RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente

2 Estructura de IndraLogic

2.1 Partes de un proyecto

Proyecto

Un proyecto contiene todos los objetos de un programa de control. Un proyecto se guarda en un archivo con el nombre del proyecto. A un proyecto pertenecen los siguientes objetos:

componentes, tipos de datos, visualizaciones, recursos y bibliotecas.

Componente

Las funciones, los bloques de función y los programas son componentes que pueden completarse mediante acciones.

Cada componente consta de una parte de declaración y una parte de código. La parte de código está escrita en uno de los lenguajes de programación IEC, tales como AWL, ST, AS, FUP, KOP o CFC.

IndraLogic soporta todos los componentes estándar IEC. Si desea utilizar estos componentes en su proyecto, debe integrar en éste la biblioteca standard.lib.

Los componentes pueden llamar a otros componentes. Sin embargo, no están permitidas las recursiones.

Función

Una función es un componente que, como resultado de la ejecución, arroja siempre un dato (que también puede constar de varios elementos, como p. ej. campos o estructuras) . En lenguajes textuales, la llamada de una función puede aparecer como un operador en expresiones.

Para la declaración de una función, es preciso recordar que la función debe recibir un tipo. Por lo tanto, detrás del nombre de la función se deben introducir dos puntos seguidos de un tipo.

Ejemplo de una declaración de función correcta:

```
FUNCTION FCT: INT
```

Fig. 2-1: Declaración de función

Además, se debe asignar un resultado a la función. Esto es, se utiliza el nombre de la función como una variable de salida.

Una declaración de función empieza con la palabra clave FUNCTION.

En AS, una llamada de función sólo puede tener lugar dentro de acciones de un paso o en una transición.

En ST puede utilizarse una llamada de función como operando en expresiones.

Ejemplos de la llamada de la función anteriormente descrita:

```
LD 7
Fct 2,4
ST  Resultado
```

Fig. 2-2: Llamada de una función en AWL

```
Resultado := Fct(7, 2, 4);
```

Fig. 2-3: Llamada de una función en ST

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente

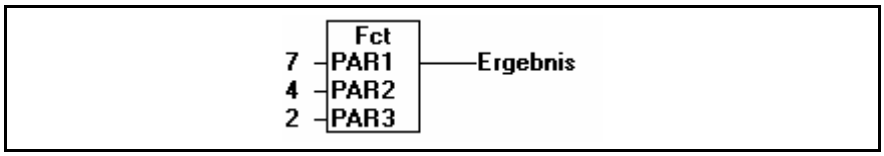

Fig. 2-4: Llamada de una función en FUP

Las funciones carecen de estados internos. Esto significa que las llamadas a una función con los mismos argumentos (parámetros de entrada) arrojan siempre el mismo valor (salida). Por este motivo, las funciones no deben contener variables ni direcciones globales.

AVISO

Si una variable local en una función se declara como RETAIN, esto no tiene consecuencias. ¡La variable no se guarda en el área Retain!

Nota: Si define en su proyecto una función con el nombre **CheckBounds**, con ella puede comprobar automáticamente si se han producido superaciones de campo en arrays (ver "Apéndice C: Tipos de datos en IndraLogic" a partir de la página 12-1).
Si define las funciones **CheckDivByte**, **CheckDivWord**, **CheckDivDWord** y **CheckDivReal**, si utiliza el operador DIV puede comprobar con ellas el valor del divisor, por ejemplo para evitar una división por 0 (ver "Apéndice A: Operadores IEC y funciones adicionales que amplían la norma" a partir de la página 10-1).
Si usted define las funciones **CheckRangeSigned** y **CheckRangeUnsigned**, en el modo Online con ellas puede interceptar automáticamente superaciones de campo en caso de variables que han sido declaradas con tipos de subcampo (ver "Apéndice C: Tipos de datos en IndraLogic" a partir de la página 12-1).
Los nombres de funciones mencionados están reservados para el uso aquí descrito.

Módulo de función (bloque de función)

Un módulo de función - también denominado bloque de función - es un componente que al ejecutarse arroja uno o varios valores. A diferencia de una función, un bloque de función no arroja ningún valor de retorno.

Una declaración de bloque de función empieza con la palabra clave FUNCTION_BLOCK.

Se pueden crear reproducciones, denominadas instancias (copias) de un bloque de función.

Ejemplo en AWL de un bloque de función con dos variables de entrada y dos variables de salida. Una de las salidas es el producto de ambas entradas, y la otra una comparación de igualdad:

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente

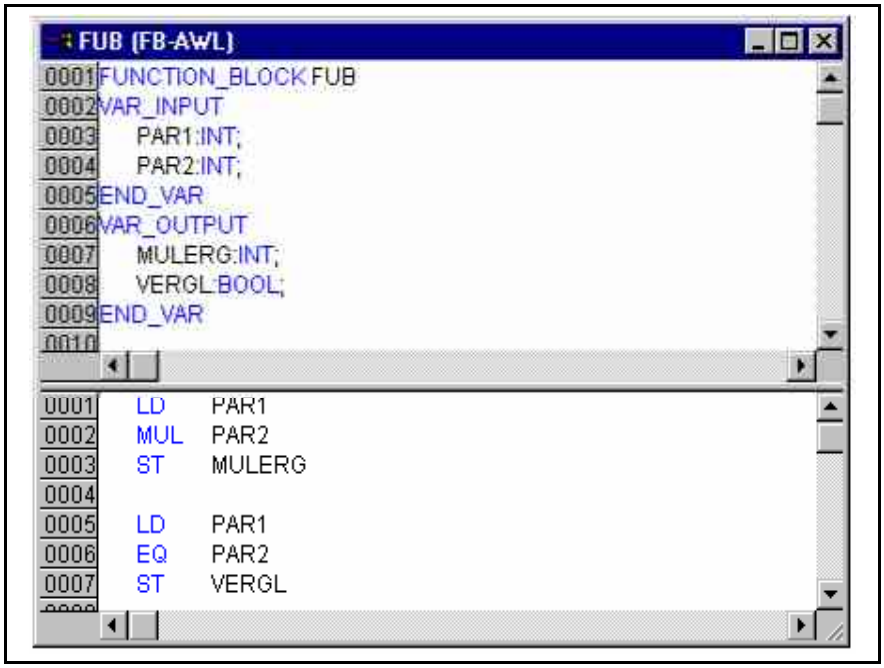

Fig. 2-5: Bloque de función con dos variables de entrada y dos variables de salida en AWL

Instancias de bloques de función

Se pueden crear reproducciones, denominadas instancias (copias) de un bloque de función.

Cada instancia posee su propio identificador (el nombre de instancia) y una estructura de datos que contiene sus entradas, salidas y variables internas. Al igual que las variables, las instancias se declaran local o globalmente, especificando como tipo de un identificador el nombre del bloque de función.

```
INSTANCIA: FUB;
```

Fig. 2-6: Instancia con el nombre INSTANCIA del bloque de función FUB

Las llamadas de bloques de función tienen lugar siempre a través de las instancias arriba descritas.

Desde fuera de una instancia de un bloque de función sólo es posible acceder a los parámetros de entrada y salida, pero no a sus variables internas.

Las partes de declaración de bloques de función y programas pueden contener declaraciones de instancias. Las declaraciones de instancias no están permitidas en funciones.

El acceso a la instancia de un bloque de función está limitado al componente en el que fue declarado, a no ser que haya sido declarado globalmente.

El nombre de instancia de una instancia de bloque funcional puede utilizarse como entrada de una función o de un bloque de función.

Nota: Tras el procesamiento de un bloque de función se conservan todos los valores hasta el siguiente procesamiento. ¡Por este motivo, las llamadas de un bloque de función con los mismos argumentos no siempre arrojan los mismos valores de salida!

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente

Nota: Si el bloque de función contiene al menos una variable Retain, se guarda la instancia en su conjunto en el área Retain.

Llamada de un bloque de función

Es posible llamar las variables de entrada y salida de un bloque de función desde otro componente, creando para ello una instancia del bloque de función e introduciendo la variable deseada utilizando la siguiente sintaxis:

<Nombre de instancia>.<Nombre de variable>

Asignación de los parámetros al producirse la llamada:

Si se desean establecer los parámetros de entrada y/o de salida al llamar el bloque de función, esto puede hacerse en los lenguajes de texto AWL y ST, asignando entre paréntesis valores a los parámetros después del nombre de instancia del bloque de función. Al igual que en la inicialización de variables en la posición de declaración, en el caso de los parámetros de entrada la asignación tiene lugar mediante ":=", mientras que para los parámetros de salida se utiliza "=>").

Si se inserta la instancia utilizando la ayuda de entrada (<F2>) con la opción **Con argumentos** en la ventana de implementación de un componente ST o AWL, se representa automáticamente en esta sintaxis con sus parámetros. Pero entonces no se deben asignar necesariamente los parámetros.

Ejemplo:

FBINST es una variable local del tipo de un bloque de función, que contiene la variable de entrada xx y la variable de salida yy. Al insertar FBINST en un programa ST mediante la ayuda de entrada, la llamada se visualizará de la siguiente forma: `FBINST1(xx:= , yy=> );`

VariablesEntradaSalida al producirse la llamada:

Tenga en cuenta que las **VariablesEntradaSalida (VAR_IN_OUT)** de un bloque de función se transmiten como **pointers**. Por este motivo, al producirse la llamada no se les pueden asignar constantes, y no es posible el acceso de lectura o escritura a ellas desde fuera.

```
VAR
 inst:fubo;
 var1:int;
END_VAR
var1:=2;
inst(inout1:=var1);
```

Fig. 2-7: Ejemplo de llamada de las variables VAR_IN_OUT inout1 del bloque de función fubo en un componente ST

Nota: No estaría permitido: inst(inout1:=2); ni inst.inout1:=2;

Ejemplos de llamada del bloque de función FUB:

(Descripción del bloque de función en la página 2-2)

El resultado de multiplicación se guarda en las variables ERG, el resultado de la comparación se guarda en QUAD. Se ha declarado una instancia de FUB con el nombre INSTANCIA.

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente

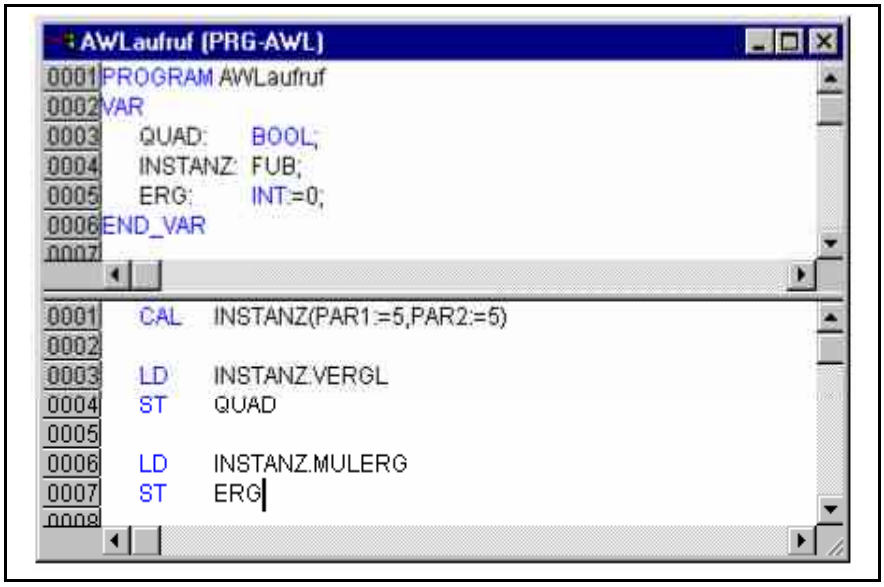

Fig. 2-8:: Llamada de la instancia de un bloque de función en AWL

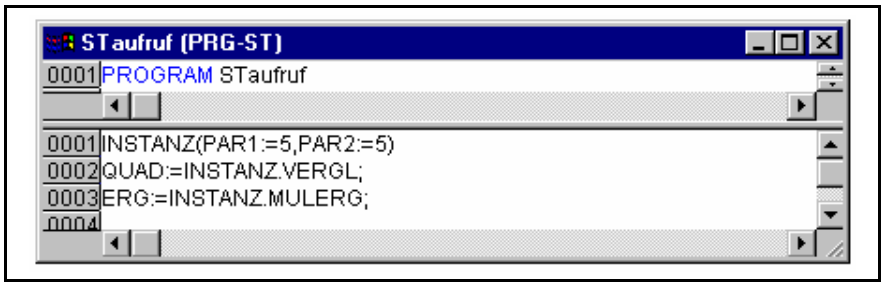

Fig. 2-9: Llamada de la instancia de un bloque de función en ST (la parte de declaración es la misma que en AWL en **Fig. 2-8:**):

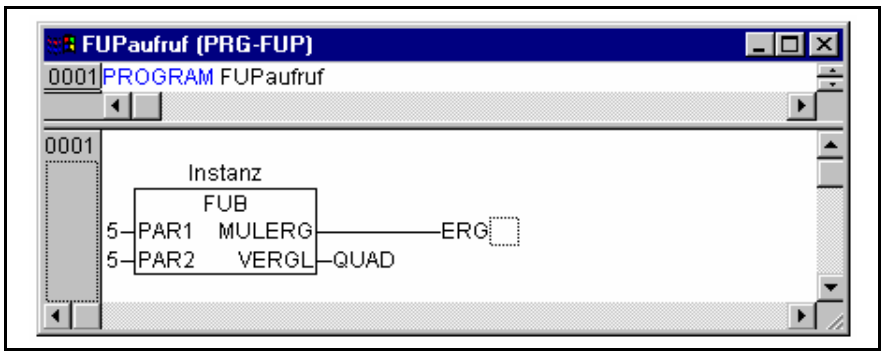

Fig. 2-10: Llamada de la instancia de un bloque de función en FUP (la parte de declaración es la misma que en AWL en **Fig. 2-8:**):

En AS, las llamadas de bloques de función sólo pueden tener lugar paso a paso.

Programa

Un programa es un componente que al ejecutarse arroja uno o varios valores. Los programas se reconocen globalmente en todo el proyecto. Tras la ejecución de un programa se conservan todos los valores hasta la siguiente ejecución.

Una declaración de programa empieza con la palabra clave PROGRAM y termina con END_PROGRAM.

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente

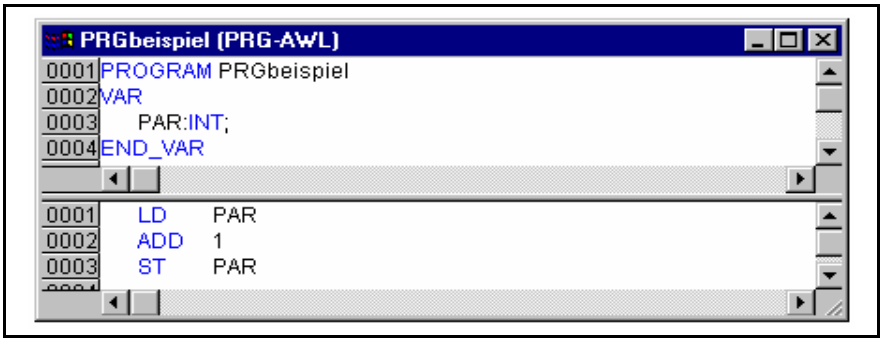

Fig. 2-11: Ejemplo de un programa

Los programas pueden llamarse desde programas y bloques de función. No está permitida la llamada de un programa en una función. Tampoco existen instancias de programas.

Cuando un componente llama un programa, y durante el proceso se modifican valores del programa, dichas alteraciones se mantienen en la siguiente llamada del programa, aunque el programa sea llamado desde otro componente.

Esto difiere de la llamada de un bloque de función. Allí sólo se modifican los valores en la instancia pertinente de un bloque de función. Por lo tanto, estos cambios sólo son relevantes si se llama la misma instancia.
Si se desea establecer los parámetros de entrada y/o de salida al llamar un programa, es decir, si se desea ajustar los valores de las variables de entrada/salida al realizar la llamada, en los lenguajes de texto AWL y ST esto se lleva a cabo asignando entre paréntesis valores a los parámetros detrás del nombre del programa. Al igual que en la inicialización de variables en la posición de declaración, la asignación tiene lugar mediante ":=".

Si se inserta el programa utilizando la ayuda de entrada (<F2>) con la opción **Con argumentos** en la ventana de implementación de un componente ST o AWL, se representa automáticamente en esta sintaxis con sus parámetros. Pero entonces no se deben asignar necesariamente los parámetros.

Ejemplos de llamadas de un programa:

En un programa PRGexample2, están declaradas la variable de entrada in_var y la variable de salida out_var, ambas del tipo INT. Está declarada localmente la variable erg, también del tipo INT:

```
CAL PRGexample2
LD  PRGexample2.out_var
ST  ERG
```

Fig. 2-12: Llamada de un programa en AWL

```
CAL PRGexample2(in_var:=33, out_var=>erg )
```

Fig. 2-13: Llamada de un programa en AWL con especificación inmediata de los parámetros (ayuda de entrada "Con argumentos", ver arriba)

```
PRGexample;
Erg := PRGexample2.out_var;
```

Fig. 2-14: Llamada de un programa en ST

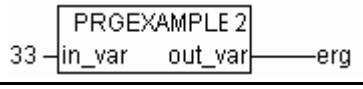

Fig. 2-15: Llamada de un programa en FUP

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente

Ejemplo de una secuencia de llamadas posible para PLC_PRG:

A este respecto, ver el programa PRGejemplo en la figura al principio de este capítulo:

```
LD 0
ST  PRGejemplo.PAR (*el ajuste predeterminado para PAR es
0*)
CAL AWLllamada  (*ERG en AWLllamada resulta en 1*)
CAL STllamada  (*ERG en STllamada resulta en 2*)
CAL FUPllamada  (*ERG en FUPllamada resulta en 3*)
```

Fig. 2-16: Secuencia de llamadas posible para PLC_PRG

Si desde un programa principal se inicializa con 0 primero la variable PAR del programa PRGejemplo, y a continuación se llaman sucesivamente programas con las llamadas de programa arriba indicadas, el resultado ERG en los programas tendrá los valores 1, 2 y 3. Si se confunde la secuencia de las llamadas, los valores de los parámetros de resultado correspondientes cambian en consecuencia.

PLC_PRG

Es posible, pero no obligatorio, controlar la ejecución del proyecto mediante las denominadas tareas (configuración de tareas). No obstante, en caso de no existir ninguna configuración de tareas, el proyecto debe contener el componente PLC_PRG. El PLC_PRG es creado automáticamente como componente del tipo programa cuando en un proyecto de nueva creación se inserta por primera vez un componente mediante **"Proyecto" "Insertar objeto"**. PLC_PRG es llamado exactamente una vez por ciclo de control.

Si existe una configuración de tareas, se permite que el proyecto no contenga ningún PLC_PRG, dado que en este caso la secuencia de ejecución depende de la asignación de tareas.

Nota: **No borre el componente PLC_PRG y tampoco le cambie el nombre** (presuponiendo que no está utilizando una configuración de tareas. Generalmente, PLC_PRG es el programa principal en un programa de una sola tarea)

Recursos

Necesita los recursos para configurar y organizar su proyecto, así como para el seguimiento de los valores de variables:

- **Variables globales** que pueden utilizarse en todo el proyecto o la red
- **Bibliotecas** que pueden incorporarse al proyecto mediante el administrador de bibliotecas
- **Registro** para registrar las actividades Online
- **Configuración del control** para configurar su hardware
- **Configuración de tareas** para controlar su programa mediante tareas
- **Administrador watch y de fórmulas** para visualizar valores de variables y establecer valores de variables predeterminados
- **Ajustes del sistema de destino** para la selección y, si fuera preciso, la configuración del sistema de destino
- Área de trabajo con una imagen de las opciones del proyecto

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente

Dependiendo del sistema de destino seleccionado y de los ajustes del sistema de destino realizados en IndraLogic, también pueden estar disponibles los siguientes recursos:

- **Administrador de parámetros** para el intercambio de datos con otros controles en una red
- Navegador PLC como monitor del control
- **Registro de seguimiento** para el registro gráfico de valores de variables
- **Herramientas** para llamar aplicaciones externas

Acción

Las acciones pueden ser definidas para bloques de función y programas y añadidas a éstos ("Proyecto" "Añadir acción"). La acción representa una implementación adicional que puede ser creada perfectamente en un lenguaje distinto al de la implementación "normal". A cada acción se le asigna un nombre.

Una acción trabaja con los datos del bloque de función o del programa al que pertenece. La acción utiliza las mismas variables de entrada/salida y variables locales que la implementación "normal".

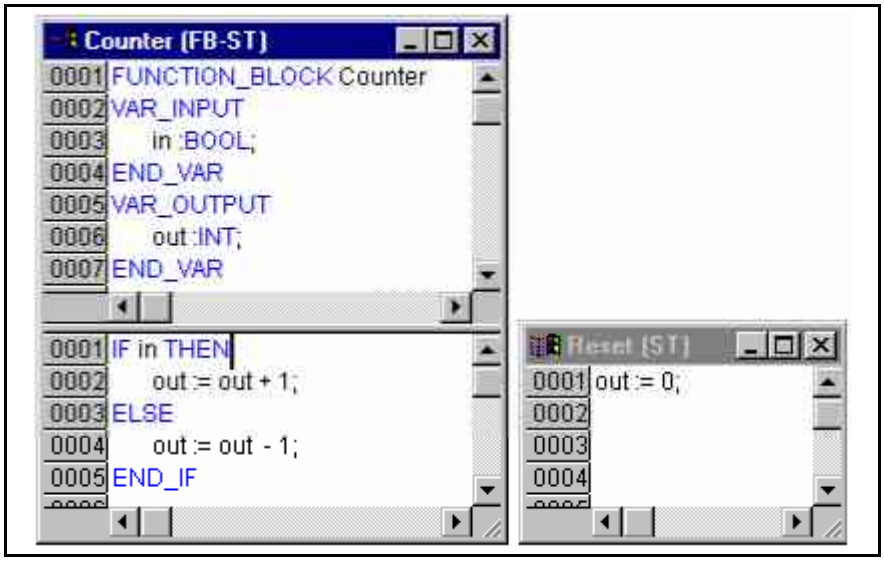

Fig. 2-17:: Ejemplo de acción de un bloque de función

En el ejemplo mostrado en **Fig. 2-17:**, al llamar el bloque de función Counter aumenta o reduce la variable de salida "out", dependiendo de la variable de entrada "in". Al llamar la acción Reset del bloque de función, se ajusta a cero la variable de salida "out". En ambos casos se escribe la misma variable "out".

Llamada de una acción:

Una acción se llama mediante <Nombre de programa>.<Nombre de acción> o bien <Nombre de instancia>.<Nombre de acción>. Tenga en cuenta la notación en FUP (ver el ejemplo inferior). Si es preciso llamar la acción dentro del componente propio, en los editores de texto se utiliza sólo el nombre de la acción y, en los lenguajes gráficos, la llamada del bloque de función sin indicación de la instancia.

Ejemplos de llamadas de la acción mencionada desde otro componente:

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente


```
PROGRAM PLC_PRG
VAR
 inst : counter;
END_VAR
```

Fig. 2-18: Declaración para los siguientes principios:

```
CAL inst.Reset(in := FALSE)
LD   inst.out
ST  ERG
```

Fig. 2-19: Llamada de la acción "Reset" en otro componente, que está programado en AWL

```
inst.Reset(in := FALSE);
Erg := inst.out;
```

Fig. 2-20: Llamada de la acción "Reset" en otro componente, que está programado en ST

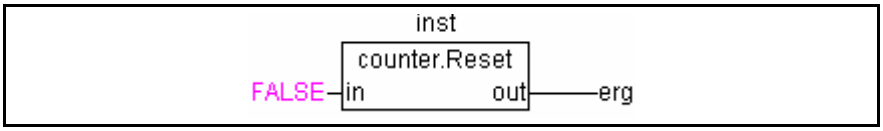

Fig. 2-21: Llamada de la acción "Reset" en otro componente, que está programado en FUP

Nota: Las acciones desempeñan un papel importante en componentes programados en lenguajes de proceso.

Bibliotecas

Puede incluir en su proyecto una serie de bibliotecas cuyos componentes, tipos de datos y variables globales pueden ser utilizados exactamente igual que los definidos por el propio usuario. Las bibliotecas standard.lib y util.lib son partes estándar del programa y están siempre disponibles.

A este respecto, ver el capítulo 6.3

Tipos de datos

Además de los tipos de datos estándar, el usuario puede definir tipos de datos propios. Se pueden crear estructuras, tipos de enumeración y referencias.

A este respecto, ver el "Apéndice C: Tipos de datos en IndraLogic" a partir de la página 12-1

Visualización

IndraLogic le ofrece la posibilidad de una visualización para la representación de las variables de su proyecto. Mediante la visualización se pueden dibujar elementos geométricos en el modo Offline. Posteriormente, en el modo Online, éstos pueden cambiar su forma/color/salida de texto en función de determinados valores de variable.

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente

2.2 Los lenguajes

Lenguajes de programación soportados

IndraLogic soporta todos los lenguajes de programación descritos en la norma IEC-61131:

Lenguajes textuales:

- Lista de instrucciones (AWL)
- Texto estructurado (ST)

Lenguajes gráficos:

- Lenguajes de proceso (AS)
- Esquema de contactos (KOP)
- Esquema de funciones (FUP)
- Además, sobre la base del esquema de funciones existe el Esquema de funciones gráfico libre (CFC)

Lista de instrucciones (AWL)

Una lista de instrucciones (AWL, siglas en alemán de "Anweisungsliste") consta de una secuencia de instrucciones. Cada instrucción se inicia en una línea nueva y contiene un operador y, según el tipo de operación, uno o varios operandos separados por comas.

Delante de una instrucción puede encontrarse un identificador *marca*, seguido de dos puntos (:). Sirve para identificar la instrucción y puede utilizarse por ejemplo como destino de salto.

El último elemento en una línea debe ser un comentario. Pueden insertarse líneas vacías entre instrucciones.

```
LD 17
ST lint (* comentario *)
GE 5
JMPC next
LD idword
EQ istruct.sdword
STN test
next:
```

Fig. 2-22: Ejemplo de programa en AWL

Modificadores y operadores en AWL

En el lenguaje AWL pueden utilizarse los modificadores y operadores recogidos en las siguientes tablas.

• C	con JMP, CAL, RET:	Sólo se ejecuta la instrucción si el resultado de la expresión precedente es VERDADERO.
• N	con JMPC, CALC, RETC:	Sólo se ejecuta la instrucción si el resultado de la expresión precedente es FALSO.
• N	en los demás casos:	Negación del operando (no del acumulador).

Fig. 2-23: Modificadores

La siguiente tabla muestra todos los operadores en el lenguaje AWL con sus posibles modificadores y su correspondiente significado:

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente

Operador	Modificadores	Significado
LD	N	Iguala el resultado actual al operando
ST	N	Guarda el resultado actual en la posición del operando
S		Ajusta el operando booleano exactamente a VERDADERO si el resultado actual es VERDADERO
R		Ajusta el operando booleano exactamente a FALSO si el resultado actual es VERDADERO
AND	N,(	Y bit a bit
OR	N,(	O bit a bit
XOR	N,(	O exclusivo bit a bit
ADD	(	Suma
SUB	(	Resta
MUL	(	Multiplicación
DIV	(	División
GT	(	>
GE	(	>=
EQ	(	=
NE	(	<>
LE	(	<=
LT	(	<
JMP	CN	Salto a marca
CAL	CN	Llamar programa o bloque de función
RET	CN	Salir del componente y volver al llamante.
)		Evaluar operación diferida

Fig. 2-24: Operadores y modificadores en AWL

En el Apéndice hallará un listado de todos los operadores IEC.

```
LD    TRUE   (*cargar TRUE [VERDADERO] en el acumulador*)
ANDN  BOOL1  (*ejecutar AND con el valor negado de la
      variable BOOL1*)
JMPC  salida del bucle    (*si el resultado era TRUE,
      saltar a la marca "marca"*)

LDN   BOOL2  (*guardar el valor negado de *)
ST    ERG    (*BOOL2 en ERG*)
Marca:
LD    BOOL2  (*guardar el valor de *)
ST    ERG    (*BOOL2 en ERG*)
```

Fig. 2-25: Programa AWL utilizando algunos modificadores

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente

En AWL también es posible colocar paréntesis detrás de una operación. En tal caso, el valor entre paréntesis se considera como operando.

```
LD 2
MUL 2
ADD 3
Erg    (*Aquφ est■ el valor de Erg 7 *)
       (*Pero si se colocan parΘntesis: *)
LD 2
MUL (2
ADD 3
)
ST Erg (*el valor resultante para Erg es 10, ya que la
        operaci≤n MUL no se eval╖a hasta que se llega a
        ")"; como operando para MUL se calcula entonces
5.*)
```

Fig. 2-26: Colocación de paréntesis en AWL

Texto estructurado (ST)

El texto estructurado consta de una serie de instrucciones que pueden ejecutarse tal como se determina en lenguajes de alto nivel ("IF..THEN..ELSE) o bien en bucles (WHILE..DO).

```
IF value < 7 THEN
 WHILE value < 8 DO
  value := value + 1;
 END_WHILE;
END_IF;
```

Fig. 2-27: Programa en ST

Expresiones

Una *expresión* es una construcción que arroja un valor tras su evaluación.

Las expresiones se componen de operadores y operandos. Un operando puede ser una constante, una variable, una llamada de función o bien otra expresión.

Evaluación de expresiones

La evaluación de una expresión tiene lugar mediante el procesamiento de los operadores conforme a ciertas *reglas de vinculación*. En primer lugar se procesa el operador con el vínculo más fuerte, a continuación el operador con el segundo vínculo más fuerte, etc., hasta que se han procesado todos los operadores.

Los operadores con la misma fuerza de vinculación se procesan de izquierda a derecha.

A continuación, se muestra una tabla de los operadores ST en el orden de su fuerza de vinculación:

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente

Operación	Símbolo	Fuerza de vinculación
Poner entre paréntesis	(expresión)	La vinculación más fuerte
Llamada de función	Nombre de función (lista de parámetros)	
Elevar a una potencia	EXPT	
Negar Formación de complementos	- NOT	
Multiplicar Dividir Módulo	* / MOD	
Sumar Restar	+ -	
Comparar	<,>,<=,>=	
Igualdad Desigualdad	= <>	
AND booleano	AND	
XOR booleano	XOR	
OR booleano	OR	La vinculación más débil

Fig. 2-28: Operadores en ST, ordenados según la fuerza de vinculación

En la siguiente tabla se indican las instrucciones posibles en ST y se ofrece un ejemplo de cada una:

Tipo de instrucción	Ejemplo
Asignación	A:=B; CV := CV + 1; C:=SIN(X);
Llamada de un bloque de función y uso de la salida FB	CMD_TMR(IN := %IX5, PT := 300); A:=CMD_TMR.Q
RETURN	RETURN;

IF	D:=B*B;
	IF D<0.0 THEN
	C:=A;
	ELSIF D=0.0 THEN
	C:=B;
	ELSE
	C:=D;
	END_IF;

CASE	CASE INT1 OF 1: BOOL1 := TRUE; 2: BOOL2 := TRUE; ELSE BOOL1 := FALSE; BOOL2 := FALSE; END_CASE;

FOR	J:=101;
	FOR I:=1 TO 100 BY 2 DO
	IF ARR[I] = 70 THEN
	J:=I;
	EXIT;
	END_IF;
	END_FOR;

WHILE	J:=1; WHILE J<= 100 AND ARR[J] <> 70 DO J:=J+2; END_WHILE;

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente

REPEAT	J:=-1; REPEAT J:=J+2; UNTIL J= 101 OR ARR[J] = 70 END_REPEAT;
EXIT	EXIT;
Instrucción vacía	;

Fig. 2-29: Instrucciones en ST

Operador de asignación

En el lado izquierdo de una asignación se halla un operando (variable, dirección), al cual se asigna el valor de la expresión situada en el lado derecho, mediante el operador de asignación :=.

```
Var1 := Var2 * 10;
```

Fig. 2-30:: Asignación en ST

Tras la ejecución de la línea indicada en **Fig. 2-30:**, Var1 posee diez veces el valor de Var2.

Llamada de bloques de función en ST

Para llamar un bloque de función en ST se escribe el nombre de la instancia del bloque de función y a continuación, entre paréntesis, se asignan a los parámetros los valores deseados. En el siguiente ejemplo se llama un Timer con asignaciones para sus parámetros IN y PT. A continuación, se asigna la variable de resultado Q a la variable A.

Al igual que en AWL, la variable de resultado se direcciona mediante el nombre del bloque de función, seguido de un punto y el nombre de la variable:

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

Fig. 2-31: Llamada de un bloque de función en ST

Instrucción RETURN

La instrucción RETURN puede utilizarse para salir de un componente, por ejemplo, dependiendo de una condición.

Instrucción CASE

La instrucción CASE permite agrupar en una construcción varias instrucciones condicionadas con las mismas variables de condición.

```
CASE <Var1>
OF
<Valor 1>: <Instrucci≤n 1>
<Valor 2>: <Instrucci≤n 2>
<Valor3, Valor4, Valor5: <Instrucci≤n 3>
<Valor6 .. Valor10 : <Instrucci≤n 4>
...
<Valor n>: <Instrucci≤n n>
ELSE <Instrucci≤n ELSE>
END_CASE;
```

Fig. 2-32: Sintaxis de la instrucción CASE en ST

Una instrucción CASE se procesa conforme al siguiente esquema:

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente

- Si la variable en <Var1> tiene el valor <Valor i>, se ejecuta la instrucción <Instrucción i>.
- Si <Var 1> no tiene ninguno de los valores indicados, se ejecuta la <Instrucción ELSE>.
- Si se debe ejecutar la misma instrucción para varios valores de las variables, se pueden escribir estos valores uno tras otro separados por comas, condicionando así la instrucción en su conjunto.
- Si se debe ejecutar la misma instrucción para una gama de valores de las variables, se pueden escribir el valor inicial y el valor final uno tras otro separados por dos puntos, condicionando así la instrucción en su conjunto.

```
CASE INT1 OF
1, 5: BOOL1 := TRUE;
 BOOL3 := FALSE;
2: BOOL2 := FALSE;
 BOOL3 := TRUE;
10..20: BOOL1 := TRUE;
 BOOL3:= TRUE;
ELSE
 BOOL1 := NOT BOOL1;
 BOOL2 := BOOL1 OR BOOL2;
END_CASE;
```

Fig. 2-33: Instrucción CASE en ST

Instrucción IF

Mediante la instrucción IF se puede comprobar una condición y ejecutar instrucciones dependiendo de dicha condición.

```
IF <Expresi≤n_booleana1>
THEN
<Instrucciones_IF>
{ELSIF <Expresi≤n_booleana2>
THEN
<Instrucciones1_ELSIF>
.
.
ELSIF <Expresi≤n_booleana n>
THEN
<Instrucciones_ELSIF n-1>
ELSE
<Instrucciones_ELSE>
}
END_IF;
```

Fig. 2-34: Sintaxis de la instrucción CASE en ST (la parte entre corchetes {} es opcional)

Si <Expresión_booleana1> resulta ser TRUE (VERDADERO), se ejecutan únicamente las <Instrucciones_IF> y ninguna de las demás instrucciones.

De lo contrario, se evalúan sucesivamente las expresiones booleanas, empezando por la <Expresión_booleana2>, hasta que una de las expresiones resulte ser TRUE. Entonces se evalúan sólo las instrucciones detrás de esta expresión booleana y delante de la siguiente ELSE o ELSIF.

Si ninguna de las expresiones booleanas arroja TRUE, se evalúan exclusivamente las <Instrucciones_ELSE>.

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente


```
IF temp<17
THEN calefacci≤n_con := TRUE;
ELSE calefacci≤n_con := FALSE;
END_IF;
```

Fig. 2-35: Instrucción IF en ST

En el ejemplo mostrado en **Fig. 2-35** se conecta la calefacción cuando la temperatura desciende por debajo de 17 grados, de lo contrario permanece apagada.

Bucle FOR

Mediante el bucle FOR se pueden programar procesos repetidos.

```
INT_Var :INT;
FOR <INT_Var> := <INIT_VALOR>
TO <VALOR_FINAL>
{BY <Incremento>}
DO
<Instrucciones>
END_FOR;
```

Fig. 2-36: Sintaxis del bucle FOR en ST. (La parte entre corchetes {} es opcional.)

Las <instrucciones> se ejecutan mientras el contador <INT_Var> no sea superior al <VALOR_FINAL>. Esto se comprueba antes de ejecutar las <instrucciones>, de modo que las <instrucciones> no se ejecutan si el <VALOR_INIC> es superior al <VALOR_FINAL>.

Cuando se han ejecutado <instrucciones>, se aumenta siempre <INT_Var> en <incremento>. El incremento puede tener cualquier valor entero. Si no está presente, se ajusta a 1. Por lo tanto, el bucle debe terminar, dado que <INT_Var> va en aumento.

```
FOR contador:=1 TO 5 BY 1 DO
Var1:=Var1*2;
END_FOR;
Erg:=Var1;
```

Fig. 2-37: Bucle FOR en ST

Supongamos que en el ejemplo mostrado en **Fig. 2-37**, el ajuste predeterminado para la variable Var1 es el valor 1. En ese caso, después del bucle FOR tendrá el valor 32.

Nota: El <VALOR_FINAL> no debe ser el valor límite del contador <INT_VAR>. P. ej. si la variable Contador es del tipo SINT, el <VALOR_FINAL> no puede ser 127, ya que de lo contrario se obtendrá un bucle infinito.

El bucle FOR está controlado en comparación con los bucles WHILE o REPEAT, dado que el número de repeticiones se fija mediante un contador. Si se conoce el número de ciclos del bucle, es preferible un bucle FOR, dado que éste no permite bucles infinitos.

Bucle WHILE

El bucle WHILE puede utilizarse como el bucle FOR, con la diferencia de que la condición de interrupción puede ser cualquier expresión booleana. Esto significa que se especifica una condición que, al cumplirse, iniciará la ejecución del bucle.

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente


```
WHILE <Expresi≤n booleana>
DO
<Instrucciones>
END_WHILE;
```

Fig. 2-38: Sintaxis de la instrucción WHILE en ST

Las <instrucciones> se ejecutan repetidamente mientras la <expresión booleana> resulte ser TRUE. Si la <expresión booleana> arroja FALSE ya en la primera evaluación, no se ejecutan nunca las <instrucciones>. Si la <expresión booleana> no adopta nunca el valor FALSE, se ejecutan las <instrucciones> en un bucle infinito, dando lugar a un error de tiempo de ejecución.

Nota: El programador debe asegurarse de que no se produzca un bucle infinito. Para ello, debe modificar la condición en la parte de instrucción del bucle, por ejemplo incrementando o descontando un contador.

```
WHILE contador<>0 DO
Var1:=Var1*2;
Contador := Contador-1;
END_WHILE
```

Fig. 2-39: Bucle WHILE en ST

Bucle REPEAT

El bucle REPEAT se diferencia de los bucles WHILE por el hecho de que la condición de interrupción no se comprueba hasta después de haberse ejecutado el bucle. En consecuencia, el bucle se ejecutará al menos una vez, independientemente de la condición de interrupción.

```
REPEAT
<Instrucciones>
UNTIL <Expresi≤n booleana>
END_REPEAT;
```

Fig. 2-40: Sintaxis de la instrucción REPEAT en ST

Las <instrucciones> se ejecutan hasta que la <expresión booleana> resulte ser TRUE.

Si la <expresión booleana> resulta ser TRUE ya en la primera evaluación, se ejecutan exactamente una vez las <instrucciones>. Si la <expresión booleana> no adopta nunca el valor TRUE, se ejecutan las <instrucciones> en un bucle infinito, dando lugar a un error de tiempo de ejecución.

Nota: El programador debe asegurarse de que no se produzca un bucle infinito. Para ello, debe modificar la condición en la parte de instrucción del bucle, por ejemplo incrementando o descontando un contador.

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente


```
REPEAT
Var1:=Var1*2;
Contador := Contador-1;
UNTIL
Contador=0
END_REPEAT
```

Fig. 2-41: Bucle REPEAT en ST

Instrucción EXIT

Si la instrucción EXIT aparece en un bucle FOR, WHILE o REPEAT, se termina el bucle más interno, independientemente de la condición de interrupción.

Lenguaje de proceso (AS)

El lenguaje de proceso es un lenguaje orientado gráficamente que permite describir la secuencia cronológica de diversas acciones dentro de un programa. Para ello, se utilizan elementos de paso a los cuales se asignan determinadas acciones, y cuya secuencia de procesamiento es controlada por elementos de transición.

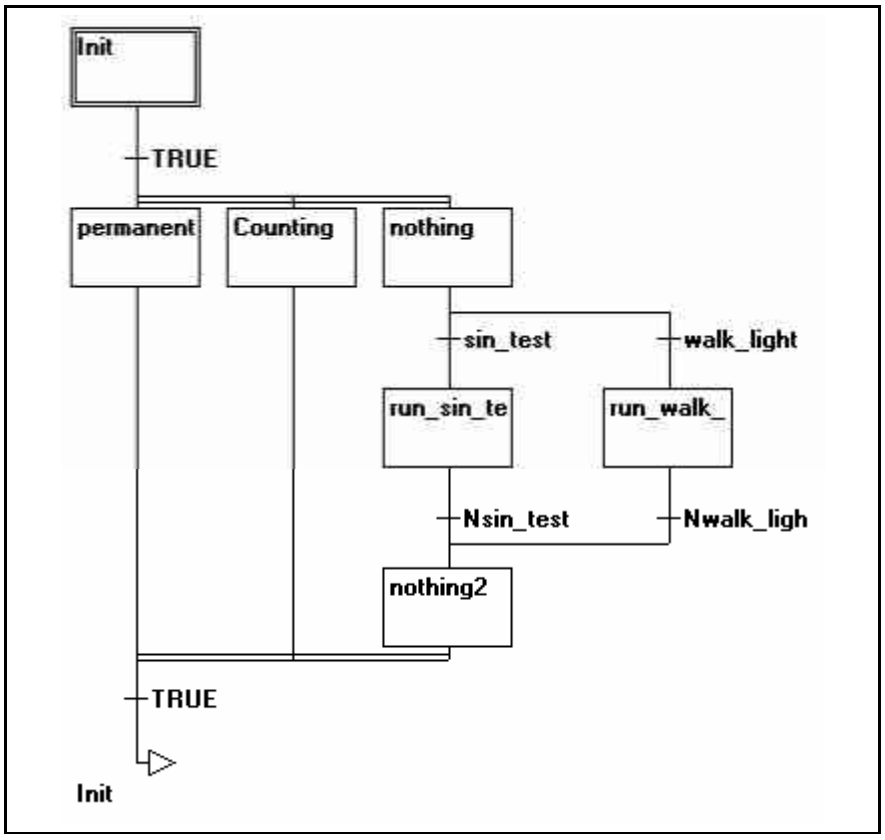

Fig. 2-42: Una red en el lenguaje de proceso (AS)

Paso

Un componente escrito en lenguaje de proceso consta de una secuencia de pasos conectados entre sí mediante conexiones dirigidas (transiciones).

Existen dos tipos de pasos:

- La forma simplificada consta de una acción y un flag (indicador), que indica si el paso está activo. Si se implementa la acción de un paso, aparece un pequeño triángulo en la esquina superior derecha de la casilla del paso.

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente

- Un paso IEC consta de un flag y de una o varias acciones o variables booleanas asignadas. Las acciones asociadas aparecen a la derecha del paso.

Acción

Una acción puede contener una serie de instrucciones en AWL o en ST, una cierta cantidad de redes en FUP o en KOP, o bien una estructura secuencial (AS).

En los pasos simplificados, una acción siempre está conectada a su paso. Para editar una acción, haga doble clic con el botón del ratón sobre el paso al que pertenece la acción, o seleccione el paso y ejecute la orden de menú "Extras" "Zoom acción/transición". Además, son posibles una acción de entrada y/o de salida por paso.

En el Object Organizer, las acciones de pasos IEC cuelgan directamente de su componente AS, y se cargan mediante doble clic o bien pulsando <Intro> en su editor. Se pueden crear nuevas acciones mediante "Proyecto" "Añadir acción". A un paso IEC se le pueden añadir un máximo de nueve acciones.

Acción de entrada y de salida

Además de la acción de paso, se pueden añadir a un paso una acción de entrada y una acción de salida. Una acción de entrada se ejecuta una sola vez, inmediatamente después de la activación del paso. Una acción de salida se ejecuta una sola vez, antes de que el paso sea desactivado.

Un paso con acción de entrada se identifica mediante una "E" en la esquina inferior izquierda, y la acción de salida mediante una "X" en la esquina inferior derecha.

Las acciones de entrada y de salida pueden implementarse en cualquier lenguaje. Para editar una acción de entrada o de salida, ejecute un doble clic con el ratón sobre la esquina correspondiente en el paso.

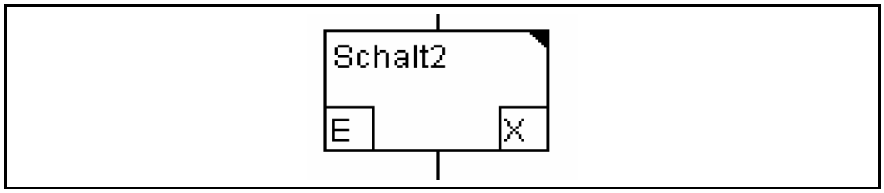

Fig. 2-43: Ejemplo de un paso con acción de entrada y de salida

Transición / Condición de transición

Entre los pasos se encuentran las denominadas transiciones.

Una condición de transición debe tener el valor TRUE o FALSE. Así pues, puede constar de una variable booleana, una dirección booleana o una constante booleana. También puede contener una secuencia de instrucciones con un resultado booleano en sintaxis ST (p. ej. (i <= 100) AND b) o en un lenguaje cualquiera (ver "Extras" "Zoom acción/transición" en la página 5-52). ¡Pero una transición no debe contener programas, bloques de función ni asignaciones!

Para el análisis de expresiones de transición se puede definir el flag SFCErrorAnalyzationTable.

En el Editor AS se puede escribir una condición de transición directamente en la marca de transición, o se puede abrir una ventana del editor propia para ello. ¡La condición presente en el Editor (ver "Extras" "Zoom acción/transición" en la página 5-52) tiene preferencia!

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente

Nota: Además de transiciones, también se puede utilizar el modo pulsatorio para saltar al siguiente paso, ver SFCtip y SFCtipmode.

Paso activo

Tras la llamada del componente AS, en primer lugar se ejecuta la acción correspondiente al paso inicial (rodeado de un borde doble). Un paso cuya acción se está ejecutando se considera "activo". En el modo Online, los pasos activos se muestran en azul.

En un ciclo se ejecutan primer todas las acciones que pertenecen a pasos activos. A continuación, los pasos que siguen a estos pasos se vuelven "activos" si las condiciones de transición para estos pasos siguientes son TRUE. Entonces los pasos ahora activos se ejecutan en el siguiente ciclo.

Nota: Si el paso activo contiene una acción de salida activa, también ésta se ejecuta primero en el siguiente ciclo, siempre y cuando la transición siguiente sea TRUE.

Paso IEC

Además de los pasos simplificados, están disponibles los pasos IEC estándar en AS.

Para poder utilizar pasos IEC, debe integrar en su proyecto la biblioteca SFC especial **Iecsfc.lib**.

A un paso IEC se le pueden asignar un máximo de nueve acciones. Las acciones IEC no están asignadas de forma fija a un paso como acción de entrada, de paso o de salida como en los pasos simplificados, sino que se encuentran separadas de los pasos y pueden utilizarse varias veces dentro de su componente. Para ello deben ser asociadas a los pasos deseados mediante la orden **"Extras" "Asociar acción"**.

Además de acciones, también se pueden asignar a los pasos variables booleanas.

La activación y desactivación de las acciones y las variables booleanas puede controlarse mediante las denominadas marcas de destino (calificador). Es posible que se produzcan retardos de tiempo. Dado que una acción todavía puede estar activa cuando ya se está procesando el siguiente paso, p. ej. mediante la marca de destino S (Set) se pueden obtener procesos concurrentes.

Una variable booleana asociada se ajusta o se repone con cada llamada del componente AS. Esto significa que cada vez se le asigna de nuevo el valor TRUE o FALSE.

Las acciones asociadas de un paso IEC se muestran a la derecha del paso en una casilla partida en dos. El campo de la izquierda contiene el calificador, posiblemente con constantes de tiempo, y el derecho el nombre de la acción o de la variable booleana.

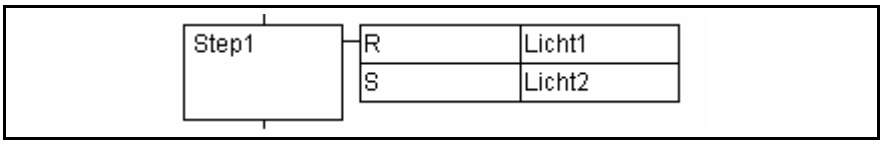

Fig. 2-44: Ejemplo de un paso IEC con dos acciones

Para un seguimiento más fácil de los procesos, todas las acciones activas en el modo Online se muestran en azul, al igual que los pasos activos. Después de cada ciclo se comprueba qué acciones están activas.

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente

¡A este respecto, tenga en cuenta también la limitación a la hora de utilizar calificadores de tiempo en acciones utilizadas varias veces en un mismo ciclo (ver más abajo en: "Calificador")!

Nota: Si se desactiva una acción, ésta se ejecuta **una vez más**. Esto significa que cada acción se ejecuta como mínimo dos veces (también una acción con el calificador P).

En caso de llamada, se procesan primero en orden alfabético las acciones desactivadas, y a continuación todas las acciones activas, también en orden alfabético.

El que un paso recientemente insertado sea un paso IEC depende de si se ha seleccionado la orden de menú **"Extras" "Utilizar pasos IEC"**.

En el Object Organizer, las acciones cuelgan directamente de su componente AS correspondiente. Se pueden crear nuevas acciones mediante "Proyecto" "Añadir acción".

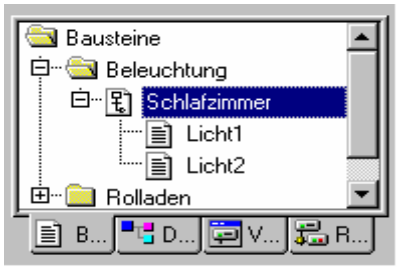

Fig. 2-45: Componente AS con acciones en el Object Organizer

Calificador

Para asociar las acciones a pasos IEC están disponibles los siguientes calificadores (marcas de destino).

N	Non-stored	la acción permanece activa tanto tiempo como el paso
R	overriding Reset	se desactiva la acción
S	Set (Stored)	se activa la acción y permanece activa hasta un Reset
L	time Limited	se activa la acción para un tiempo determinado, como máximo mientras el paso esté activo
D	time Delayed	la acción se activa al cabo de un tiempo determinado, siempre y cuando el paso todavía esté activo, y permanece activa mientras el paso esté activo
P	Pulse	la acción se ejecuta exactamente una vez al activarse el paso
SD	Stored and time Delayed	se activa la acción al cabo de un tiempo determinado y permanece activa hasta un Reset
DS	Delayed and Stored	se activa la acción al cabo de un tiempo determinado, siempre y cuando el paso todavía esté activo, y permanece activa hasta un Reset
SL	Stored and time Limited	la acción está activada durante un tiempo determinado

Fig. 2-46: Calificador (marca de destino)

Las marca de destino L, D, SD, DS y SL requieren una indicación de tiempo en el formato de constante TIME, p. ej. L T#5s.

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente

Nota: Si se desactiva una acción, ésta se ejecuta **una vez más**. Esto significa que cada acción se ejecuta como mínimo dos veces (también una acción con el calificador P).

Nota: En caso de llamada, se procesan primero en orden alfabético las acciones desactivadas, y a continuación todas las acciones activas, también en orden alfabético.

Nota: Si se utiliza la misma acción en dos pasos consecutivos con calificadores que influyen en la secuencia cronológica, en la segunda utilización el calificador de tiempo ya no puede hacerse efectivo. Para evitarlo, es preciso insertar un paso intermedio, de modo que en el ciclo que entonces se ejecutará adicionalmente pueda inicializarse de nuevo el estado de acción.

Variables implícitas en AS

En el lenguaje AS existen variables declaradas implícitamente que pueden utilizarse.

A cada paso pertenece un flag que guarda el estado del paso. El flag de paso (estado activo o inactivo del paso) se llama **<StepName>.x** en los pasos IEC o **<StepName>** en los pasos simplificados. Esta variable booleana tiene el valor TRUE cuando el paso correspondiente está activo y FALSE cuando está inactivo. Puede utilizarse en cada acción y transición del componente AS.

Mediante las variables **<AktionsName>.x** puede consultarse si una acción IEC está o no activa.

En los pasos IEC puede consultarse mediante las variables implícitas **<StepName>.t** la duración activa de los pasos.

También desde otros programas se puede acceder a las variables implícitas. Ejemplo: boolvar1:=sfc.step1.x; donde step1.x es la variable booleana implícita que expresa el estado del paso IEC step1 en el componente sfc1.

Flags AS

Para el control del proceso en el lenguaje de proceso, se pueden utilizar flags que se crean automáticamente durante la ejecución del proyecto. Para ello se deben declarar las variables correspondientes global o localmente, ya sea como variable de salida o de entrada. Ejemplo: Si en AS un paso permanece activo durante más tiempo del indicado en sus atributos, se coloca un flag al que se accede mediante una variable con el nombre SFCError (SFCError pasa a ser TRUE). Son posibles las siguientes variables de flag:

- **SFCEnableLimit**: Esta variable especial es del tipo BOOL. Si su valor es TRUE, se registran en SFCError los casos de tiempo excedido en los pasos. De lo contrario, se ignoran los casos de tiempo excedido. Su utilización puede ser útil, por ejemplo, en la puesta en servicio o en el funcionamiento manual.
- **SFCInit**: Si esta variable booleana tiene el valor TRUE, el lenguaje de proceso se devuelve al paso Init. También se reponen los demás flags AS (inicialización). Mientras la variable tenga el valor TRUE, el paso Init permanece ajustado (activo), pero no se ejecuta. Sólo cuando SFCInit vuelve a ser ajustado a FALSE se puede seguir procesando normalmente el componente.

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente

- **SFCReset**: Esta variable del tipo BOOL se comporta de forma similar a SFCInit. Sin embargo, a diferencia de ésta, el paso Init sigue siendo procesado tras la inicialización. Así, por ejemplo, en el paso Init sería posible ajustar rápidamente de nuevo a FALSE el flag SFCReset.
- **SFCQuitError**: Mientras esta variable booleana tenga el valor TRUE, se detiene la ejecución del diagrama secuencial de funciones, mientras que se repone una eventual superación del tiempo en la variable SFCError. Al ajustarse la variable de nuevo a FALSE, se reponen todos los tiempos previos en los pasos activos. La condición para ello es la declaración del flag SFCError, el cual registra la superación del tiempo.
- **SFCPause**: Mientras esta variable booleana tenga el valor TRUE, se detiene la ejecución del diagrama secuencial de funciones.
- **SFCError**: Esta variable booleana adopta el valor TRUE si se ha producido una superación del tiempo en un diagrama secuencial de funciones. Si en el programa se produce una nueva superación del tiempo después de la primera, esta segunda superación ya no se registra si previamente no se ha repuesto la variable SFCError. La declaración de SFCError es condición para el funcionamiento de las demás variables flag para el control de la secuencia cronológica (SFCErrorStep, SFCErrorPOU, SFCQuitError, SFCErrorAnalyzationTable).
- **SFCTrans**: Esta variable booleana adopta el valor TRUE cuando se activa una transición.
- **SFCErrorStep**: Esta variable es del tipo STRING. Si mediante SFCError se registra una superación del tiempo en el diagrama secuencial de funciones, en esta variable se guarda el nombre del paso que ha provocado la superación del tiempo. La condición para ello es la declaración de la variables SFCError, la cual registra la superación del tiempo.
- **SFCErrorPOU**: En caso de superación del tiempo, esta variable del tipo STRING adopta el nombre del componente en el que se ha producido la superación del tiempo. La condición para ello es la declaración de la variables SFCError, la cual registra la superación del tiempo.
- **SFCCurrentStep**: Esta variable es del tipo STRING. Es esta variable se guarda el nombre del paso que está activo, independientemente de la vigilancia del tiempo. En caso de ramificación paralela, se guarda el paso situado en la rama derecha más exterior.
- **SFCErrorAnalyzationTable**: Esta variable del tipo ARRAY [0..n] OF ExpressionResult proporciona, para cada variable de una expresión de transición agrupada que conduce a un FALSE de la transición y con ello a una superación del tiempo en el paso precedente, los siguientes datos: nombre, dirección, comentario, valor actual.
  Esto es posible para un máximo de 16 variables, esto es, la gama de array es de máx. 0..15.
  La estructura ExpressionResult, así como los módulos de análisis utilizados implícitamente, están incluidos en la biblioteca AnalyzationNew.lib. Los componentes de análisis también pueden ser utilizados explícitamente en componentes que no estén programados en SFC.
  La condición para el análisis de la expresión de transición es el registro de una superación del tiempo en el paso precedente. Por lo tanto, allí debe estar implementada una vigilancia del tiempo, y la variable SFCError (ver arriba) debe estar declarada en el componente.
- **SFCTip, SFCTipMode**: Estas variables del tipo BOOL permiten el funcionamiento pulsatorio del SFC. Si éste está activado medainte SFCTipMode=TRUE, sólo se puede pasar al siguiente paso ajustando

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente

SFCTip a TRUE. Mientras SFCTipMode permanezca ajustado en FALSE, es posible saltarse incluso las transiciones.

Rama alternativa

Dos o más ramas en AS pueden definirse como ramas alternativas. Cada rama alternativa debe empezar y terminar con una transición. Las ramas alternativas pueden contener ramas paralelas y otras ramas alternativas. Una rama alternativa empieza en una línea horizontal (principio alternativo) y termina en una línea horizontal (final alternativo) o con un salto.

Si el paso que precede a la línea de principio alternativo está activo, se evalúa de izquierda a derecha la primera transición de cada rama alternativa. Se abre la primera transición desde la izquierda cuya condición de transición tenga el valor TRUE, y se activan los pasos siguientes (ver el paso activo).

Rama paralela

Dos o más ramas en AS pueden definirse como ramas paralelas. Cada rama paralela debe empezar y terminar con un paso. Las ramas paralelas pueden contener ramas alternativas u otras ramas paralelas. Una rama paralela empieza en una línea doble (principio paralelo) y termina en una línea doble (final paralelo) o en un salto. Es posible dotarla de una marca de salto.

Si el paso que precede a la línea de principio paralelo está activo y la condición de transición tras este paso tiene el valor TRUE, se activan los primeros pasos de todas las ramas paralelas (ver el paso activo). Todas estas ramas se procesan entonces en paralelo. El paso detrás de la línea de final paralelo se activa cuando todos los pasos previos están activos y la condición de transición delante de este paso arroja el valor TRUE.

Salto

Un salto es una conexión al paso cuyo nombre está indicado bajo el símbolo de salto. Los saltos son necesarios porque no está permitido crear conexiones que conduzcan hacia arriba o se entrecrucen.

Esquema de funciones (FUP)

El esquema de funciones es un lenguaje de programación orientado gráficamente. Trabaja con una lista de redes, cada una de las cuales contiene una estructura que representa una expresión lógica o aritmética, la llamada a un bloque de función, un salto o una instrucción de retorno.

En el editor de esquemas de funciones continuo no se utilizan redes.

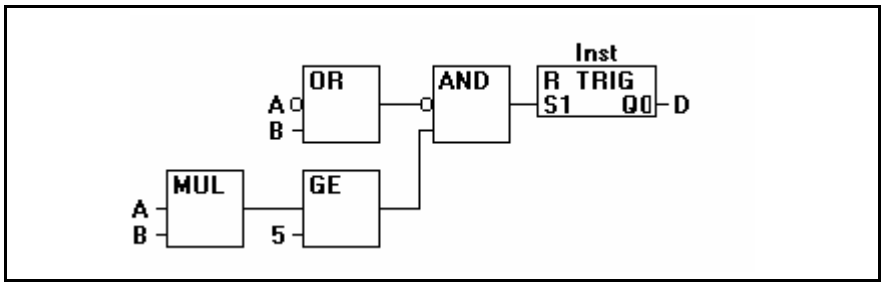

Fig. 2-47: Red en el esquema de funciones (FUP)

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente

El editor de esquemas de funciones continuo (CFC)

A diferencia del esquema de funciones FUP, el editor de esquemas de funciones continuo no trabaja con redes, sino con elementos que pueden colocarse libremente. Esto permite feedback, por ejemplo.

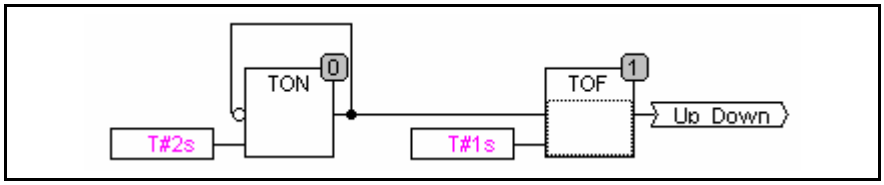

Fig. 2-48: Ejemplo de una implementación en el editor de esquemas de funciones continuo (CFC)

Esquema de contactos (KOP)

El esquema de contactos es un lenguaje de programación orientado gráficamente, que se asemeja a la estructura de un circuito eléctrico.

Por una parte, el esquema de contactos está indicado para construir interruptores lógicos, pero por otra parte también permite crear redes como en el FUP. Por lo tanto, el KOP es muy útil para controlar la llamada de otros componentes. El esquema de contactos consta de una serie de redes. Una red está limitada a los lados izquierdo y derecho por sendas líneas de corriente vertical a izquierda y derecha. Entre ambas se encuentra un esquema de circuitos compuesto por contactos, bobinas y líneas de conexión.

Cada red está formada en el lado izquierdo por una serie de contactos que transmiten de izquierda a derecha el estado "ON" u "OFF", estados que se corresponden con los valores booleanos TRUE y FALSE. A cada

contacto le pertenece una variable booleana. Si esta variable tiene el valor TRUE, se transmite el estado de izquierda a derecha por la línea de conexión; de lo contrario la conexión derecha adopta el valor OFF.

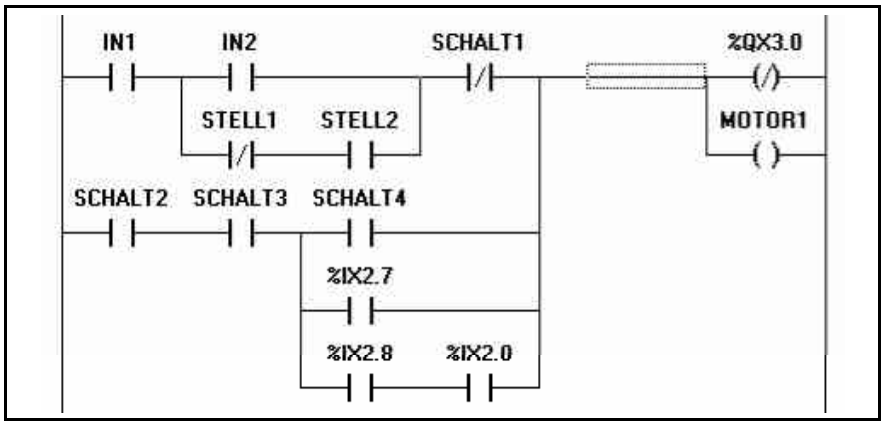

Fig. 2-49: Red de contactos y bobinas en el esquema de contactos (KOP)

Contacto

Cada red en el KOP está formada en el lado izquierdo por una red de contactos (los contactos se representan mediante dos líneas paralelas: I I), que transmiten de izquierda a derecha el estado "On" u "Off".

Estos estados se corresponden con los valores booleanos TRUE y FALSE. A cada contacto le pertenece una variable booleana. Si esta variable tiene el valor TRUE, se transmite el estado de izquierda a derecha por la línea de conexión; de lo contrario la conexión derecha adopta el valor "Off".

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente

Los contactos pueden estar conectados en paralelo, en cuyo caso *una* de las ramas paralelas debe transmitir el valor "On" para que la rama paralela transmita el valor "On", o bien los contactos están conectados en serie, y entonces *todos* los contactos deben transmitir el estado "On" para que el último contacto transmita el estado "On". Así pues, esto es análogo a un circuito eléctrico en paralelo o en serie.

Un contacto también puede estar negado, circunstancia reconocible por la barra oblicua en el símbolo de contacto: **I/I**. Entonces se transmite el valor de la línea si la variable es FALSE.

Bobina

En el lado derecho de una red en el KOP puede encontrarse cualquier cantidad de las denominadas bobinas, representadas por paréntesis:**( )**. Éstas sólo pueden conectarse en paralelo. Una bobina transmite el valor de las conexiones de izquierda a derecha, y lo copia en una variable booleana correspondiente. En la línea de entrada puede estar presente el valor ON (se corresponde con la variable booleana TRUE) o el valor OFF (se corresponde con FALSE).

Los contactos y las bobinas también pueden ser negados. Cuando una bobina está negada (circunstancia reconocible por la barra oblicua en el símbolo de bobina: **(/)**), copia el valor negado en la variable booleana correspondiente. Si un contacto está negado, sólo conecta si la variable booleana correspondiente es FALSE.

Bloques de función en el esquema de contactos

Además de contactos y bobinas, también puede introducir bloques de función y programas, los cuales deben tener en la red una entrada y una salida con valores booleanos y pueden utilizarse en los mismos lugares que los contactos, esto es, en el lado izquierdo de la red KOP.

Bobinas Set/Reset

Las bobinas también pueden estar definidas como bobinas Set o Reset. Una bobina Set (reconocible por la "S" en el símbolo de bobina: **(S)**) nunca sobrescribe el valor TRUE en la variable booleana correspondiente. Esto significa que una vez que se ha ajustado la variable a TRUE, ésta permanece así.

Una bobina Reset (reconocible por la "R" en el símbolo de bobina: **(R)**) nunca sobrescribe el valor FALSE en la variable booleana correspondiente. Una vez que se ha ajustado la variable a FALSE, ésta permanece así.

KOP como FUP

Al trabajar con el KOP puede darse fácilmente el caso de que desee utilizar el resultado del interruptor de contacto para el control de otros componentes. De este modo, por una lado puede guardar el resultado por medio de las bobinas en una variable global que se seguirá utilizando en otro lugar. Pero también puede insertar la posible llamada directamente en su red KOP. Para ello, introduce un componente con entrada EN.

Tales componentes son operandos, funciones, programas o bloques de función perfectamente normales, que cuentan con una entrada adicional identificada con EN. La entrada EN es siempre del tipo BOOL y su significado es: el componente con entrada EN se evalúa cuando EN tiene el valor TRUE.
Un componente EN se conecta en paralelo a las bobinas, de tal forma que la entrada EN se conecta a la línea de conexión entre los contactos y las bobinas. Si la información ON se transporta por esta línea, este componente se evaluará de forma totalmente normal.
Partiendo de dicho componente EN, se pueden crear redes como en FUP.

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente

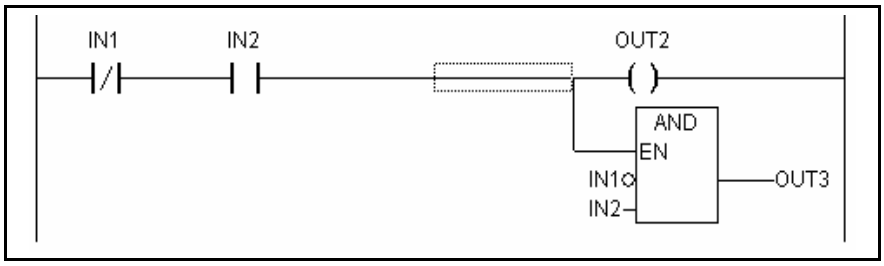

Fig. 2-50: Red KOP con un componente EN.

2.3 Depuración, funciones online

Depuración

Las funciones de depuración de IndraLogic le facilitan la localización de errores.

Para poder depurar, es preciso ejecutar la orden **"Proyecto" "Opciones"** y, en el diálogo que aparece, seleccionar la opción **Depuración** en **Opciones de traducción**.

Breakpoint

Un breakpoint es un punto en el programa en el que se interrumpe la ejecución. De este modo resulta posible observar los valores de variables en un punto determinado del programa.

Los breakpoints pueden colocarse en todos los editores. En los editores de texto, los breakpoints se colocan en números de línea; en FUP y KOP en números de red; en CFC en componentes y en AS en pasos.

Nota: No se pueden colocar breakpoints en instancias de bloques de función.

Paso individual

Paso individual significa:

- en AWL: ejecutar el programa hasta la siguiente orden CAL, LD o JMP.
- en ST: ejecutar la siguiente instrucción.
- en FUP, KOP: ejecutar la siguiente red.
- en AS: ejecutar la acción hasta el siguiente paso.
- en CFC: ejecutar el siguiente componente (casilla) en el programa CFC.

Mediante la ejecución paso a paso puede comprobar la corrección lógica de su programa.

Ciclo individual

Si se ha seleccionado Ciclo individual, se detiene la ejecución después de cada ciclo.

Modificar valores online

Durante el funcionamiento, las variables pueden ser ajustadas una vez a un valor determinado (escribir valor) o bien ser descritas de nuevo con un valor determinado después de cada ciclo (forzar valor). También puede modificar el valor de la variable en el modo Online, haciendo doble clic sobre el valor. De este modo, las variables booleanas cambian de TRUE

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente

a FALSE o viceversa, mientras que para todas las demás se le mostrará un diálogo **Escribir variable xy**, en el que puede editar el valor actual de la variable.

Monitorización

En el modo Online, para todas las variables visibles en pantalla se leen del control los valores actuales y se representan continuamente. Esta representación se encuentra en el editor de declaraciones y programas, y además puede leer valores de variables actuales en el Administrador watch y de fórmulas y en una visualización. Si se deben monitorizar variables de instancias de bloque de función, primero se debe abrir la instancia correspondiente. Al monitorizar variables VAR_IN_OUT, se da salida al valor desreferenciado.

Al monitorizar pointers, en la parte de declaración se da salida tanto al pointer como al valor desreferenciado. En la parte de programa se da salida únicamente al pointer:

```
+ --pointervar = '<'pointervalue'>'
```

Fig. 2-51: Monitorización de pointers

Los POINTERS en el valor desreferenciado también se muestran en consecuencia. Mediante un simple clic sobre la cruz o un doble clic sobre la línea se expande o se colapsa la visualización.

En las implementaciones se indica el valor del pointer. Sin embargo, para desreferenciaciones se indica el valor desreferenciado.

Monitorización de componentes ARRAY: Además de los componentes array, indexados por una constante, también se muestran componentes indexados por una variable:

```
anarray[1] = 5
anarray[i] = 1
```

Fig. 2-52: Monitorización de campos (ARRAY)

Si el índice consiste en una expresión (p. ej. [i+j] o [i+1]), no se puede mostrar el componente.

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente

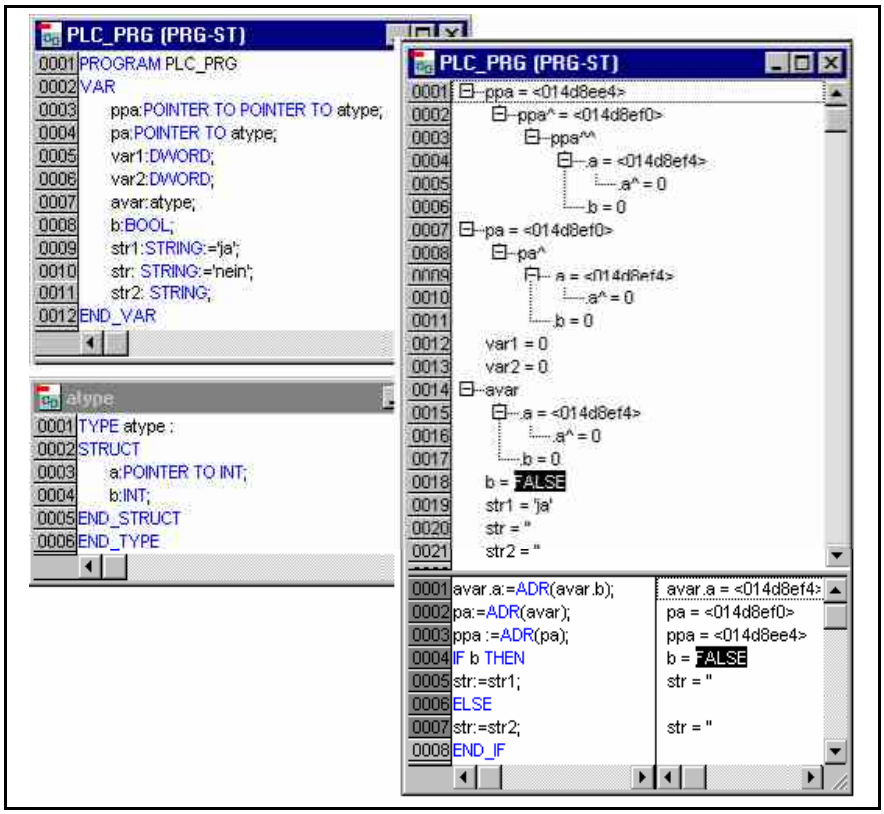

Fig. 2-53: Monitorización de pointers

Nota: Si se ha alcanzado el número máximo de variables que pueden ser monitorizadas, para cada variable adicional se muestra, en lugar del valor actual, el texto "Demasiadas variables de monitorización".

Simulación

En la simulación, el programa de control creado no se procesa en el control, sino en el ordenador en el que se ejecuta IndraLogic. Están disponibles todas las funciones online. Esto le ofrece la posibilidad de comprobar la corrección lógica de su programa sin necesidad de hardware de control.

Nota: Los componentes de bibliotecas externas no se ejecutan en la simulación.

Registro

El registro registra cronológicamente acciones del usuario, procesos internos, cambios de estado y situaciones excepcionales durante el modo Online. Sirve para la vigilancia y para la trazabilidad de errores.

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com

RODAVIGO, S.A.
RODAMIENTOS VIGO, S.A.
www.rodavigo.net
+34 986 288118
Servicio de Att. al Cliente

2.4 La norma

La norma IEC 61131-3 es un estándar internacional para lenguajes de programación de controles lógicos programables.

Los lenguajes de programación realizados en IndraLogic cumplen los requisitos de la norma.

Conforme a este estándar, un programa consta de los siguientes elementos:

- Estructuras
- Componentes
- Variables globales

Los elementos del lenguaje generales se describen en las secciones Identificador, Direcciones, Tipos, Comentarios y Constantes.

El procesamiento de un programa IndraLogic empieza por el componente especial PLC_PRG. El componente PLC_PRG puede llamar otros componentes.

Polígono Indutrial O Rebullón s/n. 36416 - Mos - España - rodavigo@rodavigo.com