10 Apéndice A: Operadores IEC y funciones adicionales que amplían la norma

Visión general

IndraLogic permite todos los operadores IEC. Al contrario que las funciones estándar (biblioteca estándar), éstos son conocidos implícitamente en todo el proyecto. Además de los operadores IEC, **IndraLogic** soporta los siguientes operadores no exigidos por la norma: INDEXOF y SIZEOF (ver Operadores aritméticos), ADR y BITADR (ver Operadores de dirección).

Nota: En operaciones con tipos de datos de coma flotante, el resultado del cálculo depende del hardware de sistema de destino utilizado.

En las implementaciones de componentes, los operadores se utilizan como funciones.

- Operadores aritméticos
- Operadores bitstring (de cadena de bits)
- Operadores bit-shift (de desplazamiento de bits)
- Operadores de selección
- Operadores de comparación
- Operadores de dirección
- Operador de llamada
- Conversiones de tipo
- Operadores numéricos

10.1 Operadores aritméticos

ADD

Suma de variables de los tipos BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL y LREAL.

También se pueden sumar dos variables TIME, en ese caso la suma es también un tiempo (p. ej. rige t#45s + t#50s = t#1m35s)

```
LD 7
ADD 2,4,7
ST  Var1
```

Fig. 10-1: Ejemplo de ADD en AWL

```
var1 := 7+2+4+7;
```

Fig. 10-2: Ejemplo de ADD en ST

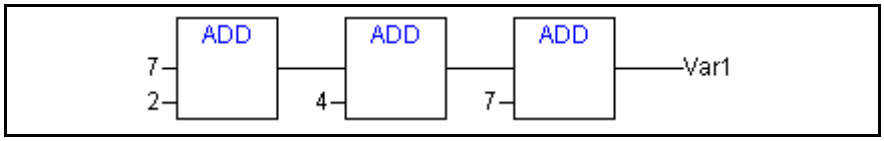


Fig. 10-3: Ejemplo de ADD en FUP

MUL

Multiplicación de variables de los tipos BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL y LREAL.

```
LD 7
MUL 2,4,7
ST  Var1
```

Fig. 10-4: Ejemplo de MUL en AWL

```
var1 := 7*2*4*7;
```

Fig. 10-5: Ejemplo de MUL en ST

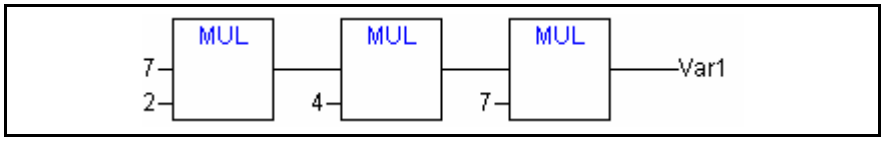


Fig. 10-6: Ejemplo de MUL en FUP

SUB

Resta de una variable del tipo BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL y LREAL de otra variable de uno de estos tipos.

Una variable TIME también puede restarse de otra variable TIME, en cuyo caso el resultado es también del tipo TIME. Tenga en cuenta que los valores TIME negativos no están definidos.

```
LD 7
SUB 2
ST  Var1
```

Fig. 10-7: Ejemplo de SUB en AWL

```
var1 := 7-2;
```

Fig. 10-8: Ejemplo de SUB en ST

Fig. 10-9: Ejemplo de SUB en FUP

DIV

División de una variable del tipo BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL y LREAL por otra variable de uno de estos tipos.

```
LD 8
DIV 2
ST  Var1 (* el resultado es 4 *)
```

Fig. 10-10: Ejemplo de DIV en AWL

```
var1 := 8/2;
```

Fig. 10-11: Ejemplo de DIV en ST

Fig. 10-12: Ejemplo de DIV en FUP

Nota: Si en su proyecto define funciones con los nombres **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. El nombre de la función es fijo y debe tener únicamente esta designación.

Nota: Tenga en cuenta que el comportamiento en caso de una división por 0 depende del sistema operativo y de destino utilizado.

```
FUNCTION CheckDivReal : REAL
VAR_INPUT
 divisor:REAL;
END_VAR

IF divisor = 0 THEN
 CheckDivReal:=1;
ELSE
 CheckDivReal:=divisor;
END_IF;
```

Fig. 10-13 : Ejemplo de implementación de la función CheckDivReal

El resultado de la función CheckDivReal es utilizado como divisor por el operador DIV. En el programa ejemplar reproducido a continuación, de este modo se evita la división por 0, ya que el divisor (d) se cambia de 0 a 1. Por lo tanto, el resultado res de la división es 799.

```
PROGRAM PLC_PRG
VAR
 res:REAL;
 v1:REAL:=799;
 d:REAL;
END_VAR

res:= v1/d;
```

Fig. 10-14 : Ejemplo de una división

Nota: ¡Las funciones CheckDiv contenidas en la CheckLib son soluciones ejemplares! Antes de utilizar la biblioteca, compruebe si las funciones trabajan tal como desea o implente una función CheckDiv apropiada directamente como componente en su proyecto.

MOD

División módulo de una variable del tipo BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT por otra variable de uno de estos tipos. Como resultado, esta función arroja el resto de la división en números enteros.

```
LD 9
MOD 2
ST  Var1 (* el resultado es 1 *)
```

Fig. 10-15 : Ejemplo de MOD en AWL

```
var1 := 9 MOD 2;
```

Fig. 10-16 : Ejemplo de MOD en ST

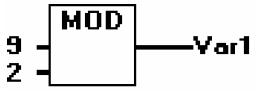


Fig. 10-17 : Ejemplo de MOD en FUP

MOVE

Asignación de una variable a otra variable de un tipo apropiado. Dado que MOVE está disponible como componente en los editores CFC y KOP, allí se puede aplicar la función EN/EN0 también a una asignación de variable. Por desgracia, esto no es posible en el editor FUP.

Ejemplo en CFC en combinación con la función EN/EN0:

Sólo si en_i es TRUE se asigna el valor de la variable var1 a la variable var2.

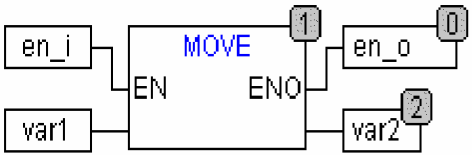


Fig. 10-18 : Ejemplo de MOVE en CFC

```
LD ivar1
MOVE
ST ivar2 (* Resultado: var2 recibe el valor de var1 *)

(* equivale a: *)
      LD ivar1
      ST ivar2
```

Fig. 10-19 : Ejemplo de MOVE en AWL

```
ivar2 := MOVE(ivar1);

(* equivale a: *)
      ivar2 := ivar1;
```

Fig. 10-20 : Ejemplo de MOVE en ST

INDEXOF

Esta función no está prescrita por la norma IEC61131-3.

Como resultado, INDEXOF arroja el indice interno de un componente.

```
var1 := INDEXOF(componente2);
```

Fig. 10-21 Ejemplo de INDEXOF en ST

SIZEOF

Esta función no está prescrita por la norma IEC61131-3.

Como resultado, SIZEOF arroja el número de bytes que necesita la variable especificada.

```
arr1:ARRAY[0..4] OF INT;
Var1 INT
LD arr1
SIZEOF
ST  Var1 (* el resultado es 10 *)
```

Fig. 10-22 : Ejemplo de SIZEOF en AWL

```
var1 := SIZEOF(arr1);
```

Fig. 10-23 : Ejemplo de SIZEOF en ST

10.2 Operadores bitstring (de cadena de bits)

AND

AND por bits de operandos de bits. Los operandos deben ser del tipo BOOL, BYTE, WORD o DWORD.

```
Var1 BYTE
LD 2#1001_0011
AND 2#1000_1010
ST Var1 (* el resultado es 2#1000_0010 *)
```

Fig. 10-24 : Ejemplo de AND en AWL

```
var1 := 2#1001_0011 AND 2#1000_1010
```

Fig. 10-25 : Ejemplo de AND en ST

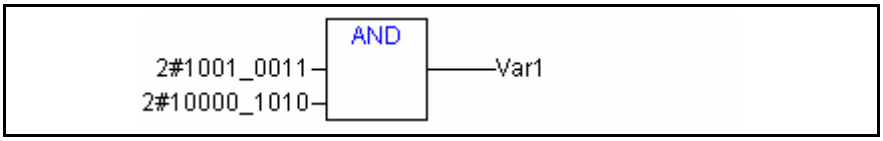
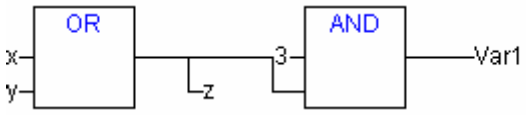


Fig. 10-26 : Ejemplo de AND en FUP

Nota: Si al utilizar generadores 68xxx o C-Code en FUP introduce un paso de programa como el aquí representado



debe tener en cuenta lo siguiente:
¡Debido al procedimiento de ejecución optimizado en el lenguaje FUP, deja de efectuarse la asignación del valor de la segunda variable de entrada en el componente de operador AND para la variable z, en cuanto la variable de entrada a tiene el valor FALSE!

OR

OR por bits de operandos de bits. Los operandos deben ser del tipo BOOL, BYTE, WORD o DWORD.

```
Var1 BYTE
LD 2#1001_0011
OR   2#1000_1010
ST   Var1    (* el resultado es 2#1001_1011 *)
```

Fig. 10-27 : Ejemplo de OR en AWL

```
Var1 := 2#1001_0011 OR 2#1000_1010
```

Fig. 10-28 : Ejemplo de OR en ST

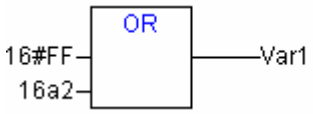
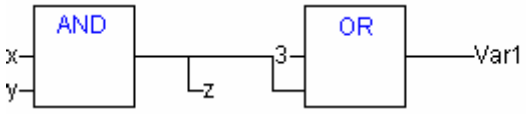


Fig. 10-29 : Ejemplo de OR en FUP

Nota: Si al utilizar generadores 68xxx o C-Code en FUP introduce un paso de programa como el aquí representado



debe tener en cuenta lo siguiente:
¡Debido al procedimiento de ejecución optimizado en el lenguaje FUP, deja de efectuarse la asignación del valor de la segunda variable de entrada en el componente de operador OR para la variable z, en cuanto la variable de entrada a tiene el valor TRUE!

XOR

XOR por bits de operandos de bits. Los operandos deben ser del tipo BOOL, BYTE, WORD o DWORD.

```
Var1 BYTE
LD 2#1001_0011
XOR  2#1000_1010
ST   Var1    (* el resultado es 2#0001_1001 *)
```

Fig. 10-30 : Ejemplo de XOR en AWL

```
Var1 := 2#1001_0011 XOR 2#1000_1010
```

Fig. 10-31 : Ejemplo de XOR en ST

Fig. 10-32 : Ejemplo de XOR en FUP

Nota: Tenga en cuenta el comportamiento del componente XOR en forma ampliada, esto es, en caso de más de 2 entradas: las entradas se comprueban por pares y después se comparan entre sí los resultados (cumple la norma, pero no necesariamente las expectativas del usuario).

NOT

NOT por bits de un operando de bits. El operando debe ser del tipo BOOL, BYTE, WORD o DWORD.

```
Var1 BYTE
LD 2#1001_0011
NOT
ST   Var1    (* el resultado es 2#0110_1100 *)
```

Fig. 10-33 : Ejemplo de NOT en AWL

```
Var1 := NOT 2#1001_0011
```

Fig. 10-34 : Ejemplo de NOT en ST

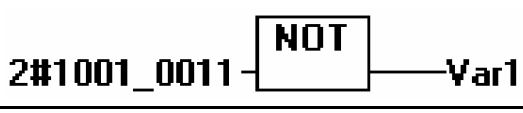


Fig. 10-35 : Ejemplo de NOT en FUP

10.3 Operadores bit-shift (de desplazamiento de bits)

SHL

Desplazamiento a la izquierda por bits de un operando: res:= SHL (in, n)

in se desplaza n bits hacia la izquierda y se llena con ceros desde la derecha.

Nota: Tenga en cuenta que el número de bits que se considera para la operación de cálculo es especificado por el tipo de dato de la variable de entrada in. Si se trata de una constante, se considera el tipo de datos más pequeño posible. El tipo de datos de la variable de salida no tiene consecuencias sobre la operación de cálculo.

En el siguiente ejemplo puede ver en representación hexadecimal cómo, con un valor idéntico de las variables de entrada in_byte e in_word, los resultados res_byte y res_word de la operación se diferencian, según si in es del tipo BYTE o WORD.

```
PROGRAM shl_st

VAR

      in_byte : BYTE:=16#45;
      in_word : WORD:=16#45;
      res_byte : BYTE;
      res_word : WORD;
      n: BYTE :=2;
END_VAR

res_byte:=SHL(in_byte,n); (* el resultado es 16#14 *)

res_word:=SHL(in_word;n); (* el resultado es 16#0114 *)
```

Fig. 10-36 : Ejemplo de SHL en ST

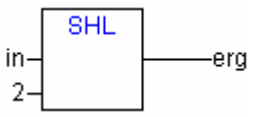


Fig. 10-37 : Ejemplo de SHL en FUP

```
LD  16#45
SHL 2
ST  res_byte
```

Fig. 10-38 : Ejemplo de SHL en AWL

SHR

Desplazamiento a la derecha por bits de un operando: res:= SHR (in, n)

in es desplazado n bits hacia la derecha. Si se utiliza un tipo de dato sin signo antepuesto (BYTE, WORD, DWORD), se llena de ceros desde la izquierda. En cambio, en caso de tipos de datos con signo antepuesto, como p. ej. INT, se efectúa un desplazamiento aritmético, esto es, se llena con el valor del bit más alto.

Nota: Tenga en cuenta que el número de bits que se considera para la operación de cálculo es especificado por el tipo de dato de la variable de entrada in. Si se trata de una constante, se considera el tipo de datos más pequeño posible. El tipo de datos de la variable de salida no tiene consecuencias sobre la operación de cálculo.

En el siguiente ejemplo se muestran en representación hexadecimal los resultados de la operación, donde una vez res_byte del tipo BYTE y una vez res_word del tipo WORD actúan como variables de entrada.

```
PROGRAM shr_st

VAR

      in_byte : BYTE:=16#45;
      in_word : WORD:=16#45;
      res_byte : BYTE;
      res_word : WORD;
      n: BYTE :=2;
END_VAR

res_byte:=SHR(in_byte,n); (* el resultado es 11 *)

res_word:=SHR(in_word;n); (* el resultado es 0011 *)
```

Fig. 10-39 : Ejemplo de SHR en ST

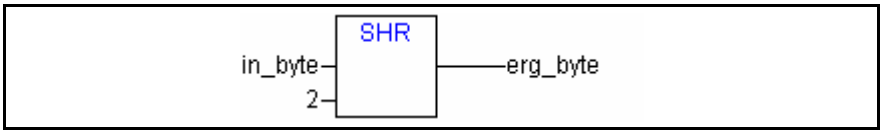


Fig. 10-40 : Ejemplo de SHR en FUP

```
LD  16#45
SHR 2
ST  res_byte
```

Fig. 10-41 : Ejemplo de SHR en AWL

ROL

Rotación a la izquierda por bits de un operando: res:= ROL (in, n)

res, in y n deberían ser del tipo BYTE, WORD o DWORD. in es desplazado n veces una posición de bit hacia la izquierda, mientras que el bit situado más a la izquierda es reinsertado desde la derecha.

Nota: Tenga en cuenta que el número de bits que se considera para la operación de cálculo es especificado por el tipo de dato de la variable de entrada in. Si se trata de una constante, se considera el tipo de datos más pequeño posible. El tipo de datos de la variable de salida no tiene consecuencias sobre la operación de cálculo.

En el siguiente ejemplo puede ver en representación hexadecimal cómo, con un valor idéntico de las variables de entrada in_byte e in_word, los resultados res_byte y res_word de la operación se diferencian, según si in es del tipo BYTE o WORD.

```
PROGRAM rol_st

VAR

      in_byte : BYTE:=16#45;
      in_word : WORD:=16#45;
      res_byte : BYTE;
      res_word : WORD;
      n: BYTE :=2;

END_VAR

res_byte:=ROL(in_byte,n); (* el resultado es 16#15 *)

res_word:=ROL(in_word;n); (* el resultado es 16#0114 *)
```

Fig. 10-42 : Ejemplo de ROL en ST

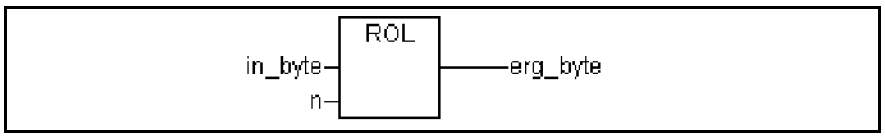


Fig. 10-43 : Ejemplo de ROL en FUP

```
LD  16#45
ROL 2
ST  res_byte
```

Fig. 10-44 : Ejemplo de ROL en AWL

ROR

Rotación a la derecha por bits de un operando: res:= ROR (in, n)

res, in y n deberían ser del tipo BYTE, WORD o DWORD. in es desplazado n veces una posición de bit hacia la derecha, mientras que el bit situado más a la derecha es reinsertado desde la izquierda.

Nota: Tenga en cuenta que el número de bits que se considera para la operación de cálculo es especificado por el tipo de dato de la variable de entrada in. Si se trata de una constante, se considera el tipo de datos más pequeño posible. El tipo de datos de la variable de salida no tiene consecuencias sobre la operación de cálculo.

En el siguiente ejemplo puede ver en representación hexadecimal cómo, con un valor idéntico de las variables de entrada in_byte e in_word, los resultados res_byte y res_word de la operación se diferencian, según si in es del tipo BYTE o WORD.

```
PROGRAM ror_st

VAR

      in_byte : BYTE:=16#45;
      in_word : WORD:=16#45;
      res_byte : BYTE;
      res_word : WORD;
      n: BYTE :=2;

END_VAR

res_byte:=ROR(in_byte,n); (* el resultado es 16#51 *)

res_word:=ROR(in_word;n); (* el resultado es 16#4011 *)
```

Fig. 10-45 : Ejemplo de ROR en ST

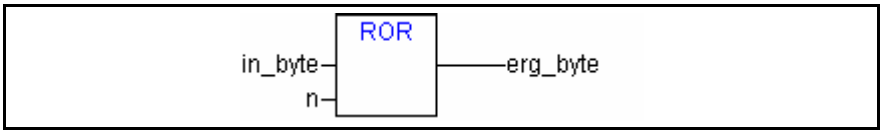


Fig. 10-46 : Ejemplo de ROR en FUP

```
LD  16#45
ROR 2
ST  res_byte
```

Fig. 10-47 : Ejemplo de ROR en AWL

10.4 Operadores de selección

Todas las operaciones de selección pueden realizarse también con variables. Por motivos de claridad, en los siguientes ejemplos nos limitamos a constantes como operadores.

SEL

Selección binaria.

`OUT := SEL(G, IN0, IN1)` significa:

```
OUT := IN0 if G=FALSE;
OUT := IN1 if G=TRUE.
```

IN0, IN1 y OUT pueden ser de cualquier tipo, mientras que G debe ser del tipo BOOL. El resultado de la selección es IN0 si G es FALSE, e IN1 si G es TRUE.

```
LD  TRUE
SEL 3,4   (* IN0 = 3, IN1 =4 *)
ST  Var1 (* el resultado es 4 *)
LD  FALSE
SEL 3,4
ST  Var1 (* el resultado es 3 *)
```

Fig. 10-48 : Ejemplo de SEL en AWL

```
Var1:=SEL(TRUE,3,4); (* el resultado para Var1 es 4 *)
```

Fig. 10-49 : Ejemplo de SEL en ST

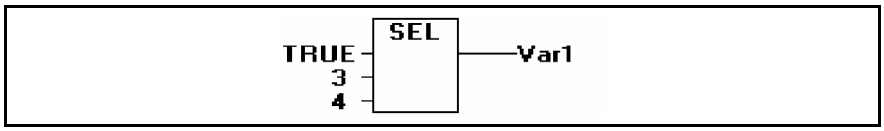


Fig. 10-50 : Ejemplo de SEL en FUP

Nota: Para fines de optimización del tiempo de ejecución, se procede de la siguiente forma: una expresión antepuesta a IN0 sólo se calcula si G es FALSE. ¡Una expresión antepuesta a IN1 sólo se calcula si G es TRUE!

En cambio, en la simulación se calculan todas las ramas.

MAX

Función de máximo. Arroja el mayor de dos valores.

```
OUT := MAX(IN0, IN1)
```

IN0, IN1 y OUT pueden ser de cualquier tipo.

```
LD 90
MAX 30
MAX 40
MAX 77
ST  Var1 (* el resultado es 90 *)
```

Fig. 10-51: Ejemplo de MAX en AWL

```
Var1:=MAX(30,40); (* el resultado es 40 *)
Var1:=MAX(40,MAX(90,30)); (* el resultado es 90 *)
```

Fig. 10-52 : Ejemplo de MAX en ST

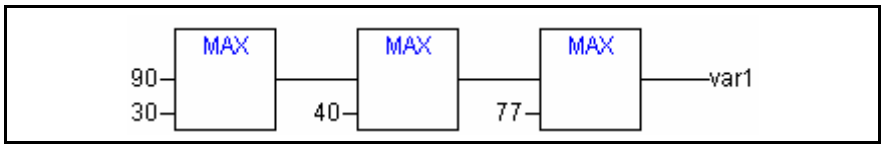


Fig. 10-53 : Ejemplo de MAX en FUP

MIN

Función de mínimo. Arroja el menor de dos valores.

```
OUT := MIN(IN0, IN1)
```

IN0, IN1 y OUT pueden ser de cualquier tipo.

```
LD 90
MIN 30
MIN 40
MIN 77
ST  Var1 (* el resultado es 30 *)
```

Fig. 10-54 : Ejemplo de MIN en AWL

```
Var1:=MIN(90,30); (* el resultado es 30 *);
Var1:=MIN(MIN(90,30),40); (* el resultado es 30 *);
```

Fig. 10-55 : Ejemplo de MIN en ST

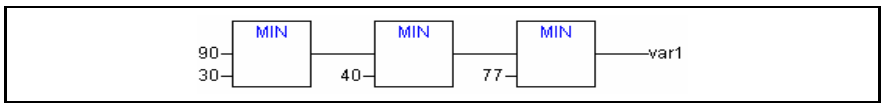


Fig. 10-56 : Ejemplo de MIN en FUP

LIMIT

Limitación

`OUT := LIMIT(Min, IN, Max)` significa:

`OUT := MIN (MAX (IN, Min), Max)`

Max es el límite superior y Min el límite inferior para el resultado. Si el valor IN supera el límite Max, LIMIT arroja Max. Si IN no alcanza Min, el resultado es Min.

IN y OUT pueden ser de cualquier tipo.

```
LD 90
LIMIT 30,80
ST  Var1 (* el resultado es 80 *)
```

Fig. 10-57 : Ejemplo de LIMIT en AWL

```
Var1:=LIMIT(30,90,80); (* el resultado es 80 *);
```

Fig. 10-58 : Ejemplo de LIMIT en ST

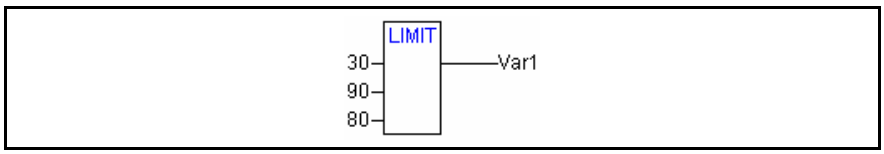


Fig. 10-59 : Ejemplo de LIMIT en FUP

MUX

Multiplexor

`OUT := MUX(K, IN0,...,INn)` significa:

`OUT := INK.`

IN0, ...,INn y OUT pueden ser de cualquier tipo. K debe ser de los tipos BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT o UDINT. MUX selecciona el valor que hace K de entre un conjunto de valores. El primer valor corresponde a K=0. Si K es mayor que el número de entradas siguientes (n) , se transmite el último valor (INn).

```
LD 0
MUX 30,40,50,60,70,80
ST  Var1 (* el resultado es 30 *)
```

Fig. 10-60 : Ejemplo de MUX en AWL

```
Var1:=MUX(0,30,40,50,60,70,80); (* el resultado es 30 *);
```

Fig. 10-61 : Ejemplo de MUX en ST

Nota: ¡Para fines de optimización del tiempo de ejecución, sólo se calcula la expresión antepuesta a INK! En cambio, en la simulación se calculan todas las ramas.

10.5 Operadores de comparación

GT

Mayor que.

Un operador booleano con el resultado TRUE si el primer operando es mayor que el segundo. Los operandos pueden ser del tipo BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME y STRING.

```
LD 20
GT 30
ST  Var1 (* el resultado es FALSE *)
```

Fig. 10-62 : Ejemplo de GT en AWL

```
VAR1 := 20 > 30 > 40 > 50 > 60 > 70;
```

Fig. 10-63 : Ejemplo de GT en ST

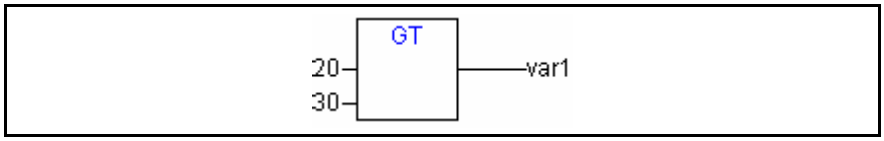


Fig. 10-64 : Ejemplo de GT en FUP

LT

Menor que.

Un operador booleano con el resultado TRUE si el primer operando es menor que el segundo. Los operandos pueden ser del tipo BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME y STRING.

```
LD 20
LT 30
ST  Var1 (* el resultado es TRUE *)
```

Fig. 10-65 : Ejemplo de LT en AWL

```
VAR1 := 20 < 30;
```

Fig. 10-66 : Ejemplo de LT en ST

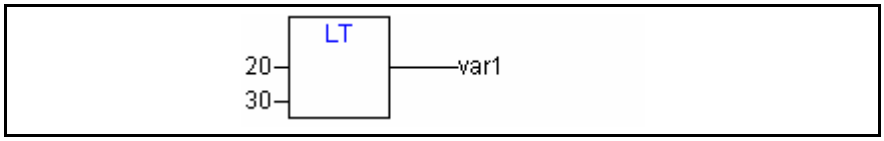


Fig. 10-67 : Ejemplo de LT en FUP

LE

Menor o igual.

Un operador booleano con el resultado TRUE si el primer operando es menor o igual que el segundo operando. Los operandos pueden ser del tipo BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT,

UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME y STRING.

```
LD 20
LE 30
ST  Var1 (* el resultado es TRUE *)
```

Fig. 10-68 : Ejemplo de LE en AWL

```
VAR1 := 20 <= 30;
```

Fig. 10-69 : Ejemplo de LE en ST

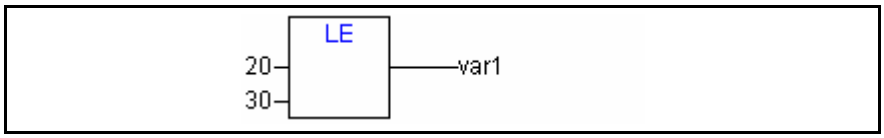


Fig. 10-70 : Ejemplo de LE en FUP

GE

Mayor o igual

Un operador booleano con el resultado TRUE si el primer operando es mayor o igual que el segundo operando. Los operandos pueden ser del tipo BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME y STRING.

```
LD 60
GE 40
ST  Var1 (* el resultado es TRUE *)
```

Fig. 10-71 : Ejemplo de GE en AWL

```
VAR1 := 60 >= 40;
```

Fig. 10-72 : Ejemplo de GE en ST

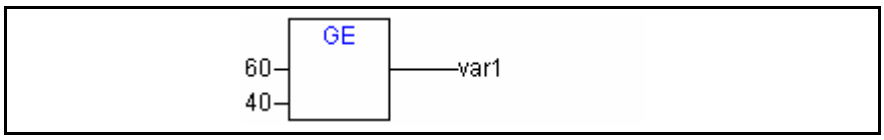


Fig. 10-73 : Ejemplo de GE en FUP

EQ

Igualdad

Un operador booleano con el resultado TRUE si los operandos son iguales. Los operandos pueden ser del tipo BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME y STRING.

```
LD 40
EQ 40
ST  Var1 (* el resultado es TRUE *)
```

Fig. 10-74 : Ejemplo de EQ en AWL

```
VAR1 := 40 = 40;
```

Fig. 10-75 : Ejemplo de EQ en ST

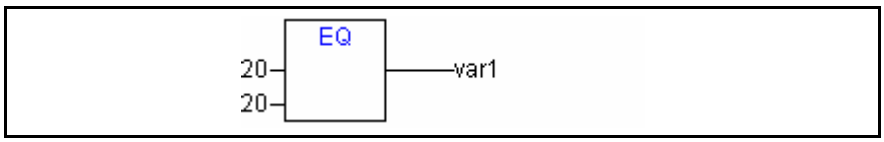


Fig. 10-76 : Ejemplo de EQ en FUP

NE

Desigualdad

Un operador booleano con el resultado TRUE si los operandos son distintos. Los operandos pueden ser del tipo BOOL, BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL, LREAL, TIME, DATE, TIME_OF_DAY, DATE_AND_TIME y STRING.

```
LD 40
NE 40
ST  Var1 (* el resultado es FALSE *)
```

Fig. 10-77 : Ejemplo de NE en AWL

```
VAR1 := 40 <> 40;
```

Fig. 10-78 : Ejemplo de NE en ST

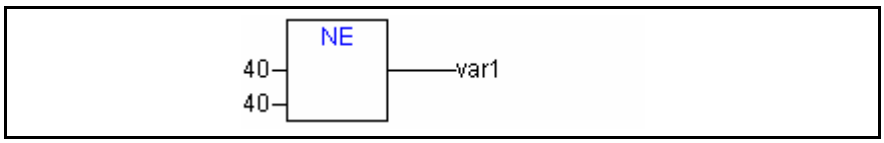


Fig. 10-79 : Ejemplo de NE en FUP

10.6 Operadores de dirección

Nota: Cuando se utiliza Cambio Online, los contenidos de las direcciones pueden desplazarse. Tenga esto en cuenta al utilizar pointers en direcciones.

ADR

Función de dirección, no prescrita por la norma IEC61131-3.

ADR arroja la dirección de su argumento en una DWORD. Esta dirección puede enviarse a funciones del fabricante y ser tratada allí como un pointer, o bien puede asignarse a un pointer dentro del proyecto.

```
dwVar:=ADR(bVAR);
```

Fig. 10-80 : Ejemplo de ADR en ST

```
LD bVar
ADR
ST dwVar
man_fun1
```

Fig. 10-81 : Ejemplo de ADR en AWL

ADRINST

Función de dirección, no prescrita por la norma IEC61131-3.

ADRINST arroja, dentro de una instancia de bloque de función, la dirección de dicha instancia en una DWORD. Esta dirección puede transmitirse a funciones y ser tratada allí como un pointer, o bien puede asignarse a un pointer dentro del proyecto.

```
dvar:=ADRINST();   (* Escribir la direccion de la
instancia
                    en la variable dvar *)
fun(a:=ADRINST()); (* Transmision de la instancia de
direccion al parametro de entrada a de la funcion fun *)
```

Fig. 10-82 : Ejemplos de ADRINST en ST (dentro de una instancia de bloque de función)

```
ADRINST
ST dvar

ADRINST
fun
```

Fig. 10-83 : Ejemplo de ADRINST en AWL

BITADR

Función de dirección, no prescrita por la norma IEC61131-3.

BITADR arroja el offset de bit dentro del segmento en una DWORD. Tenga en cuenta que el offset depende de si la opción Direccionamiento de bytes está o no activada en los ajustes del sistema de destino.

```
VAR
 var1 AT %IX2.3:BOOL;
 bitoffset: DWORD;
END_VAR
```

```
bitoffset:=BITADR(var1); (* resultado con
                            Direccionamiento de bytes=TRUE:
19, con
                            Direccionamiento de bytes=FALSE:
35 *)
```

Fig. 10-84 : Ejemplo de BITADR en ST

```
LD Var1
BITADR
ST Var2
```

Fig. 10-85 : Ejemplo de BITADR en AWL

Operador de contenido

La desreferenciación de un pointer se realiza insertando el operador de contenido "^" detrás del denominador del pointer.

```
pt:POINTER TO INT;
var_int1:INT;
var_int2:INT;
pt := ADR(var_int1);
var_int2:=pt^;
```

Fig. 10-86 : Ejemplo de operador de contenido ^ en ST

10.7 Operador de llamada

CAL

Llamada de un bloque de función

Mediante CAL se llama en AWL la instancia de un bloque de función. Detrás del nombre de la instancia del bloque de función se indica entre paréntesis la asignación de las variables de entrada del bloque de función.

```
CAL INST(PAR1 := 0, PAR2 := TRUE)
```

Fig. 10-87: Ejemplo de CAL en AWL

En la Fig. 10-87 se muestra la llamada de la instancia *Inst* de un bloque de función con asignación de las variables de entrada *Par1*, *Par2* a 0 y TRUE, respectivamente.

10.8 Conversiones de tipo

No está permitido convertir implícitamente de un tipo “mayor” a otro “menor” (por ejemplo, de INT a BYTE o del DINT a WORD). Si se desea hacerlo, es preciso utilizar conversiones de tipo especiales. Básicamente, es posible convertir desde cualquier tipo elemental a cualquier otro tipo elemental.

```
<tipoelem1>_TO_<tipoelem2>
```

Fig. 10-88 : Sintaxis de la conversión de tipo

Nota: Tenga en cuenta que, en las conversiones ...TO_STRING, el string se genera “alineado a la izquierda”. Si se define un string demasiado corto, se cortará desde la derecha.

Conversiones BOOL_TO

Conversión desde el tipo BOOL a otro tipo:

En tipos de número, el resultado es 1 si el operando es TRUE y 0 si el operando es FALSE.

En el tipo STRING, el resultado es TRUE o FALSE.

LD TRUE BOOL_TO_INT ST i	(* el resultado es 1 *)
LD TRUE BOOL_TO_STRING ST str	(* el resultado es 'TRUE' *)
LD TRUE BOOL_TO_TIME ST t	(* el resultado es T#1ms *)
LD TRUE BOOL_TO_TOD ST	(* el resultado es TOD#00:00:00.001 *)
LD FALSE BOOL_TO_DATE ST dat	(* el resultado es D#1970-01-01*)

LD TRUE BOOL_TO_DT ST dandt	(* el resultado es DT#1970-01-01-00:00:01 *)

Fig. 10-89 : Ejemplos de conversiones BOOL_TO en AWL

i:=BOOL_TO_INT(TRUE);	(* el resultado es 1 *)
str:=BOOL_TO_STRING(TRUE);	(* el resultado es 'TRUE' *)
t:=BOOL_TO_TIME(TRUE);	(* el resultado es T#1ms *)
tof:=BOOL_TO_TOD(TRUE);	(* el resultado es TOD#00:00:00.001 *)
dat:=BOOL_TO_DATE(FALSE);	(* el resultado es D#1970-01-01*)
dandt:=BOOL_TO_DT(TRUE);	(* el resultado es DT#1970-01-01-00:00:01 *)

Fig. 10-90 : Ejemplos de conversiones BOOL_TO en ST

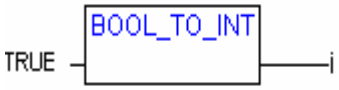
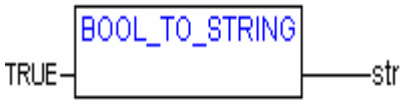
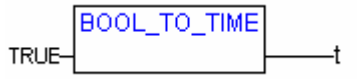
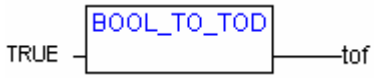
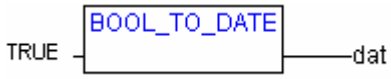
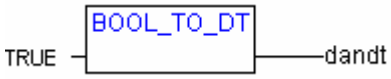
	(* el resultado es 1 *)
	(* el resultado es 'TRUE' *)
	(* el resultado es T#1ms *)
	(* el resultado es TOD#00:00:00.001 *)
	(* el resultado es D#1970-01-01*)
	(* el resultado es DT#1970-01-01-00:00:01 *)

Fig. 10-91 : Ejemplos de conversiones BOOL_TO en FUP

Conversiones TO_BOOL

Conversión desde un tipo al tipo BOOL:

El resultado es TRUE si el operando es distinto a 0. El resultado es FALSE si el operando es igual a 0.

En el tipo STRING, el resultado es TRUE si el operando es 'TRUE', de lo contrario el resultado es FALSE.

LD 213 BYTE_TO_BOOL ST b	(* el resultado es TRUE *)
LD 0 INT_TO_BOOL ST b	(* el resultado es FALSE *)
LD T#5ms TIME_TO_BOOL ST b	(* el resultado es TRUE *)
LD 'TRUE' STRING_TO_BOOL ST b	(* el resultado es TRUE *)

Fig. 10-92 : Ejemplo de conversiones TO_BOOL en AWL

b := BYTE_TO_BOOL(2#11010101);	(* el resultado es TRUE *)
b := INT_TO_BOOL(0);	(* el resultado es FALSE *)
b := TIME_TO_BOOL(T#5ms);	(* el resultado es TRUE *)
b := STRING_TO_BOOL('TRUE');	(* el resultado es TRUE *)

Fig. 10-93 : Ejemplos de conversiones TO_BOOL en ST

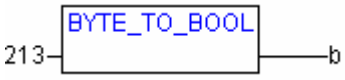
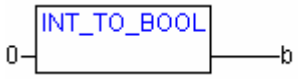
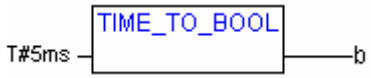
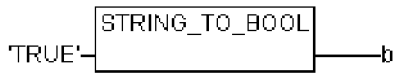
	(* el resultado es TRUE *)
	(* el resultado es FALSE *)
	(* el resultado es TRUE *)
	(* el resultado es TRUE *)

Fig. 10-94 : Ejemplos de conversiones TO_BOOL en FUP

Conversiones entre tipos de números enteros

Conversión desde un tipo de número entero a otro tipo de número:

Durante la conversión de tipos mayores a menores se puede perder información. Si el número a convertir supera el límite de la gama, no se tienen en cuenta los primeros bytes del número.

```
si := INT_TO_SINT(4223); (* el resultado es 127 *)
```

Fig. 10-95: Ejemplo de la conversión de tipos de números enteros en ST

Si, como se muestra en Fig. 10-, guarda el número entero 4223 (16#107f en representación hexadecimal) en una variable SINT, ésta contiene el número 127 (16#7f en representación hexadecimal).

```
LD 2
INT_TO_REAL
MUL
```

Fig. 10-96 : Ejemplo de la conversión de tipos de números enteros en AWL

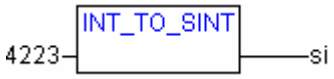


Fig. 10-97 : Ejemplo de la conversión de tipos de números enteros en FUP

Conversiones REAL_TO/LREAL_TO

Conversión desde el tipo REAL o LREAL a otro tipo:

Se redondea hacia arriba o hacia abajo a un valor entero y se convierte a los tipos en cuestión. Las excepciones a esto son los tipos STRING, BOOL, REAL y LREAL.

Durante la conversión de tipos mayores a menores se puede perder información.
Durante la conversión al tipo STRING, tenga en cuenta que el número total de dígitos está limitado a 16. Si el número (L)REAL contiene más

dígitos, se redondea el 16º dígito y se representa así en el string. Si el STRING definido para el número es demasiado corto, se cortará empezando por la derecha.

```
i := REAL_TO_INT(1.5); (* el resultado es 2 *)
j := REAL_TO_INT(1.4); (* el resultado es 1 *)
i := REAL_TO_INT(-1,5); (* el resultado es -2 *)
j := REAL_TO_INT(-1,4); (* el resultado es -1 *)
```

Fig. 10-98 : Ejemplos de REAL_TO_INT en ST

```
LD 2.7
REAL_TO_INT
GE %MW8
```

Fig. 10-99 : Ejemplo de REAL_TO_INT en AWL

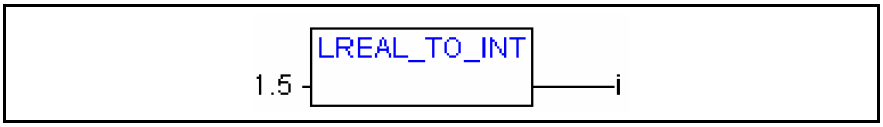


Fig. 10-100 : Ejemplo de LREAL_TO_INT en FUP

Conversiones TIME_TO/TIME_OF_DAY

Conversión desde el tipo TIME o TIME_OF_DAY a otro tipo:

Internamente se guarda el tiempo en una DWORD en milisegundos (en TIME_OF_DAY desde las 00:00 horas). Este valor se convierte.

Durante la conversión de tipos mayores a menores se puede perder información.

En el tipo STRING, el resultado es la constante de tiempo.

LD T#12ms TIME_TO_STRING ST str	(* el resultado es 'T#12ms' *)
LD T#300000ms TIME_TO_DWORD ST dw	(* el resultado es 300000 *)
LD TOD#00:00:00.012 TOD_TO_SINT ST si	(* el resultado es 12 *)

Fig. 10-101 : Ejemplos de conversiones TIME_TO y TOD_TO en AWL

str :=TIME_TO_STRING(T#12ms);	
dw:=TIME_TO_DWORD(T#5m);	
si:=TOD_TO_SINT(TOD#00:00:00.012);	

Fig. 10-102 : Ejemplos de conversiones TIME_TO y TOD_TO en ST

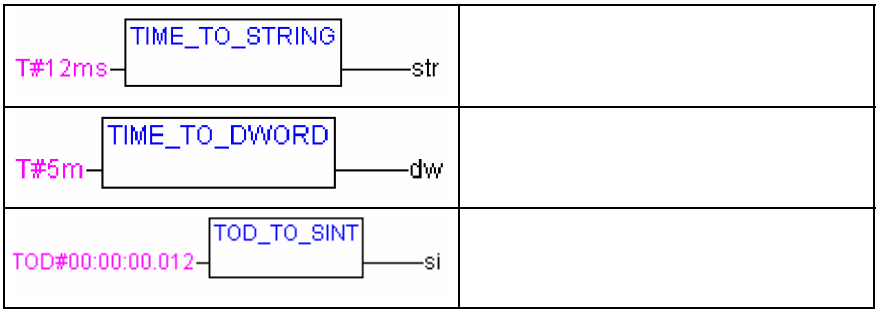


Fig. 10-103 : Ejemplos de conversiones TIME_TO y TOD_TO en FUP

Conversiones DATE_TO / DT_TO

Conversión desde el tipo DATE o DATE_AND_TIME a otro tipo:

Internamente se guarda la fecha en una DWORD en segundos desde el 1 de enero de 1970. Este valor se convierte.

Durante la conversión de tipos mayores a menores se puede perder información.

En el tipo STRING, el resultado es la constante de fecha.

LD D#1970-01-01 DATE_TO_BOOL ST b	(* el resultado es FALSE *)
LD D#1970-01-15 DATE_TO_INT ST i	(* el resultado es 29952 *)
LD DT#1970-01-15-05:05:05 DT_TO_BYTE ST byt	(* el resultado es 129 *)
LD DT#1998-02-13-14:20 DT_TO STRING ST str	(* el resultado es 'DT#1998-02-13-14:20' *)

Fig. 10-104 : Ejemplos de conversiones DATE_TO y DT_TO en AWL

b :=DATE_TO_BOOL(D#1970-01-01);	
i :=DATE_TO_INT(D#1970-01-15);	
byt :=DT_TO_BYTE(DT#1970-01-15-05:05:05);	
str:=DT_TO_STRING(DT#1998-02-13-14:20);	

Fig. 10-105 : Ejemplos de conversiones DATE_TO y DT_TO en ST

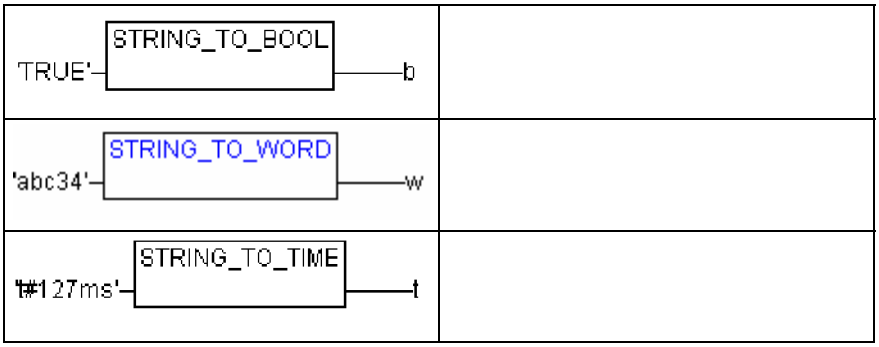


Fig. 10-106 : Ejemplos de conversiones DATE_TO y DT_TO en FUP

TRUNC

Conversión del tipo REAL al tipo INT. Sólo se toma el valor de la parte de número entero del número.

Durante la conversión de tipos mayores a menores se puede perder información.

```
LD 2.7
TRUNC
GE %MW8
```

Fig. 10-107 : Ejemplo de TRUNC en AWL

```
i:=TRUNC(1.9); (* el resultado es 1 *)
i:=TRUNC(-1,4); (* el resultado es -1 *)
```

Fig. 10-108 : Ejemplo de TRUNC en ST

10.9 Operadores numéricos

ABS

Arroja el valor absoluto de un número. ABS(-2). Son posibles las siguientes combinaciones de tipos para IN y OUT:

IN	OUT
INT	INT, REAL, WORD, DWORD, DINT
REAL	REAL
BYTE	INT, REAL, BYTE, WORD, DWORD, DINT
WORD	INT, REAL, WORD, DWORD, DINT
DWORD	REAL, DWORD, DINT
SINT	REAL
USINT	REAL
UINT	INT, REAL, WORD, DWORD, DINT, UDINT, UINT
DINT	REAL, DWORD, DINT
UDINT	REAL, DWORD, DINT, UDINT

Fig. 10-109 : Combinaciones de tipos para IN y OUT en el operador ABS

```
LD  -2
ABS
ST  i  (* el resultado es 2 *)
```

Fig. 10-110 : Ejemplo de ABS en AWL

```
i:=ABS(-2);
```

Fig. 10-111 : Ejemplo de ABS en ST

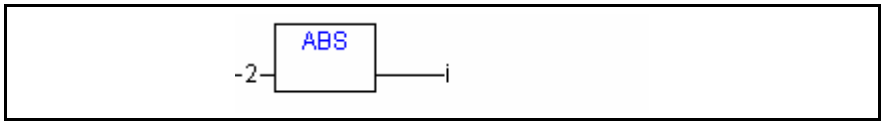


Fig. 10-112 : Ejemplo de ABS en FUP

SQRT

Arroja la raíz cuadrada de un número.

IN puede ser del tipo BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, mientras que OUT debe ser del tipo REAL.

```
LD  16
SQRT
ST  q (* el resultado es 4 *)
```

Fig. 10-113 : Ejemplo de SQRT en AWL

```
q:=SQRT(16);
```

Fig. 10-114 : Ejemplo de SQRT en ST

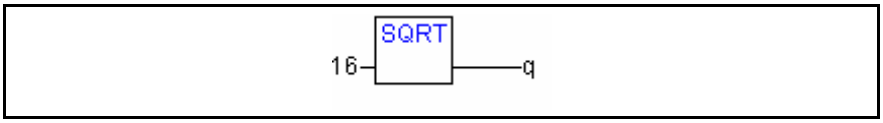


Fig. 10-115 : Ejemplo de SQRT en FUP

LN

Arroja el logaritmo natural de un número.

IN puede ser del tipo BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, mientras que OUT debe ser del tipo REAL.

```
LD  45
LN
ST  q (* el resultado es 3,80666 *)
```

Fig. 10-116 : Ejemplo de LN en AWL

```
q:=LN(45);
```

Fig. 10-117 : Ejemplo de LN en ST

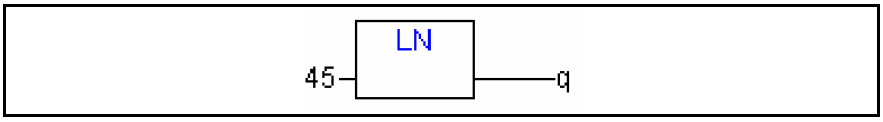


Fig. 10-118 : Ejemplo de LN en FUP

LOG

Arroja el logaritmo en base 10 de un número.

IN puede ser del tipo BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, mientras que OUT debe ser del tipo REAL.

```
LD  314.5
LOG
ST  q (* el resultado es 2,49762 *)
```

Fig. 10-119 : Ejemplo de LOG en AWL

```
q:=LOG(314.5);
```

Fig. 10-120 : Ejemplo de LOG en ST

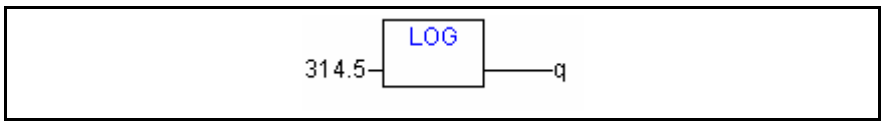


Fig. 10-121 : Ejemplo de LOG en FUP

EXP

Arroja la función exponencial.
IN puede ser del tipo BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, mientras que OUT debe ser del tipo REAL.

```
LD  2
EXP
ST  q (* el resultado es 7,389056099 *)
```

Fig. 10-122 : Ejemplo de EXP en AWL

```
q:=EXP(2);
```

Fig. 10-123 : Ejemplo de EXP en ST

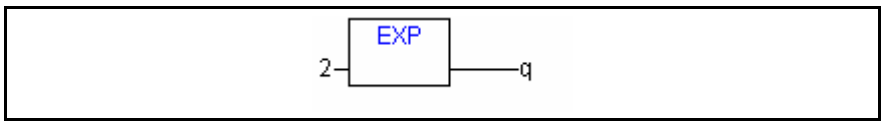


Fig. 10-124 : Ejemplo de EXP en FUP

SIN

Arroja el seno de un número. El valor se calcula en medida del arco.

IN puede ser del tipo BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, mientras que OUT debe ser del tipo REAL.

```
LD  0.5
SIN
ST  q (* el resultado es 0,479426 *)
```

Fig. 10-125 : Ejemplo de SIN en AWL

```
q:=SIN(0.5);
```

Fig. 10-126 : Ejemplo de SIN en ST

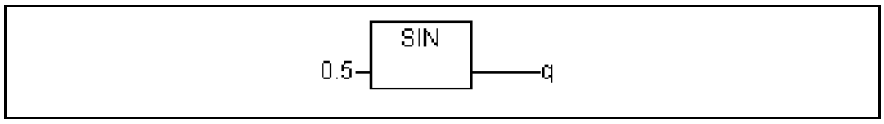


Fig. 10-127 : Ejemplo de SIN en FUP

COS

Arroja el coseno de un número. El valor se calcula en medida del arco.

IN puede ser del tipo BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, mientras que OUT debe ser del tipo REAL.

```
LD  0.5
COS
ST  q (* el resultado es 0,877583 *)
```

Fig. 10-128 : Ejemplo de COS en AWL

```
q:=COS(0.5);
```

Fig. 10-129 : Ejemplo de COS en ST

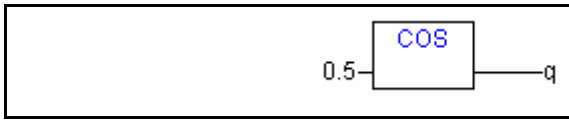


Fig. 10-130 : Ejemplo de COS en FUP

TAN

Arroja la tangente de un número. El valor se calcula en medida del arco.

IN puede ser del tipo BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, mientras que OUT debe ser del tipo REAL.

```
LD  0.5
TAN
ST  q (* el resultado es 0,546302 *)
```

Fig. 10-131 : Ejemplo de TAN en AWL

```
q:=TAN(0.5);
```

Fig. 10-132 : Ejemplo de TAN en ST

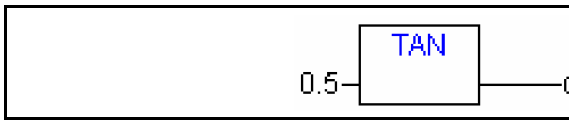


Fig. 10-133 : Ejemplo de TAN en FUP

ASIN

Arroja el arcoseno (función inversa del seno) de un número. El valor se calcula en medida del arco.

IN puede ser del tipo BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, mientras que OUT debe ser del tipo REAL.

```
LD  0.5
ASIN
ST  q (* el resultado es 0,523599 *)
```

Fig. 10-134 : Ejemplo de ASIN en AWL

```
q:=ASIN(0.5);
```

Fig. 10-135 : Ejemplo de ASIN en ST

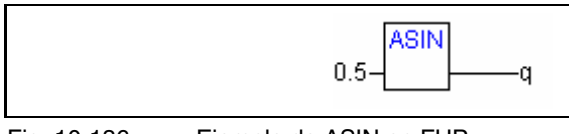


Fig. 10-136 : Ejemplo de ASIN en FUP

ACOS

Arroja el arcocoseno (función inversa del coseno) de un número. El valor se calcula en medida del arco.

IN puede ser del tipo BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, mientras que OUT debe ser del tipo REAL.

```
LD  0.5
ABS
ST  q (* el resultado es 1,0472 *)
```

Fig. 10-137 : Ejemplo de ACOS en AWL

```
q:=ACOS(0.5);
```

Fig. 10-138 : Ejemplo de ACOS en ST

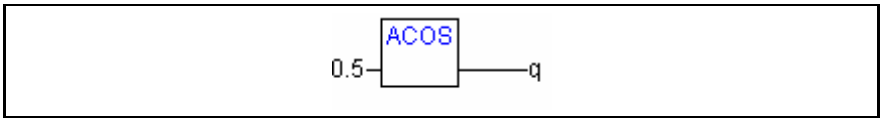


Fig. 10-139 : Ejemplo de ACOS en FUP

ATAN

Arroja la arcotangente (función inversa de la tangente) de un número. El valor se calcula en medida del arco.

IN puede ser del tipo BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, mientras que OUT debe ser del tipo REAL.

```
LD  0.5
ABS
ST  q (* el resultado es 0,463648 *)
```

Fig. 10-140 : Ejemplo de ATAN en AWL

```
q:=ATAN(0.5);
```

Fig. 10-141 : Ejemplo de ATAN en ST

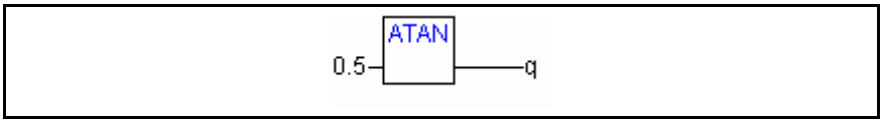


Fig. 10-142 : Ejemplo de ATAN en FUP

EXPT

Potenciación de una variable con otra:

```
OUT = IN1IN2.
```

IN1 e IN2 pueden ser del tipo BYTE, WORD, DWORD, INT, DINT, REAL, SINT, USINT, UINT, UDINT, mientras que OUT debe ser del tipo REAL.

```
LD  7
EXPT 2
ST  var1 (* el resultado es 49 *)
```

Fig. 10-143 : Ejemplo de EXPT en AWL

```
var1 := EXPT(7,2);
```

Fig. 10-144 : Ejemplo de EXPT en ST

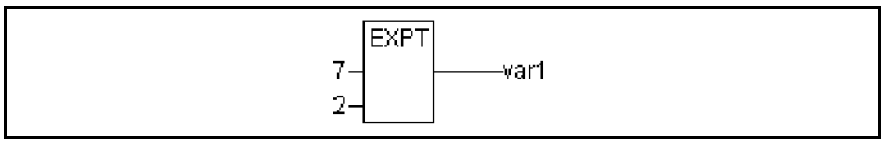


Fig. 10-145 : Ejemplo de EXPT en FUP

10.10 Operador de inicialización

Operador INI

Mediante el operador INI se pueden inicializar variables Retain de una instancia de bloque funcional utilizada en el componente.

Se debe asignar el operador a una variable booleana.

```
<variable bool> := INI(<instancia FB, TRUE|FALSE)
```

Fig. 10-146 : Sintaxis INI

Si el segundo parámetro del operador está ajustado en TRUE, se inicializan todas las variables Retain definidas en el bloque de función FB.

```
fbinst:fb;
b:bool;
```

Fig. 10-147 : Ejemplo de INI en ST – Declaración en el componente

```
b := INI(fbinst, TRUE);
ivar:=fbinst.retvar  (* => se inicializa retvar *)
```

Fig. 10-148 : Ejemplo de INI en ST – Parte de programa

```
LD fbinst
INI TRUE
ST b
```

Fig. 10-149 : Ejemplo de INI en AWL – Llamada de operador

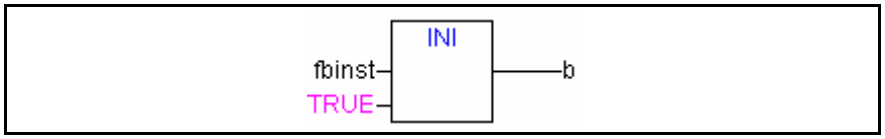


Fig. 10-150 : Ejemplo de INI en FUP – Llamada de operador