11 Apéndice B: Operandos en IndraLogic

11.1 Constantes

En **IndraLogic** se pueden utilizar como operandos constantes, variables, direcciones y en algunos casos llamadas de función.

BOOL:Constantes

Las constantes BOOL son los valores lógicos TRUE y FALSE.

Constantes TIME

En **IndraLogic** se pueden declarar constantes TIME. Se utilizan especialmente para manejar los contadores desde la biblioteca estándar. Una constante TIME consiste siempre en una "t" o "T" inicial ("time" o "TIME" en la forma no abreviada) y una cruz doble "#".

A continuación viene la declaración de tiempo propiamente dicha, que puede consistir en días (indicados con "d"), horas (indicadas con "h"), minutos (indicados con "m"), segundos (indicados con "s") y milisegundos (indicados con "ms"). Se debe tener en cuenta que las indicaciones de tiempo deben estar ordenadas según su tamaño (d antes de h antes de m antes de s antes de m antes de ms), si bien no tienen por qué aparecer todos los tiempos.

TIME1 := T#14ms;	
TIME1 := T#100S12ms;	(*Se permite la superación del límite en el componente más alto*)
TIME1 := t#12h34m15s;	

Fig. 11-1 : Ejemplos de constantes TIME correctas en una asignación ST

TIME1 := t#5m68s	(*Superación del límite en un componente más bajo*)
TIME1 := 15ms;	(*Falta T#*)
TIME1 := t#4ms13d;	(*orden incorrecto de los datos de tiempo*)

Fig. 11-2 : Ejemplos de constantes TIME **incorrectas** en una asignación ST

Constantes DATE

Mediante este tipo se pueden realizar indicaciones de fecha. Una constante DATE se declara mediante una "d", "D", "DATE" o "date" inicial, seguida de un símbolo "#". A continuación puede introducir una fecha cualquiera en el orden año-mes-día.

```
DATE#2005-05-06
d#1998-03-29
```

Fig. 11-3 : Ejemplos de constantes DATE

Los valores DATE (abreviado como D) se tratan internamente como DWORD. El tiempo se indica en **segundos**, y se calcula a partir del 1 de enero de 1970 a las 00:00 horas.

Constantes TIME_OF_DAY

Mediante este tipo puede guardar horas. Una declaración TIME_OF_DAY empieza por "tod#", "TOD#", "TIME_OF_DAY#" o "time_of_day#", y a continuación puede indicar una hora en la notación: hora:minuto:segundo. Los segundos pueden indicarse como números

reales, y por lo tanto también se pueden especificar fracciones de segundo.

```
TIME_OF_DAY#15:36:30.123
tod#00:00:00
```

Fig. 11-4 : Ejemplos de constantes TIME_OF_DAY

Los valores TIME_OF_DAY (abreviado como TOD) se tratan internamente como DWORD. El tiempo se indica en **milisegundos**, y se calcula a partir de las 00:00 horas.

Constantes DATE_AND_TIME

Las constantes de fecha y de hora también pueden combinarse para formar las denominadas constantes DATE_AND_TIME. Las constantes DATE_AND_TIME empiezan por "dt#", "DT#", "DATE_AND_TIME#" o "date_and_time#". Tras la indicación de la fecha sigue un guión y a continuación la hora.

```
DATE_AND_TIME#2005-05-06-15:36:30
dt#1998-03-29-00:00:00
```

Fig. 11-5 : Ejemplos de constantes DATE_AND_TIME

Los valores DATE_AND_TIME (abreviado como DT) se tratan internamente como DWORD. El tiempo se indica en **segundos**, y se calcula a partir del 1 de enero de 1970 a las 00:00 horas.

Constantes de número

Los valores numéricos pueden aparecer como números binarios, números octales, números decimales y números hexadecimales. Si un valor entero no es un número decimal, se debe escribir su base seguida de una cruz doble (#) delante de la constante entera. Los valores de los números 10 al 15 en números hexadecimales se indican, como habitualmente, mediante las letras A-F.

Se permiten guiones bajos dentro de un valor numérico.

Número decimal	Número binario	Número octal	Número hexadecimal
14	2#1001_0011	8#67	16#A

Fig. 11-6 : Ejemplos de valores numéricos

El tipo de estos valores numéricos puede ser BYTE, WORD, DWORD, SINT, USINT, INT, UINT, DINT, UDINT, REAL o LREAL.

No se admiten conversiones implícitas de tipos "mayores" a "menores". Es decir, una variable DINT no puede utilizarse sin más como variable INT. Para ello se utilizan las conversiones de tipo.

Constantes REAL / LREAL

Las constantes REAL y LREAL pueden indicarse como fracciones decimales y en representación exponencial. Para ello se utiliza la notación estadounidense con punto.

Ejemplos:

```
7.4 en lugar de 7,4
1.64e+009 en lugar de 1,64e+009
```

Constantes STRING

Un string es una secuencia de caracteres cualquiera. Las constantes STRING se delimitan mediante comillas simples delante y detrás. También pueden introducirse espacios y diéresis. Se tratarán exactamente igual que todos los demás caracteres.

En secuencias de caracteres, la combinación del símbolo del dólar ($) seguido de dos cifras hexadecimales se interpreta como representación hexadecimal del código de caracteres de ocho bits. Además, cuando aparecen en una secuencia de caracteres, las combinaciones de dos caracteres que empiezan con el símbolo del dólar se interpretan de la siguiente forma:

$$	Símbolos de dólar
$'	Comilla simple
$L o $l	Avance de línea
$N o $n	Nueva línea
$P o $p	Avance de página
$R o $r	Salto de línea
$T o $t	Tabulador

Fig. 11-7 : Interpretación de secuencias de caracteres que empiezan con el signo $

```
'w1W$■?'
'Pili y Juan'
':-)'
```

Fig. 11-8 : Ejemplos de constantes STRING

Constantes tipificadas (Typed Literals)

A excepción de las constantes REAL/LREAL (aquí se utiliza siempre LREAL), durante el cálculo con constantes IEC se utiliza siempre el tipo de dato más pequeño posible. Si se desea utilizar otro tipo de datos, esto puede lograrse mediante Typed Literals (constantes tipificadas), sin que sea preciso declarar explícitamente la constante. Para ello, se dota a la constante de un prefijo que establece el tipo:

La notación es: <Type>#<Literal>

<Type> indica el tipo de dato deseado, entradas posibles: BOOL, SINT, USINT, BYTE, INT, UINT, WORD, DINT, UDINT, DWORD, REAL, LREAL. El tipo debe escribirse en letras mayúsculas.

<Literal> indica la constante. La entrada debe concordar con el tipo de dato especificado en <Type>.

```
var1:=DINT#34;
```

Fig. 11-9 : Ejemplo para la constante tipificada

Si no es posible convertir la constante al tipo de destino sin pérdida de datos, se emite un mensaje de error:

Las constantes tipificadas pueden utilizarse en todos aquellos casos en los que se pueden emplear constantes normales.

11.2 Variables

Las variables se declaran localmente en la parte de declaración de un componente, o bien en las listas de variables globales.

Nota: Es posible definir una variable local con el mismo nombre que una variable global. Dentro de un componente siempre tiene prioridad la variable definida localmente. No es posible dar el mismo nombre a dos variables definidas globalmente; por ejemplo se emite un error de traducción si tanto en una lista de variables globales como en la configuración del control está definida la variable "var1".

En cuanto al identificador de la variable, es preciso recordar que no debe contener espacios ni diéresis, no debe ser declarado por duplicado y no debe ser idéntico a palabras clave. Para las variables no se tienen en cuenta las mayúsculas/minúsculas, de modo que VAR1, Var1 y var1 no son variables distintas. Los guiones bajos son significativos en el identificador, p. ej. "A_BCD" y "AB_CD" se interpretan como identificadores distintos. No se permiten varios guiones bajos seguidos al principio de un identificador o en un identificador. La longitud del identificador, así como su parte significativa, son ilimitadas. Las variables pueden utilizarse en todos aquellos casos en los que el tipo declarado lo permita.

Puede llamar las variables disponibles mediante el asistente de entrada.

Systemflags

Los systemflags son variables declaradas implícitamente que dependen de su control específico. Para determinar qué systemflags posee su sistema, seleccione la orden **"Insertar" "Operando"** y aparecerá el diálogo del asistente de entrada, donde debe escoger la categoría **Variable de sistema**.

Acceso a variables de arrays, estructuras y componentes

Con la sintaxis descrita a continuación se accede a las variables de arrays, estructuras y componentes, respectivamente.

```
<Nombre de campo>[Indice1, Indice2}]
```

Fig. 11-10 : Sintaxis para el acceso a componentes de arrays bidimensionales

```
<Nombre de estructura>.<Nombre de variable>
```

Fig. 11-11 : Sintaxis para el acceso a variables de estructuras

```
<Nombre de componente>.<Nombre de variable>
```

Fig. 11-12 : Sintaxis para el acceso a variables de bloques de función y programas

Direccionamiento de bits en variables

Se puede acceder a bits individuales en variables de número entero. Para ello se adjunta a la variable, separado por un punto, el índice del bit a direccionar. El índice de bit puede ser especificado por una constante cualquiera. La indexación está basada en 0.

```
a : INT;
b : BOOL;
...
a.2 := b;
```

Fig. 11-13 : Ejemplo del direccionamiento de un bit

En el ejemplo de la Fig. 11-13 se ajusta el tercer bit de la variable a al valor de la variable b.

Si el índice es mayor que la anchura de bit de la variable, se emite el siguiente error: ¡Índice "<n>" fuera de la gama válida para la variable "<var>"!

El direccionamiento de bits es posible en los siguientes tipos de variable: SINT, INT, DINT, USINT, UINT, UDINT, BYTE, WORD, DWORD.

Si el tipo de la variable no es admisible, se emite el siguiente mensaje de error: "Tipo de dato inadmisible '<Tipo>' para indexación directa".

¡Un acceso de bit no puede asignarse a una variable VAR_IN_OUT!

Acceso de bit mediante a una constante global:

Si se declara una constante global que define el número de bit se puede utilizar esta constante para el acceso de bit.

Nota: ¡Debe estar activada la opción de proyecto "Sustituir constantes" (categoría Opciones de traducción)!

En los siguientes ejemplos se muestran un acceso de bit a una variable normal y a una variable de estructura, respectivamente.

Declaración en lista de variables globales para ambos ejemplos siguientes:

```
VAR_GLOBAL CONSTANT
      enable:int:=2;
END_VAR
```

Fig. 11-14 : Declaración en lista de variables globales para los ejemplos 1 y 2

La variable enable de **Fig. 11-14** indica el número de bit al que se debe acceder.

Ejemplo 1, acceso de bit a variable de número entero:

```
VAR
      xxx:int;
END_VAR
```

Fig. 11-15 : Declaración en el componente

```
xxx.enable:=true;  (* -> el 3er bit en la variable xxx se
ajusta a TRUE *)
```

Fig. 11-16 : Acceso de bit

Ejemplo 2, acceso de bit a componente de estructura de número entero:

```
TYPE stru1 :

STRUCT
      bvar:BOOL;
      rvar:REAL;
      wvar:WORD;
      {bitaccess enable 42 'Liberar accionamiento'}
END_STRUCT

END_TYPE
```

Fig. 11-17 : Declaración de la estructura stru1

```
VAR
      x:stru1;
END_VAR
```

Fig. 11-18 : Declaración en el componente

```
x.enable:=true;
```

Fig. 11-19 : Acceso de bit

De este modo se ajusta a TRUE el 42º bit en la variable x. Dado que bvar contiene 8 bits y rvar contiene 32 bits, este acceso al 2º bit tiene lugar en la variable wvar, que por lo tanto recibe el valor 4.

Nota: Para representar correctamente en la monitorización, en el asistente de entrada y en la función "Intellisense" una variable que efectúe el acceso de bit a una variable de estructura mediante una constante global, utilice el **pragma {bitaccess}** mostrado en el ejemplo (ver capítulo Instrucciones de pragma en el editor de declaraciones a partir de la página 5-14). Durante la monitorización se muestra entonces además, en la ventana de declaración, la constante global debajo de la variable de estructura.

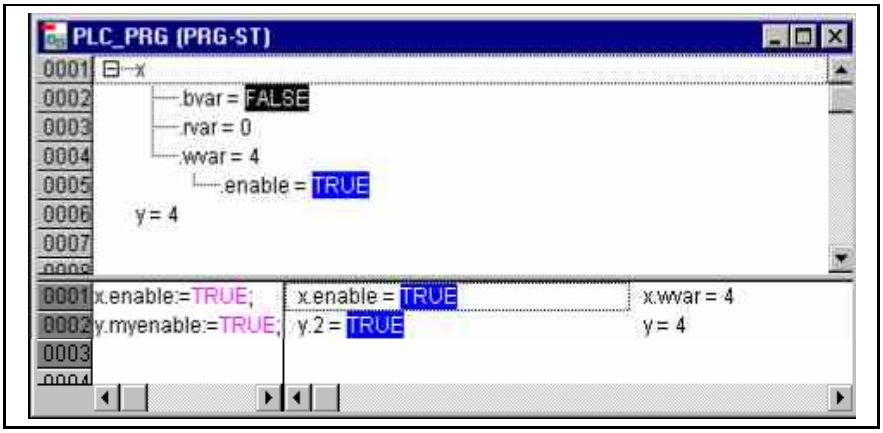

Fig. 11-20 : Ejemplo en modo Online

11.3 Direcciones

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

Dirección

La representación directa de celdas de memoria individuales tiene lugar mediante secuencias especiales de caracteres. Éstas constan de la concatenación del símbolo de porcentaje "%", un prefijo de área, un prefijo para el tamaño y uno o varios números naturales, separados entre sí por espacios.

Se admiten los siguientes prefijos de área:

I	Entrada
Q	Salida
M	Apuntador

Fig. 11-21 : Prefijos de área

Se soportan los siguientes prefijos para el tamaño:

X	Bit individual
None	Bit individual
B	Byte (8 bits)
W	Palabra (16 bits)
D	Palabra doble (32 bits)

Fig. 11-22 : Prefijos de tamaño

%QX7.5 y %Q7.5	Bit de salida 7.5
%IW215	Palabra de entrada 215
%QB7	Byte de salida 7
%MD48	Palabra doble en la posición de memoria 48 en el apuntador

Fig. 11-23 : Ejemplos de la utilización de prefijos en direcciones

La configuración actual del control en el programa determina si una dirección es válida.

Nota: Los valores booleanos se asignan por bytes si no se especifica explícitamente una dirección de bit individual. Ejemplo: Una variación del valor de varbool1 AT %QW0 afecta al rango desde QX0.0 hasta QX0.7.

A este respecto, consulte también en el Apéndice A: "Operadores IEC y funciones adicionales que amplían la norma" el capítulo "Operadores de dirección" a partir de la página 10-17.

Apuntador

Para el acceso al apuntador se pueden utilizar todos los tamaños soportados.

Por ejemplo, la dirección %MD48 direccionaría los bytes nº 192, 193, 194 y 195 en el rango de apuntador (48 * 4 = 192). El primer byte es el byte nº 0.

Asimismo, es posible acceder a palabras y bytes e incluso a bits: por ejemplo, con %MX5.0 se accede al primer bit en la quinta palabra (normalmente, los bits se guardan por palabras).

A este respecto, consulte también en el "Apéndice A: "Operadores IEC y funciones adicionales que amplían la norma" el capítulo "Operadores de dirección" a partir de la página 10-17.

11.4 Funciones

En el lenguaje ST también puede aparecer una llamada de función como operando.

Ejemplo:

```
Resultado := Fct(7) + 3;
```

Función TIME()

Esta función arroja, en base a milisegundos, el tiempo transcurrido desde la puesta en funcionamiento del sistema.

El tipo de dato es TIME.

```
TIME
ST systime  (* resultado p. ej.: T#35m11s342ms *)
```

Fig. 11-24 : Ejemplo de la función TIME en AWL

```
systime:=TIME();
```

Fig. 11-25 : Ejemplo de la función TIME en ST

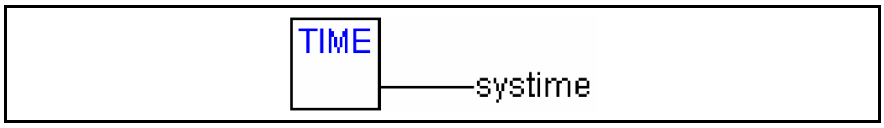

Fig. 11-26 : Ejemplo de la función TIME en FUP