the project
project_path = r"D:\PythonProjects\SetParameter.project"
# Clean up any open project:
if projects.primary:
    projects.primary.close()
# Load the project
proj = projects.open(project_path);
# Set the project as primary project
proj = projects.primary
# To set a parameter you need a device, a parameter and a new value
that should be assigned to the parameter
# At first search for the SERCOSIII node...
sercos Node = proj.find('SERCOSIII', True)[0]
# ... and add a device named Robot_XK29 to the node, assuming that no
other device is below the SERCOSIII node, otherwise the index must be c
hanged
sercosNode.insert("Robot_XK29", 0, DeviceID(4096, "1003
0082", "1.36.2.2"), 'LXM62DxS')
# Now get the children of the SERCOS node, assuming that Robot_XK29 is
the only one, otherwise the index must be changed
Robot_XK29 = sercosNode.get_children(True)[0]
# Call the get_all_parameters() function, to get a complete list of all
 parameters of that device object.
# A parameter can contain subparameter, which can be checked with
.HasSubElements property
parameter_list = treeobj.get_all_parameters()
# prints all parameters
```

```
for parameter in parameter_list:
    print("ID: " + parameter.Identifier + " Name: " + parameter.Visible
Name + " Value: " + parameter.Value + " ParameterSet: " +
str((parameter.GetAssociatedConnector).ConnectorId))
# Get the WorkingMode parameter:
# ID: 191 Name: WorkingMode Value: 1 ParameterSet: 1
working_mode = Robot_XK29.get_parameter(191, 1)
# Finally set the WorkingMode parameter to 2 = Deactivated
Robot_XK29.set_parameter(working_mode, "2")
```

Versión del compilador

Descripción general

Gracias a la extensión de la versión del compilador, puede visualizar las versiones de compilador asignadas y puede establecer una nueva versión de compilador ejecutando el script.

Ejemplo de Script Engine

```
# Enable the new python 3 print syntax
from __future__ import print_function
# The path to the project
project_path = r"D:\PythonProjects\GetCompilerVersion.project"
# Clean up any open Project:
ifprojects.primary:
    projects.primary.close()
# Load the project
proj = projects.open(project_path);
# Set the project as primary project
proj = projects.primary
print("All compiler versions")
# Get all compiler versions (filtered)
compiler_versions = compiler_settings.get_all_compiler_versions()
# Print all compiler versions (filtered)
for version in compiler_versions:
    print (" - OEM mapped version: " + version)
# Get active compiler version
compiler_version = compiler_settings.active_compiler_version
print("Current compiler version:" + compiler_version)
# Set new compiler version
compiler_settings.active_compiler_version = "3.5.0.20"
print("New compiler version: " +
compiler_settings.active_compiler_version)
# Save project
projects.primary.save()
```

Perfil de visualización

Descripción general

Gracias a la extensión del perfil de visualización, puede visualizar los perfiles de visualización y puede establecer el perfil de visualización activo del proyecto.

Ejemplo de Script Engine

```
# Enable the new python 3 print syntax
from __future__ import print_function
#The path to the project
project_path = r"D:\PythonProjects\GetVisualizationProfile.project"
# Clean up any open Project:
if projects.primary:
    projects.primary.close()
# Load the project
proj = projects.open(project_path);
# Set the project as primary project
proj = projects.primary
# Print the active profile
print("Current visual profile: " +
visualization_settings.active_profile_name)
# Get all available visualization profiles
profile_names = visualization_settings.get_all_visual_profile_names()
# Print the profiles
for visual_profile in profile_names:
    print (" - " + visual_profile)
# Set the profile to V1.35.12.0
visualization_settings.active_profile_name = "V1.35.20.0"
# Get the active profile
profile_name = visualization_settings.active_profile_name
# Print the active profile
print("New visual profile: " + profile_name)
# Save project
projects.primary.save()
```

Actualización de bibliotecas

Descripción general

Gracias a la extensión para actualizar bibliotecas, puede actualizar las bibliotecas del proyecto automáticamente. Se trata de la misma función que proporciona el comando **Bibliotecas** → **Asignación automática de versiones (todas las bibliotecas)** en la interfaz gráfica de usuario de SoMachine.

Ejemplo de Script Engine

```
# Enable the new python 3 print syntax
from __future__ import print_function
# The path to the project
project_path = r"D:\PythonProjects\Example.project"
# Clean up any open project:
if projects.primary:
    projects.primary.close()
# Load the project
proj = projects.open(project_path);
# Set the project as primary project
proj = projects.primary
# Search for die library manager objects in the project
lib_managers = [i for i in proj.get_children(True) if i.is_libman]
# Make the auto mapping for each library manager found
for lib_manager in lib_managers:
        lib_manager.make_auto_mapping()
```

Limpieza y compilación de aplicaciones

Descripción general

Gracias a la extensión para limpiar y compilar aplicaciones, puede limpiar un proyecto o compilar un proyecto nuevo.

Ejemplo de Script Engine

```
# Enable the new python 3 print syntax
from __future__ import print_function
# The path to the project
project_path = r"D:\PythonProjects\Example.project"
# Clean up any open project:
if projects.primary:
    projects.primary.close()
# Load the project
proj = projects.open(project_path);
# Set the project as primary project
proj = projects.primary
# Fetch the active application.
app = proj.active_application
# Clean application
new_project.clean_application(app)
# Compile application and store compiler messages in a list
messages = new_project.compile_application(app)
# If messages == None the build was successful
if len(messages) == 0:
    print("--- Build successful ---")
# Otherwise print results
else:
    for i in messages:
        # If serverity == 'Script Engine Exception' the plugin caused
an exception.
        # The text describes the exeption details.
        print(i.Serverity, i.Text)
```

Configuración de comunicación

Descripción general

En este ejemplo se muestra cómo cargar o establecer la dirección IP de un controlador de su elección ejecutando el script.

Ejemplo de Script Engine

```
from __future__ import print_function
def main():
    if not projects.primary:
        system.ui.error("No active project.")
        return
    project = projects.primary
    #find the controller by the object name where address should be set
and read.
    controller = project.find('LMC', True)[0]
    #reboot the controller
    controller.set_communication_address('192.168.2.25')
    #read address back and show it.
    print('get_communication_address:=' +
controller.get_communication_address())
    system.ui.info("Test complete")
main()
```

Restablecimiento de mensajes de diagnóstico

Descripción general

Una vez que ha iniciado sesión en la aplicación, puede restablecer los mensajes de diagnóstico del controlador. Se trata de un método de extensión del objeto del controlador.

En el ejemplo siguiente se muestra cómo restablecer los mensajes de diagnóstico. Debe obtener el proyecto primario e iniciar sesión en la aplicación, tal como se muestra en los otros ejemplos (compilación de una aplicación *(véase página 938)*).

Ejemplo de Script Engine

```
# get the project instance and log in to the application
# find the controller which messages shall be reset
controller = project.find("LMC", True)[0]
# Get all testelements from testseries "TS_Crank"
controller.reset_diagnosis_messages()
```

Reinicio del controlador

Descripción general

Una vez que hay una instancia del objeto del controlador en el script, puede reiniciar el controlador usando un método en ese objeto.

Ejemplo de Script Engine

```
from __future__ import print_function
def perform_application_login(project):
    app = project.active_application
    onlineapp = online.create_online_application(app)
    onlineapp.login(OnlineChangeOption.Try, True)
def main():
    if not projects.primary:
        system.ui.error("No active project.")
        return
    perform_application_login(project)
    #find the controller named 'LMC' which shall be rebooted
    controller = project.find('LMC', True)[0]
    #reboot the controller
    controller.reboot_plc()
    system.ui.info("Test complete")
main()
```

Convertir dispositivo

Descripción general

La conversión de dispositivos dentro del proyecto puede ser un procedimiento complejo. Esta API simplifica el proceso de conversión y permite evitar errores.

Uso del objeto `DeviceID`

La API de conversión utiliza el objeto `DeviceID`, que identifica un dispositivo o un módulo de dispositivo con una versión específica. `DeviceID` se crea del modo siguiente:

<tipo de dispositivo> <modelo de dispositivo> <versión del dispositivo> <nombre del módulo>

Elemento	Ejemplo	Descripción
tipo de dispositivo	`4096`	identifica un controlador
modelo de dispositivo	`1003 0082` o `1003 009D`	para LMCx00C o LMCx01C respectivamente
versión del dispositivo	`1.50.0.4`	versión de firmware del controlador
nombre del módulo	`LXM52`	módulo del dispositivo de destino

La API de conversión acepta `DeviceID` como instancia de objeto o como parámetro único. Esto permite utilizar un `DeviceID` que contenga todos los elementos mencionados en la tabla anterior o pasar cada elemento como un solo parámetro.

En el ejemplo siguiente se muestra cómo crear un `DeviceID` para un controlador LMCx00C con la versión de firmware 1.50.0.4:

```
Lmcx00c = DeviceID(4096, "1003 0082", "1.50.0.4")
```

Cómo probar si un dispositivo se puede convertir

El script siguiente permite verificar si se puede realizar una conversión a una versión determinada antes de convertir un dispositivo.

```
from __future__ import print_function
defmain():
    # Set the project as primary project
    proj = projects.primary
    controller = proj.find('LMC', True)[0]
    drive = proj.find('Drive', True)[0]
    # test if controller can be converted using DeviceID
    x01c = DeviceID(4096, "1003 009D", "1.36.2.6")
    if controller.can_convert(x01c):
        system.ui.info("Conversion to LMCx01C possible")
    # test if drive can be converted using Parameters and module id
    if drive.can_convert(4096, "1003 0082", "1.36.2.6", "LXM52"):
        system.ui.info("Conversion to LXM52 possible")
main()
```

Obtención de destinos de conversión alternativos

La API proporciona una llamada que recupera los destinos de conversión posibles para un dispositivo determinado. Devuelve el `DeviceID` de cada destino.

```
from __future__ import print_function
#help function to print the delivered device ids
def deviceid_to_string(devId):
    mystr = "ID: {0.id} Type: {0.type} Version:
{0.version}".format(devId)
    if hasattr(devId, 'module_id') and devId.module_id is not None:
        mystr += " ModuleID: {0.module_id}".format(devId)
    return mystr
def main():
    if not projects.primary:
        system.ui.error("No active project. Please open the project
PythonTestProject.projectarchive")
        return
    # Set the project as primary project
    proj = projects.primary
    controller = proj.find('LMC', True)[0]
    alternativeControllers = controller.get_alternative_devices()
    print("ALTERNATIVE DEVICES FOR LMC")
    for id in alternativeControllers:
```

```
        print(deviceid_to_string(id))
    drive = proj.find('drive', True)[0]
    alternativeDrives = drive.get_alternative_devices()
    print("ALTERNATIVE DEVICES FOR DRIVE")
    for id in alternativeDrives:
        print(deviceid_to_string(id))
    system.ui.info("Test complete. Please check the script output
window")
main()
```

Conversión del dispositivo

El proceso de convertir el dispositivo es sencillo, ya que la única acción que se requiere es llamar al método de conversión.

```
from __future__ import print_function
def main():
    proj = projects.primary
    controller = proj.find('LMC', True)[0]
    drive = proj.find('Drive', True)[0]
    # converting the controller
    controller.convert(4096, "1003 009D", "1.36.2.6")
    # converting the drive
    drive.convert(DeviceID(4096, "1003 0082", "1.50.0.4"), "LXM52")
main()
```

Comparación de proyectos

Descripción general

En varios casos resulta útil tener un script que compare automáticamente el contenido de 2 proyectos. La función de comparación de proyectos Python le permite comparar 2 proyectos. Como resultado, proporciona información si los proyectos son distintos, así como un árbol XML detallado que refleja el árbol de proyectos y muestra las diferencias para cada objeto.

Ejemplo de Script Engine

```
from __future__ import print_function
def main():
    proj = projects.primary
    # compare the Primary Project to another Project on disk
    diff = proj.compare_to("CompTest_Right.project")
    write_diff(diff, "Diff1.xml")
    # compare, but ignore whitespaces, comments and properties
    diff = proj.compare_to("CompTest_Right.project", True, True, True)
    write_diff(diff, "Diff2.xml")
def write_diff(differences, filename):
    if differences.DifferenceFound:
        f = open(filename, 'wb')
        f.writelines(differences.ResultTree)
main()
```

Funciones avanzadas de administración de bibliotecas

Descripción general

SoMachine Logic Builder ofrece funciones avanzadas para administrar bibliotecas, las llamadas bibliotecas compatibles con versiones posteriores *(véase SoMachine, Funciones y bibliotecas - Guía del usuario)*. Permiten gestionar de una forma cómoda las referencias y las dependencias entre bibliotecas.

Esta funcionalidad también está disponible mediante scripts y se puede usar en el **Administrador de bibliotecas** de todo el proyecto o en una aplicación única dentro del proyecto. En el siguiente script se muestra cómo comprobar las bibliotecas con vistas a la compatibilidad con versiones posteriores y referencias válidas. Automáticamente asigna las referencias y establece las versiones de biblioteca de forma explícita.

Ejemplo de Script Engine

```
p = projects.primary
app = p.active_application
libmgr = app.get_library_manager()
print("# Checking all libraries:")
for lib in libman.get_libraries():
    print("-     " + lib + "      Is Forward Compatible Library?  "
+ str(libmgr.is_library_forward_compatible(lib)))
if not libmgr.is_current_mapping_valid():
    for lib in libmgr.get_invalid_library_mappings():
        print("Library reference cannot be satisfied for: " + lib)
    print("Trying to auto-map libraries to valid versions")
    libmgr.make_auto_mapping()
else:
    print("All mappings valid")
# set version using individual parameters
libmgr.set_new_library_version("PD_GlobalDiagnostics",
"Schneider Electric", "1.0.1.0")
# set version using the library full name
libmgr.set_new_library_version("PD_AxisModule, 1.1.6.0 (Schneider
Electric)", "1.2.4.0")
# set version to Legacy
libmgr.set_new_library_version("PD_Template", "Schneider Electric",
None)
```

Acceso a POU

Descripción general

En el siguiente ejemplo se muestra cómo imprimir y manipular el código de una POU. Sólo está disponible para lenguajes de programación textual.

Ejemplo de Script Engine

```
if not projects.primary:
    system.ui.error('No primary project set')
p = projects.primary
pou = p.find('SR_Main', True)[0]
# read and print the declaration of the program
decl = pou.get_interface_text()
print(decl)
# read and print the implementation of the program
code = pou.get_implementation_text()
print(code)
decl = "PROGRAM SR_Main\n" + \
            "VAR\n" + \
            "    iTest: INT;\n" + \
            "END_VAR";
code = "iTest := iTest +1;"
# write new code to the declaration and implementation
pou.set_interface_text(decl)
pou.set_implementation_text(code)
```

Sección C.3
Ejemplos de CoDeSys Script Engine

Descripción general

En este capítulo se proporcionan ejemplos de miembros usados frecuentemente de CoDeSys Script Engine. Para obtener una descripción completa de los miembros de cada espacio de nombres, consulte la descripción API de CoDeSys..

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Proyecto

Descripción general

Debido a que los ejemplos de este espacio de nombres son relativamente cortos y autoexplicativos, su significado no se explica en detalle. Se proporcionan ejemplos completos cuando corresponda.

Nuevo proyecto

Con este método se crea un nuevo proyecto.

Consta de 2 parámetros:

- una cadena que especifica la ubicación donde se guardará el proyecto
- un parámetro booleano: si es TRUE, el proyecto será el nuevo proyecto primario. Este parámetro es opcional. El valor predeterminado es TRUE.

El método devuelve la instancia `IProject` (consulte la especificación en el documento *Automation Platform SDK*), que se puede utilizar en otros pasos.

```
# Creates a new project
proj = projects.create("C:\PythonProjects\Example.project", True)
```

Load Project

Con este método se carga un proyecto. Los proyectos que estén abiertos no se cerrarán.

El primer parámetro especifica la ruta del proyecto que se cargará.

```
# Load the project
proj = projects.open(project_path)
```

Guardar proyecto

Con este método se guarda el proyecto en su ubicación física.

```
# Save project
projects.primary.save()
```

Guardar archivo

Con este método se guarda el proyecto como un archivo. Se incluyen las categorías adicionales que están seleccionadas de forma predeterminada, pero sin archivos adicionales.

El primer parámetro especifica la ruta donde se guardará el archivo.

```
# Save archive
projects.primary.save_archive("D:\Archive\Example.archive")
```

Cerrar proyecto

Con este método se cierra el proyecto. Si hay cambios sin guardar en este proyecto, se descartarán.

```
# Clean up any open project:
if projects.primary:
    projects.primary.close()
```

Buscar objeto

Este método permite buscar objetos que coincidan con el nombre dado.

Consta de 2 parámetros:

- El primer parámetro es el nombre del objeto que se busca.
- El segundo parámetro especifica si se realiza una búsqueda recursiva. Este parámetro es opcional. El valor predeterminado es FALSE. Este método devuelve una colección de objetos.

```
device = proj.find('DRV_Lexium62', True)[0]
```

Los nombres no son exclusivos en el árbol. Como consecuencia, se pueden encontrar varios objetos. La búsqueda se realiza por el nombre no localizado.

Native Import

Con este método se importan los archivos especificados en formato XML nativo en el nivel superior de este proyecto.

```
system.trace
import os
project_path = r"D:\MyProjects\Example.project"
import_path = r"D:\MyProjects\ImportFiles"
# Close project if opened
if projects.primary:
    projects.primary.close()
proj = projects.open(project_path);
# Set the new project to primary
proj=projects.primary
files = os.listdir(import_path)
# create the import reporter
class Handler(NativeImportHandler):
    def conflict(self, name, obj, guid):
        print("Object already exists: ",name)
        return NativeImportResolve.skip
    def progress(self, name, obj, exception):
        print("in progess: ", name)
    def skipped(self, list):
        for obj in list:
            print("Skipped: ", obj.get_name())
```

```
def importFilter(name, guid,type,path):
    return True;
# create the importer instance.
handler = Handler()
for file in files:
    file_path = import_path + "\\" + file
    print(file)
    proj.import_native(file_path, importFilter, handler)
proj.save();
```

Importación PLCOpenXML

Con este método se importa en el nivel superior del proyecto el contenido del archivo PLCopenXML especificado.

```
# CoDeSys XML import/export functionality.
from __future__ import print_function
import sys, io
# Set target Project to primary
proj=projects.primary
# Create the import reporter
class Reporter(ImportReporter):
    def error(self, message):
        system.write_message(Severity.Error, message)
    def warning(self, message):
        system.write_message(Severity.Warning, message)
    def resolve_conflict(self, obj):
        return ConflictResolve.Copy
    def added(self, obj):
        print("added: ", obj)
    def replaced(self, obj):
        print("replaced: ", obj)
    def skipped(self, obj):
        print("skipped: ", obj)
    @property
    def aborting(self):
        return False
# Create the importer instance.
reporter = Reporter()
filename = r"D:\ExportObjects\Drv_Master.xml"
# Search for the SERCOSIII node, where the device should be added.
device = proj.find('SERCOSIII', True)[0]
# Import the data into the project.
device.import_xml(reporter, filename)
```

Native Export

Con este método se exportan los objetos especificados en formato nativo en una cadena o un archivo de la ruta indicada. Los objetos que no son exportables se detectan como error, pero la exportación continúa.

```
# Tests CoDeSys native import/export functionality.
from __future__ import print_function
project_path = r"D:\MyProjects\Example.project"
# Clean up any open Project:
if projects.primary:
    projects.primary.close()
proj = projects.open(project_path);
proj=projects.primary
import sys, io
# Collect all POU nodes in that list.
projectObjects = []
# Collect all the leaf nodes.
for node in proj.get_children(True):
    projectObjects.append(node)
# Print everything just to know what is going on.
for i in projectObjects:
    print("Found: ", i.type, i.guid, i.get_name())
# Export the files.
for candidate in projectObjects:
    # Create a list of objects to export:
    # The object itself
    objects = [candidate]
    # And sub-objects (POUs can have actions, properties, ...)
    objects.extend(candidate.get_children(True))
    # And the parent folders.
    parent = candidate.parent
    while ((not parent.is_root) and parent.is_folder):
        objects.append(parent)
        parent = parent.parent
    # Create a unique file name:
    filename = "D:\ExportFiles\\%s__%s.export" % (candidate.get_name(),
 candidate.guid)
    # print some user information
    print("Exporting ", len(objects), " objects to: ", filename)
    # and actually export the project.
    proj.export_native(objects, filename)
```

Exportación PLCOpenXML

Con este método se exportan los objetos especificados en formato PLCopenXML en una cadena o un archivo de la ruta indicada. Los objetos que no son exportables se detectan como error, pero la exportación continúa.

```
# CoDeSys XML import/export functionality.
from __future__ import print_function
import sys, io
# Set target Project to primary
proj=projects.primary
# define the printing function
def printtree(treeobj, depth=0):
    name = treeobj.get_name(False)
    if treeobj.is_device:
        deviceid = treeobj.get_device_identification()
        print("{0} - {1} {2}".format("   "*depth, name, deviceid))
    for child in treeobj.get_children(False):
        printtree(child, depth+1)
for obj in projects.primary.get_children():
    printtree(obj)
# Create the export reporter
class Reporter(ExportReporter):
    def error(self, message):
        system.write_message(Severity.Error, message)
    def warning(self, message):
        system.write_message(Severity.Warning, message)
    def nonexportable(self, message):
        print(message)
    @property
    def aborting(self):
        return False
reporter = Reporter()
# Finds the DRV_Master object in the project
device = proj.find('DRV_Master', True)
filename = r"D:\ExportObjects\Drv_Master.xml"
# Exports the object to the hard drive
proj.export_xml(reporter, device, filename, True, True)
```

Aplicaciones online

Descripción general

NOTA: Algunos de los comandos de las aplicaciones online pueden cambiar temporalmente la aplicación activa.

Esta interfaz se exporta a Python y, por tanto, cumple los estándares de nomenclatura de Python.

Creación de aplicaciones en línea

Con este método se crea una aplicación en línea.

```
app = online.create_online_application();
```

Valores preparados

Este ejemplo muestra cómo escribir valores preparados.

```
# Tests for the Online functionality - listing of prepared values.
import utils
try:
    print "Trying to create online application."
    app = online.create_online_application();
except Exception as e:
    print "Currently offline, executing startup..."
    execfile(utils.makepath("OnlineTestStartup.py"))
    print "Retrying to create online application."
    app = online.create_online_application();
print "app:", app.application_state, "op:",
app.operation_state.ToString("f")
print "Unpreparing values:"
for expression in app.get_prepared_expressions():
    app.set_prepared_value(expression, '')
    print "%s: '%s' '%s'" % (expression, app.read_value(expression),
app.get_prepared_value(expression))
print "Unforcing values:"
for expression in app.get_forced_expressions():
    app.set_unforce_value(expression)
    print "%s: '%s' '%s'" % (expression, app.read_value(expression),
app.get_prepared_value(expression))
app.force_prepared_values()
assert len(app.get_prepared_expressions()) == 0, "still some prepared
values remain..."
```

```
assert len(app.get_forced_expressions()) == 0, "still some prepared
values remain..."
print "now preparing a value and forcing it:"
app.set_prepared_value("POU.testoutput", "4711");
app.force_prepared_values()
print "now preparing a value and writing it:"
app.set_prepared_value("POU.testint", "INT#1147");
app.write_prepared_values();
print "The prepared values are now written."
```

Inicio de sesión en la aplicación

Con este método se lleva a cabo el inicio de sesión en la aplicación. Si antes ya se ha iniciado sesión en la aplicación, se cerrará y se llevará a cabo un nuevo inicio de sesión.

Consta de 2 parámetros:

- El primer parámetro es la opción de cambio.
- El segundo parámetro borrará las aplicaciones anteriores, si se establece en `True`.

```
project_path = r"D:\MyProjects\Example.project"
# Clean up any open Project:
if projects.primary:
    projects.primary.close()
proj = projects.open(PROJECT);
proj = projects.primary
# Fetch the active application.
app = proj.active_application
# Create the online application for it.
onlineapp = online.create_online_application(app)
# Log in to the device.
onlineapp.login(OnlineChangeOption.Try, True)
```

Cierre de sesión de la aplicación

Este método cierra la sesión de la aplicación. Si no se había iniciado sesión en la aplicación, no pasará nada.

```
# Logout.onlineapp.logout()
```

Inicio de la aplicación

Este método inicia la aplicación.

```
project_path = r"D:\MyProjects\Example.project"
# Clean up any open Project:
if projects.primary:
    projects.primary.close()
proj = projects.open(PROJECT);
proj = projects.primary
# Fetch the active application.
app = proj.active_application
# Create the online application for it.
onlineapp = online.create_online_application(app)
# Log in to the device.
onlineapp.login(OnlineChangeOption.Try, True)
# Start the application, if necessary.
if not onlineapp.application_state == ApplicationState.run:
    onlineapp.start()
# Let the app do its work for some time...
system.delay(1000)
```

Detención de la aplicación

Este método detiene la aplicación.

```
# Stop the application
onlineapp.stop()
```

Objetos

Buscar

Este método permite buscar objetos que coincidan con el nombre dado.

Consta de 2 parámetros:

- El primer parámetro es el nombre del objeto que se busca.
- El segundo parámetro especifica si se realiza una búsqueda recursiva. Este parámetro es opcional. El valor predeterminado es FALSE. Este método devuelve una colección de objetos.

```
# Finds the DRV_Master object in the project
device = proj.find('DRV_Master', True)
```

Quitar

Este método elimina el objeto.

```
# Finds the DRV_Master object in the project
device = proj.find('DRV_Master', True)
# Removes the DRV_Master object from the project
device.remove()
```

Cambiar nombre

Este método cambia el nombre del objeto por otro nuevo.

```
# Finds the DRV_Master object in the project
device = proj.find('DRV_Master', True)
# Removes the DRV_Master object from the project
device.rename('DRV_Master_2')
```

Importar/Exportar

Consulte la descripción de importaciones/exportaciones para proyectos *(véase página 951)*. La única diferencia es que se llama al objeto, y no al proyecto, para importación/exportación.

Dispositivos

Descripción general

En este capítulo se describen los métodos para la manipulación de objetos de dispositivo.

Agregar

Este método añade el dispositivo especificado.

Consta de 3 parámetros:

- una cadena que especifica el nombre del dispositivo
- `DeviceID` que especifica el ID del dispositivo
- una cadena que especifica el ID de módulo

```
device.insert("RobotA", DeviceID(4096, "1003 0082", "1.36.1.1"),
'LXM62DxS')
```

Deshabilitar

Este método marca este dispositivo como deshabilitado durante la descarga.

```
# Finds the DRV_Master object in the project
device = proj.find('DRV_Master', True)[0]
device.disable()
```

Habilitar

Este método marca este dispositivo como habilitado durante la descarga.

```
# Finds the DRV_Master object in the project
device = proj.find('DRV_Master', True)[0]
device.enable()
```

Get Address

Este método obtiene la dirección del dispositivo. Devuelve una cadena.

```
# Finds the DRV_Master object in the project
device = proj.find('DRV_Master', True)[0]
device.get_address()
```

Get Device Identification

Este método obtiene la identificación del dispositivo.

```
# Finds the DRV_Master object in the project
device = proj.find('DRV_Master', True)[0]
deviceid = device.get_device_identification()
print("{0} - {1} {2}".format("   "*depth, name, deviceid))
```

Get Gateway

Este método devuelve la `GUID` de la puerta de enlace.

```
# Finds the DRV_Master object in the project
device = proj.find('DRV_Master', True)[0]
gateway = device.get_gateway()
```

Insertar

Este método inserta el dispositivo especificado en el índice especificado.

Consta de 4 parámetros:

- una cadena que especifica el nombre del dispositivo
- `Int32` que especifica el índice dónde insertar el dispositivo
- `DeviceID` que especifica el ID del dispositivo
- una cadena que especifica el ID de módulo

```
device.insert("RobotA", 1, DeviceID(4096, "1003 0082", "1.36.1.1"),
'LXM62DxS')
```

Set Gateway and Address

Este método establece la puerta de enlace y la dirección. Si transmite la `GUID` vacía y una dirección vacía, se borrará la dirección de la puerta de enlace.

```
device.set_gateway_and_address(GUID gateway, string address)
```

Set Simulation Mode

Este método establece el modo de simulación. Si se establece en `True`, la simulación está habilitada.

```
# Finds the DRV_Master object in the project
device = proj.find('DRV_Master', True)[0]
device.set_simulation_mode (True)
```

Actualizar

Este método actualiza el dispositivo especificado.

```
# Finds the DRV_Master object in the project
device = proj.find('DRV_Master', True)[0]
device.update(DeviceID(4096, "1003 0082", "1.36.1.1"), 'LXM62DxS')
```

Sistema/Interfaz de usuario (UI)

Cuadro de diálogo para examinar un directorio

Abre un cuadro de diálogo para examinar un directorio. En modalidad `--noUI`, aquí puede especificar una ruta.

```
system.ui.browse_directory_dialog("Browse Directory Dialog",
r"D:\Python", Environment.SpecialFolder.Desktop, True)
```

Consta de 4 parámetros:
- una cadena que contiene el mensaje
- una cadena que contiene la ruta que estará preseleccionada cuando se abra el cuadro de diálogo
- `Environment.SpecialFolder` contiene la carpeta raíz del cuadro de diálogo de examinar
- un parámetro booleano: si es `True`, en el cuadro de diálogo se mostrará un botón que permite crear nuevas carpetas.

Este método devuelve la ruta seleccionada. Si cancela el cuadro de diálogo, no se devolverá nada.

Selección

Este método permite elegir uno de varios elementos listados.

```
list = ["RobotA", "RobotB", "RobotC"]
system.ui.choose("Choose", list, True)
```

Consta de 3 parámetros:
- una cadena que contiene el mensaje
- una lista de las opciones que se mostrarán: los objetos se convierten a cadena para poder visualizarlos.
- un parámetro booleano: si es `True`, en el cuadro de diálogo se mostrará un botón que permite crear nuevas carpetas.

Este método devuelve una tupla de Python que contiene 2 elementos:
- el índice del elemento seleccionado, o
  `-1` si el valor cancelable se estableció en `True` y canceló el cuadro de diálogo
- el elemento seleccionado o `None`

Error Detección

Este método indica un mensaje de detección de error. Impide que se produzcan más acciones hasta que se haya acusado recibo del mensaje.

```
system.ui.error("Error")
```

Información

Este método indica un mensaje de información. Impide que se produzcan más acciones hasta que se haya acusado recibo del mensaje.

```
system.ui.info("Info")
```

Cuadro de diálogo Abrir archivo

Este método muestra un cuadro de diálogo **Abrir archivo**. En modalidad `--noUI`, aquí puede especificar una ruta.

```
system.ui.open_file_dialog("Select a file")
```

Cadena de consulta

Este método consulta la entrada o la edición de una cadena de texto.

```
system.ui.query_string("Please enter a string")
```

Devuelve una cadena con el texto especificado.

Diálogo Guardar archivo

Este método muestra un cuadro de diálogo **Guardar archivo**. En modalidad `--noUI`, aquí puede especificar una ruta.

```
system.ui.save_file_dialog("Python Script: Save File")
```

Advertencia

Este método indica un mensaje de advertencia. Impide que se produzcan más acciones hasta que se haya acusado recibo del mensaje.

```
system.ui.warning("Warning")
```

Valores de lectura

Descripción general

Puede iniciar el ejemplo siguiente desde la interfaz de usuario de SoMachine o desde la línea de comandos.

Para iniciarlo desde la línea de comandos, cambie al subdirectorio **Common** de la ruta de instalación de SoMachine (<*Sustituir por la ruta a Central.exe, por ejemplo, C:\Archivos de programa (x86)\Schneider Electric\SoMachine Software*\>) y escriba el comando

```
start /wait Central.exe --
runscript="<Replace this with the full file path
where the script is stored, for example, D:\MyScripts\ReadVariable.py>".
```

El script abre una aplicación en SoMachine e inicia sesión en el dispositivo. Si el controlador no está en modalidad de ejecución, se establecerá en RUN. A continuación, la variable `iVar1` se lee y se muestra en la vista **Mensajes** o en la línea de comandos. Al final, la aplicación se cierra.

Ejemplo de script *ReadVariable.py*

```
# Close all projects
while len(projects.all) > 0:
    projects.all[0].close()
# opens project
proj = projects.open("D:\\data\\projects\\Ampel.project")
# set "Ampel.project" to active application
app = proj.active_application
onlineapp = online.create_online_application(app)
# login to device
onlineapp.login(OnlineChangeOption.Try, True)
# set status of application to "run", if not in "run"
if not onlineapp.application_state == ApplicationState.run:
    onlineapp.start()
# wait 1 second
system.delay(1000)
# read value of iVar1
value = onlineapp.read_value("PLC_PRG.iVar1")
# display value in message view or command line
print value
# log out from device and close "Ampel.project"
onlineapp.logout()
proj.close()
```

Valores de lectura de la fórmula y envío de un correo electrónico

Descripción general

Puede iniciar el ejemplo siguiente desde la interfaz de usuario de SoMachine o desde la línea de comandos.

Para iniciarlo desde la línea de comandos, cambie al subdirectorio **Common** de la ruta de instalación de SoMachine (<*Sustituir por la ruta a Central.exe, por ejemplo, C:\Archivos de programa (x86)\Schneider Electric\SoMachine Software*\>) y escriba el comando

```
start /wait Central.exe --
runscript="<Replace this with the full file path
where the script is stored, for example, D:\MyScripts\ScriptEmail.py>".
```

El script abre una aplicación en SoMachine e inicia sesión en el dispositivo. Si el controlador no está en modalidad de ejecución, se establecerá en RUN. A continuación, la variable `iVar1` se lee y se muestra en la vista **Mensajes** o en la línea de comandos. Al final, la aplicación se cierra.

Ejemplo de script *ScriptEmail.py*

```
# Close current project if necessary and open "ScriptTest.project"
if not projects.primary == None:
    projects.primary.close()
project = projects.open("D:\\Data\\projects\\scriptTest.project")
# retrieve active application
application = project.active_application
# create online application
online_application = online.create_online_application(application)
# login to application.
online_application.login(OnlineChangeOption.Try, True)
# start PLC if necessary
if not online_application.application_state == ApplicationState.run:
    online_application.start()
# wait 2 seconds
system.delay(2000)
# open recipe file to read values.
recipe_input_file = open("D:\\Data\\projects\\RecipeInput.txt", "r")
watch_expressions = []
for watch_expression in recipe_input_file:
    watch_expressions.append(watch_expression.strip())
print watch_expressions
# read values from the controllerd
watch_values = online_application.read_values(watch_expressions)
print watch_values
# open output file to write values
recipe_output_file = open("D:\\Data\\projects\\RecipeOutput.txt", "w")
for i in range(len(watch_expressions)):
```

```
    recipe_output_file.write(watch_expressions[i])
    recipe_output_file.write(" = ")
    recipe_output_file.write(watch_values[i])
    recipe_output_file.write("\n")
# Close files
recipe_input_file.close()
recipe_output_file.close()
# send Email
# import respective libraries
import smtplib
from email.mime.text import MIMEText
#open output file
recipe_output_file = open("D:\\Data\\projects\\RecipeOutput.txt", "r")
mail = MIMEText(recipe_output_file.read())
recipe_output_file.close()
#email address sender and recipient
fromm = "info@3s-software.com"
to = "info@3s-software.com"
# set sender and recipient
mail["Subject"] = "Attention value has changed"
mail["From"] = fromm
mail["To"] = to
# send email
smtp = smtplib.SMTP("name of smtp server")
smtp.sendmail(fromm, [to], mail.as_string())
smtp.quit()
# logout and close application
online_application.logout()
project.close()
```

Determinación del árbol Dispositivos del proyecto abierto

Descripción general

En este ejemplo, se determinan los objetos del árbol **Dispositivos** del proyecto abierto y se indican en la línea de comandos o en la vista **Mensajes**. Puede iniciarse desde la interfaz de usuario de SoMachine o desde la línea de comandos.

Para iniciarlo desde la línea de comandos, cambie al subdirectorio **Common** de la tuta de instalación de SoMachine (<*Sustituir por la ruta a Central.exe, por ejemplo, C:\Archivos de programa (x86)\Schneider Electric\SoMachine Software*\>) y escriba el comando
`start /wait Central.exe --runscript="<Replace this with the full file path where the script is stored, for example, D:\MyScripts\DevicePrintTree.py>"`.

Ejemplo de script *DevicePrintTree.py*

```
# We enable the new python 3 print syntax
from __future__ import print_function
import sys
# define the printing function
def printtree(treeobj, depth=0):
    if treeobj.is_root:
        name = treeobj.path
        deviceid = ""
    else:
        name = treeobj.get_name(False)
        if treeobj.is_device:
            deviceid.get_device_identification()
        else:
            deviceid = ""
    print("{0} - {1} {2}".format("   "*depth, name, deviceid))
    for child in treeobj.get_children(False):
        printtree(child, depth+1)
# Now see whether a primary project is open.
if not projects.primary:
    print("Error: Please open a project file first!", file=sys.stderr)
    sys.exit()
# And the actual output
print("--- The current tree of: ---")
printtree(projects.primary)
print("--- Script finished. ---")
```

Ejemplo de script 4: Importación de un dispositivo a PLCOpenXML desde Subversion

Descripción general

En este ejemplo se importa un dispositivo a PLCOpenXML desde Subversion mediante `svn client` en la línea de comandos. Puede iniciarse desde la interfaz de usuario de SoMachine o desde la línea de comandos.

Para iniciarlo desde la línea de comandos, cambie al subdirectorio **Common** de la ruta de instalación de SoMachine (<*Sustituir por la ruta a Central.exe, por ejemplo, C:\Archivos de programa (x86)\Schneider Electric\SoMachine Software*\>) y escriba el comando `start wait Central.exe  runscript="<Replace this with the full file path where the script is stored, for example, D:\MyScripts\DeviceImportFromSvn.py>"`.

Ejemplo de script *DeviceImportFromSvn.py*

```
# Imports a Device in PLCOpenXML from Subversion via command line svn
client.
# We enable the new python 3 print syntax
from __future__ import print_function
import sys, os
# some variable definitions:
SVNEXE = r"C:\Program Files\Subversion\bin\svn.exe"
XMLURL = "file:///D:/testrepo/testfolder/TestExport.xml"
PROJECT = r"D:\test.project"
# clean up any open project:
if projects.primary:
    projects.primary.close()
# Fetch the plcopenxml data from subversion.
# The 'with' construct automatically closes the open pipe for us.
with os.popen('"' + SVNEXE + '" cat ' + XMLURL, 'r') as pipe:
    xmldata = pipe.read()
# create a new project:
proj = projects.create(PROJECT)
# create the import reporter
class Reporter(ImportReporter):
    def error(self, message):
        system.write_message(Severity.Error, message)
    def warning(self, message):
        system.write_message(Severity.Warning, message)
    def resolve_conflict(self, obj):
        return ConflictResolve.Copy
    def added(self, obj):
        print("added: ", obj)
```

```
    def replaced(self, obj):
        print("replaced: ", obj)
    def skipped(self, obj):
        print("skipped: ", obj)

    @property
    def aborting(self):
        return False
# create the importer instance.
reporter = Reporter()
# import the data into the project.
proj.import_xml(reporter, xmldata)
# and finally save. :-)
proj.save()
print("--- Script finished. ---")
```

Apéndice D
Administración de usuarios para Soft PLC

Descripción general

En este capítulo se describen las vistas **Usuarios y grupos** y **Derechos de acceso** del editor de dispositivos. Estas vistas están sólo disponibles si el proyecto de SoMachine contiene controladores **Soft PLC**.

Contenido de este capítulo

Este capítulo contiene los siguiente apartados:

Información general sobre la administración de usuarios para Soft PLC

Descripción general

La función de administración de usuarios descrita en este capítulo le permite definir cuentas de usuario y configurar derechos de acceso (permisos) para controladores **Soft PLC**.

Los derechos para acceder a objetos de proyecto por medio de acciones especificadas se asignan sólo a grupos de usuarios, no a una sola cuenta de usuario. Por lo tanto, cada usuario debe ser miembro de un grupo.

Administración de usuarios para Soft PLC

Antes de configurar usuarios y grupos de usuarios para controladores **Soft PLC**, tenga en cuenta lo siguiente:

- De forma predeterminada, existe un grupo **Everyone**. Cada usuario definido en otros grupos es automáticamente miembro de este grupo. Por lo tanto, a cada cuenta de usuario se le proporciona automáticamente como mínimo la configuración predeterminada. El grupo **Everyone** no se puede eliminar, sólo se le puede cambiar el nombre. Los miembros no se pueden eliminar de este grupo. De forma predeterminada, **Everyone** no tiene el permiso para modificar la configuración de usuarios, grupos y permisos.
- De forma predeterminada, también existe un grupo **Owner**, que sólo contiene un usuario, **Owner**. En un proyecto nuevo, inicialmente sólo **Owner** tiene permiso para modificar la configuración actual de usuarios, grupos y permisos. Por lo tanto, sólo **Owner** puede asignar este derecho a otro grupo. Inicialmente, **Owner** puede iniciar sesión con el nombre de usuario **Owner** y una contraseña vacía. Se pueden añadir usuarios al grupo **Owner** o se pueden eliminar de dicho grupo, pero como mínimo debe quedar 1 usuario. Este grupo, al igual que **Everyone**, no se puede eliminar. Se le permiten todos los derechos de acceso. Por lo tanto, no es posible hacer que un proyecto quede inutilizable prohibiendo los derechos respectivos a todos los grupos. Se puede cambiar el nombre tanto del grupo como del usuario **Owner**.
- Al iniciar el sistema de programación o iniciar un proyecto, en un principio no hay ningún usuario registrado en el proyecto. Pero entonces el usuario puede iniciar sesión opcionalmente por medio de una cuenta de usuario definida con un nombre de usuario y una contraseña para disponer de un conjunto especial de derechos de acceso.
- Tenga en cuenta que cada proyecto tiene su propia administración de usuarios. Así, por ejemplo, para obtener un conjunto especial de derechos de acceso para una biblioteca incluida en un proyecto, el usuario debe iniciar sesión por separado en esta biblioteca. Además, los usuarios y grupos, configurados en proyectos distintos, no son idénticos aunque tengan nombres idénticos.

NOTA: Sólo se permite al usuario **Owner** del grupo **Owner** modificar los permisos, grupos y usuarios configurados actualmente. Por lo tanto, sólo **Owner** puede asignar este permiso a otro grupo.

NOTA: Las contraseñas de los usuarios se guardan de forma irreversible. Si no recuerda la contraseña, no se podrá utilizar la cuenta de usuario correspondiente. Si no recuerda la contraseña del grupo **Owner**, el proyecto completo puede quedar inutilizable.

Administración de derechos de acceso para Soft PLC

La administración de usuarios en un proyecto sólo resulta útil en combinación con la administración de derechos de acceso (=permisos).

Tenga en cuenta lo siguiente:

- En un proyecto nuevo, básicamente todos los derechos aún no están definidos explícitamente sino que están establecidos con un valor predeterminado. Este valor predeterminado se suele permitir con la excepción del derecho para modificar la configuración actual de usuarios, grupos y permisos. De forma predeterminada, sólo se permite para el grupo **Owner**.
- Cuando se crea el proyecto, un miembro del grupo con derecho para modificar los permisos puede definir derechos. Cada derecho específico se puede permitir o prohibir o volver a establecer en el valor predeterminado.
  - La administración de derechos de acceso de un proyecto se realiza en el cuadro de diálogo **Proyecto →Administración de usuarios →Permisos... →Permisos**.
  - La administración de derechos de acceso para objetos se realiza en el cuadro de diálogo **Visualizar →Propiedades... →Propiedades - Información del proyecto**, seleccionando la ficha **Control de acceso**.
- Los derechos de acceso para objetos se heredan. Si un objeto tiene un objeto padre (por ejemplo, si una acción está asignada a un objeto de programa que está insertado en el árbol de estructura debajo del programa, entonces el programa es el padre del objeto de acción), los derechos actuales del padre se convertirán automáticamente en los valores predeterminados del hijo. Las relaciones padre-hijo de los objetos referentes a los derechos de acceso suelen corresponder a las relaciones que se muestran en **DispositivosAplicaciones** y **Herramientas** y se indican en el cuadro de diálogo **Permisos** mediante la sintaxis <objeto padre>.<objeto hijo>. Ejemplo: La acción `ACT` está asignada el objeto POU `PLC_PRG`. De este modo, en **Aplicaciones** se muestra `ACT` en la estructura de árbol con sangría debajo de `PLC_PRG`. En el cuadro de diálogo **Permisos**, `ACT` se representa mediante `PLC_PRG.ACT`, lo que indica que `PLC_PRG` es el padre de `ACT`. Si el derecho de modificación se prohíbe explícitamente para `PLC_PRG` y un determinado grupo de usuarios, el valor predeterminado del derecho de modificación para `ACT` también se prohibirá automáticamente.

Usuarios y grupos

Descripción general

La vista **Usuarios y grupos** del editor de dispositivos se proporciona para los dispositivos que admiten la administración de usuarios online. Esta vista permite configurar cuentas de usuario y grupos de usuarios y, en combinación con la administración de los derechos de acceso, permite controlar el acceso a los objetos en el controlador en modalidad online.

Si desea que determinadas funciones de un controlador sólo las puedan ejecutar los usuarios autorizados, utilice la función de administración de usuarios online. Esta función permite configurar cuentas de usuario para asignar derechos de acceso a grupos de usuarios y forzar la autenticación de usuario en el inicio de sesión.

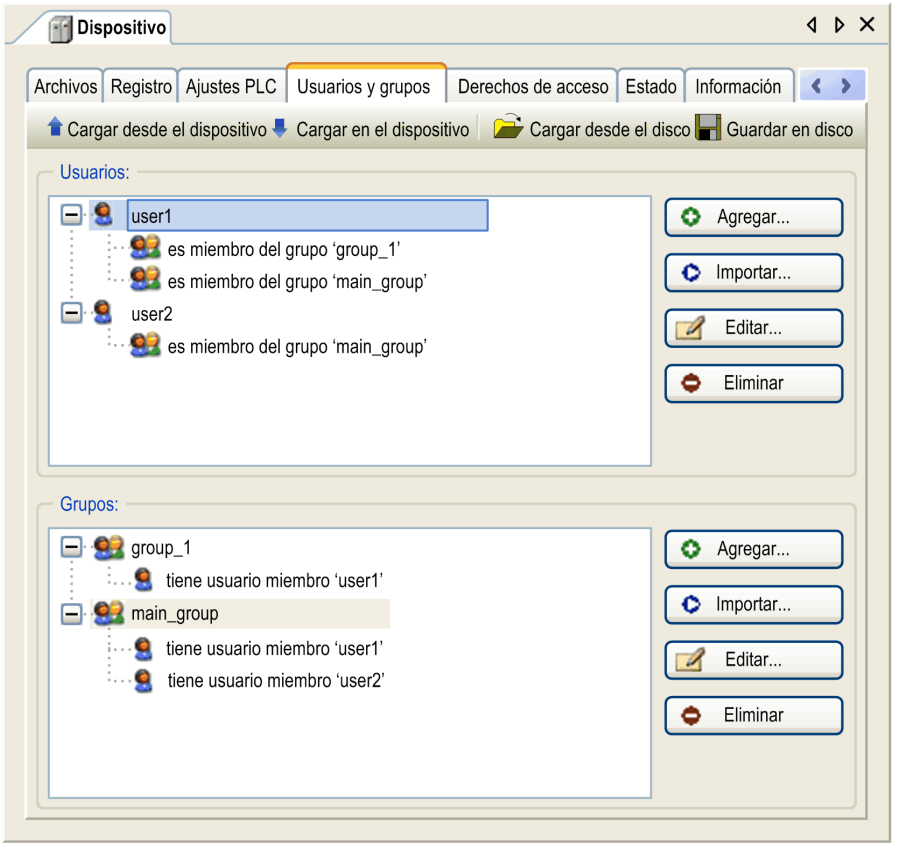
La administración de usuarios específica de dispositivo puede predefinirse mediante la descripción del dispositivo. Esta descripción también define en qué medida pueden editarse las definiciones en los cuadros de diálogo de configuración.

Como en la administración de usuarios de proyecto, los usuarios deben ser miembros de grupos. Sólo los grupos de usuarios pueden obtener determinados derechos de acceso *(véase página 976)*.

Uso del cuadro de diálogo de configuración

Básicamente, la gestión de los cuadros de diálogo de la administración de usuarios es similar a la de la administración de usuarios de proyecto. Existe incluso la opción de importar las definiciones de cuentas de usuario desde la administración de usuarios de proyecto.

Vista **Usuarios y grupos** del editor de dispositivos



Esta vista está dividida en 2 partes:

- En la parte superior se encuentra la administración de accesos de **Usuarios**.
- En la parte inferior se encuentra la administración de accesos de **Grupos**.

Área de usuarios

Para configurar cuentas de usuario están disponibles los botones siguientes:

Botón	Descripción
Agregar	Abre el cuadro de diálogo **Agregar usuario**. Escriba el **Nombre** del usuario y una **Contraseña**. Repita la contraseña en el campo **Confirmar contraseña**.
Importar	Abre el cuadro de diálogo **Importar usuarios**. Muestra todos los nombres de usuario que se han definido en la administración de usuarios del proyecto. Seleccione 1 o varias entradas y haga clic en **Aceptar** para confirmar. Se abrirá el cuadro de diálogo **Introducir contraseña**. Escriba la contraseña correspondiente como se ha definido en la administración de usuarios del proyecto. Ya podrá importar la cuenta de usuario a la administración de usuarios específica del dispositivo. Sin embargo, estas contraseñas no se importan. **NOTA:** Cada cuenta de usuario importada tendrá una definición de contraseña vacía.
Editar	Modifica la cuenta de usuario seleccionada en lo que respecta al nombre de usuario y la contraseña. El cuadro de diálogo **Editar usuario <nombre de usuario>** se corresponde con el cuadro de diálogo **Agregar usuario** (véase más arriba).
Eliminar	Elimina la cuenta de usuario seleccionada.

Área de grupos

Para configurar grupos de usuarios están disponibles los botones siguientes:

Botón	Descripción
Agregar	Abre el cuadro de diálogo **Agregar grupo**. Escriba el **Nombre** de un grupo y seleccione entre los usuarios definidos a aquellos que desea que sean **miembros** de este grupo.
Importar	Abre el cuadro de diálogo **Importar grupos**. Muestra todos los grupos que se han definido en la administración de usuarios del proyecto. Seleccione 1 o varias entradas y haga clic en **Aceptar** para integrarlas en la lista de grupos de la administración de usuarios específica del dispositivo.
Editar	Modifica el grupo seleccionado en lo que respecta al nombre de grupo y los usuarios asociados. El cuadro de diálogo **Editar el grupo <nombre de grupo>** se corresponde con el cuadro de diálogo **Agregar grupo** (véase más arriba).
Eliminar	Elimina el grupo seleccionado actualmente.

Aplicación y almacenamiento de la configuración actual

Los botones están disponibles en la barra superior del cuadro de diálogo:

Botón	Descripción
Cargar en el dispositivo	Descarga la configuración de la administración de usuarios actual en el dispositivo. Sólo empezará a aplicarse después de la descarga.
Cargar desde el dispositivo	Carga la configuración que se aplica actualmente en el dispositivo en el cuadro de diálogo de configuración.
Guardar en disco/**Cargar desde el disco**	La configuración actual se puede guardar en un archivo XML (*.DUM) y volver a cargarse desde ese archivo. Esto es útil para establecer la misma configuración de usuarios en varios sistemas. Para este fin, se proporciona el cuadro de diálogo estándar para examinar el sistema de archivos. El filtro de archivos se establece automáticamente en *.DUM, que corresponde a archivos de administración de usuarios de dispositivos.

Impresión de la configuración de la administración de usuarios

Para imprimir la configuración de la vista **Usuarios y grupos**, ejecute el comando **Imprimir** del menú **Archivo** o el comando **Documento** del menú **Proyecto**.

Derechos de acceso

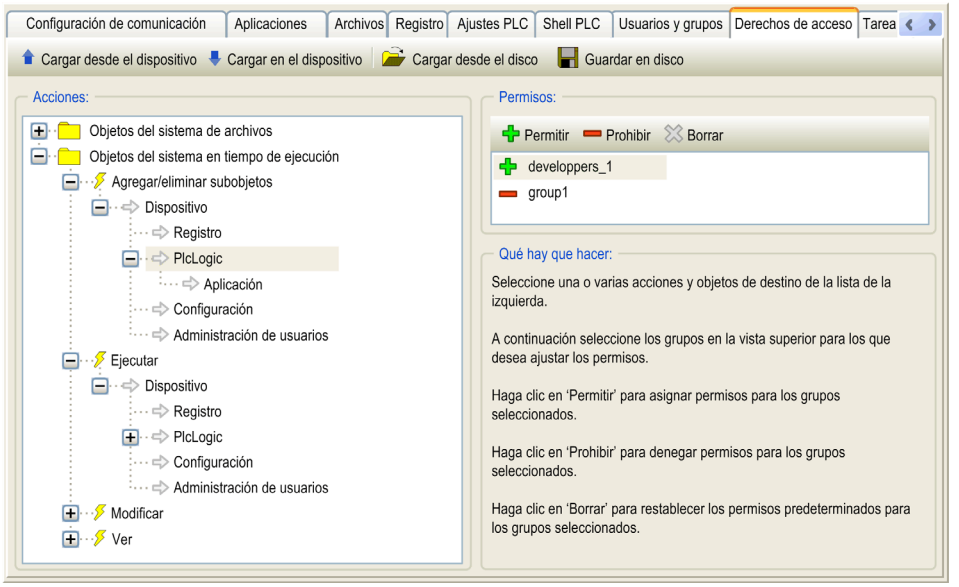
Descripción general

La vista **Derechos de acceso** del editor de dispositivos forma parte de la función de administración de usuarios online *(véase página 972)*. Sirve para otorgar o denegar determinados permisos a los grupos de usuarios que haya definido; de esta forma, se definen los derechos de acceso de los usuarios a los archivos y los objetos (por ejemplo, una aplicación) en el controlador durante el tiempo de ejecución.

Tenga en cuenta que los permisos de acceso solamente se pueden asignar a grupos, no a usuarios individuales. Por este motivo, tiene que definir un usuario como miembro de un grupo. Realice la configuración de los usuarios y los grupos en lavista **Usuarios y grupos** del editor de dispositivos *(véase página 972)*.

Consulte el ejemplo de la imagen siguiente: se otorga permiso para añadir y eliminar subobjetos en el objeto **PlcLogic** al grupo de usuarios **developpers_1**.

Vista **Derechos de acceso** del editor de dispositivos



Definición de los permisos de acceso

Para definir el permiso para realizar una acción sobre 1 o varios objetos, haga lo siguiente:

Paso	Acción
1	Seleccione las entradas de los objetos bajo el tipo de acción que desea en el área **Acciones**.
2	Seleccione el grupo que desee en el área **Permisos**.
3	Haga clic en el botón **Permitir** o **Prohibir**.

NOTA: Al establecer un derecho de acceso sobre un objeto, tenga en cuenta la tabla de asignaciones del párrafo *(véase página 979) ¿A qué acción afecta exactamente un determinado derecho de acceso sobre un objeto?*.

Consulte las instrucciones del área **Qué hay que hacer** de la vista.

Área Acciones

El área **Acciones** contiene una lista de las acciones que se pueden llevar a cabo durante el tiempo de ejecución en los archivos del sistema de archivos del controlador o los objetos en tiempo de ejecución, como, por ejemplo, las aplicaciones. El árbol está estructurado como se indica a continuación:

En el nivel superior hay 2 categorías de objetos agrupadas en carpetas:
- **Objetos del sistema de archivos**
- **Objetos del sistema en tiempo de ejecución**

Los nodos correspondientes a los 4 tipos de acciones aparecen con sangría debajo de las categorías de objetos. Se pueden realizar en los objetos individuales.
- **Modificar** (por ejemplo, descargar una aplicación)
- **Ver** (supervisión)
- **Agregar/eliminar subobjetos** (añadir o eliminar subobjetos en un objeto existente)
- **Ejecutar** (por ejemplo, iniciar o detener una aplicación, establecer puntos de interrupción)

Bajo cada nodo de tipo de acción se encuentran los objetos, o destinos de la acción. Se trata de los objetos del archivo del controlador o del tiempo de ejecución; por ejemplo, **Dispositivo**.

Estas entradas de objetos se muestran en una estructura de árbol que asigna la estructura del sistema de archivos o el árbol de dispositivos.

NOTA: La asignación de una definición de derecho de acceso a un nodo padre suele significar que los nodos hijo heredarán esta definición. Esto es válido siempre y cuando no apliquen su propia definición explícita. Sin embargo, en función del dispositivo, esto se puede gestionar de otra manera. En cualquier caso, las herencias no se muestran en esta vista.

Área Permisos

El área **Permisos** muestra los grupos de usuarios definidos.

Delante de cada grupo habrá uno de los iconos siguientes, que indican el permiso que está asignado en relación con el destino que está seleccionado en el área **Acciones**:

Icono	Descripción
– (signo menos)	Las acciones seleccionadas en el área **Acciones** se permiten al grupo.
+ (signo más)	Las acciones seleccionadas en el área **Acciones** se prohíben al grupo.
X (signo en forma de cruz)	No hay ninguna definición de derecho de acceso explícita para las acciones seleccionadas en el área **Acciones**.
ningún icono mostrado	Hay varias acciones seleccionadas en el área **Acciones** que no tienen una configuración única para el grupo que está seleccionado.

Después de seleccionar los objetos bajo la acción que desea en el área **Acciones** y el grupo que desea en el área **Permisos**, puede utilizar los botones siguientes:

Botón	Descripción
Permitir	Se otorga explícitamente el permiso de acceso.
Prohibir	Se deniega explícitamente el permiso de acceso.
Borrar	El derecho de acceso otorgado para las acciones que están seleccionadas en el área **Acciones** se eliminará y se tomará el valor predeterminado.

Aplicación y almacenamiento de la configuración actual

Los botones están disponibles en la barra superior del cuadro de diálogo:

Botón	Descripción
Cargar en el dispositivo	Descarga en el dispositivo las definiciones de derechos de acceso que haya configuradas. Entrarán en vigor únicamente después de la descarga.
Cargar desde el dispositivo	Carga los derechos de acceso que se aplican actualmente al dispositivo en el cuadro de diálogo de configuración.
Guardar en disco/**Cargar desde el disco**	La configuración actual se puede guardar en un archivo XML (*.DAR) y volver a cargarse desde ese archivo. Esto es útil para establecer la misma configuración de usuarios en varios sistemas. Para este fin, se proporciona el cuadro de diálogo estándar para examinar el sistema de archivos. El filtro de archivos se establece automáticamente en *.DAR, que significa que son archivos de derechos de acceso de dispositivos.

Impresión de la definición de derechos de acceso

Para imprimir la configuración de la vista **Derechos de acceso**, ejecute el comando **Imprimir** del menú **Archivo** o el comando **Documento** del menú **Proyecto**.

¿A qué acción afecta exactamente un determinado derecho de acceso sobre un objeto?

Objetos			Acción	Derechos			
				Agregar/eliminar subobjetos	Ejecutar	Modificar	Ver
X = el derecho debe establecerse explícitamente. – = el derecho no es relevante.							

Objetos			Acción	Derechos			
Dispositivo			**Inicio de sesión**	–	–	–	X
	Registro		**Leer entradas**	–	–	–	X
	PlcLogic						
		Aplicación	**Inicio de sesión**	–	–	–	X
			Crear	–	–	X	–
			Crear subobjeto	X	–	X	–
			Eliminar	–	–	X	–
			Descargar / Cambio en línea	–	–	X	–
			Crear proyecto de inicio	–	–	X	–
			Leer variable	–	–	–	X
			Escribir variable	–	–	X	X
			Forzar variable	–	–	X	X
			Establecer y eliminar punto de interrupción	–	X	X	–
			Definir la siguiente instrucción	–	X	X	–
			Leer pila de llamadas	–	–	–	X
			Ciclo individual	–	X	–	–
			Establecer control de flujo	–	X	X	–
			Leer control de flujo	–	–	–	X
			Ejecutar/Detener	–	X	–	–
			Restablecer	–	–	X	–
	Configuración		**Leer valores de configuración**	–	–	–	X
			Escribir valores de configuración	–	–	X	–
	Administración de usuarios		**Leer configuración**	–	–	–	X
			Escribir configuración	–	–	X	–

X = el derecho debe establecerse explícitamente.
– = el derecho no es relevante.

Apéndice E
Conjuntos de características de controlador para la migración

Conjuntos de características de controlador para la migración

Controladores Twido

Controlador	Dig In	Dig Out	MOD	FC	HSC	PWM	Serie	ETH
TWDLCAA10DRF	6	4	No	3	1	0	1	No
TWDLCDA10DRF	6	4	No	3	1	0	1	No
TWDLCAA16DRF	9	7	No	3	1	0	1+1	No
TWDLCDA16DRF	9	7	No	3	1	0	1+1	No
TWDLCAA24DRF	14	10	No	3	1	0	1+1	No
TWDLCDA24DRF	14	10	No	3	1	0	1+1	No
TWDLCAA40DRF	24	16	No	4	2	2	1+1	No
TWDLCDA40DRF	24	16	No	4	2	2	1+1	No
TWDLCAE40DRF	24	16	No	4	2	2	1+1	Sí
TWDLCDE40DRF	24	16	No	4	2	2	1+1	Sí
TWDLMDA20DTK	12	8	Sí	2	2	2	1+1	No
TWDLMDA20DUK	12	8	Sí	2	2	2	1+1	No
TWDLMDA20DRT	12	8	Sí	2	2	2	1+1	No
TWDLMDA40DTK	24	16	Sí	2	2	2	1+1	No
TWDLMDA40DUK	24	16	Sí	2	2	2	1+1	No

Dig In = número de entradas digitales
Dig Out = número de salidas digitales
MOD = módulos de expansión
FC = número de contadores rápidos
HSC = número de contadores de alta velocidad
PWM = número de generadores de pulsos
Serial = número de puertos serie
ETH = puertos Ethernet

Controladores M221

Controlador	Dig In	Dig Out	Ana In	MOD	FC	HSC	PWM	Serie	ETH	CART
TM221C16R	9	7	2	TM2 / TM3	4	2	0	1	No	1
TM221C16T	9	7	2	TM2 / TM3	4	2	2	1	No	1
TM221C24R	14	10	2	TM2 / TM3	4	2	0	1	No	1
TM221C24T	14	10	2	TM2 / TM3	4	2	2	1	No	1
TM221C40R	24	16	2	TM2 / TM3	4	2	0	1	No	2
TM221C40T	24	16	2	TM2 / TM3	4	2	2	1	No	2
TM221CE16R	9	7	2	TM2 / TM3	4	2	0	1	Sí	1
TM221CE16T	9	7	2	TM2 / TM3	4	2	2	1	Sí	1
TM221CE24R	14	10	2	TM2 / TM3	4	2	0	1	Sí	1
TM221CE24T	14	10	2	TM2 / TM3	4	2	2	1	Sí	1
TM221CE40R	24	16	2	TM2 / TM3	4	2	0	1	Sí	2
TM221CE40T	24	16	2	TM2 / TM3	4	2	2	1	Sí	2
TM221M16R/G	8	8	2	TM2 / TM3	4	2	0	2	No	0
TM221M16T/G	8	8	2	TM2 / TM3	4	2	2	2	No	0
TM221M32TK	16	16	2	TM2 / TM3	4	2	2	2	No	0
TM221ME16R/G	8	8	2	TM2 / TM3	4	2	0	1	Sí	0
TM221ME16T/G	8	8	2	TM2 / TM3	4	2	2	1	Sí	0
TM221ME32TK	16	16	2	TM2 / TM3	4	2	2	1	Sí	0

Dig In = número de entradas digitales
Dig Out = número de salidas digitales
Ana In = número de entradas analógicas
MOD = módulos de expansión
FC = número de contadores rápidos
HSC = número de contadores de alta velocidad
PWM = número de generadores de pulsos
Serial = número de puertos serie
ETH = puertos Ethernet
CART = número de cartuchos

Controladores SoMachine

Controlador	Dig In	Dig Out	Ana In	MOD	FC	HSC	PWM	Serie	ETH	CART
TM241C24R	14	10	0	TM2 / TM3	4	2	2	2	No	1
TM241C24T/U	14	10	0	TM2 / TM3	4	2	2	2	No	1
TM241C40R	24	16	0	TM2 / TM3	4	2	2	2	No	2
TM241C40T/U	24	16	0	TM2 / TM3	4	2	2	2	No	2
TM241CE24R	14	10	0	TM2 / TM3	4	2	2	2	Sí	1
TM241CE24T/U	14	10	0	TM2 / TM3	4	2	2	2	Sí	1
TM241CE40R	24	16	0	TM2 / TM3	4	2	2	2	Sí	2
TM241CE40T/U	24	16	0	TM2 / TM3	4	2	2	2	Sí	2
TM241CEC24R	14	10	0	TM2 / TM3	4	2	2	2	Sí	1
TM241CEC24T/U	14	10	0	TM2 / TM3	4	2	2	2	Sí	1
HMISCU•A5	16	10	0	No	2	1	2	1	Sí	0
HMISCU•B5	8	8	2	No	2	1	2	1	Sí	0

Dig In = número de entradas digitales
Dig Out = número de salidas digitales
Ana In = número de entradas analógicas
MOD = módulos de expansión
FC = número de contadores rápidos
HSC = número de contadores de alta velocidad
PWM = número de generadores de pulsos
Serial = número de puertos serie
ETH = puertos Ethernet
CART = número de cartuchos

Glosario

C

CFC

(*diagrama de función continua*) Un lenguaje de programación (una ampliación del estándar IEC 61131-3) basado en el lenguaje de diagrama de bloque de funciones (FBD) y que funciona como un diagrama de flujo. Sin embargo, no se utiliza ninguna red y es posible un posicionamiento libre de elementos gráficos, lo que permite bucles de realimentación. En cada bloque, las entradas se sitúan a la izquierda y las salidas, a la derecha. Las salidas del bloque se pueden conectar a las entradas de otros bloques para formar expresiones complejas.

D

dirección MAC

(*dirección de control de acceso a medios*) Un número único de 48 bits asociado a una parte específica del hardware. La dirección MAC se programa en cada tarjeta de red o dispositivo cuando se fabrica.

DTM

(*gestor de tipos de dispositivo*) Clasificado en dos categorías:

- Los DTMs del dispositivo se conectan a los componentes de configuración del dispositivo de campo.
- Los CommDTMs se conectan a los componentes de comunicaciones del software.

El DTM ofrece una estructura unificada para acceder a los parámetros de dispositivo, además de configurar, utilizar y diagnosticar los dispositivos. Los DTMs pueden incluir desde una simple interfaz gráfica de usuario para configurar parámetros de dispositivo hasta una aplicación sofisticada que permite realizar cálculos complejos en tiempo real con fines de diagnóstico y mantenimiento.

DUT

(*tipos de datos*) Junto con los tipos de datos estándar, el usuario puede definir estructuras de tipo de datos propio, tipos de enumeración y referencias como unidades de tipos de datos en un editor DUT.

E

elemento

El nombre abreviado de ARRAY.

F

FBD

(*diagrama de bloques de funciones*) Uno de los cinco lenguajes para lógica o control que cumplen con el estándar IEC 61131-3 para sistemas de control. El diagrama de bloques de funciones es un lenguaje de programación orientado gráficamente. Funciona con una lista de redes en la que cada red contiene una estructura gráfica de cuadros y líneas de conexión que representa una expresión lógica o aritmética, la llamada de un bloque de funciones, un salto o una instrucción de retorno.

FDT

(*herramienta de dispositivo de campo*) La especificación que describe el intercambio de datos estandarizado entre los dispositivos y el sistema de control o las herramientas de ingeniería o de gestión de archivos.

G

GVL

(*lista de variables globales*) Gestiona variables globales que se pueden transferir entre controladores en una red Ethernet TCP/IP Modbus.

I

IL

(*lista de instrucciones*) Un programa escrito en lenguaje que se compone de una serie de instrucciones basadas en texto y ejecutadas secuencialmente por el controlador. Cada instrucción incluye un número de línea, un código de instrucción y un operando (consulte IEC 61131-3).

L

LD

(*diagrama de contactos*) Una representación gráfica de instrucciones de un programa de controlador con símbolos para contactos, bobinas y bloques en una serie de escalones ejecutados de forma secuencial por un controlador (consulte IEC 61131-3).

P

POU

(*unidad de organización de programas*) Una declaración variable en el código fuente y el conjunto de instrucciones correspondiente. Las POUs facilitan la reutilización modular de programas de software, funciones y bloques de funciones. Una vez declaradas, cada una de las POUs está disponible para las otras.

R

RTS

(*petición de envío*) Una señal de transmisión de datos y señal CTS que reconoce la señal RTS desde el nodo de destino.

S

SFC

(*diagrama funcional secuencial*) Un lenguaje formado por pasos con acciones asociadas, transiciones con una condición lógica asociada y enlaces dirigidos entre pasos y transiciones. (La norma SFC está definida en IEC 848. Es conforme con IEC 61131-3.)

U

UDP

(*protocolo de datagramas de usuario*) Un protocolo de modalidades sin conexión (definido por la IETF RFC 768) en el que los mensajes se entregan en un datagrama (telegrama de datos) a un ordenador de destino de una red IP. El protocolo UDP generalmente se integra con el protocolo de Internet. Los mensajes UDP/IP no necesitan una respuesta y, por lo tanto, son perfectos para aplicaciones en las que los paquetes cerrados no requieren retransmisión (como redes y vídeos que necesitan rendimiento en tiempo real).

Índice

Symbols

A

B

C

D

E

F

G

I

L

M

N

O

P

R

S

T

U

V

W