SIEMENS

SINUMERIK 828D
SINUMERIK Integrate Run
MyScreens

Manual de programación

Válido para

software CNC versión 4.95

10/2020
6FC5398-4EP40-0EA0

Notas jurídicas

Filosofía en la señalización de advertencias y peligros

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

⚠ **PELIGRO**
Significa que si no se adoptan las medidas preventivas adecuadas **se producirá** la muerte o bien lesiones corporales graves.

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

⚠ **PRECAUCIÓN**
Significa que si no se adoptan las medidas preventivas adecuadas pueden producirse lesiones corporales.

ATENCIÓN
Significa que si no se adoptan las medidas preventivas adecuadas pueden producirse daños materiales.

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

Personal cualificado

El producto/sistema tratado en esta documentación sólo deberá ser manejado o manipulado por **personal cualificado** para la tarea encomendada y observando lo indicado en la documentación correspondiente a la misma, particularmente las consignas de seguridad y advertencias en ella incluidas. Debido a su formación y experiencia, el personal cualificado está en condiciones de reconocer riesgos resultantes del manejo o manipulación de dichos productos/sistemas y de evitar posibles peligros.

Uso previsto de los productos de Siemens

Considere lo siguiente:

⚠ **ADVERTENCIA**
Los productos de Siemens sólo deberán usarse para los casos de aplicación previstos en el catálogo y la documentación técnica asociada. De usarse productos y componentes de terceros, éstos deberán haber sido recomendados u homologados por Siemens. El funcionamiento correcto y seguro de los productos exige que su transporte, almacenamiento, instalación, montaje, manejo y mantenimiento hayan sido realizados de forma correcta. Es preciso respetar las condiciones ambientales permitidas. También deberán seguirse las indicaciones y advertencias que figuran en la documentación asociada.

Marcas registradas

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

Exención de responsabilidad

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

Siemens AG
Digital Industries
Postfach 48 48
90026 NÜRNBERG
ALEMANIA

Referencia del documento: 6FC5398-4EP40-0EA0
Ⓟ 07/2021 Sujeto a cambios sin previo aviso

Índice

Introducción 1

1.1 Acerca de SINUMERIK

Tanto para máquinas con CNC simples o máquinas estandarizadas como para máquinas modulares de clase "premium", los controles CNC SINUMERIK ofrecen la solución adecuada para cada tipo de máquina. Ya sea para producción en masa o de piezas únicas, simples o complejas: SINUMERIK es un control de alta productividad apto por igual para todas las áreas de fabricación, desde la de piezas de muestras y utillaje, pasando por la de moldes, y llegando a la de grandes series.

Para más información visite la página web de SINUMERIK (https://www.siemens.com/sinumerik).

1.2 Acerca de esta documentación

Destinatarios

La presente documentación está destinada a programadores y proyectistas.

Finalidad

El Manual de programación capacita a los destinatarios para diseñar, escribir y probar programas e interfaces de software y para resolver errores.

Alcance estándar

La presente documentación contiene una descripción de las funciones del alcance estándar. Este puede diferir del alcance de las funciones del sistema suministrado. Las funciones del sistema suministrado se obtienen exclusivamente de la documentación del pedido.

En el sistema pueden ejecutarse otras funciones adicionales no descritas en la presente documentación. Sin embargo, no existe derecho a reclamar estas funciones en nuevos suministros o en intervenciones de servicio técnico.

Por motivos de claridad expositiva, puede que en esta documentación no se detallen todos los datos referentes a todos los tipos de producto. Tampoco se pueden considerar aquí todos los casos posibles de instalación, servicio y mantenimiento.

El fabricante de la máquina documentará las posibles ampliaciones o modificaciones que realice en el producto.

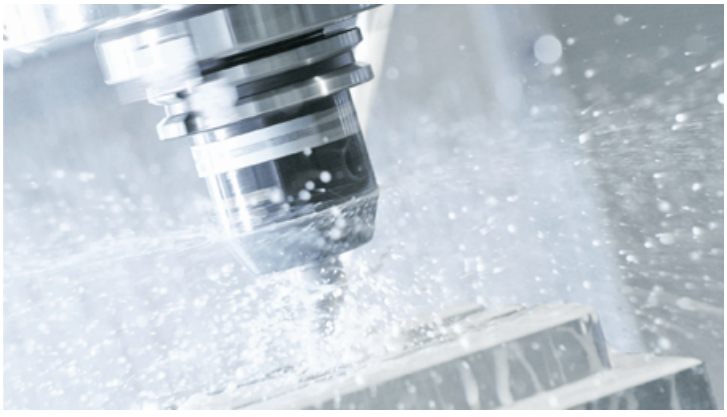
Páginas web de terceros

El presente documento puede contener enlaces a páginas web de terceros. Siemens no asume responsabilidad alguna por los contenidos de dichas páginas web ni comparte necesariamente los contenidos ni las opiniones vertidos en ellas. Siemens no controla la información publicada en estas páginas web ni tampoco es responsable del contenido o la información que ponen a disposición. Cualquier riesgo asociado a su uso es responsabilidad del usuario.

1.3 Documentación en Internet

1.3.1 Vista general de la documentación de SINUMERIK 828D

Una amplia documentación sobre las funciones de SINUMERIK 828D a partir de la versión 4.8 SP4 se encuentra en Vista general de la documentación 828D (https://support.industry.siemens.com/cs/ww/en/view/109766724).



Puede visualizar los documentos o descargarlos en formato PDF o HTML5.

La documentación se divide en las siguientes categorías:

- Usuario: Manejo
- Usuario: Programación
- Fabricante/servicio: Configuración
- Fabricante/servicio: Puesta en marcha
- Fabricante/servicio: Funciones
- Fabricante/servicio: Safety Integrated
- SINUMERIK Integrate/MindApp
- Información y cursos

1.3.2 Vista general de la documentación sobre componentes de manejo SINUMERIK

Una amplia documentación sobre los componentes de manejo SINUMERIK se encuentra en Vista general de la documentación sobre componentes de manejo SINUMERIK (https://support.industry.siemens.com/cs/document/109783841/technische-dokumentation-zu-sinumerik-bedienkomponenten?dti=0&lc=en-WW).

Puede visualizar los documentos o descargarlos en formato PDF o HTML5.

La documentación se divide en las siguientes categorías:

- Paneles de operador
- Paneles de mando de máquina
- Panel de botones de la máquina
- Unidad portátil / Miniequipos portátiles
- Más componentes de manejo

Una vista general de los documentos, artículos y enlaces más importantes sobre el tema SINUMERIK se encuentra en Página temática en vista general de SINUMERIK (https://support.industry.siemens.com/cs/document/109766201/sinumerik-an-overview-of-the-most-important-documents-and-links?dti=0&lc=en-WW).

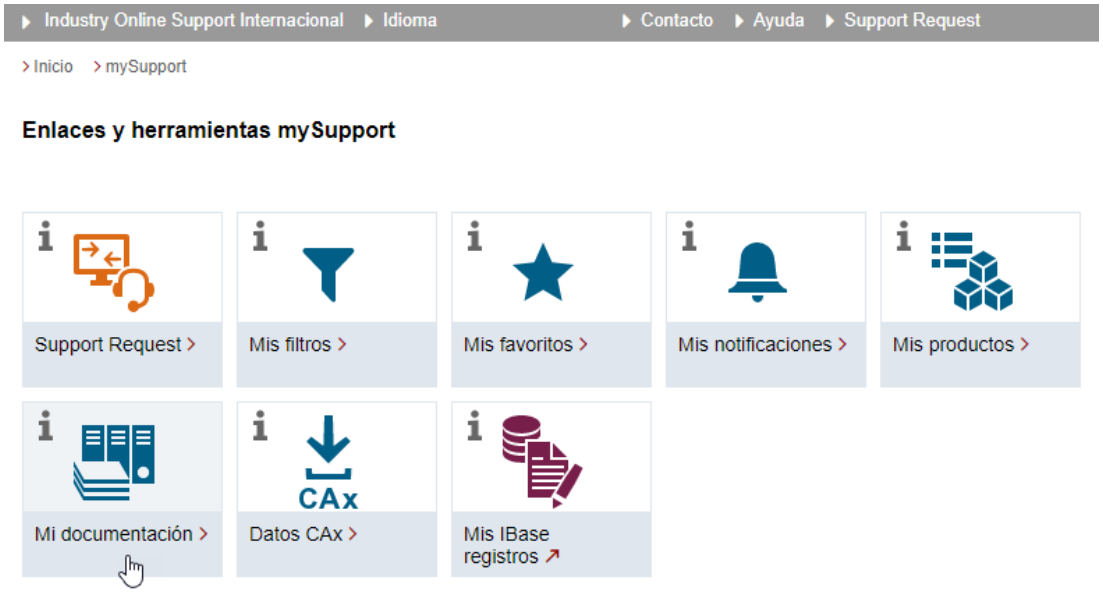
1.4 Opinión sobre la documentación técnica

En caso de preguntas, sugerencias o correcciones relacionadas con la documentación técnica publicada en el Siemens Industry Online Support, utilice el enlace "Enviar feedback" que figura al final del artículo.

1.5 Documentación de mySupport

El sistema basado en la web "Documentación de mySupport" permite recopilar de manera personalizada documentación basada en los contenidos de Siemens y adaptarla a la documentación propia de la máquina.

La aplicación se inicia mediante el icono "Mi documentación" en la página de inicio de mySupport (https://support.industry.siemens.com/cs/ww/es/my):



El manual configurado puede exportarse a los formatos RTF, PDF o XML.

Nota

Los contenidos de Siemens que soportan la aplicación Documentación de mySupport se identifican por la presencia del enlace "Configurar".

1.6 Service and Support

Product Support

Encontrará más información sobre el producto en Internet:

Product Support (https://support.industry.siemens.com/cs/ww/es/)

En esta dirección encontrará lo siguiente:

- Información actual sobre productos (notificaciones sobre productos)
- FAQ (preguntas frecuentes)
- Manuales
- Descargas
- Newsletter con la información más reciente sobre sus productos
- Foro de intercambio a escala mundial de información y experiencia para usuarios y expertos
- Personas de contacto locales a través de nuestra base de datos de personas de contacto (→ "Contacto")
- Información sobre servicio técnico in situ, reparaciones, repuestos y mucho más (→ "Servicios")

Technical Support

Los números de teléfono específicos de cada país para el asesoramiento técnico se encuentran en Internet, en la dirección (https://support.industry.siemens.com/cs/ww/es/sc/4868), en el área "Contacto".

Para formular una pregunta técnica, utilice el formulario online en el área "Support Request".

Formación

En esta dirección (https://www.siemens.com/sitrain) encontrará información sobre SITRAIN. SITRAIN ofrece formación sobre productos de Siemens, sistemas y soluciones de accionamientos y automatización.

Siemens Support en cualquier lugar

Con la galardonada aplicación "Siemens Industry Online Support" se puede acceder en cualquier momento y lugar a más de 300.000 documentos sobre productos de Siemens Industry. La aplicación le ofrece asistencia, entre otros, en los siguientes campos de aplicación:

- Solución de problemas en la implementación de un proyecto
- Eliminación de errores en caso de anomalías
- Ampliación o rediseño de una instalación

Asimismo, tendrá acceso al foro técnico y a otros artículos redactados por nuestros expertos para usted:

- FAQ
- Ejemplos de aplicación
- Manuales
- Certificados
- Información sobre productos y mucho más

La aplicación "Siemens Industry Online Support" está disponible para Apple iOS y Android.

Código de matriz de datos de la placa de características

El código de matriz de datos de la placa de características contiene los datos específicos del equipo. Este código se puede leer con cualquier smartphone y, a través de la aplicación móvil "Industry Online Support", permite visualizar información técnica sobre el equipo correspondiente.

1.7 Información importante del producto

Utilización de OpenSSL

Este producto puede contener el software siguiente:

- Software desarrollado por el Proyecto OpenSSL para su uso en el toolkit OpenSSL
- Software criptográfico creado por Eric Young
- Software desarrollado por Eric Young

Encontrará más información en Internet:

- OpenSSL (https://www.openssl.org)
- Cryptsoft (https://www.cryptsoft.com)

Cumplimiento del reglamento general de protección de datos

Siemens respeta los principios básicos de la protección de datos, en especial los preceptos relativos a la minimización de datos (privacy by design).

Para este producto, esto significa:

El producto no procesa ni almacena datos personales, únicamente datos técnicos asociados a las funciones (p. ej., etiquetas de fecha/hora). Si el usuario enlaza estos datos con otros datos (p. ej., horarios de turnos) o almacena datos personales en el mismo medio (p. ej., disco duro), creando de esta manera un vínculo con personas específicas, deberá garantizar él mismo el cumplimiento de las prescripciones legales relativas a la protección de datos.

Consignas básicas de seguridad 2

2.1 Consignas generales de seguridad

⚠ ADVERTENCIA

Peligro de muerte en caso de incumplimiento de las consignas de seguridad e inobservancia de los riesgos residuales

Si no se cumplen las consignas de seguridad ni se tienen en cuenta los riesgos residuales de la documentación de hardware correspondiente, pueden producirse accidentes con consecuencias mortales o lesiones graves.

- Respete las consignas de seguridad de la documentación de hardware.
- Tenga en cuenta los riesgos residuales durante la evaluación de riesgos.

⚠ ADVERTENCIA

Fallos de funcionamiento de la máquina a consecuencia de una parametrización errónea o modificada

Una parametrización errónea o modificada puede provocar en máquinas fallos de funcionamiento que pueden producir lesiones graves o la muerte.

- Proteja la parametrización del acceso no autorizado.
- Controle los posibles fallos de funcionamiento con medidas apropiadas, p. ej., DESCONEXIÓN o PARADA DE EMERGENCIA.

2.2 Garantía y responsabilidad para ejemplos de aplicación

Los ejemplos de aplicación no son vinculantes y no pretenden ser completos en cuanto a la configuración y al equipamiento, así como a cualquier eventualidad. Los ejemplos de aplicación tampoco representan una solución específica para el cliente; simplemente ofrecen una ayuda para tareas típicas.

El usuario es responsable del correcto manejo y uso de los productos descritos. Los ejemplos de aplicación no le eximen de la obligación de trabajar de forma segura durante la aplicación, la instalación, el funcionamiento y el mantenimiento.

2.3 Información de seguridad

Siemens ofrece productos y soluciones con funciones de seguridad industrial con el objetivo de hacer más seguro el funcionamiento de instalaciones, sistemas, máquinas y redes.

Para proteger las instalaciones, los sistemas, las máquinas y las redes contra de amenazas cibernéticas, es necesario implementar (y mantener continuamente) un concepto de seguridad industrial integral que este conforme al estado del arte. Los productos y las soluciones de Siemens constituyen una parte de este concepto.

Los clientes son responsables de impedir el acceso no autorizado a sus instalaciones, sistemas, máquinas y redes. Dichos sistemas, máquinas y componentes solo deben estar conectados a la red corporativa o a Internet cuando y en la medida que sea necesario y siempre que se hayan tomado las medidas de protección adecuadas (p. ej. cortafuegos y segmentación de la red).

Para obtener información adicional sobre las medidas de seguridad industrial que podrían ser implementadas, por favor visite https://www.siemens.com/industrialsecurity (https://www.siemens.com/industrialsecurity).

Los productos y las soluciones de Siemens están sometidos a un desarrollo constante con el fin de hacerlos más seguros. Siemens recomienda expresamente realizar actualizaciones en cuanto estén disponibles y utilizar únicamente las últimas versiones de los productos. El uso de versiones de los productos anteriores o que ya no sean soportadas y la falta de aplicación de las nuevas actualizaciones, puede aumentar el riesgo de amenazas cibernéticas.

Para mantenerse informado de las actualizaciones de productos, recomendamos que se suscriba al Siemens Industrial Security RSS Feed en https://www.siemens.com/industrialsecurity (https://new.siemens.com/global/en/products/services/cert.html#Subscriptions).

Encontrará más información en Internet:

Manual de configuración de Industrial Security (https://support.industry.siemens.com/cs/ww/es/view/108862708/en)

⚠ ADVERTENCIA

Estados operativos no seguros debidos a una manipulación del software

Las manipulaciones del software (p. ej. mediante virus, troyanos o gusanos) pueden provocar estados operativos inseguros en la instalación, con consecuencias mortales, lesiones graves o daños materiales.

- Mantenga actualizado el software.
- Integre los componentes de automatización y accionamiento en un sistema global de seguridad industrial de la instalación o máquina conforme a las últimas tecnologías.
- En su sistema global de seguridad industrial, tenga en cuenta todos los productos utilizados.
- Proteja los archivos almacenados en dispositivos de almacenamiento extraíbles contra software malicioso tomando las correspondientes medidas de protección, p. ej. programas antivirus.
- Al finalizar la puesta en marcha, compruebe todos los ajustes relevantes para la seguridad.

Funcionalidad

3

3.1 Funciones de "Run MyScreens"

Vista general

Con "Run MyScreens" es posible configurar interfaces de usuario que representan ampliaciones de funciones específicas del fabricante de la máquina o del usuario o que implementan un diseño específico del usuario.

Con las interfaces de usuario creadas por el usuario pueden generarse, entre otros, llamadas de ciclos. Los diálogos se pueden diseñar directamente en el control.

"Run MyScreens" se implementa mediante un intérprete y ficheros de configuración que contienen la descripción de las interfaces de usuario.

"Run MyScreens" se configura mediante ficheros ASCII: estos ficheros de configuración contienen la descripción de la interfaz de usuario. La sintaxis necesaria para crear los ficheros se describe en los siguientes capítulos.

Ejecución básica

Para que el fabricante de la máquina pueda configurar sus propios diálogos, está disponible la función "Run MyScreens". Ya en la ejecución básica pueden configurarse 5 diálogos en el árbol de menús del manejo o para diálogos de ciclos específicos del cliente.

Opción de software

Para aumentar el número de cuadros de diálogo se requiere una de las siguientes opciones de software:

- SINUMERIK 828D/840D sl, SINUMERIK Integrate Run MyScreens (6FC5800-0AP64-0YB0)
- SINUMERIK 840D sl, SINUMERIK Integrate Run MyHMI/3GL (6FC5800-0AP60-0YB0)
- SINUMERIK 840D sl, SINUMERIK Integrate Run MyHMI/WinCC (6FC5800-0AP61-0YB0)

Limitaciones

Se deben tener en cuenta las condiciones siguientes:

- El cambio de diálogo sólo es posible dentro de un campo de manejo.
- Las variables de usuario no deben tener el mismo nombre que las variables de sistema o de PLC (ver también Manual de listas Variables del sistema /PGAsl/).
- Los diálogos activados por el PLC forman su propio campo de manejo (al igual que las imágenes de ciclo de medida).

Herramientas

- Editor compatible con UTF8 (p. ej. el editor integrado de la interfaz de usuario o el bloc de notas)
- Para crear gráficos o imágenes se necesita un programa de elaboración de gráficos.

Nombres de fichero y codificación

Nota

Al utilizar HMI Operate en la NCU, tenga en cuenta que todos los nombres de fichero se escriben en minúsculas en la tarjeta CF (com, png, txt). Esto es necesario debido a Linux.

En la PCU pueden escribirse los nombres de fichero en mayúsculas o minúsculas indistintamente. Sin embargo, se recomienda, p. ej. para una posible transferencia posterior a Linux, utilizar también aquí solo minúsculas.

Nota

Al guardar los ficheros de configuración e idioma, asegúrese de que la codificación esté ajustada a UTF-8 en el editor que utiliza.

Rutas de almacenamiento

Al almacenar los ficheros de configuración, ficheros de ayuda, etc., tenga en cuenta la siguiente convención:

[Directorio Siemens del sistema]		
	Linux:	/card/siemens/sinumerik/hmi
	Windows 7:	C:\ProgramData\Siemens\MotionControl\siemens
[Directorio OEM del sistema]		
	Linux:	/card/oem/sinumerik/hmi
	Windows 7:	C:\ProgramData\Siemens\MotionControl\oem
[Directorio de usuario del sistema]		
	Linux:	/card/user/sinumerik/hmi
	Windows 7:	C:\ProgramData\Siemens\MotionControl\user
[Directorio de addon del sistema]		
	Linux:	/card/addon/sinumerik/hmi
	Windows 7:	C:\ProgramData\Siemens\MotionControl\addon
Lugares de almacenamiento		
	Configuración de "Run MyScreens":	[Directorio OEM del sistema]/proj
	Fichero de configuración:	[Directorio OEM del sistema]/cfg
	Ficheros de idioma:	[Directorio OEM del sistema]/lng

	Ficheros de imagen:	[Directorio OEM del sistema]/Ico/ico[Resolución] Ejemplo: [Directorio OEM del sistema]/Ico/ico640
	Ayuda en línea:	[Directorio OEM del sistema]/hlp/[Idioma] Ejemplo: [Directorio OEM del sistema]/hlp/eng

Utilización

Se pueden realizar las siguientes funciones:

- Visualizar y preparar diálogos de:
  - Pulsadores de menú
  - Variables
  - Texto y texto de ayuda
  - Gráficos y pantallas de ayuda
- Abrir cuadros de diálogo mediante:
  - Accionamiento de un pulsador de menú (de inicio)
  - Selección de PLC/CN
- Cambiar dinámicamente la estructura de los diálogos:
  - Modificar y borrar pulsadores de menú
  - Definir y diseñar campos de variables
  - Mostrar, sustituir y borrar textos de visualización (dependientes o independientes del idioma)
  - Mostrar, sustituir y borrar gráficos
- Poner en marcha acciones al:
  - Visualizar cuadros de diálogo
  - Introducir valores (variables)
  - Accionar pulsadores de menú
  - Abandonar diálogos
  - Cambio de valor de una variable de sistema o de usuario
- Intercambio de datos entre cuadros de diálogo
- Variables:
  - Leer (variables de CN, PLC, usuario)
  - Escribir (variables de CN, PLC, usuario)
  - Vincular por medio de operadores matemáticos, comparativos o lógicos

- Ejecutar funciones:
  - Subprogramas
  - Funciones de ficheros
  - Servicios de PI
- Considerar niveles de protección según los grupos de usuario

Primeros pasos

4

4.1 Introducción

Por medio del siguiente ejemplo se explican los pasos necesarios para insertar diálogos propios en la interfaz de usuario de SINUMERIK Operate mediante Run MySreens. Entre otras cosas aprenderá a crear diálogos propios, a insertar accesos a la ayuda y pantallas de ayuda contextuales, a definir pulsadores de menú y a navegar entre los diálogos.

4.2 Ejemplo de proyecto

4.2.1 Descripción de la tarea

En el ejemplo de proyecto, creará los cuadros de diálogo que se describen a continuación, incluida una ayuda contextual.

Diálogo 1

En el primer diálogo se muestran los parámetros R editables (0 y 1) y los nombres de ejes geométricos (campos de entrada). Para los dos parámetros R se integran pantallas de ayuda correspondientes. Para los ejes geométricos se integra una ayuda contextual. Además, el cuadro de diálogo contiene ejemplos de líneas de separación (horizontales y verticales), campos de alternancia, campos de entrada/salida con selección de unidades integrada y de barras de progreso (con y sin cambio de color).

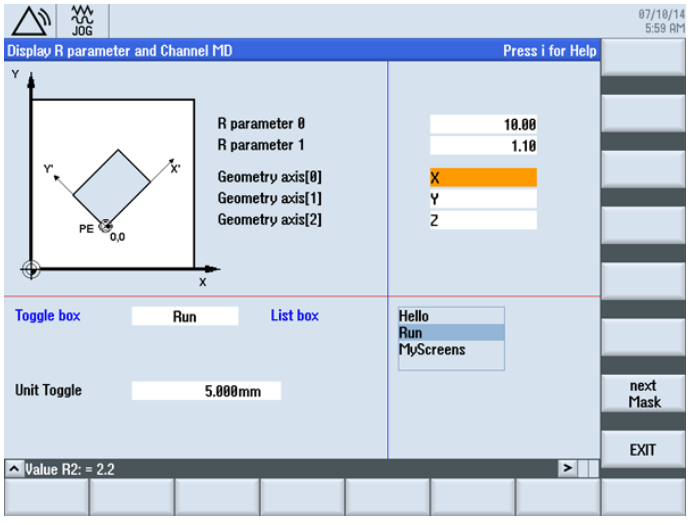


Figura 4-1 Parámetros R con pantallas de ayuda

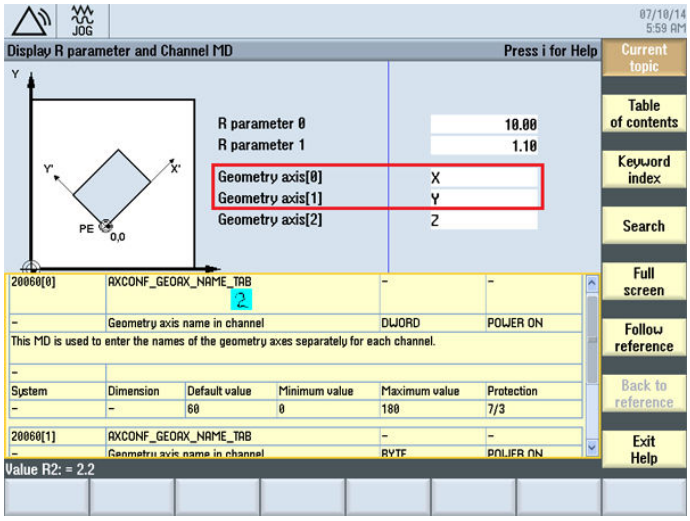


Figura 4-2 Ejes geométricos con ayuda contextual

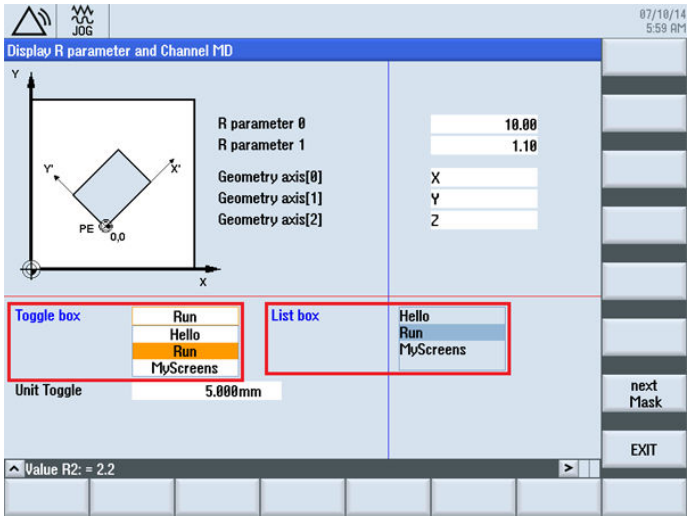


Figura 4-3 Campos de alternancia: lista y cuadro

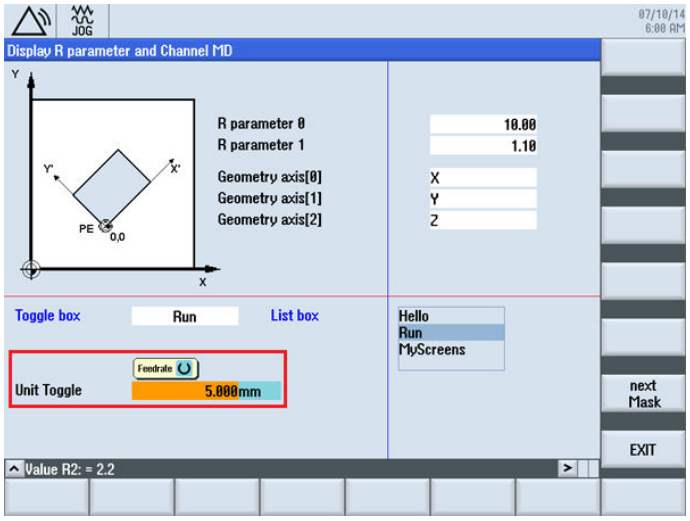


Figura 4-4 Campo de entrada/salida con selección de unidades integrada

Diálogo 2

En el segundo cuadro de diálogo se muestran los valores MKS y WKS. Además, el cuadro de diálogo contiene ejemplos de un indicador de progreso con y sin cambio de color.

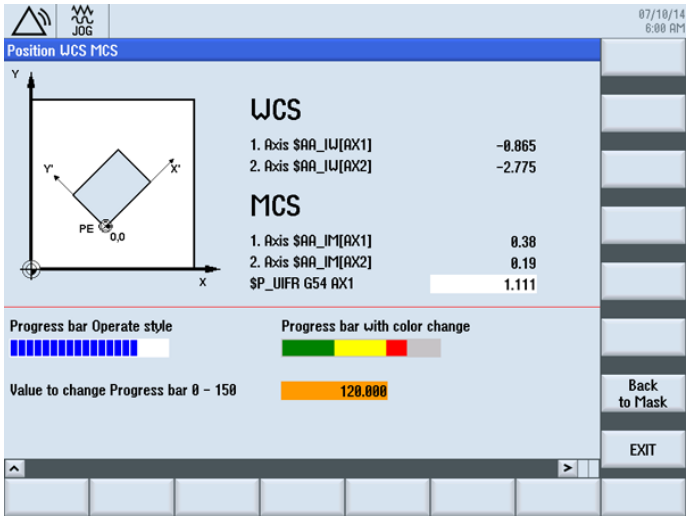


Figura 4-5 Barras de progreso con (derecha) y sin (izquierda) cambio de color

Navegación

El primer diálogo se llama con el pulsador de menú de inicio "START" del campo de manejo Diagnóstico. Se utiliza el pulsador de menú horizontal SK7.

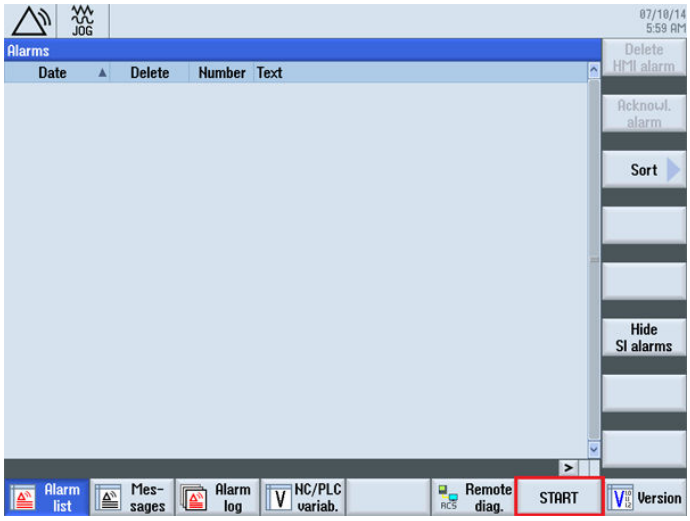


Figura 4-6 Pulsador de menú de inicio „START" del campo de manejo Máquina, modo de operación AUTO

Desde el primer diálogo se puede llamar al segundo con el pulsador de menú "Next Mask". Se vuelve a la pantalla básica del campo de manejo con el pulsador de menú "EXIT" (ver imagen de arriba).

Desde el segundo diálogo también puede volverse a la pantalla básica del campo de manejo con el pulsador de menú "EXIT" (ver imagen de arriba). Con el pulsador de menú "Back to Mask" se puede volver al primer diálogo.

Procedimiento

En los siguientes capítulos se explican los pasos que hay que seguir:

1. Creación del fichero de configuración (fichero COM) (Página 25)
2. Almacenamiento del fichero de configuración en el directorio OEM (Página 28)
3. Creación de la ayuda en línea (Página 29)
4. Integración de la ayuda en línea y almacenamiento de los ficheros en el directorio OEM (Página 32)
5. Copia de "easyscreen.ini" al directorio OEM (Página 33)
6. Declaración del fichero COM en "easyscreen.ini" (Página 33)
7. Comprobación del proyecto (Página 33)

4.2.2 Creación del fichero de configuración (ejemplo)

Contenido del fichero de configuración

Cree el fichero de configuración "diag.com" para los dos diálogos en un editor compatible con UTF8.

```
; Identificador de arranque del pulsador de menú de inicio
//S(START)
   ; Pulsador de menú de inicio solo texto
   HS7=("START")
   ; Pulsador de menú de inicio con texto dependiente del idioma y png
   ;HS7=([$80792,"\\sk_ok.png"])

   ; Método Press
   PRESS(HS7)
      ; Función LM o LS
      LM("MASK1")
      ; LM con indicación de un fichero com
      LM("MASK1","TEST.COM")
   END_PRESS
; Identificador de fin del pulsador de menú de inicio
//END

; Definición de diálogo 1 con título y pantalla
//M(MASK1/"Display R parameter and Channel MD"/"mz961_01.png")
   ; Definición de las variables
   DEF VAR1 = (R2///,"R parameter 0"///"$R[0]"/200,50,150/400,50,100,)
   ; con pantalla de ayuda
   DEF VAR2 = (R2///,"R parameter 1"//"mz961_02.png"/"$R[1]"/200,70,150/400,70,100)
   ; con ayuda en línea
   DEF ACHS_NAM1 = (S///"Press i for Help","Geometry axis[0]"////200,100,150/400,100,100//"sinume-
rik_md_1.html","20060[0]")
   ; con ayuda en línea
   DEF ACHS_NAM2 = (S///"Press i for Help","Geometry axis[1]"////200,120,150/400,120,100//"sinume-
rik_md_1.html","20060[1]")
   DEF ACHS_NAM3 = (S///,"Geometry axis[2]"////200,140,150/400,140,100)
   ; Definición de campos de alternancia y conmutador de unidades
   DEF VAR_TGL = (S/* "Hello", "Run", "MyScreens"/"Run"/,"Toggle box"////10,230,100/120,230,100/,,6)

   DEF VAR_TGB = (S/* "Hello", "Run", "MyScreens"/"Run"/,"List box"/
dt4///250,230,100/370,230,100,60/,,"#0602ee")
   ; BC, FC, BC_ST, FC_ST, BC_GT, FC_GT, BC_UT, FC_UT, SC1, SC2
   DEF VarEdit = (R///,"Unit Toggle",,,"Feedrate"///"$R[11]"/5,300,100/120,300,100///"VarTgl"),
VarTgl = (S/*0="mm",1="inch"/0//WR2////220,300,40)
```

```
; Definición de pulsadores de menú en el diálogo
HS1=("")
HS2=("")
HS3=("")
HS4=("")
HS5=("")
HS6=("")
HS7=("")
HS8=("")
VS1=("")
VS2=("")
VS3=("")
VS4=("")
VS5=("")
VS6=("")
VS7=("next Mask")
VS8=("EXIT")

; Definición de LOAD Block
LOAD
   ; Lectura de valor con RNP
   ACHS_NAM1 = RNP("$MC_AXCONF_GEOAX_NAME_TAB[0]")
   ACHS_NAM2 = RNP("$MC_AXCONF_GEOAX_NAME_TAB[1]")
   ACHS_NAM3 = RNP("$MC_AXCONF_GEOAX_NAME_TAB[2]")
   ; Emisión de una línea de diálogo
   DLGL("Value R2: = " << RNP("$R[2]"))
   ; Escritura de un valor con WNP
   WNP("$R[3]",VAR0)

  ; Línea de separación vertical
  V_SEPARATOR(360,1,6,1)
  ; Línea de separación horizontal
  H_SEPARATOR(220,1,7,1)
END_LOAD

; Método Press
PRESS(VS7)
   ; Carga de otro diálogo
   LM("MASK2")
; Identificador de fin del método Press
END_PRESS

; Método Press
PRESS(VS8)
```

```
      ; Abandono del diálogo
      EXIT
   ; Identificador de fin del método Press
   END_PRESS
; Identificador de fin del diálogo 1
//END

; Definición de diálogo 2 con título y pantalla
//M(MASK2/"Position WCS MCS"/"mz961_01.png")
   ; Definición de las variables
   DEF TEXT1 = (I///,"WCS"/WR0,fs2///230,30,120/,,1)
   DEF VAR1 = (R3///,"1. Axis $AA_IW[AX1]"/WR1//"$AA_IW[AX1]"/230,70,150/400,70,100)
   DEF VAR2 = (R3///,"2. Axis $AA_IW[AX2]"/WR1//"$AA_IW[AX2]"/230,90,150/400,90,100)

   DEF TEXT2 = (I///,"MCS"/WR0,fs2///230,120,120/,,1)
   DEF VAR3 = (R2///,"1. Axis $AA_IM[AX1]"/WR1//"$AA_IM[AX1]"/230,160,150/400,160,100)
   DEF VAR4 = (R2///,"2. Axis $AA_IM[AX2]"/WR1//"$AA_IM[AX2]"/230,180,150/400,180,100)
   DEF VAR5 = (R3///,"$P_UIFR G54 AX1"///"$P_UIFR[1,AX1,TR]"/230.200.150/400,200,100)

   ; Definición de las barras de progreso
   DEF PROGGY0 = (R/0,150//,"Progress bar Operate style"/DT2,DO0//"$R[10]"/5,240,190/5,260,150/6,10)
   DEF PROGGY4 = (R/0,150,50,100/120/,"Progress bar with color change"/DT1,DO0//"$R[10]"/
260.240.190/260,260,150/3,4,,,9,7)

   ; Definición de las variables
   DEF PROGVAL = (R/0,150//,"Value to change Progress bar 0 - 150"///"$R[10]"/5,300,230/260,300,100)

  ; Definición de pulsadores de menú en el diálogo
  HS1=("")
  HS2=("")
  HS3=("")
  HS4=("")
  HS5=("")
  HS6=("")
  HS7=("")
  HS8=("")
  VS1=("")
  VS2=("")
  VS3=("")
  VS4=("")
  VS5=("")
  VS6=("")
  VS7=("Back to Mask")
  VS8=("EXIT")
```

```
   ; Método Load
   LOAD
      H_SEPARATOR(230,1,7,1)
   END_LOAD

   ; Método Change
   CHANGE(VAR5)
      ; Función PI_START
      PI_START("/NC,201,_N_SETUFR")
   ; Identificador de fin del método Change
   END_CHANGE

   ; Método Press
   PRESS(RECALL)
      ; Retorno a máscara invocante
      LM("MASK1")
   ; Identificador de fin del método Press
   END_PRESS

   ; Método Press
   PRESS(VS7)
      ; Retorno a máscara invocante
      LM("MASK1")
   ; Identificador de fin del método Press
   END_PRESS

   ; Método Press
   PRESS(VS8)
      ; Abandono del diálogo y retorno a aplicación estándar
      EXIT
   ; Identificador de fin del método Press
   END_PRESS
; Identificador de fin del diálogo
//END
```

4.2.3 Almacenamiento del fichero de configuración en el directorio OEM (ejemplo)

Ruta de almacenamiento

Guarde el fichero de configuración "diag.com" en la siguiente ruta:

[Directorio OEM del sistema]/proj

4.2.4 Creación de la ayuda en línea (ejemplo)

Cree el fichero HTML "sinumerik_md_1.html". El ejemplo debajo de "Contenido de la ayuda en línea" muestra el acceso a la ayuda contextual a través de **name="20060[0]"** y **name="20060[1]"**.

Indicaciones

Encontrará instrucciones para crear ficheros HTML en el capítulo Crear ficheros HTML (Página 75).

Las indicaciones relativas a la integración de la ayuda en línea se encuentran en el capítulo Integración de la ayuda en línea y almacenamiento de los ficheros en el directorio OEM (ejemplo) (Página 32).

Contenido de la ayuda en línea

El ejemplo no está comentado.

```
<html><head><meta http-equiv="Content-Type" content="text/html; charset="UTF-8"/><title></
title></head>
   <body>
      <table border="1" cellspacing="0" cellpadding="2" width="100%">
         <tr>
            <td><a name="20060[0]"><b>20060[0]</b></a></td>
            <td colspan="3">AXCONF_GEOAX_NAME_TAB</td>
            <center>
               <img src="../picture/bitmap3.png">
            </center>
            <td>-</td>
            <td>-</td>
         </tr>
         <tr>
            <td>-</td>
            <td colspan="3">Geometry axis name in channel</td>
            <td>DWORD</td>
            <td>POWER ON</td>
         </tr>
         <tr>
            <td colspan="6">This MD is used to enter the names of the geometry axes
separately for each channel.<br />
         <tr>
            <td>-</td>
            <td colspan="5"> </td>
         </tr>
         <tr>
          <td width="16*">System</td><td width="16*">Dimension</td><td width="16*">Default
value</td>
            <td width="16*">Minimum value</td><td width="16*">Maximum value</td>
            <td width="16*">Protection</td>
         </tr>
         <tr>
            <td>-</td>
            <td>-</td>
            <td>60</td>
            <td>0</td>
            <td>180</td>
            <td>7/3</td>
         </tr>
      </table>
      <p></p>
      <table border="1" cellspacing="0" cellpadding="2" width="100%">
         <tr>
            <td><a name="20060[1]"><b>20060[1]</b></a></td>
            <td colspan="3">AXCONF_GEOAX_NAME_TAB</td>
            <td>-</td>
            <td>-</td>
         </tr>
         <tr>
            <td>-</td>
            <td colspan="3">Geometry axis name in channel</td><td>BYTE</td>
            <td>POWER ON</td>
         </tr>
         <tr>
```

```
            <td colspan="6">This MD is used to enter the names of the geometry axes
separately for each channel.<br />
         <tr>
            <td>-</td>
            <td colspan="5"> </td>
         </tr>
         <tr>
            <td width="16*">System</td>
            <td width="16*">Dimension</td>
            <td width="16*">Default value</td>
            <td width="16*">Minimum value</td>
            <td width="16*">Maximum value</td>
            <td width="16*">Protection</td>
         </tr>
         <tr>
            <td>-</td>
            <td>-</td>
            <td>0</td>
            <td>0</td>
            <td>2</td>
            <td>7/3</td>
         </tr>
      </table>
      <p></p>
   </body>
</html>
```

4.2.5 Integración de la ayuda en línea y almacenamiento de los ficheros en el directorio OEM (ejemplo)

Integración de la ayuda en línea

La sintaxis para la integración de la ayuda en línea es análoga a la de SINUMERIK Operate:

DEF RFP=(R//1/,"RFP","RFP"///////"sinumerik_md_1.html","9006")

Nota

¡El nombre del fichero HTML debe escribirse en minúsculas debido a LINUX!

Ruta de almacenamiento

Guarde el fichero HTML ""sinumerik_md_1.html" para la ayuda en alemán en la siguiente ruta:

[Directorio OEM del sistema]/hlp/deu

Si es necesario, deben crearse las carpetas para otros idiomas (p. ej.: chs, eng, esp, fra, ita...).

Encontrará una lista de los códigos de idioma en el anexo "Listas de referencia".

Información adicional

Encontrará información detallada sobre la creación de ayudas en línea específicas de OEM en el capítulo Configuración de la ayuda en línea (Página 74).

4.2.6 Copia de "easyscreen.ini" al directorio OEM (ejemplo)

Ruta de almacenamiento

Copie el fichero "easyscreen.ini" desde el directorio

[Directorio Siemens del sistema]/cfg

en el directorio

[Directorio OEM del sistema]/cfg.

4.2.7 Declaración del fichero COM en "easyscreen.ini" (ejemplo)

Adaptación en "easyscreen.ini"

Haga el siguiente cambio en "easyscreen.ini" en el directorio OEM. Con ello se declara el fichero de configuración "diag.com".

```
;########################
;# AREA Diagnosis       #
;########################
;<=============================================================>
;< OEM Softkey on first horizontal Main Menu                    >
;< SOFTKEY position="7"                                         >
;<=============================================================>
StartFile28 = area := AreaDiagnosis, dialog:= SlDgDialog, menu := DgGlobalHu, startfile := diag.com
```

4.2.8 Comprobación del proyecto (ejemplo)

Comprobación de llamada de diálogo

Pase al campo de manejo Diagnóstico. Haga clic en el pulsador de menú horizontal "START".

Si "Run MyScreens" detecta errores al interpretar los ficheros de configuración, estos se guardan en el fichero de texto "easyscreen_log.txt". Encontrará más información en el capítulo Tratamiento de errores (diario de incidencias) (Página 41).

Comprobación de pantallas de ayuda contextuales y ayuda en línea

Compruebe las pantallas de ayuda contextuales y la ayuda en línea. En caso de que se produzcan errores, compruebe las rutas de almacenamiento.

Conceptos básicos

5

5.1 Estructura del fichero de configuración

Introducción

La descripción de las nuevas interfaces de usuario se guarda en ficheros de configuración. Estos ficheros se interpretan automáticamente y su resultado se representa en pantalla. Los ficheros de configuración no están disponibles en el estado de entrega, por lo que primero deben crearse.

Para generar los ficheros de configuración e idioma se utiliza un editor compatible con UTF-8 (p. ej., el editor integrado en la interfaz de usuario o el bloc de notas). La descripción también puede incluir explicaciones en forma de comentarios. Antes de cada explicación se añade el carácter de comentario ";".

Nota

Al guardar los ficheros de configuración e idioma, asegúrese de que la codificación esté ajustada a UTF-8 en el editor que utiliza.

Sin embargo, las palabras clave (también en un fichero COM con codificación UTF-8) solo deben tener caracteres incluidos en el juego de caracteres ASCII. En caso contrario, no puede garantizarse la interpretación y, con ello, la visualización de máscaras/diálogos.

Configuración

Cada campo de manejo HMI dispone de pulsadores de menú de inicio fijos mediante los cuales se puede acceder a los diálogos recién creados.

Al llamar una barra de máscara de carga (LM) o un menú de pulsadores de carga (LS) en un fichero de configuración es posible indicar un nuevo nombre de fichero en el que se encuentre el objeto llamado. De esta forma se puede estructurar la configuración, por ejemplo: todas las funciones de un nivel de mando en un fichero de configuración propio.

Estructura del fichero de configuración

Un fichero de configuración se compone de los siguientes elementos:

1. Descripción de los pulsadores de menú de inicio;
2. Definición de diálogos
3. Definición de las variables
4. Descripción de los métodos
5. Definición de menús de pulsadores

Nota

Secuencia

Es imprescindible respetar la secuencia indicada en el fichero de configuración.

Ejemplo:

```
//S (START)                    ; Definición de los pulsadores de menú de inicio (opcio-
                                 nal)
....
//END
//M (.....)                    ; Definición del diálogo
DEF .....                      ; Definición de las variables
LOAD                           ; Descripción de los bloques
...
END_LOAD
UNLOAD
...
END_UNLOAD
...
//END
//S (...)                      ; Definición de un menú de pulsadores
//END
```

Almacenamiento de los ficheros de configuración

Los ficheros de configuración se guardan en el directorio [Directorio de usuario del sistema]/proj y, análogamente, también en los directorios [Directorio de addon del sistema] y [Directorio OEM del sistema].

Conversión de textos de otras aplicaciones HMI

Procedimiento para convertir un fichero de texto con codificación de página de código a codificación de texto UTF-8:

1. Abra el fichero de texto en un editor de texto de una PG/un PC.
2. Ajuste la codificación UTF-8 al guardar.

El mecanismo de lectura mediante página de código sigue siendo compatible.

5.2 Estructura del árbol de manejo

Principio del árbol de manejo

Varios diálogos vinculados entre sí conforman un árbol de manejo. Se considera que dos diálogos están vinculados cuando se puede pasar directamente de uno a otro. Los nuevos pulsadores de menú horizontales o verticales definidos en este diálogo permiten pasar al diálogo anterior o a cualquier otro.

Detrás de cada pulsador de menú de inicio se puede crear un árbol de manejo:

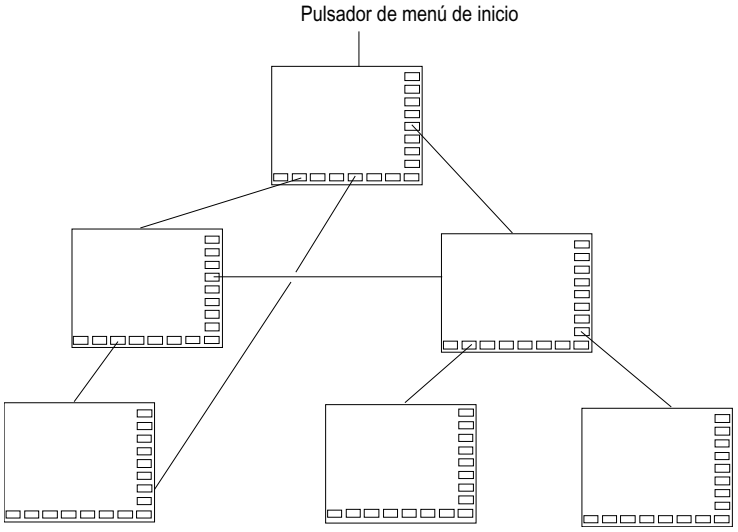


Figura 5-1 Árbol de manejo

Pulsadores de menú de inicio

En un fichero de configuración indicado en "easyscreen.ini" se definen uno o varios pulsadores de menú (pulsadores de menú de inicio) que funcionan como punto de partida para las secuencias de operaciones propias.

Al determinar un pulsador de menú se carga un diálogo propio u otro menú de pulsadores con los que se podrán ejecutar otras acciones.

Al accionar un pulsador de menú de inicio se carga el diálogo asignado. También se activan los pulsadores de menú correspondientes al diálogo. Si no se configuran posiciones específicas, las variables se emitirán en las posiciones estándar.

Retorno a la aplicación estándar

Puede salir del cuadro de diálogo recién diseñado y regresar al campo de manejo estándar.

Con la tecla <RECALL> es posible salir de las interfaces de usuario recién creadas siempre que no se haya configurado otra función para esta tecla.

Nota

Abrir diálogos en el programa de usuario del PLC

Además de los pulsadores de menú, también es posible seleccionar diálogos desde el PLC: para el intercambio de señales PLC → HMI existe una señal de interfaz en el DB19.DBB10.

5.3 Definición y funciones para los pulsadores de menú de inicio

5.3.1 Definición de un pulsador de menú de inicio

Pulsador de menú independiente del diálogo

Los pulsadores de menú de inicio son pulsadores de menú independientes del diálogo que no se llaman desde un diálogo, sino que se configuran **antes** que el primer diálogo nuevo. Para acceder al diálogo de inicio o a un menú de pulsadores de inicio, debe definirse el pulsador de menú de inicio.

Programación

El bloque descriptivo de un pulsador de menú de inicio está configurado como sigue:

```
//S(Start)                ; Identificador de arranque del pulsador de menú de inicio
HS1=(...)                 ; Definición de un pulsador de menú de inicio: pulsador de menú horizontal 1
PRESS(HS1)                ; Método
  LM...                   ; Función LM o LS
END_PRESS                 ; Fin del método
//END                     ; Identificador de fin del pulsador de menú de inicio
```

Posiciones admisibles para los pulsadores de menú de inicio

En los campos de manejo se admiten las siguientes posiciones para los pulsadores de menú de inicio de "Run MyScreens":

Campo de manejo	Posición
Máquina	HSK6
Parámetros	HSK7

Campo de manejo	Posición
Programa	HSK6 y HSK15 Ciclos de medida: HSK13 y HSK14
Gestor de programas	HSK2-8 y HSK12-16, en caso de que no esté ocupado con unidades.
Diagnóstico	HSK7
Puesta en marcha	HSK7

Los pulsadores de menú de inicio se configuran en ficheros especiales. Los nombres de estos ficheros se declaran en el fichero ""easyscreen.ini". Suelen tener un nombre específico del campo de manejo (p. ej., "startup.com" para el campo Puesta en marcha). El campo de manejo Máquina representa una excepción. En el campo de manejo Máquina existen varios ficheros específicos del modo de operación ("ma_jog.com", "ma_auto.com").

El menú de pulsadores con los pulsadores de menú de inicio se llama "Marcha". Pueden seguir utilizándose configuraciones existentes de pulsadores de menú de inicio. No se admite la funcionalidad de agrupar ("Merge") los pulsadores de menú de inicio con los pulsadores de menú de la aplicación HMI correspondiente (campo de manejo) en el menú de pulsadores de inicio.

Hasta la primera llamada de diálogo, es decir, hasta el momento a partir del cual está disponible la plena funcionalidad (p. ej., ejecución de PRESS Blocks), un menú o un menú de pulsadores solo puede sustituirse completo por otro.

Plantilla para la configuración del pulsador de menú de inicio

Encontrará una descripción detallada de todas las posiciones admisibles para los pulsadores de menú de inicio y su configuración en el fichero "easyscreen.ini", en el siguiente directorio:

[Directorio Siemens del sistema]/cfg

Este fichero sirve de plantilla para la propia configuración del pulsador de menú de inicio.

Ver también

Listas de los pulsadores de menú de inicio

5.3.2 Funciones para pulsadores de menú de inicio

Funciones para pulsadores de menú independientes del diálogo

Con los pulsadores de menú de inicio solo se pueden disparar determinadas funciones.

Se admiten las siguientes funciones:

- Con la **función LM** se puede cargar otro diálogo: **LM**(“Identificador"[,"Fichero"])
- Con la **función LS** se puede insertar otro menú de pulsadores. **LS**("Indicador"[, "Fichero"][, Merge])

- Con la **función "EXIT"** se puede salir de la interfaz de usuario recién diseñada y regresar a la aplicación estándar.
- Con la **función "EXITLS"** se puede salir de la interfaz actual de usuario y cargar un menú de pulsadores definido.

Método PRESS

El pulsador de menú se define dentro del bloque descriptivo y en el método PRESS se asigna la función "LM" o "LS".

Si la definición del pulsador de menú de inicio se marca como un comentario (punto y coma ; al comienzo de la línea) o si se elimina el fichero de configuración, el pulsador de menú no funcionará.

```
//S(Start)                          ; Identificador de arranque
HS6=("1.ª máscara")                 ; Rotular pulsador de menú horizontal 6 con "1.ª
                                    máscara"
PRESS(HS6)                          ; Método PRESS para el pulsador de menú horizontal
                                    6
  LM("Máscara1")                    ; Cargar la función Máscara1 (la máscara 1 debe
                                    estar definida en el mismo fichero).
END_PRESS                           ; Fin del método PRESS
HS7=("2.ª máscara")                 ; Rotular pulsador de menú horizontal 7 con "2.ª
                                    máscara"
PRESS(HS7)                          ; Método PRESS para el pulsador de menú horizontal
                                    7
  LM("Mascara2")                    ; Cargar la función Máscara2 (la máscara 2 debe
                                    estar definida en el mismo fichero).
END_PRESS                           ; Fin del método PRESS
//END                               ; Identificador de fin del bloque de entrada
```

Ejemplo

```
HS1 = ("nuevo menú de pulsadores")
HS2 = ("sin función")
PRESS (HS1)
  LS("Menú1")                         ; Cargar nuevo menú de pulsadores
END_PRESS
PRESS (HS2)                           ; Método PRESS vacío
END_PRESS
```

Diferentes configuraciones de los pulsadores de menú de inicio

Las diferentes configuraciones de los pulsadores de menú de inicio se agrupan. Primero se lee del "easyscreen.ini" el nombre del fichero que debe interpretarse. En los directorios siguientes se buscan ficheros *.com:

- [Directorio de usuario del sistema]/proj
- [Directorio OEM del sistema]/proj
- [Directorio de addon del sistema]/proj
- [Directorio Siemens del sistema]/proj

Las configuraciones contenidas para los pulsadores de menú de inicio se agrupan entonces en una sola configuración, es decir, que se comparan los distintos pulsadores de menú. Si hay dos o más configuraciones para un pulsador de menú, siempre se toma la mejor en la versión de Merge.

Los menús de pulsadores o diálogos que pueda haber se pasan por alto. Si un pulsador de menú contiene un comando sin indicar el fichero, p. ej., LM("prueba") porque el menú de pulsadores o el cuadro de diálogo deseado se encuentra en el mismo fichero, el nombre de fichero correspondiente se completa en la versión de Merge interna, de modo que no hace falta adaptarlo. Por último se muestra la configuración de Merge recibida.

5.4 Tratamiento de errores (diario de incidencias)

Vista general

Si "Run MyScreens" detecta errores al interpretar los ficheros de configuración, estos se guardan en el fichero de texto "easyscreen_log.txt". Solo se introducen errores del cuadro de diálogo seleccionado en ese momento. Las entradas de errores de cuadros de diálogo seleccionados anteriormente se borran.

El fichero contiene la siguiente información:

- Con qué acción se ha producido el error
- Los números de línea y columna del primer carácter erróneo
- La línea errónea completa del fichero de configuración
- Las entradas efectuadas con la función DEBUG

Nota

A cada entrada se antepone una etiqueta de fecha/hora actual (entre corchetes). Esto puede ser útil, por ejemplo, para analizar configuraciones en las que el tiempo es un factor crítico.

Almacenamiento de "easyscreen-log.txt"

El fichero "easyscreen_log.txt" se guarda en el siguiente directorio:

[Directorio de usuario del sistema]/log

Sintaxis

La interpretación de la sintaxis comienza solo una vez que se ha definido el pulsador de menú de inicio y se ha configurado un diálogo con identificador de arranque y de fin y con una línea de definición.

```
//S(Start)
HS6=("1.ª máscara")
PRESS(HS6)
  LM("Maske1")
END_PRESS
//END

//M(Maske1)
  DEF Var1=(R)
  DEF VAR2 = (R)
  LOAD
  VAR1 = VAR2 + 1              ; Aviso de error en el libro de incidencias, dado que
                               VAR2 no tiene ningún valor
...
//END

; lo correcto sería, p. ej.:
//M(Maske1)
   DEF Var1=(R)
   DEF VAR2 = (R)

   LOAD
      VAR2 = 7
      VAR1 = VAR2 + 1 ;
...
```

5.5 Notas sobre "easyscreen.ini"

A partir de SINUMERIK Operate V4.7, "easycreen.ini" se amplía con las entradas descritas en este capítulo. El fichero "easyscreen.ini" se encuentra en el directorio [Directorio Siemens del sistema]/cfg.

Posición inicial de la pantalla de ayuda

Entrada en "easyscreen.ini":

```
[GENERAL]
HlpPicFixPos=true
```

Nota:

La posición inicial de las pantallas de ayuda es, independientemente de la resolución, la posición de píxel configurada (estándar=true).

Comportamiento de extensión, altura y distancia estándar de línea

Entrada de easyscreen.ini:

```
[GENERAL]
SymmetricalAspectRatio=false
DefaultLineHeight=18
DefaultLineSpacing=3
```

Notas:

- La entrada "SymmetricalAspectRatio" determina si deben utilizarse los mismos factores de extensión al adaptar una configuración en dirección X e Y a la resolución de pantalla determinada.
  - "false" (estándar): los campos y gráficos se contraen en dirección Y en resoluciones panorámicas (extensiones asimétricas referidas a 640x480). Por ejemplo, un cuadrado configurado en 640x480 se comprime en vertical en una pantalla panorámica y se convierte así en un rectángulo.
  - "true": se extiende en dirección X e Y con el mismo factor de extensión; de esta forma los campos y gráficos mantienen las proporciones configuradas originalmente referidas a 640x480. Por ejemplo, de esta manera un cuadrado configurado en 640x480 se sigue visualizando como un cuadrado en un panel panorámico.
- Con las entradas "DefaultLineHeight" y "DefaultLineSpacing" puede establecerse la altura de línea predeterminada (estándar: 18 píxeles) y la distancia entre líneas predeterminada (estándar: 3 píxeles) referidas a 640x480. Estas solo tendrán efecto si no se ha especificado la posición Y o altura al configurar la posición del texto breve o del campo de entrada/salida.

Ciclos en la pantalla básica de la máquina

- Ingreso en el modo de operación JOG de la máquina a través de HS6 con la posibilidad de generar una llamada de ciclos en la pantalla básica de la máquina con la selección [JOBSHOPINTEGRATION]:
  ```
  [JOBSHOPINTEGRATION]
  Integration = true
  ```
  o vía el bloque de inicio en la configuración
  ```
  LM("Máscara1",,1)
  PRESS(VS8)
     GC("MOVE_RIDE") ; se genera una llamada de ciclos
     EXIT
  END_PRESS
  ```
- Mantenimiento de los menús de Operate con sección [Integration]:
  Si el cuadro de diálogo tiene solo pulsadores de menú verticales, cuando aparece se mantiene el menú de pulsadores horizontales estándar incluido el pulsador de menú de inicio. Si el cuadro de diálogo tiene pulsadores de menú horizontales y verticales, su barra horizontal reemplaza a la barra de pulsadores horizontales estándar con el pulsador de menú de inicio.
  ```
  [Integration]
  OperateMenusEnabled = true
  ```

Posición de la máscara en función de la resolución: Form Panels

Entrada en "easyscreen.ini":

```
[640x480]
MyPanel = x:=0, y:=220, width:=340, height:=174
[800x480]
MyPanel = x:=0, y:=220, width:=420, height:=174
...
```

Notas:

La posición de la máscara puede definirse de la forma siguiente:

- La posición de la máscara puede especificarse en "Píxeles referidos a 640x480".
  Ejemplo:
  `//M(MyMask/"MyCaption"/"\\myhelp.png"/`**`0,219,335,174`**`)`
- La posición de la máscara puede vincularse a la posición en función de la resolución de un FormPanel procedente de un diseño de pantalla estándar de Operate, p. ej. "FormPanel4" ("Funciones auxiliares") del diseño de pantalla "slmastandardscreenlayout.SlMaStandardScreenLayout" del área "Máquina"
  `//M(MyMask/"MyCaption"/"\`
  `\myhelp.png"/`**`"slmastandardscreenlayout.SlMaStandardScreenLayout.FormPanel4"`**
  `)`
  .
  Con ello es posible superponer formas estándar de Operate o colocar máscaras de Easyscreen en su posición exacta con comodidad.
  Ejemplo:
  `//M(Mask/"Mask"//`**`"slstandardscreenlayout.SlStandardScreenLayout.LowerForm"`**`)`
  Aunque también puede utilizarse cualquier otro diseño de pantalla.
- La posición de la máscara puede acoplarse a una definición propia de FormPanels dependiente de la resolución en el fichero "easyscreen.ini".
  Ejemplo:
  `//M(MyMask/"MyCaption"/"\\myhelp.png"/`**`"MyPanel"`**`)`

5.6 Indicaciones para nuevos usuarios sobre "Run MyScreens"

Nota

Al utilizar HMI Operate en la NCU, tenga en cuenta que todos los nombres de fichero se guardan en minúsculas en la tarjeta CF (com, png, txt).

Nota

Al guardar los ficheros de configuración e idioma, tenga en cuenta que en el editor que utiliza la codificación está ajustada a UTF-8.

Ficheros de imagen

Guarde los ficheros de imagen siempre en formato PNG "*.png".

Los datos deben guardarse, por ejemplo, en

[Directorio OEM del sistema]/ico/[Resolución]

para adaptaciones OEM.

Encontrará más información en el capítulo Utilizar imágenes/gráficos (Página 65).

Adaptación del fichero de configuración

Compruebe sus ficheros de configuración conforme a los siguientes puntos:

- Comparar los pulsadores de menú de inicio con los pulsadores de menú admitidos actualmente y efectuar ajustes en caso necesario.
- Modificar el nombre de los ficheros de imagen integrados conforme al punto de arriba "Ficheros de imagen".

Los datos, p. ej., para las adaptaciones OEM, se guardan en el siguiente directorio:

[Directorio OEM del sistema]/proj

[Directorio de usuario del sistema]/proj A

[Directorio de addon del sistema]/proj

Adaptación de los ficheros de ayuda

Todos los ficheros de ayuda deben guardarse en formato UTF-8. Compruebe los ficheros existentes y vuelva a guardarlos según corresponda con un editor adecuado.

Los ficheros HTML se guardan en el siguiente directorio, p. ej., para alemán:

[Directorio OEM del sistema]/hlp/deu

[Directorio de usuario del sistema]/hlp/deu

[Directorio de addon del sistema]/hlp/deu

Los directorios para otros idiomas deben crearse utilizando loscódigos de idioma correspondientes.

Comprobación de la licencia "Run MyScreens"

Compruebe si el número de diálogos insertados supera la capacidad máxima de 5 diálogos de la ejecución básica.

Para aumentar el número de cuadros de diálogo se requiere una de las siguientes opciones de software:

- SINUMERIK 828D/840D sl, SINUMERIK Integrate Run MyScreens (6FC5800-0AP64-0YB0)
- SINUMERIK 840D sl, SINUMERIK Integrate Run MyScreens + Run MyHMI (6FC5800-0AP65-0YB0)
- SINUMERIK 840D sl, SINUMERIK Integrate Run MyHMI/3GL (6FC5800-0AP60-0YB0)
- SINUMERIK 840D sl, SINUMERIK Integrate Run MyHMI/WinCC (6FC5800-0AP61-0YB0)

5.7 Sintaxis de configuración ampliada

A partir de SINUMERIK Operate V4.7 está disponible una sintaxis simplificada para definir máscaras, variables, pulsadores de menú y columnas de tablas. Con esta sintaxis alternativa mejora la legibilidad y se facilita el mantenimiento. Las propiedades y atributos pueden especificarse en el orden que se desee y las entradas vacías suprimirse. A diferencia de la sintaxis anterior, la lista de propiedades y atributos se escribe entre llaves "**{**" y "**}**" en lugar de entre paréntesis "**(**" y "**)**", como hasta ahora.

Las propiedades y atributos se especifican de la siguiente manera:
`{<Name> = <Wert>, <Name> = <Wert>, …}`

La sintaxis anterior sigue siendo compatible.

Sintaxis ampliada para la definición de máscaras

//M {<Nombre de máscara> [,**HD**=<Título>] [,**HLP**=<Gráfico>] [,**X**=<Posición X>] [,**Y**=<Posición Y>] [,**W**=<Anchura>] [,**H**=<Altura>] [,**VAR**=<>Variable de sistema o usuario] [,**HLP_X**=<Pantalla de ayuda para posición X>] [,**HLP_Y**=<Pantalla de ayuda para posición Y>] [,**CM**=<Ajuste de las columnas>] [,**CB**=Comportamiento al abrir el cuadro de diálogo] [,**XG**=<Interpretar pantalla de ayuda como gráfico X3d>] [,**PANEL**=<Nombre del FormPanel vinculado>][,**MC**=<Color de fondo de la máscara>][,**HD_AL**=<Ajuste del encabezado de la máscara>], [,**LANGFILELIST**=<Lista de ficheros de idioma específicos de la máscara>]}

Ejemplo:
```
//M{VariantTest, HD="My Mask"}
```

Sintaxis ampliada para la definición de variables

DEF <Nombre de variable> = {[**TYP**=<Tipo>] [,**MIN**=<Valor mínimo>] [,**MAX**=<Valor máximo>] [,**TGL**=<Valores de alternancia>] [,**VAL**=<Valor por defecto>] [,**LT**=<Texto largo>] [,**ST**=<Texto breve>] [,**GT**=<Texto de gráficos>] [,**UT**=<Texto de unidad>] [,**TT**=<Texto del tooltip>] [,**TG**=<Opción de alternancia>] [,**WR**=<Modo de entrada>] [,**AC**=<Nivel de acceso>] [,**AL**=<Ajuste de texto>] [,**FS**=<Tamaño de fuente>] [,**LI**=<Tratamiento de valores límite>] [,**UR**=<Frecuencia de actualización>] [,**CB**=<Comportamiento al abrir el cuadro de diálogo>] [,**HLP**=<Pantalla de ayuda>] [,**VAR**=<Variable de sistema o de usuario>] >] [,**TXT_X**=<Posición X de texto breve>] [,**TXT_Y**=<Posición Y de texto breve>] [,**TXT_W**=<Anchura de texto breve>] [,**TXT_H**=<Altura de texto breve>] [,**X**=<Posición X de campo de entrada/salida>] [,**Y**=<Posición Y de campo de entrada/salida>] [,**W**=<Anchura de campo de entrada/salida>] [,**H**=<Altura de campo de entrada/salida>] [,**UT_DX**=<Distancia entre el campo de entrada/salida y el campo de unidades>] [,**UT_W**=<Ancho del campo de unidades>] [,**BC**=<Color de fondo de campo de entrada/salida>] [,**FC**=<Color de primer plano de campo de entrada/salida>] [,**BC_ST**=<Color de fondo de texto breve>] [,**FC_ST**=<Color de primer plano de texto breve>] [,**BC_GT**=<Color de fondo de texto de gráficos>] [,**FC_GT**=<Color de primer plano de texto de gráficos>] [,**BC_UT**=<Color de fondo de texto de unidad>] [,**FC_UT**=<Color de primer plano de texto de unidad>] [,**SC1**=<Color de señal 1 para barra de progreso>] [,**SC2**=<Color de señal 2 para barra de progreso>] [,**SVAL1**=<Valor umbral 1 para barra de progreso>] [,**SVAL2**=<Valor umbral 2 para barra de progreso>] [,**DT**=<Tipo de visualización>] [,**DO**=<Orientación de la pantalla>] [,**OHLP**=<Ayuda en línea>][,**LINK_TGL**=<Nombre de la variable de alternancia vinculada>]}

Ejemplos:

```
DEF MyVar5={TYP="R2", ST="MyVar5", VAL=123.4567, OHLP="myhelp.html", MIN=100.1,
MAX=200.9}
DEF MyVar2={TYP="I", TGL="*1,2,3", VAL=1}
DEF MyVar3={TYP="R2", TGL="*0=""Aus"", 1=$80000", VAL=1}
DEF MyVar4={TYP="R2", TGL="*MyArray",VAL=1}
DEF MyVar1={TYP="R2", TGL="%grid99", X = 0, W=300, H=200}
DEF MyVar6={TYP="R2", TGL="+$80000", VAR="$R[10]", ST="Textoffset"}
```

Sintaxis ampliada para la definición de pulsadores de menú

SK = {[**ST**=<Rótulo>] [,**AC**=<Nivel de acceso>] [,**SE**=<Estado>]}

Ejemplos:

```
HS1={ST="""MySk""", AC=6, SE=1}
HS3={ST="SOFTKEY_CANCEL"}
HS5={ST="[$81251,""\\sk_ok.png""]"}
HS8={ST="[""Test"",""\\sk_ok.png""]"}
```

Sintaxis ampliada para la definición de columnas de tablas

{[**TYP**=<Tipo>] [,**MIN**=<Valor mínimo>] [,**MAX**=<Valor máximo>] [,**LT**=<Texto largo>] [,**ST**=<Texto breve>] [,**WR**=<Modo de entrada>] [,**AC**=<Nivel de acceso>] [,**AL**=<Ajuste de texto>] [,**FS**=<Tamaño de fuente>] [,**LI**=<Tratamiento de valores límite>] [,**UR**=<Frecuencia de actualización>] [,**HLP**=<Pantalla de ayuda>] [,**VAR**=<Variable de sistema o de usuario>] >] [,**W**=<Ancho de columnas>] [,**OF1**=<Offset1>] [,**OF2**=<Offset2>] [,**OF3**=<Offset3>]}

Ejemplo:

```
DEF MyGridVar={TYP="R", TGL="%MyGrid1", X=10, W=550, H=100}
//G(MyGrid1/0/5)
   {TYP="I", ST="Index", WR=1, VAR="1", W=80, OF1=1}
   {TYP="S", LT="LongText2", ST="Text", WR=1, VAR="$80000", AL=2, W=330, OF1=1}
   {TYP="R3", LT="LongText1", ST="R9,R11,R13,R15", WR=2, VAR="$R[1]", W=110, OF1=2}
//END
```

5.8 SmartOperation y manejo multitáctil

Para la adaptación específica a SmartOperation y para el manejo multitáctil existen las siguientes posibilidades de ajuste:

- Adaptación automática de la altura y anchura de los campos al denominado tamaño mínimo de manejo de los campos y a la distancia entre líneas (atributo de máscara MA).
- Expansión optimizada con precisión de píxeles de los campos en caso de resoluciones más altas (atributo de máscara PA).
- Ajuste de la altura de campo y de la distancia entre líneas en proporción a la fuente (atributo de máscara FA).
- Permite el desplazamiento libre de los campos para que el teclado virtual no cubra el campo de entrada (atributo de máscara KM).

Encontrará más información al respecto en el capítulo Definir propiedades del diálogo (Página 53), apartado "Programación".

Cuadros de diálogo

6

6.1 Estructura y elementos de un diálogo

6.1.1 Definir diálogos

Definición

Un diálogo es una parte de una interfaz de usuario que consta de línea de título, elementos de diálogo y/o gráfico, línea de salida para mensajes, así como de 8 pulsadores de menú horizontales y 8 verticales.

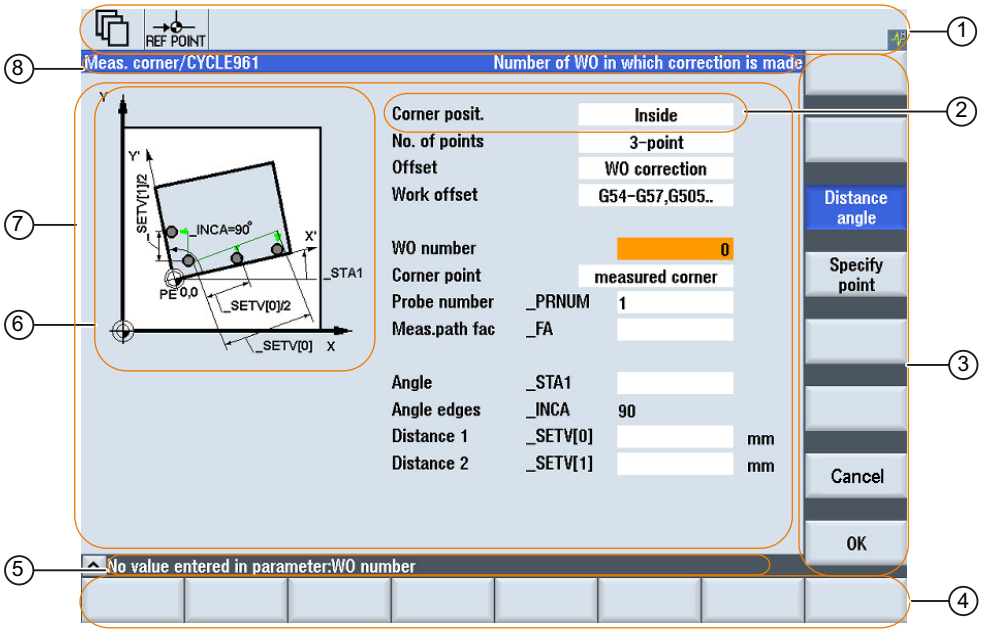
Los elementos del diálogo son:

- Variables
  - Valores límite/campo de alternancia
  - Valor por defecto de las variables
- Pantalla de ayuda
- Textos
- Atributos
- Variable de sistema o de usuario
- Posición texto breve
- Posición campo de entrada/salida
- Colores

Propiedades de un diálogo:

- Título
- Gráficos
- Dimensión
- Variable de sistema o de usuario

- Posición gráfico
- Atributos



① Indicación del estado de la máquina ("cabecera")
② Elemento de diálogo
③ 8 pulsadores de menú verticales
④ 8 pulsadores de menú horizontales
⑤ Emisión de avisos de diagnóstico
⑥ Gráficos
⑦ Cuadro de diálogo
⑧ Línea de título del diálogo con título y texto largo

Figura 6-1 Estructura del diálogo

Vista general

Por norma, la descripción de un diálogo (bloque descriptivo) está configurada como sigue:

Bloque descriptivo	Comentario	Referencia cruzada a capítulos
`//M...`	;Identificador de arranque del diálogo	
`DEF Var1=...` `...`	;Variables	Variables (Página 85)
`HS1=(...)` `...`	;Pulsadores de menú	Definir menús de pulsadores (Página 65)

`PRESS(HS1)`
`  LM...`
`END_PRESS`
	;Identificador de arranque del método
	;Acciones
	;Identificador de fin del método
		Métodos (Página 121)

`//END`	;Identificador de fin del diálogo	

Dentro del bloque descriptivo del diálogo se definen en primer lugar diferentes variables que son visibles en el diálogo como elementos de diálogo y los pulsadores de menú horizontales y verticales. A continuación se configuran distintas acciones en métodos.

Nota

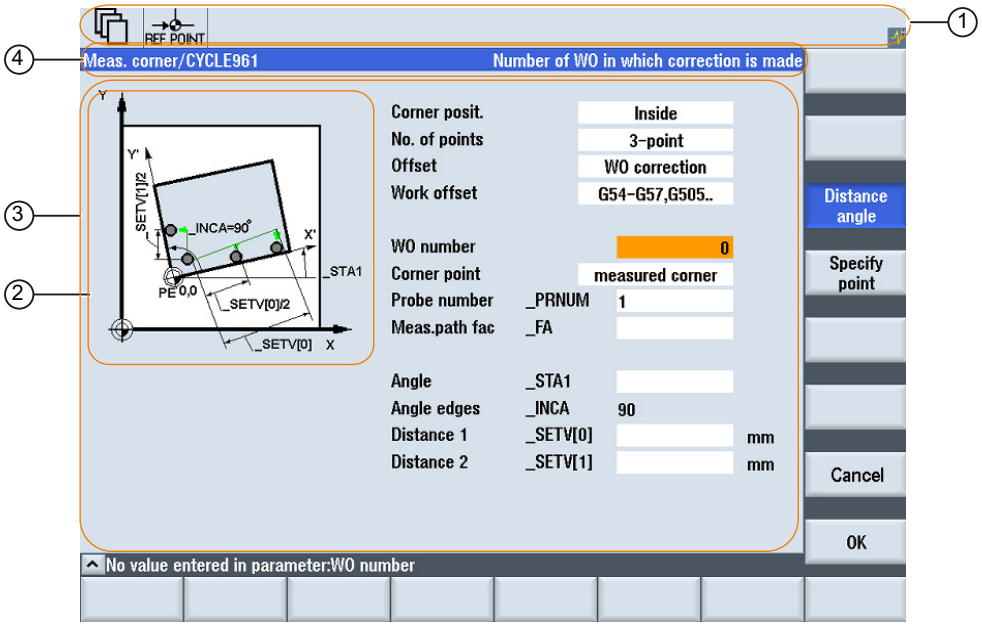
En el campo de manejo "Máquina" debe respetarse el siguiente orden en el cambio de diálogo:

Suponiendo que las dimensiones de la máscara siguiente sean menores que las de la anterior o de que la posición de la máscara siguiente sea distinta a la de la máscara anterior, el cambio de diálogo solo funciona cuando la máscara siguiente se finaliza antes de volver a la primera máscara y luego se carga de nuevo la primera máscara.

6.1.2 Definir propiedades del diálogo

Descripción

Con el identificador de arranque del diálogo se definen al mismo tiempo las propiedades del mismo:



① Indicación del estado de la máquina ("cabecera")
② Gráficos
③ Cuadro de diálogo
④ Línea de título del diálogo con título y texto largo

Figura 6-2 Propiedades de un diálogo

Programación

Sintaxis:	**//M**(Identificador/[Título]/[Gráficos]/[Dimensión]/[Variable de sistema o de usuario]/[Posición gráfico]/[Atributos]/Lista de ficheros de idioma específicos de máscara) Consulte también el capítulo Sintaxis de configuración ampliada (Página 47).
Descripción:	Definir diálogos

Parámetro:	Identificador			Nombre del diálogo
	Título			Título del diálogo como texto o llamada a un texto (p. ej. $85011) de un fichero de texto dependiente del idioma
	Gráficos			Fichero gráfico con ruta entre comillas dobles
	Dimensión			Posición y tamaño del diálogo en píxeles (distancia desde la izquierda, distancia desde arriba, anchura, altura), con relación a la esquina superior izquierda de la pantalla. Los datos se separan con comas.
	Variable de sistema o de usuario			Variable de sistema o de usuario a la cual se asigna la posición actual del cursor. La posición del cursor se puede transmitir al CN o al PLC a través de la variable de sistema o de usuario. La primera variable tiene el índice 1. El orden corresponde al orden en que se configuran las variables.
	Posición gráfico			Posición del gráfico (distancia desde la izquierda, distancia desde arriba) en píxeles, relativa a la esquina superior izquierda del diálogo. Los datos se separan con comas.
	Atributos			Los atributos se indican separados mediante comas. Los posibles atributos son:
		CM		Column Mode: ajuste de las columnas
			CM0	Ajuste predeterminado: la división de columnas se realiza por separado para cada línea.
			CM1	la división de columnas de la línea con más columnas es válida para todas las líneas.
		CB		CHANGE Block: comportamiento al abrir el diálogo: para la variable, los atributos cb indicados en una definición de variables tienen prioridad ante la indicación global en la definición del diálogo. Ver también AUTOHOTSPOT.
			CB0	Ajuste predeterminado: todos los CHANGE Blocks del diálogo se ejecutan al abrir.
			CB1	Los CHANGE Blocks sólo se ejecutan si se modifica el valor correspondiente.
		XG		Integración de una animación X3D como pantalla de ayuda (solo en la programación de cadenas secuenciales) Ejemplo: `//M(Meas/ $85605/"myx3dhelpfile.hmi,,Z_Animation,,G17"///30,10/ XG1)`
			XG0	Ajuste predeterminado = 0
			XG1	El atributo de la máscara XG debe estar ya establecido como 1 en la definición de la máscara; no es posible efectuar cambios en tiempo de ejecución. Nota: La indicación en la propiedad de máscaras HLP debe establecerse también de manera correspondiente.
		AL		Con el atributo de máscara AL puede influirse en el ajuste del título de máscara. Ejemplo: Colocación central del título de la ventana "Set password": `//M(MY_PWD_SET/"Set password"//"EasyPwdModalLayout"// AL2)`
			AL0	Justificado a la izquierda, ajuste predeterminado
			AL1	Justificado a la derecha
			AL2	Centrado

KM		El desplazamiento libre por la máscara (SlGfwScrollArea), cuando se desea mostrar el teclado virtual en modo multitáctil, puede modificarse del modo siguiente:
		• En la línea de definición de máscara con el atributo de máscara "KM" (KeyboardMode)
		Ejemplo:
		`//M(MyMTMask/"MultiTouch Mask"/////KM1)`
		• Como ajuste global para todas las máscaras Run MyScreens en "easyscreen.ini"
		Ejemplo:
		`[GENERAL]`
		`DefaultVirtualKeyboardMode=1`
		
		Nota:
		Si se ajusta el atributo de máscara KM explícitamente en la configuración de máscara, esto tiene prioridad por delante del ajuste global de "easyscreen.ini".
		
		(Ver también el capítulo SmartOperation y manejo multitáctil (Página 49))

	KM1	Desplazamiento libre activo (predeterminado)
	KM0	Desplazamiento libre inactivo
PG		Alineación y aspecto del título de la ventana en combinación con imágenes PNG (efectivo solo en el editor) Ejemplo: `//M(MyPg1SampleMask/"My PG1 Sample Mask"/////PG1)`
	PG0	(Predeterminado)
	PG1	Alineación y aspecto del título de la ventana en combinación con imágenes PNG Ruta de acceso (borde izquierdo), nombre de la máscara (derecha) En este modo ya no es posible visualizar el texto largo. También puede trabajar con tooltips.

MA		Adaptación automática de la altura de campo para manejo multitáctil (ajuste multitáctil)
		La adaptación para un panel de operador multitáctil incluye la posibilidad de ampliación en caso necesario hasta el denominado tamaño mínimo de manejo (ancho, alto) de los campos (excepciones: barra de progreso, GIF animado, widget personalizado) y la distancia entre líneas.
		La adaptación multitáctil puede
		• realizarse globalmente para todas las máscaras de Run MyScreens en el archivo "easyscreen.ini", p. ej.:
		`[GENERAL]`
		`DefaultMultiTouchAdjustmentLevel=1`
		• o de manera específica con el atributo de máscara "MA" en la línea de definición de máscara, con lo cual se sobrescribe el ajuste global en "easyscreen.ini", p. ej.:

		`//M(MyMTMask/"MultiTouch Mask"////MA1)` Si se ajusta el atributo de máscara MA explícitamente en la configuración de máscara, esto tiene prioridad por delante del ajuste global de "easyscreen.ini". (Ver también el capítulo SmartOperation y manejo multitáctil (Página 49))
	MA0	No se realiza la adaptación
	MA1	Solo se adaptan los campos para los que no se ha configurado la distancia desde arriba o la altura y anchura de campo, es decir, los campos que ocupan posiciones predeterminadas, y para los cuales Run MyScreens calcula la distancia desde arriba o la altura y anchura de campo. (Predeterminado)
	MA2	Se adaptan todos los campos, sean cuales sean los tamaños de campo configurados.

PA		Expansión optimizada con precisión de píxeles de los campos para resoluciones más altas (Pixel Fine Adjustment)
		Hasta ahora, todas las posiciones de los campos se calculaban en función de una resolución de 640 x 480 y a continuación se ampliaban aplicando en cada caso el correspondiente factor de expansión horizontal y vertical. Sin embargo, este comportamiento tiene un inconveniente: si existen "factores de expansión desfavorables", pueden producirse errores de redondeo. P. ej., la altura de campo o la distancia entre línea y línea puede aumentar un píxel. Para evitarlo, existe el modo "Pixel Fine Adjustment", en el que las posiciones de los campos se definen directamente en la resolución actual.
		`//M(MyMask/"My Mask"////PA1)`
		Este ajuste también puede realizarse globalmente para todas las máscaras de Run MyScreens en "easyscreen.ini", p. ej.,
		`[GENERAL]`
		`DefaultPixelFineAdjustment=1`
		Si se ajusta el atributo de máscara PA explícitamente en la configuración de máscara, esto tiene prioridad por delante del ajuste global de "easyscreen.ini".
		Si se visualiza una máscara con Font Adjustment = FA1, el modo Pixel Fine Adjustment está activo también automáticamente para esta máscara.
		(Ver también el capítulo SmartOperation y manejo multitáctil (Página 49))

	PA0	El comportamiento de expansión continúa como hasta ahora, es decir, las posiciones se calculan en función de una resolución de 640 x 480 y se implementan en consecuencia. (por compatibilidad: predeterminado)
	PA1	Expansión optimizada con precisión de píxeles de los campos en caso de resoluciones más altas

FA		Ajuste de la altura de campo y de la distancia entre líneas en proporción a la fuente (Font Adjustment)
		Esta función también puede realizarse globalmente para todas las máscaras de Run MyScreens en "easyscreen.ini", p. ej.,
		`[GENERAL]`
		`DefaultFontAdjustment=1`
		Si se ajusta el atributo de máscara FA explícitamente en la configuración de máscara, esto tiene prioridad por delante del ajuste global de "easyscreen.ini".
		Los ajustes de DefaultLineHeight y DefaultLineSpacing carecen de efectividad en el modo Font Adjustment.
		(Ver también el capítulo SmartOperation y manejo multitáctil (Página 49))

		FA0	La altura de campo y la distancia entre campos se amplían como hasta ahora (por compatibilidad: predeterminado)
		FA1	La altura de campo y la distancia entre campos se ajustan en proporción a la fuente (se ajusta automáticamente el atributo de máscara PA=1)
		FA2	Igual que FA1, pero además la coordenada X y el ancho de campo se amplían en proporción a la fuente. (se ajusta automáticamente el atributo de máscara PA=1) (Solo posible en combinación con el atributo XG=1)
	NT		Tipo de navegación (Navigation Type)
		NT0	NT0 = modo estándar La navegación se realiza de modo análogo a las máscaras del campo de manejo en el que se integra la máscara Run MyScreens. Es decir, en las máscaras de ciclo del editor, la navegación se realiza de modo lineal (ver NT2), y en todas las demás máscaras "normales", de modo geométrico (ver NT1).
		NT1	Navegación geométrica (arriba, abajo, izquierda, derecha) hasta alcanzar los bordes de la máscara (estándar en el entorno normal de Run MyScreens, p. ej., en el área "Custom")
		NT2	Lineal, es decir, dentro de la línea hacia la derecha hasta llegar al final de la línea (estándar en las máscaras de ciclo del editor)
	NR		Sentido de navegación al pulsar la tecla Intro (Return Navigate Direction)
		NR0	NR0 = desactivado (predeterminado), es decir, al pulsar la tecla Intro, la navegación se realiza en orden tabular dentro de la máscara (estándar en el entorno normal de Run MyScreens)
		NR1	Al pulsar la tecla Intro, la navegación se comporta como al pulsar la tecla de flecha hacia arriba
		NR2	Como NR1, pero hacia abajo
		NR3	Como NR1, pero hacia la izquierda
		NR4	Como NR1, pero hacia la derecha
	TA		Posición de los tooltips
		TA0	Automática (predeterminada)
		TA1	Izquierda
		TA2	Derecha
		TA3	Arriba
		TA4	Abajo
		TA5	Arriba a la izquierda
		TA6	Abajo a la izquierda
		TA7	Arriba a la derecha
		TA8	Abajo a la derecha

MC			Color de fondo de la máscara (Mask Color)
			Ejemplo:
			Establecer color de fondo azul (=6) para la máscara
			`//M(MyMask/"MyMask"//////6)`
			o bien
			`PRESS(HS1)`
			`  MC=6`
			`END_PRESS`

	Lista de ficheros de idioma específicos de máscara	Separados por comas

Acceso a las propiedades del diálogo

Dentro de los métodos (p. ej. PRESS Block) puede tener lugar un acceso de lectura y de escritura a las siguientes propiedades del diálogo:

- HD = Título (header)
- HLP = Pantalla de ayuda (help)
- VAR = Variable de sistema o de usuario
- MC = Color de fondo de la máscara
- CM = Ajuste de las columnas (solo lectura)
- CB = Comportamiento al abrir (solo lectura)
- XG = Integración X3d (solo lectura)
- AL = Ajuste del título de máscara (solo lectura)

Ejemplo

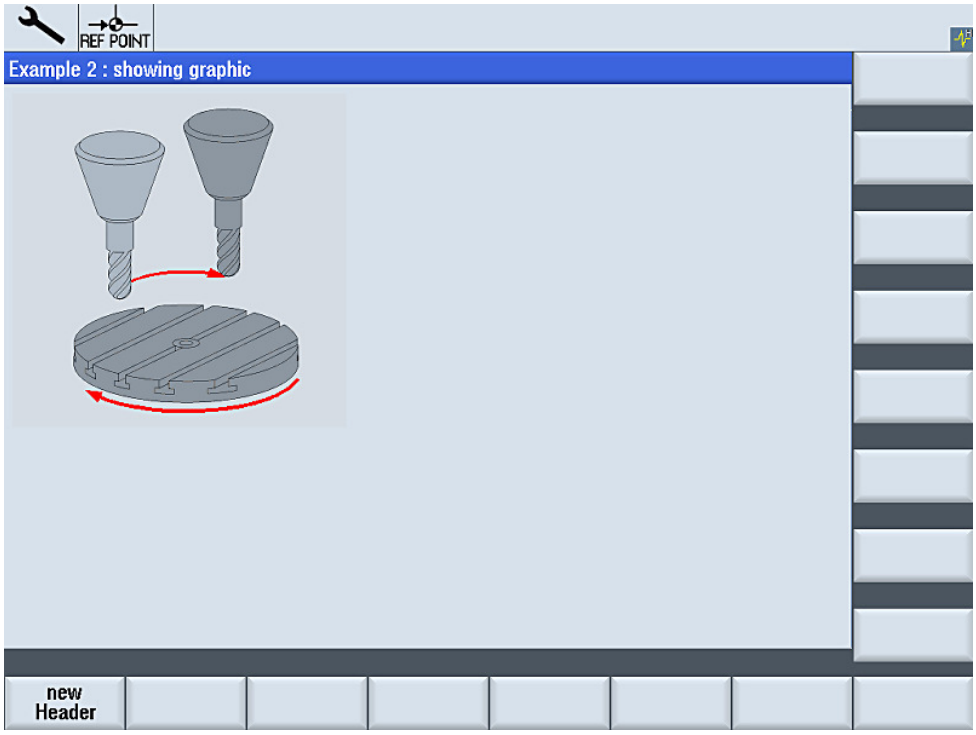


Figura 6-3 "Example 2: showing graphic"

```
//S(Start)
HS7=("Example", sel, ac7)
```

```
PRESS(HS7)
  LM("Mask2")
END_PRESS

//END
//M(Mask2/"Example 2 : showing graphic"/"example.png")
HS1=("new%nHeader")
HS2=("")
HS3=("")
HS4=("")
HS5=("")
HS6=("")
HS7=("")
HS8=("")
VS1=("")
VS2=("")
VS3=("")
VS4=("")
VS5=("")
VS6=("")
VS7=("")
VS8=("")

PRESS(HS1)
  Hd= "new Header"
END_PRESS
...
//END
```

Consulte también

Ejemplo de programación para el campo "Custom" (Página 280)

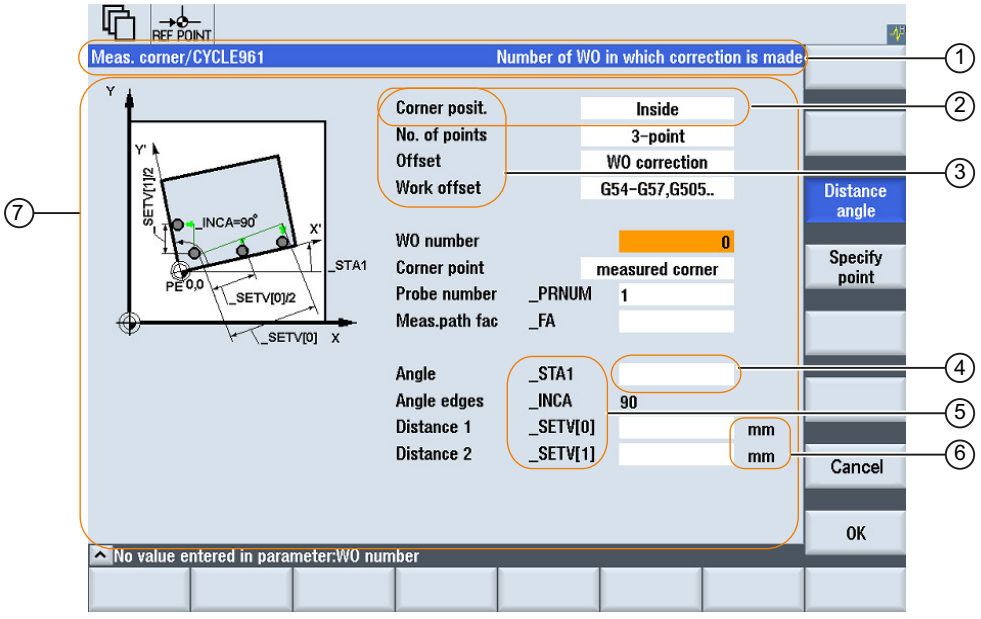
6.1.3 Definir elementos de diálogo

Elemento de diálogo

Se denomina elemento de diálogo la parte visible de una variable, esto es, texto breve, texto de gráficos, campo de entrada/salida, texto de unidad y sugerencia de herramienta. Los elementos de diálogo ocupan filas en la parte principal del diálogo. Por cada fila se pueden definir uno o varios elementos de diálogo.

Propiedades de variables

Todas las variables son válidas únicamente en el diálogo activo. Al definir una variable se le asignan propiedades. Dentro de los métodos (p. ej. de un método PRESS) se puede acceder a los valores de las propiedades del diálogo.



① Línea de título del diálogo con título y texto largo
② Elemento de diálogo
③ Texto breve
④ Campo de entrada/salida
⑤ Texto de gráficos
⑥ Texto de unidad
⑦ Parte principal del diálogo

Figura 6-4 Elementos de un diálogo

Programación - Vista general

Entre paréntesis se encuentran los parámetros individuales, que deben separarse mediante comas:

DEF [Identificador] =	**Identificador = Nombre de la variable**
	Tipo de las variables
	/[Valores límite o campo de alternancia]
	/[Valor por defecto]
	/[Textos (texto largo, texto breve/imagen, texto de gráficos, texto de unidad)]
	/[Atributos]
	/[Pantalla de ayuda]
	/[Variable de sistema o de usuario]
	/[Posición texto breve]

	/[Posición campo de entrada/salida (Left, Top, Width, Height)]
	/[Colores]
	/[Ayuda en línea] (Página 74)

Ver también

Parámetros de variables (Página 93)

6.1.4 Definir diálogos con varias columnas

Vista general

En un diálogo pueden representarse también diversas variables en una fila. En este caso todas las variables se definen en el fichero de configuración dentro de una línea de definición.

```
DEF VAR11 = (S///"Var11"), VAR12 = (I///"Var12")
```

Para representar las distintas variables en el fichero de configuración de forma más legible, las líneas de definición pueden interrumpirse después de cada definición de variable y la coma siguiente.

La palabra clave "DEF" siempre indica el comienzo de una nueva línea:

```
DEF Tnr1=(I//1/"","T ",""/wr1///,,10/20,,50),
   TOP1=(I///,"Typ="/WR2//"$TC_DP1[1,1]"/80,,30/120,,50),
   TOP2=(R3///,"L1="/WR2//"$TC_DP3[1,1]"/170,,30/210,,70),
   TOP3=(R3///,"L2="/WR2//"$TC_DP4[1,1]"/280,,30/320,,70),
   TOP4=(R3///,"L3="/WR2//"$TC_DP5[1,1]"/390,,30/420,,70)
DEF Tnr2=(I//2/"","T ",""/wr1///,,10/20,,50),
   TOP21=(I///,"Typ="/WR2//"$TC_DP1[2,1]"/80,,30/120,,50),
   TOP22=(R3///,"L1="/WR2//"$TC_DP3[2,1]"/170,,30/210,,70),
   TOP23=(R3///,"L2="/WR2//"$TC_DP4[2,1]"/280,,30/320,,70),
   TOP24=(R3///,"L3="/WR2//"$TC_DP5[2,1]"/390,,30/420,,70)
```

Nota

Si desea crear diálogos que tengan varias columnas, tenga en cuenta que muchas columnas pueden relentizar el sistema.

6.1.5 Cuadros de diálogo de contraseña

Uso de cuadros de diálogo de contraseña estándar

Los siguientes cuadros de diálogo de contraseña predefinidos pueden incorporarse en la configuración de máscara:

- Definir contraseña

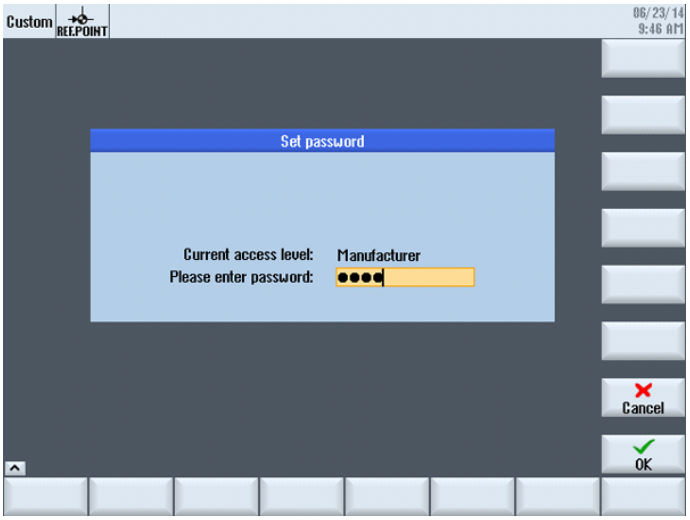


Figura 6-5 Cuadro de diálogo de definición de contraseña

- Cambiar contraseña

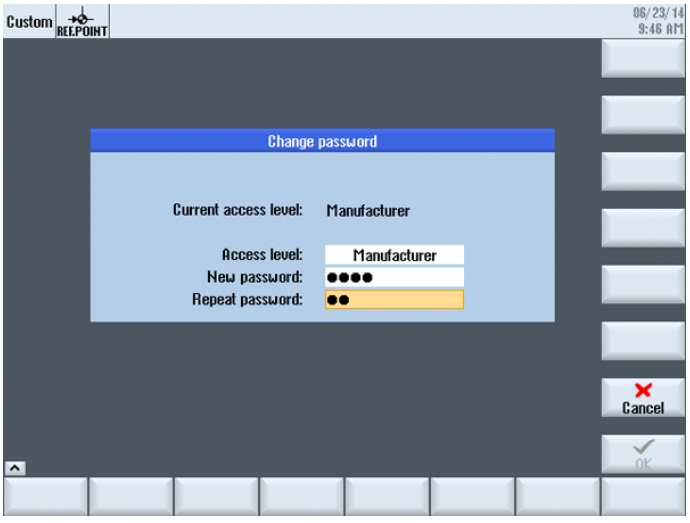


Figura 6-6 Cuadro de diálogo de cambio de contraseña

- Borrar contraseña

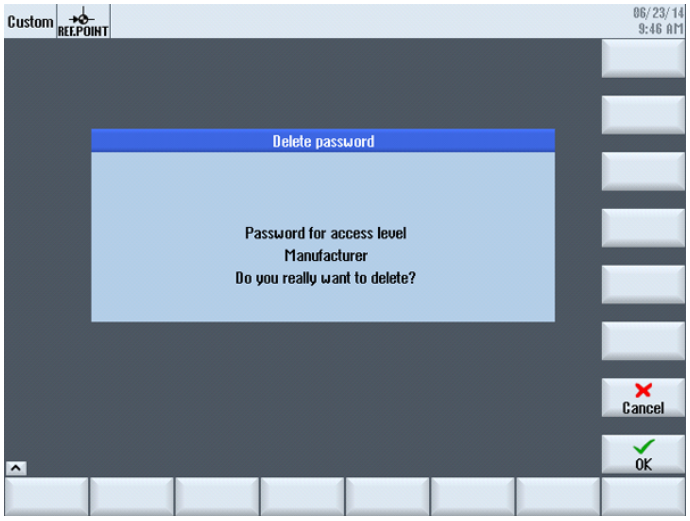


Figura 6-7 Cuadro de diálogo de borrado de contraseña

Nota

Con estos cuadros de diálogo se proporciona la funcionalidad de contraseña. Estos cuadros de diálogo no coinciden con los de SINUMERIK Operate.

El cuadro de diálogo puede abrirse a través de la función Load Softkey (LS) o la función Load Mask (LM):

1. Llamada mediante la función Load Softkey (LS):
   `LS("Passwd","slespasswd.com",1)`
   Ampliación de los pulsadores de menú verticales:
   - Pulsador de menú 1: Definir contraseña
   - Pulsador de menú 2: Cambiar contraseña
   - Pulsador de menú 3: Borrar contraseña
2. También puede saltarse directamente a las tres máscaras directamente llamando la función Load Mask (LM):
   - Definir contraseña: `LM("PWD_SET", "slespasswd.com",1)`
   - Cambiar contraseña: `LM("PWD_CHG", "slespasswd.com",1)`
   - Borrar contraseña: `LM("PWD_CLEAR", "slespasswd.com",1)`

6.1.6 Utilizar imágenes/gráficos

Uso de gráficos

Se tienen que distinguir:

- Imágenes/gráficos en el área gráfica
- Pantallas de ayuda que, por ejemplo, ilustran variables individuales y que se suceden en el área gráfica.
- En lugar de un texto breve o de un campo E/S pueden configurarse otras pantallas de ayuda que pueden posicionarse arbitrariamente.

Lugares de almacenamiento

La imagen adecuada para la resolución del monitor conectado se busca en los directorios de resolución correspondientes en el siguiente orden:

[Directorio de usuario del sistema]`/ico/ico<Resolución>`
[Directorio OEM del sistema]`/ico/ico<Resolución>`
[Directorio de addon del sistema]`/ico/ico<Resolución>`

Si la imagen no se muestra o no se encuentra, cópiela en uno de los siguientes directorios para la resolución de 640 x 480 píxeles:

[Directorio de usuario del sistema]`/ico/ico640`
[Directorio OEM del sistema]`/ico/ico640`
[Directorio de addon del sistema]`/ico/ico640`

Nota

En caso de paneles con diferente resolución, las imágenes se posicionan de forma proporcional.

6.2 Definir menús de pulsadores

Definición

Todos los pulsadores de menú horizontales y verticales se denominan conjuntamente menús de pulsadores. Adicionalmente a los menús de pulsadores existentes, pueden definirse otros menús que sobrescriban total o parcialmente a los existentes.

Los nombres de los pulsadores de menú están establecidos. No todos los pulsadores de menú deben estar asignados.

HSx x 1 - 8, Pulsadores de menú horizontales de 1 a 8

VSy y 1 - 8, Pulsadores de menú verticales de 1 a 8

Para empezar, la descripción de un menú de pulsadores (bloque descriptivo) está configurada como sigue:

Bloque descriptivo	Comentario	Referencia cruzada a capítulos
`//S...`	;Identificador de arranque del menú de pulsadores	
`HSx=...`	;Definición de pulsadores de menú	

`PRESS(HSx)`
`  LM...`
`END_PRESS`
	;Identificador de arranque del método
	;Acciones
	;Identificador de fin del método
		Ver capítulo Métodos (Página 121)

`//END`	;Identificador de fin del menú de pulsadores	

Descripción

Al definir el menú de pulsadores, al mismo tiempo se le asignan propiedades.

Programación

Sintaxis:	*//S(Identificador)* ... //END	;Identificador de arranque del menú de pulsadores ;Identificador de fin del menú de pulsadores
	Consulte también el capítulo Sintaxis de configuración ampliada (Página 47).	
Descripción:	Definir menú de pulsadores	
Parámetro:	Identificador	Nombre del menú de pulsadores
	Texto o nombre del fichero de imagen	
Sintaxis:	**SK** = (Texto[, Nivel de acceso][, Estado][, Ajuste de la imagen del pulsador de menú][, Ajuste de texto con relación a la imagen del pulsador de menú])	
Descripción:	Definición de pulsadores de menú	
Parámetro:	SK	Pulsador de menú, p. ej. de HS1 a HS8, de VS1 a VS8

	Texto	Indicar texto	
	Nombre del fichero de imagen	"\\my_pic.png" o a través de un fichero de texto separado $85199, p. ej., con el siguiente texto en el fichero de texto (dependiente del idioma): 85100 0 0 "\\my_pic.png". El tamaño de la imagen que se visualizará sobre un pulsador de menú depende del OP utilizado:	
		OP 08:	640 x 480 mm → 25 x 25 píxeles
		OP 010:	640 x 480 mm → 25 x 25 píxeles
		OP 012:	800 x 600 mm → 30 x 30 píxeles
		OP 015:	1024 x 768 mm → 40 x 40 píxeles
		OP 019:	1280 x 1024 mm → 72 x 72 píxeles
	Nivel de acceso	ac0 a ac7 (ac7: ajuste predeterminado)	
	Estado	se1: visible (ajuste predeterminado) se2: Deshabilitado (fuente en color gris) se3: Resaltado (último pulsador de menú utilizado)	
	Ajuste de la imagen del pulsador de menú	PA (PictureAlignment)	
		Valores válidos: 0: izquierda 1: derecha 2: centrado 3: arriba (estándar) 4: abajo	
	Ajuste de texto con relación a la imagen del pulsador de menú	TP (TextAlignedToPicture)	
		Valores válidos: 0: el texto no se alinea en la imagen 1: el texto se alinea en la imagen (estándar)	

Nota

Un cambio de línea en el rótulo del pulsador de menú se genera con %n.

Se dispone como máximo de 2 líneas de 9 caracteres cada una.

Asignación de los niveles de acceso

El operador solo tiene acceso a la información que corresponde a este nivel de acceso y a los niveles inferiores (ver también AUTOHOTSPOT).

Ejemplo

```
//S(Menú1)                                    ; Identificador de arranque del menú de pulsadores
```

```
HS1=("NUEVO", ac6, se2)                          ; Definir el pulsador de menú HS1 y asignar
                                                 el rótulo "NUEVO", el nivel de protección 6 y el estado
                                                 "deshabilitado"
HS2=("\\imagen1.png")                            ; Asignar un gráfico al pulsador de menú
HS3=("Salir")
HS4=(["Confirmar","\\sk_ok.png"],PA0,TP1)        ; pulsador de menú con texto y gráficos Grafik, Tex-
                                                 to="Confirm", Imagen="sk_ok.png", Ajuste de la imagen
                                                 del pulsador de menú: izquierda, el texto se alinea en
                                                 la imagen

VS1=("Submáscara")
VS2=($85011, ac7, se2)                           ; Definir el pulsador de menú VS2, asignar el texto del
                                                 fichero de idioma, el nivel de protección 1 y el estado
                                                 "deshabilitado"
VS3=("Cancelar", ac1, se3)                       ; Definir el pulsador de menú VS3 y asignar
                                                 el rótulo "Cancelar", el nivel de protección
                                                  1 y el estado "resaltado".
VS4=("OK", ac6, se1)                             ; Definir el pulsador de menú VS4 y asignar
                                                 el rótulo "OK", el nivel de protección
                                                  6 y el estado "visible"
VS5=(SOFTKEY_CANCEL,,se1)                        ; Definir el pulsador de menú estándar VS5 ("Cance-
                                                 lar") y asignar el estado "visible"
VS6=(SOFTKEY_OK,,se1)                            ; Definir el pulsador de menú estándar VS6 ("OK") y
                                                 asignar el estado "visible"
VS7=(["\\imagen1.png","Texto OEM"],,se1)         ; Definir el pulsador de menú VS7 y asignar un gráfico,
                                                 el rótulo "Texto OEM" y el estado "visible"
VS8=(["\\imagen1.png",$83533],,se1)              ; Definir el pulsador de menú VS8 y asignar un gráfico,
                                                 el texto del fichero de idioma y el estado "visible"

PRESS(HS1)                                       ; Identificador de arranque del método
  HS1.st="Calcular"                              ; Asignar un texto de rotulación al pulsador de menú
...
END_PRESS                                        ; Identificador de fin del método

PRESS(RECALL)                                    ; Identificador de arranque del método
  LM("Máscara21")                                ; Cargar diálogo
END_PRESS                                        ; Identificador de fin del método
//END                                            ; Identificador de fin del menú de pulsadores
```

6.2.1 Modificar las propiedades de pulsadores de menú en el tiempo de ejecución

Descripción

Las propiedades texto, nivel de acceso y estado de un pulsador de menú pueden modificarse en los métodos en tiempo de ejecución:

Programación

Sintaxis:	SK.st = "Texto" SK.ac = nivel de acceso SK.se = estado SK.pa = "Ajuste de la imagen del pulsador de menú" SK.tp = "Ajuste de texto referido a la imagen del pulsador de menú"		;Pulsador de menú con rotulación ;Pulsador de menú con nivel de protección ;Pulsador de menú con estado ;pulsador de menú con imagen ;pulsador de menú con imagen y rotulación
Descripción:	Asignar propiedades		
Parámetro:	Texto		Texto de rotulación entre comillas
	Nivel de acceso		Margen de valores: 0 ... 7
	Estado	1: 2: 3:	visible y habilitado Deshabilitado (fuente en color gris) Resaltado (último pulsador de menú utilizado)
	Ajuste de la imagen del pulsador de menú	0: 1: 2: 3: 4:	izquierda derecha centrado arriba (estándar) abajo
	Ajuste de texto referido a la imagen del pulsador de menú	0: 1:	el texto no se alinea en la imagen el texto se alinea en la imagen (estándar)

Ejemplo

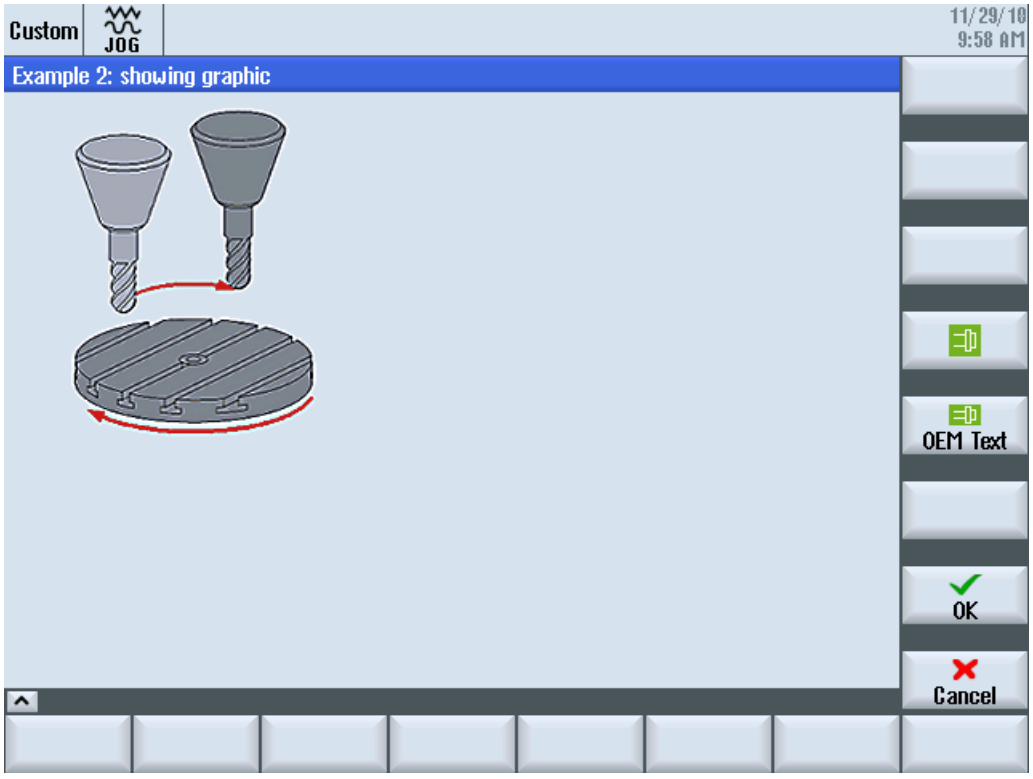


Figura 6-8 Ejemplo 3: Gráfico y pulsadores de menú

```
//S(Start)
HS7=("Example", ac7, se1)

PRESS(HS7)
  LM("Maske3")
END_PRESS

//END

//M(Maske3/"Example 2: showing graphic"/"example.png")
HS1=("")
HS2=("")
HS3=("")
HS4=("")
HS5=("")
HS6=("")
HS7=("")
HS8=("")
VS1=("")
VS2=("")
```

```
VS3=("")
VS4=("\\sp_ok.png",,SE1)
VS5=(["\\sp_ok_small.png","OEM Text"],,SE1)
VS6=("")
VS7=(SOFTKEY_OK,,SE1)
VS8=(SOFTKEY_CANCEL,,SE1)
PRESS(VS4)
  EXIT
END_PRESS
PRESS(VS5)
  EXIT
END_PRESS
PRESS(VS7)
  EXIT
END_PRESS
PRESS(VS8)
  EXIT
END_PRESS
//END
```

6.2.2 Texto dependiente del idioma

Vista general

Los textos dependientes del idioma se utilizan para:

- Rótulos de pulsadores de menú
- Títulos
- Textos de ayuda
- Cualquier otro texto

Los textos dependientes del idioma para diálogos se guardan en ficheros de texto.

Los ficheros de texto se encuentran en los siguientes directorios:

- [Directorio de usuario del sistema]/lng
- [Directorio OEM del sistema]/lng
- [Directorio de addon del sistema]/lng

Nota

Los ficheros de texto deben guardarse en una ruta análoga a la de los ficheros de proyecto.

P. ej.:

[Directorio de usuario del sistema]/lng/[Fichero de texto]

[Directorio de usuario del sistema]/proj/[Fichero de configuración]

alsc.txt	Textos dependientes del idioma para los ciclos Siemens estándar
almc.txt	Textos dependientes del idioma para los ciclos de medida de Siemens

Los ficheros de texto utilizados en tiempo de ejecución del programa se indican en el fichero "easyscreen.ini":

```
[LANGUAGEFILES]
LngFile03 = user.txt      ;->user<_xxx>.txt (p. ej.: user_eng.txt)
```

El fichero "user.txt" se ha seleccionado aquí como ejemplo para un fichero de texto. En principio, el nombre se puede elegir libremente. En función del idioma de los textos incluidos en el fichero, se debe añadir también el código de idioma de acuerdo con la siguiente sintaxis. Después del nombre se añade un guión bajo y, a continuación, el código de idioma correspondiente, p. ej., "user_eng.txt".

Nota

No se deben usar LngFile01 y LngFile02 para los textos de usuario, ya que están asignados en el fichero estándar "easyscreen.ini".

Ver también

AUTOHOTSPOT

Ficheros de idioma específicos de máscara

Para evitar solapamientos con los rangos de números utilizados en los ficheros de idioma, debe ser posible utilizar en una máscara únicamente determinados ficheros de idioma.

Por ello, en la línea de definición de máscara se muestran los ficheros de idioma que se usarán, separados por comas. La máxima prioridad corresponde al último fichero indicado (de modo análogo a la lógica de "easyscreen.ini").

Todos los textos dependientes de idioma utilizados en la máscara se buscan prioritariamente en estos ficheros de idioma. Si no se encuentra el texto en dichos ficheros, se buscará a continuación en los ficheros de texto configurados en "easyscreen.ini".

Si no hay ficheros de idioma definidos en la máscara, solo se buscará en los ficheros de idioma indicados en "easyscreen.ini".

Nota

Si un fichero de idioma no existe en el idioma seleccionado actualmente, se utilizará el fichero de idioma inglés (de manera predeterminada).

Ejemplos:

```
//M(MyMask/$85597///// //"mytexts.txt, general.txt, mask.txt")
DEF ….
```

Este procedimiento puede usarse también para los pulsadores de menú de inicio:

```
//S(Start, "mytexts.txt, general.txt, mask.txt")
VS1=($85597)
PRESS(VS1)
   LM("MyMask","masks.com")
END_PRESS
```

Formato de los ficheros de texto

Los ficheros de texto se deben guardar codificados en formato UTF-8.

Nota

Al guardar los ficheros de configuración e idioma, tenga en cuenta que en el editor que utiliza la codificación está ajustada a UTF-8.

Forma de una entrada de texto

Sintaxis:	8xxxx 0 0 "Texto"		
Descripción:	Asignación entre el número de texto y el texto en el fichero		
Parámetro:	xxxx	85.000 a 89.999 900.000 a 999.999	Rango de números de identificación de textos reservado para el usuario. Los números se tienen que asignar de forma unívoca.
	"Texto"		Texto que aparece en el diálogo
	%n		Carácter de control en el texto para un cambio de línea

Los dos parámetros 2 y 3 separados por espacios son caracteres de control para la emisión del texto de alarma. Debido a la uniformidad del formato de texto, siempre deben ser cero con los textos de alarma.

Ejemplos de textos dependientes del idioma:

```
85000 0 0 "Plano de retirada"
85001 0 0 "Profundidad de taladrado"
85002 0 0 "Paso de rosca"
85003 0 0 "Radio de la caja"
```

6.3 Configuración de la ayuda en línea

6.3.1 Resumen

Además de la completa ayuda online ya existente, tiene la posibilidad de crear una ayuda online específica del fabricante e incorporarla en SINUMERIK Operate.

Esta ayuda online se crea en formato HTML, es decir, se compone de documentos HTML vinculados entre sí. El tema buscado se abre en una ventana aparte a través de un índice de contenido o alfabético. De forma similar a un explorador de documentos (p. ej. Explorer de Windows), en la mitad izquierda de la ventana se muestra una vista de selección y, cuando se hace clic sobre el tema deseado, la explicación al respecto aparece en la mitad derecha de la ventana.

No es posible realizar una selección contextual de páginas de ayuda online.

Procedimiento

1. Crear ficheros HTML
2. Crear libro de ayuda
3. Incorporar ayuda online en SINUMERIK Operate
4. Guardar ficheros de ayuda

Otros casos de aplicación

Pueden crearse ayudas online para las siguientes ampliaciones específicas del OEM y añadirse al sistema de ayuda online de SINUMERIK Operate:

- Ayuda online para **ciclos o funciones M** del fabricante de la máquina que amplían las posibilidades de programación de los controles SINUMERIK. A esta ayuda online se accede igual que a la ayuda online de SINUMERIK Operate para "programación".
- Ayuda online para **variables** específicas de OEM del fabricante de la máquina. A esta ayuda online se accede desde la vista de variables de SINUMERIK Operate.

Programar ficheros de ayuda online

Para más opciones de diseño de la ayuda online, puede utilizar el "Paquete de programación SINUMERIK HMI sl". Con este paquete de programación es posible desarrollar aplicaciones de lenguaje de alto nivel en el lenguaje de programación C++ para SINUMERIK Operate en la NCU 7x0.

Nota

El "Paquete de programación SINUMERIK HMI sl" es una opción de software que se pide aparte. La documentación correspondiente se suministra con el paquete de programación.

6.3.2 Crear ficheros HTML

Cree los ficheros de ayuda en formato HTML. Al hacerlo es posible guardar toda la información en un solo fichero HTML o separarla en varios ficheros HTML.

Puede asignar usted mismo los nombres de los ficheros, pero debe tener en cuenta lo siguiente:

- Las referencias dentro de los ficheros HTML deben indicarse siempre con rutas relativas. Esta es la única forma de asegurar que las referencias funcionan igualmente tanto en el ordenador de desarrollo como en el sistema de destino.
- Si se debe saltar a determinados puntos dentro de un fichero HTML por medio de un enlace, se deben definir para ello las llamadas anclas.
  Ejemplo de ancla HTML:
  `<a name="myAnchor">Esto es un ancla</a>`
- El contenido de los documentos HTML debe guardarse con la codificación UTF-8. Solo así se garantiza que los documentos HTML de SINUMERIK Operate se muestren correctamente en todos los idiomas admitidos.
- Son compatibles los siguientes subconjuntos de las funciones HTML:

Tags HTML

Tag	Descripción	Comentarios
a	Anchor or link	Atributos compatibles: href y name
address	Address	
big	Larger font	
blockquote	Indented paragraph	
body	Document body	Atributos compatibles: bgcolor (#RRGGBB)
br	Line break	
center	Centered paragraph	
cite	Inline citation	Mismo efecto que el tag i
code	Code	Mismo efecto que el tag tt
dd	Definition data	
dfn	Definition	Mismo efecto que el tag i
div	Document division	Son compatibles los atributos de secuencia estándar

Tag	Descripción	Comentarios
dl	Definition list	Son compatibles los atributos de secuencia estándar
dt	Definition term	Son compatibles los atributos de secuencia estándar
em	Emphasized	Mismo efecto que el tag i
font	Font size, family, color	Atributos compatibles: size, face, and color (#RRGGBB)
h1	Level 1 heading	Son compatibles los atributos de secuencia estándar
h2	Level 2 heading	Son compatibles los atributos de secuencia estándar
h3	Level 3 heading	Son compatibles los atributos de secuencia estándar
h4	Level 4 heading	Son compatibles los atributos de secuencia estándar
h5	Level 5 heading	Son compatibles los atributos de secuencia estándar
h6	Level 6 heading	Son compatibles los atributos de secuencia estándar
head	Document header	
hr	Horizontal line	Atributos compatibles: width (se puede indicar como valor absoluto o relativo)
html	HTML document	
i	Italic	
img	Image	Atributos compatibles: src, width, height
kbd	User-entered text	
meta	Meta-information	
li	List item	
nobr	Non-breakable text	
ol	Ordered list	Son compatibles los atributos estándar para listas
p	Paragraph	Son compatibles los atributos de secuencia estándar (preajuste: left-aligned)
pre	Preformated text	
s	Strikethrough	
samp	Sample code	Mismo efecto que el tag tt
small	Small font	
span	Grouped elements	
strong	Strong	El texto plano se marca en negrita (sustituye el tag b)
sub	Subscript	
sup	Superscript	
table	Table	Atributos compatibles: border, bgcolor (#RRGGBB), cellspacing, cellpadding, width (absolut o relative), height
tbody	Table body	Sin efecto
td	Table data cell	Son compatibles los atributos estándar para celdas de tablas
tfoot	Table footer	Sin efecto
th	Table header cell	Son compatibles los atributos estándar para celdas de tablas
thead	Table header	Se utiliza para imprimir tablas que ocupan varias páginas
title	Document title	
tr	Table row	Atributos compatibles: bgcolor (#RRGGBB)
tt	Typewrite font	
u	Underlined	

Tag	Descripción	Comentarios
ul	Unordered list	Son compatibles los atributos estándar para listas
var	Variable	Mismo efecto que el tag tt

Atributos de secuencia

Los siguientes atributos son compatibles con los tags div, dl, dt, h1, h2, h3, h4, h5, h6, p:

- align (left, right, center, justify)
- dir (ltr, rtl)

Atributos estándar para listas

Los siguientes atributos son compatibles con los tags ol y ul:

- type (1, a, A, square, disc, circle)

Atributos estándar para tablas

Los siguientes atributos son compatibles con los tags td y th:

- width (absolute, relative, no-value)
- bgcolor (#RRGGBB)
- colspan
- rowspan
- align (left, right, center, justify)
- valign (top, middle, bottom)

Propiedades CSS

La siguiente tabla contiene las funciones CSS compatibles:

Propiedad	Valores	Descripción
background-color	<color>	Color de fondo para elementos
background-image	<uri>	Imagen de fondo para elementos
color	<color>	Color de primer plano para texto
text-indent	<length>px	Sangrado de la primera línea de un párrafo en píxeles
white-space	normal | pre | nowrap | pre-wrap	Determina cómo deben tratarse los caracteres de espacio en blanco en documentos HTML
margin-top	<length>px	Ancho del margen superior de párrafo en píxeles
margin-bottom	<length>px	Ancho del margen inferior de párrafo en píxeles
margin-left	<length>px	Ancho del margen izquierdo de párrafo en píxeles
margin-right	<length>px	Ancho del margen derecho de párrafo en píxeles
vertical-align	baseline | sub | super | middle | top | bottom	Orientación vertical para texto (en las tablas solo son compatibles los valores middle, top y bottom)

Propiedad	Valores	Descripción
border-color	<color>	Color de borde para tablas de texto
border-style	none | dotted | dashed | dot-dash | dot-dot-dash | solid | double | groove | ridge | inset | outset	Estilo de borde para tablas de texto
background	[<'background-color'> || <'background-image'>]	Notación abreviada para la propiedad background
page-break-before	[auto | always]	Salto de página antes de un párrafo/una tabla
page-break-after	[auto | always]	Salto de página tras un párrafo/una tabla
background-image	<uri>	Imagen de fondo para elementos

Selectores CSS compatibles

Son compatibles todas las clases de selector CSS 2.1 a excepción de las denominadas pseudoclases de selector, como :first-child, :visited y :hover.

6.3.3 Crear libro de ayuda

El libro de ayuda es un fichero XML en el que se define la estructura de la ayuda online. En este fichero se definen:

- documentos HTML;
- el índice de contenido y alfabético.

Sintaxis para el libro de ayuda

Tag	Cantidad	Significado	
HMI_SL_HELP	1	Elemento raíz del documento XML	
I-BOOK I I I I I I I I I I	+	Designa un libro de ayuda. El nombre puede elegirse libremente siempre que no se utilice ningún nombre predefinido por el sistema (como p. ej., sinumerik_alarm_plc_pmc). En el ejemplo, el nombre del libro de ayuda es: "hmi_myhelp" Atributos:	
		ref	Designa el documento HTML que se muestra como página de inicio para el libro de ayuda.
		titel	Título del libro de ayuda que se indica en el índice de contenido.
		helpdir	Directorio que contiene la ayuda online del libro de ayuda.
I-ENTRY II II II II II II II	*	Capítulo de la ayuda online Atributos:	
		ref	Designa el documento HTML que se muestra como página de inicio para el capítulo.
		titel	Título del capítulo que se indica en el índice de contenido.

Tag	Cantidad	Significado	
II-INDEX_ENTRY II II II II II II II	*	Término de búsqueda que debe visualizarse Atributos:	
		ref	Designa el documento HTML al que se salta para esa entrada de término de búsqueda.
		titel	Título del término de búsqueda que se indica en el índice alfabético.

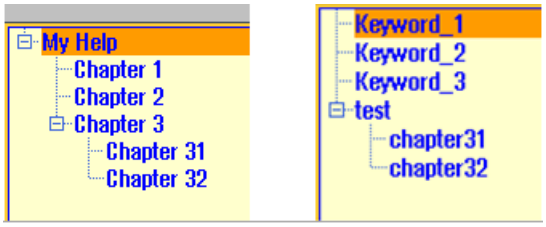
En el caso de la columna "Cantidad":
* significa 0 o varios
+ significa 1 o varios

Ejemplo de libro de ayuda

En el ejemplo siguiente se describe la estructura de un libro de ayuda con el nombre "My Help". Además, este es la base del índice de contenido y alfabético.

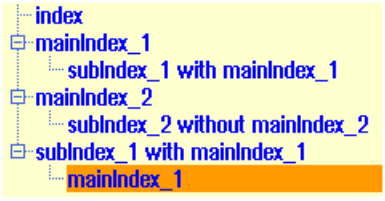
```
<?xml version="1.0" encoding="utf-8"?>
<HMI_SL_HELP language="en-US">
  <BOOK ref="index.html" title="My Help" helpdir="hmi_myhelp">
    <ENTRY ref="chapter_1.html" title="Chapter 1">
      <INDEX_ENTRY ref="chapter_1html#Keyword_1" title="Keyword_1"/>
      <INDEX_ENTRY ref="chapter_1.html#Keyword_2" title="Keyword_2"/>
    </ENTRY>
    <ENTRY ref="chapter_2.html" title="Chapter 2">
      <INDEX_ENTRY ref="chapter_2.html#Keyword_3" title="Keyword_3"/>
    </ENTRY>
    <ENTRY ref="chapter_3.html" title="Chapter 3">
      <ENTRY ref="chapter_31.html" title="Chapter 31">
        INDEX_ENTRY ref="chapter_31.html#test" title="test;chapter31"/>
      </ENTRY>
      <ENTRY ref="chapter_32.html" title="Chapter 32">
        INDEX_ENTRY ref="chapter_32.html#test" title="test;chapter32"/>
      </ENTRY>
    </ENTRY>
  </BOOK>
</HMI_SL_HELP>
```

El libro consta de tres capítulos, de los cuales el tercero incluye otros dos subcapítulos. Cada uno de los distintos términos de búsqueda está definido dentro del capítulo.



El índice alfabético puede tener los tres formatos siguientes:

1. Entrada individual:
   `<INDEX_ENTRY ...title="index"/>`
2. Dos entradas de dos niveles en las que cada título tiene una entrada principal y una subentrada. Separe las entradas entre sí con una coma.
   `<INDEX_ENTRY ...title="mainIndex_1,subIndex_1 with mainIndex_1"/>`
3. Entrada de dos niveles en la que el primer título es la entrada principal y el segundo título es la subentrada. Separe las entradas entre sí con un punto y coma.
   `<INDEX_ENTRY ...title="mainIndex_2;subIndex_2 without mainIndex_1"/>`



6.3.4 Incorporar ayuda online en SINUMERIK Operate

Si desea incorporar el libro de ayuda creado en el sistema de ayuda online de SINUMERIK Operate, necesita el fichero "slhlp.xml".

Descripción de formato de "slhlp.xml"

Tag	Canti-dad	Significado	
CONFIGURATION	1	Elemento raíz del documento XML. Indica que se trata de un fichero de configuración.	
I-OnlineHelpFiles	1	Inicia la sección de los libros de ayuda online.	
II-<help_book>	*	Inicia la sección de un libro de ayuda.	
III-EntriesFile III III III III III	1	Nombre de fichero del libro de ayuda con las entradas de contenido e índice. Atributos:	
		value	Nombre del fichero XML
		type	Tipo de datos del valor (QString)

Tag	Cantidad	Significado	
III-Technology III III III III III III III III III IIII	0, 1	Indica la tecnología para la que es válido el libro de ayuda. En este caso "All" se aplica a todas las tecnologías. Si el libro de ayuda es válido para varias tecnologías, estas últimas se indican separadas por una coma. Valores posibles: All, Universal, Milling, Turning, Grinding, Stroking, Punching Atributos:	
		value	Indicación de tecnología
		type	Tipo de datos del valor (QString)
III -DisableSearch III III III III III	0, 1	Desactivar búsqueda de términos para el libro de ayuda. Atributos:	
		value	true, false
		type	type Tipo de datos del valor (bool)
III-DisableFullTextSearch III III III III	0, 1	Desactivar búsqueda de texto completo para el libro de ayuda. Atributos:	
		value	true, false
		type	type Tipo de datos del valor (bool)
III-DisableIndex III III III III	0, 1	Desactivar índice alfabético para el libro de ayuda. Atributos:	
		value	true, false
		type	type Tipo de datos del valor (bool)
III-DisableContent III III III III	0, 1	Desactivar índice de contenido para el libro de ayuda. Atributos:	
		value	true, false
		type	type Tipo de datos del valor (bool)
III-DefaultLanguage III III III III III	0, 1	Abreviatura del idioma que debe mostrarse si está disponible el idioma de país actual para el libro de ayuda. Atributos:	
		value	chs, deu, eng, esp, fra, ita...
		type	Tipo de datos del valor (QString)

En el caso de la columna "Cantidad":
* significa 0 o varios

Ejemplo de fichero "slhlp.xml"

En el siguiente ejemplo, el libro de ayuda "hmi_myhelp.xml" se comunica a SINUMERIK Operate.

El índice alfabético no está activado para el libro de ayuda.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<!DOCTYPE CONFIGURATION>
<CONFIGURATION>
   <OnlineHelpFiles>
```

```
    <hmi_myHelp>
      <EntriesFile value="hmi_myhelp.xml" type="QString"/>
      <DisableIndex value="frue" type="bool"/>
    </hmi_myHelp>
  </OnlineHelpFiles>
</CONFIGURATION>
```

6.3.5 Guardar ficheros de ayuda

Guardar ficheros de ayuda en el sistema de destino

1. Abra el directorio /**oem**/sinumerik/hmi/hlp y cree una carpeta nueva para el idioma deseado. Utilice para ello el código de idioma especificado.
   Es imprescindible escribir los nombres de carpeta en minúsculas.
   Si, p. ej., incorpora una ayuda para los idiomas alemán e inglés, cree las carpetas "deu" y "eng".
2. Guarde el libro de ayuda, por ejemplo "hmi_myhelp.xml", en las carpetas "deu" y "eng", respectivamente.
3. Copie los ficheros de ayuda en los directorios, p. ej., /**oem**/sinumerik/hmi/hlp/deu/hmi_myhelp para los ficheros de ayuda en alemán y /**oem**/sinumerik/hmi/hlp/eng/hmi_myhelp para los ficheros de ayuda en inglés.
4. Guarde el fichero de configuración "slhlp.xml" en el directorio /**oem**/sinumerik/hmi/cfg.
5. Reinicie el HMI.

Nota

En la indicación del índice de contenido y alfabético de un libro de ayuda, los ficheros de ayuda se guardan en formato binario (slhlp_<Hilfe-Buch_*.hmi) en el directorio /siemens/sinumerik/sys_cache/hmi/hlp para que el proceso sea más rápido. Si modifica el libro de ayuda, deberá borrar siempre estos ficheros.

6.3.6 Ficheros de ayuda en formato PDF

Además de los ficheros de ayuda en formato HTML, también existe la posibilidad de incorporar información en formato PDF en el software de manejo. Las distintas ayudas en PDF pueden abrirse mediante enlaces desde el índice de contenidos o el índice o directamente desde ficheros HTML.

Guardar ayudas en PDF

Copie las ayudas en PDF en uno de los directorios siguientes:

```
/oem/sinumerik/hmi/hlp/<lng>/<hmi_myhelp>
/user/sinumerik/hmi/hlp/<lng>/<hmi_myhelp>
```

Incorporar ayudas en PDF

En la configuración de diálogos o de índices de contenidos o alfabéticos, los ficheros con extensión ".pdf" se incorporan de la misma manera que aquellos que tienen extensión ".html":

```
<ENTRY ref="myFile.pdf" title="Help 1">
```

Remita de los ficheros HTML a las ayudas en PDF por medio de un enlace:

```
<a href="myFile.pdf">My Help File</a>
```

Nota

En las ayudas en PDF no es posible seleccionar marcas de salto de forma contextualizada ni saltar a otros ficheros HTML o PDF.

La función de búsqueda solo está disponible dentro del mismo fichero PDF. No es posible realizar búsquedas más amplias en varias ayudas en PDF.

Variables

7

7.1 Definir variables

Valor de las variables

La propiedad más importante de una variable es su valor.

El valor de la variable se puede asignar a través de:

- El ajuste estándar en la definición de las variables
- La asignación de una variable de sistema o de usuario
- Un método

Programación

Sintaxis:	Identificador **val** = Valor de la variable Identificador = Valor de la variable	
Descripción:	Valor de la variable val (value)	
Parámetro:	Identificador:	Nombre de la variable
	Valor de la variable:	Valor de la variable
Ejemplo:	`VAR3 = VAR4 + SIN(VAR5)` `VAR3.VAL = VAR4 + SIN(VAR5)`	

Estado de las variables

Con la propiedad Estado de las variables puede consultarse en tiempo de ejecución si una variable contiene un valor válido. Esta propiedad es legible y modificable con el valor FALSE = 0.

Programación

Sintaxis:	Identificador.**vld**	
Descripción:	Estado de la variable vld (validation)	
Parámetro:	Identificador:	Nombre de la variable
		El resultado de la consulta puede ser:
	FALSE =	Valor no válido
	TRUE =	Valor válido

Ejemplo:	`IF VAR1.VLD == FALSE`
	`  VAR1 = 84`
	`ENDIF`	

7.2 Ejemplos de aplicación

Variables auxiliares

Las variables auxiliares son variables internas de cálculo. Las variables de cálculo se definen como variables, pero no poseen ninguna propiedad excepto el valor y el estado de variable, es decir, las variables auxiliares no son visibles en el diálogo. Las variables auxiliares son de tipo VARIANT.

Programación

Sintaxis:	DEF [Identificador]	
Descripción:	Variables de cálculo internas del tipo VARIANT	
Parámetro:	Identificador:	Nombre de la variable auxiliar
Ejemplo:	`DEF OTTO ;Definición de una variable auxiliar`	

Sintaxis:	Identificador.val = Valor de la variable auxiliar Identificador = Valor de la variable auxiliar	
Descripción:	El valor de una variable auxiliar se asigna en un método.	
Parámetro:	Identificador:	Nombre de la variable auxiliar
	Valor de la variable auxiliar:	Contenido de la variable auxiliar

Ejemplo:

```
LOAD
  OTTO = "Test"             ; Asignar el valor "Test"  a la variable auxiliar Otto
END_LOAD
LOAD
  OTTO = REG[9].VAL         ; Asignar el valor del registro a la variable auxiliar
END_LOAD                    Otto
```

Cálculo con variables

Las variables se calculan cada vez que se abandona el campo de entrada/salida (mediante la tecla ENTER o de cambio). El cálculo se configura en un método CHANGE y se ejecuta con cada modificación del valor.

A través del estado de la variable se puede consultar si una variable tiene un valor válido, p. ej.:

```
IF VAR1.VLD == FALSE
   VAR1 = 84
ENDIF
```

Acceso indirecto a una variable de sistema

Se puede acceder a una variable de sistema también de manera indirecta, es decir, en función de otra variable:

```
PRESS(HS1)
  ACHSE=ACHSE+1
  WEG.VAR="$AA_DTBW["<<ACHSE<<"]"        ;Acceso a la dirección del eje por medio de
                                          una variable
END_PRESS
```

Modificación de rótulo de pulsador de menú

Ejemplo:

```
HS3.st = "Nuevo texto"                   ;Modificación de rótulo de pulsador de menú
```

7.3 Ejemplo 1: Asignar tipo de variable, textos, pantalla de ayuda, colores y tooltips

Ejemplo 1a

En el siguiente ejemplo se define una variable para la que pueden establecerse las propiedades de tipo de variable, textos, pantalla de ayuda y colores.

DEF Var1 = (R///,"Valor real",,"mm"//"Var1.png"////8,2)			
	Tipo de variable:		REAL
	Textos:		
		Texto breve:	valor real
		Texto de unidad:	mm
	Pantalla de ayuda:		Var1.png
	Colores:		
		Color de primer plano:	8 (marrón)
		Color de fondo:	2 (naranja)

Ejemplo 1b

En el siguiente ejemplo se define una variable para la que pueden establecerse las propiedades de tipo de variable, valor por defecto, textos, sugerencia de herramienta, modo de entrada y posición de texto breve.

DEF Var2 = (I//5/"","Wert","","," Tooltip-Text"/wr2///20,250,50)			
	Tipo de variable:		INTEGER
	Valor por defecto:		5
	Textos:		
		Texto breve:	Valor (posible ID de texto de idioma)
		Tooltip:	TooltipText
	Atributos:		
		Modo de entrada	lectura y escritura
	Posición de texto breve:		
		Distancia desde la izquierda	20
		Distancia desde la parte superior	250
		Anchura:	50

Consulte también

Parámetros de variables (Página 93)

7.4 Ejemplo 2: Asignar tipo de variable, valores límite, atributos, posición texto breve

Ejemplo 2

En el siguiente ejemplo se define una variable para la que pueden establecerse las propiedades de tipo de variable, valores límite, modo de entrada, orientación y posición.

DEF Var2 = (I/0,10///wr1,al1/// , ,300)			
	Tipo de variable:		INTEGER
	Valores límite o entradas de campo de alternancia:		MÍN: 0 MÁX: 10
	Atributos:		
		Modo de entrada	sólo lectura
		Ajuste de texto para texto breve	justificado a la derecha
	Posición de texto breve:		
		Anchura:	300

Consulte también

Parámetros de variables (Página 93)

7.5 Ejemplo 3: Asignar tipo de variable, valor por defecto, variable de sistema o de usuario, posición campo de entrada/salida

Ejemplo 3

En el siguiente ejemplo se define una variable para la que pueden establecerse las propiedades de tipo de variable, valor por defecto, variable de sistema o de usuario y posición.

DEF Var3 = (R//10////"$R[1]"//300,10,200//)			
	Tipo de variable:		REAL
	Valor por defecto:		10
	Variable de sistema o de usuario:		$R[1] (parámetro R 1)
	Posición de texto breve:		Posición estándar relativa al campo de entrada/salida
	Posición de campo de entrada/salida:		
		Distancia desde la izquierda	300
		Distancia desde la parte superior	10
		Anchura:	200

Consulte también

Parámetros de variables (Página 93)

7.6 Ejemplo 4: campo de alternancia y campo de lista

Ejemplo 4a

Diferentes entradas en el campo de alternancia:

```
DEF Var1 = (I/* 0,1,2,3)                ;Campo de alternancia sencillo para alternar entre valores
                                        numéricos
DEF Var2 = (S/* "On", "Off")            ;Campo de alternancia sencillo para alternar entre cadenas
DEF Var3 = (B/* 1="On", 0="Off")        ;Campo de alternancia ampliado para alternar entre valores
                                        numéricos, en el que todos los valores numéricos tienen asig-
                                        nado un texto de visualización
DEF Var4 = (R/* ARR1)                   ;Campo de alternancia que toma sus valores de alternancia de
                                        una matriz
```

Ejemplo 4b

El campo de lista corresponde a la configuración de un campo de alternancia pero debe establecerse además el tipo de visualización para el campo de lista (atributo de variable DT = 4).

DEF VAR_LISTBOX_Text = (S/*$80000,$80001,$80002,$80003,$80004/$80001//DT4////200,,340,60)			
	Tipo de variable:		STRING
	Valores límite/campo de alternancia:		*$80000,$80001,$80002,$80003,$80004 (lista de los textos dependientes del idioma que deben mostrarse)
	Valor por defecto:		$80001
	Atributos:		
		Tipo de visualización:	4 (campo de lista)
	Posición de campo de entrada/salida:		
		Distancia desde la izquierda:	200
		Anchura:	340
		Altura:	60
	Colores:		
		Color de primer plano:	6 (azul)
		Color de fondo:	10 (blanco)

7.7 Ejemplo 5: visualización de imágenes

Ejemplo 5

Mostrar una imagen en lugar de un texto breve: El tamaño y la posición de la imagen se indican en "Posición de campo de entrada/salida (Left, Top, Width, Height)".

DEF VAR6= (V///,"\\imagen1.png" ////160,40,50,50)			
	Tipo de variable:		VARIANT
	Textos:		
		Texto breve:	imagen1.png
	Posición campo de entrada/salida:		
		Distancia desde la izquierda:	160
		Distancia desde la parte superior:	40
		Anchura:	50
		Altura:	50

7.8 Ejemplo 6: barra de progreso

La barra de progreso es un tipo de visualización especial del campo de entrada/salida y solo está concebido para la visualización sin entrada de datos.

En principio hay dos tipos de barras de progreso:

1. Barra de progreso con hasta dos cambios de color para indicar, p. ej., la temperatura o la carga (ver el ejemplo 6a)
2. Barra de progreso para indicar el progreso (sin cambio de color) en estilo de Operate (ver ejemplo 6b)

Ejemplo 6a

Barra de progreso con dos cambios de color:

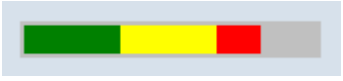


Figura 7-1 Barra de progreso con dos cambios de color

DEF PROGGY0 = (R/0,150,50,100///DT1,DO0//"$R[10]"//,,150/3,4,,,,,,9,7)		
Tipo de variable:		REAL
Valores límite/campo de alternancia:		
	MÍN:	0
	MÁX:	150
	Valor de señal SVAL1:	50
	Valor de señal SVAL2:	100
Atributos:		
	Modo de visualización DT:	1 (barra de progreso)
	Opción de visualización DO:	0 (de izquierda a derecha (por defecto))
Variable de sistema o de usuario:		$R[10]
Posición de campo de entrada/salida:		
	Anchura:	150
Colores:		
	Color de primer plano:	3 (verde oscuro)
	Color de fondo:	4 (gris claro)
	Color de señal SC1:	9 (amarillo)
	Color de señal SC2:	7 (rojo)

Para utilizar una barra de progreso con cambio de color el modo de visualización DT (DisplayType) debe estar establecido como 1.

La orientación del indicador de progreso la determina el atributo de opción de visualización DO (DisplayOption):

0: de izquierda a derecha (por defecto)

1: de derecha a izquierda

2: de abajo hacia arriba

3: de arriba hacia abajo

Para que se visualice la barra de progreso debe indicarse un valor MÍN y un valor MÁX (por ejemplo, MÍN: 0, MÁX: 150).

Con este ajuste ya se muestra, en función del valor actual de la variable PROGGY0, una barra de progreso con color de primer plano 3 (verde oscuro) y color de fondo 4 (gris claro).

También pueden definirse uno o dos valores de señal SVAL1 y SVAL2 (parámetro de valor límite), por ejemplo, SVAL1: 50 y SVAL2: 100). Con estos valores de señal cambia el color de primer plano del indicador de progreso. Los colores de señal correspondientes se establecen mediante los parámetros SC1 y SC2 (en el ejemplo anterior, SC1: 9 (amarillo) y SC2: 7 (rojo)).

Para indicar los valores límite para el indicador de progreso se aplica la siguiente regla:

MÍN < SVAL1 < SVAL2 < MÁX.

Ejemplo 6b

Barra de progreso sin cambio de color:

Figura 7-2 Barra de progreso

DEF PROGGY0 = (R/0,150///DT2,DO0//"$R[10]"//,,150/6,10)			
	Tipo de variable:		REAL
	Valores límite/campo de alternancia:		
		MÍN:	0
		MÁX:	150
	Atributos:		
		Tipo de visualización DT:	2 (barra de progreso)
		Opción de visualización DO:	0 (de izquierda a derecha (por defecto))
	Variable de sistema o de usuario:		$R[10]
	Posición de campo de entrada/salida:		
		Anchura:	150
	Colores:		
		Color de primer plano:	6 (azul)
		Color de fondo:	10 (blanco)

Para utilizar una barra de progreso sin cambio de color, el modo de visualización DT (DisplayType) debe estar establecido como 2.

La orientación del indicador de progreso la determina el atributo de visualización DO (DisplayOption) (ver descripción del ejemplo 6a).

Para que se visualice la barra de progreso debe indicarse un valor MÍN y un valor MÁX (por ejemplo, MÍN: 0, MÁX: 150).

Con este ajuste se muestra, en función del valor actual de la variable PROGGY0, una barra de progreso con color de primer plano 6 (azul) y color de fondo 10 (blanco).

7.9 Ejemplo 7: modo de entrada de contraseña (asteriscos)

Ejemplo 7

Para implementar un campo para introducir p. ej. la contraseña ocultando los caracteres, el modo de visualización DT debe ajustarse a 5. En lugar de los caracteres introducidos se muestran asteriscos.

Figura 7-3 Modo de entrada de contraseña (asteriscos)

DEF VAR_SET_PWD=(S//""//DT5)		
Tipo de variable:		STRING
Valor por defecto:		Cadena vacía
Atributos:		
	Modo de visualización DT:	5 (modo de entrada de contraseña)

7.10 Parámetros de variables

Vista general de los parámetros

En la siguiente vista de conjunto se explican brevemente los parámetros de las variables. Una descripción detallada se encuentra en los apartados siguientes.

Parámetros	**Descripción**	
Tipo de las variables (Página 100)	El tipo de variable debe indicarse.	
	R[x]:	REAL (+ cifra para las posiciones decimales)
	I:	INTEGER
	S[x]:	STRING (+ cifra para la longitud del string)
	C:	CHARACTER (carácter individual)
	B:	BOOL
	V:	VARIANT

Parámetros	Descripción	
Valores límite (Página 88)	Valor límite MÍN, valor límite MÁX Ajuste predeterminado: vacío Los valores límite se separan mediante comas. Los valores límite se pueden introducir para los tipos I, C y R en formato decimal o como carácter en la forma "A", "F".	
	Para que se visualice la barra de progreso con cambio de color (atributo de variable DT = 1) también pueden configurarse dos colores de señal SC1 y SC2 (Signal color), que se utilizarán como color de primer plano de la barra si se exceden los valores de señal SVAL1 o SVAL2. La indicación de los valores de señal se realiza como valores enteros. Ver ejemplo de aplicación de la barra de progreso (Página 90).	
Valor predeterminado (Página 104)	Si no se configura ningún valor por defecto y no se ha asignado ninguna variable de sistema o de usuario a la variable, se asigna el primer elemento del campo de alternancia. Si no está definido ningún campo de alternancia, no se ajusta ningún valor por defecto; es decir, que la variable tiene el estado "no calculada". Ajuste predeterminado: sin valor por defecto	
Campo de alternancia (Página 102)	Lista de selección con entradas predefinidas en el campo de entrada/salida: La lista se inicia con *, y las entradas se separan mediante comas. Se puede asignar un valor a las entradas. La entrada para el valor límite se interpreta en el campo de alternancia como una lista de selección. Si sólo se introduce un *, se crea un campo de alternancia variable. Ajuste predeterminado: ninguno	
Textos (Página 87)	El orden está predefinido. En lugar del texto breve también puede mostrarse una imagen. Ajuste predeterminado: vacío	
	Texto largo:	Texto en la línea de indicación
	Texto breve:	Nombre del elemento de diálogo
	Texto de gráficos:	Texto referido a los términos de un gráfico
	Texto de unidad:	Unidad del elemento de diálogo
	Sugerencia de herramienta	Sirven como información breve en una configuración de máscara para los campos de indicación y alternancia. La información se configura mediante texto plano e ID de texto de idioma.

Parámetros	Descripción	
Atributos (Página 88)	Los atributos influyen en las siguientes propiedades: • Modo de visualización • Opción de visualización • Frecuencia de actualización • Símbolo de alternancia • Sugerencia de herramienta • Modo de entrada • Nivel de acceso • Ajuste de texto para texto breve • Tamaño de fuente • Valores límite Los atributos se separan con comas; el orden puede ser cualquiera. Por cada componente puede llevarse a cabo una especificación.	
	Modo de visualización	dt0: estándar (campo de entrada/salida o campo de alternancia) (por defecto) dt1: barra de progreso (Progressbar) con cambio de color dt2: barra de progreso (Progressbar) sin cambio de color en estilo Operate dt4: campo de lista dt5: modo de entrada de contraseña (asteriscos)
	Opción de visualización	Especialmente para los modos de visualización dt1 y dt2 (barra de progreso) p. ej., puede ser necesario configurar asimismo una opción de visualización. do0: de izquierda a derecha (por defecto) do1: de derecha a izquierda do2: de abajo hacia arriba do3: de arriba hacia abajo
	Velocidad de actualización	Con el atributo UR (update rate) se puede controlar la actualización de la visualización y, con ella, la edición del correspondiente bloque CHANGE de variables o columnas grid configurado. Según la configuración, la carga de la CPU puede reducirse drásticamente y pueden acortarse así los tiempos de reacción de la interfaz de usuario. ur0: SlCap::standardUpdateRate() (actualmente = 200 ms, valor por defecto) ur1: 50 ms ur2: 100 ms ur3: 200 ms ur4: 500 ms ur5: 1000 ms ur6: 2000 ms ur7: 5000 ms ur8: 10000 ms
	Símbolo de alternancia	tg0: símbolo de alternancia apagado (por defecto) tg1: símbolo de alternancia encendido

Parámetros	Descripción	
		Si el atributo TG se establece como 1, aparece en la sugerencia de la herramienta del campo de entrada también el símbolo de alternancia. Ejemplo: `DEF OFFS = (R//123.456/,,,,"My ToolTip"/TG1)`

	Sugerencia de herramienta	El texto de la sugerencia de la herramienta también puede modificarse en tiempo de ejecución con la propiedad de la variable "TT".
		Ejemplo:
		`PRESS(VS1)`
		`  MyVar.TT = "My new ToolTip"`
		`END_PRESS`
		o bien
		`DEF MyVar=(R///,,,,"My ToolTip-text")`

	Modo de entrada	wr0: campo de entrada/salida oculto, texto breve visible wr1: lectura (no es posible ningún foco de entradas) wr2: lectura y escritura (la línea aparece en blanco) wr3: wr1 con foco wr4: todos los elementos de las variables están ocultos, no es posible ningún foco wr5: el valor introducido se guarda inmediatamente en cada pulsación de tecla (a diferencia de wr2; allí sólo se guarda al abandonar el campo o al accionar RETURN). Ajuste predeterminado: wr2
	Nivel de acceso	vacío: siempre modificable ac0...ac7: niveles de protección Cuando el nivel de acceso no es suficiente, aparece la línea en color gris, ajuste estándar: ac7
	Ajuste de texto para texto breve	al0: justificado a la izquierda al1: justificado a la derecha al2: centrado Ajuste predeterminado: al0
	Tamaño de fuente	fs1: tamaño estándar de fuente (8 puntos) fs2: tamaño doble de fuente Ajuste predeterminado: fs1 La distancia entre las líneas está fijada. Con el tamaño de fuente estándar caben 16 líneas en el diálogo. Los textos de gráficos y de unidad únicamente pueden configurarse en el tamaño estándar de fuente.
	Valores límite	De este modo se puede comprobar si el valor de la variable se encuentra entre los límites MIN y MAX indicados. Ajuste predeterminado: en función de los valores límite indicados li0: no se realiza ninguna comprobación li1: se comprueba el mínimo li2: se comprueba el máximo li3: se comprueban el mínimo y el máximo
	Comportamiento al abrir	Para la variable, los atributos cb indicados en una definición de variables tienen prioridad ante la indicación global cb en la definición del diálogo.

Parámetros	Descripción	
		Si hay varios atributos, se anotan separados mediante comas (ver también AUTOHOTSPOT).
	Llamada del método CHANGE	CB0: El método CHANGE se inicia al mostrarse la máscara si la variable tiene un valor válido en ese momento (p. ej. mediante valor predeterminado o variable de CN/PLC). CB1: El método CHANGE no se inicia de forma explícita al mostrarse la máscara. Si la variable tiene una variable de CN/PLC configurada, se llama de todos modos el método CHANGE, naturalmente.
	Empty Input (EI)	El atributo "EI" (= Empty Input) permite definir cómo debe reaccionar el campo de entrada cuando en él se ingresa una cadena vacía: EI0: Estándar, es decir, se aceptan ingresos vacíos en el campo de entrada EI1: Ingresos vacíos generan un error en el campo de entrada y se restaura el anterior valor válido (Undo).
Pantalla de ayuda (Página 87)	Fichero de pantalla de ayuda:	Nombre del fichero png Ajuste predeterminado: vacío
	El nombre del fichero de la pantalla de ayuda se escribe entre comillas dobles. La pantalla se abre automáticamente (en lugar del gráfico anterior) si el cursor pasa a esta variable.	
Variable de sistema o de usuario (Página 89)	A la variable se le puede asignar un dato de sistema o de usuario del CN/PLC. La variable de sistema o de usuario se escribe entre comillas dobles. Bibliografía: Manual de listas Variables del sistema, /PGAsl/	
Posición texto breve (Página 105)	Posición del texto breve (distancia desde la izquierda, distancia desde arriba, anchura). Las posiciones se indican en píxeles y se refieren a la esquina superior izquierda de la parte principal del diálogo. Los datos se separan mediante comas.	
Posición campo de entrada/salida (Página 105)	Posición del campo de entrada/salida (distancia desde la izquierda, distancia desde arriba, anchura). Las posiciones se indican en píxeles y se refieren a la esquina superior izquierda de la parte principal del diálogo. Los datos se separan mediante comas. Al modificar esta posición cambian también las posiciones del texto breve, del texto de gráfico y del texto de unidad.	

Parámetros	Descripción
Colores (Página 87)	Color de primer plano y de fondo para campos de entrada/salida, textos breves, textos de gráficos, textos de unidad y colores de señal para barra de progreso: los colores se separan mediante comas. Para la indicación del color, ver capítulo AUTOHOTSPOT. Ajuste predeterminado para campo de entrada/salida: Color de primer plano: negro; Color de fondo: blanco Los colores estándar del campo de entrada/salida dependen del modo de escritura "wr". Ajuste predeterminado para texto largo, texto de gráficos, texto de unidad: Color de primer plano: negro; Color de fondo: transparente.
	Para que se visualice la barra de progreso con cambio de color (atributo de variable DT = 1) también pueden configurarse dos colores de señal SC1 y SC2 (Signal color), que se utilizarán como color de primer plano de la barra si se exceden los valores de señal SVAL1 o SVAL2. Ver ejemplo de aplicación de la barra de progreso (Página 90).
	En la definición de variables se esperan los colores en el siguiente orden: 1. Color de primer plano del campo de entrada/salida: FC 2. Color de fondo del campo de entrada/salida: BC 3. Color de primer plano del texto breve: FC_ST 4. Color de fondo del texto breve: BC_ST 5. Color de primer plano del texto de gráficos: FC_GT 6. Color de fondo del texto de gráficos: BC_GT 7. Color de primer plano del texto de unidad: FC_UT 8. Color de fondo del texto de unidad: BC_UT 9. Color de señal 1 10. Color de señal 2
Fichero de ayuda en línea	El nombre del fichero de ayuda en línea se escribe entre comillas dobles.

Campo de entrada/salida con selección de unidades integrada	En los campos de entrada/salida con selección de unidades integrada puede cambiarse entre las diferentes unidades. Si se navega con el cursor en el campo de entrada, se resalta la selección de unidades integrada (el foco no tiene que definirse de manera explícita). Además, se muestra una sugerencia de herramienta con un símbolo de alternancia, que ofrece una indicación pertinente sobre esta funcionalidad.
	Ejemplos:
	`DEF VarEdit=(R/////////200,,100///"VarTgl")`
	`VarTgl=(S/*0="mm",1="inch"/0//WR2////302,,40)`
	o bien
	`DEF VarEdit_2={TYP="R", VAL=1.234, X=200, W=100, LINK_TGL="vartgl_2"},`
	`VARTgl_2={TYP="S", TGL="* 0=""mm"", 1=""inch""",WR=2, X=302, W=40}`

Variable: modificación de las propiedades

A las variables se les asigna un nuevo valor mediante la notación Identificador.Propiedad = Valor al efectuar una modificación. La expresión situada a la derecha del signo igual se evalúa y se asigna a la variable o a la propiedad de la variable.

Identificador.**ac** = nivel de acceso	(ac: access level)
Identificador.**al** = ajuste de texto	(al: alignment)
Identificador.**bc** = color de fondo del campo de entrada/salida	(bc: back color)
Identificador.**bc_gt** = color de fondo del texto de gráficos	(bc: back color) (gt: graphic text)
Identificador.**bc_st** = color de fondo del texto breve	(bc: back color) (st: short text)
Identificador.**bc_ut** = color de fondo del texto de unidad	(bc: back color) (ut: unit text)
Identificador.**do** = opción de visualización	(do: display option)
Identificador.**dt** = Modo de visualización	(dt: display type)
Identificador.**fc** = color de primer plano del campo de entrada/salida	(fc: front color)
Identificador.**fc_gt** = color de primer plano del texto de gráficos	(fc: front color) (gt: graphic text)
Identificador.**fc_st** = color de primer plano del texto breve	(fc: front color) (st: short text)
Identificador.**fc_ut** = color de primer plano del texto de unidad	(fc: front color) (ut: unit text)
Identificador.**fs** = tamaño de fuente	(fs: font size)
Identificador.**gt** = texto de gráficos	(gt: graphic text)
Identificador.**hlp** = pantalla de ayuda	(hlp: help)
Identificador.**li** = valor límite	(li: limit)
Identificador.**lt** = texto largo	(lt: long text)
Identificador.**max** = valores límite MÁX	(max: maximum)
Identificador.**min** = valores límite MÍN	(min: minimum)
Identificador.**sc** = color de señal	(sc: signal color)
Identificador.**st** = texto breve	(st: short text)
Identificador.**tg** = Símbolo de alternancia	(tg: toggle)
Identificador.**tt** = Sugerencia de herramienta	(tt: tooltip
Identificador.**typ** = tipo de variable	(typ: type)
Identificador.**ur** = Velocidad de actualización	(ur: update rate)
Identificador.**ut** = texto de unidad	(ut: unit text)
Identificador **val** = Valor de la variable	(val: value)
Identificador.**var** = variable de sistema o de usuario	(var: variable)
Identificador.**vld** = estado de la variable	(vld: validation)
Identificador.**wr** = modo de entrada	(wr: write)

Ver también

Sintaxis de configuración ampliada (Página 47)

7.11 Detalles sobre el tipo de variable

Tipo de variable INTEGER

Para el tipo "INTEGER", se puede disponer de las siguientes ampliaciones para especificar la representación en el campo de entrada/salida y el uso de memoria:

2.° carácter en el tipo de datos de ampliación

Formato de representación	
B	binario
D	decimal con signo
H	hexadecimal
Sin indicación	decimal con signo

3.er o 4.° carácter en el tipo de datos de ampliación

Uso de memoria	
B	Byte
W	Word
D	Double Word
BU	Byte sin signo
WU	Word sin signo
DU	Double Word sin signo

Orden de caracteres en el tipo de datos INTEGER

1. "I" Identificación básica como INTEGER
2. Formato de representación
3. Uso de memoria
4. "U" sin signo

Definiciones válidas de tipo para INTEGER:	
IB	Variable entera de 32 bits en representación binaria
IBD	Variable entera de 32 bits en representación binaria
IBW	Variable entera de 16 bits en representación binaria
IBB	Variable entera de 8 bits en representación binaria
I	Variable entera de 32 bits en representación decimal con signo
IDD	Variable entera de 32 bits en representación decimal con signo
IDW	Variable entera de 16 bits en representación decimal con signo
IDB	Variable entera de 8 bits en representación decimal con signo
IDDU	Variable entera de 32 bits en representación decimal sin signo
IDWU	Variable entera de 16 bits en representación decimal sin signo
IDBU	Variable entera de 8 bits en representación decimal sin signo

Definiciones válidas de tipo para INTEGER:	
IH	Variable entera de 32 bits en representación hexadecimal
IHDU	Variable entera de 32 bits en representación hexadecimal
IHWU	Variable entera de 16 bits en representación hexadecimal
IHBU	Variable entera de 8 bits en representación hexadecimal

Tipo de variable VARIANT

La variable de tipo VARIANT se determina mediante el tipo de datos de la última asignación de valores. Si el valor asignado o introducido empieza con '-'. '+', '.' o un número ('0'-'9'), se interpreta como valor numérico. En el resto de casos, como cadena.

Su valor puede consultarse mediante la función ISNUM o ISSTR. El tipo VARIANT es apto principalmente para escribir, a elección,
un nombre de variable o valores numéricos en el código de CN.

Programación

El tipo de datos de la variable se puede comprobar:

Sintaxis:	**ISNUM** (VAR)	
Parámetro:	VAR	Nombre de la variable cuyo tipo de dato debe comprobarse.
		El resultado de la consulta puede ser:
	FALSE =	no es una variable numérica (tipo de dato = STRING)
	TRUE =	variable numérica (tipo de dato = REAL)

Sintaxis:	**ISSTR** (VAR)	
Parámetro:	VAR	Nombre de la variable cuyo tipo de dato debe comprobarse.
		El resultado de la consulta puede ser:
	FALSE =	variable numérica (tipo de dato = REAL)
	TRUE =	no es una variable numérica (tipo de dato = STRING)
Ejemplo:		

```
IF ISNUM(VAR1) == TRUE
IF ISSTR(REG[4]+2) == TRUE
```

El modo de visualización de la variable se puede modificar:

- En el tipo INTEGER se puede modificar el modo de representación.

B	binario
D	decimal con signo
H	hexadecimal
sin signo	
más U (unsigned) en cada caso para variables sin signo.	

- En el tipo REAL solo puede modificarse la cantidad de posiciones decimales.
  No está permitido modificar el tipo y, en caso de hacerse, aparece un aviso de error en el fichero "easyscreen_log.txt".
  Ejemplo:
  ```
  Var1.tipo = "IBW"
  Var2.tipo = "R3"
  ```

Formatos numéricos

Los números pueden representarse en notación binaria, decimal, hexadecimal o exponencial:

binario	B01110110
decimal	123,45
hexadecimal	HF1A9
exponencial	-1.23EX-3
Ejemplos:	
	`VAR1 = HF1A9`
	`REG[0] = B01110110`
	`DEF VAR7 = (R//-1.23EX-3)`

Nota

Al generar códigos mediante la función "GC" solo se tendrán en cuenta los valores numéricos en representación decimal o exponencial, **no** en representación binaria o hexadecimal.

Consulte también

Parámetros de variables (Página 93)

7.12 Detalles sobre el campo de alternancia

Descripción

Mediante la ampliación del campo de alternancia pueden mostrarse textos (entradas en el campo de alternancia) en función de las variables del CN/PLC. Una variable que utiliza una ampliación del campo de alternancia es de sólo lectura. Al pulsar la tecla INSERT, se expande la lista del campo de alternancia.

Programación

Sintaxis:	**DEF VAR1=(IB/+ $85000/15///"DB90.DBB5")** o bien **DEF VAR_TGL = (S/* "Hello", "Run", "MyScreens"/"Run")**	
Descripción:	Al visualizar el diálogo se emite el contenido del número de texto $85015 en el campo de entrada/salida. En las variables de sistema DB90.DBB5 se introduce el valor por defecto 15. Si se modifica el valor en la variable de sistema DB90.DBB5, se volverá a formar el número de texto mostrado $(85000 + <DB90.DBB5>) con cada modificación.	
Parámetro:	Tipo de las variables	Tipo de la variable especificada en la variable de sistema o de usuario
	Número de texto	Número (base) del texto dependiente del idioma que se considera como número básico
	Variable de sistema o de usuario	Variable de sistema o de usuario (offset) a través de la cual se forma el número de texto definitivo (base + offset).

Imágenes en función del campo de alternancia

El campo de alternancia se rellena con gráficos alternantes: Si el byte de marcas tiene el valor 1, se muestra "imagen1.png" ; si tiene el valor 2, se muestra "imagen2.png".

```
DEF VAR1=(IDB/*1="\\bild1.png", 2="\\bild2.png"//,$85000/wr1//"MB[130]"//160,40,50,50)
```

La posición y el tamaño de la figura se indican en "Posición campo de entrada/salida (Left, Top, Width, Height)".

Tecla de alternancia virtual

Para los campos de alternancia sin lista configurada, como `DEF NoTglList=(R/*)`, no hay ninguna lista. El siguiente elemento se crea una vez que se acciona la tecla de alternancia o se ejecuta el método CHANGE() pertinente de la variable correspondiente.

Para estos casos, se muestra a la derecha, junto al campo de alternancia, un pequeño teclado virtual que solo consta de la tecla de alternancia para el manejo con un panel táctil.

La visualización de la tecla de alternancia virtual puede forzarse también, independientemente del panel táctil, a través de la siguiente entrada en el fichero de configuración "slguiconfig.ini".

```
[VirtualKeyboard]
; fuerza la función de teclado de alternancia virtual de easyscreen
ForceEasyscreenVirtualToggleKey = true
```

Consulte también

Parámetros de variables (Página 93)

7.13 Detalles sobre los valores por defecto

Vista general

En función de si el campo de una variable (campo de entrada/salida o campo de alternancia) tiene asignado un valor por defecto, una variable de usuario o de sistema o ambos, se obtienen diferentes estados para las variables (no calculado: la selección sólo es posible una vez que la variable tenga asignado un valor válido).

Efecto de los valores por defecto

Condición			**Reacción del tipo de campo**
Tipo de campo	**Valor predeterminado**	**Variable de sistema o de usuario**	
Campo E/S	Sí	Sí	Escribe los valores por defecto en la variable de sistema o de usuario
	No	Sí	Utiliza la variable de sistema o de usuario como valor por defecto
	Error	Sí	No calculada, no se escriben ni utilizan las variables de sistema o de usuario
	Sí	No	Valor predeterminado
	No	No	No calculada
	Error	No	No calculada
	Sí	Error	No calculada
	No	Error	No calculada
	Error	Error	No calculada
Campo de alternancia	Sí	Sí	Escribe los valores por defecto en la variable de sistema o de usuario
	No	Sí	Utiliza la variable de sistema o de usuario como valor por defecto
	Error	Sí	No calculada, no se escriben ni utilizan las variables de sistema o de usuario
	Sí	No	Valor predeterminado
	No	No	Valor por defecto = primer elemento del campo de alternancia
	Error	No	No calculada
	Sí	Error	No calculada
	No	Error	No calculada
	Error	Error	No calculada

Consulte también

Parámetros de variables (Página 93)

7.14 Detalles sobre la posición del texto breve, posición del campo de entrada/salida

Vista general

El texto breve y de gráficos, así como el campo de entrada/salida y el texto de unidad forman una unidad. Es decir, que las posiciones para el texto breve influyen también en el texto de gráficos y los datos para el campo de entrada/salida y el texto de unidad.

Programación

El dato de posición configurado sobrescribe el valor estándar, es decir, solo puede modificarse un único valor. Si no se configura ningún dato de posición para los siguientes elementos de diálogo, se tomarán los datos del último elemento de diálogo.

Si no se indican posiciones para ningún elemento de diálogo, se utiliza el ajuste predeterminado. El ancho de columna para el texto breve y el campo de entrada/salida en el caso estándar se define para cada fila a partir del número de columnas y del ancho máximo de las filas, es decir, ancho de columna = ancho máximo de filas/número de columnas.

El ancho del texto de gráficos y de unidad es fijo y está optimizado para los requisitos de ayuda a la programación. Si se ha configurado el texto de unidad o de gráficos, el ancho del texto breve o del campo de entrada/salida se acorta de la forma correspondiente.

El orden del texto breve y del campo de entrada/salida puede invertirse a través de los datos de posición.

Distancia entre el campo de entrada/salida y el campo de unidades, así como el ancho del campo de unidades

Se puede configurar la distancia entre un campo de entrada/salida y un campo de unidades, así como el ancho del campo de unidades.

Para ello, en la línea de definición, especifique en el apartado para la posición de entrada/salida, con separación por comas, la distancia entre el campo de entrada/salida y el campo de unidades (p. ej., 7 píxeles) o el ancho del campo de unidades (p. ej., 60 píxeles):

```
DEF VarDT=(R3//0.000/,"DT",,"s"////0,,24/39,,71,,7,60)
```

O:

```
DEF VarDT={TYP="R3", VAL="0.000", ST="DT", UT="s", TXT_X=0,
TXT_W=24, X=39, W=71, UT_DX=7, UT_W=60}
```

En este caso (la distancia entre el campo de entrada/salida y el campo de unidades o el ancho del campo de unidades están configurados) deben tenerse en cuenta los siguientes aspectos:

- El ancho configurado del campo de entrada/salida **no** contiene el ancho fijo del campo de unidades (siempre 50 píxeles). Es decir: se configura directamente el ancho del campo de entrada/salida.
- Si no se ha configurado el ancho para el campo de unidades, se aplica el ancho predeterminado de 50 píxeles.
- Si no se ha configurado la distancia entre el campo de entrada/salida y el campo de unidades, se aplica la distancia predeterminada de 0 píxeles.

Peculiaridad para el campo de alternancia asociado

Requisitos:

- Se ha configurado un campo de alternancia asociado para una variable,
- para la variable se ha configurado una distancia entre el campo de entrada/salida y el campo de unidades, o un ancho para el campo de unidades, y
- la variable no tiene texto de unidad.

En este caso, el campo de alternancia asociado se colocará en la posición configurada del campo de unidades de la variable.

Si existe una posición configurada para la parte de entrada/salida del campo de alternancia asociado, se omitirá.

Ejemplo

En el ejemplo siguiente, el campo de alternancia asociado **F_Unit** se posiciona automáticamente con una distancia de **7 píxeles** respecto al campo de entrada/salida de la variable **VarF** y se le asigna un ancho de **59 píxeles**.

```
DEF VarF=(R//0.0/,"F",,,////0,,24/39,,85,,7,59///"F_Unit"),  F_Unit
= (I/*3="mm/min", 1="mm/U"/3// ////181,,155)
```

Consulte también

Parámetros de variables (Página 93)

7.15 Uso de strings

Cadenas de strings

En la configuración también pueden utilizarse strings para diseñar de manera dinámica la visualización de textos o para unir distintos textos para la generación de código.

Reglas

Al emplear variables tipo string deben observarse las siguientes reglas:

- Los vínculos de caracteres se procesan de izquierda a derecha.
- Las expresiones anidadas se deshacen desde dentro hacia fuera.
- Se ignora el uso de mayúsculas y minúsculas.
- Por lo general, las variables tipo string se muestran justificadas a la izquierda.

Los strings pueden eliminarse simplemente mediante la asignación de un string vacío.

Los strings pueden añadirse a la derecha del signo igual mediante el operador "<<". Las comillas dobles (") dentro de un string se identifican mediante dos comillas seguidas. La igualdad de strings puede comprobarse mediante instrucciones IF.

Ejemplo

Valor por defecto para los siguientes ejemplos:

```
VAR1.VAL = "Esto es un"

VAR8.VAL = 4

VAR14.VAL = 15

VAR2.VAL = "error"

$85001 = "Esto es un"

$85002 = "texto de alarma"
```

Edición de strings:

- Combinación de strings:
  ```
  VAR12.VAL = VAR1 << " error."   ;Resultado: "Esto es un error"
  ```
- Borrado de una variable:
  ```
  VAR10.VAL = ""   ;Resultado: cadena vacía
  ```
- Combinación de una variable con una variable de texto:
  ```
  VAR11.VAL = VAR1.VAL   ;Resultado: "Esto es un"
  ```
- Adaptación del tipo de datos:
  ```
  VAR13.VAL ="Éste es el error número " << (VAR14 - VAR8) << ". "
              ;Resultado: "Este es el error número 11"
  ```
- Manejo de valores numéricos:
  ```
  VAR13.VAL = "Error " << VAR14.VAL << ": " << $85001 << $85002
           ;Resultado: "Error 15: Esto es un texto de alarma"
  IF VAR15 == "error"   ;Strings en una instrucción IF
    VAR16 = 18.1234
           ;Resultado: VAR16 igual a 18.1234,
           ;si VAR15 es igual a "error".
  ENDIF
  ```

- Comillas dobles dentro de un string:
```
VAR2="Hola, esto es un " test""
   ;Resultado: Hola, esto es un " test"
```
- Strings de variables de sistema o de usuario dependientes de contenidos de variables:
```
VAR2.Var = "$R[" << VAR8 << "]"   ;Resultado: $R[4]
```

Ver también

Funciones STRING (Página 182)

7.16 Variable CURPOS

Descripción

Con la variable CURPOS es posible abrir o manipular la posición del cursor en el campo de entrada activo del diálogo actual. La variable muestra cuántos caracteres se encuentran por delante del cursor. Si el cursor está posicionado al comienzo del campo de entrada CURPOS toma el valor 0. Si se efectúa una modificación en el valor de CURPOS, el cursor se coloca en la posición correspondiente en el campo de entrada.

Para poder reaccionar a las modificaciones en el valor de la variable, se pueden vigilar sus cambios con ayuda de un método CHANGE. Si cambia el valor de CURPOS, se salta al método CHANGE y se ejecutan las instrucciones contenidas en él.

7.17 Variable CURVER

Descripción

La propiedad CURVER (Current Version) permite adaptar la programación para el manejo de las versiones más diversas. La variable CURVER sólo puede ser leída.

Nota

Durante la generación de código se crea automáticamente con la versión más actual, incluso si previamente se ha decompilado con una versión más antigua. El comando "GC" genera siempre la versión más actual. Para versiones > 0, en el comentario del código generado se inserta un identificador adicional de la versión generada.

Reglas

Siempre debe verse el diálogo más reciente con todas sus variables.

- Las variables previas no deben modificarse.
- Las nuevas variables se insertan en cualquier orden en la programación (de ciclos) previa.

- No está permitido retirar variables de un diálogo de una versión a la siguiente.
- El diálogo debe contener todas las variables de todas las versiones.

Ejemplo

```
(IF CURVER==1 ...)                ; CURVER se activa automáticamente al decompilar
                                    con la versión del código decompilado.
```

7.18 Variable ENTRY

Descripción

Con la variable ENTRY se puede comprobar la forma en que se llamó al diálogo.

Programación

Sintaxis:	**ENTRY**	
Descripción:	La variable ENTRY sólo puede ser leída.	
Valor de retorno:		El resultado de la consulta puede ser:
	0 =	Sin ayuda a la programación
	1 =	Abre una máscara mediante pulsador de menú; no se ha generado aún ningún código (valor por defecto, tal como se ha configurado)
	2 =	Abre una máscara mediante pulsador de menú, se ha generado código (valor por defecto tomado del último código generado por esta máscara)
	3 =	Decompilación con comentarios (n.º de líneas)
	4 =	Code_typ = 0 : Código de CN con comentarios (n.º de líneas)
	5 =	Code_typ = 1 : Código de CN sin comentarios (n.º de líneas)

Ejemplo

```
IF ENTRY == 0
  DLGL("El diálogo no ha sido llamado en la programación")
ELSE
  DLGL("El diálogo ha sido llamado en la programación")
ENDIF
```

7.19 Variable ERR

Descripción

Con la variable ERR puede comprobarse si las líneas precedentes se han ejecutado sin errores en cada caso.

Programación

Sintaxis:	**ERR**	
Descripción:	La variable ERR sólo puede ser leída.	
Valor de retorno:		El resultado de la consulta puede ser:
	FALSE =	la línea precedente se ha ejecutado sin errores,
	TRUE =	la línea precedente se ha ejecutado con errores,

Ejemplo

```
VAR4 = Rosca[VAR1,"DIA",3]              ; Mostrar valor desde matriz
IF ERR == TRUE                          ; Consulta si el valor se ha encontrado en la
                                        matriz
  VAR5 = "Error en el acceso a la ma-
triz"
                                        ; Si el valor no se ha encontrado en la ma-
                                        triz, se le asigna a la variable el valor
                                        "Error en el acceso a la matriz".
ELSE
  VAR5 = "Todo OK"                      ; Si el valor se ha encontrado en la matriz,
                                        se le asigna a la variable el valor "Todo
                                        OK".
ENDIF
```

7.20 Variable FILE_ERR

Descripción

La variable FILE_ERR permite comprobar si el comando GC o CP previo ha sido ejecutado sin errores.

Programación

Sintaxis:	**FILE_ERR**
Descripción:	La variable FILE_ERR sólo puede ser leída.

Valor de retorno:		Los posibles resultados son:
	0 =	Operación en orden
	1 =	Unidad/ruta no existe
	2 =	Error de acceso a ruta/fichero
	3 =	Unidad no preparada
	4 =	Nombre de fichero incorrecto
	5 =	Fichero ya está abierto
	6 =	Acceso denegado
	7 =	Ruta de destino no existe o no permitida
	8 =	Fuente de copia idéntica al destino
	10 =	Error interno: con FILE_ERR = 10 se trata de un error que no se puede asignar a las demás categorías.

Ejemplo

```
CP("D:\source.mpf","E:\target.mpf")
                                          ; Copiar de source.mpf a E:\tar-
                                          get.mpf
IF FILE_ERR > 0                           ; Consultar si se ha producido un
                                          error
   IF FILE_ERR == 1                       ; Consultar determinados números de
                                          error y emitir el correspondiente
                                          texto de error
      VAR5 = "Unidad/ruta no existe"
ELSE
      IF FILE_ERR == 2
        VAR5 = "Error de acceso a ruta/fichero"
ELSE
         IF FILE_ERR == 3
            VAR5 = "Nombre de fichero erróneo"
         ENDIF
ENDIF
   ENDIF
ELSE
  VAR5 = "Todo OK"                        ; Si no se ha producido ningún error
                                          en CP
                                           (o GC), emisión de "Todo OK"
ENDIF
```

7.21 Variable FOC

Descripción

Con la variable FOC se controla el foco de entrada (campo de entrada/salida actualmente activo) en un diálogo. La reacción del cursor a la izquierda, a la derecha, arriba, abajo así como PGUP y PGDN está predefinida de forma fija.

Nota

FOC no debe dispararse con un evento de navegación. La posición del cursor solo puede modificarse en los métodos PRESS, métodos CHANGE, etc. de los pulsadores de menú.

Las variables con el modo de entrada wr = 0 y wr = 4 y las variables auxiliares no se pueden focalizar.

Programación

Sintaxis:	**FOC**	
Descripción:	La variable se puede leer y escribir.	
Valor de retorno:	Leer	Se devuelve como resultado el nombre de la variable focalizada.
	Escribir	Se puede asignar un string o un valor numérico. Un string se interpreta como un nombre de variable y un valor numérico se interpreta como un índice de variable.

Ejemplo

```
IF FOC == "Var1"              ; Lectura del foco
  REG[1] = Var1
ELSE
  REG[1] = Var2
ENDIF

FOC = "Var1"                  ; Asignación de la variable 1 al foco de entrada.
FOC = 3                       ; Asignación del 3.er elemento de diálogo al foco de
                              entrada con WR ≥ 2.
```

Consulte también

FOCUS (Página 125)

7.22 Variable S_ALEVEL

Descripción

El nivel de acceso actual puede consultarse con la propiedad de máscara S_ALEVEL en la configuración.

Programación

Sintaxis:	**S_ALEVEL**
Descripción:	Consulta del nivel de acceso actual
Valor de retorno:	0: Sistema
	1: Fabricante
	2: Servicio técnico
	3: Usuario
	4: Interruptor de llave 3
	5: Interruptor de llave 2
	6: Interruptor de llave 1
	7: Interruptor de llave 0

Ejemplo

```
REG[0] = S_ALEVEL
```

Consulte también

ACCESSLEVEL (Página 122)

7.23 Variable S_CHAN

Descripción

Con la variables S_CHAN puede determinarse el número del canal actual para fines de visualización o para una evaluación.

Programación

Sintaxis:	**S_CHAN**
Descripción:	Consulta del número de canal actual
Valor de retorno:	Número de canal

Ejemplo

```
REG[0] = S_CHAN
```

Consulte también

CHANNEL (Página 124)

7.24 Variable S_CONTROL

Descripción

El nombre de control actual puede consultarse con la propiedad de máscara S_CONTROL en la configuración.

Programación

Sintaxis:	**S_CONTROL**
Descripción:	Consulta del nombre de control actual
Valor de retorno:	El nombre de control es el nombre de sección de "mmc.ini"

Ejemplo

```
REG[0] = S_CONTROL
```

Consulte también

CONTROL (Página 124)

7.25 Variable S_LANG

Descripción

El idioma actual puede consultarse en la configuración con la propiedad de máscara S_LANG.

Programación

Sintaxis:	**S_LANG**
Descripción:	Consulta del idioma actual
Valor de retorno:	Código de idioma de resources.xml, p. ej. "deu", "eng", etc.

Ejemplo

```
REG[0] = S_LANG
```

Consulte también

LANGUAGE (Página 125)

7.26 Variable S_NCCODEREADONLY

Descripción

La propiedad de máscara S_NCCODEREADONLY es significativa exclusivamente cuando en el editor se decompila un ciclo con un cuadro de diálogo "Run MyScreens". Con S_NCCODEREADONLY se determina si puede modificarse o no un código de CN decompilado del editor.

Para el valor de retorno se aplica lo siguiente

- TRUE: El código de CN decompilado del editor se puede modificar
- FALSE: El código de CN decompilado del editor no se puede modificar (readonly), porque p. ej. ya está en decodificación previa (= TRUE).

7.27 Variables S_RESX y S_RESY

Descripción

La resolución actual o sus componentes X e Y pueden consultarse en la configuración con las propiedades de máscara S_RESX y S_RESY.

Ejemplo

```
REG[0] = S_RESY
```

Consulte también

RESOLUTION (Página 130)

Comandos de programación 8

8.1 Operadores

Vista general

En la programación se pueden utilizar los siguientes operadores:

- Operadores matemáticos
- Operadores de comparación
- Operadores lógicos (booleanos)
- Operadores de bits
- Funciones trigonométricas

8.1.1 Operadores matemáticos

Vista general

Operadores matemáticos	Explicación
+	Suma
-	Resta
*	Multiplicación
/	División
MOD	Operación de módulo
( )	Paréntesis
AND	Operador AND
OR	Operador OR
NOT	Operador NOT
ROUND	Redondear números con decimales

Ejemplo: `VAR1.VAL = 45 * (4 + 3)`

ROUND

El operador ROUND se utiliza para redondear números hasta con 12 decimales durante la ejecución de un cuadro de diálogo. Los dígitos no se pueden tomar de los campos de variables a la visualización.

Utilización

ROUND es controlado por el usuario mediante dos parámetros:

```
VAR1 = 5,2328543
VAR2 = ROUND( VAR1, 4 )
```

Resultado: `VAR2 = 5,2339`

VAR1 contiene el número a redondear. El parámetro "4" indica el número de decimales del resultado, que se guarda en VAR2.

Funciones trigonométricas

Funciones trigonométricas	Explicación
SIN(x)	Seno de x
COS(x)	Coseno de x
TAN(x)	Tangente de x
ATAN(x, y)	Arcotangente de x/y
SQRT(x)	Raíz cuadrada de x
ABS(x)	Valor absoluto de x
SDEG(x)	Conversión a grados
SRAD(x)	Conversión a radianes
CALC_ASIN(x)	Arcoseno de x
CALC_ACOS(x)	Arcocoseno de x

Nota

Las funciones trabajan con medidas de arco. Para la conversión pueden emplearse las funciones SDEG() y SRAD().

Ejemplo: `VAR1.VAL = SQRT(2)`

Funciones matemáticas

Funciones matemáticas	Explicación
CALC_CEIL(x)	Calcula el número entero inmediatamente superior a x (redondeo al alza)
CALC_FLOOR(x)	Calcula el número entero inmediatamente inferior a x (redondeo a la baja)
CALC_LOG(x)	Calcula el logaritmo (natural) de x en base e
CALC_LOG10(x)	Calcula el logaritmo de x en base 10
CALC_POW(x, y)	Calcula x elevado a y (potencia y de x)
CALC_MIN(x, y)	Calcula el número menor entre x e y
CALC_MAX(x, y)	Calcula el número mayor entre x e y

Función Random: número aleatorio

Sintaxis:	**RANDOM** (valor límite inferior, valor límite superior)	
Descripción:	La función **RANDOM** devuelve un número seudoaleatorio en un rango predefinido.	
Parámetro:	Límite inferior	Valor límite inferior >= -32767, Valor límite inferior < valor límite superior
	Límite superior	Valor límite superior <= 32767

Ejemplo

```
REG[0] = RANDOM(-10,10)         ; Posible resultado = -3
```

Constantes

Constantes	
PI	3,14159265358979323846
FALSE	0
TRUE	1

Ejemplo: `VAR1.VAL = PI`

Operadores de comparación

Operadores de comparación	
==	Igual que
<>	distinto
>	mayor
<	menor
>=	mayor o igual
<=	menor o igual

Ejemplo:

```
IF VAR1.VAL == 1
  VAR2.VAL = TRUE
ENDIF
```

Condiciones

La profundidad de anidamiento es ilimitada.

Condición con un comando:

```
IF
...
ENDIF
```

Condición con dos comandos:

```
IF
...
ELSE
...
ENDIF
```

8.1.2 Operadores de bits

Vista general

Operadores de bits	Explicación
BOR	OR bit a bit
BXOR	XOR bit a bit
BAND	AND bit a bit
BNOT	NOT bit a bit
SHL	Desplazar los bits a la izquierda
SHR	Desplazar los bits a la derecha

Operador SHL

Con el operador SHL (SHIFT LEFT) se desplazan bits a la izquierda. Se puede indicar tanto el valor que se debe desplazar como el número de pasos para el desplazamiento, de manera directa o como variable. Cuando se alcanza el límite del formato de datos los bits se desplazan sobrepasándolo sin mensaje de error.

Utilización

Sintaxis:	variable = valor **SHL**número de pasos	
Descripción:	Desplazar izquierda	
Parámetro:	valor	Valor que se debe desplazar
	número de pasos	Número de pasos que se debe desplazar

Ejemplo

```
PRESS(VS1)
```

```
  VAR01 = 16 SHL 2           ; Resultado = 64
  VAR02 = VAR02 SHL VAR04    ; El contenido de VAR02 se transforma en un valor de 32
                             bits sin signo y se desplaza a la izquierda en un número
                             de bits equivalente al contenido de VAR04. A continua-
                             ción se vuelve a transformar el valor de 32 bits al for-
                             mato de la variable VAR02.
END_PRESS
```

Operador SHR

Con el operador SHR (SHIFT RIGHT) se desplazan bits a la derecha. Se puede indicar tanto el valor que se debe desplazar como el número de pasos para el desplazamiento, de manera directa o como variable. Cuando se alcanza el límite del formato de datos los bits se desplazan sobrepasándolo sin mensaje de error.

Utilización

Sintaxis:	variable = valor **SHR**número de pasos		
Descripción:	Desplazar derecha		
Parámetro:	Valor	Valor que se debe desplazar	
	Número de pasos	Número de pasos que se debe desplazar	

Ejemplo

```
PRESS(VS1)
  VAR01 = 16 SHR 2           ; Resultado = 4
  VAR02 = VAR02 SHR VAR04    ; El contenido de VAR02 se transforma en un valor
                             de 32 bits sin signo y se desplaza a la derecha en
                             un número de bits equivalente al contenido de
                             VAR04. A continuación se vuelve a transformar el
                             valor de 32 bits al formato de la variable VAR02.
END_PRESS
```

8.2 Métodos

Vista general

En los diálogos y en los menús de pulsadores dependientes de ellos (menús de pulsadores que se abren desde un diálogo recién configurado) se pueden iniciar determinadas acciones de forma controlada mediante distintos eventos (salida del campo de entrada, operación de los pulsadores de menú). Estas acciones se configuran en métodos.

La programación básica de un método tiene el siguiente aspecto:

Bloque descriptivo	Comentario	Referencia cruzada a capítulos
`PRESS(HS1)`	;Identificador de arranque del método	
`LM...` `  LS...`	;Funciones	Ver capítulo Funciones (Página 134)
`Var1.st = ...`	;Modificación de propiedades	Ver capítulos Definir menús de pulsadores (Página 65) y Definir propiedades del diálogo (Página 53).

`Var2 = Var3 + Var4`
`...`
`EXIT`
	;Cálculo con variables	Ver capítulo Definir variables (Página 85)

`END_PRESS`	;Identificador de fin del método	

8.2.1 ACCESSLEVEL

Descripción

El método ACCESSLEVEL se ejecuta cuando se modifica el nivel de acceso actual con la máscara abierta.

Programación

Sintaxis:	ACCESSLEVEL <Instrucciones> END_ACCESSLEVEL
Descripción:	Nivel de acceso
Parámetros:	- Ninguno -

Consulte también

Variable S_ALEVEL (Página 113)

8.2.2 CHANGE

Descripción

Los métodos CHANGE se ejecutan cuando se modifica el valor de una variable. Es decir, dentro de un método CHANGE se configuran cálculos de variables que se ejecutan inmediatamente cuando tiene lugar una modificación en una variable.

Los métodos CHANGE se diferencian en específico del elemento y global:

- El **método CHANGE específico del elemento** se ejecuta cuando se modifica el valor de la variable especificada. Si una variable tiene asignada una variable de sistema o de usuario, su valor puede actualizarse cíclicamente en un método CHANGE.
- El **método CHANGE global** se ejecuta cuando se modifica el valor de cualquier variable y no se ha configurado ningún método CHANGE específico del elemento.

Programación "específica del elemento"

Sintaxis:	CHANGE(Identificador) ... END_CHANGE	
Descripción:	Modificación del valor de la variable especificada	
Parámetro:	Identificador	Nombre de la variable

Ejemplo

```
DEF VAR1=(I//////"DB20.DBB1")          ; A Var1 se asigna una variable de sistema
CHANGE(VAR1)
  IF VAR1.Val <> 1
  VAR1.st="¡Herramienta OK!"           ; Si el valor de la variable de sistema es ≠ 1,
                                       el texto breve de la variable es: ¡Herramienta
                                       OK!
    otto=1
  ELSE
    VAR1.st="Atención: error"          ; Si el valor de la variable de sistema es = 1,
                                       el texto breve de la variable es: Atención:
                                       error
    otto=2
  ENDIF
  VAR2.Var=2
END_CHANGE
```

Programación "global"

Sintaxis:	CHANGE() ... END_CHANGE
Descripción:	Modificación de cualquier valor de variable
Parámetro:	- Ninguno -

Ejemplo

```
CHANGE()
  EXIT                                ; Si se modifica cualquier valor de variable,
                                      se abandona el diálogo.
END_CHANGE
```

8.2.3 CHANNEL

Descripción

El método CHANNEL se ejecuta cuando se modifica el canal actual con la máscara abierta; es decir, si se ha producido un cambio de canal.

Programación

Sintaxis:	CHANNEL <Instrucciones> END_CHANNEL
Descripción:	Conmutación de canal
Parámetros:	- Ninguno -

Consulte también

Variable S_CHAN (Página 113)

8.2.4 CONTROL

Descripción

El método CONTROL se ejecuta cuando se ha modificado el control actual con la máscara abierta; es decir, normalmente con un cambio 1:n.

Programación

Sintaxis:	CONTROL <Instrucciones> END_CONTROL
Descripción:	Cambio de control
Parámetros:	- Ninguno -

Consulte también

Variable S_CONTROL (Página 114)

8.2.5 FOCUS

Descripción

El método FOCUS se ejecuta cuando en el diálogo el foco (cursor) se posiciona sobre otro campo.

El método FOCUS no debe dispararse con un evento de navegación. La posición del cursor solo puede modificarse en los métodos PRESS, métodos CHANGE, etc. de los pulsadores de menú. La reacción de los movimientos del cursor está predefinida de forma fija.

Nota

Dentro del método FOCUS no es posible posicionarse sobre otra variable ni se puede cargar un cuadro de diálogo nuevo.

Programación

Sintaxis:	FOCUS ... END_FOCUS
Descripción:	Posicionar el cursor
Parámetro:	- Ninguno -

Ejemplo

```
FOCUS
  DLGL("El foco se ha colocado sobre la variable"<< FOC <<".")
END_FOCUS
```

Consulte también

Variable FOC (Página 112)

8.2.6 LANGUAGE

Descripción

El método LANGUAGE se ejecuta cuando se modifica el idioma actual con la máscara abierta.

Programación

Sintaxis:	LANGUAGE <Instrucciones> END_LANGUAGE
Descripción:	Idioma
Parámetros:	- Ninguno -

Consulte también

Variable S_LANG (Página 114)

8.2.7 LOAD

Descripción

El método LOAD se ejecuta una vez interpretadas las definiciones de las variables y los pulsadores de menú (DEF Var1= ..., HS1= ...). En este momento, el diálogo aún no se muestra.

Programación

Sintaxis:	LOAD ... END_LOAD
Descripción:	Cargar
Parámetro:	- Ninguno -

Ejemplo

```
LOAD                          ; Identificador de arranque
  Máscara1.Hd =  $85111       ; Asignar el texto para el título del diálogo del fiche-
                                ro de idioma
  VAR1.Min = 0                ; Asignar variable Valor límite MÍN
  VAR1.Máx = 1000             ; Asignar variable Valor límite MÁX
END_LOAD                      ; Identificador de fin
```

Consulte también

Definir elementos gráficos (Página 193)

8.2.8 UNLOAD

Descripción

El método UNLOAD se ejecuta previamente a la descarga de un diálogo.

Programación

Sintaxis:	UNLOAD ... END_UNLOAD
Descripción:	Descargar
Parámetro:	- Ninguno -

Ejemplo

```
UNLOAD
  REG[1] = VAR1            ; Guardar variable en el registro
END_UNLOAD
```

8.2.9 OUTPUT

Descripción

El método OUTPUT se ejecuta cuando se llama la función "GC". Dentro de un método OUTPUT se configuran las variables y las variables auxiliares como código de CN. La concatenación de elementos individuales de una línea de código se lleva a cabo con espacios.

Nota

El código de CN puede generarse en un fichero suplementario con las funciones de fichero y desplazarse hasta el CN.

Programación

Sintaxis:	OUTPUT(Identificador) ... END_OUTPUT	
Descripción:	Emitir variables en el programa de CN	
Parámetro:	Identificador	Nombre del método OUTPUT

Números de secuencia y códigos de supresión

El método OUTPUT no debe contener ningún número de línea ni código de supresión cuando los números de línea y códigos de supresión definidos directamente en el programa de pieza durante la ayuda activa a la programación deban conservar su valor al decompilar.

Las modificaciones con el editor en el programa de piezas dan lugar a los siguientes comportamientos:

Condición	Comportamiento
La cantidad de secuencias no varía.	Los números de secuencia conservan su valor.
La cantidad de secuencias disminuye.	Se suprimen los números de secuencia más altos.
La cantidad de secuencias aumenta.	Las nuevas secuencias no reciben ningún número de secuencia.

Ejemplo

```
OUTPUT(CODE1)
  "CYCLE82(" Var1.val "," Var2.val "," Var3.val ","Var4.val "," Var5.val "," Var6.val ")"
END_OUTPUT
```

8.2.10 PRESS

Descripción

El método PRESS se ejecuta cuando se ha presionado el correspondiente pulsador de menú.

Programación

Sintaxis:	PRESS(Pulsador de menú) ... END_PRESS
Identificación:	Accionar un pulsador de menú

Parámetro:	Tecla de función programable	Nombre del pulsador de menú: HS1 - HS8 y VS1 - VS8	
	RECALL	Tecla <RECALL>	
	ENTER	Tecla <ENTER>, ver PRESS(ENTER) (Página 129)	
	TOGGLE	Tecla <TOGGLE>, ver PRESS(TOGGLE) (Página 130)	
	PU	Re Pág	avance página
	PD	Av Pág	retroceso página
	SL	Scroll Left	cursor a la izquierda
	SR	Scroll Right	cursor a la derecha
	SU	Scroll Up	cursor hacia arriba
	SD	Scroll Down	cursor hacia abajo

Ejemplo

```
HS1 = ("otro menú de pulsadores")
HS2 = ("sin función")
PRESS (HS1)
  LS("Menú1")                    ; Cargar otro menú de pulsadores
  Var2 = Var3 + Var1
END_PRESS
PRESS (HS2)
END_PRESS
PRESS(PU)
  INDEX = INDEX -7
  CALL("UP1")
END_PRESS
```

8.2.11 PRESS(ENTER)

Descripción

Se accede al método PRESS(ENTER) siempre que se pulsa la tecla Intro cuando haya una variable con campo de entrada/salida de modo de entrada WR3 o WR5.

- WR3: navegación en el campo y accionamiento de la tecla Intro
- WR5: en modo de entrada, adopción del valor con la tecla Intro

Programación

Sintaxis:	PRESS(ENTER) <Instrucciones> END_PRESS
Descripción:	Tecla Intro pulsada
Parámetros:	- Ninguno -

8.2.12 PRESS(TOGGLE)

Descripción

Se accede al método PRESS(TOGGLE) siempre que se acciona la tecla de alternancia, independientemente de la variable enfocada en ese momento.

Si es necesario, con ayuda de la propiedad de máscara FOC puede determinarse la variable que tiene el foco actualmente.

Programación

Sintaxis:	PRESS(TOGGLE) <Instrucciones> END_PRESS
Descripción:	Tecla de alternancia accionada
Parámetros:	- Ninguno -

Ejemplo

```
PRESS(TOGGLE)
   DLGL("Toggle key pressed at variable " << FOC)        ; Con la propiedad de máscara FOC se determi-
                                                         na la variable que tiene el foco actualmente
END_PRESS
```

8.2.13 RESOLUTION

Descripción

El método RESOLUTION se ejecuta cuando se ha modificado la resolución actual con la máscara abierta; es decir, normalmente con un cambio de TCU.

Programación

Sintaxis:	RESOLUTION <Instrucciones> END_RESOLUTION
Descripción:	Resolución
Parámetros:	- Ninguno -

Consulte también

Variables S_RESX y S_RESY (Página 115)

8.2.14 RESUME

Descripción

Se accede al método RESUME cuando la máscara se ha interrumpido, p. ej., por un cambio de área, y luego vuelve a mostrarse. Al hacerlo, vuelven a procesarse los cambios de los valores, dado el caso se inicia el temporizador y se ejecuta el bloque RESUME.

Nota

Las funciones LS, LM, DLGL, EXIT y EXITLS no deben configurarse en los métodos SUSPEND o RESUME. Si se usan esas funciones en estos métodos, se impide la ejecución de las funciones.

Programación

Sintaxis:	RESUME <Instrucción> END_RESUME
Descripción:	Máscara activa de nuevo
Parámetros:	- Ninguno -

8.2.15 SUSPEND

Descripción

Se accede al método SUSPEND cuando la máscara se interrumpe y no se descarga. Lo mismo ocurre, p. ej., cuando se sale de la máscara con un sencillo cambio de área, pero no se descarga de forma explícita. La máscara permanece entonces en segundo plano, pero no se ejecutan cambios de valores ni el temporizador, dado el caso. La máscara se detiene.

Nota

Las funciones LS, LM, DLGL, EXIT y EXITLS no deben configurarse en los métodos SUSPEND o RESUME. Si se usan esas funciones en estos métodos, se impide la ejecución de las funciones.

Programación

Sintaxis:	SUSPEND <Instrucción> END_SUSPEND
Descripción:	Interrupción de la máscara
Parámetros:	- Ninguno -

Ejemplo

```
SUSPEND
   MyVar1 = MyVar1 + 1
END_SUSPEND
```

8.2.16 Ejemplo: gestión de versiones con métodos OUTPUT

Vista general

Los diálogos existentes pueden completarse con variables adicionales en el curso de las ampliaciones. En las definiciones, tras el nombre de la variable y entre paréntesis, las variables adicionales contienen un número característico de la versión: (0 = original, no se escribe), 1 = versión 1,
2 = versión 2, ...

Ejemplo:

```
DEF var100=(R//1)                ; Original, corresponde a la versión 0
DEF var101(1)=(S//"Hallo")       ; Complemento a partir de la versión 1
```

Al escribir el método OUTPUT puede hacerse referencia a una versión concreta, referida a la totalidad de las definiciones.

Ejemplo:

```
OUTPUT(NC1)                    ; Solo las variables de la versión original se ofrecen en el método OUTPUT.
OUTPUT(NC1,1)                  ; Las variables de la versión original y los complementos con el identificador de versión 1 se ofrecen en el método OUTPUT.
```

El método OUTPUT para la versión original no necesita ningún identificador de versión; sin embargo, puede escribirse también un 0. OUTPUT(NC1) equivale a OUTPUT(NC1,0). El identificador de versión n del método OUTPUT comprende todas las variables del original 0, 1, 2,... hasta n inclusive.

Programación con indicador de versión

```
//M(XXX)                       Versión 0 (por defecto)
DEF var100=(R//1)
DEF var101=(S//"Hallo")
DEF TMP
VS8=("GC")
PRESS(VS8)
   GC("NC1")
END_PRESS

OUTPUT(NC1)
var100",,"var101
END_OUTPUT

; ************ Versión 1, definición completada ***************
//M(XXX)
DEF var100=(R//1)
DEF var101=(S//"Hallo")
DEF var102(1)=(V//"HUGO")
DEF TMP
VS8=("GC")
PRESS(VS8)
   GC("NC1")
END_PRESS
...

OUTPUT(NC1)                    Original y adicionalmente la nueva versión
var100","var101
END_OUTPUT
```

```
...

OUTPUT(NC1,1)                    Versión 1
var100","var101"," var102
END_OUTPUT
```

8.3 Funciones

Vista general

En diálogos y menús de pulsadores dependientes de ellas se dispone de diversas funciones que se pueden disparar a través de distintos eventos (p. ej.: salida del campo de entrada, operación de los pulsadores de menú) y se configuran en métodos.

Subprogramas

Las instrucciones de configuración que se repitan y también otras que resuman un determinado proceso, pueden configurarse en subprogramas. Los subprogramas pueden cargarse en cualquier momento en el programa principal o en otros subprogramas y ejecutarse tantas veces como se desee, es decir, no es necesario configurar repetidas veces las instrucciones. Como programa principal se consideran los bloques descriptivos de los diálogos o de los menús de pulsadores.

Funciones externas

Con ayuda de las funciones externas pueden introducirse otras funciones personalizadas. Las funciones externas se almacenan en un fichero DLL y se dan a conocer mediante una entrada en la línea de definición del fichero de configuración.

Servicios de PI

Con la función PI_START es posible iniciar servicios de PI (servicios de instancia de programa) desde el PLC en el área de CN.

Ver también

Function (FCT) (Página 155)

Servicios de PI (Página 170)

8.3.1 Leer y escribir parámetros de accionamiento: RDOP, WDOP, MRDOP

Descripción

Con ayuda de las funciones RDOP, WDOP y MRDOP, se pueden leer y escribir parámetros de accionamiento.

Nota

Los parámetros de accionamiento no deben leerse en ciclos de menos de 1 segundo, es mejor que se lean más despacio.

Motivo: De lo contrario, la comunicación con los accionamientos podría perturbarse en exceso o incluso provocar fallos.

Nota

Si se producen errores al leer o escribir parámetros de accionamiento, la propiedad de máscara ERR se define de forma correspondiente.

Programación

Sintaxis:	**RDOP**("Identificador del objeto de accionamiento", "Número de parámetro")	
Descripción:	Lectura de un parámetro de accionamiento (Drive Object Parameter)	
Parámetros:	Identificador del objeto de accionamiento	El identificador del objeto de accionamiento puede tomarse, p. ej., de la columna "Nombre DO" en "Diagnóstico del sistema de accionamiento" en el campo de manejo Diagnóstico (> ETC > HSK 8: sistema de accionamiento) (ver la figura siguiente).
	Número de parámetro	

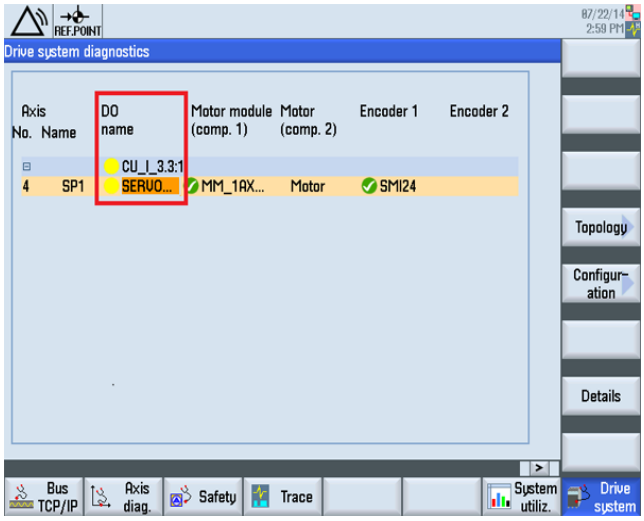


Figura 8-1 Identificador del objeto de accionamiento

Sintaxis:	**MRDOP**("identificador del objeto de accionamiento", "número de parámetro1"*"número de parámetro"[*...], índice del registro)	
Descripción:	Multi Read de parámetros de accionamiento. Con el comando MRDOP pueden transferirse varios parámetros de accionamiento de un objeto de accionamiento con un acceso al registro. Este acceso es considerablemente más rápido que la lectura mediante acceso individual.	
Parámetros:	Identificador del objeto de accionamiento	El identificador del objeto de accionamiento puede tomarse, p. ej., de la pantalla básica del campo de manejo "Puesta en marcha"
	Número de parámetro 1 ... n	Para los nombres de variables se utiliza "*" como carácter de separación. Los valores se transfieren al registro REG[índice del registro] según el orden con el que se suceden los nombres de las variables en el comando.
	Índice del registro	El valor de la primera variable se encuentra en REG[índice del registro]. El valor de la segunda variable se encuentra en REG[índice del registro + 1]

Sintaxis:	**WDOP**("identificador del objeto de accionamiento", "número de parámetro", valor)
Descripción:	Escritura de un parámetro de accionamiento (Drive Object Parameter)

Parámetros:	Identificador del objeto de accionamiento	El identificador del objeto de accionamiento puede tomarse, p. ej., de la pantalla básica del campo de manejo "Puesta en marcha"
	Número de parámetro	Número de parámetro
	Valor	Valor que se debe escribir

Ejemplos

Lectura de la temperatura del motor r0035 del objeto de accionamiento "SERVO_3.3:2":

```
MyVar=RDOP("SERVO_3.3:2","35") ;
```

Lectura de la temperatura del motor r0035 y el par real r0080 del objeto de accionamiento "SERVO_3.3:2" y almacenamiento de los resultados correspondientes a partir del índice del registro 10:

```
MRDOP("SERVO_3.3:2","35*80",10)
```

8.3.2 Llamada de un subprograma (CALL)

Descripción

Con la función CALL puede llamarse un subprograma cargado desde cualquier lugar de un método. El anidamiento, es decir, la llamada de un subprograma desde otro subprograma, está permitido.

Programación

Sintaxis:	**CALL**("Identificador")	
Descripción:	Llamada de subprograma	
Parámetro:	Identificador	Nombre del subprograma

Ejemplo

```
//M(MÁSCARA1)
DEF VAR1 = ...
DEF VAR2 = ...
CHANGE(VAR1)
  ...
  CALL("MY_UP1")              ; Llamar y ejecutar el subprograma
  ...
END_CHANGE

CHANGE(VAR2)
```

```
//M(MÁSCARA1)
  ...
  CALL("MY_UP1")              ; Llamar y ejecutar el subprograma
  ...
END_CHANGE

SUB(MY_UP1)
                              ;do something
END_SUB
//END
```

8.3.3 Definir bloque (//B)

Descripción

Los subprogramas se identifican en el fichero de programa con el identificador de bloque //B y se finalizan con //END. Por cada identificador de bloque pueden definirse múltiples subprogramas.

Nota

Las variables utilizadas en el subprograma tienen que estar definidas en el diálogo en el cual se llama al subprograma.

Programación

Un bloque tiene la siguiente estructura:

Sintaxis:	**//B**(Nombre de bloque) **SUB**(Identificador) END_SUB [SUB(Identificador) ... END_SUB] ... //END	
Descripción:	Definición de un subprograma	
Parámetro:	Nombre de bloque	Nombre del identificador de bloque
	Identificador	Nombre del subprograma

Ejemplo

```
//B(PROG1)                    ; Comienzo del bloque
```

```
SUB(UP1)                          ; Inicio de subprograma
  ...
  REG[0] = 5                      ; Asignar al registro 0 el valor 5
  ...
END_SUB                           ; Final de subprograma
SUB(UP2)                          ; Inicio de subprograma
  IF VAR1.val=="Otto"
    VAR1.val="Hans"
    RETURN
  ENDIF
  VAR1.val="Otto"
END_SUB                           ; Final de subprograma
//END                             ; Fin del bloque
```

8.3.4 Check Variable (CVAR)

Descripción

Con ayuda de la función CVAR (Check Variable) puede consultarse si todas o sólo determinadas variables o variables auxiliares de un diálogo están libres de errores.

Es conveniente consultar la validez del valor de las variables, p. ej., antes de generar código de CN mediante la función GC.

Una variable está libre de errores cuando el estado de la variable es Identificador.vld = 1.

Programación

Sintaxis:	**CVAR**(VarN)	
Descripción:	Comprobar la validez del contenido de las variables	
Parámetro:	VarN	Listado de las variables que se desean comprobar. Se pueden comprobar hasta 29 variables separadas por comas. Se tiene que observar la secuencia máxima de caracteres de 500. El resultado de la consulta puede ser:
	1 =	TRUE (todas las variables tienen contenido válido),
	0 =	FALSE (al menos una variable no tiene contenido válido).

Ejemplo

```
IF CVAR == TRUE                   ; Comprobación de todas las variables
  VS8.SE = 1                      ; Si todas las variables son correctas,
                                  se puede ver el pulsador de menú VS8
```

```
ELSE
  VS8.SE = 2                                ; Si una variable contiene un valor erró-
                                            neo, el pulsador de menú VS8 está desha-
                                            bilitado
ENDIF

IF CVAR("VAR1", "VAR2") == TRUE
                                            ; Comprobación de las variables VAR1 y
                                            VAR2
  DLGL ("VAR1 y VAR2 son correctas")
                                            ; Si VAR1 y VAR2 se han rellenado sin
                                            errores, aparece en la línea de diálogo
                                            "VAR1 y VAR2 son correctas"

ELSE
  DLGL ("VAR1 y VAR2 no son correctas")
                                            ; Si VAR1 y VAR2 se han rellenado con
                                            errores, aparece en la línea de diálogo
                                            "VAR1 y VAR2 no son correctas"

ENDIF
```

8.3.5 Función de fichero Copy Program (CP)

Descripción

La función CP (Copy Program) copia ficheros dentro del sistema de ficheros del HMI o del sistema de ficheros del CN.

Programación

Sintaxis:	**CP**("Fichero fuente", "Fichero de destino")	
Descripción:	Copiar fichero	
Parámetro:	Fichero fuente	Nombre de ruta completo del fichero fuente
	Fichero de destino	Nombre de ruta completo del fichero de destino

A través del valor de retorno (VAR1 definido como variable auxiliar) se puede consultar si la función tuvo éxito:

```
CP("//NC/MPF.DIR/HOHO.MPF","//NC/MPF.DIR/ASLAN.MPF",VAR1)
```

Ejemplo

Caso de aplicación con valor de retorno:

```
CP("//NC/MPF.DIR/HOHO.MPF","//NC/MPF.DIR/ASLAN.MPF",VAR1)
CP("CF_CARD:/wks.dir/MESS_BILD.WPD/MESS_BILD.MPF","//NC/WKS.DIR/AAA.WPD/HOHO2.MPF",VAR1)
CP("//NC/MPF.DIR/HOHO.MPF","CF_CARD:/wks.dir/WST1.WPD/MESS.MPF",VAR1)        ; WPD debe existir
```

Caso de aplicación sin valor de retorno:

```
CP("//NC/MPF.DIR/HOHO.MPF","//NC/MPF.DIR/ASLAN.MPF")
CP("CF_CARD:/mpf.dir/MYPROG.MPF","//NC/MPF.DIR/HOHO.MPF")
CP("//NC/MPF.DIR/HOHO.MPF","CF_CARD:/XYZ/MYPROG.MPF")                          ; XYZ debe existir
```

Ver también

Ayuda de FILE_ERR: Variable FILE_ERR (Página 110)

8.3.6 Función de fichero Delete Program (DP)

Descripción

La función DP (Delete Program) elimina un fichero del sistema pasivo de ficheros del HMI o del sistema activo de ficheros del CN.

Programación

Sintaxis:	**DP**("Fichero")	
Descripción:	Borrar fichero	
Parámetro:	Fichero	Nombre de ruta completo del fichero que se desea eliminar

Ejemplo

Para esta función se utiliza la siguiente sintaxis de la gestión de datos:

- Con valor de retorno

  ```
  DP("//NC/MPF.DIR/MYPROG.MPF",VAR1)
  DP("//NC/WKS.DIR/TEST.WPD/MYPROG.MPF",VAR1)
  DP("//NC/CMA.DIR/MYPROG.SPF",VAR1)
  ```

  VAR1 = 0 Se ha borrado el fichero.

  VAR1 = 1 No se ha borrado el fichero.
- Sin valor de retorno:

  ```
  DP("//NC/MPF.DIR/MYPROG.MPF")
  ```

```
DP("//NC/WKS.DIR/TEST.WPD/MYPROG.MPF")
DP("//NC/CMA.DIR/MYPROG.SPF")
```

8.3.7 Función de fichero Exist Program (EP)

Descripción

La función EP (Exist Program) comprueba si un determinado programa de CN se encuentra en el sistema de ficheros del CN o del HMI bajo la ruta indicada.

Programación

Sintaxis:	**EP**("Fichero")	
Descripción:	Comprobar la existencia del programa de CN	
Parámetro:	Fichero	Nombre de ruta completo del fichero para el sistema de ficheros de CN o de HMI
Valor de retorno:	Nombre de una variable a la que se debe asignar el resultado de la consulta.	

La función EP domina la nueva sintaxis y la antigua lógica (con la sintaxis adaptada).

Se accede al fichero directamente con un nombre calificativo:

`//NC/MPF.DIR/XYZ.MPF`

o
`CF_CARD: /MPF.DIR/XYZ.MPF` (apunta a /user/sinumerik/data/prog)
o
`LOC:` (corresponde a CF_CARD)

Ejemplos de sintaxis nueva:

```
EP("//NC/WKS.DIR/TEST.WPD/XYZ.MPF",VAR1)
EP("CF_CARD:/mpf.dir/XYZ.MPF",VAR1)
EP("LOC:/mpf.dir/XYZ.MPF",VAR1)
; con valor de retorno:
; VAR1 = 0 El fichero existe.
; VAR1 = 1 El fichero no existe.
```

Ejemplos de sintaxis antigua:

```
EP("/MPF.DIR/CFI.MPF", VAR1)
; con valor de retorno:
```

```
; VAR1 = M El fichero se encuentra en el sistema de ficheros de HMI.
; VAR1 = N El fichero se encuentra en el sistema de ficheros del CN.
; VAR1 = B El fichero se encuentra en el sistema de ficheros del HMI y del CN.
```

Ejemplo

```
EP("/MPF.DIR/GROB.MPF",VAR1)                                    ; antiguo - Ruta ahora con / en lugar de \


IF VAR1 == "M"
   DLGL("El fichero se encuentra en el sistema de ficheros
del HMI")
ELSE
   IF VAR1 == "N"
      DLGL("El fichero se encuentra en el directorio de fi-
cheros del CN")
   ELSE
     DLGL("El fichero no se encuentra en el sistema de fi-
cheros del HMI ni del CN")
   ENDIF
ENDIF
```

8.3.8 Función de fichero Move Program (MP)

Descripción

La función MP (Move Program) copia ficheros dentro del sistema de ficheros del HMI o del sistema de ficheros del CN.

Programación

Sintaxis:	**MP**("Fuente", "Destino")	
	`MP("CF_CARD:/MPF.DIR/MYPROG.MPF","//NC/MPF.DIR")`	
Descripción:	Desplazar fichero	
Parámetro:	Fichero fuente	Nombre de ruta completo
	Fichero de destino	Nombre de ruta completo
Valor de retorno	Resultado de la consulta	

Ejemplos

Con valor de retorno:

```
MP("//NC/MPF.DIR/123.MPF","//NC/MPF.DIR/ASLAN.MPF",VAR1)          ;dentro de CN
MP("//NC/MPF.DIR/123.MPF",VAR0,VAR1)                              ;destino mediante variable
MP(VAR4,VAR0,VAR1)                                                ;fuente y destino mediante variable
MP("CF_CARD:/mpf.dir/myprog.mpf","//NC/MPF.DIR/123.MPF",VAR1)     ;de tarjeta CF a CN
MP("//NC/MPF.DIR/HOHO.MPF","CF_CARD:/XYZ/123.mpf",VAR1)           ;de CN a tarjeta CF

; con valor de retorno:
; VAR1 = 0 ejecutada
; VAR1 = 1 no ejecutada
```

8.3.9 Función de fichero Select Program (SP)

Descripción

La función SP (Select Program) selecciona un fichero del sistema activo de ficheros del CN para ejecutarlo. Es decir, el fichero debe cargarse previamente en el CN.

Programación

Sintaxis:	**SP**("Fichero")	
Identificación:	Seleccionar programa	
Parámetro:	"Fichero"	Nombre de ruta completo del fichero de CN

Ejemplo

Para esta función se utiliza la siguiente sintaxis de la gestión de datos:

- Con valor de retorno

  `SP("//NC/MPF.DIR/MYPROG.MPF", VAR1)`

  VAR1 = 0 Se ha cargado el fichero.

  VAR1 = 1 No se ha cargado el fichero.
- Sin valor de retorno:

  `SP("//NC/MPF.DIR/MYPROG.MPF")`

8.3.10 Acceso a ficheros: RDFILE, WRFILE, RDLINEFILE, WRLINEFILE

Descripción

Para el acceso de lectura y escritura a ficheros con sintaxis INI se encuentran disponibles las funciones RDFILE y WRFILE.

Para el acceso de lectura y escritura a determinadas líneas de un fichero se encuentran disponibles las funciones RDLINEFILE y WRLINEFILE.

Programación

Sintaxis:	**RDFILE**("Nombre de fichero + ruta", "Section", "Key")	
Descripción:	Lectura de un fichero	
Parámetros:	Nombre de fichero + ruta	Ruta y nombre de fichero
	Section	Sección del fichero INI
	Key	Clave del fichero INI

Sintaxis:	**WRFILE**(Value, "nombre de fichero + ruta", "Section", "Key")	
Descripción:	Escritura en un fichero	
Parámetros:	Value	valor que se debe escribir
	Nombre de fichero + ruta	Ruta y nombre de fichero
	Section	Sección del fichero INI
	Key	Clave del fichero INI

Sintaxis:	**RDLINEFILE**("nombre de fichero + ruta", número de línea)	
Descripción:	Lectura de una línea de un fichero	
Parámetros:	Nombre de fichero + ruta	Ruta y nombre de fichero
	Número de línea	Número de línea La numeración empieza por 0 en la primera línea.

Sintaxis:	**WRLINEFILE**(valor, "nombre de fichero + ruta")	
Descripción:	Escritura de una línea al final de fichero	
Parámetros:	Valor	valor que se debe escribir
	Nombre de fichero + ruta	Ruta y nombre de fichero

Nota

- Los ficheros no pueden estar en el sistema de ficheros del CN (gestión de datos).
- Si el fichero no existe, si se llega al final del fichero o si se produce otro error, la variable FILE_ERR y ERR se define de forma correspondiente. Puede comprobarse de antemano si un fichero existe con la función de ficheros Exist Program (EP).
- El fichero procesado se crea con la codificación "UTF-8" (sin BOM, Byte Order Mask). Al leer un fichero se espera que su codificación sea "UTF-8".
- Con la función de ficheros Delete Program (DP) puede borrarse el fichero de forma explícita si es necesario.

Ejemplos

Lectura de un fichero INI:

Requisitos/premisa:

Contenido del fichero "C:/tmp/myfile.ini":

```
<...>
[MyData]
MyName=Daniel
<...>
```

```
MyVar = RDFILE("C:/tmp/myfile.ini", "MyData", "MyName")
```

Resultado:

MyVar contiene ahora el valor "Daniel".

Escritura en un fichero INI:

Requisitos/premisa:

VARS=12

```
WRFILE(VARS, "C:/tmp/myfile.ini", "MySession", "NrOfSessions")
```

Resultado:

Contenido del fichero "C:/tmp/myfile.ini":

```
<...>
[MySession]
NrOfSessions=12
<...>
```

Lectura de la 4.ª línea de un fichero:

Requisitos/premisa:

Contenido de c:/tmp/myfile.mpf:

```
R[0]=0
F3500 G1
MYLOOPIDX1: X0 y-50 Z0
  X150 Y50 Z10
  R[0]=R[0]+1
GOTOB MYLOOPIDX1
M30
```

```
MyVar = RDLINEFILE("C:/tmp/myfile.mpf", 4)
```

Resultado:

MyVar contiene ahora el valor " `R[0]=R[0]+1`"

Escritura al final de un fichero:

Requisitos/premisa:

```
VARX=123
VARY=456
```

```
WRLINEFILE("F100 X" << VARX << " Y" << VARY, "C:/tmp/mypp.mpf")
```

Resultado:

Contenido de c:/tmp/mypp.mpf:

```
<...>
F100 X123 Y456
```

8.3.11 Dialog Line (DLGL)

Descripción

En la línea de diálogo del diálogo pueden emitirse textos breves (mensajes o ayudas para la entrada) en función de determinadas situaciones.

Número de caracteres posible con el tamaño de fuente estándar: aprox. 50 caracteres

Nota

Las funciones LS, LM, DLGL, EXIT y EXITLS no deben configurarse en los métodos SUSPEND o RESUME. Si se usan esas funciones en estos métodos, se impide la ejecución de las funciones.

Programación

Sintaxis:	**DLGL**("String")	
Descripción:	Emitir texto en la línea de diálogo	
Parámetro:	String	Texto que aparece en la línea de diálogo

Ejemplo

```
IF Var1 > Var2
                              ; En la línea de diálogo aparece el texto "Valor dema-
                              siado grande" si variable1 > variable2.
  DLGL("Valor demasiado
grande")
ENDIF
```

8.3.12 DEBUG

Descripción

Con la función DEBUG se encuentra disponible una ayuda para el análisis en la fase de diseño de máscaras de usuario de Run MyScreens. Con la función DEBUG se evalúa en tiempo de ejecución una expresión transferida entre paréntesis. El resultado se añade al libro de incidencias "easyscreen_log.txt" como entrada independiente.

A cada entrada se antepone una etiqueta de fecha/hora actual entre corchetes (ver el ejemplo siguiente).

Se recomienda volver a eliminar en la medida de lo posible las salidas DEBUG en secciones en las que el tiempo es un factor crítico. Especialmente después de finalizar la fase de diseño se recomienda comentar las salidas DEBUG. Los accesos de escritura en el libro de incidencias, cada vez mayor, "easyscreen_log.txt" provocan una ralentización del sistema.

Programación

Sintaxis:	**DEBUG**(Expresión)
Descripción:	Efectuar una entrada en el libro de incidencias "easyscreen_log.txt"
Parámetros:	Expresión que se va a evaluar, a partir de la cual se genera una entrada en el libro de incidencias

Ejemplo

```
IF Var1 > Var2
  DEBUG("Value of ""Var1"":" << Var1)
                                        ; Entrada en el fichero "easyscreen_log_txt:
                                        [10:22:40.445] DEBUG: Value of "Var1":
                                        123,456
ENDIF
```

8.3.13 Abandonar el diálogo (EXIT)

Descripción

Con la función EXIT se abandona un diálogo y se vuelve al diálogo principal. Si no existe ningún diálogo principal, se sale de la interfaz de usuario recién diseñada y se regresa a la aplicación estándar.

Nota

Las funciones LS, LM, DLGL, EXIT y EXITLS no deben configurarse en los métodos SUSPEND o RESUME. Si se usan esas funciones en estos métodos, se impide la ejecución de las funciones.

Programación (sin parámetros)

Sintaxis:	**EXIT**
Descripción:	Abandonar diálogo
Parámetro:	- Ninguno -

Ejemplo

```
PRESS(HS1)
  EXIT
END_PRESS
```

Descripción

Si se llamó al diálogo actual con una variable de transferencia, el valor de la variable puede modificarse y devolverse al diálogo inicial.

Los valores de variable se asignan a las variables transferidas con la función "LM" del diálogo inicial al diálogo posterior. Pueden transferirse hasta 20 valores de variable separados entre sí mediante comas.

Nota

El orden de las variables o los valores de variables debe corresponder al orden de las variables de transferencia de la función LM para que la asignación sea unívoca. Si no se indican algunos de los valores de variables, estas variables de transferencia no sufren cambios. Las variables de transferencia modificadas son válidas en el diálogo inicial inmediatamente después de la función LM.

Programación con variable de transferencia

Sintaxis:	**EXIT**[(VARx)]	
Descripción:	Abandonar el diálogo con transferencia de una o varias variables	
Parámetro:	VARx	Nombre de las variables

Ejemplo

```
//M(Máscara1)
...
PRESS(HS1)
  LM("MÁSCARA2","CFI.COM",1, POSX, POSY,
DIÁMETRO)
                                        ; Interrumpir máscara1 y mostrar másca-
                                        ra2. A su vez, transferir las variables
                                        POSX, POSY y DIÁMETRO.
  DLGL("Máscara2 finalizada")           ; Al volver de máscara2, aparece en la lí-
                                        nea de diálogo de máscara1 el texto: Más-
                                        cara2 finalizada.
END_PRESS
...
//END

//M(Máscara2)
...
PRESS(HS1)
  EXIT(5, , DIÁMETRO_CALCULADO)
```

```
                                        ; Salir de máscara2 y regresar a máscara1
                                        en la línea siguiente a LM. Asignar a la
                                        variable POSX el valor 5 y a la
                                         variable DIÁMETRO el valor de la varia-
                                        ble DIÁMETRO_CALCULADO. La variable POSY
                                        conserva su valor actual.
END_PRESS
...
//END
```

8.3.14 Manipulación dinámica de listas de los campos de alternancia o de cuadros de lista

Descripción

Las funciones LISTADDITEM, LISTINSERTITEM, LISTDELETEITEM y LISTCLEAR sirven para manipular de forma dinámica las listas de los campos de alternancia o de cuadro de lista.

Estas funciones actúan solo sobre variables que tienen su propia lista, como

- Lista "sencilla"
  `DEF VAR_AC1 = (I/* 0,1,2,3,4,5,6,7,8)` o bien
- Lista "ampliada"
  `DEF VAR_AC2 = (I/* 0="AC0", 1="AC1", 2="AC2", 3="AC3", 4="AC4", 5="AC5", 6="AC6", 7="AC7", 8="AC8").`

Si la variable apunta a una matriz, p. ej. `DEF VAR_AC3 = (I/* MYARRAY)`, estas funciones no se encuentran disponibles ya que, de lo contrario, cambiaría la matriz global.

Una variable debe haber definido al menos un valor en la línea DEF. Con ello se determina el tipo, "sencilla" o "ampliada".

A continuación se permite borrar la lista por completo y, en su caso, volver a formarla. No obstante, el tipo "sencilla" o "ampliada" debe mantenerse o no puede modificarse de forma dinámica.

Programación

Sintaxis:	**LISTINSERTITEM**(nombre de variable, posición, ItemValue[, ItemDispValue])	
Descripción:	Inserción de un elemento en una posición determinada; lectura de un fichero	
Parámetros:	Nombre de variable	
	Posición	Posición en la que debe agregarse un elemento en la lista
	ItemValue	Valor de la entrada de la lista
	ItemDispValue	Valor tal como debe representarse en la lista

Sintaxis:	**LISTADDITEM**(Variablename, ItemValue[, ItemDispValue])	
Descripción:	Adición de un elemento al final de la lista	
Parámetros:	Nombre de variable	
	ItemValue	Valor de la entrada de la lista
	ItemDispValue	Valor tal como debe representarse en la lista

Sintaxis:	**LISTDELETEITEM**(Variablename, Position)	
Descripción:	Borrado de un elemento determinado	
Parámetros:	Nombre de variable	
	Posición	Posición del elemento de la lista que se va a borrar

Sintaxis:	**LISTCOUNT**(nombre de variable)	
Descripción:	Devuelve el número actual de elementos de la lista	
Parámetro:	Nombre de variable	

Sintaxis:	**LISTCLEAR**(nombre de variable)	
Descripción:	Borrado de la lista completa	
Parámetros:	Nombre de variable	

Ejemplos

Nota

Los siguientes ejemplos se basan unos en los otros. Para reproducir los resultados es importante seguir el orden de los ejemplos.

Requisitos/premisa:

```
DEF VAR_AC = (I/* 0="Off",1="On"/1/,"Switch"/WR2)
```

Adición de un elemento -1="Undefined" al final de la lista:

```
LISTADDITEM("VAR_AC", -1, """Undefined""")
```

Resultado: `0="Off", 1="On", -1="Undefined"`

Inserción de un elemento 99="Maybe" en la posición 2:

```
LISTINSERTITEM("VAR_AC", 2, 99, """Maybe""")
```

Resultado: `0="Off", 1="On", 99="Maybe", -1="Undefined"`

Cálculo del número actual de elementos de la lista:

```
REG[10]=LISTCOUNT("VAR_AC")
```

Resultado: REG[10] = 4

Borrado del elemento en la posición 1:

```
LISTDELETEITEM("VAR_AC", 1)
```

Resultado: `0="Off", 99="Maybe", -1="Undefined"`

Borrado de la lista completa:

```
LISTCLEAR("VAR_AC")
```

Resultado: La lista está vacía

8.3.15 Evaluate (EVAL)

Descripción

La función EVAL evalúa una expresión transferida y, a continuación, la ejecuta. Con ello sólo pueden generarse expresiones en tiempo de ejecución. Esto es útil, p. ej., para accesos indexados a variables.

Programación

Sintaxis:	**EVAL**(exp)	
Descripción:	Evaluación de expresiones	
Parámetro:	exp	Expresión lógica

Ejemplo

```
VAR1=(S)
VAR2=(S)
VAR3=(S)
VAR4=(S)
```

```
CHANGE()
  REG[7] = EVAL("VAR"<<REG[5])  ; La expresión entre paréntesis produce VAR3 si el
                                valor de REG[5] es igual a 3. En consecuencia, se
                                asigna a REG[7] el valor de VAR3.
  IF REG[5] == 1
    REG[7] = VAR1
  ELSE
    IF REG[5] == 2
      REG[7] = VAR2
    ELSE
      IF REG[5] == 3
        REG[7] = VAR3
      ELSE
        IF REG[5] == 4
          REG[7] = VAR4
        ENDIF
      ENDIF
    ENDIF
  ENDIF
END_CHANGE
```

8.3.16 Exit Loading Softkey (EXITLS)

Descripción

Con la función EXITLS se sale de la interfaz actual de usuario y se carga un menú de pulsadores definido.

Nota

Las funciones LS, LM, DLGL, EXIT y EXITLS no deben configurarse en los métodos SUSPEND o RESUME. Si se usan esas funciones en estos métodos, se impide la ejecución de las funciones.

Programación

Sintaxis:	**EXITLS**("Menú de pulsadores"[, "Nombre de ruta"])	
Descripción:	Cargar menú de pulsadores al abandonar	
Parámetro:	Menú de pulsadores	Nombre del menú de pulsadores que se desea cargar
	Nombre de ruta	Ruta del directorio del menú de pulsadores que se desea cargar

Ejemplo

```
PRESS(HS1)
    EXITLS( "Menú1", "AEDITOR.COM" )
END_PRESS
```

8.3.17 Function (FCT)

Descripción

Las funciones externas se almacenan en un fichero DLL y se dan a conocer mediante una entrada en la línea de definición del fichero de configuración.

Nota

La función externa debe tener como mínimo un parámetro de retorno.

Programación

Sintaxis:	**FCT**nombre de función = ("Fichero"/tipo del retorno/tipos de los parámetros fijos/ tipos de los parámetros variables)	
	FCT InitConnection = ("c:\tmp\xyz.dll"/I/R,I,S/I,S)	
Descripción:	La llamada a una función externa puede realizarse, p. ej. desde el método LOAD o en el método PRESS.	
Parámetro:	nombre de la función	Nombre de la función externa
	Fichero	Nombre de ruta completo del fichero DLL
	Tipo de retorno	Tipo de datos del valor de retorno
	Tipo de los parámetros fijos	Parámetros Value
	Tipo de los parámetros variables	Parámetros de referencia
	Los tipos de datos se separan con comas.	

La llamada a la función externa puede realizarse, p. ej. desde el método LOAD o en el método PRESS.

Ejemplo:

```
press(vs4)
RET = InitConnection(VAR1,13,"Servus",VAR2,VAR17)
end_press
```

Estructura de la función externa

La función externa debe observar una firma predeterminada concreta:

Sintaxis:	extern "C" dllexport void InitConnection (ExtFctStructPtr FctRet, ExtFctStructPtr FctPar, char cNrFctPar)	
Descripción:	Exportación DLL solo al implementar en Windows Los especificadores y los parámetros de transferencia están preestablecidos de forma fija. Mediante las estructuras transferidas se retransmiten los parámetros de llamada propiamente dichos.	
Parámetro:	cNrFctPar	Número de parámetros de llamada = número de elementos estructurales en FctPar
	FctPar	Puntero a un campo de elementos estructurales que contienen los correspondientes parámetros de llamada con tipo de datos.
	FctRet	Puntero a una estructura para el retorno del valor de la función con tipo de datos.

Definición de la estructura de transferencia

```
union CFI_VARIANT
    (
    char                                b;
    short int                           i;
    double                              r;
    char*                               s;
    )
typedef struct ExtFctStructTag
    (
    char                                cTyp;
    union CFI_VARIANT                   value;
    )ExtFctStruct;
typedef struct ExtFct* ExtFctStructPtr;
```

Si la función externa debe desarrollarse con independencia de la plataforma (Windows, Linux), no debe utilizarse la palabra clave __declspec(dllexport). Esta palabra clave únicamente es necesaria en Windows. En Qt puede utilizarse por ejemplo la siguiente macro:

```
#ifdef Q_WS_WIN
       #define MY_EXPORT __declspec(dllexport)
#else
       #define MY_EXPORT
#endif
```

La declaración de la función es la siguiente:

```
extern "C" MY_EXPORT void InitConnection
(ExtFctStructPtr FctRet, ExtFctStructPtr FctPar,char cNrFctPar)
```

Si se utilizan las imágenes configuradas con "Run MyScreens" en la NCU y en la PCU/PC, debe omitirse la extensión del fichero binario:

```
FCT InitConnection = ("xyz"/I/R,I,S/I,S)
```

Al omitir la información de ruta absoluta, "Run MyScreens" busca primero el fichero binario en el directorio configurado.

8.3.18 Generate Code (GC)

Descripción

La función GC (Generate Code) genera código de CN desde el método OUTPUT.

Programación

Sintaxis:	**GC**("Identificador"[,"Fichero de destino"][,Opt],[Append])	
Descripción:	Generar código CN (la función GC es solo posible dentro del CN)	
Parámetro:	Identificador	Nombre del método OUTPUT que sirve de base para la generación de código
	Fichero de destino	Nombre de la ruta del fichero de destino para el sistema de ficheros de HMI o de CN Si no se indica el fichero de destino (sólo posible dentro de la ayuda a la programación), el código se escribe en el lugar en el que se encuentra el cursor dentro del fichero abierto actualmente.
	Opt	Opción para la generación de comentarios
	0:	(Ajuste previo) Generar código con comentario para decompilación (ver también Decompilación (Página 175)).
	1:	No crear comentarios en el código generado **Nota:** Este código no es decompilable (ver también Decompilación (Página 175)).
	Append	Este parámetro sólo es significativo cuando se ha indicado un fichero de destino
	0:	(Ajuste predeterminado) Cuando el fichero ya existe se elimina el anterior contenido
	1:	Cuando el fichero ya existe, se escribe el nuevo código al principio del fichero
	2:	Cuando el fichero ya existe, se añade el nuevo código al final

Ejemplo

```
//M(TestGC/"Generación de código:")
```

```
DEF VAR1 = (R//1)
DEF VAR2 = (R//2)
DEF D_NAME
LOAD
  VAR1 = 123
  VAR2 = -6
END_LOAD
OUTPUT(CODE1)
  "Cycle123(" VAR1 "," VAR2 ")"
  "M30"
END_OUTPUT

PRESS(VS1)
  D_NAME = "\MPF.DIR\MESSEN.MPF"
  GC("CODE1",D_NAME)                  ; Escribir el código de CN del método OUTPUT
                                      en el fichero \MPF.DIR\MESSEN.MPF:
                                      Cycle123(123, -6)
                                      M30
END_PRESS
```

Ubicación del fichero de destino

La función GC es solo posible dentro del CN Para mover el fichero use las funciones CP o MP, resp. Ver al respecto los ejemplos siguientes:

```
; Windows directory
  GC("Code","//NC/MPF.DIR/NC1.MPF")
  MP("//NC/MPF.DIR/NC1.MPF","d:/tmp/WIN1.MPF")


; LOCAL_DRIVE
  GC("Code","//NC/MPF.DIR/NC1.MPF")
  MP("//NC/MPF.DIR/NC1.MPF","LOCAL_DRIVE:/WIN_LD1.MPF")
```

Decompilación

- **Sin indicación del fichero de destino:**
  La función GC sólo puede utilizarse en la ayuda a la programación y escribe el código de CN en el fichero actualmente abierto en el editor. El código de CN no se puede decompilar. Si la función GC se configura sin indicación del fichero de destino en "Run MyScreens", aparece un mensaje de error durante la ejecución.
- **Indicación del fichero de destino:**
  El código generado desde el método OUTPUT se introduce en el fichero de destino. Si el fichero de destino no está disponible, se crea en el sistema de ficheros de CN. Si el fichero de destino se encuentra en el sistema de ficheros del HMI, el fichero se guarda en el disco duro. Las líneas de comentario (información necesaria para la decompilación) no se crean, es decir, no se puede realizar una decompilación.

Particularidades en la decompilación

En los subdiálogos no puede llamarse la función GC, pues en los subdiálogos pueden usarse variables que proceden de diálogos principales, pero que no estarían disponibles en una llamada directa.

Al efectuar con el editor intervenciones manuales al código generado no debe modificarse la cantidad de caracteres para valores que se hayan creado mediante la generación de código. Tal modificación impide una decompilación posterior.

Solución:

1. Decompilación
2. Introducir la modificación con la ayuda del diálogo configurado (p. ej. 99 → 101)
3. GC

8.3.19 Funciones de contraseña

Vista general

Las siguientes funciones de contraseña se encuentran disponibles:

- Definir contraseña
- Borrar contraseña
- Cambiar contraseña

Función HMI_LOGIN: Definir clave nueva

Sintaxis:	**HMI_LOGIN**(Passwd)	
Descripción:	Con la función HMI_LOGIN() se envía una contraseña al NCK con la que se puede ajustar el nivel de acceso actual.	
Parámetros:	Passwd	Contraseña

Ejemplo

```
REG[0] = HMI_LOGIN("CUSTOMER")          ; Resultado = TRUE si la contraseña se ha de-
                                        finido correctamente; de lo contrario, FALSE
```

Función HMI_LOGOFF: Borrar contraseña

Sintaxis:	**HMI_LOGOFF**
Descripción:	Con la función HMI_LOGOFF puede restablecerse el nivel de acceso actual.
Parámetros:	- Ninguno -

Ejemplo

```
REG[0] = HMI_LOGOFF                     ; Resultado = TRUE si la contraseña se ha bo-
                                        rrado correctamente; de lo contrario, FALSE
```

Función HMI_SETPASSWD: cambiar contraseña

Sintaxis:	**HMI_SETPASSWD**(AC, Passwd)	
Descripción:	Con la función HMI_SETPASSWD puede cambiarse la contraseña del nivel de contraseña actual o de un nivel de contraseña inferior.	
Parámetros:	AC	Nivel de acceso
	Passwd	Contraseña

Ejemplo

```
REG[0] = HMI_SETPASSWD(4,"MYPWD")       ; Resultado = TRUE si la contraseña se ha mo-
                                        dificado correctamente; de lo contrario, FALSE
```

8.3.20 Load Array (LA)

Descripción

Con la función LA (Load Array) puede cargarse una matriz desde otro fichero.

Programación

Sintaxis:	**LA**(Identificador [, Fichero])
Descripción:	Cargar la matriz del fichero

Parámetro:	Identificador	Nombre de la matriz a recargar
	Fichero	Fichero en el que está definida la matriz

Nota

Si se desea sustituir una matriz en el actual fichero de configuración por una matriz de otro fichero de configuración, los nombres de ambas matrices deberán ser idénticos.

Ejemplo

```
                                          ; Extracto del fichero   mascara.com
DEF VAR2=(S/*ARR5/"Salida"/,"Campo de al-
ternancia")
PRESS(HS5)
  LA("ARR5","arrayext.com")               ; Cargar la matriz ARR5 del fichero arra-
                                          yext.com
  VAR2 = ARR5[0]                          ; En lugar de "DES"/"CON" aparece en el
                                          campo de conmutación de VAR2
                                          "Arriba"/"Abajo"/"Derecha"/"Izquierda"
END_PRESS
//A(ARR5)
("DES"/"CON")
//END
                                          ; Extracto del fichero arrayext.com
//A(ARR5)
("Arriba"/"Abajo"/"Derecha"/"Izquierda")
//END
```

Nota

Tenga en cuenta que a una variable debe asignársele un valor válido tras asignar una matriz diferente al campo de alternancia de la variable mediante la función LA.

8.3.21 Load Block (LB)

Descripción

Con la función LB (Load Block) se pueden cargar bloques con subprogramas en tiempo de ejecución. LB debe configurarse preferiblemente en un método LOAD, para que los subprogramas cargados puedan llamarse en cualquier momento.

Nota

Los subprogramas también pueden definirse directamente en un diálogo, en cuyo caso no tienen que cargarse.

Programación

Sintaxis:	**LB**("Nombre de bloque"[,"Fichero"])	
Descripción:	Cargar subprograma en tiempo de ejecución	
Parámetro:	Nombre de bloque	Nombre del identificador de bloque
	Fichero	Nombre de ruta del fichero de configuración Ajuste previo = fichero de configuración actual

Ejemplo

```
LOAD
  LB("PROG1")             ; El bloque "PROG1" se busca en el fichero de configura-
                          ción actual y se carga a continuación.
  LB("PROG2","XY.COM")    ; El bloque "PROG2" se busca en el fichero de configura-
                          ción XY.COM y se carga a continuación.
END_LOAD
```

8.3.22 Load Mask (LM)

Descripción

Con la función LM se carga un nuevo diálogo. En función del modo de cambio de cuadro de diálogo (ver tabla más adelante) con esta función puede implementarse también un cuadro de diálogo de mensaje.

Nota

Las funciones LS, LM, DLGL, EXIT y EXITLS no deben configurarse en los métodos SUSPEND o RESUME. Si se usan esas funciones en estos métodos, se impide la ejecución de las funciones.

Diálogo principal / subdiálogo

Un diálogo que abre otro diálogo y no se cierra por sí mismo se denomina como diálogo principal. Un diálogo que es abierto desde un diálogo principal se denomina como subdiálogo.

Programación

Sintaxis:	**LM**("Identificador"[,"Fichero"] [,MSx [, VARx]])	
Descripción:	Cargar diálogo	
Parámetro:	Identificador	Nombre del diálogo que se desea cargar
	fichero	Nombre de ruta (sistema de ficheros HMI o de CN) del fichero de configuración; ajuste estándar: Dato de configuración actual
	MSx	Modo del cambio de diálogo
	0:	(Ajuste previo) El diálogo actual se desecha, el nuevo diálogo se carga y se muestra. Con EXIT se regresa a la aplicación estándar. Mediante el parámetro MSx se puede determinar si al efectuar un cambio de diálogo el actual debe o no finalizarse. Si el diálogo actual permanece, se pueden transferir variables al nuevo diálogo. La ventaja del parámetro MSx consiste en que, al cambiar, los diálogos no se necesitan reinicializar cada vez, sino que se conservan los datos y la maquetación del diálogo actual y se facilita la transferencia de datos.
	1:	El cuadro de diálogo principal actual se interrumpe a partir de la función LM y se carga y muestra el nuevo subdiálogo (p. ej. para la implementación de un cuadro de diálogo de mensaje). Con EXIT se cierra el subdiálogo y se vuelve al punto de interrupción del diálogo principal. En el diálogo principal, el bloque UNLOAD no se ejecuta en la interrupción.
	VARx	Requisitos: MS1 Lista de variables que se pueden transferir del diálogo principal al subdiálogo. Pueden transferirse hasta 20 variables separadas entre sí mediante comas.

Nota

El parámetro VARx únicamente transfiere el valor de cada variable, es decir, las variables pueden leerse y modificarse en el subdiálogo, pero no son visibles allí. El retorno de las variables del subdiálogo al diálogo principal es posible mediante la función EXIT.

Ejemplo

```
MÁSCARA1(diálogo principal): salto al subdiálogo MÁSCARA2 con transferencia de variables
PRESS(HS1)
```

MÁSCARA1(diálogo principal): salto al subdiálogo MÁSCARA2 con transferencia de variables

```
  LM("MÁSCARA2","CFI.COM",1, POSX, POSY, DIÁMETRO)
                                    ; Interrumpir máscara1 y mostrar máscara2: A su
                                    vez, se transfieren las variables POSX, POSY y
                                    DIÁMETRO.
  DLGL("Máscara2 finalizada")       ; Al volver de máscara2, aparece en la línea de
                                    diálogo de máscara1 el texto: Máscara2 finali-
                                    zada.
END_PRESS
```

MÁSCARA2(subdiálogo): salto atrás al diálogo principal MÁSCARA1 con transferencia de los contenidos de variables

```
PRESS(VS8)

  EXIT(POSX, POSY, DIÁMETRO)
                                    ; desactivar Máscara2 y reanudar Máscara1: A su
                                    vez, se transfieren las variables modificadas
                                    POSX, POSY y DIÁMETRO.
END_PRESS
```

8.3.23 Load Softkey (LS)

Descripción

Con la función LS se puede insertar otro menú de pulsadores.

Nota

Las funciones LS, LM, DLGL, EXIT y EXITLS no deben configurarse en los métodos SUSPEND o RESUME. Si se usan esas funciones en estos métodos, se impide la ejecución de las funciones.

Programación

Sintaxis:	**LS**("Indicador"[, "Fichero"][, Merge])
Descripción:	Mostrar menú de pulsadores

Parámetro:	Identificador	Nombre del menú de pulsadores
	Fichero	Nombre de ruta (sistema de ficheros HMI o de CN) del fichero de configuración Ajuste predeterminado: fichero de configuración actual
	Merge	
	0:	Todos los pulsadores de menú existentes se borran; se introducen los nuevos pulsadores de menú configurados.
	1:	Ajuste predeterminado Solo los nuevos pulsadores de menú configurados sobrescriben a los existentes; los demás pulsadores de menú (= pulsadores de menú de la aplicación HMI) se conservan con su funcionalidad y texto.

Ejemplo

```
PRESS(HS4)
  LS("Menú2",,0)      ; Menú2 sobrescribe el menú de pulsadores existente; los pul-
                      sadores de menú mostrados se borran.
END_PRESS
```

Nota

Mientras el intérprete no haya mostrado aún ningún diálogo, es decir, no se haya procesado todavía ninguna función LM, sólo puede configurarse en los métodos PRESS del bloque descriptivo del pulsador de menú de inicio y de los menús de pulsadores un comando LS o LM y ninguna otra acción.

Las funciones LS y LM pueden abrirse exclusivamente dentro de un método PRESS de un pulsador de menú, aunque no como reacción a las teclas de navegación (PU, PD, SL, SR, SU, SD).

8.3.24 Load Grid (LG)

Descripción

La descripción de tabla (grid) puede realizarse de manera dinámica dentro de los métodos (p. ej., LOAD) mediante el método LG.

Para que pueda asignarse una tabla con el método LG, la variable debe estar previamente definida como variable de Grid y remitir a una tabla válida preexistente.

Programación

Sintaxis:	**LG**(nombre de grid, nombre de variable[,nombre de fichero])
Descripción:	Cargar tabla

Parámetro:	Nombre del grid	Nombre de la tabla (Grid) entre comillas simples
	Nombre de variable	Nombre de la variable a la que debe asignarse la tabla, entre comillas simples
	Nombre de fichero	Nombre del fichero en el que está definida la tabla (Grid), entre comillas simples. Sólo debe indicarse cuando la tabla no esté definida dentro del fichero en el que está definida también la variable

Ejemplo

```
//M(MyGridSample/"My Grid Sample")
DEF MyGridVar=(R/% MyGrid1///WR2////100,,351,100)
LOAD
   LG("MyGrid1","MyGridVar","mygrids.com")
END_LOAD
```

Contenido de mygrids.com:

```
//G(MyGrid1/0/5)
(I///,"MyGrid1"/wr1//"1"/80/1)
(R3///"LongText1","R1-R4"/wr2//"$R[1]"/80/1)
(IBB///"LongText2","M2.2-M2.5"/wr2//"M2.2"/80/,1)
(R3///"LongText3","R9,R11,R13,R15"/wr2//"$R[9]"/110/2)
//END
```

8.3.25 Selección múltiple SWITCH

Descripción

Con un comando SWITCH pueden verificarse distintos valores de una variable.

La expresión configurada en la instrucción SWITCH se compara sucesivamente con todos los valores configurados en las siguientes instrucciones CASE.

Si no hay coincidencias, se compara con la instrucción CASE siguiente. Si le sigue una instrucción DEFAULT y hasta el momento no ha habido ninguna instrucción CASE igual a la expresión configurada en la instrucción SWITCH, se ejecutan las siguientes instrucciones que siguen a la instrucción DEFAULT.

Si hay coincidencias, se ejecutan las siguientes instrucciones hasta que le siga una instrucción CASE, DEFAULT o END_SWITCH.

Ejemplo

```
FOCUS
   SWITCH (FOC)
      CASE "VarF"
         DLGL("Variable ""VarF"" has the input focus.")
      CASE "VarZ"
         DLGL("Variable ""VarZ"" has the input focus.")
      DEFAULT
         DLGL("Any other variable has the input focus.")
   END_SWITCH
END_FOCUS
```

8.3.26 Multiple Read NC PLC (MRNP)

Descripción

Con el comando MRNP pueden transferirse varias variables del CN/PLC con un acceso al registro. Este acceso es considerablemente más rápido que la lectura mediante acceso individual. Las variables del CN/PLC deben ser del mismo rango dentro de un comando MRNP.

Los rangos de las variables del CN/PLC están estructurados como sigue:

- Datos de CN generales ($MN..., $SN.., /nck/...)
- Datos de NC específicos del canal ($MC..., $SC.., /channel/...)
- Datos de PLC (DB..., MB.., /plc/...)
- Datos de CN específicos del eje para un mismo eje ($MA..., $SA...)

Programación

Sintaxis:	**MRNP**(Nombre de variable1*Nombre de variable2[* ...], Índice del registro)
Descripción:	Leer varias variables
Parámetro:	Para los nombres de variables se utiliza "*" como carácter de separación. Los valores se transfieren al registro REG[índice del registro] y siguientes según el orden con el que se suceden los nombres de las variables en el comando. Según esto, se tiene que: El valor de la primera variable se encuentra en REG[índice del registro]. El valor de la segunda variable se encuentra en REG[índice del registro + 1], etc.

Nota

Se ha de tener en cuenta que la lista de variables puede tener máx. 500 caracteres y el número de registros está limitado.

Ejemplo

```
MRNP("$R[0]*$R[1]*$R[2]*$R[3]",1)      ;REG[1] a REG[4] se escriben con el valor de
                                        las variables de $R[0] a $R[3].
```

Variable de CN

Se encuentran disponibles todos los datos del operador y de máquina así como los parámetros R, pero solo determinadas variables de sistema (ver también AUTOHOTSPOT).

Son accesibles todas las variables de usuario específicas del canal y globales (GUD). Las variables de usuario locales y globales del programa no se pueden procesar.

Datos de máquina	
Dato global de máquina	$MN_...
Dato de máquina específico del eje	$MA_...
Dato de máquina específico del canal	$MC_...

Datos de operador	
Dato del operador global	$SN_...
Dato del operador específico del eje	$SA_...
Dato del operador específico del canal	$SC_...

Variables del sistema	
Parámetro R 1	$R[1]

Variable de PLC

Están disponibles todos los datos de PLC.

Datos PLC	
Byte y bit z del bloque de datos x	DBx.DBXy.z
Byte y del bloque de datos x	DBx.DBBy
Word y del bloque de datos x	DBx.DBWy
Double Word y del bloque de datos x	DBx.DBDy
Real y del bloque de datos x	DBx.DBRy
Byte de marcas x bit y	Mx.y

Datos PLC	
Byte de marcas x	MBx
Word de marcas x	MWx
Double Word de marcas x	MDx
Byte de entrada x bit y	Ix.y o Ex.y
Byte de entrada x	IBx o EBx
Word de entrada x	IWx o EWx
Double Word de entrada x	IDx o EDx
Byte de salida x bit y	Qx.y o Ax.y
Byte de salida x	QBx o ABx
Word de salida x	QWx o AWx
Double Word de salida x	QDx o ADx
String y con longitud z del bloque de datos x	DBx.DBSy.z

8.3.27 P_LOGICAL_FILEPATH

Descripción

La función P_LOGICAL_FILEPATH devuelve el nombre y la ruta lógica del programa de pieza editado actualmente en el editor.

Nota

La función solo está disponible en el contexto de máscaras de ciclo.

Programación

Sintaxis:	P_LOGICAL_FILEPATH()	
Descripción:	Determinar el nombre y la ruta lógica del programa de pieza editado actualmente en el editor p. ej. "//NC/MPF.DIR/HUGO.MPF"	
Parámetro:	- Ninguno -	

Consulte también

P_REAL_FILEPATH (Página 170)

8.3.28 P_REAL_FILEPATH

Descripción

La función P_REAL_FILEPATH devuelve el nombre y la ruta real del programa de pieza editado actualmente en el editor.

Nota

La función solo está disponible en el contexto de máscaras de ciclo.

Programación

Sintaxis:	P_REAL_FILEPATH()	
Descripción:	Determinar el nombre y la ruta real del programa de pieza editado actualmente en el editor p. ej. "/_N_MPF_DIR/_N_HUGO_MPF"	
Parámetro:	- Ninguno -	

Consulte también

P_LOGICAL_FILEPATH (Página 169)

8.3.29 Servicios de PI

Descripción

Con la función PI_START es posible iniciar servicios de PI (servicios de instancia de programa) desde el PLC en el área de CN.

Nota

En el manual de funciones Funciones básicas hay una lista de los servicios PI disponibles.

Programación

Sintaxis:	**PI_START**("String de transferencia")	
Descripción:	Ejecutar servicio de PI	
Parámetro:	"String de transferencia"	El string de transferencia, contrariamente a la documentación OEM, debe introducirse entre comillas dobles.

Ejemplo

```
PI_START("/NC,001,_N_LOGOUT")
```

Nota

Los servicios de PI dependientes del canal están referidos siempre al canal actual.

Los servicios de PI de las funciones de herramienta (área TO) se refieren siempre al área TO asignada al canal actual.

8.3.30 Leer (RNP) y escribir (WNP) variable de sistema o de usuario

Descripción

Con el comando RNP (Read NC PLC) se pueden leer variables de CN o PLC o datos de máquina.

Programación

Sintaxis:	**RNP** ("Variable de sistema o de usuario", valor)	
Descripción:	Leer variables de CN o PLC o datos de máquina	
Parámetro:	Variable de sistema o de usuario	Nombre de la variable de CN o PLC
	Valor	Valor que debe escribirse en la variable de sistema o de usuario Si el valor es de tipo string, debe escribirse entre comillas dobles.

Ejemplo

```
VAR2=RNP("$AA_IN[2]")         ; Leer variable de CN
```

Descripción

Con el comando RNP (Write NC PLC) se pueden escribir variables de CN o PLC o datos de máquina.

Los accesos a variables de CN/PLC se vuelven a ejecutar en cada procesamiento de la función WNP, de modo que un acceso a CN/PLC en un método CHANGE se ejecuta siempre. Esto es conveniente cuando una variable de sistema o de usuario modifica con frecuencia su valor. Si sólo debe realizarse un único acceso al CN/PLC, esto debe configurarse en un método LOAD o UNLOAD.

Programación

Sintaxis:	**WNP**("Variable de sistema o de usuario", valor)	
Descripción:	Escribir variables de CN o PLC o datos de máquina	
Parámetro:	Variable de sistema o de usuario	Nombre de la variable de CN o PLC
	Valor	Valor que debe escribirse en la variable de sistema o de usuario Si el valor es de tipo string, debe escribirse entre comillas dobles.

Ejemplo

```
WNP("DB20.DBB1",1)        ; Escribir variable de PLC
```

8.3.31 RESIZE_VAR_IO y RESIZE_VAR_TXT

Descripción

Con los comandos `RESIZE_VAR_IO()` y `RESIZE_VAR_TXT()` puede modificarse la geometría de las partes de entrada/salida o de texto de una variable.

Después de definir la nueva geometría, todos los campos de la máscara se colocan como si la máscara se hubiera configurado con estas posiciones desde el principio. De este modo, todos los campos se alinean entre sí correctamente, según la configuración.

Programación

Sintaxis:	**RESIZE_VAR_IO**(nombre de variable, [X], [Y], [anchura], [altura]) **RESIZE_VAR_TXT**(nombre de variable, [X], [Y], [anchura], [altura])	
Descripción:	Modificación de la geometría de la parte de entrada/salida o de texto de una variable.	
Parámetros:	Nombre de variable	Nombre de la variable cuya geometría de la parte de entrada/salida o de texto de una variable debe modificarse.
	X	Coordenada X arriba izquierda
	Y	Coordenada Y arriba izquierda
	Anchura	Anchura
	Altura	Altura

Nota

Para cada valor de X, Y, anchura o altura no especificado se conserva el valor anterior. Si se indica -1 en uno de estos valores, se define la parte de la posición estándar predefinida por "Run MyScreens".

Ejemplo

```
RESIZE_VAR_IO("MyVar1", 200, , 100)    ; La parte de E/S de la variable "MyVar1" se
                                       desplaza 200 píxeles en la posición X; la an-
                                       chura se define en 100 píxeles. La posición Y
                                       y la altura del valor anterior se mantienen.
```

8.3.32 Registro (REG)

Descripción Registro

Los registros son necesarios para intercambiar datos entre distintos diálogos. Los registros están asignados siempre a un diálogo y se definen, al cargar el primer diálogo, con 0 o un string vacío.

Nota

Los registros no deben utilizarse directamente en un método OUTPUT para la generación de código de CN.

Programación

Sintaxis:	**REG**[x]	
Descripción:	Definir registros	
Parámetro:	x	Índice de registro con x = 0...19; Tipo: REAL o STRING = VARIANT Los registros con x ≥ 20 ya están definidos por Siemens.

Descripción valor de registro

La asignación de valores a los registros se configura en un método.

Nota

Si desde un diálogo se genera otro con la función LM, el contenido de los registros se transfiere automáticamente al nuevo diálogo y está disponible para otros cálculos en el segundo diálogo.

Programación

Sintaxis:	Identificador.val = Valor del registro o bien Identificador = Valor del registro
Descripción:	

Parámetro:	Identificador	Nombre del registro
	Valor de un registro	Valor del registro

Ejemplo

```
UNLOAD
  REG[0] = VAR1          ; Asignar al registro 0 el valor de la variable 1
END_UNLOAD

UNLOAD
  REG[9].VAL = 84        ; Asignar al registro 9 el valor 84
END_UNLOAD
                         ; En el diálogo siguiente se pueden asignar nuevamente es-
                         tos registros a variables locales en un método.
LOAD
  VAR2 = REG[0]
END_LOAD
```

Descripción estado de un registro

Con la propiedad estado puede consultarse en la configuración si un registro contiene un valor válido.

La consulta sobre el estado de un registro puede emplearse, entre otras cosas, para escribir un valor en un registro únicamente en el caso de que un diálogo se utilice como diálogo principal.

Programación

Sintaxis:	Identificador.vld	
Descripción:	Esta propiedad es de sólo lectura.	
Parámetro:	Identificador	Nombre del registro
Valor de retorno:		El resultado de la consulta puede ser:
	FALSE =	Valor no válido
	TRUE =	Valor válido

Ejemplo

```
IF REG[15].VLD == FALSE    ; Consultar validez del valor del registro
  REG[15] = 84
ENDIF
VAR1 = REG[9].VLD          ; Asignar a Var1 el valor de la consulta de estado de
                           REG[9].
```

8.3.33 RETURN

Descripción

Con la función RETURN puede cancelarse con antelación el subprograma actual y regresarse al punto de salida del último comando CALL.

Si en el subprograma no se configura ningún RETURN, el subprograma se ejecuta hasta el final y se vuelve después al punto de salida.

Programación

Sintaxis:	**RETURN**	
Descripción:	Volver al punto de salida	
Parámetro:	- Ninguno -	

Ejemplo

```
//B(PROG1)                ; Comienzo del bloque
SUB(UP2)                  ; Inicio de subprograma
  IF VAR1.val=="Otto"
    VAR1.val="Hans"
    RETURN                ; Si el valor de variable es = Otto, se asigna a la variable el valor "Hans" y se termina el subprograma en este punto.
  ENDIF
  VAR1.val="Otto"         ; Si el valor de variable es ≠ Otto, se asigna a la variable el valor "Otto".
END_SUB                   ; Final de subprograma
//END                     ; Fin del bloque
```

8.3.34 Decompilación

Descripción

En la ayuda a la programación existe la posibilidad de **decompilar** con la función GC el código de CN creado y volver a mostrar los valores de variable en el campo de entrada/salida del diálogo de entrada correspondiente.

Programación

Las variables del código de CN se aplican en el diálogo. Al hacerlo, los valores de variable del código de CN se comparan con los valores de variable calculados a partir del fichero de configuración. Si no hay coincidencias, se emite un mensaje de error al diario de incidencias, ya que los valores del código de CN generado se han modificado.

Si una variable aparece repetida en el código de CN, con la decompilación siempre se evaluará su última aparición. Además, se enviará una advertencia en el diario de incidencias.

Las variables no utilizadas durante la generación de código en el código de CN se guardan como comentarios. Con comentario se denominan todas las informaciones necesarias para la decompilación. El comentario no se puede modificar.

Nota

El bloque formado por el código de CN y el comentario sólo se puede decompilar cuando comienza al principio de una línea.

Ejemplos:

En el programa se encuentra el siguiente código de CN:

```
DEF VAR1 = (I//101)
OUTPUT(CODE1)
  "X" VAR1 " Y200"
  "X" VAR1 " Y0"
END_OUTPUT
```

En el programa de pieza se guarda el siguiente código:

```
;NCG#TestGC#\cus.dir\aeditor.com#CODE1#1#3#
X101 Y200
X101 Y0
;#END#
```

Durante la decompilación, el editor lee:

```
X101 Y200
X222 Y0          ; El valor para X fue modificado en el programa de pieza (X101→ X222)
```

En el diálogo de entrada se preselecciona el siguiente valor para VAR1: `VAR1 = 222`

Consulte también

Generate Code (GC) (Página 157)

8.3.35 Decompilación sin comentarios

Descripción

En la ayuda a la programación existe la posibilidad de **decompilar sin comentarios** el código de CN creado con la función GC y volver a mostrar los valores de variables en el campo de entrada/salida del diálogo de entrada correspondiente.

Programación

Para suprimir las líneas de comentario creadas mediante generación de código normal, el comando GC puede ejecutarse de la siguiente manera:

```
GC("CODE1",D_NAME,1)
```

El código resultante no se puede decompilar normalmente. Para poder decompilar de todos modos las llamadas de ciclos generadas, son necesarios los siguientes pasos:

- **Ampliar el fichero "easyscreen.ini"**
  En el fichero "easyscreen.ini" se introduce la sección `[RECOMPILE_INFO_FILES]`. En esta sección se da una lista de todos los ficheros ini que contienen descripciones de ciclos para decompilar sin comentarios:
  ```
  [RECOMPILE_INFO_FILES]
  IniFile01 = cycles1.ini
  IniFile02 = cycles2.ini
  ```
  Pueden indicarse varios ficheros ini, y sus nombres pueden elegirse libremente.
- **Crear fichero ini para descripción de ciclos**
  Almacene el fichero ini con las descripciones de ciclos en la siguiente ruta:
  [Directorio de usuario del sistema]/cfg
  [Directorio OEM del sistema]/cfg
  [Directorio de addon del sistema]/cfg
  Para cada ciclo se necesita una sección propia. El nombre de la sección es el mismo que el nombre del ciclo:
  ```
  [Cycle123]
  Mname = TestGC
  Dname = testgc.com
  OUTPUT = Code1
  Nump = 3
  Version = 0
  Code_typ = 1
  Icon = cycle123.png
  Desc_Text = This is describing text
  ```

Mname	Nombre de la máscara
Dname	Nombre del fichero en el que está definida la máscara
OUTPUT	Nombre del método Output en cuestión
Nump	Número de parámetros de la máscara que debe decompilarse (todas las variables creadas con DEF, también variables auxiliares)
Version	(opcional) Dato de versión del ciclo

Icon	(opcional) Símbolo que se muestra en el programa de cadenas secuenciales, formato *.png Tamaño de imagen para cada resolución: 640 x 480 mm → 16 x 16 píxeles 800 x 600 mm → 20 x 20 píxeles 1024 x 768 mm → 26 x 26 píxeles 1280 x 1024 mm → 26 x 26 píxeles 1280 x 768 mm → 26 x 26 píxeles Almacenamiento: [Directorio de usuario del sistema]/ico/ico<Resolución> **Notas:** • Para la resolución de 1280 píxeles se utiliza la carpeta de 1024 x 768 mm (solo adecuado para programas de cadenas secuenciales). • El tamaño de imagen no solo depende de la resolución, sino también del tamaño de fuente ajustado en el editor.
Desc_Text	(opcional) Texto explicativo que se muestra en el programa de cadenas secuenciales, longitud máx. de cadena 17 caracteres (solo adecuado para programas de cadenas secuenciales)
Param_Text	(opcional) Texto para la lista de parámetros T=%1 D=%2

Nota

Indicaciones sobre Icon, Desc_Text y Param_Text:

1. El símbolo de usuario se visualiza siempre en el editor de código G. En el editor ShopMill/ShopTurn se visualiza siempre el símbolo de código G. Esto permite distinguir más fácilmente entre los pasos ShopMill/ShopTurn y los pasos que NO son ShopMill/ShopTurn.
2. El paso Run MyScreens depende de cómo se haya ajustado "Representar ciclos como paso de trabajo" en el editor. Esto vale para el código G y el editor JobShop.
   Es decir, si "Representar ciclos como paso de trabajo" = "SÍ", se toma el "Desc_Text" de la configuración de Run MyScreens, y para "NO", se visualiza el ciclo.
3. Desc_Text y Param_Text
   – se pueden indicar en función del idioma mediante la notación $xxxxx
   – a través de %1, %2, ... se puede acceder a los parámetros de la llamada cíclica generada. Para ello, el comodín es reemplazado por el parámetro. En la lista de parámetros generada, el parámetro figura en la posición que se indica detrás de '%'.

Ejemplo

```
//M(TestGC/"Generación de código:")
DEF VAR1 = (R//1)
DEF VAR2 = (R//2)
DEF D_NAME
```

```
LOAD
  VAR1 = 123
  VAR2 = -6
END_LOAD
OUTPUT(CODE1)
  "Cycle123(" VAR1 "," VAR2 ")"
  "M30"
END_OUTPUT

PRESS(VS1)
  D_NAME = "\MPF.DIR\MESSEN.MPF"
  GC("CODE1",D_NAME)                    ; Escribir el código de CN del método
                                        OUTPUT en el fichero \MPF.DIR\MES-
                                        SEN.MPF:
                                        Cycle123(123, -6)
                                        M30

END_PRESS
```

Limitaciones

- La funcionalidad "Decompilación sin comentarios" no dispone del abanico de funciones completo, como "Decompilar con comentarios".
  Con "Decompilar sin comentarios" se admiten llamadas de ciclos típicas como, p. ej. MYCYCLE(PAR1, PAR2, PAR3, ...). En la línea de la llamada a la función no puede haber ningún comentario. Sin embargo, los parámetros opcionales que no se transfieren al llamar la función y son de tipo string S deben indicarse al menos con comillas vacías, p. ej. "". Si no, "Run MyScreens" trata de completar estos parámetros con comas para decompilar a continuación la llamada de ciclo rellenada así.
- Los parámetros de tipo string no pueden contener caracteres de coma ni punto y coma en el string que se va a transferir.
- Al realizar "Decompilar sin comentarios" todas las variables contenidas en el método OUTPUT deben ir siempre entre paréntesis para que pueda actuar la funcionalidad para "rellenar con comas los parámetros que faltan en el ciclo".

Ejemplo:

Permitido:

```
OUTPUT
    "MYCYCLE(" MYPAR1 "," MYPAR2 "," MYPAR3 ")"
END_OUTPUT
```

No permitido (la variable MYCOMMENT se coloca tras el paréntesis de cierre):

```
OUTPUT
```

```
    "MYCYCLE(" MYPAR1 "," MYPAR2 "," MYPAR3 ")" MYCOMMENT
END_OUTPUT
```

8.3.36 Search Forward, Search Backward (SF, SB)

Descripción

Con la función **SF, SB (Search Forward, Search Backward)** en el programa de CN actual del editor se puede buscar un string desde la posición actual del cursor y emitir su valor.

Programación

Sintaxis:	**SF**("String")	
Identificación:	**S**earch **F**orward: Buscar desde la posición actual del cursor hacia delante	
Sintaxis:	**SB**("String")	
Identificación:	**S**earch **B**ackward: Buscar desde la posición actual del cursor hacia atrás	
Parámetro:	String	Texto de la búsqueda

Reglas para la búsqueda

- Antes y después de la unidad formada por el string de búsqueda y su valor, en el programa de CN actual debe haber un espacio en blanco.
- El término de búsqueda no se buscará ni en comentarios ni dentro de un string.
- El valor introducido debe ser una expresión numérica; no se reconocerán expresiones del tipo "X1=4+5".
- Las constantes hexadecimales del tipo X1='HFFFF', las constantes binarias del tipo X1='B10010' y las constantes exponenciales del tipo X1='-.5EX-4' sí se reconocen.
- El valor de la señal se puede emitir cuando entre el string y el valor aparece lo siguiente:
  - Nada
  - Espacio en blanco
  - Signo igual

Ejemplo

Son posibles las siguientes notaciones:

```
X100 Y200               ; La variable Abc recibe el valor 200
Abc = SB("Y")
X100 Y 200              ; La variable Abc recibe el valor 200
Abc = SB("Y")
```

```
X100 Y=200              ; La variable Abc recibe el valor 200
Abc = SB("Y")
```

8.3.37 SL_HMI_INSTALL_DIR

Descripción

La función SL_HMI_INSTALL_DIR devuelve la ruta del directorio de instalación de SINUMERIK Operate.

Programación

Sintaxis:	SL_HMI_INSTALL_DIR()	
Descripción:	Determinar el directorio de instalación de SINUMERIK Operate	
Parámetro:	- Ninguno -	

8.3.38 SL_TEMP_DIR

Descripción

La función SL_TEMP_DIR devuelve una ruta para ficheros temporales en SINUMERIK Operate.

Nota

El contenido de este directorio no es persistente, es decir, se borra cada vez que se inicia SINUMERIK Operate.

Programación

Sintaxis:	SL_TEMP_DIR()	
Descripción:	Determinar la ruta del directorio no persistente (volátil) de SINUMERIK Operate	
Parámetro:	- Ninguno -	

8.3.39 Funciones STRING

Vista general

Las siguientes funciones permiten procesar strings:

- Cálculo de la longitud de strings
- Búsqueda de un carácter en un string
- Extracción de parte de un string comenzando por la izquierda
- Extracción de parte de un string comenzando por la derecha
- Extracción de la parte central de un string
- Sustitución de partes de un string
- Comparación de strings
- Inserción de un string en otro string
- Eliminación de un string incluido en otro string
- Eliminación de espacios en blanco (a la izquierda o a la derecha)
- Inserción de valores o cadenas con identificadores de formato

Función LEN: Longitud de un string

Sintaxis:	**LEN**(string | varname)	
Descripción:	Determinar el número de caracteres de un string	
Parámetros:	string	Cualquier expresión válida de tipo string. En el caso de un string vacío se retorna CERO.
	varname	Cualquier nombre de variable válido y declarado
	Solo es admisible uno de los dos posibles parámetros.	

Ejemplo

```
DEF VAR01
DEF VAR02

LOAD
  VAR01="HALLO"
  VAR02=LEN(VAR01)          ; Resultado = 5
END_LOAD
```

Función INSTR: Buscar caracteres en una cadena de caracteres

Sintaxis:	**INSTR**(Inicio, String1, String2 [,Dirección])
Descripción:	Buscar caracteres

Parámetros:	Inicio	Posición inicial desde la cual se busca el string1 en string2. Cuando la búsqueda deba comenzar al principio de string2, debe introducirse 0.
	String1	Carácter que se busca
	String2	Cadena de caracteres en la que se busca
	Dirección (opción)	Dirección en la que se busca 0: de izquierda a derecha (ajuste predeterminado) 1: de derecha a izquierda
	Si string1 no está contenido en string2, se devuelve el valor 0.	

Ejemplo

```
DEF VAR01
DEF VAR02

LOAD
  VAR01="HALLO/WELT"
  VAR02=INST(1,"/",VAR01)        ; Resultado = 6
END_LOAD
```

Función LEFT: String desde la izquierda

Sintaxis:	**LEFT**(string, longitud)	
Descripción:	LEFT devuelve una cadena de caracteres que contiene el número de caracteres indicado comenzando por la izquierda de un string.	
Parámetros:	string	Cadena de caracteres o variable con la cadena de caracteres que se desea procesar
	longitud	Número de caracteres que se desea leer

Ejemplo

```
DEF VAR01
DEF VAR02
LOAD
  VAR01="HALLO/WELT"
  VAR02=LEFT(VAR01,5)           ; Ergebnis = "HALLO"
END_LOAD
```

Función RIGHT: String desde la derecha

Sintaxis:	**RIGHT**(string, longitud)	
Descripción:	RIGHT devuelve una cadena de caracteres que contiene el número de caracteres indicado comenzando por la derecha de un string.	
Parámetros:	string	Cadena de caracteres o variable con la cadena de caracteres que se desea procesar
	longitud	Número de caracteres que se desea leer

Ejemplo

```
DEF VAR01
DEF VAR02
LOAD
  VAR01="HALLO/WELT"
  VAR02=LEFT(VAR01,4)         ; Resultado = "WELT"
END_LOAD
```

Función MIDS: String del centro

Sintaxis:	**MIDS**(string, inicio [, longitud])	
Descripción:	MIDS devuelve una cadena de caracteres que contiene el número de caracteres indicado comenzando por el lugar especificado de un string.	
Parámetros:	string	Cadena de caracteres o variable con la cadena de caracteres que se desea procesar
	inicio	Punto inicial desde el que se lee en la cadena de caracteres
	longitud	Número de caracteres que se desea leer

Ejemplo

```
DEF VAR01
DEF VAR02
LOAD
  VAR01="HALLO/WELT"
  VAR02=LEFT(VAR01,4,4)           ; Ergebnis = "LO/W"
END_LOAD
```

Función REPLACE: Reemplazo de caracteres

Sintaxis:	**REPLACE**(string, FindString, ReplaceString [, inicio [, count]])
Descripción:	La función REPLACE reemplaza un carácter o una cadena de caracteres dentro de un string por otro carácter u otra cadena de caracteres.

Parámetros:	string	String en el que debe reemplazarse FindString por ReplaceString.
	FindString	String que se desea reemplazar
	ReplaceString	String de reemplazo (se inserta en lugar de FindString)
	inicio	Posición inicial a partir de la cual se busca y sustituye
	count	Número de caracteres dentro de los que se debe buscar FindString a partir de la posición de inicio
Valor de retorno:		
	string = string vacío	Copia de String
	FindString = string vacío	Copia de String
	ReplaceString = string vacío	Copia de String en la que se han eliminado todas las apariciones de FindString
	inicio > Len(String)	Cadena vacía
	count = 0	Copia de String

Función STRCMP: comparación de strings

Sintaxis:	**STRCMP**(string1, string2 [, CaseInsensitive])	
Descripción:	STRCMP compara una cadena de caracteres con otra.	
Parámetros:	string1	Cadena de caracteres que debe compararse con una segunda cadena de caracteres
	string2	Cadena de caracteres que debe compararse con la primera cadena de caracteres
	CaseInsensitive	Comparación con/sin distinción de mayúsculas/minúsculas: sin indicación o FALSE = distinción entre mayúsculas/minúsculas TRUE = sin distinción entre mayúsculas/minúsculas

Ejemplo

```
REG[0]=STRCMP("Hugo", "HUGO")              ; Resultado = 32
                                           ; (<> 0, los strings no son idénticos)

REG[0]=STRCMP("Hugo", "HUGO", TRUE)        ; Resultado = 0
                                           ; (== 0, los strings son idénticos
```

Función STRINSERT: inserción de strings

Sintaxis:	**STRINSERT**(string1, string2, insert position)
Descripción:	STRINSERT inserta una cadena de caracteres en una posición determinada de la cadena de caracteres.

Parámetros:	string1	Cadena de caracteres en la que debe insertarse otra cadena de caracteres
	string2	Cadena de caracteres que debe insertarse
	insert position	Posición en la que debe insertarse la cadena de caracteres

Ejemplo

```
REG[0]=STRINSERT("Hello!", " world", 5)    ; Resultado = "Hello world!")
```

Función STRREMOVE: eliminación de cadena

Sintaxis:	**STRREMOVE**(string1, remove position, count)	
Descripción:	STRREMOVE elimina un número determinado de caracteres en una posición determinada de una cadena de caracteres.	
Parámetros:	string1	Cadena de caracteres de la que se deben eliminar caracteres
	remove position	Posición desde la que deben eliminarse caracteres de la cadena de caracteres
	count	Número de caracteres que se deben eliminar

Ejemplo

```
REG[0]=STRREMOVE("Hello world!", 5, 6)     ; Resultado = "Hello!")
```

Función TRIMLEFT: eliminación de espacios en blanco a la izquierda del string

Sintaxis:	**TRIMLEFT**(string1)	
Descripción:	TRIMLEFT elimina espacios en blanco a la izquierda de la cadena de caracteres.	
Parámetros:	string1	Cadena de caracteres de la que deben eliminarse espacios en blanco situados a la izquierda

Ejemplo

```
REG[0]=TRIMLEFT("    Hello!")              ; Resultado = "Hello!")
```

Función TRIMRIGHT: eliminación de espacios en blanco a la derecha del string

Sintaxis:	**TRIMRIGHT**(string1)	
Descripción:	TRIMRIGHT elimina espacios en blanco a la derecha de la cadena de caracteres.	
Parámetros:	string1	Cadena de caracteres de la que deben eliminarse espacios en blanco situados a la derecha

Ejemplo

```
REG[0]=TRIMRIGHT("Hello!    ")            ; Resultado = "Hello!")
```

Función FORMAT: inserción de valores o de string con identificador de formato

Sintaxis:	**FORMAT**(texto con identificador de formato [, valor1] ... [, valor28])	
Descripción:	La función FORMAT ofrece la posibilidad, usando identificadores de formato, de insertar hasta 28 valores o strings en determinadas posiciones de un texto predefinido.	
Identificadores de formato:	Sintaxis	%[Flags] [anchura] [.posiciones decimales] Tipo
	Flags	Carácter opcional para establecer el formato de salida: • justificado a la derecha o a la izquierda ("-" para justificado a la izquierda) • Adición de ceros a la izquierda ("0") • default: rellenado con espacios en blanco si el valor por emitir tiene menos posiciones de las indicadas con [anchura]
	Anchura	El argumento define la anchura de salida mínima de un número no negativo. Si el valor tiene menos posiciones de las definidas por el argumento, se completan las posiciones que falten con espacios en blanco.
	Posiciones decimales	Con un número de coma flotante los parámetros opcionales definen el número de posiciones decimales.
	Tipo	El carácter de tipo define los formatos de datos que debe transferir la instrucción de imprimir. Estos caracteres deben especificarse. • d: valor entero • f: Número en coma flotante • s: String • o: octal • x: hexadecimal • b: binario

Ejemplo

```
DEF VAR1
DEF VAR2
LOAD
VAR1 = 123
VAR2 = FORMAT("Hello %08b %.2f %s!", VAR1 + 1, 987.654321, "world")
                                    ; Resultado = "Hello 01111100 987.65 world!"
END_LOAD
```

Consulte también

Uso de strings (Página 106)

8.3.40 Bucles WHILE/UNTIL

Descripción

Con los comandos DO-LOOP es posible implementar un bucle. Según la configuración, este se ejecutará mientras una condición se cumpla (WHILE) o hasta que sea cierta una condición (UNTIL).

Puesto que los bucles pueden afectar al rendimiento del sistema en función de la configuración, deben aplicarse con prudencia y debe prescindirse en ellos de acciones que requieran mucho tiempo.

Se recomienda utilizar como variable de proceso, p. ej., un registro (REG[]), ya que las variables de visualización normales (en particular las ligadas a variables de sistema o de usuario) pueden afectar asimismo al rendimiento del sistema a causa de la alta frecuencia de actualización o de procesos de escritura.

Con ayuda de la función DEBUG (ver capítulo DEBUG (Página 148)) puede determinarse el tiempo de ejecución de los métodos de "Run MyScreens". Esto permite, dado el caso, identificar a través de bucles los problemas que surjan (alta carga de la CPU, tiempo de reacción reducido).

Nota

Puesto que cada bucle FOR puede ser reemplazado por un bucle WHILE, en Easyscreen no se admite la sintaxis para la formulación de un bucle FOR.

Programación

```
DO
   <Instrucciones>
LOOP_WHILE <Condición para que continúe el bucle>
```

```
DO
   <Instrucciones>
LOOP_UNTIL <Condición para finalizar el bucle>

DO_WHILE <Condición para que continúe el bucle>
   <Instrucciones>
LOOP

DO_UNTIL <Condición para finalizar el bucle>
   <Instrucciones>
LOOP
```

Ejemplo

```
REG[0] = 5
DO
   DEBUG("OUTER: " << REG[0])
   REG[0] = REG[0] + 1
   REG[1] = -5

   DO
      DEBUG("INNER: " << REG[1])
      REG[1] = REG[1] + 1
   LOOP_WHILE REG[1] < 0

LOOP_WHILE REG[0] < 10
```

```
REG[0] = 5
DO
   DEBUG("OUTER: " << REG[0])
   REG[0] = REG[0] + 1
   REG[1] = -5

   DO
      DEBUG("INNER: " << REG[1])
      REG[1] = REG[1] + 1
   LOOP_UNTIL 0 <= REG[1]

LOOP_WHILE 10 > REG[0]
```

```
REG[0] = 5
DO_WHILE 10 > REG[0]
   DEBUG("OUTER: " << REG[0])
   REG[0] = REG[0] + 1
   REG[1] = -5

   DO_UNTIL 0 <= REG[1]
     DEBUG("INNER: " << REG[1])
     REG[1] = REG[1] + 1
   LOOP

LOOP
```

```
REG[0] = 5
DO_WHILE 10 > REG[0]
   DEBUG("OUTER: " << REG[0])
   REG[0] = REG[0] + 1
   REG[1] = -5

   DO
     DEBUG("INNER: " << REG[1])
     REG[1] = REG[1] + 1
   LOOP_UNTIL 0 <= REG[1]

LOOP
```

8.3.41 Ejecución cíclica de scripts: START_TIMER, STOP_TIMER

Descripción

Con ayuda de temporizadores se pueden abrir cíclicamente los métodos SUB. Para ello existen las funciones START_TIMER() y STOP_TIMER().

Nota

Solo se puede configurar un temporizador por método SUB.

Programación

Sintaxis:	**START_TIMER**("Nombre de SUB", intervalo)	
Descripción:	Inicio del procesamiento cíclico de un método SUB	
Parámetros:	Nombre de SUB:	Nombre del método SUB al que se debe acceder cíclicamente
	Intervalo:	Intervalo en milisegundos

Sintaxis:	**STOP_TIMER**("Nombre de SUB")	
Descripción:	Fin del procesamiento cíclico de un método SUB	
Parámetros:	Nombre de SUB:	Nombre del método SUB cuyo temporizador debe detenerse

Ejemplo

```
//M(TimerSample/"My timer")
DEF MyVariable=(I//0/,"Number of cyclic calls:"/WR1)
VS1=("Start%ntimer")
VS2=("Stop%ntimer")

SUB(MyTimerSub)
   MyVariable = MyVariable + 1
END_SUB

PRESS(VS1)
   ;llama a SUB "MyTimerSub" cada 1000 milisegundos
   START_TIMER("MyTimerSub", 1000)
END_PRESS

PRESS(VS2)
   STOP_TIMER("MyTimerSub")
END_PRESS
```

Si se accede de nuevo a START_TIMER para un temporizador ya asignado a un método SUB, después se toma el nuevo intervalo si es diferente. De lo contrario, se ignora esta segunda llamada.

El intervalo más pequeño es de 100 milisegundos y viene determinado por el sistema.

Si se accede a STOP_TIMER para un método SUB para el que actualmente no se ejecuta ningún temporizador, se ignora esta llamada.

Elementos gráficos y lógicos

9

9.1 Línea, línea de separación, rectángulo, círculo y elipse

9.1.1 Definir elementos gráficos

Descripción

Las líneas, las líneas de separación, los rectángulos y las elipses/los círculos se configuran en el método LOAD:

- Para conseguir rectángulos transparentes debe establecerse como color de relleno el color de fondo del sistema.
- Con el elemento ELLIPSE se forma un círculo definiendo altura y anchura iguales.
- Las líneas de separación horizontales y verticales tienen siempre la anchura o altura exacta de la ventana. Las líneas de separación no dan lugar a barras de desplazamiento.
  Si para representar las líneas de separación se ha usado hasta ahora el elemento RECT, p. ej. RECT(305,0,1,370,0,0,1), la ayuda a la programación lo reconoce y se convierte automáticamente en V_SEPARATOR(305,1,"#87a5cd", 1). Esto se aplica tanto a líneas de separación verticales como horizontales.

Elemento LINE

Programación:

Sintaxis:	LINE (x1, y1, x2, y2, color, style, tag, visible)	
Descripción:	Definir línea	
Parámetros:	x1	Coordenada x del punto inicial
	y1	Coordenada y del punto inicial
	x2	Coordenada x del punto final
	y2	Coordenada y del punto final
	color	Color de la línea
	style	Estilo de la línea: 1 = continuo 2 = con trazos 3 = punteado 4 = con trazos y puntos
	tag	Mediante este tag y las funciones SHOW_GRAPHIC, HIDE_GRAPHIC y DELETE_GRAPHIC se puede mostrar/ocultar o bien borrar un elemento gráfico o un grupo de elementos gráficos.
	visible	Elemento gráfico visible (TRUE/FALSE)

Elemento RECT

Programación:

Sintaxis:	RECT (x, y, w, h, frame color, fill color, style, tag, visible)	
Descripción:	Definir rectángulo	
Parámetros:	x	Coordenada x arriba izquierda
	y	Coordenada y arriba izquierda
	w	Anchura
	h	Altura
	frame color	Color del marco
	fill color	Color de relleno
	style	Estilo del marco: 1 = continuo 2 = con trazos 3 = punteado 4 = con trazos y puntos
	tag	Mediante este tag y las funciones SHOW_GRAPHIC, HIDE_GRAPHIC y DELETE_GRAPHIC se puede mostrar/ocultar o bien borrar un elemento gráfico o un grupo de elementos gráficos.
	visible	Elemento gráfico visible (TRUE/FALSE)

Elemento ELLIPSE

Programación:

Sintaxis:	ELLIPSE(x, y, w, h, frame color, fill color, style, tag, visible)	
Descripción:	Definición de una elipse o un círculo	
Parámetros:	x	Coordenada x arriba izquierda
	y	Coordenada y arriba izquierda
	w	Anchura
	h	Altura
	frame color	Color del marco
	fill color	Color de relleno
	style	Estilo del marco: 1 = continuo 2 = con trazos 3 = punteado 4 = con trazos y puntos
	tag	Mediante este tag y las funciones SHOW_GRAPHIC, HIDE_GRAPHIC y DELETE_GRAPHIC se puede mostrar/ocultar o bien borrar un elemento gráfico o un grupo de elementos gráficos.
	visible	Elemento gráfico visible (TRUE/FALSE)

Con el mismo valor para la altura y la anchura se forma un círculo.

Elemento V_SEPARATOR

Programación:

Sintaxis:	V_SEPARATOR (x,w,color,pen)	
Descripción:	Definición de línea de separación vertical	
Parámetros:	x	Posición X
	pen width	Grosor de línea
	color	Color
	style	Estilo de la línea: 1 = continuo 2 = con trazos 3 = punteado 4 = con trazos y puntos
	tag	Mediante este tag y las funciones SHOW_GRAPHIC, HIDE_GRAPHIC y DELETE_GRAPHIC se puede mostrar/ocultar o bien borrar un elemento gráfico o un grupo de elementos gráficos.
	visible	Elemento gráfico visible (TRUE/FALSE)

Elemento H_SEPARATOR

Programación:

Sintaxis:	H_SEPARATOR (y,h,color,pen)	
Descripción:	Definición de línea de separación horizontal	
Parámetros:	y	Posición Y
	pen width	Grosor de línea
	color	Color
	style	Estilo de la línea: 1 = continuo 2 = con trazos 3 = punteado 4 = con trazos y puntos
	tag	Mediante este tag y las funciones SHOW_GRAPHIC, HIDE_GRAPHIC y DELETE_GRAPHIC se puede mostrar/ocultar o bien borrar un elemento gráfico o un grupo de elementos gráficos.
	visible	Elemento gráfico visible (TRUE/FALSE)

Información adicional

En el apartado Funciones gráficas (Página 196) se describen las funciones SHOW_GRAPHIC, HIDE_GRAPHIC, CLEAR_BACKGROUND y DELETE_GRAPHIC. Estas funciones permiten controlar la visualización de los elementos gráficos.

9.1.2 Funciones gráficas

Sinopsis

Las siguientes funciones gráficas se encuentran disponibles:

- CLEAR_BACKGROUND
- DELETE_GRAPHIC
- HIDE_GRAPHIC
- SHOW_GRAPHIC

Las funciones se pueden utilizar en los elementos gráficos LINE, RECT, ELLIPSE, V_SEPARATOR y H_SEPARATOR (Página 193).

Función CLEAR_BACKGROUND: borrar elementos gráficos

Con la función CLEAR_BACKGROUND se pueden borrar los elementos gráficos.

Sintaxis:	CLEAR_BACKGROUND()	
Descripción:	Borrar elementos gráficos. La memoria ocupada por los objetos gráficos se libera.	
Parámetro:	- Ninguno -	

Función DELETE_GRAPHIC: borrar un elemento gráfico o un grupo de elementos gráficos

Con la función DELETE_GRAPHIC se puede borrar un elemento gráfico específico o un grupo de elementos gráficos. En los elementos gráficos correspondientes debe definirse un valor en el parámetro "tag".

- Con respecto a la función CLEAR_BACKGROUND se pueden borrar elementos gráficos específicos.
- Con respecto a la función HIDE_GRAPHIC se libera la memoria ocupada por los objetos gráficos.

Sintaxis:	**DELETE_GRAPHIC**(tag)	
Descripción:	Borrar un elemento gráfico o un grupo de elementos gráficos cuyo valor es idéntico en el parámetro "tag".	
Parámetro:	tag	El parámetro "tag" puede definirse en los elementos gráficos. También es posible definir varios elementos gráficos con el mismo valor y, con ello, formar un grupo de elementos gráficos.

Ejemplo

```
LINE(0, 10, 100, 10, "black", "dotted", 1, true)
...
LINE(0, 10, 100, 10, "red", "dotted", 5, true)
LINE(0, 20, 100, 20, "red", "dotted", 5, true)
```

```
LINE(0, 30, 100, 30, "red", "dotted", 5, true)
...
DELETE_GRAPHIC(5)                                    ;los 3 elementos gráfi-
                                                     cos del tipo LINE con
                                                     tag=5 se borran
```

Función HIDE_GRAPHIC: ocultar un elemento gráfico o un grupo de elementos gráficos

Con la función HIDE_GRAPHIC se puede ocultar un elemento gráfico específico o un grupo de elementos gráficos. En los elementos gráficos correspondientes debe definirse un valor en el parámetro "tag".

Sintaxis:	HIDE_GRAPHIC(tag)	
Descripción:	Ocultar un elemento gráfico o un grupo de elementos gráficos cuyo valor es idéntico en el parámetro "tag".	
Parámetro:	tag	El parámetro "tag" puede definirse en los elementos gráficos. También es posible definir varios elementos gráficos con el mismo valor y, con ello, formar un grupo de elementos gráficos.

Ejemplo

```
LINE(0, 10, 100, 10, "black", "dotted", 1, true)
...
LINE(0, 10, 100, 10, "red", "dotted", 5, true)
LINE(0, 20, 100, 20, "red", "dotted", 5, true)
LINE(0, 30, 100, 30, "red", "dotted", 5, true)
...
HIDE_GRAPHIC(5)                                      ;los 3 elementos gráfi-
                                                     cos del tipo LINE con
                                                     tag=5 se ocultan
```

Función SHOW_GRAPHIC: mostrar un elemento gráfico o un grupo de elementos gráficos

Con la función SHOW_GRAPHIC se puede mostrar un elemento gráfico específico o un grupo de elementos gráficos. En los elementos gráficos correspondientes debe definirse un valor en el parámetro "tag".

Sintaxis:	SHOW_GRAPHIC(tag)	
Descripción:	Mostrar un elemento gráfico o un grupo de elementos gráficos cuyo valor es idéntico en el parámetro "tag".	
Parámetro:	tag	El parámetro "tag" puede definirse en los elementos gráficos. También es posible definir varios elementos gráficos con el mismo valor y, con ello, formar un grupo de elementos gráficos.

Ejemplo

```
LINE(0, 10, 100, 10, "black", "dotted", 1, true)
...
LINE(0, 10, 100, 10, "red", "dotted", 5, true)
LINE(0, 20, 100, 20, "red", "dotted", 5, true)
LINE(0, 30, 100, 30, "red", "dotted", 5, true)
...
SHOW_GRAPHIC(5)                                          ;los 3 elementos gráfi-
                                                         cos del tipo LINE con
                                                         tag=5 se muestran
```

9.2 Definir matriz

Definición

Con la ayuda de una matriz, los datos de un tipo de datos uniforme se guardan en la memoria ordenados de manera a permitir el acceso a los datos a través de un índice.

Descripción

Las matrices pueden ser unidimensionales o bidimensionales. Una matriz unidimensional puede considerarse como una matriz bidimensional con una fila o una columna.

Las matrices se definen con la identificación //A y se cierran con //END. La cantidad de filas y columnas es arbitraria. Una matriz tiene la siguiente estructura:

Programación

Sintaxis:	**//A**(Identificador) (a/b...) (c/d...) ... //END	
Descripción:	Definir matriz	
Parámetro:	Identificador	Nombre de la matriz
	a, b, c, d	Valores de la matriz Los valores de tipo STRING deben indicarse entre comillas dobles.

Ejemplo

```
//A(Rosca)                    ; Tamaño/Paso/Diámetro interior
```

```
(0.3 / 0.075 / 0.202)
(0.4 / 0.1   / 0.270)
(0.5 / 0.125 / 0.338)
(0.6 / 0.15  / 0.406)
(0.8 / 0.2   / 0.540)
(1.0 / 0.25  / 0.676)
(1.2 / 0.25  / 0.676)
(1.4 / 0.3   / 1.010)
(1.7 / 0.35  / 1.246)
//END
```

9.2.1 Acceder al valor de un elemento de matriz

Descripción

El valor del acceso a una matriz se puede transmitir mediante la propiedad valor (Identificador.val).

El índice de filas (número de la fila en la matriz) o el índice de columnas (número de la columna en la matriz) comienza respectivamente con 0. Si un índice de fila o de columna señala fuera de la matriz, se muestra el valor 0 o una cadena vacía y la variable ERR tiene el valor TRUE. La variable ERR también es TRUE cuando no se encuentra el término buscado.

Programación

Sintaxis:	Identificador [Z,[M[,C]]].val o Identificador [Z,[M[,C]]]
Descripción:	Acceso a una matriz unidimensional con una única columna
Sintaxis:	Identificador [S,[M[,C]]].val] o Identificador[S,[M[,C]]] o
Descripción:	Acceso a una matriz unidimensional con una única fila
Sintaxis:	Identificador [Z,S,[M[,C]]].val o Identificador [Z,S,[M[,C]]]
Descripción:	Acceso a una matriz bidimensional

Parámetro:	Identificador:	Nombre de la matriz	
	Z:	Valor de las filas (índice de filas o término buscado)	
	S:	Valor de las columnas (índice de columnas o término buscado)	
	M:	Modo de acceso	
		0	Directo
		1	búsqueda para filas, directo para columnas
		2	directo para filas, búsqueda para columnas
		3	Búsqueda
		4	Buscar índice de fila
		5	Buscar índice de columna
	C:	Modo de comparación	
		0	el término buscado se tiene que encontrar en el margen de valores de la línea o columna.
		1	el término buscado debe encontrarse exactamente.

Ejemplo:		
	`VAR1 = MET_G[REG[3],1,0].VAL`	`; Asignar a Var1 un valor de la matriz MET_G`

Modo de acceso

- **Modo de acceso "Directo"**
  En el modo de acceso "Directo" (M = 0) el acceso a la matriz se efectúa con el índice de fila en F y el índice de columna en C. El modo de comparación P no se evalúa.
- **Modo de acceso "Búsqueda"**
  En el modo de acceso M = 1, 2 o 3 la búsqueda se efectúa siempre en la fila 0 o la columna 0.

Modo M	Valor de fila F	Valor de columna C	Valor de salida
0	Índice de fila	Índice de columna	Valor de la fila F y la columna C
1	Elemento de búsqueda: Se busca en la columna 0	Índice de la columna de la cual se lee el valor	Valor de la fila encontrada y la columna C
2	Índice de la fila de la cual se lee el valor de retorno	Elemento de búsqueda: Se busca en la fila 0	Valor de la fila F y la columna encontrada
3	Elemento de búsqueda: Se busca en la columna 0	Elemento de búsqueda: Se busca en la fila 0	Valor de la fila y la columna encontradas
4	Elemento de búsqueda: Se busca en la columna C	Índice de la columna en la que se busca	Índice de fila
5	Índice de la fila en la que se busca	Elemento de búsqueda: Se busca en la fila F	Índice de columna

Modo de comparación

Al emplear el modo de comparación P = 0 el contenido de la fila o la columna de búsqueda debe estar clasificado en orden ascendente. Si el término buscado es menor que el primer elemento o mayor que el último, se muestra el valor 0 o un string vacío y la variable de error ERR es TRUE.

Al emplear el modo de comparación P = 1 el término buscado debe encontrarse en la fila o columna de búsqueda. Si no se encuentra el término buscado, se muestra el valor 0 o un string vacío y la variable de error ERR es TRUE.

9.2.2 Ejemplo: Acceso a un elemento de matriz

Requisitos

A continuación se definen dos matrices que son necesarias para los ejemplos siguientes:

```
//A(Rosca)
                (0.3 / 0.075 / 0.202)
                (0.4 / 0.1   / 0.270)
                (0.5 / 0.125 / 0.338)
                (0.6 / 0.15  / 0.406)
                (0.8 / 0.2   / 0.540)
                (1.0 / 0.25  / 0.676)
                (1.2 / 0.25  / 0.676)
                (1.4 / 0.3   / 1.010)
                (1.7 / 0.35  / 1.246)
//END


//A(Matriz2)
                ("IND" /     "PAS" /     "DIA")
                (0.3 /       0.075 /     0.202 )
                (0.4 /       0.1 /       0.270 )
                (0.5 /       0.125 /     0.338 )
                (0.6 /       0.15 /      0.406 )
                (0.8 /       0.2 /       0.540 )
                (1.0 /       0.25 /      0.676 )
                (1.2 /       0.25 /      0.676 )
                (1.4 /       0.3 /       1.010 )
                (1.7 /       0.35 /      1.246 )
//END
```

Ejemplos

- **Modo de acceso ejemplo 1:**
  En F se encuentra el término buscado. Este término siempre se busca en la columna 0. Se muestra el valor de la columna C con el índice de fila del término encontrado:
  ```
  VAR1 = rosca[0.5,1,1]   ;VAR1 tiene el valor 0.125
  ```
  Aclaración:
  Se busca el valor 0.5 en la columna 0 de la matriz "Rosca" y se devuelve el valor encontrado en la columna 1 de la misma fila.
- **Modo de acceso ejemplo 2:**
  En C se encuentra el término buscado. Este término siempre se busca en la fila 0. Se muestra el valor de la fila F con el índice de columna del término encontrado:
  ```
  VAR1 = MATRIZ2[3,"PAS",2]   ;VAR1 tiene el valor 0.125
  ```
  Aclaración:
  Se busca en la fila 0 de la matriz "Matriz2" la columna que contiene "PAS". Se devuelve el valor de la columna encontrada y la fila con el índice 3.
- **Modo de acceso ejemplo 3:**
  Tanto en F como en C hay un término buscado. Con el término en F se busca en la columna 0 el índice de fila y con el término en C se busca en la fila 0 el índice de columna. Con los índices de fila y columna encontrados se muestra el valor de la matriz:
  ```
  VAR1 = MATRIZ2[0.6,"PAS",3]   ;VAR1 tiene el valor 0.15
  ```
  Aclaración:
  Se busca en la columna 0 de la matriz "Matriz2" la fila que contiene 0.6, y se busca en la fila 0 de la "Matriz2" la columna que contiene "PAS". Se devuelve el valor de la fila y la columna encontradas según VAR1.
- **Modo de acceso ejemplo 4:**
  En F se encuentra el término buscado. En C se encuentra el índice de la columna en la que se busca. Se muestra el índice de fila del término encontrado:
  ```
  VAR1 = rosca[0.125,1,4]   ;VAR1 tiene el valor 2
  ```
  Aclaración:
  Se busca el valor 0.125 en la columna 1 de la matriz "Rosca" y se devuelve el índice de fila del valor encontrado según VAR1.
- **Modo de acceso ejemplo 5:**
  En F se encuentra el índice de la fila en la que se busca. El término buscado se encuentra en C. Se muestra el índice de columna del término encontrado:
  ```
  VAR1 = rosca[4,0.2,5,1]  ;VAR1 tiene el valor 1
  ```
  Aclaración:
  Se busca el valor 0.2 en la fila 4 de la matriz "Rosca" y se devuelve el índice de columna del valor encontrado según VAR1. Se ha elegido el modo de comparación 1, ya que los valores de la fila 4 no están clasificados en orden ascendente.

9.2.3 Consultar el estado de un elemento matriz

Descripción

Con la propiedad estado puede consultarse si el acceso a una matriz devuelve un valor válido.

Programación

Sintaxis:	Identificador [Z, S, [M[,C]]]**.vld**
Descripción:	Esta propiedad es de sólo lectura.
Parámetro:	Identificador Nombre de la matriz
Valor de retorno:	FALSE = Valor no válido
	TRUE = Valor válido

Ejemplo

```
DEF MPIT = (R///"MPIT",,"MPIT",""/wr3)
DEF PIT  = (R///"PIT",,"PIT",""/wr3)
PRESS(VS1)
  MPIT = 0.6
  IF MET_G[MPIT,0,4,1].VLD == TRUE
    PIT    = MET_G[MPIT,1,0].VAL
    REG[4] = PIT
    REG[1] = "OK"
  ELSE
    REG[1] = "ERROR"
  ENDIF
END_PRESS
```

9.3 Descripción de tabla (grid)

Definición

A diferencia de una matriz, los valores de una tabla (grid) se actualizan constantemente. Se trata de una representación tabular de valores que pueden proceder, p. ej. del CN o del PLC. La tabla se organiza en columnas que contienen siempre variables del mismo tipo.

Asignación

La remisión a una descripción de tabla se lleva a cabo en la descripción de las variables:

- La definición de variables determina los valores que deben mostrarse; la definición de los elementos de tabla determina la apariencia y disposición en la pantalla. Las propiedades de los campos de entrada/salida de la línea de definición de las variables se adoptan en la tabla.
- El área visible de la tabla se determina mediante la anchura y altura del campo de entrada/salida. Si hay más filas o columnas de las que caben en el área visible, se puede desplazar en sentido vertical y horizontal.

Identificador de tablas

Identificador de una tabla del mismo tipo de valores que el NCK o del PLC, a los que se puede asignar un canal mediante bloques. El identificador de tablas se distingue de los valores límite o de los campos de alternancia mediante un signo % antepuesto. Al identificador de tablas puede seguir, separado mediante coma, un nombre de fichero que indica el fichero en el que está definida la descripción de la tabla.

Identificador de una tabla del mismo tipo de valores, p. ej., procedentes de NCK o PLC. El identificador de tablas se introduce en la posición en la que se configura la indicación de valores límite o del campo de alternancia. El identificador de tablas comienza con un caracter % antepuesto. Después puede seguirle un nombre de fichero, separado por una coma. Este indica el fichero en el que está definida la descripción de la tabla. La tabla se busca de forma predeterminada en el fichero de configuración en el que está configurada la máscara.

Una definición de tabla también puede cargarse de manera dinámica, p. ej. en el método LOAD, mediante la función Load Grid (LG).

Variable de sistema o de usuario

Este parámetro no toma ningún valor para tablas, ya que las variables que deben mostrarse se indican de manera detallada en las líneas de definición de las columnas. La descripción de tablas puede realizarse de forma dinámica.

Ejemplo

Definición de una variable MyGridVar que representa un grid "MyGrid1" en su campo de entrada/salida (distancia desde la izquierda: 100, distancia desde arriba: estándar, anchura: 350, altura: 100).

```
DEF MyGridVar=(V/% MyGrid1///////100,,350,100)
```

Ver también

Parámetros de variables (Página 93)

Load Grid (LG) (Página 165)

9.3.1 Definición de tabla (grid)

Descripción

El bloque de tablas consta de:

- Encabezamiento descriptivo
- De 1 a n columnas descriptivas

Programación

Sintaxis:	**//G**(Identificador de tablas/tipo de tabla/número de filas/ [atributo fila fija],[atributo columna fija])		
Descripción:	Definición de tabla (grid)		
Parámetro:	Identificador de tablas	En este caso el identificador de tablas se utiliza sin anteponer el signo %. Únicamente puede utilizarse una vez en un diálogo.	
	Tipo de tabla	0 (ajuste predeterminado)	Tabla para datos PLC o de usuario (datos NCK y específicos del canal)
		1	y otros reservados
	Líneas	Cantidad de filas incluida la de encabezado	
		La fila o columna fija no se desplaza. La cantidad de columnas es el resultado del número de columnas configuradas.	
	Visualización de la fila de título de las columnas	-1:	No se visualizará la fila de título de las columnas.
		0:	Se visualizará la fila de título de las columnas (por defecto)
	Número de columnas fijas	0:	Ninguna columna fija
		1 ... n:	Número de columnas que deben estar siempre visibles; es decir, no se desplazan en horizontal

Ejemplos

`//G(grid1/0/5/-1,2)`	Fila de título de las columnas oculta y 2 filas fijas
`//G(grid1/0/5/,1)`	Fila de título de las columnas mostrada y 1 columna fija
`//G(grid1/0/5)`	Fila de título de las columnas mostrada y sin filas fijas

9.3.2 Definir columnas

Descripción

Para las tablas (grid) conviene utilizar variables con índice. El número de índice es importante en el caso de variables de PLC o CN con uno o más índices.

Pueden modificarse de manera directa los valores mostrados en una tabla en el marco de los derechos determinados mediante los atributos y dentro de los límites definidos dado el caso.

Programación

Sintaxis:	(Tipo/valores límite/vacío/texto largo, título de columna/atributos/pantalla de ayuda/ variable de sistema o de usuario/ancho de columnas/Offset1, Offset2, Offset3) Ver también Sintaxis de configuración ampliada (Página 47).	
Descripción:	Definir columnas	
Parámetro:	por analogía a las variables	
	Tipo	Tipo de datos
	Valores límite	Valor límite MÍN, valor límite MÁX
	Texto largo, título de columna	
	Atributos	
	Pantalla de ayuda	
	Variable de sistema o de usuario	Como variables deben introducirse variables de PLC o CN entre comillas.
	Ancho de columnas	Indicación en píxeles
	Offset (Decalajes)	Los incrementos de consigna en los que debe aumentar el respectivo índice para rellenar las columnas se indica en el parámetro de offset asignado. • Offset1: Incremento de paso para el 1.er índice • Offset2: Incremento de paso para el 2.º índice • Offset3: Incremento de paso para el 3.er índice

Título de columna de un fichero de texto

El título de columna puede indicarse como texto o como número ($8xxxx) y, dado el caso, no se desplazará.

Modificación de las propiedades de las columnas

Las propiedades de columna que son modificables dinámicamente (tienen acceso de escritura) son:

- Valores límite (mín., máx.)
- Título de columna (st)
- Atributos (wr, ac y li)
- Pantalla de ayuda (hlp)

La modificación de una propiedad de columna se lleva a cabo mediante el identificador de la variable de la línea de definición y mediante el índice de la columna (que comienza con 1).

Ejemplo: `VAR1[1].st="Columna 1"`

Para la definición de columna se pueden indicar los atributos wr, ac y li.

Ver también

Load Grid (LG) (Página 165)

9.3.3 Control del foco en la tabla (grid)

Descripción

Con las propiedades Row y Col el foco puede activarse y calcularse dentro de una tabla:

- Identificador.**Row**
- Identificador.**Col**

Programación

Cada celda de una tabla posee las propiedades Val y Vld.

Para la lectura y escritura de las propiedades de las celdas debe indicarse un índice de fila y columna junto al identificador de la variable de la línea de definición.

Sintaxis:	Indicador[índice de fila, índice de columna].Val o Identificador[índice de fila, índice de columna]
Descripción:	Propiedades Val
Sintaxis:	Indicador[índice de fila, índice de columna].Vld
Descripción:	Propiedades Vld

Ejemplo

```
Var1[2,3].val=1.203
```

Si no se indica ningún índice de fila ni de columna, se aplican los índices de la celda enfocada. Esto corresponde a:

```
Var1.Row=2
Var1.Col=3
Var1.val=1.203
```

9.4 Widgets personalizados

9.4.1 Definir widgets personalizados

Descripción

A través de un widget personalizado se configuran elementos de visualización específicos del usuario en el diálogo.

Opción de software

Para utilizar widgets personalizados en los diálogos, se necesita además la siguiente opción de software:

"SINUMERIK Integrate Run MyHMI/3GL" (6FC5800-0AP60-0YB0)

Programación

Definición:	**DEF** (Nombre)	
Sintaxis:	**(W///"","**(Nombre de la librería)**.**(Nombre de la clase)**"////**a,b,c,d**);**	
Descripción:	W	Definir widget personalizado
Parámetro:	Name	Nombre del widget personalizado, de libre elección
	Nombre de la librería	Nombre del fichero de librería dll (Windows) o so (Linux), de libre elección
	Nombre de la clase	Nombre de la función de clase de la librería antes mencionada, de libre elección
	a, b, c, d	Posición y tamaño de la configuración

Ejemplo

Un widget personalizado se define en la configuración de diálogo de la manera siguiente:

```
DEF Cus = (W///"","slestestcustomwidget.SlEsTestCustomWidget"/////
20,20,250,100);
```

9.4.2 Estructura de la librería de widgets personalizados

Descripción

La librería de widgets personalizados contiene principalmente una clase definida. El nombre de esta clase debe indicarse en la configuración de diálogo junto al nombre de la librería. A partir del nombre de la librería, "Run MyScreens" accede a un fichero dll del mismo nombre, p. ej.:

```
slestestcustomwidget.dll
```

Programación

La definición de clase del fichero dll debe ser así:

```
#define SLESTESTCUSTOMWIDGET_EXPORT Q_DECL_EXPORT

class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget  : public QWidget
{
   Q_OBJECT
....
public slots:
   bool serialize(const QString& szFilePath, bool bIsStoring);
....
}
```

9.4.3 Estructura de la interfaz de widgets personalizados

Descripción

Para poder mostrar el widget personalizado en el diálogo, la librería se completa con una interfaz. Esta contiene definiciones de macros con las que "Run MyScreens" inicia el widget personalizado. La interfaz tiene el formato de un fichero cpp. El nombre del fichero se puede elegir libremente, p. ej.:
`sleswidgetfactory.cpp`

Programación

La interfaz se define del modo siguiente:

```
#include "slestestcustomwidget.h"      ; El fichero header (de cabecera) del widget
                                       personalizado correspondiente se inserta al
                                       inicio del fichero
....
//Makros                               ; Las definiciones de macros no se modifican
....
WIDGET_CLASS_EXPORT(SlEsTestCustom-    ; El widget personalizado en cuestión se de-
Widget)                                clara al final del fichero
```

Ejemplo

Contenido del fichero sleswidgetfactory.cpp para un widget personalizado con el nombre de clase "SlEsTestCustomWidget":

```
#include <Qt/qglobal.h>
#include "slestestcustomwidget.h"

////////////////////////////////////////////////////////////////////////
// MAKROS FOR PLUGIN DLL-EXPORT - DO NOT CHANGE
////////////////////////////////////////////////////////////////////////

#ifndef Q_EXTERN_C
#ifdef __cplusplus
#define Q_EXTERN_C    extern "C"
#else
#define Q_EXTERN_C    extern
#endif
#endif

#define SL_ES_FCT_NAME(PLUGIN) sl_es_create_ ##PLUGIN
#define SL_ES_CUSTOM_WIDGET_PLUGIN_INSTANTIATE( IMPLEMENTATION , PARAM) \
    { \
    IMPLEMENTATION *i = new PARAM; \
    return i; \
    }

#ifdef Q_WS_WIN
#   ifdef Q_CC_BOR
#       define EXPORT_SL_ES_CUSTOM_WIDGET_PLUGIN(PLUGIN,PARAM) \
    Q_EXTERN_C __declspec(dllexport) void* \
    __stdcall SL_ES_FCT_NAME(PLUGIN) (QWidget* pParent) \
    SL_ES_CUSTOM_WIDGET_PLUGIN_INSTANTIATE( PLUGIN,PARAM )
#   else
#       define EXPORT_SL_ES_CUSTOM_WIDGET_PLUGIN(PLUGIN,PARAM) \
    Q_EXTERN_C __declspec(dllexport) void* SL_ES_FCT_NAME(PLUGIN) \
    (QWidget* pParent) \
    SL_ES_CUSTOM_WIDGET_PLUGIN_INSTANTIATE( PLUGIN,PARAM )
#   endif
#else
#   define EXPORT_SL_ES_CUSTOM_WIDGET_PLUGIN(PLUGIN,PARAM) \
    Q_EXTERN_C void* SL_ES_FCT_NAME(PLUGIN) (QWidget* pParent) \
    SL_ES_CUSTOM_WIDGET_PLUGIN_INSTANTIATE( PLUGIN,PARAM )
#endif
```

```
#define WIDGET_CLASS_EXPORT(CLASSNAME)  \
    EXPORT_SL_ES_CUSTOM_WIDGET_PLUGIN(CLASSNAME,CLASSNAME(pParent))

////////////////////////////////////////////////////////////////////////
// FOR OEM USER - please declare here your widget classes for export
////////////////////////////////////////////////////////////////////////

WIDGET_CLASS_EXPORT(SlEsTestCustomWidget)
```

9.4.4 Interacción entre widget personalizado y diálogo: intercambio de datos automático

Los widgets personalizados interactúan con los diálogos y pueden mostrar o manipular valores.

Condiciones

Se produce un intercambio de datos automático si se cumplen las siguientes condiciones:

Condición	Dirección
Al iniciar o decompilar un diálogo	Diálogo → widget personalizado
Al ejecutar el comando GC para generar llamadas de ciclos	Widget personalizado → diálogo

Programación

Para las interacciones son necesarias las siguientes definiciones:

Ampliación de la configuración de diálogo

Definición:	**DEF** (Variable)	
Sintaxis:	**(**(tipo)**//5/"","**(variable)**","/wr2/)**	
Tipo de variable:	Tipo	Campo de entrada estándar (no grid ni campo de alternancia) con cualquier tipo de datos (no W)
Parámetro:	Variable	Cualquier denominación de una variable para el intercambio de datos
Modo de entrada:	wr2	Lectura y escritura

Ejemplo

```
DEF CUSVAR1 = (R//5/"","CUSVAR1","/wr2/)
```

Ampliación de la definición de clase

En la definición de clase del widget personalizado debe crearse una QProperty cuyo nombre sea idéntico a la variable seleccionada de la configuración de diálogo, p. ej.:

```
Q_PROPERTY(double CUSVAR1 READ cusVar1 WRITE setCusVar1);
```

Ejemplo

La definición de clase del fichero dll debe ser así:

```
#define SLESTESTCUSTOMWIDGET_EXPORT Q_DECL_EXPORT


class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget  : public QWidget
{
 Q_OBJECT
 Q_PROPERTY(double CUSVAR1  READ cusVar1    WRITE setCusVar1);
....
....
}
```

9.4.5 Interacción entre widget personalizado y diálogo: intercambio de datos manual

Además de automático, el intercambio de datos también puede ser manual. Los datos se intercambian de forma dinámica, es decir, en el tiempo de ejecución del diálogo. Son posibles las siguientes acciones:

- Las propiedades del widget personalizado pueden leerse y escribirse.
- Los métodos del widget personalizado pueden llamarse desde la configuración de Run MyScreens.
- Es posible reaccionar a una determinada señal del widget personalizado para llamar subprogramas (SUB) en la configuración de Run MyScreens.

9.4.5.1 Lectura y escritura de propiedades

Descripción

Para leer y escribir propiedades del widget personalizado se dispone de las funciones ReadCWProperties y WriteCWProperties en la configuración de Run MyScreens.

Programación

Sintaxis:	**ReadCWProperty**("Nombre de variable", "nombre de propiedad")
Descripción:	Leer una propiedad de un widget personalizado

Parámetro:	Nombre de variable	Nombre de la variable diálogo asignada a un widget personalizado
	Nombre de propiedad	Nombre de la propiedad del widget personalizado que desea leerse
Valor de retorno:	Valor actual de la propiedad del widget personalizado	

Sintaxis:	**WriteCWProperty**("Nombre de variable", "nombre de propiedad", "valor")	
Descripción:	Escribir una propiedad de un widget personalizado	
Parámetro:	Nombre de variable	Nombre de la variable diálogo asignada a un widget personalizado
	Nombre de propiedad	Nombre de la propiedad del widget personalizado que desea escribirse
	Valor	Valor que debe escribirse en la propiedad del widget personalizado

Ejemplos

Ejemplo 1:

Lectura de la propiedad "MyStringVar" del widget personalizado conectado con la variable diálogo "MyCWVar1" y asignación del valor al registro 7.

Widget personalizado, declaración de clase:

```
class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget  : public QWidget
{
   Q_OBJECT
   Q_PROPERTY(QString MyStringVar
              READ myStringVar
              WRITE setMyStringVar);
   ...
```

Configuración de diálogo:

```
DEF MyCWVar1 = (W///,"slestestcustomwidget.SlEsTestCustomWidget")
PRESS(VS1)
   REG[7]=ReadCWProperty("MyCWVar1", "MyStringVar")
END_PRESS
```

Ejemplo 2:

Escritura del resultado del cálculo "3 + sin(123.456)" en la propiedad "MyRealVar" del widget personalizado conectado con la variable diálogo "MyCWVar1".

Widget personalizado, declaración de clase:

```
class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget  : public QWidget
{
   Q_OBJECT
   Q_PROPERTY(double MyRealVar
              READ myRealVar
              WRITE setMyRealVar);
   ...
```

Configuración de diálogo:

```
DEF MyCWVar1 = (W///,"slestestcustomwidget.SlEsTestCustomWidget")
PRESS(VS1)
   WriteCWProperty("MyCWVar1", "MyRealVar", 3 + sin(123.456))
END_PRESS
```

9.4.5.2 Ejecución de un método del widget personalizado

Descripción

Para ejecutar métodos del widget personalizado se dispone de la función CallCWMethod en la configuración de Run MyScreens.

El método del widget personalizado que desee llamarse no debe tener más de 10 parámetros de transferencia.

Se soportan los siguientes formatos de datos de los parámetros de transferencia:

- bool
- uint
- int
- double
- QString
- QByteArray

Programación

Sintaxis:	**CallCWMethod**("nombre de variable", "nombre de método[, argumento 0][, argumento 1 ... [,argumento 9]")
Descripción:	Llamar método del widget personalizado

Parámetro:	Nombre de variable	Nombre de la variable diálogo asignada a un widget personalizado
	Nombre de método	Nombre del método del widget personalizado que desea llamarse
	Argumento 0 - 9	Parámetro de transferencia para el método del widget personalizado Formatos de datos soportados: ver arriba **Nota:** Los parámetros de transferencia siempre se transfieren "ByVal", lo que significa que siempre se transfiere el valor solamente y no, p. ej., la referencia a una variable.
Valor de retorno:	Valor de retorno del método del widget personalizado Se soportan los siguientes formatos de datos de los parámetros de transferencia: • void • bool • uint • int • double • QString • QByteArray **Nota:** Incluso si el formato de datos del valor de retorno del método del widget personalizado es "void", dicho valor de retorno debe asignarse formalmente, p. ej., a una variable.	

Ejemplo

Widget personalizado, declaración de clase:

```
class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget  : public QWidget
{
   Q_OBJECT
   public slots:
       void myFunc1(int nValue, const QString& szString, double dValue);
...
```

Configuración de diálogo:

```
DEF MyCWVar1 = (W///,"slestestcustomwidget.SlEsTestCustomWidget")
DEF MyStringVar1 = (S)
DEF MyRealVar = (R)

PRESS(VS3)
   REG[9] = CallCWMethod("MyCWVar1", "myFunc1", 1+7, MyStringVar1, sin(MyRealVar) -
8)
END_PRESS
```

Nota

El widget personalizado debe implementar el método "serialize". En este método es posible escribir los datos internos de un widget personalizado
en un fichero predefinido, o volver a crearlo a partir de este. Esto es necesario, sobre todo, cuando, estando abierta la máscara "Run MyScreens", se pasa a otro campo de manejo y a continuación se vuelve al anterior. De lo contrario, los datos internos se perderían al revisualizar.

Sintaxis:	public slots: bool **serialize**(const QString& szFilePath, bool bIsStoring);	
Descripción:	Lectura o escritura de datos internos y estados desde un fichero o en él.	
Parámetros:	szFilePath	Nombre del fichero (con la ruta completa) en el que se escribirán los datos y estados del widget personalizado o del que se leerán. Es posible que el fichero deba ser creado por el propio widget personalizado.
	bIsStoring	TRUE = escribir FALSE = leer

Ejemplo

```
bool SlEsTestCustomWidget::serialize(const QString& szFilePath, bool bIsStoring)
{
   QFileInfo fi(szFilePath);
   bool bReturn = false;
   QDir dir;
   if (dir.mkpath(fi.canonicalPath()))
   {
      QFile fileData(szFilePath);
      QIODevice::OpenMode mode;

      if (bIsStoring)
      {
         mode = QIODevice::WriteOnly;
      }
      else
      {
         mode = QIODevice::ReadOnly;
      }

      if (fileData.open(mode))
      {
         QDataStream streamData;
         streamData.setDevice(&fileData);

         if (bIsStoring)
         {
            streamData << m_nDataCount << m_dValueX;
```

bool SlEsTestCustomWidget::serialize(const QString& szFilePath, bool bIsStoring)

```
        }
        else
        {
           streamData >> m_nDataCount >> m_dValueX;
        }

        streamData.setDevice(0);
           fileData.flush();
           fileData.close();
        bReturn = true;
     }
  }
  return bReturn;
}
```

9.4.5.3 Reacción a una señal del widget personalizado

Descripción

En Run MyScreens es posible reaccionar a una señal determinada (invokeSub()) del widget personalizado para llamar un subprograma (SUB).

Para la transferencia de valores (señal del widget personalizado -> SUB) existen 10 variables globales, denominadas SIGARG, que en la configuración pueden compararse con los registros (REG). En ellas se almacenan los valores transferidos con la señal del widget personalizado.

Se soportan los siguientes formatos de datos de los parámetros de transferencia:

- bool
- uint
- int
- double
- QString
- QByteArray

Programación

Llamada del subprograma:

Sintaxis:	`void invokeSub(const QString& rszSignalName, const QVariantList& rvntList);`
Descripción:	Señal del widget personalizado con la que se llama un subprograma de Run MyScreens.

Parámetro:	rszSignalName	Nombre del subprograma de Run MyScreens que desea llamarse
	rvntList	Matriz QVariantList para transferir parámetros almacenados en los parámetros globales SIGARG y disponibles en la configuración. Tamaño máximo: 10 elementos Formatos de datos soportados: ver arriba **Nota:** los parámetros de transferencia siempre se transfieren "ByVal", lo que significa que siempre se transfiere el valor solamente y no, p. ej., la referencia a una variable.

Subprograma que se desea llamar:

Sintaxis:	`SUB(on_<nombre de variable>_<nombre de la señal>)`
	`…`
	`END_SUB`	

Descripción:	Reacción a una señal del widget personalizado	
Parámetro:	Nombre de variable	Nombre de la variable diálogo asignada a un widget personalizado.
	Nombre de la señal	Nombre de la señal del widget personalizado
	SIGARG 0 - 9	Parámetro de transferencia para el método del widget personalizado. Formatos de datos soportados: ver arriba **Nota:** los parámetros de transferencia siempre se transfieren "ByVal", lo que significa que siempre se transfiere el valor solamente y no, p. ej., la referencia a una variable.

Ejemplo

Widget personalizado, declaración de clase:

```
class SLESTESTCUSTOMWIDGET_EXPORT SlEsTestCustomWidget : public QWidget
{
   Q_OBJECT
signals:
   void invokeSub(const QString& szSubName, const QVariantList& vntList);
…
}
```

Widget personalizado, clase:

```
QVariantList vntList;
vntList << 123.456;
emit invokeSub("MySub", vntList);
```

Configuración de diálogo:

```
DEF MyCWVar1 = (W///,"slestestcustomwidget.SlEsTestCustomWidget")
SUB(on_MyCWVar1_MySub)
   DEBUG("SUB(on_MyCWVar1_MySub) was called with parameter: """ << SIGARG[0] <<
""""")
END_SUB
```

Resultado "easyscreen_log.txt":

```
[10:22:40.445] DEBUG: SUB(on_MyCWVar1_MySub) was called with parameter: "123.456"
```

9.5 SlEsGraphCustomWidget

9.5.1 SlEsGraphCustomWidget

Generalidades

Con ayuda de SlEsGraphCustomWidget pueden representarse objetos geométricos (punto, línea, rectángulo, rectángulo con esquinas redondeadas, elipse, arco, texto) y curvas a partir de nodos de interpolación (p. ej., valores medidos, evolución).

Los objetos se organizan en uno o varios contornos. Tras ello, los contornos pueden visualizarse, vaciarse, seleccionarse o borrarse uno por uno o combinados.

Los objetos se agregan al contorno seleccionado actualmente con ayuda de funciones, y se dibujan en el mismo orden. Si se desea dibujar objetos con un determinado color, debe definirse el color de dibujo correspondiente antes de agregar los objetos. Todos los objetos agregados a continuación se dibujarán en ese color. Además del color de dibujo, también puede modificarse el ancho y el estilo de dibujo. Para los objetos cerrados como el rectángulo, el rectángulo con esquinas redondeadas y la elipse, puede seleccionarse un color de relleno antes de agregar los objetos.

Cada contorno puede adoptar distintos modos:

- Visualización normal de todos los tipos de objetos.
- Los puntos pueden conectarse para formar una polilínea y mostrarse como curva o gráfico.
- La superficie entre los puntos, líneas o arcos hasta el eje X puede colorearse con el color de relleno actual. Esta posibilidad puede usarse, por ejemplo, para visualizar el material sobrante del torneado.
  Si en este modo, además, se visualizan puntos en forma de polilínea, se rellenará toda la superficie situada por debajo de la curva hasta el eje X. Los puntos, líneas y arcos pueden encontrarse en cualquiera de los cuatro cuadrantes.

Un modo especial del cursor permite "recorrer" uno o varios contornos. Para ello, se coloca el cursor sobre el primer o el último objeto de punto de un contorno o se mueve un objeto de punto hacia delante o hacia atrás dentro del contorno partiendo de la posición actual del cursor.

En caso necesario, puede visualizarse un segundo eje Y (derecha) con escala propia. Este eje está acoplado al primer eje Y (izquierda) mediante un offset y un factor.

La función de búsqueda permite encontrar un punto agregado anteriormente a un contorno a partir de una coordenada X especificada y, si se desea, colocar el cursor sobre él.

Los contornos pueden configurarse también como búfer circular con tamaño configurable.

Gracias a la serialización, es posible almacenar el estado actual del SlEsGraphCustomWidget en forma binaria en un fichero, así como restablecerlo.

El SlEsGraphCustomWidget puede manejarse mediante el gesto "Pan" (desplazamiento de la vista) y los gestos "Pinch"/"Spread" (aumentar en la vista/fuera de la vista).

La vista puede desplazarse (botón izquierdo del ratón + movimiento) y aumentarse (ruedecilla del ratón).

Por motivos de rendimiento, no se actualiza automáticamente la visualización. Según el caso de aplicación, puede forzarse la actualización activando una función determinada.

Ejemplo

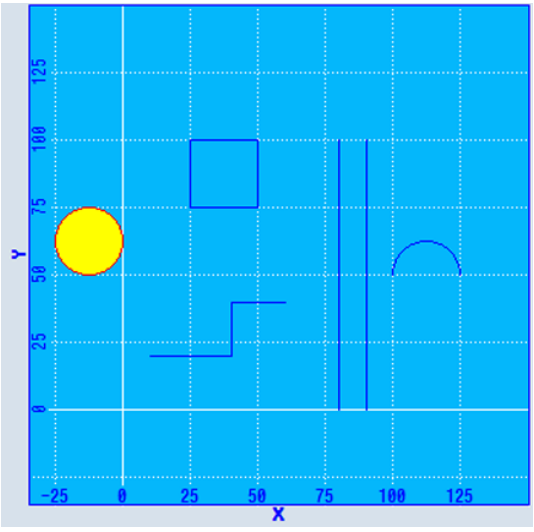


Figura 9-1 SlEsGraphCustomWidget: ejemplo

```
//M(MyGraphSampleMask/"SlEsGraphCustomWidget Sample")
DEF MyGraphVar = (W///,"slesgraphcustomwidget.SlEsGraphCustomWidget"/////10,10,340,340/0,0,0,0)
VS1=("Add objects",,se1)
LOAD
   WRITECWPROPERTY("MyGraphVar", "AxisNameX", "X")
   WRITECWPROPERTY("MyGraphVar", "AxisNameY", "Y")
   WRITECWPROPERTY("MyGraphVar", "ScaleTextOrientationYAxis", 2)
   WRITECWPROPERTY("MyGraphVar", "KeepAspectRatio", TRUE)

   REG[0]= CALLCWMETHOD("MyGraphVar", "addContour", "MyContour", TRUE)
   REG[0]= CALLCWMETHOD("MyGraphVar", "showContour", "MyContour")
   REG[0]= CALLCWMETHOD("MyGraphVar", "setView", -35, -35, 150, 150)
```

```
//M(MyGraphSampleMask/"SlEsGraphCustomWidget Sample")
END_LOAD

PRESS(VS1)
   REG[0]= CALLCWMETHOD("MyGraphVar", "addLine", 10, 20, 40, 20)
   REG[0]= CALLCWMETHOD("MyGraphVar", "addLine", 40, 20, 40, 40)
   REG[0]= CALLCWMETHOD("MyGraphVar", "addLine", 40, 40, 60, 40)
   REG[0]= CALLCWMETHOD("MyGraphVar", "addLine", 80, 0, 80, 100)
   REG[0]= CALLCWMETHOD("MyGraphVar", "addLine", 90, 0, 90, 100)
   REG[0]= CALLCWMETHOD("MyGraphVar", "addRect", 25, 100, 50, 75)
   REG[0]= CALLCWMETHOD("MyGraphVar", "setPenColor", "#ff0000");red
   REG[0]= CALLCWMETHOD("MyGraphVar", "setFillColor", "#ffff00");yellow
   REG[0]= CALLCWMETHOD("MyGraphVar", "addCircle", -12.5, 62.5, 12.5)
   REG[0]= CALLCWMETHOD("MyGraphVar", "setFillColor");off/default
   REG[0]= CALLCWMETHOD("MyGraphVar", "setPenColor");off/default
   REG[0]= CALLCWMETHOD("MyGraphVar", "addArc", 100, 37.5, 125, 62.5, 0, 180)
   REG[0]= CALLCWMETHOD("MyGraphVar", "update")
END_PRESS
```

9.5.2 Indicaciones sobre el rendimiento

En función del hardware utilizado y de la carga a la que esté sometido el sistema, al utilizar SlEsGraphCustomWidgets deben tenerse en cuenta los siguientes valores orientativos. Además del hardware utilizado y de la carga a la que esté sometido el sistema, los valores varían en función de la configuración de los SlEsGraphCustomWidgets.

Tenga en cuenta estos valores orientativos para no mermar la capacidad de reacción y la estabilidad del sistema en su conjunto.

- Número de SlEsGraphCustomWidgets configurados al mismo tiempo (p. ej., máximo 1)
- Número de contornos configurados (p. ej., máximo 6)
- Número de objetos gráficos configurados por contorno (p. ej., máximo 1000)
- Frecuencia de solicitud de actualización (p. ej., máximo 500 ms)

Nota

La visualización no es apta para tiempo real.

9.5.3 Lectura y escritura de propiedades

Descripción

Las propiedades mencionadas en el siguiente capítulo se leen con la función ReadCWProperty() y se ajustan con WriteCWProperty().

Ejemplos

Lectura de las propiedades "CursorX" del SlEsGraphCustomWidget conectado mediante la variable de visualización "MyGraphVar". El resultado se escribe en el registro 0.

```
REG[0] = ReadCWProperty("MyGraphVar", "CursorX")
```

Escritura del valor "MyFirstContour" en la propiedad "SelectedContour" del SlEsGraphCustomWidget conectado mediante la variable de visualización "MyGraphVar".

```
WriteCWProperty("MyGraphVar ", "SelectedContour", "MyFirstContour")
```

9.5.4 Propiedades

Vista general

Propiedad	Descripción
AxisNameX	Identificaciones de eje, eje X
AxisNameY	Identificaciones de eje, eje Y
AxisNameY2	Identificaciones de eje, segundo eje Y (derecha)
AxisY2Visible	Ver/ocultar segundo eje Y (derecha)
AxisY2Offset	Offset del segundo eje Y (derecha)
AxisY2Factor	Factor segundo eje Y (derecha)
ScaleTextEmbedded	Posicionamiento de los textos de escala
ScaleTextOrientationYAxis	Alineación de los textos de escala del eje Y
KeepAspectRatio	Conservar las proporciones de la página
SelectedContour	Nombre del contorno actualmente seleccionado
BackColor	Color de fondo
ChartBackColor	Color de fondo de la superficie de dibujo
ForeColor	Color de primer plano para textos y color de dibujo predeterminado
ForeColorY2	Color de primer plano para los textos del segundo eje Y (derecha)
GridColor	Color de las líneas de retícula
GridColorY2	Color de las líneas de retícula horizontales del segundo eje Y (derecha)
CursorColor	Color de la retícula del cursor
ShowCursor	Mostrar/ocultar cursor
CursorX	Posición X del cursor
CursorY	Posición Y del cursor

Propiedad	Descripción
CursorY2	Posición Y del cursor referida al segundo eje Y (derecha)
CursorStyle	Tipo de representación del cursor
ViewMoveZoomMode	Comportamiento al ampliar y desplazar mediante gestos

AxisNameX: identificaciones de eje, eje X

Sintaxis:	ReturnValue = ReadCWProperty(GraphVarName, "**AxisNameX**") WriteCWProperty(GraphVarName, "**AxisNameX**", Value)	
Descripción:	Lee o define el identificador del eje X. Si no se especifica ningún identificador, para la superficie de dibujo se utilizará el espacio disponible.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Valor leído de la propiedad (QString)
	Value	Valor que se desea ajustar (QString)

AxisNameY: identificaciones de eje, eje Y

Sintaxis:	ReturnValue = ReadCWProperty(GraphVarName, "**AxisNameY**") WriteCWProperty(GraphVarName, "**AxisNameY**", Value)	
Descripción:	Lee o define el identificador del eje Y. Si no se especifica ningún identificador, para la superficie de dibujo se utilizará el espacio disponible.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Valor leído de la propiedad (QString)
	Value	Valor que se desea ajustar (QString)

AxisY2Visible: ver/ocultar segundo eje Y (derecha)

Sintaxis:	ReturnValue = ReadCWProperty(GraphVarName, "**AxisY2Visible**") WriteCWProperty(GraphVarName, "**AxisY2Visible**", Value)	
Descripción:	Ver/ocultar segundo eje Y (derecha)	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Valor leído de la propiedad (bool)
	Value	Valor que se desea ajustar (bool): TRUE o FALSE

AxisY2Offset: offset del segundo eje Y (derecha)

Sintaxis:	ReturnValue = ReadCWProperty(GraphVarName, "**AxisY2Offset**") WriteCWProperty(GraphVarName, "**AxisY2Offset**", Value)
Descripción:	Offset del segundo eje Y (derecha) referido al primer eje Y (izquierda). Ver ejemplo para la propiedad AxisY2Factor.

Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Valor leído de la propiedad (double)
	Value	Valor que se desea ajustar (double)

AxisY2Factor: factor segundo eje Y (derecha)

Sintaxis:	ReturnValue = ReadCWProperty(GraphVarName, "**AxisY2Factor**") WriteCWProperty(GraphVarName, "**AxisY2Factor**", Value)	
Descripción:	Factor del segundo eje Y (derecha) referido al primer eje Y (izquierda).	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Valor leído de la propiedad (double)
	Value	Valor que se desea ajustar (double)

Junto con la propiedad "AxisY2Offset" es posible visualizar un segundo eje Y (derecha) con escala propia. La escala está acoplada al primer eje Y (izquierda) mediante el offset y el factor.

Fórmula para convertir Y2 a Y:

Y = Y2 / factor - offset

Fórmula para convertir Y a Y2:

Y2 = (Y + offset) * factor

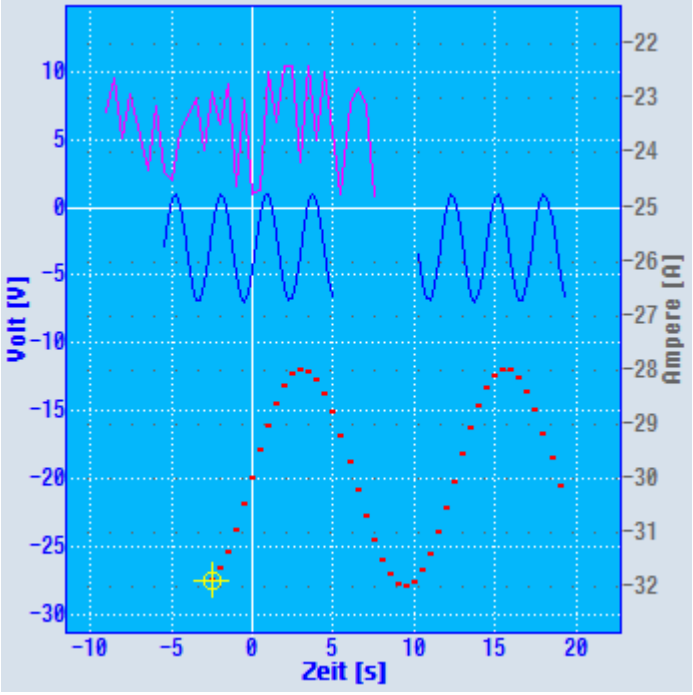


Figura 9-2 Ejemplo con el segundo eje Y con escala propia

Ejemplo

Y: de 0 a 200 corresponderá a Y2: de -25 a 25.

- Offset: -100
- Factor: 0,25

Ejemplo de cálculo:

Y2 = -30

Y = -30 / 0.25 – (-100) = -20

Ejemplo de cálculo:

Y = 0

Y2 = (0 + (-100)) * 0,25 = -25

El eje X es igual para los dos sistemas de coordenadas.

Los colores pueden definirse con setForeColorY2() y setGridColorY2() (ver Colores).

La rotulación de los ejes se define con setAxisNameY2() (ver Identificaciones de eje).

ScaleTextEmbedded: posicionamiento de los textos de escala

Sintaxis:	ReturnValue = ReadCWProperty(GraphVarName, "**ScaleTextEmbedded**") WriteCWProperty(GraphVarName, "**ScaleTextEmbedded**", Value)	
Descripción:	Con esta propiedad se define si los textos de escala se posicionarán dentro o fuera de la superficie de dibujo.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Valor leído de la propiedad (bool)
	Value	Valor que se desea ajustar (bool): TRUE o FALSE

Ejemplo

Textos de escala fuera del área de dibujo:

ScaleTextEmbedded = FALSE

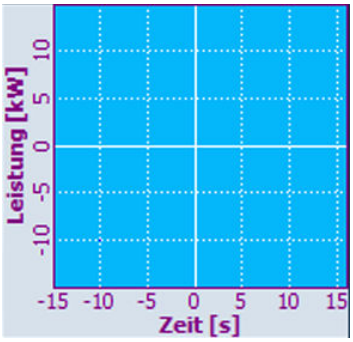


Figura 9-3 Textos de escala fuera del área de dibujo

Textos de escala dentro del área de dibujo:

ScaleTextEmbedded = TRUE

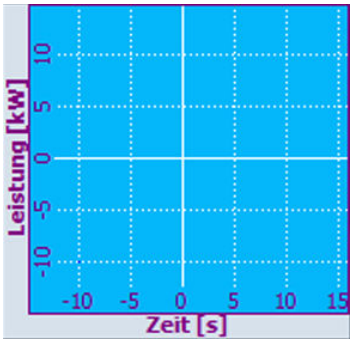


Figura 9-4 Textos de escala dentro del área de dibujo

ScaleTextOrientationYAxis: alineación de los textos de escala del eje Y

Sintaxis:	ReturnValue = ReadCWProperty(GraphVarName, "**ScaleTextOrientationYAxis**") WriteCWProperty(GraphVarName, "**ScaleTextOrientationYAxis**", Value)	
Descripción:	Con esta función se alinean verticalmente los textos de escala del eje Y.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Valor leído de la propiedad (int)
	Value	Valor que se desea ajustar (int): 1 (= horizontal) o 2 (= vertical)

Ejemplo

Textos de escala dentro del área de dibujo:

- ScaleTextEmbedded = TRUE

Alineación de texto horizontal:

- ScaleTextOrientationYAxis = 1

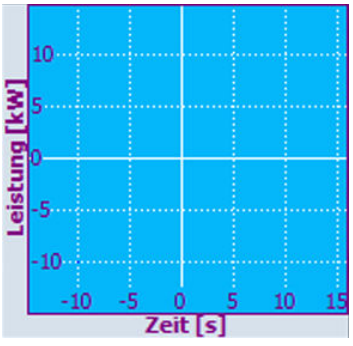


Figura 9-5 Textos de escala dentro del área de dibujo, alineación de texto horizontal

Alineación de texto vertical:

- ScaleTextOrientationYAxis = 2

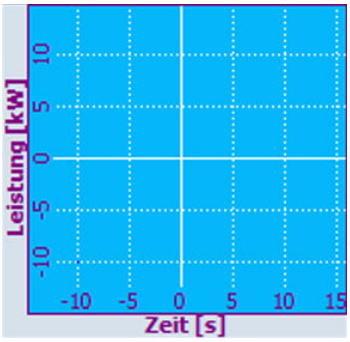


Figura 9-6 Textos de escala dentro del área de dibujo, alineación de texto vertical

Textos de escala fuera del área de dibujo:

- ScaleTextEmbedded = FALSE

Alineación de texto horizontal:

- ScaleTextOrientationYAxis = 1

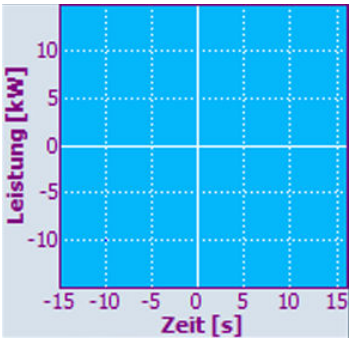


Figura 9-7 Textos de escala fuera del área de dibujo, alineación de texto horizontal

Alineación de texto vertical:

- ScaleTextOrientationYAxis = 2

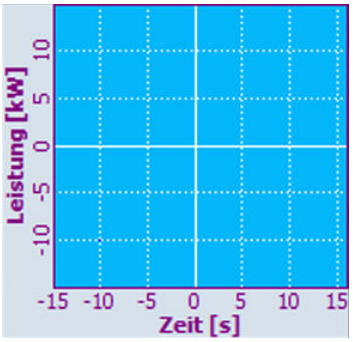


Figura 9-8 Textos de escala fuera del área de dibujo, alineación de texto vertical

KeepAspectRatio: conservar las proporciones de la página

Sintaxis:	ReturnValue = ReadCWProperty(GraphVarName, "**KeepAspectRatio**") WriteCWProperty(GraphVarName, "**KeepAspectRatio**", Value)	
Descripción:	Esta propiedad determina si la vista definida con setView() se alineará, adaptará o ampliará automáticamente a fin de que la relación de aspecto entre el eje X y el eje Y permanezca invariable. Esta propiedad es relevante en especial al visualizar figuras geométricas. Con ella, p. ej., un círculo o un cuadrado aparece siempre como tal y no deformado como elipse o rectángulo.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Valor leído de la propiedad (bool)
	Return Value	Valor que se desea ajustar (bool): TRUE o FALSE

Ejemplo

```
setView(-15, -15, 15, 15)
setFillColor("#A0A0A4")
addCircle(0, 0, 5)
KeepAspectRatio = TRUE
```

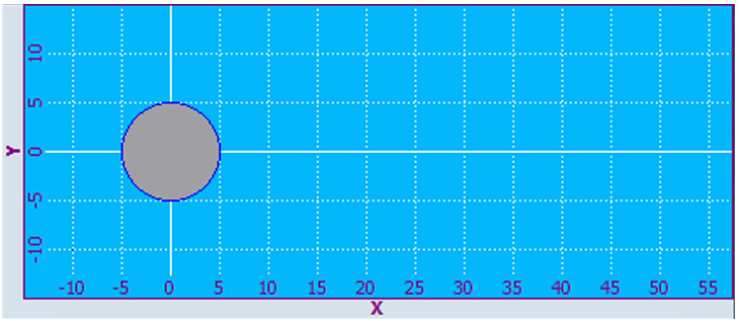


Figura 9-9 La relación de aspecto entre los ejes X e Y no varía

```
KeepAspectRatio = FALSE
```

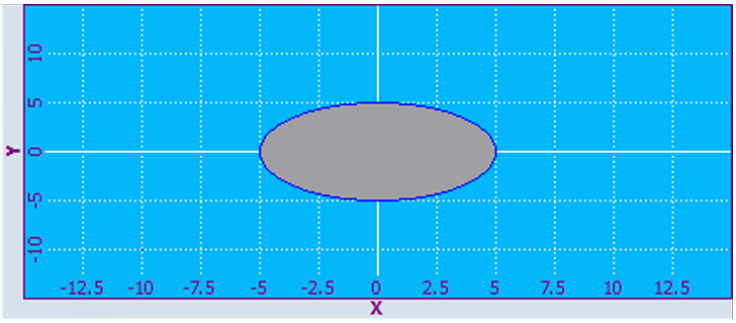


Figura 9-10 La relación de aspecto entre los ejes X e Y se deforma

SelectedContour: nombre del contorno actualmente seleccionado

Sintaxis:	ReturnValue = ReadCWProperty(GraphVarName, " **SelectedContour**") WriteCWProperty(GraphVarName, " **SelectedContour**", Value)	
Descripción:	Lee o ajusta la selección del contorno actual.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Valor leído de la propiedad (QString)
	Value	Valor que se desea ajustar (QString)

Nota

Para poder ejecutar operaciones en un contorno, p. ej., para agregar objetos gráficos, es necesario seleccionarlos previamente.

BackColor: color de fondo

Sintaxis:	ReturnValue = ReadCWProperty(GraphVarName, " **BackColor**") WriteCWProperty(GraphVarName, " **BackColor**", Value)	
Descripción:	Color de fondo para la rotulación de ejes, y en función de la propiedad ScaleTextInsideChartArea, también de los textos de escala.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Color leído de la propiedad (QString)
	Value	Valor que se debe ajustar (QString) como valor RGB en el formato "#RRGGBB", p. ej., "#04B7FB"

ChartBackColor: color de fondo de la superficie de dibujo

Sintaxis:	ReturnValue = ReadCWProperty(GraphVarName, "**ChartBackColor**") WriteCWProperty(GraphVarName, "**ChartBackColor**", Value)
Descripción:	Color de fondo del área de dibujo.

Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Color leído de la propiedad (QString)
	Value	Valor que se debe ajustar (QString) como valor RGB en el formato "#RRGGBB", p. ej., "#04B7FB"

ForeColor: color de primer plano para la rotulación y color de dibujo predeterminado

Sintaxis:	ReturnValue = ReadCWProperty(GraphVarName, "**ForeColor**") WriteCWProperty(GraphVarName, "**ForeColor**", Value)	
Descripción:	Color de primer plano para textos y color de dibujo predeterminado.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Color leído de la propiedad (QString)
	Value	Valor que se debe ajustar (QString) como valor RGB en el formato "#RRGGBB", p. ej., "#04B7FB"

ForeColorY2: color de primer plano para la rotulación del segundo eje Y (derecha)

Sintaxis:	ReturnValue = ReadCWProperty(GraphVarName, "**ForeColorY2**") WriteCWProperty(GraphVarName, "**ForeColorY2**", Value)	
Descripción:	Color de primer plano para los textos del segundo eje Y (derecha).	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Color leído de la propiedad (QString)
	Value	Valor que se debe ajustar (QString) como valor RGB en el formato "#RRGGBB", p. ej., "#04B7FB"

GridColor: color de las líneas de retícula

Sintaxis:	ReturnValue = ReadCWProperty(GraphVarName, "**GridColor**") WriteCWProperty(GraphVarName, "**GridColor**", Value)	
Descripción:	Color de las líneas de retícula.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Color leído de la propiedad (QString)
	Value	Valor que se debe ajustar (QString) como valor RGB en el formato "#RRGGBB", p. ej., "#04B7FB"

GridColorY2: color de las líneas de retícula horizontales del segundo eje Y (derecha)

Sintaxis:	ReturnValue = ReadCWProperty(GraphVarName, "**GridColorY2**") WriteCWProperty(GraphVarName, "**GridColorY2**", Value)
Descripción:	Color de las líneas de retícula horizontales del segundo eje Y (derecha).

Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Color leído de la propiedad (QString)
	Value	Valor que se debe ajustar (QString) como valor RGB en el formato "#RRGGBB", p. ej., "#04B7FB"

CursorColor: color del cursor

Sintaxis:	ReturnValue = ReadCWProperty(GraphVarName, "**CursorColor**") WriteCWProperty(GraphVarName, "**CursorColor**", Value)	
Descripción:	Color del cursor.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Color leído de la propiedad (QString)
	Value	Valor que se debe ajustar (QString) como valor RGB en el formato "#RRGGBB", p. ej., "#04B7FB"

ShowCursor: mostrar/ocultar cursor

Sintaxis:	ReturnValue = ReadCWProperty(GraphVarName, " **ShowCursor**") WriteCWProperty(GraphVarName, " **ShowCursor**", Value)	
Descripción:	Mostrar/ocultar cursor.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Valor leído de la propiedad (bool)
	Value	Valor que se desea ajustar (bool): TRUE o FALSE

Nota

Esta función actualiza la vista automáticamente.

CursorX: posición X del cursor

Sintaxis:	ReturnValue = ReadCWProperty(GraphVarName, "**CursorX**") WriteCWProperty(GraphVarName, "**CursorX**", Value)	
Descripción:	Posición X del cursor.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Valor leído de la propiedad (double)
	Value	Valor que se desea ajustar (double)

Nota

Esta función actualiza la vista automáticamente.

CursorY: posición Y del cursor

Sintaxis:	ReturnValue = ReadCWProperty(GraphVarName, "**CursorY**") WriteCWProperty(GraphVarName, "**CursorY**", Value)	
Descripción:	Posición Y del cursor.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Valor leído de la propiedad (double)
	Value	Valor que se desea ajustar (double)

Nota

Esta función actualiza la vista automáticamente.

CursorY2: posición Y del cursor referida al segundo eje Y (derecha)

Sintaxis:	ReturnValue = ReadCWProperty(GraphVarName, "**CursorY2**") WriteCWProperty(GraphVarName, "**CursorY2**", Value)	
Descripción:	Posición Y del cursor referida al segundo eje Y (derecha).	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Valor leído de la propiedad (double)
	Value	Valor que se desea ajustar (double)

Nota

Esta función actualiza la vista automáticamente.

CursorStyle: tipo de representación del cursor

Sintaxis:	ReturnValue = ReadCWProperty(GraphVarName, "**CursorStyle**") WriteCWProperty(GraphVarName, "**CursorStyle**", Value)	
Descripción:	Tipo de representación del cursor.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Valor leído de la propiedad (int)
	Value	Valor que se desea ajustar (int): 0 (= cruz) 1 (= retícula) 2 (= línea vertical) 3 (= línea horizontal) 4 (= línea horizontal y vertical)

ViewMoveZoomMode: comportamiento al ampliar y desplazar mediante gestos

Sintaxis:	ReturnValue = ReadCWProperty(GraphVarName, "**ViewMoveZoomMode**") WriteCWProperty(GraphVarName, "**ViewMoveZoomMode**", Value)	
Descripción:	Es posible modificar la vista del SlEsGraphCustomWidgets mediante gestos. Esta propiedad permite definir de qué modo se desea permitir esto. Por ejemplo, en determinados casos de aplicación puede ser útil que la vista solo pueda desplazarse y ampliarse en sentido horizontal, p. ej., al visualizar la evolución de valores medidos.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Valor leído de la propiedad (int)
	Value	Valor que se desea ajustar (int): 0 (= desactivado) 1 (= solo horizontal) 2 (= solo vertical) 3 (= horizontal y vertical)

9.5.5 Funciones

Las funciones que se indican a continuación pueden llamarse con la función CallCWMethod().

Ejemplo

Ajustar la vista del SlEsGraphCustomWidget conectado mediante la variable de visualización "MyGraphVar".

El resultado se escribe en el registro 0.

```
REG[0] = CallCWMethod("MyGraphVar", "setView", -8, -5, 11- 20)
```

Vista general

Función	Descripción
setView	Definir el sistema de coordenadas
setMaxContourObjects	Definir el número máximo de objetos de un contorno (búfer circular)
addContour	Agregar contorno
showContour	Ver contorno
hideContour	Invisibilizar contorno
hideAllContours	Invisibilizar todos los contornos
removeContour	Eliminar contorno
clearContour	Borrar los objetos gráficos de un contorno
fitViewToContours	Adaptación automática de la vista
fitViewToContour	Adaptación automática de la vista
findX	Buscar en un contorno
setPolylineMode	Mostrar puntos de un contorno en forma de polilínea/curva

Función	Descripción
setIntegralFillMode	Mostrar puntos de un contorno en forma de polilínea/curva
repaint, update	Actualizar vista
addPoint	Agregar punto a un contorno
addLine	Agregar línea a un contorno
addRect	Agregar rectángulo a un contorno
addRoundedRect	Agregar rectángulo con esquinas redondeadas a un contorno
addEllipse	Agregar elipse a un contorno
addCircle	Agregar círculo a un contorno
addArc	Agregar arco a un contorno
addText	Agregar texto a un contorno
setPenWidth	Ajustar el ancho de dibujo
setPenStyle	Definir el estilo de dibujo
setPenColor	Definir el color de dibujo
setFillColor	Ajustar el color de relleno
setCursorPosition	Posicionar el cursor
setCursorPositionY2	Posicionar el cursor referido al segundo eje Y (derecha)
setCursorOnContour	Posicionar el cursor en un contorno
moveCursorOnContourBegin	Posicionar el cursor en el primer objeto gráfico de un contorno
moveCursorOnContourEnd	Posicionar el cursor en el último objeto gráfico de un contorno
moveCursorOnContourNext	Posicionar el cursor en el siguiente objeto gráfico de un contorno
serialize	Guardar/restablecer el estado actual

setView: definir el sistema de coordenadas

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**setView**", x1, y1, x2, y2)	
Descripción:	Con esta función se define el tamaño del sistema de coordenadas que se mostrará dentro del SlEsGraphCustomWidget. Ver al respecto también la propiedad KeepAspectRatio.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	x1	Borde izquierdo (double)
	y1	Borde superior (double)
	x2	Borde derecho (double)
	y2	Borde inferior (double)

Nota

La función actualiza la vista automáticamente.

Ejemplo

```
setView(-8, -5, 11, 20)
AxisNameX = "X"
AxisNameY = "Y"
ScaleTextOrientationYAxis = 1
ScaleTextEmbedded = false
KeepAspectRatio = false
update
```

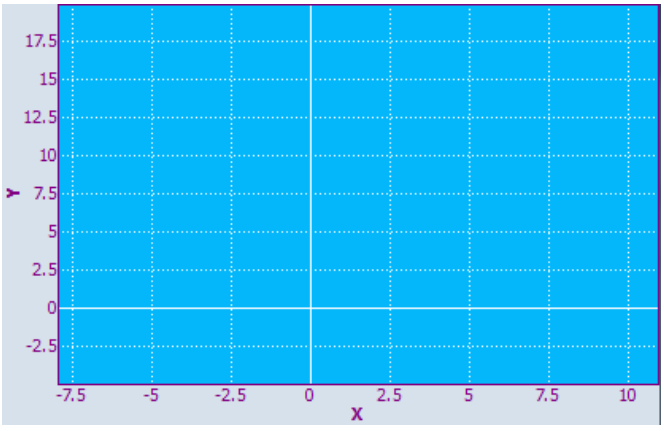


Figura 9-11 Ejemplo: setView

setMaxContourObjects: definir el número máximo de objetos de un contorno (búfer circular)

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**setMaxContourObjects**", Value) ReturnValue = CallCWMethod(GraphVarName, "**setMaxContourObjects**", Value, ContourName)	
Descripción:	Con esta función se define el tamaño del búfer circular de un contorno. Si el número de objetos agregados al contorno supera dicho límite, se borrará el objeto más antiguo.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	Value	Número máximo de objetos gráficos (uint)
	ContourName	Nombre del contorno (QString). Si no se indica un contorno, la llamada se refiere automáticamente al contorno seleccionado actualmente.

Nota

La función actualiza la vista automáticamente.

addContour: agregar contorno

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**addContour**", ContourName) ReturnValue = CallCWMethod(GraphVarName, "**addContour**", ContourName, Selected) ReturnValue = CallCWMethod(GraphVarName, "**addContour**", ContourName, Selected, AssignedY2)	
Descripción:	Con esta función se crea un contorno nuevo. Opcionalmente puede asignar el contorno al sistema de coordenadas del segundo eje Y (derecha).	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	ContourName	Nombre del contorno (QString). Si no se indica un contorno, la llamada se refiere automáticamente al contorno seleccionado actualmente.
	Selected	Se desea seleccionar un contorno (bool): TRUE o FALSE
	AssignedY2	Se desea asignar el contorno al segundo eje Y (derecha) (bool): TRUE o FALSE

showContour: ver contorno

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**showContour**") ReturnValue = CallCWMethod(GraphVarName, "**showContour**", ContourName)	
Descripción:	Esta función hace visible un contorno.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	ContourName	Nombre del contorno (QString). Si no se indica un contorno, la llamada se refiere automáticamente al contorno seleccionado actualmente.

hideContour: invisibilizar contorno

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**hideContour**") ReturnValue = CallCWMethod(GraphVarName, "**hideContour**", ContourName)	
Descripción:	Esta función hace invisible un contorno.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	ContourName	Nombre del contorno (QString). Si no se indica un contorno, la llamada se refiere automáticamente al contorno seleccionado actualmente.

showAllContours: visibilizar todos los contornos

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**showAllContours**")	
Descripción:	Esta función hace visibles todos los contornos.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto

hideAllContours: invisibilizar todos los contornos

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**hideAllContours**")	
Descripción:	Esta función hace invisibles todos los contornos.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto

removeContour: invisibilizar contorno

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**removeContour**") ReturnValue = CallCWMethod(GraphVarName, "**removeContour**", ContourName)	
Descripción:	Elimina un contorno.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	ContourName	Nombre del contorno (QString). Si no se indica un contorno, la llamada se refiere automáticamente al contorno seleccionado actualmente.

clearContour: borrar los objetos gráficos de un contorno

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**clearContour**") ReturnValue = CallCWMethod(GraphVarName, "**clearContour**", ContourName)	
Descripción:	Borra todos los objetos gráficos contenidos en un contorno.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	ContourName	Nombre del contorno (QString). Si no se indica un contorno, la llamada se refiere automáticamente al contorno seleccionado actualmente.

fitViewToContours: adaptación automática de la vista

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**fitViewToContours**") ReturnValue = CallCWMethod(GraphVarName, "**fitViewToContours**", OnlyVisible)	
Descripción:	Con esta función se adapta la vista automáticamente a los contornos. En función del parámetro de transferencia, se determina si serán visibles solo todos los contornos visibles o todos los contornos.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	OnlyVisible	Valor que se desea ajustar (bool): TRUE o FALSE

fitViewToContour: adaptación automática de la vista

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**fitViewToContour**", ContourName)	
Descripción:	Con esta función se adapta la vista automáticamente, de modo que solo es visible un contorno determinado.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	ContourName	Nombre del contorno (QString)

findX: buscar en un contorno

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**findX**", x) ReturnValue = CallCWMethod(GraphVarName, "**findX**", x, ContourName) ReturnValue = CallCWMethod(GraphVarName, "**findX**", x, ContourName, SetCursor)	
Descripción:	Con estas funciones puede buscar en un contorno un punto previamente definido con una determinada coordenada X. El resultado obtenido será la coordenada Y. Opcionalmente también puede posicionar el cursor directamente en este punto.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (QString): si la operación es correcta, se devuelve el valor Y correspondiente
	x	Valor X que se desea buscar (double)
	ContourName	Nombre del contorno (QString). Si no se indica un contorno, la llamada se refiere automáticamente al contorno seleccionado actualmente
	SetCursor	Situar el cursor (bool): TRUE o FALSE

Nota

Si se sitúa el cursor con esta función, la vista se actualiza automáticamente.

setPolylineMode: mostrar puntos de un contorno en forma de polilínea/curva

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**setPolylineMode**", SetPoly)	
Descripción:	Todos los puntos agregados al contorno actual después de esta llamada de función se unen en una polilínea/curva. Esta función es específica de contorno y puede activarse o desactivarse por segmentos.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	SetPoly	PolylineMode (bool): TRUE = activado

Ejemplo

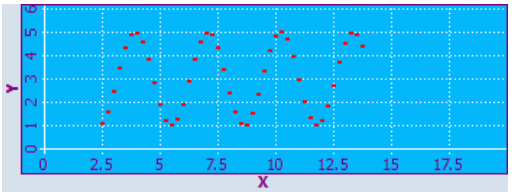


Figura 9-12 Ejemplo: PolylineMode: desactivado

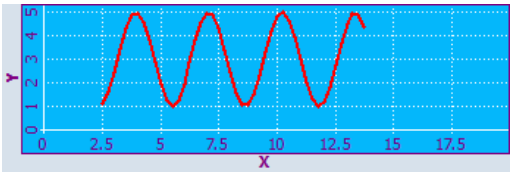


Figura 9-13 Ejemplo: PolylineMode: activado

setIntegralFillMode: rellenar las superficies entre puntos, líneas y arcos y el eje X

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**setIntegralFillMode**", SetIntFill)	
Descripción:	Las superficies entre puntos, líneas y arcos y el eje X se rellenan con el color de relleno seleccionado (independientemente de que los puntos sean +Y o –Y). Esta función es específica de contorno y puede activarse o desactivarse por segmentos.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	SetIntFill	IntegralFillMode (bool): TRUE = activado

Ejemplo

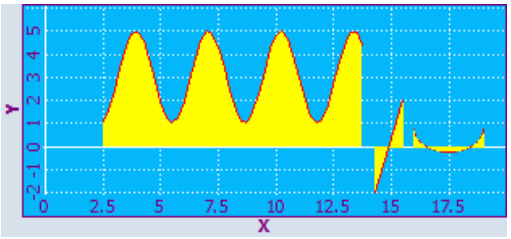


Figura 9-14 Ejemplo: setIntegralFillMode: activado

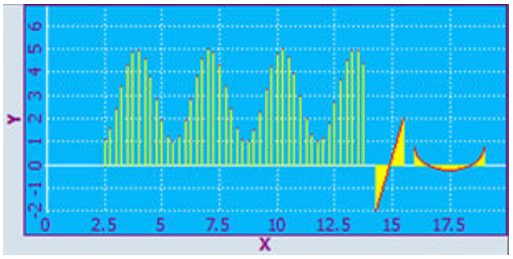


Figura 9-15 Ejemplo: setIntegralFillMode: desactivado

repaint, update: actualizar la vista

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**repaint**") ReturnValue = CallCWMethod(GraphVarName, "**update**")	
Descripción:	Estas funciones permiten actualizar manualmente la vista.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto

- update()
  La función actualiza la vista, a condición de que la actualización del widget no esté desactivada y el widget no esté oculto. La función está optimizada para mantener el rendimiento de la aplicación y la vista no se haga inestable debido a una actualización demasiado frecuente.
- repaint()
  La función actualiza la vista justo después de la llamada. Por ello, la función repaint solo debe usarse si es imprescindible la actualización inmediata, p. ej. en animaciones.

Se recomienda trabajar siempre con la función update(). Internamente, el SlEsGraphCustomWidget trabaja siempre con la función update(), p. ej., con setView().

addPoint: agregar punto a un contorno

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**addPoint**", x, y) ReturnValue = CallCWMethod(GraphVarName, "**addPoint**", x, y, DummyPoint)	
Descripción:	Agregar un punto al contorno seleccionado. Además puede definirse un punto de prueba (dummy) que no se dibujará, pero puede posicionarse con el cursor.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	x	Coordenada x (double)
	y	Coordenada y (double)
	DummyPoint	El punto se trata como invisible (bool): TRUE = sí

addLine: agregar línea a un contorno

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**addLine**", x1, y1, x2, y2)	
Descripción:	Agregar una línea al contorno seleccionado actualmente	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	x1	Coordenada x del punto inicial (double)
	y1	Coordenada y del punto inicial (double)
	x2	Coordenada x del punto final (double)
	y2	Coordenada y del punto final (double)

addRect: agregar rectángulo a un contorno

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**addRect**", x1, y1, x2, y2)	
Descripción:	Agregar un rectángulo al contorno seleccionado actualmente.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	x1	Borde izquierdo (double)
	y1	Borde izquierdo (double)
	x2	Borde derecho (double)
	y2	Borde inferior (double)

addRoundedRect: agregar rectángulo con esquinas redondeadas a un contorno

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**addRoundedRect**", x1, y1, x2, y2, r)
Descripción:	Agregar un rectángulo con esquinas redondeadas al contorno seleccionado actualmente.

Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	x1	Borde izquierdo (double)
	y1	Borde superior (double)
	x2	Borde superior (double)
	y2	Borde superior (double)
	r	Radio (double)

addEllipse: agregar elipse a un contorno

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**addEllipse**", x1, y1, x2, y2, radius)	
Descripción:	Agregar una elipse al contorno seleccionado actualmente.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	x1	Borde izquierdo (double)
	y1	Borde superior (double)
	x2	Borde derecho (double)
	y2	Borde inferior (double)
	radio	Radio (double)

addCircle: agregar círculo a un contorno

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**addCircle**", x, y, radius)	
Descripción:	Agregar un círculo al contorno seleccionado actualmente.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	x	Borde izquierdo (double)
	y	Borde superior (double)
	radio	Radio (double)

addArc: agregar arco a un contorno

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**addArc**", x1, y1, x2, y2, StartAngle, SpanAngle
Descripción:	Agregar un arco al contorno seleccionado actualmente.

Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	x1	Borde izquierdo (double)
	y1	Borde superior (double)
	x2	Borde derecho (double)
	y2	Borde inferior (double)
	StartAngle	Ángulo inicial en grados (double)
	SpanAngle	Ángulo de apertura en grados (double)

addText: agregar texto a un contorno

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**addText**", x, y, Text)	
Descripción:	Agregar un texto al contorno seleccionado actualmente.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	x	Borde izquierdo (double)
	y	Borde superior (double)
	Texto	Texto (QString)

setPenWidth: ajustar el ancho de dibujo

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**setPenWidth**") ReturnValue = CallCWMethod(GraphVarName, "**setPenWidth**", width)	
Descripción:	Define el ancho de dibujo a partir de la llamada de la función para los objetos que se agreguen posteriormente al contorno seleccionado.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	width	Ancho de dibujo (double) Si no se indica ningún ancho de dibujo, se ajustará el ancho de dibujo predeterminado.

setPenStyle: definir el estilo de dibujo

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**setPenStyle**") ReturnValue = CallCWMethod(GraphVarName, "**setPenStyle**", style)
Descripción:	Define el estilo de dibujo a partir de la llamada de la función para los objetos que se agreguen posteriormente al contorno seleccionado.

Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	style	1: línea continua (___) 2 : línea discontinua (_ _) 3: línea de puntos (...) 4 : Línea-punto-línea (_._) 5 : línea-punto-punto (_..)

setPenColor: ajustar el color de dibujo

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**setPenColor**") ReturnValue = CallCWMethod(GraphVarName, "**setPenColor**", Color)	
Descripción:	Define el color de dibujo a partir de la llamada de la función para los objetos que se agreguen posteriormente al contorno seleccionado.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	Color	Valor que se debe ajustar (QString) como valor RGB en el formato "#RRGGBB", p. ej., "#04B7FB"

setFillColor: ajustar el color de relleno

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**setFillColor**") ReturnValue = CallCWMethod(GraphVarName, "**setFillColor**", Color)	
Descripción:	Define el color de relleno a partir de la llamada de la función para los objetos que se agreguen posteriormente al contorno seleccionado.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	Color	Valor que se debe ajustar (QString) como valor RGB en el formato "#RRGGBB", p. ej., "#04B7FB"

setCursorPosition: posicionar el cursor

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**setCursorPosition**", x, y)	
Descripción:	Posiciona el cursor en el lugar especificado.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	x	Coordenada x (double)
	y	Coordenada y (double)

Nota

Esta función actualiza la vista automáticamente.

setCursorPositionY2Cursor: posicionar el cursor referido al segundo eje Y (derecha)

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**setCursorPositionY2**", x, y2)	
Descripción:	Posiciona el cursor en el lugar especificado referido al segundo eje Y (derecha).	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	x	Coordenada x (double)
	y2	Coordenada Y referida al segundo eje Y (derecha) (double)

Nota

Esta función actualiza la vista automáticamente.

setCursorOnContour: posicionar el cursor en un contorno

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**setCursorOnContour**") ReturnValue = CallCWMethod(GraphVarName, "**setCursorOnContour**", ContourName)	
Descripción:	Coloca el cursor en la última posición del cursor dentro de un contorno.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	ContourName	Nombre del contorno (QString). Si no se indica un contorno, la llamada se refiere automáticamente al contorno seleccionado actualmente.

Nota

Esta función actualiza la vista automáticamente.

Ejemplo

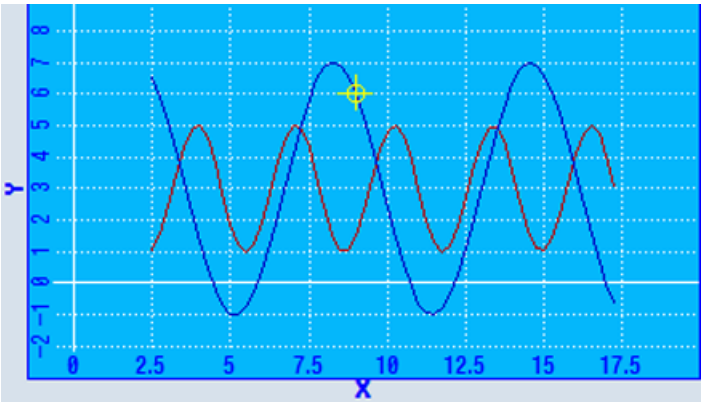


Figura 9-16 Ejemplo: setCursorOnContour

moveCursorOnContourBegin: posicionar el cursor en el primer objeto gráfico de un contorno

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourBegin**") ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourBegin**", ContourName)	
Descripción:	Partiendo de la posición actual del cursor dentro de un contorno, esta función permite navegar con el cursor hasta el primer objeto de punto de un contorno.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	ContourName	Nombre del contorno (QString). Si no se indica un contorno, la llamada se refiere automáticamente al contorno seleccionado actualmente.

Nota

Esta función actualiza la vista automáticamente.

moveCursorOnContourEnd: posicionar el cursor en el último objeto gráfico de un contorno

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourEnd**") ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourEnd**", ContourName)	
Descripción:	Partiendo de la posición actual del cursor dentro de un contorno, esta función permite navegar con el cursor hasta el último objeto de punto de un contorno.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	ContourName	Nombre del contorno (QString). Si no se indica un contorno, la llamada se refiere automáticamente al contorno seleccionado actualmente.

Nota

Esta función actualiza la vista automáticamente.

moveCursorOnContourNext: posicionar el cursor en el siguiente objeto gráfico de un contorno

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourNext**") ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourNext**", ContourName)	
Descripción:	Partiendo de la posición actual del cursor dentro de un contorno, esta función permite navegar con el cursor hasta el siguiente objeto de punto de un contorno.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	ContourName	Nombre del contorno (QString). Si no se indica un contorno, la llamada se refiere automáticamente al contorno seleccionado actualmente.

Nota

Esta función actualiza la vista automáticamente.

moveCursorOnContourBack: posicionar el cursor en el anterior objeto gráfico de un contorno

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourBack**") ReturnValue = CallCWMethod(GraphVarName, "**moveCursorOnContourBack**", ContourName)	
Descripción:	Partiendo de la posición actual del cursor dentro de un contorno, esta función permite navegar con el cursor hasta el anterior objeto de punto de un contorno.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	ContourName	Nombre del contorno (QString). Si no se indica un contorno, la llamada se refiere automáticamente al contorno seleccionado actualmente.

Nota

Esta función actualiza la vista automáticamente.

serialize: guardar/restablecer el estado actual

Sintaxis:	ReturnValue = CallCWMethod(GraphVarName, "**serialize**", FilePath, IsStoring)	
Descripción:	En caso necesario, esta función permite escribir el estado actual y el contenido del SlEsGraphWidget en formato binario en un fichero o DataStream, así como restablecerlo.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	Return Value	Errorcode (bool): TRUE = correcto
	FilePath	Ruta completa con nombre de fichero (QString)
	IsStoring	Guardar/restablecer (bool): TRUE = guardar

Nota

Esta función actualiza la vista automáticamente.

9.5.6 Señales

La señal que se indica a continuación (ViewChanged) puede captarse en la configuración a fin de definir una respuesta adecuada.

Vista general

Función	Descripción
ViewChanged	Modificación de la vista

ViewChanged: modificación de la vista

Sintaxis:	SUB(on_<GraphVarName>**_ViewChanged**) ... END_SUB	
Descripción:	Si la vista cambia, se emite la señal "ViewChanged". De este modo se puede responder a la señal en la configuración en un determinado método SUB. Los parámetros SIGARG se ajustan en consecuencia.	
Parámetros:	GraphVarName	Nombre de la variable de visualización que contiene un SlEsGraphCustomWidget
	SIGARG[0]	Borde izquierdo (double)
	SIGARG[1]	Borde superior (double)
	SIGARG[2]	Borde derecho (double)
	SIGARG[3]	Borde inferior (double)

Ejemplo

```
DEF MyGraphVar = (W///,"slesgraphcustomwidget.SlEsGraphCustomWidget"/////
10,10,340,340/0,0,0,0)
```

```
SUB(on_MyGraphVar_ViewChanged
     DLGL("Current view: " << SIGARG[0] << ", " << SIGARG[1] << ", " <<   SIGARG[2]
<< ", " << SIGARG[3]
END_SUB
```

9.6 SlEsTouchButton

9.6.1 SlEsTouchButton

Generalidades

Run MyScreens permite crear fácilmente aplicaciones de distinto grado de funcionalidad. En especial facilita la configuración de botones de libre ubicación para manejo táctil (TouchButtons). La configuración de los TouchButtons debe cumplir lo siguiente:

- Las dos modificaciones de estado posibles del botón, "clicked" (pulsado) y "checked" (enclavado), pueden recogerse en la configuración para disparar las acciones correspondientes.
- Los TouchButtons pueden manejarse por toque único o múltiple, ratón o teclado.
- La visualización del TouchButton depende de la resolución actual. Esto se aplica también a la fuente y a las imágenes visualizadas.
- Para el TouchButton pueden aplicarse dos estilos de representación: "Softkey Look&Feel" y "Específico del usuario". En el modo "Softkey Look&Feel", los TouchButtons se representan al estilo de los pulsadores de menú de Operate. En ambos estilos de representación están disponibles funciones como, por ejemplo, el escalado de imágenes.
- Los TouchButtons se implementan y están disponibles en forma de CustomWidget.

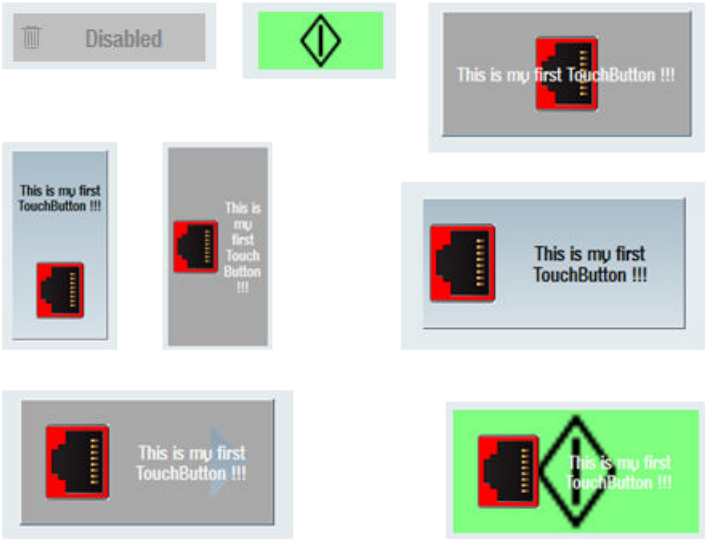


Figura 9-17 Ejemplos de TouchButton

Ejemplo

```
//M(MyTBMask/"My CustomWidget TouchButton ...")
DEF MyTB1 = (W///,"slesstdcw.SlEsTouchButton"/////70,20,200,100/0,0,0,0)
LOAD
   WRITECWPROPERTY("MyTB1", "text", "This is my first TouchButton !!!")
   WRITECWPROPERTY("MyTB1", "textPressed", "This is my first TouchButton (pressed)!!!")
   WRITECWPROPERTY("MyTB1", "picture", "dsm_remove_trashcan_red.png") WRITECWPROPERTY("MyTB1", "pic-
tureAlignment", "left")
   WRITECWPROPERTY("MyTB1", "scalePicture", FALSE)
   WRITECWPROPERTY("MyTB1", "picturePressed", "slsu_topology_empty_round_slot.png") WRITECWPRO-
PERTY("MyTB1", "picture", "slsu_topology_empty_slot_left_error.png")
END_LOAD
```

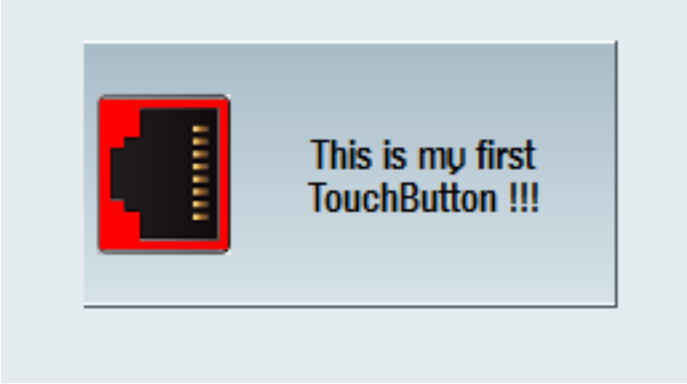


Figura 9-18 Ejemplo "This is my first TouchButton !!!"

9.6.2 Lectura y escritura de propiedades

Descripción

Las propiedades mencionadas en el siguiente capítulo se leen con la función ReadCWProperty() y se ajustan con WriteCWProperty().

Ejemplos

Lectura de la propiedad "Text" del SlEsTouchButton conectado a través de la variable de visualización "MyTouchButton". El resultado se escribe en el registro 0.

```
REG[0] = ReadCWProperty("MyTouchButton", "Text")
```

Escritura del valor "sk_ok.png" en la propiedad "Picture" del SlEsTouchButton conectado a través de la variable de visualización "MyTouchButton".

```
WriteCWProperty("MyTouchButton", "Picture", "sk_ok.png")
```

9.6.3 Propiedades

Vista general

Propiedad	Descripción
ButtonStyle	Estilo de representación del TouchButton
Flat	Representación plana o 3D
Enabled	Activar/desactivar manejo
Checkable	Activar/desactivar función de alternancia
Checked	El TouchButton está enclavado (checked)
ShowFocusRect	Mostrar recuadro de foco cuando el TouchButton tenga el foco de entrada
Picture	Imagen que se mostrará cuando el TouchButton se encuentre en el estado no pulsado normal
PicturePressed	Imagen que se mostrará cuando el TouchButton se encuentre pulsado o enclavado (checked)
PictureAlignment PictureAlignmentString	Orientación de la imagen
ScalePicture	La imagen se escala (se expande o se contrae)
PictureKeepAspectRatio	Comportamiento de expansión de la imagen
Text	Texto visualizado normal
TextPressed	Texto visualizado cuando el TouchButton está pulsado
textAlignment textAlignmentString	Orientación del texto
TextAlignedToPicture	El texto se alinea en relación a la imagen: activar o desactivar

Propiedad	Descripción
BackgroundPicture	Imagen de fondo visualizada
BackgroundPictureAlignment BackgroundPictureAlignmentString	Orientación de la imagen de fondo
ScaleBackgroundPicture	La imagen de fondo se escala (se expande o se contrae)
BackgroundPictureKeepAspectRatio	Comportamiento de expansión de la imagen
BackColor	Color de fondo cuando el TouchButton se encuentra en el estado no pulsado normal
BackColorChecked	Color de fondo cuando el TouchButton está enclavado (checked)
BackColorPressed	Color de fondo cuando el TouchButton está pulsado momentáneamente
BackColorDisabled	Color de fondo cuando el TouchButton no está habilitado
TextColor	Color del texto cuando el TouchButton se encuentra en el estado no pulsado normal
TextColorChecked	Color del texto cuando el TouchButton está enclavado (checked)
TextColorPressed	Color del texto cuando el TouchButton está pulsado momentáneamente
TextColorDisabled	Color del texto cuando el TouchButton no está habilitado

ButtonStyle: Estilo de representación

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**ButtonStyle**") WriteCWProperty(TouchButtonVarName, "**ButtonStyle**", Value)	
Descripción:	Lee o ajusta el estilo de visualización del TouchButton.	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Valor leído de la propiedad (int)
	Value	Valor que se desea ajustar (int) 0 = Softkey Look&Feel (predeterminado) 1 = Específico del usuario

Con el ajuste "0 = Softkey Look&Feel", el TouchButton tiene el mismo aspecto y comportamiento que un pulsador de menú. Se tiene en cuenta la skin ajustada actualmente (ver dato de máquina de visualización 9112 = $MM_HMI_SKIN).

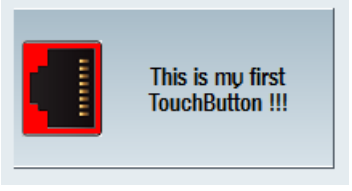


Figura 9-19 ButtonStyle: Softkey Look&Feel

Con el ajuste "1 = Específico del usuario", el color del texto y del fondo puede definirse en función del estado del TouchButton. Además puede conseguirse un efecto 3D, el escalado automático de imágenes y el ajuste de las distancias respecto a los márgenes.

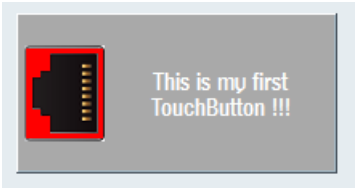


Figura 9-20 ButtonStyle: Específico del usuario

Flat: Tipo de representación

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**Flat**") WriteCWProperty(TouchButtonVarName, "**Flat**", Value)	
Descripción:	Lee o ajusta la representación plana (predeterminada) o en 3D del TouchButton. **Nota:** Esta propiedad solo está disponible con el estilo de representación (ButtonStyle) "1 = específico del usuario" activado.	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Valor leído de la propiedad (bool)
	Value	Valor que se desea ajustar (bool): TRUE (predeterminado) o FALSE

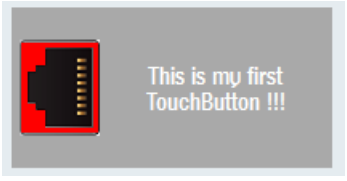


Figura 9-21 Tipo de representación plana

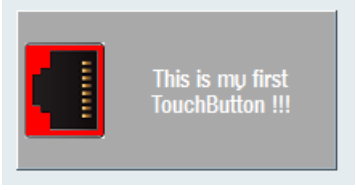


Figura 9-22 Tipo de representación 3D

Enabled: Habilitación del TouchButton

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**Enabled**") WriteCWProperty(TouchButtonVarName, "**Enabled**", Value)	
Descripción:	Lee o ajusta la habilitación (= TRUE) o deshabilitación (= FALSE) del manejo del TouchButton.	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Valor leído de la propiedad (bool)
	Value	Valor que se desea ajustar (bool): TRUE (predeterminado) o FALSE

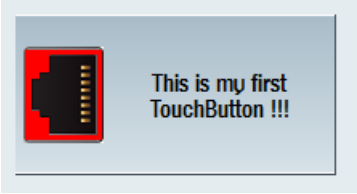


Figura 9-23 TouchButton habilitado

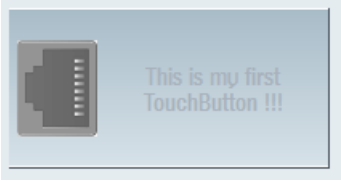


Figura 9-24 TouchButton inhabilitado

Nota

Al hacer clic en un TouchButton deshabilitado, se envía la señal "clickedDisabled". La señal se puede evaluar y, p. ej., se puede emitir un mensaje para explicar por qué el TouchButton y la función asociada a él no están habilitados en ese momento. En estado desactivado, la imagen visualizada o la imagen de fondo se muestran automáticamente en escala de grises.

Checkable: Activar/desactivar función de alternancia

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**Checkable**") WriteCWProperty(TouchButtonVarName, "**Checkable**", Value)	
Descripción:	Lee o ajusta la funcionalidad de alternancia del TouchButton.	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Valor leído de la propiedad (bool)
	Value	Valor que se desea ajustar (bool): TRUE o FALSE (predeterminado)

Nota

De modo predeterminado, el TouchButton vuelve automáticamente a su estado normal al soltarlo (como un pulsador). Si, por el contrario, la función de alternancia del TouchButton está activada (TRUE), este permanece inicialmente en la posición presionada al pulsarlo. Si se vuelve a accionar, pasa a su estado normal (como un interruptor).

Ver también la propiedad "Checked".

Checked: Estado actual de alternancia

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**Checked**") WriteCWProperty(TouchButtonVarName, "**Checked**", Value)	
Descripción:	Lee o ajusta el estado actual de alternancia del TouchButton.	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Valor leído de la propiedad (bool)
	Value	Valor que se desea ajustar (bool): TRUE o FALSE (predeterminado)

Si la función de alternancia está ajustada a "TRUE" con la propiedad "Checkable", la propiedad "Checked" permite leer o ajustar el estado de alternancia actual.

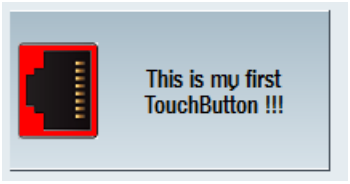


Figura 9-25 Unchecked

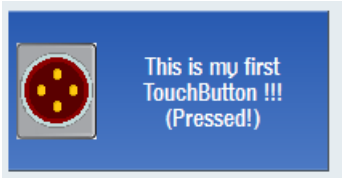


Figura 9-26 Checked

Pueden mostrarse un texto y una imagen específicos para los dos estados.

Nota

Ver también la propiedad "Checkable".

ShowFocusRect: Representación de un recuadro de foco

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**ShowFocusRect**") WriteCWProperty(TouchButtonVarName, "**ShowFocusRect**", Value)	
Descripción:	Lee o ajusta si debe mostrarse un recuadro de foco en el TouchButton cuando este tiene el foco de entrada.	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Valor leído de la propiedad (bool)
	Value	Valor que se desea ajustar (bool): TRUE o FALSE (predeterminado)

El color del recuadro de foco se define automáticamente en función de la configuración de color de Operate, en el siguiente ejemplo, "orange".

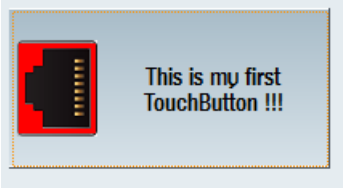


Figura 9-27 ShowFocusRect

Picture: Imagen visualizada

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**Picture**") WriteCWProperty(TouchButtonVarName, "**Picture**", Value)	
Descripción:	Lee o ajusta la imagen que se mostrará cuando el TouchButton se encuentre en estado de reposo (no pulsado).	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Valor leído de la propiedad (String)
	Value	Valor que se desea ajustar (String)

Se especifica como valor el nombre de fichero, por ejemplo, "sk_ok.png". El TouchButton determina automáticamente el fichero de imagen correcto en el directorio de resolución actual (ver capítulo Utilizar imágenes/gráficos (Página 65)).

En estado desactivado, la imagen visualizada se muestra automáticamente en escala de grises.

Nota

Ver también las propiedades "Text", "TextAlignedToPicture", "PictureAlignment", "ScalePicture", "PictureKeepAspectRatio".

PicturePressed: Imagen visualizada en estado pulsado

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**PicturePressed**") WriteCWProperty(TouchButtonVarName, "**PicturePressed**", Value)	
Descripción:	Lee o ajusta la imagen que se visualizará cuando el TouchButton se encuentre en estado pulsado o enclavado.	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Valor leído de la propiedad (String)
	Value	Valor que se desea ajustar (String)

Si no se especifica ningún valor (cadena vacía " "), en este estado el TouchButton muestra la imagen especificada en la propiedad "Picture".

Se especifica como valor el nombre de fichero, por ejemplo, "sk_ok.png". El TouchButton determina automáticamente el fichero de imagen correcto en el directorio de resolución actual (ver capítulo Utilizar imágenes/gráficos (Página 65)).

En estado desactivado, la imagen visualizada se muestra automáticamente en escala de grises.

Nota

Ver también las propiedades "Text", "TextAlignedToPicture", "PictureAlignment", "ScalePicture", "PictureKeepAspectRatio".

PictureAlignment: Orientación de la imagen

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**PictureAlignment**") WriteCWProperty(TouchButtonVarName, "**PictureAlignment**", Value)	
Descripción:	Lee o ajusta la orientación de la imagen visualizada en el TouchButton.	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Valor leído de la propiedad (int)
	Value	Valor que se desea ajustar (int): 1 = izquierda (predeterminado) 2 = derecha 32 = arriba 64 = abajo 128 = centrada

Nota

Ver capítulo Posicionamiento y orientación de imagen y texto (Página 272).
Ver también las propiedades "Text", "TextAlignedToPicture", "PictureAlignmentString", "ScalePicture", "PictureKeepAspectRatio".

PictureAlignmentString: Orientación de la imagen

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**PictureAlignmentString**") WriteCWProperty(TouchButtonVarName, "**PictureAlignmentString**", Value)	
Descripción:	Lee o ajusta la orientación de la imagen visualizada. Esta propiedad tiene el mismo significado que la propiedad "PictureAlignment". Sin embargo, en este caso el valor no se envía como número (int), sino como cadena de caracteres.	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Valor leído de la propiedad (String)
	Value	Valor que se desea ajustar (String): "Left" = izquierda (predeterminado) "Right" = derecha "Top" = arriba "Bottom" = abajo "Center" = centrada

Nota

Ver capítulo Posicionamiento y orientación de imagen y texto (Página 272).

Ver también las propiedades "Text", "TextAlignedToPicture", "PictureAlignment", "ScalePicture", "PictureKeepAspectRatio".

ScalePicture: Escalar la imagen visualizada

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**ScalePicture**") WriteCWProperty(TouchButtonVarName, "**ScalePicture**", Value)	
Descripción:	Lee o ajusta el escalado de la imagen visualizada en el TouchButton en función de la orientación.	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Valor leído de la propiedad (bool)
	Value	Valor que se desea ajustar (bool): TRUE o FALSE (predeterminado)

Nota

Ver capítulo Posicionamiento y orientación de imagen y texto (Página 272).

PictureKeepAspectRatio: Escalar la imagen visualizada manteniendo la relación de aspecto

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**PictureKeepAspectRatio**") WriteCWProperty(TouchButtonVarName, "**PictureKeepAspectRatio**", Value)	
Descripción:	Lee o ajusta el mantenimiento de la relación de aspecto de la imagen visualizada en caso de escalado (propiedad "ScalePicture" = "TRUE").	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Valor leído de la propiedad (bool)
	Value	Valor que se desea ajustar (bool): TRUE (predeterminado) o FALSE

Nota

Ver capítulo Posicionamiento y orientación de imagen y texto (Página 272).

Text: Texto visualizado

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**Text**") WriteCWProperty(TouchButtonVarName, "**Text**", Value)	
Descripción:	Lee o ajusta el texto que se mostrará cuando el TouchButton se encuentre en estado de reposo (no pulsado).	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Valor leído de la propiedad (String)
	Value	Valor que se desea ajustar (String)

En los textos se introducen automáticamente saltos de línea en los límites de palabras en el recuadro disponible.

Por requisitos del sistema, los saltos de línea solo pueden forzarse si el texto visualizado procede de un fichero de idioma.

Nota

Ver capítulo Textos dependientes del idioma (Página 274).

TextPressed: Texto visualizado en estado pulsado

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextPressed**") WriteCWProperty(TouchButtonVarName, "**TextPressed**", Value)	
Descripción:	Lee o ajusta el texto que se visualizará cuando el TouchButton se encuentre en estado pulsado o enclavado.	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Valor leído de la propiedad (String)
	Value	Valor que se desea ajustar (String)

En los textos se introducen automáticamente saltos de línea en los límites de palabras en el recuadro disponible.

Por requisitos del sistema, los saltos de línea solo pueden forzarse si el texto visualizado procede de un fichero de idioma.

Nota

Ver capítulo Textos dependientes del idioma (Página 274).

TextAlignment: Orientación del texto

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextAlignment**") WriteCWProperty(TouchButtonVarName, "**TextAlignment**", Value)
Descripción:	Lee o ajusta la orientación del texto en el TouchButton.

Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Valor leído de la propiedad (int)
	Value	Valor que se desea ajustar (int): 1 = izquierda 2 = derecha 32 = arriba 64 = abajo 128 = centrado (predeterminado) (ver ejemplos abajo)

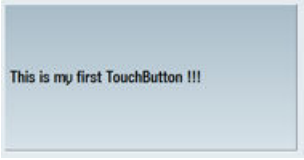


Figura 9-28 TextAlignment: izquierda

Figura 9-29 TextAlignment: derecha

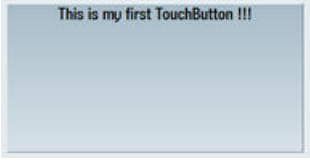


Figura 9-30 TextAlignment: arriba

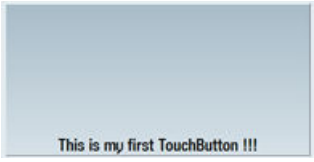


Figura 9-31 TextAlignment: abajo

Figura 9-32 TextAlignment: centrado

Nota

Ver capítulo Posicionamiento y orientación de imagen y texto (Página 272).

Ver también las propiedades "Text", "TextAlignmentString", "TextAlignedToPicture".

TextAlignmentString: Orientación del texto

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextAlignmentString**") WriteCWProperty(TouchButtonVarName, "**TextAlignmentString**", Value)	
Descripción:	Lee o ajusta la orientación del texto. Esta propiedad tiene el mismo significado que la propiedad "TextAlignment". En este caso el valor no se envía como número (int), sino como cadena de caracteres.	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Valor leído de la propiedad (String)
	Value	Valor que se desea ajustar (String): "Left" = izquierda "Right" = derecha "Top" = arriba "Bottom" = abajo "Center" = centrado (predeterminado)

Nota

Ver capítulo Posicionamiento y orientación de imagen y texto (Página 272).

Ver también las propiedades "Text", "TextAlignment", "TextAlignedToPicture".

TextAlignedToPicture: Orientación del texto respecto a la imagen

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "TextAlignedToPicture") WriteCWProperty(TouchButtonVarName, "**TextAlignedToPicture**", Value)	
Descripción:	Lee o ajusta el posicionamiento del texto visualizado respecto a la imagen. Si se ajusta FALSE, el texto se muestra centrado en el TouchButton.	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Valor leído de la propiedad (bool)
	Value	Valor que se desea ajustar (bool): TRUE (predeterminado) o FALSE

Nota

Ver capítulo Posicionamiento y orientación de imagen y texto (Página 272).
Ver también las propiedades "Text", "TextPressed", "Picture", "PicturePressed".

BackColor: Color de fondo

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, " **BackColor**") WriteCWProperty(TouchButtonVarName, " **BackColor**", Value)	
Descripción:	Lee o ajusta el color de fondo que se mostrará cuando el TouchButton se encuentre en estado de reposo (no pulsado).	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Color leído de la propiedad (String)
	Value	Valor que se debe ajustar (String) como valor RGB en el formato "#RRGGBB", p. ej., "#04B7FB"

Nota

Esta propiedad solo está disponible con el estilo de representación "1 = específico del usuario" activado.

BackColorChecked: Color de fondo en estado enclavado

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackColorChecked**") WriteCWProperty(TouchButtonVarName, " **BackColorChecked**", Value)	
Descripción:	Lee o ajusta el color de fondo cuando el TouchButton se encuentra en estado enclavado (función de alternancia). Ver las propiedades "Checked", "Checkable".	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Color leído de la propiedad (String)
	Value	Valor que se debe ajustar (String) como valor RGB en el formato "#RRGGBB", p. ej., "#04B7FB"

Nota

Esta propiedad solo está disponible con el estilo de representación "1 = específico del usuario" activado.

Nota

Ver también las propiedades "Checked", "Checkable".

BackColorPressed: Color de fondo en estado pulsado

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackColorPressed**") WriteCWProperty(TouchButtonVarName, " **BackColorPressed**", Value)	
Descripción:	Lee o ajusta el color de fondo cuando el TouchButton se encuentra en estado pulsado.	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Color leído de la propiedad (String)
	Value	Valor que se debe ajustar (String) como valor RGB en el formato "#RRGGBB", p. ej., "#04B7FB"

Nota

Esta propiedad solo está disponible con el estilo de representación "1 = específico del usuario" activado.

BackColorDisabled: Color de fondo en estado desactivado

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackColorDisabled**") WriteCWProperty(TouchButtonVarName, " **BackColorDisabled**", Value)	
Descripción:	Lee o ajusta el color de fondo cuando el TouchButton se encuentra en estado desactivado.	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Color leído de la propiedad (String)
	Value	Valor que se debe ajustar (String) como valor RGB en el formato "#RRGGBB", p. ej., "#04B7FB"

Nota

Esta propiedad solo está disponible con el estilo de representación "1 = específico del usuario" activado.

Nota

Ver también la propiedad "Enabled".

TextColor: Color del texto

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextColor**") WriteCWProperty(TouchButtonVarName, " **TextColor**", Value)
Descripción:	Lee o ajusta el color del texto que se mostrará cuando el TouchButton se encuentre en estado de reposo (no pulsado).

Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Color leído de la propiedad (String)
	Value	Valor que se debe ajustar (String) como valor RGB en el formato "#RRGGBB", p. ej., "#04B7FB"

Nota

Esta propiedad solo está disponible con el estilo de representación "1 = específico del usuario" activado.

TextColorChecked: Color del texto en estado enclavado

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextColorChecked**") WriteCWProperty(TouchButtonVarName, " **TextColorChecked**", Value)	
Descripción:	Lee o ajusta el color del texto cuando el TouchButton se encuentra en estado enclavado (función de alternancia).	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Color leído de la propiedad (String)
	Value	Valor que se debe ajustar (String) como valor RGB en el formato "#RRGGBB", p. ej., "#04B7FB"

Nota

Esta propiedad solo está disponible con el estilo de representación "1 = específico del usuario" activado.

Nota

Ver también las propiedades "Checked", "Checkable".

TextColorPressed: Color del texto en estado pulsado

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextColorPressed**") WriteCWProperty(TouchButtonVarName, "**TextColorPressed**", Value)	
Descripción:	Lee o ajusta el color del texto cuando el TouchButton se encuentra en estado pulsado.	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Color leído de la propiedad (String)
	Value	Valor que se debe ajustar (String) como valor RGB en el formato "#RRGGBB", p. ej., "#04B7FB"

Nota

Esta propiedad solo está disponible con el estilo de representación "1 = específico del usuario" activado.

TextColorDisabled: Color del texto en estado desactivado

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**TextColorDisabled**") WriteCWProperty(TouchButtonVarName, "**TextColorDisabled**", Value)	
Descripción:	Lee o ajusta el color del texto cuando el TouchButton se encuentra en estado desactivado.	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Color leído de la propiedad (String)
	Value	Valor que se debe ajustar (String) como valor RGB en el formato "#RRGGBB", p. ej., "#04B7FB"

Nota

Esta propiedad solo está disponible con el estilo de representación "1 = específico del usuario" activado.

Nota

Ver también la propiedad "Enabled".

BackgroundPicture: Imagen de fondo

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackgroundPicture**") WriteCWProperty(TouchButtonVarName, "**BackgroundPicture**", Value)	
Descripción:	Lee o ajusta la imagen de fondo visualizada permanente, es decir, independiente del estado.	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Valor leído de la propiedad (String)
	Value	Valor que se desea ajustar (String)

Se especifica como valor el nombre de fichero, por ejemplo, "sk_ok.png". El TouchButton determina automáticamente el fichero de imagen correcto en el directorio de resolución actual (ver capítulo Utilizar imágenes/gráficos (Página 65)).

En estado desactivado, la imagen de fondo se muestra automáticamente en escala de grises.

Nota

Ver capítulo Utilizar imágenes/gráficos (Página 65).

Ver también las propiedades "BackgroundPictureAlignment", "ScaleBackgroundPicture", "BackgroundPictureKeepAspectRatio".

BackgroundPictureAlignment: Orientación de la imagen de fondo

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackgroundPictureAlignment**") WriteCWProperty(TouchButtonVarName, "**BackgroundPictureAlignment**", Value)	
Descripción:	Lee o ajusta la orientación de la imagen de fondo.	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Valor leído de la propiedad (int)
	Value	Valor que se desea ajustar (int): 1 = izquierda 2 = derecha 32 = arriba 64 = abajo 128 = centrada (predeterminada)

Nota

Ver capítulo Posicionamiento y orientación de imagen y texto (Página 272).

Ver también las propiedades "BackgroundPictureAlignment", "ScaleBackgroundPicture", "BackgroundPictureKeepAspectRatio".

BackgroundPictureAlignmentString: Orientación de la imagen de fondo

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackgroundPictureAlignmentString**") WriteCWProperty(TouchButtonVarName, "**BackgroundPictureAlignmentString**", Value)
Descripción:	Lee o ajusta la orientación de la imagen de fondo. Esta propiedad tiene el mismo significado que la propiedad "BackgroundPictureAlignment". Sin embargo, en este caso el valor no se envía como número (int), sino como cadena de caracteres.

Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Valor leído de la propiedad (int)
	Value	Valor que se desea ajustar (int): "Left" = izquierda "Right" = derecha "Top" = arriba "Bottom" = abajo "Center" = centrada (predeterminada)

Nota

Ver capítulo Posicionamiento y orientación de imagen y texto (Página 272).

Ver también las propiedades "BackgroundPictureAlignment", "ScaleBackgroundPicture", "BackgroundPictureKeepAspectRatio".

ScaleBackgroundPicture: Escalar la imagen de fondo

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**ScalePicture**") WriteCWProperty(TouchButtonVarName, "**ScalePicture**", Value)	
Descripción:	Lee o ajusta el escalado de la imagen visualizada en el TouchButton en función de la orientación.	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Valor leído de la propiedad (bool)
	Value	Valor que se desea ajustar (bool): TRUE o FALSE (predeterminado)

Nota

Ver capítulo Posicionamiento y orientación de imagen y texto (Página 272).

Ver también las propiedades "BackgroundPictureAlignment", "ScaleBackgroundPicture", "BackgroundPictureKeepAspectRatio".

BackgroundPictureKeepAspectRatio: Escalar la imagen de fondo manteniendo la relación de aspecto

Sintaxis:	ReturnValue = ReadCWProperty(TouchButtonVarName, "**BackgroundPictureKeepAspectRatio**") WriteCWProperty(TouchButtonVarName, "**BackgroundPictureKeepAspectRatio**", Value)
Descripción:	Lee o ajusta el mantenimiento de la relación de aspecto de la imagen de fondo en caso de escalado (propiedad "ScaleBackgroundPicture" = "TRUE").

Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Valor leído de la propiedad (bool)
	Value	Valor que se desea ajustar (bool): TRUE (predeterminado) o FALSE

Nota

Ver capítulo Posicionamiento y orientación de imagen y texto (Página 272).

9.6.4 Funciones

Las funciones que se indican a continuación pueden llamarse con la función CallCWMethod().

Ejemplo

Ajuste de los márgenes del SlEsTouchButton conectado a través de la variable de visualización "MyTouchButton".

El resultado se escribe en el registro 0.

```
REG[0] = CallCWMethod("MyTouchButton", "setMargins", 20, 20, 20, 20,
20)
```

Vista general

Función	Descripción
setMargins	Ajuste de los márgenes
serialize	Guardar/restablecer el estado actual

setMargins: Ajuste de los márgenes

Sintaxis:	ReturnValue = CallCWMethod(TouchButtonVarName, "**setMargins**", left, top, right, bottom, center) ReturnValue = CallCWMethod(TouchButtonVarName, "**setMargins**", left, top, right, bottom, center, marginAlignment)
Descripción:	Con esta función se definen los márgenes y la distancia entre la imagen y el texto. Si se especifica un valor con "-1", para este valor se aplicará el margen predeterminado del sistema. Si el valor es mayor o igual que "0", se aplicará como margen.

Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Errorcode (bool): TRUE = correcto
	left	margen izquierdo (int)
	top	margen superior (int)
	right	margen derecho (int)
	bottom	margen inferior (int)
	center	Distancia entre imagen y texto (int) si la alineación no es "center"
	marginAlignment	Opcional: Determina si el margen, además de para la imagen visualizada, debe aplicarse también para la posición del texto visualizado (bool), TRUE (predeterminado)

serialize: Guardar/restablecer el estado actual

Sintaxis:	ReturnValue = CallCWMethod(TouchButtonVarName, "**serialize**", FilePath, IsStoring)	
Descripción:	Si es necesario, esta función permite escribir el estado actual y el contenido del SlEsTouchButton en formato binario en un fichero o DataStream, así como restablecerlo. Esta función actualiza la vista automáticamente.	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	Return Value	Errorcode (bool): TRUE = correcto
	FilePath	Ruta completa con nombre de fichero (QString)
	IsStoring	Guardar/restablecer (bool): TRUE = guardar

Nota

El personal de configuración no utiliza directamente esta función.

9.6.5 Señales

Las señales que se describen a continuación pueden recogerse en la configuración para definir una respuesta adecuada.

Vista general

Función	Descripción
clickedDisabled	Accionamiento de un TouchButton desactivado
clicked	Se ha efectuado un clic o un toque
checked	Se ha conmutado

- Si un TouchButton está habilitado y no es conmutable, al accionarlo se envía la señal "clicked".
- Si un TouchButton está habilitado y es conmutable, al accionarlo se envía la siguiente secuencia de señales:
  "checked" → "clicked"
- Si un TouchButton no está habilitado, al accionarlo se envía la señal "clickedDisabled".

Todas las señales mencionadas se envían siempre después de una intervención del operador, es decir, después de que se haya soltado el ratón o la barra espaciadora o después de efectuar un toque en modo multitáctil.

Esto hace que el SlEsTouchButton posea una funcionalidad de mero interruptor, es decir, no permite implementar funciones de pulsador.

Nota

En el modo de manejo multitáctil, tenga en cuenta que el clic no se envía hasta que se ha detectado por completo el gesto de toque (pulsar y soltar en un intervalo de aprox. 0,7 s). Si se realizara una acción al efectuar una sola pulsación, no sería posible, p. ej., el desplazamiento de la ScrollArea situada en segundo plano o bien se producirían fácilmente fallos de manejo no deseados.

clicked: Clic en un TouchButton

Sintaxis:	SUB(on_<TouchButtonVarName>_clicked) ... END_SUB	
Descripción:	Una secuencia completa de pulsar y soltar un TouchButton habilitado da como resultado una señal "clicked". Se recomienda trabajar habitualmente con esta señal.	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	SIGARG[0]	Devuelve el estado de alternancia del TouchButton (bool)

Ejemplo

```
DEF MyTouchButton = (W///,"slesstdcw.SlEsTouchButton"/////70,20,200,100/0,0,0,0)
SUB(on_MyTouchButton_clicked)
   DLGL("checked: " << SIGARG[0])
END_SUB
```

checked: Conmutación de un TouchButton

Sintaxis:	SUB(on_<TouchButtonVarName>_checked) ... END_SUB	
Descripción:	Se ha pulsado un botón conmutable, por lo que su estado se ha modificado.	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	SIGARG[0]	Devuelve el estado de alternancia del TouchButton (bool)

Ejemplo

```
DEF MyTouchButton = (W///,"slesstdcw.SlEsTouchButton"/////70,20,200,100/0,0,0,0)
SUB(on_MyTouchButton_toggled)
   DLGL("toggled: " << SIGARG[0])
END_SUB
```

clickedDisabled: Clic en un TouchButton no habilitado

Sintaxis:	SUB(on_<TouchButtonVarName>_clickedDisabled) ... END_SUB	
Descripción:	Una secuencia completa de pulsar y soltar un TouchButton no habilitado da como resultado una señal "clickedDisabled".	
Parámetros:	TouchButtonVarName	Nombre de la variable de visualización que contiene un SlEsTouchButton
	SIGARG[0]	Devuelve el estado de alternancia del TouchButton (bool)

Ejemplo

```
DEF MyTouchButton = (W///,"slesstdcw.SlEsTouchButton"/////70,20,200,100/0,0,0,0)
SUB(on_MyTouchButton_clickedDisabled)
   DLGL("checkedDisabled: " << SIGARG[0])
END_SUB
```

9.6.6 Posicionamiento y orientación de imagen y texto

Posicionamiento de imágenes

Con la orientación (alignment) especificada se posiciona una imagen de la siguiente manera:

- En el primer paso se determina la imagen adecuada para la resolución actual del directorio de resolución correspondiente.
- Después, el recuadro de la interfaz (ClientArea) se reduce en los valores correspondientes a los márgenes especificados (MarginArea).
  Se puede modificar la MarginArea con la función "setMargins":
  - setMargins(-1, -1, -1, -1, -1)

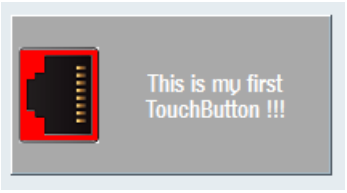


Figura 9-33 setMargins(-1, -1, -1, -1, -1)

  - setMargins(0, 0, 0, 0, 0)

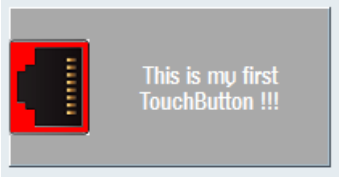


Figura 9-34 setMargins(0, 0, 0, 0, 0)

  - setMargins(20, 20, 20, 20, 20)

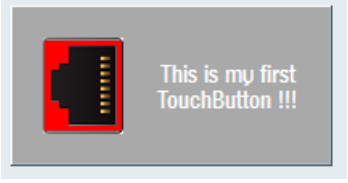


Figura 9-35 setMargins(20, 20, 20, 20, 20)

Ejemplo de orientación "izquierda"

La superficie se divide horizontalmente en dos mitades, de modo que la imagen visualizada puede ocupar como máximo la mitad de la superficie.

La pantalla se representa entonces del modo siguiente:

- Propiedad "scalePicture": FALSE
  La imagen se reproduce en el lugar indicado sin escalado. Hay que procurar que la imagen original no sea demasiado grande para que pueda visualizarse por completo dentro del TouchButton.

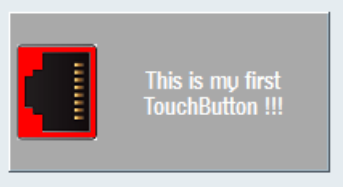


Figura 9-36 scalePicture FALSE

- Propiedad "scalePicture": TRUE
  La imagen se escala (se expande o se contrae) hasta que cabe en la mitad izquierda del TouchButton. Al mismo tiempo se aplica la propiedad "pictureKeepAspectRatio" del modo siguiente:
  - Propiedad "pictureKeepAspectRatio": FALSE
    La imagen se escala en horizontal y vertical hasta que cabe exactamente en la mitad izquierda del TouchButton. No se mantiene la relación de aspecto de la imagen original.

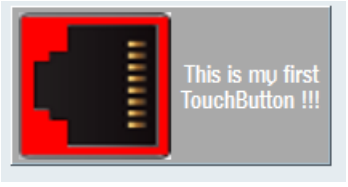


Figura 9-37 scalePicture - pictureKeepAspectRatio FALSE

  - Propiedad "pictureKeepAspectRatio": TRUE
    La imagen se escala en la mitad izquierda del TouchButton en el máximo tamaño posible manteniendo la relación de aspecto de la imagen original.

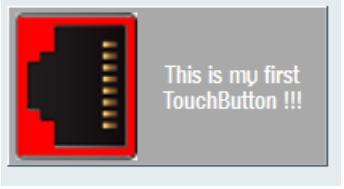


Figura 9-38 scalePicture - pictureKeepAspectRatio TRUE

Nota

Para las orientaciones "derecha", "arriba" y "abajo" se aplica el mismo principio.

Con la orientación "centrada" se intenta encajar la imagen en toda la extensión de la MarginArea aplicando las propiedades "scalePicture" y "pictureKeepAspectRatio".

Posicionamiento del texto

El texto se posiciona de la siguiente manera en función de la propiedad "textAlignedToPicture":

- Propiedad "textAlignedToPicture": FALSE
  El texto se muestra centrado en vertical y horizontal en la MarginArea.

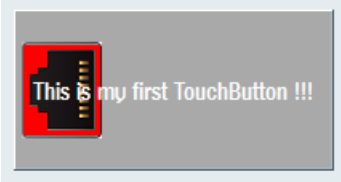


Figura 9-39 textAlignedToPicture FALSE

- Propiedad "textAlignedToPicture": TRUE
  El texto se muestra centrado en horizontal y vertical en la extensión de la MarginArea descontando el área de la imagen visualizada.

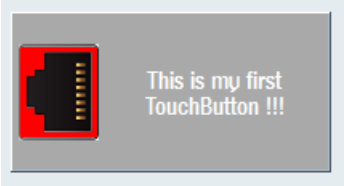


Figura 9-40 textAlignedToPicture TRUE

9.6.7 Textos dependientes del idioma

SlEsTouchButton no posee soporte propio para idiomas extranjeros. Sin embargo, es posible implementar una dependencia de idioma del modo que se muestra en el siguiente ejemplo:

Ejemplo

easyscreen.ini:

```
[LANGUAGEFILES]LngFile03 = user.txt
```

user_eng.txt:

```
85000 0 0 "This is my first Touchbutton !!!"
```

user_deu.txt:

```
85000 0 0 "Este es mi primer TouchButton"
```

Fichero de configuración:

```
//M(MyTBMask/"My CustomWidget TouchButton ...")
DEF MyTB1 = (W///,"slesstdcw.SlEsTouchButton"/////70,20,200,100/0,0,0,0)
LOAD
```

```
//M(MyTBMask/"My CustomWidget TouchButton ...")
DEF MyTB1 = (W///,"slesstdcw.SlEsTouchButton"/////70,20,200,100/0,0,0,0)
   WRITECWPROPERTY("MyTB1", "text", $85000)
END_LOAD
LANGUAGE
   WRITECWPROPERTY("MyTB1", "text", $85000)
END_LANGUAGE
```

En los textos se introducen automáticamente saltos de línea en los límites de palabras en el recuadro disponible. El salto de línea forzado solo es posible si el texto visualizado procede de un fichero de idioma. Para ello es necesario insertar un "%n" en el lugar correspondiente del texto (ver el ejemplo siguiente).

user_eng.txt:

```
85001 0 0 "This is%nmy%nfirst%nTouchButton !!!"
```

Fichero de configuración:

```
WRITECWPROPERTY("MyTB1", "text", $85001)
```

Resultado

Figura 9-41 Ejemplo de salto de línea en texto dependiente de idioma

Campo de manejo "Custom" 10

10.1 Procedimiento para activar el campo de manejo "Custom"

Activar campo de manejo "Custom"

El campo de manejo "Custom" no está activado en el momento de la entrega.

1. Copie primero el fichero "slamconfig.ini" del directorio [Directorio Siemens del sistema]/cfg al [Directorio OEM del sistema]/cfg
   o, análogamente, al [Directorio de addon del sistema]/cfg o [Directorio de usuario del sistema]/cfg.
2. Para activar el campo de manejo "Custom", se necesita la siguiente entrada:
   ```
   [Custom]
   Visible=True
   ```

Resultado

Después de la activación, el pulsador de menú para el campo de manejo "Custom" se encuentra en el menú principal (F10) en la barra de conmutación de menús en HSK4 (= ajuste predeterminado).

El campo de manejo "Custom" muestra una ventana vacía en todo el campo de manejo con un título configurable. Todos los pulsadores de menú horizontales y verticales son configurables.

10.2 Procedimiento para configurar el pulsador de menú para "Custom"

Configurar el pulsador de menú para el campo de manejo "Custom"

En el fichero "slamconfig.ini" se configuran el rótulo y la posición del pulsador de menú para el campo de manejo "Custom".

La configuración del pulsador de menú de inicio se puede realizar de distintas maneras:

1. Para sustituir el rótulo del pulsador de menú por un **texto dependiente del idioma**, en la sección [Custom] deben realizarse las siguientes entradas:
   ```
   TextId=MY_TEXT_ID
   TextFile=mytextfile
   TextContext=mycontext
   ```
   En este ejemplo, el pulsador de menú muestra el texto dependiente del idioma que se ha guardado con la ID de texto "MY_TEXT_ID" dentro de "MyContext" en el fichero de texto mytextfile_xxx.qm (siendo xxx la abreviatura del idioma).
2. Para sustituir el rótulo del pulsador de menú por un **texto independiente del idioma**, en la sección [Custom] deben realizarse las siguientes entradas:
   ```
   TextId=HELLO
   TextFile=<empty>
   TextContext=<empty>
   ```
   En este ejemplo, el pulsador de menú para el campo de manejo "Custom" muestra el texto "HELLO" en todos los idiomas.
3. Además del texto, en el pulsador de menú se puede mostrar también **un pictograma**. Para ello, en la sección [Custom] se necesita la siguiente entrada:
   ```
   Picture=mypicture.png
   ```
   Entonces, el pulsador de menú muestra el pictograma del fichero mypicture.png. Los gráficos y los mapas de bits se guardan en la siguiente ruta: [Directorio OEM del sistema]/ico/ico<Resolución>. Se debe utilizar el directorio correspondiente a la resolución de la pantalla.
4. Además, puede ajustarse **la posición** del pulsador de menú. Para ello, en la sección [Custom] existe la siguiente entrada:
   ```
   SoftkeyPosition=12
   ```
   La posición 12 es el ajuste predeterminado. Esto corresponde a HSK4 en la barra de conmutación de menús del menú de campos de manejo. La posición 1-8 corresponde a HSK1 hasta HSK8 en la barra de menús; la posición 9-16 corresponde a HSK1 hasta HSK8 en la barra de conmutación de menús.

10.3 Procedimiento para configurar el campo de manejo "Custom"

Configurar el pulsador de menú para el campo de manejo "Custom"

Para configurar el campo de manejo, necesita los ficheros "easyscreen.ini" y "custom.ini". En el directorio [Directorio Siemens del sistema]/templates/cfg se encuentran plantillas para ambos ficheros.

1. Copie primero los ficheros en el directorio [Directorio OEM del sistema]/cfg y realice allí las modificaciones.
2. En el fichero "easyscreen.ini" ya se incluye una línea de definición para el campo de manejo "Custom":
   ```
   ;StartFile02 = area := Custom, dialog := SlEsCustomDialog,
   startfile := custom.com
   ```
   El signo ";" al principio de la línea representa el carácter de comentario. Así pues, la línea tiene un comentario y, por lo tanto, no está activa. Para modificarlo, debe borrarse el signo ";". Con el atributo "startfile" en esta línea se define que la entrada hace referencia al fichero de proyecto "custom.com" al seleccionar el campo de manejo "Custom".

3. El **fichero de proyecto "custom.com"** se crea en el directorio [Directorio OEM del sistema]/proj. En él está incluida la configuración, que se crea de forma análoga al fichero "aeditor.com" del campo de manejo "Programa". Los pulsadores de menú de inicio configurados se visualizan acto seguido en el campo de manejo "Custom".
4. En el fichero "custom.ini" se configura el **texto independiente del idioma** para la línea de título del diálogo.
   Para ello, en la plantilla se encuentra la siguiente entrada:
   ```
   [Header]
   Text=Custom
   ```
   Puede sustituir este texto por un texto específico del usuario.
5. Para configurar una **imagen inicial** del campo de manejo "Custom", en la plantilla se encuentra la siguiente entrada:
   ```
   [Picture]
   Picture=logo.png
   ```
   Logo.png es el nombre de la imagen inicial que se visualiza en el diálogo inicial del campo de manejo "Custom". Aquí puede, p. ej., mostrar un logo de empresa u otra imagen. El fichero se debe guardar en el directorio de la correspondiente resolución en: [Directorio OEM del sistema]/ico/ ...
6. Para mostrar una máscara de "Run MyScreens" determinada directamente al visualizar por primera vez el campo de manejo "Custom", se especifica la siguiente entrada en la plantilla:
   ```
   ; Mask shown with startup of area "custom"
   [Startup]
   ;Startup = Mask:=MyCustomStartupMask, File:=mycustommasks.com
   ```
   El caso de aplicación típico consiste, p. ej., en que al iniciar SINUMERIK Operate se inicie directamente una máscara de "Run MyScreens" determinada.
   Para ello, en la sección "[miscellaneous]" del fichero de configuración "systemconfiguration.ini" debe ajustarse la clave "startuparea" como sigue:
   ```
   [miscellaneous]
   ;name of the area to be shown at system startup
   startuparea = Custom
   ```

10.4 Ejemplo de programación para el campo "Custom"

Ejemplo

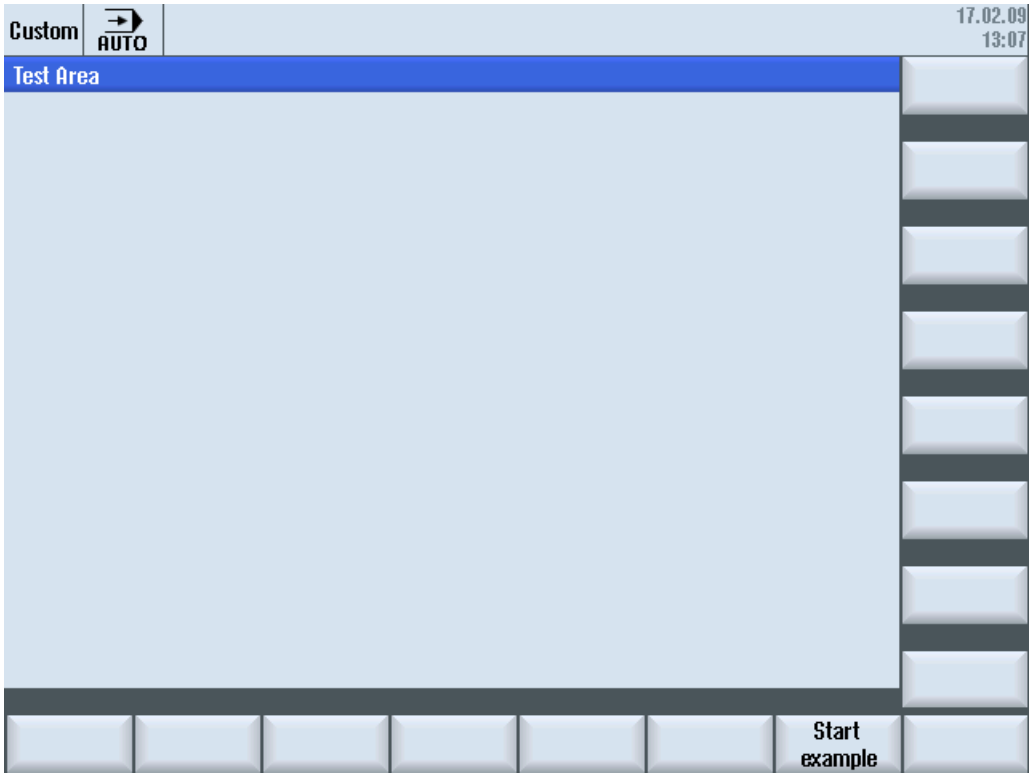


Figura 10-1 Ejemplo con el pulsador de menú "Start example"

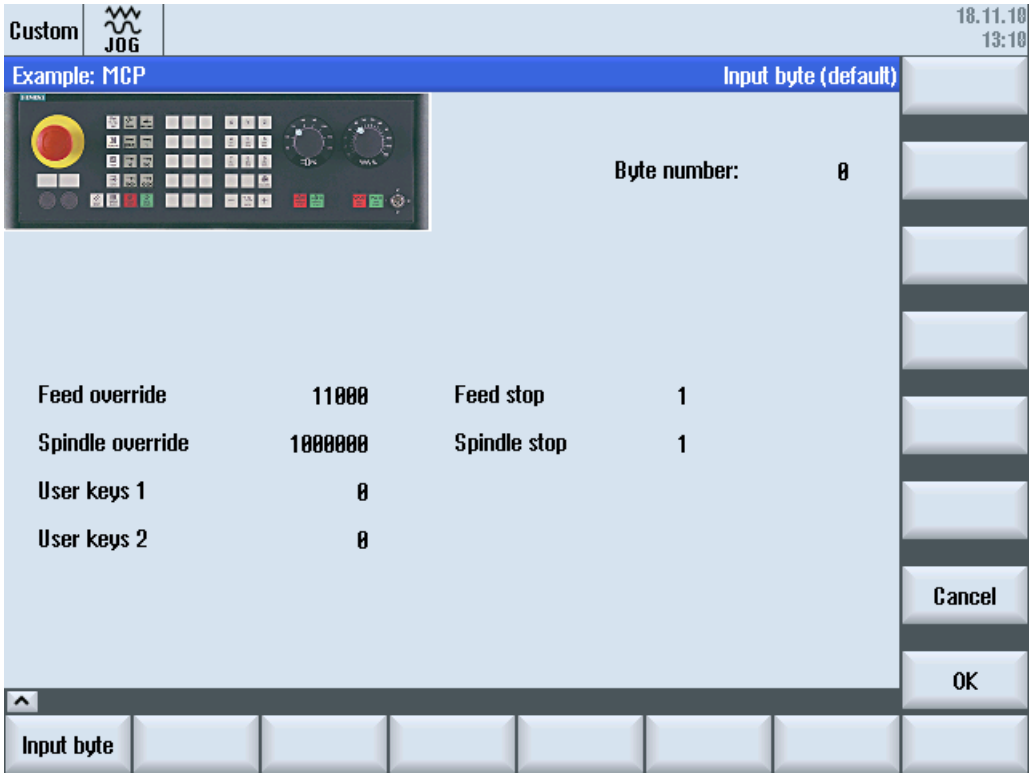


Figura 10-2 Ejemplo con mapa de bits y campos de texto

Lista de ficheros

Se necesitan los siguientes ficheros:

- "custom.com"
- "easyscreen.ini"

Programación

Contenido del fichero "custom.com":

Nota

El fichero gráfico "mcp.png" se incluye a modo de ejemplo. Si desea programar por su parte este ejemplo deberá reemplazar tal gráfica por una que tenga disponible.

```
//S(Start)
   HS7=("Start example", se1, ac7)
   PRESS(HS7)
      LM("Maske4")
   END_PRESS
```

```
//END
//M(Maske4/"Example: MCP"/"mcp.png")
   DEF byte=(I/0/0/"Input byte=0 (default)","Byte number:",""/wr1,li1///380,40,100/480,40,50)
   DEF Feed=(IBB//0/"","Feed override",""/wr1//"EB3"/20,180,100/130,180,100), Axistop=(B//0/"","Feed
stop",""/wr1//"E2.2"/280,180,100/380,180,50/100)
   DEF Spin=(IBB//0/"","Spindle override",""/wr1//"EB0"/20,210,100/130,210,100), spinstop=(B//
0/"","Spindle stop",""/wr1//"E2.4"/280,210,100/380,210,50/100)
   DEF custom1=(IBB//0/"," User keys 1",""/wr1//"EB7.7"/20,240,100/130,240,100)
   DEF custom2=(IBB//0/"","User keys 2",""/wr1//"EB7.5"/20,270,100/130,270,100)
   DEF By1
   DEF By2
   DEF By3
   DEF By6
   DEF By7

   HS1=("Input byte", SE1, AC4)
   HS2=("")
   HS3=("")
   HS4=("")
   HS5=("")
   HS6=("")
   HS7=("")
   HS8=("")
   VS1=("")
   VS2=("")
   VS3=("")
   VS4=("")
   VS5=("")
   VS6=("")
   VS7=("Cancel", SE1, AC7)
   VS8=("OK", SE1, AC7)

   PRESS(VS7)
      EXIT
   END_PRESS

   PRESS(VS8)
      EXIT
   END_PRESS

   LOAD
      By1=1
      By2=2
      By3=3
      By6=6
```

```
      By7=7
   END_LOAD

   PRESS(HS1)
      Byte.wr=2
   END_PRESS

   CHANGE(Byte)
      By1=byte+1
      By2=byte+2
      By3=byte+3
      By6=byte+6
      By7=byte+7
      Feed.VAR="EB"<<By3
      Spin.VAR="EB"<<Byte
      Custom1.VAR="EB"<<By6
      Custom2.VAR="EB"<<By7
      Axisstop.VAR="E"<<By2<<".2"
      Spinstop.VAR="E"<<By2<<".4"
      Byte.wr=1
   END_CHANGE

   CHANGE(Axis stop)
      IF Axistop==0
         Axistop.BC=9
      ELSE
         Axistop.BC=11
      ENDIF
   END_CHANGE

   CHANGE(Spin stop)
      IF Spinstop==0
         Spinstop.BC=9
      ELSE
         Spinstop.BC=11
      ENDIF
   END_CHANGE
//END
```

Selección de diálogos 11

11.1 Selección de diálogos mediante pulsadores de menú de PLC

Configuración

Descripción del procedimiento:

- En "systemconfiguration.ini" existe una sección [keyconfiguration]. La entrada especifica una acción para un pulsador de menú de PLC especial.
- Se indica como acción un número que, si es mayor o igual que 100, se trata de una llamada de "Run MyScreens".
- En el fichero "easyscreen.ini" debe crearse una sección para definir la acción que va a ejecutarse, cuyo nombre se obtiene a partir de los nombres del campo de manejo y del diálogo (ver entrada en [keyconfiguration] → Area:=..., Dialog:=...) → [<Área>_<Diálogo>] → p. ej., [AreaParameter_SlPaDialog]
- En esta sección se definen los números de acción (que se han indicado en "systemconfiguration.ini" → ver Action:=...). Se trata de dos comandos:
  1. LS("MenuPulsadores1","param.com") ... Cargar un menú de pulsadores
  2. LM("Mascara1","param.com") ... Cargar una máscara

Selección de menús de pulsadores a través de pulsadores de menú de PLC

En "Run MyScreens", la selección de menús de pulsadores y diálogos de "Run MyScreens" se puede realizar a través de pulsadores de menú de PLC. Para ello, el atributo "action" que debe indicarse al configurar los pulsadores de menú de PLC en cuestión debe tener un valor mayor o igual que 100.

La configuración de los pulsadores de menú de PLC se realiza en el fichero "systemconfiguration.ini" en la sección [keyconfiguration]:

```
[keyconfiguration]
KEY75.1 = Area:=area, Dialog:=dialog, Screen:=screen, Action:= 100,
```

La configuración de los comandos LM y LS, que deben ejecutarse al recibir los pulsadores de menú de PLC correspondientes, se realiza en el fichero "easyscreen.ini", concretamente en secciones cuyo nombre tiene el patrón siguiente:

`[areaname_dialogname]`	La primera parte del nombre, "areaname", designa el campo de manejo; la segunda parte, "dialogname" designa el diálogo para el cual son válidos los comandos configurados en esta sección.

`[AreaParameter_SlPaDialog]`
`100.screen1 =`
`LS("PulsadorMenu1","param.com")`
`101.screen3 = LM("Mascara1","param.com")`
	Deben utilizarse los nombres que se han asignado al campo de manejo y al diálogo en el fichero "systemconfiguration.ini". La indicación del diálogo es opcional.
	Puede omitirse, especialmente en campos de manejo que se implementan solamente con un único diálogo (ver ejemplo de al lado).
	Si en el campo de manejo AreaParameter, que se implementa con el diálogo SlPaDialog, se muestra "screen1", al producirse la "action" con el valor 100 se ejecuta el comando "LS("Pulsador-Menu1","param.com")".

`action.screen=Comando`	Los dos atributos "action" y "screen" indican de forma unívoca cuándo se ejecutará el comando indicado.
	La indicación de "screen" es opcional.
	Los comandos admisibles son:
	`LM (LoadMask)`
	`LS (LoadSoftkeys)`

11.1.1 Llamada de máscaras de ciclo Run MyScreens en el editor de programas

Campo de aplicación

Por medio de teclas de PLC se pueden abrir directamente máscaras de ciclo Run MyScreens en el editor de programas.

Condición

Solo se puede seleccionar ciclo tecnológico estando abierto el editor de programas. El ciclo tecnológico parametrizado en la máscara se inserta en la posición actual del cursor en el editor de programas.

Procedimiento

1. Cree la llamada de ciclo Run MyScreens con hardkeys (Página 288) de PLC.
2. Para que la máscara Run MyScreens se integre correctamente en el editor de programas, debe ampliarse la configuración de teclas de PLC como se describe en el siguiente ejemplo:
   Ejemplo de una configuración completa de tecla:
   systemconfiguration.ini
   ```
   [keyconfiguration]
   KEY101.0 = action:=209, runmyscreenmode:=HandledByDialog
   ```
   easyscreen.ini
   ```
   [AreaProgramEdit_SlProgramEdit]
   209 = LM("MASK_E_DR_O1_MAIN","e_dr_o1.com")
   ```
3. La respuesta de la máscara de ciclo abierta Run MyScreens se puede comprobar en la interfaz de PLC "DB19.DBW24".
   Ejemplo:
   En este ejemplo se escribe el valor "9876" en la interfaz de PLC cuando se abre la máscara Run MyScreens:
   Configuración con ID de PLC=9876:
   //M(MASK_E_DR_O1_MAIN/$85305////XG1,FA2,TA7//"drilling.txt"/**9876**)
   o bien
   //M{ MASK_E_DR_O1_MAIN, HD=$85305, LANGFILELIST=" drilling.txt", **PLC_ID=9876**}
   Ver también: Manual de funciones SINUMERIK 840D sl Funciones básicas

Estado del editor de programas en la interfaz OA

Se puede consultar el estado del editor de programas mediante la variable local de CAP "/Hmi/StepEditorInfo".

Nombre de la variable:	/Hmi/StepEditorInfo
Descripción de la variable:	Informaciones del editor sobre el programa seleccionado.
La cadena contiene la siguiente información:	
[fileName]	Ruta de acceso del programa
[channel]	Número de canal
[technlogy]	Tecnología
[appearance]	Identificador de programa (ShopMill/ShopTurn/G-Code)
[state]	Estado del editor Step: • EditorClosed: Editor cerrado (p. ej., HMI ausente del campo de manejo Programa) • EditEnabled: Permitido editar • EditDisabled: No está permitido editar (p. ej., readOnly o posición no válida del cursor)

11.2 Selección de diálogos mediante hardkeys de PLC

Campo de aplicación

Desde el PLC pueden iniciarse en el software de manejo las siguientes funciones:

- Selección de campo de manejo
- Selección de escenarios concretos dentro de los campos de manejo
- Ejecución de funciones configuradas en pulsadores de menú

Hardkeys

Todas las teclas (también las del PLC) se denominan en adelante "hardkeys" (teclas físicas). Es posible definir un máximo de 254 hardkeys. Se aplica la siguiente división:

Número de tecla	Utilización
Tecla 1 – tecla 9	Teclas del panel de operador
Tecla 10 – tecla 49	Reservado
Tecla 50 – tecla 254 Tecla 50 – tecla 81	Teclas del PLC: reservadas para OEM
Tecla 255	Predefinidas con información de control.

Las hardkeys 1 - 9 están predefinidas del siguiente modo:

Nombre de teclas		Acción/efecto
HK1	MACHINE	Selección de campo de manejo "Máquina", último diálogo
HK2	PROGRAM	Selección de campo de manejo "Programa", último diálogo o último programa
HK3	OFFSET	Selección de campo de manejo "Parámetros", último diálogo
HK4	PROGRAM MANAGER	Selección de campo de manejo "Programa", pantalla básica "Gestor de programas"
HK5	ALARM	Selección de campo de manejo "Diagnóstico", diálogo "Lista de alarmas"
HK6	CUSTOM	Campo de manejo "Custom"
HK7 1)	MENU SELECT	Selección "Menú inicial"
HK8 1)	MENU FUNCTION	Campo de manejo "Function"
HK9 1)	MENU USER	Campo de manejo "User"

1) solo para 828D

Configuración

La configuración se efectúa en el fichero de configuración "systemconfiguration.ini" en la sección [keyconfiguration]. Cada línea define un "evento de hardkey". Los eventos de hardkey se refieren a las diferentes pulsaciones de una determinada hardkey. La segunda y la tercera pulsación de una determinada hardkey pueden tener reacciones diferentes, por ejemplo.

Las entradas del fichero de configuración "systemconfiguration.ini" pueden complementarse con ajustes específicos del usuario. Para ello están disponibles los directorios [Directorio de usuario del sistema]/cfg y [Directorio OEM del sistema]/cfg.

Las líneas para la configuración de los eventos de hardkey tienen la siguiente estructura:

```
KEYx.n = area:=area, dialog:=dialog, screen:=screen, forms:=form,
menus:=menu,
action:=menu.action, cmdline:=cmdline
KEYx.n = area:=area, dialog:=dialog, cmdline:=cmdline, action:=
action
```

x: Número de la hardkey, rango de valores: 1 – 254

n: número de evento (se corresponde con la enésima pulsación de la tecla), rango de valores: 0 – 9

Requisitos

El programa de usuario del PLC debe cumplir el siguiente requisito:

Siempre se ejecuta únicamente una hardkey. Por esta razón, solo debe establecerse una nueva solicitud cuando el software de manejo haya confirmado la solicitud anterior. Si el programa de usuario del PLC utiliza la hardkey desde una tecla de MCP, debe procurarse que la tecla quede guardada en una memoria intermedia de forma que tampoco se pierda ninguna pulsación en caso de manejo rápido.

Interfaz de PLC

En la interfaz del PLC se prevé un campo para la selección de una tecla. El campo se encuentra en DB19.DBB10. Aquí, el PLC puede especificar directamente un valor de tecla entre 50 y 254.

El acuse por parte del software de manejo se efectúa en dos pasos. Este procedimiento es necesario para que el software de manejo pueda reconocer el mismo código de tecla dos veces seguidas de forma correcta como dos eventos independientes. En el primer paso se escribe la información de control 255 en el byte DB19.DBB10. Esta pulsación de teclas definida virtualmente permite reconocer de forma unívoca cualquier secuencia de teclas del PLC. La información de control no tiene ningún significado para el programa de usuario del PLC y no se debe modificar. En el segundo paso, se efectúa el acuse de recibo propiamente dicho frente al PLC con el borrado de DB19.DBB10 A partir de ese momento, el programa de usuario del PLC puede especificar una nueva hardkey. Al mismo tiempo se procesa la solicitud de la hardkey actual en el software de manejo.

Ejemplo

Fichero de configuración:

```
; Hardkeys del PLC (KEY50-KEY254)
[keyconfiguration]
KEY50.0 = name := AreaMachine, dialog := SlMachine
KEY51.0 = name := AreaParameter, dialog := SlParameter
```

```
KEY52.0 = name := AreaProgramEdit, dialog := SlProgramEdit
KEY53.0 = name := AreaProgramManager, dialog := SlPmDialog
KEY54.0 = name := AreaDiagnosis, dialog := SlDgDialog
KEY55.0 = name := AreaStartup, dialog := SlSuDialog
KEY56.0 = name := Custom, dialog := SlEsCustomDialog
```

El identificador del área y del cuadro de diálogo se indican en "systemconfiguration.ini" en [Directorio Siemens del sistema]/cfg.

11.3 Selección de diálogos a través del CN

Comando MMC en HMI Operate

Los comandos MMC pueden utilizarse como se describe a continuación:

1. **Definición de comandos MMC**
   En el fichero estándar "systemconfiguration.ini" se encuentran las siguientes combinaciones:
   address:=MCYCLES --> command:=LM
   address:=CYCLES --> command:=PICTURE_ON
   Esta diferenciación es necesaria para distinguir entre ciclos de medida y otros ciclos. Es decir:
   - LM se aplica siempre a ciclos de medida y PICTURE_ON a otros ciclos.
   - Los comandos MMC de nueva definición no deben denominarse PICTURE_ON ni LM.
2. **Licencia de "Run MyScreens"**
   Todos los diálogos abiertos por "Run MyScreens" (a excepción de los ciclos de medida) están sujetos a la licencia de "Run MyScreens", por lo que, sin licencia, solo pueden utilizarse en número limitado.
   Llamada de ejemplo con "test.com" (Run MyScreens):

```
g0 f50
MMC("CYCLES,PICTURE_ON,test.com,máscara1","A")
m0
MMC("CYCLES,PICTURE_OFF","N")
M30
```

Ejemplos de máscaras de ciclo 12

12.1 Ejemplos de máscaras de ciclo

Al instalar SINUMERIK Operate se guardan también algunos ejemplos de máscaras de ciclo creadas con Run MyScreens.

Encontrará los ejemplos de Run MyScreens en las siguientes rutas de almacenamiento:

[Directorio Siemens del sistema]		
...\siemens\sinumerik\hmi\template\easyscreen\		
...		
	standard\	Pantallas de ayuda, PNG
	x3d\	Pantallas de ayuda, animaciones
...		
	Milling\	Fresado
	Turning\	Torneado
...		
	deu\	Descripción en alemán
	eng\	Descripción en inglés

En la interfaz HMI encontrará ejemplos en el campo de manejo de puesta en marcha:

Pulsador de menú horizontal Datos de sistema → Datos HMI → Plantillas → Ejemplos → Easyscreen → ... (ver arriba).

Descripción abreviada de los ejemplos

- Fresado
  - Ejemplo de taladrado: aquí es posible añadir una posición.
  - Ejemplo de medir pieza: con el cuadro de diálogo abierto, aquí es posible generar y ejecutar una llamada de ciclo mediante arranque del CN.
  - Ejemplo de contador de piezas: en este ejemplo se muestra el manejo con GUD
- Torneado
  - Dispositivos: estos incluyen un contrapunto, un cargador de barras, un receptor de piezas
  - Ejemplo de medir pieza: con el cuadro de diálogo abierto, aquí es posible generar y ejecutar una llamada de ciclo mediante arranque del CN.
  - Ejemplo de taladrado: aquí es posible añadir una posición.

Configuración en Sidescreen

13

Dispone de las siguientes posibilidades para visualizar configuraciones de Run MyScreens en Sidescreen:

1. Visualización de una configuración de Run MyScreens en una página (aparte) de Sidescreen.
   En este caso, la configuración de Run MyScreens ocupa toda la página de Sidescreen.
2. Visualización de varias configuraciones de Run MyScreens en una página (aparte) de Sidescreen.
   Dentro de la página de Sidescreen se muestran una sobre otra cada una de las configuraciones de Run MyScreens formando los denominados elementos Sidescreen. Es posible desplazarse de forma vertical por los elementos Sidescreen y, por tanto, también por las configuraciones de Run MyScreens.
3. Agregar una o varias configuraciones de Run MyScreens a una página de Sidescreen existente.
   La configuración de Run MyScreens se agrega a una página de Sidescreen incluida en el volumen de suministro de SIEMENS. Este caso coincide con el punto 2 hasta la página de Sidescreen afectada.
4. Agregar una o varias configuraciones de Run MyScreens a un elemento Sidescreen.
   Las distintas configuraciones de Run MyScreens se muestran una junto a otra, formando los llamados widgets Sidescreen, y es posible desplazarse por ellas de forma horizontal.

13.1 Visualización de una configuración de Run MyScreens en una página de Sidescreen

Para visualizar una configuración de Run MyScreens en una página (aparte) de Sidescreen, debe agregarse una nueva página de Sidescreen a la configuración de Sidescreen existente ("slsidescreen.ini"). A continuación, en esta página de Sidescreen se mostrará/ejecutará la configuración de Run MyScreens.

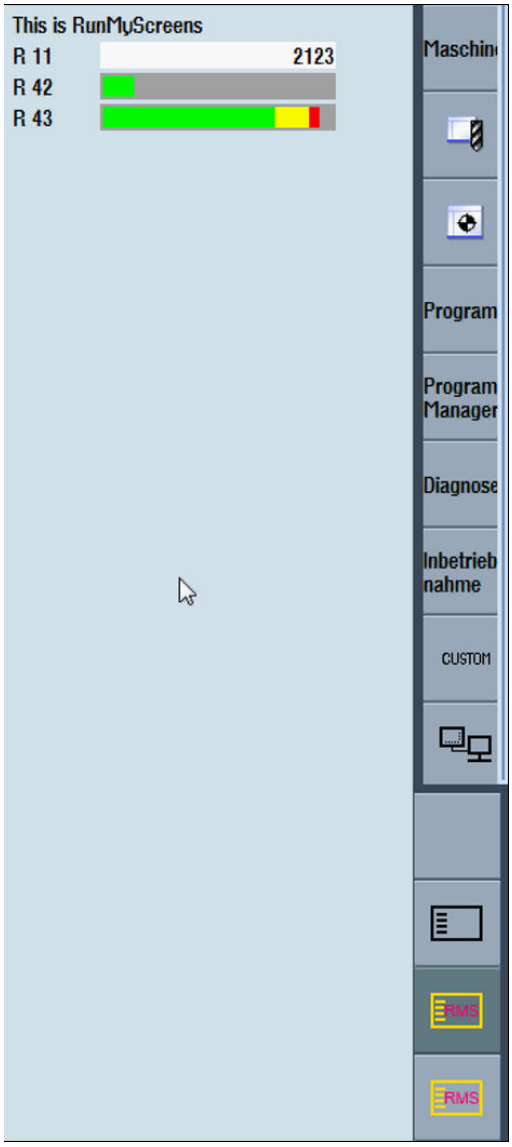


Figura 13-1 **Visualización de una configuración de Run MyScreens en una página de Sidescreen**

Procedimiento

Amplíe "slsidescreen.ini" con la siguiente entrada:

```
[Sidescreen]
PAGE100= name:=RMSPage,
implementation:=sleseasyscreen.SlEsSideScreenPage
```

- El número de página (en este caso, "100") debe incrementarse en función de las páginas de Sidescreen existentes. El número de la página debe ser mayor o igual que 100. De este modo se evitan conflictos con las páginas de SIEMENS.
- El nombre de la página (en este caso, "RMSPage") se puede elegir libremente.
- La indicación para la implementación de la página (en este caso, "sleseasyscreen.SlEsSideScreenPage") no puede modificarse.

En el siguiente paso, ajuste la configuración de Run MyScreens que va a ejecutarse. Para ello, cree una nueva sección cuyo nombre contenga el nombre de la página recién creada:

```
[Page_RMSPage]
Icon=rmspage.png
PROPERTY001= name:=maskPath, type:=QString,
value:="sidescreenmask.com"
PROPERTY002= name:=maskName, type:=QString, value:="Mask"
PROPERTY003= name:=focusable, type:=bool, value:="true"
```

- El nombre de la sección se forma añadiendo el nombre de la página (en este caso, "RMSPage") tras el prefijo "Page_".
- En Icon, introduzca el nombre de fichero (en este caso, "rmspage.png"). El fichero contiene el icono del botón de selección de página. Guarde el fichero en el directorio /user/sinumerik/hmi/ico/ico640 o /oem/sinumerik/hmi/ico/ico640.
- En la propiedad maskPath, indique el nombre del fichero que contiene la configuración de Run MyScreens (en este caso, "sidescreenmask.com").
- En la propiedad maskName, indique el nombre de la pantalla Run MyScreens que debe mostrarse (en este caso, "Mask").
- En la propiedad focusable, defina el foco de entrada. La página solo puede obtener el foco de entrada si este atributo tiene el valor "true". Este atributo se necesita para páginas que tengan elementos de entrada.

Guarde "slsidescreen.ini" en el directorio /oem/sinumerik/hmi/cfg o /user/sinumerik/hmi/cfg.

La configuración de Run MyScreens estará disponible tras el próximo arranque del HMI.

Ejemplo

Ejemplo de configuración completa en "slsidescreen.ini":

```
[Sidescreen]
PAGE005= name:=RMSPage,
implementation:=sleseasyscreen.SlEsSideScreenPage
[Page_RMSPage]
PROPERTY001= name:=maskPath, type:=QString,
value:="sidescreenmask.com"
PROPERTY002= name:=maskName, type:=QString, value:="Mask"
```

13.2 Visualización de varias configuraciones de Run MyScreens en una página de Sidescreen

Para visualizar varias configuraciones de Run MyScreens en una página (aparte) de Sidescreen, deben configurarse los llamados elementos Sidescreen, además de la página de Sidescreen. Los elementos Sidescreen sirven para visualizar las configuraciones de Run MyScreens.

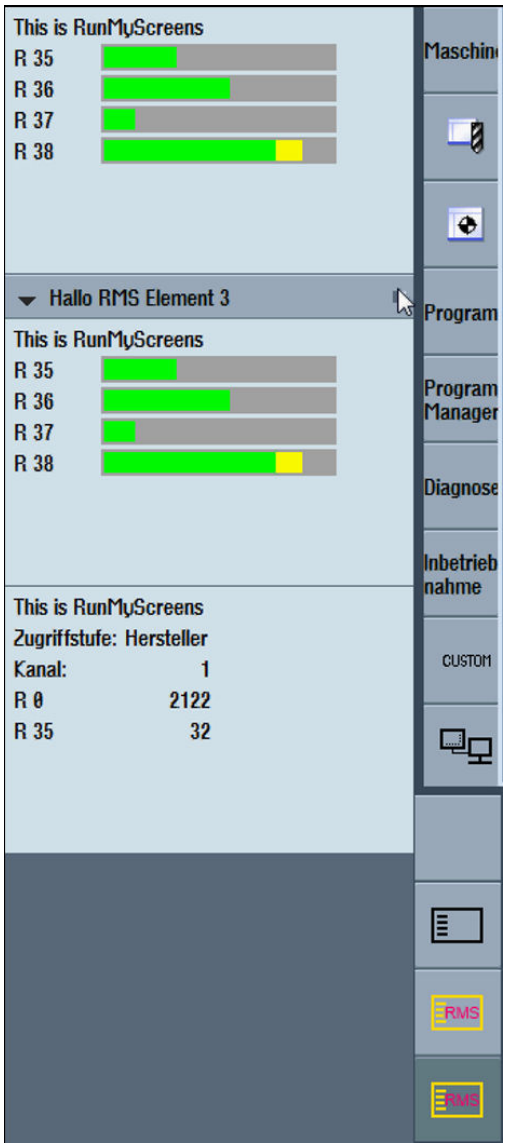


Figura 13-2 **Visualización de varias configuraciones de Run MyScreens en una página de Sidescreen**

Procedimiento

Amplíe "slsidescreen.ini" con las siguientes entradas:

```
[Sidescreen]
PAGE100= name:=RMSPage, implementation:=SlSideScreenPage
```

- El número de página (en este caso, "100") debe incrementarse en función de las páginas de Sidescreen existentes. El número de la página debe ser mayor o igual que 100. De este modo se evitan conflictos con las páginas de SIEMENS.
- El nombre de la página (en este caso, "RMSPage") se puede elegir libremente.
- La indicación para la implementación de la página (en este caso, "SlSideScreenPage") no puede modificarse.

```
[Page_RMSPage]
ELEMENT001= name:=RMSElement001,
implementation:=sleseasyscreen.SlEsSideScreenElement
ELEMENT002= name:=RMSElement002,
implementation:=sleseasyscreen.SlEsSideScreenElement
```

- Hay dos elementos Sidescreen asignados a la página (RMSElement001 y RMSElement002). Los nombres de los elementos pueden elegirse libremente.
- La indicación para la implementación de los elementos (en este caso, "sleseasyscreen.SlEsSideScreenElement") no puede modificarse.

Por último, también deben asignarse a los elementos Sidescreen las configuraciones de Run MyScreens que deben mostrarse en esos elementos:

```
[Element_RMSElement001]
PROPERTY001= name:=maskPath, type:=QString,
value:="sidescreenmask.com"
PROPERTY002= name:=maskName, type:=QString, value:="Mask001"
…
[Element_RMSElement002]
PROPERTY001= name:=maskPath, type:=QString,
value:="sidescreenmask.com"
PROPERTY002= name:=maskName, type:=QString, value:="Mask002"
```

Guarde "slsidescreen.ini" en el directorio /oem/sinumerik/hmi/cfg o /user/sinumerik/hmi/cfg.

La configuración de Run MyScreens estará disponible tras el próximo arranque del HMI.

13.3 Agregar configuraciones de Run MyScreens a una página de Sidescreen

El procedimiento para integrar una o varias configuraciones de Run MyScreens en una página de Sidescreen existente es idéntico al procedimiento descrito en el capítulo Visualización de varias configuraciones de Run MyScreens en una página de Sidescreen (Página 296). La única diferencia es que los elementos Sidescreen empleados para la visualización de configuraciones de Run MyScreens se agregan a una página de Sidescreen ya existente.

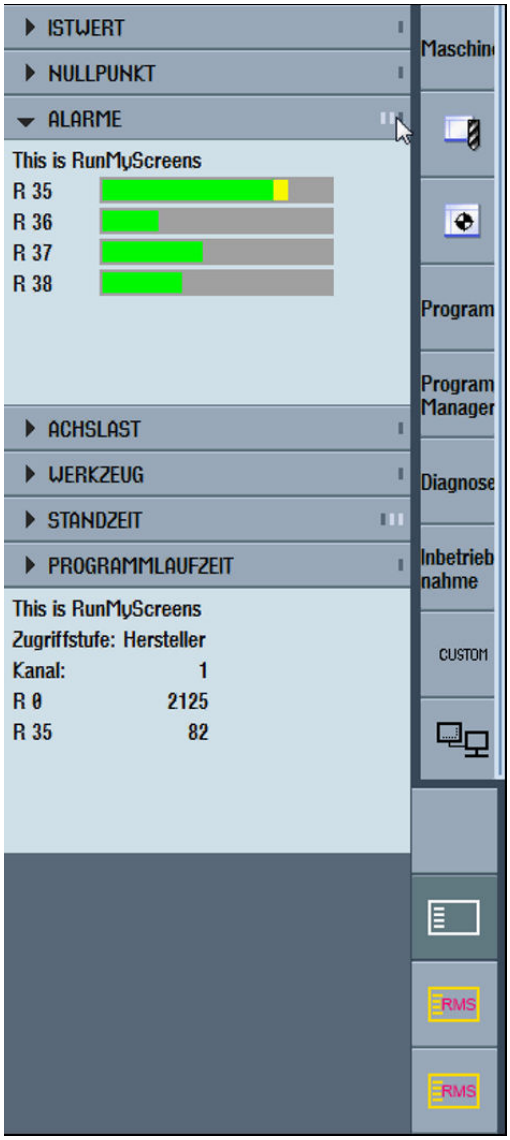


Figura 13-3 **Agregar configuraciones de Run MyScreens a una página de Sidescreen**

Nota

De las páginas de Sidescreen incluidas en el volumen de suministro de SIEMENS, solo puede ampliar WidgetsPage (ver fichero "slsidescreen.ini") con configuraciones de Run MyScreens.

Procedimiento

Amplíe "slsidescreen.ini" con las siguientes entradas:

En la sección [Page_WidgetsPage], agregue un elemento que corresponda al registro de la configuración de Run MyScreen.

```
[Page_WidgetsPage]
ELEMENT100= name:=RMSElement,
implementation:=sleseasyscreen.SlEsSideScreenElement
```

- Para nuevos elementos, utilice números a partir del 100, inclusive. De este modo se evitan conflictos con los elementos de visualización de SIEMENS. RMSElement se inserta en WidgetsPage de acuerdo con esta numeración por debajo de los elementos SIEMENS.

```
[Element_RMSElement]
PROPERTY001= name:=maskPath, type:=QString,
value:="sidescreenmask.com"
PROPERTY002= name:=maskName, type:=QString, value:="Mask001"
```

13.4 Agregar configuraciones de Run MyScreens a un elemento Sidescreen

En el procedimiento descrito a continuación a modo de ejemplo hay una página de Sidescreen con un elemento Sidescreen. En el elemento Sidescreen, se muestran dos configuraciones de Run MyScreens, una junto a la otra.

Configure una página de Sidescreen con un elemento Sidescreen tal como se describe a continuación. Asigne dos widgets Sidescreen con sus respectivas configuraciones de Run MyScreens al elemento Sidescreen.

Procedimiento

Amplíe "slsidescreen.ini" con las siguientes entradas:

```
[Sidescreen]
PAGE100= name:=RMSPage, implementation:=SlSideScreenPage
```

- La indicación de la implementación (en este caso, "SlSideScreenPage") no puede modificarse.

```
[Page_RMSPage]
ELEMENT001= name:=RMSElement, implementation:= SlSideScreenElement
```

- La indicación de la implementación (en este caso, "SlSideScreenElement") no puede modificarse.

```
[Element_RMSElement]
TextId=RMS_ELEMENT_TITLE
TextFile=rmstexts
```

```
TextContext=rmstexts
TextContext=rmstexts
WIDGET001= name:=RMSWidget001,
implementation:=sleseasyscreen.SlEsSideScreenWidget
WIDGET002= name:=RMSWidget002,
implementation:=sleseasyscreen.SlEsSideScreenWidget
```

- Si se utiliza la implementación SlSideScreenElement para el elemento Sidescreen (ver sección [Page_RMSPage]), el elemento Sidescreen tiene una línea de título en la que puede visualizarse un texto.
  Este texto se configura con las entradas TextId, TextFile y TextContext:
  - TextId: ID del texto que debe mostrarse
  - TextFile: fichero en el que está guardado el texto
  - TextContext: contexto en el que se encuentra el texto dentro de este fichero

  Para cada idioma debe proporcionarse un fichero por separado.
  El nombre de estos ficheros se forma según este esquema:
  <fichero de texto>_<código de idioma>.ts, p. ej., rmstexts_deu.ts para el ejemplo de más arriba.
  Si no se ha indicado ningún fichero ni contexto (TextFile=<empty> y TextContext=<empty>), la ID de texto se mostrará como texto. Se puede indicar un string cualquiera como ID de texto.
  El fichero de texto (p. ej., rmstexts_deu.ts) presenta la siguiente estructura:

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE TS>
<TS>
 <context>
 <name>rmstexts</name>
 <message>
  <source>Meine Applikation</source>
 <translation>My Application</translation>
 </message>
 </context>
</TS>
```

  Guarde el fichero en el directorio /oem/sinumerik/hmi/lng o /user/sinumerik/hmi/lng. Cuando el HMI vuelva a arrancar, el fichero se convertirá al formato de fichero requerido durante la ejecución.
- Los nombres de los widgets asignados al elemento (en este caso, "RMSWidget001" y "RMSWidget002") pueden elegirse libremente.
- La implementación de Widget "sleseasysscreen.SlEsSideScreenWidget" no puede modificarse.

```
[Widget_RMSWidget001]
PROPERTY001= name:=maskPath, type:=QString, value:="mymask001.com"
PROPERTY002= name:=maskName, type:=QString, value:="MyMask001"

[Widget_RMSWidget002]
PROPERTY001= name:=maskPath, type:=QString, value:="mymask002.com"
PROPERTY002= name:=maskName, type:=QString, value:="MyMask002"
```

- Los nombres de las secciones de configuración de los widgets Sidescreen se forman añadiendo el nombre del widget que debe configurarse en esta sección tras el prefijo "Widget_".
- En la propiedad maskPath, indique el nombre del fichero que contiene la configuración de Run MyScreens (en este caso, "mymask001.com" o "mymask002.com").
- En la propiedad maskName, indique el nombre de la pantalla Run MyScreens que debe mostrarse (en este caso, "MyMask001" o "MyMask002").

13.5 Indicaciones sobre la ejecución de configuraciones de Run MyScreens en Sidescreen

Tenga en cuenta las siguientes indicaciones:

- Si se ejecutan las configuraciones de Run MyScreens en Sidescreen, las funciones LS, LM, DLGL, EXIT y EXITLS no están disponibles.
- No es posible consultar el tamaño de la pantalla en el LOAD Block.
- Los textos se muestran siempre con la fuente empleada en SINUMERIK Operate para resoluciones de pantalla de 640 × 480 píxeles.
- Las posiciones configuradas de los elementos de manejo se implementan sin modificaciones (p. ej., expansión).
- El tamaño del cuadro de diálogo se ajusta automáticamente.
- Si se ejecutan las configuraciones de Run MyScreens en una página de Sidescreen, se dispone de toda la superficie de la página para la visualización. Si la ejecución se produce en los widgets o elementos Sidescreen, el sistema especifica en contexto la superficie disponible para la visualización.
- La cabecera y los pulsadores de menú no son configurables.
- Todas las salidas de depuración y de fallo se escriben de forma secuencial en el fichero de texto "easyscreen_log.txt". No existe ninguna identificación específica que indique de qué configuración proviene un aviso.
- Al ocultar una configuración de Run MyScreens mostrada en Sidescreen, se ignora el estado de esta.

Nota

La integración de configuraciones de Run MyScreens en Sidescreen puede provocar que se ejecute simultáneamente más de una configuración de Run MyScreens. Para mantener la capacidad operativa de todo el sistema tenga en cuenta, en función del hardware empleado, la carga adicional del sistema que esto supone.

Configuración en el Display Manager 14

14.1 Visualización de configuraciones de Run MyScreens en el Display Manager

Dispone del cuadro de diálogo SlSideScreenDialog para visualizar configuraciones de Run MyScreens en el Display Manager: En este cuadro de diálogo pueden mostrarse una o varias páginas de Sidescreen, en función del espacio disponible (ver IM9, capítulo 3.13). Se muestran varias páginas en forma de columna una junto a otra. En este caso, el espacio disponible (ancho) se distribuye de manera uniforme entre todas las columnas. Se configura el número de páginas que deben mostrarse y su contenido. Cada página tiene su propio fichero de configuración. En el caso de la configuración de diálogo, los nombres de estos ficheros de configuración se indican en el fichero "systemconfiguration.ini", en el parámetro de línea de comandos "cmdline". Tras el identificador de parámetro "–sidescreen1" se encuentra el nombre del fichero de configuración de la primera columna; tras el identificador de parámetro "–sidescreen2", el nombre del fichero de configuración de la segunda columna; y así sucesivamente.

Ejemplo

```
[dialogs]
DLG500= name:=RMSApp,
implementation:=sldmsidescreenapp.SlSideScreenDialog,
process:=SlHmiHost1, preload:=false, cmdline:="-sidescreen1
rmspage1.ini –sidescreen2 rmspage2.ini –spacing 3”
```

La instancia del diálogo SlSideScreenDialog creada en este ejemplo recibe el nombre de RMSApp. Esta tiene dos páginas/columnas. Ambas páginas/columnas se configuran en los ficheros "rmspage1.ini" y "rmspage2.ini". Entre ambas columnas hay una distancia de tres píxeles.

Nota

Los nombres de los ficheros con configuraciones de página (en este caso, "rmspage1.ini" y "rmspage2.ini") pueden elegirse libremente.

En el capítulo Configuración en Sidescreen (Página 293) se describe la utilización de configuraciones de Run MyScreens en el Display Manager. La única diferencia consiste en que la integración de las configuraciones no se realiza en el fichero "slsidescreen.ini", sino en los respectivos ficheros previstos para la configuración de las páginas existentes.

Existen dos posibilidades para integrar configuraciones de Run MyScreens en el Display Manager de SINUMERIK Operate. Estas se describen en los siguientes apartados.

14.2 Integración en SISideScreenDialog

Para integrar la configuración de Run MyScreens, debe crearse la instancia de diálogo de SlSideScreenDialog en el fichero "systemconfiguration.ini" y crear el fichero para integrar la configuración de Run MyScreens.

El siguiente ejemplo muestra cómo se integra una configuración de Run MyScreens. La configuración de Run MyScreens ocupa toda la ventana del diálogo SlSideScreenDialog, es decir, solo existe una página/columna.

Procedimiento

En primer lugar, cree una instancia del diálogo SlSideScreenDialog en el fichero "systemconfiguration.ini", en la sección `[dialogs]`. La instancia recibe el nombre "RMSApp".

```
[dialogs]
DLG500= name:=RMSApp,
implementation:=sldmsidescreenapp.SlSideScreenDialog,
process:=SlHmiHost1, preload:=false,
cmdline:="-sidescreen1 rmsapp.ini"
```

En el siguiente paso se ajusta/integra la configuración de Run MyScreens en el fichero "rmsapp.ini":

```
[Sidescreen]
PAGE001= name:=RMSPage, implementation:=SlSideScreenPage
```

- Cree una página con el nombre "RMSPage".

```
[Page_RMSPage]
ELEMENT001= name:=RMSElement,
implementation:=sleseasyscreen.SlEsSideScreenElement
```

- En la página RMSPage, cree un elemento con el nombre RMSElement. Este elemento sirve para visualizar la configuración de Run MyScreens.

```
[Element_RMSElement]
PROPERTY001= name:=maskPath, type:=QString,
value:="sidescreenmask.com"
PROPERTY002= name:=maskName, type:=QString, value:="Mask"
```

- En la propiedad maskPath, indique el nombre del fichero que contiene la configuración de Run MyScreens (en este caso, "sidescreenmask.com").
- En la propiedad maskName, indique el nombre de la pantalla Run MyScreens que debe mostrarse (en este caso, "Mask").

Guarde los ficheros "rmsapp.ini" y "systemconfiguration.ini" (ver arriba) en el directorio /oem/sinumerik/hmi/cfg o /user/sinumerik/hmi/cfg.

La configuración de Run MyScreens estará disponible tras el próximo arranque del HMI.

14.3 Integración en SIWidgetsApp

A continuación se describe la integración de una configuración de Run MyScreens en la aplicación SlWidgetsApp, incluida en el volumen de suministro de SIEMENS.

En función de la página/columna de la aplicación SlWidgetsApp en la que deba integrarse la configuración de Run MyScreens, deberá ampliar el fichero "sldmwidgets1.ini" o bien el fichero "sldmwidgets2.ini".

El siguiente ejemplo muestra cómo se integra una configuración de Run MyScreens en la página/columna izquierda de la aplicación SlWidgetsApp.

Procedimiento

Amplíe el fichero "sldmwidgets1.ini" del siguiente modo:

```
[Sidescreen]
PAGE001= name:=WidgetsPage, implementation:=SlSideScreenPage
[Page_WidgetsPage]
ELEMENT100= name:=RMSElement,
implementation:=sleseasyscreen.SlEsSideScreenElement
```

- Cree un elemento con el nombre "RMSElement" para la visualización de la configuración de Run MyScreens.

Nota

Para nuevos elementos, utilice números a partir del 100, inclusive. De este modo se evitan conflictos con los elementos de visualización de SIEMENS. RMSElement se inserta en WidgetsPage de acuerdo con esta numeración por debajo de los elementos SIEMENS.

```
[Element_RMSElement]
PROPERTY001= name:=maskPath, type:=QString,
value:="sidescreenmask.com"
PROPERTY002= name:=maskName, type:=QString, value:="Mask"
```

- En la propiedad maskPath, indique el nombre del fichero que contiene la configuración de Run MyScreens (en este caso, "sidescreenmask.com").
- En la propiedad maskName, indique el nombre de la pantalla Run MyScreens que debe mostrarse (en este caso, "Mask").

Guarde el fichero "sldmwidgets1.ini" en el directorio /oem/sinumerik/hmi/cfg o /user/sinumerik/hmi/cfg.

La configuración de Run MyScreens estará disponible tras el próximo arranque del HMI.

14.4 Indicaciones sobre la ejecución de configuraciones de Run MyScreens en el Display Manager

Tenga en cuenta las siguientes indicaciones:

- Si se ejecutan las configuraciones de Run MyScreens en el Display Manager, las funciones LS, LM, DLGL, EXIT y EXITLS no están disponibles.
- No es posible consultar el tamaño de la pantalla en el LOAD Block.
- Los textos se muestran siempre con la fuente empleada en SINUMERIK Operate para resoluciones de pantalla de 640 × 480 píxeles.
- Las posiciones configuradas de los elementos de manejo se implementan sin modificaciones (p. ej., expansión).

- Si se ejecutan las configuraciones de Run MyScreens en una página de Sidescreen, se dispone de toda la superficie de la página para la visualización. Si la ejecución se produce en los widgets o elementos Sidescreen, el sistema especifica en contexto la superficie disponible para la visualización.
- Todas las salidas de depuración y de fallo se escriben de forma secuencial en el fichero de texto easyscreen_log.txt. No existe ninguna identificación específica que indique de qué configuración proviene un aviso.
- Al ocultar una configuración de Run MyScreens mostrada en el Display Manager, se ignora el estado de esta.

Nota

La integración de configuraciones de Run MyScreens en el Display Manager puede provocar que se ejecute simultáneamente más de una configuración de Run MyScreens. Para mantener la capacidad operativa de todo el sistema tenga en cuenta, en función del hardware empleado, la carga adicional del sistema que esto supone.

Listas de referencia

A

A.1 Listas de los pulsadores de menú de inicio

A.1.1 Lista de los pulsadores de menú de inicio para torneado

Campo de manejo Programa para torneado

HSK1	HSK2	HSK3	HSK4	HSK5	HSK6	HSK7	HSK8
Editar	**Taladrar**	**Tornear**	**Tornear contorno**	**Fresar**	**Otros**	**Simulación**	**CN Selección**
HSK9	HSK10	HSK11	HSK12	HSK13	HSK14	HSK15	HSK16
--	Recta Arco (solo con ShopTurn)	--	--	Medir pieza	Medir herramienta	**OEM**	--

Torneado

En las tablas siguientes se muestran los pulsadores de menú de inicio posibles para la tecnología de torneado. Las asignaciones de determinados pulsadores de menú de inicio pueden variar debido a las características del sistema. Los pulsadores de menú OEM indicados se admiten para "Run MyScreens".

Pulsadores de menú de inicio programGUIDE (código G):

	Taladrar	**Tornear**	**Tornear contorno**		**Fresar**		**Otros**	
	HSK2	**HSK3**	**HSK4**		**HSK5**		**HSK6**	
VSK1	Puntear	Desbaste	Contorno	--	Planeado	Contorno	Ajustes	High Speed Settings
VSK2	Taladrar escariar	Entallado	Desbaste	--	Caja	Contorneado	Orientación plano	Ejes paralelos
VSK3	Taladrado profundo	Garganta	Desbastar resto	--	Saliente poliedro	Pretaladrar	Orientar herramienta	--
VSK4	Mandrinado	Rosca	Ranurado	--	Ranura	Caja	--	--
VSK5	Rosca	Tronzado	Ranurar resto	--	Fresado de roscas	Mat. rest. caja	--	--
VSK6	**OEM**	--	Ranurado derecha/izquierda	--	Grabado	Saliente	Subprograma	--

VSK7	Posiciones	**OEM**	Ranurar d/i resto	**OEM**	**OEM**	Mat. rest. saliente	--	**OEM**
VSK8	Repetir posición	--	>>	<<	Fresado del contorno	<<	>>	<<

Pulsadores de menú de inicio de ShopTurn:

	Taladrar	**Tornear**	**Tornear contorno**		**Fresar**		**Otros**		
	HSK2	**HSK3**	**HSK4**		**HSK5**		**HSK6**		**HSK10**
VSK1	Taladro:centrado	Desbaste	Nuevo contorno	--	Planeado	Nuevo contorno	Ajustes	High Speed Settings	Herramienta
VSK2	Puntear	Entallado	Desbaste	--	Caja	Contorneado	Orientación plano	Ejes paralelos	Recta
VSK3	Taladrar escariar	Garganta	Desbastar resto	--	Saliente poliedro	Pretaladrar	Orientar herramienta	Repetición programa	Centro círculo
VSK4	Taladrado profundo	Rosca	Ranurado	--	Ranura	Caja	Contracabezal	--	Arco con radio
VSK5	Rosca	Tronzado	Ranurar resto	--	Fresado de roscas	Mat. rest. caja	Transformaciones	--	Polar
VSK6	**OEM**	--	Ranurado derecha/izquierda	--	Grabado	Saliente	Subprograma	--	Retirar/aproximar
VSK7	Posiciones	**OEM**	Ranurar d/i resto	**OEM**	**OEM**	Mat. rest. saliente	--	**OEM**	--
VSK8	Repetir posición	--	>>	<<	Fresado del contorno	<<	>>	<<	--

A.1.2 Lista de los pulsadores de menú de inicio para fresado

Campo de manejo Programa para fresado

HSK1	HSK2	HSK3	HSK4	HSK5	HSK6	HSK7	HSK8
Editar	**Taladrar**	**Fresar**	**Fresar contorno**	**Tornear** (solo con código G)	**Otros**	**Simulación**	**CN Selección**
HSK9	HSK10	HSK11	HSK12	HSK13	HSK14	HSK15	HSK16
--	Recta Arco (solo con ShopMill)	--	--	Medir pieza	Medir herramienta	**OEM**	--

Fresado

En las tablas siguientes se muestran los pulsadores de menú de inicio posibles para la tecnología de fresado. Las asignaciones de determinados pulsadores de menú de inicio pueden variar debido a las características del sistema. Los pulsadores de menú OEM indicados se admiten para "Run MyScreens".

Pulsadores de menú de inicio programGUIDE (código G):

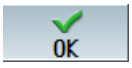
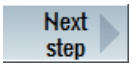
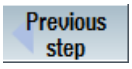
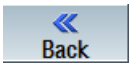
	Taladrar	**Fresar**	**Fresar contorno**		**Tornear**		**Otros**	
	HSK2	**HSK3**	**HSK4**		**HSK5**		**HSK6**	
VSK1	Puntear	Planeado	Contorno	--	Desbaste	Contorno	Ajustes	--
VSK2	Taladrar escariar	Caja	Contorneado	--	Entallado	Desbaste	Orientación plano	Ejes paralelos
VSK3	Taladrado profundo	Saliente poliedro	Pretaladrar	--	Garganta	Desbastar resto	Orientar herramienta	--
VSK4	Mandrinado	Ranura	Caja	--	Rosca	Ranurado	High Speed Settings	--
VSK5	Rosca	Fresado de roscas	Mat. rest. caja	--	Tronzado	Ranurar resto	--	--
VSK6	**OEM**	Grabado	Saliente	--	--	Ranurado derecha/ izquierda	Subprograma	--
VSK7	Posiciones	**OEM**	Mat. rest. saliente	**OEM**	**OEM**	Ranurar d/i resto	--	**OEM**
VSK8	Repetir posición	--	>>	<<	Torneado de contorno	<<	>>	<<

Pulsadores de menú de inicio de ShopMill:

	Taladrar	**Fresar**	**Fresar contorno**		**Otros**		**Recta arco**
	HSK2	**HSK3**	**HSK4**		**HSK6**		**HSK10**
VSK1	Puntear	Planeado	Nuevo contorno	--	Ajustes	--	Herramienta
VSK2	Taladrar escariar	Caja	Contorneado	--	Orientación plano	Ejes paralelos	Recta
VSK3	Taladrado profundo	Saliente poliedro	Pretaladrar	--	Orientar herramienta	Repetición programa	Centro círculo
VSK4	Mandrinado	Ranura	Caja	--	High Speed Settings	--	Arco con radio
VSK5	Rosca	Fresado de roscas	Mat. rest. caja	--	Transformaciones	--	Hélice
VSK6	**OEM**	Grabado	Saliente	--	Subprograma	--	Polar
VSK7	Posiciones	**OEM**	Mat. rest. saliente	**OEM**	--	**OEM**	--
VSK8	Repetir posición	--	>>	<<	>>	<<	--

A.2 Lista de los pulsadores de menú predefinidos

Tabla A-1 Pulsadores de menú predefinidos

Nombre	Pulsador de menú
SOFTKEY_OK	
SOFTKEY_CANCEL	
SOFTKEY_APPLY	
SOFTKEY_MORE	
SOFTKEY_BACK	
SOFTKEY_ASSISTANT_NEXT	
SOFTKEY_ASSISTANT_PREVIOUS	
SOFTKEY_NAV_BACK	

A.3 Lista de los niveles de acceso

Los diferentes niveles de acceso tienen el siguiente significado:

Nivel de protección	Bloqueado por	Área
ac0	Reservado para Siemens	
ac1	Contraseña	Fabricante de la máquina
ac2	Contraseña	Servicio técnico
ac3	Contraseña	Usuario
ac4	Interruptor de llave posición 3	Programador, preparador
ac5	Interruptor de llave posición 2	Operador cualificado
ac6	Interruptor de llave posición 1	Operador formado
ac7	Interruptor de llave posición 0	Operador entrenado

A.4 Lista de los colores

Colores del sistema

Para la configuración de diálogos se dispone de una tabla de colores unitaria (algunos de los colores estándar correspondientes). Para un elemento (texto, campo de entrada, fondo, etc.) se puede seleccionar uno de los siguientes colores, que pueden encontrarse entre 0 y 133.

Como alternativa a los colores predefinidos pueden especificarse también colores como valores RGB ("#RRGGBB").

Índice	Pictograma	Cambiar	Descripción del color
1		negro	
2		Naranja	
3		Verde oscuro	
4		Gris claro	
5		Gris oscuro	
6		Azul	
7		Rojo	
8		marrón	
9		Amarillo	
10		Blanco	
126		Negro	Color de texto de un campo de entrada/salida que tiene el foco actualmente
127		Naranja claro	Color de fondo de un campo de entrada/salida que tiene el foco actualmente
128		Naranja	Color de sistema: foco
129		Gris claro	Color de fondo
130		Azul	Color de encabezado (activo)
131		Negro	Color de la letra del encabezado (activo)

Índice	Pictograma	Cambiar	Descripción del color
132		Turquesa	Color de fondo de un campo de alternancia
133		Azul claro	Color de fondo del cuadro de lista

A.5 Lista de códigos de idioma en el nombre del fichero

Idiomas admitidos

Idiomas estándar:

Idioma	Abreviatura en el nombre del fichero
Chino simplificado	chs
Alemán	deu
Inglés	eng
Francés	fra
Italiano	ita
Español	esp

Otros idiomas:

Idioma	Abreviatura en el nombre del fichero
Chino tradicional	cht
Danés	dan
Finlandés	fin
Indonesio	ind
Japonés	jpn
Coreano	kor
Malayo	msl
Neerlandés	nld
Polaco	plk
Portugués (Brasil)	ptb
Rumano	rom
Ruso	rus
Sueco	sve
Eslovaco	sky
Esloveno	slv
Tailandés	tha
Checo	csy
Turco	trk

Idioma	Abreviatura en el nombre del fichero
Húngaro	hun
Vietnamita	vit

A.6 Lista de las variables de sistema accesibles

Bibliografía

Manual de listas Variables del sistema /PGAsl/

A.7 Comportamiento al abrir el cuadro de diálogo (atributo CB)

La siguiente tabla le da una vista general de las condiciones en las que se llama al método CHANGE.

Para el atributo CB se aplica:

CB0

El método CHANGE se inicia al mostrar la máscara si la variable tiene un valor válido en ese momento (p. ej. mediante valor predeterminado o variable de CN/PLC).

CB1 (predeterminado)

El método CHANGE no se inicia de forma explícita al mostrar la máscara. Si la variable tiene una variable de CN/PLC configurada, se llama de todos modos al método CHANGE, naturalmente.

Condición				Reacción
Tipo	**Variable de sistema o de usuario**	**Valor predeterminado**	**Ejecutar método CHANGE**	
Campo E/S	Sí	Sí	Sí	Con la variable de CN/PLC configurada se obtiene siempre al menos una llamada automática del método CHANGE con el valor actual de la variable CN/PLC. El valor predeterminado de CB no tiene efecto.
			No	
		No	Sí	
			No	
	No	Sí	Sí	Se llama al método CHANGE con el valor predeterminado.
			No	No se llama al método CHANGE.
		No	Sí	Sin llamada, ya que no existe ningún valor válido para llamar al método CHANGE.
			No	

Condición				**Reacción**
Tipo	**Variable de sistema o de usuario**	**Valor predeterminado**	**Ejecutar método CHANGE**	
Campo de alternancia	Sí	Sí	Sí	Con la variable de CN/PLC configurada se obtiene siempre al menos una llamada automática del método CHANGE con el valor actual de la variable CN/PLC. El valor predeterminado de CB no tiene efecto.
			No	
		No	Sí	
			No	
	No	Sí	Sí	Se llama al método CHANGE con el valor predeterminado.
			No	No se llama al método CHANGE.
		No (de forma predeterminada se predefine con el primer valor de la lista de alternancia)	Sí	Se llama al método CHANGE con el primer valor de la lista de alternancia como valor predeterminado.
			No	No se llama al método CHANGE.

Sugerencias y trucos

B

B.1 Sugerencias generales

- En la medida de lo posible, se utiliza la variante BTSS para variables de sistema (variables $).
  Motivo:
  Así se evita, en determinadas circunstancias, una laboriosa resolución del nombre.
  Ejemplo:
  En lugar de "$R[10]" se utiliza preferiblemente "/Channel/Parameter/R[u1,10]".
- Debe evitarse la definición cíclica (del mismo tipo), p. ej. de la propiedad HLP de máscaras y variables. Debe compararse el valor que debía definirse anteriormente con el valor actual y definirlos realmente si hay alguna divergencia.
  Motivo:
  Evitar la laboriosa búsqueda de pantallas de ayuda que dependen de la resolución.
  Ejemplo:

```
DEF MyVar=(R3///,"X1",,"mm"/WR1//"$AA_IM[0]")
CHANGE(MyVar)
   IF MyVar.VAL < 100
      HLP="mypic1.png"
   ELSE
      HLP="mypic2.png"
   ENDIF
END_CHANGE
```

Recomendación:

```
CHANGE(MyVar)
   IF MyVar.VAL < 100
      IF HLP <> "mypic1.png"
         HLP="mypic1.png"
      ENDIF
   ELSE
      IF HLP <> "smypic2.png"
         HLP="mypic2.png"
      ENDIF
   ENDIF
END_CHANGE
```

- Deben reemplazarse varias funciones RNP() sucesivas por una función MRNP()-.
  Deben reemplazarse varias funciones RDOP() sucesivas por una función MRDOP().
  Motivo:
  Reducción de la carga de comunicación y aumento del rendimiento.
- Los parámetros de accionamiento no deben leerse en ciclos de menos de 1 segundo, es mejor que se lean más despacio.
  Motivo: De lo contrario, la comunicación con los accionamientos podría perturbarse en exceso o incluso provocar fallos.
- Deben evitarse operaciones aritméticas sucesivas con variables de diálogo ligadas a variables de sistema o de usuario. Para ello, deben utilizarse, p. ej., registro (REG[x]) o variables auxiliares (invisibles).
  Motivo: Cada asignación de un valor da lugar también a una escritura en la variable de sistema o de usuario ligada.
- Por motivos de claridad, facilidad de mantenimiento y rendimiento (al mostrar las máscaras) el código del mismo tipo que se utilice en distintos bloques debe agruparse en un bloque SUB. Este puede abrirse entonces en las posiciones correspondientes con la función CALL().
- Mediante la observación de la carga de la CPU en la línea de diálogo (ajuste en slguiconfig.xml) puede analizarse cómo afectan al rendimiento las modificaciones en la máscara.

B.2 Consejos para la depuración

- La función DEBUG() debe utilizarse para el diagnóstico. Al llamar a la función DEBUG() se escribe en "easyscreen_log.txt" el string transferido. La salida de información en la línea de diálogo a través de la función DLGL() también puede ser útil.
  No obstante, al finalizar el desarrollo de las máscaras esta función debe volver a eliminarse o debe comentarse por motivos de rendimiento.
- El fichero de registro "easyscreen_log.txt" debe quedar siempre sin entradas una vez que haya acabado el desarrollo de una máscara.

B.3 Consejos para el método CHANGE

- Los métodos CHANGE deben mantenerse siempre muy breves, en especial aquellos con variables ligadas a una variable de sistema o de usuario que se modifica a alta frecuencia.
  Motivo:
  Aumento del rendimiento de la máscara.
- En la medida de lo posible, no deben configurarse funciones RNP() en métodos CHANGE. En lugar de ello, es preferible crear al mismo tiempo una variable invisible con la variable de sistema o de usuario que se va a leer y utilizar esta.
  Motivo:
  Con cada llamada se emitiría inevitablemente una función RNP(). Si no, se accedería simple y exclusivamente al valor actual ya existente de todos modos.

Ejemplo:

con cada cambio del movimiento del eje por RNP() se efectúa una resolución de nombres para leer un dato de máquina específico del canal:

```
DEF AXIS_POSITION_X =  (R///,""///"$AA_IM[X]")

CHANGE(AXIS_POSITION_X)
      DLGL("Axis """ << RNP("$MC_AXCONF_GEOAX_NAME_TAB[0]") << """
se ha movido: "
<< AXIS_POSITION_X)
END_CHANGE
```

Con ayuda de una variable invisible, se mantiene actual el dato de máquina específico del canal y cada cambio de valor se copia en una variable temporal, como el registro.

Esta variable temporal puede entonces utilizarse en el método CHANGE del cambio de valor de la posición de eje sin efectuar cada vez una resolución de nombres del dato de máquina y crear el posterior acceso de lectura:

```
DEF AXIS_POSITION_X =  (R///,""///"$AA_IM[X]")
DEF AXIS_NAME_X = (S///,""/WR0//"$MC_AXCONF_GEOAX_NAME_TAB[0]")

CHANGE(AXIS_NAME_X)
      REG[0] = AXIS_NAME_X
END_CHANGE

CHANGE(AXIS_POSITION_X1)
   DLGL("Axis """ << REG[0] << """ se ha movido " <<
AXIS_POSITION_X)
END_CHANGE
```

- La frecuencia de actualización —y, con ella, la ejecución del método CHANGE correspondiente de variables ligadas a variables de sistema o de usuario con cambio de valor de alta frecuencia— puede disminuirse utilizando la propiedad de variables UR, p. ej. la variable que está vinculada a los valores de los ejes.
  Motivo:
  De este modo, al cambiar el valor el método CHANGE correspondiente se ejecuta en una retícula fija predefinida.

B.4 Consejos sobre bucles DO-LOOP

Puesto que los bucles pueden afectar al rendimiento de SINUMERIK Operate en función de la configuración, deben aplicarse con prudencia y debe prescindirse, en la medida de lo posible, de acciones que requieran mucho tiempo.

Asimismo debe utilizarse, p. ej., un registro (REG[]) como variable de proceso, ya que las variables de visualización normales (en particular las ligadas a BTSS) pueden afectar al rendimiento del sistema a causa de la extremada frecuencia de actualización o de procesos de escritura.

Debe tenerse en cuenta que el número de líneas de script procesadas "de un golpe" está limitado actualmente a 10.000. Si se alcanza este número, el procesamiento de scripts se interrumpe con una entrada correspondiente en "easyscreen_log.txt".

Motivos:

- Se evitan bucles sin fin
- "Run MyScreens" debe seguir operable

Elementos animados

C

C.1 Introducción

Introducción

El manual describe el trabajo con el visor de X3D, el cual permite integrar escenas gráficas (elementos animados) con o sin movimiento –en lo sucesivo denominadas pantallas de ayuda– en la interfaz gráfica de SINUMERIK Operate a partir de la versión V4.7 SP1.

Existe la posibilidad de representar secuencias de movimiento y parámetros concretos en pantallas de ayuda contextual. De esta forma podrá mejorar el manejo de las aplicaciones y diseñar una interfaz de usuario más atractiva.

El proceso de realización de una idea hasta la creación de la pantalla de ayuda comprende los siguientes pasos:

- Creación de elementos gráficos y modelos 3D para las pantallas de ayuda que se mostrarán después en el visor de X3D (ver capítulo Modelado (Página 320)).
- Creación del fichero de descripción de escenas, en el que los datos modelo del fichero gráfico se asignan a las escenas y animaciones concretas que desean representarse (ver capítulo Estructura del fichero de descripción de escenas (Página 327)).
- Conversión de los ficheros X3D y XML en ficheros HMI (ver capítulo Conversión a fichero hmi (Página 331)).
- Integración C++ del visor de X3D en la propia aplicación (ver capítulo Ejemplo de implementación (Página 334)).
- Aplicación en Run MyScreens (ver capítulo Visualización en Run MyScreens (Página 334)).

Vista general de los formatos de fichero

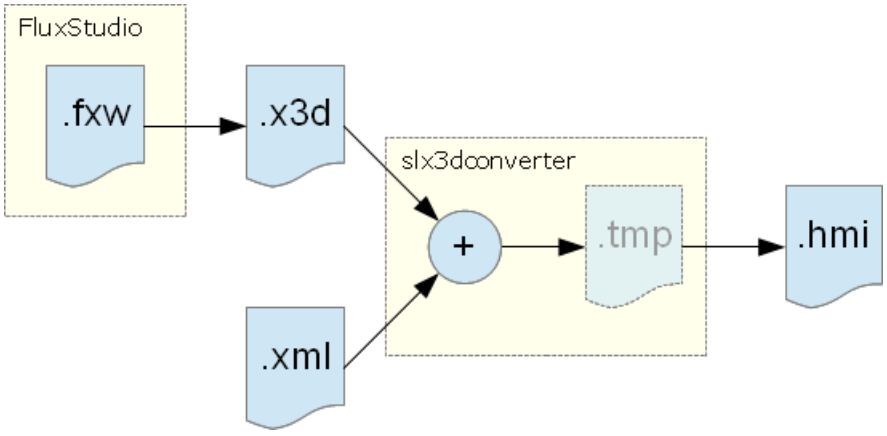


Figura C-1 Formatos de fichero

Formato de fichero	Descripción
.fxw	Fichero de gráficos de Vivaty Studio
.x3d	Fichero modelo en formato de intercambio
.xml	Fichero de definición para animaciones y escenas
.hmi	Fichero de salida para el visor de X3D

C.2 Modelado

C.2.1 Requisitos

El objetivo del modelado es crear un fichero con los modelos 3D generados en el formato .x3d, un estándar oficial para contenidos en 3D.

El modelado se realiza en **Vivaty Studio**. Vivaty Studio es una herramienta de modelado 3D de libre disposición con múltiples posibilidades de exportación e importación.

Requisitos

- Ha instalado Vivaty Studio, versión 1.0.
- Está familiarizado con el modelado y el manejo de Vivaty Studio.
- En esta descripción no se profundiza en el formato de fichero .x3d, por lo que se precisan los correspondientes conocimientos.

Nota

Puede descargar Vivaty Studio de Internet a través del enlace (https://www.web3d.org/projects/vivaty-studio).

C.2.2 Reglas para el modelado

Tenga en cuenta las siguientes reglas al realizar el modelado. Observarlas es necesario para poder preparar el fichero de la pantalla de ayuda a continuación.

Reglas para el modelado

1. El TimeSensor debe llamarse "Animación".
2. Todos los elementos que le pertenezcan deben asignarse a un grupo.
3. Todos los grupos deben ajustarse en la siguiente posición:
   x = 0, y = 0, z = 0

4. Para ocultar los grupos, los valores de transformación deben ajustarse del siguiente modo: x = 1000000, y = 1000000 y z = 1000000
5. Los siguientes elementos deben tener el mismo nombre en los modelos gráficos de Vivaty Studio que en las definiciones de escenas XML (a este respecto, ver capítulo Vista general (Página 327)).
   - Herramienta
   - Cámaras
6. Para crear un fichero .x3d debe realizar el siguiente ajuste en el diálogo "Export Options":

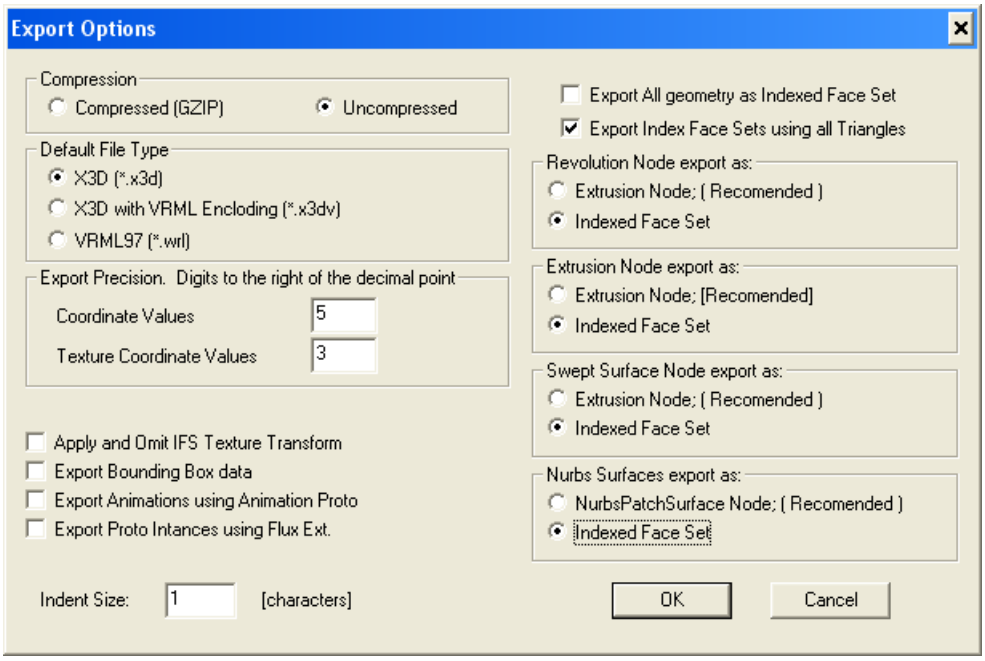


Figura C-2 Ajustes en el diálogo Export Options

7. Para textos se recomiendan los siguientes ajustes (ver también el diálogo a continuación):
   – Los textos siempre deben alinearse en el centro.
   – El tamaño del texto debe ser 0.2. De este modo resulta fácil posicionar correctamente el texto. La salida del texto con el visor de X3D no depende de este valor, sino que se muestra en el tamaño de fuente de la interfaz de usuario.

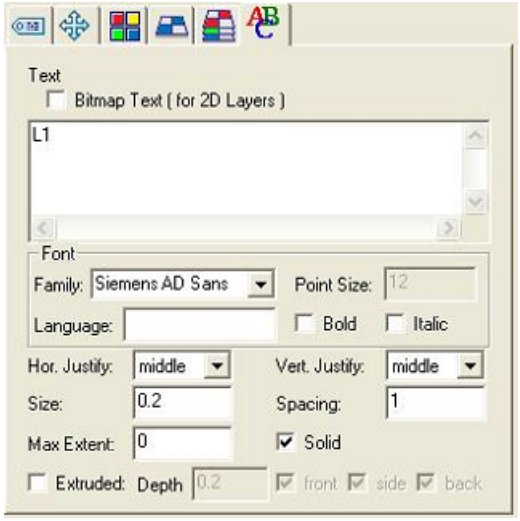


Figura C-3 Ajustes para el texto

8. Para crear un sombreado utilice el fichero schnitt.png como textura. Las líneas siempre van de abajo arriba y de izquierda a derecha en un ángulo de 45°. Se recomienda ajustar el escalado a 3 en ambas dimensiones. La rotación depende del tipo de diseño y debe adaptarse manualmente.

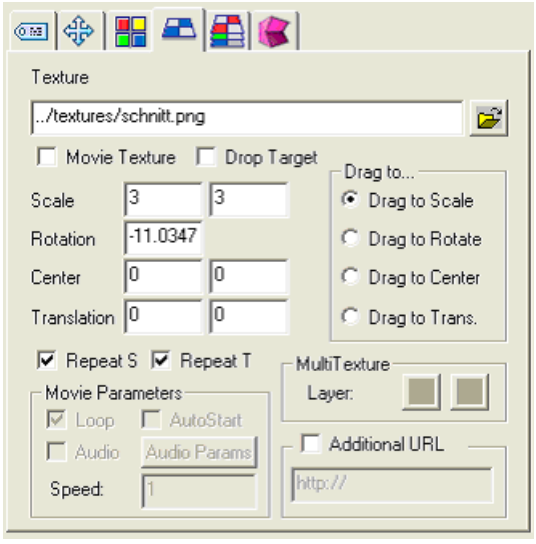


Figura C-4 Ajustes para el sombreado

9. Se recomiendan los siguientes tamaños/medidas para las piezas en bruto:
   - Cilindro para torneado:
     Longitud = 5,5; radio = 1,4 (ver plantilla turning_blank.fxw)
   - Paralelepípedo para fresado:
     x = 4,75; y = 3; z = 3 (ver plantilla milling_blank.fxw)

Consulte también

Comandos XML (Página 327)

C.2.3 Importación de gráficos (modelos)

En Flux Studio es posible importar gráficos (modelos) externos. Los modelos empleados a menudo, como las herramientas, pueden guardarse de forma centralizada como ficheros .x3d o .hmi y volver a utilizarse en otras pantallas de ayuda como elementos ya listos. Encontrará una lista con las plantillas de modelado suministradas en el capítulo Plantillas para el modelado (Página 325).

Los objetos más complejos también pueden crearse con otras herramientas de modelado, como Cinema4D, e importarse después.

A continuación se describe la forma de poder volver a utilizar estos modelos.

Importación como Inline

Para la importación de ficheros .x3d, Flux Studio ofrece el objeto "Inline". Con él pueden integrarse elementos externos sin tener que multiplicar los datos 3D de estos modelos.

El comando de menú **Create** -> **Create Inline** permite insertar el objeto. El nombre de fichero se indica a continuación en las propiedades del objeto.

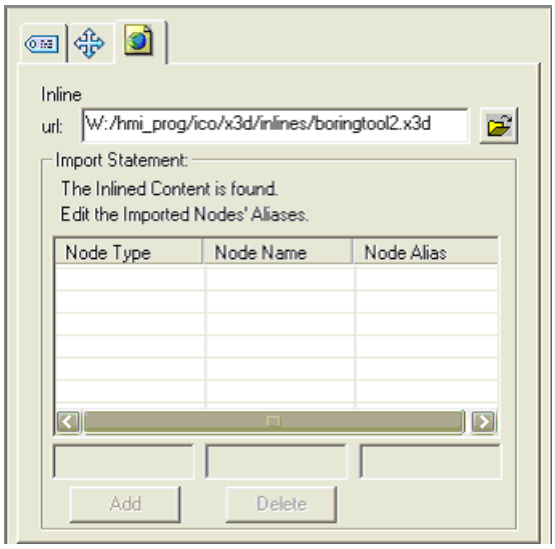


Figura C-5 Importación mediante el objeto "Inline"

Importación de datos modelo 3D

Para importar datos modelo desde un fichero, proceda del siguiente modo.

1. Abra el diálogo "Flux Studio Accutrans3D Translation Utility" a través del comando de menú **File** -> **Import Other Format**.
2. Seleccione el formato correspondiente en el campo de selección "File Types". Para importar, por ejemplo, un fichero de Cinema4D debe seleccionar el tipo de fichero "3D Studio".
3. A continuación, seleccione el fichero deseado con **Select Files** e inicie la importación.

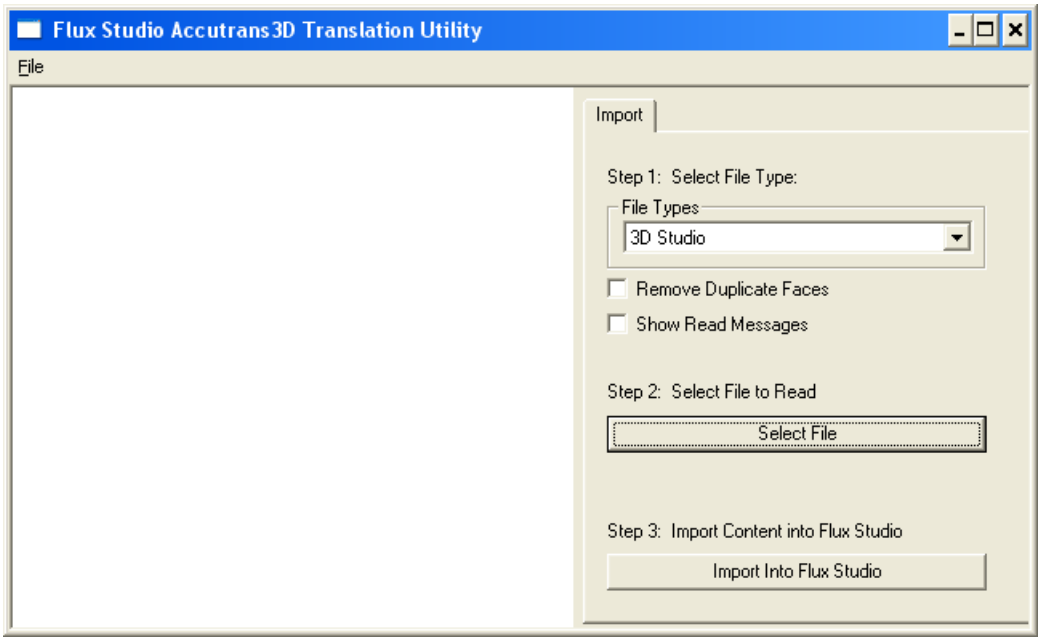


Figura C-6 Importación de datos modelo 3D

Los datos modelo del fichero de Cinema4D se insertan como grupo completo. Conviene borrar cualquier cámara u otro tipo de información que también deba importarse. Solo se precisan los elementos gráficos del fichero de Cinema4D.

C.2.4 Plantillas para el modelado

En este capítulo encontrará una lista de los elementos gráficos disponibles como base para los diseños, colores y tamaños. A fin de lograr un aspecto unitario de la interfaz de usuario, se recomienda utilizar el estilo de las plantillas o las propias plantillas a la hora de crear gráficos propios.

Generalidades

Plantilla	Descripción
intersection_texture.png	Textura para una superficie de corte
rapid_traverse_line_hori.fxw	Línea horizontal para ilustrar el recorrido de desplazamiento de la herramienta en rápido

Plantilla	Descripción
rapid_traverse_line_vert.fxw	Línea vertical para ilustrar el recorrido de desplazamiento de la herramienta en rápido
feed_traverse_line_hori.fxw	Línea horizontal para ilustrar la trayectoria de la herramienta en avance
feed_traverse_line_vert.fxw	Línea vertical para ilustrar la trayectoria de la herramienta en avance
dimensioning_lines_hori.fxw	Líneas auxiliares de dimensión horizontales
dimensioning_lines_vert.fxw	Líneas auxiliares de dimensión verticales
z1_inc.fxw	Línea auxiliar horizontal con el identificador Z1
x1_inc.fxw	Línea auxiliar vertical con el identificador X1
3d_coordinate_origin.fxw	Cruz de coordenadas 3D
3d_zero_point.fxw	Origen 3D

Torneado

Plantilla	Descripción
turning_blank.fxw	Pieza en bruto para la tecnología Tornear: un cilindro simple
turning_centerline.fxw	Línea central
turning_centerpoint.fxw	Cruz de coordenadas para representar el origen en WKS
turning_refpoint.fxw	Punto de referencia para, p. ej., un mecanizado
turning_machining_area.fxw	Líneas de limitación para la superficie de mecanizado

Fresado

Plantilla	Descripción
milling_blank.fxw	La pieza en bruto para la tecnología Fresar: un paralelepípedo simple
milling_centerline.fxw	Línea central
milling_refpoint.fxw	Punto de referencia para, p. ej., un mecanizado

C.3 Comandos XML

C.3.1 Vista general

Para que el visor de X3D pueda acceder a los gráficos se necesita el fichero de descripción de escenas (*.xml). En el fichero de descripción de escenas se asignan los nombres de escena que se llaman desde la configuración a los tiempos del fichero *.x3d (TimeSensor) en los que se encuentran los gráficos.

C.3.2 Estructura del fichero de descripción de escenas

A continuación se muestra la estructura del fichero de descripción de escenas y se explican los comandos XML correspondientes:

Estructura y descripción

```
<!-- Tratamiento de textos dependientes del plano en el torneado -->
<TextPlane plane="G18_ZXY" />
```

Descripción

<TextPlane plane="G18_ZXY" /> — Tratamiento de textos dependientes del plano (nombres de eje) específicos para torneados. Si, p. ej., se utiliza un texto con la identificación "Z" (con G18, el primer eje geométrico), el correspondiente identificador de eje geométrico del control se representa en la pantalla de ayuda (p. ej., "Z").

```
<!-- Alineación del texto -->
<TextPosition center='true' />
```

Descripción

<TextPosition center='true' /> — Identificación de que los textos están alineados en el centro. Esto es relevante para la representación de los textos en pantallas giradas. Se recomienda utilizar este ajuste.

<!-- Adaptar la velocidad de la herramienta -->

```
<ToolSpeedFactors planeSpeedFactor="0.4" rapidSpeedFactor="0.7"
reducedSpeedFac¬tor="1.0"/>
```

Descripción

<ToolSpeedFactors	Si es necesario modificar las velocidades de la herramienta, puede agregarse esta entrada.
planeSpeedFactor="0.4"	Factor para la velocidad de avance (0.4 = 40%)
rapidSpeedFactor="0.7"	Factor para la velocidad en rápido (0.7 = 70%)
reducedSpeedFactor="0.1" />	Factor para una tercera velocidad (0.1 = 10%)

<!-- Definición de una animación -->

```
<SceneKey name='BoringAnimation' masterRotationSpeed="-64.0"
maxRotAngle="45.0" begin-Time='0.110 endTime='0.118 view='camiso'
speedMaster='boringtool' Type="VIEW_3D_TURN_CYL">
```

Descripción

<SceneKey name='BoringAnimation'	Nombre de la animación
masterRotationSpeed="-64.0"	Sentido de giro y velocidad de giro de la herramienta. El ajuste es opcional.
maxRotAngle="45.0"	Prevención del efecto alias (la herramienta parece girar en el sentido incorrecto). El valor suele corresponder a una cuarta parte de la simetría de la herramienta. El ajuste es opcional.
beginTime='0.110	Instante inicial de la animación en el TimeSensor
endTime='0.118'	Instante final de la animación en el TimeSensor
view='camiso'	Cámara que se utiliza para la animación
speedMaster='boringtool'	Nombre de la herramienta para la animación
type="VIEW_3D_TURN_CYL">	El tipo de vista determina la vista (ver capítulo Tipo de vista (Página 331))

<!-- Elementos -->

```
<Element name='1' time='0.110' feed='rapid' dwell='2'/>
<Element name='2' time='0.112' feed='rapid'/>
<Element name='3' time='0.114' feed='rapid'/>
<Element name='4' time='0.116' feed='plane'/>
<Element name='5' time='0.118' feed='plane'/>
```

Descripción

<Element name='1'	Indica que se trata del primer elemento de la animación.
time='0.110'	Instante del primer elemento en el TimeSensor
feed='rapid'	Velocidad de desplazamiento del elemento plane = avance de mecanizado rapid = rápido reduced = velocidad reducida, más lento que "plane"
dwell='2'/>	Pausa en el movimiento de desplazamiento de la animación. Una herramienta giratoria permanece en rotación.

<!-- Fin de la definición de animación -->

```
</SceneKey>
```

Descripción

</SceneKey>	Fin de la definición de animación

<!-- Animación existente como origen de una animación nueva -->

```
<SceneKey source=''BoringAnimation' name=''BoringAnimationRear' mirror='
MirrorScreenX ' />
```

Descripción

<SceneKey source="BoringAnimation'	Nombre de una animación que debe utilizarse como origen de otra animación.
name="BoringAnimationRear'	Nombre de la animación nueva, basada en la animación origen
mirror=' MirrorScreenX ' />	Simetría en sentido del eje X; el resultado encuentra salida en la animación nueva.

<!-- Definición de una escena estática (pantalla), (4 ejemplos) -->

```
<SceneKey name='Default' time='0.026' view='camiso'
type="VIEW_3D_DRILL_CUT"/>
<SceneKey name='Cut' time='0.046' view='camside' type="VIEW_SIDE"/>
<SceneKey name='Z1ink' time='0.056' view='camside' type="VIEW_SIDE"
 highLightedGroup='dad_Z1ink'/>
<SceneKey name='Z1abs' time='0.066' view='camside' type="VIEW_SIDE"
 highLightedGroup='dad_Z1abs'/>
```

Descripción

<SceneKeyname='Z1abs'	Nombre de la imagen
time='0,066'	Instante de la pantalla en el TimeSensor
view='camside'	Cámara que se utiliza para la pantalla
type="VIEW_SIDE"	El tipo de vista determina la vista (ver capítulo Tipo de vista (Página 331))
highLightedGroup='dad_Z1abs'/>	Nombre del grupo que se desea resaltar. El grupo se ha denominado "Z1abs" en FluxStudio. Flux antepone el suplemento "dad_" automáticamente.

<!-- Fin del fichero xml –!>

C.3.3 Simetrías y rotaciones

Con el comando **mirror** se determina la simetría o la rotación de la pantalla/el modelo.

Descripción

mirror="RotateScreenX"	La pantalla gira sobre el eje X
mirror="RotateScreenY"	La pantalla gira sobre el eje Y
mirror="RotateScreenZ"	La pantalla gira sobre el eje Z
mirror="MirrorScreenX"	La pantalla se refleja simétricamente en el sentido del eje X
mirror="MirrorScreenY"	La pantalla se refleja simétricamente en el sentido del eje Y
mirror="MirrorScreenZ"	La pantalla se refleja simétricamente en el sentido del eje Z
mirror="RotatePieceX"	El modelo gira sobre el eje X
mirror="RotatePieceY"	El modelo gira sobre el eje Y
mirror="RotatePieceZ"	El modelo gira sobre el eje Z
mirror="MirrorPieceX"	El modelo se refleja simétricamente en el sentido del eje X
mirror="MirrorPieceY"	El modelo se refleja simétricamente en el sentido del eje Y
mirror="MirrorPieceZ"	El modelo se refleja simétricamente en el sentido del eje Z

Todas las posibilidades de simetría y giro pueden combinarse. Para la rotación es necesario determinar el ángulo de rotación.

Ejemplos

mirror="RotatePieceZ=180"

mirror="RotatePieceZ=-90 MirrorScreenX"

mirror="RotateScreenY=90"

mirror="MirrorPieceX MirrorPieceY"

mirror="MirrorPieceZ RotatePieceX=-90"

C.3.4 Tipo de vista

Con el tipo de vista se determinan las vistas. Las vistas se basan en valores empíricos y reglas que permiten representar convenientemente los objetos.
De esta forma se garantiza que los objetos se representen de acuerdo con los ajustes del sistema de coordenadas de la interfaz de usuario (MD 52000 $MCS_DISP_COORDINATE_SYSTEM o bien MD 52001 $MCS_DISP_COORDINATE_SYSTEM_2).

Descripción

"VIEW_STATIC"	Vista directa sin conversión
"VIEW_3D_TURN_CYL"	Vista 3D para torneados (cilindro)
"VIEW_3D_MILL_CUBE"	Vista 3D para fresados (paralelepípedo)
"VIEW_3D_DRILL_CUT"	Vista 3D para taladros (representación en sección)
"VIEW_SIDE"	Vista 2D para taladros (representación en sección)
"VIEW_TOP_GEO_AX_1"	Vista 2D desde el sentido del 1.er eje geométrico
"VIEW_TOP_GEO_AX_2"	Vista 2D desde el sentido del 2.° eje geométrico
"VIEW_TOP_GEO_AX_3"	Vista 2D desde el sentido del 3.er eje geométrico

C.4 Conversión a fichero hmi

Nota

Los ficheros X3D, junto con los correspondientes ficheros XML, se convierten en ficheros HMI durante el arranque del HMI.

Para cada fichero X3D debe crearse el fichero XML correspondiente con el mismo nombre. Para ello es necesario guardar los ficheros X3D y los ficheros XML en el directorio de la `ruta de búsqueda de HMI\ico\x3d\turning` o `milling`. El procedimiento es similar al de los ficheros .ts.

C.5 Visualización en Create MyHMI/3GL

C.5.1 Visor de X3D

Para representar pantallas de ayuda concretas en una aplicación OA propia se debe integrar el widget del visor de X3D en dicha aplicación OA.

El widget del visor de X3D ofrece interfaces que permiten presentar contenidos en X3D dentro del HMI.

Para representar escenas gráficas se dispone de la clase SlX3dViewerWidget. La definición de la clase se encuentra en el correspondiente fichero de cabecera (header) slx3dviewerwidget.h en el directorio GUI Include global `\hmi_prog\gui\include`.

C.5.2 Clase SlX3dViewerWidget

Esta clase ofrece un widget de aplicación flexible que representa de forma autónoma los contenidos de un fichero modelo indicado durante el tiempo de ejecución y que, en caso necesario, permite ejecutar la animación.

La interfaz de la clase está formada por constructor, destructor y dos métodos para controlar la salida gráfica.

Como derivación directa de la clase Qt QWidget se dispone de una interfaz mucho más amplia, no descrita con mayor detalle aquí (p. ej., show(), hide() y resize(...); hallará más información al respecto en la documentación de Qt).

C.5.3 Métodos públicos

SlX3dViewerWidget (QWidget* pParent = 0)

Constructor del widget del visor de X3D.

Parámetro	Significado
pParent	El parámetro se trasfiere al constructor del Qwidget.

~SlX3dViewerWidget ( )

Destructor del widget del visor de X3D.

Parámetro	Significado
-	-

C.5.4 Slots públicos

void viewSceneSlot (const QString& rsFileName, const QString& rsScene, const QString& rsAnimationScene, int nChannel, int nPlane, SlStepTechnology nTechnology)

Con el método viewSceneSlot se le indica al visor de X3D que debe cargar la escena estática rsScene y la escena animada rsAnimationSzene del fichero rsFileName y representarlas intercaladas.

En concreto se repite primero la escena estática durante un intervalo de tiempo fijo y, a continuación, se muestra la escena animada.

Si no se indica ninguna escena estática, la animación se representa de inmediato; de la misma forma también puede no haberse indicado ninguna escena animada.

El número de canal, el plano y la tecnología se emplean para girar las escenas hasta la posición correcta (en función del sistema de coordenadas de máquina ajustado).

Parámetro	Significado
rsFileName	Nombre del fichero que contiene las escenas que se desean representar.
rsScene	Nombre de la escena estática.
rsAnimationScene	Nombre de la escena animada.
nChannel	Número de canal
nPlane	Plano
nTechnology	Tecnología Para el tipo de enumeración SlStepTechnology están definidas las siguientes constantes: • SL_STEP_NO_TECHNOLOGY • SL_STEP_MILLING • SL_STEP_TURNING • SL_STEP_SURFACE_GRINDING • SL_STEP_CIRCULAR_GRINDING

void viewSceneSlot (const QString& rsFileName, const QString& rsScene, const QString& rsAnimationScene)

Esta es una forma simplificada del método viewSceneSlot, con el que se le puede indicar al visor de X3D que debe cargar y representar la escena estática rsScene o la escena animada rsAnimationScene del fichero rsFileName.

Parámetro	Significado
rsFileName	Nombre del fichero que contiene las escenas que se desean representar.
rsScene	Nombre de la escena estática.
rsAnimationScene	Nombre de la escena animada.

C.5.5 Librerías

Para poder utilizar el visor de X3D en proyectos propios es necesario agregar la entrada 'slx3dviewer.lib' a la lista de dependencias de librerías.

C.5.6 Ejemplo de implementación

Encontrará un ejemplo de implementación en el paquete Create MyHMI/3GL, en **\examples \GUIFrameWork\SlExGuiX3D**.

C.5.7 Datos de máquina

Visualización de MD 9104 $MM_ANIMATION_TIME_DELAY

Retardo hasta utilizar la animación en pantallas de ayuda (en segundos). El ajuste no tiene efecto en pantallas de ayuda únicamente animadas.

El ajuste tiene un efecto global en todas las animaciones de SINUMERIK Operate.

C.5.8 Indicaciones sobre la aplicación

- La animación se interrumpe cuando se oculta el widget del visor de X3D. No es necesario seleccionar escenas no animadas en estos casos.
- No conviene instanciar periódica o repetidamente el widget del visor de X3D a fin de no menoscabar el rendimiento y la memoria necesaria. Para estos casos de aplicación se recomienda el uso (la implementación) de un singleton del visor de X3D.
- El visor de X3D también permite el uso en sistemas signal/slot.
- Si se produce un error (p. ej., no se encuentra el fichero o se desconoce el nombre de escena), el widget del visor de X3D muestra un área de mensajes. Dicha área vuelve a ocultarse automáticamente en cuanto se activa otro fichero de la pantalla de ayuda.

C.6 Visualización en Run MyScreens

Este capítulo se dirige a desarrolladores de "Run MyScreens" expertos. Los conocimientos básicos necesarios pueden consultarse en la documentación correspondiente.

Además del uso de pantallas en los formatos .bmp o .png, el visor de X3D también permite representar pantallas de ayuda animadas.

Para incorporarlas deberá utilizar, como de costumbre, la interfaz Run MyScreens para pantallas de ayuda.

Para que se muestren pantallas en formato .bmp o .png debe introducirse la indicación XG0 en la definición de un diálogo, en el apartado de atributos, o bien dejar el parámetro vacío. Para incorporar pantallas de ayuda en X3D en un diálogo, debe introducirse la indicación **XG1** en los atributos.

```
//M(MASK_F_DR_O1_MAIN/$85407////52,80/XG1)
```

Para controlar las diferentes pantallas de ayuda se necesitan los siguientes parámetros, cuyo significado se describe en el capítulo 5.3 Slots públicos:

1. Nombre de fichero
2. StaticScene (opcional)
3. AnimationScene (opcional)
4. Tecnología (opcional)
5. Plano (opcional)

Los parámetros se agrupan en una cadena de caracteres siguiendo este orden y separados por una coma.

```
Hlp = "nombre de fichero,StaticScene,AnimationScene,tecnología,plano"
```

Ejemplos

- La pantalla de ayuda predeterminada puede ajustarse con la variable Hlp especificada.
  En el siguiente ejemplo se emite la animación "MyAnimation" desde el fichero "MyDlgHelp.hmi". No se ha indicado ninguna StaticScene, por lo que no se emite ninguna escena estática. Tampoco se ha hecho ninguna indicación en cuanto a la tecnología y el plano, por lo que se utilizan valores predeterminados.
  ```
  Hlp = "MyDlgHelp.hmi,,MyAnimation"
  ```
- Para los diferentes parámetros de la máscara de entrada, en las variables correspondientes puede asignarse la propiedad hlp a una pantalla de ayuda específica.
  En el siguiente ejemplo se muestran la escena estática "MyParam" y la animación "MyAnimation" del fichero "MyDlgHelp.hmi" en el plano G17. No se ha hecho ninguna indicación en cuanto a la tecnología, por lo que se utiliza un valor predeterminado.
  ```
  VarMyParam.hlp = "MyDlgHelp.hmi,MyParam,MyAnimation,,G17"
  ```

Índice alfabético

G

H

I

L

M

N

O

P

R

S

T

V

W