ISaGRAF

Versión 3.4

MANUAL
DE USUARIO

CJ INTERNATIONAL

La información contenida en este documento está sujeta a cambios sin previo aviso y no representa compromiso alguno por parte de CJ International. El *software* y/o base de datos descritos en este documento se proporcionan bajo un convenio de licencia o convenio de no divulgación y pueden utilizarse o copiarse únicamente en conformidad con los términos del mencionado convenio. Es ilegal copiar el *software* salvo en los casos contemplados específicamente en el convenio de licencia o de no divulgación. No está permitida la reproducción total o parcial de este manual de ninguna forma ni por ningún método, ya sea electrónico, mecánico, por fotocopia o por registro, para propósito alguno sin el consentimiento expreso y por escrito de CJ International.

Impreso en Francia por CJ International.
3 Rue Hector Berlioz
F-38600 FONTAINE
Phone: 33 (0)4 76 26 87 30
Fax: 33 (0)4 76 26 87 39

Traducido por SAINCO.

Índice general

Nota: Se facilita un índice detallado al principio de cada sección.
Al término del documento, aparece un índice de búsqueda por términos.

A. Manual del Usuario

A.1 Primeros pasos

Este apartado está dedicado a la instalación del banco de trabajo ISaGRAF. Incluye asimismo un pequeño ejemplo de una aplicación ISaGRAF, aportando al usuario una breve descripción de sus principales características y permitiendo a éste el uso inmediato de ISaGRAF.

A.1.1 Instalación de ISaGRAF

Este apartado está dedicado a la instalación del banco de trabajo ISaGRAF y a la manera de configurar el ordenador para el desarrollo de aplicaciones.

Requisitos de hardware y software

Se puede instalar el banco de trabajo ISaGRAF en cualquier ordenador que cumpla los requisitos mínimos de Windows Versión 3.1. Sin embargo, se recomienda el siguiente *hardware* para el desarrollo de aplicaciones:

- Ordenador personal equipado con microprocesador 80486 o superior
- 8 MB de memoria convencional y extendida
- Unidad de disco de 3,5 pulgadas (1,44 MB)
- Unidad de disco duro con al menos 20 MB de espacio libre
- Tarjeta gráfica VGA o SVGA con monitor compatible
- Ratón (necesario para las herramientas gráficas de desarrollo)
- Puerto paralelo LPT1 (necesario para la llave de protección)

Antes de instalar el banco de trabajo ISaGRAF, el siguiente *software* debe estar ya instalado en el sistema:

- Windows Versión 3.1 operando en 'modo mejorado (enhanced) 386'
- Windows 95
- Windows NT Versión 3.51 ó 4.00

Cómo utilizar el programa de instalación

Se instala el banco de trabajo ISaGRAF con INSTALL, el programa de instalación de ISaGRAF. Este programa copia el *software* ISaGRAF de los discos de ISaGRAF al disco duro del usuario. INSTALL también se encarga de añadir el grupo "ISaGRAF" a la ventana del Gestor de Programas y de crear un fichero de inicialización denominado "ISA.ini" en el subdirectorio **EXE** que se instala.

INSTALL es un programa para el entorno Windows, que debe ejecutarse desde el Gestor de Programas de Windows o desde el comando Ejecutar del menú de Inicio de Windows 95. Para instalar ISaGRAF, deben llevarse a cabo los siguientes pasos:

• Insertar el Disco nº 1 en la unidad de disco apropiada
• Desde el Gestor de Programas, seleccionar la opción de “Ejecutar” (“Run”) en el menú “Fichero” (“File”) y teclear "A:\INSTALL.exe" a modo de línea de comando de programa. Alternativamente, se puede teclear el comando "WIN A:\INSTALL.exe" desde el *prompt* de MS-DOS.
• Seguir las instrucciones que aparecen en pantalla para completar la instalación. Se recomienda la instalación del banco de trabajo ISaGRAF en un directorio nuevo para evitar posibles conflictos con versiones anteriores de ISaGRAF.

INSTALL preguntará si se desea instalar los siguientes componentes:

• Programas ejecutables ISaGRAF
• Ficheros de información y ayuda en línea
• Librerías estándares ISaGRAF
• Muestras de aplicaciones ISaGRAF

Si se instala ISaGRAF por primera vez, es altamente recomendable la instalación de todos los componentes. No obstante, pueden añadirse componentes adicionales en fechas posteriores mediante la reinstalación del banco de trabajo ISaGRAF.

El nombre por defecto del directorio principal de ISaGRAF es "\ISAWIN". Esto facilita la instalación de ISaGRAF para Windows en el mismo disco que contenga una versión de ISaGRAF para MS-DOS. Para más información sobre la arquitectura de disco de ISaGRAF, véase la sección dedicada a “Directorios ISaGRAF” en el apartado “Técnicas avanzadas”. Una vez que se hayan copiado todos los ficheros ISaGRAF, se añade el siguiente grupo a la ventana del Gestor de Programas:

Los principales iconos de ISaGRAF son:

Projects: ..... (Proyectos) Gestión de proyectos
Libraries: ..... (Librerías) Gestión de librerías
Archive: ..... (Archivo) Herramienta de copia de seguridad / restauración de ficheros
Book: ..... (Libro) Información en línea sobre ISaGRAF
Diagnosis: ..... (Diagnóstico) Herramienta de diagnóstico para el usuario final
Read Me: ..... (Léeme) Información sobre la nueva versión de ISaGRAF
Report: ..... (Informe) Impreso estándar para informar sobre errores

En el caso de que surja cualquier problema, se puede utilizar el impreso estándar para informar sobre errores. Abrir el impreso, aportar los datos solicitados

y utilizar el comando de menú Fichero/Guardar Como (Fichero/Guardar como) para salvar el documento con un nombre de fichero determinado. Después, enviar el fichero a CJ International mediante fax o correo electrónico.

Actualización de ficheros de sistema

Una vez que se haya concluido la instalación, será necesario actualizar el fichero CONFIG.SYS antes de reinicializar el ordenador. No es necesaria la inserción de la ruta del directorio ISaGRAF en la variable PATH.
ISaGRAF no utiliza las variables del entorno MS-DOS. Sin embargo, se pueden añadir las siguientes sentencias al fichero CONFIG.SYS:

```
files=20
buffers=20
```

El banco de trabajo ISaGRAF utiliza un puerto serie para comunicar con el PLC (Controlador de Lógica Programable) objeto. El puerto serie por defecto es COM1. Si el ratón también utiliza un puerto serie, deberá seleccionarse COM2 para el ratón para que la especificación por defecto de COM1 sea válida para cualquier nueva aplicación ISaGRAF.

Después de actualizar el fichero CONFIG.SYS, será necesario reiniciar el ordenador para que tengan efecto los cambios.

⇒ **Nota importante para el usuario de Windows NT:**
Cuando se utiliza el banco de trabajo bajo Windows NT 3.51 ó 4.00, se tiene que insertar la siguiente línea en la sección [WS001] del fichero ISA.ini, ubicado en el directorio \ISAWIN\EXE:

```
[WS001]
NT=1
Isa=C:\ISAWIN
IsaExe=C:\ISAWIN\EXE
IsaApl=C:\ISAWIN\APL1
IsaTmp=C:\ISAWIN\TMP
```

Esta inserción es absolutamente indispensable para la comunicación RS.

La llave de seguridad

Una llave *hardware* protege al programa ISaGRAF contra las copias ilegales. Sin embargo, la mayoría de las funciones del banco de trabajo ISaGRAF siguen estando disponibles cuando la llave no está colocada. La llave de seguridad también define la opción del banco de trabajo ISaGRAF y el tamaño máximo de las aplicaciones desarrolladas. Cuando la llave no está colocada o está conectada incorrectamente, algunas de las funciones del banco de trabajo ISaGRAF no funcionarán. Esto constituye un comportamiento NORMAL. Para asegurar que la llave está conectada correctamente, seleccionar "**Acerca de**" en el menú de "**Ayuda**" de cualquier ventana de ISaGRAF. Se podrá visualizar la opción disponible del banco de trabajo ISaGRAF.

Se puede conectar la llave a cualquier puerto paralelo del ordenador. Si el equipo dispone de más de un puerto paralelo, es preferible que se conecte la llave y la impresora a puertos diferentes. En algunas configuraciones de PC/impresora, es posible que no se reconozca la llave cuando su salida esté conectada a una

impresora *fuera de línea*. En este caso, desconectar la impresora, o encenderla en modo en línea y reiniciar el banco de trabajo ISaGRAF.

Nota: No se necesita llave para el banco de trabajo **ISaGRAF-32**.

⇒ **Nota importante para el usuario de Windows NT:**
En los sistemas Windows NT, se debe instalar el controlador Sentinel/Rainbow™ para que se reconozca la llave de protección. Se proporciona un disquete independiente para esta finalidad.

A.1.2 Como usar la información en línea

Junto con el banco de trabajo ISaGRAF, se instala información en línea que cubre las siguientes áreas:
- Referencia de lenguajes ISaGRAF
- Manual del Usuario completo (para cualquier herramienta ISaGRAF)
- Notas técnicas para los elementos ubicados en las librerías

Desde cualquier ventana ISaGRAF, seleccionar las opciones necesarias del menú de "**Ayuda**" para visualizar la información en línea sobre las características principales (como los lenguajes) y la herramienta que se esté utilizando en ese momento.
ISaGRAF dispone asimismo de un icono estándar de Ayuda de Windows, que viene a completar la ayuda relativa a ISaGRAF.

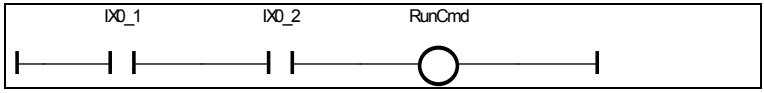
A.1.3 Aplicación de muestra

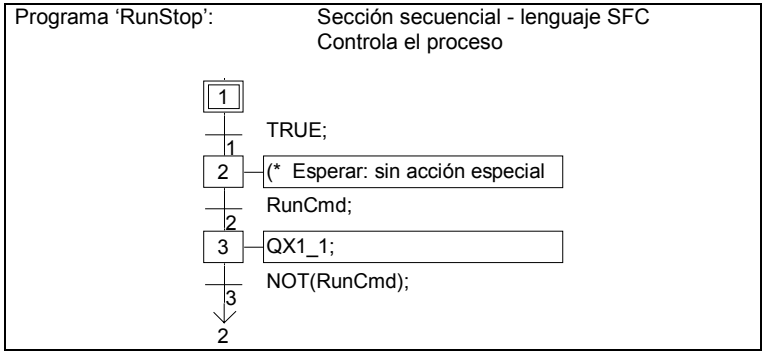
Este apartado aporta una explicación paso a paso de todas las operaciones básicas que son necesarias para realizar, diseñar, generar y probar una aplicación multi-lenguaje corta pero completa.

A continuación se muestra las especificaciones completas de esta aplicación, combinando representaciones LD y SFC:

Variables booleanas:	
IX0_1, IX0_2:	variables de entrada para orden de proceso
RunCmd:	comando interno "run/stop"
QX1_1:	variable de salida: estado del proceso

Comando de programa:	Sección inicial cíclica - lenguaje LD Evalúa el comando interno "run/stop"





Cómo ejecutar el banco de trabajo ISaGRAF

Para ejecutar el banco de trabajo ISaGRAF, hacer doble click con el ratón sobre el icono "Proyectos" del grupo "ISaGRAF". De esta manera se abre la ventana de Gestión de Proyectos.

Cómo crear el proyecto

Crear el proyecto (denominado "RunStop") utilizando el comando "Nuevo" del menú "Fichero" o pulsando el botón "Nuevo". En la ventana de diálogo que se abre:

Introducir el nombre del proyecto: "**RunStop**"
Seleccionar la configuración de E/S: "**Sim_Boo**"
Pulsar el botón "**Aceptar**".
El proyecto ya ha sido creado.

Cómo abrir el proyecto

Los programas de proyecto se definen abriendo la ventana del Gestor de Programas de ISaGRAF. Utilizar el comando "Edición" de la ventana del Gestor de Proyectos, hacer doble click con el ratón sobre el nombre del proyecto o utilizar el botón "Edición".

Cómo crear los programas

Ahora la ventana del Gestor de Programas se encuentra abierta y vacía (sin programas definidos). Se crea el primer programa utilizando bien el comando "Nuevo" del menú de "Fichero" o bien el botón "Nuevo". En la ventana de diálogo que aparece:

Introducir el nombre del programa: "**Command**".
Seleccionar el lenguaje "**Quick LD**".
Seleccionar la sección "**Comienzo de ciclo**".
Pulsar el botón "**Aceptar**" para crear el programa.

Se tiene que repetir la misma operación para el segundo programa:

Utilizar bien el comando “Nuevo” del menú de "Fichero" o el botón “Nuevo”. En la ventana de diálogo que aparece:

Introducir el nombre del programa: "**RunStop**".
Seleccionar el lenguaje "**SFC**".
Seleccionar la sección "**Secuencial**".
Pulsar el botón "**Aceptar**" para crear el programa.

Los programas ya han sido creados, y deben aparecer en la ventana del Gestor de Programas.

Declaración de las variables

Antes de ejecutar los programas, se tiene que declarar la variable interna que se usará en la programación. Este se logra mediante la utilización del comando "Diccionario" del menú de "Fichero" o de el botón de “Diccionario”. Se declaran automáticamente las variables de E/S en el momento de crearse el proyecto.

Se abre la ventana del diccionario. Dentro del menú de "Fichero" y los submenús “Otro" y "Variables globales" y mediante el comando "Booleanos", seleccionar el diccionario booleano global. Se pueden utilizar los botones “Objetos globales” y “Booleano” para lograr el mismo efecto.

El comando "Nuevo" del menú "Edición" se utiliza para crear nuevas variables booleanas. También se puede utilizar el botón de “Insertar objetos”. En la ventana activa de diálogo, introducir la descripción de la variable interna:

nombre: **RunCmd**
comentario: **Ejecutar/Parar comando: interna**
atributo: Seleccionar el atributo "**Interno**"
Pulsar el botón "**Almacenar**": se crea la variable.
Pulsar el botón "**Cancelar**" para salir de la ventana de diálogo.

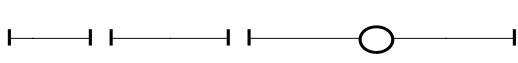
Por último, salir del editor de diccionarios y guardar las modificaciones que se han introducido: Menú "**Fichero**" - Comando "**Salir**". Hacer click sobre "**SI**" para guardar las modificaciones.

Edición del Programa Quick LD

Para empezar a editar el programa LD "Command", hacer doble click sobre su nombre en la ventana del Gestor de Programas o utilizar el botón “Editar programa”.

Se abre la ventana del Editor del programa Quick LD de ISaGRAF. Para aumentar el área de trabajo, redimensionar la ventana para utilizar la pantalla completa.

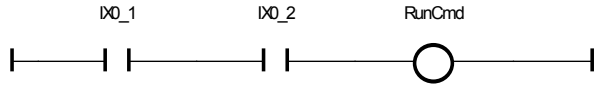
F2 F3 Pulsar los botones F2 y F3:



Intro Asociar variables a los símbolos LD: desplazar el cursor utilizando las flechas del teclado. Situar el cursor sobre cada símbolo y pulsar la tecla de “Intro”. Se abre la ventana de diálogo de la sección variable.
Para el primer contacto, teclear en la ventana de selección de variables: IX0_1 y después pulsar “Intro”.
Para el segundo contacto, teclear en la ventana de selección de variables: IX0_2 y después pulsar “Intro”.

Para la bobina (salida), teclear en la ventana de selección de variables: RunCmd y después pulsar “Intro”.

El programa ya está completo. A continuación se muestra el resultado:



Salir del editor y guardar las modificaciones introducidas: Menú "**Fichero**" - Comando "**Salir**". Hacer click sobre "**SI**" para guardar las modificaciones.

Edición del Programa SFC

Para empezar a editar el programa SFC "RunStop", hacer doble click sobre su nombre en la ventana del Gestor de Programas o utilizar el botón “Edición”.

Se abre la ventana del Editor SFC. Para aumentar el área de trabajo, redimensionar la ventana para utilizar la pantalla completa:

Seleccionar las opciones de edición apropiadas:
Si no estuviera seleccionada, marcar la opción "**Comentarios**" en el menú "**Opciones**".
Si no estuviera seleccionada, marcar la opción "**Entrada Automática**" en el menú "**Opciones**".
Seleccionar el comando de "**Distribución**" en el menú "**Opciones**". En la ventana de diálogo de “Distribución”, seleccionar todas las opciones.
Se muestran las coordenadas del cursor en la parte inferior de las ventanas SFC, en la barra de estado.
El paso inicial ya existe.

Para la primera transición, seleccionar el botón correspondiente en la barra de herramientas del editor.
Hacer click sobre la celda (**0,1**) para colocar la transición en el diagrama.
No se definen comentarios para esta transición del tipo “siempre verdadero”.
Pulsar el botón "**Aceptar**".

Ahora, para los restantes pasos y transiciones, se selecciona el botón correspondiente de la barra de herramientas de forma automática. Ya se está en disposición de realizar un paso:

Hacer click sobre la celda (**0,2**).
Introducir los comentarios para este paso.
Introducir "**Esperar por comando**" y pulsar el botón "**Aceptar**".
Hacer click sobre la celda (**0,3**).
Introducir los comentarios para este paso.
Introducir "**Comando Run**" y pulsar el botón "**Aceptar**".
Hacer click sobre la celda (**0,4**).
Introducir los comentarios para este paso.
Introducir "**El proceso está corriendo**" y pulsar el botón "**Aceptar**".
Hacer click sobre la celda (**0,5**).
Introducir los comentarios para este paso.

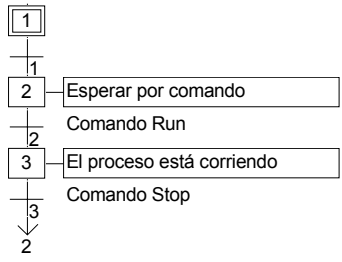
Introducir "**Comando Stop**" y pulsar el botón "**Aceptar**".

Para el símbolo de “Saltar a un paso”, seleccionar este botón en la barra de herramientas. Hacer click sobre la celda (**0,6**) para colocar el símbolo en el diagrama.

Introducir el número de referencia del paso destino.
Introducir "**3**" y pulsar el botón "**Aceptar**".

El diagrama ya está completo. Utilizar bien el comando "Renumerar" del menú "Edición" o bien el botón correspondiente en la barra de herramientas para establecer números de referencia consecutivos.

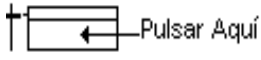
El diagrama resultante tiene el siguiente aspecto:



Para introducir la descripción de “Nivel 2” de los pasos y las transiciones, seleccionar la representación apropiada. Ya no es necesaria la barra de herramientas. Seleccionar la opción "**Nivel 2**" en el menú "**Opciones**".

Comprobar si se está en el modo de selección. Seleccionar este icono en la barra de herramientas.

Para introducir la programación de la transición “2”, hacer doble click sobre el rectángulo inferior del símbolo de transición:



(**0,3**) Hacer doble click sobre la celda (**0,3**):
Se abre una ventana de texto. Introducir la programación de nivel 2 para la transición 2:

RunCmd;

Hacer click sobre el botón “Cerrar” para salir de la ventana de nivel 2.

Se realiza la misma operación para el paso 3:

(**0,4**) Hacer doble click sobre la celda (**0,4**):
Se abre una ventana de texto. Introducir la programación de nivel 2 para el paso 3:

QX1_1;

Hacer click sobre el botón “Cerrar” para salir de la ventana de nivel 2.

(**0,5**) Si la transición 3 no está completamente visible en pantalla, desplazar la ventana del editor (*scroll*).
Hacer doble click sobre la celda (**0,5**):

Se abre una ventana de texto. Introducir la programación de nivel 2 para el paso 3:

Not (RunCmd);

Hacer click sobre el botón “Cerrar” para salir de la ventana de nivel 2.
El programa SFC ya está terminado. Salir del editor por medio del Menú "**Fichero**" y el Comando "**Salir**", y guardar las modificaciones introducidas haciendo click sobre "**SI**".

Elaboración del código de aplicación

Utilizar bien el menú "**Ensamblar**" y el comando de "**Construir aplicación**" desde la ventana del Gestor de Programas, o bien el botón correspondiente de la barra de herramientas, para elaborar el código de aplicación.

Cuando finaliza la generación de código, aparece una ventana de diálogo que pregunta al usuario si desea salir de la generación de código **ahora** o si quiere **continuar** trabajando. Pulsar el botón "**Salir**".

Simulación

Utilizar bien el menú "**Depurar**" y el comando "**Simular**" desde la ventana del Gestor de Programas, o bien el botón correspondiente de la barra de herramientas, para ejecutar el simulador del *kernel* de ISaGRAF.

Cuando aparezca la ventana del Simulador, se puede probar la aplicación. En este ejemplo, se debe actuar sobre las entradas 1 y 2 (botones verdes) para ejecutar el proceso (salida LED’s rojos).

Cerrar la ventana del **Depurador** y salir de la simulación: Menú "**Fichero**" - Comando "**Salir**".

A.2 Gestión de proyectos

Para ejecutar la herramienta de gestión de proyectos de ISaGRAF, hacer doble click sobre el icono de "Proyecto", en el grupo de ISaGRAF. Se abrirá la venta del "Gestor de Proyectos".
Cada proyecto corresponde a una operación en bucle sobre un PLC objeto. La ventana superior contiene la lista de proyectos existentes. El texto que describe el proyecto seleccionado se muestra en la ventana inferior.

Cómo redimensionar ventanas

Sólo hay que hacer click sobre el separador (divisor) que existe entre la lista y el descriptor de proyectos para redimensionar las ventanas correspondientes. La ventana del descriptor no se puede ocultar en su totalidad. Contiene siempre al menos un renglón de texto.

Cómo insertar separadores

Se puede insertar una línea de separación delante de cualquier nombre de proyecto. Esto permite la agrupación de determinados proyectos que están vinculados a la misma aplicación en la lista. Utilizar el comando "**Edición / Separador transversal**" para insertar o borrar un separador ubicado delante del proyecto seleccionado.

Cómo desplazar proyectos en la lista

Para desplazar un proyecto en la lista, primero se tiene que seleccionar (resaltar). Después, hacer click sobre su nombre y arrastrarlo hasta su nueva posición en la lista. Cuando se arrastra un proyecto, aparece una flecha pequeña en el margen izquierdo que señala el lugar en el que va a ser depositado. También se puede utilizar los comandos "**Subir/Bajar en la lista**" del menú "**Edición**" para desplazar el proyecto seleccionado línea por línea. Obsérvese que si se coloca un separador delante del proyecto seleccionado, se desplazará junto con el proyecto.

A.2.1 Cómo crear y trabajar con proyectos

Se emplean los comandos del menú del gestor de proyectos para crear nuevos proyectos, para editarlos y para gestionar proyectos existentes.

Creación de un nuevo proyecto

Para crear un nuevo proyecto, el primer paso es la introducción de un nombre. De esta manera se crea un proyecto vacío, con ningún objeto en él. Se puede asociar una configuración de E/S al recién creado proyecto. Esta configuración de E/S debe estar definida en la librería. Si se elige una configuración, ISaGRAF configurará la conexión E/S automáticamente y declarará las correspondientes

variables E/S en el nuevo diccionario de proyecto. A la hora de crear o renombrar un proyecto, se debe cumplir con las siguientes normas de denominación:

- el nombre no puede superar los **8** caracteres
- el primer carácter tiene que ser una **letra**
- los restantes caracteres pueden ser **letras**, **dígitos** o el carácter de subrayado "_".
- el nombre del proyecto es insensible al uso de mayúsculas o minúsculas

Al crear un proyecto, utilizar el comando "**Edición / Establecer texto de comentario**" para introducir el texto que se mostrará junto al nombre del proyecto en la lista.

Edición del descriptor de proyecto

Se emplea el comando "**Proyecto / Descriptor de proyecto**" para editar el descriptor textual del proyecto. Este documento aporta una identificación completa del proyecto, diferenciándolo de los otros existentes en la lista. También se puede utilizar el descriptor de proyectos para registrar comentarios durante la vida del proyecto.

Edición del proyecto

El comando "**Fichero / Abrir**" abre la ventana del Gestor de Programas para el proyecto seleccionado. Desde esta ventana se pueden gestionar todos los elementos que contiene el proyecto (programas, parámetros de aplicaciones, etc.). Otra manera de iniciar la edición es mediante un doble click en el nombre de un proyecto.

Historial de modificaciones

El sistema ISaGRAF almacena cualquier modificación relativa a un componente de proyecto en un fichero histórico. En dicho fichero, se identifica a cada modificación con un título, una fecha y una hora. El fichero histórico contiene las últimas **500** modificaciones. Existe un fichero histórico para cada proyecto. El historial de modificaciones del proyecto es el complemento de los ficheros "diarios" asociados a los programas del proyecto. El comando "**Proyecto / Histórico**" permite al usuario la visualización o impresión del historial de modificaciones relativo al proyecto seleccionado. El usuario puede elegir uno o más elementos de la lista principal y pulsar los siguientes botones:

Aceptar..... cierra esta ventana
Imprimir..... envía el contenido de la lista a la impresora
Ayuda ..... para visualizar ayuda relativa a esta ventana de diálogo
[borrado] Selección..... elimina (borra) de la lista las líneas seleccionadas
[borrado] Todo..... borra la lista completa
Buscar..... localiza un patrón en la lista

Se utiliza la ventana de introducción de datos localizada encima de el botón "**Buscar**" para introducir un patrón de búsqueda. Esta función es insensible al uso de mayúsculas o minúsculas. Cuando la búsqueda llega al final de la lista, continúa desde el principio de la lista hasta alcanzar la posición de inicio.

Impresión de un documento completo

El comando "**Proyecto / Imprimir**" permite al usuario la elaboración e impresión de un documento completo sobre el proyecto seleccionado. Este documento puede agrupar a cualquier componente (programa, variable, parámetros, etc.) del proyecto seleccionado. Para redactar un documento específico (no completo), el usuario sólo tiene que definir su índice de materias.

Protección por contraseña

El comando "**Proyecto / Establecer contraseña**" permite al usuario definir la protección por contraseña de las herramientas y datos del proyecto seleccionado. Para mayor información sobre los niveles de acceso y la protección de datos, véase la sección titulada "**Protección con contraseña**" al final de la primera parte de este manual. Las claves de acceso están relacionadas con el proyecto seleccionado. No influyen sobre otros proyectos y librerías ISaGRAF.

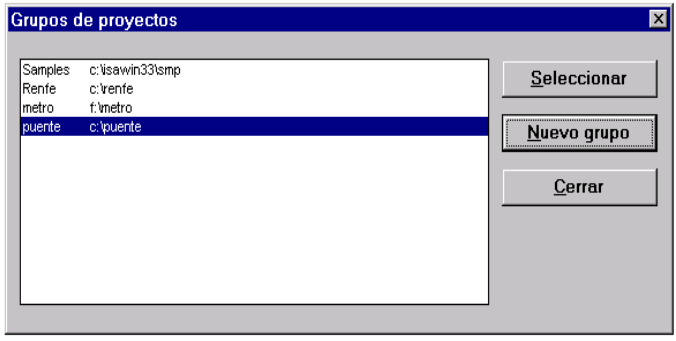
A.2.2 Trabajar con varios grupos de proyectos

Cada proyecto ISaGRAF corresponde a un directorio del disco en el que se guardan todos los ficheros del proyecto. Un “Grupo de Proyectos” corresponde a una lista de directorios de proyecto que han sido agrupados bajo el mismo directorio raíz. A un grupo de proyectos se le identifica con un nombre. Por defecto, ISaGRAF crea dos grupos de proyectos:

"**Default**"	en "\ISAWIN\APL"	su área de trabajo
"**Samples**"	en "\ISAWIN\SMP"	aplicaciones de ejemplo desarrollados con el banco de trabajo de ISaGRAF

El nombre del grupo de proyectos se escribe en la barra de herramientas, junto al botón que se utiliza para seleccionar el grupo de proyecto:

Se puede ejecutar el comando "**Fichero / Seleccionar grupo de proyecto**" para seleccionar un grupo existente o crear uno nuevo. Se abre la siguiente ventana de diálogo:



Seleccionar un grupo en la lista y pulsar **"Seleccionar** " para activarlo en la lista del gestor de proyectos. También se puede hacer doble click en su nombre para seleccionarlo. Utilizar el comando "**Nuevo grupo**" para crear un grupo nuevo. Este comando se puede utilizar, tanto para asignar un nombre de grupo a un directorio existente, como para crear un grupo nuevo con un nuevo directorio.

No se puede seleccionar o crear ningún grupo cuando hay otras ventanas de ISaGRAF abiertas (gestor de programas, editores...).

A.2.3 Opciones

Se utilizan los comandos del menú "**Opciones**" para visualizar u ocultar la barra de herramientas, seleccionar la fuente de caracteres para el texto y establecer el modo "auto cierre" del Gestor de Proyectos. La fuente de caracteres seleccionada es la que se utiliza para visualizar el descriptor de proyecto y la que utilizan todos los editores de texto ISaGRAF.

Si se anula la opción de "**Mantener el gestor de proyectos abierto**", la ventana del Gestor de Proyectos se cerrará automáticamente cuando se entre en un proyecto.

A.2.4 Herramientas

Los comandos del menú "**Herramientas**" se utilizan para ejecutar otras aplicaciones de ISaGRAF. El comando "**Herramientas / Archivo / Proyectos**" ejecuta el gestor de proyectos de ISaGRAF para guardar o recuperar proyectos. El comando "**Herramientas / Archivo / Datos comunes** " se utiliza para guardar o recuperar ficheros utilizados por todos los proyectos (tales como palabras comunes definidas).
El comando "**Herramientas / Librerías**" ejecuta el gestor de librerías de ISaGRAF en una ventana separada.

El comando "**Herramientas / Importar programa IL** " se puede utilizar para importar un proyecto descrito como un único programa IL en un fichero de texto, según el formato de intercambio de archivo abierto PLC.

A.3 Gestión de programas

La ventana del Gestor de Programas muestra los programas (también denominados módulos o unidades de programación) de la aplicación y agrupa en menús los comandos disponibles para la creación de la arquitectura del proyecto y la ejecución de editores, el compilador y el depurador (*debugger*). Esta ventana es el centro del banco de trabajo durante el desarrollo de una aplicación. La ventana del Gestor de Programas se abre al ejecutar el comando “Abrir” desde la ventana del Gestor de Proyectos.

A.3.1 Componentes de un proyecto

Los componentes de un proyecto se llaman **programas**. Un programa es una entidad lógica que describe una parte del control de la ejecución. Las variables globales (tales como las variables de E/S) pueden utilizarse por cualquier programa de la aplicación. Las variables locales sólo se pueden utilizar en un único programa. Los programas se organizan en un **árbol jerárquico**, dividido en diferentes **secciones lógicas**. La ventana muestra los programas y los enlaces entre ellos. Los programas de " **Nivel Superior** " aparecen en la parte izquierda del árbol jerárquico.

Programas de nivel superior

Los programas de nivel superior aparecen en el lado izquierdo del árbol jerárquico. Los programas de nivel superior de las tres primeras secciones están siempre activos y se ejecutan en el siguiente orden durante el ciclo de ejecución (*scan*):

- (Leer entradas)
- Ejecutar los programa de nivel superior de la sección **COMIENZO**
- Ejecutar los programa de nivel superior de la sección **SECUENCIAL**
- Ejecutar los programa de nivel superior de la sección **FIN**
- (Refrescar salidas)

Los programas de las secciones "**Comienzo**" y "**Fin**" describen las operaciones cíclicas. No dependen del tiempo. Los programas de la sección "**Secuencial**" describen las operaciones secuenciales, donde la variable de tiempo aparece explícitamente para diferenciar las operaciones básicas. Los programas principales de la sección "**Comienzo**" se ejecutan sistemáticamente al principio de cada ciclo de ejecución. Los programas principales de la sección "**Fin**" se ejecutan sistemáticamente al final de cada ciclo de ejecución. Los programas principales de la sección "**Secuencial**" se ejecutan en base a las normas **SFC** o **FC** y deben estar escritos en el lenguaje **SFC** o **FC**. Los programas de las secciones cíclicas no pueden ser descritos con el lenguaje **SFC** o **FC**. Cualquier programa de cualquier sección puede poseer uno o más **subprogramas**.

Funciones y bloques de función

Los programas de la sección "**Funciones**" pueden ser invocados por cualquier programa perteneciente a cualquier sección del proyecto. Una función es un algoritmo que procesa un valor de salida a partir de varios valores de entrada. Un algoritmo de función sólo trabaja con variables intermedias volátiles, que se eliminan entre una llamada y otra. De ello se desprende que una función jamás debería invocar a un bloque de función. Los programas de la sección "**Funciones**" no puede ser descrita con el lenguaje **SFC** o **FC**.
A diferencia de las funciones, los "**Bloques de función**" asocian un algoritmo que procesa valores de entrada con datos estáticos ocultos, que son copiados (instanciados) por el sistema en cada uso diferente del bloque de función. Los programas de la sección "**Bloques de función**" pueden ser invocados por cualquier programa de cualquier sección del proyecto. No se pueden programar en el lenguaje **SFC** o **FC**.

Subprogramas

Los subprogramas son funciones dedicadas a un programa padre (SFC, FC u otro). Un subprograma puede ser ejecutado (invocado) únicamente por su programa padre. Cada programa de cada sección puede tener uno o más subprogramas. Se puede utilizar cualquier lenguaje, con la excepción de **SFC** y **FC**, para describir un subprograma.

Programas SFC hijo

Un **programa SFC hijo** es un programa paralelo que puede ser ejecutado o finalizado por el programa padre correspondiente. Tanto el programa padre como el programa hijo tienen que ser descritos con el lenguaje **SFC**.
Cuando un programa padre inicia un programa **SFC** hijo, introduce una **marca SFC** en cada paso inicial del programa hijo. Cuando un programa padre finaliza un programa **SFC** hijo, elimina todas las marcas que pudieran existir en los pasos del hijo.

Cualquier programa **FC** de la sección secuencial puede controlar otros subprogramas **FC**. Un programa **FC** padre se bloquea (espera) durante la ejecución de un subprograma FC. No es posible hacer operaciones simultáneas en el programa FC padre y uno de sus subprogramas FC hijo.

Enlaces entre programas y subprogramas:

Los subprogramas y los programas hijo están vinculados a sus programas padre mediante una línea en el árbol jerárquico. El enlace entre un programa **SFC** y un programa **SFC** hijo termina en una flecha. Obsérvese que este tipo de enlace representa operaciones **paralelas**.

Lenguajes de programación

Cada programa se describe en un único **lenguaje**. Este lenguaje, que se selecciona en el momento de crear el programa, no puede cambiarse posteriormente. Sin embargo, los diagramas **FBD** pueden incluir partes en **LD** y los diagramas **LD** pueden contener llamadas a bloques de función. Se dispone de los siguientes lenguajes gráficos: **SFC** (Diagrama de Funciones Secuenciales), **FC**

(Diagrama de flujo), **FBD** (Diagrama de Bloques de función) y **LD** (Diagrama de Escalera (Contactos)). Se dispone además de los siguientes lenguajes literales: **ST** (Texto Estructurado) e **IL** (Lista de Instrucciones). Los lenguajes **SFC** y **FC** están reservados para los programas principales e hijo de la sección secuencial. El lenguaje de cada programa se muestra en forma de icono situado al lado del nombre del programa, dentro de la ventana del Gestor de Programas. A continuación se reflejan los iconos utilizados para representar los diversos lenguajes:


SFC ... Diagrama de Funciones Secuenciales

FC ..... Diagrama de flujo

FBD ... Diagrama de Bloques de función

LD..... Diagrama de Escalera (Contactos) (introducido con el editor *Quick LD*)

ST..... Texto Estructurado

IL ..... Lista de Instrucciones

A.3.2 Trabajando con programas

El menú "**Fichero**" agrupa a todos los comandos que se necesitan para crear, actualizar o modificar programas. Así mismo, ejecuta los editores apropiados para poder introducir el contenido de las aplicaciones.

Creando un programa nuevo

La función "**Nuevo**" del menú "**Fichero**" permite la creación de programas de nivel superior, programas hijo o subprogramas en cada sección de programa. El primer dato que debe ser introducido es el nombre del programa nuevo, conforme a las siguientes normas de nombrado:

- la longitud máxima de un nombre es de **8** caracteres
- el primer carácter tiene que ser una **letra**
- los restantes caracteres tienen que ser **letras**, **dígitos** o el carácter de subrayado " _ "
- los nombres de programas son insensibles al uso de mayúsculas o minúsculas

Luego, seleccionar el lenguaje de edición elegido para describir el programa nuevo:

SFC..... Diagrama de Funciones Secuenciales
FC ..... Diagrama de flujo
FBD..... Diagrama de Bloques de función (puede incluir partes en LD)
LD ..... Diagrama de Escalera (Contactos) (introducido con el editor *Quick LD*)
ST ..... Texto Estructurado
IL..... Lista de Instrucciones

Por último, seleccionar un modo de ejecución para el programa :

Comienzo.....nivel superior de la sección "Comienzo"
Secuencial .....nivel superior de la sección "Secuencial"
Fin.....nivel superior de la sección "Fin"
Función .....en la sección "Funciones"

Bloque de función.....en la sección "Bloques de función"
Hijo de.. .....hijo SFC o FC o subprograma de un programa existente

Al elegir una de las primeras cinco selecciones, el programa se sitúa en el nivel superior de una sección **Comienzo**, **Fin**, **Secuencial**, **Funciones** o **Bloques de función**. Le selección del último indica que el programa nuevo es un programa **SFC** hijo o un subprograma **FC** o un subprograma.. Recordar que se tienen que describir los programas secuenciales de nivel superior con el lenguaje **SFC** o **FC**, y que este lenguaje no se puede utilizar para programa cíclicos y sus correspondientes subprogramas.

Introducción de comentarios para cada programa

ISaGRAF permite al usuario adosar un texto descriptivo a cada programa del proyecto. Este comentario aparece en un tipo de letra más pequeño al lado del nombre del programa. Utilizar el comando "**Fichero / Texto de comentario del programa**" para introducir o modificar el comentario que está vinculado al programa seleccionado.

Edición del contenido de un programa

Este comando permite modificar el contenido de un programa. El editor que se utiliza para introducir un programa depende del lenguaje que se haya elegido para ese programa. La edición de programas de lleva a cabo en ventanas individuales, lo que posibilita la edición de más de un programa por medio de ventanas paralelas. Al pulsar la tecla **INTRO**, el usuario puede proceder a editar el programa seleccionado. Para entrar en modo edición, también puede hacer doble clic con el ratón sobre el nombre del programa.

Edición del fichero "diario"

Se asocia un **fichero diario** a cada programa. Este es un fichero de texto que contiene todos los apuntes relativos a las modificaciones realizadas durante la vida del programa. El fichero diario puede ser editado, modificado libremente o impreso en cualquier momento. Al salir del editor que se ha utilizado para modificar el código fuente de un programa, se abre automáticamente una ventana que permite la introducción de apuntes en la lista diaria. Estos apuntes, junto con la fecha y hora correcta, se incorporan posteriormente en el fichero diario.

El diccionario de variables

El comando "**Fichero / Diccionario**" ejecuta el editor del diccionario, en el que se registran todas las variables del proyecto. Las variables pueden ser globales (conocidas por cualquier programa del proyecto) o locales para el programa seleccionado. También se puede utilizar el editor del diccionario para registrar **palabras definidas**, que son alias semánticos que se emplean para sustituir un nombre o expresión en el código fuente del programa.

Parámetros de funciones, subprogramas o bloques de función

El comando "**Fichero / Parámetros**" permite al usuario definir los parámetros de llamada y retorno del subprograma, la función o el bloque de función seleccionado. Este comando no tiene efecto si se selecciona un programa principal de las

secciones “**Comienzo**” o “**Fin**”, o un programa SFC, desde la ventana del Gestor de Programas.
Los subprogramas, las funciones y los bloques de función pueden tener hasta **32** parámetros (entrada y salida). Una función o un subprograma siempre tiene un único parámetro de retorno, que debe tener el mismo nombre que la función para poder cumplir con las normas de escritura del lenguaje ST.

La lista que aparece en la zona superior izquierda de la ventana muestra los parámetros, dispuestos en el orden del modelo de invocación: los parámetros de llamada en primer lugar y los parámetros de retorno en el último lugar. La parte inferior de la ventana contiene una descripción detallada del parámetro que en ese momento está seleccionado de la lista. Se puede utilizar cualquiera de los tipos de datos ISaGRAF para un parámetro. Los parámetros de retorno tiene que estar situados después de los parámetros de llamada en la lista. La denominación de parámetros debe cumplir con las siguientes normas:

- la longitud del nombre no puede superar los 16 caracteres
- el primer carácter tiene que ser una letra
- los restantes caracteres tiene que ser letras, dígitos o el carácter ‘_’
- los nombres no son sensibles al uso de mayúsculas o minúsculas

Se utiliza el comando "**Insertar**" para insertar un parámetro nuevo delante del parámetro seleccionado. Se utiliza el comando "**Borrar**" para borrar el parámetro seleccionado. El comando "**Ordenar**" vuelve a disponer (ordena) los parámetros de manera automática, para que los parámetros de retorno se sitúen al final de la lista.

Desplazamiento de un programa en el árbol jerárquico

El comando "**Renombrar / Mover**" del menú "**Fichero**" se emplea para cambiar el nombre de un programa o para desplazarlo a otra sección del árbol jerárquico. Sin embargo, no se puede cambiar el lenguaje de descripción de un programa ya existente. Al ejecutar este comando, se abre la misma ventana que se utiliza para la creación de programas y se completan todos los campos con los atributos del programa seleccionado. Se puede modificar el nombre de un programa y se puede seleccionar otra sección o programa padre para su introducción en el árbol jerárquico.
Se utiliza el comando "**Ordenar programas**" del menú "**Fichero**" para dar una orden explícita entre una lista de programas del mismo nivel y padre. Si el programa seleccionado se encuentra en el nivel superior, el comando se utiliza para ordenar los programas de nivel superior pertenecientes a la sección seleccionada. Si el programa seleccionado se encuentra en el nivel inferior, el comando sólo ordena los hijos SFC y los subprogramas que tengan el mismo padre que el seleccionado. Cuando se abra la ventana de diálogo "**Ordenar programas**", seleccionar el programa que se desea mover y pulsar los botones "**Arriba**" o "**Abajo**" para desplazarlo dentro de la lista.

Realización de copias de programas

Para hacer una copia de un programa, seleccionar el programa fuente de la lista de programas y ejecutar el comando "**Fichero / Copiar**". Al ejecutar este comando, se abre la misma ventana que se utiliza para la creación de programas y se completan todos los campos con los atributos del programa seleccionado. Introducir el nombre del programa destino y su ubicación en las secciones del árbol jerárquico. Si el

programa destino no existe, se crea en la posición especificada. Si ya existe el programa destino, queda sobrescrita. Junto con el programa, se copian todas las declaraciones y palabras definidas locales existentes. El lenguaje utilizado para la descripción del programa destino tiene que ser el mismo que se haya utilizado para el programa fuente. Pulsar el botón "**Aceptar**" para copiar el programa.
El comando "**Copiar a otro proyecto**" del menú "**Fichero**" copia el programa seleccionado a otro proyecto con el mismo nombre. Los programas SFC hijo y los subprogramas del programa seleccionado pueden ser copiados junto con el programa. No se puede utilizar los nombres del programa seleccionado y de sus hijos en el proyecto objeto. Este comando no permite la sobrescritura de programas. Todas las declaraciones y palabras definidas locales se copian junto con los programas.

Eliminación de programas

Para eliminar un programa, primero hay que seleccionarlo de la lista de programas y después ejecutar el comando "**Fichero / Borrar**". Los programas que poseen programas hijo o subprogramas no pueden ser eliminados. Para poder borrar un programa que tenga programas hijo o subprogramas, primero hay que eliminar a éstos. Todas las declaraciones y palabras locales definidas se borran junto con el programa.

Importación de funciones o bloques de función de la librería

El comando "**Herramientas / Importar de librería**" se utiliza para copiar una función o un bloque de función, escritos en un lenguaje IEC y descritos en la librería, a las secciones "**Funciones**" o "**Bloques de función**" del proyecto abierto. Las variables y palabras locales definidas que estén asociadas a la función importada se copian junto con ésta. Cuando una función ha sido importada correctamente de la librería, se puede colocar en otra sección o en otra posición del árbol jerárquico mediante la utilización del comando "**Fichero / Renombrar/Mover**". Para evitar que surjan conflictos entre nombres, se deberá renombrar la función o el bloque de función importado cuando se importe en un área del proyecto. Así mismo, hay que renombrar el parámetro de retorno en el caso de las funciones.

Exportación de funciones o bloques de función a la librería

El comando "**Herramientas / Exportar a librería**" se utiliza para enviar un programa de la sección "**Funciones**" o "**Bloques de función**" (en el proyecto abierto) a la librería apropiada. Las variables locales y palabras definidas asociadas a la función o bloque de función exportado se copian junto con el programa. Será necesario volver a compilar (verificar) la función o el bloque de función exportado con el Gestor de Librerías ISaGRAF, para asegurar que se pueda utilizar en un entorno de librería. Las funciones y los bloques de función pertenecientes a la librería no pueden utilizar variables globales.

A.3.3 Ejecución de las herramientas de generación de código

Los comandos del menú "**Ensamblar**" se utilizan para ejecutar el generador de código y para introducir las opciones y los datos adicionales que se emplean al producir el código de una aplicación. Para mayor información sobre estas herramientas, véase el apartado del presente documento titulado "**Cómo utilizar el generador de código**".

Generación del código de la aplicación

El comando "**Construir aplicación**" inicia la generación del código de proyecto. Las opciones de generación de código objeto deben estar establecidas correctamente antes de ejecutar este comando. Antes de generar el código objeto, se comprueban aquellos programas que todavía no están verificados para detectar posibles errores de sintaxis. ISaGRAF incluye un compilador incremental que no vuelve a compilar programas que ya han sido compilados.

Verificación del programa seleccionado

El comando "**Verificar**" permite al usuario verificar la sintaxis del programa que en ese momento está seleccionado en la lista. Cuando se verifica un programa, sin que se detecten errores, no se vuelve a verificar durante el proceso de generación de código hasta que cambien su contenido o sus palabras definidas o variables dependientes.

Simulación de una modificación

El comando "**Marcar**" simula la modificación de cada programa para que todos puedan volver a ser compilados en la siguiente generación de código.

Opciones operativas de aplicaciones en tiempo de proceso

Este comando abre una ventana de diálogo, en la que se introducen los principales parámetros de tiempo de proceso para la ejecución de la aplicación. Esto incluye la programación de la sincronización de ciclos, la gestión de errores de tiempo de proceso, el modo de arranque y la implementación *hardware* de las variables retenidas. Para mayor información acerca de este comando, véase el apartado titulado "**Cómo utilizar el generador de código**".

Opciones del compilador

Este comando se utiliza para establecer las opciones utilizadas por el Generador de Código ISaGRAF para producir y optimizar código objeto. Para mayor información acerca de este comando, véase el apartado titulado "**Cómo utilizar el generador de código**".

Definición de recursos

Un "**recurso**" consiste en datos definidos por el usuario (por ejemplo, un fichero) que deben quedar integrados en el código objeto para poder cargarse o ser transferido con él. Para mayor información sobre el formato del fichero de definición de recursos, véase el apartado titulado "**Cómo utilizar el generador de código**".

A.3.4 Otras herramientas ISaGRAF

El menú "**Proyecto** agrupa los comandos que se encargan de ejecutar las herramientas ISaGRAF para el proyecto seleccionado. Para más información acerca de estas herramientas, véanse los apartados correspondientes en este documento.

Conexión de variables de E/S

El comando "**Conexión E/S**" ejecuta el editor ISaGRAF de conexión de variables de E/S. Esta herramienta se utiliza para establecer la relación entre las variables de E/S que estén registradas en el diccionario del proyecto y el correspondiente *hardware* de E/S.

Ejecución del editor de referencias cruzadas

El comando "**Referencias cruzadas**" brinda al usuario la posibilidad de calcular, visualizar o imprimir las referencias cruzadas del proyecto. Las referencias cruzadas muestran al usuario todas las apariciones de cada variable en el código fuente de los programas, en el conjunto del proyecto. Esta función resulta muy útil para detectar un acceso a una variable o a cualquier recurso global, o para listar todas las apariciones de una variable global en el código fuente.

Introducción del descriptor de proyectos

El comando "**Descriptor de proyecto**" se utiliza para editar el descriptor textual del proyecto. Este documento identifica y diferencia al proyecto totalmente de todos los demás de la lista de proyectos. También se puede utilizar el descriptor de proyectos para registrar cualquier comentario que surja durante la vida del proyecto. El descriptor de proyectos puede visualizarse en la ventana del Gestor de Proyectos.

Impresión de un documento completo

El comando "**Imprimir documento de proyecto**" permite al usuario elaborar e imprimir un documento completo sobre el proyecto seleccionado. Este documento puede abarcar cualquier componente (programa, variable, parámetros, etc.) del proyecto seleccionado. Para elaborar un documento específico (no completo), el usuario no tiene más que definir su índice de materias.

Historial de modificaciones

Este comando abre una ventana de diálogo en el que se muestra el historial de modificaciones del proyecto. Para mayor información acerca de este comando, véase el apartado titulado "**Gestión de proyectos**".

A.3.5 Añadir comandos al menú de Herramientas

ISaGRAF proporciona el método para insertar comandos en el menú de **"Herramientas"**. Los comandos definidos por el usuario que pueden ser añadidos se describen en el fichero de texto "**\ISAWIN\COM\ISA.MNU**". Se pueden añadir

hasta 10 comandos. Pueden insertarse comentarios en cualquier línea, comenzando con un carácter "**;**". Cada comando se describe en dos líneas de texto, de acuerdo con la siguiente sintaxis:

M=menu_string
C=*command_line*

La cadena de menú es el texto que aparece en el menú "**Herramientas**". La línea de mando es cualquier ejecutable de MS-DOS o Windows, y puede completarse con argumentos. En la línea de mando, se pueden utilizar caracteres **"%A"** para reemplazar el nombre del proyecto abierto, y caracteres **"%P"** para reemplazar el nombre del programa seleccionado. En el siguiente ejemplo, se ejecuta la utilidad de Windows "Bloc de Notas" para editar el programa seleccionado (sólo para programas ST e IL):

M=Edit with Notepad

C=Notepad.exe \isawin\apl\%A\%P.lsf

A.3.6 Simulación y depuración de la aplicación

Se utilizan los comandos del menú "**Depuración**" para ejecutar el depurador gráfico de ISaGRAF, bien en modo simulado o bien en modo de conexión real.

Simulación

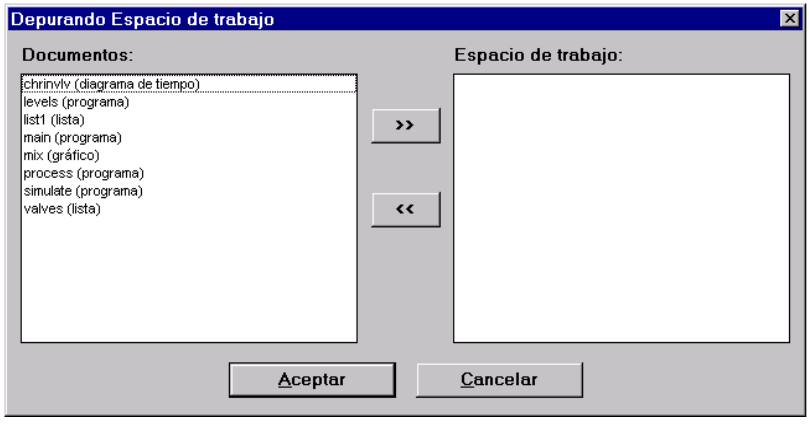
El comando "**Simular**" abre el depurador en modo simulado. En este modo de funcionamiento se abre otra ventana, denominada el simulador. Este comando es muy útil para comprobar una aplicación cuando la máquina objeto no está disponible. Al iniciar el simulador se cierra la ventana del Gestor de Programas; se abre de nuevo en modo depuración una vez que ya están abiertas las ventanas tanto del depurador como del simulador. No se puede ejecutar el simulador si previamente no se ha generado el código objeto. Tampoco se puede ejecutar el simulador si hay ventanas de hijos (editores, generación de código, conexión E/S, etc.) abiertas. Se debe cerrar cada una de ellas antes de ejecutar este comando. También se puede acceder a este comando desde los menús de los editores ISaGRAF.

Depuración real

El comando "**Depurar**" abre la ventana principal del depurador y cierra la ventana del Gestor de Programas. La ventana del Gestor de Programas se vuelve a abrir en modo depuración una vez que se haya establecido la comunicación entre el depurador y la aplicación objeto. No se puede ejecutar el simulador si previamente no se ha generado el código objeto. Tampoco se puede ejecutar el depurador si hay ventanas de hijos (editores, generación de código, conexión E/S, etc.) abiertas. Se debe cerrar cada una de ellas antes de ejecutar este comando. También se puede acceder a este comando desde los menús de los editores ISaGRAF.

Preparación del espacio de depuración

El comando "**Depurar / Espacio de trabajo**" permite definir una lista de documentos para el espacio inicial. Dichos documentos pueden ser programas, gráficos SpotLight, listas de variables. Gráficos y listas de diagramas de tiempo de versiones de ISaGRAF previas se listan también con los documentos del proyecto. Los documentos definidos en el espacio de trabajo inicial se abren automáticamente cuando se lanza la monitorización en línea.



La ventana de diálogo muestra los documentos existentes en el proyecto en la izquierda , y los documentos seleccionados para el espacio de trabajo inicial a la derecha. Utilizar los botones ">>" y "<<" para mover documentos de una lista a otra. Cada proyecto tiene su propia lista de documentos para el espacio de trabajo inicial.

Configuración de enlaces

El comando "**Configuración de enlace**" abre la siguiente ventana de diálogo. Permite al usuario definir los parámetros del enlace que se utilizan para la comunicación entre el depurador, ubicado en el ordenador principal, y el sistema ISaGRAF objeto.

El "**Número de esclavo**", identifica al sistema o tarea objeto del entorno ISaGRAF. Puede ser cualquier número entre **1** y **255**. Véase la documentación facilitada por el proveedor para el número de esclavo del sistema objeto que se utilice.
El "**Puerto de comunicación**", identifica el canal *hardware* entre el banco de trabajo ISaGRAF y el objeto. Puede ser bien el nombre de un puerto serie o bien "**Ethernet**", que está reservado para la comunicación TCP/IP utilizando el "Winsock" versión 1.1.
El término "**Tiempo de espera (seg)**", es el tiempo permitido al sistema objeto para sus operaciones de comunicaciones entre el término de una interrogación del depurador y el inicio de su respuesta. Este tiempo se expresa como un número de **segundos**. El campo de "**Reintentos**", es el número de intentos automáticos ejecutados por el depurador para llevar a cabo una operación de comunicación, antes de detectar un error de comunicación.

- ***Configuración del enlace serie***

  Cuando se selecciona un puerto serie (COM1..4), se utiliza el botón "**Configurar**" para acceder a otros parámetros de comunicación del enlace serie.

  Se puede seleccionar la velocidad de transmisión en baudios, la paridad y el formato. Cuando se opta por la selección de "**hardware**" en relación a "**Control de flujo**", el banco de trabajo ISaGRAF controla las líneas CTS y DSR para habilitar el diálogo hardware (hardware handshaking) entre los intercambios.

- ***Configuración del enlace Ethernet***

  Cuando se selecciona "Ethernet" como el puerto de comunicación, se utiliza el botón "**Configurar**" para introducir la "Dirección Internet" y el número del "Puerto Internet" para la comunicación usando TCP/IP.

  Estos campos emplean los formatos estándares definidos para la interfaz 'Socket'. El banco de trabajo emplea la librería WINSOCK.DLL Versión 1.1 para las comunicaciones TCP/IP. Este fichero tiene que estar instalado correctamente en el disco duro del ordenador. "**1100**" es el número de puerto por defecto que se utiliza si no se especifica otro al ejecutar el objeto ISaGRAF.

A.4 Utilización del editor SFC

El lenguaje SFC se utiliza para describir las operaciones de un proceso secuencial. Utiliza una representación gráfica sencilla para exponer los diferentes pasos de un proceso, junto con condiciones que permiten la modificación de pasos activos. Se entra en un programa SFC por medio del editor gráfico SFC de ISaGRAF. SFC es el núcleo de la norma IEC 1131-3. Los restantes lenguajes suelen describir las acciones que tienen lugar dentro de los pasos y las condiciones lógicas de las transiciones. El editor gráfico SFC de ISaGRAF permite al usuario introducir programas SFC completos. Combina la capacidad de editar gráficos y texto, lo que permite la incorporación tanto del diagrama SFC como de las acciones y condiciones correspondientes.

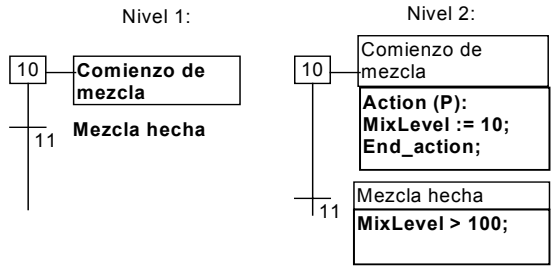
A.4.1 Aspectos principales del lenguaje SFC

Se utiliza el lenguaje SFC para representar procesos secuenciales. Divide el ciclo de proceso en un número de **pasos** (situaciones independientes) sucesivos y bien definidos, que se separan mediante **transiciones**. Para mayor información acerca del lenguaje SFC, véase el Manual de Referencia de Lenguajes ISaGRAF.

Los componentes SFC están unidos por **líneas orientadas**. La orientación por defecto de una línea es de **arriba** hacia **abajo**. A continuación de muestran los componentes gráficos básicos que se utilizan para elaborar un diagrama SFC:

 ..... Paso inicial
 ..... Paso
 ..... Transición
 ..... Salto a un paso
 ..... Macropaso
 ..... Paso inicial de Macropaso
 ..... Paso final de Macropaso

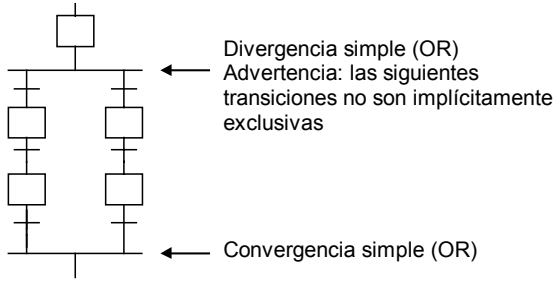
Generalmente, la programación SFC se divide en dos niveles diferentes: el **Nivel 1** muestra el diagrama gráfico, los números de referencia de los pasos y las transiciones, y los comentarios asociados a los pasos y las transiciones. El **Nivel 2** es la programación **ST** o **IL** de las acciones contenidas en cada paso, o las condiciones que afectan a las transiciones. Las acciones o condiciones pueden hacer referencia a **subprogramas** escritos en otros lenguajes (**FBD**, **LD**, **ST** o **IL**). A continuación aparece un ejemplo de programación de niveles 1 y 2 :



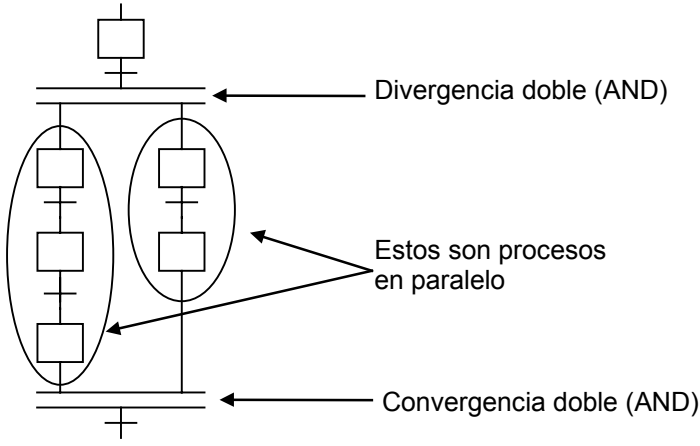
La programación de Nivel 2 de un paso se introduce con un editor de textos. Puede incluir bloques de acciones programados en los lenguajes ST o IL. La programación de Nivel 2 de una transición puede realizarse bien en IL o bien en ST, o con el editor *Quick LD*.

Divergencias y convergencias

Se utilizan las divergencias y las convergencias para representar **enlaces múltiples** entre pasos y transiciones. Las divergencias o convergencias simples representan diferentes posibilidades **inclusivas** entre las distintas subpartes del proceso.

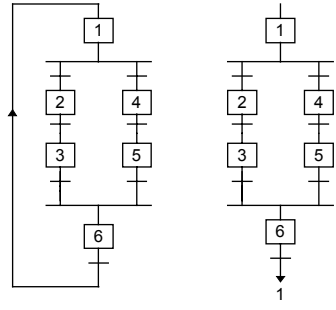


Las divergencias dobles representan procesos **paralelos**.



Salto a un paso

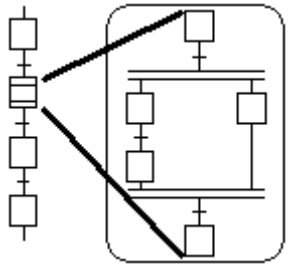
El editor SFC sólo le permite al usuario dibujar enlaces en dirección ascendente (**Arriba**) o descendente (**Abajo**). Se puede utilizar un **salto** a un paso para representar un enlace con una parte superior del diagrama. Los siguientes diagramas son equivalentes :



Los saltos a una transición están prohibidos, y deben representarse explícitamente como convergencias dobles (AND).

Macropasos

Un macropaso es una representación **única** de un grupo **autónomo** de pasos y transiciones. Un macropaso comienza con un **paso inicial** y finaliza con un **paso final**.



La representación detallada de un macropaso tiene que estar descrita con el mismo lenguaje SFC. El símbolo de macropaso debe tener el mismo **número de referencia** que el paso inicial de la macro. La descripción del macropaso puede contener a su vez otro macropaso.

A.4.2 Introducción de una tabla SFC

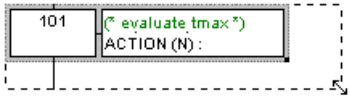
Para dibujar un diagrama SFC, el usuario sólo tiene que introducir los componentes significantes del diagrama. El editor SFC dibuja todas las líneas simples que unen dos elementos (horizontal o verticalmente) automáticamente. Para colocar un componente SFC en el diagrama el usuario tiene que llevar la selección al lugar apropiado y seleccionar el tipo de componente en la barra de herramientas del editor. El símbolo se inserta en la posición actual. También se pueden usar las siguientes secuencias de teclado.

F2:	Inserción de un paso inicial
F3:	Inserción de un paso
F4:	Inserción de una transición
F5:	Inserción de un salto a un paso
F6: F7:	Inserción de una divergencia o convergencia (OR) Adición de ramas
↑F6: ↑F7:	Inserción de una (AND) Adición de ramas
F8:	Inserción de macropasos
F9: ↑F9:	Insertar pasos inicial o final para el cuerpo de un macropaso

(El símbolo "↑" indica una combinación con la tecla de mayúsculas SHIFT)

La **rejilla de edición** muestra **celdas de matriz**. Una opción del editor le permite al usuario mostrar u ocultar la rejilla durante la introducción de datos en el diagrama. La rejilla es muy útil para el diseño inicial del diagrama SFC, o para seleccionar subpartes del diagrama. Utilizar el comando "**Opciones / Distribución**" para mostrar u ocultar la rejilla.

El editor gráfico SFC de ISaGRAF siempre muestra la posición actual en la matriz. La celda seleccionada se marca en gris. El pequeño cuadrado en la esquina inferior izquierda se puede utilizar para cambiar el tamaño de las celdas. La relación X/Y de las celdas se puede cambiar también de esta forma..



Creación de divergencias o convergencias

Las divergencias y las convergencias siempre se dibujan **de izquierda a derecha**. Para dibujar una divergencia o una convergencia, sus **ramas izquierda** tienen que situarse dentro del área del diagrama. Se determina el tipo de dibujo (simple o doble) seleccionando el botón correspondiente en la barra de herramientas.

Adición de ramas a divergencias

Las posiciones de **start** (inicio) y **stop** (fin) de cada **rama auxiliar** se sitúan sobre la línea de divergencia o convergencia utilizando estos botones en la barra de herramientas. La esquina izquierda de la divergencia o convergencia debe estar presente antes de insertar ramas nuevas. Las esquinas de la derecha no se pueden colocar si no se ha añadido previamente la principal esquina izquierda.

Inserción de macropasos

Se utiliza este botón para insertar un macropaso en el diagrama principal. Se tiene que introducir el cuerpo del macropaso en otra parte del mismo programa SFC.

Cuerpo de un macropaso

Los macropasos tienen que ser descritos en el mismo programa SFC que el diagrama principal. Un macropaso empieza con un **paso inicial** y finaliza con un **paso final**. El subdiagrama que se describe como la implementación de la macro tiene que ser **autocontenido**. El paso inicial del macropaso debe tener la misma **referencia** que el símbolo de macropaso de la rama principal.

A.4.3 Trabajar con diagramas SFC existentes

Se puede utilizar tanto el ratón como las flechas del teclado para seleccionar un área rectangular en el diagrama. Todo el área seleccionada se marca en gris. Se pueden utilizar los comandos del menú de edición.

Comandos Cortar / Copiar / Borrar

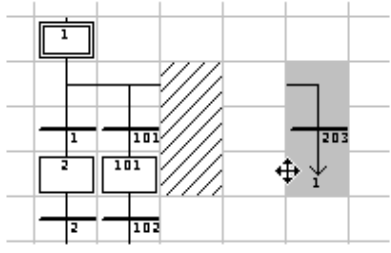
Los siguientes comandos están disponibles en el menú "**Edición**" cuando está seleccionada el botón de "**flecha**" en la barra de herramientas del editor:
Cortar ..... Mover el rectángulo seleccionado de la pantalla al portapapeles SFC
Copiar..... Copiar el rectángulo seleccionado de la pantalla al portapapeles SFC
Borrar ..... Borrar el rectángulo seleccionado

Pegar ..... Inserta el contenido del portapapeles SFC en la posición actual

El comando "**Edición / Pegar**" copia el contenido del portapapeles SFC a la pantalla. Estos comandos funcionan tanto en los diagramas SFC como en la programación de Nivel 2 de pasos y transiciones. También es posible copiar un diagrama de un programa y pegarlo en otro programa SFC. Los elementos se insertan antes de la posición actualmente seleccionada.

Mover elementos

Cuando se seleccionan elementos SFC de un diagrama SFC, se pueden mover a otra posición del diagrama arrastrando la selección con el ratón. Cuando se arrastra la selección, la posición inicial de los elementos seleccionados se marca con un rayado.



El área a la que se mueven los objetos debe de estar vacía. No es posible insertar mientras se mueve símbolos SFC.

Renumeración de pasos y transiciones

Cada paso o transición está identificado por un número lógico en el diagrama SFC. El comando "**Edición / Renumerar**" permite al usuario el establecimiento automático de números de referencia numéricamente secuenciales para cualquiera de los pasos o las transiciones del programa SFC que en ese momento se está editando. Cuando se cambia un número de paso, todos los saltos a ese paso se actualizan automáticamente con el nuevo número de referencia. Lo mismo ocurre con los macropasos y los pasos iniciales.

Acceso directo a un paso o una transición

El comando "**Edición / Ir a**" le permite al usuario acceder a un paso o una transición existente. Se adapta automáticamente la posición de *scrolling* para que el paso o la transición esté visible.

Encontrar y sustituir textos

Se puede utilizar el comando "**Edición / Buscar Reemplazar**" para encontrar o sustituir cadenas de texto en el conjunto del programa (en todos los pasos y transiciones). La ventana de diálogo se utiliza para introducir un texto determinado y abrir directamente la sección de la programación de nivel 2 donde aparece el texto.

A.4.4 Introducción de la programación de Nivel 2

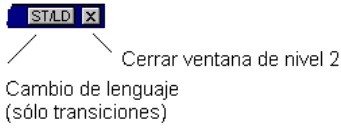
Para introducir el texto de Nivel 2, el usuario debe hacer doble click sobre el rectángulo de Nivel 2 de un paso o una transición. La programación de nivel 2 se muestra a la derecha de la ventana SFC. La línea de separación entre el diagrama SFC y la programación de nivel 2 se puede mover libremente.

Se pueden abrir simultáneamente dos áreas de nivel 2 al mismo tiempo. Los siguientes comandos están disponibles desde el teclado, el ratón o el menú de "Edición".

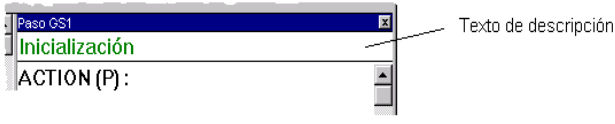
	Teclado	*Ratón*	*Menú "Edición"*
Abre la última ventana por defecto	Intro	Doble Click	Edita el nivel 2
Lo abre en otra ventana	Ctrl+Intro	Ctrl + 2 Click	Edita el nivel 2 en otra ventana

Cuando se tienen abiertas dos ventanas de nivel 2, la separación entre ellas se puede mover libremente. El botón de la derecha de la barra de título de nivel 2 se utiliza para cerrar una ventana de nivel 2.

El lenguaje por defecto para la programación del nivel 2 es ST (Texto Estructurado). En el caso de las transiciones, la programación del nivel 2 se puede hacer también con el editor **Quick LD**. Usar el botón "**ST/LD**" en la barra de título del nivel 2 para cambiar el lenguaje activo. Este comando sólo es valido si la ventana de programación de nivel 2 se encuentra vacía.



Una única línea aparece en la parte superior de la ventana de nivel 2. Se utiliza para introducir un pequeño texto de descripción. Este texto se ve como un comentario IEC al dibujar los símbolos SFC. Es muy útil al utilizarse por otros comandos tales como "Ir a..." y también en la impresión SFC para documentar pasos y transiciones SFC.



El comando "**Opciones / Refresco**" se puede utilizar en cualquier instante cuando hay abiertas ventanas de nivel 2 para refrescar el diagrama SFC principal

Mientras se trabaja con una ventana de programación de Nivel 2, el usuario puede acceder a los comandos del menú "**Edición**", en la ventana principal, para trabajar en el texto o el diagrama activos de Nivel 2.

Inserción de un nombre de variable

Al programar en lenguaje de texto, pulsar este botón para seleccionar una variable registrada en el diccionario del proyecto e insertar su nombre en la posición actual del cursor. Cuando se programa en *Quick LD*, pulsar este botón para seleccionar la variable que se va a asociar al contacto o parámetro de bloque de E/S seleccionado.

Inserción de un bloque de acción pulsante en un paso

Al programar el Nivel 2 de un paso, pulsar este botón para insertar la plantilla de un bloque de acciones pulsante en la posición actual del cursor. A continuación se muestra el formato de un bloque de acciones de alteración:

```
Action (P) :
    sentencia ST;
    ...
End_Action;
```

Las acciones pulsantes son instrucciones que sólo se ejecutan una vez, cuando el paso se vuelve activo. Para mayor información sobre la programación SFC, véase el documento de referencia de lenguajes ISaGRAF.

Inserción en un paso de un bloque de acciones no almacenado

Cuando se está programando el Nivel 2 de un paso, pulsar este botón para insertar la plantilla de un bloque de acciones no almacenado en la posición actual del cursor. A continuación se muestra el formato de un bloque de acciones no almacenado:

```
Action (N) :
    sentencia ST;
    ...
End_Action;
```

Las acciones no almacenadas son instrucciones que se ejecutan en cada ciclo de PLC, estando activo el paso. Para mayor información sobre la programación SFC, véase el documento de referencia de lenguajes ISaGRAF.

Nuevos calificadores de acciones P0 y P1

ISaGRAF soporta los nuevos calificadores de acciones **P0** y **P1**. Al realizar la programación de Nivel 2 de un paso, pulsar estos botones para insertar la plantilla

de un bloque de acciones P0 ó P1 en la posición actual del cursor. A continuación se muestra el formato de estos bloques:

Action (P0) :	**Action (P1) :**
sentencia ST;	sentencia ST;
...	...
End_Action;	**End_Action;**

Las acciones P1 son instrucciones que sólo se ejecutan una vez cuando el paso se vuelve activo (como sucede con las acciones pulsantes). Las acciones P0 son instrucciones que sólo se ejecutan una vez cuando el paso se vuelve inactivo. Para mayor información sobre la programación SFC, véase el documento de referencia de lenguajes ISaGRAF.

Acciones booleanas

Se dispone de otras semánticas de texto para actuar directamente sobre una variable booleana, de acuerdo con la actividad del paso. Las acciones de este tipo consisten en asociar una **señal de actividad del paso** a una variable booleana interna o de salida. A continuación se muestra la sintaxis de las acciones booleanas básicas:

<variable_booleana> (N);	asigna la señal de actividad del paso a la variable
<variable_booleana>;	mismo efecto (el atributo N es opcional)
/ <variable_booleana>;	asigna la negación de la señal de actividad del paso a la variable

Se dispone de otras funciones para establecer o restablecer una variable booleana cuando el paso se activa. Esta es la sintaxis de las acciones booleanas de establecer o restablecer:

<variable_booleana> (S);	fija la variable en VERDADERO cuando la señal de actividad del paso se convierte en VERDADERO.
<variable_booleana> (R);	restablece la variable en FALSO cuando la señal de actividad del paso se convierte en VERDADERO

Acciones SFC

Se dispone de otras semánticas de texto para controlar la ejecución de un programa SFC hijo. Una acción SFC es una secuencia SFC hijo que se inicia o se finaliza según el estado de la señal de actividad del paso. Una acción SFC puede tener un calificador **N** (No almacenado), **S** (Establecer) o **R** (Restablecer). A continuación se muestra la sintaxis de las acciones SFC básicas:

<programa_hijo> (N);	inicia la secuencia hijo cuando se activa el paso, y finaliza la secuencia hijo cuando el paso se vuelve inactivo
<programa_hijo>;	mismo efecto que la sintaxis precedente (el atributo N es opcional)

<programa_hijo> (S); ..... inicia la secuencia hijo cuando el paso se activa; no se hace nada cuando el paso de vuelve inactivo
<programa_hijo> (R); ..... finaliza la secuencia hijo cuando el paso se activa; no se hace nada cuando el paso de vuelve inactivo

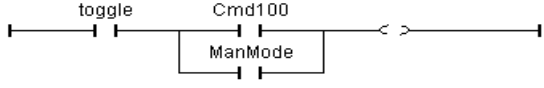
La secuencia SFC que se especifique como una acción tiene que ser un **programa SFC hijo** perteneciente al programa que se esté editando en ese momento, creado con el Gestor de Programas SFC.

Transiciones escritas en ST

El Nivel 2 de una transición es una expresión booleana. Para programarlo en el lenguaje ST, sólo hay que introducir la condición booleana de acuerdo con la sintaxis ST. Opcionalmente, puede añadirse un punto y coma ';' al final de la expresión.

Transiciones escritas en Quick Ladder

El editor *Quick LD* está disponible para la programación de la condición de Nivel 2 de una transición. En este caso, el diagrama se compone de un único escalón (línea), con una sola bobina (salida) en representación de la transición. No se repite el nombre de la transición con el símbolo de la bobina (salida). A continuación se muestra un ejemplo de una condición de transición programada en *Quick LD*:



Cuando se programa en *Quick LD*, utilizar las flechas del teclado para desplazar la selección dentro de la rejilla lógica de programación y después utilizar los siguientes aceleradores de teclado para insertar símbolos:

F2:..... insertar un contacto después del símbolo seleccionado / iniciar el escalón (línea)
F3:..... insertar un contacto antes del símbolo seleccionado
F4:..... insertar un contacto en paralelo con el símbolo seleccionado
F6:..... insertar un bloque después del símbolo seleccionado
F7:..... insertar un bloque antes del símbolo seleccionado
F8:..... insertar un bloque en paralelo con el símbolo seleccionado

También se puede hacer click sobre la barra de botones de funciones, localizada en la parte inferior de la ventana de Nivel 2, en lugar de pulsar los botones de funciones.

Cuando está seleccionado un contacto o un parámetro de bloque de E/S, pulsar 'Intro' para seleccionar una variable o introducir un valor constante. Cuando está seleccionado un bloque de función, pulsar 'Intro' para seleccionar el tipo de bloque de función. También se puede hacer doble click sobre un símbolo para lograr el mismo efecto.

Cuando está seleccionado un contacto, pulsar la barra espaciadora para cambiar el tipo de contacto (directo, negado o con detección de impulsos). Para más información sobre las capacidades de *Quick LD*, véase el apartado titulado "**Utilización del editor *Quick LD***".

A.4.5 Utilización de la galería SFC

El editor SFC de ISaGRAF maneja una galería SFC: es una colección de estructuras SFC que se pueden insertar en cualquier diagrama SFC. Los elementos de la galería SFC se pueden incluir opcionalmente la programación de nivel 2 de pasos y transiciones. Utilizar los siguientes comandos del menú "**Herramientas**":

Copiar a galería SFC copia los elementos seleccionados a la galería SFC
Pegar de galería SFC pega un elemento de la galería SFC a la posición actual

Cuando se copia a la galería SFC (i.e. se crea un nuevo elemento de la galería SFC), se puede incluir adicionalmente la programación de nivel 2 de los símbolos seleccionados.

A.5 Utilización del editor Diagrama de Flujo

El editor gráfico de Diagrama de flujo de ISaGRAF permite al usuario introducir programas FC (Diagrama de flujo) completos con acciones y decisiones programadas en lenguaje ST, IL o Quick LD. El Diagrama de flujo es un diagrama de decisión que también se puede utilizar para describir operaciones secuenciales al permitir algunas características avanzadas como no bloquear los saltos hacia atrás.

A.5.1 Aspectos básicos del lenguaje FC

El lenguaje **Diagrama de flujo** (**FC**) se utiliza para describir **operaciones secuenciales**. Un diagrama de flujo está formado por **Acciones** y **Decisiones**. Entre las acciones y decisiones hay **enlaces orientados** que representan el flujo de datos. A continuación están los componentes gráficos del lenguaje Diagrama de flujo:

Inicio del diagrama FC: Un símbolo de "Inicio" debe aparecer al principio del programa Diagrama de flujo. Es único y no se puede omitir. Representa el estado inicial del diagrama cuando se activa el diagrama.

Fin del diagrama FC Un símbolo de "Fin" debe aparecer al final del programa en diagrama de flujo. Es único y no se puede omitir. Es posible que no haya ninguna conexión dibujada al símbolo de fin (diagrama de bucle infinito), pero el símbolo de "Fin" debe aparecer en cualquier caso dibujado en la parte inferior del diagrama. Representa el estado final del diagrama cuando su ejecución se ha completado.

Enlaces de flujo FC: Un enlace de flujo es una línea que representa un flujo entre dos puntos del diagrama. Un enlace siempre termina en una flecha. Dos enlaces no pueden estar conectados al mismo punto fuente de conexión.

Acciones FC: Un símbolo de acción FC representa las acciones a realizar. Una acción está identificada por un número y un nombre. Dos objetos diferentes del mismo diagrama no pueden tener el mismo nombre o número lógico. El lenguaje de programación para una acción puede ser ST, LD o IL. Una acción está siempre conectada con enlaces, uno llegando a ella, otro saliendo de ella.

Condiciones FC: Una Condición representa una decisión booleana. Una condición está identificada por un número y un nombre. Según la evaluación de la correspondiente expresión en ST, LD o IL, el flujo se dirige por el camino de "SÍ" o "NO". Cuando se programa en texto ST, la expresión puede estar seguida

opcionalmente por un punto y coma. Cuando se programa en LD, la única bobina (salida) representa el valor de la condición

Subprograma FC: El sistema permite la descripción de la estructura vertical de los programas FC. Los programas FC están organizados según un árbol jerárquico. Cada programa FC puede llamar a otros programas FC. Este programa se llama programa hijo del programa FC que lo llama. Los programas FC que llaman a subprogramas FC se llaman programas padre. Los programas FC se unen juntos a un árbol jerárquico principal usando una relación padre-hijo. Un símbolo de subprograma en un diagrama de flujo representa una llamada a un subprograma diagrama de flujo. La ejecución del programa padre está suspendida hasta que el subprograma termine de ejecutarse.

Acción específica de E7S FC Un símbolo de una acción específica de E/S FC representa las acciones a ser realizadas. Como otras acciones, una acción de E/S concreta se identifica por un número y un nombre. La misma semántica se utiliza en las acciones standard y especificaciones de E/S. El objetivo de una acción específica de E/S es sólo hacer el diagrama más legible y centrarse en las partes no portables del diagrama. Utilizar las acciones especificas es una característica opcional. Los bloques concretos de E/S tienen exactamente el mismo comportamiento que las acciones normales.

Conectores FC: Los conectores se utilizan para representar un enlace entre dos puntos del diagrama sin dibujarlo. Un conector se representa por un círculo y se conecta a la fuente de flujo. El dibujo del conector se completa, en el lado apropiado (dependiendo de la dirección del flujo de datos), por la identificación del punto de destino (generalmente el nombre del símbolo objeto). Un conector siempre se dirige a un símbolo definido del diagrama de flujo. El símbolo de destino se identifica por un número lógico.

Comentarios FC : Un bloque de comentario contiene texto que no tiene sentido para la semántica del diagrama. Se puede insertar en cualquier lugar del espacio no usado de la ventana documento del diagrama de flujo y se utiliza para documentar el programa.

A.5.2 Introducción de un Diagrama de flujo

Para introducir un diagrama de flujo se tienen que colocar los elementos (acciones, decisiones, conectores…) en el área gráfica y dibujar enlaces de datos entre ellos.

Inserción de Objetos

Para insertar un objeto en un diagrama, seleccionar el botón correspondiente en la barra de herramientas y hacer click en el área gráfica donde se quiere insertar éste. Se puede colocar el elemento sobre una zona vacía o insertarlo en un flujo haciendo click en el enlace de flujo. La inserción de un enlace sólo está permitida de arriba a abajo para enlaces verticales.

Se pueden insertar los siguientes elementos básicos:

Acción programada en ST, IL o Quick LD
Acción específica de E/S (destaca una acción concreta no portable)
Condición (decisión) programada en ST, IL o Quick LD
conector
Llamada a un subprograma FC
Comentario (Texto de descripción)

El editor de Diagrama de flujo de ISaGRAF también propone una lista de estructuras de diagramas de flujo clásicas. Dichas estructuras sólo pueden insertarse en un flujo ya existente. No se pueden colocar en una área vacía:

If / Then / Else (selección binaria)
Repeat until (espera una condición)
While (Hazlo mientras la condición sea verdadera)

Selección de Objetos

Se necesita la selección de objetos gráficos para la mayor parte de los comandos de edición. El editor gráfico de ISaGRAF permite la selección de uno o más objetos existentes en el área del diagrama. Para seleccionar objetos, la opción de "**selección**" (botón con una flecha) debe seleccionarse en la barra de herramientas del editor. Para seleccionar un objeto, el usuario sólo tiene que hacer click en su símbolo.

Para seleccionar una lista de objetos, arrastre el ratón por el diagrama para seleccionar un área rectangular. Todos los objetos gráficos de la selección se marcan como "**seleccionado**".

Un objeto seleccionado se dibuja en color azul oscuro con pequeños cuadraditos negros alrededor de su símbolo gráfico. Es posible también añadir o eliminar un objeto a una selección múltiple haciendo click en su símbolo con las teclas **Shift** o **Ctrl** presionadas.

Al hacer una nueva selección se elimina la selección de todos los objetos previamente seleccionados. Para eliminar la selección existente, simplemente hacer click con el ratón en un área vacía, fuera del rectángulo que rodea a los objetos seleccionados.

Para una sola selección es posible usar las flechas del teclado para desplazar la selección de un objeto a otro en el diagrama. Los enlaces de flujo se pueden también seleccionar.

Inserción de comentarios

Se pueden insertar comentarios en cualquier parte vacía del diagrama. Los comentarios no tienen influencia en la ejecución del programa. Permiten una mayor

legibilidad del diagrama. Para insertar un bloque de comentario seleccionar el botón correspondiente en la barra de herramientas y hacer click en el diagrama donde se quiera poner. Haga. Doble click en un comentario para introducir / cambiar su texto. No se necesita ninguna inserción adicional de caracteres, tales como "**(***" y "***)**" cuando se introduce el texto de un bloque de comentario. Un bloque de comentario puede cambiarse de tamaño arrastrando las esquinas de su frontera cuando se selecciona.

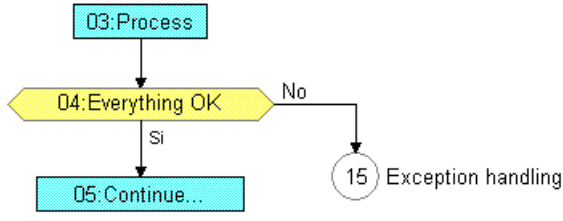
Dibujo enlaces de flujo

Seleccionar este botón de la barra de herramientas para dibujar un flujo entre dos elementos existentes. Un enlace debe dibujarse siempre en la dirección del flujo. Primero selecciona un punto de salida no conectada de un elemento FC, y se arrastra el ratón a l punto de destino para insertar el enlace. El punto de destino puede ser tanto la parte de arriba de un elemento FC no conectado o cualquier posición de un flujo existente. Los puntos de convergencia entre enlaces se muestran con pequeños círculos grises en el diagrama de flujo. Los puntos de convergencia pueden seleccionarse y moverse para arreglar el dibujo.

Utilización de conectores

El editor de Diagrama de flujo de ISaGRAF permite el uso de conectores gráficos, como sustituto de un enlace de flujo visible. Los conectores pueden ser muy útiles para evitar enlaces largos e incrementar la legibilidad del diagrama. Un conector no se puede utilizar para establecer un enlace con otro programa FC.

Un conector se coloca en el diagrama como otros objetos FC. Se representa por un círculo que contiene una referencia numérica de el elemento apuntado (el destino del enlace de flujo). El pequeño texto de descripción del elemento apuntado se muestra junto al círculo del conector.



Desplazamiento de objetos

Para mover objetos en el diagrama, se tienen que seleccionar y arrastrar el ratón para moverlos dentro del diagrama. Se puede mover bien un sólo elemento bien una selección múltiple. No se pueden superponer elementos cuando se mueven. El desplazamiento de elementos no se puede utilizar para conectarlos a un flujo existente.

Cuando se mueve un único elemento (acción, decisión...), el editor de diagrama de flujo ISaGRAF se mueve automáticamente con todos los elementos por debajo conectados a él. Esta característica no actúa en el caso de una selección múltiple.

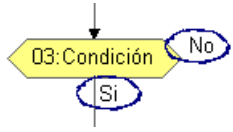
Cambiar el tamaño de los objetos

Cualquier elemento gráfico de un flujo, excepto los símbolos de "Inicio", y "Fin" , se pueden cambiar de tamaño libremente. Para cambiar el tamaño de un elemento primero hay que seleccionarlo. Después se arrastran con el ratón las esquinas dibujadas en su frontera para cambiar su tamaño.

Cuando un elemento se conecta a un flujo de datos, cambiar su tamaño horizontalmente actúa sobre su esquina izquierda y derecha, de modo que el elemento todavía está centrado en el enlace cuando se cambia su tamaño.

Conmutar las salidas de una decisión

Se pueden cambiar las posiciones de SÍ / NO en las salidas de una decisión. Para hacer esto simplemente hacer doble click en las marcas de "Sí" o "No" junto al símbolo de decisión.



A.5.3 Introducción de un Diagrama de flujo

Los comandos del menú "**Edición**" se utilizan para cambiar o completar un diagrama existente. La mayor parte de estos comandos actúan sobre elementos seleccionados actualmente en el diagrama.

Corregir un diagrama

La tecla de borrado DEL se puede utilizar para eliminar los elementos seleccionados. Los enlaces se borran junto a los elementos. Utilizar el comando "**Edición / Deshacer**" para recuperar los elementos después de un comando DEL. El comando se puede aplicar también a un grupo de elementos seleccionados en el diagrama. Los comandos "**Cortar**", "**Copiar**", "**Pegar**" del menú de "**Edición**" se utilizan para mover o copiar los elementos seleccionados.

Encontrar y reemplazar

Los comandos "**Edición / Encontrar Reemplazar**" se utilizan para encontrar o reemplazar cadenas de texto en todo el diagrama (todas las acciones y decisiones programadas en ST, IL o Quick LD). La ventana de diálogo de Encontrar/Reemplazar se utiliza para introducir un texto a buscar y abrir directamente la sección de programación donde ese texto aparece.

Acceso directo a un elemento

El comando "**Edición / Ir a**" permite al usuario acceso a un elemento gráfico existente en el diagrama. Se adapta automáticamente la posición de *scrolling* de forma que el elemento sea visible. Cuando se alcanza el elemento se selecciona.

Renumeración de elementos

El comando "**Edición / Renumerar**" se utiliza para renumerar elementos del diagrama de flujo. Cualquier elemento FC colocado en el diagrama se identifica con un único numero de referencia. Los números de referencia los da el editor cada vez que se insertan nuevos elementos. El comando "**Renumerar**" permite reajustar la numeración de elementos según su posición en el diagrama. La numeración creciente se hace de arriba a abajo y de izquierda a derecha.

A.5.4 Introducción de la programación de Nivel 2

Para introducir el texto de Nivel 2, el usuario debe hacer doble click sobre el rectángulo de Nivel 2 de un paso o una transición. La programación de nivel 2 se muestra a la derecha de la ventana SFC. La línea de separación entre el diagrama SFC y la programación de nivel 2 se puede mover libremente.

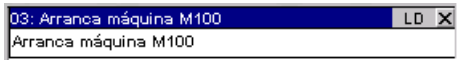
Se pueden abrir simultáneamente dos áreas de nivel 2 al mismo tiempo. Los siguiente comandos están disponibles desde el teclado, el ratón o el menú de "Edición".

	Teclado	*Ratón*	*Menú "Edición"*
Abre la última ventana por defecto	Intro	Doble Click	Edita el nivel 2
Lo abre en otra ventana	Ctrl+Intro	Ctrl + Doble Click	Edita el nivel 2 en otra ventana

Cuando se tienen abiertas dos ventanas de nivel 2, la separación entre ellas se puede mover libremente. El botón de la derecha de la barra de título de nivel 2 se utiliza para cerrar una ventana de nivel 2.

EL lenguaje por defecto para la programación del nivel 2 es ST (Texto Estructurado). La programación del nivel 2 se puede hacer también con el editor **Quick LD**. El nombre del lenguaje seleccionado se despliega en una pequeña ventana en la barra de título de nivel 2. Ejecutar el comando "**Opciones / Establecer el lenguaje de nivel 2**" desde los menús o haciendo click en la ventana para cambiar el lenguaje activo. Este comando sólo es valido si la ventana de programación de nivel 2 se encuentra vacía.

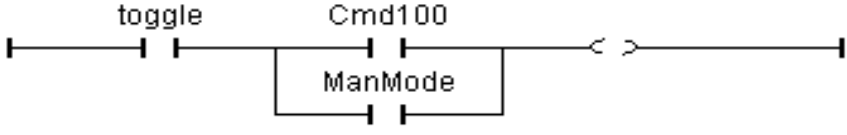
Una única línea aparece en la parte superior de la ventana de nivel 2. Se utiliza para introducir un pequeño texto de descripción. Este texto se ve como un comentario IEC al dibujar los símbolos FC. Es muy útil al utilizarse por otros comandos tales como "Ir a..." y también en la impresión FC para documentar acciones y decisiones FC.



El comando "**Opciones / Refrescar**" se puede utilizar en cualquier instante cuando hay abiertas ventanas de nivel 2 para refrescar el diagrama SFC principal

A.5.5 Programación con Quick Ladder

El editor *Quick LD* está disponible para la programación de Nivel 2. En el caso de una acción o decisión, el diagrama LD se compone de un único escalón (línea), con una sola bobina (salida) en representación de la decisión. No se repite el nombre de la decisión con el símbolo de la bobina (salida). A continuación se muestra un ejemplo de una condición de transición programada en *Quick LD*:



Cuando se programa en *Quick LD*, utilizar las flechas del teclado para desplazar la selección dentro de la rejilla lógica de programación y después utilizar los siguientes aceleradores de teclado para insertar símbolos:

F2: ..... insertar un contacto después del símbolo seleccionado / iniciar el escalón (línea)
F3: ..... insertar un contacto antes del símbolo seleccionado
F4: ..... insertar un contacto en paralelo con el símbolo seleccionado
F6: ..... insertar un bloque después del símbolo seleccionado
F7: ..... insertar un bloque antes del símbolo seleccionado
F8: ..... insertar un bloque en paralelo con el símbolo seleccionado
F9:..... añadir un símbolo de salto en paralelo con la bobina seleccionada (no para decisiones)

Un salto lleva a un nombre de escalón. El nombre de un escalón se puede introducir pulsando la tecla INTRO cuando la selección está encima del escalón. El editor de ISaGRAF lleva memoria de las etiquetas de los escalones ya introducidos, ya se hayan especificado para nombres de escalones o una operación de salto. La ventana de diálogo "Salto /Etiqueta" te da la posibilidad de introducir una nueva etiqueta o seleccionar una ya existente. Si se introduce un nombre nuevo, será añadido automáticamente a la lista. El botón de "**Eliminar**" se utiliza para eliminar el nombre seleccionado de la lista. No elimina la etiqueta del escalón que se seleccionó en el diagrama. Para hacer esto, simplemente presionar **Aceptar** cuando la ventana de edición está vacía.

También se puede hacer click sobre la barra de herramientas LD, en lugar de pulsar los botones de funciones.

Cuando está seleccionado un contacto o un parámetro de bloque de E/S, pulsar 'Retorno' para seleccionar una variable o introducir un valor constante. Cuando está seleccionado un bloque de función, pulsar 'Retorno' para seleccionar el tipo de bloque de función. También se puede hacer doble click sobre un símbolo para lograr el mismo efecto.

Cuando está seleccionado un contacto, pulsar Ctrl + Barra espaciadora para cambiar el tipo de contacto (directo, negado o con detección de impulsos). Para más información sobre las capacidades de *Quick LD*, véase el apartado titulado **Utilización del editor *Quick LD***".

A.5.6 Opciones de visualización

El comando "**Opciones / Distribución**" abre una ventana de diálogo donde se agrupan todos los parámetros del editor del espacio de trabajo y el dibujo del diagrama. Utilizar las opciones en el grupo de espacio de trabajo para ver u ocultar la barra de herramientas del editor y la barra de estado en la pantalla del espacio de trabajo. Las opciones del grupo "Documento" permiten ver u ocultar los puntos de la rejilla y ver el diagrama en blanco y negro o con colores.

Utilizar el botón de "Zoom" de la barra de herramientas para cambiar la relación de zoom actual. Este comando está también disponible cuando se trabaja en un programa Quick LD asociado a una acción o a una decisión.

Utilizar el botón de "Rejilla" de la barra de herramientas para ver o ocultar los puntos de la rejilla de edición. Este comando está también disponible cuando se trabaja en un programa Quick LD asociado a una acción o a una decisión.

Utilizar el comando "**Opciones / Fuente**" para seleccionar el nombre de la fuente de caracteres que se utilice en todos los documentos de ISaGRAF. Cuando se llama desde un bloque ST o IL, se puede especificar el tamaño de la fuente. Cuando se selecciona una fuente para un editor gráfico (FC o Quick LD), el estilo de fuente y el tamaño no son relevantes y no necesita especificarse los editores gráficos de ISaGRAF siempre calculan el tamaño de la fuente según la relación actual de zoom.

A.6 Utilización del editor *Quick LD*

El lenguaje LD permite la representación gráfica de expresiones booleanas. Los operadores booleanos AND, OR, NOT están representados explícitamente por la topología del diagrama. Las variables booleanas de entrada están asociadas a contactos gráficos. Las variables booleanas de salida están asociadas a bobinas (salidas) gráficas. El editor Quick LD de ISaGRAF permite una sencilla entrada de diagramas LD utilizando bien el teclado o bien el ratón. El editor Quick LD enlaza y ordena los elementos en escalones (líneas) de forma automática. El usuario no tiene que dibujar las conexiones manualmente. El editor Quick LD también ordena los escalones (líneas) del diagrama para que el espacio ocupado por el diagrama siempre esté optimizado.

A.6.1 Aspectos básicos del lenguaje LD

Un programa LD se expresa como una lista de ***escalones (líneas)*** en los que se disponen contactos y bobinas (salidas). A continuación se muestran los componentes básicos de un diagrama LD:

Principio de escalón (línea) (carril de potencia izquierdo)

Cada escalón (línea) comienza con un carril de potencia izquierdo, que representa el estado VERDADERO inicial. El editor *Quick LD* de ISaGRAF crea el carril de potencia izquierdo automáticamente cuando el usuario coloca el primer contacto del escalón (línea). Cada escalón (línea) puede tener un nombre lógico, que puede utilizarse como etiqueta para las instrucciones de salto.

Contactos

Un contacto modifica el flujo de datos booleanos, de acuerdo con el estado de una variable booleana. El nombre de la variable se muestra encima del símbolo del contacto. El editor *Quick LD* de ISaGRAF soporta los siguientes tipos de contacto:

 ..... contacto directo

 ..... contacto invertido

 ..... contacto con detección de flancos positivos (de subida)

 ..... contacto con detección de flancos negativos (de bajada)

Bobinas (salidas)

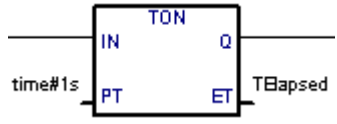
Una bobina (salida) representa una acción. Se utiliza el estado del escalón (línea) (el estado del vínculo situado a la izquierda de la bobina) para forzar una variable booleana. Se muestra el nombre de la variable encima del símbolo de la bobina. El editor *Quick LD* de ISaGRAF soporta los siguientes tipos de bobina:

 ..... bobina directa

-(\)- ..... bobina negada
-(S)- ..... bobina de acción “establecer”
-(R)- ..... bobina de acción “restablecer”
-(P)- ..... bobina con detección de flancos positivos (de subida)
-(N)- ..... bobina con detección de flancos negativos (de bajada)

Bloques de función

Un bloque en un diagrama LD puede representar una función, un bloque de función, un subprograma o un operador. Sus primeros parámetros de entrada y salida siempre están conectados al escalón (línea). Los restantes parámetros de entrada y salida se escriben de forma literal fuera del rectángulo de bloque.



Fin de escalón (línea) (carril de potencia derecho)

Un escalón (línea) finaliza con un carril de potencia derecho. El editor *Quick LD* de ISaGRAF inserta el carril de potencia derecho automáticamente cuando el usuario coloca una bobina (salida).

Símbolo de salto

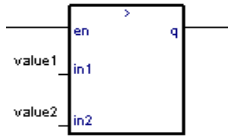
Un símbolo de salto siempre se refiere a una etiqueta de escalón (línea), es decir, el nombre de un escalón (línea) que está definido en otra parte del mismo diagrama LD. Se coloca al término de un escalón (línea). Cuando el estado del escalón (línea) es VERDADERO, la ejecución del diagrama salta directamente a este escalón (línea) destino. Obsérvese que los saltos hacia atrás son peligrosos, ya que en algunos casos pueden llevar al bloqueo del ciclo de PLC.

Símbolo de retorno

Se coloca un símbolo de retorno al término del escalón (línea), indicando que se debe parar la ejecución del programa si el estado del escalón (línea) es VERDADERO.

La entrada “EN”

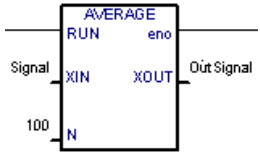
En algunos operadores, funciones o bloques de función, la primera entrada no posee datos del tipo booleano. Como la primera entrada tiene que estar conectada siempre al escalón (línea), se inserta otra entrada, denominada “**EN**”, en la primera posición de forma automática. El bloque sólo se ejecuta si la entrada **EN** es VERDADERO. A continuación se muestra un ejemplo del operador de comparación y el código equivalente expresado en ST:




```
IF rung_state THEN
   q := (value1 > value 2);
ELSE
   q := FALSE;
END_IF;
(*continuar escalón con estado q *)
```

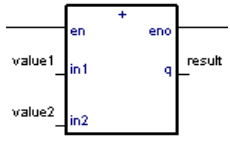
La salida "ENO"

En algunos operadores, funciones o bloques de función, la primera salida no posee datos del tipo booleano. Como la primera salida tiene que estar conectada siempre al escalón (línea), se inserta otra salida, denominada "**ENO**", en la primera posición de forma automática. La salida "**ENO**" siempre toma el mismo estado que la primera entrada del bloque. A continuación se muestra un ejemplo con el bloque de función AVERAGE y el código equivalente expresado en ST:




```
AVERAGE(rung_state, Signal, 100);
OutSignal := AVERAGE.XOUT;
eno := rung_state;
    (*continuar escalón con estado eno *)
```

En algunos casos se requieren ambas, **EN** y **ENO**. A continuación hay un ejemplo con un operador aritmético , y el código equivalente expresado en ST:




```
IF rung_state THEN
   result := (value1 + value2);
END_IF;
eno := rung_state;
(*continuar escalón con estado eno *)
```

Limitaciones del editor Quick LD

El editor *Quick LD* de ISaGRAF no permite extender un escalón (línea) (insertar otros contactos o bobinas) a la derecha de una bobina (salida). Si se necesitan incluir varias salidas en el mismo escalón (línea), las bobinas (salidas) correspondientes tienen que dibujarse en paralelo.

A.6.2 Introducción de un diagrama LD

Se puede acceder a todos los comandos de edición del editor *Quick LD* por medio del teclado o del ratón.

La rejilla de edición

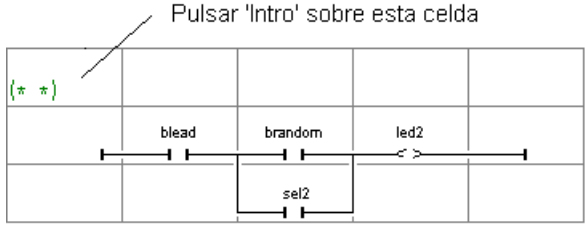
El diagrama LD se introduce en una matriz lógica. Cada celda de la matriz puede contener hasta un símbolo LD. Utilizar las flechas del teclado o hacer click sobre una celda para mover la selección actual. La celda seleccionada está señalada en inverso. Para algunas operaciones de cortar/copiar/pegar, se permite la selección de varias celdas. Para hacer lo mismo con el ratón, sólo hay que arrastrar el cursor del ratón dentro del diagrama. Con el teclado, utilizar los botones de flecha a la vez que se pulsa el botón de SHIFT o mayúsculas.

Comienzo de un escalón (línea) nuevo

Para añadir un escalón (línea) nuevo al diagrama, colocar la selección detrás del último escalón (línea) e insertar un contacto (pulsar F2 o el botón correspondiente en la barra de herramientas de LD). De esta manera, se creará un nuevo escalón (línea) con un contacto y una bobina (salida).

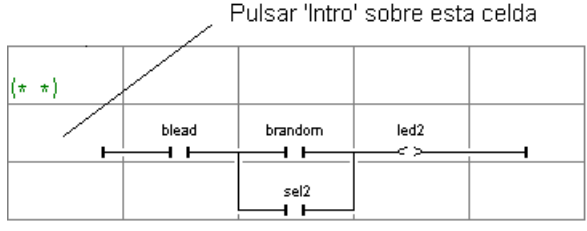
Introducción de comentarios del escalón (línea)

Cada escalón (línea) puede estar documentado con hasta dos renglones de texto. Para introducir un texto de comentario, desplazar la selección de celda sobre el escalón (línea) y pulsar la tecla 'Intro', o hacer doble click sobre la celda con el ratón:



Introducción de la etiqueta del escalón (línea)

Se puede identificar cada escalón (línea) con un nombre. Este nombre puede ser utilizado como etiqueta destino en operaciones de salto. Para introducir o modificar el nombre de un escalón (línea), desplazar la selección al inicio de escalón (línea) y pulsar la tecla 'Intro', o hacer doble click sobre esta celda con el ratón:



El editor *Quick LD* de ISaGRAF memoriza las etiquetas de escalón (línea) que ya han sido introducidas por el usuario, con independencia de si han sido especificados como nombres de escalón (línea) o para operaciones de salto. La

ventana de diálogo “Salto/Etiqueta” permite al usuario introducir una etiqueta nueva o seleccionar una que ya existe.

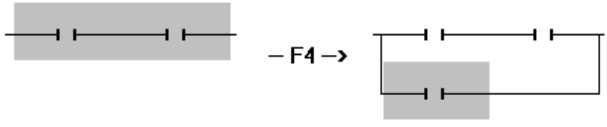
Si se selecciona un nombre nuevo, se añadirá automáticamente a la lista. Se utiliza el botón "**Borrar**" para eliminar de la lista el nombre seleccionado. No elimina la etiqueta del escalón (línea) que se ha seleccionado en el diagrama. Para ello, sólo hay que pulsar **Aceptar** cuando la ventana de edición está vacía.

Inserción de símbolos en un escalón (línea)

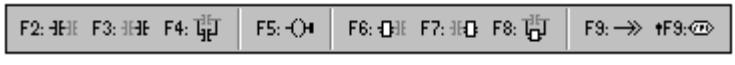
La inserción de símbolos (contactos, bobinas, bloques, etc.) en un escalón (línea) ya existente siempre se hace de acuerdo con la selección actual. Seleccionar una posición de celda válida en el escalón (línea) y pulsar uno de los siguientes botones de funciones para insertar:
F2..... un contacto antes del símbolo seleccionado (a la izquierda)
F3..... un contacto después del símbolo seleccionado (a la derecha)
F4..... un contacto en paralelo al símbolo seleccionado
F6..... un bloque antes del símbolo seleccionado (a la izquierda)
F7..... un bloque después del símbolo seleccionado (a la derecha)
F8..... un bloque en paralelo al símbolo seleccionado

Los siguientes comandos son válidos cuando la selección se encuentra en la salida del escalón (línea):
F5..... añadir una bobina (salida) en paralelo al seleccionado
F9..... añadir un símbolo de “Salto” en paralelo al seleccionado
May + F9 ..... añadir un símbolo de "Retorno" en paralelo al seleccionado

En el caso de la inserción en paralelo (F4/F8), si se seleccionan a la vez varios contactos de un escalón (línea), el símbolo se inserta en paralelo al grupo de elementos seleccionados. A continuación se muestra un ejemplo:



Para insertar símbolos en el diagrama, también se pueden utilizar los comandos del menú "**Insertar**". Con el ratón, hacer click sobre la barra de herramientas clave situada en la parte inferior de la pantalla, sobre el tipo de símbolo que se desea insertar:



Introducción de símbolos

Para asociar un símbolo de variable a un contacto o una bobina (salida), seleccionarlo y pulsar ‘Retorno’. Con el ratón, hacer doble click en el contacto o el bucle. Se abre una ventana de selección de variables. Para más información sobre la manera de utilizar esta ventana, véase el apartado titulado “**Más sobre los editores de programas**”. Para asociar una función, un bloque de función o un operador a un bloque, pulsar ‘Retorno’ cuando la selección se encuentra en el

interior de su rectángulo. Para asociar un símbolo de variable a un parámetro de bloque de entradas o salidas, la selección tiene que estar en la posición correspondiente, **fuera** del rectángulo del bloque.

La ventana de diálogo incluyendo variables o bloque de selección de listas son normalmente utilizadas como entradas de texto. Si el modo "**Entrada manual por teclado**" está seleccionado en el menú de "**Opciones**", los símbolos de variables y los nombres de bloque se introducen directamente en una única ventana de edición. Introduzca el nuevo texto y pulse la tecla "**Intro**" para comprobrarlo, o pulse la tecla "**Escape**" para salir de las modificaciones y cerrar la ventana de edición de texto . La ventana de edición de texto utilizada en el modo "Entrada manual por teclado" no se puede cerrar con el ratón.

Cambio del tipo de contactos y bobinas (salidas)

El comando "**Edición / Cambiar tipo de contacto/bobina (salida)**" cambia el tipo de contacto o bucle que está seleccionado. Un contacto puede ser directo, invertido o con detección de flancos positivos o negativos. Una bobina puede ser directa, negada, establecida o restablecida, con detección de flancos positivos o negativos. Al pulsar la barra espaciadora se logra el mismo efecto.

Inserción de un escalón (línea) en un diagrama

El comando "**Edición / Insertar escalón (línea)**" inserta un escalón (línea) nuevo en el diagrama, delante del que está seleccionado. El escalón (línea) se inicia con un contacto y una bobina (salida).

A.6.3 Trabajar con un diagrama LD ya existente

Se utilizan los comandos del menú "**Edición**" para modificar o completar un diagrama ya existente. La mayoría de estos comandos actúa sobre los elementos actualmente seleccionados en el diagrama.

Corrección de diagramas

Se puede utilizar el botón DEL (Supr) para eliminar elementos seleccionados. No se puede eliminar una bobina (salida), un salto o un símbolo de retorno cuando es la única salida de un escalón (línea). Utilizar el comando "**Edición / Deshacer**" para restaurar elementos después de ejecutar un comando DEL. También se puede aplicar el comando DEL a un grupo de elementos que se seleccione en el diagrama. Puede emplearse el comando DEL cuando la selección se encuentra en el texto de comentario del escalón (línea), para borrarlo. El comando DEL, si se utiliza cuando la selección se encuentra sobre el inicio del escalón (línea), elimina todo el escalón (línea).

Copia de símbolos

Los comandos "**Cortar**", "**Copiar**", "**Pegar**" del menú "**Edición**" se utilizan para mover o copiar elementos seleccionados. Estos comandos no actúan sobre los comentarios de escalón (línea). El comando "**Edición / Pegado especial**" le da al usuario la opción de insertar los elementos pegados:

- antes del elemento seleccionado (a la izquierda)
- después del elemento seleccionado (a la derecha)
- en paralelo al elemento seleccionado

Gestión de escalones (líneas) completos

Todos los comandos de edición (borrar, copiar, cortar, etc.) actúan sobre el conjunto del escalón (línea) si la selección se sitúa sobre el inicio del escalón (línea) (carril de potencia izquierdo). Esta es una manera sencilla de ordenar escalones (líneas) en el diagrama, con sólo desplazar la selección a la primera columna. También es posible extender la selección en el sentido vertical para que incluya varios inicios de escalón (línea). En este caso, se pueden aplicar los comandos de edición a un lista de escalones (líneas) completos.

Buscar y reemplazar

Se utilizan los comandos de menú "**Edición / Buscar**" y "**Edición / Reemplazar**" para encontrar y sustituir textos del diagrama. Sólo se pueden buscar nombres completos. La función de búsqueda actúa sobre contactos, bobinas (salidas), nombres de bloque, parámetros de bloque y etiquetas de ejecución. No se puede utilizar para encontrar una cadena dentro de un comentario de escalón (línea). No se puede utilizar el comando **Reemplazar** para cambiar el tipo de bloque. La búsqueda puede ser en sentido ascendente o descendente, empezando en la posición de la selección actual. Ejecuta un bucle cuando alcanza los límites del diagrama. También se dispone de los siguientes aceleradores de teclado para la búsqueda rápida de nombres de variables:
ALT + F2 encuentra el siguiente elemento con el mismo nombre de variable que el elemento que está seleccionado en ese momento. Esta prestación también puede aplicarse a los bloques de función y las etiquetas de escalones (líneas).
ALT + F5 encuentra la siguiente bobina (salida) con el mismo nombre de variable que el elemento que está seleccionado en ese momento. Esta función se utiliza principalmente en el modo de depuración, para averiguar rápidamente cuáles son los escalones (líneas) que fuerzan una variable sospechosa.

A.6.4 Opciones de visualización

Se utilizan los comandos del menú "**Opciones**" para personalizar la manera de dibujarse el diagrama LD en pantalla, y para ocultar o mostrar determinados tipos de información.

Comentarios de escalón (línea)

Utilizar el comando "**Opciones / Comentarios de escalón (línea)**" para ocultar o mostrar los comentarios de escalón (línea) de todo el diagrama. Puede que sea necesario ocultar los comentarios de escalón (línea) para obtener una vista más condensada de un diagrama enorme, ya que cada comentario consume un renglón de la matriz de edición. Esta opción no afecta al contenido de los comentarios de escalón (línea) ya existentes, y puede permutarse en cualquier momento.

Nombres y alias

Cada variable, al asociarse a un contacto, bobina (salida) o parámetro de bloque de E/S, está identificado por su nombre simbólico. El editor *Quick LD* de ISaGRAF también introduce la noción o "**alias**" de cada variable. El alias de la variable es su texto de comentario, truncado antes del primer carácter '**:**' y limitado a 16 caracteres. A continuación se muestran algunos ejemplos:

comentario variable:	*alias:*
short text	short text
long text with no separator	long text with n
short text: long description	short text

Los alias no afectan a la ejecución del diagrama LD y deberán ser considerados como comentarios desde el punto de vista sintáctico. Un alias de variable se extrae automáticamente del comentario de la variable cuando se selecciona el nombre de la lista de variables. No se puede cambiar manualmente. Utilizar los comandos "**Opciones / Contactos y bobinas (salidas)**" para seleccionar un modo de visualización para la identificación de variables. Se dispone de los siguientes modos:

- mostrar sólo los nombres de las variables
- mostrar sólo los alias de las variables
- mostrar tanto nombres como alias

Opciones de dibujo

El comando "**Opciones / Distribución**" abre una ventana de diálogo que agrupa a todos los parámetros y opciones relacionados con el espacio de trabajo del editor y el dibujo del diagrama gráfico LD.

Utilizar las casillas de la sección "Espacio de trabajo" para mostrar u ocultar la barra de herramientas del editor, la barra de estado y la barra de herramientas LD. Las opciones de la sección "Documento" le permiten al usuario mostrar u ocultar los puntos de la rejilla de edición, además de habilitar o deshabilitar el uso de colores en el dibujo.

Las opciones de la sección "Zoom" le permiten al usuario seleccionar el principal ratio de aumento o zoom. También puede utilizarse el botón "zoom" de la barra de herramientas del editor para permutar entre las relaciones de zoom por defecto.

El usuario también puede personalizar el ratio de aspecto X/Y de las celdas pertenecientes a la rejilla de edición. Se puede utilizar esta opción para reducir el ancho por defecto de un celda, si se suelen utilizar nombres cortos para las variables. También se puede utilizar el botón "ancho" de la barra de herramientas del editor para cambiar la relación de aspecto X/Y sin necesidad de entrar en la ventana de diálogo de **Distribución**.

Utilizar el comando "**Opciones / Fuente**" para seleccionar el nombre de la fuente de caracteres que se usará en todos los documentos gráficos de ISaGRAF. Cuando se selecciona una fuente, el estilo de fuente y su tamaño no son relevantes y no necesitan especificarse. Los editores gráficos de ISaGRAF siempre calculan el tamaño de la fuente según la relación de zoom seleccionada.

A.7 Utilización del editor FBD/LD

El editor gráfico FBD/LD de ISaGRAF permite al usuario introducir programas FBD completos, pudiéndose incluir partes en LD. Combina las capacidades de edición de textos y gráficos, por lo que admite la introducción tanto de diagramas como de las entradas y salidas correspondientes. Ya que este editor está más orientado al lenguaje FBD, es preferible que los diagramas LD puros se introduzcan utilizando el editor *Quick LD* de ISaGRAF.

A.7.1 Aspectos básicos de los lenguajes FBD/LD

El lenguaje **FBD** es una representación gráfica de muchos tipos de ecuaciones diferentes. Los **operadores** están representados por cajas de funciones rectangulares. Las entradas de funciones se conectan al lateral izquierdo de la caja. Las salidas de funciones se conectan al lateral derecho. Las entradas y salidas del diagrama (**variables**) están conectadas a las cajas de funciones por medio de **enlaces lógicos**. Una salida de una caja de funciones puede estar conectada a la entrada de otra caja.

El lenguaje **LD** permite la representación gráfica de expresiones booleanas. Los operadores booleanos **AND**, **OR**, **NOT** están representados explícitamente por la topología de diagrama. Las variables booleanas de entrada están asociadas a **contactos** gráficos. Las variables booleanas de salida están asociadas a **bobinas (salidas)** gráficas. Los contactos y las bobinas (salidas) están conectados entre sí y a los carriles de potencia izquierdo y derecho por medio de **líneas horizontales**. Cada segmento de línea tiene un estado booleano **FALSO** o **VERDADERO**. El estado booleano es el mismo para todos los segmentos que estén enlazados directamente entre sí. Cualquier línea horizontal que esté conectada al **carril de potencia vertical** de la izquierda posee el estado **VERDADERO**.
Los diagramas LD y FBD siempre se interpretan de izquierda a derecha y de arriba abajo. Para mayor información sobre los lenguajes LD y FBD, véase el Manual de Referencia de Lenguajes ISaGRAF. A continuación se muestran los componentes gráficos básicos de los lenguajes LD y FBD que soporta el editor FBD/LD:

Carril de potencia izquierdo

Los escalones (líneas) tienen que estar conectados por la izquierda al **carril de potencia izquierdo**, que representa el estado “VERDADERO” inicial. El editor FBD de ISaGRAF permite asimismo la conexión de cualquier símbolo booleano a un carril de potencia izquierdo.

Carril de potencia derecho

Se pueden conectar bobinas (salidas) por la derecha a un **carril de potencia derecho**. Esta es una característica opcional del editor FBD/LD de ISaGRAF. Si un

bucle no está conectado por la derecha, incluye un carril de potencia derecho en su propio dibujo.

Conexión vertical LD "OR"

Las conexiones verticales LD aceptan varias conexiones a la izquierda y varias conexiones a la derecha. Cada conexión de la derecha es equivalente a la combinación **OR** de las conexiones de la izquierda.

Contactos

Un contacto modifica el flujo de datos booleanos, de acuerdo con el estado de una variable booleana. Se muestra el nombre de la variable encima del símbolo de contacto. El editor FBD/LD de ISaGRAF soporta los siguientes tipos de contactos:

- ..... contacto directo
- ..... contacto invertido
- ..... contacto con detección de flancos positivos (de subida)
- ..... contacto con detección de flancos negativos (de bajada)

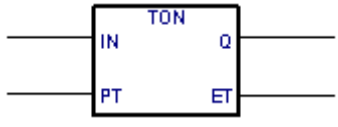
Bobinas (salidas)

Una bobina (salida) representa una acción. Tiene que estar conectada por la izquierda con un símbolo booleano como un contacto. El nombre de la variable aparece encima del símbolo de la bobina. El editor FBD/LD de ISaGRAF soporta los siguientes tipos de bobina:

- ..... bobina directa
- ..... bobina invertida
- ..... bobina de acción “establecer”
- ..... bobina de acción “restablecer”

Bloques de función

Un bloque en un diagrama FBD puede representar una función, un bloque de función, un subprograma o un operador. Las entradas y salidas tiene que estar conectados a variables, contactos o bobinas (salidas), o a otros bloques de entradas o salidas. Los nombres formales de los parámetros aparecen en el interior del rectángulo correspondiente al bloque.



Etiquetas

Se pueden colocar etiquetas en cualquier parte del diagrama. Las etiquetas se utilizan como destinos en las instrucciones de salto, para cambiar el orden de ejecución del diagrama. Las etiquetas no están conectadas a otros elementos. Es altamente recomendable la colocación de etiquetas en la parte izquierda del diagrama, para aumentar la legibilidad del diagrama.

Saltos

Un símbolo de salto se refiere siempre a una etiqueta situada en otra parte del diagrama. Su conexión izquierda tiene que estar vinculada a un punto booleano. Cuando la conexión izquierda es VERDADERO, la ejecución del diagrama salta directamente a esta etiqueta destino. Obsérvese que los saltos hacia atrás son peligrosos, ya que en algunos casos pueden llevar al bloqueo del bucle del PLC.

Símbolo de retorno

Un símbolo de retorno está conectado a un punto booleano. Indica que se deberá interrumpir la ejecución del programa si el estado del escalón (línea) es VERDADERO.

Variables

Las variables se representan en el diagrama dentro de rectángulos pequeños que están conectados a izquierda o derecha con otros elementos del diagrama.

Enlaces de conexión

Se dibujan enlaces de conexión entre los elementos que se colocan en el diagrama. Los enlaces siempre se dibujan entre un punto de salida y un punto de entrada (en el sentido del flujo de datos).

Enlaces de conexión con negación booleana

Algunos enlaces booleanos están representados por un círculo pequeño en su extremo derecho. Esto representa una negación booleana de la información que pasa por el enlace.

Esquinas definidas por el usuario

Se pueden establecer puntos definidos por el usuario en los enlaces. Estos puntos le permiten al usuario el control manual del *routing* o encaminamiento de un enlace. Si no se coloca una esquina, el editor FBD/LD de ISaGRAF utiliza un algoritmo de *routing* por defecto.

A.7.2 Introducción de un diagrama FBD

Para introducir un diagrama, el usuario tiene que colocar elementos (bloques, variables, contactos, bobinas, etc.) en la zona gráfica y después dibujar enlaces entre ellos.

Inserción de objetos

Para insertar un objeto en el diagrama, seleccionar el botón correspondiente en la barra de herramientas y hacer click en la zona gráfica, en el lugar en que se desea hacer la inserción.

Selección de objetos

La selección de objetos gráficos es necesaria para la mayoría de los comandos de edición. El editor gráfico LD/FBD de ISaGRAF permite la selección de uno o más objetos existentes en la zona del diagrama. Para poder seleccionar objetos, antes debe estar marcada la opción "**Seleccionar**" (botón con una flecha) en la barra de herramientas del editor. Para seleccionar un único objeto, el usuario sólo tiene que hacer click sobre el símbolo correspondiente. Para seleccionar una lista de objetos, arrastrar el cursor del ratón dentro del diagrama y seleccionar un área rectangular. Todos los objetos gráficos que cortan el rectángulo de la selección quedan marcados como "**seleccionados**". Los objetos seleccionados muestran unos recuadros negros pequeños alrededor de su símbolo gráfico. Al realizar una nueva selección, todos los objetos que previamente estaban seleccionados dejan de estarlo. Para eliminar la selección actual, sólo hay que hacer click con el ratón sobre una zona vacía, fuera del rectángulo que bordea a los objetos seleccionados.

Inserción de comentarios

Se pueden insertar comentarios en cualquier punto del diagrama. Los comentarios no influyen en la ejecución del programa. Permiten una mayor legibilidad del diagrama. Para insertar un bloque de comentarios, seleccionar este botón en la barra de herramientas y arrastrar el cursor del ratón para seleccionar el área rectangular en el que se desea insertar el comentario. Después, introducir el texto del comentario. No hacen falta caracteres iniciales o finales especiales, como "**(***" y "***)**", cuando se introduce el texto de un bloque de comentarios. Para redimensionar un bloque de comentarios, arrastrar las esquinas del borde cuando se encuentre seleccionado.

Desplazamiento de objetos

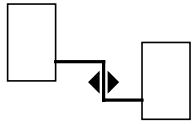
Para desplazar objetos en el diagrama, primero se seleccionan y después se arrastra el cursor del ratón para desplazar la zona seleccionada dentro del diagrama. Para mover objetos conectados, el usuario sólo tiene que mover los símbolos gráficos que se encuentran en el diagrama. El editor LD/FBD de ISaGRAF volverá a dibujar automáticamente las líneas de conexión existentes entre los objetos que fueron desplazados, con base en su nueva ubicación.

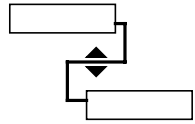
Dibujo de enlaces

Seleccionar uno de estos botones en la barra de herramientas para dibujar un enlace entre los puntos de conexión de elementos existentes. Si se dibuja un enlace desde un punto de conexión hasta una zona vacía del diagrama, se termina automáticamente con una esquina definida por el usuario para que se pueda continuar dibujando otro segmento.

Cambio de dibujos de enlace

Cuando un enlace está seleccionado en el diagrama, se utiliza el comando de "**Herramientas / Mover línea**" para cambiar su *routing* automático. Este comando no tiene efecto cuando el enlace está conectado a una esquina definida por el usuario. Cuando un enlace se dibuja en tres segmentos, este comando cambia la posición del segundo segmento. A continuación se muestran algunos ejemplos:





Cambio del tipo de enlace

Se puede cambiar el tipo de enlace (con o sin negación booleana) de forma sencilla, con sólo hacer doble click con el ratón sobre su extremidad derecha.

Dibujo de escalones (líneas) LD

Para dibujar un nuevo escalón (línea) LD, primero hay que insertar el carril de potencia izquierdo. Después se coloca una bobina (salida), que quedará enlazada automáticamente al carril de potencia. Se pueden insertar los demás contactos y conexiones verticales OR directamente en la línea del escalón (línea), sin necesidad de dibujar un nuevo enlace de conexión.
Cuando se inserta un nuevo contacto o bobina (salida) LD en un espacio vacío del área de edición, la nueva línea horizontal del escalón (línea) se dibuja automáticamente desde el elemento que se acaba de insertar hasta los carriles de potencia ya existentes a la izquierda y la derecha. No se dibuja esta línea automáticamente si no se sitúa el nuevo contacto o bobina (salida) entre los carriles de potencia. El contacto o bobina (salida) recién insertado puede moverse libremente sobre el escalón (línea). Las líneas horizontales creadas por el editor al insertarse un contacto o símbolo de bobina (salida) LD pueden ser seleccionadas y eliminadas. Se puede insertar un nuevo contacto o símbolo de bobina (salida) LD en la línea horizontal de un escalón (línea) existente. El editor corta los escalones (líneas) de forma automática y los conecta a los puntos de conexión izquierdo y derecho del contacto o bobina (salida) que se acaba de insertar.

Conexiones múltiples

Se puede crear una conexión múltiple a la derecha de cualquier punto de **salida**. Eso significa que la información será **emitida** a diversos puntos del diagrama. Se propaga el mismo estado a cada extremidad de la derecha. No está limitado el número de líneas que se puede dibujar a la derecha de un punto de conexión de salidas. Dos líneas de conexión no pueden tener su extremidad derecha conectada al mismo punto de **entrada**, con la excepción de los siguientes símbolos LD:

..... carril de potencia derecho

..... conexión múltiple en el operador (OR) izquierdo

Estos símbolos LD pueden poseer un número ilimitado de entradas.

A.7.3 Trabajar con un diagrama ya existente

Se utilizan los comandos del menú "**Edición**" para cambiar o completar un diagrama existente. La mayoría de estos comandos actúa sobre los elementos que en ese momento están seleccionados en el diagrama.

Corrección de diagrama

Se puede utilizar el botón DEL (Supr) para eliminar elementos seleccionados. Se borran los enlaces pendientes con los elementos seleccionados. Utilizar el comando "**Edición / Deshacer**" para restaurar elementos después de ejecutar un comando DEL. También se puede aplicar el comando DEL a un grupo de elementos que se seleccione en el diagrama. Se utilizan los comandos "**Cortar**", "**Copiar**", "**Pegar**" del menú "**Edición**" para mover o copiar los elementos seleccionados.

Buscar y reemplazar

Se utilizan los comandos de menú "**Edición / Buscar**" y "**Edición / Reemplazar**" para encontrar y sustituir textos del diagrama. Sólo se pueden buscar nombres completos. La función de búsqueda actúa sobre contactos, bobinas (salidas), nombres de bloque, variables y etiquetas. No se puede utilizar para encontrar una cadena dentro del texto de un comentario. No se puede utilizar el comando **Reemplazar** para cambiar el nombre de un bloque. La búsqueda puede ser en sentido ascendente o descendente, empezando en la posición de la selección actual. Se ejecuta un bucle cuando se alcanzan los límites del diagrama.

Visualización de la orden de ejecución

Cuando un diagrama FBD incluye bucles hacia atrás, el orden de ejecución no puede observar el método izquierda a derecha / arriba abajo. Para evitar confusiones, utilizar el comando "**Herramientas / Mostrar orden de ejecución**" o pulsar los botones **Control + F1** para visualizar la orden de ejecución que se empleará a la hora de compilar. Se muestran unas etiquetas numeradas de 1 a N cerca de los símbolos que provocan acciones (bobinas, variables de referencia y bloques de función).

Introducción de símbolos y textos

Hacer doble click con el ratón sobre un elemento para introducir el símbolo o texto asociado. Esto es aplicable a variables, contactos y bobinas (salidas), textos de comentario y etiquetas. Cuando se aplica a un contacto o una bobina (salida), también permite cambiar su tipo (directo, invertido, etc.).

Se utilizan para entradas de texto ventanas de diálogo que incluyen variables o listas de selección en bloque. Si el modo "**Entrada manual por teclado**" se selecciona en el menú de "**Opciones**" , los símbolos de variables y nombres de bloque se introducen directamente en una única ventana de edición de texto. Introducir el nuevo texto y pulsar la tecla "**Intro**" para comprobarlo, o pulsar la tecla "**Escape**" para salir de la modificación y cerrar la ventana de edición de texto. La ventana de edición d texto en el modo "Entrada por teclado manual" no se puede cerrar con el ratón.

Si se ha seleccionado el modo "**Entrada Automática**" en el menú "**Opciones**", cada vez que se inserta un nuevo contacto o bobina (salida) se debe introducir el símbolo de la variable de forma inmediata. Cada vez que se inserta una variable o una etiqueta, se tiene que introducir el símbolo inmediatamente.

Selección del tipo de bloque de función

Hacer doble click con el ratón sobre un bloque para cambiar su tipo. Se selecciona el tipo del bloque de la lista de operadores, funciones y bloques de función

disponibles. Este comando también permite cambiar el número de puntos de entrada, en el caso de un operador conmutativo (p.ej. AND, OR, ADD, MUL...).

- ***Espacio libre***

Cuando se pulsa el botón derecho del ratón en un área de dibujo, se despliega un menú de entrada/salida. Contiene los siguientes comandos que se pueden utilizar para insertar, o eliminar espacio libre en las posición del cursor del ratón.:

Insertar filas: Este comando inserta espacio libre horizontal, hecho de 4 filas según el paso de la rejilla, comenzando en la posición del cursor del ratón donde se abre el menú desplegable.

Borrar filas: Este comando elimina espacio horizontal no utilizado (filas) comenzando en la posición del cursor del ratón donde se abrió el menú desplegable. Este comando no se puede utilizar para eliminar elementos FBD.

Cuando se abre el menú desplegable una línea gris en el dibujo FBD indica donde se insertará o eliminará el espacio libre.

A.7.4 Opciones de visualización

Se utilizan los comandos del menú "**Opciones**" para personalizar la presentación del diagrama FDB en pantalla.

- ***Personalización de la distribución***

El comando "**Opciones / Distribución**" abre una ventana de diálogo en la que se agrupan todos los parámetros y opciones relacionados con el espacio de trabajo del editor y la forma de dibujar el diagrama gráfico. Utilizar las casillas seleccionables de la sección "Espacio de trabajo" para mostrar u ocultar las barras de herramientas y la barra de estado del editor. Las opciones de la sección "Documento" le permiten al usuario mostrar u ocultar los puntos de la rejilla de edición.

Las opciones de la sección "Zoom" permiten seleccionar una relación de zoom principal. También se puede utilizar el botón de "zoom" en la barra de herramientas del editor para conmutar entre las relaciones de zoom establecidas por defecto.

Utilizar el comando "**Opciones / Fuente**" para seleccionar el nombre de la fuente de caracteres que se utilizará en los documentos gráficos de ISaGRAF. Cuando se selecciona una fuente, el estilo de fuente o tamaño no son relevantes y no se necesita especificarlos. Los editores gráficos de ISaGRAF siempre calculan el tamaño de la fuente según la relación de zoom seleccionada.

A.7.5 Estilos y rasteo de modificaciones

El editor LD/FBD de ISaGRAF permite asignar un estilo gráfico a cualquier componente de un diagrama LD/FBD. Un estilo se define principalmente por un color especial del diagrama. Pero ISaGRAF también utiliza estilos para permitir el rastreo de modificaciones en diagramas para control de versiones.
Notar que los estilos no son visibles durante la simulación o la depuración en línea, tal como se usan los colores (rojo y azul) en estos modos para resaltar los estados VERDADERO / FALSO de las variables espiadas.

- ***Estilos predefinidos***

Están predefinidos los siguiente estilos:

Normal Dibujo por defecto (negro). Para el rastreo de modificaciones, el estilo “normal" indica que los elementos que tienen dicho estilo son parte del diagrama original. Los elementos estilo "Normal" son normalmente mostrados durante la ejecución.

Modificados Los elementos marcados como "modificados" se pintan en rosa. Para el rastreo de modificaciones, el estilo, "modificado" se usa para resaltar los elementos que se han añadido o cambiado después de la versión original del programa. Los elementos estilo "Modificado" son normalmente mostrados durante la ejecución.

Borrados Los elementos marcados como "borrado" se pintan en gris. Dichos elementos no se tienen en cuenta para la ejecución del programa. Este estilo es el que se usa para llevar un rastreo de los elementos eliminados después de la versión original cuando se necesita el control de versiones.

Configurable Además de los estilos predefinidos el editor LD/FBD de ISaGRAF permite seleccionar cualquier color para utilizarse en cualquier parte del diagrama. Estos elementos se consideran de estilo "Configurable". La utilización del estilo "Configurable" no tiene efectos en el diagrama durante el tiempo de ejecución.

Utilizar los comandos del submenú de "**Estilo**" en el menú "**Edición**" para aplicar manualmente un estilo a los elementos seleccionados.

- ***Rastreo de modificaciones***

El uso de estilos y la disponibilidad del estilo "Borrado" permite un rastreo de modificaciones inmediato en el diagrama existente. Utilizar el comando "**Marcar modificaciones**" en el menú "**Edición / Estilo**" para habilitar o deshabilitar el rastreo de modificaciones.

Cuando la opción "Marcar modificaciones" está habilitada, todos los elementos cambiados o añadidos al diagrama se configuran con el estilo "Modificado" . Cuando un elemento se borra, usando el comando "Eliminar" o "Cortar", no son eliminados visualmente del diagrama, sino marcados con el estilo "Borrado". Esto

permite al usuario llevar un historial de todas las modificaciones realizadas en el diagrama.

Utilizar "**Edición / Estilo / Eliminar todos los items borrados**" para eliminar realmente los elementos marcados con estilo "Borrado" del diagrama LD/FBD. Este comando no tiene en cuenta la selección actual y se aplica siempre al diagrama entero.

Para "recuperar" un elemento marcado con el estilo "**Borrado**", seleccionar el elemento deseado y aplicarle el estilo "**Normal**" , el estilo "**Modificado**" o cualquier estilo "**Configurable**". Esta operación puede llevar a conexiones invalidas (más de un enlace conectado al mismo punto de entrada) que se detectaran durante la siguiente verificación del programa.

A.8 Utilización del editor de textos

Este apartado sólo describe características y comandos específicos del editor de textos de ISaGRAF, en especial cuando se utiliza para introducir el código fuente de programas ST e IL.
El editor de textos de ISaGRAF también se utiliza para introducir el descriptor de proyectos, para editar los ficheros diarios y notas técnicas (documentación en línea) para elementos de librería, y cada vez que el usuario tiene que introducir un documento de texto.

A.8.1 Edición de comandos

Se utilizan los comandos del menú "**Edición**" para trabajar con el texto editado. La mayoría de estos comandos actúa sobre los caracteres seleccionados en ese momento en el diagrama, o realiza alguna acción en la posición del cursor.

Cortar y pegar

Se puede emplear el botón DEL (Supr) para eliminar el texto seleccionado. Utilizar el comando "**Edición / Deshacer**" para restaurar elementos después de ejecutar un comando DEL. Los comandos "**Cortar**", "**Copiar**", "**Pegar**" del menú "**Edición**" se utilizan para mover o copiar texto dentro del programa, o para insertar bloques de texto copiados al portapapeles desde otras aplicaciones.

Localizar y sustituir

Los comandos de menú "**Edición / Buscar**" y "**Edición / Reemplazar**" se utilizan para localizar y sustituir textos en el programa. Se puede encontrar cualquier cadena de caracteres. La búsqueda puede efectuarse en sentido ascendente o hacia atrás, comenzando en la posición actual del cursor. No se efectúa un bucle cuando se alcanzan los límites del programa.

Ir a la línea

Se utiliza el comando "**Edición / Ir a línea**" para desplazar el cursor hasta un número de línea específico. Esto puede ser muy útil cuando se quiera acceder a una línea que contenga algún error detectado por el compilador ISaGRAF en un programa ST o IL y que esté referenciado por un número de línea.

Insertar símbolo desde el diccionario

Utilizar el comando "**Edición / Insertar variable**" para insertar, en la posición del cursor, el símbolo de una variable o un objeto que se haya declarado en el diccionario del proyecto. El símbolo se selecciona por medio de la ventana de selección de variables comunes, que se describe en el apartado titulado “Más sobre los editores de programas”.

- ***Insertar fichero***

  El comando "**Edición / Insertar Fichero**" inserta el contenido íntegro de un fichero en la posición actual del cursor. Nota: Este comando sólo puede manejar ficheros de texto ASCII puro.

A.8.2 Opciones

Los comandos del menú "**Opciones**" se utilizan para mostrar u ocultar las barras de herramientas del editor y para seleccionar el tipo de letra. Se empleará el tipo de letra seleccionado para cualquier tarea de edición de textos en el banco de trabajo ISaGRAF.

Cuando se utiliza para introducir el código fuente de un programa ST o IL, el comando "**Opciones / Mostrar palabras clave**" se emplea para mostrar u ocultar una caja de herramientas que agrupa a las palabras clave más comunes de los lenguajes ST e IL. Hacer click sobre un botón de la barra de herramientas para insertar las correspondientes palabras claves u operadores en la posición actual del cursor.

A.9 Más sobre los editores de programas

Este apartado contiene información útil sobre las funciones de edición que son comunes a todos los editores de programas ISaGRAF. Tiene que ver principalmente con los enlaces con otras herramientas ISaGRAF y las ventanas de diálogo comunes de ISaGRAF.

A.9.1 Llamadas a otras herramientas ISaGRAF

Verificar (compilar) el programa

El comando "**Fichero / Verificar**" ejecuta el generador de código ISaGRAF para verificar la sintaxis de programación del programa que se está editando en ese momento. En el caso del lenguaje SFC, se comprueban los niveles 1 y 2. Cuando se completa la verificación de sintaxis, la ventana del generador de código se tiene que cerrar para que puede continuar el trabajo con el programa. Si sólo hay un programa en la aplicación (la que se está editando) se genera el código de la aplicación siempre que no se haya detectado un error de sintaxis. Se utiliza el comando "**Opciones / Opciones del compilador**" para fijar los parámetros de compilación y optimización. Para más información sobre compilación y generación de código, véase el apartado titulado "**Cómo utilizar el generador de código**".

Simular o depurar la aplicación

Los comandos "**Fichero / Simulación**" y "**Fichero / Depuración**" ejecutan el depurador gráfico de ISaGRAF bien en modo simulado o bien en modo de conexión real y vuelve a abrir el programa SFC editado en modo depuración. Cuando se utiliza en modo depuración, no se puede introducir modificación alguna en el programa.

Editar el diccionario de variables

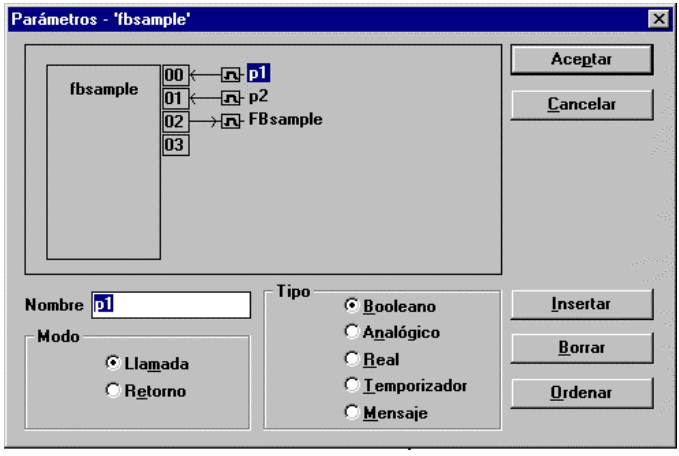
Se utiliza el comando "**Fichero / Diccionario**" para editar el diccionario de variables de la aplicación actual y el programa actual. También contiene los puntos de entrada necesarios para editar las palabras definidas por el usuario. Las declaraciones o palabras definidas **locales** están relacionadas con el programa que se está editando.

A.9.2 Parámetros del programa

Cuando el programa editado es una función, un bloque de función o un subprograma, se utiliza el comando "**Fichero / Parámetros**" para definir sus

parámetros de llamada y retorno. Este comando no tiene efecto si el programa editado es un programa SFC o de nivel superior de las secciones **Comienzo** o **Fin**.

Los subprogramas, las funciones y los bloques de función pueden tener hasta **32** parámetros (de entrada o salida). Las funciones y subprogramas siempre tienen un único parámetro de retorno, que tiene que tener el mismo nombre que la función para poder cumplir con las normas de escritura del lenguaje ST. Se utiliza la siguiente ventana de diálogo para describir los parámetros del subprograma:



La lista situada en la parte superior izquierda de la ventana muestra los parámetros, dispuestos en el orden del modelo de llamada: los parámetros de llamada al principio y los parámetros de retorno al final. La parte inferior de la ventana muestra una descripción detallada del parámetro que en ese momento esté seleccionado en la lista. Se puede utilizar cualquiera de los tipos de datos ISaGRAF para un parámetro. Los parámetros de retorno tienen que estar situados después de los parámetros de llamada en la lista. Los nombres de los parámetros tienen que cumplir con las siguientes normas:

- la longitud del nombre no puede superar los 16 caracteres
- el primer carácter tiene que ser una letra
- los restantes caracteres tienen que ser letras, dígitos o caracteres de subrayado ' _ '
- los nombres son insensibles al uso de mayúsculas o minúsculas

Se utiliza el comando "**Insertar**" para insertar un parámetro nuevo delante del parámetro seleccionado. Se utiliza el comando "**Borrar**" para eliminar el parámetro seleccionado. El comando "**Ordenar**" reordena (clasifica) los parámetros, colocando los parámetros de retorno al final de la lista.

A.9.3 Otros comandos del menú "Fichero"

Los siguientes comandos están disponibles en el menú "**Fichero**" de todos los editores de programas:

Abrir otro programa

El comando "**Fichero / Abrir**" le permite al usuario cerrar el programa que está editando en ese momento y empezar a editar otro programa perteneciente al proyecto actual, con el mismo lenguaje. No se puede utilizar esta función para editar un programa escrito en otro lenguaje. El nuevo programa sustituye al programa actual en la ventana de edición.

Imprimir el programa

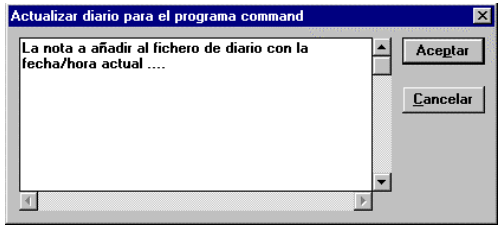
El comando "**Fichero / Imprimir**" da salida al programa editado por impresora. Cuando se editan programas ST o IL, este comando ejecuta el generador de documentos de proyecto para imprimir el programa.

En el caso de los programas gráficos (SFC, FBD y *Quick LD*), también se puede utilizar el comando "**Edición / Copiar imagen (metafile)**" para copiar el dibujo del diagrama al portapapeles en formato *metafile*, para que pueda pegarse en otras aplicaciones tales como procesadores de texto. Para los programas SFC, sólo la información de Nivel 1 (diagrama, numeración y comentarios de Nivel 1) aparece en el *metafile* copiado.

A.9.4 Actualización del diario del programa

El fichero diario asociado al programa editado puede introducirse manualmente por medio del comando "**Fichero / Diario**". El fichero diario se actualiza automáticamente, con los mensajes de salida de chequeo de sintaxis cada vez que se compila el programa. Las salidas de compilación se completan con el sello de fecha / hora de la compilación.

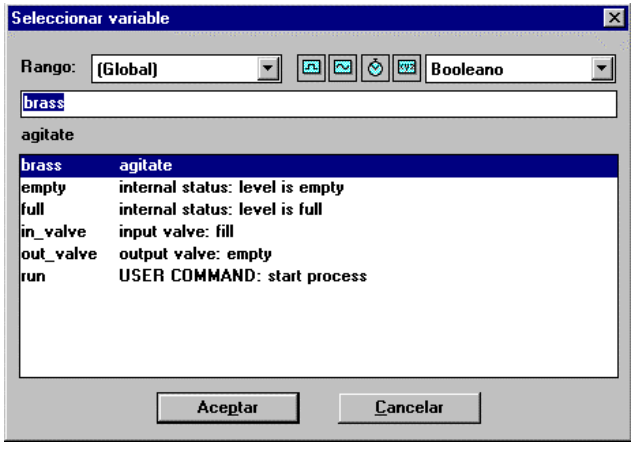
Si se selecciona el modo "**Actualizar diario**" en el menú "**Opciones**" del editor de programas, se abre la siguiente ventana de diálogo cada vez que se guarda el programa en el disco:



Si se pulsa el botón **Aceptar**, la anotación introducida se guarda al final del fichero diario con el sello de fecha / hora actual. Esta función es muy útil para el mantenimiento de programas completos, ya que proporciona una eficaz ayuda sobre el ciclo de vida del programa.

A.9.5 Selección de una variable del diccionario

Cuando se edita un programa de texto (ST o IL), el comando "**Edición / Insertar variable**" permite seleccionar el nombre de la variable declarada que se desea insertar en la posición actual del cursor. Cuando se editan programas LD o FBD, es necesaria la selección de variables para la descripción de contactos, bobinas (salidas), parámetros de bloques de E/S o ventana variables FBD. En ambos casos, se abre la siguiente ventana de diálogo para seleccionar la variable declarada:



Se utiliza el campo de selección "**Visibilidad**" para elegir entre variables globales y locales. El campo de selección situado a la derecha permite elegir el tipo de datos. Los iconos pequeños dispuestos al lado del campo de selección de tipos son botones que pueden utilizarse como aceleradores para seleccionar los tipos de datos más usuales:

 ..... Booleanos
 ..... Enteros / Reales
 ..... Temporizador
 ..... Mensaje

Para seleccionar una variable, hacer click sobre su nombre en la lista. El nombre y el comentario asociado aparecerán en la parte superior de la lista. Pulsar el botón

"**Aceptar**" para confirmar la selección. También se puede introducir el nombre de una variable directamente en el campo de control de edición, sin necesidad de utilizar la lista.

A.9.6 Comandos del menú "Herramientas"

Los siguientes comandos están disponibles el menú Herramientas. Se usan para mostrar la información en una pequeña lista de texto en la parte inferior de la ventana de edición:

Mostrar salida del compilador	saca en la ventana de salida los mensajes de error desde la última compilación (sólo para programas ST/IL)
Buscar en texto	Encuentra las ocurrencias de un texto en todo el texto y las lista en la ventana de salida
Ocultar lista de salida	cierra la ventana de salida de lista

Cuando en la ventana de salida se muestran ocurrencias o mensajes de error, se puede hacer doble click en una línea para hacer directamente visible la selección en la posición correspondiente.

A.10 Utilización del editor de diccionarios

El diccionario ISaGRAF es una herramienta de edición para la declaración de variables internas, variables de E/S, instancias de bloques de función y "palabras definidas" de la aplicación. El diccionario agrupa a las variables declaradas e instancias de bloques de función de la aplicación, junto con las palabras definidas como cadenas constantes.

Las variables, los bloques de función y las palabras definidas deben declararse en el diccionario antes de ser utilizados en el código fuente. Se pueden utilizar las variables y las palabras definidas con cualquiera de los lenguajes de automatización: SFC, FBD, LD, ST e IL. Los bloques de función utilizados en el lenguaje FBD no tienen que ser declarados, ya que los editores FBD y *Quick LD* de ISaGRAF declaran las instancias de los bloques utilizados de forma automática.

Variables

Las variables están clasificadas por **rango** y por **tipo**. Sólo se pueden introducir variables del mismo tipo y rango en una misma plantilla de entrada. Los rangos básicos para variables son:

GLOBALES .....pueden ser utilizadas por cualquier programa del proyecto actual

LOCALES .....pueden ser utilizadas por un único programa

Los tipos básicos de variables son:

BOOLEANAS.....valores binarios verdaderos/falsos

ANALÓGICAS.....valores reales o enteros

TEMPORIZADOR .....valores de tiempo

MENSAJE.....cadenas de caracteres

Una variable está identificada por un nombre, un comentario, un atributo, una dirección de red y por otros campos específicos. A continuación se muestran los atributos básicos de una variable:

INTERNA .....variable de memoria
ENTRADA.....variable asociada a un dispositivo de entrada
SALIDA.....variable asociada a un dispositivo de salida
CONSTANTE .....variable interna de sólo lectura (con valor inicial)

Nota: Los **Temporizadores** siempre son variables **internas**. Las variables de **Entrada** y **Salida** siempre tienen el rango **GLOBAL**.

Palabras definidas

Una 'palabra definida' es un alias que puede utilizarse en cualquier lenguaje para reemplazar una cadena de texto. El texto reemplazado puede ser el nombre de una variable, una expresión constante o una expresión compleja. Se clasifica las

palabras definidas de acuerdo con su rango. Sólo se pueden introducir palabras definidas del mismo tipo y rango en una misma plantilla de entrada. Los rangos básicos son:

COMUNES... pueden ser utilizadas por cualquier programa de cualquier proyecto

GLOBALES . pueden ser utilizadas por cualquier programa del proyecto actual

LOCALES .... pueden ser utilizadas por un único programa

Una palabra definida está identificada por un nombre, un bloque bien definido de equivalencias en texto ST y un comentario libre.

Instancias de bloques de función

Las instancias de los bloques de función utilizados en los lenguajes ST e IL deben declararse en el diccionario. Teniendo en cuenta que los bloques de función tienen datos internos "ocultos", se tiene que identificar cada copia de un bloque de función. En el siguiente ejemplo se muestra el bloque de función "R_TRIG" (detección de flanco de subida), que está definido en el diccionario y que se utiliza para la detección de flancos en diversas variables. Se tiene que identificar cada copia del bloque con un nombre único. Se efectúa la denominación del tipo de bloque y la definición de sus parámetros mediante la utilización del gestor de librerías:

Nombre del bloque: ***R_TRIG***
Parámetros*:* ***Input=CLK***
Output=Q

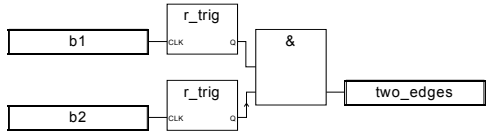
Se lleva a cabo la denominación de instancias mediante la utilización del editor de diccionarios:

Nombre instancia*: TRIG_B1* **Nombre bloque*: R_TRIG***
Nombre instancia*: TRIG_B2* **Nombre bloque*: R_TRIG***

Se pueden utilizar las instancias declaradas en programas ST:

```
TRIG_B1 (b1);
edge_b1 := TRIG_B1.Q;  (* b1 variable detección de flanco *)
TRIG_B2 (b2);
edge_b2 := TRIG_B2.Q;  (* b2 variable detección de flanco *)
```

Las instancias declaradas de bloques de función pueden ser **GLOBALES** (conocidas por cualquier programa del proyecto) o **LOCALES** para un único programa. Los bloques de función utilizados en los lenguajes FBD y LD no tienen que ser declarados, ya que los editores FBD de ISaGRAF declaran las instancias de los bloques utilizados de forma automática.



(* los bloques de función siempre tienen el nombre del bloque que se haya definido en la librería. Los editores FBD y *Quick LD* de ISaGRAF declaran una instancia automáticamente cada vez que se inserta un bloque en el diagrama *)

Las instancias de bloques de función declaradas por los editores FBD y *Quick LD* siempre son **LOCALES** para el programa editado.

Direcciones de red

Las direcciones de red son **opcionales**. Una variable con una dirección de red 'no cero' puede ser **espiada** por un sistema externo (por ejemplo, un sistema de visualización de procesos) durante el tiempo de proceso. En términos más generales, la dirección de red facilita un mecanismo de identificación para cada sistema de comunicaciones que no puede manejar nombres simbólicos. Se puede introducir una dirección de red para cada variable, durante su descripción completa, cuando se crea o se modifica una variable.

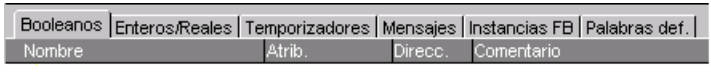
A.10.1 Ventana principal del diccionario

La ventana de edición muestra únicamente los campos principales de descripción de variables: nombre, atributo y dirección de red, junto con un comentario de texto. La descripción completa de la variable seleccionada está presente en todo momento en la barra de estado.

Utilizar los siguientes botones de la barra de herramientas para seleccionar el rango de la variable que se desea editar:

.**COMUNES**…….....los puede utilizar cualquier programa de cualquier proyecto
.**GLOBALES**……...los puede utilizar cualquier programa del proyecto actual
.**LOCALES**……...los puede utilizar un único programa

Utilizar el control de tabulación desplegado con la barra de títulos para seleccionar el tipo de objeto que se desea editar:



Utilizar el campo de introducción de textos situado a la izquierda de la barra de herramientas para buscar un prefijo de nombre de variable. En este caso,

se efectúa la búsqueda en toda la lista, desde el principio, con base en la selección actual. También se dispone de los comandos del menú "**Edición / Buscar**" para buscar una cadena de texto en nombres variables y comentarios, y desplazar la selección a esta variable. La búsqueda es siempre **insensible** al uso de mayúsculas o minúsculas.

A.10.2 Gestión de variables

Los comandos del menú "**Fichero**" actúan sobre toda la clase seleccionada de variables, instancias de bloques de función o palabras definidas. Utilizar el comando "**Otro**" para seleccionar el tipo y rango de los objetos que se desea editar.

Impresión de variables

Utilizar el comando "**Fichero / Imprimir**" para imprimir la lista que se está editando de variables o palabras definidas, en una impresora estándar de Windows™. La impresión se realiza utilizando el generador de documentos de ISaGRAF. Se incluye la descripción completa de cada variable o palabra definida del grupo que se está editando en ese momento.

Creación de variables nuevas

El comando "**Edición / Nuevo**" le permite al usuario la creación de nuevas variables, instancias de bloques de función o palabras definidas para el rango y tipo seleccionados. Se insertan las variables nuevas justo delante de la variable señalada por la barra de selección. Cuando se ejecuta este comando, se abre una ventana para introducir la descripción de la variable. Cuando se completa la descripción, se pulsa el botón "**Almacenar**" para incorporarla en la lista. Se vuelve a abrir la ventana de entrada de forma automática para que el usuario pueda introducir otras variables con el mismo comando "**Edición**". Al pulsar el botón "**Cancelar**" de la ventana de diálogo, se interrumpe el proceso de creación de variables.

Modificación de variables existentes

El comando "**Editar**" del menú "**Edición**" le permite al usuario modificar la descripción de la variable que en ese momento está señalando la barra de selección. Cuando se ejecuta este comando, se abre una ventana de introducción de textos para modificar la descripción de la variable. Cuando se completa la descripción, se pulsa el botón "**Almacenar**" para habilitar la modificación. El usuario también puede pulsar los botones "**Siguiente**" y "**Anterior**" para hacer extensivo el comando de modificación a las variables adjuntas. Para cerrar la ventana de diálogo sin guardar las modificaciones, pulsar el botón "**Cancelar**".

Cortar y pegar

La herramienta de edición de diccionario de ISaGRAF permite la **selección de líneas múltiples**. Existen muchos comandos para trabajar con la lista de variables que se está editando en un momento dado. Se dispone de los siguientes comandos del menú "**Edición**":

COPIAR ..... Copia el grupo seleccionado de variables al portapapeles del diccionario
CORTAR ..... Copia el grupo seleccionado de variables y lo elimina de la lista editada
PEGAR ..... Insertar el contenido del portapapeles del diccionario delante de la variable seleccionada.

Se pueden utilizar las funciones de Copiar/Cortar/Pegar entre una lista de variables y otra. No se pueden utilizar entre listas compuestas por tipos de objetos diferentes.

Clasificación de variables

El comando "**Herramientas / Ordenar**" clasifica las variables o palabras definidas de la lista que en ese momento se está editando. El orden de clasificación está determinado por los atributos de las variables:

- primero, las variables internas
- después, las variables de entrada
- por último, las variables de salida

Las variables con el mismo atributo están clasificadas en orden alfabético. Las palabras definidas siempre están clasificadas en orden alfabético.

Establecimiento de direcciones de red

Las direcciones de red son **opcionales**. Las variables con direcciones de red 'no cero' pueden ser **espiadas** por un sistema externo (por ejemplo, un sistema de visualización de procesos) durante el tiempo de proceso. Se puede introducir una dirección de red para cada variable, durante el proceso de descripción completa, cuando se crea o se modifica una variable. El comando "**Herramientas / Renumerar direcciones**" le permite al usuario definir las direcciones de red de un grupo entero de variables. Cuando se ejecuta este comando, actúa sobre el grupo de variables que está seleccionado en la lista. La introducción de una **dirección básica** hexadecimal (dirección para la primera variable del grupo) resulta en la definición de **direcciones de red consecutivas** para todas las variables del grupo. La introducción de una dirección básica nula restablece en cero las direcciones de red de todas las variables seleccionadas.

Importación de cadenas booleanas "verdadero/falso"

Cuando se editan palabras definidas, el comando "**Herramientas / Importar definiciones verdadero/falso**" le permite al usuario definir automáticamente como palabras clave del lenguaje a las cadenas asociadas a las variables booleanas para representar estados TRUE y FALSE. Normalmente se define este tipo de cadena para formatear la depuración. Tienen que estar especificadas como palabras definidas si se van a utilizar en un programa. Este comando realiza una búsqueda de cadenas booleanas verdadero/falso en las declaraciones con el mismo rango que aquella que está seleccionada en la actualidad para la edición de las palabras definidas.

A.10.3 Descripción de objetos

Se introduce una descripción completa para cada variable, instancia de bloque de función o palabra definida. Los campos de descripción son distintos para cada tipo de objeto. Los siguientes campos son comunes para cualquier tipo de variable:

Nombre .....Nombre de la variable: el primer carácter tiene que ser una letra y los restantes caracteres tienen que ser letras, dígitos o el símbolo '_'.
Dirección de red .....Dirección de red hexadecimal (opcional). Cuando este campo no equivale a cero, la variable puede ser espiada por sistemas externos durante el tiempo de proceso.
Comentario .....Comentario libre para la descripción de variables.
Retener.....Esta opción indica que se tiene que guardar la variable en la memoria de seguridad.

Estos son otros campos de descripción para una variable **booleana**:

Atributo .....Especifica una variable interna, constante, de entrada o de salida.
Cadena "**Falso**".....Cadena utilizada para el valor falso durante la depuración.
Cadena "**verdadero**".....Cadena utilizada para el valor falso durante la depuración.
Establecer "**verdadero**"
al inicio.....El valor inicial es TRUE si se selecciona esta opción. En caso contrario, el valor inicial es FALSE.

Estos son otros campos de descripción para una variable analógica **entera o real**:

Atributo .....Especifica una variable interna, constante, de entrada o de salida.
Formato.....Especifica una variable entera o real (flotante). Se puede seleccionar el formato de visualización que se vaya a utilizar durante el proceso de depuración.
Cadena unidad .....La cadena utilizada para identificar a la unidad física durante el proceso de depuración.
Conversión .....El nombre de la tabla de conversión o la función de conversión asociada a la variable (sólo para variables de entrada o salida).
Valor inicial.....Valor inicial de la variable (tiene que tener el mismo formato que la variable). Si no se especifica lo contrario, el valor inicial es 0.

Estos son otros campos de descripción para una variable **temporizador**:

Atributo .....Especifica una variable interna o constante.
Valor inicial.....Valor inicial de la variable (valor de tiempo). Si no se especifica lo contrario, el valor inicial es **time#0s**.

Estos son otros campos de descripción para una variable **mensaje**:

Atributo .....Especifica una variable interna, constante, de entrada o de salida.
Longitud máx......Especifica el número máximo de caracteres que pueden almacenarse en el mensaje.
Valor inicial.....Valor inicial de la variable (la longitud no puede superar la capacidad del mensaje). Si no se especifica lo contrario, el valor inicial es una cadena vacía.

Los campos de descripción para una **palabra definida** son:

Nombre .....El nombre utilizado en los ficheros fuente ST: el primer carácter tiene que ser una letra, los siguientes caracteres tienen que ser letras, dígitos o el símbolo '_'.
Definición.....Cadena de acuerdo con la sintaxis ST, que sustituye la palabra definida durante la compilación. Ejemplo: Nombre = PI - Equivalencia = 3.14159
Comentario .....Comentario libre para la equivalencia definida

Los campos de descripción para una **instancia de bloque de función** son:

Nombre .....Nombre de la instancia, utilizado en ficheros fuente ST: el primer carácter tiene que ser una letra, los siguientes caracteres tienen que ser letras, dígitos o el símbolo '_'.
Tipo .....Nombre del correspondiente bloque de función en la librería.
Comentario .....Comentario libre para la instancia del bloque de función.

A.10.4 Declaración rápida

El comando "**Herramientas / Declaración rápida**" permite declarar varias variables al mismo tiempo. Las variables creadas mediante declaración rápida se nombran usando una convención numérica. Para eso hay que definir:

- El índice (numero) de la primera y de la última variable,
- El texto a añadir, antes y después del número, en caracteres de la variable
- El número de dígitos que se utilizan para expresar el número en caracteres de la variable

Además se pueden especificar atributos básicos de las variables creadas (internas, entrada o salida…), y algunas propiedades dependiendo del tipo de variable (atributo de "Retención", formato entero o real, longitud máxima de la cadena de mensaje).

Siempre se necesita definir un texto para insertar antes de un número de variable, ya que el símbolo de una variable no puede empezar con un dígito. Cuando el

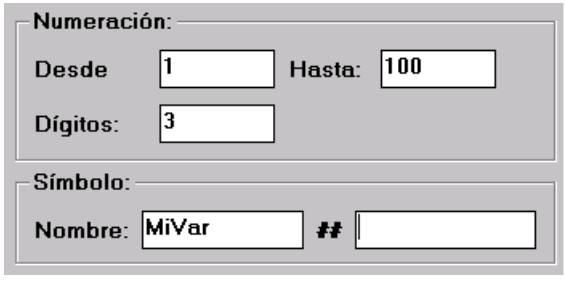
número de dígitos se configura a "Auto", ISaGRAF formatea el número de variable al mínimo número de dígitos necesario. Cuando se especifica el número de dígitos, ISaGRAF formatea todos los números a la longitud especificada añadiendo caracteres '0' por delante. Configurar un número fijo de dígitos para los números de variable puede ser muy útil para prevenir contra una mala clasificación alfabética. Aquí están algunos ejemplos:

Ejemplo 1: Esta configuración para declaración rápida:

creará las siguientes tres variables:

Var9xx Var10xx Var11xx

Ejemplo 2: Esta configuración para declaración rápida:



creará 100 variables con nombres desde MiVar001 a MiVar100.

A.10.5 Mapa de direcciones Modbus SCADA

Las "direcciones de red" de ISaGRAF se usan a menudo para establecer enlaces entre el sistema ISaGRAF y un SCADA basado en una comunicación MODBUS. EN este caso, el SCADA es un MODBUS maestro y el sistema ISaGRAF es un Modbus esclavo. Las direcciones de red se usan para crear un mapa virtual Modbus para todas las variables ISaGRAF que deben ser controladas por el SCADA. El comando "**Herramientas / Mapa de direcciones Modbus SCADA**" es

muy potente para crear rápidamente un mapa virtual Modbus con las variables de la aplicación.

Las herramientas de mapeo muestran dos listas. La superior es un segmento del mapa Modbus (4096 posiciones), con las variables mapeadas (las que tienen dirección de red). La lista inferior muestra las variables no mapeadas (sin dirección de red definida). la dirección "0" no puede ser usada para mapear una variable.

Utilizar los comandos "**Mapear**" y "**Eliminar**" del menú "**Edición**" para mover una variable de una lista a otra, y de este modo construir el mapa. las mismas acciones pueden realizarse haciendo doble click en un símbolo de una variable de la lista, para mandarla a la otra lista. En cualquier momento se puede desplegar la lista "Segmento" para ver otro segmento del mapa.

Los comandos del menú "**Opciones**" se pueden utilizar en cualquier momento para ver las direcciones en decimal o en hexadecimal.

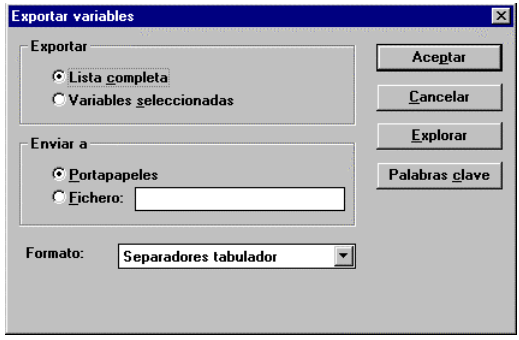
Los comandos "**Edición / Buscar**" se usan para buscar una variable declarada, ya esté mapeada o no.

A.10.6 Intercambio de información con otras aplicaciones

La herramienta de edición del diccionario de ISaGRAF incluye funciones de importación/exportación que permiten el intercambio de información con otras aplicaciones, tales como procesadores de texto, hojas de cálculo, gestores de bases de datos, etc. Estos comandos están agrupados en el menú "**Edición**". El comando "**Exportar texto**" crea una descripción en texto ASCII puro de los campos que describen un conjunto de objetos editados, y almacena este texto bien en el portapapeles de Windows o bien en un fichero. Por lo general, la información de este tipo es utilizada por otras aplicaciones. El comando "**Importar texto**" importa campos de descripción de declaraciones variables, descritos en formato de texto ASCII puro y almacenados bien en el portapapeles de Windows o bien en un fichero, y actualiza la lista que en ese momento se está editando con los campos importados. Por lo general, la información de este tipo está producida por otra aplicación.

Exportación de datos

Aparece la siguiente ventana de diálogo cuando se ejecuta el comando "**Exportar texto**". Le permite al usuario controlar el mecanismo de exportación.



Marcando la opción "**Lista completa**" indica que se tiene que exportar la lista editada completa. En este caso, se ignora la selección actual. Marcando la opción "**Variables seleccionadas**" indica que sólo se exportarán las variables seleccionadas.

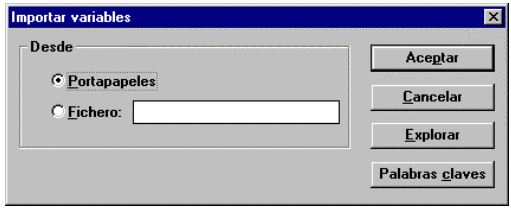
Al seleccionar la opción "**Portapapeles**", se guarda la información exportada en el portapapeles de Windows en formato de texto ASCII puro, quedando a disposición de los comandos de "pegar" de otras aplicaciones. Al seleccionar la opción "**Fichero**", el texto exportado se guarda en un fichero ASCII. Tiene que introducirse el nombre completo de la ruta de acceso de este fichero. Se puede utilizar el comando "**Explorar**" para encontrar un nombre de ruta existente.

Seguidamente, el usuario elige un formato para el texto exportado. Los formatos disponibles están descritos en secciones posteriores. Pulsar el botón "**Aceptar**" para ejecutar la función de exportación. Pulsar el botón "**Cancelar**" para cerrar la ventana de diálogo y salir del comando de exportación.

Todos los campos de los objetos seleccionados están almacenados en el texto exportado, en el orden estándar de declaración. La primera línea del texto exportado contiene los nombres de los campos. Cada objeto se describe en una línea de texto. El separador de "fin de línea" es la secuencia estándar de MS-DOS "**0d-0a**". Se pueden cambiar los nombres que se emplean para identificar los campos de la primera línea exportada, pulsando el botón "**Palabras clave**". Se describe este comando en secciones posteriores.

Importación de datos

Aparece la siguiente ventana de diálogo cuando se ejecuta el comando de "**Importar texto**". Permite al usuario controlar el mecanismo de importación.



Si se selecciona la opción "**Portapapeles**", la información importada se toma del portapapeles de Windows, en formato de texto ASCII puro. Si se selecciona la opción "**Fichero**", se lee el texto exportado en un fichero ASCII. Se tiene que introducir la ruta de acceso completa del fichero. Se puede utilizar el comando "**Explorar**" para localizar un nombre de ruta de acceso existente.

La función de importación reconoce automáticamente el formato (los separadores) que se utiliza en el texto importado. Los formatos disponibles están descritos en secciones posteriores. Pulsar el botón "**Aceptar**" para ejecutar la función de importación. Pulsar el botón "**Cancelar**" para cerrar la ventana de diálogo y salir del comando de importación. Se pueden cambiar los nombres que se emplean para identificar los campos de la primera línea importada, pulsando el botón "**Palabras clave**". Se describe este comando en secciones posteriores.

La primera línea del texto tiene que contener los nombres de los campos, de acuerdo con el orden que se utilice para las líneas posteriores. Cada objeto se describe en una línea de texto. El separador de "fin de línea" es la secuencia estándar de MS-DOS "**0d-0a**". Los campos pueden aparecer en cualquier orden. En el caso de que falten campos, se rellenan automáticamente en la descripción de objeto importada con valores por defecto. Si un objeto importado ya existe en la lista editada, el usuario tiene que confirmar su sobrescritura y se actualiza la descripción del objeto con los campos importados. En el caso de que falten campos, no se actualizan en la descripción del objeto.

Formatos de texto disponibles

A continuación aparece una lista de los formatos que están disponibles para el comando de exportación. El comando de importación reconoce estos formatos automáticamente.

- separación por tabuladores

Descripción: ***Los campos se separan mediante caracteres de tabulación.***

Ejemplo:	***Nombre***	***Atributo***	***Comentario***
	nivel	***interno***	***cálculo interno nivel de agua***
	alrm1	***salida***	***salida de alarma principal***

- separación por comas

Descripción: ***Los campos se separan mediante comas.***

Ejemplo: ***Nombre,Atributo,Comentario***
nivel,interno,cálculo interno nivel de agua
alrm1,salida,salida de alarma principal

- separación por punto y coma

Descripción: ***Los campos de separan mediante puntos y comas.***

Ejemplo: ***Nombre;Atributo;Comentario***
nivel;interno;cálculo interno nivel de agua
alrm1;salida;salida de alarma principal

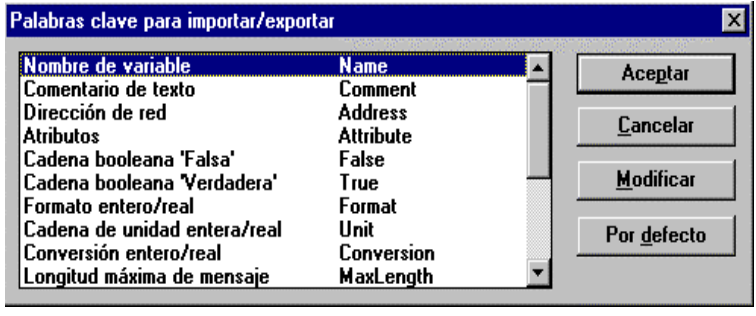
- comas y comillas

Descripción: ***Los campos están separados por comas.***
Cada campo aparece entre comillas.

Ejemplo: ***"Nombre","Atributo","Comentario"***
"nivel","interno","cálculo interno nivel de agua "
"alrm1","salida"," salida de alarma principal"

Palabras clave

Para cambiar los nombres utilizados para identificar los campos de la primera línea importada o exportada, pulsar el botón "**Palabras clave**". Este comando abre la siguiente ventana de diálogo:



La ventana muestra la lista de campos de objetos y las palabras clave asociadas. Para modificar una palabra clave, el usuario tiene que seleccionar un campo de la lista y pulsar el botón "**Modificar**". Pulsar el botón "**Por defecto**" para restaurar la lista original de palabras clave. Los nombres de las palabras clave tienen que cumplir con las siguientes normas:

- el nombre no puede superar los **16** caracteres
- el primer carácter tiene que ser una **letra**
- los restantes caracteres pueden ser **letras**, **dígitos** o el símbolo '_'
- no se puede usar el mismo nombre para palabras clave diferentes

Las palabras clave estándares utilizadas en ISaGRAF son:

Nombre de objeto..... **Name**
Comentario de texto..... **Comment**
Dirección de red..... **Address**
Atributos (interno, entrada, salida)..... **Attribute**
Cadena booleana 'Falso'..... **False**
Cadena booleana 'Verdadero'..... **True**
Formato analógico (real o entero)..... **Format**
Cadena unidad analógica..... **Unit**
Nombre conversión analógica..... **Conversion**
Longitud máxima mensaje..... **MaxLength**
Bloque funciones, tipo librería..... **Library**
Equivalencia palabra definida..... **Equivalence**
Atributo interno..... **Internal**
Atributo de entrada..... **Input**
Atributo de salida..... **Output**
Atributo constante..... **Constant**
Formato analógico real..... **Real**
Formato valor analógico entero..... **Integer**

A.11 Utilización del editor de conexiones de E/S

El objetivo de la operación de conexión de E/S es el de establecer un enlace lógico entre las variables de E/S de la aplicación y los canales físicos de las tarjetas que están presentes en el equipo objeto. Para realizar este enlace, el usuario tiene que identificar y configurar todas las tarjetas del equipo objeto y situar las variables de E/S en los correspondientes canales de E/S.
La lista de la izquierda muestra el chasis del equipo objeto, con sus **ranuras de tarjeta**. Una ranura puede estar libre, o puede estar utilizada por una tarjeta de E/S o un equipo complejo. Se identifica a cada ranura con un **número de orden**. Un chasis puede contener hasta **255** tarjetas. La lista de la derecha muestra los parámetros de la tarjeta y las variables que están conectadas a la tarjeta seleccionada. Una tarjeta puede tener hasta **128** canales de E/S. El número total de tarjetas de E/S (incluyendo equipos y tarjetas de equipo complejo) no puede superar las **255**.

Iconos

Los iconos que se exhiben en el frontal indican el tipo y los atributos de las variables que se pueden conectar a los canales de la tarjeta. El sistema ISaGRAF no permite la conexión de variables de diferentes tipos en la misma tarjeta. A continuación se indica el significado de los iconos utilizados:

- ..... tipo booleano
- ..... tipo datos enteros/reales (pueden conectarse ambos tipos de variables)
- ..... tipo mensaje
- ..... entradas - sin conectar canal
- ..... salidas - sin conectar canal
- ..... entradas - al menos un canal conectado
- ..... salidas - al menos un canal conectado

A continuación se muestran los iconos utilizados para mostrar el tipo de dispositivo de E/S que está instalado en una ranura:

- ..... equipo complejo de E/S
- ..... tarjeta de E/S real
- ..... tarjeta de E/S virtual

Se utilizan los siguientes iconos para dibujar un parámetro o un canal:

- ..... parámetro de tarjeta
- ..... canal libre
- ..... canal conectado

Desplazamiento de tarjetas en la lista

Utilizar estos botones de la barra de herramientas o los comandos de menú "**Edición / Subir tarjeta / Bajar tarjeta**" para mover la tarjeta de E/S seleccionada

una línea hacia arriba o una hacia abajo en la lista principal. El comando "**Edición / Insertar ranura**" inserta una ranura vacía en la posición actual.

A.11.1 Definición de tarjetas de E/S

El menú "**Edición**" contiene los comandos necesarios para definir la tarjeta seleccionada (configurar sus parámetros) y conectar las variables de E/S a sus canales.
El menú de "**Herramientas**" contiene otros comandos útiles para trabajar en las tarjetas seleccionadas.

Selección del tipo de tarjeta de E/S

Antes de conectar las variables de E/S a una tarjeta, se tiene que introducir la identificación de la tarjeta. El banco de trabajo ISaGRAF dispone de una librería de tarjetas predefinidas. Puede que esta librería esté compilada por uno o más proveedores de dispositivos de E/S. Se utiliza el comando "**Edición / Definir Tarjeta / Equipo**" para establecer la identificación de la tarjeta. Se puede utilizar este comando para seleccionar tanto una tarjeta simple como un equipo complejo de E/S de la librería ISaGRAF. También se puede hacer doble click sobre una ranura para configurar la ranura o equipo correspondiente. Todos los canales de una tarjeta simple son del mismo tipo (booleanos, enteros/reales o de mensajes) y dirección (entradas o salidas). No se distingue entre las variables reales y enteras durante la conexión de E/S. Un equipo complejo de E/S representa un dispositivo de E/S con canales de diferentes tipos o direcciones. Los equipos complejos de E/S están representados como una lista de tarjetas simples de E/S. Sólo ocupa una ranura en la lista del chasis.

Eliminación de una tarjeta

Se utiliza el comando "**Edición / Borrar ranura**" para eliminar la tarjeta o equipo de E/S seleccionado. Si ya se hubieran conectado variables a los canales correspondientes, se desconectan automáticamente al eliminar la ranura.

Tarjetas reales y tarjetas virtuales

El comando "**Edición / Tarjeta real/virtual**" establece la validez de la tarjeta o equipo complejo de E/S seleccionado. Se exhiben los siguientes iconos en la lista del *chasis* para mostrar la validez de una tarjeta:
 ..... tarjeta de E/S real
 ..... tarjeta de E/S virtual

En el **Modo Real**, las variables de E/S se enlazan directamente con los dispositivos correspondientes. Las operaciones de entrada o salida del programa de aplicación se vinculan directamente con las correspondientes condiciones de entrada o salida de los dispositivos E/S de campo. En el **Modo Virtual**, la variables de E/S se procesan de la misma manera que las variables internas. Pueden ser leídas y actualizadas por el depurador para que el usuario pueda simular el procesamiento de E/S, aunque no se realiza conexión alguna con el mundo real.

Notas técnicas

El comando "**Herramientas / Nota técnica**" muestra la guía en línea del usuario para la tarjeta o el equipo complejo seleccionado. Las notas técnicas de la tarjeta están redactadas por el proveedor de la tarjeta de E/S. Estas instrucciones contienen una información completa sobre la gestión de la tarjeta de E/S. También explica el significado de sus parámetros.

Desconexión de variables conectadas

El comando "**Herramientas / Liberar canales de tarjeta**" desconecta todas las variables de E/S que ya estén conectadas a la tarjeta seleccionada.

Comentarios

El texto de comentario asociado a una variable de E/S declarada se muestra, con su nombre, en la lista de tarjetas. Ya que ISaGRAF permite la utilización de variables de representación directa (notación %), también se pueden asociar comentarios a los canales libres. Utilizar el comando "**Herramientas / Definir comentario de canal**" para introducir un comentario para el canal libre que está seleccionado en la lista de tarjetas. Este comando no puede ser utilizado para cambiar el comentario asociado a una variable de E/S que se haya declarado en el diccionario del proyecto.

A.11.2 Definir parámetros de tarjeta

Para definir el valor de un parámetro de tarjeta, el usuario tiene que hacer doble click sobre su nombre en la lista que aparece a la derecha. También lo puede seleccionar (resaltar) y elegir el comando "**Definir canal/parámetro**" del menú "**Edición**". Los parámetros aparecen ordenados al principio de la lista. Se utiliza el siguiente icono para representarlos en la lista:
..... parámetro de tarjeta

El proveedor de la correspondiente tarjeta o equipo de E/S determina el significado y el formato de entrada del parámetro. Para mayor información sobre parámetros de tarjetas, utilizar el comando "**Herramientas / Nota técnica**" o véase el manual del *hardware.*

A.11.3 Conexión de canales E/S

Para establecer la conexión de un canal, el usuario tiene que hacer doble click sobre su posición en la lista situada a la derecha. También existe la posibilidad de seleccionarlo (destacarlo) y ejecutar el comando "**Edición / Definir canal/parámetro**". Se utilizan los siguientes iconos para representar los canales de la lista:
..... canal libre
..... canal conectado

La lista contiene todas las variables que coincidan con el tipo y la dirección de la tarjeta seleccionada. Sólo aparecen en esta lista las variables que todavía no han sido conectadas. El botón "**Conectar**" conecta la variable seleccionada de la lista con el canal seleccionado. El botón "**Liberar**" elimina (desconecta) la variable del canal seleccionado. Los botones "**Siguiente**" y "**Anterior**" se utilizan para seleccionar otro canal de la tarjeta. La ubicación del canal seleccionado se exhibe siempre en la barra de título de la ventana de diálogo.

A.11.4 Variables de representación directa

Los canales libres son aquellos que no están vinculados a una variable de E/S declarada. ISaGRAF permite el uso de **variables de representación directa** en las fuentes de los programas para representar a los canales libres. El identificador de una variable de representación directa siempre empieza por un carácter **"%"**.

A continuación se indican las normas de denominación de una variable de representación directa para un canal de una tarjeta simple. "***s***" es el número de ranura de la tarjeta. "***c***" es el número del canal.

%IX*s.c* ..... canal libre en tarjeta de entrada booleana
%ID*s.c* ..... canal libre en tarjeta de entrada de valores enteros
%IS*s.c* ..... canal libre en tarjeta de entrada de mensajes
%QX*s.c*..... canal libre en tarjeta de salida booleana
%QD*s.c* ..... canal libre en tarjeta de salida de valores enteros
%QS*s.c*..... canal libre en tarjeta de salida de mensajes

A continuación se muestran las normas de denominación de una variable de representación directa para un canal de un equipo complejo. "***s***" es el número de ranura del equipo. "***b***" es el índice de la tarjeta simple dentro del equipo complejo. "***c***" es el número del canal.

%IX*s.b.c* ..... canal libre en tarjeta de entrada booleana
%ID*s.b.c* ..... canal libre en tarjeta de entrada de valores enteros
%IS*s.b.c* ..... canal libre en tarjeta de entrada de mensajes
%QX*s.b.c* .... canal libre en tarjeta de salida booleana
%QD*s.b.c* .... canal libre en tarjeta de salida de valores enteros
%QS*s.b.c* .... canal libre en tarjeta de salida de mensajes

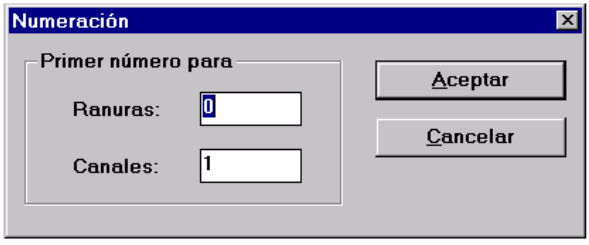
Algunos ejemplos:

%QX1.6 6º canal de la tarjeta nº 1 (salida booleana)
%ID2.1.7 7º canal de la tarjeta nº 1 del equipo nº 2 (entrada de valores enteros)

Una variable de representación directa no puede ser del tipo de datos "**real**".

A.11.5 Numeración

Use el comando "**Opciones / Numeración**" para configurar los convenios de numeración. Se puede especificar el número utilizado para el primer slot y el número utilizado para el primer canal de cada tarjeta en el siguiente cuadro de diálogo:



Por defecto la numeración de slots comienza con el índice **0** y la numeración de canales con el índice **1**-

Advertencia: Sea muy cuidadoso cuando cambie los convenios de numeración al tener efecto en los símbolos usados para las variables de representación directa y poder llevar a errores de compilación si las variables E/S de representación directa se usan en los programas.

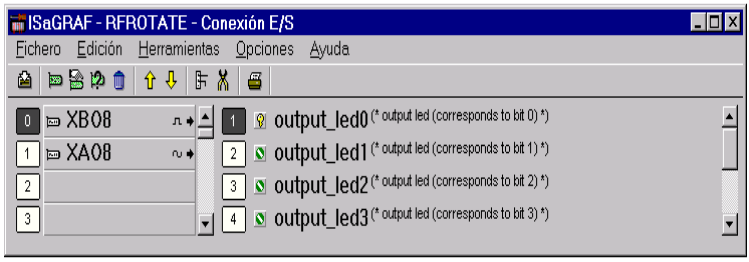
A.11.6 Definir protecciones individuales

El banco de trabajo de ISaGRAF proporciona una protección de datos completa basada en una jerarquía de contraseñas. Las conexiones E/S pueden protegerse globalmente mediante una contraseña. Además ISaGRAF permite establecer protecciones individuales a cualquier canal de E/S. Esto supone que:

- Las contraseñas ya están definidas en el sistema de definición de contraseñas (usar el comando "**Proyecto / Establecer contraseña**" de la ventana del gestor de proyectos) de modo que los niveles de protección estén disponibles para protecciones individuales.

- Se usan niveles de protección con mayor nivel de prioridad para protecciones individuales en comparación con la protección global de E/S.

Cuando un canal de E/S tiene una protección individual, aparece un pequeño icono junto a su nombre en la ventana de conexiones E/S.



Use el comando "**Establecer protección**" y "**Eliminar protección**" del menú de "**Editar**" para configurar o eliminar una protección individual para el canal seleccionado. Ambos comandos preguntaran por una contraseña valida de modo que un nivel de protección pueda asociarse al canal. Después, cada vez que se quiera cambiar la condición de un canal con una protección individual se debe introducir una contraseña con suficiente nivel de prioridad.

Advertencia: si un canal se protege con un determinado nivel y la contraseña correspondiente se elimina del sistema de protección, y no hay definida ninguna contraseña de nivel superior la conexión al canal no puede ser cambiada nunca más a menos que se defina una nueva contraseña con suficiente nivel.

A.12 Creación de tablas de conversión

El banco de trabajo ISaGRAF permite al usuario la creación de tablas de conversión. Una tabla de conversión es un conjunto de puntos que se utiliza para definir una conversión analógica. Se puede asociar una tabla de conversión a una variable analógica de entrada o salida. Una tabla crea una relación proporcional entre los valores eléctricos (leídos en el sensor de entradas o enviados al dispositivo de salida) y físicos (utilizados en la programación de la aplicación).
Las tablas de conversión se editan a través de un cuadro de diálogo usando el comando "**Herramientas / Tablas de conversión**" en la ventana del diccionario de ISaGRAF.
Se puede utilizar una tabla de conversión definida para filtrar los valores de cualquier variable analógica de entrada o salida del proyecto seleccionado. Para asociar una tabla de conversión a una variable, se hace utilizando los comandos del diccionario de ISaGRAF, el editor de declaración de variables. Después, se selecciona una variable analógica de entrada o salida y se editan sus parámetros. Una variable no puede estar asociada a una tabla de conversión que todavía no está definida.

A.12.1 Comandos principales

La ventana de diálogo de "**Tablas de conversión**" muestra la lista de las tablas de conversión definida para los comandos principales, para editar una tabla existente (definir sus puntos), para crear una nueva tabla, y también para renombrar o borrar una tabla.
Presione Aceptar para salir del cuadro de diálogo de "**Tablas de conversión**" y guárdelo en el disco.

Creación de una tabla nueva

El comando "**Nueva**" le permite al usuario la creación de una tabla de conversión nueva. Se pueden crear hasta **127** tablas de conversión para cada proyecto. Sólo se introducen las tablas que están utilizadas (aquellas que están vinculadas a variables analógicas) en el código ejecutable de la aplicación. Los nombres de las tablas tienen que cumplir con las siguientes normas:

- La longitud del nombre no puede superar los **16** caracteres
- El primer carácter tiene que ser una **letra**
- Los restantes caracteres deben ser **letras**, **dígitos** o caracteres de subrayado '**_**'
- Los nombres no son sensibles al uso de mayúsculas o minúsculas

Modificación del contenido de una tabla

Se utiliza el comando "**Editar**" para introducir los puntos de una tabla seleccionada de la lista. También existe la posibilidad de hacer doble click sobre el nombre de la

tabla. Se llama al comando "**Editar**" automáticamente cuando se crea una tabla nueva. Se deben introducir al menos dos puntos por cada tabla.

A.12.2 Introducción de puntos en una tabla

La ventana de diálogo "**Editar**" le permite al usuario definir los puntos de una tabla de conversión. A la izquierda de la ventana se muestra una lista de los puntos que ya han sido definidos La subventana inferior derecha muestra la tabla definida en forma de curva gráfica. Para introducir los puntos, se utilizan los comandos de la ventana. El usuario tiene que cumplir con las normas numéricas que afectan a la definición de puntos y que se describen al final del presente apartado. La sección de la izquierda siempre contiene la lista de puntos que existen para la tabla que se está editando. La columna de la izquierda muestra el valor eléctrico (externo) de los puntos. La columna de la derecha muestra los valores físicos (internos). El usuario tiene que seleccionar un punto de la lista para poder modificar sus valores o eliminarlo. Se utiliza la última selección de la lista ("**... ...**") para definir un punto nuevo. La sección inferior derecha muestra la tabla que se está editando en forma de curva gráfica. No se indican los ejes ni las coordenadas, ya que se trata de una representación proporcional de la curva. Esta representación es útil para efectuar una comprobación rápida de que la curva está correctamente definida.

- ***Definición de puntos nuevos***

  Para definir un punto nuevo, seleccionar la última posición ("**... ...**") de la lista de puntos. Este también es el modo por defecto cuando se empieza a definir una tabla de conversión nueva. El usuario tiene que introducir los valores eléctricos (externos) y físicos (internos) de cada punto. Se almacenan los valores como valores de coma flotante de simple precisión. No debe olvidarse que se tienen que introducir al menos **dos puntos** para definir una curva. Cuando ambos valores están introducidos, pulsar el botón "**Almacenar**" para añadir el punto a la tabla. Se puede definir un máximo de **32** puntos por cada tabla de conversión.

- ***Modificación de un punto***

  Para modificar los valores de un punto existente, el primer paso es seleccionarlo de la lista. Seguidamente, pueden introducirse los nuevos valores eléctricos (externos) y físicos (internos) el punto. Se almacenan los valores como valores de coma flotante de simple precisión. Cuando ambos valores están introducidos, pulsar el botón "**Almacenar**" para actualizar el punto en la tabla.

- ***Eliminación de un punto***

  Para eliminar un punto existente, seleccionarlo de la lista y pulsar el botón "**Eliminar**". No debe olvidarse que se tienen que introducir al menos **dos puntos** para definir una tabla.

A.12.3 Normas y límites

Se tiene que cumplir con las siguientes normas a la hora de definir una tabla de conversión. La tabla puede utilizarse para convertir variables analógicas tanto de entrada como de salida:

- No se pueden definir dos puntos con el mismo valor eléctrico.
- La curva tiene que crecer o decrecer continuamente.
- No se pueden definir dos puntos con el mismo valor físico.

Son aplicables los siguientes límites a la hora de definir las tablas de conversión de un proyecto:

- No se pueden definir más de **127** tablas de conversión en un mismo proyecto.

No se pueden definir más de **32** puntos en una misma tabla de conversión.

A.13 Utilización del generador de código

Los comandos “Verificar” y “Construir aplicación”, pertenecientes a las otras ventanas del banco de trabajo ISaGRAF, abren la ventana de generación de código de forma automática. Esta última no se cierra automáticamente cuando finaliza la operación de generación de código solicitada, para que el usuario pueda seguir teniendo acceso a todos los comandos y opciones de generación de código desde el menú de la ventana.

A.13.1 Comandos principales

El menú **"Fichero"** contiene los comandos necesarios para la comprobación de sintaxis de programa y la generación de código.
Hacer doble click sobre el mensaje de error para abrir el programa y posicionar el cursor sobre el error.

- ***Construcción del código de la aplicación***

  El comando "**Ensamblar**" construye todo el código del proyecto. Antes de generar nada, este comando comprueba la sintaxis de las declaraciones y los programas. Cualquier error que no puede detectarse durante la compilación de un programa se detecta durante la generación del código. Esto es aplicable a las tablas de conversión, las conexiones variables de E/S y los enlaces con las librerías. El proceso de generación de código suspende la compilación de un programa cuando se detectan errores. Se tiene que corregir el programa en cuestión antes de poder proseguir con la generación del código. No se vuelven a compilar los programas que ya han sido comprobados (sin que se hayan detectado errores) y que no han sido modificados desde su última operación de “**Verificar**”. Siempre se lleva a cabo la verificación de las declaraciones variables y la comprobación de la coherencia de la aplicación. Para abortar la operación “**Ensamblar**” una vez iniciado el proceso de comprobación de programas, pulsar el botón **ESCAPE**.

  Nota: Si se ha modificado la declaración de la variable local de un programa, se comprueba el programa. Si se ha modificado una variable global, se comprueban todos los programas.

- ***Verificación de sintaxis de programa***

  El comando "**Verificar programa**" le permite al usuario verificar un único programa. Se compila el programa seleccionado incluso si no ha sido modificado desde su última verificación. El comando "**Verificar diccionario**" le permite al usuario verificar las declaraciones de todas las variables del proyecto.
  El comando "**Verificar todos los programas**" comprueba la sintaxis de todos los programas pertenecientes al proyecto, incluso si algunos no han sido modificados. Este comando **no se para** cuando detecta un error en un programa. Se puede

utilizar para producir un listado completo de todos los errores que queden en los programas del proyecto. Para abortar este comando, pulsar el botón **ESCAPE**.

- ***Simulación de una modificación***

  El comando "**Marcar**" simula una modificación de todos los programas del proyecto para que todas sean verificadas en la siguiente operación "**Ensamblar**". Se utiliza el comando "**Abrir**" para abrir el último programa que haya sido verificado. Este comando es muy útil para lograr acceder directamente a un programa en el que se han detectado errores de sintaxis.

A.13.2 Opciones del compilador

Se utiliza el comando "**Opciones del compilador**" para configurar los parámetros principales que usa el Generador de Código ISaGRAF para construir y optimizar el código objeto. La finalidad de este comando es la de seleccionar el tipo de código que se va a generar, de acuerdo con los objetos ISaGRAF correspondientes, y configurar los parámetros del 'optimizador' de acuerdo con el tiempo de compilación previsto y los requisitos de tiempo de proceso de la aplicación.
El botón de "**Cargar**" abre una segunda ventana de diálogo con otras opciones que permiten pasar de código fuente incrustado comprimido a código de descarga, para permitir la característica de "Descarga". Acudir a la documentación de "Descarga" para más información.

- ***Selección de objetos***

  La ventana superior muestra la lista de códigos destino que se puede producir. Se utiliza el signo "**>>**" para indicar el(los) objeto(s) seleccionado(s). El Generador de Código ISaGRAF puede producir hasta **3** códigos diferentes con la misma operación de compilación. Utilizar los botones "**Seleccionar**" y "**Deseleccionar**" para determinar la lista de códigos objeto que se requiera, de acuerdo con el *hardware* objeto. A continuación se relacionan los objetos ISaGRAF estándares:

  SIMULATE:..... Este código está dedicado al **Simulador** del banco de trabajo ISaGRAF. No se puede ejecutar el simulador si previamente no se ha seleccionado este destino para generar el código de aplicación.
  ISA86M: ..... Este es un código **TIC** (*Target Independent Code*), dedicado a los *kernels* ISaGRAF instalados en procesadores basados en tecnología Intel. El tipo de procesador sólo tiene que ver con el orden de bytes en el código generado.
  ISA68M: ..... Este es un código **TIC** (*Target Independent Code*), dedicado a los *kernels* ISaGRAF instalados en procesadores basados en tecnología Motorola. El tipo de procesador sólo tiene que ver con el orden de bytes en el código generado.
  SCC: ..... Si se selecciona este destino el compilador ISaGRAF produce código fuente en lenguaje "C" estructurado que se compila y se enlaza con las librerías *kernel* objeto de ISaGRAF para producir un código ejecutable incrustado.

CC86M:..... Si se selecciona este destino, el compilador ISaGRAF produce un código fuente en lenguaje "C" no estructurado que se compila y se enlaza con las librerías *kernel* objeto de ISaGRAF para producir un código ejecutable incrustado. Se proporciona esta selección para compatibilidad con versiones de ISaGRAF anteriores a la V3.23, donde la generación e integración de código "C" estructurado no son soportadas.

Para conocer el tipo de *kernel* ISaGRAF objeto que está instalado en un PLC determinado, véase el manual de *hardware* correspondiente. Existe la posibilidad de que se soporten otros tipos de destino (código máquina, código fuente "C", etc.) en versiones futuras del banco de trabajo ISaGRAF.

Procesamiento SFC

Seleccionar la casilla titulada "**Usar máquina SFC incrustada**" para permitir el uso del motor SFC de ISaGRAF. ***Este debe ser el modo preferido ya que permite un mayor rendimiento en términos de tiempo de proceso***. Sin embargo, puede que no se disponga del motor objeto en determinadas implementaciones del objeto ISaGRAF, como sucede más habitualmente en los objetos personalizados basados en el post-procesamiento de código ISaGRAF. En tal caso, quizás se tenga que deseleccionar esta opción y dejar que el compilador ISaGRAF traduzca los diagramas SFC a instrucciones de bajo nivel. Para mayor información sobre la utilización de esta opción, véase la documentación de *hardware*.

Opciones del optimizador

Los siguientes parámetros, utilizados por el Generador de Código ISaGRAF para optimizar el código objeto, pueden configurarse desde la ventana de diálogo "**Opciones del compilador**". Se utiliza el botón "**Por defecto**" para eliminar todas las opciones de optimización, reduciéndose así el tiempo de compilación.

- Cuando se selecciona la opción "**Ejecutar dos pasadas del optimizador**", se ejecuta dos veces el Optimizador de Código ISaGRAF. Las optimizaciones realizadas durante la segunda pasada suelen ser menos significantes que las que se realizan en el transcurso de la primera.

- Al seleccionar la opción "**Evaluar expresiones constantes**", el compilador evalúa las expresiones constantes. Por ejemplo, la expresión numérica "**2 + 3**" queda reemplazada por "**5**" en el código destino. Si no se selecciona esta opción, se calculan las expresiones constantes durante el tiempo de proceso.

- Si se selecciona la opción "**Suprimir etiquetas sin usar**", el Optimizador simplifica el sistema de saltos y etiquetas de los programas mediante la supresión de 'etiquetas de destino' no utilizadas o altos nulos.

- Cuando se selecciona la opción "**Optimizar el copiado de variables**", se optimiza la utilización de las variables temporales (aquellas que se usan para almacenar resultados intermedios). Se suele emplear esta opción conjuntamente con la opción "**Optimizar expresiones**". Al seleccionar esta opción, el Optimizador reutiliza el

resultado de las expresiones y subexpresiones que se usan en más de una ocasión en el programa.

- Al seleccionar la opción "**Suprimir código no usado**", el Optimizador suprime el código que no sea significante. Por ejemplo, si se programan las siguientes sentencias: "**var := 1; var := X;**", el código generado es sólo: "**var := X;**".

- Si se selecciona la opción "**Optimizar operaciones aritméticas**", el Optimizador simplifica las operaciones aritméticas de acuerdo con unos operandos especiales. Por ejemplo, la sentencia "**A + 0**" queda reemplazada por "**A**". Cuando se selecciona la opción "**Optimizar operaciones booleanas**", el Optimizador simplifica las operaciones booleanas de acuerdo con unos operandos especiales. Por ejemplo, la sentencia booleana "**A & A**" queda reemplazada por "**A**".

- Si se selecciona la opción "**Construir diagramas de decisión binaria**", el Optimizador sustituye las ecuaciones booleanas (mezclando los operadores **AND**, **OR** , **XOR** y **NOT**) por una lista reducida de operaciones de saltos condicionales. Sólo se lleva a cabo la traducción si el tiempo de ejecución previsto de la secuencia de salto es inferior al tiempo previsto para la sentencia original.

La siguiente tabla resume los tiempos previstos de optimización y los tiempos solicitados de compilación correspondientes a cada parámetro:

	ganancia (rendimiento)	tiempo de compilación
Ejecutar 2 pasadas	□□□□	(*)
Optimizar expr. Constantes	□□□□□□□□□□	□□□□□□
Suprimir etiquetas no utilizadas	□□□□	□□□□□□□□□□
Optimizar copiado de variables	□□□□	□□□□□□□□□□
Optimizar expr.	□□□□	□□□□□□□□□□
Suprimir código no utilizado	□□□□	□□□□□□□□□□
Optimizar operac. aritméticas	□□□□□□□□□□	□□□□□□□□□□
Optimizar operac. booleanas	□□□□□□□□□□	□□□□
Construir diag. de decisión binarios	□□□□□□□□□□□□□□□□	□□□□□□□□□□□□□□□□

(*) el tiempo de compilación también se multiplica por 2.

A.13.3 Producción de código fuente en "C"

El banco de trabajo ISaGRAF permite la producción de código fuente en lenguaje "C". En este caso, el contenido completo de la aplicación - incluyendo la

descripción de diagramas SFC, la definición de bases de datos y las secuencias de código - está generado en el formato “C” de código fuente. Hay dos posibilidades, propuestas como dos estilos de código generado:

CC86M (Código fuente C- V3.04) produce código fuente "C" no estructurado. Se debe seleccionar este estilo si su software objeto está basado en versiones de ISaGRAF anteriores a la V3.23.

SCC (Código fuente C estructurado) produce código fuente "C" estructurado. Este estilo debe ser preferible si el software objeto está basado en versiones de ISaGRAF V3-23 o posteriores.

Se crean los dos ficheros siguientes en el directorio del proyecto:

APPLI.C ..... código fuente de la aplicación
APPLI.H ..... definiciones del lenguaje "C"
Estos ficheros tienen que ser compilados y enlazados a las librerías objeto de ISaGRAF para poder producir el código ejecutable final. Para más información sobre las técnicas de implementación recomendadas, véase la “Guía del Usuario de herramientas de desarrollo de E/S ISaGRAF”.

Nota: Algunas funciones de depuración, tales como la carga remota de aplicaciones, la modificación en línea y los *breakpoint*s, dejan de estar disponibles cuando se compila la aplicación ISaGRAF en “C”.

A.13.4 Visualización de información

El menú "**Edición**" contiene los comandos necesarios para visualizar los diferentes ficheros de texto que se crean durante las operaciones de generación de código o comprobación de sintaxis en la ventana de generación de código. La ventana de generación de código es un área de texto que contiene mensajes durante las operaciones de generación de código o comprobación de sintaxis. Se almacena toda la información en el disco para su posterior consulta por medio de los comandos del menú "**Edición**".

A.13.5 Definición de recursos

El comando "**Recursos**" del menú "**Opciones**" le permite al usuario definir recursos. Un recurso es cualquier dato definido por el usuario (configuración de red, configuración de *hardware*, etc.), en cualquier formato (fichero, lista de valores), que tenga que fusionarse con el código generado para su transmisión al PLC objeto. El *kernel* ISaGRAF no actúa directamente sobre este tipo de datos, que generalmente están dedicados a otro *software* instalado en el PLC objeto. Para mayor información sobre la disponibilidad de recursos, véase el manual del *hardware*.

A.14 Referencias cruzadas

El banco de trabajo ISaGRAF incluye un editor de referencias cruzadas que proporciona al usuario una visión global de las variables declaradas existentes en los programas del proyecto, así como de los lugares donde se utiliza cada una de ellas. La finalidad de las referencias cruzadas es la de listar todas las variables declaradas en el proyecto a la vez que localiza, en la fuente de cada programa, las partes del código fuente en las que se utilizan estas variables. Las referencias cruzadas son muy útiles para obtener una visión global del ciclo de vida de una variable. Ayudan a localizar efectos secundarios y reducen el tiempo que se necesita para entender el proyecto durante el proceso de mantenimiento. También se pueden utilizar las referencias cruzadas para obtener una visión global del diccionario completo de un proyecto, con la finalidad de encontrar fácilmente variables no utilizadas y medir la complejidad del proyecto.
La lista de la izquierda muestra los objetos declarados del proyecto (programas, variables y palabras definidas) y los elementos de librería (funciones y bloques de función) que están referenciados en el proyecto. La lista de la derecha muestra las ocurrencias en los programas del objeto que actualmente está seleccionado en la primera lista.
La descripción de una búsqueda encontrada incluye el nombre de programa, el número de paso, transición, o decisión FC SFC, más el número de línea para lenguajes de texto, o coordenadas para diagramas LD o FBD. Para diagramas Quick LD, la descripción es completa con el número de escalón. Si la variable se utiliza como salida, el número de escalón viene seguido por un carácter asterisco ("**").
Configurar la opción "**Mostrar variables no usadas**" del menú "**Opciones**" para mostrar en una lista principal variables que no se usan en los programas de aplicación.

- ***Selección del tipo objeto***

  Un proyecto puede contener un número enorme de objetos declarados. Por esa razón, se utiliza la ventana combinada de la barra de herramientas del editor para seleccionar el tipo de objeto que debe aparecer en la ventana. De esta manera, el usuario puede acceder a información seleccionada.
  Cada vez que se recalculan las referencias cruzadas, se restablece la selección en la posición "**todo**" para poder presentar la lista completa.

- ***Re-calcular referencias cruzadas***

  Se puede utilizar el comando "**Fichero / Re-calcular**" en cualquier momento para actualizar las referencias cruzadas de acuerdo con las modificaciones que se introduzcan en otras ventanas de edición ISaGRAF.

- ***Exportar referencias cruzadas***

  Se utiliza el comando "**Herramientas / Exportar (fichero de texto)**" para escribir el listado completo de referencias cruzadas en un fichero de texto ASCII. Se puede

abrir este fichero con otras aplicaciones, como el NotePad de Windows o un procesador de textos.

Errores en el diccionario

El comando "**Edición / Errores de diccionario**" muestra, en una ventana de diálogo, la lista de errores detectados cuando se cargó el diccionario del proyecto.

Estadística

El comando "**Herramientas / Estadísticas**" muestra, en una ventana de diálogo, el número de objetos y variables que fueron declarados en el proyecto, en función de los tipos y atributos de las variables. Este comando también se puede utilizar para averiguar el número de variables de E/S que fueron declarados en el proyecto, con la finalidad de asegurar que se puede compilar en el caso de que utilice una versión limitada del banco de trabajo ISaGRAF.

Búsqueda en la lista de objetos

El comando "**Edición / Buscar**" le permite al usuario la selección directa de un objeto en la lista del editor. No se puede encontrar el objeto buscado si no aparece en la lista (cuando se utiliza un filtro de visualización). Antes de buscar un objeto, se recomienda la activación de la opción "**todo**" en la barra de herramientas.

Apertura de programas

La lista de la derecha contiene las incidencias del objeto seleccionado en los ficheros fuente y la conexión E/S del programa abierto. El comando "**Edición / Abrir programa**" le permite al usuario abrir un programa directamente en el lugar en el que aparece el objeto. También existe la posibilidad de hacer doble click con el ratón sobre la ocurrencia (en la lista de ocurrencias) para abrir el programa correspondiente.

A.15 Utilización del depurador gráfico

ISaGRAF incluye un depurador gráfico y simbólico completo. El comando "**Depurar**" de la ventana del Gestor de Programas ejecuta el depurador para controlar la aplicación que se haya cargado en el PLC objeto. En este modo de funcionamiento, el depurador comunica con el sistema objeto por medio de un enlace *hardware*. El comando "**Simular**" de la ventana del Gestor de Programas lanza la ejecución simultánea del depurador y de un simulador completo del objeto. Esto le permite al usuario comprobar su aplicación cuando el sistema de E/S objeto todavía no está completo. La ventana del depurador contiene los comandos necesarios para controlar toda la aplicación.
Al iniciarse el depurador, si la aplicación del PLC objeto es la misma que la del banco de trabajo, se abre automáticamente la **ventana del Gestor de Programas**, en modo depuración. Se pueden utilizar los comandos de esta ventana para abrir otras ventanas ISaGRAF (editores gráficos y de texto, diccionario, listas de variables, conexión de E/S, etc.). Todas las ventanas que se abren durante una sesión de depuración operan en "**modo depuración**", lo que significa que el comando de edición está deshabilitado. Se muestran los componentes de programas (pasos, transiciones, variables, etc.) con su estado o valor actual de tiempo de operación. Hacer doble click sobre un objeto para cambiar su estado o valor en la aplicación objeto.
Al ejecutar el depurador en **modo simulación**, se interrumpe la comunicación con el sistema ISaGRAF objeto. El depurador sólo se comunica con la ventana del simulador. Ya que el sistema objeto no existe en este modo, los comandos "**carga**", "**parada**" o "**activar**" no están disponibles en el menú del depurador.

A.15.1 La ventana del depurador

La ventana del depurador sólo contiene información relativa al estado de la aplicación en su conjunto. Está vinculada a otras ventanas ISaGRAF, creando un sistema interactivo de depuración completo. Los errores de tiempo de operación detectados se muestran en la parte inferior de la ventana del depurador. Se utilizan los comandos del menú "**Opciones**" para ocultar, mostrar o eliminar la lista de errores.

El panel de control (la zona situada debajo del menú del depurador) muestra el estado global de la aplicación destino, junto con información sobre tiempos de los ciclos de ejecución. El sistema objeto puede presentar los siguientes estados:

Presentación:.....El depurador establece la comunicación con el sistema objeto.
Desconectado:.....El depurador no puede comunicar con el sistema objeto. Revisar el cable de conexión y asegurar la validez de los parámetros de comunicación.

Sin aplicación: .....La conexión es correcta, pero en la actualidad no existe ninguna aplicación ISaGRAF en el sistema objeto. Proceder a la carga remota de una aplicación.
Aplicación activa: ....La conexión es correcta y existe una aplicación activa en el sistema objeto. El depurador está estableciendo la comunicación con esta aplicación, siempre que sea la misma que la que está activa en el banco de trabajo.
RUN: .....La aplicación objeto está en modo “Tiempo Real”.
STOP: .....La aplicación objeto está en modo "Ciclo a Ciclo”.
Breakpoint: .....La aplicación objeto está en modo "Ciclo a Ciclo” porque se encontrado con un *breakpoint*.
Error Fatal: .....La aplicación objeto falló debido a un error grave

Se dispone de la siguiente información sobre tiempos de los ciclos de ejecución:

Permitido: .....tiempo programado.
Actual: .....tiempo exacto del último ciclo completo de ejecución.
Máximo:.....tiempo máximo detectado desde el inicio de la aplicación.
Rebasamiento:.....número de ciclos de ejecución detectados con tiempos superiores al permitido.

Todos los valores de tiempo se expresan en milisegundos. No se muestran los valores de tiempo cuando se utiliza el depurador en modo simulación.

A.15.2 Control de la aplicación

Los menús "**Fichero**" y "**Control**" contienen todos los comandos necesarios para la instalación y el control de la aplicación ISaGRAF actual en el sistema ISaGRAF objeto.

Nota: No se dispone de algunos de estos comandos durante la simulación, ya que el banco de trabajo ISaGRAF instala automáticamente la aplicación que está siendo procesada por el simulador.

Interrupción de la aplicación objeto

El comando "**Fichero / Parar aplicación**" interrumpe la ejecución de la aplicación activa en el sistema ISaGRAF objeto.

Activación de la aplicación objeto

El comando "**Fichero / Iniciar aplicación**" ejecuta la aplicación que está presente en el sistema objeto. Cuando se carga una aplicación se inicia automáticamente, por lo que no se tiene que utilizar el comando "**Iniciar**". El comando "**Iniciar**" suele emplearse después de un comando "**Parar**".

Nota: La aplicación objeto debe estar parada (inactiva) antes de realizar la carga remota de una nueva aplicación.

Carga de la aplicación

Se utiliza el comando "**Fichero / Cargar**" para efectuar la carga del código de aplicación en el sistema objeto. Seleccionar el tipo de código que se va a cargar, de acuerdo con el procesador del sistema objeto y las opciones de la aplicación.

Visualización del número de versión

Se utiliza el comando "**Fichero / Obtener número de versión**" para visualizar la identificación completa tanto de la aplicación del banco de trabajo como de la aplicación objeto. La aplicación del banco de trabajo es la que está abierta en el banco de trabajo ISaGRAF. La aplicación objeto es la que se ejecuta en el PLC ISaGRAF objeto. Se muestran los siguientes puntos:

VERSION:.....Es el número de versión del código de aplicación. Este número ha sido calculado por el generador de código.

FECHA:.....Este punto muestra la fecha y hora a la que se construyó el código.

CRC:.....Se trata de un valor de comprobación que se calcula con base en el contenido de la tabla de símbolos. Este número ha sido calculado por el generador de código. Su valor depende del contenido del diccionario de variables.

Nota: El comando "**Obtener número de versión**" también está disponible durante la simulación. En el modo de depuración real, no se puede utilizar este comando si no está conectado el PLC objeto.

Modificación en línea

El comando "**Fichero / Actualizar aplicación**" le permite al usuario llevar a cabo la "modificación en línea" de la aplicación objeto que se está ejecutando. Este comando se describe en mayor detalle en secciones posteriores de este apartado. No está disponible cuando se ejecuta el depurador en modo simulación.

Modo Tiempo Real

No se dispone del comando "**Control / Tiempo real**" cuando no hay una aplicación activa. Este comando establece el modo normal de "tiempo real" en la aplicación objeto. Modo Normal: los ciclos de ejecución se activan por tiempo programado de ciclo.

Modo Ciclo a Ciclo

No se dispone del comando "**Control / Ciclo a ciclo**" cuando no hay una aplicación activa. Este comando establece el modo normal de "ciclo a ciclo" en la aplicación objeto. En este modo, se ejecutan los ciclos uno a uno, de acuerdo con los comandos "**Ejecutar un ciclo**" realizados por el usuario desde el menú del depurador.

'Ejecutar un ciclo'

Cuando el sistema objeto está en modo ciclo a ciclo, el comando "**Control / Ejecutar un ciclo**" provoca la ejecución de un ciclo.

Tiempo de ciclo

El comando "**Control / Cambiar tiempo de ciclo**" le permite al usuario modificar el tiempo programado de ciclos. Este tiempo se denomina "**Permitido**", en la ventana de la barra de control del depurador. Se debe establecer el modo "**Ciclo a ciclo**" antes de modificar los tiempos de ciclo. Se introducen los tiempos de ciclo como números enteros expresados en milisegundos.

- ***Eliminación de todos los breakpoints***

  El comando "**Control / Eliminar todos los *breakpoint*s**" elimina todos los *breakpoint*s que actualmente estén instalados (encontrados o todavía activos) en toda la aplicación. Los *breakpoint*s actuales no se eliminan automáticamente cuando se cierra la ventana del depurador.

- ***Desbloqueo de variables de E/S***

  El comando "**Control / Desbloquear todas las variables ES**" desbloquea todas las variables de E/S que actualmente están bloqueadas en la aplicación. Cuando una variable de E/S está bloqueada, no se pueden realizar cambios en el estado de entrada o salida del dispositivo de E/S correspondiente. La aplicación o el depurador pueden seguir escribiendo las variables asociadas a una E/S. Las variables de E/S bloqueadas no se desbloquean automáticamente cuando se cierra la ventana del depurador.

A.15.3 Opciones

El menú "**Opciones**" contiene todas las opciones necesarias para controlar la información mostrada en la ventana del depurador.

- ***Parámetros de comunicación***

  Se pueden ajustar los parámetros de tiempos de comunicación cuando el depurador está activo. Sólo se puede configurar el **tiempo de espera** de comunicación desde aquí. Los demás parámetros de comunicación (baudios, paridad, etc.) se configuran desde el menú "**Depuración**" de la ventana del Gestor de Programas.
  El "**Tiempo permitido de comunicación**" es el tiempo que queda para que el sistema objeto comience a contestar a una petición del banco de trabajo. La "**Duración del refresco de ciclo**" es el periodo de tiempo que se requiere para que el depurador envíe las peticiones de "**lectura**" para refrescar los datos que aparecen en las ventanas abiertas.
  Todos los valores de tiempo se muestran y se introducen como números enteros, expresados en **milisegundos**. No se puede configurar los parámetros de tiempo de comunicación cuando el depurador se utiliza en modo simulación.

- ***Opciones de visualización***

  La opción "**Mostrar tiempo de ciclo**" le permite al usuario ocultar o mostrar los valores de **tiempos de ciclos** en la barra de control del depurador. Cuando se selecciona esta opción, se muestran y se refrescan todos los componentes de tiempos de ciclos (permitido, actual, máximo, rebasamientos, etc.). La desactivación de esta opción reduce la carga de comunicación del depurador.

Cuando se selecciona la opción "**Mostrar errores**", los errores de tiempo de proceso que se detectan aparecen en la zona inferior de la ventana de depuración. Si se desactiva esta opción, se cierra la lista de errores. La eliminación de esta opción reduce la carga de visualización y comunicación del depurador. El comando "**Opciones / Limpiar errores**" elimina la lista de errores de tiempo de proceso que se está mostrando en la ventana del depurador.

El comando "**Opciones / Minimizar ventana**" reduce el tamaño de la ventana del depurador para que aparezca como un panel pequeño, siempre visible, que sólo contiene el estado de la aplicación y unos botones gráficos para los comandos de uso más frecuente.

A.15.4 Comandos "Escritura"

El depurador simbólico de ISaGRAF posee numerosos comandos para cambiar el **valor** o **estado** de los componentes de una aplicación. Seleccionar el componente que se desea cambiar haciendo **doble click** sobre su nombre o dibujo en una ventana de edición, cuando se abre la ventana del depurador.

- ***Variables***

Para cambiar el estado de una variable, hacer doble click sobre su nombre en una de las siguientes ventanas:
- Diccionario
- Listas de variables o diagramas de tiempo
- Programas LD o FBD
- Conexión de E/S

Se ofrecen los siguientes comandos en la ventana de diálogo del depurador:
- Introducir un nuevo valor para la variable
- **Bloquear** la variable (sólo variables de E/S)
- **Desbloquear** la variable (sólo variables de E/S bloqueadas)
- **Iniciar** o **parar** una variable temporizador (establecer modo de refresco automático)

Los valores simbólicos que se utilizan para representar los valores booleanos **FALSO** y **VERDADERO** son las cadenas que se han definido para esa variable específica en el diccionario. El valor analógico que se especifica para un comando "**Escritura**" se tiene que introducir en formato de valores enteros o reales, según la definición de variables del diccionario. La cadena que se establezca para un mensaje no puede ser superior a la capacidad de mensaje que tenga asociada esa variable en el diccionario.

- ***Objetos SFC***

Para observar una operación de control sobre un **programa SFC** mientras se depura la aplicación, se utilizan los comandos del menú "**Fichero**" en la ventana

del Gestor de Programas. El programa SFC debe estar seleccionado desde la lista de programas. Se dispone de los siguientes comandos:

Iniciar programa SFC: ..... Habilita el programa seleccionado mediante la colocación de una marca SFC en cada uno de sus pasos iniciales.

Matar programa SFC: ..... Deshabilita el programa seleccionado mediante la eliminación de todas las marcas existentes.

Congelar programa SFC: ..... Elimina todas las marcas existentes en el programa seleccionado y memoriza su posición.

Continuar programa SFC:..... Reinicia un programa 'congelado' mediante la colocación de todas las marcas que fueron eliminadas por el comando "Congelar".

En el caso de los programas hijo, estos comandos equivalen a las funciones "**GSTART**", "**GKILL**", "**GFREEZE**" y "**GRST**" del lenguaje de programación.

Para visualizar una operación de control en un **paso SFC** mientras se depura la aplicación, hacer doble click sobre su representación gráfica en la ventana de edición SFC. Se dispone de los siguientes comandos en la ventana de diálogo del depurador:

- Establecer *breakpoint* en la **activación** del paso
- Establecer *breakpoint* en la **desactivación** del paso
- **Eliminar** un *breakpoint* que se haya añadido al paso

Nota: No se pueden añadir *breakpoint*s de activación y desactivación al mismo paso.

Para visualizar una operación de control en una **transición SFC** mientras se depura la aplicación, hacer doble click sobre su representación gráfica en la ventana de edición SFC. Se dispone de los siguientes comandos en la ventana de diálogo del depurador:

- Establecer un ***breakpoint*** al franquear la transición
- **Eliminar** un *breakpoint* que se haya añadido a la transición
- **Franquear** la transición manualmente (mover o añadir marcas)

Franqueo condicional: se coloca una marca en los pasos que siguen a la transición. Se eliminan las marcas que existan en los pasos precedentes.
Franqueo incondicional: se coloca una marca en los pasos que siguen a la transición. No se eliminan las marcas que existan en los pasos precedentes.

A.15.5 Modificación en línea

La función de "modificación en línea" le permite al usuario modificar la aplicación mientras se ejecuta el proceso. A veces resulta necesario para los procesos químicos, donde cualquier interrupción puede resultar perjudicial para la producción o la seguridad. Se debe utilizar esta función **con mucho cuidado**. ISaGRAF podría ser incapaz de detectar todos los posibles conflictos provocados por operaciones definidas por el usuario, como resultado de estos cambios en línea.

Secuencias de código

Teniendo en cuenta que ISaGRAF ofrece muchas posibilidades de acceso a variables, programas o tarjetas de E/S desde el depurador, la función de "modificación en línea" que se describe aquí sólo es aplicable a la modificación de secuencias de código. Una secuencia de código es un conjunto completo de instrucciones ST, IL, LD o FBD que se ejecutan en serie. En un programa de "comienzo de ciclo" o "fin de ciclo", una secuencia de código es la lista entera de instrucciones escritas en el programa. En un programa SFC, una secuencia de código es la programación de Nivel 2 de un paso o una transición. La "modificación en línea" consiste en sustituir uno o más secuencias de código, sin interrumpir el ciclo de ejecución del PLC. Ya que el control de las marcas SFC es altamente crítico, **no se puede modificar una estructura SFC, añadir, renumerar o eliminar un paso, una transición o un programa SFC**.

Variables

La base de datos de variables es una parte muy crítica de la aplicación, por lo que otros procesos (en PLCs de multitarea) pueden acceder a ella en cualquier momento. También existe la posibilidad de modificar valores de variables desde el depurador. Por lo tanto, **ISaGRAF no permite al usuario añadir, renombrar o eliminar una variable** en línea. Sin embargo, sí es posible modificar la forma en la que se utiliza una variable en la aplicación. Existe asimismo la posibilidad de reservar variables internas o de E/S "no utilizadas" en la primera versión de la aplicación, para que las futuras modificaciones puedan hacer uso de ellas.

Hay diferentes tipos de variables en la base de datos del sistema objeto de ISaGRAF . Las limitaciones afectan a todas ellas.

- *Variables declaradas*

Son las que están declaradas usando el diccionario de ISaGRAF . No pueden cambiarse ni eliminarse para cambios en línea. Se recomienda que algunas de estas variables adicionales se declaren e inicialicen en la aplicación, incluso si no se usan actualmente. Estas variables adicionales permitirán futuras modificaciones para seguir trabajando sin cambiar el dato de comprobación de la aplicación

- *Instancias de bloques de función*

Cada instancia de bloques de función "C" o IEC escritos corresponde a datos almacenados en la base de datos en tiempo real del sistema objeto de ISaGRAF. Cuando se añaden o eliminan bloques de función no son posibles cambios en línea. De ahí que sea mejor trabajar en ST con instancias FBD declaradas en el diccionario, mejor que añadiendo bloques (esto correspondería a instancias nuevas automáticamente declaradas) en diagramas Quick LD o FBD. También cualquier modificación en la definición de bloques de función disponibles en la librería de ISaGRAF llevaría a un cambio en línea imposible.

- *Pasos*

Cada paso SFC corresponde a un conjunto de datos donde se almacenan atributos dinámicos de los pasos (su actividad temporal y bandera). Añadir o eliminar pasos SFC cambia la base de datos de la aplicación y se deniega para cambios en línea.

- *Variables ocultas alocadas por compiladores*

El compilador de ISaGRAF genera variables temporalmente ocultas para resolver expresiones complicadas. En algunos casos el cambio de una expresión puede dar lugar a un conjunto diferente de variables temporales, y eso lleva a la imposibilidad de cambios en línea. Para evitar esta situación se pueden añadir las siguientes entradas en el fichero ISA.INI, para forzar un número mínimo de variables temporales que se aloquen para cada programa, incluso si no se usa para compilar la versión de la primera aplicación. Los valores dados aquí son ejemplos:

```
[DEBUG]
MNTVboo=8  ; para booleanas
MNTVana=4  ; para enteros y reales
MNTVtmr=4  ; para temporizadores
MNTVmsg=2  ; para mensajes
```

Cuando se escribe una configuración como ésta en el archivo ISA.INI el compilador da un mensaje de aviso si una nueva compilación de la aplicación lleva a un número mayor de variables alocadas temporalmente.

Entradas y salidas

El sistema de E/S de ISaGRAF es muy abierto, por lo que las modificaciones necesarias deberán estar implementadas por los OEM, utilizando funciones específicas del *hardware* correspondiente. El sistema ISaGRAF **no permite al usuario añadir, conectar o eliminar una variable de E/S, o modificar la descripción de una tarjeta de E/S** en línea. Se dispone de operaciones como la modificación de los parámetros de una tarjeta y el bloqueo de los canales de E/S por medio de las funciones OEM estándares y la función "**OPERATE**".

Operaciones en tiempo de proceso

La modificación de una aplicación en funcionamiento consiste de las siguientes operaciones:

- modificar el código fuente de la aplicación en el banco de trabajo
- generar el nuevo código de aplicación
- realizar la carga del nuevo código de aplicación utilizando el comando "**actualizar**" en lugar de "**cargar**"
- cambiar de la aplicación vieja a la nueva, entre ciclos de ejecución del PLC, utilizando el comando "**Realizar actualización**".

Este procedimiento garantiza que el PLC objeto disponga siempre de una aplicación completa y fiable, y permite al usuario controlar los tiempos de las operaciones de muestra de forma muy segura y eficiente. Así mismo, permite al usuario modificar el proyecto tantas veces como sea necesario. Con independencia del proceso, la “modificación en línea” es básicamente igual que un conjunto normal de comandos "**parar**, **iniciar** y **cargar**". La única diferencia que existe es que no se pierden estados variables y el tiempo de cambio es muy corto (duración típica: 1 ó 2 ciclos). Durante el cambio no se modifica ninguna variable, y **todas las variables internas, de entrada o de salida tienen el mismo valor** antes y

después de la modificación de la aplicación. Durante el cambio, no se lleva a cabo ninguna acción y **no se mueven las marcas SFC**.

Requisitos de memoria

Para poder soportar la capacidad de “modificación en línea”; el PLC objeto tiene que tener suficiente memoria libre para poder guardar la versión modificada del código de aplicación. Se tienen que guardar ambas versiones del código de aplicación en la memoria del PLC durante la operación de cambio.

Limitaciones

Como ya se dijo, sólo se permiten las modificaciones de las secuencias de código. No se pueden modificar las definiciones de variables, parámetros de aplicaciones y conexiones de E/S. Cuando se realiza la carga de una versión modificada de la aplicación, ISaGRAF lleva a cabo una comparación entre la aplicación modificada y la que actualmente está en funcionamiento, para detectar cualquier cambio que no sea seguro. Si el cambio pudiera ser peligroso o imposible, se genera un error de carga. Una de las medidas de seguridad que lleva a cabo ISaGRAF es la comparación del valor de comprobación de la tabla de símbolos, para así detectar el cambio de nombre de cualquier variable, programa o elemento SFC. Si un paso está activo cuando se produce el cambio, sus acciones (N) no almacenadas se pierden. No se ejecutan las nuevas acciones de activación de pasos. Las acciones que se ejecutan en el momento de desactivación del paso son aquellas que proceden del nuevo código de aplicación. Si una transición es válida cuando se produce el cambio, se actualiza su ecuación de receptividad. El PLC no realiza una copia de seguridad del código de aplicación recién cargado. La copia de seguridad corresponde a la versión previamente cargada por medio de comandos estándares de carga remota.

Operaciones

Para actualizar el código de una aplicación activa, se tienen que realizar las siguientes operaciones:

- Antes de efectuar cambio alguno en una aplicación activa, es muy recomendable la realización de una copia del proyecto actual con un nombre nuevo. Se pueden efectuar las modificaciones sobre las copias.
- Antes de editar un programa, el usuario debe comprobar que está seleccionada la opción "**actualizar diario**" de las herramientas de edición, para facilitar el mantenimiento del programa en el futuro.
- Cuando se modifican una o más secuencias (sin modificar las estructuras y jerarquía de programa SFC), se debe generar el código de la aplicación nueva en el banco de trabajo antes de proceder a su carga remota.
- Utilizando el depurador, el usuario tiene que conectar con el PLC objeto desde el proyecto antiguo y realizar cualquier operación que permita que la aplicación se actualice con mayor rapidez o seguridad.
- Utilizando el depurador, el usuario tiene que conectar con el PLC objeto desde el proyecto nuevo. Si se ha cambiado el nombre de la aplicación, no se podrá acceder a la base de datos objeto. En este caso, el usuario tiene que ejecutar el comando "**Fichero / Actualizar**".
- Se realiza la carga remota de la aplicación modificada mediante la selección de la opción "**actualizar después**". Esto puede suponer una pequeña ralentización del PLC durante la transferencia.

- Cuando se complete la carga, el usuario puede ejecutar el comando "**Fichero / Realizar actualización**" para habilitar el cambio en el momento más oportuno. El cambio tendrá una duración de 1 ó 2 ciclos.

Si el cambio se ha llevado a cabo correctamente, se muestran los programas de la aplicación modificada que está activa. En caso contrario, la aplicación activa existente permanece como estaba.

A.15.6 Intercambios DDE

El depurador ISaGRAF incluye un servidor DDE (*Dynamic Data Exchange*). Se puede instalar un bucle de aviso entre el depurador ISaGRAF y otras aplicaciones, para la visualización dinámica del valor actual de las variables en aplicaciones ajenas a ISaGRAF.

El servidor DDE del depurador ISaGRAF sólo soporta transacciones del tipo "*advise*" y "*poke*". Sólo se puede utilizar la transacción "*request*" para variables que ya hayan sido espiadas en un bucle de aviso. Otros servicios DDE, como "*execute*", no están disponibles. Cuando se establece un bucle de aviso en una variable, el valor de esta variable se actualiza en la aplicación cliente cada vez que cambia. Se pueden espiar variables de cualquier tipo. La identificación del enlace dinámico incluye los siguientes nombres:

Nombre de Servicio:..... "ISaGRAF"
Nombre de Tópico:..... Nombre del proyecto ISaGRAF
Nombre de Item: ..... Nombre de la variable

Si la variable es local para un programa, su nombre debe venir seguido por el nombre del programa padre, expresado entre paréntesis y con la siguiente sintaxis:

nombre_variable(nombre_programa)

El servidor DDE del depurador ISaGRAF está dedicado a la aplicación ISaGRAF que actualmente está siendo espiada por el depurador. El servidor ISaGRAF puede espiar hasta **256** variables. Se puede utilizar el servidor DDE cuando se ejecuta el depurador ISaGRAF bien en modo conectado o bien en modo simulación. La duración del refresco es la que se establece para la comunicación entre el depurador y el sistema objeto o simulador de ISaGRAF.

A.16 Espiar variables

El comando "**Espiar listas**" del menú "**Herramienta**" de la ventana del Depurador le permite al usuario la construcción de listas no contiguas de variables que se refrescan con sus valores actuales. Las listas se crean a la hora de depurar la aplicación. Se pueden guardar las listas en el disco y abrirse de nuevo en el transcurso de otras sesiones de depuración. Una lista puede contener hasta **32** variables. Se pueden mezclar variables de diferentes tipos en una misma lista. Se pueden insertar variables globales y locales en una lista. Una lista de variables está dedicada a un proyecto en particular. Las listas de variables son muy útiles a la hora de realizar la comprobación funcional de una aplicación. Le permiten al usuario observar los cambios que se producen en una parte limitada del proceso controlado, con independencia del correspondiente código fuente de los programas de aplicación. Las listas de variables también son útiles cuando se depuran programas de texto ST e IL. El usuario puede agrupar en una lista al conjunto de variables utilizadas en un programa, con la finalidad de controlar o supervisar la ejecución de las instrucciones programadas.
Para cada variable de la lista , ISaGRAF muestra su nombre, su valor actual, y el correspondiente texto de comentario. Se pueden cambiar de tamaño las columnas arrastrando las líneas de separación con el ratón en la barra de títulos de la lista.

Guardar listas en el disco duro

Se utilizan los comandos del menú "**Fichero**" para crear, abrir y guardar las listas de variables. ISaGRAF no limita el número de listas para cada proyecto. Cuando se asignen nombres a las listas de variables que van a guardarse en el disco, se tiene que cumplir las siguientes normas:

- el nombre no puede superar los **8** caracteres
- el primer carácter tiene que ser una **letra**
- los restantes caracteres pueden ser **letras**, **dígitos** o el carácter de subrayado '_'
- los nombres de las listas no son sensibles al uso de mayúsculas o minúsculas

El editor de listas no puede representar más de una lista de variables a la vez en la misma ventana. Sin embargo, se puede ejecutar el editor de listas más de una vez, para poder espiar diferentes listas simultáneamente.

Inserción de variables en la lista

El comando "**Edición / Insertar**" inserta otra variable en la lista. Se selecciona el nombre de la variable de la lista de objetos que están definidos en el diccionario del proyecto. De esta forma, el usuario no tiene que introducir el identificador manualmente. Se inserta la variable delante de la variable que está seleccionada en la lista. La lista no puede contener más de **32** variables. La misma variable no puede aparecer más de una vez en la misma lista.

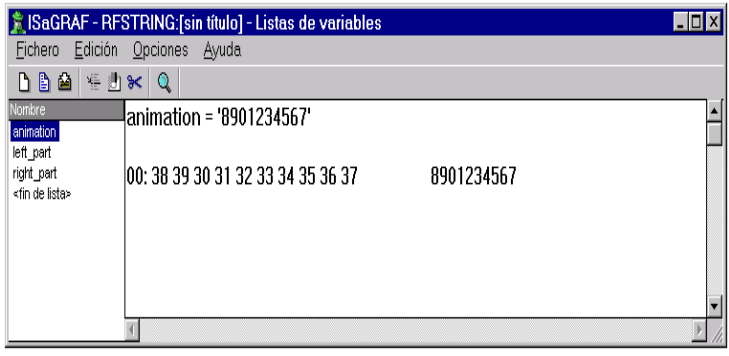
Modificación de la variable seleccionada

El comando "**Edición / Modificar**" sustituye la variable seleccionada por otra variable. También se puede utilizar el comando "**Cortar**" para eliminar la variable seleccionada de la lista.

Vista hexadecimal

En cualquier instante se puede conmutar entre el modo lista y la vista "Hexadecimal". Pulsar el botón de "zoom" en la barra de herramientas o usar el comando "**Opciones / Hexadecimal**" para conmutar el modo de vista.

En el modo "Hexadecimal", sólo se muestra el valor de una variable. Su valor se muestra en formato numérico/simbólico en la parte de arriba de la ventana, y se ve también en formato binario. Este modo permite espiar el valor hexadecimal de cada byte en el del valor de la variable.



El modo hexadecimal es muy útil para espiar y comprender mensajes que contienen caracteres no imprimibles.

A.17 Depuración de programas ST e IL

Durante la simulación o la depuración en línea de un programa ST e IL, no se pueden introducir modificaciones en el texto del programa.

IL Para programas IL las instrucciones de formatean en una vista de lista. El valor actual de la variable utilizada en una instrucción se muestra en la misma línea. Se puede hacer doble click en una instrucción para cambiar el correspondiente valor de la variable.

ST Para programas ST, una ventana de Lista de Espía se integra en la ventana de edición. Se puede cambiar el tamaño de las ventanas arrastrando mediante el ratón la línea de separación entre las mismas.
Para cada variable de la lista, ISaGRAF muestra su nombre, su valor actual y su texto de comentario. Se puede cambiar el tamaño de las columnas arrastrando con el ratón las líneas de separación en la barra de títulos.

Guardar listas en el disco duro

El comando "**Fichero / Guardar**" guarda la lista de variables en el disco, bajo el mismo nombre que el del programa editado. La lista se cargará automáticamente cada vez que el programa ST o IL se abre en modo depuración. Esta lista se puede también abrir y modificar libremente utilizando la herramienta Espiar Lista ejecutado por el comando "**Herramientas / Espiar listas**" de la ventana de depuración.

Inserción de variables en la lista

El comando "**Edición / Insertar**" inserta otra variable en la lista de diagramas de tiempo. Se selecciona el nombre de la variable de la lista de objetos que están definidos en el diccionario del proyecto. De esta forma, el usuario no tiene que introducir el identificador manualmente. Se inserta la variable delante de la variable que está seleccionada en la lista. La lista no puede contener más de **8** variables. La misma variable no puede aparecer más de una vez en la misma lista.

Cuando el nombre de una variable se destaca en texto ST, pulsar este botón en la barra de herramientas o ejecutar el comando "**Edición / Espiar selección**" para mandar directamente la variable a la lista de espía.

Modificación de la variable seleccionada

El comando "**Edición / Modificar**" sustituye la variable seleccionada por otra variable. También se puede utilizar el comando "**Cortar**" para eliminar la variable seleccionada de la lista.

A.18 Spotlight

La herramienta SpotLight de ISaGRAF permite definir al usuario listas de observación (watch) que pueden ser presentadas bien como dibujos gráficos o como listas durante la depuración. Los items gráficos deben estar enlazados a variables del proyecto ISaGRAF. El dibujo gráfico se define y anima “en línea”.
Para forzar el valor de una variable, hacer doble click en el ítem correspondiente de la composición gráfica o lista, o pulse INTRO cuando esté seleccionado.
Se puede también bloquear el documento (denegar cualquier modificación) utilizando el comando "**Fichero / Bloquear**". Cuando un documento este bloqueado, se pueden forzar variables haciendo doble click sobre su símbolo.

A.18.1 Construyendo la composición gráfica

Un diagrama esta constituido por dibujos de fondo (bitmaps o metafiles), y un conjunto de items gráficos que serán animados durante la depuracion. Para crear el diagrama, se deben realizar las siguientes operaciones: Insertar dibujos de fondo, insertar items gráficos, enlazar objetos a las variables del proyecto.

Dibujos de fondo

Los dibujos de fondo son ficheros "bitmap" (.BMP) o “metafile” (.WMF). El número de dibujos incluidos en la composición gráfica no está limitado. Los dibujos se pueden mover o redimensionar en la composición gráfica. No aparecen en la composición de lista. Los dibujos se construyen con otras herramientas. SpotLight no incluye una herramienta de dibujo. El comando "**Opciones / Color de fondo**" se utiliza para seleccionar un color sólido para el espacio vacío en una composición gráfica.

Nota: Los bitmaps consumen una elevada cantidad de memoria. Es altamente recomendable dimensionar correctamente el dibujo, y limitar el espacio no utilizado dentro del rectángulo del bitmap.

Presentación solo texto

Un ítem “solo texto” es un texto escrito en un rectángulo. El texto presentado es el valor de la variable asociada. Así, dicho ítem puede estar enlazado a una variable cadena de mensaje.

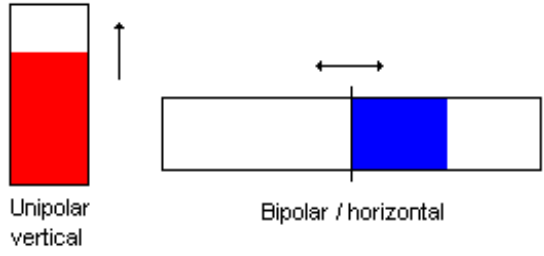
El rectángulo donde se presenta el texto puede bien estar coloreado o bien ser transparente. Cuando se redimensiona el ítem la fuente de caracteres empleada para presentar el texto se ajusta para encajarse a la altura del rectángulo.

Gráficos de barras unipolares y bipolares

Un gráfico de barras es un rectángulo con una parte coloreada que representa el valor numérico de la variable asociada. Opcionalmente, el resto de la barra puede estar coloreada. Un gráfico de barras puede ser bien horizontal o vertical.

Los gráficos de barras unipolares pueden crecer en cualquier dirección, hacia arriba, hacia abajo, a la izquierda, a las derecha.

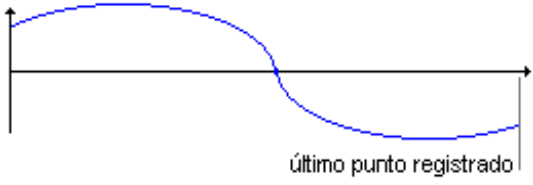
Los gráficos bipolares pueden crecer bien en dirección positiva o negativa, según el valor de la variable asociada. En el caso de un gráfico bipolar el valor máximo permitido es el mismo para ambas escalas positiva y negativa.



Curvas

Es posible insertar una curva en un documento. Una curva muestra la historia de la variable asociada. Aunque no es una herramienta de medida precisa, sí puede dar una información de depuración sobre la sincronización entre varias variables.

Una curva almacena los 200 últimos valores de la variable. El número de muestras no cambia cuando se redimensiona el ítem curva en la composición gráfica.

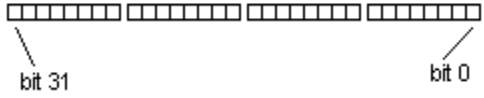


Iconos booleanos

Un ítem “Icono booleano” se utiliza para presentar un estado binario. Se define un fichero icono (.ICO) para FALSO o valor 0. Se define otro icono para el resto de los valores distinto a cero. Como SpotLight no incluye un editor de iconos, los ficheros de iconos deben ser preparados con otra herramienta.

Campos de bits

Un ítem "campo de bits" muestra en un panel gráfico los 32 bits de un valor entero. El bit menos significativo siempre se presenta a la derecha. No se recomienda el uso de campo de bits para otro tipo de datos como valores analógicos reales, ya que la información presentada puede inducir a confusiones.



Seleccionar, mover o redimensionar items

La selección de objetos gráficos se necesita para la mayoría de los comandos de edición. SpotLight permite la selección de uno o más objetos gráficos existentes en el área del diagrama. Para seleccionar objetos, el botón de "**selección**" (botón con una flecha) debe estar seleccionado en la barra de herramientas del editor. Para seleccionar un objeto, el usuario simplemente tiene que hacer click en su símbolo. Para seleccionar una lista de objetos, arrastre el ratón en el área de dibujo para seleccionar un área rectangular. Todos los objetos gráficos que intersectan el rectángulo de selección se marcan como "**seleccionados**". Un objeto seleccionado se dibuja con pequeños cuadrados negros alrededor de su símbolo gráfico.

Haciendo una nueva selección, cualquiera de los objetos previamente seleccionados se deseleccionan. Para deshacer la(s) selección(es) actual(es), simplemente hacer click con el ratón en un área vacía fuera del rectángulo que bordea los objetos seleccionados.

Para mover objetos, primero tienen que estar seleccionados. Poner entonces el cursor en el borde del ítem seleccionado y arrastrarlo a otra posición.

Para redimensionar un objeto, primero tiene que estar seleccionado. Poner entonces el cursor en uno de los pequeños rectángulos presentados en el borde de selección, y arrastrar en la dirección apropiada para redimensionar el objeto. Los dibujos también pueden redimensionarse. En tal caso, el bitmap o metafile se modifica para encajar el nuevo rectángulo de ítem especificado.

Agrupar items / Desasociar grupos

Se puede agrupar items juntos de forma que se manejen como un solo ítem. Para hacer un grupo, seleccionar items en la composición gráfica y ejecutar el comando "**Edición / Grupo**". Se utiliza el comando "**Edición / Desasociar**" para restaurar los items del grupo seleccionado como separados.

Un grupo puede contener un dibujo. Un grupo puede contener también otro grupo.

Cuando se agrupan los items, su estilo ya no puede ser cambiado. Los items del grupo se siguen presentando, pero no se pueden utilizar (con doble click) para modificar el valor de las variables asociadas.

En la composición de lista un grupo aparece como una sola línea

A.18.2 Composición de lista

Pulsando este botón, se puede cambiar en cualquier momento entre composición gráfica y lista. También se puede utilizar el comando "**Opciones / Distribución de listas / gráficos**".

En la composición de lista, los items se muestran en un listado clásico (list box). El tamaño de cada ítem se calcula de acuerdo a su estilo gráfico. Los dibujos (bitmaps y metafiles) no son visibles desde la composición de lista. Existe una función de selección en la composición de lista, que deberá ser utilizada para definir el estilo de un ítem o para cambiar el valor de una variable. En este modo no están disponibles selecciones y comandos múltiples

Los items de la lista se pueden reordenar usando los comandos "**Edición / Subir / Bajar en la lista**". Se deberá seleccionar el ítem a mover en la lista.

A.18.3 Definiendo el estilo del ítem

El estilo gráfico y la configuración de un ítem existente se puede modificar, haciendo doble click sobre su símbolo en el área gráfica, o ejecutando el comando "**Edición / Establecer Estilo de ítem**" cuando el ítem está seleccionado en la composición gráfica o de lista. Se abre también la ventana de diálogo "Estilo" siempre que se añade un ítem al documento. Este diálogo agrupa las siguientes opciones a seleccionar por parte del usuario.

Estilo gráfico y configuración:

El estilo de presentación (solo texto, gráfico de barra, curva…) de un ítem se puede cambiar dinámicamente. Cuando se utilizan los colores de fondo y relieve, se pueden personalizar usando las cajas de selección correspondientes. Cuando el estilo es "icono booleano", se debe especificar la ruta de acceso de los ficheros .ICO correspondientes. Utilice los botones "**...**" próximos a estos controles para explorar los ficheros de iconos existentes en el disco.

Escala:

Este es el máximo valor que se puede presentar en los gráficos de barras y curvas. Para gráficos de barras y curvas, se utiliza el mismo valor absoluto para ambos ejes positivo y negativo.

Nombre de variable:

Cuando esté activo el campo "**Nombre**", pulsando el botón "**...**" próximo al control de edición se permite al usuario encontrar los nombres de las variables declaradas en el diccionario del proyecto.

Título (Caption):

Se puede presentar un título cercano al ítem gráfico en la composición gráfica. Se puede personalizar la posición del texto del título (superior, inferior, izquierda o derecha) y su contenido. El título puede ser cualquier combinación del nombre de la variable y su valor formateado como texto. La personalización del título no tiene ningún efecto sobre la composición de lista.

Control de variable (Command variable):

Si está activa la opción "Control de variable", el usuario puede modificar el valor de una variable enlazada durante la depuración haciendo doble click en el símbolo gráfico del ítem.

A.18.4 Comandos del menú "Fichero"

El menú "**Fichero**" contiene los comandos que permiten manejar el documento completo al usuario

El comando "**Nuevo**" del menú "**Fichero**" inicia la edición de un nuevo documento. ISaGRAF no limita el número de documentos definidos para un proyecto. Antes de editar un nuevo proyecto, se cierra el documento previamente abierto. SpotLight no se puede emplear para editar varios diagramas simultáneamente. Sin embargo, se pueden abrir simultáneamente múltiples ventanas SpotLight que se usan cada una para editar un documento diferente.

El comando "**Abrir**" del menú "**Fichero**" permite al usuario cerrar el documento editado actualmente e iniciar la edición de otro documento del proyecto actual. El documento nuevo seleccionado reemplaza el actual en la ventana de edición. Con objeto de limpiar el directorio de proyecto, cuando se selecciona un nuevo documento, se puede utilizar el botón "**Borrar**" para borrar algún fichero existente. Los ficheros de iconos y bitmaps referenciados en un diagrama no se borran cuando se borra el diagrama.

El comando "**Guardar**" del menú "**Fichero**" almacena en el disco el documento editado actualmente. Si es un documento nuevo sin título, el usuario debe asignarle un nombre antes de salvarlo. El nombre de un documento tiene que cumplir con las siguientes normas:

- La longitud del nombre no puede exceder 8 caracteres
- El primer carácter debe ser una letra
- Las restantes tienen que ser letras, dígitos o el carácter de subrayado '_'
- Los nombres son insensibles al uso de mayúsculas o minúsculas

El comando "**Guardar como**" del menú "**Fichero**" permite al usuario almacenar bajo otro nombre el documento actualmente editado.

A.18.5 Nota para usuarios de ISaGRAF V3.2

Spotlight puede leer gráficos y listas de diagramas de tiempo construidos con las herramientas de ISaGRAF V3.0 o V3.2. Dichos ficheros aparecen en la ventana de diálogo "**Abrir**", con la descripción de su origen. Los ficheros se pueden leer y modificar libremente con SpotLight.

Cuando se abre un gráfico de ISaGRAF V3.2, el documento se marca automáticamente como "Bloqueado". Eliminar la opción "Bloqueo" del menú "**Fichero**" si se quieren hacer cambios en el gráfico.

Cuando se abre un gráfico o una lista de diagramas de tiempo de ISaGRAF 3.2, SpotLight siempre propone guardarlos en el formato nativo de SpotLight. La ventana de diálogo "**Guardar como**" se abre sistemáticamente cuando se cierra un documento de este tipo.

A.19 Descarga (Upload)

ISaGRAF soporta la descarga de la aplicación almacenada en el objeto. El procedimiento de descarga se comunica con el objeto para recuperar el código fuente comprimido (EZS) incrustado y después restaurar el proyecto cargado en el entorno del banco de trabajo.
Se puede descargar el proyecto de un sistema objeto conectado si la versión del objeto es V3.22 o superior, y si se ha incrustado con la aplicación el código fuente comprimido. La incrustación de código fuente para descarga es una característica (feature) opcional.

A.19.1 Descargando de un proyecto

La ventana de diálogo "**Descargar proyecto**" se ejecuta desde el comando "**Ficheros**" del Gestor de Proyectos ISaGRAF. La descarga no hace referencia a un proyecto existente en el Banco de trabajo. El proyecto seleccionado actualmente en la lista de gestión de proyectos no está relacionado con el mecanismo de descarga. Para descargar la aplicación ejecutándose en el objeto se debe:

1- asegurarse de que el objeto está conectado adecuadamente
2- configurar los parámetros de comunicación de acuerdo con el enlace de conexión
3- presionar el botón "**Ejecutar**"

La descarga de la fuente comprimida incrustada (EZS) y posterior descompresión puede llevar varios segundos. Los mensaje en la caja de diálogo le informarán cuando la descarga se completa, o en caso de error.

El nombre que se utiliza para crear el proyecto ISaGRAF es el que se ha leído por comunicación en el objeto. Si este nombre ya está usado por un proyecto existente en el banco de trabajo, se preguntará si se sobrescribe o se selecciona un nombre no utilizado. Cuando la descarga se ha completado, no se puede cancelar el registro de las fuentes cargadas. El proyecto cargado está ahora disponible y puede ser abierto.

Errores posibles

Pueden ocurrir los siguientes errores cuando se descarga un proyecto. Se informa del error en la ventana de diálogo "Descarga".

- no se puede establecer comunicación con el objeto
- el objeto conectado es un sistema ISaGRAF anterior a la versión 3.22
- no hay aplicación corriendo en el objeto
- no hay EZS incrustado en el objeto

A.19.2 Parámetros de comunicación

Pulsando el botón "**Configurar**" permite al usuario definir los parámetros del enlace utilizados para comunicación en la descarga entre el banco de trabajo ISaGRAF y el sistema ISaGRAF objeto. Se debe asegurar que los parámetros configurados coincidan con los del objeto conectado antes de activar la descarga.

A.19.3 Preparar un proyecto para descarga

Si se quiere habilitar la descarga mas adelante, se debe informar al Generador de Código ISaGRAF que debe incrustar el código fuente comprimido con el código de la aplicación. Para ello, pulse el botón "**Cargar**" en la ventana de diálogo "**Opciones del compilador**". Otra ventana de diálogo permite seleccionar, como opción, la incrustación de código fuente comprimido. En este caso, sólo se incrustarán el número mínimo requerido de ficheros fuente. Seleccione otras opciones para incrustar también ficheros opcionales.

Nota importante: Las librerías no se cargan con el código fuente incrustado. Esto incluye las funciones, bloques de función, tarjetas E/S y equipos.

Para una mejor comprensión del mecanismo de Descarga, véanse los siguientes tópicos:

Cómo se almacena la fuente comprimida en el objeto
Necesidades de memoria en el objeto
Acerca del proyecto descargado
Aspectos de compatibilidad

- ***Ficheros opcionales***

Adicionalmente al mínimo código fuente requerido, se pueden incrustar los siguientes ficheros. Son opciones, ya que su selección implica requerimientos de memoria extras en el objeto.

Descriptor de proyecto

Si no esta incrustado el descriptor de proyecto, después de la descarga éste indicará solo la fecha de la descarga.

Protección con contraseña

La función de descarga no está protegida por una contraseña. Si se quiere tener el proyecto descargado protegido, se debe incrustar su sistema de protección por contraseña con el código fuente.

Comentarios para canales E/S no conectados

ISaGRAF da la posibilidad de introducir descripciones de texto para canales E/S no conectados. No seleccione esta opción si se trabaja solo con E/S conectadas.

Histórico de modificaciones

Esta es la historia global de modificaciones del proyecto.

Ficheros de diario

El fichero de diario de cada programa contiene notas de usuario más la historia de los mensajes de salida del compilador referentes al programa. La incrustación de ficheros de diario puede consumir una cantidad elevada de memoria en el objeto.

Listas de variables y diagramas de tiempo

Estos son los ficheros creados durante la depuración, y que contienen listas de nombres de variables para listas o monitorización de diagramas de tiempo.

Gráficos, iconos y bitmaps

Esto incluye los ficheros de gráficos ISaGRAF, más todos los iconos y bitmaps asociados, si están localizados en el directorio del proyecto. Advertencia: La incrustación de ficheros diarios puede consumir una cantidad elevada de memoria en el objeto.

A.19.4 Cómo se almacena la fuente comprimida en el objeto

La fuente comprimida incrustada (EZS) se almacena en el código generado con recursos. El recurso generado se llama "**EZS**". Cuando se selecciona incrustación de código fuente, no se puede utilizar dicho nombre para otro recurso. La incrustación de código fuente no lleva consigo ninguna limitación en la definición de los recursos. El fichero de definición de recursos escritos por el usuario no se ve afectado por la incrustación de código.

Véase la documentación ISaGRAF relativa al Generador de Código para mas detalles e información acerca de los recursos.

A.19.5 Requerimientos de memoria en el objeto

El código fuente comprimido incrustado requiere memoria extra para almacenarse con el código de la aplicación en el objeto. Una estimación aproximada general es que el EZS mínimo (sin opciones extra para incrustación de fuentes) tiene una vez y media el tamaño del código ejecutable. Esto significa que la incrustación de EZS multiplica el tamaño del código cargado por 2,5.

Pueden darse limitaciones especiales en algunos sistemas objetos basados en memoria segmentada. Dado que los EZS se almacenan como recursos en el código generado, deberán ser almacenados en el mismo segmento de datos que el código de la aplicación.

A.19.6 Acerca del proyecto descargado

El proyecto descargado contiene todos los ficheros y datos requeridos para la recompilación. Dependiendo de las opciones seleccionadas en operaciones de compilación previas, puede también contener ficheros auxiliares como descriptor de proyecto y ficheros de diarios de programa.

Se debe compilar (ensamblar) el proyecto antes de depurarlo o monitorizarlo.
Advertencia: debido a que ISaGRAF utiliza el sello de fecha de compilación para comparar aplicaciones, se informara al activar el depurador que las aplicaciones del banco de trabajo y el objeto tienen códigos de versión diferentes.

Nota importante: Las librerías no se cargan con el código fuente incrustado. Antes de recompilar la aplicación descargada debe asegurarse de que estén instaladas en el banco de trabajo ISaGRAF las librerías de funciones y bloques de función.

A.19.7 Aspectos de compatibilidad

La descarga está soportada por objeto y banco de trabajo ISaGRAF versión 3.22 y posteriores. Se han realizado extensiones al protocolo de comunicaciones para soportar la descarga.

No existe restricción en la incrustación de código fuente comprimido (EZS) en un objeto basado en sistemas ISaGRAF de las versiones 3.03 a la 3.21, ya que EZS se almacena en el código de la aplicación como los recursos estándar. Sin embargo la información incrustada no puede ser descargada en este caso ya que el objeto no soporta los servicios de comunicación requeridos.

A.20 Utilización de la Herramienta de Diagnóstico

La "**Herramienta de Diagnóstico**" es un subconjunto que depende del depurador de ISaGRAF. Permite al usuario final trabajar con un conjunto predefinido de variables, con vistas a examinar y controlar el proceso. El depurador de ISaGRAF es una herramienta muy potente que incluye funciones de alto nivel. La Herramienta de Diagnóstico proporciona una manera segura de controlar la operación final o el mantenimiento de la aplicación objeto. Para ejecutar la Herramienta de Diagnóstico de ISaGRAF directamente desde el grupo ISaGRAF del Gestor de Programas, hacer doble click sobre el siguiente icono:

Aparece la lista de proyectos existentes en una ventana de diálogo. Permite al usuario ejecutar el depurador ISaGRAF limitado sobre una aplicación ISaGRAF existente y previamente cargada. Pulsar el botón "**Aceptar**" para que el depurador limitado comience con el proyecto seleccionado. Pulsar el botón "**Cancelar**" para cerrar la ventana de diálogo. Se utiliza el comando "**Configuración**" para configurar el enlace de comunicación entre el banco de trabajo ISaGRAF y el PLC objeto. Para mayor información sobre este comando, véase el apartado titulado "**Gestión de Programas**" en este manual.

Nota: No se puede utilizar la Herramienta de Diagnóstico de ISaGRAF (depurador limitado) para cargar, interrumpir o actualizar la aplicación que se ejecuta en el PLC objeto. No se puede realizar operación alguna si el proyecto que está seleccionado en la ventana de diálogo de la Herramienta de Diagnóstico no es el mismo que está instalado y funcionando en el PLC.

Cuando el depurador limitado de ISaGRAF se ejecuta y está conectado correctamente a la aplicación objeto, se dispone de los siguientes comandos:

- Espiar listas de variables
- Espiar trazados de variables (diagramas de tiempo)
- Refrescar imágenes gráficas

Nota: Cuando se refresca una imagen gráfica, el usuario sólo puede modificar (escribir) aquellas variables que fueron definidas como "**Variables comando**" cuando se generó la imagen. Cualquier variable que esté incluida en una lista de variables o de diagramas de tiempo puede ser modificada por el usuario.

A.21 Utilización del simulador ISaGRAF

El simulador del *Kernel* ISaGRAF se inicia junto con el depurador cuando se ejecuta el comando "**Simular**" del menú "**Depurar**" en la ventana del Gestor de Programas. El simulador del *kernel* es un sistema objeto IsaGRAF completo que soporta las funciones estándares de ISaGRAF y todas las funciones y los bloques de función "C" de la librería estándar suministrada por CJ International. Las tarjetas de E/S se simulan gráficamente en una ventana. Se puede simular cualquier tipo de tarjeta de E/S. Las tarjetas definidas como "tarjetas virtuales" durante la conexión de E/S también aparecen en la ventana de simulación.

A.21.1 Enlaces con el depurador

El simulador del *kernel* soporta la comunicación plena con el depurador ISaGRAF, por lo que se puede utilizar cualquiera de las posibilidades de depuración durante la simulación. El simulador del *kernel* siempre opera sobre la aplicación ISaGRAF actual. Durante la simulación, los comandos del depurador "**Iniciar**", "**Parar**", "**Cargar**" o "**Actualizar**" no están disponibles. No se puede utilizar el depurador si el objeto "**SIMULATE**" no ha sido seleccionado en las opciones del compilador antes de construir el código destino. El cierre de la ventana del simulador implica que la ventana del depurador (así como cualquier otra ventana ISaGRAF que se haya abierto durante la sesión de depuración) se cierra también.

A.21.2 Simulación de E/S

Las tarjetas de E/S aparecen en la ventana de simulación, tituladas por su nombre y número de ranura. Se soporta cualquiera de los tipos estándares ISaGRAF de E/S (booleanas, analógicas o de mensaje). Se muestran los canales de las tarjetas de entrada con botones y campos especiales. Se muestran los canales de las tarjetas de salida con indicadores gráficos de estado y campos de datos.

Entradas booleanas: Las entradas booleanas están representadas por botones cuadrados de color verde. Se muestra el número del canal con el botón de E/S. El valor de entrada es VERDADERO cuando se pulsa el botón. Al hacer click sobre el botón, se cambia el valor de E/S correspondiente. Unicamente cuando está pulsado el botón de entrada, se puede utilizar el botón derecho del ratón para configurar la entrada.

Salidas booleanas: Las salidas booleanas están representadas por círculos pequeños. Se muestra el número del canal con la E/S. El valor de salida es VERDADERO cuando está resaltado el símbolo gráfico.

Entradas analógicas: Un canal analógico de entrada es un campo numérico simple en el que se puede introducir el valor de la entrada correspondiente. Hacer click sobre la ventana para visualizar el cursor y poder proceder a la introducción de un nuevo valor para el canal. No es necesario el uso de la tecla **INTRO** (Retorno) después de realizar la introducción de datos. Se pueden introducir las entradas analógicas bien en base decimal o bien en base hexadecimal. Utilizar los botones de arriba/abajo para aumentar o reducir al valor actual.

Salidas analógicas: Un canal de salida analógica es un campo de salida numérico. Se puede mostrar el valor de salida como un número bien decimal o bien hexadecimal. El usuario no puede actuar sobre un canal de salida.

Entradas de mensajes: Un canal de entrada de mensajes es un campo simple de texto en el que se introduce el valor de la entrada correspondiente. Hacer click sobre la ventana para visualizar el cursor y poder proceder a la introducción de un nuevo valor para el canal. No es necesario el uso de la tecla **INTRO** después de realizar la introducción de datos.

Salidas de mensajes: Un canal de salida de mensajes es un campo de salida de texto. El usuario no puede actuar sobre un canal de salida.

A.21.3 Componentes de la librería

El simulador ISaGRAF soporta todas las conversiones, funciones y bloques de función estándares proporcionadas por CJ International. Se soportan los siguientes objetos:

- *Funciones de conversión:*

bcd, scale

- *Funciones:*

abs, acos, ArCreate, ArRead, ArWrite, ascii, asin, atan, char, cos, delete, expt, find, insert, left, limit, log, max, mid, min, mlen, mod, mux4, mux8, odd, rand, replace, right, rol, ror, sel, shl, shr, sin, sqrt, tan, trunc

- *Bloques de función:*

average, blink, cmp, ctd, ctu, ctud, derivate, f_trig, hyster, integral, lim_alrm, r_trig, rs, sema, sr, stackint, tof, ton, tp

⚠ Por lo general, las conversiones, funciones “C” y bloques de función “C” definidos por el usuario no están integrados en el Simulador ISaGRAF. Normalmente, estos objetos están diseñados para utilizar los recursos de *software* y *hardware* del sistema objeto. El sistema Windows no suele disponer de este tipo de recursos. El Simulador ISaGRAF proporciona el siguiente comportamiento

estándar para cualquier conversión, función o bloque de función definido por el usuario:
- Cuando el simulador procesa una nueva conversión, la sustituye por una conversión “nula”. Esto significa que el valor físico de las variables analógicas es siempre igual al valor eléctrico (según se introduce o se representa en el panel del Simulador).
- Cuando el simulador ejecuta una nueva función o bloque de función “C”, no procesa operación alguna. No se establece el valor del resultado.

A.21.4 Opciones

Los comandos del menú "**Opciones**" le permiten al usuario controlar la representación de E/S en el panel del simulador. El usuario puede establecer o eliminar estas opciones en cualquier momento de la depuración.

- Si se selecciona la opción "**Presentación en color**", los canales de E/S se muestran como bitmaps en color. Si no se pueden distinguir los colores en determinados tipos de pantalla de cristal líquido, el usuario deberá deseleccionar esta opción para utilizar gráficos de entrada y salida en blanco y negro puro para los canales de E/S.

- Al seleccionar la opción "**Nombres de variables**", se muestra una pegatina al lado del canal de E/S, con el nombre de la variable de E/S que está conectada. Si se elimina esta opción, el usuario puede reducir el tamaño del panel de simulación.

- Cuando se selecciona la opción "**Valores hexadecimales**", los canales analógicos de entrada y salida se muestran y se introducen en formato hexadecimal.

- Al seleccionar la opción "**Siempre arriba**", la ventana del simulador está siempre visible, incluso si el foco de entrada está en otra ventana.

A.21.5 Salvar y recuperar estados de entrada

Utilizando el simulador de ISaGRAF, los canales de entrada se fuerzan durante las operaciones manuales, actuando en las salidas de señalización y los controles de edición del panel de simulación panel. Siempre se pueden utilizar los siguientes comandos del menú "**Herramientas**" para guardar y recuperar el estado de todas las variables de entrada:

Cargar esquema de entrada: Configura los valores de los canales de entrada con los valores almacenados en un fichero que has ido creado en el disco mediante el comando "Salvar esquema de entrada".

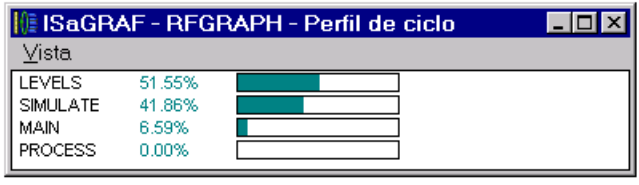
Guardar esquema de entrada: Salva el estado de los canales de entrada en un fichero, de modo que pueda recuperarse posteriormente utilizando el comando "Guardar esquema de entrada ". El fichero se almacena en disco en el directorio del proyecto, y por tanto se guarda con otros ficheros de proyecto por la utilidad de archivos de ISaGRAF.

Nota: Sólo los canales de entrada nombrados (los que tienen una variable conectada) se guardan en disco.

A.21.6 El perfil de ciclo

El perfil de ciclo de ISaGRAF es una potente herramienta de diagnóstico que muestra como se distribuye el tiempo de ciclo entre varios programas, funciones y bloques de función de una aplicación. Esta herramienta es muy útil para tener un diagnóstico rápido en la ejecución de la aplicación y lleva el programador a las partes del código que pueden necesitar optimizaciones.

El perfil de ciclo se ejecuta mediante el comando "**Herramientas / Perfil de ciclo**" en los menús de la ventana del simulador de ISaGRAF. Muestra para cada programa, función o bloque de función el porcentaje de tiempo de ciclo empleado en su ejecución.



Cuando la opción "**Vista / Media**" está activada, la información desplegada es una media de porcentajes calculada desde que se inició la aplicación, o desde la última vez que se ejecutó el comando "**Vista / Reset**".

Si no se activa la opción "**Vista / Media**" la información desplegada muestra las medidas hechas durante la ejecución del último ciclo. Se puede utilizar también esta característica cuando la aplicación está en el modo "**Ciclo a Ciclo**" para tener un grupo de medidas dependiendo del contexto de la aplicación.

Utilizar el comando "**Vista / Copiar**" para copiar los nombres de programa y porcentajes al portapapeles de Windows en formato ASCII. Después se pueden pegar los datos en documentos de texto u hojas de distribución comunes.

Notas importantes.

Estas no son medidas precisas. El cálculo de porcentajes está basado en instrucciones TIC, teniendo en cuenta varios tiempos de ejecución de

instrucciones. El cálculo no incluye el tiempo empleado en funciones y bloques de función "C".

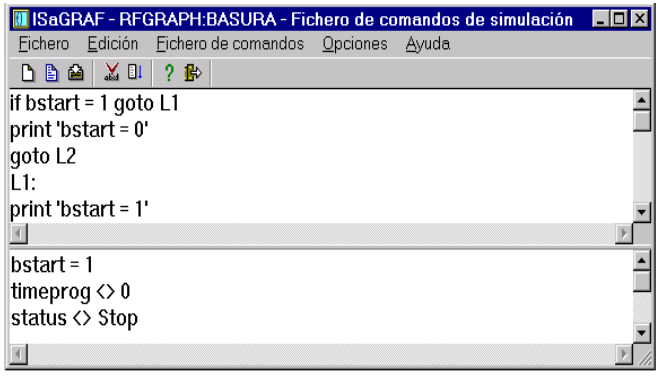
El valor dado para una función o bloque de función es la suma de todos los " tiempos de llamada " desde programas de aplicación en el mismo ciclo.

El tiempo de cálculo se basa en código TIC y no proporciona información fiable si el código de la aplicación actual está generado en lenguaje "C" y construido usando un compilador de "C".

A.21.7 Ficheros de comandos de simulación

El simulador de ISaGRAF incluye una herramienta para construir y ejecutar Ficheros de comandos de simulación. Un fichero de comandos de simulación está descrito con un lenguaje, parecido al ST, y se usa para automatizar pruebas con el simulador de ISaGRAF.

El fichero de comandos de simulación se ejecuta mediante el comando "**Herramientas / Ficheros de comandos de simulación**" de la ventana del Simulador. A continuación está el marco del editor de ficheros de comandos de simulación:



La ventana superior es un editor de texto donde se introducen las instrucciones del fichero de comandos de simulación. Se usa como otras herramientas de edición de texto de ISaGRAF e incluye características de alto nivel como selección con el ratón de un símbolo de variable. Se pueden utilizar los comandos del menú "**Opciones**" para configurar la anchura de la tabulación y seleccionar la fuente de caracteres.

La ventana inferior muestra todos los mensajes de salida cuando se ejecuta el simulador. La línea de separación entre ventanas puede moverse libremente para cambiar el tamaño de las ventanas. La ventana de salida se puede ocultar durante

la edición del fichero de comandos de simulación, pero se abre automáticamente cada vez que se ejecuta el fichero de comandos de simulación.

Edición de ficheros de comandos de simulación

Utilizar los comandos del menú **"Fichero"** para gestionar los ficheros de comandos de simulación:

Nuevo	crea un nuevo fichero de comandos de simulación sin título
Abrir	carga un fichero de comandos de simulación existente desde un archivo
Guardar	guarda el texto del fichero de comandos de simulación y el contenido de la ventana de salida a un disco, en el directorio de proyecto
Guardar como	Guarda el fichero de comandos de simulación con otro nombre

Se crean dos ficheros en directorio de proyecto de ISaGRAF para cada fichero de comando de simulación:

< *nombre_guion* >.SCC	texto del fichero de comando de simulación (instrucciones)
< *nombre_guion* >.SCO	contenido de la ventana de salida

donde <*nombre_guion*> es el nombre del fichero de comandos de simulación. Ambos ficheros son ficheros standard de texto, y se pueden abrir utilizando cualquier otro editor de texto.

 Cuando se edita un fichero de comandos de simulación se puede utilizar el comando "**Edición / Insertar símbolo**" para seleccionar un nombre de una variable declarada que se inserte en la posición actual.

Ejecución de ficheros de comandos de simulación

El fichero de comandos de simulación se debe verificar y compilar entes de ejecutarse. Si es necesario la verificación de la sintaxis se ejecuta automáticamente con un comando de "Ejecución". Utilizar los siguientes comandos del menú "**Fichero de comandos**":

 **Verificar**	verifica la sintaxis y compila el fichero de comandos de simulación
 **Ejecutar fichero de comandos**	Inicia la ejecución del fichero de comandos de simulación actualmente editado

En el caso de un nuevo fichero de comandos de simulación sin título, se debe salvar (y se debe introducir un nombre para ello) antes de verificarlo. En el caso de un fichero de comandos de simulación con nombre, el fichero de comandos de simulación se guarda automáticamente en el disco antes de comprobar la sintaxis.

Cuando el fichero de comandos de simulación se ejecuta los contenidos no pueden cambiarse. Se muestra un mensaje cuando se alcanza el final del fichero de

comandos de simulación. También se puede abortar su ejecución usando el siguiente comando del menú "**Fichero de comandos**":


Abortar fichero de comandos termina el fichero de comandos que se está ejecutando

La ejecución del fichero de comandos de simulación se ejecuta entre ciclos de sistema. En el caso de que se programe un bucle infinito en el ciclo, el simulador de ISaGRAF se asegura de que esté siempre se rompa, de modo que los ciclos de ISaGRAF se puedan seguir ejecutando, y no se bloqueen otras aplicaciones. El intérprete de ficheros de comandos de simulación de ISaGRAF decide romper la ejecución de un fichero de comandos de simulación si encuentra la misma etiqueta más de una vez en el mismo ciclo. La ejecución del fichero de comandos de simulación se puede también interrumpir normalmente con las instrucciones "Cycle" o "Wait".

Lenguaje de descripción de ficheros de comandos de simulación

El lenguaje de descripción de ficheros de comandos de simulación es un lenguaje de texto muy simple similar al ST, pero donde cada instrucción se introduce en una línea de texto separada, y no necesita terminarse con un punto y coma. Utilizar el siguiente botón de la barra de herramientas para colocar la lista de instrucciones disponible e insertar una palabra clave en la posición actual:


insertar instrucción (palabra clave y ayuda como comentarios)

Hay varios tipos de instrucciones. Primero está la asignación (forzada) de una variable:

:= asignación

Otras instrucciones permiten la salida de mensajes por la ventana de salida:

Print saca una cadena de texto o el valor de una variable
PrintTime saca la hora actual

Otras instrucciones se utilizan para sincronizar ficheros de comandos de simulación con el ciclo de ISaGRAF:

Cyle deja al simulador de ISaGRAF ejecutar un ciclo
Wait espera durante el tiempo especificado

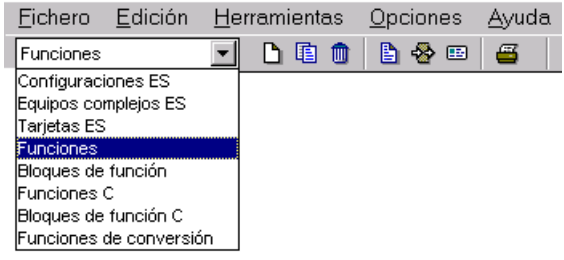
Se utilizan otras instrucciones para controlar el flujo de instrucciones en un fichero de comandos de simulación

Labels se puede colocar en cualquier lugar en un fichero de comandos de simulación
Goto salto incondicional a una etiqueta
If goto salto condicional a una etiqueta
End Termina el fichero de comandos de simulación

El lenguaje de ficheros de comandos de simulación no es sensible a Mayúsculas/minúsculas. Se pueden insertar comentarios al final de una línea de texto. Los comentarios se pueden escribir según las reglas de ST (entre caracteres "**(*"** y **"*)**"), o a continuación de un carácter punto y coma "**;**".

A.22 Utilización del Gestor de Librerías

Las librerías de ISaGRAF proporcionan una interfase estándar entre el desarrollo de automatizaciones y las capacidades de *software* y *hardware* del sistema ISaGRAF objeto. Existe una librería para cada tipo de interfase. El Gestor de Librerías del banco de trabajo ISaGRAF está orientado al proveedor de *hardware* o al ingeniero de *software*, quienes utilizan el gestor de librerías para describir la interfase de programación ISaGRAF de los objetos que crean.
El Gestor de Librerías del banco de trabajo ISaGRAF muestra los elementos que contiene una de las librerías ISaGRAF. En la parte izquierda de la ventana se encuentra la **lista de elementos** de la librería seleccionada. En la parte derecha se encuentran las **notas técnicas** (manual del usuario) del elemento que está seleccionado en la lista de elementos. Los menús del Gestor de Librerías contienen los comandos necesarios para crear, definir o modificar los elementos de la librería activa. El comando "**Fichero / Otra librería**" permite la selección de una de las librerías ISaGRAF. También se puede utilizar la ventana desplegable situada a la izquierda de la barra de herramientas para seleccionar una librería:



A.22.1 Gestión de elementos de librería

Utilizar los comandos del menú "**Fichero**" para crear elementos y trabajar con elementos existentes en la librería abierta.

Creación de elementos nuevos

El comando "**Nuevo**" del menú "**Fichero**" añade un elemento nuevo a la librería seleccionada. Se introduce el nombre del elemento nuevo, con base en las siguientes normas de denominación:

- la longitud máxima de un nombre es de **8** caracteres
- el primer carácter tiene que ser una **letra**
- los restantes caracteres tienen que ser **letras**, **dígitos** o el carácter de subrayado ' _ '

- los nombres de los elementos de librería son insensibles al uso de mayúsculas o minúsculas

Se asocia un comentario de texto a cada elemento de librería. Se introduce este comentario a la hora de crear el elemento. Cuando se crea un elemento nuevo, se tiene que introducir lo siguiente:
- su definición para una configuración de E/S,
- sus parámetros para una tarjeta de E/S, y
- su interfase de usuario para una función o un bloque de función.

Cuando se crea una conversión “C”, una función “C” o un bloque de función “C”, se genera automáticamente un marco completo de código fuente.

Trabajar con elementos existentes

El comando "**Fichero / Renombrar**" permite al usuario cambiar el nombre o el comentario del elemento que ha seleccionado en la lista de elementos. El comando "**Fichero / Copiar**" permite al usuario copiar el elemento que está seleccionado en la librería activa sobre otro elemento de la misma librería. Si el elemento destino ya existe, todo su contenido queda sobrescrito. Si el elemento destino no existe, se crea automáticamente. El comando "**Fichero / Borrar**" elimina el elemento seleccionado de la librería activa. Los siguientes componentes del elemento están gestionados por los comandos "**Renombrar**", "**Copiar**" y "**Borrar**":
- notas técnicas
- definición completa para una configuración de E/S
- parámetros para una tarjeta de E/S o un equipo complejo
- definición de interfase para una función o un bloque de función
- código fuente para una función o un bloque de función, escrito en un lenguaje IEC
- código fuente para una conversión “C”, una función o un bloque de función

⚠ Si el elemento es una conversión “C”, una función “C” o un bloque de función “C”, los comandos "**Renombrar**" o "**Copiar**" no actualizan el nombre automáticamente en el código fuente asociado.

⚠ Si el elemento es una función escrita en un lenguaje IEC, los comandos "**Renombrar**" o "**Copiar**" no cambian el nombre del parámetro de retorno.

Configuración de protección por contraseña

El comando "**Fichero / Establecer contraseña**" permite al usuario definir la protección por contraseña del elemento seleccionado en la librería que está abierta. Para mayor información sobre los niveles de contraseña y la protección de datos, véase la sección titulada "**Protección por contraseñas**", localizada al final de la primera parte de este manual. Las contraseñas sólo guardan relación con el elemento seleccionado. No influyen en otros elementos de las librerías ISaGRAF.

Compilación de funciones y bloques de función

Cuando se selecciona la librería de funciones o bloques de función escritos en lenguajes IEC, se utiliza el comando "**Verificar (compilar)**" del menú "**Fichero**" para comprobar la sintaxis del elemento seleccionado y crear su código objeto. Las funciones y los bloques de función escritos en lenguajes IEC tienen que compilarse

sin errores antes de que puedan ser utilizados en un proyecto ISaGRAF. Este comando no tiene efecto si se ha seleccionado otra librería.

Notas técnicas

El comando "**Edición / Nota técnica**" permite al usuario introducir las notas técnicas relativas al elemento que está seleccionado en la librería activa. Se introducen las notas técnicas con el editor de textos ISaGRAF. Las notas técnicas de un elemento son la **guía del usuario** y serán consultadas por el usuario del elemento durante su integración en un proyecto ISaGRAF. Las notas técnicas sobre cómo utilizar el elemento deben contener la descripción de sus funciones principales y una explicación detallada de su interfase de programación y parámetros, así como de su contexto y sus límites.

El comando "**Herramientas / Formato estándar de nota** " le permite al usuario definir un formato estándar de texto para todos los elementos de la librería seleccionada. Cuando se editan las notas técnicas para un elemento nuevo, se utiliza este formato como marco principal. Esto permite al usuario la optimización de sus tareas de edición de notas técnicas.

Parámetros

Los parámetros de un elemento describen la **interfase** que existe entre las operaciones de cómputo que proporciona el elemento y el uso del elemento en una aplicación ISaGRAF. Los parámetros tienen un significado diferente para cada tipo de elemento de librería.

Los parámetros de una configuración de E/S definen el conjunto completo de tarjetas de E/S de la configuración y los nombres de variables por defecto que se utilizan para los canales de E/S. Los parámetros de una tarjeta de E/S o un equipo complejo definen la configuración física y lógica de la tarjeta. Los parámetros de una función o un bloque de función definen la interfase del elemento, de acuerdo con las convenciones de llamada del lenguaje ST. No existen parámetros para una función de conversión, ya que utiliza una interfase estándar predefinida.

Código fuente "C"

El banco de trabajo ISaGRAF permite al programador gestionar el código fuente de una conversión, función o bloque de función procedente de la librería. El código fuente de una función o de un bloque de función escrito en un lenguaje IEC es un texto o un diagrama descrito en el lenguaje asociado a la función. El código fuente de los componentes "C" (funciones "C", bloques de función "C" y funciones de conversión) se dividen en dos ficheros independientes: una **cabecera de fuente C** que contiene la definición exacta de la **interfase**, de acuerdo con la definición de parámetros del elemento, y un fichero de **código fuente C** que contiene la implementación operativa del elemento.

El banco de trabajo ISaGRAF genera el fichero de código fuente cuando se crea un nuevo elemento de librería. También crea y actualiza la cabecera de fuente, basado en la definición de los parámetros. El programador puede utilizar el editor de textos ISaGRAF para completar el fichero de código fuente.

Archivar los elementos de librería

El comando de menú "**Herramientas / Archivo**" ejecuta el gestor de archivos de ISaGRAF para guardar, gestionar o recuperar elementos de librería. Primero se necesita seleccionar una librería antes de ejecutar el comando "**Archivo**". El gestor de archivos muestra la lista de elementos para una única librería al mismo tiempo.

A.22.2 Configuración de E/S

La librería ISaGRAF de configuración de E/S proporciona una forma sencilla de inicializar los nuevos proyectos ISaGRAF con una configuración de E/S predefinida. Una configuración de E/S define:

- un conjunto de tarjetas de E/S
- los valores por defecto de los parámetros de las tarjetas de E/S
- los nombres por defecto de los canales de E/S

Cuando se crea un nuevo proyecto ISaGRAF con una configuración de E/S procedente de la librería, se establece automáticamente la conexión de E/S correspondiente y las variables de E/S que correspondan a los nombres de canales se declaran automáticamente en el diccionario del proyecto.

Para realizar la definición de una configuración de E/S, se utiliza la herramienta de conexión de E/S de ISaGRAF (la misma herramienta que se utiliza dentro de un proyecto). Para mayor información sobre la manera de utilizar esta herramienta, véase la sección titulada “Conexiones de E/S”, de este manual. Cuando se inserta una nueva tarjeta de E/S en la configuración, se declaran todos los canales de la tarjeta nueva con nombres estándares por defecto. El nombre estándar por defecto de un canal de E/S tiene el siguiente formato:

<dirección><tipo><número_ranura>_<número_canal>

El primer carácter indica la dirección del canal de E/S:
"**I**" ..... canal de entrada
"**O**" ..... canal de salida
El segundo carácter indica el tipo de canal de E/S:
"**X**"..... booleano
"**D**"..... analógico
"**M**" ..... mensajes

A continuación se muestran algunos ejemplos de nombres estándares de canal de E/S:
IX0_7..... entrada booleana - tarjeta #0 - canal #7
QD2_4..... salida valores enteros - tarjeta #2 - canal #4

Se utiliza el comando "**Definir canal / parámetro** " del Editor de Conexiones de E/S para modificar el nombre por defecto que se asocia a un canal de E/S.

A.22.3 Equipos complejos de E/S

Todos los canales de un única tarjeta son del mismo tipo (booleanos, analógicos o mensaje) y dirección (entrada o salida). Un equipo complejo de E/S representa un dispositivo de E/S con canales de diferentes tipos o direcciones. Los equipos complejos de E/S están representados en forma de una lista de tarjetas individuales de E/S. Utiliza una sola ranura en la lista del chasis de conexiones de E/S.

Para definir un equipo complejo de E/S, el usuario tiene que especificar la lista de tarjetas individuales que define al equipo de E/S. También tiene que introducir los parámetros detallados de cada tarjeta individual. Se introduce la lista de tarjetas individuales de E/S por medio de una ventana de diálogo.

Se utiliza el botón "**Añadir**" para añadir una tarjeta individual al final de la lista actual. El botón "**Insertar**" se utiliza para insertar una tarjeta nueva delante de la selección actual en la lista. El botón "**Borrar**" elimina de la lista la tarjeta que está seleccionada. Los botones "**Renombrar**" y "**Parámetros**" se utilizan para cambiar el nombre y los parámetros de la tarjeta individual que está seleccionada. Para una explicación completa de los parámetros de una tarjeta individual, véase la siguiente sección. Un equipo complejo de E/S puede agrupar hasta **16** tarjetas individuales de E/S. El nombre de una tarjeta individual (dentro de un equipo de E/S) no puede superar los **8** caracteres.

A.22.4 Tarjetas de E/S

La librería ISaGRAF de tarjetas de E/S define una interfase estándar entre las variables de una aplicación y el *hardware* objeto. Durante la descripción de la aplicación, se conectan todas las variables de E/S a los canales de las tarjetas objeto de E/S. Se definen las tarjetas ISaGRAF de E/S con un **nombre** y una "**clave OEM**" que identifica al **proveedor**. Existen otros parámetros que describen la topología de la tarjeta de E/S (número de canales, dirección y tipo de canal) y su configuración *hardware* y *software*.

Parámetros de tarjetas de E/S

Existen dos tipos diferentes de parámetros para una tarjeta de E/S: los parámetros comunes que se definen para cualquier tarjeta de la librería ISaGRAF, y los parámetros OEM que son específicos para la implementación de la tarjeta, facilitados por el proveedor del *hardware*. Los parámetros comunes se introducen en la parte superior de la ventana en la que se definen los parámetros de la tarjeta de E/S. Estos parámetros (junto con el nombre de la tarjeta de E/S) identifican la interfase ISaGRAF estándar de las tarjetas de E/S.

La "**Clave OEM**" es un número simple que define al **proveedor del *hardware***. Todas las tarjetas que estén definidas por el mismo proveedor tienen que tener la misma clave OEM. La Clave OEM es una **palabra sin signo de 16 bits** que se

introduce en formato **hexadecimal**. La Clave OEM que se ha reservado para **CJ International** es "**1**".

Los parámetros principales definen la topología de la tarjeta de E/S. El **número de canales** define el número de canales disponibles en la tarjeta. El **tipo** de la tarjeta es el tipo de variables que pueden conectarse a los canales de la tarjeta. La **dirección** define si las variables que se conectan a la tarjeta son variables de **entrada** o de **salida**.

Nota: No se pueden agrupar las variables de E/S de diferentes tipos o direcciones en una misma tarjeta ISaGRAF de E/S. Para ello se requiere un equipo complejo de E/S.

Parámetros OEM

Se introducen los parámetros OEM en la parte inferior de la ventana de definición de los parámetros de la tarjeta de E/S. Estos parámetros están definidos por el proveedor de la tarjeta de E/S y son específicos para ésta. Una tarjeta puede tener un máximo de **16** parámetros OEM, aunque también existe la posibilidad de que no tenga ninguno. El Gestor de Librerías ISaGRAF le permite al proveedor que defina la identificación y el formato de cada parámetro, así como la manera en la que el programador de automatización debe introducirlo.

La subventana de la izquierda contiene la lista de parámetros OEM. Cada parámetro está identificado con un **nombre** y un **número** lógico entre **0** y **15**. La zona de la derecha contiene la descripción detallada del parámetro que está seleccionado en la lista. Se selecciona un parámetro en la lista para poder acceder a su descripción completa. Pulsar el botón "**Borrar**" para volver a iniciar la descripción del parámetro y eliminarlo de la lista de parámetros. Advertencia: no se puede "deshacer" este comando.

Se utiliza el nombre de un parámetro para identificar el campo de entrada correspondiente durante la conexión de una tarjeta de E/S, si el campo tiene que estar definido por el operador de la automatización. Los nombres de los parámetros tienen que cumplir con las siguientes normas:

- la longitud del nombre no puede superar los **16** caracteres
- el primer carácter tiene que ser una **letra**
- los restantes caracteres tienen que ser **letras**, **dígitos** o caracteres de subrayado ' _ '

El tipo de un parámetro define el **formato interno** del parámetro y su **formato de entrada** durante la conexión de E/S de la aplicación. Se dispone de los siguiente formatos internos:

word ..... palabra de 16 bits, sin signo
long ..... palabra de 32 bits, sin signo
word hexa ..... palabra de 16 bits, sin signo
long hexa ..... palabra de 32 bits, sin signo
booleano ..... palabra de 16 bits, sin signo (sólo se usa el bit más bajo)
carácter ..... palabra de 16 bits, sin signo (sólo se usa el byte más bajo)
cadena..... matriz de 16 bytes que contiene una cadena de terminación nula
flotante ..... valor flotante de 32 bits, de simple precisión

Se dispone de los siguientes formatos de entrada:
word ..... palabra decimal, sin signo
long ..... palabra decimal larga
word hexa ..... palabra hexadecimal, sin signo
long hexa ..... palabra hexadecimal larga, sin signo
booleano ..... "verdadero" o "falso"
carácter ..... carácter simple
cadena..... cadena ASCII (sólo 15 caracteres como máximo)
flotante ..... valor flotante de simple precisión

Se utiliza la subventana "**Acceso**" para definir la manera que tiene el usuario final de acceder al parámetro. Si se selecciona la opción "**Definido por usuario** ", se muestra el parámetro como un campo de entrada durante la conexión de la tarjeta de E/S. Se utiliza el valor por defecto de los parámetros OEM como el valor por defecto para la edición de parámetros. Si se selecciona la opción "**Oculto**", el parámetro es una constante y no aparece en la ventana de conexión de la tarjeta de E/S. El valor por defecto de los parámetros OEM define el valor del parámetro constante. La opción "**Solo lectura**" indica que el parámetro es visible para el usuario, pero no puede ser modificado. Se utiliza su valor por defecto como un valor constante.

A.22.5 Funciones y bloques escritos en lenguajes IEC

ISaGRAF maneja una librería de funciones y bloques de función escritos en lenguajes IEC. Se dispone de los siguientes lenguajes para describir estas funciones o bloques de función: **FBD** (Diagrama de Bloques de función), **LD** (Diagrama de Escalera (Contactos)), **ST** (Texto Estructurado) o **IL** (Lista de Instrucciones). Se pueden mezclar los lenguajes LD y FBD en un mismo diagrama. No se puede utilizar el lenguaje **SFC** (Diagrama de Funciones Secuenciales) para describir una función o un bloque de función en una librería. Se selecciona el lenguaje asociado a un elemento de librería cuando se crea la función, y no se puede cambiar posteriormente.

Compilación

Las funciones y los bloques de función que están definidos en la librería tienen que ser compilados (verificados) antes de que puedan ser utilizados en un proyecto ISaGRAF. No hay que realizar más cambios a nivel de librería en relación a funciones y bloques de función. Los elementos de la librería aparecerán automáticamente en el menú de selección cuando se utiliza el editor de gráficos LD/FBD en un proyecto.

⚠ Las funciones definidas en la librería pueden invocar a otras funciones de la librería. Sin embargo, el sistema ISaGRAF **no soporta la recursividad** en la invocación de funciones. Un bloque de función escrito en lenguaje IEC no puede invocar a otros bloques de función (ni en lenguaje IEC ni en lenguaje “C”).

Introducción de código fuente

Se introduce el código fuente de una función o un bloque de función de la librería por medio de las herramientas ISaGRAF estándares: el editor de gráficos para los programas LD o FBD y el editor de textos para los programas ST o IL. Para mayor información sobre estas herramientas, véanse las secciones correspondientes de este manual. Se puede invocar al Generador de Código ISaGRAF directamente desde la ventana de edición de gráficos o texto, con la finalidad de compilar el código fuente de una función o bloque de función de la librería.

Diccionario de variables locales

Una función o bloque de función de librería puede tener variables locales y palabras definidas locales. Para acceder a la declaración variable, el usuario tiene que ejecutar las ordenes del comando "**Diccionario**" del menú "**Fichero**", en la ventana de edición, mientras edita el código fuente de la función.

⚠ Una función o bloque de función de la librería no puede acceder a las variables globales o a las instancias de los bloque de función. Se tienen que inicializar las variables locales de una función en el cuerpo de la función.

Las variables locales de un bloque de función escrito en lenguaje IEC son copiadas (instanciadas) cada vez que se utiliza el bloque en un proyecto. Las variables locales de una instancia mantienen sus valores entre una invocación y otra.

Definición de la interface

Las funciones y los bloques de función pueden poseer hasta **32** parámetros (de entrada o salida). Una función siempre tiene un único parámetro de retorno, que debe tener el mismo nombre que la función para cumplir con las normas de escritura del lenguaje ST.

La lista que aparece en la parte superior izquierda de la ventana muestra los parámetros, ordenados de acuerdo con el modelo de invocación: los parámetros de llamada en primer lugar y los de retorno en el último lugar. La parte inferior de la ventana muestra una descripción detallada del parámetro que está seleccionado en la lista. Se puede utilizar cualquiera de los tipos de datos ISaGRAF para un parámetro. Los parámetros de retorno tienen que estar ubicados después de los de llamada, dentro de la lista. Los nombres de los parámetros tienen que cumplir con las siguientes normas:

- la longitud máxima de un nombre es de 16 caracteres
- el primer carácter tiene que ser una letra
- los restantes caracteres tienen que ser letras, dígitos o el carácter de subrayado '_'
- los nombres son insensibles al uso de mayúsculas o minúsculas

Se utiliza el comando "**Insertar**" para insertar un parámetro nuevo delante del parámetro seleccionado. Se utiliza el comando "**Borrar**" para borrar el parámetro seleccionado. El comando "**Ordenar**" reordena (clasifica) los parámetros de forma automática, colocando los parámetros de retorno al final de la lista.

A.22.6 Funciones y bloques de función “C”

Las funciones y los bloques de función “C” son **funciones de cómputo** que se invocan desde la aplicación de automatización, según la interfase de invocación de funciones del lenguaje ST.

Las funciones son procesos **síncronos**. Se suspende la aplicación ISaGRAF objeto durante la ejecución de funciones. Los bloques de función asocian operaciones y datos estáticos ocultos. Por ejemplo, un bloque de función de “contador” representa tanto la operación de contaje como su resultado. Se pueden utilizar las funciones y los bloques de función para completar las capacidades estándares de los lenguajes de automatización, o para acceder a los recursos del sistema.

Se utiliza la ventana de definición de parámetros para definir el nombre y tipo de cada parámetro de llamada o retorno de la función o bloque de función. Se utilizan los comandos del menú "**Edición**" para definir los parámetros de la función o el bloque de función seleccionado. Una función puede tener hasta **31** parámetros de llamada, y siempre tiene un **único** parámetro de retorno. Un bloque de función puede tener hasta **32** parámetros y cualquier combinación de parámetros de llamada y retorno. A continuación se muestra la correspondencia entre tipos ISaGRAF y tipos “C”:

BOOLEANO	unsigned long	palabra de 32 bits, sin signo: 1=verdadero / 0=falso
ANALOGICO	long	palabra entera de 32 bits, con signo
REAL	float	valor flotante, simple precisión
TEMPORIZADOR	unsigned long	palabra entera de 32 bits, sin signo (la unidad es 1 ms)
MENSAJE	char *	cadena de caracteres

Cuando se transmite un valor de mensaje a una función o un bloque de función “C”, no puede contener caracteres nulos. La cadena que se transmite al código “C” tiene la terminación nula.

Para mayor información sobre la manera de gestionar el código fuente en “C” de una función o un bloque de función y de cómo integrar un elemento nuevo en el sistema objeto de ISaGRAF, véase la Guía de Usuario de Objeto ISaGRAF.

A.22.7 Funciones de conversión

Una función de conversión es una función “C” que invoca el Gestor de E/S de ISaGRAF cada vez que se da de entrada al, o salida del, proyecto a las variables analógicas que utilizan esta conversión.

La función crea la relación entre el **valor eléctrico** de la variable (el valor leído por el sensor de entradas o enviado al dispositivo de salida) y su **valor físico** (el valor utilizado en las sentencias de la aplicación). La función se divide, por tanto, en dos

partes: la conversión de entradas y la conversión de salidas. El Gestor de Librerías de ISaGRAF le permite al usuario controlar el código fuente en “C” de una función de conversión.

Se puede utilizar la función de conversión para las variables analógicas **enteras** o **reales**. Esto implica que la interfase de la función de conversión siempre esté definida por valores flotantes. La interfase es siempre la misma para cualquier función de conversión. Se realiza la definición en “C” de esta interfase en el fichero de definición "**TACN0DEF.H**".

Para mayor información sobre la manera de gestionar el código fuente en “C” de una función de conversión y de cómo integrar un elemento nuevo en el sistema objeto de ISaGRAF, véase la Guía de Usuario de Objeto ISaGRAF.

A.23 Utilización de la utilidad de Archivo

El gestor de archivos de ISaGRAF permite al usuario guardar los proyectos y las librerías ISaGRAF en disquetes o en un directorio de copia de seguridad. El administrador de archivos de ISaGRAF es una ventana de diálogo que se puede llamar desde el administrador de proyectos de ISaGRAF o desde el administrador de librerías de windows.

⚠ Para crear y mantener unos archivos fiables, se recomienda la observación de las siguientes directrices:
Escribir el nombre y la descripción del objeto guardado en la pegatina del disco
No guardar proyectos y librerías en el mismo disquete
No guardar proyectos diferentes en el mismo disquete

A.23.1 Llamando al gestor de archivos

La ventana de diálogo "**Archivo**" puede llamarse desde el menú "**Herramientas / Archivo**" de la ventana del gestor de proyectos para salvar o recuperar bien un proyecto, bien un dato común..

La ventana de diálogo puede ejecutarse desde el comando "**Herramientas / Archivo**" del gestor de librerías de ISaGRAF para guardar o recuperar elementos de la librería actualmente seleccionada en la ventana del gestor de librerías.

- ***Proyectos***

  Siempre se guardan los proyectos en su forma íntegra. Se guardan todos los componentes del proyecto (ficheros fuente de programas, código objeto y código ejecutable de aplicaciones) en el mismo fichero de archivo. Seleccionar la opción "**Comprimir**" para reducir el tamaño del archivo de proyecto.

- ***Elementos de librería***

  Se pueden guardar individualmente los elementos de las librerías ISaGRAF. Se guardan todos los componentes de un elemento de librería (notas técnicas, definición, interfase, código fuente, etc.) en el mismo fichero de archivo.

- ***Datos comunes***

  El comando "**Herramienta / Archivo / Datos comunes**" de la ventana del administrador de proyectos permite al usuario realizar copias de seguridad o restaurar los datos de “rango común” que existan en el banco de trabajo ISaGRAF. Este comando no actúa sobre las librerías ISaGRAF. Se pueden copiar los siguientes ficheros con este comando:

  common.eqv.....palabras definidas comunes

oem.bat .....fichero de comando MS-DOS definido por el usuario

Se guardan estos ficheros en el disco de archivo, uno por uno, en su forma original. Nunca se comprimen los ficheros de archivo correspondientes.

A.23.2 Opciones

La ruta que se usa para los archivos de ISaGRAF se despliega en la parte de abajo de la ventana de diálogo. Se pulsa el botón "Explorar" para ver los discos y seleccionar otro disco de archivo y directorio.

Cuando se selecciona la opción "**Comprimir**" todos los ficheros de archivo que se crean durante un procedimiento de "**Copia de seguridad**" se comprimen. Esta opción es muy útil a la hora de reducir el tamaño de un fichero grande de proyecto, para poder guardarlo en un único disquete. No suele necesitarse la compresión de archivos para los componentes de librerías. El gestor de archivos de ISaGRAF reconoce automáticamente el estado (comprimido o no) de un fichero de archivo cuando restaura el archivo. De ahí se desprende que la opción de "**Comprimir**" no tiene efecto para un procedimiento de "**Recuperar**"

A.23.3 Realizar copias de seguridad (archivar) y recuperaciones

La lista "**Banco de trabajo**" (a la izquierda) muestra los objetos que existen en el banco de trabajo ISaGRAF que está instalado en el disco duro. La lista "**Archivo**" (a la derecha) muestra los objetos que están guardados en el disco y directorio especificado.

- ***Copias de seguridad***

  Para archivar un objeto, se selecciona el objeto en la lista de la **izquierda** (objetos pertenecientes al banco de trabajo ISaGRAF) y se pulsa el botón "**Archivar**". Se puede seleccionar más de un objeto de la lista. Se deshabilita el botón "**Archivar**" al seleccionar un elemento de la lista de la **derecha** (modo restaurar).

- ***Operación de restaurar***

Para copiar un objeto del archivo al banco de trabajo ISaGRAF, se selecciona el objeto en la lista de la **derecha** (objetos del archivo) y se pulsa el botón "**Recuperar**". Se puede seleccionar más de un objeto de la lista. Se deshabilita el botón " **Recuperar**" al seleccionar un elemento de la lista de la **izquierda** (modo realizar copia de seguridad).

A.23.4 Ficheros de archivo

El Gestor de Archivos ISaGRAF crea un único fichero de archivo para cada objeto guardado. El fichero de archivo tiene el mismo nombre que el objeto. La extensión del fichero indica su tipo. Se utilizan las siguientes extensiones:

.pia ..... proyecto
.bia ..... tarjeta de E/S
.iia ..... función en lenguaje IEC
.aia ..... bloque de función en lenguaje IEC
.uia ..... función C
.fia ..... bloque de función C
.cia ..... función de conversión C
.ria ..... configuración de E/S
.xia ..... equipo de E/S

A.24 Impresión de un documento completo

El Generador de Documentos de ISaGRAF le permite al usuario elaborar e imprimir un documento completo para el proyecto seleccionado. Se puede llamar desde el comando "**Proyecto / Imprimir**" del gestor de proyectos de la ventana de programas para imprimir un documento completo. El Generador de Documentos se puede ejecutar también desde el comando "**Imprimir** de todos los otros editores de ISaGRAF para imprimir un único documento de ISaGRAF. Sin embargo el generador de documentos proporciona las mismas características en ambos casos. Se utilizan los comandos del menú "**Edición**" para definir los elementos del proyecto que tienen que insertarse en el documento. De esta manera, el usuario elabora la "Tabla de contenidos" del documento en cuestión. Se puede insertar en el documento del proyecto cualquier información sobre el proyecto (programas, variables, opciones, conexiones de E/S, etc.). En este documento no puede aparecer información relativa a otro proyecto o información procedente de las librerías ISaGRAF.

El comando "**Fichero / Imprimir**" genera el documento y lo envía a la impresora, de acuerdo con la tabla de contenidos especificada. La tarea "Imprimir" puede tardar unos minutos en construir y formatear el documento. Se recomienda esperar a que concluya el mensaje de "Printing Job" en la ventana del Generador de Documentos antes de ejecutar otros comandos del banco de trabajo ISaGRAF. La construcción del documento completo podría requerir un espacio grande en el disco duro. Se mostrará un mensaje de error si se llena el disco, en cuyo caso el usuario bien tendrá que liberar espacio de disco mediante la eliminación de ficheros, o bien tendrá que reducir el tamaño del trabajo de impresión. Al ejecutarse el comando "**Imprimir**", aparece una ventana de diálogo que permite al usuario introducir una anotación que describe el comando de impresión. Estas anotaciones se guardan en un fichero histórico y se imprimen en la primera página de cualquier documento futuro (incluyendo el actual).

A.24.1 Personalización de la tabla de contenidos

El menú "**Edición**" contiene los comandos necesarios para definir la "Tabla de Contenidos" del documento. Una selección de comandos permite al usuario utilizar una tabla por defecto (con todos los componentes del proyecto), construir una tabla específica (sólo con determinados componentes) o desplazar puntos dentro de la tabla y modificarla.

La lista por defecto

El comando "**Lista por defecto**" del menú "**Edición**" define una tabla de contenidos estándar para el documento que contiene todos los componentes del proyecto. La tabla estándar consiste en:

- Descriptor del proyecto
- Árbol jerárquico (enlaces entre programas)
- Código fuente para cualquier programa
- Fichero diario para cualquier programa
- Definiciones comunes
- Definiciones globales
- Definiciones locales para cualquier programa
- Variables globales
- Variables locales para cualquier programa
- Opciones de la aplicación
- Conexión E/S
- Listas espía
- Tablas de conversión
- Referencias cruzadas resumidas
- Referencias cruzadas detalladas
- Resumen de declaraciones
- Mapa de direcciones de red
- Historial de modificaciones

La tabla de contenidos se puede salvar en disco usando el comando "**Fichero / Guardar**". Este comando se inhabilita cuando el generador de documentos se ejecuta desde un editor de ISaGRAF para imprimir un único documento.

Cortar y pegar

Utilicen los comandos "**Edición / Cortar**" y "**Edición / Pegar**" para desplazan los items dentro de la lista, y así se pueda personalizar el orden de la tabla. El Generador de Documentos permite la selección múltiple, por lo que se puede cortar y pegar un grupo de items.

Eliminación de la tabla

El comando "**Edición / Borrar**" restablece la tabla de contenidos para que se pueda reconstruir por completo mediante la inserción de items individuales.

Inserción de items en la tabla

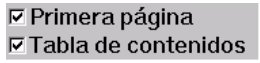
Al ejecutarse el comando "**Edición / Insertar**", se abre la ventana de diálogo "**Añadir ítem**". Esta ventana le permite al usuario insertar items (componentes del proyecto) en la tabla de contenidos.

Para un ítem relativo a un programa, utilizar la ventana desplegable "**Programa**" para seleccionar un nombre de programa. Pulsar el botón "**Añadir**" para insertar el punto seleccionado en la tabla de contenidos. El mismo ítem sólo puede aparecer una vez en la tabla.

A.24.2 Opciones

Se utilizan los comandos del menú "**Opciones**" para definir y personalizar el formato del documento generado.

Otras opciones están disponibles directamente mediante botones de la ventana del documento generador.



Cuando se configura la opción "**Primera página**", una cabecera de página se imprime al comienzo del documento, conteniendo el título del proyecto y la historia de las impresiones. Cuando no se selecciona esta opción el primer objeto a imprimir empieza en la primera página.

Cuando se activa la opción "**Tabla de contenidos**" , se imprime una tabla de contenidos al final del documento generado.

Ambas opciones están inhabilitadas cuando el generador de documentos se ejecuta desde un comando "**Imprimir**" de un editor de ISaGRAF (programa, diccionario...).

Diagramas SFC

La opción "**Separar niveles SFC**" indica al sistema que, para cada programa SFC, imprima primero el Nivel 1 (diagrama SFC y comentarios) y después la programación de Nivel 2. Si no se selecciona esta opción, los Niveles 1 y 2 aparecen juntos en la misma impresión.

Formato de página

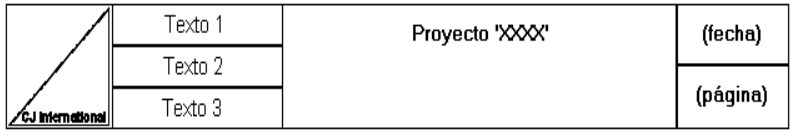
Se utiliza el comando "**Formato de página**" del menú "**Opciones**" para definir los principales parámetros sobre los cuales actúa el Generador de Documentos a la hora de formatear una página. Se pueden especificar los siguientes parámetros:

- **Margen izquierdo**: (1 ó 2 centímetros, o sin margen)

Borde de página: cuando se selecciona esta opción, se dibuja un borde alrededor de cualquier página impresa.

Plantilla de título de página

Se utiliza el comando "**Título de página**" del menú "**Opciones**" para definir el contenido del cajetín de título que se imprime a pie de cualquier página. Este cajetín tiene la siguiente distribución típica de elementos:



La primera línea del título principal (con el nombre del proyecto ISaGRAF), la fecha actual y el número de página están generados automáticamente por el Gestor de Documentos y no se pueden cambiar.

Las tres líneas de texto situadas en la parte izquierda del cajetín (text1, text2, text3) y la segunda línea del título principal están definidas por el usuario. El usuario también puede cambiar el logotipo que aparece en el extremo izquierdo del cajetín. Para utilizar otro logotipo, el usuario tiene que especificar la ruta de acceso de un fichero bitmap (.**BMP**). Esta imagen puede ser de cualquier tamaño. Se estirará o se encogerá en función de las dimensiones exactas de la página impresa. Al hacer click sobre el área del logotipo, dentro de la ventana de diálogo, se muestra la imagen recién especificada. El fichero de imagen tiene que estar en el disco (en el directorio y con el nombre de fichero especificados) cuando se ejecute el comando "**Imprimir**".

Selección del tipo de letra

Se utilizan los comandos "**Fuente del texto**" y "**Fuente del título**" del menú "**Opciones**" para definir los tipos de letras que se utilizan para imprimir textos y para los títulos de cualquier punto del documento. También se puede elegir el tamaño y el estilo de las letras para el texto y los títulos. Se lleva a cabo la selección del tipo de letra con la ventana de diálogo estándar definida por Windows. Se imprimirá cualquier texto (programas literales, nombres en diagramas, etc.) con el tamaño, el estilo y la fuente de caracteres seleccionada. Sólo los títulos se imprimirán con el tipo de letra seleccionado para títulos.

Si no se define el tipo de letra, se utilizará el tipo estándar de la impresora para la impresión de cualquier texto, con los siguientes estilos:

- Estilo "normal" para textos y rótulos dentro de diagramas
- Estilo "negrita" para títulos

A.25 Protección por contraseña

El banco de trabajo de IsaGRAF incluye un sistema completo de protección de datos que permite al usuario proteger proyectos y elementos de librería con contraseñas. Un elemento de librería puede ser una configuración de E/S, una tarjeta de E/S o un equipo complejo, una función o un bloque de función escrito en un lenguaje IEC, o una función, un bloque de función o una función de conversión en “C”. Existe una base de datos de contraseñas por cada proyecto o elemento de librería, y no se puede compartir entre varios.

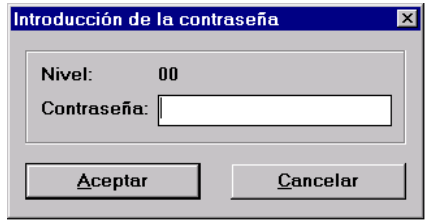
Niveles de protección

Dentro de un proyecto o elemento de librería, el usuario puede definir hasta **16** niveles de acceso que corresponden a diferentes contraseñas. Los niveles de acceso están ordenados en un sistema jerárquico y numerados del **0** al **15**. El nivel de acceso más elevado es el **0**. Cuando un usuario conoce una clave, puede acceder a todos los puntos protegidos en el nivel de acceso correspondiente, además de todos aquellos que estén protegidos en niveles inferiores. Se pueden proteger los comandos y datos elementales de cada proyecto o elemento de librería individualmente con un nivel de acceso. Por ejemplo, se puede proteger el comando "Construir el código de la aplicación" de los menús ISaGRAF de forma individual. Los datos elementales pueden ser un programa, una lista de opciones, las notas técnicas de un elemento de librería, etc...

Definición de la protección por claves de acceso

Se utiliza el comando "**Establecer contraseña**" de los menús de ISaGRAF para definir las claves y niveles de acceso de un proyecto o elemento de librería. Se invoca este comando desde los menús del Gestor de Proyectos de ISaGRAF (en el caso de un proyecto) o del Gestor de Librerías de ISaGRAF (en el caso de un elemento de librería). No se exige una contraseña la primera vez que se ejecuta este comando. Si las claves de acceso ya están definidas, el usuario tiene que introducir la clave de mayor nivel que conozca antes de poder acceder a este comando. Las claves de acceso a puntos protegidos de niveles superiores no podrán ser modificadas. El comando "**Establecer contraseña**" le permite al usuario definir las claves correspondientes a los diferentes niveles de acceso y proteger los comandos y datos elementales con los niveles que están definidos.

Se introducen las claves de acceso (correspondientes a los niveles de protección) haciendo doble click en una línea de la lista. El siguiente cuadro de diálogo se utiliza para introducir una contraseña.



La lista en el área inferior muestra los diferentes objetos (datos o funciones) que pueden ser protegidos, y el nivel de protección actual dado a cada permiso de "acceso lectura" o permiso de "acceso total". Asignar un nivel de protección para leer permisos permite impedir que usuarios sin suficiente permiso incluso abran o impriman un documento.

Acceso a datos protegidos

No se solicita una contraseña o el nombre del usuario cuando se inicializa el banco de trabajo. Sin embargo, cada vez que un usuario intente acceder a datos o funciones protegidos, deberá introducir la contraseña correspondiente en una ventana de diálogo.

Si el usuario introduce la contraseña solicitada (o una contraseña asociada a un nivel de acceso superior), puede continuar con normalidad. Cada vez que el usuario introduce una contraseña, se guarda en memoria para que no tenga que volver a introducirla. Se mantienen las contraseñas almacenadas cada vez que se ejecuta una herramienta ISaGRAF desde otra herramienta ISaGRAF (por ejemplo, el Gestor de Proyectos ejecuta el Gestor de Programas). Las contraseñas almacenadas se pierden al cerrarse la última ventana ISaGRAF. No pueden compartirse las contraseñas que se introduzcan durante la edición de proyectos, o al utilizar el Gestor de Librerías o el Gestor de Archivos. Si el usuario introduce una contraseña incorrecta, no podrá ejecutar la función seleccionada.

Enlaces con el Gestor de Archivos

Al salvarse un objeto (elementos de proyecto o librería) en el disco de archivo, se invoca el elemento de protección de datos denominado "**Copia de seguridad al archivar**". Este elemento corresponde al sistema de protección de datos asociado al objeto en el banco de trabajo (disco duro). No se comprueba el sistema de protección de datos del objeto en el disco de archivo si ya existe. El comando "**Archivar**" del Gestor de Archivos de ISaGRAF guarda la información de protección de datos junto con el objeto en el disco de archivo.

Al restaurar un objeto que ya existe en el banco de trabajo (disco duro), se invoca el elemento de protección de datos denominado "**Sobrescritura con archivo**". Este elemento corresponde al sistema de protección de datos asociado al objeto en el banco de trabajo (disco duro). No se comprueba el sistema de protección de datos del objeto en el disco de archivo. Si se valida este comando, la información de protección de datos restaurada reemplaza a la información existente en el disco duro.

Configurar protecciones individuales para variables y canales E/S

El banco de trabajo de IsaGRAF proporciona una protección completa de datos basada en una jerarquía de contraseñas. La declaración de variables y las conexiones E/S pueden protegerse globalmente por una contraseña. Adicionalmente, ISaGRAF permite establecer una protección individual para cualquier variable o canal E/S. Esto asume que:

- la contraseña ya está definida en el sistema de definición de contraseñas (use el comando "**Proyecto / Establecer contraseña**" de la ventana de manejo de proyectos) de modo que los niveles de protección estén disponibles para protecciones individuales.

- Se usan niveles de protección con prioridad mayor para protecciones individuales comparado con variables globales o protección E/S.

Cuando una variable o un canal E/S tiene una protección individual, aparece un pequeño icono cerca de su nombre, en el diccionario o en la ventana de conexiones E/S.

Use los comandos "**Establecer protección**" y "**Eliminar protección**" del menú "**Edición**" en el diccionario o ventanas de conexión E/S para establecer o eliminar una protección individual para la variable o el canal seleccionado. Ambos comandos piden la introducción de una contraseña valida para que un nivel de protección pueda asociarse a la variable o el canal. Entonces, cada vez que se quiera cambiar una variable o una conexión a un canal con una protección individual se debe introducir una contraseña con un nivel de prioridad suficiente.

Advertencia: si una variable o canal se protege con un nivel, y la contraseña correspondiente se elimina del sistema de protección, y si no se define una contraseña de nivel superior, la variable o el canal no pueden cambiarse más a menos que se defina una nueva contraseña con suficiente nivel.

A.26 Técnicas de programación avanzadas

Este apartado contiene más información sobre el banco de trabajo ISaGRAF y el sistema objeto. Se recomienda al usuario que esté familiarizado con las herramientas y métodos ISaGRAF antes de leer esta sección.

A.26.1 Más sobre las herramientas ISaGRAF

Al utilizar las herramientas de edición de ISaGRAF, el usuario puede pulsar **el botón derecho del ratón** para abrir un menú pop-up, que contiene los comandos principales de edición. El menú se abre en la posición actual del cursor. Esta función es muy útil para reducir el número de operaciones del ratón durante las tareas de cortar y pegar.

Las herramientas ISaGRAF soportan la **ejecución múltiple**. Aunque no se puede abrir la misma herramienta dos veces para editar el mismo documento, sí se pueden abrir ventanas diferentes con la misma herramienta para editar objetos diferentes como operaciones en paralelo.

Se dispone de otros comandos para encontrar información sobre los botones gráficos de las barras de herramientas. Hacer doble click en una zona vacía de una barra de herramientas para ver el contenido de la barra en forma de menú pop-up. Si se deja el cursor del ratón apoyado sobre un botón gráfico, aparece el texto del comando correspondiente.

A.26.2 E/S bloqueadas y E/S virtuales

Si se define una tarjeta de E/S como **virtual**, se deshabilita el procesamiento de los canales físicos de E/S. Cuando se define una tarjeta como virtual, no cambian las operaciones del *kernel* ISaGRAF. La única diferencia está en que no se leen los sensores de entradas y no se actualizan los dispositivos de salida. En este modo de funcionamiento, se puede utilizar el depurador ISaGRAF para modificar los valores de entrada. El atributo **Virtual** es aplicable a una tarjeta completa. Se programa durante la definición de la tarjeta de E/S, **antes** de la generación del código de aplicación. El atributo **virtual** es una característica **estática**, y se almacena al pararse y reiniciarse la aplicación.

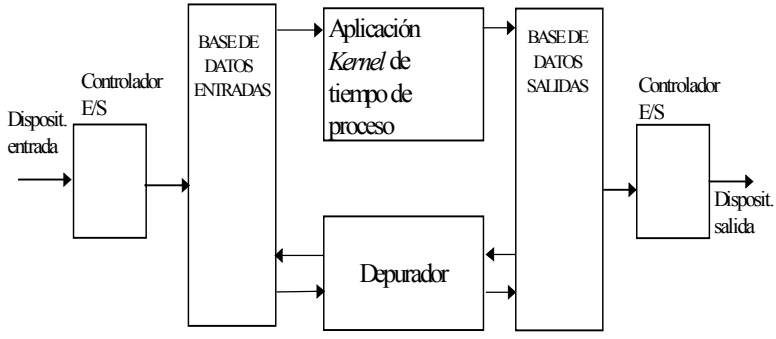
Otra posibilidad es el **bloqueo** de variables de E/S. Consiste en desconectar un dispositivo físico y la correspondiente variable de E/S de ISaGRAF. El bloqueo y desbloqueo de variables se realiza a través del depurador. El bloqueo de variables es una operación **dinámica** y no queda memorizada cuando se reinicia la

aplicación. Una operación de **bloqueo** sólo es aplicable a **una** variable (un canal de E/S) a la vez.

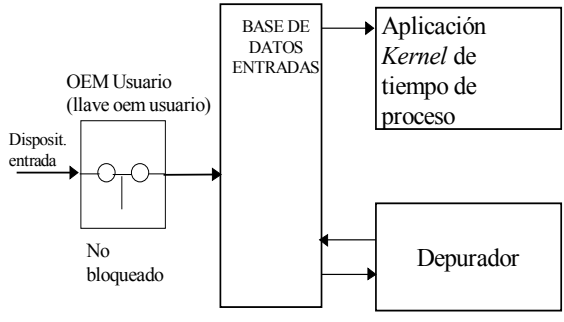
A continuación se resumen las características principales de control de E/S:

	Atributo virtual	Comando bloqueo
Herramienta de selección	Conexión de la tarjeta de E/S	Depurador
Definición	Estático	Dinámico
Modo de selección	Tarjeta	Variable
Aplicación	Validación y pruebas	Mantenimiento

El siguiente diagrama ilustra el flujo de datos de E/S entre las diversas tareas ISaGRAF:

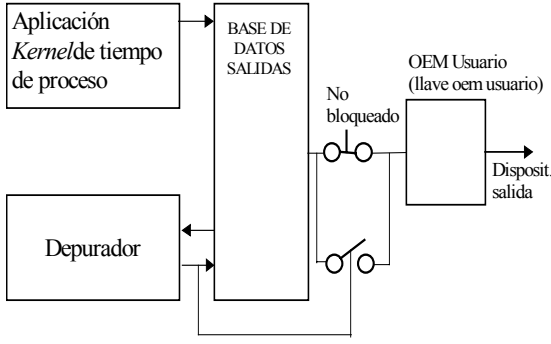


Cuando una variable de entrada está bloqueada, no se cambian los diversos accesos a la base de datos pero se desconecta el dispositivo de entrada. Los valores de entrada pueden establecerse con el depurador y procesarse por el *kernel* de ISaGRAF:

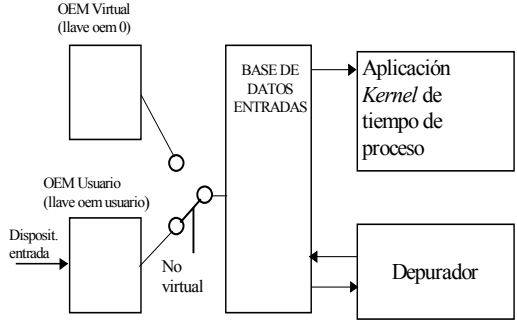


Cuando una variable de salida está bloqueada, se desconectan el *kernel* de tiempo de proceso y el controlador de salidas. En este caso, se sigue permitiendo el

acceso al dispositivo de salida con el depurador ISaGRAF, por medio del controlador de salidas:



Cuando se establece el atributo virtual de una entrada, se desconectan la base de datos de entradas y los dispositivos de entrada asociados. Un controlador virtual de E/S sustituye al auténtico.



La configuración del atributo virtual sigue las mismas directrices para una tarjeta de entradas o una tarjeta de salidas. En el caso de las tarjetas de salidas, el *kernel* de ISaGRAF actualiza la base de datos de salidas. Sin embargo, esta base de datos y los dispositivos de salida asociados están desconectados. Un controlador virtual de E/S sustituye al auténtico.

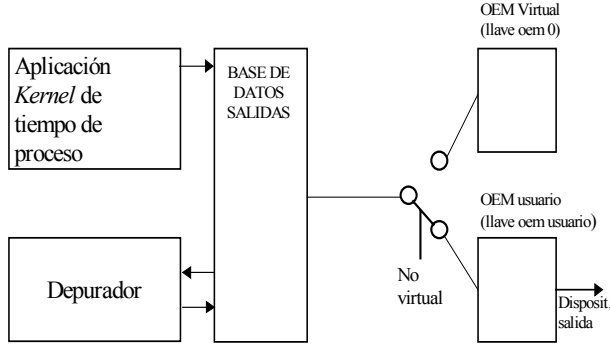
A.26.3 Validación de enlaces PC-PLC

La mayoría de los problemas relacionados con la mala comunicación entre el banco de trabajo ISaGRAF y el PLC objeto se representan en la ventana del depurador por el mensaje de estado "**desconectado**". Antes de implementar una prueba de diagnóstico, se debe validar la comunicación cuando **no haya aplicaciones activas** en el PLC objeto. De esta manera, se puede validar

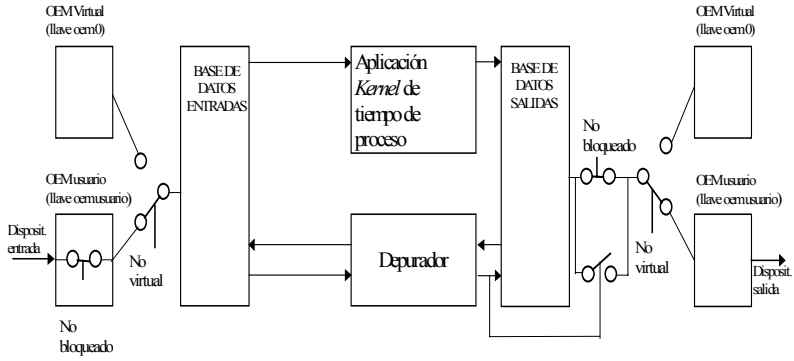
únicamente el enlace de comunicación serie, aislándolo de los efectos relacionados con la ejecución.

El lenguaje “C”, que se utiliza para la descripción de las funciones de conversión y las funciones “C”, permite el acceso directo al sistema objeto. Un error de programación en un componente de *software* de este tipo podría generar errores de sistema o un comportamiento incorrecto del sistema ISaGRAF. Pueden surgir errores de esta naturaleza cuando se desarrollan los controladores de E/S con la opción de **herramienta de desarrollo de E/S** de ISaGRAF. Se podrían producir errores de sistema, por ejemplo, si se conectara una tarjeta de E/S a una dirección de *bus* no válida. La siguiente tabla aporta un resumen sintetizado de los diagnósticos de error:

Estado	Contexto	Diagnóstico
"desconectado" (antes de la carga remota)		- el objeto no está operativo - no hay cable / cable no válido - parámetros de enlace no válidos - objeto ISaGRAF mal instalado
"desconectado" (después de la carga remota)	Inicio de modo ciclo a ciclo	- configuración de E/S no válida - caída del sistema
	Inicio de modo tiempo real	- configuración de E/S no válida - caída del sistema (debida a la programación en “C”)
"sin aplicación"		- no se ha cargado la aplicación - no se ha iniciado la aplicación (debido a la programación en “C”) - diferencia Intel/Motorola - versión de objeto no válida

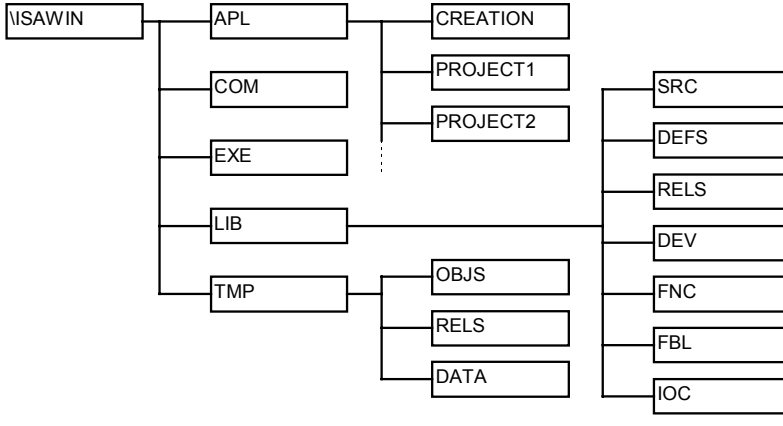


Para resumir todas las posibilidades:



A.26.4 Directorios ISaGRAF

El banco de trabajo ISaGRAF funciona sobre una estructura dedicada de directorios de disco. El usuario especifica el directorio raíz de esta arquitectura durante la instalación de ISaGRAF. EL nombre por defecto del directorio raíz de ISaGRAF es **ISAWIN**. El programa de instalación crea la siguiente arquitectura estándar de disco:



Los subdirectorios estándares de ISaGRAF son:

DIRECTORIO	CONTENIDO
APL	Directorio raíz para los proyectos ISaGRAF. Cada proyecto corresponde a un subdirectorio que contiene todos los datos del proyecto.

	Pueden existir otros directorios para otros grupos de proyecto. El programa de instalación de ISaGRAF crea un subdirectorio "**SMP**" donde se almacenan ejemplos de aplicación.
COM	Datos de rango "común". Los datos pueden ser utilizados por cualquier proyecto.
EXE	Programas y ficheros de ayuda ISaGRAF.
LIB	Librerías ISaGRAF: - listas de elementos - parámetros o interfase para cada elemento - notas técnicas
LIB\IOC	Código fuente para configuraciones de E/S.
LIB\FNC	Código fuente de funciones escritas en lenguajes IEC.
LIB\FBL	Código fuente de bloques de función escritos en lenguajes IEC
LIB\SRC	Código fuente para conversiones y funciones "C"
LIB\DEFS	Cabecera de código para conversiones y funciones "C"
LIB\RELS	Código objeto para conversiones y funciones "C"
LIB\DEV	Ficheros de comandos para desarrollar librerías "C" *makefile*, enlazar listas, etc. ...
TMP	Ficheros temporales: los subdirectorios de TMP están reservados para el Generador de Código de ISaGRAF y no se pueden borrar

Los subdirectorios pueden moverse a otras ubicaciones dentro del disco. Cuando el usuario tiene una arquitectura no estándar, deben declararse las rutas de acceso de los subdirectorios en la sección **WS001**, del fichero de inicialización **ISA.ini**, en el subdirectorio **EXE** de ISaGRAF. Las entradas de la sección **WS001** son:

Isa	Directorio raíz para arquitectura ISaGRAF
IsaExe	Directorio raíz para programas y ficheros de ayuda ISaGRAF
IsaApl	Directorio raíz para proyectos IsaGRAF
IsaTmp	Directorio para ficheros temporales
IsaSrc	Directorio para código fuente de librerías
IsaDefs	directorio para cabeceras fuente de librerías

Observe que: Si se desplaza la entrada IsaTmp a otro directorio, se tienen que crear los subdirectorios OBJS, RELS y DATA en el directorio nuevo. El siguiente ejemplo utiliza las entradas de la sección **WS001** para redefinir la arquitectura estándar de disco ISaGRAF:

```
;file c:\ISAWIN\EXE\ISA.ini

[WS001]
Isa=c:\isawin
IsaExe=c:\isawin\exe
IsaApl=c:\isawin\apl
```

```
IsaTmp=c:\isawin\tmp
IsaSrc=c:\isawin\lib\src
IsaDefs=c:\isawin\lib\defs
```

Para añadir funciones o bloques de función “C” al objeto ISaGRAF, utilizar el directorio **\ISAWIN\LIB\DEV** para almacenar los ficheros de desarrollo: ficheros de comandos, *makefiles*, mapas, etc. Se utiliza el directorio **\ISAWIN\LIB\RELS** para almacenar los ficheros objeto que se generan durante la compilación “C” y las librerías “C” de ISaGRAF que se necesitan para las operaciones de LINK.

A.26.5 Símbolos de aplicación

Cada objeto de una aplicación ISaGRAF está referenciado por un nombre (que se introduce durante la declaración de variables) y una **dirección virtual** interna que calcula el generador de código. La dirección virtual de una variable no es su **dirección de red**, introducida durante la declaración de la variable. Se utilizan las direcciones variables para las tareas de comunicación y para aplicaciones especiales en “C”, utilizando la opción de **herramienta de desarrollo de E/S**. Cuando se ejecuta el generador de código de ISaGRAF, éste crea un fichero ASCII con las correspondencias lógicas entre los nombres y las direcciones virtuales de todos los objetos (variables, programas, pasos, etc.) del proyecto. Se puede interrogar este fichero fácilmente desde cualquier aplicación del usuario para obtener información sobre la base de datos estática de ISaGRAF. El fichero se llama "**APPLI.TST**" y se encuentra en el directorio del proyecto ISaGRAF: "**\ISAWIN\APL\proname**" (‘proname’ es el nombre del proyecto). Esta sección describe en detalle el formato del fichero "**APPLI.TST**". Se utilizan las notaciones principales que aparecen a continuación para las siguientes descripciones:

VA dirección virtual
ATTR atributo de una variable
USP función "C"

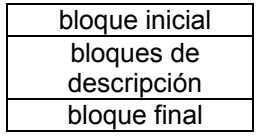
Los atributos de una variable pueden tener los siguientes valores. Estos valores aparecen en los campos de "**atributos**":

+X variable interna
+C variable interna ‘sólo lectura’
+I variable de entrada
+O variable de salida

Todos los números, con la excepción de las direcciones virtuales, están expresados como valores decimales enteros. Las direcciones virtuales (**VA**) están expresadas como números hexadecimales de 4 dígitos y están precedidas por el carácter “**!**”. Por ejemplo:

123 este es un número decimal
!A003 esta es una dirección virtual hexadecimal

Se muestra a continuación la estructura principal del fichero "**APPLI.TST**". El fichero está estructurado como una lista de **bloques**. Un bloque es una lista de **registros**. Cada registro está descrito en una línea de texto. Cada bloque empieza con una cabecera, que se coloca en una línea de texto.



La sintaxis general de un bloque es la siguiente:

@ <nombre_bloque> <argumentos>
#registro...
#registro...
...

La estructura del primer bloque - el que contiene la información principal sobre la aplicación - es la siguiente:

@ISA_SYMBOLS,<crc_apli>
#NAME,<nombre_apli>,<versión>
#DATE,<fecha_creación>
#SIZE,G=<nbprg>,S=<nbpaso>,T=<nbtra>,L=0,P=<nbpro>,V=<nbvar>
#COMMENT,cj international

crc_apli	valor de comprobación de los símbolos de la aplicación
nombre_apli	nombre de la aplicación
versión	número de versión del banco de trabajo ISaGRAF
fecha_creación	fecha de generación de la aplicación
nbprg	número de programas
nbpaso	número de pasos SFC
nbtra	número de transiciones SFC
nbpro	número de funciones “C” utilizadas
nbvar	número total de variables

El último bloque - el que señala el final del fichero - tiene la siguiente estructura:

@END_SYMBOLS

El bloque que se utiliza para describir los programas de la aplicación tiene la siguiente estructura:

@PROGRAMS,<nbprg>
#<va>,<nombre>
#...

nbprg	número de programas definidos en este bloque

va dirección virtual del programa
nombre nombre del programa

El bloque que se utiliza para describir los pasos SFC de la aplicación tiene la siguiente estructura. (Obsérvese que se define un paso virtual para cada programa no SFC):

@STEPS,<nbpasos>
#<va>,<nombre>,<padre>
#...

nbpasos número de pasos definidos en este bloque
va dirección virtual del paso
nombre nombre del paso
padre dirección virtual del padre

El bloque que se utiliza para describir las transiciones SFC de la aplicación tiene la siguiente estructura:

@TRANSITIONS,<nbtrans>
#<va>,<nombre>,<padre>
#...

nbtrans número de transiciones definidas en este bloque
va dirección virtual de la transición
nombre nombre de la transición
padrea dirección virtual del padre

El bloque que se utiliza para describir las variables booleanas de la aplicación tiene la siguiente estructura:

@BOOLEANS,<nb_boo>
#<va>,<nombre>,<atrib>,<programa>,<eq_falso>,<eq_verdadero>
#...

y si el número de variables supera las 4095

X# (1.<numvar>),<nombre>,<atrib>,<programa>,<eq_falso>,<eq_verdadero>

nb_boo número de variables en este bloque
va dirección virtual de la variable
numvar rango de la dirección (dentro de los tipos de datos booleanos)
nombre nombre de la variable
atrib atributo de la variable
programa dirección virtual del programa padre o "**!0000**" para una variable global
eq_falso cadena utilizada para el valor Falso
eq_verdadero cadena utilizada para el valor Verdadero

El bloque que se utiliza para describir las variables analógicas de la aplicación tiene la siguiente estructura:

@ANALOGS,<nb_ana>
#<va>,<nombre>,<atrib>,<programa>,<formato>,<unidad>
#...

y si el número de variables supera las 4095

X# (2.<numvar>),<nombre>,<atrib>,<programa>,<eq_falso>,<eq_verdadero>

nb_ana	número de variables en este bloque
va	dirección virtual de la variable
numvar	rango de la dirección (dentro de los tipos de datos enteros/reales)
nombre	nombre de la variable
atrib	atributo de la variable
programa	dirección virtual del programa padre o "**!0000**" para una variable global
formato	= "**I**" para una variable entera = **"F**" para una variable real
unidad	cadena de unidad

El bloque que se utiliza para describir las variables de temporizador de la aplicación tiene la siguiente estructura:

@TIMERS,<nb_tmr>
#<va>,<nombre>,<atrib>,<programa>
#...

y si el número de variables supera las 4095

X# (3.<numvar>),<nombre>,<atrib>,<programa>,<eq_falso>,<eq_verdadero>

nb_tmr	número de variables en este bloque
va	dirección virtual de la variable
numvar	rango de la dirección (dentro de los tipos de datos temporizadores)
nombre	nombre de la variable
atrib	atributo de la variable (siempre +X: interno)
programa	dirección virtual del programa padre o "**!0000**" para una variable global

El bloque que se utiliza para describir las variables de mensaje de la aplicación tiene la siguiente estructura:

@MESSAGES,<nb_msg>
#<va>,<nombre>,<atrib>,<programa>,<max_lon>
#...

y si el número de variables supera las 4095

X# (4.<numvar>),<nombre>,<atrib>,<programa>,<eq_falso>,<eq_verdadero>

nb_msg	número de variables en este bloque
va	dirección virtual de la variable
numvar	rango de la dirección (dentro de los tipos de datos mensajes)
nombre	nombre de la variable
atrib	atributo de la variable
programa	dirección virtual del programa padre o "**!0000**" para una variable global
max_lon	longitud máxima (capacidad declarada)

El bloque que se utiliza para describir las funciones “C” utilizadas en la aplicación tiene la siguiente estructura:

@USP,<nb_usp>
#<va>,<nombre>
#...

nb_usp	número de funciones “C” en este bloque
va	dirección virtual de la función “C”
nombre	nombre de la función “C”

El bloque que se utiliza para describir las instancias de bloques de función “C” utilizadas en la aplicación tiene la siguiente estructura:

@FBINSTANCES,<nb_fb>
#<va>,<nombre_inst>,<nombre_fb>
#...

nb_fb	número de instancias de bloques de función “C” en este bloque
va	dirección virtual de la instancia de bloque de función “C”
nombre_inst	nombre de la instancia de bloque de función “C”
nombre_fb	nombre del bloque de función “C” de referencia

A.26.6 Límites del banco de trabajo "GRANDE" (WDL) de ISaGRAF

Existen algunas limitaciones para los objetos que se utilizan en el banco de trabajo ISaGRAF. Sin duda existirán muchos otros límites prácticos imputables a la configuración del ordenador que se utilice (disponibilidad de memoria y espacio de disco) y las capacidades del sistema objeto (disponibilidad de memoria y recursos de *hardware* y *software*, etc.). No se pueden superar los siguientes límites absolutos:

❑ Para un proyecto:

Objeto	Máximo	Observaciones
Programas	255	sumando programas principales, subprogramas y programas hijo
Niveles en la jerarquía	20	

El número de proyectos instalados en el banco de trabajo sólo está limitado por el espacio disponible en el disco duro.

❑ Para nombres:

Nombre de:	Máximo	Observaciones
Proyecto	8 car	
Programa	8 car	
Variable	16 car	+ 60 caracteres para comentarios
Etiqueta palabra definida	16 car	
Equivalencia definida	255 car	+ 60 caracteres para comentarios
Tabla de conversión	16 car	
Lista de variables	16 car	
Función / bloque de función (librería)	8 car	aplicable a funciones “C”, bloques de función “C” o funciones escritas en lenguajes IEC
Parámetro de función (librería)	16 car	aplicable a funciones “C”, bloques de función “C” o funciones escritas en lenguajes IEC
Tarjeta de E/S	8 car	
Configuración de E/S	8 car	
Parámetro OEM de tarjeta	16 car	
Función de conversión	8 car	

❑ Edición (para un programa):

Objeto	Máximo	Observaciones
Filas SFC	600	
Columnas SFC	20	
Pasos SFC	4095	Para el proyecto completo, sumando pasos iniciales, pasos inicial y final
Transiciones SFC	4095	Para toda la aplicación
Edición LD/FBD	200 col. 2000 filas	este es el tamaño del área de edición, expresado en unidades de celda.
Edición *Quick LD*	sin límite	los límites están impuestos por la capacidad del PC
Etiquetas IL	251	en el mismo programa IL
Edición de texto	40KBytes	

❑ Para el diccionario (para un proyecto):

Objeto	Máximo	Observaciones
Variables booleanas	65535	
Variables analógicas	65535	sumando variables reales y enteras
Temporizadores	65535	
Variables de mensaje	65535	
Palabras definidas	4095	en la misma lista (mismo rango)

Palabras definidas	255	utilizadas en el mismo programa
Tablas de conversión	127	utilizadas en la aplicación
Puntos de una tabla	32	definidos en la misma tabla de conversión

Los límites indicados para el número máximo de variables booleanas, analógicas o de mensaje abarcan a las variables internas, de entrada y de salida. También incluye todas las variables temporalmente ocultas o variables alocadas por los compiladores. El número de variables editadas conjuntamente (del mismo tipo, misma pantalla), en el editor del diccionario no pueden superar las 16000. Dependiendo de la configuración del PC, el límite puede ser menor que 16000. La aplicación no puede correr en un sistema objeto versión 1.21 o inferior si el número total de variables de un tipo supera las 4095. El enlace standard "Modbus" usando direcciones de red no puede ser utilizado si el número de variables de un tipo supera las 4095.

❑ Conexiones de E/S:

Objeto	Máximo	Observaciones
Tarjetas de E/S	256	definidas para la misma aplicación (tarjetas o equipos complejos)

El número de tarjetas de E/S sumando tarjetas individuales y equipos complejos no puede superar las 256.

Canales de E/S	128	en la misma tarjeta

❑ Para librerías:

Objeto	Máximo	Observaciones
Funciones (lenguaje IEC)	255	instaladas juntas en la librería
Bloques de función (lenguaje IEC)	255	instaladas juntas en la librería
Funciones “C”	255	instaladas juntas en la librería
Bloques de función “C”	255	instaladas juntas en la librería
Instancias de bloques de función	4095	para el mismo tipo de bloque de función en la misma aplicación
Parámetros de entrada de funciones	31	aplicable a funciones C y funciones escritas en lenguajes IEC
Parámetros de bloques de función	32	libremente distribuidos entre parámetros de entrada y salida. Se requiere al menos 1 parámetro de salida.
Función de conversión	128	instaladas juntas en la librería
Configuraciones de E/S	255	Instaladas juntas en la librería
Tarjetas de E/S	255	Instaladas juntas en la librería

Equipos complejos de E/S	255	Instaladas juntas en la librería
Parámetros OEM de tarjetas	16	

B. Referencia de lenguajes

B.1 Arquitectura del proyecto

El proyecto ISaGRAF está dividido en varias unidades de programación denominadas **programas**. Los programas del proyecto están vinculados entre sí por medio de una estructura de árbol. Para describir los programas, puede utilizarse cualquiera de los lenguajes gráficos o literales **SFC, FC (Diagrama de flujo),FBD, LD, ST** o **IL**.

B.1.1 Programas

Un **programa** es una unidad de programación lógica, que describe las operaciones que tienen lugar entre las **variables** del proceso. Los programas describen operaciones bien **secuenciales** o bien **cíclicas**. Los programas cíclicos se ejecutan en cada ciclo del sistema objeto. La ejecución de programas secuenciales sigue las reglas dinámicas del lenguaje **SFC** o del lenguaje **FC**

Los programas están vinculados entre sí en un árbol jerárquico. Los programas situados en la parte superior de la jerarquía los activa el sistema. Los subprogramas (nivel inferior de la jerarquía) son activados por su padre. Se puede describir un programa mediante cualquiera de los siguientes lenguajes gráficos o literales:

Diagrama de Funciones Secuenciales (SFC) para la programación de alto nivel
Diagrama de flujo (FC) para la programación de alto nivel
Diagrama de Bloques de función (FBD) para operaciones cíclicas complejas
Diagrama de Escalera (Contactos) (LD), sólo para operaciones booleanas
Texto Estructurado (ST) para cualquier operación cíclica
Lista de Instrucciones (IL) para operaciones de bajo nivel

El mismo programa no puede mezclar varios lenguajes. Como excepción, LD y FBD pueden combinarse en un único diagrama.

B.1.2 Operaciones cíclicas y secuenciales

La jerarquía de programas se divide en cuatro **secciones** o grupos principales:

Comienzo	programas ejecutados al inicio de cada ciclo objeto
Secuencial	programas que siguen las reglas dinámicas del lenguaje SFC o FC
Fin	programas ejecutados al final de cada ciclo objeto
Funciones	conjunto de subprogramas no dedicados

Los programas pertenecientes a las secciones **'Comienzo'** o **'Fin'** describen las operaciones cíclicas y no son dependientes del tiempo. Los programas pertenecientes a la sección

'Secuencial' describen las operaciones secuenciales, donde la variable de tiempo sincroniza las operaciones básicas de forma explícita. Los programas principales de la sección **'Comienzo'** se ejecutan sistemáticamente al inicio de cada ciclo de proceso. Los programas principales de la sección **'Fin'** se ejecutan sistemáticamente al término de cada ciclo de proceso. Los programas principales de la sección **'Secuencial'** se ejecutan de acuerdo con las reglas dinámicas del lenguaje **SFC** o del **FC**.

Los programas de la sección **'Funciones'** son subprogramas que pueden ser invocados por cualquier otro programa perteneciente al proyecto. Un programa de la sección **'Funciones'** puede invocar a otro programa perteneciente a la misma sección.
Los programas pertenecientes a la sección **'Bloques de función'** son programas que pueden ser invocados por cualquier otro programa del proyecto. Un programa de la sección **'Bloques de función'** puede invocar a programas pertenecientes a la sección Funciones, pero no a otros bloques de función.
Los programas principales e hijos de la sección secuencial deben ser descritos con el lenguaje **SFC** o el **FC** Los programas pertenecientes a las secciones cíclicas (**Comienzo** y **Fin**) no pueden ser descritos con el lenguaje **SFC** o el **FC**. Cualquier programa de cualquiera de las secciones puede poseer uno o más subprogramas. Cualquier programa de la sección secuencial puede poseer uno o más programas **SFC** o **FC** hijo. Los subprogramas no pueden ser descritos con el lenguaje **SFC** o el **FC**.

Una aplicación típica de los programas pertenecientes a la sección de **Comienzo** es la de describir operaciones preliminares sobre dispositivos de entrada para elaborar variables filtradas de alto nivel. Los programas de la sección **Secuencial** utilizan este tipo de variables con frecuencia. Los programas pertenecientes a la sección **Final** suelen utilizarse para describir operaciones de seguridad que afectan a las variables sobre las que actúa la sección **Secuencial**, antes de lanzar valores a los dispositivos de salida.

B.1.3 Programas SFC hijo

Cualquier programa **SFC** de la sección secuencial tiene la capacidad de controlar a otros programas **SFC**. Dichos programas de bajo nivel se denominan **programas SFC hijo**. Un **programa SFC hijo** es un programa en paralelo que puede ser iniciado, finalizado, congelado o reiniciado por su programa padre. Tanto el programa padre como los programas hijo deben estar descritos en el lenguaje **SFC**. Un programa SFC hijo puede poseer variables locales y palabras definidas.

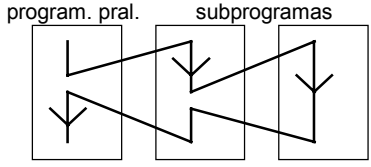
Cuando un programa padre arranca un programa **SFC** hijo, coloca una **marca** SFC en el paso inicial del programa hijo. Se describe este comando con la sentencia **GSTART**. Cuando un programa padre finaliza un programa **SFC** hijo, elimina todas las marcas que pudieran existir en los **pasos** del hijo. Este tipo de comando se describe con la sentencia **GKILL**.

Cuando un programa padre congela un programa **SFC** hijo, elimina todas las marcas que pudieran existir en el programa hijo y mantiene sus posiciones en memoria. Este tipo de comando se describe en la sentencia **GFREEZE**. Cuando un programa padre reinicia un programa **SFC** hijo que ha sido previamente congelado, restaura todas las marcas que eliminara en el momento de congelar el programa hijo. Este tipo de comando se describe con la sentencia **GRST**.

Cualquier programa **FC** de la sección secuencial tiene la capacidad de controlar a otros programas **FC**. Un programa **FC** padre se bloquea durante la ejecución de un subprograma FC. No es posible que se hagan operaciones simultaneas en el programa FC padre y en uno de sus subprogramas FC.

B.1.4 Funciones y subprogramas

Los subprogramas y la ejecución de funciones están dirigidos por el correspondiente programa padre. La ejecución del programa padre se suspende hasta que finalice el subprograma o la función en cuestión:



Cualquier programa perteneciente a cualquier sección puede poseer uno o más subprogramas. Un subprograma pertenece a un único programa padre. Un subprograma puede tener variables y definiciones locales. Puede utilizarse cualquier lenguaje, con la excepción de **SFC** o **FC** para describir un subprograma. Los programas pertenecientes a la sección **'Funciones'** son subprogramas que pueden ser invocados por cualquier otro programa del proyecto. A diferencia de otros subprogramas, no están dedicados a un único programa padre. Los programas de la sección **'Funciones'** pueden invocar a otros programas pertenecientes a la sección. Una función puede ser colocada en la Librería.

Advertencia: El sistema ISaGRAF no soporta la **recursividad** durante la invocación de funciones. Se producirá un error de tiempo de proceso si uno de los programas de la sección **'Funciones'** se invoca a sí mismo o si es invocado por uno de sus subprogramas.
Advertencia: Una función o subprograma no "almacena" el valor local de sus variables locales. Una función o subprograma no se instancia y por tanto no puede llamar a bloques de función.

La interfaz de un subprograma tiene que definirse de forma explícita, indicándose un **tipo** y un **nombre único** para cada uno de sus parámetros de llamada o retorno. Con vistas a sostener las convenciones del lenguaje **ST**, el parámetro de retorno debe tener el mismo nombre que el subprograma.

La siguiente tabla muestra la forma de especificar el valor del parámetro de retorno en el texto de un subprograma, en los diferentes lenguajes:

ST: asignar el parámetro de retorno utilizando su nombre (el mismo nombre que el subprograma):

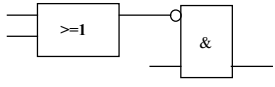
```
nombre_subprograma := <expresión>;
```

IL: el valor del resultado actual (registro IL)

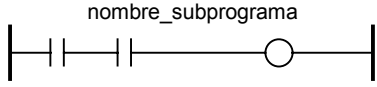
al final de la secuencia se almacena en el parámetro de retorno:

```
LD    10
ADD   20   (* valor de parámetro de retorno = 30 *)
```

FBD: fijar el parámetro de retorno utilizando su nombre:

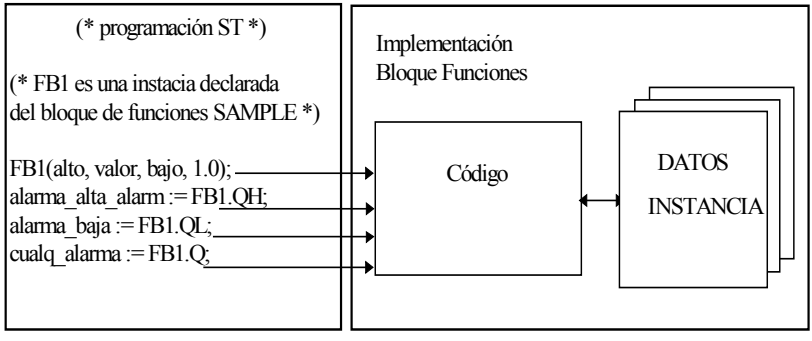


LD: utilizar un símbolo de bucle con el nombre del parámetro de retorno:



B.1.5 Bloques de función

Los bloques de función pueden utilizar los lenguajes LD, FBD, ST o IL. Los bloques de función son del tipo instanciado, lo que significa que se copian las variables locales de un bloque de función para cada instancia. Cuando se invoca un bloque desde un programa, en realidad se invoca la instancia del bloque: aunque se invoca el mismo código, los datos que se utilizan son aquellos que hayan sido asignados a la instancia. Los valores de las variables de la instancia se almacenan de un ciclo a otro.



Advertencias:

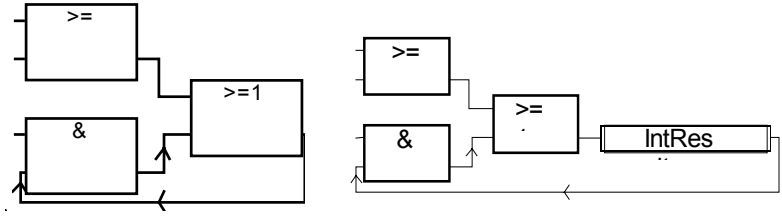
- Los bloques de función escritos con uno de los lenguajes IEC no pueden invocar a otros bloques de función: el mecanismo de instanciación solo gestiona las variables locales del propio bloque. A continuación se indica una lista de los bloques de función estándares que no pueden utilizarse dentro de un bloque de función IEC:
  SR, RS, R_Trig, F_Trig, SEMA, CTU, CTD, CTUD, TON, TOF, TP, CMP, StackInt, AVERAGE, HYSTER, LIM_ALRM, INTEGRAL, DERIVATE, BLINK, SIG_GEN

- Por la misma razón, no se pueden utilizar contactos o bobinas (salidas) Positivos o Negativos, o bobinas (salidas) de Establecer (Set) y Restablecer (Reset).

- Las funciones de TSTART y TSTOP para iniciar y parar temporizadores no pueden utilizarse en bloques de función para objetos o *targets* 3.0x. Sölo pueden utilizarse a partir del *target* 3.20.

- Cuando se tenga la necesidad de incorporar un bucle en el bloque de función, se deberá usar una variable local antes de hacer el bucle. Véase el siguiente ejemplo:

Esto no funcionará: Esto sí funcionará:



B.1.6 Lenguaje de descripción

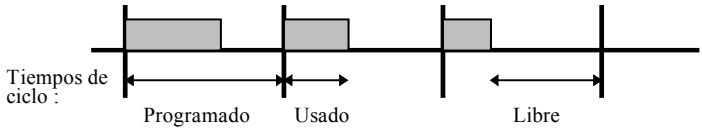
Se puede describir un programa con cualquiera de los siguientes lenguajes gráficos o literales:

Tabla de Funciones Secuenciales (SFC) para operaciones de alto nivel
Diagrama de flujo (FC) para operaciones de alto nivel
Diagrama de Bloques de función (FBD) para operaciones cíclicas complejas
Diagrama de Escalera (Contactos) (LD), para operaciones booleanas solamente
Texto Estructurado (ST) para cualquier operación cíclica
Lista de Instrucciones (IL) para operaciones de bajo nivel

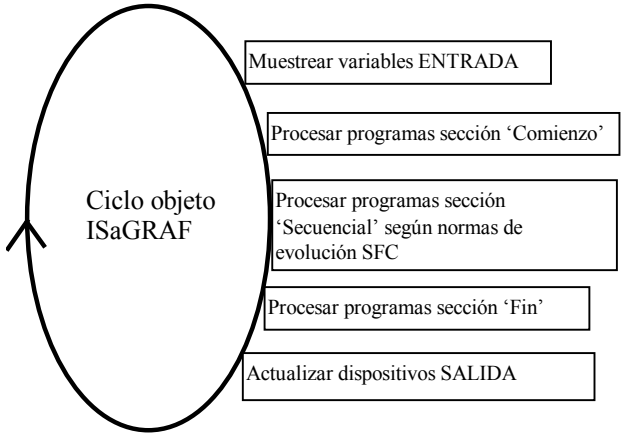
No se pueden mezclar varios lenguajes en un mismo programa. Se elige el lenguaje utilizado para describir un programa a la hora de crear el programa, y no se puede cambiar posteriormente. Como excepción, es posible combinar FBD y LD en un mismo programa.

B.1.7 Reglas de ejecución

ISaGRAF es un sistema **síncrono**. Se activan todas las operaciones por medio de un reloj. La duración básica del reloj se denomina el tiempo de ciclo:



Las operaciones básicas que se llevan a cabo durante un ciclo objeto son:



Este sistema permite:

- garantizar que una variable de entrada mantendrá el mismo valor durante un ciclo,
- garantizar que un dispositivo de salida no se actualizará más de una vez durante un ciclo,
- trabajar de manera segura con la misma variable global, desde diferentes programas,
- cuantificar y controlar los tiempos de respuesta de la aplicación en su conjunto.

B.2 Objetos comunes

Estos son las principales características y los **objetos** comunes de la base de datos de programación ISaGRAF. Estes objetos pueden ser utilizados en cualquier programa que haya sido escrito con alguno de los lenguajes **SFC, FBD, FC, LD, ST** o **IL**.

B.2.1 Tipos básicos

Cualquier constante, expresión o variable que se utilice en un programa (escrito en cualquier lenguaje) debe estar caracterizada por un tipo. La coherencia de tipos deberá mantenerse en las operaciones gráficas y sentencias literales. Estos son los tipos básicos disponibles para objetos de programación:

- **BOOLEANO**: valor lógico (verdadero -true- o falso -false-)
- **ANALÓGICO**: valor contínuo entero o real (flotante)
- **TEMPORIZADOR**: valor en tiempo
- **MENSAJE**: cadena de caracteres

Nota: Los temporizadores contienen valores inferiores a un día y no pueden utilizarse para almacenar fechas.

B.2.2 Expresiones constantes

Una expresión constante está asociada a un único tipo. No se puede utilizar la misma notación para representar expresiones constantes de diferentes tipos.

B.2.2.1 Expresiones constantes booleanas

Sólo existen dos expresiones constantes booleanas:

- **VERDADERO** equivale al valor entero 1
- **FALSO** equivale al valor entero 0

Las palabras clave “Verdadero” y “Falso” son insensibles al uso de mayúsculas o minúsculas.

B.2.2.2 Expresiones constantes analógicas enteras

Las expresiones constantes analógicas representan valores enteros largos (32 bit) y con signo:
entre **-2147483647** y **+2147483647**
Se pueden expresar las constantes analógicas enteras con una de las siguientes **bases**. Las constantes enteras deben empezar por un **prefijo** que identifique las bases utilizadas:

Base	Prefijo	Ejemplo
DECIMAL	**(ninguno)**	**-908**
HEXADECIMAL	**"16#"**	**16#1A2B3C4D**
OCTAL	**"8#"**	**8#1756402**
BINARIO	**"2#"**	**2#1101_0001_0101_1101**

Se puede emplear el carácter de subrayado ('**_**') para separar grupos de dígitos. No posee ningún significado particular, y se utiliza para aumentar la legibilidad de las expresiones constantes.

B.2.2.3 Expresiones constantes analógicas reales

Las expresiones constantes analógicas reales se pueden escribir con representación **decimal** o **científica**. El **punto decimal** ('**.**') separa las partes enteras de las decimales. Se debe utilizar el punto decimal para diferenciar entre las expresiones constantes reales y las enteras. La representación científica emplea las letras '**E**' o '**F**' para separar la **mantisa** del exponente. El exponente de una expresión científica real tiene que ser un valor entero con signo de entre **–37** y **+37**. A continuación se dan ejemplos de expresiones constantes analógicas reales:

3.14159	**-1.0E+12**
+1.0	**1.0F-15**
-789.56	**+1.0E-37**

La expresión "**123**" no representa una expresión constante real. Su representación real correcta es "**123.0**".

B.2.2.4 Expresiones constantes temporizadores

Las expresiones constantes temporizadores representan valores de tiempo entre **0 segundos** y **23h59m59s999ms**. El valor más pequeño que se permite es un milisegundo. Las unidades de tiempo estándares que se utilizan en las expresiones constantes son:

- **Hora** Debe indicarse la letra "**h**" después del número de horas
- **Minuto** Debe indicarse la letra "**m**" después del número de minutos
- **Segundo** Debe indicarse la letra "**s**" después del número de segundos
- **Milisegundo** Deben indicarse las letras "**ms**" después del número de milisegundos

La expresión constante temporizador debe empezar por el prefijo "**T#**" o "**TIME#**". Los prefijos y las letras de las unidades pueden indicarse en mayúsculas o minúsculas, indistintamente. Es posible que algunas unidades no aparezcan. Estos son ejemplos de expresiones constantes temporizadores:

T#1H450MS 1 hora, 450 milisegundos
time#1H3M 1 hora, 3 minutos

La expresión "0" no representa un valor de tiempo, sino una constante analógica.

B.2.2.5 Expresiones constantes con cadenas de mensajes

Las expresiones constantes con cadenas o mensajes representan cadenas de caracteres. Los caracteres deben estar flanqueados por apóstrofes. Por ejemplo:

'ESTO ES UN MENSAJE'

Advertencia: No se puede utilizar el carácter de apóstrofe '" en las expresiones constantes con cadenas. Una expresión constante con cadenas debe ser expresada en una única línea del código fuente del programa. Su longitud no puede superar los 255 caracteres, incluyendo espacios.

Las expresiones constantes de cadenas vacías se representan con dos apóstrofes, sin caracteres de espacio o tabulación entre ellos:

'' (* esta es una cadena vacía *)

En una expresión constante con cadenas, se puede utilizar el carácter especial de dólar ('**$**'), seguido por otros caracteres especiales, para representar un carácter no imprimible:

Secuencia	Significado	ASCII (hexa)	Ejemplo
$$	Carácter '$'	16#24	'pagué $$5 por esto'
$'	Apóstrofe	16#27	'introd. $'Y$' para SÍ'
$L	Cambio línea	16#0a	'siguiente línea $L'
$R	Retorno carro	16#0d	' llo $R He'
$N	Nueva linea	16#0d0a	'esta es una línea$N'
$P	Nueva página	16#0c	'última línea $P primera linea
$T	Tabulación	16#09	'nombre$Ttamaño$Tfecha'
$hh (*)	Cualquier carácter	16#hh	'ABCD = $41$42$43$44'

(*) "**hh**" es el valor hexadecimal del código ASCII correspondiente al carácter expresado.

B.2.3 Variables

Las variables pueden ser **LOCALES** a un programa, o **GLOBALES**. Las variables locales sólo pueden ser utilizadas por un programa. Las variables globales pueden ser utilizadas en cualquier programa del proyecto. Los nombres de las variables deben cumplir con las siguientes normas:

- el nombre no puede superar los **16** caracteres
- el primer carácter debe ser una **letra**
- los restantes caracteres pueden ser **letras**, **dígitos** o el carácter de subrayado (_)

B.2.3.1 Palabras clave reservadas

A continuación se muestra una relación de palabras clave reservadas. Tales identificadores no pueden ser empleados para nombrar programas, variables o funciones o bloques de función “C”:

A	ANA, ABS, ACOS, ADD, ANA, AND, AND_MASK, ANDN, ARRAY, ASIN, AT, ATAN,
B	BCD_TO_BOOL, BCD_TO_INT, BCD_TO_REAL, BCD_TO_STRING, BCD_TO_TIME, BOO, BOOL, BOOL_TO_BCD, BOOL_TO_INT, BOOL_TO_REAL, BOOL_TO_STRING, BOOL_TO_TIME, BY, BYTE,
C	CAL, CALC, CALCN, CALN, CALNC, CASE, CONCAT, CONSTANT, COS,
D	DATE, DATE_AND_TIME, DELETE, DINT, DIV, DO, DT, DWORD,
E	ELSE, ELSIF, EN, END_CASE, END_FOR, END_FUNCTION, END_IF, END_PROGRAM, END_REPEAT, END_RESSOURCE, END_STRUCT, END_TYPE, END_VAR, END_WHILE, ENO, EQ, EXIT, EXP, EXPT,
F	FALSO, FEDGE, FIND, FOR, FUNCTION,
G	GE, GFREEZE, GKILL, GRST, GSTART, GSTATUS, GT,
I	IF, INSERT, INT, INT_TO_BCD, INT_TO_BOOL, INT_TO_REAL, INT_TO_STRING, INT_TO_TIME,
J	JMP, JMPC, JMPCN, JMPN, JMPNC,
L	LD, LDN, LE, LEFT, LEN, LIMIT, LINT, LN, LOG, LREAL, LT, LWORD,
M	MAX, MID, MIN, MOD, MOVE, MSG, MUL, MUX,
N	NE, NOT,
O	OF, ON, OPERATE, OR, OR_MASK, ORN,
P	PROGRAM
R	R, REDGE, READ_ONLY, READ_WRITE, REAL, REAL_TO_BCD, REAL_TO_BOOL, REAL_TO_INT, REAL_TO_STRING, REAL_TO_TIME, REDGE, REPEAT, REPLACE, RESSOURCE, RET, RETAIN, RETC, RETCN, RETN, RETNC, RETURN, RIGHT, ROL, ROR,
S	S, SEL, SHL, SHR, SIN, SINT, SQRT, ST, STN, STRING, STRING_TO_BCD, STRING_TO_BOOL, STRING_TO_INT, STRING_TO_REAL, STRING_TO_TIME, STRUCT, SUB, SYS_ERR_READ, SYS_ERR_TEST, SYS_INITALL, SYS_INITANA, SYS_INITBOO, SYS_INITTMR, SYS_RESTALL, SYS_RESTANA, SYS_RESTBOO, SYS_RESTTMR, SYS_SAVALL, SYS_SAVANA, SYS_SAVBOO,

	SYS_SAVTMR, SYS_TALLOWED, SYS_TCURRENT, SYS_TMAXIMUM, SYS_TOVERFLOW, SYS_TRESET, SYS_TWRITE, SYSTEM,
T	TAN, TASK, THEN, TIME, TIME_OF_DAY, TIME_TO_BCD, TIME_TO_BOOL, TIME_TO_INT, TIME_TO_REAL, TIME_TO_STRING, TMR, TO, TOD, VERDADERO, TSTART, TSTOP, TYPE,
U	UDINT, UINT, ULINT, UNTIL, USINT,
V	VAR, VAR_ACCESS, VAR_EXTERNAL, VAR_GLOBAL, VAR_IN_OUT, VAR_INPUT, ,VAR_OUTPUT,
W	WHILE, WITH, WORD,
X	XOR, XOR_MASK, XORN

Todas las palabras clave que empiecen por un carácter de subrayado ('_') son palabras clave internas y no deben utilizarse en las instrucciones textuales.

B.2.3.2 Variables de representación directa

ISaGRAF permite el uso de **variables de representación directa** en la fuente de los programas para representar un canal libre. Los canales libres son aquellos que no están vinculados a una variable declarada de E/S. El identificador de una variable de representación directa siempre empieza por un carácter **"%"**.
A continuación se muestran las convenciones de nombrado de variables de representación directa para un canal de una tarjeta simple. "***s***" es el número de la ranura de la tarjeta. "***c***" es el número del canal.

%IX*s.c*	canal libre de una tarjeta de entrada booleana
%ID*s.c*	canal libre de una tarjeta de entrada de valores enteros
%IS*s.c*	canal libre de una tarjeta de entrada de mensajes
%QX*s.c*	canal libre de una tarjeta de salida booleana
%QD*s.c*	canal libre de una tarjeta de salida de valores enteros
%QS*s.c*	canal libre de una tarjeta de salida de mensajes

A continuación se indican las convenciones de nombrado de variables de representación directa para un canal de un equipo complejo. "***s***" es el número de la ranura del equipo. "***b***" es el identificador de la tarjeta individual dentro del equipo complejo. "***c***" es el número del canal.

%IX*s.b.c*	canal libre de una tarjeta de entrada booleana
%ID*s.b.c*	canal libre de una tarjeta de entrada de valores enteros
%IS*s.b.c*	canal libre de una tarjeta de entrada de mensajes
%QX*s.b.c*	canal libre de una tarjeta de salida booleana
%QD*s.b.c*	canal libre de una tarjeta de salida de valores enteros
%QS*s.b.c*	canal libre de una tarjeta de salida de mensajes

Algunos ejemplos:

%QX1.6	6º canal de la tarjeta nº 1 (salida booleana)
%ID2.1.7	7º canal de la tarjeta nº 1 del equipo nº 2 (entrada valores enteros)

Una variable de representación directa no puede tener el tipo de datos “**reales**”.

B.2.3.3 Variables booleanas

Booleano significa **lógico**. Tales variables pueden tomar uno de los dos valores booleanos: **VERDADERO** o **FALSO**. La aplicación típica de las variables booleanas se encuentra en las expresiones booleanas. Las variables booleanas pueden tener uno de los siguientes **atributos**:

Interna:	variable de memoria actualizada por el programa
Constante:	variable de memoria, de solo lectura, con un valor inicial
Entrada:	variable conectada a un dispositivo de entrada (refrescada por el sistema)
Salida:	variable conectada a un dispositivo de salida

Advertencia: Cuando se declara una variable booleana, pueden definirse cadenas para reemplazar los valores de "Verdadero" y "Falso" durante el proceso de depuración. Estas cadenas no pueden ser empleadas en los programas si no se han incorporado como **'palabras definidas'** para el lenguaje.

B.2.3.4 Variables analógicas

Analógico significa **continuo**. Este tipo de variable posee valores enteros con signo o reales (flotantes). Los formatos disponibles para variables analógicas son:

Entero	valor entero con signo de 32 bits: de **-2147483647** a **+2147483647**
Real	valor flotante de 32 bits, según el estándar IEEE (precisión simple) 1 bit signo + 23 bits mantisa + 8 bits exponente

El valor REAL del exponente analógico no puede ser inferior a **-37** o superior a **+37**. Las variables analógicas pueden tener uno de los siguientes **atributos**:

Interna:	variable de memoria actualizada por el programa
Constante:	variable de memoria, solo lectura, con un valor inicial
Entrada:	variable conectada a un dispositivo de entrada (refrescada por el sistema)
Salida:	variable conectada a un dispositivo de salida

Nota: Cuando se conecta una variable real a un dispositivo de E/S, el correspondiente controlador de E/S opera con el valor entero equivalente.

Advertencia: No pueden mezclarse las variables o expresiones constantes analógicas enteras y reales en una misma expresión analógica.

B.2.3.5 Variables de temporizador

Temporizador significa **reloj** o **contador**. Las variables de este tipo se emplean típicamente en expresiones de tiempo. Un valor de temporizador no puede superar **23h59m59s99** y no

puede ser negativo. Las variables de temporizador se almacenan en palabras de 32 bits. La representación interna es un número positivo expresado en milisegundos.
Las variables de temporizador pueden tener uno de los siguientes **atributos**:

Interno: variable de memoria gestionada por el programa y refrescada por el sistema ISaGRAF
Constante: variable de memoria, de solo lectura, con un valor inicial

Advertencia: Las variables de temporizador no pueden tener atributos de ENTRADA o SALIDA.

Las variables de temporizador puede ser refrescadas automáticamente por el sistema ISaGRAF. Cuando un temporizador se encuentra **activo**, su valor se incrementa automáticamente de acuerdo con el reloj en tiempo real del sistema objeto. Pueden utilizarse las siguientes instrucciones en lenguaje **ST** para controlar un temporizador:

TSTART inicia el refresco automático de un temporizador
TSTOP termina el refresco automático de un temporizador

B.2.3.6 Variables de cadenas de mensajes

Las variables de mensajes o cadenas contienen cadenas de caracteres. La longitud de la cadena puede variar en el transcurso de una operación de proceso. La longitud de una variable de mensaje no puede superar la capacidad (longitud máxima) especificada a la hora de declarar la variable. La capacidad de mensaje está limitada a 255 caracteres. Las variables de mensaje pueden tener uno de los siguientes **atributos**:

Interna: variable de memoria actualizada por el programa
Constante: variable de memoria, de solo lectura, con un valor inicial
Entrada: variable conectada a un dispositivo de entrada (refrescada por el sistema)
Salida: variable conectada a un dispositivo de salida

Las variables de cadena pueden contener cualquier carácter del juego estándar de caracteres ASCII (códigos ASCII del **0** al **255**). Se permiten caracteres nulos en una cadena de caracteres. Algunas de las funciones “C” de la librería estándar de ISaGRAF no funcionarán correctamente con mensajes que contengan caracteres nulos (**0**).

B.2.4 Comentarios

En los lenguajes literales como **SI** e **IL** se pueden insertar comentarios libremente. Los comentarios deberán empezar por los caracteres especiales "**(***" y terminar por los caracteres "***)**". Se pueden insertar comentarios en cualquier punto de un programa **ST**, y pueden ocupar más de una línea.

Estos son ejemplos de comentarios:

```
counter := ivalue; (* asigna el contador principal *)
(* este es un comentario expresado
en dos líneas *)
c := counter (* se puede colocar comentarios en cualquier lugar *) + base_value +
1;
```

No se pueden utilizar comentarios intercalados. Es decir, no se pueden emplear los caracteres "**(***" dentro de un comentario.

Advertencia: El lenguaje **IL** sólo acepta comentarios como el último componente de una línea de instrucciones.

B.2.5 Palabras definidas

El sistema ISaGRAF permite la redefinición de expresiones constantes, expresiones booleanas verdaderas y falsas, palabras clave y expresiones **ST** complejas. Para ello, se debe asignar un **identificador** a la expresión correspondiente. Por ejemplo:

SÍ	es	**VERDADERO**
PI	es	**3.14159**
OK	es	**(auto_mode AND NOT (alarm))**

Cuando se define una equivalencia de este tipo, se puede utilizar su **identificador** en cualquier punto de un programa **ST** para reemplazar la expresión asociada. A continuación aparece un ejemplo de programación en **ST** utilizando definiciones:

```
If OK Then
  angle := PI / 2.0;
  isdone := SI;
End_if;
```

Las palabras definidas pueden ser **LOCALES** para un programa, **GLOBALES** o **COMUNES**.
Las palabras definidas locales sólo pueden ser utilizadas por un único programa.
Las palabras definidas globales pueden ser utilizadas en cualquier programa del proyecto.
Las palabras definidas comunes pueden ser utilizadas en cualquier programa de cualquier proyecto.
Nota: Las palabras definidas comunes pueden almacenarse de forma independiente en el Gestor de Archivos.

Advertencia: Cuando se define el mismo identificador dos veces, con diferentes equivalencias **ST**, se utiliza la última expresión que haya sido definida. Por ejemplo:

La definición:	**ABIERTO**	es	**FALSO**
	ABIERTO	es	**VERDADERO**
significa:	**ABIERTO**	es	**VERDADERO**

La denominación de palabras definidas debe cumplir con las siguientes normas:

- el nombre no puede superar los **16** caracteres
- el primer carácter debe ser una **letra**
- los restantes caracteres pueden ser **letras**, **dígitos** o el carácter de subrayado ('**_**')

Advertencia: Una palabra definida no puede emplear otra palabra definida en su definición. Por ejemplo, no se puede escribir:

PI es **3,14159**
~~**PI2** es **PI*2**~~

Debe escribirse la equivalencia completa, empleando constantes o variables y operaciones:

PI2 es **6,28318**

B.3 Lenguaje SFC

El Diagrama de Funciones Secuenciales (SFC) es un lenguaje **gráfico** que se utiliza para describir **operaciones secuenciales**. El proceso se representa como un conjunto de **pasos** bien definidos, vinculados por medio de **transiciones**. Se asocia una **condición booleana** a cada transición. Las **acciones** dentro de cada paso se detallan por medio de otros lenguajes (**ST**, **IL**, **LD** y **FDB**).

B.3.1 Principal formato de diagramas SFC

Un programa SFC es un conjunto gráfico de **pasos** y **transiciones**, unidos por **vínculos orientados**. Se emplean múltiples vínculos de conexión para representar divergencias y convergencias. Se puede separar una parte determinada del programa completo, quedando representada en el esquema de conjunto por un único símbolo denominado **macropaso**. Las **reglas gráficas** básicas de SFC son las siguientes:

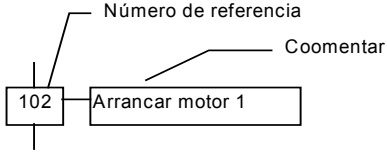
- A un paso no le puede seguir otro paso
- A una transición no le puede seguir otra transición

B.3.2 Componentes básicos SFC

Los componentes básicos (símbolos gráficos) del lenguaje SFC son: pasos y pasos iniciales, transiciones, vínculos orientados, y saltos a un paso.

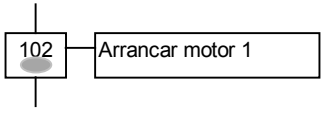
B.3.2.1 Pasos y pasos iniciales

Se representa un paso con un único **cuadrado**. Cada paso está **referenciado** con un número, escrito dentro del símbolo cuadrado del paso. Se incluye la descripción principal del paso en un rectángulo vinculado al símbolo del paso. Esta descripción es un **comentario libre** (es decir, no es parte del lenguaje de programación). La información arriba indicada se denomina el **Nivel 1** del paso:

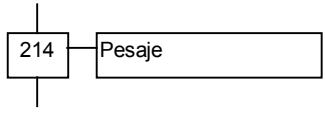


En el tiempo de ejecución, una **marca** indica que el paso se encuentra **activo**:

Paso activo:

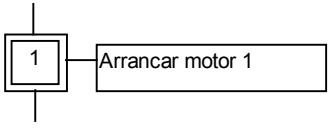


Paso inactivo:



La **situación inicial** de un programa SFC se expresa mediante **pasos iniciales**. Un paso inicial posee un símbolo gráfico con un **marco doble**. Se coloca una marca automáticamente en cada paso inicial cuando se arranca el programa.

Paso inicial:



Un programa SFC debe contener **al menos un** paso inicial.

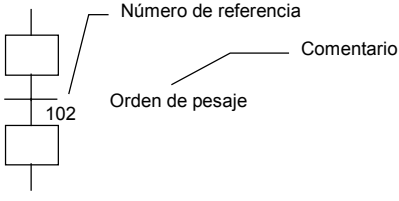
Estos son los atributos de un paso. Se puede utilizar este tipo de campo en cualquiera de los otros lenguajes:

GSnnn.x..... actividad del paso (valor booleano)
GSnnn.t..... duración de la activación del paso (valor en tiempo)

(donde **nnn** es el número de referencia del paso)

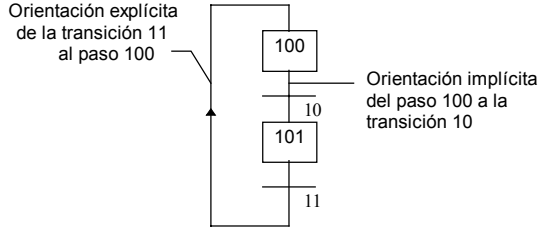
B.3.2.2 Transiciones

Las transiciones se representan por medio de unas barras horizontales pequeñas que atraviesan el vínculo de conexión. Cada transición está **referenciada** con un número escrito al lado del símbolo de transición. Se incluye la descripción principal del paso a la derecha del símbolo de transición. Esta descripción es un **comentario libre** (no es parte del lenguaje de programación). La información arriba indicada se denomina el **Nivel 1** del paso:



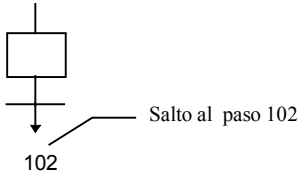
B.3.2.3 Vínculos orientados

Se utilizan líneas simples para vincular pasos y transiciones. Estos son vínculos orientados. Cuando no se indica explícitamente la orientación, el vínculo se orienta de arriba abajo.

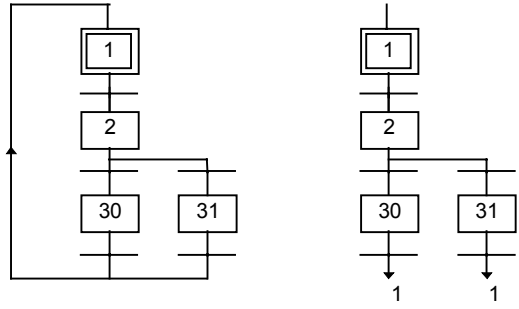


B.3.2.4 Saltar a un paso

Pueden emplearse símbolos de salto para indicar un vínculo de conexión entre una transición y un paso, sin necesidad de dibujar la línea de conexión. El símbolo de salto debe estar referenciado con el número del paso destino:



No se puede utilizar un símbolo de salto para representar un vínculo entre un paso y una transición. Ejemplo de saltos – los siguientes diagramas son equivalentes:

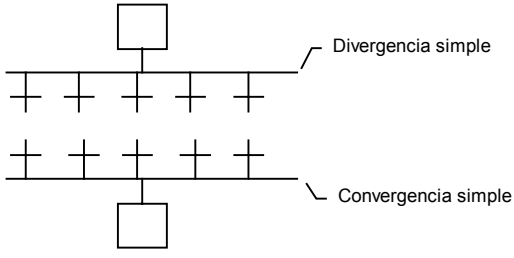


B.3.3 Divergencias y convergencias

Las divergencias son **vínculos de conexión múltiples** desde un símbolo SFC (paso o transición) a muchos otros simbolos SFC. Las convergencias son vínculos de conexión múltiples desde más de un símbolo SFC a un símbolo SFC. Tanto las divergencias como las convergencias pueden ser simples o dobles.

B.3.3.1 Divergencias simples

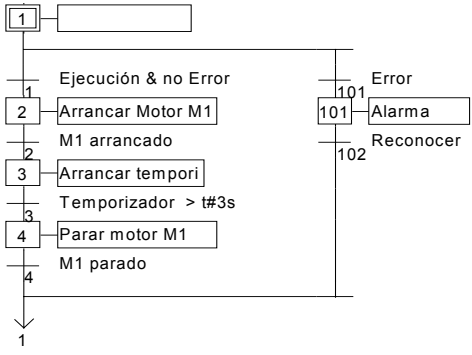
Una divergencia simple es un enlace múltiple entre un paso y muchas transiciones. Permite que la marca activa pase a una de un número de ramas. Una convergencia simple es un enlace múltiple desde muchas transiciones a un mismo paso. Se suelen utilizar las convergencias simples para agrupar a los ramales SFC que fueron iniciados en una divergencia simple. Las divergencias y convergencias simples se representan con líneas horizontales simples.



Advertencia: Las condiciones asociadas a las diferentes transiciones al inicio de una divergencia simple **no son implícitamente exclusivas**. Las condiciones de las transiciones

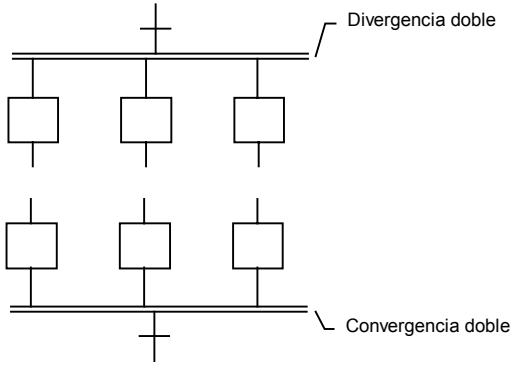
deben incluir un detalle explícito de la exclusividad para asegurar que sólo una marca pueda progresar en un ramal de la divergencia durante el tiempo de ejecución. A continuación se muestra un ejemplo de divergencias y convergencias simples:

(* Programa SFC con divergencia y convergencia simple *)



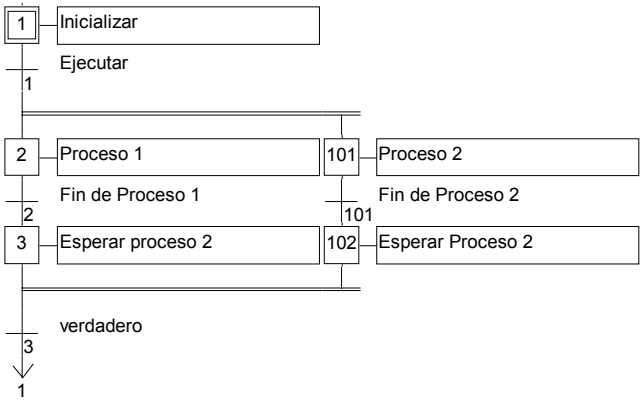
B.3.3.2 Divergencias dobles

Una divergencia doble es un enlace múltiple desde una transición a muchos pasos. Corresponde a operaciones en paralelo del proceso. Una convergencia doble es un enlace múltiple desde numerosos pasos a una única transición. Las convergencias dobles se utilizan generalmente para agrupar a los ramales SFC iniciados en una divergencia doble. Las divergencias y convergencias dobles se representan con líneas horizontales dobles.



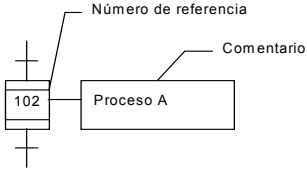
Ejemplo de divergencia y convergencia doble:

(* Programa SFC con divergencia y convergencia doble *)



B.3.4 Macropaso

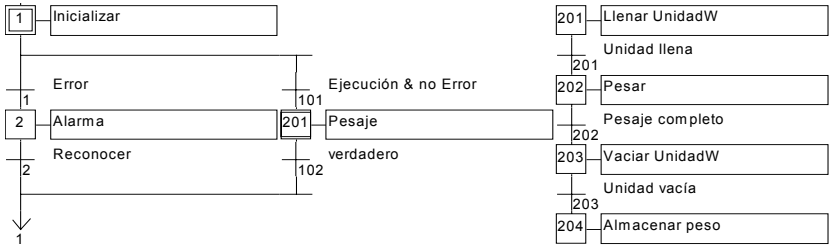
Un macropaso es una representación única de un grupo único de pasos y transiciones. El cuerpo de la macro se describe por separado, en otro lugar del mismo programa SFC. Aparece como un único símbolo en el esquema general SFC. Este es el símbolo que se emplea para representar un macropaso:



El número de referencia que se indica en el símbolo de la macro es el número de referencia del primer paso que figura en el cuerpo de la macro. El macropaso debe comenzar con un **paso inicial** y finalizar con un **paso final**. El diagrama debe ser autocontenido. El paso inicial no debe poseer enlaces ascendentes (ninguna transición hacia atrás) y el paso final no debe poseer enlaces descendentes (ninguna transición hacia delante). Se puede colocar un símbolo de macropaso en el cuerpo de otro macropaso.

Advertencia: Teniendo en cuenta que un macropaso es un conjunto **único** de pasos y transiciones, la misma macro no puede utilizarse más de una vez en un programa SFC.

Ejemplo de macropaso:
(* Programa SFC con macropaso *)
(* Diagrama principal *) **(* Cuerpo de la macropaso *)**



B.3.5 Acciones dentro de los pasos

El **Nivel 2** de un paso SFC es la descripción detallada de las **acciones** que se ejecutan **durante la actividad del paso**. Esta descripción se realiza utilizando las **características literales SFC**, y otros lenguajes como Texto Estructurado (**ST**). Los tipos básicos de acciones son:

- Acciones booleanas
- Acciones pulsantes programadas en ST
- Acciones no almacenadas, programadas en ST
- Acciones SFC

Pueden describirse varias acciones (de tipos iguales o diferentes) en el mismo paso. Las características especiales que permiten el uso de otros lenguajes son:

- Invocación de subprogramas
- Convenciones del lenguaje Lista de Instrucciones (IL)

B.3.5.1 Acciones booleanas

Las acciones booleanas asignan una variable booleana con la actividad del paso. Esta variable puede ser bien de salida o bien interna. Se asigna cada vez que empieza o finaliza la actividad del paso. La sintaxis de las acciones booleanas básicas es:

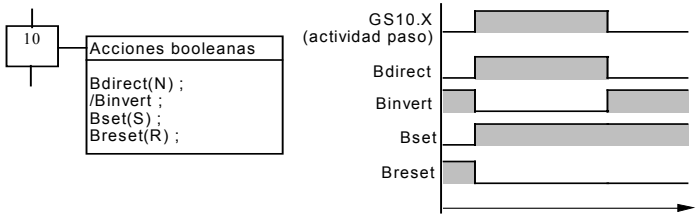
<variable_ booleana> (N) ;	asigna la señal de actividad del paso a la variable
<variable_ booleana> ;	mismo efecto (el atributo N es opcional)
/ <variable_ booleana> ;	asigna a la variable la negación de la señal de actividad del paso

Se dispone de otras funciones para establecer o restablecer una variable booleana, una vez que el paso esté activo. La sintaxis es:

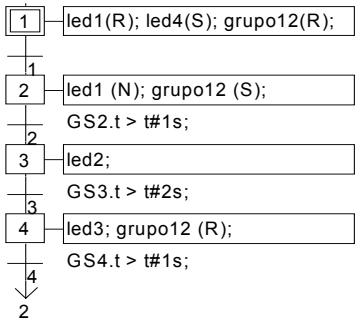
<variable_ booleana> (S) ;	activa la variable a VERDADERO cuando la señal de actividad del paso se convierte en VERDADERO

<variable_ booleana> (R) ; desactiva la variable a FALSO cuando la señal de actividad del paso se convierte en VERDADERO

La variable booleana tiene que ser de SALIDA o INTERNA. La programación SFC que aparece a continuación conduce al siguiente comportamiento:



Ejemplo de acciones booleanas:
(* Programa SFC utilizando acciones BOOLEANAS *)

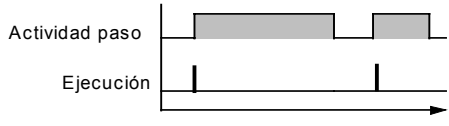


B.3.5.2 Acciones pulsantes

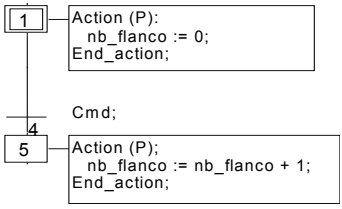
Una acción pulsante es una lista de instrucciones ST o IL, que se ejecutan sólo **una** vez al producirse la **activación** del paso. Las instrucciones se escriben de acuerdo con la siguiente sintaxis SFC:

```
ACCION (P) :
        (* sentencias ST *)
FIN_ACCION ;
```

La siguiente ilustración muestra los resultados de una acción pulsante:



Ejemplo de acción pulsante:

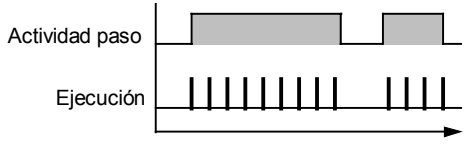


B.3.5.3 Acciones no almacenadas

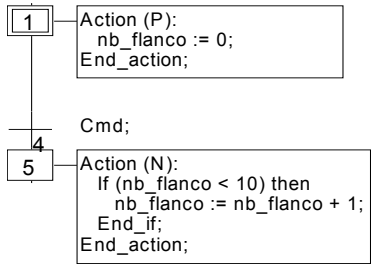
Una acción no almacenada (normal) es una lista de instrucciones ST o IL que se ejecutan **en cada ciclo** durante todo el periodo **activo** del paso. Las instrucciones se escriben de acuerdo con la siguiente sintaxis SFC:

```
ACCION (N) :
        (* instrucciones ST *)
FIN_ACCION ;
```

La siguiente ilustración muestra los resultados de una acción no almacenada:



Ejemplo de acción no almacenada:



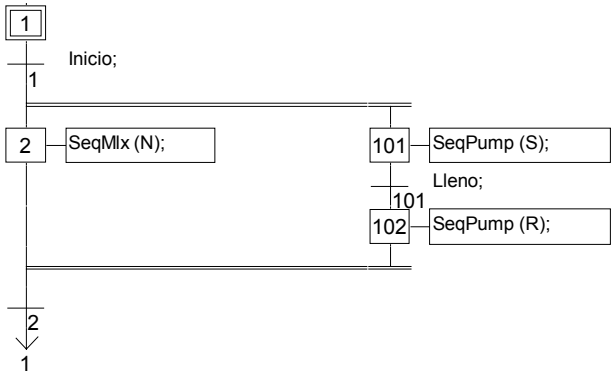
B.3.5.4 Acciones SFC

Una acción SFC es una secuencia SFC hija, iniciada o finalizada de acuerdo con los cambios que se producen en la señal de actividad de paso. Una acción SFC puede tener un calificador **N** (No almacenado), **S** (Establecer) o **R** (Restablecer). La sintaxis de las acciones SFC básicas es:

<prog_hijo> (N);	inicia la secuencia hija cuando el paso se activa y la finaliza cuando el paso se inactiva
<prog_hijo> ;	mismo efecto (el atributo N es opcional)
<prog_hijo> (S);	inicia la secuencia hija cuando el paso se activa; no se hace nada cuando el paso se inactiva
<prog_hijo> (R);	finaliza la secuencia hija cuando el paso se activa; no hace nada cuando el paso se inactiva

La secuencia SFC que se haya especificado como una acción tiene que ser un **programa SFC hijo** del programa que se esté editando en ese momento. Obsérvese que el uso de los calificadores **S** (Establecer) o **R** (Restablecer) para una acción SFC tiene exactamente el mismo efecto que las instrucciones **GSTART** y **GKILL,** que se programan en **ST** para una acción pulsante.
A continuación se muestra un ejemplo de una acción SFC. El principal programa SFC se denomina **padre**. Tiene dos hijos SFC, llamados **SeqMlx** y **SeqPump**. La programación SFC del programa SFC padre es:

(* Programa SFC utilizando acciones SFC *)



B.3.5.5 Invocación de funciones y bloques de función desde una acción

Se pueden invocar subprogramas, funciones o bloques de función (escritos en los lenguajes ST, IL, LD o FBD) o funciones “C” y bloques de función “C” directamente desde un bloque de acciones SFC, con base en la siguiente sintaxis:

Para subprogramas, funciones y funciones “C”:

```
ACTION (P) :
        result := sub_program ( ) ;
END_ACTION;
```

o

```
ACTION (N) :
        result := sub_program ( ) ;
END_ACTION;
```

Para bloques de función en “C” o en ST, IL, LD, FBD:

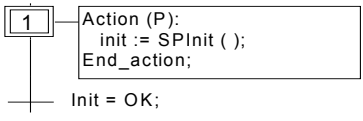
```
ACTION (P) :
        Fbinst(in1, in2);
        result1 := Fbinst.out1;
        result2 := Fbinst.out2;
END_ACTION;
```

o

```
ACTION (N) :
        Fbinst(in1, in2);
        result1 := Fbinst.out1;
        result2 := Fbinst.out2;
END_ACTION;
```

Para una sintaxis detallada, véase la sección dedicada al lenguaje ST.
Ejemplo de invocación de un subprograma en un bloque de acciones:

(* Programa SFC con invocación de subprograma en un bloque de acciones *)



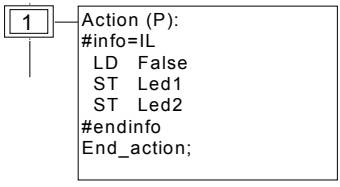
B.3.5.6 Convenciones IL

Se puede incorporar la programación IL (Lista de Instrucciones) directamente en un bloque de acciones SFC, con base en la siguiente sintaxis:

```
ACTION (P) :            (* or N *)
#info=IL
    <sentencia>
    <sentencia>
    ....
#endinfo
END_ACTION;
```

Las palabras claves especiales "#info=IL" y "#endinfo" deben ser introducidas exactamente como se indica, y son **sensibles al caso** (mayúsculas o minúsculas). No se puede insertar caracteres de espacio o tabulación dentro, después o antes de palabras clave. A continuación se muestra un ejemplo de un programa IL dentro de un bloque de acciones:

(* Programa SFC con secuencia IL en un bloque de acciones *)



B.3.6 Condiciones vinculadas a transiciones

En cada transición, se asocia una **expresión booleana** que condiciona el franqueo de la transición. Esta condición suele expresarse con el lenguaje ST, o bien utilizando el lenguaje LD (editor *Quick LD*). Este es el **Nivel 2** de la transición. Sin embargo, también se pueden utilizar otras estructuras:

- Convenciones del lenguaje ST
- Convenciones del lenguaje LD
- Convenciones del lenguaje IL
- Invocación de funciones desde una transición

Advertencia: Cuando no se añade una expresión a la transición, la condición por defecto es **VERDADERO**.

B.3.6.1 Convenciones ST

Se puede utilizar el lenguaje de **Texto Estructurado** (ST) puede utilizarse para describir la **condición** que se asocia a una transición. La expresión completa debe ser del tipo **booleano** y finalizar con **punto y coma**, de acuerdo con la siguiente sintaxis:

<expresión_booleana> ;

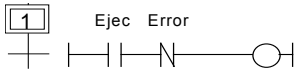
La expresión puede ser una expresión constante VERDADERA o FALSA, una entrada simple o una variable booleana interna, o una combinación de variables que conduce a un valor booleano. A continuación se muestra un ejemplo de programación ST para transiciones:

(* Programa SFC con programación ST para transiciones *)

B.3.6.2 Convenciones LD

El lenguaje de **Diagrama de Escalera (Contactos)** (LD) puede utilizarse para describir la **condición** que se asocia a una transición. El diagrama se compone de un único escalón (línea) con una bobina. El valor de la bobina (salida) representa el valor de la transición.

A continuación se muestra un ejemplo de programación LD para transiciones:



B.3.6.3 Convenciones IL

La programación de **Lista de Instrucciones** (IL) puede utilizarse directamente para describir una transición SFC, de acuerdo con la siguiente sintaxis:

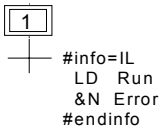
```
#info=IL
        <sentencia>
        <sentencia>
        ....
#endinfo
```

El valor que contiene el **resultado actual** (registro IL) al final de la secuencia IL provoca la asociación del resultado de la condición a la transición:

Resultado actual = 0	→	condición es **FALSA**
Resultado actual <> 0	→	condición es **VERDADERA**

Las palabras claves especiales "#info=IL" y "#endinfo" deben ser introducidas exactamente como se indica, y son sensibles al caso (mayúsculas o minúsculas). No se puede insertar caracteres de espacio o tabulación dentro, después o antes de palabras clave. A continuación se muestra un ejemplo de programación IL para transiciones:

(* Programa SFC con programa IL para transiciones *)



B.3.6.4 Invocación de funciones desde una transición

Puede invocarse cualquier subprograma o función (escritos en los lenguajes FBD, LD, ST o IL), o función "C", para evaluar la condición asociada a una transición, de acuerdo con la siguiente sintaxis:

< sub_program > ( ) ;

El valor de retorno del subprograma o de la función deberá ser booleano y producirá la condición correspondiente:

valor de retorno = FALSO → condición es **FALSA**
valor de retorno = VERDADERO → condición es **VERDADERO**

Ejemplo de un subprograma invocado durante una transición:

(* Programa SFC con invocación de subprograma para transiciones *)

B.3.7 Reglas dinámicas SFC

Las **cinco** reglas dinámicas del lenguaje SFC son:

Situación inicial

La situación inicial está caracterizada por los **pasos iniciales** que están, por definición, en el estado activo al principio de la operación. **Al menos un** paso inicial tiene que estar presente en cada programa SFC.

Franqueo de una transición

Una transición puede estar **habilitada** o **inhabilitada**. Se considera habilitada cuando todos los pasos inmediatamente anteriores y asociados al símbolo de transición correspondiente se encuentran **activos**. En caso contrario, se considera inhabilitada. Una transición no puede ser franqueada a no ser que:
- esté habilitada, y
- la condición de transición asociada sea verdadera.

Cambio de estado de pasos activos

El franqueo de una transición conduce simultáneamente al estado activo de los pasos inmediatamente posteriores y al estado inactivo de los pasos inmediatamente anteriores.

Franqueo simultáneo de transiciones

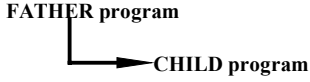
Se pueden utilizar líneas dobles para indicar aquellas transiciones que tienen que ser franqueadas simultáneamente. Si las transiciones de este tipo se muestran por separado, puede emplearse el estado de actividad de los pasos anteriores (GSnnn.x) para expresar su condición.

Activación y desactivación simultánea de un paso

Si durante el tiempo de operación se activa y desactiva un paso simultáneamente, se dará prioridad a la activación.

B.3.8 Jerarquía de programas SFC

El sistema ISaGRAF permite la descripción de la estructura vertical de los programas SFC. Los programas SFC se organizan en un **árbol jerárquico**. Cada programa SFC puede controlar (iniciar, finalizar...) a otros programas SFC. Los programas de este tipo se conocen como **hijos** del programa SFC que los controla. Los programas SFC están unidos entre sí en un **árbol jerárquico** principal, valiéndose de una relación "**padre-hijo**":



Las reglas básicas implícitas en la estructura jerárquica son:

- Los programas SFC que carecen de padre se denominan programas SFC "**principales**"
- El sistema activa los programas SFC principales cuando se inicia la aplicación
- Un programa puede tener varios programas hijo
- El hijo de un programa no puede tener más de un padre
- Un programa hijo sólo puede ser controlado por su padre
- Un programa no puede controlar a los hijos de uno de sus propios hijos

Las acciones básicas que puede llevar a cabo un programa SFC padre para controlar a su programa hijo son:

- Iniciar (**GSTART**) Arranca el programa hijo: activa cada uno de sus pasos iniciales. Los hijos de este programa hijo no se inician automáticamente.
- Terminar (**GKILL**) Termina el programa hijo mediante la desactivación de cada uno de sus pasos activos. También se finalizan todos los hijos del programa hijo.
- Congelar (**GFREEZE**) Desactiva cada uno de los pasos activos del programa y los memoriza para que el programa pueda

reiniciarse. Se congela asimismo a todos los hijos del programa hijo.

- Reiniciar: (**GRST**) Reinicia un programa SFC congelado mediante la reactivación de todos los pasos suspendidos. Los hijos del programa no se reinician automáticamente.
- Obtener estado: (**GSTATUS**) Obtiene el estado actual (activo, desactivo o congelado) de un programa hijo.

B.4 Lenguaje FC

El Diagrama de flujo (**FC**) es un lenguaje gráfico que se utiliza para describir operaciones secuenciales. Un diagrama de flujo FC está compuesto por acciones y decisiones. Entre las acciones y decisiones hay vínculos orientados representando flujo de datos. Los vínculos de conexión múltiple se usan para representar divergencias y convergencias. Se pueden describir Acciones y decisiones con los lenguajes ST, LD o IL. Se pueden llamar a funciones y a bloques de función de cualquier lenguaje (excepto SFC) desde acciones y decisiones. Un programa diagrama de flujo puede llamar a otro programa diagrama de flujo, El programa FC llamado es un subprograma del programa FC que lo llama.

B.4.1 Componentes FC

A continuación están los componentes gráficos del lenguaje de diagrama de flujo

- ***Inicio del diagrama FC***

  Un símbolo de "Inicio" debe aparecer al comienzo del programa de diagrama de flujo. Es único y no se puede omitir, representa el estado inicial del diagrama cuando está activado. A continuación está el dibujo de un símbolo de "Inicio":

  El símbolo de "Inicio" siempre tiene una conexión (por debajo) con los otros objetos del diagrama. Un diagrama de flujo no es valido si no hay ninguna conexión dibujada de "Inicio" a otro objeto.

- ***Fin del diagrama FC***

  Un símbolo de "**Fin**" debe aparecer al final del programa en diagrama de flujo. Es único y no se puede omitir, Es posible que no haya ninguna conexión dibujada al símbolo de fin (diagrama de bucle infinito), pero el símbolo de "Fin" debe aparecer en cualquier caso dibujado en la parte inferior del diagrama. Representa el estado final del diagrama cuando su ejecución se ha completado. A continuación está el símbolo de "Fin":

  El símbolo de "Fin" tiene normalmente una conexión por arriba a otros objetos del diagrama. Un diagrama de flujo puede no tener conexión a un objeto "Fin"

(diagrama de bucle infinito). El símbolo de "Fin" debe ser aún visible en la parte inferior del diagrama en este caso.

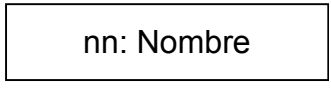
Enlaces de flujo FC

Un enlace de flujo es una línea que representa un flujo entre dos puntos del diagrama. Un enlace siempre termina en una flecha. A continuación está el dibujo de un enlace de flujo:

Dos enlaces no pueden estar conectados al mismo punto fuente de conexión.

Acciones FC

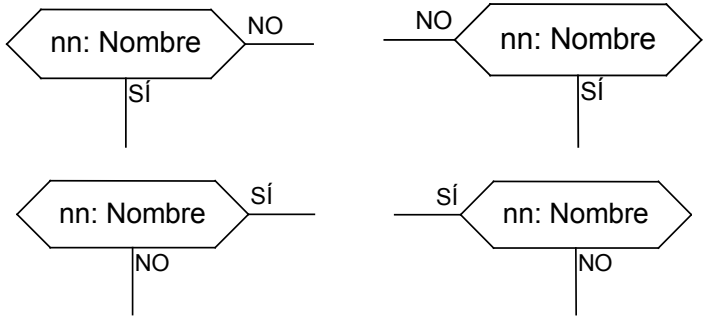
Un símbolo de **acción FC** representa las acciones a realizar. Una acción está identificada por un número y un nombre. Debajo está el dibujo de un símbolo de "acción":



Dos objetos diferentes del mismo diagrama no pueden tener el mismo nombre o número lógico. El lenguaje de programación para una acción puede ser ST, LD o IL. Una acción está siempre conectada con enlaces, uno llegando a ella, otro saliendo de ella.

Condiciones FC

Una **Condición** representa una decisión booleana . Una condición está identificada por un número y un nombre.
Según la evaluación de la correspondiente expresión en ST, LD o IL, el flujo se dirige por el camino de "SÍ" o "NO". Debajo están los posible dibujos para un símbolo de condición:



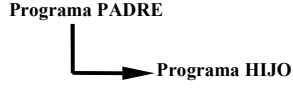
Dos objetos diferentes del mismo diagrama no pueden tener el mismo nombre o número lógico. La programación de una decisión es bien
- una expresión en ST, o
- un único contacto en LD, sin ningún símbolo para su única bobina, o
- diversas instrucciones en IL. El registro IL (o resultado actual) se usa para evaluar la condición.

Cuando se programa en texto ST la expresión puede ser opcionalmente seguida por un punto y coma. Cuando se programa en LD, la bobina única representa el estado de la condición. Una condición es igual:
- 0 o FALSO dirige el flujo hacia NO
- 1 o VERDADERO dirige el flujo hacia SÍ

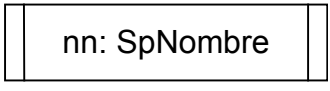
Una condición está siempre conectada con un enlace de llegada y deben definirse las dos condiciones de salida.

Subprograma FC

El sistema permite la descripción de la estructura vertical de los programas FC. Los programas FC están organizados según un árbol jerárquico. Cada programa FC puede llamar a otros programas FC. Este programa se llama programa hijo del programa FC que lo llama. Los programas FC que llaman a subprogramas FC se llaman programas padre. Los programas FC se unen juntos a un árbol jerárquico principal usando una relación padre-hijo:



Un símbolo de subprograma en un diagrama de flujo representa una llamada a un subprograma diagrama de flujo. La ejecución del programa padre está suspendida hasta que el subprograma termine de ejecutarse. Un subprograma diagrama de flujo se identifica con un numero y un nombre, como otros programas, funciones o bloques de función. Debajo está el dibujo de un símbolo de "llamada a subprograma":

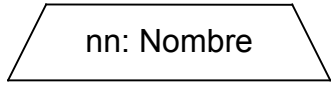


Dos objetos diferentes del mismo diagrama no pueden tener el mismo nombre o mismo número lógico. Las reglas básicas que implica la estructura jerárquica FC son:
- Programas FC que no tienen padre se llaman programas FC principales.
- Programas FC principales se activan por el sistema cuando empieza la aplicación
- Un programa puede tener varios programas hijos
- Un programa hijo no puede tener más que un padre
- Un programa hijo sólo puede ser llamado por su padre
- Un programa no puede llamar al hijo de uno de sus hijos

El mismo programa puede aparecer varias veces en diagrama del padre. No puede aparecer en ramas diferentes de las misma divergencia paralela. Una llamada a un subprograma en diagrama de flujo representa la ejecución completa del subdiagrama. La ejecución del padre está suspendida durante la actuación del subdiagrama. Los bloques de llamada del subprograma deben seguir las mismas reglas de conexión como los definidos para Acciones.

Acción específica de E/S FC

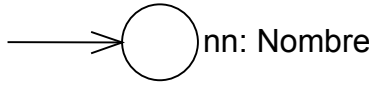
Un símbolo de una acción específica de E/S FC representa las acciones a ser realizadas. Como otras acciones, una acción de E/S específica se identifica por un número y un nombre. La misma semántica se utiliza en las acciones standard y especificaciones de E/S. El objetivo de una acción específica de E/S es sólo hacer el diagrama más legible y centrarse en las partes no portables del diagrama. Usar las acciones especificas es una característica opcional. Debajo está el diagrama de un símbolo de "acción específica de E/S:



Los bloques específicos de E/S tienen el mismo comportamiento que las acciones standard. Esto afecta a sus propiedades, a la programación ST, LD o IL, y a las reglas de conexión.

Conectores FC

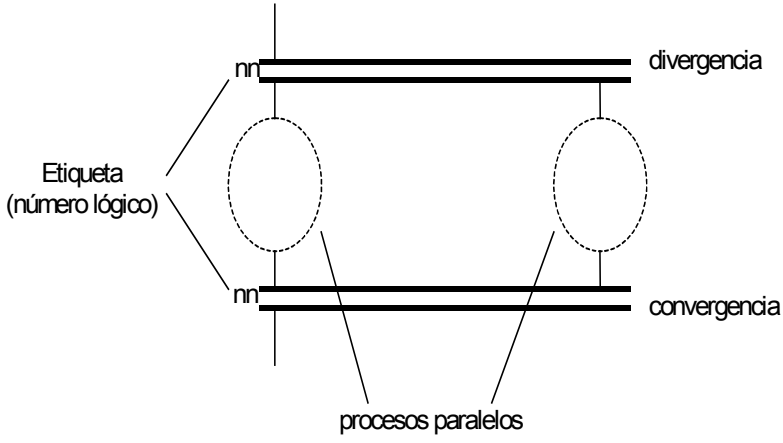
Los conectores se utilizan para representar un enlace entre dos puntos del diagrama sin dibujarlo. Un conector se representa por un círculo y se conecta a la fuente de flujo. El dibujo del conector se completa, en el lado apropiado (dependiendo de la dirección del flujo de datos), por la identificación del punto de destino (generalmente el nombre del símbolo objeto). Debajo está el dibujo standard de un conector:



Un conector siempre se dirige a un símbolo definido del diagrama de flujo. El símbolo de destino se identifica por un número lógico. El objetivo de un conector puede ser también el número lógico (etiqueta) de una divergencia paralela.

Ramas paralelas FC

Las ramas paralelas representan ejecución simultánea de subdiagramas. El sistema de "divergencia / convergencia" se identifica por un número lógico((etiqueta). Cada subprograma tiene un único "inicio" de flujo y un único "Fin" de flujo. Las ramas paralelas comienzan en un símbolo de "divergencia", y terminan en un símbolo de "convergencia". Las líneas horizontales dobles se utilizan para representar convergencias y divergencias:



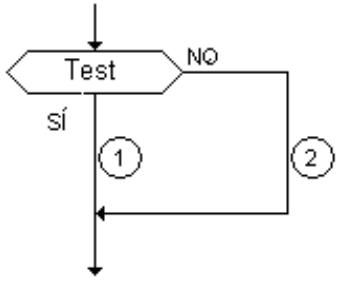
El número lógico (etiqueta) de una rama paralela puede ser utilizado como objetivo de un conector. El número lógico es el mismo para la divergencia y la convergencia del mismo sistema paralelo. No puede utilizarse para otros símbolos en el diagrama.

- ***Comentarios FC***

Un bloque de **comentario** contiene texto que no tiene sentido para la semántica del diagrama. Se puede insertar en cualquier lugar del espacio no usado de la ventana documento del diagrama de flujo y se utiliza para documentar el programa. Debajo está el dibujo del símbolo de "comentario" :

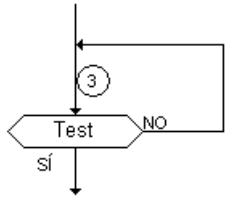
B.4.2 Ejemplos de estructuras complejas FC

Esta sección muestra ejemplos de estructuras complejas que pueden definirse en un diagrama de flujo. Dichas estructuras son combinaciones de objetos básicos enlazados.



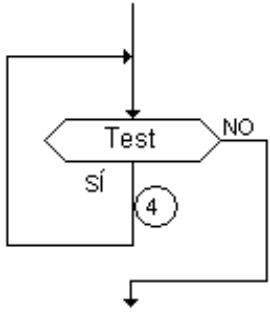
IF / THEN / ELSE

(1) lugar para acciones “THEN” que se inserten
(2) lugar para acciones "ELSE" que se inserten



REPEAT / UNTIL

(3) lugar para acciones repetidas a ser insertadas



WHILE / DO

(4) lugar para acciones repetidas a ser insertadas

B.4.3 Comportamiento dinámico FC

La **ejecución** de un programa en diagrama de flujo se puede explicar como sigue:

- El símbolo de Inicio lleva un ciclo del sistema
- El símbolo de Fin lleva un ciclo del sistema y termina la ejecución del programa. Después de que se alcanza este símbolo no se ejecuta ninguna acción más sobre el diagrama.
- La ejecución de una acción lleva un ciclo de sistema.
- En el caso de ramas paralelas:
- Todas las acciones (una por rama) que sigan a la divergencia se ejecutan en el mismo ciclo.
- La convergencia se "pasa" sólo cuando todas las ramas han finalizado su ejecución.

Nota: Contrariamente a SFC, una acción no es un estado estable. No hay repetición de instrucciones mientras el símbolo de acción está destacado.

B.4.4 Verificación FC

Aparte de la programación unida a ST, LD o IL, otras **reglas sintácticas** se aplican al propio diagrama de flujo. Debajo está la lista de las reglas principales:

- Todos los puntos de "conexión" deben de estar unidos. (la conexión al símbolo de "Fin" puede omitirse)
- Todos los símbolos deben de estar unidos (no debe aparecer ninguna parte aislada)
- Todos los conectores deben tener un destino valido
- Un símbolo en una rama paralela no puede estar unido a otros símbolos fuera de la rama (cada rama debe de estar conectada)
- El mismo subprograma no puede ser llamado desde diferentes ramas paralelas en la misma divergencia

Se presentan otros errores sintácticos menores:

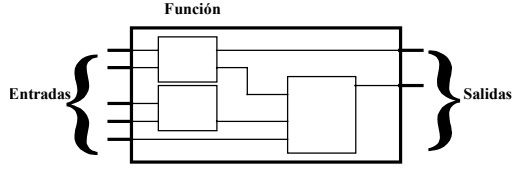
- Se ignoran las ramas vacías en una divergencia
- Una divergencia con sólo una rama llena no es una divergencia
- Las acciones vacías (sin programación) se consideran como pasos durante el tiempo programado de ejecución
- las decisiones vacías (sin programación) se consideran como "siempre verdad".

B.5 Lenguaje FBD

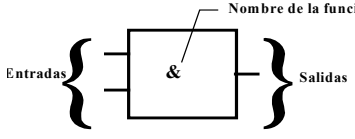
El **Diagrama de Bloques de función** (FBD) es un lenguaje gráfico. Permite al programador la construcción de procedimientos complejos, tomando **funciones** existentes de la librería ISaGRAF y **enlazándolas** en la zona del diagrama gráfico.

B.5.1 Formato principal del diagrama FBD

El diagrama FBD describe una función entre **variables de entrada** y **variables de salida**. Una función se describe como un conjunto de **bloques de función elementales**. Las variables de entrada y salida están vinculadas a los bloques por medio de **líneas de conexión**. Una salida de un bloque de función también puede conectarse a la entrada de otro bloque.



Una función completa manejada por un programa FBD está construida con bloques de función **elementales** y estándares procedentes de la librería ISaGRAF. Cada bloque de función tiene un número fijo de **puntos de conexión de entradas** y un número fijo de **puntos de conexión de salidas**. Un bloque de función está representado por un único **rectángulo**. Las entradas se conectan a su borde **izquierdo**. Las salidas se conectan a su borde **derecho**. Un bloque de función elementales lleva a cabo una única **función** entre sus entradas y salidas. El nombre de la función que tiene que desempeñar el bloque está escrito en su símbolo rectangular. Cada entrada o salida de un bloque posee un **tipo** bien definido.



Las variables de entrada de un programa FBD tienen que estar conectadas a los puntos de conexión de entradas de los bloques de función. El tipo de cada variable tiene que ser el mismo que se espera para la entrada asociada. Una entrada de un diagrama FBD puede ser una expresión **constante**, cualquier variable **interna** o de **entrada**, o una variable de **salida**.

Las variables de salida de un programa FBD tienen que estar conectadas a los puntos de conexión de salidas de los bloques de función. El tipo de cada variable tiene que ser el mismo que se espera para la salida de bloque asociada. Una salida de un diagrama FBD

puede ser cualquier variable **interna** o de **salida**, o el nombre del programa (sólo en el caso de **subprogramas**). Cuando una salida es el nombre del subprograma que se está editando en ese momento, representa la asignación del valor de retorno del subprograma (retorno al programa invocante).

Las variables de entrada y salida y las entradas y salidas de los bloques de función están unidas por medio de **líneas de conexión**. Pueden utilizarse líneas simples para **conectar** dos puntos lógicos del diagrama:

- Una variable de entrada y una entrada de un bloque de función
- Una salida de un bloque de función y una entrada de otro bloque
- Una salida de un bloque de función y una variable de salida

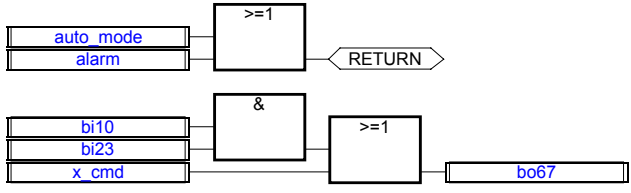
La conexión está **orientada**, lo que significa que la línea lleva datos asociados desde la extremidad izquierda hasta la extremidad derecha. Las extremidades izquierda y derecha de la línea de conexión deben ser del **mismo tipo**.

Puede utilizarse una conexión múltiple en la extremidad derecha para difundir una información desde la extremidad izquierda hacia cada una de las extremidades derechas. Todas las extremidades de la conexión deben ser del mismo tipo.

B.5.1.1 Sentencia RETURN

La palabra clave "**<RETURN>**", o RETORNO, puede aparecer como una salida del diagrama. Tiene que estar conectada a un punto de conexión de salida booleana de un bloque de función. La sentencia RETURN representa una **terminación condicional** del programa: si la salida de la caja que está conectada a la sentencia tiene el valor booleano **VERDADERO**, el final (la parte restante) del diagrama no se ejecuta.

(* **Ejemplo de un programa FBD que utiliza la sentencia RETURN** *)




```
(* equivalencia ST: *)
If auto_mode OR alarm Then
        Return;
End_if;
bo67 := (bi10 AND bi23) OR x_cmd;
```

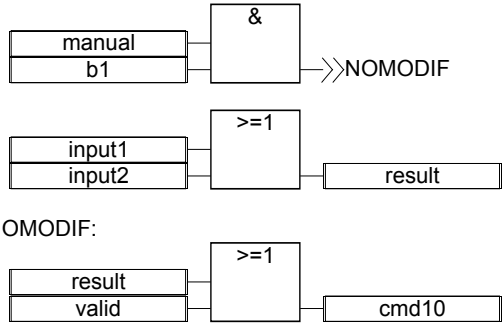
B.5.1.2 Saltos y etiquetas

Se utilizan los saltos y las etiquetas para controlar la ejecución del diagrama. No pueden conectarse otros objetos a la derecha de un símbolo de salto o etiqueta. Se emplean las siguientes notaciones:

>>LAB ..... salto a una etiqueta (el nombre de la etiqueta es "LAB")
LAB: ..... definición de una etiqueta (el nombre de la etiqueta es "LAB")

Si la línea de conexión situada a la **izquierda** del símbolo de salto muestra el estado booleano **VERDADERO**, la ejecución del programa salta directamente después del símbolo de etiqueta correspondiente.

(* **Ejemplo de un programa FBD que utiliza etiquetas y saltos** *)



(* Equivalencia IL: *)

```
                ld      manual
                and     b1
                jmpc    NOMODIF
                ld      input1
                or      input2
                st      result
NOMODIF:        ld      result
                or      valid
                st      cmd10
```

B.5.1.3 Negación booleana

Una línea de conexión simple, con su extremidad derecha conectada a una entrada de un bloque de función, puede terminar con una **negación booleana**. La negación está representada por un círculo pequeño. Cuando se utiliza una negación booleana, las extremidades izquierda y derecha de la línea de conexión deben ser del tipo **BOOLEANO**.

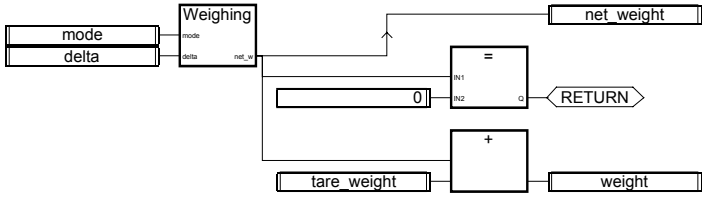
(* **Ejemplo de un programa FBD utilizando etiquetas y saltos** *)

(* Equivalencia ST: *)
output1 := input1 AND NOT (input2);

B.5.1.4 Invocación de funciones o bloques de función desde FBD

El lenguaje FBD permite la invocación de subprogramas, funciones o bloques de función. Un subprograma, función o bloque de función está representado por una caja de función. El nombre que aparece en la caja es el nombre del subprograma, de la función o del bloque de función.
En el caso de un subprograma o una función, el valor de retorno es la única salida de la caja de función. Los bloques de función pueden tener más de una salida.

(* **Ejemplo de un programa FBD utilizando el bloque de SUBPROGRAMA** *)

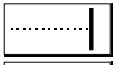


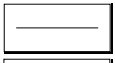
(* Equivalencia ST *)
net_weight := Weighing (mode, delta); (* llamar a sub-programa *)
If (net_weight = 0) Then Return; End_if;
weight := net_weight + tare_weight;

B.6 Lenguaje LD

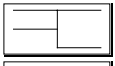
El Diagrama de Escalera (Contactos) (LD) es una representación gráfica de ecuaciones booleanas que combina **contactos** (argumentos de entrada) con **bobinas** (resultados de salida). El lenguaje LD permite la descripción de pruebas y modificaciones de datos **booleanos** mediante la colocación de **simbolos gráficos** en el diagrama del programa. Los simbolos gráficos LD se organizan dentro del diagrama del mismo modo que en un diagrama de contactos eléctricos. Un diagrama LD está conectado en sus laterales izquierdo y derecho con **carriles de potencia** verticales. Estos son los componentes gráficos básicos de un diagrama LD:

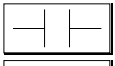

---------- Carril de potencia vertical izquierdo

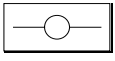

---------- Carril de potencia vertical derecho


---------- Línea de conexión horizontal


---------- Línea de conexión vertical

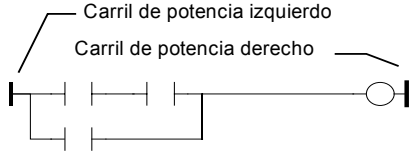

---------- Líneas de conexión múltiples (todas interconectadas)


---------- Contacto asociado a una variable

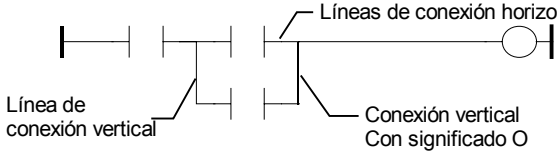

---------- Bobina asociada a una salida o a una variable interna

B.6.1 Carriles de potencia y líneas de conexión

Un diagrama LD está delimitado en sus laterales izquierdo y derecho por unas líneas verticales, denominadas **carril de potencia izquierdo** y **carril de potencia derecho**, respectivamente.



Los símbolos gráficos de los diagramas LD están conectados a carriles de potencia o a otros símbolos por medio de **líneas de conexión**. Las líneas de conexión pueden ser horizontales o verticales.



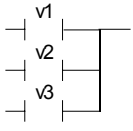
Cada segmento de línea tiene un estado booleano **FALSO** o **VERDADERO**. El estado booleano es el mismo para todos los segmentos que estén conectados directamente entre sí. Cualquier línea horizontal que esté conectada al **carril de potencia izquierdo** tiene el estado **VERDADERO**.

B.6.2 Conexiones múltiples

El estado booleano que se atribuye a una única línea de conexión horizontal es el mismo que en las extremidades izquierda y derecha de la línea. La combinación de líneas de conexión horizontales y verticales permite la construcción de **conexiones múltiples**. El estado booleano de las extremidades de una conexión multiple obedece una reglas lógicas.

Una **conexión múltiple a la izquierda** combina **más de una** línea horizontal conectada en el lateral izquierdo de una línea vertical, y **una** línea conectada en su lateral **derecho**. El estado booleano de la extremidad derecha es la '**O**' **LÓGICA** entre todas las extremidades de la izquierda.

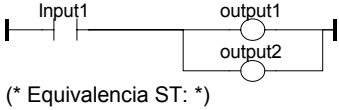
(* **Ejemplo de conexión múltiple a la IZQUIERDA** *)



(* estado de extremidad derecha es (v1 OR v2 OR v3) *)

Una **conexión múltiple a la derecha** combina **una** línea horizontal conectada al lateral **izquierdo** de una línea vertical, con **más de una** línea conectada a su lateral **derecho**. El estado booleano de la extremidad izquierda se propaga a cada una de las extremidades derechas.

(* **Ejemplo de una conexión múltiple a la DERECHA** *)

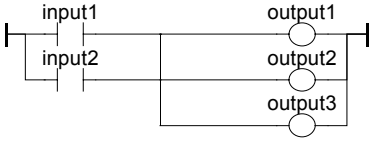


(* Equivalencia ST: *)

output1 := input1;
output2 := input1;

Una **conexión múltiple a la derecha y a la izquierda** combina **más de una** línea horizontal conectada al lateral **izquierdo** de una línea vertical, y **más de una** línea conectada a su lateral **derecho**. El estado booleano de cada una de sus extremidades de la derecha es el '**O**' **LÓGICO** del conjunto de extremidades de la izquierda.

(* **Ejemplo de conexión múltiple IZQUIERDA y DERECHA** *)



(* Equivalencia ST: *)
output1 := input1 O input2;
output2 := input1 O input2;
output3 := input1 O input2;

B.6.3 Contactos y bobinas básicos del lenguaje LD

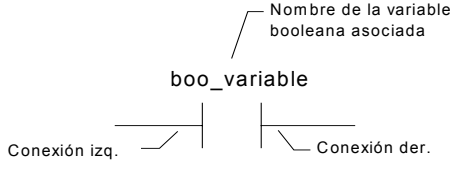
Se dispone de diversos símbolos para los contactos de entrada :

- Contacto directo
- Contacto invertido
- Contactos con detección de flancos

Se dispone de diversos símbolos para las bobinas de salida :

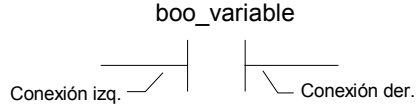
- Bobina directa
- Bobina invertida
- Bobina **SET**
- Bobina **RESET**
- Bobinas con detección de flancos

El nombre de la variable se escribe por encima de cualquiera de estos símbolos gráficos:



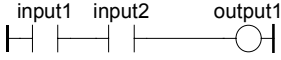
B.6.3.1 Contacto directo

Un contacto directo permite llevar a cabo una **operación booleana** entre un estado de **línea de conexión** y una **variable** booleana.



El estado de la línea de conexión situada a la derecha del contacto es el **'Y' (AND) LÓGICO** entre el estado de la línea de conexión izquierda y el valor de la variable asociada al contacto.

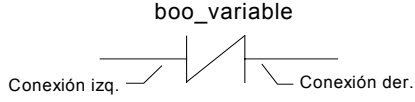
(* **Ejemplo utilizando contactos DIRECTOS** *)




```
(* Equivalencia ST: *)
output1 := input1 AND input2;
```

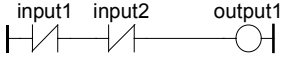
B.6.3.2 Contacto invertido

Un contacto invertido permite llevar a cabo una **operación** booleana entre un estado de **línea de conexión** y la negación booleana de una **variable** booleana.



El estado de la línea de conexión situada a la derecha del contacto es la '**Y**' **(AND) LÓGICO** entre el estado de la línea de conexión izquierda y la **negación booleana** del valor de la variable asociada al contacto.

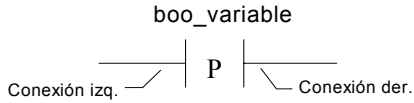
(* **Ejemplo utilizando contactos INVERTIDOS** *)




```
(* Equivalencia ST: *)
output1 := NOT (input1) AND NOT (input2);
```

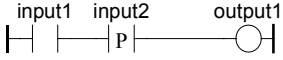
B.6.3.3 Contacto con detección de flancos de subida

Este contacto (positivo) permite llevar a cabo una **operación booleana** entre el estado de una **línea de conexión** y el flanco de subida de una **variable** booleana.



El estado de la línea de conexión situada a la derecha del contacto pasa a **VERDADERO** cuando el estado de la línea de conexión de la izquierda es **VERDADERO**, y el estado de la variable asociada **se eleva** de FALSO a VERDADERO. Se repone en FALSO en cualquier otro caso.

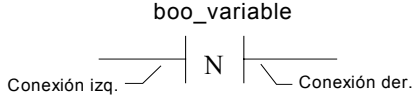
(* **Ejemplo utilizando contactos de FLANCO DE SUBIDA** *)




```
(* Equivalencia ST: *)
output1 := input1 AND (input2 AND NOT (input2prev));
(* input2prev es el valor de input2 del ciclo anterior *)
```

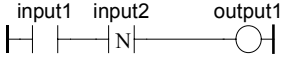
B.6.3.4 Contacto con detección de flanco de bajada

Este contacto (negativo) permite llevar a cabo una **operación booleana** entre un estado de **línea de conexión** y el flanco de bajada de una **variable** booleana.



El estado de la línea de conexión situada a la derecha del contacto pasa a **VERDADERO** cuando el estado de la línea de conexión de la izquierda es **VERDADERO**, y el estado de la variable asociada **desciende** de VERDADERO a FALSO. Se repone a FALSO en cualquier otro caso.

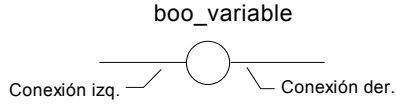
(* **Ejemplo utilizando contactos de FLANCO DE BAJADA** *)




```
(* Equivalencia ST: *)
output1 := input1 AND (NOT (input2) AND input2prev);
(*input2prev es el valor de iniput2 del ciclo anterior *)
```

B.6.3.5 Bobina directa

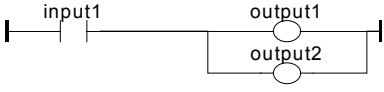
Las bobinas directas permiten la **salida booleana** del estado booleano de una **línea de conexión**.



La variable asociada se asigna al **estado booleano de la conexión izquierda**. El estado de la conexión izquierda se propaga hacia la conexión derecha. La conexión derecha puede conectarse al carril de potencia vertical de la derecha.

La variable booleana asociada tiene que ser de **SALIDA** o **INTERNA**.
El nombre asociado puede ser el nombre del programa (sólo en el caso de **subprogramas**). Esto corresponde a la asignación del valor de retorno del subprograma.

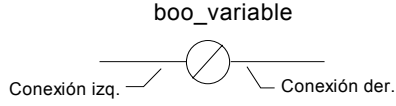
(* **Ejemplo utilizando bobinas DIRECTAS** *)




```
(* Equivalencia ST: *)
output1 := input1;
output2 := input1;
```

B.6.3.6 Bobina invertida

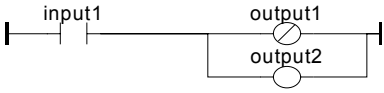
Las bobinas invertidas permiten una **salida booleana** de acuerdo con la **negación** booleana de un estado de **línea de conexión**.



La variable asociada se asigna a la **negación booleana del estado de la conexión izquierda**. El estado de la conexión izquierda se propaga hacia la conexión derecha. La conexión derecha puede conectarse al carril de potencia vertical de la derecha.

La variable booleana asociada tiene que ser de **SALIDA** o **INTERNA**.
El nombre asociado puede ser el nombre del programa (sólo en el caso de **subprogramas**). Esto corresponde a la asignación del valor de retorno del subprograma.

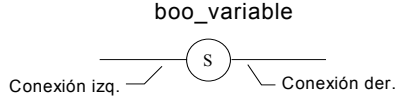
(* **Ejemplo utilizando bobinas INVERTIDAS** *)




```
(* Equivalencia ST: *)
output1 := NOT (input1);
output2 := input1;
```

B.6.3.7 Bobina SET (Establecer)

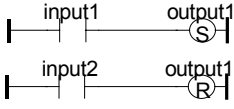
Las bobinas SET permiten la **salida booleana** del estado booleano de una **línea de conexión**.



La variable asociada **PASA A VERDADERO** cuando el **estado booleano de la conexión izquierda** se convierte en VERDADERO. La variable de salida mantiene este valor hasta que una bobina de "RESET" emite una orden inversa. El estado de la conexión izquierda se propaga hacia la conexión derecha. La conexión derecha puede conectarse al carril de potencia vertical de la derecha.

La variable booleana asociada tiene que ser de **SALIDA** o **INTERNA**.

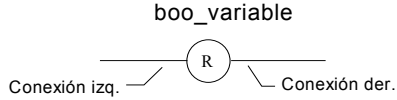
(* **Ejemplo utilizando bobinas de "SET" y "RESET"** *)




```
(* Equivalencia ST: *)
IF input1 THEN
  output1 := VERDADERO;
END_IF;
IF input2 THEN
  output1 := FALSO;
END_IF;
```

B.6.3.8 Bobinas RESET (Restablecer)

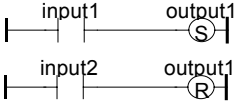
Las bobinas RESET permiten la **salida booleana** del estado booleano de una **línea de conexión**.



La variable asociada **PASA A FALSO** cuando el **estado booleano de la conexión izquierda** se convierte en VERDADERO. La variable de salida mantiene este valor hasta que una bobina de SET emite una orden inversa. El estado de la conexión izquierda se propaga hacia la conexión derecha. La conexión derecha puede conectarse al carril de potencia vertical de la derecha.

La variable booleana asociada tiene que ser de **SALIDA** o **INTERNA**.

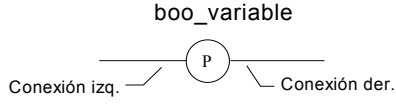
(* **Ejemplo utilizando bobinas de "SET" y "RESET"** *)




```
(* Equivalencia ST: *)
IF input1 THEN
  output1 := VERDADERO;
END_IF;
IF input2 THEN
  output1 := FALSO;
END_IF;
```

B.6.3.9 Bobina con detección de flancos de subida

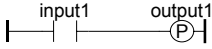
Las bobinas "Positivas" permiten la **salida booleana** del estado booleano de una **línea de conexión**. Este tipo de bucle sólo está disponible cuando se utiliza el editor *Quick Ladder*.



La variable asociada **PASA A VERDADERO** cuando el **estado booleano de la conexión izquierda** se eleva de FALSO a VERDADERO. La variable de salida se reinicia en FALSO en cualquier otro caso. El estado de la conexión izquierda se propaga hacia la conexión derecha. La conexión derecha puede conectarse al carril de potencia vertical de la derecha.

La variable booleana asociada tiene que ser de **SALIDA** o **INTERNA**.

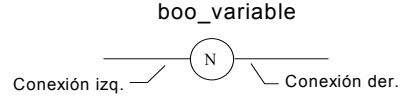
(* **Ejemplo utilizando una bobina "Positiva"** *)




```
IF (input1 and NOT(input1prev)) THEN
  output1 := VERDADERO;
ELSE
  output1 := FALSO;
END_IF;
(* input1prev es el valor de input1 en el ciclo anterior *)
```

B.6.3.10 Bobina con detección de flanco de bajada

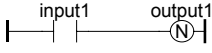
Las bobinas "Negativas" permiten la **salida booleana** del estado booleano de una **línea de conexión**. Este tipo de bucle sólo está disponible cuando se utiliza el editor *Quick Ladder*.



La variable asociada **PASA A VERDADERO** cuando el **estado booleano de la conexión izquierda** desciende de VERDADERO a FALSO. La variable de salida se reinicia en FALSO en cualquier otro caso. El estado de la conexión izquierda se propaga hacia la conexión derecha. La conexión derecha puede conectarse al carril de potencia vertical de la derecha.

La variable booleana asociada tiene que ser de **SALIDA** o **INTERNA**.

(* **Ejemplo utilizando una bobina "Negativa"** *)



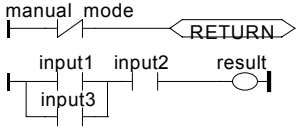

```
(* Equivalencia ST: *)
IF (NOT(input1) and input1prev) THEN
  output1 := VERDADERO;
ELSE
  output1 := FALSO;
END_IF;
(* input1prev es el valor de input1 en el ciclo anterior *)
```

B.6.4 Sentencia RETURN

Se puede utilizar la etiqueta **RETURN**, como salida para representar un final condicional del programa. No se puede colocar conexión alguna a la derecha de un símbolo de RETURN.

Si la línea de **conexión izquierda** muestra un estado booleano **VERDADERO**, el programa finaliza sin ejecutar las ecuaciones que aparecen en las siguientes líneas del diagrama.
Nota: Cuando el programa LD es un subprograma, su nombre tiene que estar asociado con una bobina de salida para poder establecer el valor de retorno (retorno al programa invocante).

(* **Ejemplo utilizando el símbolo de RETURN** *)




```
(* Equivalencia ST: *)
If Not (manual_mode) Then RETURN; End_if;
result := (input1 OR input3) AND input2;
```

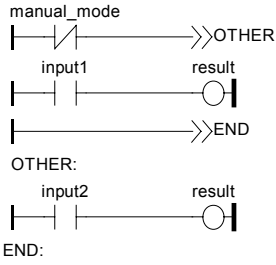
B.6.5 Saltos y etiquetas

Pueden emplearse etiquetas y símbolos de SALTOS condicionales e incondicionales para controlar la ejecución del diagrama. No se puede colocar conexión alguna a la derecha del símbolo de etiqueta y salto. Se utilizan las siguientes notaciones:

>>LAB ..... saltar a la etiqueta llamada "LAB"
LAB: ..... definición de la etiqueta llamada "LAB"

Si la **conexión situada a la izquierda** del símbolo de salto tiene el estado booleano **VERDADERO**, se lanzará la ejecución del programa después del símbolo de etiqueta.

(* **Ejemplo utilizando los símbolos de SALTO y ETIQUETA** *)

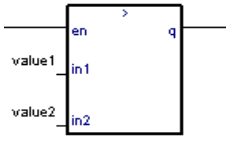



```
(* Equivalencia IL: *)
                ldn         manual_mode
                jmpc        other
                ld          input1
                st          result
                jmp         END
OTHER:          ld          input2
                st          result
END:            (* fin del programa *)
```

B.6.6 Bloques en LD

Utilizando el editor *Quick LD*, se pueden conectar cajas de funciones a líneas booleanas. Una función puede ser en realidad un operador, un bloque de función o una función. Ya que todos los bloques no siempre tienen una entrada booleana y/o una salida booleana, la inserción de bloques en un diagrama LD conduce a la incorporación de nuevos parámetros EN, ENO a la interfaz del bloque. No se añaden los parámetros EN, ENO si se utiliza el editor FBD/LD, ya que se puede conectar la variable con el tipo necesario.

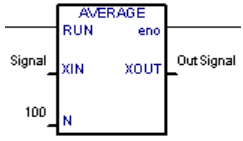
• **La entrada "EN"**
En algunos operadores, funciones o bloques de función, la primera entrada no incluye datos de tipo booleano. Dado que la primera entrada tiene que estar siempre conectada al escalón (línea), se inserta otra entrada, denominada "**EN**", en la primera posición de forma automática. El bloque sólo se ejecuta si la entrada **EN** es VERDADERA. A continuación se muestra el ejemplo de un operador de comparación, junto con el código equivalente expresado en ST:




```
IF rung_state THEN
  q := (value1 > value 2);
ELSE
  q := FALSO;
END_IF;
(* continúa el escalón con el estado q *)
```

• **La salida "ENO"**
En algunos operadores, funciones o bloques de función, la primera entrada no incluye datos del tipo booleano. Dado que la primera salida tiene que estar conectada siempre al escalón

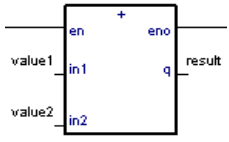
(línea), se inserta otra salida, denominada "**ENO**", en la primera posición de forma automática. La salida **ENO** siempre asume el mismo estado que la primera entrada del bloque. A continuación se muestra un ejemplo con el bloque de función AVERAGE, o PROMEDIO, con el código equivalente expresado en ST:




```
AVERAGE(rung_state, Signal, 100);
OutSignal := AVERAGE.XOUT;
eno := rung_state;
(* continúa el escalón con estado eno *)
```

● **Los parámetros "EN" y "ENO"**

En algunos casos, se necesitan tano **EN** como **ENO**. A continuación se muestra un ejemplo que incluye un operador aritmético, con el código equivalente expresado en ST:




```
IF rung_state THEN
  result := (value1 + value2);
END_IF;
eno := rung_state;
(* continue rung with eno state *)
```

B.7 Lenguaje ST

ST (**Texto Estructurado**) es un lenguaje estructurado de alto nivel diseñado para procesos de automatización. Este lenguaje se usa principalmente para implementar procesos complejos que no pueden expresarse fácilmente con lenguajes gráficos. ST es el lenguaje por defecto para la descripción de las acciones contenidas en los pasos y condiciones asociados a las transiciones del lenguaje **SFC**.

B.7.1 Sintaxis principal de ST

Un programa ST es una lista de **sentencias** ST. Cada sentencia acaba con un separador constituido por un punto y coma (**";"**). Los nombres utilizados en el código fuente (identificadores de variables, constantes, palabras clave del lenguaje...) se espacian con **separadores inactivos** (carácter de espacio, fin de línea o pasos de tabulador) o con **separadores activos**, que poseen un significado bien definido (por ejemplo, el separador "**>**" indica la comparación "mayor que"). Se pueden insertar comentarios en el texto libremente. Un comentario tiene que empezar por **"(*"** y finalizar por **"*)"**. Cada sentencia termina con un separador en forma de punto y coma (**";"**). Estos son los tipos básicos de sentencias ST:

- sentencia de **asignación** (variable := expresión;)
- invocación de **subprogramas** o **funciones**
- invocación de **bloques de función**
- sentencias de **selección** (IF, THEN, ELSE, CASE...)
- sentencias de **iteración** (FOR, WHILE, REPEAT...)
- sentencias de **control** (RETURN, EXIT...)
- sentencias especiales para enlaces con otros lenguajes tales como **SFC**

Se puede introducir separadores inactivos libremente entre los separadores activos, expresiones constantes e identificadores. Los separadores inactivos en ST son: **Espacios** o caracteres en blanco, **Tabulaciones** y caracteres de **Fin de línea**. A diferencia de los lenguajes con formato de líneas, como IL, pueden introducirse fines de línea en cualquier parte del programa. Se deberá tener en cuenta las siguientes normas a la hora de utilizar separadores inactivos, para aumentar la legibilidad del programa ST.

- No escribir más de una sentencia por línea
- Utilizar pasos de tabulador para indentar sentencias complejas
- Insertar comentarios para aumentar la legibilidad de líneas o párrafos

Legibilidad de fuente - ejemplos:

Baja legibilidad	Alta legibilidad

imax := max_ite; cond := X12;
if not(cond (* alarm *)
then return; end_if;
for i (* index *) := 1 to max_ite
do if i <> 2 then Spcall();
end_if; end_for;
(* no effect if alarm *)
	(* imax : number of iterations *)
	(* i: FOR statement index *)
	(* cond: process validity *)
	imax := max_ite;
	cond := X12;
	if not (cond) then
	  return;
	end_if;
	
	(* process loop *)
	for i := 1 to max_ite do
	  if i <> 2 then
	  Spcall ();
	  end_if;
	end_for;

B.7.2 Expresiones y paréntesis

Las expresiones ST combinan **operadores** ST y **operandos** variables o constantes. Para cada expresión individual (que combine operandos con un operador ST), los operandos deben ser del mismo **tipo**. Dicha expresión tiene el mismo tipo que sus operandos, y se puede utilizar en una expresión más compleja. Por ejemplo:

(boo_var1 AND boo_var2)	posee tipo BOO
not (boo_var1)	posee tipo BOO
(sin (3.14) + 0.72)	posee tipo ANALÓGICO REAL
(t#1s23 + 1.78)	no es una expresión válida

Se utilizan los **paréntesis** para aislar las subpartes de la expresión y para estructurar explícitamente la prioridad de las operaciones. Cuando no aparecen paréntesis en una expresión compleja, la secuencia de operaciones viene indicada implícitamente por la **prioridad** por defecto que exista entre los operadores ST. Por ejemplo:

2 + 3 * 6	es igual a 2+18=20	ya que el operador de multiplicación tiene una prioridad mayor
(2+3) * 6	es igual a 5*6=30	la prioridad viene dada por los paréntesis

Advertencia: Se puede incluir un número máximo de **8** niveles de paréntesis dentro de una expresión.

B.7.3 Invocación de funciones o bloques de función

Puede utilizarse la invocación estándar de funciones ST para cada uno de los siguientes objetos:

- Subprogramas
- Librerías de funciones y bloques de función escritos en lenguajes IEC
- Funciones y bloques de función "C"
- Funciones de conversión de tipos

B.7.3.1 Invocación de subprogramas o funciones

Nombre: nombre del subprograma o función de librería invocado, escrito en un lenguaje IEC o en “C”.

Significado: invoca un subprograma o función ST, IL, LD o FBD o una función "C" y obtiene su valor de retorno.

Sintaxis: **<variable> := <subprog> (<par1>,** ... **<parN>);**

Operandos: El tipo de valor de retorno y los parámetros de invocación deben estar conformes con la interfaz definida para el subprograma.

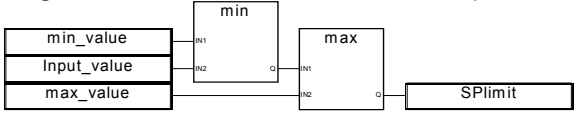
Valor de retorno: valor devuelto por el subprograma.

Se puede utilizar la invocación de subprogramas en cualquier expresión. Puede utilizarse igualmente en una transición SFC.

Ejemplo1: Invocación de subprograma

```
(* Programa ST principal *)
(* consigue un valor analógico y lo convierte a un valor de tiempo limitado*)
ana_timeprog := SPlimit ( tprog_cmd );
appl_timer := tmr (ana_timeprog * 100);
```

(* Programa FBD invocado de nombre 'SPlimit' *)



Ejemplo2: Invocación de función

```
(* las funciones usadas en expresiones complejas: min, max, right, mlen, y left son funciones
“C” estandar *)
limited_value := min (16, max (0, input_value) );
```

rol_msg := right (message, mlen (message) - 1) + left (message, 1);

B.7.3.2 Invocación de bloques de función

Nombre: nombre de la instancia de bloque de función.

Significado: invoca un bloque de función de la librería ISaGRAF o de la librería del usuario y accede a sus parámetros de retorno

Sintaxis:
(* llamada del bloque de función *)
<nombre_bloque> (<p1>, <p2> ...);
(* consigue sus parámetros de retorno *)
<resultado> := <nombre_bloque>. <ret_param1>;
...
<resultado> := <nombre_bloque>. <ret_paramN>;

Operandos: los parámetros son expresiones que concuerdan con el tipo de los parámetros especificados para ese bloque de función

Valor de retorno: Véase Sintaxis para obtener los parámetros de retorno.

Debe consultarse la librería ISaGRAF para hallar el significado y tipo de cada parámetro de bloque de función. Se tiene que declarar la instancia del bloque de función (nombre de la copia) en el diccionario.

Ejemplo :

(* Programa ST invocando un bloque de función *)

(* declarar la instancia del bloque en el diccionario: *)
(* trigb1 : bloque R_TRIG - detección flanco positivo *)

(* activación de bloque de función desde lenguaje ST *)
trigb1 (b1);
(* acceso a parámetros de retorno *)
If (trigb1.Q) Then nb_edge := nb_edge + 1; End_if;

B.7.4 Operadores booleanos específicos de ST

Los siguientes operadores booleanos son específicos del lenguaje ST:

- **REDGE** detección de flanco de subida
- **FEDGE** detección de flanco de bajada

También pueden utilizarse otros operadores booleanos estándares como:

- NOT negación booleana
- AND (&) 'Y' lógica
- OR 'O' lógica
- XOR 'O' exclusiva lógica

Su descripción se encuentra en la sección 'Operadores, bloques de función y funciones estándares'.

B.7.4.1 Operador "REDGE"

Nombre: **REDGE**

Significado: evalúa el flanco de subida de una expresión booleana completa

Sintaxis: **<flanco> := REDGE (<expresión_booleana>,<variable_memoria>);**

Operandos: el primer operando es cualquier variable booleana o expresión compleja
el segundo operando es una variable booleana interna que se utiliza para almacenar el último estado de la expresión

Valor de retorno: VERDADERO cuando la expresión cambia de FALSO a VERDADERO
FALSO en todos los demás casos

No se puede detectar el flanco de subida de una expresión en más de una ocasión durante el mismo ciclo de ejecución, utilizando el operador REDGE. Este operador se puede utilizar para describir la condición asociada a una transición SFC.

Advertencia: La variable booleana de "memoria" utilizada para almacenar el último estado de la expresión no puede ser utilizada como activador de los flancos de diferentes expresiones.

Cuando la expresión es una variable booleana denominada "**xxx**", se debe declarar una variable interna única llamada "**EDGE_xxx**" y utilizarla en las expresiones REDGE asociadas a esta variable. Este método asegura que la variable de memoria no quede sobreescrita durante otras evaluaciones REDGE.

Ejemplo:

(* Programa ST utilizando el operador REDGE *)

```
(* este programa cuenta los flancos positivos de una entrada booleana *)
(* Bi120 es una variable booleana de entrada *)
(* Edge_Bi120 es la memoria de variable de estado Bi120 *)

If REDGE (Bi120, Edge_Bi120) Then
        Counter := Counter + 1;
```

End_if;

Nota: Este operador no figura en la norma IEC1131-3. Existe la opción de utilizar el bloque estándar R_TRIG. Se ha mantenido por razones de compatibilidad.

B.7.4.2 Operador "FEDGE"

Nombre: **FEDGE**

Significado: evalúa el flanco de bajada de una expresión booleana

Sintaxis: **<flanco> := FEDGE (<expresión_booleana>, <variable_memoria>);**

Operandos: el primer operando es cualquier variable booleana o expresión compleja
el segundo operando es una variable booleana interna que se utiliza para almacenar el último estado de la expresón

Valor de retorno: VERDADERO cuando la expresión cambia de VERDADERO a FALSO
FALSO en todos los otros casos.

No se puede detectar el flanco de bajada de una expresión en más de una ocasión durante el mismo ciclo de ejecución, utilizando el operador FEDGE. Este operador se puede utilizar para describir la condición asociada a una transición SFC.

Advertencia: La variable booleana de “memoria” utilizada para almacenar el último estado de la expresión no puede ser utilizada como activador de los flancos de diferentes expresiones.

Cuando la expresión es una variable booleana denominada “**xxx**”, se debe declarar una variable interna única llamada "**EDGE_xxx**" y utilizarla en las expresiones FEDGE asociadas a esta variable. Este método asegura que la variable de memoria no quede sobreescrita durante otras evaluaciones FEDGE.

Ejemplo:

(* Programa ST utilizando el operador FEDGE *)

```
(* este programa cuenta los flancos negativos de una entrada booleana *)
(* Bi120 es una variable de entrada booleana *)
(* Edge_Bi120 es la memoria de variable de estado Bi120 *)

If FEDGE (Bi120, Edge_Bi120) Then
        Counter := Counter + 1;
End_if;
```

Nota: Este operador no figura en la norma IEC1131-3. Existe la opción de utilizar el bloque estándar F_TRIG. Se ha mantenido por razones de compatibilidad.

B.7.5 Sentencias básicas ST

Las sentencias básicas del lenguaje ST son:

- Asignación
- Sentencia **RETURN**
- Estructura **IF-THEN-ELSIF-ELSE**
- Sentencia **CASE**
- Sentencia de iteración **WHILE**
- Sentencia de iteración **REPEAT**
- Sentencia de iteración **FOR**
- Sentencia **EXIT**

B.7.5.1 Asignación

Nombre: **:=**

Significado: asigna una variable a una expresión

Sintaxis: **<variable> := <cualquier_expresión> ;**

Operandos: la variable debe ser interna o de salida
La variable y la expresión deben tener el mismo tipo

La expresión puede ser la invocación de un subprograma o una función de la librería ISaGRAF.

Ejemplo:

```
(* Programa ST con asignaciones *)

(* variable <<= variable *)
bo23 := bo10;

(* variable <<= expresión *)
bo56 := bx34 OR alrm100 & (level >= over_value);
result := (100 * input_value) / scale;

(* asignación con retorno de subprograma *)
rc := PSelect ( );

(* asignación con invocación de función *)
limited_value := min (16, max (0, input_value) );
```

B.7.5.2 Sentencia RETURN

Nombre: **RETURN**

Significado: termina la ejecución del programa actual

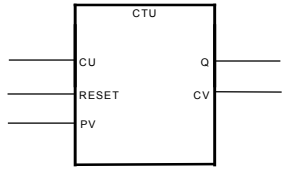
Sintaxis: **RETURN ;**

Operandos: (ninguno)

En un bloque de acciones SFC, la sentencia RETURN indica el final de la ejecución de ese bloque exclusivamente.

Ejemplo:

(* Especificación FBD del programa: contador programable *)



(* Implementación ST del programa, utilizando la sentencia RETURN *)

```
If not (CU) then
        Q := FALSO;
        CV := 0;
        RETURN;   (* termina el programa *)
end_if;

if R then
        CV := 0;
else
        if (CV < PV) then
                CV := CV + 1;
        end_if;
end_si;
Q := (CV >= PV);
```

B.7.5.3 Sentencia IF-THEN-ELSIF-ELSE

Nombre: **IF ... THEN ... ELSIF ... THEN ... ELSE ... END_IF**

Significado: ejecuta una de varias listas de sentencias ST
La selección se realiza en base al valor de una expresión booleana

Sintaxis:

```
IF <expresión_booleana> THEN
  <sentencia> ;
  <sentencia> ;
  ...
ELSIF <expresión_ booleana> THEN
  <sentencia> ;
  <sentencia> ;
  ...
ELSE
  <sentencia> ;
  <sentencia> ;
  ...
END_IF;
```

Las sentencias ELSE y ELSIF son opcionales. Si no se incluye la sentencia ELSE, no se ejecutan instrucciones cuando la condición es FALSO.

Ejemplo:

```
(* Programa ST utilizando la sentencia IF *)

IF manual AND not (alarm) THEN
        level := manual_level;
        bx126 := bi12 OR bi45;
ELSIF over_mode THEN
        level := max_level;
ELSE
        level := (lv16 * 100) / scale;
END_IF;

(* Estructura IF sin ELSE *)
If overflow THEN
          alarm_level := VERDADERO;
END_IF;
```

B.7.5.4 Sentencia CASE

Nombre: **CASE ... OF ... ELSE ... END_CASE**

Significado: ejecuta una de varias listas de sentencias ST
La selección se realiza en base a una expresión entera

Sintaxis:

```
CASE <expresión_entera> OF
  <valor> : <sentencias> ;
  <valor> , <valor> : <sentencias> ;
  ...
ELSE
  <sentencias> ;
END_CASE;
```

Los valores CASE tienen que ser expresiones constantes enteras. Varios valores, separados por comas, pueden conducir a la misma lista de sentencias. La sentencia ELSE es opcional.

Ejemplo:

(* Programa ST utilizando la sentencia CASE *)

```
CASE error_code OF
        255:    err_msg := 'División por cero';
                fatal_error := VERDADERO;
        1:      err_msg := 'Rebasamiento';
        2, 3:   err_msg := 'Signo erróneo';
ELSE
        err_msg := 'Error desconocido';
END_CASE;
```

B.7.5.5 Sentencia WHILE

Nombre: **WHILE ... DO ... END_WHILE**

Significado: estructura de iteración para un grupo de sentencias ST
La condición de "continuar" se evalúa ANTES que cualquier iteración

Sintaxis:

```
WHILE <expresión_booleana> DO
  <sentencia> ;
  <sentencia> ;
  ...
END_WHILE ;
```

Advertencia: Teniendo en cuenta que ISaGRAF es un sistema **síncrono**, las variables de entrada no se refrescan durante las iteraciones WHILE. No se puede utilizar el cambio de estado de una variable de entrada para describir la condición de una sentencia WHILE.

Ejemplo:

(* Programa ST utilizando la sentencia WHILE *)

```
(* este programa usa funciones “C” específicas para leer caracteres en un puerto serie *)

string := ''; (* cadena vacía *)
nbchar := 0;

WHILE ((nbchar < 16) & ComIsReady ( )) DO
        string := string + ComGetChar ( );
        nbchar := nbchar + 1;
END_WHILE;
```

B.7.5.6 Sentencia REPEAT

Nombre: **REPEAT ... UNTIL ... END_REPEAT**

Significado: estructura de iteración para un grupo de sentencias ST
La condición de "continuar" se evalúa DESPUÉS de cualquier iteración

Sintaxis:
REPEAT
<sentencia> ;
<sentencia> ;
...
UNTIL <condición_booleana>
END_REPEAT ;

Advertencia: Teniendo en cuenta que ISaGRAF es un sistema **síncrono**, las variables de entrada no se refrescan durante las iteraciones REPEAT. No se puede utilizar el cambio de estado de una variable de entrada para describir la condición de finalización de una sentencia REPEAT.

Ejemplo:

```
(* Programa ST utilizando la sentencia REPEAT *)

(* este programa utiliza funciones “C” específicas para leer caracteres en un puerto serie *)

string := ''; (* cadena vacía *)
nbchar := 0;
SI ComIsReady ( ) THEN
        REPEAT
                string := string + ComGetChar ( );
                nbchar := nbchar + 1;
        UNTIL ( (nbchar >= 16) OR NOT (ComIsReady ( )) )
        END_REPEAT;
END_IF;
```

B.7.5.7 Sentencia FOR

Nombre: **FOR ... TO ... BY ... DO ... END_FOR**

Significado: ejecuta un número limitado de iteraciones,
utilizando una variable analógica entera de índice

Sintaxis:
FOR <index> := <mini> TO <maxi> BY <step> DO
<sentencia> ;
<sentencia> ;
END_FOR;

Operandos:	**index**:	variable analógica interna incrementada en cualquier bucle
	mini:	valor inicial de referencia (antes del primer bucle)
	maxi:	valor máximo permitido para la referencia
	step:	incremento de la referencia en cada bucle

La sentencia [BY step] es opcional. Si no se especifica lo contrario, el paso incremental es 1.

Advertencia: Teniendo en cuenta que ISaGRAF es un sistema **síncrono**, las variables de entrada no se refrescan durante las iteraciones FOR.

Este es el equivalente "while" de una sentencia FOR:

```
index := mini;
while (index <= maxi) do
  <sentencia> ;
  <sentencia> ;
  index := index + step;
end_while;
```

Ejemplo:

```
(* Programa ST utilizando la sentencia FOR *)
(* este programa extrae los caracteres numéricos de una cadena *)

length := mlen (message);
target := ''; (* cadena vacía *)
FOR index := 1 TO length BY 1 DO
        code := ascii (message, index);
        IF (code >= 48) & (code <= 57) THEN
                target := target + char (code);
        END_IF;
END_FOR;
```

B.7.5.8 Sentencia EXIT

Nombre: **EXIT**

Significado: salir de una sentencia de iteración FOR, WHILE o REPEAT

Syntax: **EXIT;**

Operands: (ninguno)

EXIT se suele usar en una sentencia IF, dentro de un bloque FOR, WHILE o REPEAT.

Ejemplo:

```
(* Programa ST utilizando la sentencia EXIT *)
(* este programa busca un carácter en una cadena *)

length := mlen (message);
found := NO;
FOR index := 1 TO length BY 1 DO
        code := ascii (message, index);
        IF (code  = searched_char) THEN
                found := YES;
                EXIT;
        END_IF;
END_FOR;
```

B.7.6 Extensiones ST

Las siguientes funciones son extensiones del lenguaje ST:

- TSTART - TSTOP: control de temporizador

Se dispone de las siguientes sentencias y funciones para controlar la ejecución de programas hijos SFC. Pueden utilizarse dentro de bloques ACTION(): ... END_ACTION; en pasos SFC.

- GSTART — inicia un programa SFC
- GKILL — finaliza un programa SFC
- GFREEZE — congela un programa SFC
- GRST — reinicia un programa SFC congelado
- GSTATUS — obtiene el estado actual de un programa SFC

Advertencia: Estas funciones no aparecen en la norma IEC 1131-3.

GSTART y GKILL tienen una sencilla equivalencia en la siguiente sintaxis para un paso SFC:

```
child_name(S); (* equivalente a GSTART(child_name); *)
child_name(R); (* equivalente a GKILL(child_name); *)
```

Se puede emplear los siguientes campos para acceder al estado de un paso SFC:

GSnnn.x — valor booleano que representa la actividad del paso
GSnnn.t — tiempo transcurrido desde la última activación del paso
("**nnn**" es el número de referencia del paso SFC)

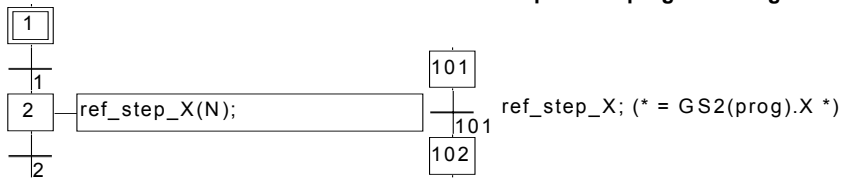
También es posible verificar la actividad de un paso que haya sido declarado en otro programa SFC, utilizando la siguiente sintaxis:

GSnnn(progname).x

Advertencia: Esta manera de referenciar un paso de otro programa, utilizando la expresada sintaxis, no aparece en la norma IEC 1131-3. Una forma sencilla de lograr el mismo objetivo, pero respetando las normas IEC, es la de declarar en el diccionario una variable booleana global que represente la actividad del paso que va a ser verificada (por ejemplo,

ref_paso_X)., Después, se inserta en el paso la variable con el calificador N (ref_paso_X(N);). Posteriormente, se utiliza la variable en el programa en el cual se desea verificar la actividad del paso.

Programa Prog **el otro programa que necesita conocer la actividad del paso del programa Prog**



B.7.6.1 Sentencia TSTART

Nombre: **TSTART**

Significado: inicia el conteo de una variable temporizador.
el comando TSTART no modifica el valor del temporizador; es decir, el conteo comienza a partir del valor actual del temporizador.

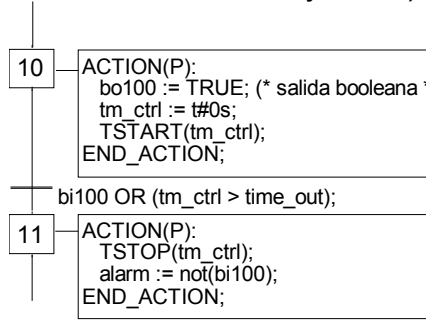
Sintaxis: **TSTART (<variable_temporizador>);**

Operandos: cualquier variable temporizador inactiva

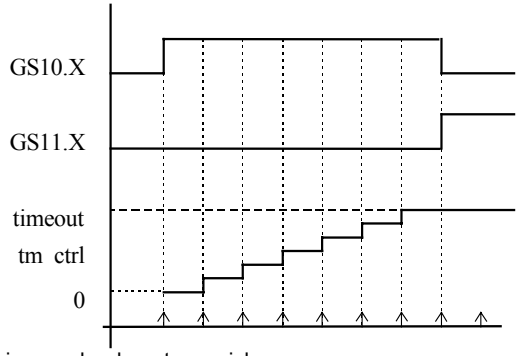
Valor de retorno: (ninguno)

Ejemplo:

(* Programa SFC utilizando las sentencias TSTART y TSTOP *)



Si bi100, el diagrama de tiempo es siempre FALSO:



El reloj mantiene el mismo valor durante un ciclo.

B.7.6.2 Sentencia TSTOP

Nombre: **TSTOP**

Significado: finaliza la actualización de una variable de temporizador
El comando TSTOP no modifica el valor de temporizador

Sintaxis: **TSTOP (<variable_temporizador>);**

Operandos: cualquier variable temporizador activa

Valor de retorno: (ninguno)

Ejemplo: Véase TSTART (la función se describe en el apartado anterior)

B.7.6.3 Sentencia GSTART

Nombre: **GSTART**

Significado: inicia un programa hijo SFC mediante la colocación de una marca en cada uno de sus pasos iniciales

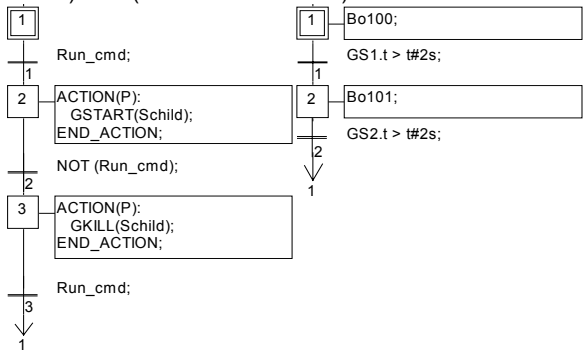
Sintaxis: **GSTART (<programa_hijo>);**

Operandos: el programa SFC especificado tiene que ser un hijo de aquel que contiene la sentencia

Valor de retorno: (ninguno)

La sentencia GSTART no arranca los hijos del programa hijo de forma automática.
Nota: Dado que GSTART no está incluida en la norma IEC 1131-3, existe la alternativa de usar el calificador S con la siguiente sintaxis para arrancar un programa hijo SFC:
Nombre_hijo(S);

Ejemplo: Uso de GSTART y GKILL



B.7.6.4 Sentencia GKILL

Nombre: **GKILL**

Significado: finaliza un programa hijo SFC mediante la eliminación de las marcas que en ese momento se encuentren en sus pasos

Sintaxis: **GKILL (<programa_hijo>);**

Operandos: el programa SFC especificado tiene que ser un hijo de aquel que contiene la sentencia

Valor de retorno: (ninguno)

Se finalizan los hijos del programa hijo de forma automática con el programa especificado.
Nota: Dado que GKILL no está incluida en la norma IEC 1131-3, existe la alternativa de usar el calificador R con la siguiente sintaxis para finalizar un programa hijo SFC:
Nombre_hijo(R);

Ejemplo: Véase GSTART (la función se describe en el apartado anterior)

B.7.6.5 Sentencia GFREEZE

Nombre: **GFREEZE**

Significado: elimina todas las marcas de un programa hijo SFC y almacena sus posiciones para que el programa pueda ser reiniciado por la sentencia GRST.

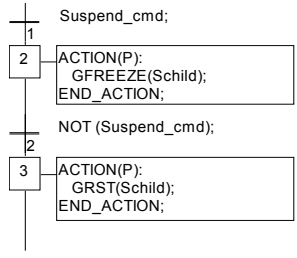
Sintaxis: **GFREEZE (<programa_hijo>);**

Operandos: el programa SFC especificado tiene que ser un hijo de aquel que contiene la sentencia

Valor de retorno: (ninguno)

Se congelan los hijos del programa hijo de forma automática junto con el programa especificado.
Nota: GFREEZE no se encuentra en la norma IEC 1131-3.

Ejemplo:



B.7.6.6 Sentencia GRST

Nombre: **GRST**

Significado: reinicia un programa hijo SFC que ha sido congelado por la sentencia GFREEZE: se restauran todas las marcas que fueron eliminadas por GFREEZE.

Sintaxis: **GRST (<programa_hijo>);**

Operandos: el programa SFC especificado tiene que ser un hijo de aquel que contiene la sentencia

Valor de retorno: (ninguno)

Se reinician los hijos del programa hijo de forma automática con la sentencia GRST.
Nota: GRST no se encuentra en la norma IEC 1131-3.

Example: Véase GFREEZE (la función se describe en el apartado anterior)

B.7.6.7 Sentencia GSTATUS

Nombre: **GSTATUS**

Significado: obtiene el estado actual de un programa SFC

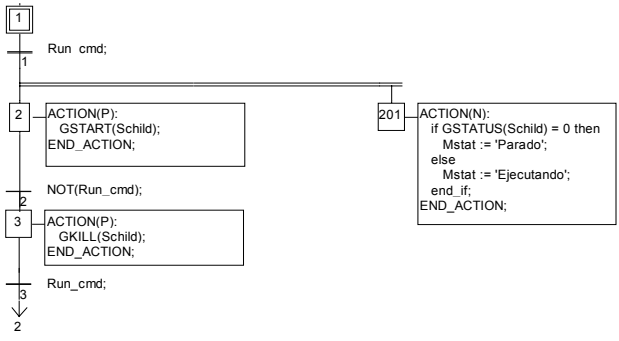
Sintaxis: **<variable_analógica> := GSTATUS (<programa_hijo>);**

Operandos: el programa SFC especificado tiene que ser un hijo de aquel que contiene la sentencia

Valor de retorno: 0 = el programa está inactivo (finalizado)
1 = el programa está activo (iniciado)
2 = el programa está congelado

Nota: GSTATUS no se encuentra en la norma IEC 1131-3.

Ejemplo:



B.8 Lenguaje IL

Lista de Instrucciones, o **IL**, es un lenguaje de bajo nivel. Las instrucciones siempre están relacionadas con el **resultado actual** (o **registro IL**). El operador indica la operación que debe llevarse a cabo entre el valor actual y el operando. El resultado de la operación se almacena de nuevo en el resultado actual.

B.8.1 Sintaxis principal IL

Un programa IL es una lista de **instrucciones**. Cada instrucción tiene que comenzar en una línea nueva y contener un **operador**, complementado con **modificadores** opcionales. Adicionalmente, y si fuera necesario para la operación específica, tiene que contener uno o más **operandos**, separados por comas (**','**). Una **etiqueta** seguida por dos puntos (**':'**) puede preceder a la instrucción. Si se añade un **comentario** a la instrucción, debe ser el último componente de la línea. Los comentarios siempre comienzan por **'(*'** y concluyen por **'*)'**. Se puede introducir líneas vacías entre instrucciones. Pueden colocarse los comentarios en líneas vacías. A continuación se muestran algunos ejemplos de líneas de instrucciones:

Etiqueta	Operador	Operando	Comentarios
Inicio:	LD	IX1	(* pulsar interruptor *)
	ANDN	MX5	(* comando no prohibido *)
	ST	QX2	(* arrancar motor *)

B.8.1.1 Etiquetas

Una **etiqueta** seguida por dos puntos (**':'**) puede preceder a la instrucción. Puede situarse una etiqueta en una línea vacía. Las etiquetas se utilizan como operandos para algunas operaciones tales como saltos. La denominación de etiquetas debe cumplir con las siguientes normas:

- el nombre no puede superar los **16** caracteres
- el primer carácter tiene que ser una **letra**
- los siguientes caracteres tiene que ser **letras**, **dígitos** o el carácter '**_**'

No se puede utilizar el mismo nombre para más de una etiqueta en el mismo programa IL. Una etiqueta puede tener el mismo nombre que una variable.

B.8.1.2 Modificadores de operadores

Se dispone de los modificadores de operadores que aparecen a continuación. El carácter modificador tiene que completar el nombre del operador, sin dejar espacios en blanco entre ellos:

N negación booleana del operando
(operación demorada
C operación condicional

El modificador **'N'** indica la negación booleana del operando. Por ejemplo, la instrucción **ORN IX12** se interpreta como: **resultado := resultado OR NOT (IX12)**.

El modificador de paréntesis **'('** indica que la evaluación de la instrucción debe demorarse hasta que se llegue al operador de paréntesis **')'** de cierre.

El modificador **'C'** indica que la instrucción asociada sólo debe ejecutarse si el resultado actual muestra el valor booleano VERDADERO (diferente a 0, en el caso de valores no booleanos). El modificador **'C'** puede combinarse con el modificador **'N'** para indicar que la instrucción sólo debe ejecutarse si el resultado actual muestra el valor booleano FALSO (o 0, en el caso de valores no booleanos).

B.8.1.3 Operaciones demoradas

Al existir un solo registro IL (resultado actual), algunas operaciones tienen que ser demoradas para que la orden de ejecución o las instrucciones puedan ser cambiadas. Se emplean paréntesis para indicar las operaciones demoradas:

'('	es un modificador	Indica la operación que va a ser demorada
')'	es un operador	Ejecuta la operación demorada

El modificador de paréntesis de apertura **'('** indica que se tiene que demorar la evaluación de la instrucción hasta que se llegue al operador de paréntesis **')'** de cierre. Por ejemplo, la siguiente secuencia:

```
AND(    IX12
OR      IX35
)
```

se interpreta como:

result := result AND (IX12 OR IX35)

B.8.2 Operadores IL

La siguiente tabla resume los operadores estándares del lenguaje IL:

Operador	Modificadores	Operando	Descripción
LD	N	Variable, constante	Carga operando
ST	N	Variable	Almacena resultado actual
S		variable BOO	Pone a VERDADERO
R		variable BOO	Repone a FALSO
AND	N (	BOO	'Y' booleana
&	N (	BOO	'Y' booleana
OR	N (	BOO	'O' booleana
XOR	N (	BOO	'O' exclusiva
ADD	(	Variable, constante	Suma
SUB	(	Variable, constante	Resta
MUL	(	Variable, constante	Multiplicación
DIV	(	Variable, constante	División
GT	(	Variable, constante	Prueba: >
GE	(	Variable, constante	Prueba: >=
EQ	(	Variable, constante	Prueba: =
LE	(	Variable, constante	Prueba: <=
LT	(	Variable, constante	Prueba: <
NE	(	Variable, constante	Prueba: <>
CAL	C N	Nombre instancia bloque	Invoca bloque de función
JMP	C N	función	Salta a etiqueta
RET	C N	Etiqueta	Retorna de subprograma
)			Ejecuta operación demorada

En la siguiente sección, sólo se describen operadores que son específicos del lenguaje IL. Pueden encontrarse otros operadores estándares en la sección titulada, "operadores, bloques de función y funciones estándares".

B.8.2.1 Operador LD

Operación carga un valor en el resultado actual

Modificadores permitidos N

Operando expresión constante
Variable interna, de entrada o de salida

Ejemplo:

```
                    (* EJEMPLOS DE OPERACIONES LD *)
LDex:       LD      FALSO       (* resultado := FALSO constante booleana *)
            LD      VERDADERO   (* resultado := VERDADERO constante booleana *)
```

```
LD      123          (*resultado:= constante entero *)
LD      123.1        (*resultado:= constante real *)
LD      t#3ms        (*resultado:= constante de tiempo *)
LD      boo_var1     (*resultado:= variable booleana *)
LD      ana_var1     (*resultado:= variable analógica *)
LD      tmr_var1     (*resultado:= variable de reloj *)
LDN     boo_var2     (*resultado:= NOT ( variable booleana ) *)
```

B.8.2.2 Operador ST

Operación almacena el resultado actual en una variable
Esta operación no modifica el resultado actual

Modificadores permitidos N

Operando variable interna o de salida

Ejemplo:

```
                (* EJEMPLOS DE OPERACIONES ST *)
STboo:  LD      FALSO
        ST      boo_var1    (* boo_var1 := FALSO*)
        STN     boo_var2    (* boo_var2 := VERDADERO*)
STana:  LD      123
        ST      ana_var1    (* ana_var1 := 123 *)
STtmr:  LD      t#12s
        ST      tmr_var1    (* tmr_var1 := t#12s *)
```

B.8.2.3 Operador S

Operación: almacena el valor booleano VERDADERO en una variable booleana, si el resultado actual tiene el valor booleano VERDADERO. No se procesan operaciones si el resultado actual es FALSO. Esta operación no modifica el resultado actual.

Modificadores permitidos: (ninguno)

Operando: variable booleana interna o de salida

Ejemplo:

```
                (* EJEMPLOS DE OPERACIONES S *)
SETex:  LD      VERDADERO       (* resultado actual := VERDADERO *)
        S       var_boo1        (*var_boo1 := VERDADERO *)
```

```
                                    (* resultado actual no está modificado *)
        LD      FALSO               (*resultado actual := FALSO *)
        S       var_boo1            (* sin acciones – var_boo1 sin cambios *)
```

B.8.2.4 Operador R

Operación almacena el valor booleano FALSO en una variable booleana, si el resultado actual muestra el valor booleano VERDADERO. No se procesan operaciones si el resultado actual es FALSO. Esta operación no modifica el resultado actual.

Modificadores permitidos (ninguno)

Operando variable booleana interna o de salida

Ejemplo:

```
                    (* EJEMPLOS DE OPERACIONES R *)
RESETex:    LD      VERDADERO       (* resultado actual := VERDADERO *)
            R       boo_var1        (* boo_var1 := FALSO *)
                                    (* resultado actual no se modifica *)
            ST      boo_var2        (* boo_var2 := VERDADERO *)
            LD      FALSO           (* resultado actual := FALSO *)
            R       boo_var1        (* no se hace nada - boo_var1 no cambiado
*)
```

B.8.2.5 Operador JMP

Operación salta a la etiqueta especificada

Modificadores permitidos C N

Operando etiqueta definida en el mismo programa IL

Ejemplo:

```
(* el siguiente ejemplo verifica el valor de un selector analógico (0 ó 1 ó 2) *)
(* para fijar una de 3 salidas booleanas. Verificación "es igual a 0" se realiza con *)
(* el operador JMPC *)

JMPex:      LD      selector        (* selector es 0 ó 1 ó 2 *)
            BOO                     (* conversión a booleano *)
            JMPC    test1           (* si selector = 0 entonces *)
```

```
            LD      VERDADERO
            ST      bo0             (* bo0 := VERDADERO *)
            JMP     JMPend          (* fin de programa *)
test1:      LD      selector
            SUB     1               (* decrementar selector: ahora es 0 ó 1 *)
            BOO                     (* conversión a booleano *)
            JMPC    test2           (* si selector = 0 entonces *)
            LD      VERDADERO
            ST      bo1             (* bo1 := VERDADERO *)
            JMP     JMPend          (* fin de programa *)
test2:      LD      VERDADERO       (* última posibilidad *)
            ST      bo2             (* bo2 := VERDADERO *)
JMPend:                             (* fin de programa IL *)
```

B.8.2.6 Operador RET

Operación finaliza la lista actual de instrucciones. Si la secuencia IL es un subprograma, se devuelve el resultado actual al programa invocante

Modificadores permitidos C N

Operando (ninguno)

Ejemplo:

```
(* el siguiente ejemplo verifica el valor de un selector analógico (0 ó 1 ó 2) *)
(* para fijar una de 3 salidas booleanas. Verificación "es igual a 0" se realiza con *)
(* el operador JMPC *)

JMPex:      LD      selector        (* selector es 0 ó 1 ó 2 *)
            BOO                     (* conversión a booleano *)
            JMPC    test1           (* si selector = 0 entonces *)
            LD      VERDADERO
            ST      bo0             (* bo0 := VERDADERO *)
            RET                     (* fin - retorno 0 *)
                                    (* decrementar selector *)
test1:      LD      selector
            SUB     1               (* selector es ahora 0 ó 1 *)
            BOO                     (* conversión a booleano *)
            JMPC    test2           (* si selector = 0 entonces *)
            LD      VERDADERO
            ST      bo1             (* bo1 := VERDADERO *)
            LD      1               (* cargar valor selector real *)
            RET                     (* fin - retorno 1 *)
                                    (* última posibilidad *)
test2:      RETNC                   (* retornar si el selector tiene *)
                                    (* un valor inválido *)
```

```
LD      VERDADERO
ST      bo2             (* bo2 := VERDADERO *)
LD      2               (* cargar valor selector real *)
                        (* fin - retorno 2 *)
```

B.8.2.7 Operador ")"

Operación ejecuta una operación demorada. La operación demorada fue notificada por un '('

Modificadores permitidos (ninguno)

Operando (ninguno)

Ejemplo:

```
(* El siguiente programa intercala operaciones demoradas: *)
(* res := a1 + (a2 * (a3 - a4) * a5) + a6; *)

Delayed:    LD      a1      (* resultado := a1; *)
            ADD(    a2      (* ADD retardado - resultado := a2; *)
            MUL(    a3      (* MUL retardado - resultado := a3; *)
            SUB     a4      (* resultado := a3 - a4; *)
            )               (* ejecutar MUL retardado - resultado := a2 * (a3-a4);
*)
            MUL     a5      (* resultado := a2 * (a3 - a4) * a5; *)
            )               (* ejecutar ADD retardado *)
                            (* resultado := a1 + (a2 * (a3 - a4) * a5); *)
            ADD     a6      (* resultado := a1 + (a2 * (a3 - a4) * a5) + a6; *)
            ST      res     (* almacenar resultado actual en variable res *)
```

B.8.2.8 Invocación de subprogramas o funciones

Un subprograma o una función (escritos en cualquiera de los lenguajes IL, ST, LD, FBD o "C") se invoca desde el lenguaje IL, utilizando su nombre como operador.

Operación ejecuta un subprograma o una función – el valor devuelto por el subprograma o la función se almacena en el resultado actual IL.

Modificadores permitidos (ninguno)

Operand El primer parámetro de invocación debe estar almacenado en el resultado actual antes de producirse la llamada. Los siguientes se expresan en el campo del operando, separados por comas.

Ejemplo:

(* Programa de invocación : convierte un valor analógico en un valor de tiempo *)

```
Main:       LD          bi0
            SUBPRO      bi1,bi2     (* invoca subprograma para conseguir valor
analógico *)
            ST          resultado   (* resultado := valor retornado por subprograma *)
            GT          vmax        (* test de rebasamiento de valor *)
            RETC                    (* retorno si hay rebasamiento *)
            LD          resultado
            MUL         1000        (* convierte segundos en milisegundos *)
            TMR                     (* convierte a temporizador *)
            ST          tmval       (* almacena valor convertido en un temporizador
*)

(* Invocado subprograma llamado 'SUBPRO' : evalúa el valor analógico *)
(* dado como valor binario en tres entradas booleanas: in0, in1, in2 son los tres parámetros
booleanos de entrada del subprograma *)
            LD          in2
            ANA                     (* resultado = ana (in2); *)
            MUL         2           (* resultado := 2*ana (in2); *)
            ST          temporary   (* temporary := resultado *)
            LD          in1
            ANA
            ADD         temporary   (* resultado := 2*ana (in2) + ana (in1); *)
            MUL         2           (* resultado := 4*ana (in2) + 2*ana (in1); *)
            ST          temporary   (* temporary := resultado *)
            LD          in0
            ANA
            ADD         temporary   (* resultado := 4*ana (in2) + 2*ana (in1)+ana (in0); *)
            ST          SUBPRO      (* retorna resultado actual a programa invocante *)
```

B.8.2.9 Invocación de bloques de función: operador CAL

Operación invoca un bloque de función

Modificadores permitidos C N

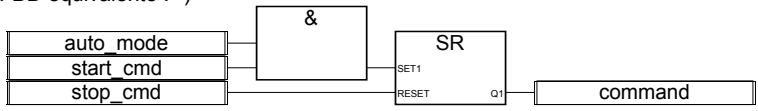
Operando Nombre del bloque de función específico.
Tienen que asignarse los parámetros de entrada de los bloques antes de la invocación, utilizando la secuencia de operaciones LD/ST.
Se utilizan parámetros de salida si se conocen.

Ejemplo 1:

(* Invocación de bloque de función SR : SR1 es una instancia de SR *)

```
LD      auto_mode
Y       start_cmd
ST      SR1.set1
LD      stop_cmd
ST      SR1.reset
CAL     SR1
LD      SR1.Q1
ST      command
```

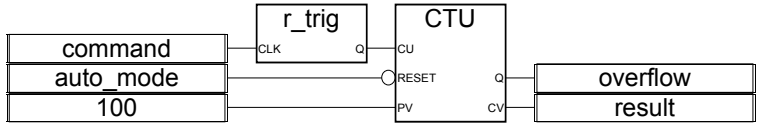
(* FBD equivalente : *)



Ejemplo 2
(* Se supone que R_TRIG1 es una instancia del bloque R_TRIG y CTU1 es una instancia del bloque CTU *)

```
LD      command
ST      R_TRIG1.clk
CAL     R_TRIG1
LD      R_TRIG1.Q
ST      CTU1.cu
LDN     auto_mode
ST      CTU1.reset
LD      100
ST      CTU1.pv
CAL     CTU1
LD      CTU1.Q
ST      overflow
LD      CTU1.cv
ST      result
```

(* FBD equivalente: *)



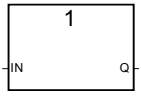
B.9 Operadores, bloques de función y funciones estándares

B.9.1 Operadores estándares

A continuación se exponen los operadores estándares de los lenguajes IEC:

- Manejo de datos.....Asignación, Negación analógica
- Operaciones booleanas .....AND booleana
  OR booleana
  OR exclusiva booleana
- Operaciones aritméticas.....Suma
  Resta
  Multiplicación
  División
- Operaciones lógicas.....Máscara analógica bit-a-bit AND
  Máscara analógica bit-a-bit OR
  Máscara analógica bit-a-bit OR exclusiva
  Negación bit-a-bit
- Pruebas comparativas.....Menor que
  Menor o igual que
  Mayor que
  Mayor o igual que
  Es igual a
  No es igual a
- Conversión de datos .....Convertir a Booleano
  Convertir a Analógico Entero
  Convertir a Analógico Real
  Convertir a Temporizador
  Convertir a Mensaje
- Otros .....Concatenación de mensajes
  Acceso al sistema
  Operar canal E/S

1 gain



Argumentos:

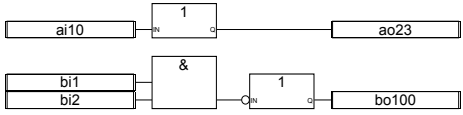
IN	cualquier tipo
Q	cualquier tipo

Descripción:

Asignación de una variable a otra

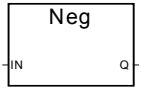
Este bloque resulta muy útil para crear un vínculo directo entre una entrada de diagrama y una salida de diagrama. También se puede utilizar (con una línea de negación booleana) para invertir el estado de una línea que esté conectada a una salida de diagrama.

(* Ejemplo FBD con Bloques de Asignación *)




```
(* Equivalencia ST: *)
ao23 := ai10;
bo100 := NOT (bi1 Y bi2);

(* Equivalencia IL: *)
LD      ai10
ST      ao23
LD      bi1
Y       bi2
STN     bo100
```

NEG



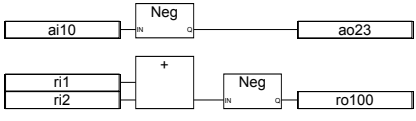
Argumentos:

IN	INT-REAL	entrada y salida tienen que tener el mismo formato
Q	INT-REAL	

Descripción:

Asignación de la negación de una variable.

(* Ejemplo FBD con Bloques de Negación *)



(* Equivalencia ST: *)

```
ao23 := - (ai10);
ro100 := - (ri1 + ri2);

(* Equivalencia IL: *)
LD          ai10
MUL         -1
ST          ao23
LD          ri1
ADD         ri2
MUL         -1.0
ST          ro100
```

& AND

Nota: Para este operador, el número de entradas puede extenderse a más de dos.

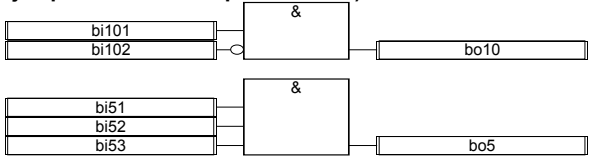
Argumentos:

(entradas)	BOOLEANO	
salida	BOOLEANO	AND booleana de los términos de entrada

Descripción:

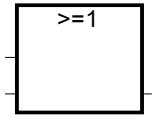
AND booleana entre dos o más términos.

(* Ejemplo FBD con Bloques "AND" *)




```
(* Equivalencia ST: *)
bo10 := bi101 AND NOT (bi102);
bo5 := (bi51 AND bi52) Y bi53;

(* Equivalencia IL *)
            LD      bi101   (* resultado actual := bi101 *)
            ANDN    bi102   (* resultado actual := bi101 AND not(bi102) *)
            ST      bo10    (* bo10 := resultado actual *)
            LD      bi51    (* resultado actual := bi51;
            &       bi52    (* resultado actual := bi51 AND bi52 *)
            &       bi53    (* resultado actual := (bi51 AND bi52) AND bi53 *)
            ST      bo5     (* bo5 := resultado actual *)
```

>=1 OR



Nota: Para este operador, el número de entradas puede extenderse a más de dos.

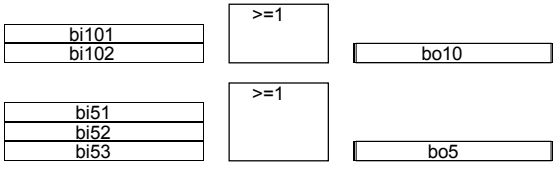
Argumentos:

(entradas)	BOOLEANO	
salida	BOOLEANO	OR booleana de los términos de entrada

Descripción:

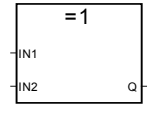
OR booleana de dos o más términos.

(* Ejemplo FBD con Bloques "OR" *)




```
(* Equivalencia ST: *)
bo10 := bi101 OR NOT (bi102);
bo5 := (bi51 OR bi52) OR bi53;

(* Equivalencia IL: *)
LD      bi101
ORN     bi102
ST      bo10
LD      bi51
OR      bi52
OR      bi53
ST      bo5
```

=1 XOR



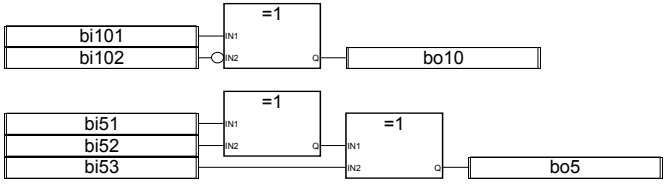
Argumentos:

IN1	BOOLEANO	
IN2	BOOLEANO	
Q	BOOLEANO	OR exclusiva booleana de los 2 términos de entrada

Descripción:

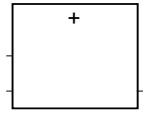
OR exclusiva booleana entre dos términos.

(* Ejemplo FBD con Bloques "XOR" *)




```
(* Equivalencia ST: *)
bo10 := bi101 XOR NOT (bi102);
bo5 := (bi51 XOR bi52) XOR bi53;

(* Equivalencia IL: *)
LD          bi101
XORN        bi102
ST          bo10
LD          bi51
XOR         bi52
XOR         bi53
ST          bo5
```

+ (Suma)



Nota: Para este operador, el número de entradas puede extenderse a más de dos.

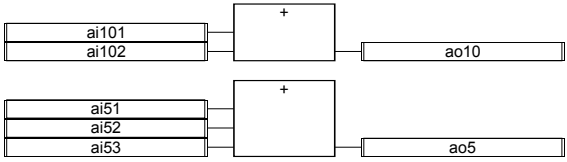
Argumentos:

(entradas)	INT-REAL	puede ser ENTERO o REAL (todas las entradas tienen que tener el mismo formato)
salida	INT-REAL	suma con signo de los términos de entrada

Descripción:

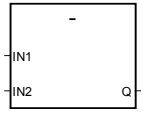
Adición de dos o más variables analógicas.

(* Ejemplo FBD con Bloques de Suma *)




```
(* Equivalencia ST: *)
ao10 := ai101 + ai102;
ao5 := (ai51 + ai52) + ai53;

(* Equivalencia IL: *)
LD          ai101
ADD         ai102
ST          ao10
LD          ai51
ADD         ai52
ADD         ai53
ST          ao5
```

- (Resta)



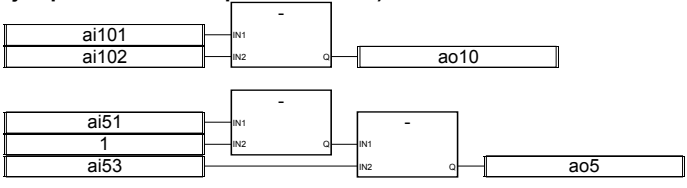
Argumentos:

IN1	INT-REAL	puede ser ENTERO o REAL
IN2	INT-REAL	(IN1 y IN2 tienen que tener el mismo formato)
Q	INT-REAL	resta (primera - segunda)

Descripción:

Resta de dos variables analógicas (primera - segunda).

(* Ejemplo FBD con Bloques de Resta *)




```
 (* Equivalencia ST: *)
ao10 := ai101 - ai102;
ao5 := (ai51 - 1) - ai53;

(* Equivalencia IL: *)
LD          ai101
```

```
SUB     ai102
ST      ao10
LD      ai51
SUB     1
SUB     ai53
ST      ao5
```

*** (Multiplicación)**

Nota: Para este operador, el número de entradas puede extenderse a más de dos.

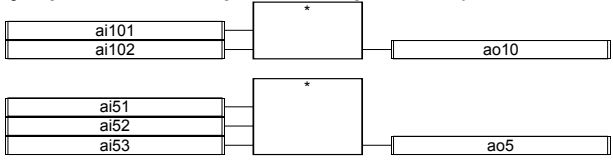
Argumentos:

(entradas)	INT-REAL	puede ser ENTERO o REAL (todas las entradas tienen que tener el mismo formato)
salida	INT-REAL	multiplicación con signo de los términos de entrada

Descripción:

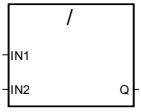
Multiplicación de dos o más variables analógicas.

(* Ejemplo FBD con Bloques de Multiplicación *)




```
(* Equivalencia ST *)
ao10 := ai101 * ai102;
ao5 := (ai51 * ai52) * ai53;

(* Equivalencia IL: *)
LD      ai101
MUL     ai102
ST      ao10
LD      ai51
MUL     ai52
MUL     ai53
ST      ao5
```

/ (División)



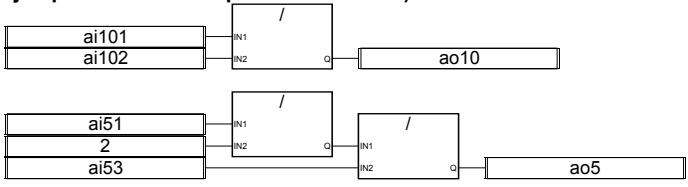
Argumentos:

IN1	INT-REAL	puede ser ENTERO o REAL (operando)
IN2	INT-REAL	valor analógico distinto de cero (divisor) (IN1 y IN2 tienen que tener el mismo formato)
Q	INT-REAL	valor entero con signo o división real de IN1 por IN2

Descripción:

División de dos variables analógicas (la primera dividida por la segunda).

(* Ejemplo FBD con Bloques de División *)




```
(* Equivalencia ST: *)
ao10 := ai101 / ai102;
ao5 := (ai5 / 2) / ai53;

(* Equivalencia IL: *)
LD      ai101
DIV     ai102
ST      ao10
LD      ai51
DIV     2
DIV     ai53
ST      ao5
```

AND_MASK

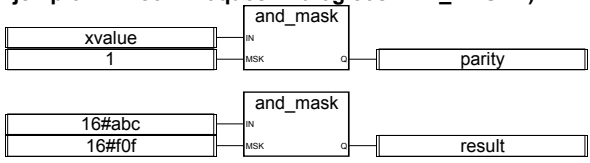
Argumentos:

IN	INT	tiene que tener formato entero
MSK	INT	tiene que tener formato entero
Q	INT	AND lógico bit-a-bit entre IN y MSK

Descripción:

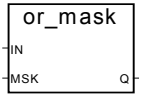
Analógico entero AND máscara bit a bit.

(* Ejemplo FBD con Bloques Analógicos AND_MASK *)




```
(* Equivalencia ST: *)
parity := AND_MASK (xvalue, 1); (* 1 si xvalue es impar *)
result := AND_MASK (16#abc, 16#f0f); (* igual a 16#a0c *)

(* Equivalencia IL: *)
LD          xvalue
AND_MASK    1
ST          parity
LD          16#abc
AND_MASK    16#f0f
ST          result
```

OR_MASK



Argumentos:

IN	INT	tiene que tener formato entero
MSK	INT	tiene que tener formato entero
Q	INT	OR lógico de bit a bit entre 'IN' y 'MSK'

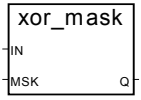
Descripción:

Analógico entero OR máscara bit a bit.

(* Ejemplo FBD con Bloques Analógicos OR_MASK *)

```
(* Equivalencia ST: *)
is_odd := OR_MASK (xvalue, 1); (* hace el valor siempre impar *)
result := OR_MASK (16#abc, 16#f0f); (* igual a 16#fbf *)

(* Equivalencia IL: *)
LD          xvalue
OR_MASK     1
ST          is_odd
LD          16#abc
OR_MASK     16#f0f
ST          result
```

XOR_MASK



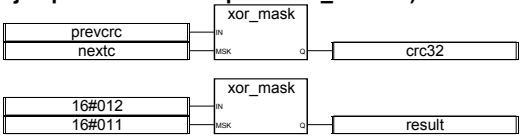
Argumentos:

IN	INT	tiene que tener formato entero
MSK	INT	tiene que tener formato entero
Q	INT	OR exclusivo lógico de bit a bit entre 'IN' y 'MSK'

Descripción:

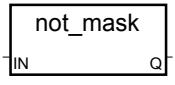
Analógico entero XOR máscara bit a bit.

(* Ejemplo FBD con Bloques XOR_MASK *)




```
(* Equivalencia ST: *)
crc32 := XOR_MASK (prevcrc, nextc);
resultado := XOR_MASK (16#012, 16#011); (* igual a 16#003 *)

(* Equivalencia IL: *)
LD          prevcrc
XOR_MASK    nextc
ST          crc32
LD          16#012
XOR_MASK    16#011
ST          resultado
```

NOT_MASK



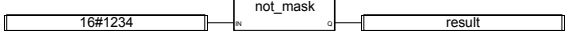
Argumentos :

IN	INT	tiene que tener formato entero
Q	INT	negación de bit a bit en 32 bits de 'IN'

Descripción:

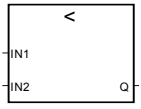
Máscara de negación bit a bit de analógico entero.

(* Ejemplo FBD con Bloques NOT_MASK *)



```
(* Equivalencia ST: *)
result := NOT_MASK (16#1234);
(* el resultado es 16#FFFF_EDCB *)

(* Equivalencia IL: *)
LD          16#1234
NOT_MASK
ST          result
```

< (Menor que)



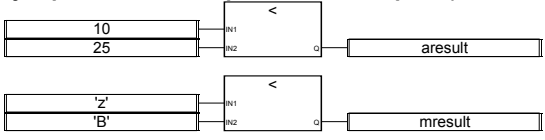
Argumentos:

IN1	INT-REAL-TMR-MSG	
IN2	INT-REAL-TMR-MSG	ambas entradas tienen que tener el mismo tipo
Q	BOOLEANO	VERDADERO si IN1 < IN2

Descripción:

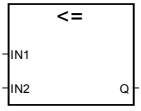
Comprueba si un valor es MENOR QUE otro valor (para analógicos, temporizadores o mensajes)

(* Ejemplo FBD con Bloques de "Menor que" *)




```
(* Equivalencia ST: *)
aresult := (10 < 25); (* aresult es VERDADERO *)
mresult := ('z' < 'B'); (* mresult es FALSO*)

(* Equivalencia IL: *)
LD          10
LT          25
ST          aresult
LD          'z'
LT          'B'
ST          mresult
```

<= (Menor o igual que)



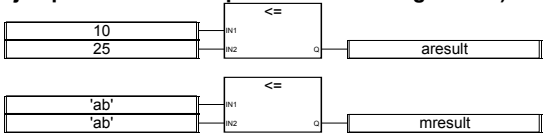
Argumentos:

IN1	INT-REAL-MSG	
IN2	INT-REAL-MSG	ambas entradas tienen que tener el mismo tipo
Q	BOOLEANO	VERDADERO si IN1 <= IN2

Descripción:

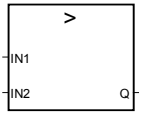
Comprueba si un valor es MENOR O IGUAL A otro valor (para analógicos, o mensajes)

(* Ejemplo FBD con Bloques de "Menor o igual a" *)




```
(* Equivalencia ST: *)
aresult := (10 <= 25); (* aresult es VERDADERO *)
mresult := ('ab' <= 'ab'); (* mresult es VERDADERO *)

(* Equivalencia IL: *)
LD      10
LE      25
ST      aresult
LD      'ab'
LE      'ab'
ST      mresult
```

> (Mayor que)



Argumentos:

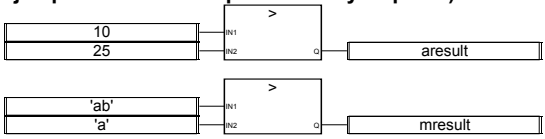
IN1	INT-REAL-TMR-MSG	
IN2	INT-REAL-TMR-MSG	ambas entradas tienen que tener el mismo tipo

Q BOOLEANO VERDADERO si IN1 > IN2

Descripción:

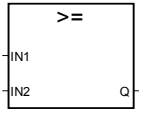
Comprueba si un valor es MAYOR QUE otro valor (para analógicos, temporizadores o mensajes)

(* Ejemplo FBD con Bloques de "Mayor que" *)




```
(* Equivalencia ST: *)
aresult := (10 > 25); (* aresult es FALSO *)
mresult := ('ab' > 'a'); (* mresult es VERDADERO *)

(* Equivalencia IL: *)
LD      10
GT      25
ST      aresult
LD      'ab'
GT      'a'
ST      mresult
```

>= (Mayor o igual que)



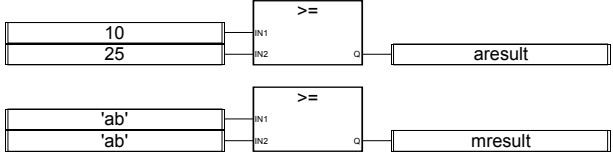
Argumentos:

IN1	INT-REAL-MSG
IN2	INT-REAL-MSG ambas entradas tienen que tener el mismo tipo
Q	BOOLEANO VERDADERO si IN1 >= IN2

Descripción:

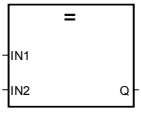
Comprueba si un valor es SUPERIOR O IGUAL A otro valor (para analógicos o mensajes)

(* Ejemplo FBD con bloques de "Mayor o igual que" *)



```
(* Equivalencia ST: *)
aresult := (10 >= 25); (* aresult es FALSO *)
mresult := ('ab' >= 'ab'); (* mresult es VERDADERO *)

(* Equivalencia IL: *)
LD          10
GE          25
ST          aresult
LD          'ab'
GE          'ab'
ST          mresult
```

= (Igual a)



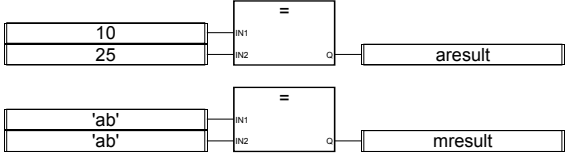
Argumentos:

IN1	INT-REAL-MSG
IN2	INT-REAL-MSG ambas entradas tienen que tener el mismo tipo
Q	BOOLEANO VERDADERO si IN1 = IN2

Descripción:

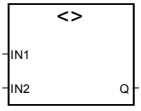
Comprueba si un valor es IGUAL A otro valor (para analógicos o mensajes)

(* Ejemplo FBD con Bloques "Igual a" *)




```
(* Equivalencia ST: *)
aresult := (10 = 25); (* aresult es FALSO *)
mresult := ('ab' = 'ab'); (* mresult es VERDADERO *)

(* Equivalencia IL: *)
LD          10
EQ          25
ST          aresult
LD          'ab'
EQ          'ab'
ST          mresult
```

<> (No es igual a)



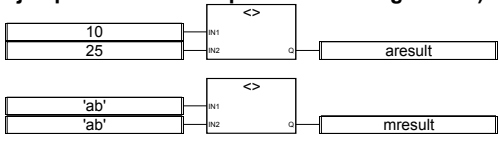
Argumentos:

IN1	INT-REAL-MSG	
IN2	INT-REAL-MSG	ambas entradas tienen que tener el mismo tipo
Q	BOOLEANO	VERDADERO si primera <> segunda

Descripción:

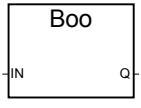
Comprueba si un valor NO ES IGUAL A otro valor (para analógicos o mensajes)

(* Ejemplo FBD con Bloques de "No es Igual a" *)




```
(* Equivalencia ST: *)
aresult := (10 <> 25); (* aresult es VERDADERO *)
mresult := ('ab' <> 'ab'); (* mresult es FALSO *)

(* Equivalencia IL: *)
LD      10
NE      25
ST      aresult
LD      'ab'
NE      'ab'
ST      mresult
```

BOO



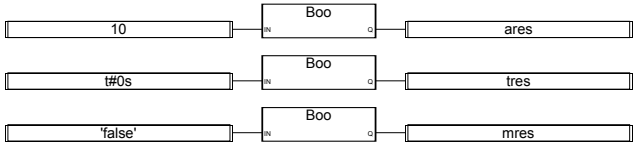
Argumentos:

IN	ANY	cualquier valor no booleano
Q	BOO	VERDADERO para valores numéricos distintos a cero FALSO para valores numéricos cero VERDADERO para mensaje 'VERDADERO' FALSO para mensaje 'FALSO'

Descripción:

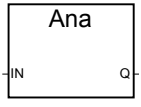
Convierte cualquier variable a una variable booleana

(* Ejemplo FBD con Bloques de “Convertir a Booleana” *)




```
(* Equivalencia ST: *)
ares := BOO (10);                (* ares es VERDADERO *)
tres := BOO (t#0s);              (* tres es FALSO *)
mres := BOO ('FALSO');           (* mres es FALSO *)

(* Equivalencia IL: *)
LD          10
BOO
ST          ares
LD          t#0s
BOO
ST          tres
LD          'FALSO'
BOO
ST          mres
```

ANA



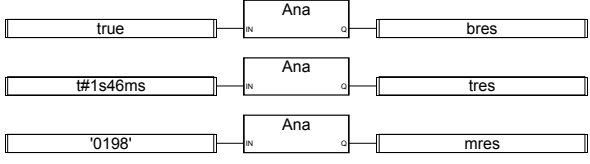
Argumentos:

IN	ANY	cualquier valor analógico no entero
Q	INT	0 si IN es FALSO / 1 si IN es VERDADERO número de milisegundos para un temporizador parte entera para analógico real número decimal representado por una cadena

Descripción:

Convierte cualquier variable a una entera

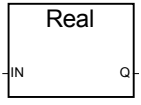
(* Ejemplo FBD con Bloques de “Convertir a Analógico” *)




```
(* Equivalencia ST: *)
bres := ANA (VERDADERO);         (* bres es 1 *)
```

```
tres := ANA (t#1s46ms);          (* tres es 1046 *)
mres := ANA ('0198');            (* mres es 198 *)

(* Equivalencia IL: *)
LD          VERDADERO
ANA
ST          bres
LD          t#1s46ms
ANA
ST          tres
LD          '0198'
ANA
ST          mres
```

REAL



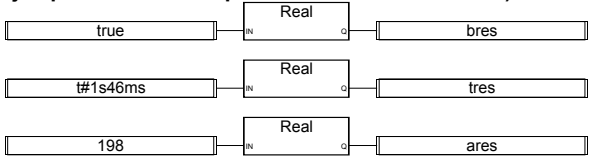
Argumentos:

IN	BOO-INT-TMR	cualquier valor analógico no real (excepto mensaje)
Q	REAL	0.0 si IN es FALSO / 1.0 si IN es VERDADERO número de milisegundos para un temporizador número equivalente para analógico entero

Descripción:

Convierte cualquier variable a una real

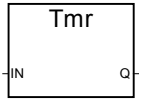
(* Ejemplo FBD con Bloques de "Convertir a Real" *)




```
(* Equivalencia ST: *)
bres := REAL (VERDADERO);        (* bres es 1.0 *)
tres := REAL (t#1s46ms);         (* tres es 1046.0 *)
ares := REAL (198);              (* ares es 198.0 *)

(* Equivalencia IL: *)
LD          VERDADERO
REAL
ST          bres
LD          t#1s46ms
REAL
```

```
ST          tres
LD          198
REAL
ST          ares
```

TMR



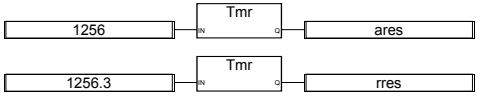
Argumentos:

IN	INT-REAL	cualquier valor excepto temporizador IN (o parte entera de IN si es real) es el número de milisegundos
Q	TIMER	valor en tiempo, representado por IN

Descripción:

Convierte cualquier variable analógica en una variable de temporizador

(* Ejemplo FBD con Bloques de "Convertir a Temporizador" *)




```
 (* Equivalencia ST: *)
ares := TMR (1256);                 (* ares := t#1s256ms *)
rres := TMR (1256.3);               (* rres := t#1s256ms *)

(* Equivalencia IL: *)
LD          1256
TMR
ST          ares
LD          1256.3
TMR
ST          rres
```

MSG

Argumentos:

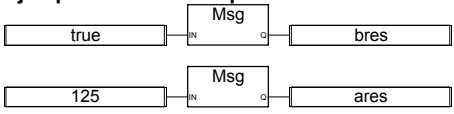
IN	BOO- INT-REA	cualquier valor excepto mensaje
Q	MSG	'falso' o 'verdadero' si IN es booleano

representación decimal si IN es analógico

Descripción:

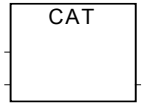
Convierte cualquier variable en una de mensaje

(* Ejemplo FBD con Bloques de “Convertir a Mensaje” *)




```
(* Equivalencia ST: *)
bres := MSG (VERDADERO); (* bres es 'VERDADERO' (VERDADERO) *)
ares := MSG (125); (* ares es '125' *)

(* Equivalencia IL: *)
LD          VERDADERO
MSG
ST          bres
LD          125
MSG
ST          ares
```

CAT



Nota: Para este operador, el número de entradas puede extenderse a más de dos.

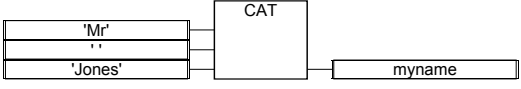
Argumentos:

(entradas)	MSG	(la suma de todas la longitudes de mensajes no debe superar la capacidad del mensaje de salida)
salida	MSG	concatenación de los mensajes de entrada

Descripción:

Concatena varios mensajes en uno solo

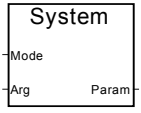
(* Ejemplo FBD con Bloques de “Concatenación de Mensajes” *)




```
(* Equivalencia ST: usar el operador + *)
myname := ('Mr' + ' ') + 'Jones';
(* significa: myname := 'Mr Jones' *)

(* Equivalencia IL: *)
```

```
LD      'Mr'
ADD     ' '
ADD     'Jones'
ST      myname
```

SYSTEM



Argumentos:

Mode	INT	identifica el parámetro de sistema y el modo de acceso
Arg	INT-TMR	valor nuevo para un acceso de “escritura”
Param	INT	valor del parámetro al cual se ha accedido

Descripción:

Acceso a los parámetros del sistema

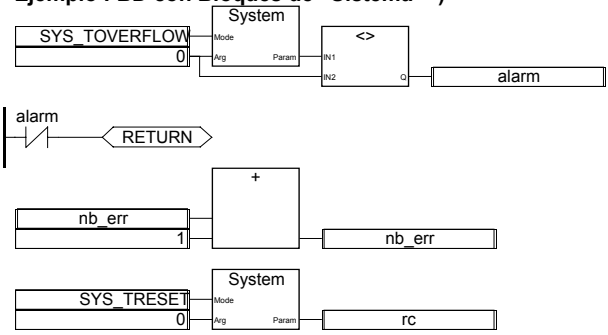
A continuación se expone una relación de los comandos (palabras clave predefinidas) disponibles para la función de SISTEMA:

Comando	Significado
SYS_TALLOWED	leer tiempo de ciclo permitido
SYS_TCURRENT	leer tiempo de ciclo actual
SYS_TMAXIMUM	leer tiempo de ciclo máximo
SYS_TOVERFLOW	leer desbordamientos de tiempo de ciclo
SYS_TRESET	restaurar contadores de tiempo
SYS_TWRITE	cambiar tiempos de ciclo
SYS_ERR_TEST	comprobar errores de tiempo de proceso
SYS_ERR_READ	leer error de tiempo de proceso más antiguo

A continuación se indican los argumentos esperados para las funciones predefinidas de la función de SISTEMA:

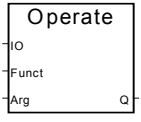
Comando	Argumento	Valor de retorno
SYS_TALLOWED	0	tiempo de ciclo permitido
SYS_TCURRENT	0	tiempo de ciclo actual
SYS_TMAXIMUM	0	tiempo máximo detectado
SYS_TOVERFLOW	0	número de desbordamientos de tiempo
SYS_TRESET	0	0
SYS_TWRITE	Nuevo tiempo de ciclo permitido	tiempo escrito
SYS_ERR_TEST	0	0 si no se detectan errores
SYS_ERR_READ	0	código de error más antiguo

(* Ejemplo FBD con Bloques de "Sistema" *)




```
(* Equivalencia ST: *)
alarm := (SYSTEM (SYS_TOVERFLOW, 0) <> 0);
If (alarm) Then
        nb_err := nb_err + 1;
        rc := SYSTEM (SYS_TRESET, 0);
End_If
```

OPERATE



Argumentos:

IO	ANY	variable de entrada o salida
Funct	INT	acción a operar
Arg	INT	argumento para acción de E/S
Q	INT	comprobación retorno

Descripción:

Acceso a un canal de E/S

El significado de los argumentos OPERATE depende de la implementación de la interfaz de E/S. Para conocer en mayor detalle las capacidades OPERATE, véanse el manual del *hardware* o las correspondientes notas técnicas de la tarjeta de E/S.

B.9.2 Bloques de función estándares

Estos son bloques de función estándares soportados por el sistema ISaGRAF. Dichos bloques están predefinidos y no tienen que ser declarados en la librería.

● Booleanos	SR	Establecer biestable dominante
	RS	Restablecer biestable dominante
	R_Trig	Detección de flanco de subida
	F_Trig	Detección de flanco de bajada
	SEMA	Semáforo
● Contaje	CTU	Contador ascendente
	CTD	Contador descendente
	CTUD	Contador ascendente-descendente
● Temporizadores	TON	Temporización con retardo en conexión
	TOF	Temporización con retardo en desconexión
	TP	Temporización de impulsos
● Analógicos enteros	CMP	Bloque de función de comparación total
	StackInt	Grupo de valores analógicos enteros
● Analógicos reales	AVERAGE	Promedio dinámico basado en N muestras
	HYSTER	Histéresis booleana por diferencias entre reales
	LIM_ALRM	Límites superior/inferior de alarma con histéresis
	INTEGRAL	Integración a lo largo del tiempo
	DERIVATE	Diferenciación según tiempos
● Generación de señales	BLINK	Señal booleana parpadeante
	SIG_GEN	Generador de señales

Nota: Cuando se crean nuevos bloques “C”, pueden ser invocados desde el lenguaje FBD.

SR

Argumentos:

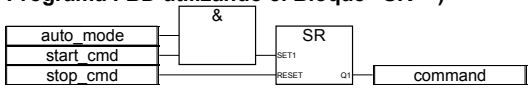
SET1	BOO	si VERDADERO, fija Q1 en VERDADERO (dominante)
RESET	BOO	si VERDADERO, restablece Q1 en FALSO
Q1	BOO	estado booleano de memoria

Descripción:

Establece biestable dominante: Véase Tabla de VERDAD a continuación:

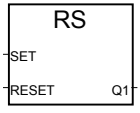
SET1	RESET	Q1	resultado Q1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

(* Programa FBD utilizando el Bloque “SR” *)




```
(* Equivalencia ST: se supone que SR1 es una instancia del bloque SR *)
SR1((auto_mode & start_cmd), stop_cmd);
command := SR1.Q1;

(* Equivalencia IL: *)
LD          auto_mode
Y           start_cmd
ST          SR1.set1
LD          stop_cmd
ST          SR1.reset
CAL         SR1
LD          SR1.Q1
ST          command
```

RS



Argumentos:

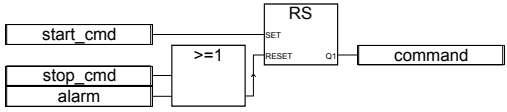
SET	BOO	si VERDADERO, fija Q1 en VERDADERO (dominante)
RESET	BOO	si VERDADERO, restablece Q1 en FALSO (dominante)
Q1	BOO	estado booleano de memoria

Descripción:

Restablecer biestable dominante: Véase Tabla de VERDAD a continuación:

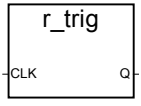
SET	RESET	Q1	resultado Q1
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

(* Programa FBD utilizando el Bloque "RS" *)




```
(* Equivalencia ST: se supone que SR1 es una instancia del bloque RS *)
RS1(start_cmd, (stop_cmd OR alarm));
command := RS1.Q1;

(* Equivalencia IL: *)
LD          start_cmd
ST          RS1.set
LD          stop_cmd
OR          alarm
ST          RS1.reset1
CAL         RS1
LD          RS1.Q1
ST          command
```

R_TRIG



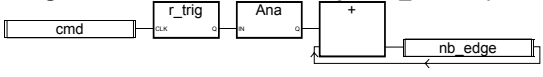
Argumentos:

CLK	BOO	cualquier variable booleana
Q	BOO	VERDADERO cuando CLK pasa de FALSO a VERDADERO FALSO en todos los demás casos

Descripción:

Detecta el flanco de subida de una variable booleana

(* Programa FBD utilizando el Bloque "R_TRIG" *)




```
(* Equivalencia ST: se supone que R_TRIG1 es una instancia del bloque R_TRIG *)
R_TRIG1(cmd);
nb_edge := ANA(R_TRIG1.Q) + nb_edge;

(* Equivalencia IL: *)
LD          cmd
ST          R_TRIG1.clk
CAL         R_TRIG1
LD          R_TRIG1.Q
ANA
ADD         nb_edge
```

ST nb_edge

F_TRIG

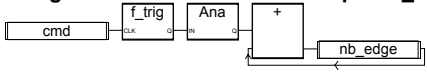
Argumentos:

CLK	BOO	cualquier variable booleana
Q	BOO	VERDADERO cuando CLK pasa de VERDADERO a FALSO FALSO en todos los demás casos

Descripción:

Detecta el flanco de bajada de una variable booleana

(* Programa FBD utilizando el Bloque "F_TRIG" *)




```
(* Equivalencia ST: se supone que F_TRIG1 es una instancia del bloque FR_TRIG *)
F_TRIG1(cmd);
nb_edge := ANA(F_TRIG1.Q) + nb_edge;

(* Equivalencia IL: *)
LD      cmd
ST      F_TRIG1.clk
CAL     F_TRIG1
LD      F_TRIG1.Q
ANA
ADD     nb_edge
ST      nb_edge
```

SEMA

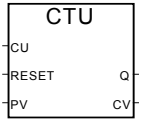
Argumentos:

CLAIM	BOOLEANO	comando de “comprobar y establecer”
RELEASE	BOOLEANO	libera el semáforo
BUSY	BOOLEANO	estado del semáforo

Descripción:

```
(* "x" es una variable booleana inicializada en FALSO *)
busy := x;
If claim Then
        x := VERDADERO;
Else
        If release Then
                busy := FALSO;
                x := FALSO;
        End_If;
End_If;
```

CTU



Argumentos:

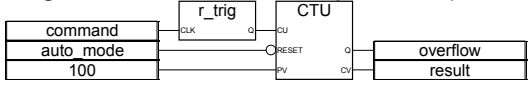
CU	BOO	entrada de contaje (contaje creciente cuando CU es VERDADERO)
RESET	BOO	comando de restablecer (dominante)
PV	INT	valor máximo programado
Q	BOO	desbordamiento: VERDADERO cuando CV = PV
CV	INT	resultado del contaje

Advertencia: El bloque CTU no detecta los flancos de subida o bajada de la entrada de contaje (CU). Tiene que estar asociado a bloques "R_TRIG" o "F_TRIG" para crear un contador de impulsos.

Descripción:

Cuenta (números enteros) desde 0 hasta un valor determinado en pasos de 1

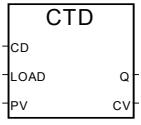
(* Programa FBD utilizando el Bloque "CTU" *)




```
(* Equivalencia ST: se supone que R_TRIG1 es una instancia del bloque R_TRIG y que
CTU1 es una instancia del bloque CTU *)
CTU1(R_TRIG1(command),NOT(auto_mode),100);
overflow := CTU1.Q;
result := CTU1.CV;

(* Equivalencia IL: *)
LD      command
ST      R_TRIG1.clk
CAL     R_TRIG1
LD      R_TRIG1.Q
ST      CTU1.cu
```

```
LDN     auto_mode
ST      CTU1.reset
LD      100
ST      CTU1.pv
CAL     CTU1
LD      CTU1.Q
ST      overflow
LD      CTU1.cv
ST      result
```

CTD



Argumentos:

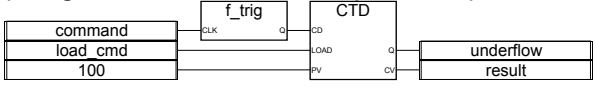
CD	BOO	entrada de contaje (cuenta decreciente cuando CD es VERDADERO)
LOAD	BOO	comando de cargar (dominante) (CV = PV cuando LOAD es VERDADERO)
PV	INT	valor inicial programado
Q	BOO	subdesbordamiento: VERDADERO cuando CV = 0
CV	INT	resultado del contaje

Advertencia: El bloque CTD no detecta los flancos de subida o bajada de la entrada de contaje (CD). Tiene que estar asociado a bloques "R_TRIG" o "F_TRIG" para crear un contador de impulsos.

Descripción:

Cuenta (números enteros) desde un valor determinado hasta 0 en pasos de 1

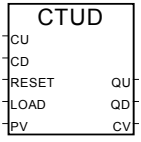
(* Programa FBD utilizando el Bloque "CTD" *)




```
(* Equivalencia ST: se supone que F_TRIG1 es una instancia del bloque F_TRIG y que CTD1
es una instancia del bloque CTD *)
CTD1(F_TRIG1(command),load_cmd,100);
underflow := CTD1.Q;
result := CTD1.CV;

(* Equivalencia IL: *)
LD      command
ST      F_TRIG1.clk
CAL     F_TRIG1
LD      F_TRIG1.Q
ST      CTD1.cd
```

```
LD      load_cmd
ST      CTD1.load
LD      100
ST      CTD1.pv
CAL     CTD1
LD      CTD1.Q
ST      underflow
LD      CTD1.cv
ST      result
```

CTUD



Argumentos:

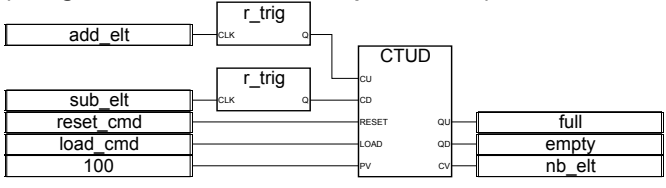
CU	BOO	contaje creciente (cuando CU es VERDADERO)
CD	BOO	contaje decreciente (cuando CD es VERDADERO)
RESET	BOO	comando de restablecer (dominante) (CV = 0 cuando RESET es VERDADERO)
LOAD	BOO	comando de cargar (CV = PV cuando LOAD es VERDADERO)
PV	INT	valor inicial programado
QU	BOO	desbordamiento: VERDADERO cuando CV = PV
QD	BOO	subdesbordamiento: VERDADERO cuando CV = 0
CV	INT	resultado del contaje

Advertencia: El bloque CTUD no detecta los flancos de subida o bajada de las entradas de contaje (CU y CD). Tiene que estar asociado a bloques "R_TRIG" o "F_TRIG" para crear un contador de impulsos.

Descripción:

Cuenta (números enteros) desde 0 hasta un valor determinado, en pasos de 1 o desde un valor determinado hasta 0, en pasos de 1

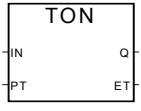
(* Programa FBD utilizando el Bloque "CTUD" *)




```
(* Equivalencia ST: se supone que R_TRIG1 y R_TRIG2 son dos instancias del bloque
R_TRIG y que CTUD1 es una instancia del bloque CTUD *)
CTUD1(R_TRIG1(add_elt), R_TRIG2(sub_elt), reset_cmd, load_cmd,100);
```

```
full := CTUD1.QU;
empty := CTUD1.QD;
nb_elt := CTUD1.CV;

(* Equivalencia IL: *)
LD          add_elt
ST          R_TRIG1.clk
CAL         R_TRIG1
LD          R_TRIG1.Q
ST          CTUD1.cu
LD          sub_elt
ST          R_TRIG2.clk
CAL         R_TRIG2
LD          R_TRIG2.Q
ST          CTUD1.cd
LD          reset_cmd
ST          CTUD1.reset
LD          load_cmd
ST          CTUD1.load
LD          100
ST          CTUD1.pv
CAL         CTUD1
LD          CTUD1.QU
ST          full
LD          CTUD1.QD
ST          empty
LD          CTUD1.CV
ST          nb_elt
```

TON



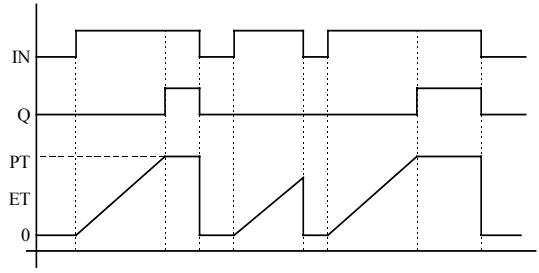
Argumentos:

IN	BOO	Con flanco de subida, comienza a aumentar el temporizador interno Con flanco de bajada, para y reinicia el temporizador interno
PT	TMR	tiempo máximo programado
Q	BOO	Si estado VERDADERO:ha transcurrido el tiempo programado
ET	TMR	tiempo transcurrido actual

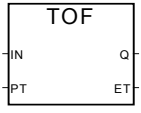
Descripción:

Incrementa un temporizador interno hasta un valor determinado.

Diagrama de tiempo:



TOF



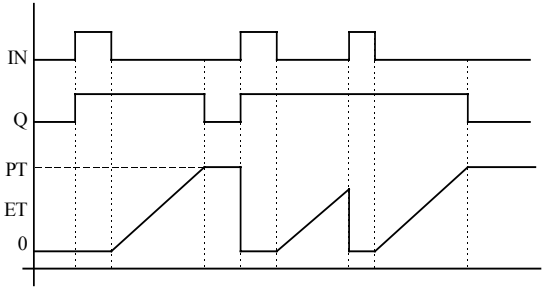
Argumentos:

IN	BOO	Con flanco de bajada, comienza a aumentar el temporizador interno Con flanco de subida, para y reinicia el temporizador interno
PT	TMR	tiempo máximo programado
Q	BOO	Si estado VERDADERO: no ha transcurrido el tiempo programado
ET	TMR	tiempo transcurrido actual

Descripción:

Incrementa un temporizador interno hasta un valor determinado.

Diagrama de tiempo:



TP

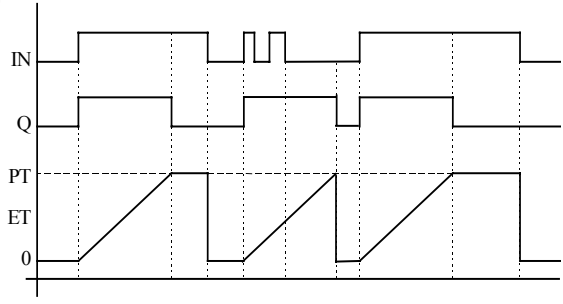
Argumentos:

IN	BOO	Con flanco de subida, comienza a aumentar el temporizador interno (si no se encuentra ya en aumento) Con estado FALSO y sólo si ha transcurrido el temporizador, reinicia el temporizador interno Cualquier cambio en IN durante el contaje queda sin efecto
PT	TMR	tiempo máximo programado
Q	BOO	Si estado VERDADERO: el temporizador está contando
ET	TMR	tiempo transcurrido actual

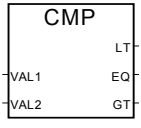
Descripción:

Incrementa un temporizador interno hasta un valor determinado.

Diagrama de tiempo:



CMP



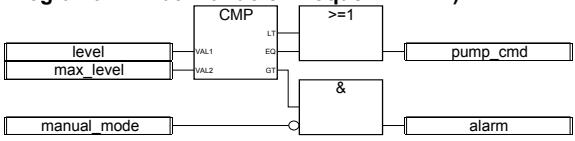
Argumentos:

VAL1	INT	cualquier valor analógico entero con su correspondiente signo
VAL2	INT	cualquier valor analógico entero con su correspondiente signo
LT	BOO	VERDADERO si val1 es inferior a val2
EQ	BOO	VERDADERO si val1 es igual a val2
GT	BOO	VERDADERO si val1 es mayor que val2

Descripción:

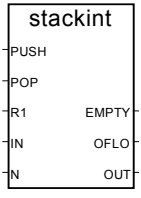
Comparar dos valores, determinando si son iguales o si el primero es inferior o mayor que el segundo.

(* Programa FBD utilizando el Bloque "CMP" *)




```
(* Equivalencia ST: se supone que CMP1 es una instancia del bloque CMP *)
CMP1(level, max_level);
pump_cmd := CMP1.LT OR CMP1.EQ;
alarm := CMP1.GT Y NOT(manual_mode);

(* Equivalencia IL: *)
LD          level
ST          CMP1.val1
LD          max_level
ST          CMP1.val2
CAL         CMP1
LD          CMP1.LT
OR          CMP1.EQ
ST          pump_cmd
LD          CMP1.GT
ANDN        manual_mode
ST          alarm
```

STACKINT



Argumentos:

PUSH	BOO	comando *push* (introducir) (sólo flanco de subida) añade el valor IN a la cabeza de la pila
POP	BOO	comando *pop* (sacar) (sólo flanco de subida) elimina de la pila el último valor 'introducido' (cabeza de la pila)
R1	BOO	restablece la pila a su estado vacío
IN	INT	valor 'introducido'
N	INT	tamaño del pila definido por la aplicación
EMPTY	BOO	VERDADERO si la pila está vacía
OFLO	BOO	desbordamiento: VERDADERO si la pila está completa

OUT	INT	valor situado a la cabeza de la pila

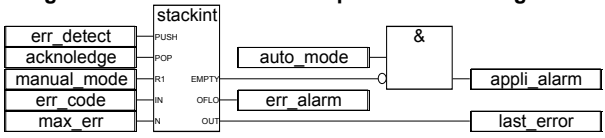
Descripción:

Gestionar una pila de valores enteros.

El bloque de función "STACKINT" incluye la detección de flanco de subida para los comandos *PUSH* y *POP*. El tamaño máximo del grupo es de **128**. El tamaño de pila definido por la aplicación, **N**, no puede ser inferior a 1 o superior a 128.

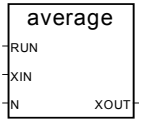
Observe que el valor OFLO sólo es válido después de un restablecimiento (R1 se sitúa en VERDADERO al menos en una ocasión, volviendo posteriormente a FALSO).

(* Programa FBD utilizando el Bloque "STACKINT": gestión de errores *)




```
(* Equivalencia ST: se supone que STACKINT1 es una instancia del bloque STACKINT *)
STACKINT1(err_detect, ackneledge, manual_mode, err_code, max_err);
appli_alarm := auto_mode Y NOT(STACKINT1.EMPTY);
err_alarm := STACKINT1.OFLO;
last_error := STACKINT1.OUT;

(* Equivalencia IL: *)
LD      err_detect
ST      STACKINT1.push
LD      ackneledge
ST      STACKINT1.pop
LD      manual_mode
ST      STACKINT1.r1
LD      err_code
ST      STACKINT1.IN
LD      max_err
ST      STACKINT1.N
CAL     STACKINT1
LD      auto_mode
ANDN    STACKINT1.empty
ST      appli_alarm
LD      STACKINT1.OFLO
ST      err_alarm
LD      STACKINT1.OUT
ST      last_error
```

AVERAGE



Argumentos:

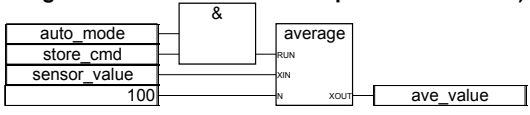
RUN	BOO	VERDADERO=promediar / FALSO=restablecer
XIN	REAL	cualquier variable analógica real
N	INT	número de muestreos definido por la aplicación
XOUT	REAL	media dinámica del valor XIN

Descripción:

Almacena un valor en cada ciclo y calcula el valor medio de todos los valores ya almacenados. Sólo se almacenan los ultimos valores N.

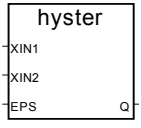
El número de muestreos **N** no puede superar **128**.
Si el comando "**RUN**" es **FALSO** (modo restablecer), el valor de salida es igual al valor de entrada.
Cuando se alcanza el número máximo de valores almacenados N, el primer valor almacenado queda reemplazado por el último.

(* Programa FBD utilizando el Bloque "AVERAGE": *)




```
(* Equivalencia ST: AVERAGE1 es instancia del bloque AVERAGE *)
AVERAGE1((auto_mode & store_cmd), sensor_value, 100);
ave_value := AVERAGE1.XOUT;

(* Equivalencia IL: *)
LD      auto_mode
Y       store_cmd
ST      AVERAGE1.run
LD      sensor_value
ST      AVERAGE1.xin
LD      100
ST      AVERAGE1.N
CAL     AVERAGE1
LD      AVERAGE1.XOUT
ST      ave_value
```

HYSTER



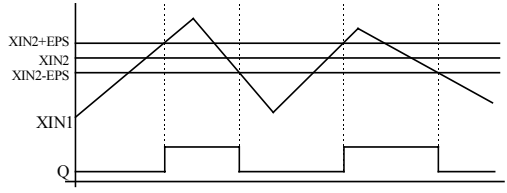
Argumentos:

XIN1	REAL	cualquier valor analógico real
XIN2	REAL	para comprobar si XIN1 ha sobrepasado XIN2+EPS
EPS	REAL	valor de histéresis (debe ser superior a cero)
Q	BOO	VERDADERO si XIN1 ha sobrepasado XIN2+EPS y todavía no está por debajo de XIN2-EPS

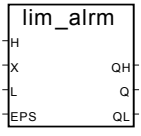
Descripción:

Histéresis de valor real para un límite superior.

Ejemplo de diagrama de tiempo:



LIM_ALRM



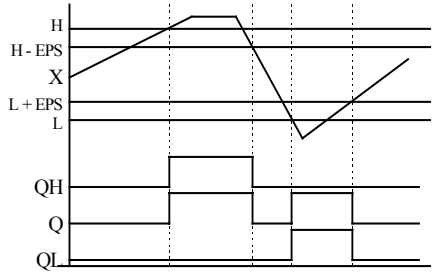
Argumentos:

H	REAL	valor de límite superior
X	REAL	entrada: cualquier valor analógico real
L	REAL	valor de límite inferior
EPS	REAL	valor de histéresis (debe ser superior a cero)
QH	BOO	alarma “superior”: VERDADERO si X está por encima de límite superior H
Q	BOO	salida de alarma: VERDADERO si X está fuera de límites
QL	BOO	alarma “inferior”: VERDADERO si X está por debajo del límite inferior L

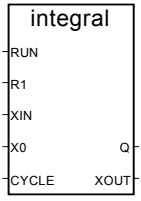
Descripción:

Histéresis de valor real para límites superior e inferior.

Se aplica una histéresis a los límites superior e inferior. El delta de histéresis que se utiliza para cualquiera de los dos límites es la mitad del parámetro EPS. A continuación se muestra un ejemplo de un diagrama de tiempo:



INTEGRAL



Argumentos:

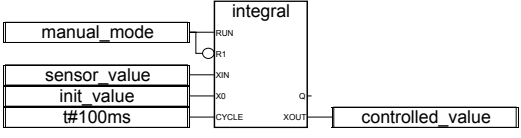
RUN	BOO	modo: VERDADERO=integrar / FALSO=mantener
R1	BOO	restablecimiento prioritario
XIN	REAL	entrada: cualquier valor analógico real
X0	REAL	valor inicial
CYCLE	TMR	periodo de muestreo
Q	BOO	R1 negado
XOUT	REAL	salida integrada

Descripción:

Integración de un valor real.

Si el valor del parámetro "**CYCLE**" es inferior al tiempo de ciclo de la aplicación, el periodo de muestreo equivale al tiempo de ciclo de la aplicación.

(* Programa FBD utilizando el Bloque "INTEGRAL": *)




```
(* Equivalencia ST: INTEGRAL1 es una instancia del bloque INTEGRAL *)
INTEGRAL1(manual_mode, NOT(manual_mode), sensor_value, init_value, t#100ms);
controlled_value := INTEGRAL1.XOUT;

(* Equivalencia IL: *)
LD              manual_mode
```

```
ST          INTEGRAL1.run
STN         INTEGRAL1.R1
LD          sensor_value
ST          INTEGRAL1.XIN
LD          init_value
ST          INTEGRAL1.X0
LD          t#100ms
ST          INTEGRAL1.CYCLE
CAL         INTEGRAL1
LD          INTEGRAL1.XOUT
ST          controlled_value
```

DERIVATE

Argumentos:

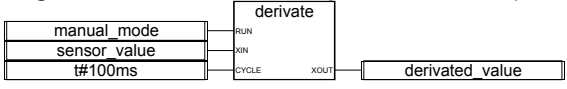
RUN	BOO	modo: VERDADERO=normal / FALSO=restablecer
XIN	REAL	entrada: cualquier valor analógico real
CYCLE	TMR	periodo de muestreo
XOUT	REAL	salidas diferenciadas

Descripción:

Diferenciación de un valor real.

Si el valor del parámetro "**CYCLE**" es inferior al tiempo de ciclo de la aplicación ISaGRAF, el periodo de muestreo equivale al tiempo de ciclo de la aplicación.

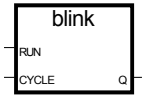
(* Programa FBD utilizando el Bloque "DERIVATE": *)




```
(* Equivalencia ST: DERIVATE1 es una instancia del bloque DERIVATE *)
DERIVATE1(manual_mode, sensor_value, t#100ms);
derivated_value := DERIVATE1.XOUT;

(* Equivalencia IL: *)
LD          manual_mode
ST          DERIVATE1.run
LD          sensor_value
ST          DERIVATE1.XIN
LD          t#100ms
ST          DERIVATE1.CYCLE
CAL         DERIVATE1
LD          DERIVATE1.XOUT
```

ST derivated_value

BLINK



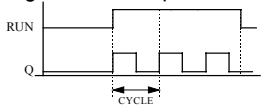
Argumentos:

RUN	BOO	modo: VERDADERO=parpadeante / FALSO=restablecer la salida en estado falso
CYCLE	TMR	periodo de parpadeo
Q	BOO	señal de salida parpadeante

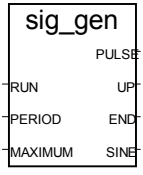
Descripción:

Genera una señal parpadeante.

Diagrama de tiempo:



SIG_GEN



Argumentos:

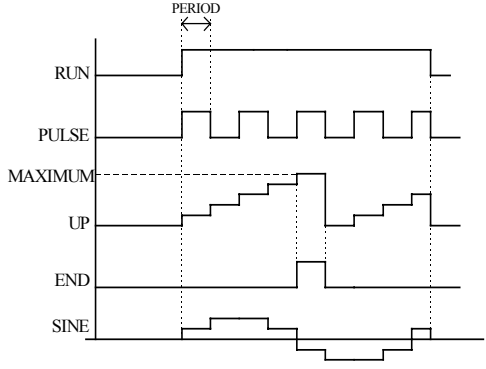
RUN	BOO	modo: VERDADERO=en operación / FALSO=restablecer en estado falso
PERIOD	TMR	duración de una muestra
MAXIMUM	INT	valor máximo de contaje
PULSE	BOO	invertido después de cada muestra
UP	INT	contador creciente, incrementado con cada muestra
END	BOO	VERDADERO cuando finaliza la cuenta creciente
SENO	REAL	señal senoidal (periodo = duración de contaje)

Descripción:

Genera varias señales: parpadeo para booleanos, cuenta creciente de valores enteros y ondas senoidales reales.

Cuando el proceso de contaje alcanza su valor máximo, se reinicia desde 0 (cero). Por tanto, END sólo mantiene el valor VERDADERO durante un PERIOD.

Diagrama de tiempo:



B.9.3 Funciones estándares

Estas son funciones estándares soportadas por el sistema ISaGRAF. Dichas funciones están predefinidas y no tienen que ser declaradas en la librería.

● Matemática	ABS	Valor absoluto
	EXPT, POW	Cálculo de Exponente, Potencia
	LOG	Logaritmos
	SQRT	Raíz cuadrada
	TRUNC	Truncar parte decimal
● Trigonometría.....	ACOS, ASIN,	Arco coseno, Arco seno,
	ATAN	Arco tangente
	COS, SIN,	Coseno, Seno,
	TAN	Tangente
● Control de registros	ROL, ROR	Rotar a la izquierda, Rotar a la derecha
	SHL, SHR	Desplazar a la izquierda, Desplazar a la derecha
● Tratamiento de datos	MIN, MAX,	Mínimo, Máximo,
	LIMIT	Límite
	MOD	Módulo
	MUX4, MUX8	Multiplexor (4 u 8 entradas),
	SEL	Selector binario
	ODD	Paridad impar
	RAND	Valor aleatorio
● Conversión de datos	ASCII	Carácter → Código ASCII
	CHAR	Código ASCII → Carácter
● Gestión de cadenas	MLEN	Obtener longitud de cadena
	DELETE	Borrar subcadena,
	INSERT	Insertar cadena

	FIND,	Encontrar subcadena,
	REPLACE	Sustituir subcadena
	LEFT, MID	Extraer parte izquierda, central
	RIGHT	o derecha de una cadena
	DAY_TIME	Hora del día
● Tratamiento de grupos	ARCREATE	Crear vector de valores enteros
	ARREAD	Leer /
	ARWRITE	Escribir elementos de vector
● Gestión de ficheros binarios..	F_ROPEN	Abrir un fichero binario en modo Lectura
	F_WOPEN	Abrir un fichero binario en modo Escritura
	F_CLOSE	Cerrar un fichero binario
	F_EOF	Comprobar el final de un fichero binario
	FA_READ	Leer un valor analógico en un fichero binario
	FA_WRITE	Escribir un valor analógico en un fichero binario
	FM_READ	Leer una cadena de mensajes en un fichero binario
	FM_WRITE	Escribir una cadena de mensajes en un fichero binario

ABS

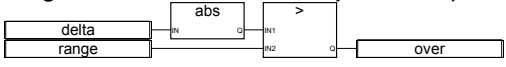
Argumentos:

IN	REAL	cualquier valor analógico real con signo
Q	REAL	valor absoluto (siempre positivo)

Descripción:

Proporciona el valor absoluto (positivo) de un valor real.

(* Programa FBD utilizando el Bloque "ABS" *)




```
(* Equivalencia ST: *)
over := (ABS (delta) > range);

(* Equivalencia IL: *)
LD          delta
ABS
GT          range
ST          over
```

EXPT

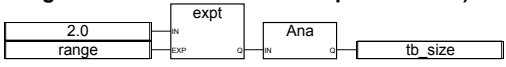
Argumentos:

IN	REAL	cualquier valor analógico real con signo
EXP	INT	exponente analógico entero
Q	REAL	(IN ^{EXP})

Descripción:

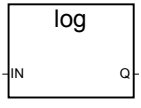
Proporciona el resultado real de la operación: (base ^{exponente}), siendo 'base' el primer argumento y 'exponente' el segundo.

(* Programa FBD utilizando el Bloque "EXPT" *)




```
(* Equivalencia ST: *)
tb_size := ANA (EXPT (2.0, range) );

(* Equivalencia IL: *)
LD          2.0
EXPT        range
ANA
ST          tb_size
```

LOG



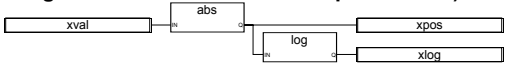
Argumentos:

IN	REAL	debe ser superior a cero
Q	REAL	logaritmo (base 10) del valor de entrada

Descripción:

Calcula el logaritmo (base 10) de un valor real.

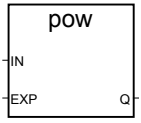
(* Programa FBD utilizando el Bloque "LOG" *)




```
(* Equivalencia ST: *)
xpos := ABS (xval);
```

```
xlog := LOG (xpos);

(* Equivalencia IL: *)
LD          xval
ABS
ST          xpos
LOG
ST          xlog
```

POW



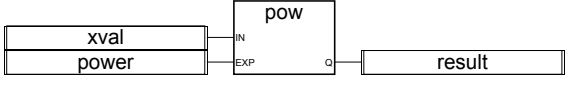
Argumentos:

IN	REAL	número analógico real a elevar
EXP	REAL	potencia (exponente)
Q	REAL	(IN ^{EXP}) 1.0 si IN no es 0.0 y EXP es 0.0 0.0 si IN es 0.0 y EXP es negativo 0.0 si tanto IN como EXP son 0.0 0.0 si IN es negativo e Y no corresponde a un número entero

Descripción:

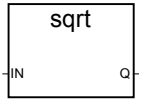
Proporciona el resultado real de la operación: (base ^{exponente}), siendo 'base' el primer argumento y 'exponente' el segundo. El exponente es un valor real.

(* Programa FBD utilizando el Bloque "POW" *)




```
(* Equivalencia ST: *)
resultado := POW (xval, power);

(* Equivalencia IL: *)
LD          xval
POW         power
ST          resultado
```

SQRT



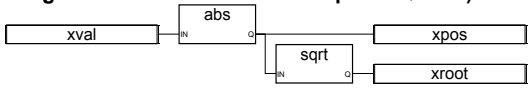
Argumentos:

IN	REAL	debe ser superior o igual a cero
Q	REAL	raíz cuadrada del valor de entrada

Descripción:

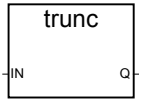
Calcula la raiz cuadrada de un valor real.

(* Programa FBD utilizando el Bloque "SQRT" *)




```
(* Equivalencia ST: *)
xpos := ABS (xval);
xroot := SQRT (xpos);

(* Equivalencia IL: *)
LD          xval
ABS
ST          xpos
SQRT
ST          xrout
```

TRUNC



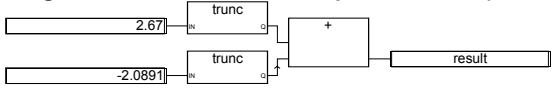
Argumentos:

IN	REAL	cualquier valor analógico REAL
Q	REAL	si IN>0, mayor número entero inferior o igual a la entrada si IN<0, menor número entero mayor o igual a la entrada

Descripción:

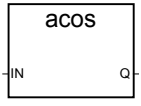
Trunca un valor real para que tenga sólo la parte entera

(* Programa FBD utilizando el Bloque "TRUNC" *)



```
(* Equivalencia ST: *)
resultado := TRUNC (+2.67) + TRUNC (-2.0891);
(* significa : resultado := 2.0 + (-2.0) := 0.0; *)

(* Equivalencia IL: *)
LD          2.67
TRUNC
ST          temporary   (* resultado temporal del primer TRUNC *)
LD          -2.0891
TRUNC
ADD         temporary
ST          result
```

ACOS



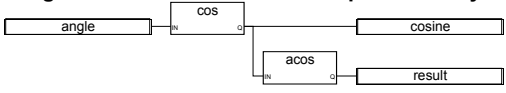
Argumentos:

IN	REAL	debe estar en el conjunto [-1.0 .. +1.0]
Q	REAL	arco coseno del valor de entrada (en el conjunto [0.0 .. PI]) = 0.0 para entradas no válidas

Descripción:

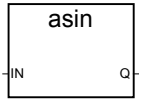
Calcula el Arco coseno de un valor real.

(* Programa FBD utilizando los Bloques "COS" y "ACOS" *)




```
(* Equivalencia ST: *)
cosine := COS (angle);
result := ACOS (cosine); (* resultado es igual a un ángulo *)

(* Equivalencia IL: *)
LD          angle
COS
ST          cosine
ACOS
ST          result
```

ASIN



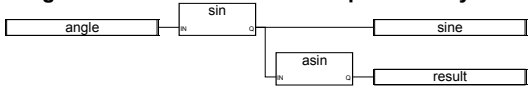
Argumentos:

IN	REAL	debe estar en el conjunto [-1.0 .. +1.0]
Q	REAL	arco coseno del valor de entrada (en el conjunto [-PI/2 .. +PI/2]) = 0.0 para entradas no válidas

Descripción:

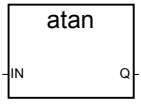
Calcula el Arco seno de un valor real.

(* Programa FBD utilizando los Bloques "SIN" y "ASIN" *)




```
(* Equivalencia ST: *)
sine := SIN (angle);
result := ASIN (sine); (* resultado es igual a un ángulo *)

(* Equivalencia IL: *)
LD          angle
SIN
ST          sine
ASIN
ST          result
```

ATAN



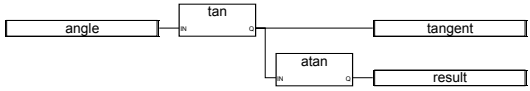
Argumentos:

IN	REAL	cualquier valor analógico REAL
Q	REAL	arco tangente del valor de entrada (en el conjunto [-PI/2 .. +PI/2]) = 0.0 para entradas no válidas

Descripción:

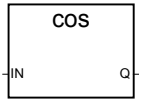
Calcula el Arco tangente de un valor real.

(* Programa FBD utilizando los Bloques "TAN" y "ATAN" *)




```
(* Equivalencia ST: *)
tangent := TAN (angle);
result := ATAN (tangent); (* resultado es igual a un ángulo *)

(* Equivalencia IL: *)
LD          angle
TAN
ST          tangent
ATAN
ST          result
```

COS



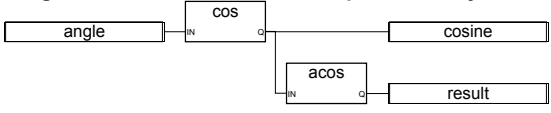
Argumentos:

IN	REAL	cualquier valor analógico REAL
Q	REAL	coseno del valor de entrada (en el conjunto [-1.0 .. +1.0])

Descripción:

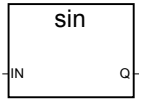
Calcula el Coseno de un valor real.

(* Programa FBD utilizando los Bloques "COS" y "ACOS" *)




```
(* Equivalencia ST: *)
cosine := COS (angle);
result := ACOS (cosine); (* resultado es igual a ángulo *)

(* Equivalencia IL: *)
LD          angle
COS
ST          cosine
ACOS
ST          result
```

SIN



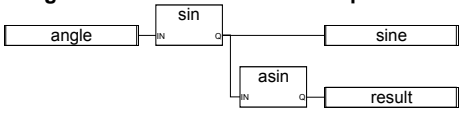
Argumentos:

IN	REAL	cualquier valor analógico REAL
Q	REAL	seno del valor de entrada (en el conjunto [-1.0 .. +1.0])

Descripción:

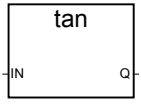
Calcula el Seno de un valor real.

(* Programa FBD utilizando los Bloques "SIN" y "ASIN" *)




```
(* Equivalencia ST: *)
sine := SIN (angle);
result := ASIN (sine); (* resultado es igual a ángulo *)

(* Equivalencia IL: *)
LD          angle
SIN
ST          sine
ASIN
ST          result
```

TAN



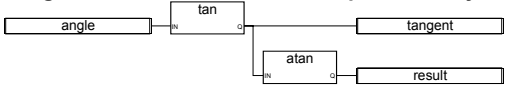
Argumentos:

IN	REAL	no puede ser igual a PI/2 módulo PI
Q	REAL	tangente del valor de entrada = 1E+38 para entradas no válidas

Descripción:

Calcula la Tangente de un valor real.

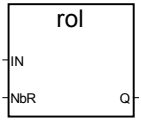
(* Programa FBD utilizando los Bloques "TAN" y "ATAN" *)




```
(* Equivalencia ST: *)
```

```
tangent := TAN (angle);
result := ATAN (tangent); (* resultado es igual a ángulo *)

(* Equivalencia IL: *)
LD          angle
TAN
ST          tangent
ATAN
ST          result
```

ROL

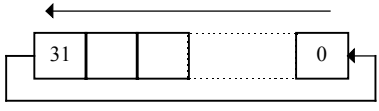


Argumentos:

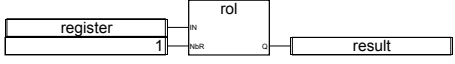
IN	INT	cualquier valor analógico entero
NbR	INT	número de rotaciones de 1 bit (en el conjunto [1..31])
Q	INT	valor rotado a la izquierda Sin efecto si NbR <= 0

Descripción:

Provoca la rotación a la izquierda de los bits de un valor entero. La rotación se realiza sobre 32 bits:

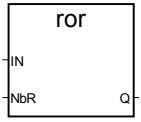


(* Programa FBD utilizando el Bloque "ROL" *)




```
(* Equivalencia ST: *)
result := ROL (register, 1);
(* register = 2#0100_1101_0011_0101*)
(* result = 2#1001_1010_0110_1010*)

(* Equivalencia IL: *)
LD          register
ROL         1
ST          result
```

ROR

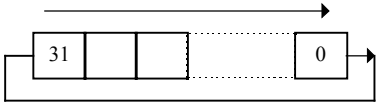


Argumentos:

IN	INT	cualquier valor analógico entero
NbR	INT	número de rotaciones de 1 bit (en el conjunto [1..31])
Q	INT	valor rotado a la derecha Sin efecto si NbR <= 0

Descripción:

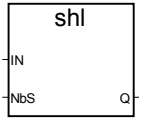
Provoca la rotación a la derecha de los bits de un valor entero. La rotación se realiza sobre 32 bits:



(* Programa FBD utilizando el Bloque "ROR" *)


```
(* Equivalencia ST: *)
result := ROR (register, 1);
(* register = 2#0100_1101_0011_0101 *)
(* result = 2#1010_0110_1001_1010 *)

(* Equivalencia IL: *)
LD          register
ROR         1
ST          result
```

SHL



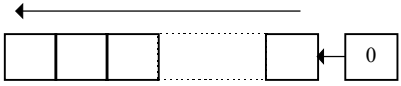
Argumentos:

IN	INT	cualquier valor analógico entero
NbS	INT	número de desplazamientos de 1 bit (en el conjunto [1..31])

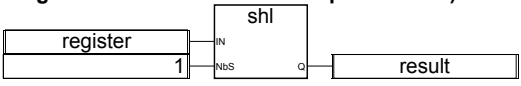
Q	INT	valor desplazado a la izquierda Sin efecto si NbS <= 0 Se emplea 0 para sustituir al bit inferior

Descripción:

Provoca el desplazamiento a la izquierda de los bits de un valor entero. El desplazamiento se realiza sobre 32 bits:

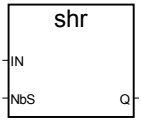


(* Programa FBD utilizando el Bloque "SHL" *)




```
(* Equivalencia ST: *)
result := SHL (register,1);
(* register = 2#0100_1101_0011_0101 *)
(* result = 2#1001_1010_0110_1010 *)

(* Equivalencia IL: *)
LD          register
SHL         1
ST          result
```

SHR

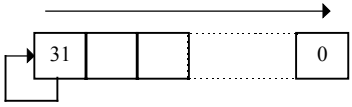


Argumentos:

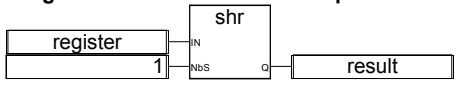
IN	INT	cualquier valor analógico entero
NbS	INT	número de desplazamientos de 1 bit (en el conjunto [1..31])
Q	INT	valor desplazado a la derecha Sin efecto si NbS <= 0 Se copia el bit superior en cada desplazamiento

Descripción:

Provoca el desplazamiento a la derecha de los bits de un valor entero. El desplazamiento se realiza sobre 32 bits:



(* Programa FBD utilizando el Bloque "SHR" *)




```
(* Equivalencia ST: *)
result := SHR (register,1);
(* register = 2#1100_1101_0011_0101 *)
(* result = 2#1110_0110_1001_1010 *)

(* Equivalencia IL: *)
LD          register
SHR         1
ST          result
```

MIN

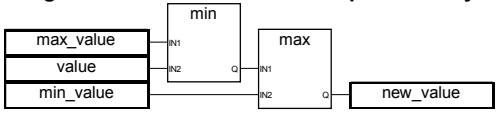
Argumentos:

IN1	INT	cualquier valor analógico entero con signo
IN2	INT	(no puede ser REAL)
Q	INT	mínimo de ambos valores de entrada

Descripción:

Proporciona el mínimo de dos valores enteros.

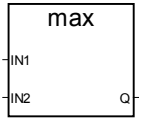
(* Programa FBD utilizando los Bloques "MIN" y "MAX" *)




```
(* Equivalencia ST: *)
new_value := MAX (MIN (max_value, value), min_value);
(* limita el valor al conjunto [min_value..max_value] *)

(* Equivalencia IL: *)
LD          max_value
MIN         value
MAX         min_value
```

ST new_value

MAX



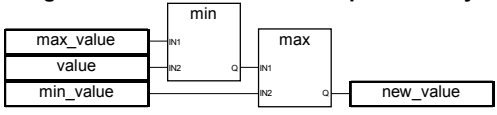
Argumentos:

IN1	INT	cualquier valor analógico entero con signo
IN2	INT	(no puede ser REAL)
Q	INT	máximo de ambos valores de entrada

Descripción:

Proporciona el máximo de dos valores enteros.

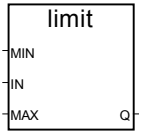
(* Programa FBD utilizando los Bloques "MIN" y "MAX" *)



(* Equivalencia ST: *)
new_value := MAX (MIN (max_value, value), min_value);
(* limita el valor al conjunto [min_value..max_value] *)

(* Equivalencia IL: *)
LD max_value
MIN value
MAX min_value
ST new_value

LIMIT



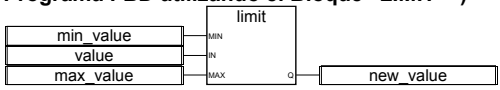
Argumentos:

MIN	INT	valor mínimo permitido
IN	INT	cualquier valor analógico entero con signo
MAX	INT	valor máximo permitido
Q	INT	valor de entrada limitado al rango permitido

Descripción:

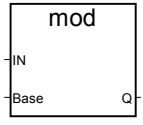
Limita un valor entero a un intervalo determinado. Mantiene su valor si se encuentra entre el mínimo y el máximo, cambia a máximo si se encuentra por encima o cambia a mínimo si se encuentra por debajo.

(* Programa FBD utilizando el Bloque "LIMIT" *)




```
(* Equivalencia ST: *)
new_value := LIMIT (min_value, value, max_value);
(* limita el valor al conjunto [min_value..max_value] *)

(* Equivalencia IL: *)
LD          min_value
LIMIT       value, max_value
ST          new_value
```

MOD



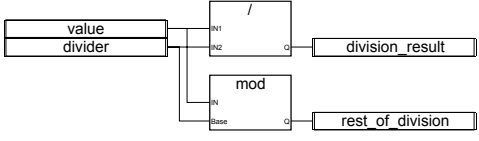
Argumentos:

IN	INT	cualquier valor analógico entero con signo
Base	INT	debe ser superior a cero
Q	INT	cálculo de módulo (entrada MOD base) retorna -1 si Base <= 0

Descripción:

Calcula el módulo de un valor entero.

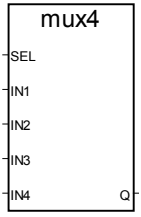
(* Programa FBD utilizando el Bloque "MOD" *)




```
(* Equivalencia ST: *)
division_result := (value / divider); (* división entera *)
rest_of_division := MOD (value, divider); (* resto de la división *)

(* Equivalencia IL: *)
LD          value
DIV         divider
ST          division_result
```

```
LD          value
MOD         divider
ST          rest_of_division
```

MUX4



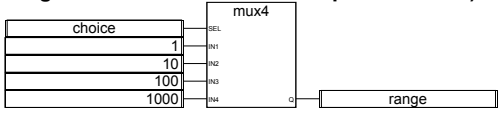
Argumentos:

SEL	INT	valor entero del selector (tiene que estar en el conjunto [0..3])
IN1..IN4	INT	cualquier valor analógico entero
Q	INT	= valor1 si SEL = 0 = valor2 si SEL = 1 = valor3 si SEL = 2 = valor4 si SEL = 3 = 0 para los restantes valores del selector

Descripción:

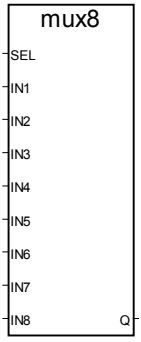
Multiplexor con 4 entradas: selecciona un valor entre 4 valores enteros.

(* Programa FBD utilizando el Bloque "MUX4" *)




```
(* Equivalencia ST: *)
range := MUX4 (choice, 1, 10, 100, 1000);
(* selecciona entre 4 rangos predefinidos, por ejemplo, si la elección es 1, el rango será 10 *)

(* Equivalencia IL: *)
LD          choice
MUX4        1,10,100,1000
ST          range
```

MUX8



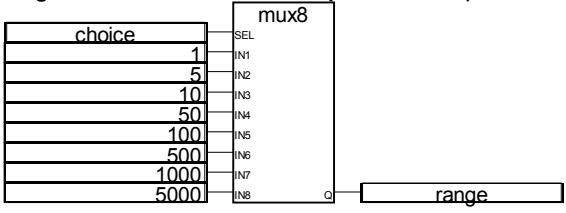
Argumentos:

SEL	INT	valor entero del selector (tiene que estar en el conjunto [0..7])
IN1..IN8	INT	cualquier valor analógico entero
Q	INT	= valor1 si selector = 0 = valor2 si selector = 1 ... = valor8 si selector = 7 = 0 para los restantes valores del selector

Descripción:

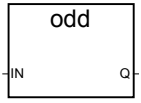
Multiplexor con 8 entradas: selecciona un valor entre 8 valores enteros.

(* Programa FBD utilizando el Bloque "MUX8" *)




```
(* Equivalencia ST: *)
range := MUX8 (choice, 1, 5, 10, 50, 100, 500, 1000, 5000);
(* selecciona entre 8 rangos predefinidos, por ejemplo, si la elección es 3, el rango será 50 *)

(* Equivalencia IL: *)
LD      choice
MUX8    1,5,10,50,100,500,1000,5000
ST      range
```

ODD



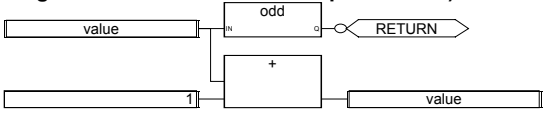
Argumentos:

IN	INT	cualquier valor analógico entero con signo
Q	BOO	VERDADERO si el valor de entrada es impar FALSO si el valor de entrada es par

Descripción:

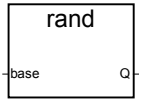
Comprueba la paridad de un valor entero: el resultado es impar o par.

(* Programa FBD utilizando el Bloque "ODD" *)




```
(* Equivalencia ST: *)
If Not (ODD (value)) Then Return; End_If;
value := value + 1;
(* hace el valor siempre par *)

(* Equivalencia IL: *)
LD          value
ODD
RETNC
LD          value
ADD         1
ST          value
```

RAND



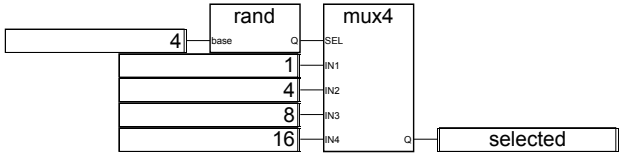
Argumentos:

base	INT	define el conjunto de números permitido
Q	INT	valor aleatorio en el conjunto [0..base-1]

Descripción:

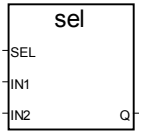
Asigna un valor entero aleatorio a un rango determinado.

(* Programa FBD utilizando el Bloque "RAND" block *)




```
(* Equivalencia ST: *)
selected := MUX4 ( RAND (4), 1, 4, 8, 16 );
(*
selección aleatoria de uno entre 4 valores predefinidos
el valor proporcionado por la llamada RAND pertence al conjunto [0..3], por ello el valor
'seleccionado' por MUX4, tomará 'aleatoriamente' el valor:
1 si RAND genera 0
ó 4 si RAND genera 1
ó 8 si RAND genera 2
ó 16 si RAND genera 3
*)

(* Equivalencia IL: *)
LD          4
RAND
MUX4        1,4,8,16
ST          selected
```

SEL



Argumentos:

SEL	BOO	indica el valor seleccionado
IN1, IN2	INT	cualquier valor analógico entero
Q	INT	= valor1 si SEL es FALSO = valor2 si SEL es VERDADERO

Descripción:

Selector binario: selecciona uno entre 2 valores enteros.

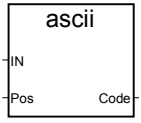
(* Programa FBD utilizando el Bloque "SEL" *)


```
(* Equivalencia ST: *)
ProCmd := SEL (AutoMode, ManuCmd, InpCmd);
```

(* procesa la selección de comando *)

(* Equivalencia IL: *)
LD AutoMode
SEL ManuCmd,InpCmd
ST ProCmd

ASCII



Argumentos:

IN	MSG	cualquier cadena no vacía
Pos	INT	posición del carácter seleccionado En el conjunto [1.. len] (len es la longitud del mensaje IN)
Code	INT	código del carácter seleccionado (en el conjunto [0 .. 255]) retorna 0 si Pos está fuera de la cadena

Descripción:

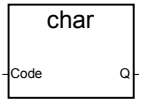
Proporciona el código ASCII de un carácter perteneciente a una cadena de mensaje.

(* Programa FBD utilizando el Bloque "ASCII" *)

(* Equivalencia ST: *)
FirstChr := ASCII (message, 1);
(* FirstChr es el código Ascii del primer carácter de la cadena *)

(* Equivalencia IL: *)
LD message
ASCII 1
ST FirstChr

CHAR



Argumentos:

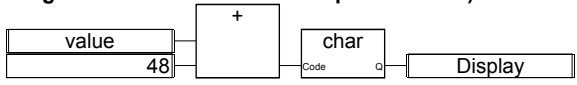
Code	INT	código en el conjunto [0 .. 255]
Q	MSG	una cadena de caracteres

El carácter tiene el código ASCII especificado en el Código de entrada
(el código ASCII utilizado es módulo 256)

Descripción:

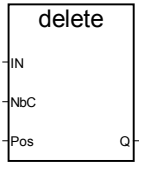
Proporciona una cadena de mensaje de un carácter a partir de un código ASCII determinado.

(* Programa FBD utilizando el Bloque "CHAR" *)




```
(* Equivalencia ST: *)
Display := CHAR ( value + 48 );
(* valor en el conjunto [0..9] *)
(* 48 es el código ascii de '0' *)
(* resultado es un cadena de un carácter de '0' a '9' *)

(* Equivalencia IL: *)
LD          value
ADD         48
CHAR
ST          Display
```

DELETE



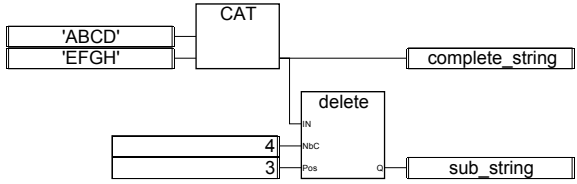
Argumentos:

IN	MSG	cualquier mensaje no vacío
NbC	INT	número de caracteres a borrar
Pos	INT	posición del primer carácter borrado (el primer carácter de la cadena tiene la posición 1)
Q	MSG	cadena modificada cadena vacía si Pos < 1 cadena inicial si Pos > IN longitud de cadena cadena inicial si NbC <= 0

Descripción:

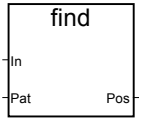
Borra parte de una cadena de mensaje.

(* Programa FBD utilizando el Bloque "DELETE" *)




```
(* Equivalencia ST: *)
complete_string := 'ABCD' + 'EFGH'; (* complete_string es 'ABCDEFGH' *)
sub_string := DELETE (complete_string, 4, 3); (* sub_string es 'ABGH' *)

(* Equivalencia IL: *)
LD          'ABCD'
ADD         'EFGH'
ST          complete_string
DELETE      4,3
ST          sub_string
```

FIND



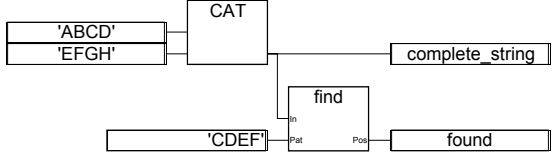
Argumentos:

In	MSG	cualquier cadena de mensajes
Pat	MSG	cualquier cadena no vacía (patrón)
Pos	INT	= 0 si no se encuentra la subcadena Pat = posición del primer carácter de la primera ocurrencia de la subcadena Pat (primera posición es 1) esta función es **sensible al uso de mayúsculas o minúsculas**

Descripción:

Localiza una subcadena en una cadena de mensajes. Indica la posición de la subcadena en la cadena.

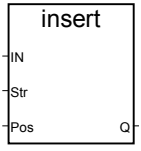
(* Programa FBD utilizando el Bloque "FIND" *)




```
(* Equivalencia ST: *)
```

```
complete_string := 'ABCD' + 'EFGH'; (* complete_string es 'ABCDEFGH' *)
found := FIND (complete_string, 'CDEF'); (* found es 3 *)

(* Equivalencia IL: *)
LD          'ABCD'
ADD         'EFGH'
ST          complete_string
FIND        'CDEF'
ST          found
```

INSERT



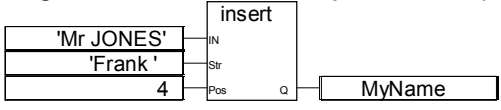
Argumentos:

IN	MSG	cadena inicial
Str	MSG	cadena a insertar
Pos	INT	posición de la inserción la inserción se realiza delante de la posición (primera posición válida es 1)
Q	MSG	cadena modificada cadena vacía si Pos <= 0 concatenación de ambas cadenas si Pos es mayor que la longitud de la cadena IN

Descripción:

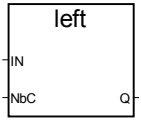
Inserta una subcadena en una posición determinada de una cadena de mensaje.

(* Programa FBD utilizando el Bloque "INSERT" *)




```
(* Equivalencia ST: *)
MyName := INSERT ('Mr JONES', 'Frank ', 4);
(* MyName es 'Mr Frank JONES' *)

(* Equivalencia IL: *)
LD          'Mr JONES'
INSERT      'Frank ',4
ST          MyName
```

LEFT



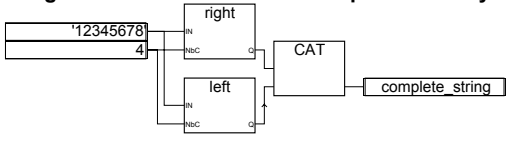
Argumentos:

IN	MSG	cualquier cadena no vacía
NbC	INT	el número de caracteres a extraer no puede ser superior a la longitud de la cadena IN
Q	MSG	parte izquierda de la cadena IN (su longitud = NbC) cadena vacía si NbC <= 0 cadena IN completa si NbC >= longitud de cadena IN

Descripción:

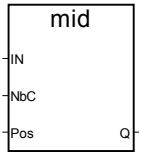
Extrae la parte izquierda de una cadena de mensajes. Se indica el número de caracteres a extraer.

(* Programa FBD utilizando los Bloques "LEFT" y "RIGHT" *)




```
(* Equivalencia ST: *)
complete_string := RIGHT ('12345678', 4) + LEFT ('12345678', 4);
(* complete_string es '56781234'
el valor proporcionado por la llamada a RIGHT es '5678'
el valor proporcionado por la llamada a LEFT es '1234'
*)

(* Equivalencia IL: Primero se llama a LEFT *)
LD          '12345678'
LEFT        4
ST          sub_string  (* resultado intermedio *)
LD          '12345678'
RIGHT       4
ADD         sub_string
ST          complete_string
```

MID



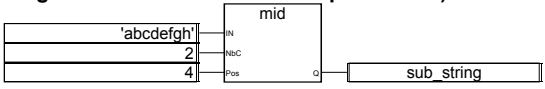
Argumentos:

IN	MSG	cualquier cadena no vacía
NbC	INT	el número de caracteres a extraer no puede ser superior a la longitud de la cadena IN
Pos	INT	posición de la subcadena el primer carácter de la subcadena será el señalado por Pos (la primera posición válida es 1)
Q	MSG	parte central de la cadena (su longitud = NbC) cadena vacía si los parámetros no son válidos

Descripción:

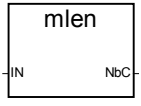
Extrae una parte de una cadena de mensaje. Se indica el número de caracteres a extraer y la posición del primer carácter.

(* Programa FBD utilizando el Bloque "MID" *)




```
(* Equivalencia ST: *)
sub_string := MID ('abcdefgh', 2, 4);
(* sub_string es 'de' *)

(* Equivalencia IL: *)
LD          'abcdefgh'
MID         2,4
ST          sub_string
```

MLEN



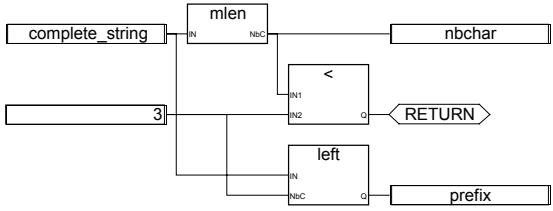
Argumentos:

IN	MSG	cualquier cadena de mensaje
NbC	INT	número de caracteres en la cadena IN

Descripción:

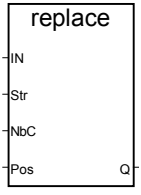
Calcula la longitud de una cadena de mensaje.

(* Programa FBD utilizando el Bloque "MLEN" *)




```
(* Equivalencia ST: *)
nbchar := MLEN (complete_string);
If (nbchar < 3) Then Return; End_if;
prefix := LEFT (complete_string, 3);
(* este programa extrae los 3 caracteres situados en la parte izquierda de la cadena y coloca
el resultado en la variable del prefijo de mensaje
no se hace nada si la longitud de cadena es inferior a 3 caracteres *)

(* Equivalencia IL: *)
LD          complete_string
MLEN
ST          nbchar
LT          3
RETC
LD          complete_string
LEFT        3
ST          prefix
```

REPLACE



Argumentos:

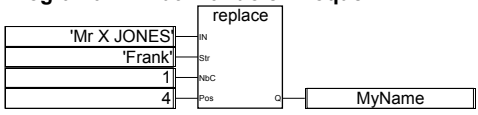
IN	MSG	cualquier cadena
Str	MSG	cadena a insertar (para reemplazar NbC caracteres)
NbC	INT	número de caracteres a borrar
Pos	INT	posición del primer carácter modificado (la primera posición válida es 1)
Q	MSG	cadena modificada: - se borran los NbC caracteres en la posición Pos - después, se inserta la subcadena Str en esa misma posición se devuelve una cadena vacía si Pos <= 0 se devuelve una concatenación de cadenas (IN+Str) si Pos es mayor que la longitud de la cadena IN

se devuelve la cadena inicial IN si NbC <= 0

Descripción:

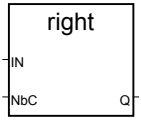
Sustituye una parte de una cadena de mensaje con un nuevo conjunto de caracteres.

(* Programa FBD utilizando el Bloque "REPLACE" *)




```
(* Equivalencia ST: *)
MyName := REPLACE ('Mr X JONES, 'Frank', 1, 4);
(* MyName es 'Mr Frank JONES' *)

(* Equivalencia IL: *)
LD          'Mr X JONES'
REPLACE     'Frank',1,4
ST          MyName
```

RIGHT



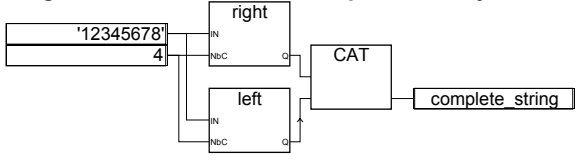
Argumentos:

IN	MSG	cualquier cadena no vacía
NbC	INT	no puede ser mayor que la longitud de la cadena IN
Q	MSG	parte derecha de la cadena (longitud = NbC) cadena vacía si NbC <= 0 cadena completa si NbC >= longitud de cadena

Descripción:

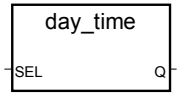
Extrae la parte derecha de una cadena de mensaje. Se indica el número de caracteres a extraer.

(* Programa FBD utilizando los Bloques "LEFT" y "RIGHT" *)



```
(* Equivalencia ST: *)
complete_string := RIGHT ('12345678', 4) + LEFT ('12345678', 4);
(* complete_string es '56781234'
el valor proporcionado por la llamada a RIGHT es ‘5678’
el valor proporcionado por la llamada a LEFT es ‘1234’
*)

(* Equivalencia IL: Primero se llama a LEFT *)
LD          '12345678'
LEFT        4
ST          sub_string  (* resultado intermedio *)
LD          '12345678'
RIGHT       4
ADD         sub_string
ST          complete_string
```

DAY_TIME



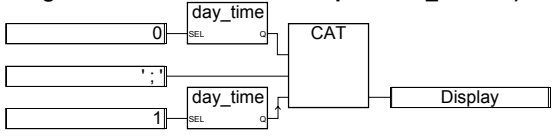
Argumentos:

SEL	INT	selección de salida 0= obtener fecha actual 1= obtener hora actual 2= obtener día de la semana
Q	MSG	hora/fecha expresadas en una cadena de caracteres: 'AAAA/MM/DD' si SEL = 0 'HH:MM:SS' si SEL = 1 nombre día si SEL = 2 (p.ej.: ‘Lunes’)

Descripción:

Proporciona la fecha o la hora del día como una cadena de mensaje.

(* Programa FBD utilizando el Bloque "DAY_TIME" *)




```
(* Equivalencia ST: *)
Display := Day_Time (0) + ' ; ' + Day_Time (1);
(* El formato del texto a visualizar es: 'YYYY/MM/DD ; HH:MM:SS' *)

(* Equivalencia IL: Primero se llama a day_time(1) *)
LD          1
DAY_TIME
```

```
ST          hour_str    (* resultado intermedio *)
LD          0
DAY_TIME
ADD         ' ; '
ADD         hour_str
ST          Display
```

ARCREATE

Argumentos:

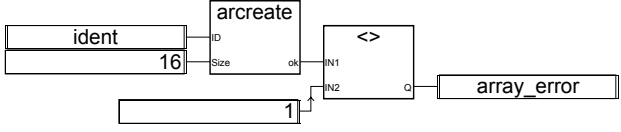
ID	INT	identificador del vector (tiene que estar en el conjunto [0..15])
Size	INT	número de elementos en el vector
ok	INT	estado de ejecución : **1** = si el vector ha sido creado con éxito **2** = identificador de vector no válido o vector ya creado **3** = tamaño no válido **4** = memoria insuficiente

Descripción:

Creación de una vector de enteros.

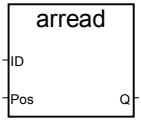
Advertencia: Existe un máximo de **16** vectores en una aplicación. Los vectores contienen valores **analógicos enteros**. Al llevarse a cabo la asignación dinámica de memoria, esta función puede provocar un error de sistema si el tamaño del vector es demasiado próximo al tamaño de la memoria disponible.

(* Programa FBD creando un vector de enteros *)




```
(* Equivalencia ST: *)
array_error := (ARCREATE (ident, 16) <> 1));

(* Equivalencia IL: *)
LD          ident
ARCREATE    16
NE          1
ST          array_error
```

ARREAD



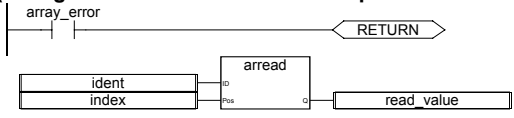
Argumentos:

ID	INT	identificador del vector (tiene que estar en el conjunto [0..15])
Pos	INT	la posición del elemento en el vector tiene que estar en el conjunto [0 .. size-1]
value	INT	valor del elemento leído 0 si los argumentos no son válidos

Descripción:

Lee un elemento en el vector de enteros.

(* Programa FBD utilizando los Bloques de Gestión de Vectores*)




```
(* Equivalencia ST: *)
If (array_error) Then Return; End_If;
read_value := ARREAD (ident, index);
(* array_error procede de la llamada a ARCREATE *)

(* Equivalencia IL: *)
LD          array_error
RETC
LD          ident
ARREAD      index
ST          read_value
```

ARWRITE

Argumentos:

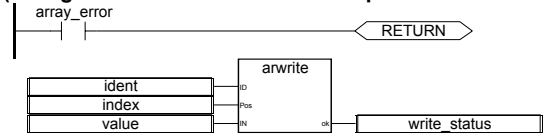
ID	INT	identificador del vector (tiene que estar en el conjunto [0..15])
Pos	INT	la posición del elemento en el vector

		tiene que estar en el conjunto [0 .. size-1]
IN	INT	nuevo valor del elemento
ok	INT	estado de ejecución **1** = éxito en el intento de escritura **2** = identificador de vector no válido **3** = índice no válido

Descripción:

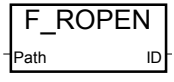
Almacena (escribe) un valor en la vector de enteros.

(* Programa FBD utilizando los Bloques de Gestión de Vectores*)




```
(* Equivalencia ST: *)
If (array_error) Then Return; End_If;
write_status := ARWRITE (Ident, Index, value);
(* array_error proviene de la llamada a ARCREATE *)

(* Equivalencia IL: *)
LD          array_error
RETC
LD          ident
ARWRITE     index,value
ST          write_status
```

F_ROPEN



Argumentos:

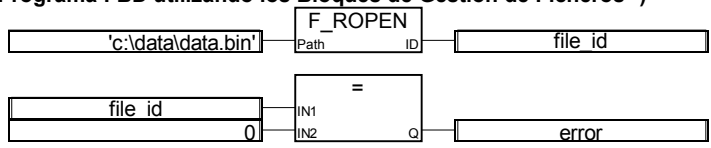
Path	MSG	nombre de fichero puede incluirse la ruta de acceso al fichero por medio de los símbolos \ o / para especificar un directorio. Para facilitar la portabilidad de las aplicaciones ambos símbolos son equivalentes.
ID	INT	número de fichero 0 si se produce un error: fichero no existe.

Descripción:

Abre un fichero binario en modo lectura. Para ser utilizado con FX_READ y F_CLOSE.

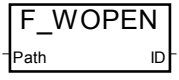
Esta función no se incluye en el simulador ISaGRAF.

(* Programa FBD utilizando los Bloques de Gestión de Ficheros *)




```
(* Equivalencia ST: *)
file_id := F_ROPEN('c:\data \data.bin');
error := (file_id=0);

(* Equivalencia IL: *)
LD          'c:\data\data.bin'
F_ROPEN
ST          file_id
EQ          0
ST          error
```

F_WOPEN



Argumentos:

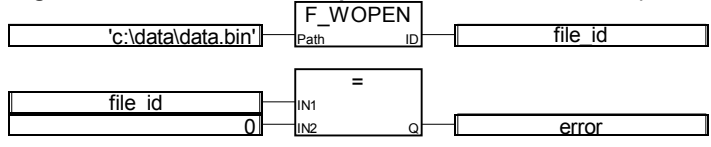
Path	MSG	nombre de fichero puede incluirse la ruta de acceso al fichero por medio de los símbolos \ o / para especificar un directorio. Para facilitar la portabilidad de las aplicaciones ambos símbolos son equivalentes.
ID	INT	número de fichero 0 si se produce un error. Si el fichero ya existe, se sobreescribe.

Descripción:

Abre un fichero binario en modo escritura. Para ser utilizado con FX_WRITE y F_CLOSE.

Esta función no se incluye en el simulador ISaGRAF.

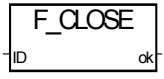
(* Programa FBD utilizando los Bloques de Gestión de Ficheros *)




```
(* Equivalencia ST: *)
file_id := F_WOPEN('c:\data \data.bin');
error := (file_id=0);
```

```
(* Equivalencia IL: *)
LD              'c:\data\data.bin'
F_WOPEN
ST              file_id
EQ              0
ST              error
```

F_CLOSE



Argumentos:

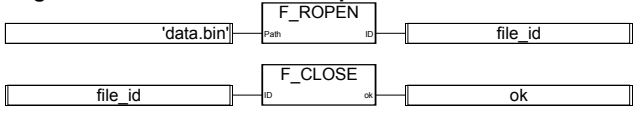
ID	INT	número de fichero: devuelto por F_ROPEN o F_WOPEN.
ok	BOO	estado de retorno VERDADERO si el cierre de fichero es OK FALSO si se produjo un error

Descripción:

Cierra un fichero binario que previamente se ha abierto con las funciones F_ROPEN o F_WOPEN.

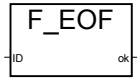
Esta función no se incluye en el simulador ISaGRAF.

(* Programa FBD utilizando los Bloques de Gestión de Ficheros *)




```
(* Equivalencia ST: *)
file_id := F_ROPEN('data.bin');
ok := F_CLOSE(file_id);

(* Equivalencia IL: *)
LD              'data.bin'
F_ROPEN
ST              file_id
F_CLOSE                 (* file_id ya está en el resultado IL actual *)
ST              ok
```

F_EOF



Argumentos:

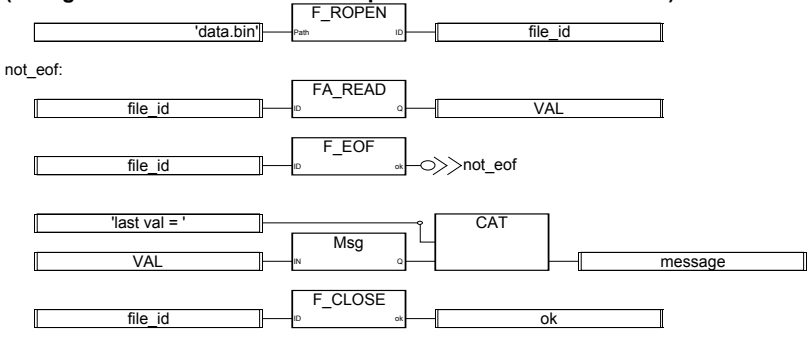
ID	INT	número de fichero: devuelto por F_ROPEN o F_WOPEN.
ok	BOO	indicador de final de fichero VERDADERO si se llegó al final del fichero en la última invocación del procedimiento de lectura o escritura. Con FM_READ, el último mensaje que se ha leido de un fichero pudiera ser incorrecto, si el último carácter no es un terminador de cadenas.

Descripción:

Comprueba si se ha llegado al final del fichero.

Esta función no se incluye en el simulador ISaGRAF.

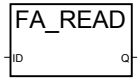
(* Programa FBD utilizando los Bloques de Gestión de Ficheros *)




```
(* Equivalencia ST: *)
file_id := F_ROPEN('data.bin');
WHILE not(F_EOF(file_id))
        VAL := FA_READ(file_id);
END_WHILE;
MESSAGE := 'last val = ' + msg(VAL);
ok := F_CLOSE(file_id);

(* Equivalencia IL: *)
                LD      'data.bin'
                F_ROPEN
                ST      file_id
                LD      file_id
                F_EOF
                JMPC    END_OF_FILE
NOT_EOF:        LD      file_id
                FA_READ
                ST      VAL
```

```
                LD          file_id
                F_EOF
                JMPNC       NOT_EOF  (* si no final de fichero, se continúa leyendo *)
END_OF_FILE:    LD          VAL
                MSG
                ST          val_msg     (* conversión de VAL en un mensaje *)
                LD          'last val = '
                ADD         val_msg
                ST          MESSAGE
                LD          file_id
                F_CLOSE
                ST          ok
```

FA_READ



Argumentos:

ID	INT	número de fichero: devuelto por F_ROPEN.
Q	INT	valor analógico entero leido del fichero

Descripción:

Lee variables ANALÓGICAS de ficheros binarios. Para ser utilizado con F_ROPEN y F_CLOSE.

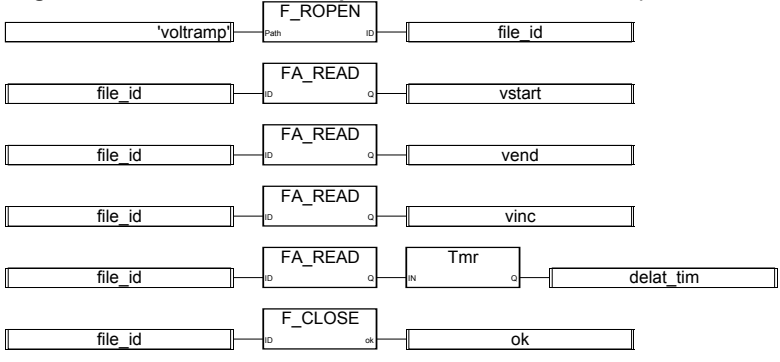
Este procedimiento lleva a cabo un acceso secuencial al fichero desde la posición previa.

La primera llamada después de F_ROPEN lee los primeros 4 bytes del fichero, y cada llamada avanza el puntero de lectura.

Para comprobar si se ha llegado al final del fichero, debe utilizarse F_EOF.

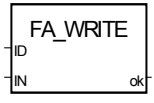
Esta función no se incluye en el simulador ISaGRAF.

(* Programa FBD utilizando los Bloques de Gestión de Ficheros *)



```
(* Equivalencia ST: *)
file_id := F_ROPEN('voltramp.bin');
vstart := FA_READ(file_id);
vend := FA_READ(file_id);
vinc := FA_READ(file_id);
delta_tim := tmr(FA_READ(file_id));
ok := F_CLOSE(file_id);

(* Equivalencia IL: *)
        LD      'voltramp.bin'
        F_ROPEN
        ST      file_id
        FA_READ             (* leer vstart *)
        ST      vstart
        LD      file_id
        FA_READ             (* leer vend *)
        ST      vend
        LD      file_id
        FA_READ             (* leer vinc *)
        ST      vinc
        LD      file_id
        FA_READ             (* leer delta_tim *)
        TMR                 (* conversión a un temporizador *)
        ST      delta_tim
        LD      file_id
        F_CLOSE
        ST      ok
```

FA_WRITE



Argumentos:

ID	INT	número de fichero: devuelto por F_WOPEN.
IN	INT	valor analógico entero. Para escribirse en el fichero
OK	BOO	estado de ejecución: VERDADERO si OK

Descripción:

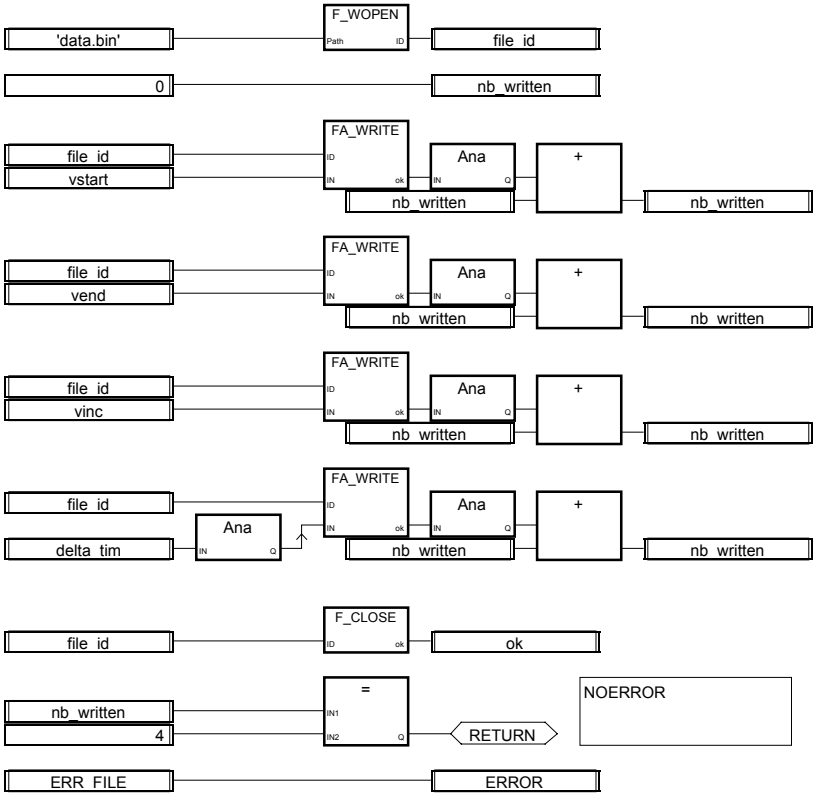
Escribe variables ANALÓGICAS a un fichero binario.

Con este procedimiento, se realiza un acceso secuencial al fichero desde la posición anterior.

La primera llamada después de F_WOPEN escribe los primeros 4 bytes del fichero, y cada llamada avanza el puntero de escritura.

Esta función no se incluye en el simulador ISaGRAF.

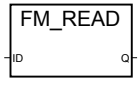
(* Programa FBD *)




```
(* Equivalencia ST: *)
file_id := F_WOPEN('voltramp.bin');
nb_written  := 0;
nb_written := nb_written + ana(FA_WRITE(file_id,vstart));
nb_written := nb_written + ana(FA_WRITE(file_id,vend));
nb_written := nb_written + ana(FA_WRITE(file_id,vinc));
nb_written := nb_written + ana(FA_WRITE(file_id,ana(delta_tim)));
ok := F_CLOSE(file_id);
SI ( nb_written <> 4) THEN
        ERROR := ERR_FILE;
END_SI;

(* Equivalencia IL: *)
                LD          'voltramp.bin'
                F_ROPEN
                ST          file_id
```

```
LD          0
ST          nb_written
LD          file_id       (* escribir vstart *)
FA_WRITE    vstart
ANA
ADD         nb_written
ST          nb_written
LD          file_id       (* escribir vend *)
FA_WRITE    vend
ANA
ADD         nb_written
ST          nb_written
LD          file_id       (* escribir vinc *)
FA_WRITE    vinc
ANA
ADD         nb_written
LD                        (* escribir delta_tim *)
ANA                       (* convertirlo en un entero *)
ST          ana_delta_tim
LD          file_id
FA_WRITE    ana_delta_tim
ANA
ADD         nb_written
ST          nb_written
F_CLOSE
ST          ok
LD          nb_written
EQ          4
RETC                      (* retorno si igual a 4 *)
LD          ERR_FILE      (* en otro caso error *)
ST          ERROR
```

FM_READ



Argumentos:

ID	INT	número de fichero: devuelto por F_ROPEN.
Q	MSG	valor de mensaje leido del fichero

Descripción:

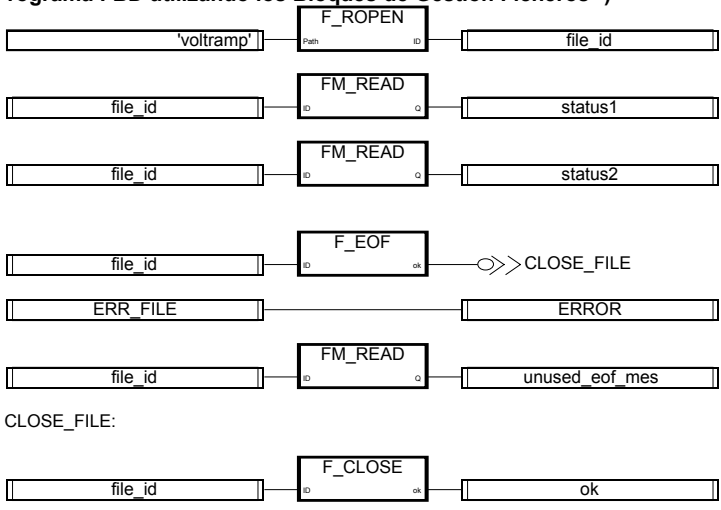
Lee variables de MENSAJE de un fichero binario.
Para ser utilizado con F_ROPEN y F_CLOSE.

Con este procedimiento, se realiza un acceso secuencial al fichero desde la posición anterior.

La primera llamada después de F_ROPEN lee la primera cadena del fichero, y cada llamada avanza el puntero de lectura.

Las cadenas finalizan con cero (0), fin de renglón ('\n') o retorno ('\r');
Para comprobar si se ha alcanzado el final del fichero, se utiliza F_EOF.
Esta función no se incluye en el simulador ISaGRAF.

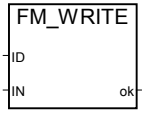
(* Programa FBD utilizando los Bloques de Gestión Ficheros *)




```
(* Equivalencia ST: *)
file_id := F_ROPEN('voltramp.bin');
status1 := FM_READ(file_id);
status2 := FM_READ(file_id);
IF (F_EOF(file_id)) THEN
        ERROR := ERR_FILE;
        unused_eof_mes := FM_READ(file_id);
END_IF;
ok := F_CLOSE(file_id);

(* Equivalencia IL: *)
        LD      'voltramp.bin'
        F_ROPEN
        ST      file_id
        FM_READ             (* leer status1 *)
        ST      status1
        LD      file_id
        FM_READ             (* leer status2 *)
        ST      status2
        LD      file_id
        F_EOF
        JMPNC   CLOSE_FILE (* si final de fichero no saltar *)
        LD      ERR_FILE
        ST      ERROR
```

```
            LD          file_id
            FM_READ             (* leer unused_eof_mes *)
            ST          unused_eof_mes
CLOSE_FILE  LD          file_id
            F_CLOSE
            ST          ok
```

FM_WRITE



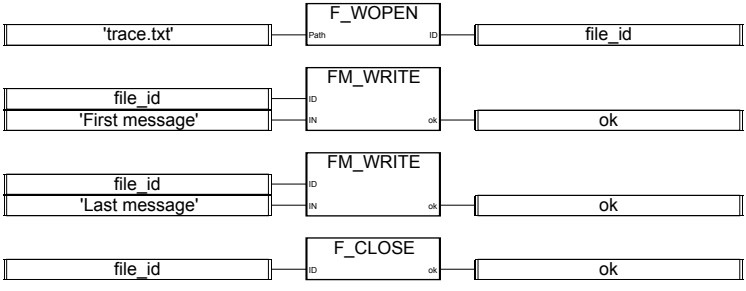
Argumentos:

ID	INT	número de fichero: devuelto por F_WOPEN.
IN	MSG	valor de mensaje a escribir en el fichero
ok	BOO	estado de ejecución VERDADERO si tiene éxito

Descripción:

Escribe variables de MENSAJE a un fichero binario.
Para ser utilizado con F_WOPEN y F_CLOSE.
Se escribe un mensaje en el fichero como una cadena terminada en nulo.
Con este procedimiento, se realiza un acceso secuencial al fichero desde la posición anterior.
La primera llamada después de F_WOPEN escribe la primera cadena del fichero, y cada llamada avanza el puntero de escritura.
Esta función no se incluye en el simulador ISaGRAF.

(* Programa FBD utilizando los Bloques de Gestión de Ficheros *)




```
(* Equivalencia ST: *)
file_id := F_WOPEN('trace.txt');
ok := FM_WRITE(file_id,'Primer mensaje');
ok := FM_WRITE(file_id,'Ultimo mensaje');
```

```
ok := F_CLOSE(file_id);

(* Equivalencia IL: *)
                LD          'trace.txt'
                F_WOPEN
                ST          file_id
                FM_WRITE‘Primer mensaje’      (* escribe el primer msg *)
                ST          ok
                LD          file_id
                FM_WRITE‘Ultimo mensaje’      (* escribe el segundo msg *)
                ST          ok
                LD          file_id
                F_CLOSE
                ST          ok
```

C. Manual de usuario del sistema objeto

C.1 Introducción

El objeto ISaGRAF es un software en tiempo real que ejecuta una aplicación ISaGRAF en un sistema computador industrial o una tarjeta, de acuerdo con el siguiente esquema bien conocido:

El ciclo objeto consiste en el muestreo de las entradas físicas del proceso que se va a ejecutar, procesándose los datos de la aplicación de acuerdo con los programas de aplicación del banco de trabajo ISaGRAF y después realizando la actualización de las salidas físicas.

- La primera parte de esta sección explica como se empieza a funcionar con un sistema objeto específico. Se tratarán los objetos DOS, OS-9, VxWorks y NT, respectivamente. Para cada objeto, se explicará la manera de ejecutar el objeto ISaGRAF en primer lugar. Posteriormente, se facilitará información sobre sus características específicas, como por ejemplo: arranque del objeto al iniciar el sistema, gestión de errores, comportamiento en general, etc.

- La segunda parte está dedicada al método de implementación de las funciones C, los bloques de función C y las funciones de conversión C del usuario para realzar el objeto ISaGRAF.

- La tercera parte facilita información sobre el Modbus y la implementación en ISaGRAF. Describe el formato de *tramas* de los diversos códigos de funciones.

La cuarta parte proporciona algunas herramientas para la gestión de caídas de tensión y el rearranque del objeto.

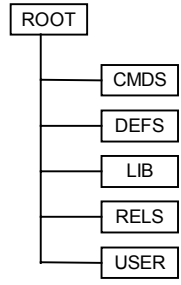
C.2 Instalación

La instalación necesita alrededor de 1 MB de espacio libre en el disco duro del ordenador.

El fichero *install.bat* que se incluye en el disco se encarga de instalar todos los ficheros que se necesitan para una plataforma determinada en un PC.

Ejemplo: a:\install a: c:*path*
instalará los ficheros de la unidad de disco a: al directorio *path* de c:.

Se utiliza la siguiente arquitectura de directorios:



el directorio raíz ROOT contiene algunas herramientas y ficheros 'readme'
el directorio CMDS contiene ficheros ejecutables
el directorio DEFS contiene ficheros de definición de cabeceras
el directorio LIB contiene librerías
el directorio RELS contiene ficheros reubicables (objeto)
el directorio USER contiene los procedimientos 'C' del usuario para las funciones, bloques de función y funciones de conversión C (ficheros fuente y cabecera)

El siguiente paso es el de empezar a trabajar con la plataforma instalada.

C.3 Primeros pasos con el objeto DOS de ISaGRAF

C.3.1 Ejecución de ISaGRAF: ISA.EXE

En la implementación MS-DOS, el objeto funciona como un único programa: ISA.EXE
Para empezar a trabajar, basta con ejecutar el comandos de ayuda **isa -?** desde el directorio CMDS.

En un sistema de este tipo, las operaciones pueden jugar un papel crítico. Por ejemplo, se recomienda no sobrecargar el enlace de comunicaciones para asegurar un buen rendimiento del sistema.
El programa objeto no impide la ejecución de rutinas accionadas por interrupciones.

Enlace y configuración de comunicaciones: Opción -t

El objeto ISaGRAF utiliza un enlace serie para la comunicación con el depurador. Se especifica el nombre del puerto con la opción -t. Ya que la interfaz de comunicaciones fue diseñada para ser compatible con cualquier máquina, se pueden utilizar los puertos COM1, COM2 o COM3, según la versión del BIOS.

Sin valor por defecto: Si no se utiliza esta opción, no se podrá comunicar con el objeto. En este caso, podría mostrarse el error número 7.

No se dispone de la comunicación mediante enlace Ethernet con el objeto DOS de ISaGRAF. Consultar al proveedor la posibilidad de una implementación especial.

Se tiene que configurar los parámetros de comunicación antes de ejecutar ISaGRAF, para que el usuario pueda tener una libertad absoluta para utilizar los parámetros que necesite. Al utilizar el depurador del banco de trabajo, el usuario deberá asegurarse de que los parámetros de comunicación del banco de trabajo (véase Manual del Usuario: Gestión de Programas) coincidan con los parámetros objeto.

Ejemplo:
MODE COM1:9600,N,8,1
Configura los parámetros de comunicación con los siguientes valores:
- Velocidad de línea 9600 baudios
- Sin control de paridad
- 8 bits de datos
- 1 bit de parada

Observe que en algunas versiones de BIOS, no se autoriza la configuración por defecto del banco de trabajo a 19200 baudios.

CJ suministra la utilidad ISAMOD.EXE para configurar los parámetros del banco de trabajo:
ISAMOD COM1
es equivalente a MODE COM1:19200,N,8,1

Número de esclavo: Opción -s

Esta opción especifica el número de esclavo del objeto. Puede ser cualquier número entre el 1 y el 255, con la excepción del 13 ($0D). Este número de esclavo es utilizado por el protocolo del enlace de comunicación. Está diseñado principalmente para diferenciar entre esclavos cuando se han conectado juntos más de un objeto. Al utilizar el depurador del banco de trabajo, el usuario deberá asegurarse de que el parámetro de esclavo del banco de trabajo (véase Manual del Usuario: Gestión de Programas) coincide con el parámetro objeto.

Valor por defecto: el número de esclavo por defecto es el 1 (igual que el del banco de trabajo)

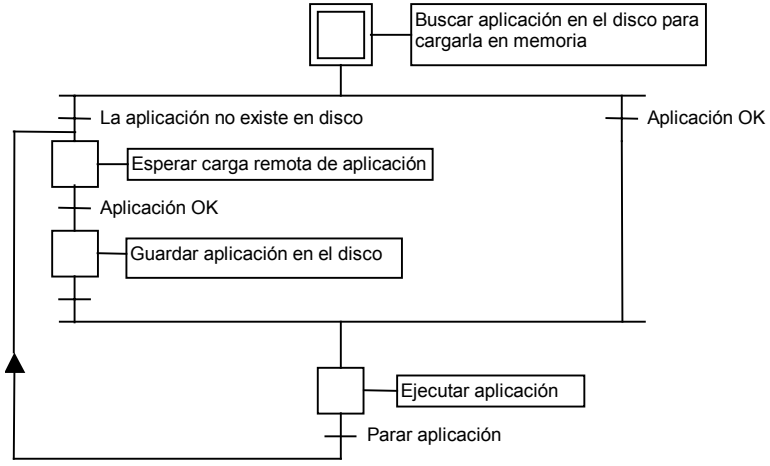
Ejemplos:

isamod COM1	Configurar COM1 a 19200 baudios, sin paridad, 8 bits datos, 1 bit parada.
isa -t=COM1	Arranca el objeto ISaGRAF con el número de esclavo por defecto (1) y con COM1 como el puerto de comunicaciones.
isa -s=3 -t=COM1	Arranca el objeto ISaGRAF con el número de esclavo 3 y con COM1 como el puerto de comunicaciones.

C.3.2 Características específicas

Arranque de ISaGRAF

Cuando se arranca el objeto, se ejecuta el siguiente algoritmo:



• Definiciones

El código de aplicación es la base de datos binaria que fue generada y cargada por el banco de trabajo, y posteriormente ejecutada por el objeto. Se puede completar con la tabla de símbolos.

La tabla de símbolos de la aplicación es una base de datos ASCII generada y cargada por el banco de trabajo. Esta tabla crea el enlace entre los objetos simbólicos y los objetos internos del objeto. No es necesaria en el objeto salvo para la gestión específica de símbolos por parte del usuario. Para mayor información sobre la tabla de símbolos, véase "Manual del Usuario: Técnicas Avanzadas de Programación".

• Copia de seguridad de la aplicación

Cuando se carga una aplicación nueva desde el depurador del banco de trabajo hasta el objeto, se guarda el código de aplicación en el directorio actual del objeto con el nombre de fichero:

ISAx1 fichero de copia de seguridad de código de aplicación (donde x es el número de esclavo)

Además, si ya se hubiera cargado la tabla de símbolos de la aplicación, también se guarda en el directorio actual del objeto con el nombre de fichero:

ISAx6 fichero de copia de seguridad de símbolos de aplicación (donde x es el número de esclavo)

Cuando se arranca el objeto ISaGRAF, se realiza la búsqueda de estos ficheros de código y símbolos de la aplicación en el directorio actual y se cargan en memoria.
Si no se dispone del fichero de símbolos, el objeto arrancará con el código de la aplicación, sin cargar los símbolos.

Si no se dispone del código de la aplicación, el objeto se pondrá a la espera para poder cargar una aplicación.

Para inicializar el objeto con una aplicación específica en el momento del arranque, sin utilizar el enlace con el depurador, se pueden copiar estos ficheros directamente al disco del directorio actual del objeto, si el banco de trabajo está en el mismo PC, o por medio de un disquete. Si el equipo objeto carece de disco, se puede utilizar un disco virtual.

Si el banco de trabajo ISaGRAF está instalado en el directorio estándar \ISAWIN:
el fichero del código de aplicación del proyecto MYPROJ es.
\ISAWIN\APL\MYPROJ\appli.x8m
el fichero de símbolos de la aplicación del proyecto MYPROJ es:
\ISAWIN\APL\MYPROJ\appli.tst

Ejemplo:
Desde el directorio en el cual está instalado isa.exe, si se introduce el siguiente comando:
copy \ISAWIN\APL\MYPROJ\appli.x8m isa11
isa.exe encontrará y ejecutará la aplicación 'myproj'.

Todos estos comandos pueden ser agrupados, por ejemplo, en un fichero de comandos que posteriormente podrá ser ejecutado desde el menú de herramientas del banco de trabajo (Véase "Manual del Usuario: Gestión de Programas).

Gestión de errores y mensajes de salida

El *software* objeto de ISaGRAF incorpora una función de detección y gestión de errores. La relación de avisos de error y sus descripciones aparece en el apéndice.

La detección de errores se procesa de la siguiente manera:
- Un error está compuesto por un número de error y de argumento que se remite a la rutina de errores de ISaGRAF.
- Si el señalizador de detección de errores está configurado en las opciones Ensamblar del banco de trabajo, se procesa el error. Si no lo está, se pierde la información y se finaliza la gestión de errores.

Si se procesa el error:
- Se muestran el número de error (valor decimal) y el argumento (valor hexadecimal) en la salida por defecto 'stdout'
- El número y argumento del error pasan a un *buffer* o memoria intermedia de errores FIFO, configurada en anillo, para su recuperación en un momento posterior. Se establece el tamaño del *buffer* de errores en las opciones Ensamblar del banco de trabajo. Cuando el *buffer* está lleno, al registrarse cada nuevo error se pierde el de mayor antigüedad.
- Los errores pueden ser extraídos bien del depurador o bien de la aplicación activa por medio de la llamada SYSTEM (véase el Manual del Usuario).

Cuando el depurador detecta un error, aparece en la pantalla de errores un mensaje que lo describe. Dependiendo del contexto de la aplicación (activa o no), el depurador podrá mostrar el nombre del objeto (variable o programa) del cual procede el error, o el argumento del error (valor decimal) entre paréntesis [x], que posee un significado diferente para cada error.

Aparecen en la salida por defecto 'stdout' un mensaje de bienvenida y los valores de los errores cuando arranca el objeto y cuando se detecta un error. Si se desea que no aparezca

esta representación en el canal de salida estándar, se puede utilizar un comando de redireccionamiento como:

isa -t=COM1 -s=1 **>NUL**

Reloj del sistema

Ya que el objeto ISaGRAF está diseñado para operar en cualquier sistema, la referencia horaria que se utiliza tanto para la sincronización de ciclos como para el refresco de las variables horarias es el tick estándar, que dura unos 55 milisegundos.

En consecuencia, no se puede obtener una precisión mejor que 55 ms en las variables horarias. Por el mismo motivo, un tiempo de ciclo especificado que sea inferior o igual a 55 ms, y diferente a cero, provocará un error de desbordamiento de tiempo de ciclo (error 62) y ningún ciclo activado.

La ventaja de no modificar el tick del sistema es que cualquiera de la aplicaciones residentes, o funciones y bloques de función C que estén integradas en la aplicación, jamás se verán perjudicadas por la ejecución de ISaGRAF.

El usuario deberá solicitar una implementación especial al proveedor si su aplicación requiere una mayor precisión.

Tecla de salida

Al probar una aplicación en condiciones no industriales, en un PC de sobremesa, el usuario podría tener la necesidad de interrumpir la ejecución de ISaGRAF. Esto se logra mediante la pulsación de una combinación compleja de teclas que evita las paradas inesperadas. Esta secuencia de teclas es:

shift + ctrl + alt

Claro está que si no se desea que la aplicación industrial se interrumpa al pulsar una tecla, se deberán tomar las medidas oportunas para deshabilitar estas combinaciones.

Uno de los efectos secundarios peligrosos de estas 'salidas rápidas' es que no se cierra la interfaz de la tarjeta de E/S. Por lo tanto, para finalizar el objeto ISaGRAF de la manera correcta:
- parar la aplicación desde el depurador (así se cerrarán las tarjetas de E/S)
- parar el objeto ISaGRAF desde el teclado

Tamaño de la aplicación

El objeto MS-DOS de ISaGRAF está diseñado para funcionar en el modo real de Intel, por lo que el tamaño máximo de una estructura de datos es de 64K. En consecuencia, el código de aplicación que se cargue del banco de trabajo no debe superar este límite. En casos muy raros, la estructura interna asignada por ISaGRAF también puede superar este límite y provocar un error interno grave de la aplicación tras su carga. Además, el total de memoria disponible está limitado a los 640K de memoria convencional.

El usuario deberá solicitar una implementación especial al proveedor si su aplicación requiere una mayor capacidad de memoria.

C.4 Primeros pasos con el objeto OS-9 de ISaGRAF

En primer lugar, se tienen que transferir ficheros (al menos los ficheros ejecutables del directorio CMDS) al objeto OS-9 utilizando cualquier herramienta de transferencia de ficheros.

Para empezar a trabajar, basta con ejecutar los comandos de ayuda desde el directorio CMDS del sistema OS-9:

```
isa -?
isaker -?
isatst -?
Isanet -?
```

C.4.1 Ejecución de ISaGRAF en modo simple tarea: isa

Se puede ejecutar el objeto ISaGRAF en modo simple tarea. En una configuración de este tipo, las operaciones pueden jugar un papel crítico. Por ejemplo, se recomienda no sobrecargar el enlace de comunicaciones para asegurar un buen rendimiento del sistema. El sistema de multitarea OS-9 soporta la ejecución en la misma CPU de diversos objetos ISaGRAF en modo simple tarea, siempre que sus números de esclavo y puertos de comunicación sean diferentes.

La implementación en modo simple tarea ha sido diseñada principalmente para plataformas de *hardware* escasas, tales como tarjetas o PCs MS-DOS de bajo coste, o para realizar un primer prototipo al trabajar sobre una plataforma nueva. Por consiguiente, es preferible optar por la implementación de objetos en el modo multitarea de ISaGRAF.
El objeto ISaGRAF en modo simple tarea no impide la ejecución de procesos no prioritarios o rutinas accionadas por interrupciones.

Enlace y configuración de comunicaciones: Opción -t

El objeto ISaGRAF en modo simple tarea utiliza un enlace serie para la comunicación con el depurador. Se especifica el nombre del descriptor con la opción -t.

Sin valor por defecto: Si no se utiliza esta opción, no se podrá comunicar con el objeto. En este caso, podría mostrarse el error número 7.

No se dispone de la comunicación mediante enlace Ethernet con la implementación en modo simple tarea.

El dispositivo de enlace vía serie se abre en el modo de transferencia de datos binarios (sin caracteres de control, sin XON/XOFF). Los demás parámetros de comunicaciones tienen que estar configurados antes de ejecutarse ISaGRAF, para que el usuario puede tener una total

libertad de utilizar los parámetros que necesite. Al utilizar el depurador del banco de trabajo, el usuario deberá asegurarse de que los parámetros de comunicaciones del banco de trabajo (véase Manual del Usuario: Gestión de Programas) coinciden con los parámetros objeto.

Ejemplo:
xmode /t0 baud=19200
Establece la velocidad de línea de comunicación del dispositivo /t0 en 19200 baudios

Número de esclavo: Opción -s

Esta opción especifica el número de esclavo del objeto. Puede ser cualquier número entre el 1 y el 255, con la excepción del 13 ($0D). Este número de esclavo es utilizado por el protocolo del enlace de comunicación. Es necesario para poder diferenciar entre esclavos cuando más de un objeto está en funcionamiento. Al utilizar el depurador del banco de trabajo, el usuario deberá asegurarse de que el parámetro de esclavo del banco de trabajo (véase Manual del Usuario: Gestión de Programas) coincide con un objeto existente.

Valor por defecto: el número de esclavo por defecto es el 1 (igual que el del banco de trabajo)

Ejemplos:

isa -t=/t0 Arranca un objeto ISaGRAF en modo simple tarea, con el número de esclavo por defecto (1) y con /t0 como puerto de comunicaciones.

isa -s=3 -t=/t1 Arranca el objeto ISaGRAF en modo simple tarea, con el número de esclavo 3 y con /t1 como puerto de comunicaciones.

isa -t=/t0 &
isa -s=3 -t=/t1 Arranca dos objetos ISaGRAF en modo simple tarea. Uno con el número de esclavo por defecto (1) y con /t0 como puerto de comunicaciones. El otro, con el número de esclavo 3 y con /t1 como puerto de comunicaciones.

C.4.2 Ejecución de multitareas ISaGRAF: isaker, isatst, isanet

Para mejorar los tiempos de respuesta del *kernel* del objeto ISaGRAF y del enlace de comunicación, se divide el objeto en dos tareas que separan el trabajo de comunicación (tareas de comunicación isatst o isanet) de la ejecución de la aplicación (tarea *kernel* isaker). Este tipo de arquitectura es más flexible. Le permite al usuario ejecutar más de una tarea de comunicación vinculada a la misma tarea *kernel*, o ejecutar hasta 4 *kernels* con la misma tarea de comunicación. Esto facilita algunas integraciones, como un enlace de visualización de procesos y el enlace del depurador del banco de trabajo en la misma aplicación, o en un único enlace hasta 4 aplicaciones diferentes a través del mismo puerto físico.

Las tareas del *kernel* y de comunicaciones son independientes y pueden bifurcarse (*fork*)por separado. El único requisito es que la(s) tarea(s) del *kernel* se ejecute(n) en primer lugar para que se inicialice el entorno de sistema y la(s) tarea(s) de comunicaciones puedan establecer los enlaces correspondientes.

El objeto multitarea ISaGRAF no impide la ejecución de procesos no prioritarios o rutinas accionadas por interrupciones.

C.4.2.1 Ejecución de la tarea del *kernel*: isaker

Número de esclavo: Opción -s

Esta opción especifica el número de esclavo del *kernel* objeto. Puede ser cualquier número entre el 1 y el 255, con la excepción del 13 ($0D). Este número de esclavo es utilizado por el protocolo del enlace de comunicación y por parte de la(s) tarea(s) de comunicación que esté(n) enlazada(s) con el *kernel*. Es necesario para poder diferenciar entre esclavos cuando más de un objeto está en funcionamiento.

Valor por defecto: el número de esclavo por defecto es el 1 (igual que el del banco de trabajo).

C.4.2.2 Ejecución de la tarea de comunicación serie: isatst

Enlace y configuración de comunicaciones: Opción -t

La tarea de comunicación serie del objeto, 'isatst', utiliza un enlace serie para la comunicación con el depurador. Se especifica el nombre del descriptor con la opción -t.

Sin valor por defecto: Si no se utiliza esta opción, no se podrá comunicar con el objeto. En este caso, podría mostrarse el error número 7.

No se dispone de la comunicación mediante enlace Ethernet con la implementación de la tarea isatst.

El dispositivo de enlace vía serie se abre en el modo de transferencia de datos binarios (sin caracteres de control, sin XON/XOFF). Los demás parámetros de comunicaciones tienen que estar configurados antes de ejecutarse ISaGRAF, para que el usuario pueda tener una total libertad de utilizar los parámetros que necesite. Al utilizar el depurador del banco de trabajo, el usuario deberá asegurarse de que los parámetros de comunicaciones del banco de trabajo (véase Manual del Usuario: Gestión de Programas) coinciden con los parámetros objeto.

Ejemplo:
xmode /t0 baud=19200
Establece la velocidad de línea de comunicación del dispositivo /t0 en 19200 baudios

Número de esclavo: Opción -s

Esta opción especifica el número o los números de esclavo a los que está vinculada la tarea de comunicación. Puede ser cualquier número entre el 1 y el 255, con la excepción del 13 ($0D). Se puede repetir esta operación hasta 4 veces para vincular a un máximo de 4 esclavos *kernel* diferentes. Este número de esclavo es utilizado por el protocolo del enlace de comunicación. Es necesario para poder diferenciar entre esclavos cuando más de un objeto está en funcionamiento. Al utilizar el depurador del banco de trabajo, el usuario deberá asegurarse de que el parámetro de esclavo del banco de trabajo (véase Manual del Usuario:

Gestión de Programas) coincide con un objeto existente (tareas del *kernel* y de comunicaciones).

Valor por defecto: el número de esclavo por defecto es el 1 (igual que el del banco de trabajo)

Número lógico de la tarea de comunicación: Opción -c

Esta opción especifica el número lógico de la tarea de comunicación. Se utiliza para gestionar más de una tarea de comunicación a la vez. Puede ser cualquier número entre el 1 y el 255, y se tiene que utilizar uno diferente para cada tarea de comunicación.

Valor por defecto: Se utiliza la última opción -s especificada. El valor por defecto asegura la compatibilidad con las versiones previas (3.0) de ISaGRAF.

C.4.2.3 Ejecución de la tarea de comunicación Ethernet: isanet

Enlace y configuración de comunicación: Opción -t

La tarea de comunicación del objeto, 'isanet', utiliza un enlace Ethernet estándar para la comunicación con el depurador. Se especifica el nombre del puerto con la opción -t.

Sin valor por defecto: Si no se utiliza esta opción, no se podrá comunicar con el objeto. En este caso, podría mostrarse el error número 7.

Al utilizar el depurador del banco de trabajo, el usuario deberá asegurarse de que los parámetros de comunicación del banco de trabajo (véase Manual del Usuario: Gestión de Programas) coinciden con los parámetros objeto.

Para ISaGRAF, el objeto OS-9 representa el servidor y el depurador es el cliente que se conecta al número de puerto especificado.
Antes de comenzar la primera sesión de depuración vía Ethernet, el usuario deberá cerciorarse de que su dispositivo Ethernet OS-9 está correctamente configurado. Puede enviar un 'ping' al sistema OS-9, por ejemplo.

Número de esclavo: Opción -s

Esta opción especifica el número o los números de esclavo a los que está vinculada la tarea de comunicación. Puede ser cualquier número entre el 1 y el 255, con la excepción del 13 ($0D). Se puede repetir esta operación hasta 4 veces para vincular a un máximo de 4 esclavos *kernel* diferentes. Este número de esclavo es utilizado por el protocolo del enlace de comunicación. Es necesario para poder diferenciar entre esclavos cuando más de un objeto está en funcionamiento. Al utilizar el depurador del banco de trabajo, el usuario deberá asegurarse de que el parámetro de esclavo del banco de trabajo (véase Manual del Usuario: Gestión de Programas) coincide con un objeto existente (tareas del *kernel* y de comunicaciones).

Valor por defecto: el número de esclavo por defecto es el 1 (igual que el del banco de trabajo).

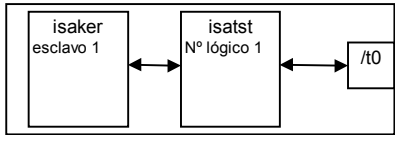
Número lógico de la tarea de comunicación: Opción -c

Esta opción especifica el número lógico de la tarea de comunicación. Se utiliza para gestionar más de una tarea de comunicación a la vez. Puede ser cualquier número entre el 1 y el 255, y se tiene que utilizar uno diferente para cada tarea de comunicación.

Valor por defecto: Se utiliza la última opción -s especificada. El valor por defecto asegura la compatibilidad con las versiones previas (3.0) de ISaGRAF.

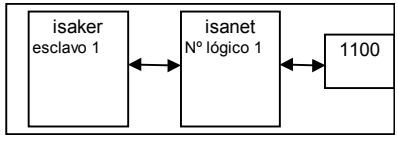
C.4.2.4 Ejemplos:

isaker &
isatst -t=/t0



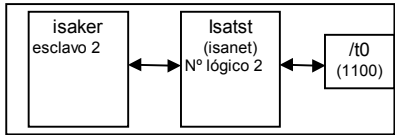
Arranca:
Una tarea del *kernel* ISaGRAF con el número de esclavo por defecto (1).
Una tarea de comunicación vía serie de ISaGRAF, en el puerto de comunicaciones /t0, enlazada con el número de esclavo por defecto (1) y con el número lógico por defecto (el último número de esclavo especificado = defecto = 1).

isaker &
isanet -t=1100



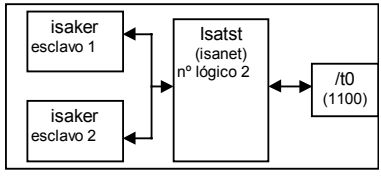
Arranca:
Una tarea del *kernel* ISaGRAF con el número de esclavo por defecto (1).
Una tarea de comunicación vía Ethernet de ISaGRAF, en el puerto número 1100, enlazada con el número de esclavo por defecto (1) y con el número lógico por defecto (el último número de esclavo especificado = defecto = 1).

isaker -s=2 &
isatst -t=/t0 -s=2 (isanet -t=1100 -s=2, respectivamente)



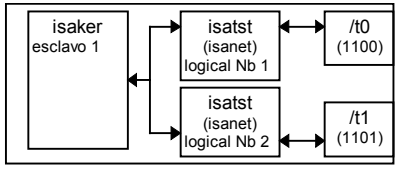
Arranca:
Una tarea del *kernel* ISaGRAF con el número de esclavo 2.
Una tarea de comunicación vía serie (Ethernet) de ISaGRAF, en el puerto de comunicaciones /t0 (Puerto número 1100), enlazada con el número de esclavo 2 y con el número lógico por defecto (el último número de esclavo especificado = 2).

Isaker -s=1 &
isaker -s=2 &
isatst -t=/t0 -s=1 -s=2 (isanet -t=1100 -s=1 -s=2, respectivamente)



Arranca:
Una tarea del *kernel* ISaGRAF con el número de esclavo 1.
Una tarea del *kernel* ISaGRAF con el número de esclavo 2.
Una tarea de comunicación vía serie (Ethernet) de ISaGRAF, en el puerto de comunicaciones /t0 (Puerto número 1100), enlazada con los números de esclavo 1 y 2 y con el número lógico por defecto (el último número de esclavo especificado = 2).

Isaker -s=1 &
isatst -t=/t0 -s=1 -c=1 & (isanet -t=1100 -s=1 -c=1 &, respectivamente)
isatst -t=/t1 -s=1 -c=2 (isanet -t=1101 -s=1 -c=2, respectivamente)



Arranca:
Una tarea del *kernel* ISaGRAF con el número de esclavo 1.
Una tarea de comunicación vía serie (Ethernet) de ISaGRAF, en el puerto de comunicaciones /t0 (Puerto número 1100), enlazada con el esclavo nº 1 y el número lógico 1.
Una tarea de comunicación vía serie (Ethernet) de ISaGRAF, en el puerto de comunicaciones /t1 (Puerto número 1101), enlazada con el esclavo nº 1 y el número lógico 2.

Nota:
Se pueden combinar las tareas de comunicaciones serie y Ethernet.

C.4.3 Características específicas

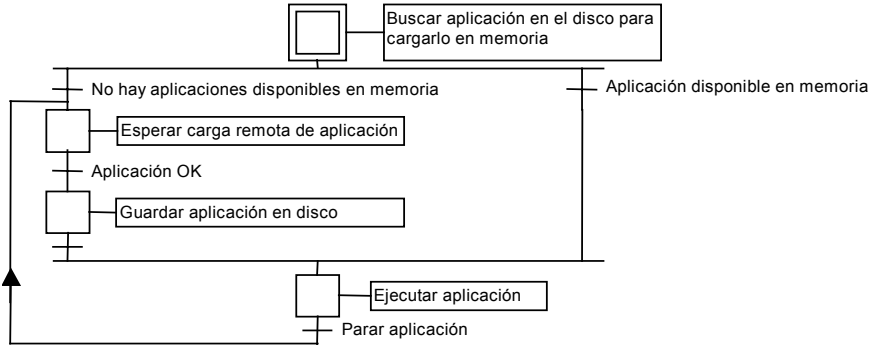
- ***Enlaces de comunicación***

El Gestor de Caracteres Serie de OS-9 es altamente flexible, por lo que permite la utilización de casi cualquier dispositivo físico bidireccional soportado por Microware :

Ejemplo:
El enlace serie puede ser una ruta de red a un puerto físico ubicado en otra CPU.
En este supuesto, se podría utilizar la opción -t de la siguiente manera: -t=/nr/MASTER/t0 donde se deporta el enlace de comunicación a una CPU denominada MASTER en una red *ramnet*. Se utiliza el puerto físico /t0.

- ***Arranque de ISaGRAF***

Cuando se arranca el objeto, se ejecuta el siguiente algoritmo:



• Definiciones

El código de aplicación es la base de datos binaria que fue generada y cargada por el banco de trabajo, y posteriormente ejecutada por el objeto. Se puede completar con la tabla de símbolos.
La tabla de símbolos de la aplicación es una base de datos ASCII que generada y cargada por el banco de trabajo. Esta tabla crea el enlace entre los objetos simbólicos y los objetos internos del objeto. No es necesaria en el objeto salvo para la gestión específica de símbolos por parte del usuario. Para mayor información sobre la tabla de símbolos, véase “Manual del Usuario: Técnicas Avanzadas de Programación”.

• Objetos y multiaplicación ISaGRAF OS-9

Todos los nombres de objetos públicos de ISaGRAF comienzan por '**ISAxn**', donde **x** es el número del esclavo *kernel* y **n** es un número de espacio con un significado específico, con la excepción de **ISAy3**, donde **y** es el número lógico de la tarea de comunicación en la implementación de multitareas.
Se pueden ejecutar diferentes aplicaciones (tareas de *kernel* y de comunicación) simultáneamente en una CPU, siempre que tengan diferentes números de esclavo y diferentes números lógicos de tarea de comunicación. No obstante, al ejecutar aplicaciones diferentes, el usuario debe tener cautela con determinados objetos de aplicación con acceso compartido, como las tarjetas de E/S. Por ejemplo, aplicaciones diferentes (*kernel*s) pueden utilizar tarjetas físicas distintas al no ser que se implemente algún tipo de servidor o semáforo de E/S a través del controlador de E/S.

Nombres de objeto OS-9:
Ficheros de disco:

ISAx1 fichero de copia de seguridad del código de aplicación ISaGRAF
ISAx6 fichero de copia de seguridad de símbolos de aplicación ISaGRAF

Módulos de memoria:

ISAx0 datos de sistema del *kernel* de ISaGRAF
ISAx1 código de aplicación ISaGRAF
ISAx2 base de datos en tiempo real del *kernel* ISaGRAF
ISAy3 *buffer* de intercambio de datos de comunicación ISaGRAF
ISAx4 código de aplicación para modificación en línea 1 de ISaGRAF
ISAx5 código de aplicación para modificación en línea 2 de ISaGRAF
ISAx6 símbolo de aplicación ISaGRAF

El usuario debe tener cuidado con no utilizar los mismos nombres de objeto.

• Copia de seguridad de la aplicación

Cuando se carga una aplicación nueva desde el depurador del banco de trabajo hasta el objeto, se guarda el código de aplicación en el directorio actual del objeto con el nombre de fichero:

ISAx1 fichero de copia de seguridad de código de aplicación (donde x es el número de esclavo)

Además, si ya se hubiera cargado la tabla de símbolos de la aplicación, también se guarda en el directorio actual del objeto con el nombre de fichero:

ISAx6 fichero de copia de seguridad de símbolos de aplicación (donde x es el número de esclavo)

Cuando se arranca el objeto ISaGRAF, se realiza la búsqueda de estos ficheros de código y símbolos de la aplicación en el directorio actual y se cargan en memoria como módulos de datos con los mismos nombres.

Si no se dispone del fichero de símbolos en memoria, el objeto arrancará con el código de la aplicación, sin cargar los símbolos.

Si no se dispone del código de la aplicación en memoria, el objeto se pondrá a la espera para poder cargar una aplicación.

Para inicializar el objeto con una aplicación específica en el momento del arranque, sin utilizar el enlace con el depurador:

- La primera manera consiste en copiar estos ficheros directamente al disco del directorio actual del objeto desde el PC principal en el cual está instalado el banco de trabajo, utilizando para ello cualquier herramienta de transferencia. Se puede utilizar el menú de herramientas del banco de trabajo (véase "Manual del Usuario: Gestión de Programas") para facilitar estas maniobras.

- Una segunda manera consiste en guardar el código de aplicación (y, si fuera necesario, la tabla de símbolos de la aplicación) en una memoria no volátil (como una PROM o una EPROM), procedente de ficheros del PC principal en el cual está instalado el banco de trabajo y utilizando herramientas propias.

  Posteriormente, en el momento de arrancar el sistema y si fuera preciso (por ejemplo, debido a un acceso más rápido o a la gestión de *breakpoints*[1]), el usuario puede cargar el código de aplicación (y si fuera necesario, la tabla de símbolos de la aplicación) desde la PROM hasta la RAM como módulo(s) de memoria **ISAx1** (y **ISAx6**, si fuera necesario) con sus propias herramientas.

  ADVERTENCIA:
  El gestor de breakpoints del depurador ISaGRAF no funcionará correctamente si no se puede acceder al módulo de código de aplicación para escribir. Esto no representa un problema, ya que normalmente la aplicación habrá sido sometida a una completa prueba previa.

En el ordenador principal, si el banco de trabajo ISaGRAF está instalado en el directorio estándar \ISAWIN:
el fichero de código de aplicación del proyecto MYPROJ es:
\ISAWIN\APL\MYPROJ\appli.x6m (correspondiente a isax1 en el objeto).
el fichero de símbolos de aplicación del proyecto MYPROJ es:
\ISAWIN\APL\MYPROJ\appli.tst (correspondiente a isax6 en el objeto).

Gestión de errores y mensajes de salida

El *software* objeto de ISaGRAF incorpora una función de detección y gestión de errores. La relación de avisos de error y sus descripciones aparece en el apéndice.

La detección de errores se procesa de la siguiente manera:
- Un error está compuesto por un número de error y de argumento que se remite a la rutina de errores de ISaGRAF.
- Si el señalizador de detección de errores está configurado en las opciones Ensamblar del banco de trabajo, se procesa el error. Si no lo está, se pierde la información y se finaliza la gestión de errores.

Si se procesa el error:
- Se muestran el número de error (valor decimal) y el argumento (valor hexadecimal) en la salida por defecto 'stdout'
- El número y argumento del error pasan a un *buffer* o memoria intermedia de errores FIFO, configurada en anillo, para su recuperación en un momento posterior. Se establece

[1] Punto de ruptura

el tamaño del *buffer* de errores en las opciones Ensamblar del banco de trabajo. Cuando el *buffer* está lleno, al registrarse cada nuevo error se pierde el de mayor antigüedad.

- Los errores pueden ser extraídos bien del depurador o bien de la aplicación activa por medio de la llamada SYSTEM (véase el Manual del Usuario).

Cuando el depurador detecta un error, aparece en la pantalla de errores un mensaje que lo describe. Dependiendo del contexto de la aplicación (activa o no), el depurador podrá mostrar el nombre del objeto (variable o programa) del cual procede el error, o el argumento del error (valor decimal) entre paréntesis [x], que posee un significado diferente para cada error.

Aparecen en la salida por defecto 'stdout' un mensaje de bienvenida y los valores de los errores cuando arranca el objeto y cuando se detecta un error. Si se desea que no aparezca esta representación en el canal de salida estándar, se puede utilizar un comando de redireccionamiento como:

nombre_programa [opciones] **>>>/nil**

Duración de ciclo, comportamiento de tareas y prioridades de tareas

- Al término de un ciclo ISaGRAF, justo antes de comenzar un ciclo nuevo, se ejecuta el siguiente algoritmo:

  Si se ha especificado un tiempo de ciclo (desde el banco de trabajo: véase el "Manual del Usuario: Gestión de Programas"), se cede la CPU durante el periodo de tiempo que queda (tiempo de ciclo especificado – tiempo de ciclo actual de la aplicación). Si este periodo de tiempo restante es negativo, se genera un desbordamiento y se cede la CPU durante 1 tick para forzar la entrada del gestor de tareas.

  Si no se ha especificado un tiempo de ciclo, o si el tiempo restante es inferior o igual a 1 tick o igual a cero, se renuncia a la CPU durante 1 tick para forzar la entrada del gestor de tareas.

  La precisión horaria del objeto corresponde a la del tick uno del sistema OS-9.

  Normalmente se escoge un tiempo específico de ciclo bien para "disparar" los ciclos de ejecución o bien para ceder la CPU a otras tareas que estén ejecutándose en el sistema OS-9.

- La tarea de comunicaciones permanece en estado 'dormida' cuando no entran datos a través del enlace de comunicaciones. Cuando sea preciso, esta tarea obtiene información de la aplicación activa mediante el uso de un protocolo de pregunta/respuesta, con la tarea del *kernel*. La tarea de comunicaciones le solicita una pregunta al *kernel*. Al final del ciclo (para tener una imagen síncrona de la aplicación), el *kernel* le facilita la respuesta a la tarea de comunicaciones.

Las tareas de ISaGRAF no modifican la prioridad que se les asigna. El usuario tiene libertad para ajustar estas prioridades de acuerdo con el comportamiento de las tareas ISaGRAF y los requisitos de la aplicación en su conjunto.
Por ejemplo, para asegurarse de que una tarea de baja prioridad no se adueña de ISaGRAF, se pueden modificar determinados parámetros de gestión de tareas de OS-9, como **MIN_AGE** y **MAX_AGE**.

Modo terminal

El protocolo de comunicación serie del objeto reconoce una secuencia de tres caracteres de retorno de carro ($0D) y después inicia una tarea del *shell* OS-9, si está disponible, en el dispositivo asociado al enlace serie.
Esto permite obtener el *prompt* del *shell* OS-9 en cualquier terminal, utilizando el enlace serie del objeto ISaGRAF.

Ejemplo:
Desde el PC principal (*host*):
- Cerrar el depurador ISaGRAF.
- Iniciar una sesión de Terminal de Windows (grupo Accesorios) con los parámetros de comunicaciones apropiados.
- Introducir 3 retornos de carro.

Ahora, se está de alta en un *shell* OS-9.
- Teclear **logout** para salir del modo terminal.

ADVERTENCIA:
Es imprescindible salir de la sesión en modo terminal de la forma correcta, utilizando únicamente la orden 'logout'. En caso contrario, fallará la siguiente conexión con el banco de trabajo.

C.5 Primeros pasos con el objeto VxWorks de ISaGRAF

Para ejecutar el(los) objeto(s) ISaGRAF, primero hay que ejecutar varios comandos en el sistema VxWorks para establecer el entorno de configuración y, en último término, crear el(los) objeto(s) ISaGRAF. Pueden iniciarse todos estos comandos desde un fichero de texto. Se describen en los siguientes apartados.

C.5.1 Gestor de recursos del sistema: isassr.o

Siempre se necesita este módulo en cualquier configuración del objeto ISaGRAF, y debe ser el primer módulo que se cargue en el objeto. Permite al gestor de recursos del sistema hacerse cargo de la ejecución de múltiples objetos.

C.5.2 Características comunes de isa.o, isakerse.o y isakeret.o

Para ejecutar ISaGRAF, uno de los siguientes módulos puede estar cargado:

isa.o: permite el arranque de objetos ISaGRAF de monotarea (sólo enlace de comunicación serie).

Isakerse.o: permite el arranque de objetos ISaGRAF de multitarea (sólo enlace de comunicación serie).

Isakeret.o: permite el arranque de objetos ISaGRAF de multitarea (enlace de comunicación serie y/o vía Ethernet).

Se describen estos módulos en los siguientes apartados.

Configuración del enlace de comunicación serie

Básicamente, el objeto ISaGRAF utiliza un enlace serie para la comunicación con el depurador. Cuando está abierto este enlace, el objeto ISaGRAF no puede realizar la configuración de los dispositivos asociados al enlace serie. Así, el usuario tiene absoluta libertad para utilizar los parámetros que necesite. No obstante, se necesita el modo de transferencia de datos binarios (modo RAW). Para ello, se proporciona la subrutina *ISAMOD ()*.

```
uchar ISAMOD
  (
  char *desc,        /* Nombre de dispositivo serie */
  uint32 baudrate    /* Velocidad en baudios        */
  )
```

Descripción:

Configura un dispositivo de enlace serie para la transferencia de datos binarios, con una velocidad de línea determinada (en baudios).

Valor de retorno:
0 si tiene éxito, BAD_RET si se produce algún error

Al utilizar el depurador del banco de trabajo, hay que asegurarse de que los parámetros de comunicación del banco de trabajo (véase "Manual del Usuario: Gestión de Programas") coinciden con los parámetros del objeto.

- ***Frecuencia base del reloj de sistema***

Se tiene que inicializar la variable global CLKRATE (uint32) de acuerdo con la frecuencia base del reloj del sistema VxWorks. De esta manera, se puede utilizar:

```
CLKRATE = sysClkRateGet ()
```

El valor por defecto de CLKRATE es de 60 Hz.

C.5.3 Ejecución de ISaGRAF en modo simple tarea: isa.o

Se puede ejecutar el objeto ISaGRAF en modo simple tarea. En una configuración de este tipo, las operaciones pueden jugar un papel crítico. Por ejemplo, se recomienda no sobrecargar el enlace de comunicaciones para asegurar un buen rendimiento del sistema. El sistema de multitarea VxWorks soporta la ejecución en la misma CPU de diversos objetos ISaGRAF en modo simple tarea, siempre que sus números de esclavo y puertos de comunicación sean diferentes.
La implementación en modo simple tarea ha sido diseñada principalmente para plataformas de *hardware* escasas, tales como tarjetas o PCs MS-DOS de bajo coste, o para realizar un primer prototipo al trabajar sobre una plataforma nueva. Por consiguiente, es preferible optar por la implementación de objetos en el modo multitarea de ISaGRAF.
El objeto ISaGRAF en modo simple tarea no impide la ejecución de procesos no prioritarios o rutinas accionadas por interrupciones.

- ***Registro de esclavo(s)***

Los objetos ISaGRAF se caracterizan por sus números de esclavo. Su valor puede ser cualquier número entre el 1 y el 255, con la excepción del 13 ($0D). Este número de esclavo es utilizado por el protocolo del enlace de comunicación. Es necesario para diferenciar entre esclavos cuando se está ejecutando más de un objeto. Por consiguiente, antes de arrancar una tarea ISaGRAF, tiene que estar registrada. Para ello, se proporciona la subrutina *isa_register_Esclavo()*.

```
uchar isa_register_Esclavo
  (
  uchar Esclavo     /* Número de esclavo */
  )
```

Descripción:
Añade un nuevo registro de esclavo al sistema de gestión de objetos múltiples.

Valor de retorno:
0 si tiene éxito, BAD_RET si se produce algún error

Unidad de almacenamiento de copias de seguridad de la aplicación

Se puede inicializar la variable global TSK_FUNIT (char *) con una cadena que contenga la ruta de la unidad en la que se van a realizar las copias de seguridad de los ficheros de la aplicación. El objeto ISaGRAF simplemente utiliza las rutinas estándares de gestión de ficheros fopen, fread, fwrite, fclose para realizar las copias de seguridad de los ficheros de la aplicación.
El valor por defecto es una cadena vacía ("") que especifica que no existen unidades de almacenamiento.

Ejemplo:
TSK_FUNIT = "*host name:*/C:/ISaGRAF/target/apl/"
Especifica ISaGRAF\target\apl\, en la raíz de la unidad C: del PC denominado *host_name*, como el directorio de copias de seguridad de los ficheros de la aplicación. No debe olvidarse la barra inclinada final, ya que en caso contrario se realizarán las copias de seguridad en el directorio ISaGRAF\target\ con los nombres de fichero preestablecidos por la aplicación.

Si fuera necesario, se puede configurar esta variable para unidades con rutas diferentes, para cada objeto, antes de cada creación.
Para mayor información sobre las copias de seguridad de la aplicación, véase la sección “Características específicas” en el apartado dedicado a las copias de seguridad de la aplicación.

Control de final de ciclo

Se puede configurar la variable TSK_NBTCKSCHED (uint 32) con un valor que especifica una demora en ticks, utilizada por el objeto ISaGRAF al final del ciclo.
El valor por defecto es 0 (programación de tarea con igual prioridad).
Si fuera necesario, se puede configurar esta variable con valores diferentes, para cada objeto que haya que arrancar, antes de cada creación.
Para mayor información de características específicas, véase el apartado dedicado a duración de ciclos, comportamiento de tareas y prioridad de tareas.

Creación de objetos ISaGRAF

Una vez que se haya establecido el entorno de configuración, el último paso consiste en crear el(los) objeto(s) ISaGRAF: isa_main.

```
uchar isa_main
  (
  uchar Esclavo,    /* Número de esclavo */
  char *com         /* Nombre de dispositivo serie */
  )
```

Descripción:
Arranca una tarea del objeto ISaGRAF.

Valor de retorno:
retorna un valor diferente a cero si se produce algún error.

El número de esclavo es el mismo que se ha tratado en el apartado dedicado al registro de esclavos.

Se puede arrancar más de un esclavo siempre que sus números de esclavo y puertos de comunicación sean diferentes.
Al utilizar el depurador del banco de trabajo, hay que asegurarse de que los parámetros del esclavo del banco de trabajo (véase “Manual del Usuario: Gestión de Programas”) coinciden con los de un objeto existente.

Ejemplo

Este ejemplo muestra la manera de arrancar un objeto ISaGRAF en modo monotarea con el número de esclavo 1 y el dispositivo /tyCo/1 para el enlace serie.
El directorio *anfitrión* actual es aquél en el que está instalado el objeto.

Cargar de módulo isassr.o
ld < RELS/isassr.o

Cargar de módulo isa.o
ld < CMDS/isa.o

Configuración de comunicación serie
ISAMOD ("/tyCo/1", 19200)

Frecuencia base reloj del sistema
CLKRATE = sysClkRateGet ()

Registro de esclavo
isa_register_Esclavo (1)

Unidad almacenamiento ficheros (podría omitirse y usar la configuración por defecto)
TSK_FUNIT = ""

Control de final de ciclo (podría omitirse y usar la configuración por defecto)
TSK_NBTCKSCHED = 0

Creación de objeto ISaGRAF
sp (isa_main, 1, "/tyCo/1")

C.5.4 Ejecución de multitareas ISaGRAF: isakerse.o y isakeret.o

Para mejorar los tiempos de respuesta del *kernel* del objeto ISaGRAF y del enlace de comunicación, se divide el objeto en dos tareas que separan el trabajo de comunicación (tarea de comunicación) de la ejecución de la aplicación (tarea *kernel*).
Este tipo de arquitectura es más flexible. Le permite al usuario ejecutar más de una tarea de comunicación vinculada a la misma tarea *kernel*, o ejecutar hasta 4 *kernels* con la misma tarea de comunicación. Esto facilita algunas integraciones, como un enlace de visualización de procesos y el enlace del depurador del banco de trabajo en la misma aplicación, o en un único enlace hasta 4 aplicaciones diferentes a través del mismo puerto físico.

Las tareas del *kernel* y de comunicaciones son independientes y pueden crearse por separado. El único requisito es que la(s) tarea(s) del *kernel* se ejecuten en primer lugar para que se inicialice el entorno de sistema y la(s) tarea(s) de comunicaciones puedan establecer los enlaces correspondientes.

El objeto multitarea ISaGRAF no impide la ejecución de procesos no prioritarios o rutinas accionadas por interrupciones.

Se proponen dos módulos, en función de la capacidad del *hardware* de comunicaciones:
- *Kernel* y enlace serie: isakerse.o
Este módulo permite arrancar las tareas del *kernel* y de comunicación serie.
- *Kernel,* enlace serie y/o enlace Ethernet: isakeret.o
Este módulo permite arrancar las tareas del *kernel* y de comunicación serie y/o vía Ethernet.

El modo de arrancar ISaGRAF es el mismo para los módulos isakerse.o e isakeret.o. Como excepción en el caso de isakeret.o, se puede especificar bien un nombre de dispositivo serie o bien un número de puerto de enlace Ethernet como el parámetro de nombre del dispositivo de comunicación al arrancar la(s) tarea(s) de comunicación de ISaGRAF: tst_main_ex (véase información más adelante).

Para ISaGRAF, el objeto VxWorks representa el servidor y el depurador es el cliente que se conecta al número de puerto especificado.

Registro de kernel(s)

Los *kernels* ISaGRAF se caracterizan por sus números de esclavo. Su valor puede ser cualquier número entre el 1 y el 255, con la excepción del 13 ($0D). Este número de esclavo es utilizado por el protocolo del enlace de comunicación y por parte de la(s) tarea(s) de comunicación que está(n) vinculada(s) al *kernel*. Es necesario para diferenciar entre esclavos cuando se está ejecutando más de un objeto. Por consiguiente, antes de arrancar una tarea *kernel* ISaGRAF, tiene que estar registrada. Para ello, se proporciona la subrutina *isa_register_Esclavo()*.

```
uchar isa_register_Esclavo
  (
  uchar Esclavo     /* Número de esclavo */
  )
```

Descripción:
Añade un nuevo registro de esclavo *kernel* al sistema de gestión de objetos múltiples.

Valor de retorno:
0 si tiene éxito, BAD_RET si se produce algún error

Registro de tareas de comunicación

Las tareas de comunicación de ISaGRAF se caracterizan por sus números lógicos. Se utilizan para poder gestionar más de una tarea de comunicación a la vez. Puede ser cualquiera entre el 1 y el 255, y tiene que ser diferente para cada tarea de comunicación. Por consiguiente, antes de arrancar una tarea de comunicación ISaGRAF, tiene que estar registrada. Para ello, se proporciona la subrutina *isa_register_com()*.

```
uchar isa_register_com
```

```
(
uchar com_id      /* Identificador de tarea */
)
```

Descripción:
Añade un nuevo registro de tarea de comunicación al sistema de gestión de objetos múltiples.

Valor de retorno:
0 si tiene éxito, BAD_RET si se produce algún error

Unidad de almacenamiento de copias de seguridad de la aplicación

Se puede inicializar la variable global TSK_FUNIT (char *) con una cadena que contenga la ruta de la unidad en la que se van a realizar las copias de seguridad de los ficheros de la aplicación. El objeto ISaGRAF simplemente utiliza las rutinas estándares de gestión de ficheros fopen, fread, fwrite, fclose para realizar las copias de seguridad de los ficheros de la aplicación.
El valor por defecto es una cadena vacía ("") que especifica que no existen unidades de almacenamiento.

Ejemplo:
TSK_FUNIT = "*host_name:*/C:/ISaGRAF/target/apl/"
Especifica ISaGRAF\target\apl\, en la raíz de la unidad C: del PC denominado *host_name*, como el directorio de copias de seguridad de los ficheros de la aplicación. No debe olvidarse la barra inclinada final, ya que en caso contrario se realizarán las copias de seguridad en el directorio ISaGRAF\target\ con los nombres de fichero preestablecidos por la aplicación.

Si fuera necesario, se puede configurar esta variable para unidades con rutas diferentes, para cada objeto, antes de cada creación de *kernel*.
Para mayor información sobre las copias de seguridad de la aplicación, véanse las características específicas en el apartado dedicado a las copias de seguridad de la aplicación.

Control de final de ciclo

Se puede configurar la variable TSK_NBTCKSCHED (uint 32) con un valor que especifica una demora en ticks, utilizada por el objeto ISaGRAF al final del ciclo.
El valor por defecto es 0 (programación de tarea con igual prioridad).
Si fuera necesario, se puede configurar esta variable con valores diferentes para cada *kernel* antes de que sean creados.
Para mayor información de características específicas, véase el apartado dedicado a duración de ciclos, comportamiento de tareas y prioridad de tareas.

Creación de kernels ISaGRAF

Una vez que se haya establecido el entorno de configuración, el último paso consiste en crear el o los *kernels* ISaGRAF: isa_main.

```
uchar isa_main
  (
  uchar Esclavo,    /* Número de esclavo        */
  char *com         /* NO USADO. Una cadena vacía es OK */
```

```
)
```

Descripción:
Arranca una tarea del *kernel* ISaGRAF

Valor de retorno:
retorna un valor diferente a cero si se produce algún error.

El número de esclavo es el mismo que se ha tratado en el apartado dedicado al registro de esclavos.
Se puede arrancar más de un *kernel* siempre que sus números de esclavo sean diferentes.

Creación de tareas de comunicación ISaGRAF

Una vez que se haya establecido el entorno de configuración, unos de los últimos pasos consiste en crear las tareas de comunicación ISaGRAF: tst_main_ex.

```
uchar tst_main_ex
  (
  char *com,        /* Nombre de dispositivo de comunicación */
  uchar *Esclavo,   /* Dirección de un campo de 4 bytes que especifica el (los)
                       esclavo(s) a enlazar */
  uchar com_id      /* Identificador de tarea de comunicaciones */
  )
```

Descripción:
Arranca una tarea de comunicación ISaGRAF

Valor de retorno:
retorna un valor diferente a cero si se produce algún error.

El campo de 4 Bytes especifica el (los) esclavo(s) de *kernel* al (a los) que está asociada la tarea de comunicación. Si se necesitan menos de 4 esclavos de *kernel*, se debe rellenar el campo con cero. Una vez que se inicie la tarea, este campo ya no es necesario.
El nombre del dispositivo de comunicación corresponde al nombre del dispositivo serie que se utilice para el enlace de comunicación.
Se puede arrancar más de una tarea de comunicación, siempre que sus identificadores de tareas sean diferentes.
Al utilizar el depurador del banco de trabajo, hay que asegurarse de que los parámetros de enlace de comunicación del banco de trabajo (véase “Manual del Usuario: Gestión de Programas”) coinciden con los de un objeto existente (tareas de *kernel* y de comunicación).

Ejemplo:

Este ejemplo muestra la manera de arrancar:
Una tarea del *kernel* ISaGRAF con el número de esclavo 1.
Una tarea de comunicación de ISaGRAF identificada por el número 1, asociada al esclavo de *kernel* nº 1 y con el dispositivo /tyCo/1 para el enlace serie.
Una tarea de comunicación de ISaGRAF identificada por el número 2, asociada al esclavo de *kernel* nº 1 y con el número de puerto 1100 para el enlace de comunicación vía Ethernet.
El directorio *anfitrión* actual es aquél en el que está instalado el objeto.

Carga de módulo isassr.o

ld < RELS/isassr.o

Carga de módulo isakeret.o (se puede cargar isakerse cuando no hace falta un enlace de comunicación vía Ethernet)
ld < CMDS/isakeret.o

Configuración de comunicación serie
ISAMOD ("/tyCo/1", 19200)

Frecuencia base reloj del sistema
CLKRATE = sysClkRateGet ()

Registro de esclavo
isa_register_Esclavo (1)

Registro de comunicaciones
isa_register_com (1)
isa_register_com (2)

Unidad almacenamiento ficheros (podría omitirse y usar la configuración por defecto)
TSK_FUNIT = ""

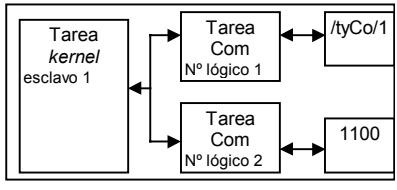
Control de final de ciclo (podría omitirse y usar la configuración por defecto)
TSK_NBTCKSCHED = 0

Creación del *kernel* ISaGRAF
sp (isa_main, 1, "")

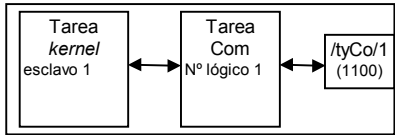
Tarea de comunicación, enlace esclavos
EsclavosLink = 0x01000000

Creación de tareas de comunicación de ISaGRAF
sp (tst_main_ex, "/tyCo/1", &EsclavosLink, 1)
sp (tst_main_ex, "1100", &EsclavosLink, 2)

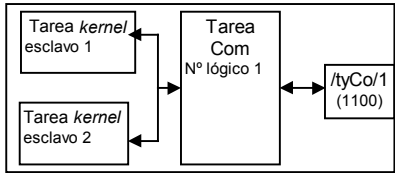
Este arranque corresponde a la siguiente figura:



También se puede elegir entre las siguientes configuraciones básicas:



La configuración más básica consiste en una tarea de *kernel* asociada a una tarea de comunicación en un enlace serie (Ethernet).

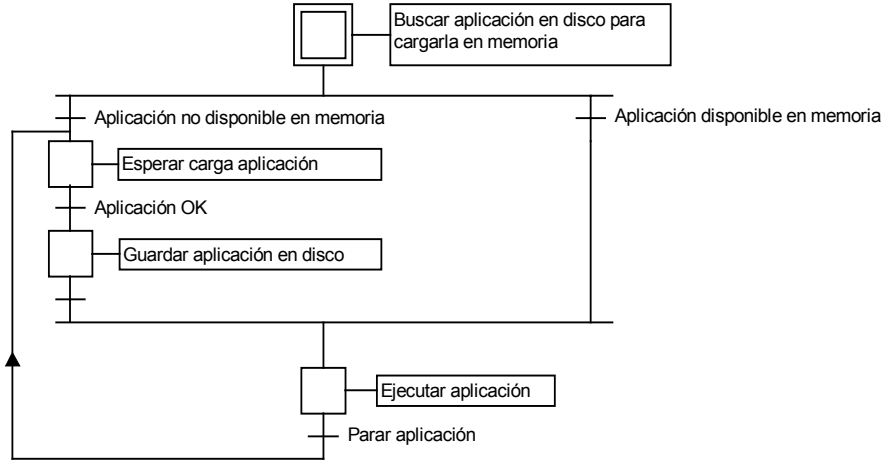


Otra configuración consiste en 2 *kernels* asociados a una tarea de comunicación en un enlace serie (Ethernet). En este caso, EsclavosLink = 0x01020000.

C.5.5 Características específicas

- ***Arranque de ISaGRAF***

Al arrancar el objeto, se ejecuta el siguiente algoritmo:



• Definiciones

El código de aplicación es la base de datos binaria que fue generada y cargada por el banco de trabajo, y posteriormente ejecutada por el objeto. Se puede completar con la tabla de símbolos.
La tabla de símbolos de la aplicación es una base de datos ASCII que genera y carga el banco de trabajo. Esta tabla crea el enlace entre los objetos simbólicos y los objetos internos del objeto. No es necesaria en el objeto salvo para la gestión específica de símbolos por parte del usuario. Para mayor información sobre la tabla de símbolos, véase “Manual del Usuario: Técnicas Avanzadas de Programación”.

Se especifica la ruta de la unidad de almacenamiento del disco en el momento de arrancar el objeto ISaGRAF, por medio de la variable global TSK_FUNIT (valor por defecto = "" para especificar que no existen unidades de almacenamiento en disco).

• Aplicaciones múltiples ISaGRAF

Se pueden ejecutar diferentes aplicaciones (tareas de *kernel* y de comunicación) simultáneamente en una CPU, siempre que tengan diferentes números de esclavo y diferentes números lógicos de tarea de comunicación. No obstante, al ejecutar aplicaciones diferentes, el usuario debe tener cautela con determinados objetos de aplicación con acceso compartido, como las tarjetas de E/S. Por ejemplo, aplicaciones diferentes (*kernels*) pueden utilizar tarjetas físicas distintas a no ser que se implemente algún tipo de servidor de E/S o semáforo a través del controlador de E/S.

• Copia de seguridad de la aplicación

Cuando se carga una nueva aplicación desde el depurador del banco de trabajo al objeto, se guarda el código de la aplicación (el objeto utiliza las rutinas estándares de gestión de ficheros fopen, etc.) con el nombre de fichero:

*ruta***ISAx1** fichero de copia de seguridad del código de aplicación ISaGRAF (donde x es el número de esclavo)

Además, si ya se hubiera cargado la tabla de símbolos de la aplicación, también se guarda en el directorio actual del objeto con el nombre de fichero:

*ruta***ISAx6** fichero de copia de seguridad de símbolos de la aplicación (donde x es el número de esclavo)

La *ruta* se especifica en el momento de arrancar el objeto ISaGRAF, utilizando la variable global TSK_FUNIT. Una cadena vacía ("") indicará que no existe ninguna unidad de almacenamiento en disco (valor por defecto).

Cuando se arranca el objeto ISaGRAF, se realiza la búsqueda de estos ficheros de código y símbolos de la aplicación en el directorio actual y se cargan en memoria.

Si no se dispone del fichero de símbolos en memoria, el objeto arrancará el código de la aplicación, sin cargar los símbolos.

Si no se dispone del código de la aplicación en memoria, el objeto se pondrá a la espera para poder cargar una aplicación.

Para inicializar el objeto con una aplicación específica en el momento del arranque, sin utilizar el enlace con el depurador:

- Una primera forma consiste en copiar estos ficheros directamente a la unidad de almacenamiento de copias de seguridad desde el PC principal donde está instalado el banco de trabajo, utilizando para ello cualquier herramienta de transferencia. Se puede utilizar el menú de “Herramientas” del banco de trabajo (véase “Manual del Usuario: Gestión de Programas”) para facilitar estas maniobras.

- Una segunda manera consiste en guardar el código de aplicación (y, si fuera necesario, la tabla de símbolos de la aplicación) en una memoria no volátil (como una PROM o una EPROM), procedente de ficheros del PC principal en el que está instalado el banco de trabajo, utilizando herramientas propias.

  Posteriormente, en el momento de arrancar el sistema y si fuera preciso (por ejemplo, debido a un acceso más rápido o a la gestión de breakpoints), el usuario puede cargar el código de aplicación (y si fuera necesario, la tabla de símbolos de la aplicación) desde la PROM hasta la RAM, utilizando sus propias herramientas.

  Después, al arrancarse ISaGRAF (justo antes de crear las tareas), se tiene que especificar la(s) dirección(es) de memoria en las que se encuentra el código de aplicación (y si fuera necesario, la tabla de símbolos de la aplicación). En este sentido, se tiene que inicializar la variable global SSR de la siguiente manera:

  SSR[x][1].space = *dirección de código de aplicación*

  Y si fuera necesario:

  SSR[x][6].space = *dirección de la tabla de símbolos de la aplicación*

  Para ello, se puede escribir un procedimiento corto. Se declara la variable global SSR como un tipo de estructura str_ssr, que está definido en el fichero tasy0ssr.h.

  ADVERTENCIA:

El gestor de breakpoints del depurador ISaGRAF no funcionará correctamente si no se puede acceder al módulo de código de aplicación para escribir. Esto no representa un problema, ya que normalmente la aplicación habrá sido sometida a una completa prueba previa.

En el PC principal, si el banco de trabajo ISaGRAF está instalado en el directorio estándar \ISAWIN:
el fichero de código de aplicación del proyecto MYPROJ es:
\ISAWIN\APL\MYPROJ\appli.x6m (correspondiente a isax1 en el objeto).
el fichero de símbolos de aplicación del proyecto MYPROJ es:
\ISAWIN\APL\MYPROJ\appli.tst (correspondiente a isax6 en el objeto).

Gestión de errores y mensajes de salida

El *software* objeto de ISaGRAF incorpora una función de detección y gestión de errores. La relación de avisos de error y sus descripciones aparece en el apéndice.

La detección de errores se procesa de la siguiente manera:
- Un error está compuesto por un número de error y de argumento que se remite a la rutina de errores de ISaGRAF.
- Si el señalizador de detección de errores está configurado en las opciones Ensamblar del banco de trabajo, se procesa el error. Si no lo está, se pierde la información y se finaliza la gestión de errores.

Si se procesa el error:
- Se muestran el número de error (valor decimal) y el argumento (valor hexadecimal) en la salida por defecto 'stdout'
- El número y argumento del error pasan a un *buffer* o memoria intermedia de errores FIFO, configurada en anillo, para su recuperación en un momento posterior. Se establece el tamaño del *buffer* de errores en las opciones Ensamblar del banco de trabajo. Cuando el *buffer* está lleno, al registrarse cada nuevo error se pierde el de mayor antigüedad.
- Los errores pueden ser extraídos bien del depurador o bien de la aplicación activa por medio de la llamada SYSTEM (véase el Manual del Usuario).

Cuando el depurador detecta un error, aparece en la pantalla de errores un mensaje que lo describe. Dependiendo del contexto de la aplicación (que esté activa o no), el depurador podrá mostrar el nombre del objeto (variable o programa) del que procede el error, o el argumento del error (valor decimal) entre paréntesis [x], que posee un significado diferente para cada error.

En el objeto, cuando se detecta un error se presentan los valores correspondientes en la salida por defecto 'stdout'. De esta manera, se puede dirigir el *display* por medio de rutinas de VxWorks tales como
ioGlobalStdSet()
o *ioTaskStdSet()*
En este último caso, observe que tanto las tareas del *kernel* como las de comunicaciones pueden generar errores.

Duración de ciclos, comportamiento de tareas y prioridades de tareas

- Al término de un ciclo ISaGRAF, justo antes de comenzar un ciclo nuevo, se ejecuta el siguiente algoritmo:
  Si se ha especificado un tiempo de ciclo (desde el banco de trabajo: véase el "Manual del Usuario: Gestión de Programas"), se renuncia a la CPU durante el periodo de tiempo que queda (tiempo de ciclo especificado – tiempo de ciclo actual de la aplicación). Si este periodo de tiempo restante es negativo, se genera un desbordamiento y se cede la CPU durante TSK_NBTCKSCHED (variable que se configura al arrancar ISaGRAF) ticks para forzar la entrada del gestor de tareas

  Si no se ha especificado un tiempo de ciclo, o si el tiempo restante es inferior o igual a 1 tick o igual a cero, se cede la CPU durante TSK_NBTCKSCHED ticks para forzar la entrada del gestor de tareas.

  La precisión horaria del objeto corresponde a un tick del sistema VxWorks.

  Se suelen utilizar tiempos de ciclo específicos para activar ciclos o para ceder la CPU a otras tareas que estén ejecutándose en el sistema VxWorks.

- La tarea de comunicaciones permanece en estado 'dormida' cuando no entran datos a través del enlace de comunicaciones. Cuando sea preciso, esta tarea obtiene información de la aplicación activa mediante el uso de un protocolo de pregunta/respuesta, con la tarea del *kernel*. La tarea de comunicaciones le solicita una pregunta al *kernel*. Al final del ciclo (para tener una imagen síncrona de la aplicación), el *kernel* le facilita la respuesta a la tarea de comunicaciones.

Las tareas de ISaGRAF no modifican la prioridad que se les asigna. El usuario tiene libertad para ajustar estas prioridades de acuerdo con el comportamiento de las tareas ISaGRAF descrito anteriormente y los requisitos de la aplicación en su conjunto.

C.6 Primeros pasos con el objeto NT de ISaGRAF

C.6.1 Ejecución de ISaGRAF

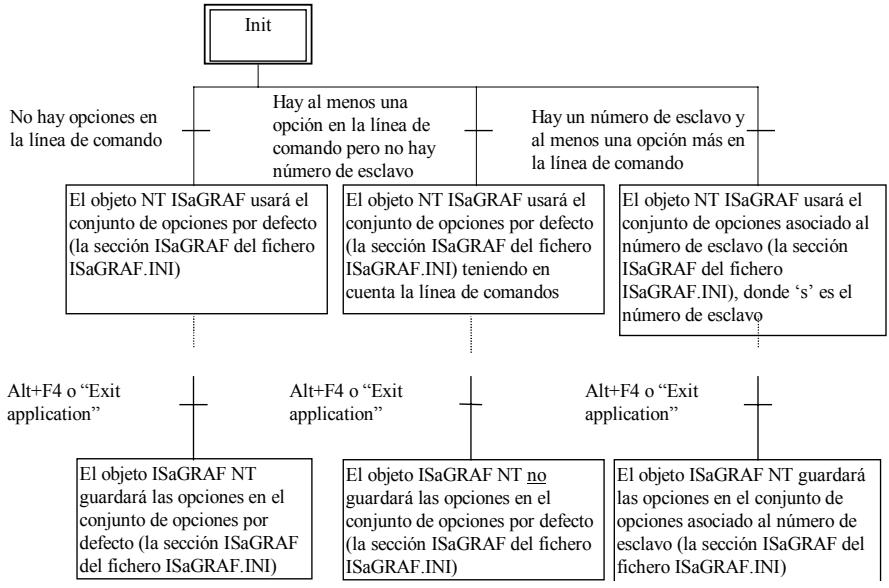
En la implementación NT, el objeto funciona como un único programa: WISAKER.EXE, que puede ejecutarse varias veces. Esto permite tener tantos objetos NT ISaGRAF como se desee, ya que cada instancia tiene un número de esclavo diferente.

El programa objeto no impide la ejecución de rutinas accionadas por interrupciones.

El *software* WISAKER está diseñado para operar bajo Windows NT 3.51 o versiones posteriores.

C.6.2 Información general sobre opciones

Las opciones de guardan y se recuperan de acuerdo con el siguiente diagrama:



Obsérvese que se guarda el fichero ISAGRAF.INI en el directorio actual de trabajo.

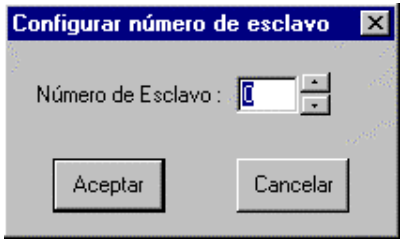
Número de esclavo: Opción -s

Esta opción especifica el número de esclavo del objeto. Puede ser cualquier número entre el 1 y el 255, con la excepción del 13 ($0D). Este número de esclavo es utilizado por el protocolo del enlace de comunicación. Está diseñado principalmente para diferenciar entre esclavos cuando se ha conectado más de un objeto al banco de trabajo del PC principal o cuando se está ejecutando más de un objeto en el mismo PC. Al utilizar el depurador del banco de trabajo, el usuario deberá asegurarse de que el parámetro de esclavo del banco de trabajo (véase Manual del Usuario: Gestión de Programas) coincide con el parámetro objeto.

Valor por defecto: el número de esclavo por defecto es el 1 o el que se indique en el fichero ISaGRAF.INI.

Ejemplo:
WISAKER.EXE -s=2

Interfaz del usuario: Se presenta esta ventana con el comando "**Opciones/Esclavo**" de la ventana principal del objeto NT ISaGRAF.



Por medio del ratón o de las flechas de 'arriba' y 'abajo', se puede cambiar el valor de esta opción. Para poder utilizar el valor nuevo, hay que reinicializar el objeto ISaGRAF NT.

Enlace y configuración de comunicaciones: Opción -t

El objeto ISaGRAF puede utilizar enlaces serie o vía Ethernet para la comunicación con el depurador.
Se especifica el nombre del puerto con la opción -t. Ya que la interfaz de comunicaciones fue diseñada para ser compatible con cualquier máquina, se pueden utilizar los puertos COM1, COM2, COM3 ó COM4 para la comunicación serie, y los puertos numerados a partir de 1100 para la comunicación vía Ethernet.

Valor por defecto: Los puertos por defecto son 1100 para la comunicación vía Ethernet y COM1 para la comunicación serie, o el puerto que se especifique en el fichero ISaGRAF.INI.

N.B.: El enlace de comunicación por defecto es el de Ethernet.

Ejemplos:
WISAKER -t=COM2
WISAKER -t=1101

Configuración serie:
Existen algunas opciones que sólo pueden utilizarse si se especifica la opción -t=COMx.

Las siguientes opciones de configuración son para el enlace serie:

Opción	Valores	Significado
baud	600 1200 2400 4800 9600 **19200**	Velocidad en baudios
parity	**n** e o	Sin paridad Par Impar
data	7 u **8**	Número de bits
stop	**1** ó 2	Longitud del bit de parada

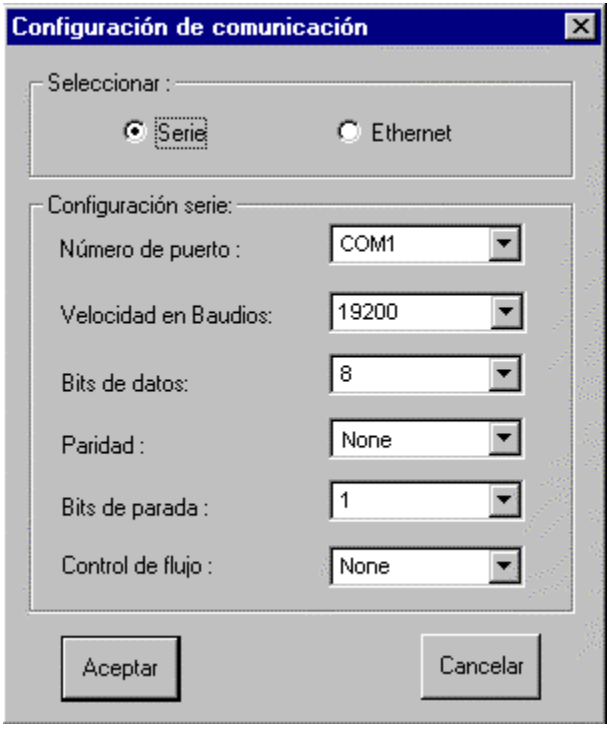
flow	h	Control por *hardware*
	n	Sin control

Los valores por defecto son: 19200, sin paridad, 8 bits de datos, 1 bit de parada, sin control de flujo.

Ejemplo:
WISAKER -t=COM1 baud=1200 data=8 parity=n stop=2

Interfaz del usuario: Se presenta esta ventana con el comando "**Opciones/ Comunicación**" de la ventana principal del objeto ISaGRAF NT.

Se puede elegir la comunicación serie o la comunicación vía Ethernet. Esta última aporta la posibilidad de modificar el número de puerto. Éste debería ser el mismo que se refleja en las especificaciones de enlace PC-PLC del banco de trabajo.



Cuando se elige la comunicación serie, aparece la configuración. Esta configuración debería ser la misma que se refleja en las especificaciones de enlace PC-PLC del banco de trabajo.

Simulación gráfica de tarjetas virtuales: Opción -x

Al seleccionarse esta opción, se simularán las tarjetas que se declaren como virtuales en el editor de conexiones de E/S (Véase Parte A).
Los posibles valores son 0 ó 1. 0 significa “sin simulación” y 1 significa “con simulación”.

Valor por defecto: El valor por defecto es 0, o el que aparece en el fichero ISaGRAF.INI.

Ejemplo:
WISAKER -x=1 simulará las tarjetas virtuales.

Interfaz del usuario: El correspondiente elemento de menú aparecerá como seleccionado o no seleccionado, de acuerdo con el estado de la opción. Las tarjetas simuladas aparecen en un panel gráfico.

Prioridad del objeto NT ISaGRAF: Opción -p

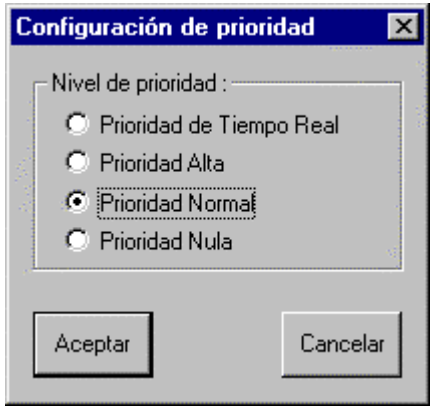
Teniendo en cuenta que el objeto opera bajo NT, es muy útil la especificación de un nivel de prioridad. Por ejemplo, se puede estar ejecutando en un objeto una aplicación ISaGRAF

crítica en tiempo con una prioridad mayor, junto con uno o dos objetos adicionales operando en segundo plano con prioridades menores.

Los posibles valores son 0, 1, 2 ó 3. 0 es la prioridad más alta y 3 es la más baja.

Ejemplos:
WISAKER -p=0
WISAKER -p=1

Interfaz del usuario: Se presenta esta ventana con el comando "**Opciones/Prioridad**" de la ventana principal del objeto NT ISaGRAF.



La prioridad más alta es la de tiempo real; la más baja es la de reposo.
0: Prioridad tiempo real
1: Prioridad alta
2: Prioridad normal
3: Prioridad reposo

- ***Ejemplos:***

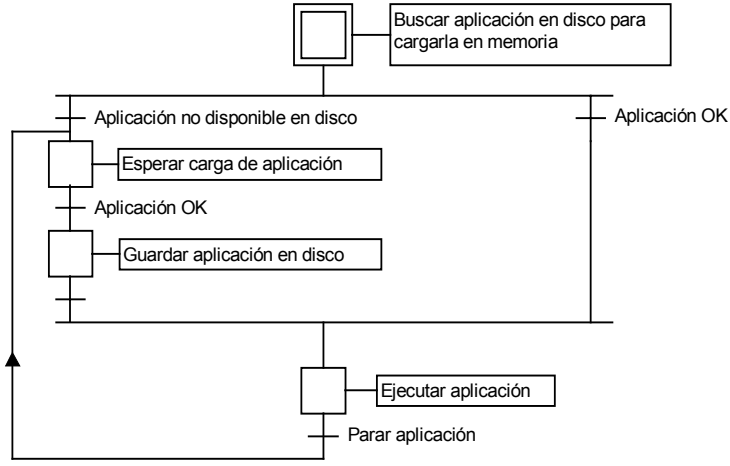
wisaker -t=COM1 Arranca el objeto ISaGRAF con el número de esclavo por defecto (1) y con COM1 como el puerto de comunicaciones.

wisaker -s=3 -t=COM1 Arranca el objeto ISaGRAF con el número de esclavo 3 y con COM1 como el puerto de comunicaciones.

C.6.3 Características específicas

- ***Arranque de ISaGRAF***

Al arrancarse el objeto, se ejecuta el siguiente algoritmo:



• Definiciones

El código de aplicación es la base de datos binaria que fue generada y cargada por el banco de trabajo, y posteriormente ejecutada por el objeto. Se puede completar con la tabla de símbolos.

La tabla de símbolos de la aplicación es una base de datos ASCII que genera y carga el banco de trabajo. Esta tabla crea el enlace entre los objetos simbólicos y los objetos internos del objeto. No es necesaria en el objeto salvo para la gestión específica de símbolos por parte del usuario, como por ejemplo las funciones DDE o de simulación de E/S con nombres de símbolos. Para mayor información sobre la tabla de símbolos, véase “Manual del Usuario: Técnicas Avanzadas de Programación”.

• Aplicaciones múltiples ISaGRAF

Se pueden ejecutar diferentes aplicaciones simultáneamente en una CPU, siempre que tengan diferentes números de esclavo y diferentes números lógicos de tarea de comunicación. No obstante, al ejecutar aplicaciones diferentes, el usuario debe tener cautela con determinados objetos de aplicación con acceso compartido, como las tarjetas de E/S. Por ejemplo, aplicaciones diferentes pueden utilizar tarjetas físicas distintas al no ser que se implemente algún tipo de servidor de E/S o semáforo a través del controlador de E/S.

• Copia de seguridad de la aplicación

Cuando se transfiere una nueva aplicación desde el depurador del banco de trabajo al objeto, se guarda el código de la aplicación en el directorio actual del objeto con el nombre de fichero:

ISAx1 fichero de copia de seguridad de código de aplicación ISaGRAF (donde x es el número de esclavo)

Además, si ya se hubiera cargado la tabla de símbolos de la aplicación, también se guarda en el directorio actual del objeto con el nombre de fichero:

ISAx6 fichero de copia de seguridad de símbolos de aplicación ISaGRAF (donde x es el número de esclavo)

Cuando se arranca el objeto ISaGRAF, se realiza la búsqueda de estos ficheros de código y símbolos de la aplicación en el directorio actual y se cargan en memoria.
Si no se dispone del fichero de símbolos, el objeto arrancará con el código de la aplicación, sin cargar los símbolos.
Si no se dispone del código de la aplicación, el objeto se pondrá a la espera para poder cargar una aplicación.

Para inicializar el objeto con una aplicación específica en el momento del arranque, sin utilizar el enlace con el depurador, se pueden copiar estos ficheros directamente al disco del directorio actual del objeto desde el mismo disco, si el banco de trabajo está en el mismo PC, o por medio de un disquete.

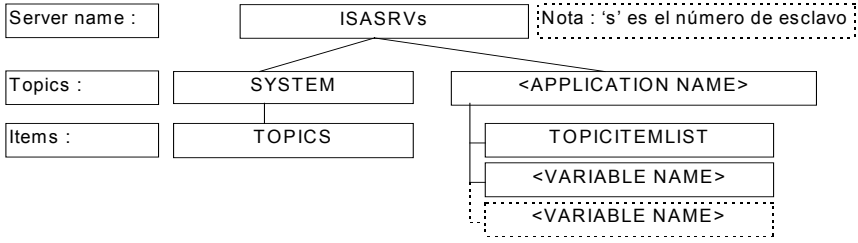
Si el banco de trabajo ISaGRAF está instalado en el directorio estándar \ISAWIN:
el fichero del código de aplicación del proyecto MYPROJ es.
\ISAWIN\APL\MYPROJ\appli.x8m
el fichero de símbolos de la aplicación del proyecto MYPROJ es:
\ISAWIN\APL\MYPROJ\appli.tst

Ejemplo:
Desde el directorio en el cual está instalado WISAKER.EXE, se introduce el siguiente comando:
copy \ISAWIN\APL\MYPROJ\appli.x8m isa11
WISAKER.EXE encontrará y ejecutará la aplicación 'myproj'.

Todos estos comandos pueden ser agrupados, por ejemplo, en un fichero de comandos que posteriormente podrá ser ejecutado desde el menú de herramientas del banco de trabajo (Véase "Manual del Usuario: Gestión de Programas).

Especificaciones DDE

El objeto ISaGRAF NT es un servidor DDE (*Dynamic Data Exchange* o Intercambio Dinámico de Datos). Cualquier *software* con la capacidad de funcionar como 'cliente,' puede conectarse al objeto para intercambiar variables. Por ejemplo, MSEXCEL puede animar gráficos basados en valores importados del objeto ISaGRAF vía DDE.
La función DDE requiere la presencia en el objeto de la tabla de símbolos de la aplicación.
Los temas tratados por DDE se definen de la siguiente manera:



« ISASRVs » es el nombre del servidor DDE, donde 's' es el número de esclavo.
« SYSTEM » es un asunto estándar que da acceso al tema « TOPICS »,
« TOPICS » facilita la lista de tópicos actualmente definidos: sistema y el nombre de la aplicación que se está ejecutando en el objeto NT ISaGRAF.
« APPLICATION NAME » es el nombre de la aplicación.
« TOPICITEMLIST » es la lista de ítems disponibles bajo el tema actual; esto facilita la lista de variables a las que se puede acceder vía DDE.
« VARIABLE NAME » es el nombre de una variable.

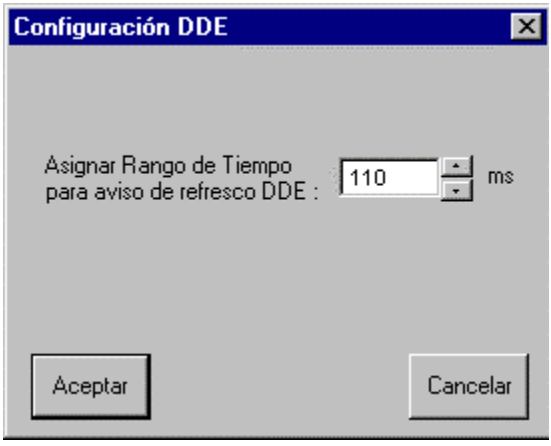
Cadencia del bucle de 'aviso' DDE para el objeto NT ISaGRAF: Opción -d
Generalmente, el cliente DDE realiza una llamada cíclica a las variables cada vez que las necesita. Esto puede tomar un tiempo considerable si existen muchas variables. Existe otro modo denominado el 'modo aviso' (bucle de aviso), en el que el propio servidor sólo envía variables modificadas, para que la comunicación esté minimizada y sea eficiente. En este modo, el servidor se remite periódicamente a las variables que están marcadas como 'variables avisadas' para saber cuáles deberían enviarse. Este periodo se llama la cadencia del bucle de 'aviso' DDE.

Con esta opción, se puede especificar la cadencia (en ms) del bucle de aviso DDE.

Valor por defecto: El valor por defecto es de 1000 ms, o el que se indique en el fichero ISaGRAF.INI.

Ejemplo:
WISAKER -d=100

Interfaz del usuario: Se presenta esta ventana con el comando "**Opciones/DDE**" de la ventana principal del objeto ISaGRAF NT.



Gestión de errores y mensajes de salida

El *software* objeto de ISaGRAF incorpora una función de detección y gestión de errores. La relación de avisos de error y sus descripciones aparece en el apéndice.

La detección de errores se procesa de la siguiente manera:
- Un error está compuesto por un número de error y de argumento que se remite a la rutina de errores de ISaGRAF.
- Si el señalizador de detección de errores está configurado en las opciones Ensamblar del banco de trabajo, se procesa el error. Si no lo está, se pierde la información y se finaliza la gestión de errores.

Si se procesa el error:
- Se presentan el número de error (valor decimal) y el argumento (valor hexadecimal) en la salida (ventana de WISAKER.EXE).
- El número y argumento del error pasan a un *buffer* o memoria intermedia de errores FIFO, configurada en anillo, para su recuperación en un momento posterior. Se establece el tamaño del *buffer* de errores en las opciones Ensamblar del banco de trabajo. Cuando el *buffer* está lleno, al registrarse cada nuevo error se pierde el de mayor antigüedad.
- Los errores pueden ser extraídos bien del depurador o bien de la aplicación activa por medio de la llamada SYSTEM (véase el Manual del Usuario).

Cuando el depurador detecta un error, aparece una ventana de error con un mensaje de descripción del error. Dependiendo del contexto de la aplicación (que esté activa o no), el depurador podrá mostrar el nombre del objeto (variable o programa) del que procede el error, o el argumento del error (valor decimal) entre paréntesis [x], que posee un significado diferente para cada error.

Se presenta un mensaje de bienvenida en la salida cuando arranca el objeto. Está compuesto por el número de esclavo, la configuración de comunicación y el nombre del servidor DDE.

Reloj del sistema

Ya que el objeto ISaGRAF está diseñado para operar en cualquier sistema, la referencia horaria que se utiliza tanto para la sincronización de ciclos como para el refresco de las variables horarias es el tick estándar, que dura 10 milisegundos.

En consecuencia, no se puede obtener una precisión mejor que 10 ms en las variables horarias. Por el mismo motivo, un tiempo de ciclo especificado que sea inferior o igual a 10 ms, y diferente a cero, provocará un error de desbordamiento de tiempo de ciclo (error 62). Véase el siguiente apartado para más información al respecto.

El usuario deberá solicitar una implementación especial al proveedor si su aplicación requiere una mayor precisión.

Duración de ciclos y comportamiento de tareas

- Al término de un ciclo ISaGRAF, justo antes de comenzar un ciclo nuevo, se ejecuta el siguiente algoritmo:

  Si se ha especificado un tiempo de ciclo (desde el banco de trabajo: véase el "Manual del Usuario: Gestión de Programas"), se cede la CPU durante el periodo de tiempo que queda (tiempo de ciclo especificado – tiempo de ciclo actual de la aplicación). Si este periodo de tiempo restante es negativo, se genera un desbordamiento y se cede la CPU durante 1 tick para forzar la entrada del gestor de tareas.

  Si no se ha especificado un tiempo de ciclo, o si el tiempo restante es inferior o igual 1 tick o igual a cero, se cede la CPU durante 1 tick para forzar la entrada del gestor de tareas.

  La precisión horaria del objeto corresponde a la del tick del sistema Windows NT.

  Se suelen utilizar tiempos de ciclo específicos para activar ciclos o para ceder la CPU a otras tareas que estén ejecutándose en el sistema Windows NT.

Tecla 'Exit'

Al probar una aplicación en condiciones no industriales, sobre un PC de sobremesa, le puede surgir al usuario la necesidad de interrumpir la ejecución de ISaGRAF. Esto se logra mediante la pulsación de una combinación de teclas para prevenir las paradas inesperadas. La secuencia es:

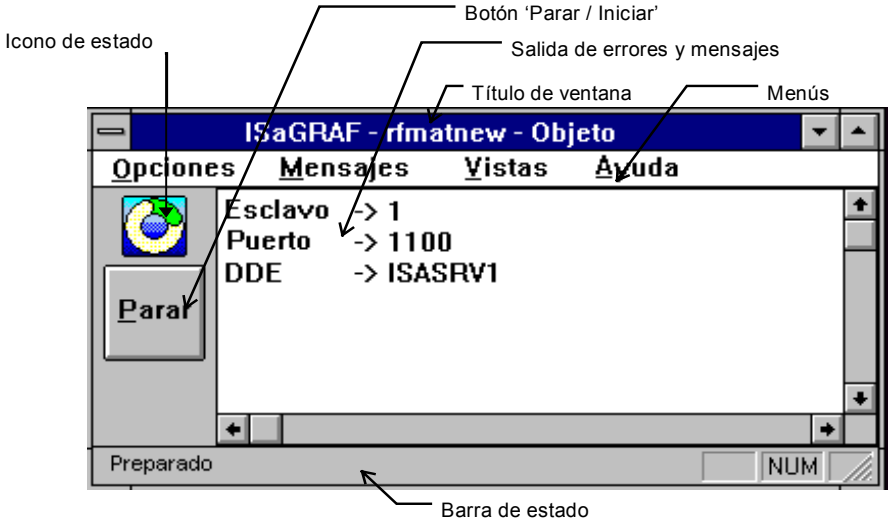
alt + F4

Un efecto secundario peligroso de esta 'salida rápida' es que no se cierra la interfaz de las tarjetas de E/S. Por consiguiente, la manera más apropiada de interrumpir el objeto ISaGRAF es:

- parar la aplicación desde el depurador o por medio de la tecla 'Parar / Start' (esto cerrará las tarjetas de E/S).
- parar el objeto ISaGRAF desde el menú del sistema.

C.6.4 Interfaz del usuario

La interfaz del usuario del objeto NT ISaGRAF tiene el siguiente aspecto:



Los principales componentes de la ventana son:

El título de la ventana
La barra del menú
Un icono de estado operativo
Un botón de '**Iniciar / Parar**'
Una salida de errores y mensajes
Y una barra de estado.

El título de la ventana contiene « ISaGRAF – nombre_de_applic_objeto », donde nombre_de_applic_objeto es el nombre de la aplicación activa. Sólo contiene « ISaGRAF - - Objeto » cuando no se está ejecutando una aplicación.

Barra de menú del objeto ISaGRAF NT:

La barra de menú tiene cuatro menús:

Opciones
Mensajes
Ver
Ayuda

• Menú "Opciones"

(véase también la primera sección dedicada a NT: Información general sobre opciones)
El menú "**Opciones**" permite acceder a las opciones de ejecución. Propone las siguientes opciones:

Esclavo permite acceder a la modificación del número de esclavo. La opción modificada se activa tras el siguiente arranque del objeto. Esta función no está

disponible si se ha arrancado el objeto con al menos una opción en la línea de comando.

Comunicación permite acceder a la configuración de la comunicación. La opción modificada se activa tras el siguiente arranque del objeto. Esta función no está disponible si se ha arrancado el objeto con al menos una opción que sea diferente a la opción -s.

DDE permite acceder a la modificación de la cadencia del bucle de 'aviso' DDE. La opción modificada se activa tras el siguiente arranque del objeto. Esta función no está disponible si se ha arrancado el objeto con al menos una opción que sea diferente a la opción -s.

Simulación E/S aparece seleccionada o no seleccionada, reflejando así el estado de la opción. La opción modificada se activa tras el siguiente inicio/parada de la aplicación.

Prioridad permite acceder a la modificación de prioridades. La opción modificada se activa inmediatamente.

Opciones por defecto recupera las opciones de ejecución por defecto sólo para los siguientes:
- Comunicación
- DDE
- Coordenadas de la ventana en pantalla

La opción modificada se activa tras el siguiente arranque del objeto. Esta función no está disponible si se ha arrancado el objeto con al menos una opción que sea diferente a la opción -s.

• Menú **"Mensajes"**

El menú "**Mensajes**" se encarga de la gestión de las salidas. Contiene dos comandos:

Reconocer para el parpadeo rojo en caso de producirse errores o mensajes.

Borrar elimina la salida por completo.

***Icono del objeto ISaGRAF** NT:*

Este icono refleja el estado del objeto:

- cuando se está ejecutando la aplicación, el icono gira
- cuando no hay aplicación (o se ha parado), el icono deja de girar
- hay errores o mensajes en la ventana de salidas. Una luz roja parpadea en el centro del icono. Para parar el parpadeo, se puede elegir la opción « Reconocer » del menú « Mensajes » o la opción « Borrar » del mismo menú (ojo: este último comando borrará el contenido de la ventana de salidas por completo). Para mayor información sobre los errores, véase el capítulo dedicado a gestión de errores y mensajes de salida.

Se resumen los diferentes estados en la siguiente tabla:

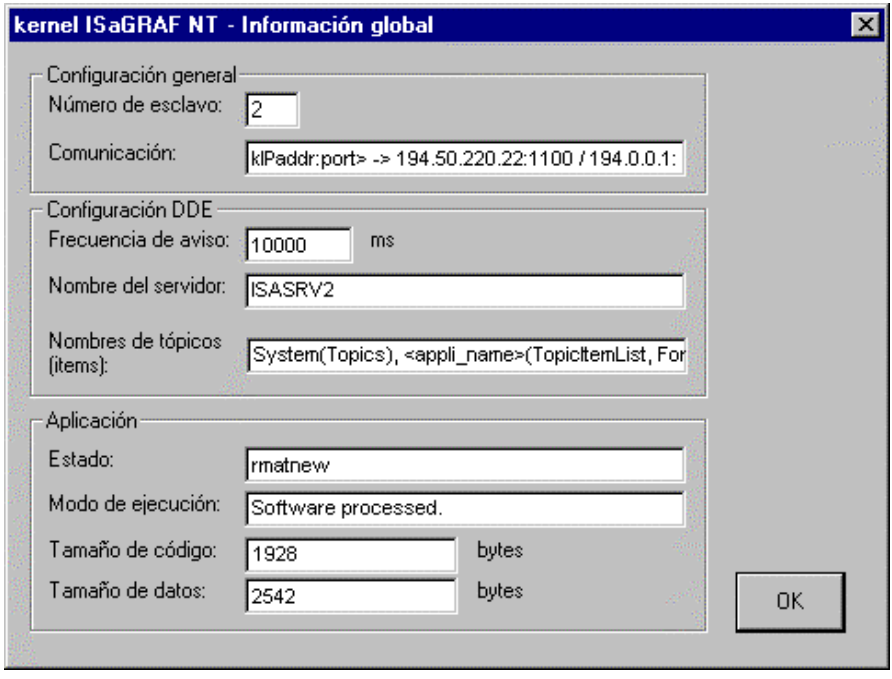
	sin errores	Errores o mensajes (centro rojo)
Aplicación activa		
Sin aplicación		

Tecla 'Iniciar / Parar' del objeto NT ISaGRAF:

La tecla Iniciar/Parar es estrictamente idéntica a la función '**Iniciar/Parar**' del depurador. El texto de la tecla reflejará el estado operativo de la aplicación. Si la aplicación está activa, el texto será « Parar »; si la aplicación está parada (o si no hay ninguna aplicación), el texto será « Iniciar » (téngase en cuenta que si no hay ninguna aplicación y se solicita la acción de inicio, la tecla pasará al modo 'Parar' y luego volverá al modo 'Iniciar').

Objeto ISaGRAF NT, información general

Con el comando "**Ver / Información**", la siguiente ventana de diálogo proporciona información general sobre la configuración del objeto y la aplicación activa:



Existen tres áreas:

a) Configuración general

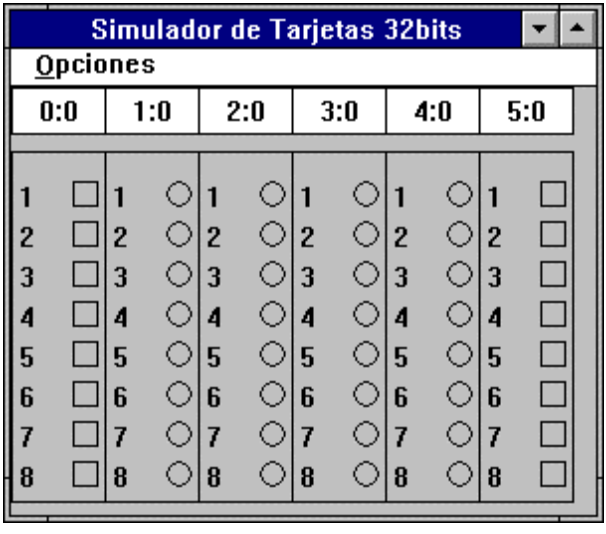
- Número de esclavo
- Configuración de la comunicación (Si el enlace de comunicación es el de Ethernet, además del número del puerto se presenta la lista de direcciones IP disponibles en el sistema NT actual)

b) Configuración DDE
- Cadencia de bucle de 'aviso'
- Nombre del servidor DDE
- Nombre de tópicos e ítems DDE. Esta información es de carácter general y no afecta a los valores reales. Es más, los campos situados entre < > deben sustituirse por los valores reales.

c) Aplicación
- El estado de la aplicación, que equivale a su nombre cuando existe una aplicación activa, y la cadena "Sin aplicación" cuando no hay una aplicación activa.
- El modo de ejecución de la aplicación, que indica si la aplicación se está ejecutando a través del procesador de *software*. En este caso, contiene la cadena « Procesada software ». O si la aplicación ha sido compilada con un compilador C, en cuyo caso contiene la cadena « Compilada C ». Si no se está ejecutando ninguna aplicación, contiene la cadena « Sin aplicación ».
- El tamaño del código, expresado en bytes. Si la aplicación activa está « Compilada C », este campo es cero.
- El tamaño de los datos, expresado en bytes. Esta es la suma de los datos internos de tiempo de ejecución y la base de datos de variables.

Simulación de tarjetas virtuales en el objeto ISaGRAF NT:

Al seleccionarse la opción « **Simulación E/S** », aparece esta ventana en el siguiente arranque de la aplicación:



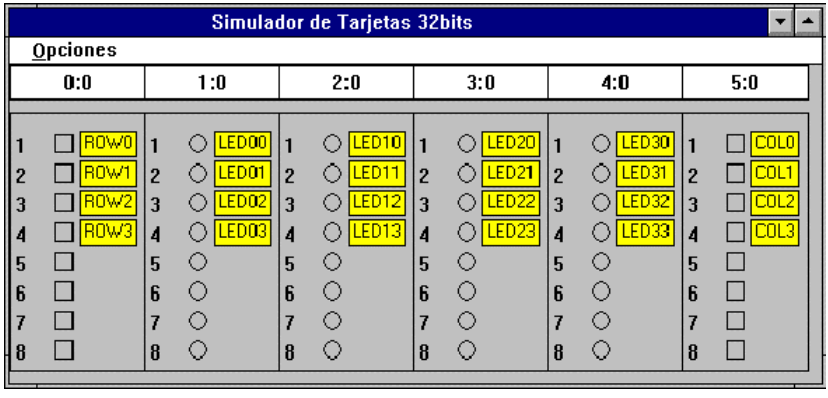
Dependiendo de la configuración de E/S, habrá más o menos tarjetas (y diferentes) y más o menos variables (y diferentes). Los caracteres « s:b » en la parte superior de cada tarjeta representan los identificadores de ranuras (s) y de tarjetas (b). El recuento comienza en cero, y no existe la posibilidad de modificarlo.
La ventana titulada 'Simulador tarjetas 32bits' funciona de acuerdo con el estado de 'Inicio / Paro' de la aplicación. Por lo tanto, si existe una aplicación activa con tarjetas virtuales (o que use tarjetas de simulador) y se selecciona la casilla « Simulación E/S », aparecerá esta ventana. Por el contrario, cuando se pulsa la tecla de 'Parar' la ventana se cierra. Esta ventana funciona a la vez que las llamadas de E/S.

El menú "**Opciones**" contiene dos elementos:

Nombres de variables muestra los nombres de las variables, siempre y cuando se haya 'cargado' la tabla de símbolos antes que el código tic.

Valores hexadecimales muestra cada valor entero en formato hexadecimal en lugar del formato por defecto, que es decimal.

Los nombres de las variables tienen el siguiente aspecto:



C.7 Programación en "C"

C.7.1 Descripción general

Este manual está orientado al usuario que ya posee experiencia con los conceptos de ISaGRAF y las herramientas del *Banco de trabajo*. Después de desarrollar aplicaciones de automatización pura, utilizando las **funciones de conversión**, las **funciones "C"** y los **bloques de función "C"** de las librerías estándares de CJ International, existe la posibilidad de desarrollar funciones de conversión, funciones "C" y bloques de función "C" "definidos por el usuario." Esto le permite al usuario realzar el PLC ISaGRAF objeto mediante la creación de librerías nuevas y sacar el máximo provecho de la flexibilidad de la estación de trabajo y la plataforma *hardware*.

Con un sistema de desarrollo en "C" y algo de experiencia previa en la programación en "C", este manual le permitirá al usuario personalizar el PLC ISaGRAF objeto para optimizar su control. Este tipo de desarrollo mejora tanto el rendimiento del PLC objeto como la comodidad y calidad de desarrollo con el banco de trabajo ISaGRAF para el programador de automatizaciones.

La información que contiene este documento no está dedicada a un único sistema objeto en especial. Sin embargo, existen algunas características (como por ejemplo las capacidades multitarea) que no son aplicables a determinados sistemas monotarea.

Características estándares del banco de trabajo de ISaGRAF

El *Banco de trabajo* de ISaGRAF incorpora numerosas funciones para la gestión de librerías de componentes "C" en el entorno de desarrollo de automatizaciones. Para la programación de automatizaciones, una conversión "C", una función "C" o un bloque de función "C" es una "**caja negra**" que está completamente definida por su interfaz.

Se utiliza el Gestor de Librerías ISaGRAF para añadir componentes a las librerías existentes y para definir la interfaz entre la implementación "C" y el uso de estos componentes en la programación **ST**/**FBD**. El Gestor de Librerías ISaGRAF también facilita la generación automática del *esqueleto* del código fuente en "C" para conversiones, funciones y bloques de función. Incluye además herramientas para la edición de ficheros "C" de este tipo. Para mayor información sobre las funciones del Gestor de Librerías, véase el **Manual del Usuario de ISaGRAF**.

Desarrollo en lenguaje "C"

El banco de trabajo ISaGRAF no incluye un compilador "C" o herramientas de compilación cruzada. El usuario tiene que tener su propio compilador "C", dedicado al sistema ISaGRAF objeto, para integrar sus componentes "C" en el *kernel* ISaGRAF.

Cuando se utiliza un compilador cruzado, el banco de trabajo ISaGRAF ofrece al usuario puntos de entrada para ejecutar ficheros de comando MS-DOS (.bat) definidos por el usuario en una ventana DOS. El compilador cruzado que se utilice tiene que ejecutarse en una ventana de emulación del DOS. Si no, se tiene que cerrar Windows antes de ejecutar los compiladores y enlazadores en un contexto puro de MS-DOS.

Notas técnicas

El Gestor de Librerías ISaGRAF le permite al usuario redactar un texto descriptivo para cada uno de los componentes de la librería. Estas **notas técnicas** constituyen la guía del usuario del componente "C" que se desarrolle, beneficiando al programador de automatizaciones con su descripción de las correspondientes conversiones, funciones o bloques de función en aplicaciones ISaGRAF.

La conversión, la función "C" o el bloque de función "C" tiene que estar definido de forma precisa en las notas técnicas, para que el programador de automatizaciones realmente lo pueda utilizar como una función integrada en ISaGRAF. Para una función "C", las notas técnicas deben describir:

- ❑ la función detallada que procesa la función
- ❑ la descripción completa de sus parámetros de llamada
- ❑ el significado de su valor de retorno
- ❑ la redacción detallada de sus parámetros de llamada y valor de retorno
- ❑ el contexto de la aplicación

Para un bloque de función "C", las notas técnicas deben describir:

- ❑ la función detallada que procesa la función de activación de bloques
- ❑ la descripción completa de sus parámetros de llamada
- ❑ el significado de su valor de retorno
- ❑ la redacción detallada de sus parámetros de llamada y de retorno
- ❑ el contexto de la aplicación

Para una función de conversión, las notas técnicas deben describir:

- ❑ el significado exacto de la conversión cuando se utiliza con una variable de entrada
- ❑ el significado exacto de la conversión cuando se utiliza con una variable de salida
- ❑ los límites de los valores que puede procesar la conversión

Las notas técnicas también pueden incluir información sobre:

- ❑ la identificación completa de la conversión, la función o el bloque de función
- ❑ cualquier dato sobre su mantenimiento y actualizaciones
- ❑ el sistema objeto soportado
- ❑ las características especiales de multitarea
- ❑ los servicios requeridos del sistema, memoria, controladores, etc.

C.7.2 Funciones de conversión "C"

El banco de trabajo ISaGRAF incluye una utilidad de **conversión lineal** que lleva a cabo una conversión analógica sencilla en el PLC objeto durante el tiempo de proceso. Esta utilidad no requiere ningún desarrollo “C”, al estar limitada a funciones estrictamente crecientes o decrecientes. Para una descripción completa de estas herramientas, véase el Manual del Usuario de ISaGRAF.

Las funciones de conversión le permiten al usuario la aplicación de cualquier conversión compleja, con operaciones específicas descritas en el lenguaje “C”. En términos básicos, se define una función de conversión **tanto para entradas como para salidas**. Incluso si no se utilizara una de las direcciones, se tiene que realizar la implementación y pruebas antes de integrar la conversión en el *kernel* de ISaGRAF, previniendo así cualquier error grave de sistema debido a una llamada errónea.

Las funciones de conversión están escritas en el lenguaje “C”, y compiladas y enlazadas con el *kernel* de ISaGRAF. Se tiene que instalar el *kernel* aumentado en el PLC objeto antes de utilizar las nuevas funciones de conversión en un proyecto ISaGRAF. No se pueden integrar las nuevas funciones de conversión en el Simulador de ISaGRAF. Se tienen que simular las aplicaciones ISaGRAF **antes** de insertar las funciones de conversión no estándares.

El código fuente en “C” de las conversiones estándares escritas por CJ International se instala junto con el banco de trabajo ISaGRAF. Se pueden utilizar como ejemplos para la creación de nuevas funciones. Se recomienda que **no se modifiquen** las funciones estándares para que puedan utilizarse en cualquier aplicación ISaGRAF. Las conversiones estándares que se suministran con el banco de trabajo ISaGRAF están soportadas por el Simulador de ISaGRAF.

Advertencia: Las funciones de conversión son operaciones **sincrónicas**, activadas por el Gestor de E/S de ISaGRAF en el tiempo de proceso, durante las fases de entrada o salida del ciclo de la aplicación. El tiempo que se invierte en la ejecución de una función de conversión está incluido en el **tiempo de ciclo** de la aplicación ISaGRAF. El usuario tiene que asegurarse de que la función de conversión no incluye ninguna “operación de espera”, para que el procesamiento de ciclos de ISaGRAF no se vea extendido sin necesidad.

Añadir una función a la librería ISaGRAF

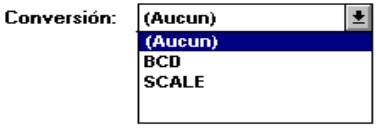
Se utiliza el Gestor de Librerías de ISaGRAF para añadir nuevas funciones de conversión a la librería ISaGRAF, al nivel de banco de trabajo. Para ello, se emplea el comando "**Nuevo**" del menú "**Ficheros**" cuando está seleccionada la librería de funciones de conversión. No hay necesidad de definir parámetro alguno en el banco de trabajo, ya que las funciones de conversión utilizan una interfaz predefinida estándar.

Cuando se termina de crear una nueva función de conversión, se debe escribir sus **notas técnicas**. El Gestor de Librerías de ISaGRAF genera el *esqueleto* del código fuente “C” de la nueva función de conversión, de forma automática.

Utilización de conversiones en un proyecto ISaGRAF

Se pueden utilizar determinadas funciones de conversión para filtrar los valores de cualquier variable analógica de entrada o salida del proyecto seleccionado. Para vincular una función de conversión a una variable, se ejecuta el editor de declaración de variables, se selecciona

una variable analógica de entrada o salida y se editan sus parámetros. Se utiliza el campo "**conversión**" de la ventana de declaración de variables para configurar la función de conversión que está vinculada a una variable analógica de E/S:



Aparecen tanto funciones como tablas de conversión en la lista. De ello se desprende que no se puede utilizar el mismo nombre para una función y una tabla. No se puede vincular una variable a una función de conversión que esté pendiente de definir o integrar en el *kernel* de ISaGRAF.

Interfaz estándar de "C"

La interfaz de una función de conversión siempre tiene el mismo formato. Los parámetros de llamada y retorno pasan a través de una estructura. Esta estructura está definida en el fichero "**TACN0DEF.h**":

```
/*
  Nombre: tacn0def.h
  Fichero de definición de conversión en objeto
*/

#define DIR_INPUT 0                 /* dirección = conversión entrada */
#define DIR_OUTPUT 1                /* dirección = conversión salida */

typedef int32   T_ANA;              /* tipo ANA entero          */
typedef float   T_REAL;             /* tipo ANA real            */

typedef struct {                    /* estructura de conversión       */
     uint16 number;                 /* número de conversión (reservado)  */
     uint16 direction;              /* dirección de conversión        */
     T_REAL *before;                /* valor antes de conversión      */
     T_REAL *after;                 /* valor después de conversión       */
  } str_cnv;

#define ARG_BEFORE (*(arg->before))
#define ARG_AFTER  (*(arg->after))
#define DIRECTION  (arg->direction)

/* fin de fichero */
```

La estructura "**str_cnv**" aporta una descripción completa de la interfaz. El único parámetro de una función de conversión "C" es un puntero que señala la estructura. El campo "**number**" es

el número lógico de la función de conversión (ubicado en la librería ISaGRAF) y no se tiene que utilizar en la programación.

El campo "**direction**" indica si se tiene que aplicar la conversión a una variable de entrada o a una variable de salida. Contiene el valor **DIR_INPUT** para una conversión de entrada, o el valor **DIR_OUTPUT** para una conversión de salida.

El campo "**before**" indica el valor que existía antes de la conversión. Este campo tiene un significado diferente, según se trate de una conversión de entrada o de salida. Representa el valor eléctrico (leído en el dispositivo de entrada) en el caso de una conversión de entrada, cuando el campo **direction** asume el valor **DIR_INPUT**. Representa el valor físico (el valor que se utiliza en las ecuaciones programadas) en el caso de una conversión de salidas, cuando el campo **direction** asume el valor **DIR_OUTPUT**.

El campo "**after**" indica el valor que existe después de la conversión. Este campo tiene un significado diferente, según se trate de una conversión de entrada o de salida. Representa el valor físico (el valor que se utiliza en las ecuaciones programadas) en el caso de una conversión de entrada, cuando el campo **direction** asume el valor **DIR_INPUT**. Representa el valor eléctrico (enviado al dispositivo de salida) en el caso de una conversión de salidas, cuando el campo **direction** asume el valor **DIR_OUTPUT**.

El programador puede utilizar las definiciones "**ARG_BEFORE**" y "**ARG_AFTER**" para acceder directamente a los campos **before** y **after** de la estructura que se transfiere a la función de conversión "C". Los valores procesados son **valores flotantes de simple precisión**. Se convierte el resultado a un valor entero largo cuando la conversión se aplica a una variable analógica entera. Esto significa que se puede utilizar la misma conversión para las variables analógicas de E/S tanto reales como de valores enteros.

Código fuente

Teniendo en cuenta que se puede utilizar la función de conversión para las variables analógicas tanto de entrada como de salida, el código fuente "C" de la función se divide en dos partes principales: la conversión de entrada y la conversión de salida. Se utiliza el campo **direction** de la estructura de la interfaz para seleccionar la conversión que va a ser aplicada. El Gestor de Librerías de ISaGRAF genera automáticamente el *esqueleto* completo de la función, una vez que la función de conversión ha sido creada. Esto incluye la estructura principal de selección "**IF**". A continuación se muestra el *esqueleto* estándar de una función de conversión:

```
/*
  Función de conversión
  Nombre: sample
*/

#include <tasy0def.h>
#include <tacn0def.h>

void CNV_sample (str_cnv *arg)
{
        if (DIRECTION == DIR_INPUT) { /*CONVERSIÓN ENTRADA*/
```

```
        }
        else { /*CONVERSIÓN SALIDA*/

        }
}
```

/* La siguiente función muestra el enlace con el gestor E/S ISaGRAF, utilizando el nombre de la conversión. Esta función está completamente generada por el Gestor de Librerías ISaGRAF. */

```
UFP cnvdef_sample (char *name)
{
        sys_strcpy (name, "SAMPLE");            /* da el nombre de la conversión */
        return (CNV_sample);            /* retorna la función de implementación */
}
```

La mejor manera de completar la parte específica de la función es mediante la creación de dos funciones locales independientes para la conversión de entradas y la conversión de salidas. Estas funciones serán invocadas por el algoritmo principal, como se explica en los comentarios del ejemplo anterior, en la estructura **IF** principal.

Se requiere el fichero "**TASY0DEF.H**", del *kernel* de ISaGRAF, para las definiciones dependientes del sistema. También contiene la definición del tipo **UFP**, que representa un puntero a funciones *void* y que se utiliza para la función de declaración.

Enlaces entre proyectos e implementación "C"

El enlace lógico entre la implementación de una función de conversión y la utilización de la conversión en un proyecto ISaGRAF se realiza por medio del nombre de la conversión. Se añade una función de "declaración" al código fuente "C" de la función de conversión. Esta función, que sólo se invoca una vez al arrancarse la aplicación, indica al Gestor de E/S de ISaGRAF el nombre de conversión correspondiente a la función que se va a implementar. Una función de declaración tiene el siguiente formato estándar:

```
UFP cnvdef_xxx (char *name)
{
        strcpy (name, "XXX");           /* da el nombre de la conversión */
        return (CNV_xxx);               /* retorna la función de implementación */
}
/* (xxx es el nombre de la conversión) */
```

Cuando se utiliza en la sentencia **strcpy**, el nombre de la función debe estar en mayúsculas. Cuando se utiliza en el nombre de la función de implementación de la conversión y en el nombre de la función de declaración, se tiene que escribir en minúsculas.

La utilización de los prefijos "**CNV_**" y "**cnvdef_**" para las funciones de implementación y definición le permite al usuario nombrar una conversión con una palabra clave reservada del lenguaje "C", o el nombre de una función ya existente de las librerías "C" de ISaGRAF.

Se pueden añadir otras sentencias a la función de declaración para realizar cualquier operación específica de inicialización que esté relacionada con esta conversión. El sistema ISaGRAF asegura al usuario que esta función **sólo se invoca una vez** cuando arranca la aplicación.

Se invoca la función de declaración para cualquier función integrada de conversión, incluso si no se utiliza en la aplicación ISaGRAF. El *kernel* de ISaGRAF detectará un error fatal si la conversión que se utiliza en la aplicación no está integrada en el *kernel*.

Antes de enlazar nuevas funciones con el *kernel*, el usuario tiene que escribir otro fichero fuente en "C", llamado "**GRCN0LIB.C**", e insertarlo (junto con las funciones de conversión) en la lista de ficheros para el enlazador. El fichero "**GRCN0LIB.C**" sólo contiene un vector de funciones de declaración. Se lee este vector durante las inicializaciones de la aplicación, para establecer un enlace dinámico con las funciones de conversión escritas en "C". A continuación se muestra un ejemplo de un fichero de este tipo:

```
/* Fichero "GRCN0LIB.c" - Ejemplo con conversiones de librería estándar */

#include <tasy0def.h>                    /* requerido para definición de tipos */

extern UFP cnvdef_scale (char *name);          /* declaración de función SCALE */
extern UFP cnvdef_bcd (char *name);  /* declaración de función BCD */

UFP_LIST CNVDEF[ ] = {              /* vector de declaraciones de funciones */
        /* para funciones de conversión integradas */
        cnvdef_scale,
        cnvdef_bcd,

NULL };

/* fin de fichero */
```

Se tiene que finalizar el vector **CNVDEF** por medio de un puntero NULO. Pueden producirse conflictos si no se cumple con esta condición. Se producirán referencias no resueltas a la hora de enlazar el nuevo *kernel* de ISaGRAF si no está definido el vector **CNVDEF**.

Al escribir este fichero se puede crear un nuevo *kernel*, incluyendo todas las conversiones existentes. También se puede construir un *kernel* personalizado para un único proyecto, mediante la inserción en el vector **CNVDEF** de sólo las conversiones que se utilizan en el proyecto. El Generador de Código de ISaGRAF genera el fichero "**GRCN0LIB.C**" automáticamente cuando se construye el código de una aplicación. El fichero, que se ubica en el directorio del proyecto ISaGRAF, abarca únicamente las conversiones que se utilizan en el proyecto.

Límites

La librería de ISaGRAF puede contener hasta **128** funciones de conversión. Se puede procesar cualquier tipo de operación con una función de conversión. Se debe tener en cuenta

que las funciones se invocan en el ciclo ISaGRAF de manera **sincrónica**, por lo que la ejecución de la función tiene un efecto directo sobre los tiempos de ciclo.

C.7.3 Funciones "C"

Se utilizan las funciones "C" para aumentar las capacidades estándar de los lenguajes **ST** y **FBD**. Se pueden utilizar para realizar cualquier cálculo específico, llamadas de sistema o comunicaciones específicos, o para instalar un conjunto de servicios para el diálogo entre una aplicación ISaGRAF y otras tareas. Las funciones están escritas en el lenguaje "C" y están compiladas y enlazadas con el *kernel* de ISaGRAF. Se tiene que instalar el *kernel* aumentado en el PLC objeto antes de utilizar las nuevas funciones en un proyecto ISaGRAF.

No se pueden integrar las nuevas funciones en el Simulador de ISaGRAF. Se tienen que simular las aplicaciones ISaGRAF **antes** de utilizarse las funciones no estándares.

Advertencia: Las funciones son operaciones **síncronas**, activadas por el *kernel* de ISaGRAF durante el ciclo de la aplicación. El tiempo que se invierte en la ejecución de una función está incluido en el **tiempo de ciclo** de la aplicación ISaGRAF. El usuario tiene que asegurarse de que la función no incluye ninguna "operación de espera", para que el procesamiento de ciclos de ISaGRAF no se vea extendido sin necesidad.

Añadir una función a la librería ISaGRAF

Se utiliza el Gestor de Librerías de ISaGRAF para añadir nuevas funciones "C" a la librería ISaGRAF, al nivel de banco de trabajo. Para ello, se emplea el comando **"Nuevo"** del menú **"Ficheros"** cuando está seleccionada la librería de funciones "C". Cuando se termina de crear una nueva función, se redactan sus **notas técnicas**. El Gestor de Librerías de ISaGRAF genera el *esqueleto* del código fuente "C" de la nueva función, de forma automática.

Se utiliza el comando **"Parámetros"** del menú **"Edición"** para definir los parámetros de llamada y retorno de la nueva función.

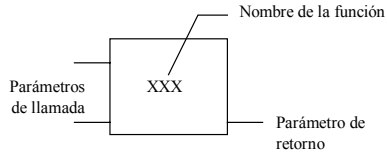
Utilización de una función "C" en un proyecto ISaGRAF

Se puede utilizar cualquier función "C" integrada como función estándar en los programas de un proyecto ISaGRAF. Se pueden invocar las funciones "C" desde los lenguajes **ST** y **FBD**, y desde algunas sentencias especiales del lenguaje **SFC**.

La invocación de una función "C" desde el lenguaje **ST** se ajusta a las convenciones de invocación de función del lenguaje. Los parámetros de invocación de la función están escritos después del nombre de la función, entre paréntesis y separados por comas. La expresión representa el valor retornado por la función. Se puede insertar la invocación de una función "C" en cualquier sentencia de asignación o expresión compleja. A continuación se muestra un ejemplo de invocación de una función "C" en una sentencia de asignación:

result := ProcName (par1, par2, ... parN);

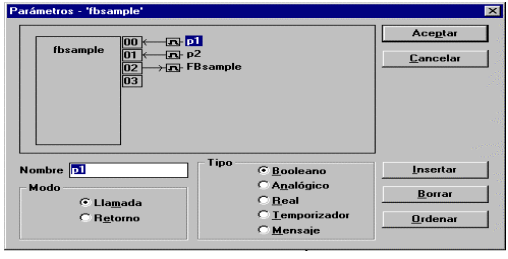
Un programa **FBD** puede invocar a cualquier función “C”. Se utiliza una función como una ‘caja’ estándar de función. Los parámetros de llamada están conectados al lateral izquierdo de la caja de función. El parámetro de retorno está conectado al lateral derecho de la caja. Una caja de función tiene el siguiente aspecto estándar:



Se puede invocar a una función “C” desde cualquier bloque de acciones **SFC**, o en cualquier condición booleana asociada a una transición.

Definición de la interfaz de una función “C”

Se utiliza el comando "**Parámetros**" del menú "**Edición**" para definir los parámetros de llamada y retorno de una función nueva. Una función puede tener hasta **31** parámetros de llamada y siempre tiene **un** único parámetro de retorno. Se describen los parámetros de la función “C” en la siguiente ventana de diálogo:



La lista que aparece en la parte superior de la ventana muestra los parámetros de la función “C”, de acuerdo con el orden marcado por el prototipo de invocación de funciones: los parámetros de llamada al principio y el de retorno al final. La parte inferior de la ventana presenta una descripción detallada del parámetro que está seleccionado en la lista.:

- el nombre del parámetro
- la dirección (llamada/retorno) del parámetro
- el tipo de parámetro

Se puede utilizar cualquiera de los tipos de datos de ISaGRAF como un parámetro: Booleano, Analógico entero, Analógico real, Temporizador o Mensaje. Se tiene que diferenciar entre analógicos enteros y reales.
A continuación se presenta la correspondencia entre los tipos ISaGRAF y los tipos “C”:

BOOLEANO	unsigned long	Palabra entera de 32 bits, sin signo: 1=verdadero / 0=falso
ANALÓGICO	long	Palabra entera de 32 bits, con signo
REAL	float	Valor flotante, simple precisión

TEMPORIZADOR	unsigned long	Palabra entera de 32 bits, sin signo: (unidad es 1 milisegundo)
MENSAJE	char *	Cadena de caracteres

Cuando se transfiere el valor de un mensaje a una función “C”, no puede contener caracteres nulos. La cadena que se transfiere al código “C” tiene la terminación en nulo. No debe olvidarse que el parámetro de retorno tiene que ser el último de la lista. Se observarán las siguientes reglas a la hora de nombrar parámetros:

- la longitud del nombre no puede superar los 16 caracteres
- el primer carácter tiene que ser una letra
- los siguientes caracteres tiene que ser letras, dígitos o el carácter de subrayado ‘_’
- los nombres son insensibles al caso (mayúsculas/minúsculas)

No se puede utilizar el mismo nombre para más de un parámetro de la función. Un parámetro de llamada no puede tener el mismo nombre que el de retorno. Se **puede** utilizar el mismo nombre para parámetros de funciones diferentes. El nombre por defecto del parámetro de retorno es "**Q**". Este nombre puede modificarse libremente. Se utiliza el nombre de un parámetro para identificar el parámetro en el código fuente “C”.

Se utiliza el comando "**Insertar**" para insertar un parámetro nuevo delante del parámetro seleccionado. Se utiliza el comando "**Borrar**" para borrar el parámetro seleccionado. El comando "**Ordenar**" reordena (clasifica) los parámetros de forma automática, colocando el parámetro de retorno al final de la lista. Se pulsa el botón “**Aceptar**” para guardar la definición de la interfaz de funciones y cerrar la ventana de diálogo. Se pulsa el botón "**Cancelar**" para cerrar la ventana de diálogo sin efectuar cambios en la definición de la interfaz de funciones.

Interfaz de la función "C"

La interfaz de una función depende de la definición de sus parámetros. Los parámetros de llamada y de retorno pasan a través de una estructura. Esta estructura está definida en el fichero "**GRUS0nnn.H**", donde "**nnn**" es el número lógico de la función en la librería ISaGRAF. A continuación se presenta un ejemplo de la interfaz “C” de la función "**SIN**" (cálculo de senos):

```
/* Fichero: GRUS0255.h - función "sample" */

typedef  long                   T_BOO;
typedef  long                   T_ANA;
typedef  float                  T_REAL;
typedef  long                   T_TMR;
typedef  char                   *T_MSG;

typedef struct {
        /* CALL   */            T_REAL   _param1;
        /* RETURN */            T_REAL   _param2;
} str_arg;

#define PARAM1                  (arg->_param1)
```

```
#define PARAM2                  (arg->_param2)

/* fin de fichero */
```

En la siguiente tabla, se presenta la relación que existe entre los tipos ISaGRAF y los tipos "C". Los tipos ISaGRAF están definidos como tipos "C" en el fichero de definiciones de la función.

Booleano	T_BOO	long (32 bits)
Analógico entero	T_ANA	long
Analógico real	T_REAL	float (32 bits – simple precisión)
Temporizador	T_TMR	long
Mensaje	T_MSG	char * (32 bits – puntero car)

Cada campo de la estructura **"str_arg"** corresponde a un parámetro de la función. El parámetro de retorno es el último de la estructura. Los parámetros de llamada aparecen en la estructura en el mismo orden que se estableciera para la definición de funciones. Se define un identificador en mayúsculas para facilitar el acceso directo a un parámetro de la estructura que se pasó a la implementación "C" de la función. Los nombres de los identificadores son aquellos que se introducen durante la definición de la función, con el Gestor de Librerías de ISaGRAF.

Se actualiza el fichero de definiciones de "C" cada vez que se cambia la interfaz de la función por medio del Gestor de Librerías de ISaGRAF. Se asegura así la coherencia completa entre la implementación de la función y su utilización en los programas de las aplicaciones ISaGRAF.

Código fuente

A continuación se muestra el *esqueleto* estándar de la implementación de una función "C":

```
/* Ejemplo de función de usuario - Número: 255 - Nombre: "SAMPLE" */

#include "tasy0def.h"           /* definiciones comunes de kernel */
#include "grus0255.h"           /* definición de interfase para función 255 */

void USP_sample (str_arg *arg)
{
        /* cuerpo de función */
}

/* La siguiente función se utiliza para la inicialización de la función y la declaración de su
implementación. Realiza el enlace con el kernel ISaGRAF, usando el nombre de la función.
Esta función está generada completamente por el Gestor de Librería ISaGRAF. */

UFP uspdef_sample (char *name)
{
```

```
        strcpy (name, "SAMPLE");      /* dar el nombre a la función */
        return (USP_sample);          /* retorna la función implementada */
}

/* fin de fichero */
```

Se requiere el fichero cabecera "**TASY0DEF.H**", del *kernel* de ISaGRAF, para las definiciones dependientes del sistema. También contiene la definición del tipo **UFP**, que representa un puntero a funciones *void* y que se utiliza para la función de declaración.

Enlaces entre proyectos e implementación "C"

El enlace lógico entre la implementación de una función "C" y su utilización en los programas de un proyecto ISaGRAF se realiza con el nombre de la función. Se añade una función de "declaración" al código fuente "C" de la función. Esta función, que sólo se invoca una vez al arrancarse la aplicación, indica al *kernel* de ISaGRAF el nombre de la función "C" correspondiente a la función que se va a implementar. Una función de declaración tiene el siguiente formato estándar:

```
UFP uspdef_xxx (char *name)
{
        strcpy (name, "XXX");         /* da el nombre de la función */
        return (USP_xxx);             /* retorna la función implementada */
}
/* (xxx es el nombre de la función) */
```

Cuando se utiliza en la sentencia **strcpy**, se escribe el nombre de la función "C" en **mayúsculas**. Cuando se utiliza en el nombre de la función de implementación y en el nombre de la función de declaración, se tiene que escribir en minúsculas. La utilización de los prefijos "**USP_**" y "**uspdef_**" para las funciones de implementación y definición le permite al usuario nombrar una función con una palabra clave reservada del lenguaje "C", o el nombre de una función ya existente de las librerías "C" de ISaGRAF.

Se pueden añadir otras sentencias a la función de declaración para realizar cualquier operación específica de inicialización que esté relacionada con esta función. El sistema ISaGRAF asegura al usuario que esta función **sólo se invoca una vez** cuando arranca la aplicación. Se invoca la función de declaración para cualquier función "C" integrada, incluso si no se utiliza en los programas de la aplicación ISaGRAF. El *kernel* de ISaGRAF detectará un error fatal si una función "C" que se utiliza en la aplicación no está integrada en el *kernel*.

Antes de enlazar nuevas funciones con el *kernel*, el usuario tiene que escribir otro fichero fuente en "C", llamado "**GRUS0LIB.C**", e insertarlo (junto con las funciones) en la lista de ficheros para el enlace. El fichero "**GRUS0LIB.C**" sólo contiene un vector de funciones de declaración. Se lee este vector durante la inicialización de la aplicación, para establecer un enlace dinámico con las funciones escritas en "C". A continuación se muestra un ejemplo de un fichero de este tipo:

```
/* Fichero "GRUS0LIB.c" - Ejemplo usando funciones trigonométricas */

#include <tasy0def.h>                 /* requerido para definición de tipos */
```

```
extern UFP uspdef_fc1 (char *name);  /* declaración de funciones */
extern UFP uspdef_fc2 (char *name);
extern UFP uspdef_fc3 (char *name);
extern UFP uspdef_fc4 (char *name);

UFP_LIST USPDEF[ ] = {              /* vector de declaración de funciones*/
                                    /* para funciones integradas */
        uspdef_fc1,
        uspdef_fc2,
        uspdef_fc3,
        uspdef_fc4,

NULL };

/* fin de fichero */
```

Se tiene que finalizar el vector **USPDEF** por medio de un puntero NULO. Pueden producirse conflictos si no se cumple con esta condición. Se producirán referencias no resueltas a la hora de enlazar el nuevo *kernel* de ISaGRAF si no está definido el vector **USPDEF**. Al escribir este fichero se puede crear un nuevo *kernel*, incluyendo todas las funciones existentes. También se puede construir un *kernel* personalizado para un único proyecto, mediante la inserción en el vector **USPDEF** de sólo las funciones que se utilizan en el proyecto. El Generador de Código de ISaGRAF genera el fichero "**GRUS0LIB.C**" automáticamente cuando se construye el código de una aplicación. El fichero, que se ubica en el directorio del proyecto ISaGRAF, abarca únicamente las funciones que se utilizan en el proyecto.

Límites

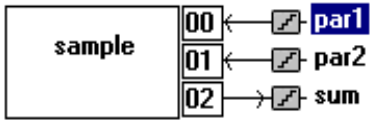
La librería de ISaGRAF puede contener hasta **255** funciones "C". Se puede procesar cualquier tipo de operación con una función. Se debe tener en cuenta que las funciones se invocan en el ciclo ISaGRAF de manera **sincrónica**, por lo que la ejecución de la función tiene un efecto directo sobre los tiempos de ciclo.

Ejemplo completo

A continuación se presenta la programación completa de una función de muestra titulada "**sample**", que realiza una simple suma. A continuación se da la nota técnica de la función.

nombre:	SAMPLE
descripción:	realiza una simple suma de valores analógicos enteros
fecha de creación:	1 de julio de 1992
autor:	CJ International
llamada:	operandos valores enteros par1, par2
retorno:	suma valor entero
prototipo:	sum := sample (par1, par2);

La interfaz de la función tiene este aspecto:



A continuación se muestra la cabecera fuente “C” de la función:

```
/* Fichero: GRUS0255.h - definiciones función C de usuario - Nombre: sample */

/* definición de tipos de datos estándar ISaGRAF */

typedef long  T_BOO;
typedef long  T_ANA;
typedef float T_REAL;
typedef long  T_TMR;
typedef char  *T_MSG;

/* definición de estructura de parámetros de llamada y retorno */

typedef struct {
        T_ANA _par1;            /* parámetro de llamada #1 */
        T_ANA _par2;            /* parámetro de llamada #2 */
        T_ANA _sum;             /* parámetro de retorno*/
} str_arg;

/* identificadores usados para parámetros de llamada y retorno */

#define PAR1                    (arg->_par1)
#define PAR2                    (arg->_par2)
#define SUM                     (arg->_sum)

/* fin de fichero */
```

A continuación está el código fuente “C” de la función. Sólo las líneas impresas con caracteres en negrita fueron introducidas manualmente por el programador de “C”.

```
/* Fichero: GRUS0255.c - función C de usuario - Nombre: SAMPLE */

#include "tasy0def.h"           /* requerido para definición de tipos */
#include "grus0255.h"           /* cabecera fuente de función C */

/* servicio C principal: calcula la suma */
```

```
void USP_sample (str_arg *arg)
{
        SUM = PAR1 + PAR2;
}

/* declaración de servicio requerido para enlace con kernel ISaGRAF */

UFP uspdef_sample (char *name)
{
        strcpy (name, "SAMPLE");
        return (USP_sample);
}
/* fin de fichero */
```

C.7.4 BLOQUES DE FUNCIÓN "C"

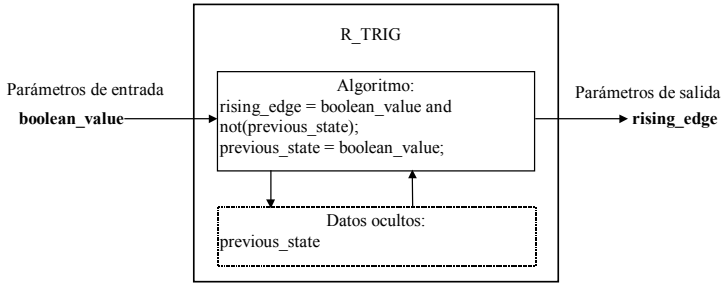
Los bloques de función "C" asocian operaciones con datos estáticos. Completan el conjunto de funciones “C” al permitir el procesamiento de objetos estáticos. Se suelen utilizar para aumentar las capacidades estándar de los lenguajes **ST** y **FBD**. A diferencia de las funciones, que procesan valores, los bloques de función pueden procesar datos estáticos. Esto significa que un algoritmo de bloque de función puede gestionar variaciones en datos a lo largo del tiempo.

Los bloques de función están escritos en el lenguaje “C”, y están compilados y enlazados al *kernel* de ISaGRAF. Se tiene que instalar el *kernel* aumentado en el PLC objeto antes de utilizar los nuevos bloques de función en proyectos ISaGRAF. No se pueden integrar nuevos bloques de función en el Simulador de ISaGRAF. Las aplicaciones de ISaGRAF tienen que ser simuladas **antes** de utilizarse las funciones no estándares.

Advertencia: Las llamadas a bloque de función son operaciones **síncronas**, activadas por el *kernel* de ISaGRAF durante el ciclo de la aplicación. El tiempo que se invierte en la ejecución de un servicio de activación o lectura de bloque de función está incluido en el **tiempo de ciclo** de la aplicación ISaGRAF. El usuario tiene que asegurarse de que el bloque de función no incluye ninguna “operación de espera”, para que el procesamiento de ciclos de ISaGRAF no supere el tiempo máximo permitido.

Declaración de instancias de bloque de función

Un bloque de función es un objeto que combina operaciones y datos estáticos. A continuación se muestra un ejemplo del bloque de función "**R_TRIG**", que detecta el flanco de subida de una expresión booleana. El bloque tiene la siguiente descripción funcional:



Se necesita la variable estática oculta "**previous_state**" para el cálculo del flanco. Esta variable tiene que ser diferente para cada utilización del bloque de función "**TRIG**" en la aplicación. Las instancias de los bloques de función que se utilizan en el lenguaje ST tienen que ser declaradas en el diccionario. Dado que el bloque de función pose datos internos ocultos, cada copia (instancia) de un bloque de función tiene que estar identificada por un nombre único. La asignación de nombre al tipo de bloque se realiza por medio del Gestor de Librerías. La asignación de nombres a las instancias se realiza por medio del editor de diccionarios.

No se tienen que declarar los bloques de función utilizados en el lenguaje FBD, ya que el editor FBD de ISaGRAF declara las instancias de los bloques utilizados automáticamente. Las instancias de bloque de función declaradas automáticamente por el editor FBD de ISaGRAF son siempre **LOCALES** con respecto al programa editado.

Añadir un bloque de función a la librería ISaGRAF

Se utiliza el Gestor de Librerías de ISaGRAF para añadir un nuevo bloque de función “C” a la librería ISaGRAF, al nivel de banco de trabajo. Para ello, se emplea el comando "**Nuevo**" del menú "**Ficheros**" cuando está seleccionada la librería de bloques de función “C”. Cuando se termina de crear un nuevo bloque de función, se redactan sus **notas técnicas**. El Gestor de Librerías de ISaGRAF genera el *esqueleto* del código fuente “C” del nuevo bloque de función, de forma automática. Se utiliza el comando "**Parámetros**" del menú "**Edición**" para definir los parámetros de llamada y retorno del nuevo bloque de función.

Utilización de un bloque de función "C" en un proyecto ISaGRAF

Se puede utilizar cualquier bloque de función “C” integrado en los programas de un proyecto ISaGRAF. Se pueden invocar las funciones “C” desde los lenguajes **ST** y **FBD**.

La invocación de un bloque de función “C” desde el lenguaje **ST** se ajusta a las convenciones de invocación de bloques de función del lenguaje. Los parámetros de invocación del bloque están escritos después del nombre de la función, entre paréntesis y separados por comas. Se accede a los parámetros de retorno uno por uno. Cada parámetro de retorno está representado por un nombre, combinando el nombre de la instancia del bloque, y el nombre de los parámetros. Los componentes del nombre están separados por un punto. Por ejemplo, el nombre:

FBINSTNAME.parname

se utiliza para representar el parámetro de retorno llamado "**parname**", de la instancia de bloque de función llamada "**FBINSTNAME**".

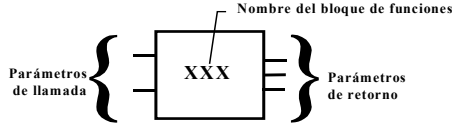
Las instancias de los bloques de función que se utilizan en el lenguaje ST tienen que estar declaradas en el diccionario. Cada copia (instancia) de un bloque de función debe estar identificada por un nombre único. A continuación se presenta un ejemplo de la declaración de instancias en el diccionario de ISaGRAF:

instancia:	TRIG1	tipo:	R_TRIG
	TRIG2		R_TRIG

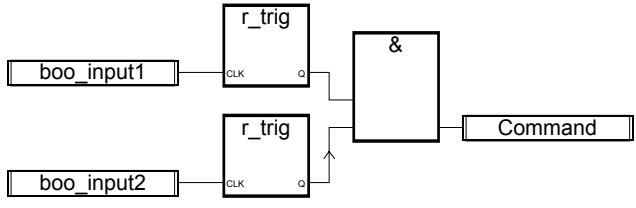
El siguiente ejemplo es de la utilización de estas instancias declaradas en un programa ST:

```
TRIG1 (boo_input1);
TRIG2 (boo_input2);
Command := (TRIG1.Q & TRIG2.Q);
```

Un programa **FBD** puede invocar a cualquier bloque de función "C". Se utiliza un bloque de función como una 'caja' estándar de funciones. Los parámetros de llamada están conectados al lateral izquierdo de la caja de funciones. Sus parámetros de retorno están conectados al lateral derecho de la caja. Una caja de funciones tiene el siguiente formato estándar:



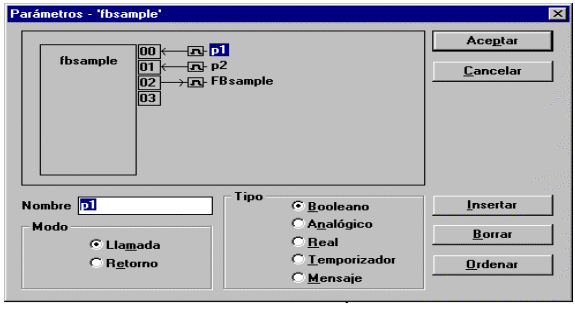
No se tienen que declarar los bloques de función utilizados en el lenguaje FBD, ya que el editor FBD de ISaGRAF declara las instancias de los bloques utilizados automáticamente. Las instancias de bloques de función declaradas automáticamente por el editor FBD de ISaGRAF son siempre **LOCALES** con respecto al programa editado. A continuación se presenta el ejemplo anterior, programado en el lenguaje FBD:



Definición de la interfaz de un bloque de función "C"

Se utiliza el comando "**Parámetros**" del menú "**Edición**" para definir los parámetros de llamada y retorno de un bloque de función nuevo. Un bloque de función puede tener hasta **32** parámetros, dispuestos libremente como parámetros de llamada o de retorno. A diferencia de

las funciones “C”, un bloque de función puede tener varios parámetros de retorno. Se describen los parámetros de un bloque de función “C” en la siguiente ventana de diálogo:



La lista que aparece en la parte superior de la ventana muestra los parámetros del bloque de función “C”, de acuerdo con el orden marcado por el prototipo de invocación de funciones: primero los parámetros de llamada y después los de retorno. La parte inferior de la ventana presenta una descripción detallada del parámetro que está seleccionado en la lista.:

- el nombre del parámetro
- la dirección (llamada/retorno) del parámetro
- el tipo de parámetro

Se puede utilizar cualquiera de los tipos de datos de ISaGRAF como un parámetro: Booleano, Analógico entero, Analógico real, Tiempo o Mensaje. Se tiene que diferenciar entre analógicos enteros y reales.
A continuación se presenta la correspondencia entre los tipos ISaGRAF y los tipos “C”:

BOOLEANO	unsigned long	Palabra entera de 32 bits, sin signo: 1=verdadero / 0=falso
ANALÓGICO	long	Palabra entera de 32 bits, con signo
REAL	float	Valor flotante, simple precisión
TEMPORIZADOR	unsigned long	Palabra entera de 32 bits, sin signo: (unidad es 1 milisegundo)
MENSAJE	char *	Cadena de caracteres

Cuando se transfiere el valor de un mensaje a una función “C”, no puede contener caracteres nulos. La cadena que se transfiere al código “C” tiene la terminación en nulo. No debe olvidarse que los parámetros de retorno tienen que ser los últimos de la lista. Se observarán las siguientes reglas a la hora de nombrar parámetros:

- la longitud del nombre no puede superar los 16 caracteres
- el primer carácter tiene que ser una letra
- los siguientes caracteres tiene que ser letras, dígitos o el carácter de subrayado ‘_’
- los nombres son insensibles al caso (mayúsculas/minúsculas)

No se puede utilizar el mismo nombre para más de un parámetro del bloque de función. Un parámetro de llamada no puede tener el mismo nombre que otro de retorno. Se **puede**

utilizar el mismo nombre para los parámetros de funciones diferentes. Se utiliza el nombre de un parámetro para identificar el parámetro en el código fuente “C”.

Se utiliza el comando "**Insertar**" para insertar un parámetro nuevo delante del parámetro seleccionado. Se utiliza el comando "**Borrar**" para borrar el parámetro seleccionado. El comando "**Ordenar**" reordena (clasifica) los parámetros de forma automática, colocando los parámetros de retorno al final de la lista. Se pulsa el botón "**Aceptar**" para guardar la definición de la interfaz del bloque de función y cerrar la ventana de diálogo. Se pulsa el botón "**Cancelar**" para cerrar la ventana de diálogo sin efectuar cambios en la definición de la interfaz del bloque de función.

Interfaz del bloque de función “C”

La interfaz de un bloque de función depende de la definición de sus parámetros. Los parámetros de llamada pasan a través de una estructura. Esta estructura está definida en el fichero "**GRFB0nnn.H**", donde "**nnn**" es el número lógico del bloque de función en la librería ISaGRAF. Los parámetros de retorno están representados por números lógicos, que también están definidos en el fichero "**GRFB0nnn.h**". A continuación se presenta un ejemplo de la interfaz “C” del bloque de función "**LIM_ALARM**" (alarma de límites):

```
/* Interfase function block - Nombre: sample */

/* tipos de datos estándar ISaGRAF */

typedef long                    T_BOO;
typedef long                    T_ANA;
typedef float                   T_REAL;
typedef long                    T_TMR;
typedef char                    *T_MSG;

/* estructura de parámetros de llamada */

typedef struct {
        /* CALL   */            T_BOO   _par1;
        /* CALL   */            T_BOO   _par2;
} str_arg;

/* acceso a campo de estructura str_arg */

#define PAR1                    (arg->_par1)
#define PAR2                    (arg->_par2)

/* retorno de números lógicos de parámetros */

#define FBLPNO_Q1               0
#define FBLPNO_Q2               1

/* fin de fichero */
```

En la siguiente tabla, se presenta la relación que existe entre los tipos ISaGRAF y los tipos "C". Los tipos ISaGRAF están definidos como tipos "C" en el fichero de definiciones de la función.

Booleano	T_BOO	long (32 bits)
Analógico	T_ANA	long
Real	T_REAL	float (32 bits – simple precisión)
Temporizador	T_TMR	long
Mensaje	T_MSG	char * (32 bits – puntero car)

Cada campo de la estructura "**str_arg**" corresponde a un parámetro del bloque de función. Los parámetros aparecen en la estructura en el mismo orden que se estableciera para la definición de los bloques de función. Se define un identificador en mayúsculas para facilitar el acceso directo a un parámetro de la estructura que se pasó a la implementación "C" del servicio de activación del bloque de función. Los nombres de los identificadores son aquellos que se introducen durante la definición del bloque de función, con el Gestor de Librerías de ISaGRAF.

El orden que se utiliza para la numeración de los parámetros de retorno es aquél que se estableciera para la definición del bloque de función. El número lógico del primer parámetro de retorno es siempre el **0**.

Se deben utilizar identificadores definidos en lugar de valores numéricos para representar los parámetros de retorno durante la programación fuente en "C". Se asegura de esta manera que el fichero fuente pueda volver a compilarse fácilmente después de una modificación de la interfaz.

Se actualiza el fichero de definiciones de "C" cada vez que se cambia la interfaz del bloque de función por medio del Gestor de Librerías de ISaGRAF. Se asegura así una coherencia absoluta entre la implementación del bloque de función y su utilización en los programas de las aplicaciones ISaGRAF.

Código fuente

La implementación en lenguaje "C" de un bloque de función está dividida en tres puntos de entrada diferentes:

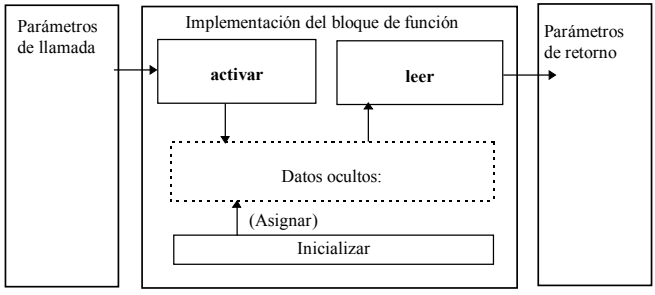
- ❑ servicio de inicialización
- ❑ servicio de activación – procesamiento de los parámetros de llamada
- ❑ servicio de lectura de parámetros retornados

Se utiliza el mismo código para cada instancia de un mismo bloque de función, y no se duplica. Se asocia una estructura de datos estáticos a cada instancia. La programación de ISaGRAF no puede acceder directamente a los datos de este tipo, que contienen las variables ocultas de la instancia de bloque de función.

El "servicio de activación" se invoca una vez para cada instancia de cada bloque que se utilice, en cada ciclo objeto. Procesa los parámetros de llamada y actualiza los datos asociados. Representa el "algoritmo principal" del bloque de función.

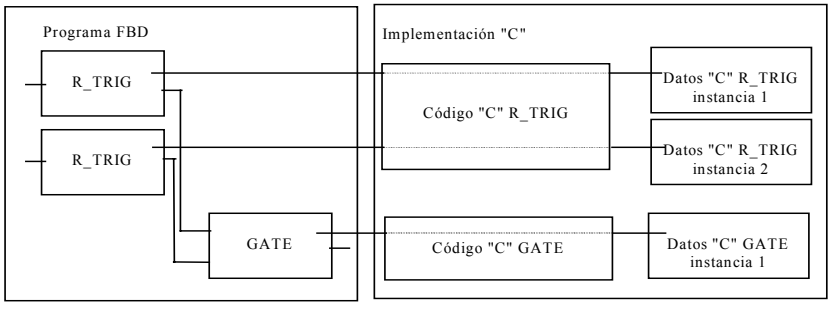
El *kernel* de ISaGRAF invoca al "servicio de lectura" para leer el valor actual de un parámetro de retorno de una instancia. Este servicio no tiene que realizar cálculos especiales; sólo acciona la transferencia entre los datos ocultos y la aplicación ISaGRAF.

Diagrama funcional:



• Datos estáticos del bloque de función

Un bloque de función asocia operaciones con datos estáticos. Una estructura de datos está asociada a cada instancia del mismo bloque de función. Cada vez que se utiliza un bloque de función en la programación en ST o FBD, corresponde a una instancia y a una estructura de datos. El siguiente ejemplo presenta la correspondencia entre las estructuras de datos "C" y las instancias de bloques de función que se utilizan en un programa FBD:



La memoria que se necesita para la estructura de datos de cada instancia está asignada por el sistema ISaGRAF, cuando arranca la aplicación. Se pasa un puntero que señala la estructura de datos de instancia asociada a los servicios de "activar" y "leer".

El Gestor de Librería de ISaGRAF genera automáticamente el *esqueleto* del código fuente "C" para la definición del tipo de estructura de datos. El tipo de estructura de datos siempre se llama "**str_data**". El programador no debe cambiar este nombre, para asegurar la compatibilidad con las cabeceras de servicio. Generalmente, los datos ocultos agrupan a las variables internas con una imagen de los parámetros de retorno. El servicio de 'lectura' de bloques de función sólo se utiliza para acceder al parámetro de retorno, y no se debe usar para llevar a cabo otras operaciones.

• Servicio de inicialización

El *kernel* de ISaGRAF invoca el servicio de "inicialización" de un bloque de función cuando arranca la aplicación. Le permite al programador "C" solicitar al sistema que asigne memoria para una instancia. A continuación se muestra la programación estándar del servicio de inicialización:

```
uint16 FBINIT_xxx (uint16 hinstance)
/* "xxx" es el nombre del bloque de función */
{
  return (sizeof (str_data));
}
```

El argumento "**hinstance**" es el número lógico de la instancia. Está reservado para las operaciones internas de ISaGRAF y no se puede utilizar en la programación del servicio. El servicio de inicialización retorna el número de bytes de memoria que son necesarios para los datos de **una** instancia. La cantidad de memoria que se necesita (valor de retorno) no puede superar los **64** kBytes. No debe realizarse ninguna otra operación con este servicio. El Gestor de Librerías de ISaGRAF genera el código fuente "C" de este servicio automáticamente cuando se crea el bloque de función.

• Servicio de activación

Se invoca el servicio de "activación" en cada ciclo objeto, para cada instancia de bloque de función que se utilice en la aplicación. Este servicio procesa los parámetros de llamada y ejecuta el algoritmo principal de bloque de función, con la finalidad de actualizar los datos estáticos ocultos y el valor de los parámetros de retorno. A continuación de presenta el *esqueleto* estándar del servicio de activación:

```
void FBACT_xxx (
uint16 hinstance,          /* "xxx" es el nombre del bloque de función */
                           /* número lógico de la instancia */
str_data *data,            /* puntero a la estructura de datos de la
                           instancia */
str_arg *arg               /* puntero a la estructura de los parámetros de
                           invocación */
)
{
}
```

El argumento "**hinstance**" es el número lógico de la instancia. Está reservado para las operaciones internas de ISaGRAF y no se puede utilizar en la programación del servicio. El argumento "**data**" es un puntero que señala la estructura de datos asociada a la instancia. El argumento "**arg**" es un puntero que señala la estructura que contiene el valor de los parámetros de llamada. El programador deberá utilizar los identificadores que están definidos en la cabecera "C" del bloque de función para poder acceder a los campos de la estructura "**arg**".

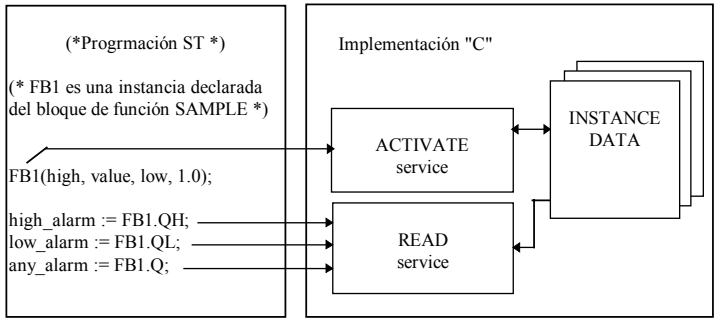
El algoritmo "activación" procesa los parámetros de llamada (almacenados en la estructura "**arg**") y actualiza los campos de la estructura "**data**". El siguiente ejemplo muestra el servicio de "activación" del bloque de función **TRIG** (detección de flanco de subida):

```
/* definiciones almacenadas en la cabecera C del bloque de función */

typedef struct {                        /* parámetros de llamada */
        T_BOO  _clk;                    /* entrada de disparo */
} str_arg;

#define CLK      (arg->_clk)

/* estructura de datos de instancia de bloque de función */

typedef struct {
        T_BOO prev_state;               /* estado previo de la entrada de disparo */
        T_BOO edge_detect;              /* valor de flanco: imagen de parámetro de retorno
                                        */
} str_data;

/* servicio de activación */

void FBACT_trig (uint16 hinstance, str_data *data, str_arg *arg)
{
        data->edge_detect = (T_BOO)(CLK && !data->prev_state);
        data->prev_state = CLK;  /* parámetro de llamada */
}
```

El Gestor de Librerías de ISaGRAF genera el *esqueleto* del código fuente "C" de este servicio automáticamente, cuando se crea el bloque de función.

- **Lectura de los parámetros de retorno**

Se invoca el servicio de "leer" cada vez que se referencia el parámetro de retorno de una instancia de bloque de función en un programa ST o FBD. Se utiliza para obtener el valor de **un** parámetro de retorno. El siguiente ejemplo muestra las invocaciones del servicio de "leer" que se producen mientras se ejecuta un programa ST:



Teniendo en cuenta que el servicio “leer” puede ser invocado en más de una ocasión durante el mismo ciclo, con relación al mismo parámetro de retorno o la misma instancia de bloque de función, no se tienen que realizar cálculos especiales en un servicio de este tipo. Se limita a accionar la transferencia entre datos ocultos y la aplicación ISaGRAF. A continuación se muestra el *esqueleto* estándar del servicio de lectura:

```
/* conversión de tipo usada para copiar el valor de un parámetro de retorno */

#define BOO_VALUE          ((T_BOO *)value)
#define ANA_VALUE          ((T_ANA *)value)
#define REAL_VALUE         ((T_REAL *)value)
#define TMR_VALUE          ((T_TMR *)value)
#define MSG_VALUE          ((T_MSG *)value)

/* servicio de lectura de parámetros de retorno: llamado por cada parámetro de retorno */

void FBREAD_xxx (          /* "xxx" es el nombre del bloque de función */
uint16 hinstance,          /* número lógico de instancia */
str_data *data,            /* puntero a estructura de datos de instancia */
uint16 parno,              /* número lógico de parámetro leído */
void *value)               /* buffer de copia de valor de parámetro */
{
        switch (parno) {
            case FBLPNO_XX: /* ... */ break;
            case FBLPNO_YY: /* ... */ break;
             /* .... */
        }
}
```

El argumento "**hinstance**" es el número lógico de la instancia. Está reservado para las operaciones internas de ISaGRAF y no se puede utilizar en la programación del servicio. El argumento “**data**” es un puntero que señala la estructura de datos asociada a la instancia.

El argumento "**parno**" es el número lógico del parámetro de retorno cuyo valor se desea conocer. Utilizar los identificadores que están definidos en la cabecera “C” del bloque de

función para identificar los parámetros de retorno. Este tipo de identificador comenzará con el prefijo "**FBLPNO_**". El argumento "**value**" es un puntero que señala el *buffer* (memoria intermedia) en el que se debe copiar el valor actual del parámetro de retorno al cual se ha accedido. El tipo de dato al que se señala con este argumento depende del tipo ISaGRAF del parámetro de retorno. La siguiente tabla ilustra la relación que existe entre los tipos ISaGRAF y los tipos de datos de un *buffer* "C":

Booleano	long	Palabra de 32 bits, sin signo 1=verdadero / 0=falso
Analógico	long	Palabra de 32 bits, con signo
Real	float	Valor flotante de 32 bits, simple precisión
Temporizador	long	Palabra de 32 bits, sin signo (unidad es 1 ms)
Mensaje	char *	Cadena de caracteres

Se utilizan las siguientes macros para acceder al *buffer* de copias, de acuerdo con el tipo de parámetro de retorno al que se haya accedido:

```
#define     BOO_VALUE     ((T_BOO *)value)
#define     ANA_VALUE     ((T_ANA *)value)
#define     REAL_VALUE    ((T_REAL *)value)
#define     TMR_VALUE     ((T_TMR *)value)
#define     MSG_VALUE     ((T_MSG *)value)
```

A continuación se muestran algunas operaciones que se emplean con frecuencia para copiar el valor o el parámetro al *buffer* de ISaGRAF:

```
/* para un parámetro booleano: */
  *BOO_VALUE = parameter_value;
/* para un parámetro analógico entero:*/
  *ANA_VALUE = parameter_value;
/* para un parámetro analógico real:*/
  *REAL_VALUE = parameter_value;
/* para un parámetro temporizador:*/
  *TMR_VALUE = parameter_value;
/* para un parámetro cadena:*/
  strcpy (*MSG_VALUE, parameter_value);
```

El Gestor de Librerías de ISaGRAF genera el *esqueleto* del código fuente "C" de este servicio automáticamente, cuando se crea el bloque de función.

• Ejemplo de fichero fuente "C"

El *esqueleto* estándar para la implementación de un bloque de función "C" tiene el siguiente aspecto:

```
/* bloque de función (xxx es el nombre del bloque de función) */

#include <tasy0def.h>
```

```
#include <grfb0nnn.h>   /* nnn es el número del bloque de función en librería */

/* estructura de datos ocultos para cada instancia del bloque */
typedef struct {
        /* definición de campos */
} str_data;

/* servicio de activación: retorna el tamaño de los datos ocultos necesitados */
word FBINIT_xxx (uint16 hinstance)
{
        return (sizeof (str_data));
}

/* servicio de activación: procesa los parámetros de llamada */
void FBACT_xxx (uint16 hinstance, str_data *data, str_arg *arg)
{
        /* ... */
}

/* conversión de tipo para copiar el valor de un parámetro de retorno */
#define BOO_VALUE   ((T_BOO *)value)
#define ANA_VALUE   ((T_ANA *)value)
#define REAL_VALUE  ((T_REAL *)value)
#define TMR_VALUE   ((T_TMR *)value)
#define MSG_VALUE   ((T_MSG *)value)

/* servicio de lectura de parámetros de retorno: llamada por cada parámetro retornado */
void FBREAD_xxx (uint16 hinstance, str_data *data, uint16 parno, void *value)
{
        switch(parno)
        {
          case FBLPNO_XX: *???_VALUE = ...; break;
          case FBLPNO_YY: *???_VALUE = ...; break;
        ....

}
```

/* La siguiente función se utiliza para la inicialización del bloque de función y la declaración de su implementación. Realiza el enlace con el kernel ISaGRAF, usando el nombre del bloque de función. Este servicio es generado completamente por el Gestor de Librerías de ISaGRAF. */

```
ABP fbldef_xxx (char *name, IBP *initproc, RBP *readproc)
{
        strcpy (name, "XXX");
```

```
        *initproc = (IBP)FBINIT_xxx;
        *readproc = (RBP)FBREAD_xxx;
        return ((ABP)FBACT_xxx);
}

/* fin de fichero */
```

Se requiere el fichero "**TASY0DEF.H**", del *kernel* de ISaGRAF, para las definiciones dependientes del sistema. También contiene la definición de los tipos de datos que representan punteros que señalan los servicios implementados.

Enlaces entre proyectos e implementación "C"

El enlace lógico entre la implementación de un bloque de función "C" y su utilización en los programas de un proyecto ISaGRAF se realiza por medio del nombre de la función. Se añade un servicio de "declaración" al código fuente "C" del bloque de función. Este servicio, que sólo se invoca una vez al arrancarse la aplicación, indica al *kernel* de ISaGRAF el nombre del bloque de función "C" correspondiente a los servicios implementados. Un servicio de declaración tiene el siguiente formato estándar:

```
ABP fbldef_xxx (char *name, IBP *initproc, RBP *readproc)
{
        strcpy (name, "XXX");                   /* nombre del bloque de función */
        *initproc = (IBP)FBINIT_xxx;            /* servicio de inicialización */
        *readproc = (RBP)FBREAD_xxx;            /* servicio de lectura */
        return ((ABP)FBACT_xxx);                /* servicio de activación*/
}
/* xxx es el nombre del bloque de función */
```

Cuando se utiliza en la sentencia **strcpy**, se escribe el nombre del bloque de función en **mayúsculas**. Se tiene que utilizar minúsculas para el nombre de los servicios implementados y el nombre del servicio de declaración.

La utilización de los prefijos "**FBACT_**", "**FBINIT_**", "**FBREAD_**" y "**fbldef_**" para los servicios implementados y el servicio de definición le permite al usuario nombrar un bloque de función con una palabra clave reservada al lenguaje "C", o el nombre de una función ya existente en las librerías "C" de ISaGRAF. No se debe añadir ninguna otra sentencia al servicio de declaración.

Se invoca la función de declaración para cualquier bloque de función "C" integrado, incluso si no se utiliza en los programas de la aplicación ISaGRAF. El *kernel* de ISaGRAF detectará un error fatal si un bloque de función "C" que se utiliza en la aplicación no está integrado en el *kernel*.

Antes de enlazar nuevos bloque de función con el *kernel*, el usuario tiene que escribir otro fichero fuente en "C", llamado "**GRFB0LIB.C**", e insertarlo (junto con los bloques de función) en la lista de ficheros para el enlace. El fichero "**GRFB0LIB.C**" sólo contiene un vector de servicios de declaración. Se lee este vector durante las inicializaciones de la aplicación, para establecer un enlace dinámico con los bloque de función escritos en "C". A continuación se muestra un ejemplo de un fichero de este tipo:

```
/* Fichero: grfb0lib.c - bloques de función implementados */

#include <tasy0def.h>

extern ABP fbldef_fb1(char *name, IBP *init, RBP *read);
extern ABP fbldef_fb2(char *name, IBP *init, RBP *read);

FBL_LIST FBLDEF[ ] = {
        fbldef_fb1,
        fbldef_fb2,

NULL };

/* fin de fichero */
```

Se tiene que finalizar el vector **FBLDEF** por medio de un puntero NULO. Pueden producirse conflictos si no se cumple con esta condición. Se producirán referencias no resueltas a la hora de enlazar el nuevo *kernel* de ISaGRAF si no está definido el vector **FBLDEF**.

Al escribir este fichero se puede crear un nuevo *kernel*, incluyendo todos los bloques de función existentes. También se puede construir un *kernel* personalizado para un único proyecto, mediante la inserción en el vector **FBLDEF** de sólo los bloques de función que se utilizan en el proyecto. El Generador de Código de ISaGRAF genera el fichero "**GRFB0LIB.C**" automáticamente cuando se construye el código de una aplicación. El fichero, que se ubica en el directorio del proyecto ISaGRAF, abarca únicamente los bloques de función que se utilizan en el proyecto.

Límites

La librería de ISaGRAF puede contener hasta **255** bloques de función "C". Se puede procesar cualquier tipo de operación con una función. Se puede copiar (instanciar) cada tipo de bloque de función hasta **255** veces en el mismo proyecto.

Se debe tener en cuenta que los servicios de bloques de función se invocan en el ciclo ISaGRAF de manera **síncrona**, por lo que la ejecución del bloque de función tiene un efecto directo sobre los tiempos de ciclo.

Ejemplo completo

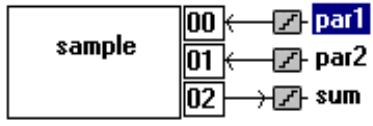
A continuación se presenta la programación completa de una función de muestra titulada "**sample**", que es un contador ascendente. Se encuentran las notas técnicas del bloque de función al final.

nombre:	SAMPLE
descripción:	Contador ascendente
fecha de creación:	1 de febrero de 1994
autor:	CJ international

llamada	CU : contar entradas R : resetear comando PV : valor máx. programado
retorno:	Q : detección máx. CV : contar resultado

prototipo:	SAMPLE (count, reset_command, maximum_value);
	max_detect := SAMPLE.Q;
	count_result := SAMPLE.CV;

La interfaz del bloque de función tiene este aspecto:



A continuación se muestra la cabecera fuente "C" del bloque de función:

```
/* Interfase de bloque de función - Nombre: SAMPLE */

/* definición de tipos de datos estándar de ISaGRAF */

typedef long  T_BOO;
typedef long  T_ANA;
typedef float T_REAL;
typedef long  T_TMR;
typedef char  *T_MSG;

/* definición de estructura de parámetros de llamada */

typedef struct {
  T_BOO  _cu;
  T_BOO  _r;
  T_ANA  _pv;
} str_arg;

/* identificadores usados para acceso a los parámetros de llamada */

#define CU                    (arg->_cu)
#define R                     (arg->_r)
#define PV                    (arg->_pv)

/* numeración lógica de los parámetros de retorno */

#define FBLPNO_Q              0
#define FBLPNO_CV             1
```

/* *fin de fichero* */

A continuación está el código fuente "C" del bloque de función. Sólo las líneas impresas con caracteres en negrita fueron introducidas manualmente por el programador de "C".

```
/* Bloque de función - Nombre: SAMPLE */

#include <tasy0def.h>                   /* requerido para definición de tipos de datos */
#include <grfb0255.h>                   /* cabecera fuente de bloque de función */

/* definición de estructura que contiene los datos de una instancia */

typedef struct {
        T_BOO overflow;                 /* verdadero: valor de conteo >= valor programado*/
        T_ANA value;                    /* valor de conteo actual */
} str_data;

/* servicio de inicialización: requiere memoria para datos de instancia */

word FBINIT_sample (uint16 hinstance)
{
        return (sizeof (str_data));
}

/* servicio de activación: algoritmo de conteo creciente */

void FBACT_sample (uint16 hinstance, str_data *data, str_arg *arg)
{
        if (R) data->value = 0;
        else if (CU && data->value < PV) (data->value)++;
        data->overflow = (data->value >= PV) ? (T_BOO)1 : (T_BOO)0;
}

/* conversión de tipo requerida para copiar parámetros al buffer de ISaGRAF */

#define BOO_VALUE   ((T_BOO *)value)
#define ANA_VALUE   ((T_ANA *)value)
#define REAL_VALUE  ((T_REAL *)value)
#define TMR_VALUE   ((T_TMR *)value)
#define MSG_VALUE   ((T_MSG *)value)

/* servicio de lectura: consigue el valor de un parámetro de retorno */

void FBREAD_sample (uint16 hinstance, str_data *data, uint16 parno, void *value)
{
```

```
        switch (parno) {
          case FBLPNO_Q   : *BOO_VALUE = data->overflow; break;
          case FBLPNO_CV : *ANA_VALUE = data->value; break;
        }
}

/* servicio de declaración usado para enlace dinámico con el kernel ISaGRAF */

ABP fbldef_sample (char *name, IBP *initproc, RBP *readproc)
{
        strcpy (name, "SAMPLE");
        *initproc = (IBP)FBINIT_sample;
        *readproc = (RBP)FBREAD_sample;
        return ((ABP)FBACT_sample);
}

/* fin de fichero */
```

C.7.5 Técnicas de compilación y de enlazado

El banco de trabajo de ISaGRAF no incluye un compilador o enlazador “C”. Sin embargo, este capítulo explica las técnicas principales que se pueden aplicar para utilizar los ficheros que se crean con el Gestor de Librerías de ISaGRAF y pasarlos a otras herramientas tales como compiladores y enlazadores.

Ficheros fuente "C"

El Gestor de Librerías de ISaGRAF coloca los ficheros fuente "C" de las conversiones, las funciones y los bloques de función en los directorios **ISAWIN\LIB\DEFS** y **ISAWIN\LIB\SRC**. El nombre de un fichero fuente se construye con el número de la conversión, la función o el bloque de función correspondiente en la librería de ISaGRAF. Se utilizan los siguientes nombres de fichero:

\isawin\lib\defs\TACN0DEF.H	fichero de definición de cualquier función de conversión
\isawin\lib\src\GRCN0nnn.H	fichero fuente de una función de conversión
\isawin\lib\defs\GRUS0nnn.H	fichero de definición de una función
\isawin\lib\src\GRUS0nnn.C	fichero fuente de una función
\isawin\lib\defs\GRFB0nnn.H	fichero de definición de un bloque de función
\isawin\lib\src\GRFB0nnn.C	fichero fuente de un bloque de función

(**nnn** es el número de la conversión, la función o el bloque de función)

Advertencia: Al renombrar o copiar elementos de librería, el Gestor de Librerías no actualiza su texto o líneas de programación según el nuevo nombre de elemento y número lógico. Se deben actualizar manualmente en el fichero fuente “C”.

El fichero **\ISAWIN\LIB\USPNUMS** proporciona la relación entre nombres y números lógicos para las funciones "C" que existan en la librería ISaGRAF. Un ejemplo de este tipo de fichero es:

```
1       funct_A
10      funct_B
16      funct_C
```

El fichero **\ISAWIN\LIB\FBLNUMS** proporciona la relación entre nombres y números lógicos para los bloques de función "C" que existan en la librería ISaGRAF. Un ejemplo de este tipo de fichero es:

```
0       fbl_A
1       fbl_B
2       fbl_C
```

El fichero **\ISAWIN\LIB\CNVNUMS** proporciona la relación entre nombres y números lógicos para las funciones de conversión que existan en la librería ISaGRAF. A modo de ejemplo, se muestra a continuación el contenido de este fichero para las conversiones de la librería estándar:

```
0       SCALE
1       BCD
```

El Gestor de Librerías de ISaGRAF actualiza estos ficheros automáticamente cada vez que se crea, se renombra, se copia o se borra una conversión, una función o un bloque de función. El Generador de Código de ISaGRAF genera los siguientes ficheros de forma automática cuando se construye una aplicación:

\isawin\apl\ppp\GRCN0LIB.C	Declaración como vector de todas las funciones de conversión utilizadas en el proyecto.
\isawin\apl\ppp\GRUS0LIB.C	Declaración como vector de todas las funciones utilizadas en el proyecto.
\isawin\apl\ppp\GRFB0LIB.C	Declaración como vector de todos los bloques de función utilizados en el proyecto.

(**ppp** es el nombre del proyecto ISaGRAF)

Se pueden utilizar estos ficheros durante las operaciones de enlace para crear un nuevo *kernel* de ISaGRAF dedicado al proyecto, que contenga sólo las conversiones, las funciones y los bloques de función que se utilizan en el proyecto.

Carga de ficheros fuente en un sistema nativo

Se pueden cargar los ficheros fuente y de definición en "C" creados por el Gestor de Librerías de ISaGRAF al sistema ISaGRAF objeto, siempre y cuando soporte una herramienta nativa de compilación. Para ello, se puede utilizar la herramienta estándar de **TERMINAL** que se suministra dentro de Windows.

Cuando se gestionan ficheros fuente en el sistema objeto, se tienen que actualizar los ficheros de definición mediante una nueva operación de carga cada vez que el Gestor de Librerías de ISaGRAF modifica una interfaz de función.

Las líneas de comando que se utilizan para efectuar la carga de ficheros pueden agruparse, por ejemplo, en un fichero de comandos que se puede ejecutar posteriormente desde el menú de herramientas del banco de trabajo (véase “Manual del Usuario: Gestión de Programas”).

Utilización de un compilador cruzado

Así mismo, se pueden gestionar los ficheros fuente directamente en el PC del usuario – si el objeto es un PC – o si se dispone de un compilador cruzado que se esté ejecutando en el PC y generando código para el sistema objeto.

En este caso, el usuario puede ejecutar el Gestor de Librerías de ISaGRAF para completar y modificar las fuentes de conversiones, funciones o bloques de función.
Las líneas de comando que se utilizan para ejecutar el compilador y el enlazador pueden agruparse, por ejemplo, en un fichero de comandos que se puede ejecutar posteriormente desde el menú de herramientas del banco de trabajo (véase “Manual del Usuario: Gestión de Programas”).

Cuando se compilan las conversiones, las funciones o los bloques de función en el PC, el usuario sólo tiene que efectuar la carga del nuevo *kernel* de ISaGRAF (enlazado con nuevos componentes) en el sistema objeto antes de ejecutar aplicaciones. Si el objeto es otro PC, se puede cargar el nuevo *kernel* de ISaGRAF en el equipo objeto por medio de un disquete o a través de una conexión de red.

Enlace con las librerías del kernel de ISaGRAF

Advertencia:
La siguiente información es de carácter general y existe la posibilidad de que no corresponda con exactitud al sistema objeto del usuario.

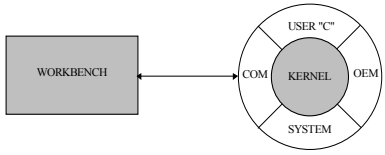
En cualquier caso, consúltense los ficheros ‘léeme’ y .TXT que se suministran en el disco duro del sistema objeto.

El disquete del objeto ISaGRAF contiene numerosas utilidades para compilar y enlazar las conversiones, las funciones y los bloques de función con las librerías del *kernel* de ISaGRAF.

Existen dos implementaciones:
- ISaGRAF monotarea: se realizan todas las funciones en el mismo programa
- ISaGRAF multitarea: se dedica una tarea independiente a la comunicación

En ambos casos, los componentes “C” están agrupados en las mismas librerías: para el programador “C”, no se distingue entre monotarea y multitarea. Para las versiones monotarea, las librerías “C” del usuario están enlazadas con la tarea simple (generalmente conocida como **isa**), mientras que para la versión multitarea las versiones están enlazadas con la tarea del *kernel* (generalmente conocida como **isaker**).

Sistema de desarrollo	**Sistema objeto**

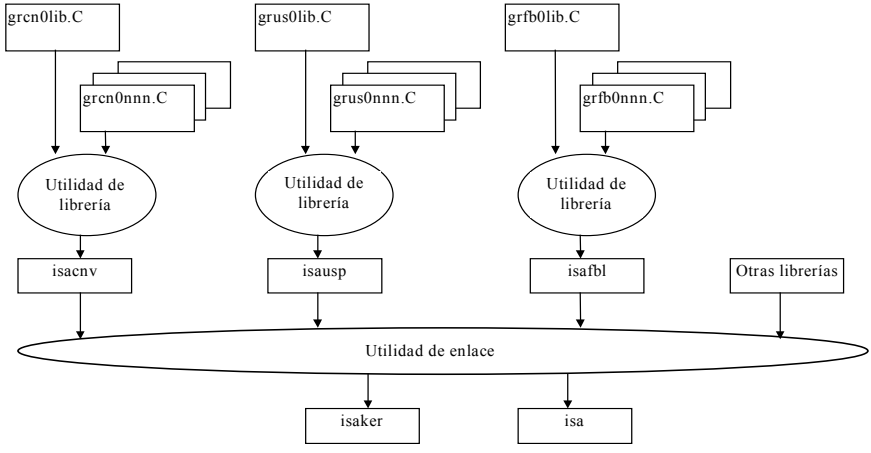


La parte interior del *software* ISaGRAF es independiente del *hardware*. Ejecuta los lenguajes IEC y tiene su propia base de datos de variables.

El primer paso, cuando se está haciendo el enlace con el kernel, es el de construir librerías de todas las conversiones, las funciones y los bloques de función que se vayan a necesitar para el proyecto específico.

Librería	Contenido
ISAUSP	- fichero objeto GRUS0LIB (vector de funciones declaradas) - fichero objeto de cada función que se haya integrado
ISAFBL	- fichero objeto GRFB0LIB (vector de bloques de función declarados) - fichero objeto de cada bloque de función que se haya integrado
ISACNV	- fichero objeto GRCN0LIB (vector de conversiones declaradas) - fichero objeto de cada función de conversión que se haya integrado

Después, el programador tiene que vincular estas nuevas librerías con los demás ficheros y librerías objeto del *kernel* de ISaGRAF. En el siguiente diagrama, se perfilan las diferentes fases de la integración de un desarrollo en "C" del usuario:



A continuación se muestra la lista exacta de módulos y librerías objeto que se tienen que unir durante el enlace:

Para construir **isaker**:
Módulo objeto: **tast0mai**

Módulo objeto:	**tats0com**	
Librería *kernel*:	**isaker**	
Librería *kernel*:	**isaoem**	
Librería usuario:	**isausp**	funciones definidas por el usuario
Librería usuario:	**isafbl**	bloques de función definidos por el usuario
Librería usuario:	**isacnv**	funciones de conversión definidas por el usuario
Librería *kernel*:	**isasys**	
Librerías del sistema:	(véase el manual del compilador "C")	

Para construir **isa**:

Módulo objeto:	**tast0mai**	
Módulo objeto:	**tast0com**	
Librería *kernel*:	**isaker**	
Librería *kernel*:	**isatst**	
Librería *kernel*:	**isaoem**	
Librería usuario:	**isausp**	funciones definidas por el usuario
Librería usuario:	**isafbl**	bloques de función definidos por el usuario
Librería usuario:	**isacnv**	funciones de conversión definidas por el usuario
Librería *kernel*:	**isasys**	
Librerías del sistema:	(véase el manual del compilador "C")	

Puede que el programador tenga que seguir el orden exacto de los módulos y librerías objeto que se presentan en estas tablas. Los módulos y librerías objeto tienen extensiones estándares (**".lib"**, **".obj"**, **".l"**, **".r"**...) según el sistema objeto.

Opciones necesarias de compilación y enlace

Pueden seleccionarse opciones de conveniencia durante el proceso de compilación y enlace. Estas dependen del tipo de operaciones procesadas en las conversiones, las funciones o los bloques de función. Algunas operaciones requieren otras librerías de sistema (matemáticas, gráficas, etc.) durante el proceso de enlace.

Todos los ficheros fuente "C" del *kernel* de ISaGRAF han sido compilados según el modelo de memoria **LARGE**. El programador debe disponer del mismo modelo para la compilación de conversiones, funciones y bloques de función.

Se tiene que definir una constante especial para la compilación de los componentes de librerías "C". Indica el tipo de sistema y procesadores objeto para que la fuente de conversiones, funciones y bloques de función pueda ser independiente del sistema. Estos valores constantes tienen los siguientes nombres:

DOS.....para sistemas basados en DOS (procesador INTEL)
ISAWNT .....para sistemas basados en Windows-NT (procesador INTEL)
OS9 .....para el sistema OS-9 (procesador MOTOROLA)
VxWorks .....para un sistema VxWorks (procesador MOTOROLA)

Los ficheros de comando de las utilidades (para compilar y enlazar) que se suministran con el *software* ISaGRAF objeto enseñan al usuario la manera de definir el valor constante más apropiado en la línea de comando del compilador.

Compiladores soportados

Se soportan los siguientes compiladores para el desarrollo de conversiones, funciones y bloques de función, y para el enlace con el *kernel* de ISaGRAF:

Compilador Microsoft MSC 7.00	para objetos basados en DOS
Compilador Microsoft MSVC 4.00	para objetos basados en Windows-NT
Compilador Microware ULTRA-C	para objetos OS-9
Tornado 1.0; GNU Toolkit 2.6	para objetos VxWorks

Contactar con CJ International para más información sobre el uso de otros compiladores.

Resumen

A continuación se facilita un resumen de las operaciones que deben llevarse a cabo al desarrollar una conversión, una función o un bloque de función nuevo.

⇨**1.** Con el Gestor de Librerías de ISaGRAF, crear el nuevo elemento: asignarle un nombre y un texto de comentario. El *esqueleto* del fichero fuente “C” se genera automáticamente.

⇨**2.** Con el Gestor de Librerías de ISaGRAF, describir la interfaz (los parámetros de llamada y retorno), si el elemento es una función o un bloque de función. El fichero de cabecera fuente “C” se genera automáticamente.

⇨**3.** Con el Gestor de Librerías de ISaGRAF, introducir el texto de las notas técnicas detalladas (manual del usuario) del elemento.

⇨**4.** Con el Gestor de Librerías de ISaGRAF, completar el fichero fuente "C" mediante la introducción de la programación “C” del algoritmo de la conversión, la función o el bloque de función. El código fuente el elemento ya está completo. Observe que se puede utilizar otro editor.

⇨**5.** Seleccionar la opción "**Mostrar número lógico**" del Gestor de Librerías para saber qué número lógico está asociado al nuevo elemento. Se utiliza este número en los nombres de las rutas de acceso de los correspondientes ficheros fuente “.C” y “.H”.

⇨**6.** Copiar / Transferir los ficheros .C y .H al objeto (en el caso de un compilador nativo) o al entorno (en el caso de un compilador cruzado) en el cual se han instalado las librerías y tareas del objeto ISaGRAF.

⇨**7.** Ejecutar el compilador “C” sobre el nuevo fichero fuente y corregir errores de sintaxis si los hubiera.

⇨**8.** Insertar el nombre del servicio de declaración de elementos nuevos en el fichero fuente "**GR??0LIB.C**", que define el vector de elementos que están insertados.

⇨**9.** Ejecutar el compilador "C" para compilar el fichero "**GR??0LIB.C**".

⇨**10.** Insertar el nombre del módulo objeto en la lista de ficheros objeto que se utilizan para construir la librería correspondiente.

⇨**11.** Ejecutar el ‘constructor’ de librerías “C”. Ejecutar el enlazador “C” para construir el *kernel* nuevo.

⇨**12.** Instalar el *kernel* recién creado en la máquina objeto.

⇨**13.** Escribir una aplicación ISaGRAF de ensayo para comprobar la activación y la interfaz del elemento nuevo.

C.8 Enlace Modbus

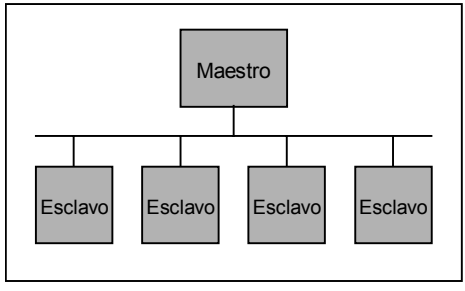
Una vez que la aplicación está completamente desarrollada y probada, se puede conectar a un sistema de visualización de procesos.

ISaGRAF es un sistema abierto que ofrece una gran variedad de posibilidades de operación en red.
La red industrial más sencilla es el protocolo estándar MODBUS/MODICON, que está disponible en prácticamente todos los sistemas de visualización de procesos y que sólo necesita un enlace vía serie (RS232, RS485, Bucle de corriente).

El protocolo del depurador de comunicaciones de ISaGRAF es compatible con MODBUS para permitir el acceso a variables en modo leer/escribir desde una estación maestra de Modbus.

C.8.1 Red y protocolo MODBUS

Una red Modbus está compuesta por una única estación maestra (que suele ser un sistema de visualización de procesos) y una o más estaciones esclavas (que suelen ser PLCs).



El maestro envía las peticiones de uno en uno a cada esclavo (utilizando su número de esclavo) y espera que el esclavo conteste antes de enviarle la siguiente pregunta. Los esclavos que no están afectados no responden.

Cada *trama* contiene un número de esclavo, un número de petición y los correspondientes datos de petición, y un valor de comprobación de 16 bits (CRC).

Si no se recibe una respuesta antes de transcurrir un determinado tiempo de espera, se puede repetir la pregunta un cierto número de veces antes de que el maestro declare al esclavo “desconectado”.

El valor del tiempo de espera y el número de nuevos intentos tienen que ajustarse en la estación maestra, para adaptarlos a los requisitos del esclavo (dependiendo de la aplicación, etc.).

Si se produce un error en el procesamiento de una interrogación, el esclavo puede generar un mensaje de error en lugar de enviar la *trama* de respuesta que se esperaba.

Modbus es un protocolo Modicon, pero no es un estándar internacional. Existen muchas implementaciones diferentes de protocolos compatibles con 'Modbus', con muchas diferencias posibles tales como:

– Lista de códigos de función implementados
– Mapeo de direcciones
– Protocolo RTU (códigos binarios) o ASCII
etc...

C.8.2 Implementación en ISaGRAF

Acceso a las variables de la aplicación

El enlace de comunicación de ISaGRAF reconoce cinco códigos de función Modbus:

1	Leer N bits
3	Leer N palabras
5	Escribir 1 bit
6	Escribir 1 palabra
16	Escribir N palabras

Se puede acceder a las variables de la aplicación ISaGRAF por medio de sus "direcciones de red", siempre y cuando hayan sido definidas en el diccionario del banco de trabajo. Estas variables pueden ser:
- Variables booleanas o analógicas
- Variables de entrada, de salida o internas
- Variables locales o globales

Para escribir una variable booleana, se pueden utilizar las funciones 5, 6 ó 16. A la hora de escribir, un valor VERDADERO es cualquier valor no igual a cero.

Para leer una variable booleana, se pueden utilizar las funciones 1 ó 3. Con la función 1, se recuperan los valores en un campo de bits; con la función 3, se recuperan en Bytes (un valor VERDADERO corresponde a 0xFFFF).

Para escribir una variable analógica, se pueden utilizar las funciones 6 ó 16. El valor es un valor entero de 16 bits entre -32768 y +32767 (las variables del objeto ISaGRAF son de 32 bits).

Para leer una variable analógica, se debe utilizar la función 3. El valor leído es un valor entero de 16 bits entre -32768 y +32767. Al nivel de objeto, las variables analógicas son de 32 bits, por lo que un valor en el objeto que supere el rango de los 16 bits (positivo o negativo) será leído con un valor de rango máximo de 16 bits (positivo o negativo).

No se puede acceder a las variables reales con una petición Modbus.

Advertencia:
La implementación ISaGRAF no gestiona los códigos de error tales como “dirección modbus desconocida”.

Notaciones:

slv	Número de esclavo
nbw	Número de palabras
nbb	Número de bytes
nbi	Número de bits
addH	Dirección de red (Byte alto)
addL	Dirección de red (Byte bajo)
vH	Valor (Byte alto)
vL	Valor (Byte bajo)
V	Valor Byte
bfd	Campo de bits (Bytes nbb)
crcH	Valor de comprobación (Byte alto)
crcL	Valor de comprobación (Byte bajo)

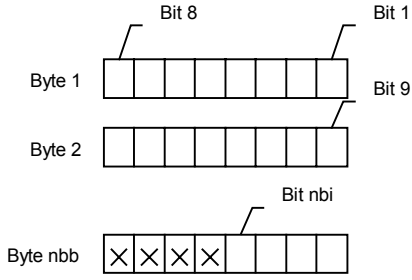
FUNCIÓN 1: leer N bits

Leer bits nbi (Booleanos), partiendo de la dirección de red addH/addL

Pregunta	slv	01	addH	addL	00	nbi	crcH	crcL

Respuesta	slv	01	nbb	bfd	...		crcH	crcL
				Byte 1		Byte nbb		

bfd es un campo de bits de Bytes nbb, con el siguiente formato:



Bit 1 corresponde al valor de la variable en la dirección de red addH/addL.
Bit nbi corresponde al valor de la variable en la dirección de red addH/addL + nbi -1.
X significa un valor indefinido.

FUNCIÓN 3: leer N palabras

Leer palabras nbw, partiendo de la dirección de red addH/addL

Pregunta	slv	03	addH	addL	00	nbw	crcH	crcL

Respuesta	slv	03	nbb	vH	vL	...	crcH	crcL

nbb corresponde al número de bytes vH, vL

FUNCIÓN 5: escribir 1 bit

Escribir un bit (Booleano) en la dirección de red addH/addL

Pregunta	slv	05	addH	addL	vH	00	crcH	crcL

Respuesta	slv	05	addH	addL	vH	00	crcH	crcL

FUNCIÓN 6: escribir 1 palabra

Escribir una palabra en la dirección de red addH/addL

Pregunta	slv	06	addH	addL	vH	vL	crcH	crcL

Respuesta	slv	06	addH	addL	vH	vL	crcH	crcL

FUNCIÓN 16: escribir N palabras

Escribir palabras nbw, partiendo de la dirección de red addH/addL (nbb = 2nbw)

Pregunta	slv	10	addH	addL	00	nbw	nbb	vH	vL	...	crcH	crcL

Respuesta	slv	10	addH	addL	00	nbw	CrcH	crcL

Ejemplos:

- Función 1: leer 15 bits, partiendo de la dirección de red 0x1020, en el esclavo 1

Pregunta	01	01	10	20	00	0F	79	04

Respuesta	01	01	02	00	12	39	F1

El valor leído es 0x0012, que da 00000000 00010010 como campo de bits.
En este ejemplo, las variables 0x1029 y 0x102C son VERDADERAS y todas las demás son FALSAS.

- Función 16: escribir 2 palabras en la dirección 0x2100, en el esclavo 1. Los valores escritos son 0x1234 y 0x5678.

Pregunta	01	10	21	00	00	02	04	12	34	56	78	1C	CA

Respuesta	01	10	21	00	00	02	4B	F4

Transferencia de ficheros

En comparación con los buses de campo modernos, el protocolo Modbus ofrece unos servicios muy pobres si no se amplía con códigos de funciones específicos de los fabricantes.

En nuestra situación, en la que se ejecuta ISaGRAF sobre una base informática potente y flexible, existen dos restricciones en el protocolo Modbus:
- Sólo se puede acceder a las variables de ISaGRAF
- Es difícil ejecutar la transferencia rápida de grandes cantidades de datos

Por estas razones, ISaGRAF ofrece un conjunto de peticiones de transferencia de datos “tipo Modbus”, o un protocolo de “gestión remota de ficheros”. Se han implementado estas características para permitir:
- Carga de ficheros binarios o ASCII
- Recuperación de ficheros binarios o ASCII
- Intercambio dinámico de datos a través de ficheros compartidos virtuales o físicos

En consecuencia, con el enlace de comunicación de ISaGRAF, cualquier aplicación “independiente de ISaGRAF” se puede comunicar fácilmente con un objeto remoto.

El protocolo está basado en los siguientes conceptos:
- El fichero del lado del objeto ISaGRAF se llama **fichero remoto**
- El fichero que reside en el ordenador maestro se llama **fichero local**
- Se accede a cada byte de un fichero con una **dirección de base** de 32 bits más una **dirección de byte** de 16 bits

Existen peticiones para seleccionar el nombre del fichero remoto, para seleccionar la dirección de base, para leer o escribir datos en el fichero remoto utilizando la dirección de byte de 16 bits.

FUNCTION 17: escribir datos

nbb corresponde al número de bytes vH, vL

Pregunta	slv	11	addH	addL	00	nbb	nbb	vH	vL	...	crcH	crcL

Respuesta	slv	11	addH	addL	00	nbb	crcH	crcL

El significado de esta petición difiere de acuerdo con el rango de dirección addH/addL:

- **0xF000: Inicializar nombre de fichero remoto**
  nbb corresponde al número de caracteres del nombre de fichero, especificado en los campos vH vL (en este caso, Alto y Bajo no son significantes), **incluyendo \0** para final de cadena.
  Si el fichero no existe, se crea con los atributos 'escribible' + leíble + ejecutable.

- **0xF002: Cambiar dirección de base al valor especificado**
  nbb debería ser igual a 4. El primer byte vH/vL corresponde a la palabra Alta del valor especificado. Se acepta cualquier valor de 32 bits.
  Todas las peticiones futuras de 'leer' o 'escribir' utilizarán esta dirección de base. Cuando no se utiliza esta petición, el valor de la dirección de base por defecto es cero.

- **0xF004: Borrar fichero**
  nbb debería ser igual a cero.
  Se borrará el fichero si existe y si es posible hacerlo.

- **Mayor que 0xF004: Reservado**

- **Menor que 0xF000: Escribir bytes**
  La dirección en la que se pueden escribir bytes está especificada en addH/addL. Debe ser inferior a F000. Se escriben los bytes especificados (bytes nbb especificados en los campos vH vL, donde Alto y Bajo quizás hayan dejado de ser significativos) en el orden que se indica (de izquierda a derecha) al nombre de fichero remoto que previamente se ha seleccionado. La dirección de inicio en la que se ha escrito es la dirección especificada que se ha añadido a la dirección de base que previamente se ha seleccionado. Si el acceso a direcciones resultante supera el tamaño del fichero actual, se extiende el fichero. No se puede reducir el tamaño del fichero.

FUNCIÓN 18: leer datos

Pregunta	slv	12	addH	addL	00	nbb	crcH	crcL

Respuesta	slv	12	nbb	V	V	...	crcH	crcL

La dirección en la que se leen bytes se especifica en addH/addL. Debe ser inferior a F000. Leer el número (nbb) especificado de bytes desde el nombre de fichero remoto previamente seleccionado, partiendo de la dirección especificada (addH/addL con cualquier valor de 16 bits) añadida a la dirección de base previamente seleccionada.
Los valores de recuperan (campos V de izquierda a derecha) en el orden en el que se leen del fichero.

Ejemplo:

Seleccionar nombre de fichero remoto: 'target.fil'.

Pregunta	01	11	F0	00	00	0B	0B	74	...	00	25	9F

Respuesta	01	11	F0	00	00	0B	8F	0E

Seleccionar dirección de base: 0x10000.

Pregunta	01	11	F0	02	00	04	04	00	01	00	00	76	11

Respuesta	01	11	F0	02	00	04	6E	CA

Escribir 4 bytes: dirección absoluta 0x107D0, valores 01,02,03,04.

Pregunta	01	11	07	D0	00	04	04	01	02	03	04	28	6F

Respuesta	01	11	07	D0	00	04	FC	87

Leer 4 bytes: dirección absoluta 0x107D0.

Pregunta	01	12	07	D0	00	04	B8	87

Respuesta	01	12	04	01	02	03	04	58	7D

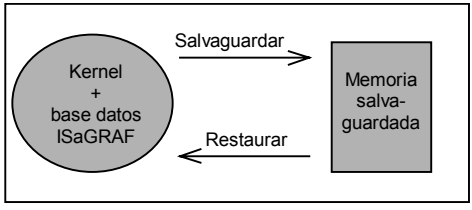
C.9 Gestión de fallos de tensión

C.9.1 Conceptos básicos

La gestión de los fallos de tensión es un aspecto muy crítico de una aplicación, por tres razones:
- Depende de las especificaciones del proceso
- Depende de la capacidad del *hardware*
- Depende de los métodos de programación

Por lo tanto, la respuesta de ISaGRAF a la gestión de fallos de tensión no es un método universal completo y absoluto, sino un conjunto de principios, métodos y herramientas que deben combinarse de una manera específica para cada aplicación o al menos para cada *hardware*.

Para que un sistema de control de procesos pueda volver a arrancar correctamente después de un fallo de tensión, se tienen que resolver tres problemas:



- Realizar una copia de seguridad de los datos
- Detectar que se ha producido un fallo de tensión al arrancar
- Restaurar los datos salvaguardados

Aunque el segundo problema no puede tener una solución *software* estándar, el proveedor del sistema puede proporcionar las herramientas necesarias para poder acceder al estado del *hardware* desde la aplicación ISaGRAF o desde un programa en “C”.

Además, lo importante es decidir qué tipo de datos tiene que ser almacenado y recuperado. Pueden definirse dos tipos de datos:
- Variables de la aplicación:
  Variables de proceso tales como número de ítems procesados, fecha del fallo de tensión, valores de los parámetros de la aplicación, etc.
  Variables de programa tales como contadores, temporizadores, valores intermedios y señalizadores.

- Estado del programa:
  P.ej. referencia de pasos activos, estado de cada programa “C”, etc.

Se estudiarán estos dos casos en los siguientes apartados para ver qué respuestas puede aportar ISaGRAF.

C.9.2 Salvaguarda de variables de la aplicación

Variables retenidas

El editor de variables del banco de trabajo ofrece la posibilidad de seleccionar el atributo “retener” para cada variable interna (variables no de E/S).

Al final de cada ciclo objeto, se copian los valores de las variables retenidas a una posición de memoria determinada. Esta posición de memoria suele ser una RAM salvaguardada por batería.

En el momento del arranque, si al menos una variable posee el atributo "retenido", ISaGRAF busca las variables retenidas:
- Si la misma aplicación ya ha sido ejecutada con anterioridad, ISaGRAF reconoce los valores almacenados y los asigna a cada una de las variables retenidas.
- Si la aplicación anterior es diferente, o si no se ha ejecutado ninguna aplicación anteriormente, ISaGRAF reconoce que los valores retenidos no son válidos y reinicializa todas las variables retenidas en valor nulo.

Las especificaciones del área de memoria que se utiliza para almacenar los diferentes tipos de variables están establecidas en el banco de trabajo, en el menú **Ensamblar**: **Opción de aplicación en ejecución ; variables retenidas.**
La cadena especificada tiene que tener el siguiente formato:

boo_add , boo_size , ana_add , ana_size , tmr_add , tmr_size , msg_add , msg_size

con:

boo_add: Dirección hexadecimal utilizada para almacenar variables booleanas. Siempre tiene que ser diferente a cero.

boo_size: Tamaño hexadecimal, expresado en bytes, disponible en esta dirección. Se requiere un byte por cada variable booleana que se desee almacenar.

ana_add: Dirección hexadecimal utilizada para almacenar variables analógicas. Siempre tiene que ser diferente a cero.

ana_size: Tamaño hexadecimal, expresado en bytes, disponible en esta dirección. Siempre se requiere un mínimo de 4 bytes más 4 bytes por cada variable analógica que se desee almacenar.

tmr_add: Dirección hexadecimal utilizada para almacenar variables temporizadores. Siempre tiene que ser diferente a cero.

tmr_size: Tamaño hexadecimal, expresado en bytes, disponible en esta dirección. Se requieren 5 bytes por cada variable temporizador que se desee almacenar.

msg_add: Dirección hexadecimal utilizada para almacenar variables mensajes. Siempre tiene que ser diferente a cero.

msg_size: Tamaño hexadecimal, expresado en bytes, disponible en esta dirección. Se requieren 256 bytes por cada variable mensaje que se desee almacenar.

Requisitos:

- Tienen que especificarse todos los campos de todos los tipos, incluso si no existiera la necesidad de salvaguardar todos los tipos de variables. En este caso, se tiene que especificar un tamaño de cero (con la excepción de las analógicas, para las cuales se tiene que especificar un tamaño de 4 bytes) y cualquier dirección que sea diferente a cero, para los tipos de variables que no sean necesarias.

Ejemplo:

Supongamos que tenemos que realizar una copia de seguridad de:

20 variables booleanas
0 variables analógicas
0 variables temporizadores
3 variables mensajes

La memoria salvaguardada está ubicada en la dirección hexadecimal 0xA2F200.

Supongamos también que:

Se almacenarán las booleanas en la dirección 0xA2F200, con el tamaño exacto requerido de 20 bytes.

Las analógicas - con un tamaño mínimo requerido de 4 bytes - se almacenarán en la dirección 0xA2F214.

La dirección ficticia de los temporizadores será 0xA2F200 y se le especificará un tamaño de cero.

Se almacenarán los mensajes en la dirección 0x A2F218, con el tamaño exacto requerido de 256*3 bytes.

Consecuentemente, la cadena introducida en el banco de trabajo tendría que ser:

```
A2F200,14,A2F214,4,A2F200,0,A2F218,300
```

Invocación de la función SYSTEM

Si existe la necesidad de almacenar la mayoría de las variables de la aplicación, debe utilizarse las prestaciones de la función SYSTEM para gestionar un conjunto completo de variables (para mayor información sobre la función SYSTEM, véase el Manual del Usuario). Cabe destacar que tanto la salvaguarda como la restauración están gestionadas por el programador a nivel de aplicación.

En primer lugar, se tiene que definir la posición de salvaguarda de memoria para un tipo determinado de variable o para todos los tipos de variables:

<nueva_dirección> := SYSTEM(SYS_INITxxx,<dirección>);

donde:

- <dirección> es la dirección de salvaguarda de memoria (valor 16# en formato hexadecimal). Tiene que ser una dirección par, o en caso contrario, fallaría la operación.
- SYS_INITxxx puede ser:
  - SYS_INITBOO para definir la posición de salvaguarda de memoria de todas las variables booleanas.
  - SYS_INITANA para definir la posición de salvaguarda de memoria de todas las variables analógicas.
  - SYS_INITTMR para definir la posición de salvaguarda de memoria de todas las variables temporizadores.
  - SYS_INITALL para definir la posición de salvaguarda de memoria de todas las variables booleanas, analógicas y temporizadores.
- <nueva_dirección> obtiene la siguiente dirección libre, es decir, <dirección> + tamaño de las variables salvaguardadas (en bytes), según SYS_INITxxx. Esto permite comprobar el tamaño requerido de la copia de seguridad de memoria. Si falla la operación, <nueva_dirección> recibe el valor cero.

Entonces se puede solicitar la copia de seguridad. Se puede invocar este procedimiento en cualquier punto de la aplicación, pero se efectuará la copia de seguridad al final del ciclo y sólo una vez. Si el *hardware* proporciona una entrada booleana, o una función "C" para informar al usuario cuando se produce el fallo de tensión, y permite al menos una demora de ciclo ISaGRAF antes de la 'caída', se podría hacer la copia de seguridad sólo al detectarse el fallo de tensión:

<error> :=SYSTEM(SYS_SAVxxx,0);

donde:
- SYS_SAVxxx puede ser:
  - SYS_SAVBOO para pedir la salvaguarda de todas las variables booleanas.
  - SYS_SAVANA para pedir la salvaguarda de todas las variables analógicas.
  - SYS_SAVTMR para pedir la salvaguarda de todas las variables temporizadores.
  - SYS_SAVALL para pedir la salvaguarda de todas las variables booleanas, analógicas y temporizadores.
- <error> obtiene un estado de error diferente a cero cuando falla la operación (no se ha invocado SYS_INITxxx).

Por último, existe la posibilidad de restaurar las variables. Se puede invocar este procedimiento en cualquier punto de la aplicación, pero se efectuará la restauración al final del ciclo y sólo una vez. Para asegurar la validez de los datos salvaguardados, se debe configurar una variable analógica como valor constante para su utilización como firma:

<error> := SYSTEM(SYS_RESTxxx,0);

donde:
- SYS_RESTxxx puede ser:
  - SYS_RESTBOO para restaurar todas las variables booleanas.
  - SYS_RESTANA para restaurar todas las variables analógicas.
  - SYS_RESTTMR para restaurar todas las variables temporizadores.
  - SYS_RESTALL para restaurar todas las variables booleanas, analógicas y temporizadores.
- <error> obtiene un estado de error diferente a cero cuando falla la operación (no se ha invocado SYS_INITxxx).

A continuación se presenta un resumen de los comandos de la función SYSTEM que se utilizan para gestionar las variables salvaguardadas:

Comando		Significado
clave predefinida	**Valor**	
SYS_INITBOO	16#20	inicia salvaguarda booleanas
SYS_SAVBOO	16#21	guardar booleanas
SYS_RESTBOO	16#22	restaurar booleanas
SYS_INITANA	16#24	inicia salvaguarda analógicas
SYS_SAVANA	16#25	guardar analógicas
SYS_RESTANA	16#26	restaurar analógicas
SYS_INITTMR	16#28	inicia salvaguarda temporizador
SYS_SAVTMR	16#29	guardar temporizadores
SYS_RESTTMR	16#2A	restaurar temporizadores
SYS_INITALL	16#2C	inicia salvaguarda todos tipos
SYS_SAVALL	16#2D	guardar todos tipos
SYS_RESTALL	16#2E	restaurar todos tipos

Comando (clave predefinida)	Argumento	Valor de retorno
SYS_INITxxx	dirección memoria	siguiente dirección libre
SYS_SAVxxx	0	cero si es OK
SYS_RESTxxx	0	cero si es OK

Implementación personalizada

Por último, mediante la utilización de funciones o bloques de función “C”, se pueden desarrollar procedimientos específicos del usuario de acceso a la memoria respaldada por batería, para almacenar y restaurar variables en cualquier momento de la aplicación.

Ejemplos:

1) Procedimiento dedicado a una aplicación:
Los procedimientos del usuario “C” serían backup, restore_temp, restore_date, restore_cpt.

```
backup(temperature, date, cnt);   almacenar 3 datos críticos

temperature := restore_temp();    restaurar temperatura
date := restore_date();           restaurar fecha
cnt := restore_cnt();             restaurar contador
```

2) Procedimientos de propósito general:
Los procedimientos del usuario “C” serían backup_init, backup, backup_link, restore.

```
save_id := backup_init(address, size);   reservar un vector respaldado por batería.
backup(save_id, cpt1, 3);                guardar cpt1 como el 3er elemento.

rest_id := backup_link(address, size)    enlazar memoria respaldada.
cpt1 := restore(rest_id, 3);             restaurar valor respaldado de cpt1.
```

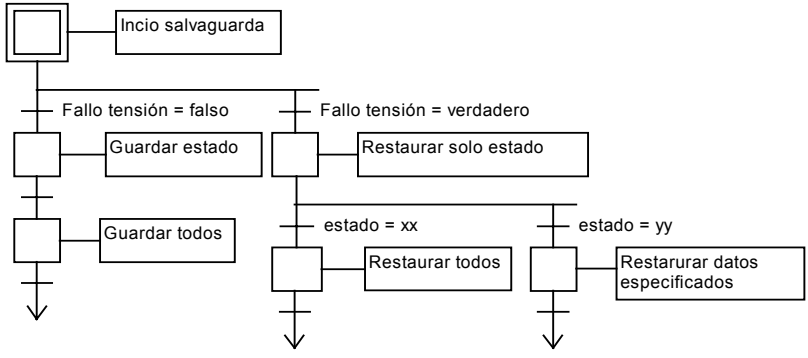
C.9.3 Copia de seguridad del estado del programa

Se podría almacenar cada uno de los estados del programa de aplicación, pero sería peligroso restaurar cada programa al estado en el cual se encontraba en el momento de la última copia de seguridad, por al menos tres motivos:

- Algunos procesos requieren la ejecución de operaciones específicas antes de arrancar de nuevo.
- Ocuparse de cada estado de una aplicación completa resultaría tedioso.
- Algunos recursos externos, tales como programas “C”, periféricos, etc. no pueden reinicializarse automáticamente.

La mejor solución parece ser la de salvaguardar las variables analógicas o booleanas que describen el estado del proceso cuando el programador piensa que las fases de reinicialización podrán utilizar esta información.
Después, debería ser posible que una ‘imagen’ incompleta pero inteligente del proceso pudiera iniciar, finalizar o congelar programas SFC e inicializar variables para situar a la aplicación en el estado apropiado. ISaGRAF no puede facilitar un procedimiento automático de arranque.

Ejemplo:



C.10 Apéndice: Lista y descripción de errores

Lista de errores:

Cód.	Mensaje	Tipo
1	no se puede asignar memoria para base de datos de tiempo de ejecución	Sistema
2	base de datos de aplicación incorrecta (Motorola/Intel)	Aplicación
3	no se puede asignar buzón de comunicaciones	Sistema
4	no se puede enlazar la base de datos del kernel	Sistema
5	tiempo excedido enviando pregunta al kernel	Sistema
6	tiempo excedido esperando respuesta del kernel	Sistema
7	no se puede inicializar la comunicación	Sistema
8	no se puede asignar memoria para las variables retentivas	Aplicación
9	aplicación parada	Aplicación
10	demasiadas acciones P o N simultáneas	Aplicación
11	demasiadas acciones de activación simultáneas	Aplicación
12	demasiadas acciones de desactivación simultáneas	Aplicación
13	instrucción TIC desconocida	Aplicación
16	no se puede responder la petición de lectura de datos	Sistema
17	no se puede responder la petición de escritura de datos	Sistema
18	no se puede responder la petición del depurador de sesión	Sistema
19	no se puede responder la petición modbus	Sistema
20	no se puede responder la petición del depurador de aplicación	Sistema
21	no se puede responder a la depuración	Sistema
23	código de petición desconocido	Sistema
24	error de comunicación Ethernet	Sistema
25	error de sincronización de comunicación	Sistema
28	no se puede asignar memoria para la aplicación	Sistema
29	no se puede asignar memoria para la actualización de la aplicación	Sistema
30	código identificativo OEM desconocido	Aplicación
31	no se puede inicializar tarjeta de entrada booleana	Aplicación
32	no se puede inicializar tarjeta de entrada analógica	Aplicación
33	no se puede inicializar tarjeta de entrada mensaje	Aplicación
34	no se puede inicializar tarjeta de salida booleana	Aplicación
35	no se puede inicializar tarjeta de salida analógica	Aplicación
36	no se puede inicializar tarjeta de salida mensaje	Aplicación
37	no se puede leer datos de tarjeta booleana	Aplicación
38	no se puede leer datos de tarjeta analógica	Aplicación
39	no se puede leer datos de tarjeta mensaje	Aplicación

40	no se puede sacar variable de salida booleana	Aplicación
41	no se puede sacar variable de salida analógica	Aplicación
42	no se puede sacar variable de salida mensaje	Aplicación
43	no se puede operar variable booleana	Aplicación
44	no se puede operar variable analógica	Aplicación
45	no se puede operar variable mensaje	Aplicación
46	no se puede abrir tarjeta	Aplicación
47	no se puede cerrar tarjeta	Aplicación
50	no se puede sobreescribir variable de salida booleana	Programa
51	no se puede sobreescribir variable de salida analógica	Programa
52	no se puede sobreescribir variable de salida mensaje	Programa
61	código desconocido de petición al sistema	Programa
62	desbordamiento de periodo de muestreo	Programa
63	función de usuario no implementada	Aplicación
64	entero dividido por cero	Programa
65	función de conversión no implementada	Aplicación
66	bloque de función no implementado	Aplicación
67	función estándar no implementada	Aplicación
68	real dividido por cero	Programa
69	parámetros de OPERATE no válidos	Aplicación
72	los símbolos de la aplicación no pueden ser modificados	Aplicación
73	no se puede actualizar: conjunto de variables booleanas diferente	Aplicación
74	no se puede actualizar: conjunto de variables analógicas diferente	Aplicación
75	no se puede actualizar: conjunto de variables temporizadores diferente	Aplicación
76	no se puede actualizar: conjunto de variables mensajes diferente	Aplicación
77	no se puede actualizar: no se puede encontrar la nueva aplicación	Aplicación
> 100	códigos de error específicos de OEM, pregunte a su suministrador por más detalles	

Los tres tipos de error corresponden a las tres fuentes de problemas:

– Errores de sistema:
Es probable que estos problemas sean imputables al *software* o *hardware* objeto, y no a la configuración de la aplicación o la ejecución del programa.
Probar el 'reinicio duro' (apagado) del objeto, e intentar ejecutar otras aplicaciones.
Se debe informar de estos errores al soporte ISaGRAF.

– **Errores de aplicación:**
Los problemas de este tipo son debidos a los parámetros, el tamaño o el contenido de la aplicación.

Estos errores deberían desaparecer al cargar una aplicación conocida y previamente validada. Si el problema persiste, pasa a considerarse como un error de sistema (véase posición anterior).

– **Errores de programa:**
Estos problemas son imputables a una secuencia determinada del programa.
Este tipo de error debería desaparecer al arrancar el programa en modo 'ciclo por ciclo', o cuando se pare el programa crítico.

Descripción de errores:

1. no se puede asignar memoria para base de datos de tiempo de ejecución	***Sistema***

No se dispone de memoria libre. Comprobar el *hardware*.

2. base de datos de aplicación incorrecta (Motorola/Intel)	***Aplicación***

El fichero de aplicación que se ha cargado o salvaguardado no es correcto. Este error aparece si se genera la aplicación para INTEL y se carga en una plataforma MOTOROLA (y viceversa), o si el fichero ha sido alterado.

3. no se puede asignar buzón de comunicaciones	***Sistema***

La tarea de comunicación genera este error si no puede asignar el espacio 3 para la comunicación entre tareas.

4. no se puede enlazar la base de datos del kernel	***Sistema***

La tarea de comunicación genera este error si no puede encontrar un *kernel* en funcionamiento con el número de esclavo que se especifica en su línea de comando.

5. tiempo excedido enviando pregunta al kernel	***Sistema***

La tarea de comunicación no puede enviar una petición al *kernel*. Lo más probable es que el *kernel* esté fuera de servicio o esté ocupado.

6. tiempo excedido esperando respuesta del kernel	***Sistema***

La tarea de comunicación no puede recibir una respuesta del *kernel*. Lo más probable es que el *kernel* esté fuera de servicio o esté ocupado.

7. no se puede inicializar la comunicación	***Sistema***

Se genera esta advertencia cuando la capa de comunicación no puede inicializar el enlace físico. También se presenta esta advertencia si no se especifica la ruta de

comunicación. Esto no le impide al objeto que funcione correctamente, pero no puede comunicar.

8. no se puede asignar memoria para las variables retentivas	***Aplicación***

ISaGRAF no puede manejar variables retentivas. Pueden existir dos razones para este tipo de problema:
- la cadena que se traslada como parámetro al objeto anfitrión no es sintácticamente correcta
- el tamaño de la memoria que se ha especificado para cada bloque es insuficiente
Verificar la sintaxis del parámetro 'retener variable' y probar con un número reducido de variables retentivas.

9. aplicación parada	***Aplicación***

Se genera esta advertencia cada vez que se para la aplicación desde el depurador.

10. demasiadas acciones P o N simultáneas	***Aplicación***

Se produce este error si uno de los ciclos del objeto tiene que ejecutar demasiadas acciones de impulso no almacenadas o bloques cíclicos. Existe la posibilidad de localizar el problema en modo CC. El número máximo de acciones simultáneas es de 2 + 4 por programa SFC.

11. demasiadas acciones de activación simultáneas	***Aplicación***

Se produce este error si uno de los ciclos del objeto tiene que ejecutar demasiadas acciones de configuración (que se ejecutan cuando se activa un paso). Proceder como se mencionó en el anterior.

12. demasiadas acciones de desactivación simultáneas	***Aplicación***

Se produce este error si uno de los ciclos objeto tiene que ejecutar demasiadas acciones de reactivación (que se ejecutan cuando un paso está desactivado). Proceder como se mencionó en el anterior.

13. instrucción TIC desconocida	***Aplicación***

El *kernel* ha detectado algo que no funciona bien en el código de aplicación (llamado Target Independent Code o Código Independiente del Objeto), en un programa. Existen dos posibles explicaciones:
- es probable que un programa externo esté escribiendo en el código de aplicación. Intentar localizar el *crash* en modo CC y asegurarse de que no hay ningún interfaz de E/S con parámetros erróneos.
- el objeto tiene un conjunto reducido de instrucciones, y la aplicación utiliza una instrucción o tipo de variable no autorizado.

16. no se puede responder la petición de lectura de datos	***Sistema***

Se detecta un error de comunicación al responder a la petición específica del código de función 18 (Lectura fichero) del Modbus ISaGRAF.
Comprobar la conexión y la configuración del sistema a nivel tanto de objeto como de estación maestra.

17. no se puede responder la petición de escritura de datos	***Sistema***

Se detecta un error de comunicación al responder a la petición específica del código de función 17 (escritura fichero) del Modbus ISaGRAF.
Comprobar la conexión y la configuración del sistema a nivel tanto de objeto como de estación maestra.

18. no se puede responder la petición del depurador de sesión	***Sistema***

Se detecta un error de comunicación al responder a una petición de depuración. Comprobar la conexión y la configuración del sistema a nivel tanto de objeto como de estación maestra.

19. no se puede responder la petición modbus	***Sistema***

Se detecta un error de comunicación al responder a una petición del Modbus. Comprobar la conexión y la configuración del sistema a nivel tanto de objeto como de estación maestra.

20. no se puede responder la petición del depurador de aplicación	***Sistema***

Se detecta un error de comunicación al responder a una petición de depurador. Comprobar la conexión y la configuración del sistema a nivel tanto de objeto como de estación maestra.

21. no se puede responder a la depuración	***Sistema***

Se detecta un error de comunicación al responder a una petición de depurador. Comprobar la conexión y la configuración del sistema a nivel tanto de objeto como de estación maestra.

23. código de petición desconocido	***Sistema***

Una petición del depurador carece de sentido.

24. error de comunicación Ethernet	***Sistema***

Aparece este mensaje cada vez que se cierra la conexión, estando cerrado el depurador: el sistema está funcionando correctamente. En caso contrario, significa que se ha detectado un error de comunicación Ethernet. Comprobar la conexión y la configuración del sistema a nivel tanto de objeto como de estación maestra.
Se indica un segundo campo, que puede ser:

1: error al enviar o recibir
2: error al crear el *socket*
3: error al enlazar o escuchar al *socket*
4: error al aceptar un cliente nuevo

25. error de sincronización de comunicación	***Sistema***

Falta de sincronización entre la tarea de comunicación en el objeto y la estación maestra. Comprobar la conexión y la configuración del sistema (parámetros de comunicación) a nivel tanto de objeto como de estación maestra.

28. no se puede asignar memoria para la aplicación	***Sistema***

No se dispone de memoria libre. Comprobar el *hardware*, de acuerdo con el tamaño de la aplicación.

29. no se puede asignar memoria para la actualización de la aplicación	***Sistema***

No se dispone de memoria libre. Comprobar el *hardware*, de acuerdo con el tamaño de la aplicación.

30. código identificativo OEM desconocido	***Aplicación***

La aplicación está utilizando una tarjeta cuya clave de fabricante no es reconocida por el objeto. Comprobar la conexión de E/S en el banco de trabajo y utilizar el atributo 'VIRTUAL' para localizar la tarjeta incorrecta. Es posible que la librería del banco de trabajo no corresponda con la versión del objeto del usuario.

31. no se puede inicializar tarjeta de entrada booleana	***Aplicación***

Ha fallado la inicialización de una tarjeta de entradas booleanas. Comprobar la conexión de E/S en el banco de trabajo y los parámetros de las tarjetas de entradas booleanas.

32. no se puede inicializar tarjeta de entrada analógica	***Aplicación***

Ha fallado la inicialización de una tarjeta de entradas analógicas. Comprobar la conexión de E/S en el banco de trabajo y los parámetros de las tarjetas de entradas analógicas.

33. no se puede inicializar tarjeta de entrada mensaje	***Aplicación***

Ha fallado la inicialización de una tarjeta de entradas de mensajes. Comprobar la conexión de E/S en el banco de trabajo y los parámetros de las tarjetas de entradas de mensajes.

34. no se puede inicializar tarjeta de salida booleana	***Aplicación***

Ha fallado la inicialización de una tarjeta de salidas booleanas. Comprobar la conexión de E/S en el banco de trabajo y los parámetros de las tarjetas de salidas booleanas.

35. no se puede inicializar tarjeta de salida analógica	***Aplicación***

Ha fallado la inicialización de una tarjeta de salidas analógicas. Comprobar la conexión de E/S en el banco de trabajo y los parámetros de las tarjetas de salidas analógicas.

36. no se puede inicializar tarjeta de salida mensaje	***Aplicación***

Ha fallado la inicialización de una tarjeta de salidas de mensajes. Comprobar la conexión de E/S en el banco de trabajo y los parámetros de las tarjetas de salidas de mensajes.

37. no se puede leer datos de tarjeta booleana	***Aplicación***

Se ha detectado un error al refrescar una tarjeta de entradas booleanas. Comprobar la conexión de E/S en el banco de trabajo y los parámetros de la tarjeta.

38. no se puede leer datos de tarjeta analógica	***Aplicación***

Se ha detectado un error al refrescar una tarjeta de entradas analógicas. Comprobar la conexión de E/S en el banco de trabajo y los parámetros de la tarjeta.

39. no se puede leer datos de tarjeta mensaje	***Aplicación***

Se ha detectado un error al refrescar una tarjeta de entradas de mensajes. Comprobar la conexión de E/S en el banco de trabajo y los parámetros de la tarjeta.

40. no se puede sacar variable de salida booleana	***Aplicación***

Se ha detectado un error al actualizar una variable de salida booleana. Comprobar la conexión de E/S en el banco de trabajo y los parámetros de la tarjeta.

41. no se puede sacar variable de salida analógica	*Aplicación*

Se ha detectado un error al actualizar una variable de salida analógica. Comprobar la conexión de E/S en el banco de trabajo y los parámetros de la tarjeta.

42. no se puede sacar variable de salida mensaje	*Aplicación*

Se ha detectado un error al actualizar una variable de salida de mensajes. Comprobar la conexión de E/S en el banco de trabajo y los parámetros de la tarjeta.

43. no se puede operar variable booleana	*Aplicación*

Se ha detectado un error al ejecutar una invocación OPERATE de una variable booleana. Verificar los parámetros de OPERATE y las notas del usuario de la tarjeta.

44. no se puede operar variable analógica	*Aplicación*

Se ha detectado un error al ejecutar una invocación OPERATE de una variable analógica. Verificar los parámetros de OPERATE y las notas del usuario de la tarjeta.

45. no se puede operar variable mensaje	*Aplicación*

Se ha detectado un error al ejecutar una invocación OPERATE de una variable de mensajes. Verificar los parámetros de OPERATE y las instrucciones del usuario de la tarjeta.

46. no se puede abrir tarjeta	*Aplicación*

La aplicación está utilizando una referencia de tarjeta que es desconocida para el objeto. Comprobar la conexión de E/S en el banco de trabajo. Es posible que la librería del banco de trabajo no corresponda a la versión del objeto.

47. no se puede cerrar tarjeta	*Aplicación*

La aplicación está utilizando una referencia de tarjeta que es desconocida para el objeto. Comprobar la conexión de E/S en el banco de trabajo.

50. no se puede sobreescribir variable de salida booleana	*Programa*

Dos secuencias SFC están escribiendo la misma variable de salida booleana en el mismo ciclo objeto. Esto se debe evitar para prevenir el comportamiento peligroso de las E/S. En caso de producirse un conflicto de este tipo, se concede la prioridad al programa que más alto está en la jerarquía. Si los dos programas SFC están situados en el mismo nivel, el resultado será impredecible.

51. no se puede sobreescribir variable de salida analógica	***Programa***

Dos secuencias SFC están escribiendo la misma variable de salida analógica en el mismo ciclo objeto. Es aplicable el anterior comentario.

52. no se puede sobreescribir variable de salida mensaje	***Programa***

Dos secuencias SFC están escribiendo la misma variable de salida de mensajes en el mismo ciclo objeto. Es aplicable el anterior comentario.

61. código desconocido de petición al sistema	***Programa***

Un programa está utilizando la invocación SYSTEM con un código que no es válido.

62. desbordamiento de periodo de muestreo	***Programa***

El periodo de ciclo objeto es más largo de lo que se especificó en el menú del banco de trabajo.
En un sistema multitarea, esto significa que no se dispone de suficiente tiempo de CPU para ejecutar un ciclo, incluso si la 'duración del ciclo actual' es inferior al periodo especificado.
En un sistema monotarea, esto siempre significa que hay demasiadas operaciones en uno de los ciclos objeto.

Existen numerosas maneras de eliminar esta advertencia:
- reducir el número de operaciones que se realizan en el punto en el cual se detecta la advertencia.
- reducir el número de *marcas* y de transiciones válidas, optimizar el procesamiento complejo, etc.
- reducir otras tareas que consuman tiempo de CPU para dejarle más tiempo a ISaGRAF.
- reducir el tráfico de comunicación para dejarle más tiempo a ISaGRAF.
- utilizar la modificación dinámica de duración de ciclos para adaptar la duración de ciclos a las diferentes etapas del proceso.
- fijar la duración de ciclo en cero para que el *kernel* de ISaGRAF pueda ejecutarse a la mayor velocidad posible, sin control de desbordamiento.

63. función de usuario no implementada	***Aplicación***

Un programa está utilizando una función “C” que es desconocida por el objeto. Es posible que la librería del banco de trabajo no corresponda a la versión del objeto.

64. entero dividido por cero	***Programa***

Un programa intenta dividir un valor analógico entero por cero. La aplicación debe evitar que se produzca este tipo de operación, ya que puede tener un efecto impredecible.
Cuando esto sucede, ISaGRAF presenta el máximo valor analógico como resultado.
Cuando el operando es negativo, se invierte el resultado.

65. función de conversión no implementada	***Aplicación***

Un programa está utilizando una función de conversión “C” que es desconocida por el objeto. Es posible que la librería del banco de trabajo no corresponda a la versión del objeto.
Cuando esto sucede, ISaGRAF no convierte el valor.

66. bloque de función no implementado	***Aplicación***

Un programa está utilizando un bloque de función “C” que es desconocido por el objeto. Es posible que la librería del banco de trabajo no corresponda a la versión del objeto.

67. función estándar no implementada	***Aplicación***

Un programa está utilizando un bloque de función que es desconocido por el objeto, aunque se supone que tiene que estar disponible en la mayoría de objetos. Consultar al proveedor.

68. real dividido por cero	***Programa***

Un programa intenta dividir un valor analógico real por cero. La aplicación debe evitar que se produzca este tipo de operación, ya que puede tener un efecto impredecible.
Cuando esto sucede, ISaGRAF presenta el máximo valor analógico real como resultado.
Cuando el operando es negativo, se invierte el resultado.

69. parámetros de OPERATE no válidos	***Aplicación***

La aplicación está utilizando una invocación OPERATE con parámetros erróneos. Esto suele estar filtrado por el compilador. Podría ser un parámetro temporizador o una variable que no es de entrada ni de salida.

72. los símbolos de la aplicación no pueden ser modificados	***Aplicación***

Al intentar efectuar la actualización de una aplicación, no se puede arrancar la aplicación modificada debido a que los símbolos son diferentes. Se podrían haber

añadido, eliminado o modificado una o más variables o instancias de bloques de función, en comparación con la aplicación actual.

73. no se puede actualizar: conjunto de variables booleanas diferente	***Aplicación***

No se puede arrancar la aplicación modificada al haberse añadido o eliminado determinadas variables booleanas, en comparación con la aplicación actual.

74. no se puede actualizar: conjunto de variables analógicas diferente	***Aplicación***

No se puede arrancar la aplicación modificada al haberse añadido o eliminado determinadas variables analógicas, en comparación con la aplicación actual.

75. no se puede actualizar: conjunto de variables temporizadores diferente	***Aplicación***

No se puede arrancar la aplicación modificada al haberse añadido o eliminado determinadas variables termporizadores, en comparación con la aplicación actual.

76. no se puede actualizar: conjunto de variables mensajes diferente	***Aplicación***

No se puede arrancar la aplicación modificada al haberse añadido o eliminado determinadas variables de mensajes, en comparación con la aplicación actual.

77. no se puede actualizar: no se puede encontrar la nueva aplicación	***Aplicación***

No se encuentra la aplicación modificada en memoria; habrá surgido algún problema durante la carga.

D. Glosario

Acción Lista de sentencias o asignaciones que se ejecuta cuando está activo un paso de un programa SFC.

Acción booleana Acción SFC: se asigna una variable booleana a la señal de actividad de un paso.

Acción pulsante Acción SFC: es una lista de sentencias que se ejecuta una sola vez al activarse el paso correspondiente.

Acción no almacenada Acción SFC: es una lista de sentencias que se ejecuta en cada ciclo objeto, cuando está activo el paso correspondiente.

Actividad de un paso Atributo de un paso que está señalado por una marca SFC. Las acciones que están asociadas al paso se ejecutan en función de su actividad.

Analógico Tipo de variable. Son variables contínuas de valores enteros o reales.

Atributo Clase de variable. Los atributos de variable existentes son interna, de entrada o de salida.

Bloque de funciones Componente gráfico del lenguaje FBD, que representa una función elemental estándar de las librerías ISaGRAF.

Booleano Tipo de variable. Las variables de este tipo sólo asumen valores de verdadero o falso.

Bobina Componente gráfico de un programa LD, que se utiliza para representar la asignación de una variable de salida.

Cabecera fuente C Fichero de texto que contiene las definiciones y los tipos C que son necesarios para la programación de una función C o una función de conversión.

Cadena Conjunto de caracteres almacenado en una variable de mensaje.

Caja de herramientas Pequeña ventana hija de la ventana de herramientas de edición gráfica, que reúne los principales botones para la selección de componentes gráficos.

Canal de E/S	Punto de conexión individual de una tarjeta de E/S. Un canal de E/S puede recibir una variable de E/S.
Carril de potencia	Principales carriles verticales izquierdo y derecho, situados en las extremidades de un diagrama de contactos.
Celda	Área elemental de la matriz gráfica para lenguajes gráficos como SFC, FBD o LD.
Cíclico	Atributo de un programa que siempre se ejecuta.
Ciclo objeto	Conjunto de operaciones ejecutadas cada vez que se activa el sistema objeto de ISaGRAF. Se activan los ciclos por medio de la programación de tiempos.
Clave OEM (tarjeta de E/S)	Código hexadecimal de 16 bits que se define para cada tarjeta de E/S de la librería ISaGRAF. La clave OEM identifica al proveedor de la tarjeta.
Código fuente C	Fichero de texto que contiene el código fuente C de una función o función de conversión.
Comentario	Texto que se incluye en un programa, sin impacto en la ejecución del programa.
Comentario (SFC)	Texto asociado a un paso o una transición SFC, sin impacto en la ejecución del programa.
Comunes	Rango de palabras definidas. Se puede utilizar este tipo de objeto en cualquier programa de cualquier proyecto.
Condición (para una transición)	Expresión booleana asociada a una transición SFC. No se puede franquear la transición cuando su condición es 'falsa'.
Conexión de E/S	Definición de los enlaces entre las variables de la aplicación y los canales de las tarjetas existentes en el sistema objeto.
Contacto	Componente gráfico de un programa LD. Representa el estado de una variable de entrada.
Conversión	Filtro asociado a una variable analógica de entrada o salida. Se aplica la conversión automáticamente cada vez que se produce una entrada o salida de variable.
Franquear una transición	Operación en tiempo de proceso: se eliminan todas las marcas existentes en los pasos precedentes. Se coloca una marca en cada uno de los pasos subsiguientes.
Diagrama de Escalera	Lenguaje gráfico que combina contactos y bobinas, destinado al diseño de ecuaciones booleanas.

Diagrama de Funciones Secuenciales	Lenguaje gráfico: el proceso está descrito como un conjunto de pasos enlazados por transiciones. Se asocian acciones a los pasos. Las transiciones están detalladas como condiciones booleanas.
Diagramas de Bloques de Funciones	Lenguaje gráfico: se construyen las ecuaciones con bloques elementales estándares de la librería ISaGRAF, enlazados en el diagrama.
Diario	Fichero de texto que contiene todas las notas sobre los cambios que se realizan en un programa. Cada nota se completa con su fecha de edición.
Diccionario	Conjunto de variables internas, de entrada o de salida y palabras definidas que se utilizan en los programas de un proyecto.
Dirección de red	Dirección hexadecimal opcional, libremente definible para cada variable. El protocolo Modbus utiliza esta dirección cuando el sistema objeto está conectado a un sistema externo.
Marca (SFC)	Señalizador gráfico que se utiliza para mostrar los pasos activos de un programa SFC.
E/S bloqueada	Variable de entrada o salida, con desconexión lógica del correspondiente dispositivo de E/S por medio de un comando de bloqueo “Lock” que emite el usuario desde el depurador.
Entero	Clase de variable analógica, almacenada en formato de valor entero de 32 bits, con signo.
Entrada	Atributo de una variable. Esto tipo de variable está enlazado con un dispositivo de entrada.
Error de tiempo de proceso	Error de aplicación detectado por el sistema ISaGRAF objeto en el tiempo de proceso.
Etiqueta (IL)	Identificador que se coloca al principio de una línea de instrucciones IL, que identifica la instrucción y puede utilizarse como operando para las operaciones con el comando JMP.
Expresión	Conjunto de operadores e identificadores que representa la evaluación de un valor.
Expresión constante	Expresión literal que se emplea para describir un valor constante. Una expresión constante está dedicada a un tipo.
FBD	Lenguaje de Diagramas de Bloques de Función.
Flanco	Cambio en una variable booleana. Un flanco de subida significa un cambio de falso a verdadero. Un flanco de bajada significa un cambio de verdadero a falso.

Función C Función escrita en lenguaje C e invocada desde los programas IsaGRAF (escritos en otros lenguajes) de manera síncrona. Las funciones C están suministradas por CJ International, o desarrolladas por el usuario.

Función de conversión Función escrita en C que describe una conversión. Se pueden asociar las conversiones de este tipo a cualquier variable analógica de entrada o salida.

Global Rango de variables o palabras definidas. Se puede utilizar este tipo de objeto en cualquier programa de un proyecto.

Identificador Palabra exclusiva que se utiliza para representar una variable o una expresión constante en la programación.

IL Lenguaje de Lista de Instrucciones.

Instrucción Operación elemental de un programa IL, introducido en una línea de texto.

Notas técnicas Manual del usuario de un elemento de las librerías ISaGRAF (función o bloque de funciones C, función de conversión o tarjeta de E/S). Las notas técnicas están escritas por el diseñador del elemento.

Interna Atributo de una variable que no está vinculada a un dispositivo de entrada o salida.

Jerarquía Arquitectura de un proyecto, dividida en varios programas. El árbol jerárquico representa los enlaces existentes entre los programas padre y los programas hijo.

LD Lenguaje de Diagrama de Escalera.

Lenguaje C Lenguaje literal de alto nivel que se utiliza para describir operaciones informáticas, tales como funciones C y funciones de conversión.

Librería Conjunto de recursos de *hardware* o *software* que pueden insertarse directamente en cualquier aplicación.

Lista de Instrucciones Lenguaje literal de bajo nivel que se introduce como una lista secuencial de operaciones elementales.

Local Rango de variables o palabras definidas. Los objetos de este tipo sólo se pueden utilizar en un programa de un proyecto.

Macropaso Componente gráfico SFC. Un macropaso es un grupo único de pasos y transiciones que se representa como un solo icono en el diagrama principal, y se describe de forma independiente.

Matriz División lógica del área de edición en celdas rectangulares, al

	editarse un programa escrito con un lenguaje gráfico.
Mensaje	Tipo de variable. Las variables de este tipo contienen cadenas de caracteres de longitud variable.
Modbus	Protocolo Maestro-Esclavo. Un sistema ISaGRAF objeto puede ser un esclavo Modbus en el enlace con un sistema externo (por ejemplo, un sistema de supervisión) en una arquitectura completa.
Modificador (IL)	Un único carácter colocado al final de una palabra clave de operación IL, que modifica el significado de la operación.
Modo ciclo a ciclo	Modo de ejecución: en este modo, se ejecutan los ciclos de uno en uno, de acuerdo con las órdenes emitidas por el usuario del depurador.
Modo tiempo real	Modo normal de ejecución en tiempo de proceso: los ciclos objeto están activados de acuerdo con los tiempos de ciclo programados.
Nivel 1 del lenguaje SFC	Descripción principal de un programa SFC. El nivel 1 agrupa el diagrama (pasos y transiciones) y los comentarios asociados.
Nivel 2 del lenguaje SFC	Descripción detallada de un programa SFC. Es la descripción de las acciones existentes dentro de los pasos y de las condiciones booleanas que están asociadas a las transiciones.
Número de referencia (SFC)	Número decimal (entre 1 y 65535) que identifica un paso o transición SFC en un programa SFC.
Objeto	Máquina objeto de ISaGRAF, que soporta el *software* del *kernel* de ISaGRAF.
Operación (IL)	Instrucción básica del lenguaje IL. Generalmente, una operación está asociada a un operando de una instrucción.
Operación demorada (IL)	Operación de un programa IL que se ejecuta cuando aparece la instrucción "(", más adelante en el programa.
Operando (IL)	Variable o expresión constante procesada por una instrucción elemental IL.
Palabra clave	Palabra reservada del lenguaje.
Palabra definida	Identificador exclusivo que se utiliza para sustituir cualquier expresión en un programa.
Parámetro (función C)	Valor atribuido como entrada a una función C. Un parámetro está caracterizado por un tipo.
Parámetro (tarjeta E/S)	Parámetro definido por el usuario o constante de una tarjeta estándar de E/S. El programador introduce los parámetros

	definidos por el usuario durante la conexión de E/S.
Parámetro OEM (tarjeta de E/S)	Parámetro de tarjeta de E/S, definido por el diseñador de la tarjeta. Puede ser una constante o un parámetro variable que introduce el usuario durante la conexión de E/S.
Paso	Componente gráfico básico del lenguaje SFC. Un paso representa una situación estable del proceso y se dibuja en forma de cuadrado. Los pasos están referenciados con números. Se utiliza la actividad de un paso para controlar la ejecución de las acciones correspondientes.
Paso comienzo	El primer paso que aparece en el cuerpo de un macropaso. Los pasos iniciales no están vinculados a las transiciones precedentes.
Paso fin	El último paso que aparece en el cuerpo de un macropaso SFC. Un paso final no está vinculado a ninguna de las transiciones posteriores.
Paso inicial	Paso especial de un programa SFC, que se activa cuando arranca el sistema.
Programa	Unidad básica de programación de un proyecto. Se describe el proyecto con un lenguaje y se coloca en la arquitectura jerárquica del proyecto.
Programa padre	Programa escrito en cualquier lenguaje, que controla (invoca) a otro programa no SFC, conocido como su subprograma.
Programa de nivel superior	Programa situado en lo más alto del árbol jerárquico. Un programa de nivel superior está activado por el sistema.
Programa SFC hijo	Programa SFC controlado por otro programa SFC, conocido como su padre.
Programa SFC padre	Programa SFC que controla a otros programas SFC, conocidos como sus hijos.
Proyecto	Área de programación que aglutina toda la información (programas, variables, código destino, etc.) de una aplicación ISaGRAF.
Breakpoint	Señal colocada por el usuario en un paso o una transición SFC, a la hora de depurar. El sistema objeto se para cuando se desplaza un marca SFC sobre un breakpoint.
Rango	Conjunto de programas que puede utilizar un objeto. Los rangos predefinidos de ISaGRAF son comun, global y local.
Real	Clase de variable analógica que se almacena en formato flotante IEEE de 32 bits y simple precisión.
Referencias cruzadas	Información calculada por el banco de trabajo ISaGRAF en relación

	al diccionario de variables, y los lugares en los cuales se utilizan esas variables en un proyecto.
Registro (IL)	Resultado actual de una secuencia IL.
Resultado actual (IL)	Resultado de una instrucción en un programa IL. El resultado actual puede ser modificado por una instrucción, o puede utilizarse para asignar a una variable.
Salida	Atributo de una variable. Las variables de este tipo están enlazadas al dispositivo de salida del equipo objeto.
Salto a un paso	Componente gráfico SFC que representa el enlace entre una transición y un paso. El símbolo gráfico de un salto es una flecha que presenta el número de referencia del paso destino.
Sección	Grupo de programas ejecutado con las mismas reglas dinámicas.
Sección de comienzo	Grupo de programas cíclicos que se ejecuta al principio de cada ciclo objeto.
Sección de fin	Grupo de programas cíclicos que se ejecuta al término de cada ciclo objeto.
Sección secuencial	Grupo de programas de un proyecto. La ejecución de estos programas cumple con las reglas dinámicas del lenguaje SFC.
Sentencia	Operación ST básica completa.
Separador	Carácter especial (o grupo de caracteres) que se utiliza para separar a los identificadores en un lenguaje literal. Un separador puede representar una operación.
SFC	Lenguaje 'Diagrama de Funciones Secuencial'.
Situación inicial	Conjunto de pasos iniciales de un programa SFC, que representa el contexto del programa cuando se arranca.
ST	Lenguaje 'Textos Estructurados'.
Subprograma	Programa escrito en cualquier lenguaje (salvo SFC) e invocado por otro programa, conocido como su programa propietario.
Tabla de conversión	Conjunto de puntos que define una conversión lineal (por segmentos). Se puede aplicar las conversiones de este tipo a cualquier variable analógica de entrada o salida.
Tarjeta de E/S	Recurso *hardware*. Una tarjeta de E/S está caracterizada por un tipo y una dirección (entrada o salida). Se describen los parámetros de una tarjeta de E/S en la librería ISaGRAF.
Tarjeta real	Tarjeta de E/S conectada físicamente a un dispositivo de E/S en la

	máquina objeto.
Tarjeta virtual	Tarjeta de E/S que no está conectada físicamente a un dispositivo de E/S en la máquina objeto.
Temporizador	Tipo de variable. Las variables de este tipo contienen valores de tiempo y pueden ser refrescadas automáticamente por el sistema ISaGRAF durante el tiempo de proceso.
Texto Estructurado	Lenguaje literal estructurado de alto nivel, que combina asignaciones, estructuras de alto nivel como 'If/Then/Else,' e invocaciones a funciones.
Tiempo de ciclo	Duración del ciclo de ejecución objeto.
Tipo	Clase de variables que tienen el mismo formato. Los tipos básicos son: booleanas, analógicas, temporizadores y mensajes.
Transición	Componente gráfico básico SFC. Una transición representa la condición existente entre diferentes pasos SFC. Las transiciones están referenciadas por un número. Se asocia una condición booleana a cada transición.
Validez de una transición	Atributo de una transición. Se valida una transición cuando todos los pasos precedentes están activos.
Valor de retorno de un subprograma	Valor retornado por un subprograma al final de su ejecución. Se utiliza el valor de retorno en las operaciones del programa propietario.
Variable	Representación única de datos elementales que se procesan en los programas del proyecto.
Variable de E/S	Variable asociada a un dispositivo de entrada o salida. Se tiene que conectar una variable de E/S a un canal de una tarjeta de E/S.

E. Símbolos

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

X