Manual de Referencia

PL7 Micro/Junior/Pro

Descripción detallada de las Instrucciones y Funciones

spa Marzo 2005

35009570_00

a brand of Schneider Electric

Telemecanique

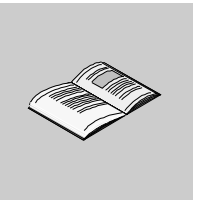
Estructura de la documentación

Presentación

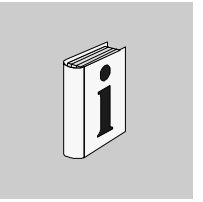
Este manual consta de tres tomos:

- Tomo 1: Descripción del programa PL7
  - Generalidades
  - Lenguaje de contactos
  - Lenguaje de la lista de instrucciones
  - Lenguaje Literal estructurado
  - Lenguaje Grafcet
  - Bloques de función DFB
  - Módulos Funcionales
- Tomo 2: Descripción detallada de las Instrucciones y de las Funciones
  - Instrucciones básicas
  - Instrucciones avanzadas
  - Objetos bits y palabras del sistema
- Tomo 3: Anexos
  - Diferencias entre PL7-2/3 y PL7-Micro/Junior
  - Ayuda-memoria
  - Lista de las palabras reservadas
  - Conforme a la norma CEI 1131-3
  - Servidor OLE Automation
  - Rendimiento

Tabla de materias



Acerca de este libro



Presentación

Objeto

Este manual describe las instrucciones de los lenguajes de programación de los autómatas Micro, Premium y Atrium.

Campo de aplicación

La actualización de esta publicación toma en cuenta las funcionalidades de PL7 V4.5. Sin embargo, también permite poner en marcha las versiones anteriores de PL7.

Comentarios del usuario

Envíe sus comentarios a la dirección electrónica techpub@schneider-electric.com

Instrucciones de base

1

Presentación

Contenido Este capítulo describe las instrucciones de base del lenguaje PL7.

Contenido: Este capítulo contiene las siguientes secciones:

1.1 Presentación de las instrucciones PL7

Instrucciones de PL7

Generalidades

Todos los lenguajes PL7 utilizan el mismo juego de instrucciones.

Las instrucciones booleanas y los bloques de función tienen distintas representaciones según el lenguaje.

Ejemplo: instrucción de carga

Instrucción	Lenguaje de contactos	Lista de instrucciones	Literal
Carga		LD	:=

Las instrucciones numéricas (aritméticas, lógicas, específicas...) tienen representaciones similares.

En este documento se describe de forma detallada el conjunto de las instrucciones; para facilitar su descripción, las mismas se clasifican en 2 juegos:
- instrucciones de base (Véase *Instrucciones de base, p. 13*)
- instrucciones avanzadas (Véase *Instrucciones avanzadas, p. 87*)

Instrucciones de base

Incluyen las instrucciones booleanas de base, los bloques de función predefinidos y las instrucciones aritméticas y lógicas de enteros.

Instrucciones avanzadas

Incluyen instrucciones que responden a necesidades de programación avanzada. Dichas instrucciones son de 2 tipos:
- lenguaje PL7: aumentan las posibilidades de tratamiento del lenguaje mediante funciones específicas (manipulación de cadenas de caracteres, gestión del tiempo...),
- específicas: ofrecen funciones para las funciones específicas que se van a tratar; ejemplo para la función específica de comunicación:
  - `PRINT` para enviar un mensaje de tipo cadena de caracteres a un terminal o una impresora,
  - `SEND` para enviar un mensaje a una aplicación,
  - `PID` función PID de regulación.

1.2 Instrucciones booleanas

Presentación

Objeto de este apartado

Este apartado describe las instrucciones booleanas del lenguaje PL7.

Contenido

Esta sección contiene los siguientes apartados:

Instrucciones sobre objetos bits

Instrucciones sobre bits

Las instrucciones siguientes se aplican a objetos bits.

Designación	Función
:=	Asignación de un bit
OR	O booleana
AND	Y booleana
XOR	O exclusiva booleana
NOT	Inversión
RE	Flanco ascendente
FE	Flanco descendente
SET	Puesta a 1
RESET	Puesta a 0

Instrucciones sobre tablas de bits

Las instrucciones siguientes se aplican a objetos de tipo tabla de bits.

Designación	Función
Tabla:= Tabla	Asignación entre dos tablas
Tabla:= Palabra	Asignación de una palabra a una tabla
Palabra:= Tabla	Asignación de una tabla a una palabra
Tabla:= Palabra doble	Asignación de una palabra doble a una tabla
Palabra doble:= Tabla	Asignación de una tabla a una palabra doble
COPY_BIT	Copia de una tabla de bits a una tabla de bits
AND_ARX	Y entre dos tablas
OR_ARX	O entre dos tablas
XOR_ARX	O exclusiva entre dos tablas
NOT_ARX	Negación en una tabla
BIT_W	Copia de una tabla de bits a una tabla de palabras
BIT_D	Copia de una tabla de bits a una tabla de palabras dobles
W_BIT	Copia de una tabla de palabras en una tabla de bits
D_BIT	Copia de una tabla de palabras dobles en una tabla de bits
LENGHT_ARX	Cálculo de la longitud de una tabla en número de elementos

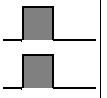
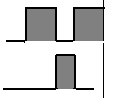
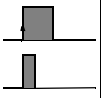
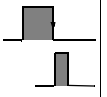
Definición de los principales objetos booleanos

Descripción En la siguiente tabla se describen los principales objetos booleanos.

Bits	Descripción	Ejemplos	Acceso de escritura
Valores inmediatos	0 ó 1 (False o True)	0	_
Entradas/ salidas	Estos bits son las "imágenes lógicas" de los estados eléctricos de las entradas/salidas. Se guardan en la memoria de datos y se actualizan en cada explotación de la tarea en la que se configuran. **Nota:** Los bits de entradas/salidas que no se utilizan no se pueden emplear como bits internos.	%I23.5 %Q51.2	No Sí
Internos	Los bits internos permiten almacenar los estados intermedios durante la ejecución del programa.	%M200	Sí
Sistema	Los bits de sistema %S0 a %S127 supervisan el correcto funcionamiento del autómata, así como el desarrollo del programa de aplicación.	%S10	Según i
Bloques de función	Los bits de bloques de función corresponden a la salidas de los bloques de función estándar o instancia de DFB. Estas salidas pueden conectarse directamente o bien utilizarse como objetos.	%TM8.Q	No
Extractos de palabras	El programa PL7 ofrece la posibilidad de extraer uno de los 16 bits de un objeto palabra.	%MW10:X5	Según el tipo de palabra
Etapas y macroetapas Grafcet	Los bits Grafcet de estado de las etapas, las macroetapas y las etapas de macroetapas permiten conocer el estado de la etapa i, de la macroetapa j o de la etapa i de la macroetapa j del Grafcet.	%X21 %X5.9	Sí Sí

Instrucciones de carga

Función En la siguiente tabla se describe la función de cada instrucción.

Lenguaje de contactos	Lista de instrucciones	Literal estructurado	Descripción	Cronograma
	LD	:=	Contactos de cierre: contacto de paso (resultado en 1) cuando el objeto bit que lo controla está en estado 1.	
	LDN	:=NOT	Contactos de apertura: contacto de paso (resultado en 1) cuando el objeto bit que lo controla está en estado 0.	
	LDR	:=RE	Contactos de flanco ascendente: detección del paso de 0 a 1 del objeto bit que lo controla. La puesta a 1 del resultado se efectúa durante 1 ciclo.	
	LDF	:=FE	Contactos de flanco descendente: detección del paso de 1 a 0 del objeto bit que lo controla. La puesta a 1 del resultado se efectúa durante 1 ciclo.	

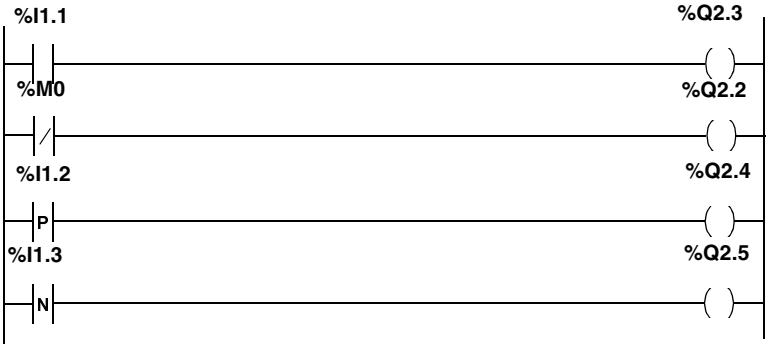
Operandos autorizados

En la siguiente tabla figura la lista de los operandos utilizados para estas instrucciones.

Lenguaje de contactos	Lista de instrucciones	Literal estructurado	Operandos
	LD	:=	%I,%Q,%M,%S,%BLK,%•:Xk, %Xi, (True y False en lista de instrucciones o literal estructurado)
	LDN	:=NOT	%I,%Q,%M,%S,%BLK,%•:Xk, %X (True y False en lista de instrucciones o literal estructurado)
	LDR	:=RE	%I,%Q,%M
	LDF	:=FE	%I,%Q,%M

Ejemplo en el lenguaje de contactos

El siguiente ejemplo muestra la programación de las instrucciones de carga en el lenguaje de contactos.



Ejemplo en lista de instrucciones

El siguiente ejemplo muestra la programación de las instrucciones de carga en el lenguaje lista de instrucciones.

```
LD    %I1.1
ST    %Q2.3
LDN   %M0
ST    %Q2.2
LDR   %I1.2
ST    %Q2.4
LDF   %I1.3
ST     %Q2.5
```

Ejemplo en literal estructurado

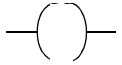
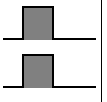
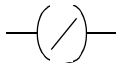
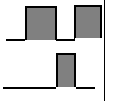
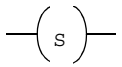
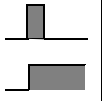
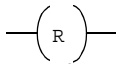
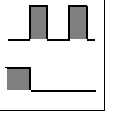
El siguiente ejemplo muestra la programación de las instrucciones de carga en el lenguaje literal estructurado.

```
%Q2.3:=%I1.1;
%Q2.2:=NOT %M0;
%Q2.4:=RE %I1.2;
%Q2.5:=FE %I1.3;
```

Instrucciones de asignación

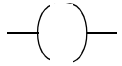
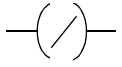
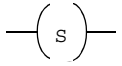
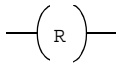
Función

En la siguiente tabla se describe la función de cada instrucción.

Lenguaje de contactos	Lista de instrucciones	Literal estructurado	Descripción	Cronograma
	ST	:=	en las bobinas directas: el objeto bit asociado toma el valor del resultado de la ecuación.	
	STN	:=NOT	en las bobinas inversas: el objeto bit asociado toma el valor inverso del resultado de la ecuación.	
	S	SET	en las bobinas en la conexión: el objeto bit asociado se pone a 1 cuando el resultado de la ecuación es 1	
	R	RESET	en las bobinas en la desconexión: el objeto bit asociado se pone a 0 cuando el resultado de la ecuación es 1	

Operandos permitidos

En la siguiente tabla figura la lista de los operandos utilizados para estas instrucciones.

Lenguaje de contactos	Lista de instrucciones	Literal estructurado	Operandos
	ST	:=	%I,%Q,%M,%S,%•:Xk
	STN	:=NOT	%I,%Q,%M,%S,%•:Xk
	S	SET	%I,%Q,%M,%S,%•:Xk,%Xi Únicamente en el tratamiento preliminar.
	R	RESET	%I,%Q,%M,%S,%•:Xk,%Xi Únicamente en el tratamiento preliminar.

Ejemplo en el lenguaje de contactos

El siguiente ejemplo muestra la programación de las instrucciones de asignación en el lenguaje de contactos.

Ejemplo en lista de instrucciones

El siguiente ejemplo muestra la programación de las instrucciones de asignación en el lenguaje lista de instrucciones.

```
LD   %I1.1
ST    %Q2.3

STN   %Q2.2

S     %Q2.4

LD   %I1.2
R     %Q2.4
```

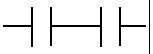
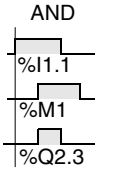
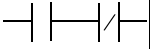
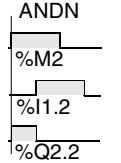
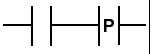
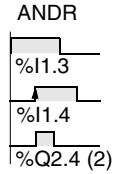
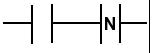
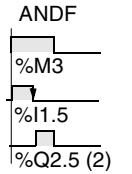
Ejemplo en literal estructurado

El siguiente ejemplo muestra la programación de las instrucciones de asignación en el lenguaje literal estructurado.

```
%Q2.3 := %I1.1;
%Q2.2 := NOT %I1.1;
IF %I1.1 THEN
    SET %Q2.4;
END_IF;
IF %I1.2 THEN
    RESET %Q2.4;
END_IF;
```

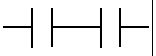
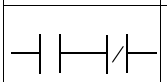
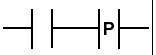
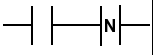
Instrucción Y lógico

Función En la siguiente tabla se describe la función de cada instrucción.

Lenguaje de contactos	Lista de instrucciones	Literal estructurado	Descripción	Cronograma
	AND	AND	Y lógico entre el operando y el resultado booleano de la instrucción anterior	
	ANDN	AND (NOT...)	Y lógico entre el operando inverso y el resultado booleano de la instrucción anterior	
	ANDR	AND (RE...)	Y lógico entre el flanco ascendente del operando y el resultado booleano de la instrucción anterior (2) Puesta a 1 durante 1 ciclo	
	ANDF	AND (FE...)	Y lógico entre el flanco descendente del operando y el resultado booleano de la instrucción anterior (2) Puesta a 1 durante 1 ciclo	

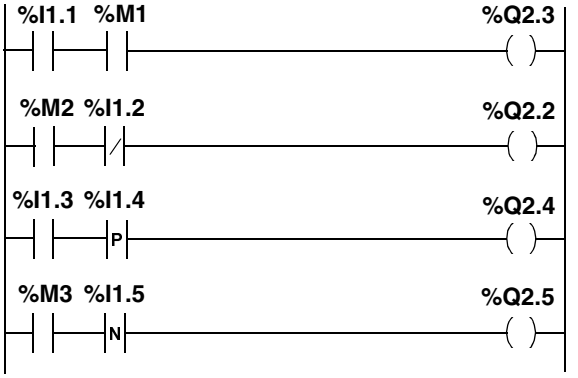
Operandos permitidos

En la siguiente tabla figura la lista de los operandos utilizados para estas instrucciones.

Lenguaje de contactos	Lista de instrucciones	Literal estructurado	Operandos
	AND	AND	%I, %Q, %M, %S, %BLK, %•:Xk, %Xi True (1)/False (0) en lenguaje lista de instrucciones o literal estructurado
	ANDN	AND (NOT...)	%I, %Q, %M, %S, %BLK, %•:Xk, %Xi True (1)/False (0) en lenguaje lista de instrucciones o literal estructurado
	ANDR	AND (RE...)	%I, %Q, %M
	ANDF	AND (FE...)	%I, %Q, %M

Ejemplo en el lenguaje de contactos

El siguiente ejemplo muestra la programación de las instrucciones Y lógico en el lenguaje de contactos.



Ejemplo en lista de instrucciones

El siguiente ejemplo muestra la programación de las instrucciones Y lógico en el lenguaje lista de instrucciones.

```
LD   %I1.1
AND  %M1
ST   %Q2.3
LD   %M2
ANDN %I1.2
ST   %Q2.2
LD   %I1.3
ANDR %I1.4
ST   %Q2.4
LD   %M3
ANDF %I1.5
ST   %Q2.5
```

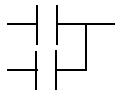
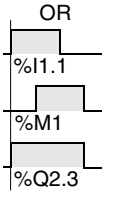
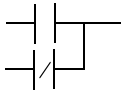
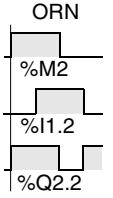
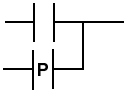
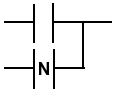
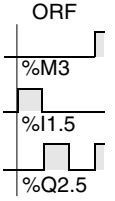
Ejemplo en lenguaje literal estructurado

El siguiente ejemplo muestra la programación de las instrucciones Y lógico en el lenguaje literal estructurado.

```
%Q2.3:=%I1.1 AND %M1;
%Q2.2:=%M2 AND (NOT%I1.2);
%Q2.4:=%I1.3 AND (RE%I1.4);
%Q2.5:=%M3 AND (FE%I1.5);
```

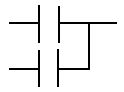
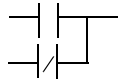
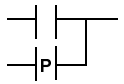
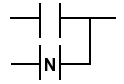
Instrucción O lógico

Función En la siguiente tabla se describe la función de cada instrucción.

Lenguaje de contactos	Lista de instrucciones	Literal estructurado	Descripción	Cronograma
	OR	OR	O lógico entre el operando y el resultado booleano de la instrucción anterior	
	ORN	OR (NOT...)	O lógico entre el operando inverso y el resultado booleano de la instrucción anterior	
	ORR	OR (RE...)	O lógico entre el flanco ascendente del operando y el resultado booleano de la instrucción anterior	
	ORF	OR (FE...)	O lógico entre el flanco descendente del operando y el resultado booleano de la instrucción anterior	

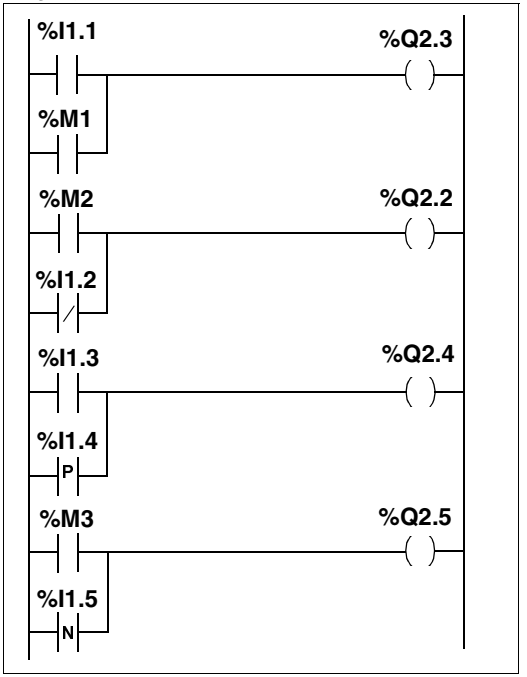
Operandos permitidos

En la siguiente tabla figura la lista de los operandos utilizados para estas instrucciones.

Lenguaje de contactos	Lista de instrucciones	Literal estructurado	Operandos
	OR	OR	%I, %Q, %M, %S, %BLK, %•:Xk, %Xi True (1)/False (0) en lenguaje lista de instrucciones o literal estructurado
	ORN	OR (NOT...)	%I, %Q, %M, %S, %BLK, %•:Xk, %Xi True (1)/False (0) en lenguaje lista de instrucciones o literal estructurado
	ORR	OR (RE...)	%I, %Q, %M
	ORF	OR (FE...)	%I, %Q, %M

Ejemplo en el lenguaje de contactos

El siguiente ejemplo muestra la programación de las instrucciones O lógico en el lenguaje de contactos.



Ejemplo en lista de instrucciones

El siguiente ejemplo muestra la programación de las instrucciones O lógico en el lenguaje lista de instrucciones.

```
LD %I1.1
OR %M1
ST %Q2.3

LD %M2
ORN %I1.2
ST %Q2.2

LD %I1.3
ORR %I1.4
ST %Q2.4

LD %M3
ORF %I1.5
ST %Q2.5
```

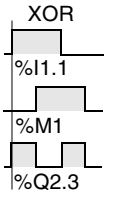
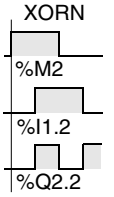
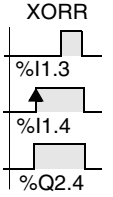
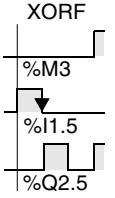
Ejemplo en lenguaje literal estructurado

El siguiente ejemplo muestra la programación de las instrucciones O lógico en el lenguaje literal estructurado.

```
%Q2.3:=%I1.1 OR %M1;
%Q2.2:=%M2 OR (NOT%I1.2);
%Q2.4:=%I1.3 OR (RE%I1.4);
%Q2.5:=%M3 OR (FE%I1.5);
```

Instrucción O exclusivo

Función En la siguiente tabla se describe la función de cada instrucción.

Lista de instrucciones	Literal estructurado	Descripción	Cronograma
XOR	XOR	O exclusivo entre el operando y el resultado booleano de la instrucción anterior	
XORN	XOR (NOT...)	O exclusivo entre el operando inverso y el resultado booleano de la instrucción anterior	
XORR	XOR (RE...)	O exclusivo entre el flanco ascendente del operando y el resultado booleano de la instrucción anterior	
XORF	XOR (FE...)	O exclusivo entre el flanco descendente del operando y el resultado booleano de la instrucción anterior.	

Nota: No hay elementos gráficos específicos para el O exclusivo en el lenguaje de contactos. No obstante, el O exclusivo puede programarse utilizando una combinación de contactos de apertura y de cierre (véase el ejemplo siguiente).

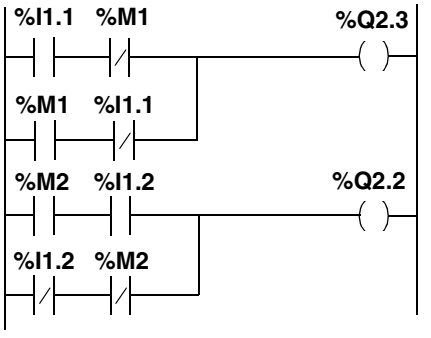
Operandos permitidos

En la siguiente tabla figura la lista de los operandos utilizados para estas instrucciones.

Lista de instrucciones	Literal estructurado	Operandos
XOR	XOR	%I, %Q, %M, %S, %BLK, %•:Xk, %Xi
XORN	XOR (NOT...)	%I, %Q, %M, %S, %BLK, %•:Xk, %Xi
XORR	XOR (RE...)	%I, %Q, %M
XORF	XOR (FE...)	%I, %Q, %M

Ejemplo en el lenguaje de contactos

El siguiente ejemplo muestra la programación de las instrucciones O exclusivo en el lenguaje de contactos.



Ejemplo en lista de instrucciones

El siguiente ejemplo muestra la programación de las instrucciones O exclusivo en el lenguaje lista de instrucciones.

```
LD %I1.1
XOR %M1
ST %Q2.3

LD %M2
XORN %I1.2
ST %Q2.2

LD %I1.3
XORR %I1.4
ST%Q2.4

LD %M3
XORF %I1.5
ST %Q2.5
```

Ejemplo en lenguaje literal estructurado

El siguiente ejemplo muestra la programación de las instrucciones O exclusivo en el lenguaje literal estructurado:

```
%Q2.3:=%I1.1 XOR%M1;
%Q2.2:=%M2 XOR (NOT%I1.2);
%Q2.4:=%I1.3 XOR (RE%I1.4)
%Q2.5:=%M3 XOR (FE%I1.5);
```

Nota: Los paréntesis son opcionales, pero facilitan la lectura del programa.

1.3 Bloque de función predefinidos

Presentación

Objeto de este apartado

En este apartado se describen los bloques de función predefinidos del lenguaje PL7

Contenido

Esta sección contiene los siguientes apartados:

Presentación del bloque de función del temporizador %TMi

Generalidades

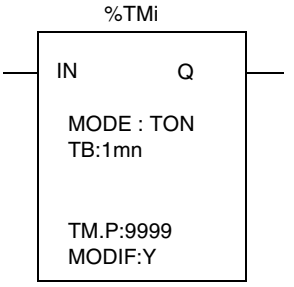
El temporizador tiene 3 modos de funcionamiento:

- **TON**: permite gestionar retardos en la conexión,
- **TOF**: permite gestionar retardos en la desconexión,
- **TP**: permite elaborar un impulso de duración precisa.

Los retardos o las duraciones de impulso se pueden programar y modificar o no a través del terminal.

Figura

La representación gráfica del bloque de función del temporizador es la siguiente:



Características El temporizador posee las siguientes características:

Característica	Variable	Valor
Número de temporizador	%TMi	0 a 63 para un TSX 37, 0 a 254 para un TSX 57
Modo	TON	• retardo en la conexión (por defecto)
	TOF	• retardo en la desconexión
	TP	• monoestable
Base de tiempo	TB	1mn (por defecto), 1s, 100ms, 10ms (16 temporizadores como máx. a 10ms). Cuanto más baja sea la base de tiempo, más elevada será la precisión del temporizador.
Valor actual	%TMi.V	Palabra que aumenta de 0 a %TMi.P al terminar el temporizador. Puede leerse y probarse, pero no escribirse por programa (%TMi.V puede modificarse mediante terminal).
Valor de preselección	%TMi.P	0-%TMi.P-9999. Palabra que puede leerse, probarse y escribirse por programa. Pasa al valor 9999 por defecto. La duración o el retardo elaborado es igual a %TMi.P x TB.
Ajuste por terminal (MODIF)	Y/N	Y: posibilidad de modificar el valor de preselección %TMi.P en ajuste. N: sin acceso al ajuste.
Entrada (instrucción) "Activación"	IN	En el flanco ascendente (modo TON o TP) o flanco "Activación" descendente (modo TOF), se inicia el temporizador.
Salida "Temporizador"	Q	Bit asociado %TMi.Q, su puesta en 1 depende de la función realizada TON, TOF o TP.

Modo de funcionamiento del bloque temporizador %TMi

Descripción

En la siguiente tabla se describen los modos de funcionamiento específicos del bloque temporizador.

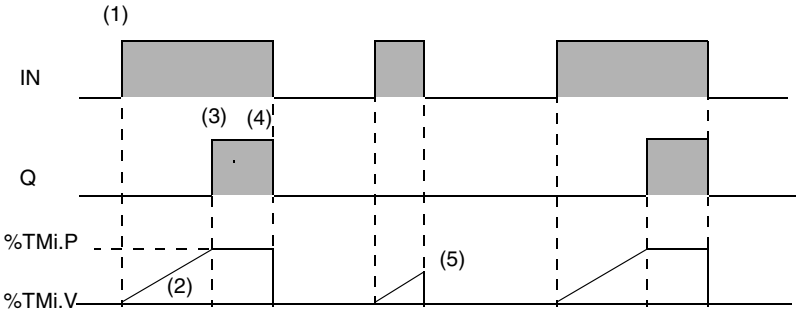
Incidencia...	Descripción
de un rearranque en frío	(%S0=1), provoca la puesta a 0 del valor actual, la puesta a 0 de la salida %TMi.Q y el valor de preselección se reinicializa al valor definido en la configuración.
de un rearranque en caliente	(%S1=1) no tiene incidencia sobre el valor actual del temporizador ni sobre el valor de preselección. El valor actual no evoluciona mientras dura el corte de corriente.
de un paso a parada, desactivación de una tarea o ejecución de un punto de parada	no fija el valor actual.
de un salto de programa	El hecho de no explorar las instrucciones donde está programado el bloque temporizador no fija el valor actual %TMi.V, que continúa aumentando hacia %TMi.P. De la misma forma, el bit %TMi.Q asociado a la salida Q del bloque temporizador conserva su funcionamiento normal y puede probarse por otra instrucción. Por el contrario, la salida, directamente conectada a la salida del bloque, no se activa, ya que el autómata no la explora.
de la modificación de la preselección	La modificación del valor de preselección por instrucción o ajuste sólo se tiene en cuenta en la siguiente activación del temporizador. La modificación del valor de preselección en el editor de variables sólo se tiene en cuenta después de un rearranque en frío (%S0=1).

Nota: se recomienda probar el bit %TMi.Q únicamente una sola vez en el programa.

Funcionamiento del bloque de función del temporizador %TMi en modo TON

Generalidades El funcionamiento del temporizador en modo TON permite gestionar retardos en la conexión.

Figura El cronograma muestra el funcionamiento del temporizador en modo TON.



Funcionamiento En la siguiente tabla se describe el funcionamiento del temporizador en modo TON.

Fase	Descripción
1	En un flanco ascendente en la entrada IN, el temporizador se inicia
2	El valor actual %TMi.V del temporizador aumenta de 0 hacia %TMi.P de una unidad en cada impulso de la base de tiempo TB
3	El bit de salida %TMi.Q pasa a 1 en el momento en el que el valor actual alcanza %TMi.P
4	El bit de salida %TMi.Q permanece en 1 mientras la entrada IN esté en 1.
5	Cuando la entrada IN está en 0, el temporizador se detiene aunque estuviera en curso de evolución: %TMi.V toma el valor 0.

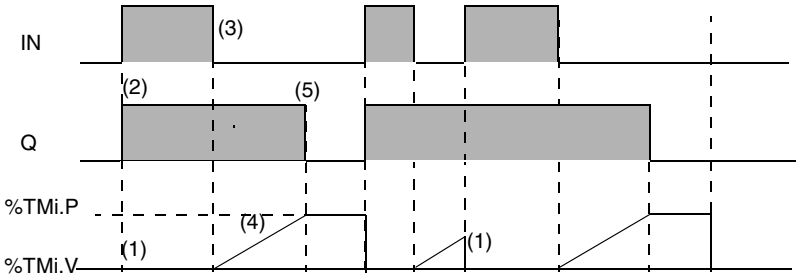
Funcionamiento del bloque de función del temporizador %TMi en modo TOF

Generalidades

El funcionamiento del temporizador en modo TOF permite gestionar retardos en la desconexión.

Figura

El cronograma muestra el funcionamiento del temporizador en modo TOF.



Funcionamiento

En la siguiente tabla se describe el funcionamiento del temporizador en modo TOF.

Fase	Descripción
1	El valor actual %TMi.V toma el valor 0 en un flanco ascendente de la entrada IN (aunque el temporizador esté en curso de evolución)
2	El bit de salida %TMi.Q pasa a 1.
3	En un flanco descendente en la entrada IN, el temporizador se inicia.
4	El valor actual aumenta hacia %TMi.P de una unidad en cada impulso de la base de tiempo TB.
5	El bit de salida %TMi.Q vuelve a 0 cuando el valor actual alcanza %TMi.P

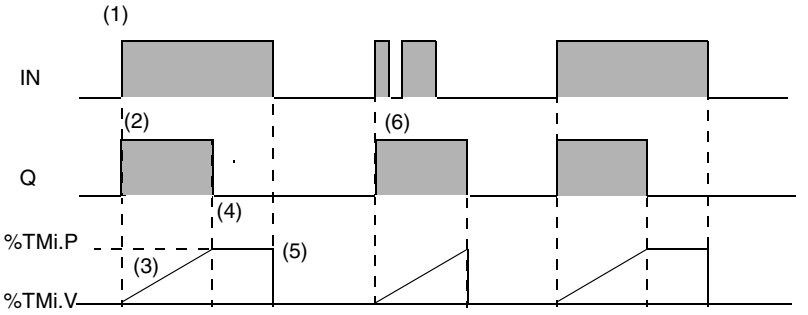
Funcionamiento del bloque de función del temporizador %TMi en modo TP

Generalidades

El funcionamiento del temporizador en modo TP permite elaborar un impulso de duración precisa (función monoestable).

Figura

El cronograma muestra el funcionamiento del temporizador en modo TP.



Funcionamiento

En la siguiente tabla se describe el funcionamiento del temporizador en modo TP.

Fase	Descripción
1	En un flanco ascendente en la entrada IN, el temporizador se inicia
2	El bit de salida %TMi.Q pasa a 1.
3	El valor actual %TMi.V del temporizador aumenta de 0 hacia %TMi.P de una unidad en cada impulso de la base de tiempo TB
4	El bit de salida %TMi.Q vuelve a 0 cuando el valor actual alcanza %TMi.P.
5	Cuando la entrada IN y la salida %TMi.Q están en 0, %TMi.V toma el valor 0.
6	Este monoestable no se puede reactivar.

Programación y configuración de los bloques de función del temporizador

Generalidades

La programación de los bloques de función del temporizador es idéntica independientemente del modo de utilización seleccionado.

La elección del funcionamiento TON, TOF o TP se realiza en el editor de variables.

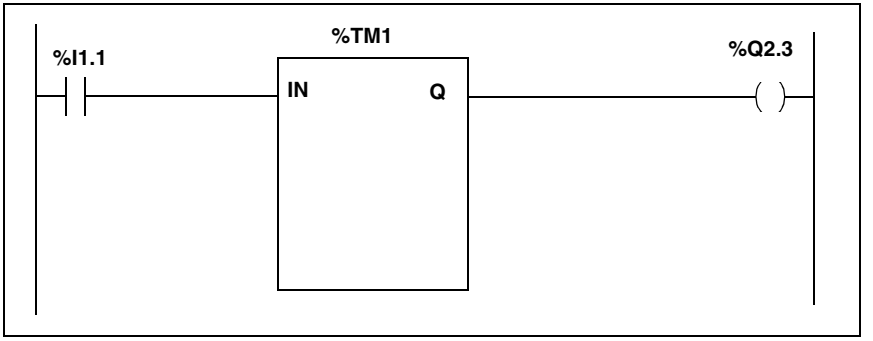
Configuración

Consiste en determinar los parámetros siguientes :

Parámetro	Valores
Modo	TON, TOF o TP.
TB	1min, 1s, 100ms o 10ms
%TMi.P	0 a 9999
MODIF	Y o N

Programación en el lenguaje de contactos

El siguiente programa muestra la utilización de un bloque de función de temporizador en lenguaje de contactos.



Programación en lista de instrucciones

El siguiente programa muestra la utilización de un bloque de función de temporizador en lenguaje lista de instrucciones.

```
LD      %I1.1
IN      %TM1
LD      %TM1.Q
ST      %Q2.3
```

Programación en literal estructurado

El siguiente programa muestra la utilización de un bloque de función de temporizador en lenguaje literal estructurado.

```
IF RE %I1.1 THEN
    START %TM1;
ELSIF FE %I1.1 THEN
    DOWN %TM1;
END_IF;
%Q2.3 := %TM1.Q;
```

La instrucción `START %TMi` genera un flanco ascendente en la entrada IN del bloque del temporizador.

La instrucción `DOWN %TMi` genera un flanco descendente en la entrada IN del bloque del temporizador.

Casos específicos del funcionamiento del temporizador serie 7

Casos específicos

- **Incidencia de un "rearranque en frío"**: (%S0 = 1) provoca la carga del valor de preselección (definido por el editor de variables) en el valor actual y la puesta a 0 de la salida %Ti.D; el valor de preselección eventualmente modificado por el terminal se pierde.
- **Incidencia de un "rearranque en caliente"**: (%S1=1) no tiene ninguna incidencia en el valor actual del temporizador.
- **Incidencia de un paso a stop**: el paso a stop del autómata no fija el valor actual. Ocurre lo mismo cuando la tarea en curso se desactiva o cuando se ejecuta un punto de parada.
- **Incidencia de un salto de programa**: El hecho de no explorar la red donde está programado el bloque temporizador no fija el valor actual %Ti.V, que continúa disminuyendo hacia 0. De igual forma, los bits %Ti.D asociados a las salidas D y R del bloque temporizador conservan su funcionamiento normal y pueden por lo tanto probarse en otra red. Por el contrario, las bobinas directamente "conectadas" a la salida del bloque no se activan, puesto que el autómata no las explora.
- **Prueba de los bits %Ti.D y %Ti.R**: estos bits pueden cambiar de estado en el transcurso de un ciclo.

Presentación del bloque de función contador-descontador

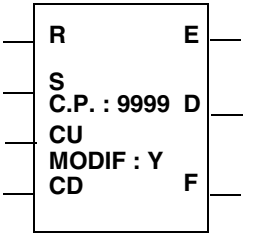
Generalidades

Este bloque de función permite:
- El contaje de sucesos
- El descontaje de sucesos

Estas operaciones pueden ser simultáneas

Figura

Representación gráfica del bloque de función contador-descontador



Características El contador-descontador presenta las siguientes características:

Características	Variable	Valor
Número del contador	%Ci	0 a 31 para un TSX 37, 0 a 254 para un TSX 57
Valor actual	%Ci.V	Palabra incrementada o disminuida en función de las entradas CU y CD. Puede leerse y probarse, pero no escribirse mediante programa. Puede modificarse mediante terminal.
Valor de preselección	%Ci.P	$0 \leq \%Ci.P \leq 9999$. Palabra que puede leerse, probarse y escribirse (valor 9999 por defecto).
Ajuste por terminal (MODIF)	Y/N	• Y : posibilidad de modificar el valor de preselección en ajuste. • N : sin acceso al ajuste.
Entrada (instrucción) Reset	R	En estado 1: %Ci.V = 0
Entrada (instrucción) Preselección	S	En estado 1: %Ci.V = %Ci.P
Entrada (instrucción) Contaje	CU	Incrementa %Ci.V en el flanco ascendente
Entrada (instrucción) Descontaje	CD	Disminuye %Ci.V en el flanco ascendente
Salida Rebasamiento	E (Empty)	El bit asociado %Ci.E=1, cuando %Ci.V pasa de 0 a 9999 (se pone a 1 cuando %Ci.V es igual a 9999) vuelve a 0 si el contador sigue descontando. Cuando se produce un rebasamiento, el bit %S18 pasa a 1
Salida Preselección alcanzada	D (Done)	El bit asociado %Ci.D=1, cuando %Ci.V=%Ci.P.
Salida Rebasamiento	F (Full)	El bit asociado %Ci.F, cuando %Ci.V pasa de 9999 a 0 (se pone a 1 cuando %Ci.V es igual a 0) vuelve a 0 si el contador sigue descontando. Cuando se produce un rebasamiento, el bit %S18 pasa a 1

Funcionamiento del bloque de función Contador/Descontador

Funcionamiento

Función Contaje

Acción	Resultado
Aparece un flanco ascendente en la entrada de contaje CU	El valor actual %Ci.V se incrementa de una unidad
El valor actual %Ci.V es igual al valor de preselección %Ci.P	El bit de salida %Ci.D "preselección alcanzada" asociado a la salida D pasa al estado 1
El valor actual %Ci.V pasa de 9999 a 0	El bit de salida %Ci.F (rebasamiento de contaje) pasa al estado 1
El contador sigue contando	El bit de salida %Ci.F (rebasamiento de contaje) vuelve a 0

Función Descontaje

Acción	Resultado
Aparece un flanco ascendente en la entrada de descontaje CD	El valor actual %Ci.V disminuye de una unidad
El valor actual %Ci.V pasa de 0 a 9999	El bit de salida %Ci.E (rebasamiento de descontaje) pasa al estado 1
El contador sigue descontando	El bit de salida %Ci.E (rebasamiento de descontaje) vuelve a 0

Función Contaje/Descontaje

Acción	Resultado
Aparece un flanco ascendente en la entrada de contaje CU	El valor actual %Ci.V se incrementa de una unidad
Aparece un flanco ascendente en la entrada de descontaje CD	El valor actual %Ci.V disminuye de una unidad
Las dos entradas están en 1 simultáneamente	El valor actual no cambia

Reset

Cuando	Resultado
La entrada R se pone a 1 (esta entrada tiene prioridad sobre el resto)	El valor actual %Ci.V se fuerza a 0. Las salidas %Ci.V, %CI.D y %CI.F están en 0

Preselección

Acción	Resultado
La entrada S "Preselección" se encuentra en el estado 1 y la entrada R "Reset"	El valor actual %Ci.V toma el valor %Ci.P y la salida %Ci.D pasa a 1

Observación

En la puesta a 0 (entrada R o instrucción R):

- En el lenguaje de contactos, los historiales de las entradas CU y CD se actualizan con los valores conectados.
- En los lenguajes lista de instrucciones y literal estructurado, los historiales de las entradas CU y CD no se actualizan; cada entrada conserva el valor que tenía antes de la llamada.

Casos específicos

Diferentes casos específicos

Acción	Resultado
• Rearranque en frío (%S0=1)	• El valor actual %Ci.V se pone a cero • Los bits de las salidas %Ci.E, %Ci.D y %Ci.F se ponen a cero • El valor de preselección se inicializa al valor definido en la configuración
• Rearranque en caliente (%S1=1) • Paso a stop • Desactivación de una tarea • Ejecución de un punto de parada	• Ninguna incidencia en el valor actual del contador (%Ci.V)
• Modificación de la preselección %Ci.P	• La modificación del valor de preselección mediante instrucción o ajuste se toma en cuenta en la gestión del bloque por parte de la aplicación (activación de una de las entradas)

Configuración y programación

Ejemplo

Contaje de un número de piezas = 5000. Cada impulso en la entrada %I1.2 (cuando el bit interno %M0 está en 1) provoca el incremento del contador %C8 hasta el valor de preselección final del contador %C8 (bit %C8.D=1). La entrada %I1.1 provoca el reset del contador.

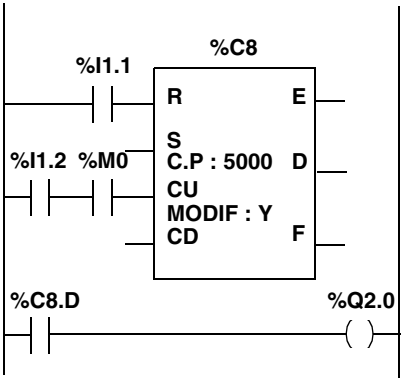
Configuración

A continuación, se indican los parámetros que debe introducir el editor de variables:

- %Ci.P, fijado a 5000 en este ejemplo
- MODIF: Y

Programación

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD %I1.1
R   %C8
LD  %I1.2
AND %M0
CU  %C8
LD  %C8.D
ST  %Q2.0
```

Lenguaje literal estructurado

```
IF %I1.1 THEN
    RESET %C8
END_IF;
%M1:=%I1.2 THEN
    UP %C8;
END_IF;
%Q2.0 :=%C8.D;
```

En el lenguaje literal estructurado, 4 instrucciones permiten programar los bloques de función del contador/descontador:

- **RESET** %Ci: Reset del valor actual
- **PRESET** %Ci: Carga del valor de preselección en el valor actual
- **UP** %Ci: Incrementa el valor actual
- **DOWN** %Ci: Disminuye el valor actual

En el caso del lenguaje literal estructurado, se hace un reset del historial de las entradas CU y CD cuando se utilizan las instrucciones UP y DOWN. Por lo tanto, es el usuario quien debe gestionar los flancos ascendentes para estas dos instrucciones.

1.4 Tratamientos numéricos en enteros

Presentación

Objeto de este apartado

En este apartado se describen los tratamientos numéricos en enteros del lenguaje PL7

Contenido

Esta sección contiene los siguientes apartados:

Presentación de los tratamientos numéricos en enteros

Generalidades

Las instrucciones numéricas descritas en este capítulo se aplican a los objetos de tipo:
- tablas de bits
- palabras
- palabras dobles

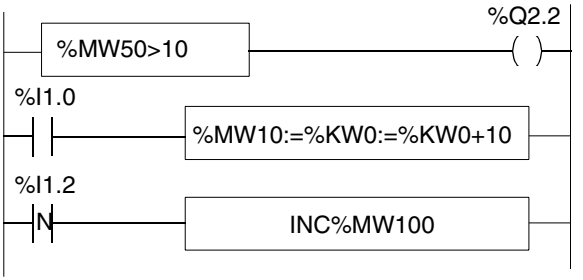
Las instrucciones sobre los demás tipos de objetos se describen en el capítulo "Instrucciones avanzadas (Véase *Instrucciones avanzadas, p. 87*)".

Programación en el lenguaje de contactos

Las instrucciones numéricas se introducen en bloques:
- en la zona de prueba para los bloques de comparación
- en la zona de acción para los bloques de operaciones

Dichos bloques pueden contener:
- una expresión de formato simple; ej: OP3:=OP1+OP2
- una expresión de formato complejo; ej: OP5:=(OP1+OP2)*OP3-OP4.

Ejemplo de programación:



Programación en lenguaje lista de instrucciones

Las instrucciones se escriben entre corchetes.
Se ejecutan si el resultado booleano de la instrucción de prueba que precede a la instrucción numérica está en 1.

Ejemplo de programación:

```
LD      [%MW50>10]
ST      %Q2.2
LD      %I1.0
[%MW10:=%KW0+10]
LDF     %I1.2
[INC%MW100]
```

Programación en el lenguaje literal estructurado

Las instrucciones numéricas se introducen directamente.
La instrucción condicional IF permite condicionar estas instrucciones numéricas mediante una expresión booleana.

Ejemplo de programación:

```
%Q2.2:=%MW50 > 10;
IF %I1.0 THEN
   %MW10:=%KW0 + 10;
END_IF;
IF FE %I1.2 THEN
   INC %MW100;
END_IF;
```

Lista de operandos

Lista de las tablas de bits

Abreviaturas	Direccionamiento completo	Tipo de palabra	Acceso
%M:L	%Mi:L	tabla de bits internos	R/W
%I:L	%Ixy.i:L	tabla de bits de entrada	R/W
%Q:L	%Qxy.i:L	tabla de bits de salida	R/W
	%Xi:L o %Xj.i:L	tabla de bits de etapas	R

Lista de palabras de formato simple

Abreviaturas	Direccionamiento completo	Tipo de palabra	Acceso	Forma indexada
Valor inm.	-	valores inmediatos	R	-
%MW	%MWi	palabra interna	R/W	%MWi[index]
%KW	%KWi	constante interna	R	%KWi[index]
%SW	%SWi	palabra de sistema	R/W (1)	-
%IW	%IWxy.i(.r)	palabra de entrada	R	-
%QW	%QWxy.i(.r)	palabra de salida	R/W	-
%NW	%NW{j}k	palabra común	R/W	-
%BLK	ej: %TMi.P	palabra extraída de bloque de función estándar o de bloque de función	R/W (2)	-
%Xi.T	%Xi.T o %Xj.i.T	tiempo de actividad de etapa	R	%Xi.T[índice]

(1) escritura según i.
(2) escritura según el tipo de palabra, por ejemplo: los valores de preselección (%Ci.P pueden escribirse, mientras que los valores actuales %Ci.V sólo se pueden leer).

Lista de palabras dobles

Abreviaturas	Direccionamiento completo	Tipo de palabra	Acceso	Forma indexada
Valor inm.	-	valores inmediatos	R	-
%MD	%MDi	palabra doble interna	R/W	%MDi[index]
%KD	%KDi	constante doble interna	R	%KDi[index]
%SD	%SDi	palabra de sistema doble	R/W (1)	-
%ID	%IDxy.i(.r)	palabra doble de entrada	R	-
%QD	%QDxy.i(.r)	palabra doble de salida	R/W	-

(1) únicamente palabra doble %SD18

Nota: Existen otros tipos de palabras dobles, como %MWxy.i %KWxy.i y %MDxy.i %KDxy.i asociadas a las funciones específicas, que se comportan respectivamente como las palabras y palabras dobles %MWi %KWi y %MDi %KDi.

Nota: Conversiones implícitas palabras <--> palabras dobles
El programa PL7 permite combinar operaciones que utilicen palabras y palabras dobles. Las conversiones en uno u otro de los formatos se efectúa de forma implícita. Una operación en la que interviene una palabra doble o varios valores inmediatos se ejecuta de forma interna automáticamente en formato doble.

Instrucciones de comparación

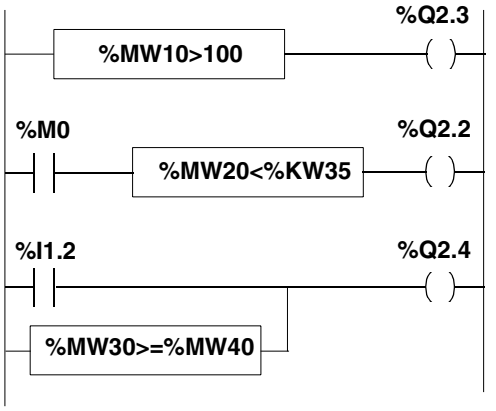
Generalidades

Las instrucciones de comparación permiten comparar dos operandos.

- **>** : prueba si el operando 1 es superior al operando 2,
- **>=** : prueba si el operando 1 es superior o igual al operando 2,
- **<** : prueba si el operando 1 es inferior al operando 2,
- **<=** : prueba si el operando 1 es inferior o igual al operando 2,
- **=** : prueba si el operando 1 es diferente del operando 2.

Estructura

Lenguaje de contactos



Nota: Los bloques de comparación se programan en la zona de prueba.

Lenguaje lista de instrucciones

```
LD   [%MW10>100]
ST   %Q2.3
LD %M0
AND  [%MW20<%KW35]
ST   %Q2.2
LD   %I1.2
OR   [%MW30>=%MW40]
ST   %Q2.4
```

Nota: La comparación se efectúa entre corchetes que figuran a continuación de las instrucciones LD, AND y OR.

Lenguaje literal estructurado

```
%Q2.3:=%MW10>100;
%Q2.2:=%M0 AND (%MW20<%KW35);
%Q2.4:=%I1.2 OR (%MW30>=%MW40);
```

Nota: Los paréntesis son opcionales, pero facilitan la lectura del programa.

Sintaxis

Operadores de instrucciones de comparación

Operadores	**Sintaxis**
>,>=,<,<=,=,<>	Op1 Operador Op2

Operandos

Tipo	**Operandos 1 y 2 (Op1 y Op2)**
Palabras indexables	%MW,%KW,%Xi.T
Palabras no indexables	Val.inm.,%IW,%QW,%SW,%NW,%BLK,Expr. numérica
Palabras dobles indexables	%MD,%KD
Palabras dobles no indexables	Val.inm.,%ID,%QD,%SD,Expr.numérica

Nota:
- en el lenguaje de contactos, la operación de comparación puede efectuarse también con el Bloque de comparación vertical (Véase *Presentación del bloque de operación comparador vertical, p. 120*)
- en el lenguaje lista de instrucciones, las instrucciones de comparación se pueden utilizar entre paréntesis.

Instrucciones de asignación

Generalidades

Realizan la carga de un operando Op2 en un operando Op1
Las operaciones de asignación pueden efectuarse:
- en tablas de bits,
- en palabras o palabras dobles.

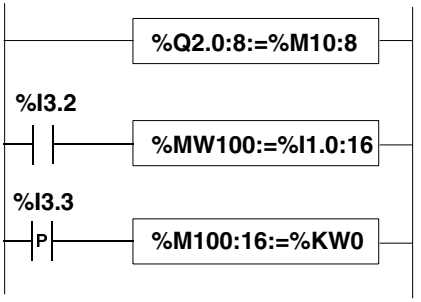
En un mismo bloque pueden encadenarse varias instrucciones de asignación:
Op1:=Op2:=Op3:=Op4:=...

Asignación de tablas de bits

Se pueden realizar las siguientes operaciones en tablas de bits (("Objetos PL7 de tipo tabla" - Manual de referencia Tomo 1)):
- tabla de bits -> tabla de bits (ej: 1)
- tabla de bits -> palabra o palabra doble (indexada) (ej: 2)
- palabra o palabra doble (indexada) -> tabla de bits (ej: 3)

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones
Ejemplo 1:

```
LD TRUE
[%Q2.0:8]
```

Ejemplo 2:

```
LD %I3.2
[%MW100:=%I1.0:16]
```

Ejemplo 3:

```
LDR %I3.3
[%MW100:16=%KW0]
```

Lenguaje literal estructurado
Ejemplos 1 y 2:

```
%Q2.0:8:=%M10:8;
IF %I3.2 THEN
  %MW100:=%I1.0:16;
END_IF;
```

Ejemplo 3:

```
IF RE %I3.3 THEN
  %M100:16:=%KW0;
END_IF;
```

Sintaxis

Operador y sintaxis

Operador	Sintaxis
:=	Op1:=Op2

Operandos

Tipo	Operando 1 (Op1)	Operando 2 (Op2)
Tabla de bits	%M:L,%Q:L,%I:L	%M:L,%Q:L,%I:L,%Xi:L
Palabras indexables	%MW	%MW,%KW,%Xi.T
Palabras no indexables	%QW,%SW,%NW,%BLK	Val.inm.,%IW,%QW,%SW,%NW,%BLK,Expr.num.
Palabras dobles indexables	%MD	%MD,%KD
Palabras dobles no indexables	%QD,%SD	Val.inm.,%ID,%QD,%SD,Expr. numérica

Reglas de utilización

- las tablas de bits de origen y de destino no tienen obligatoriamente la misma longitud. En caso de que la tabla de origen sea más larga que la de destino, sólo se transferirán los bits de peso menos significativo. En caso contrario, la tabla de destino se completa con 0.
- Caso de una asignación tabla de bits -> palabra (o palabra doble): los bits de la tabla se transfieren a la palabra (de peso menos significativo para una palabra doble) comenzando por la derecha (primer bit de la tabla en el bit 0 de la palabra), los bits de la palabra no afectados por la transferencia (longitud<16 ó 32) se sitúan en 0.
- Caso de una asignación palabra -> tabla de bits: los bits de la palabra se transfieren a partir de la derecha (el bit 0 de la palabra en el primer bit de la tabla)

Asignación de palabras

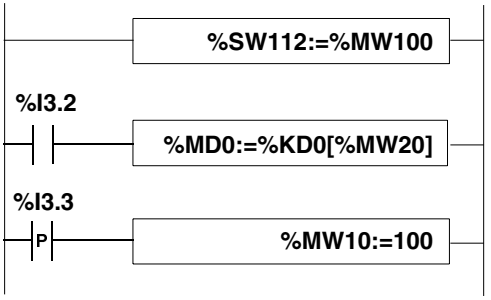
Generalidades

Se pueden llevar a cabo las siguientes operaciones de asignación en palabras:

- palabra (indexada) -> palabra (indexada) o palabra doble (indexada) (ej: 1)
- palabra doble (indexada) -> palabra doble (indexada) o palabra (indexada) (ej: 2)
- valor inmediato -> palabra (indexada) o palabra doble (indexada) (ej: 3)

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones
Ejemplo 1:

```
LD TRUE
[%SW112:=%MW100]
```

Ejemplo 2:

```
LD %I3.2
[%MD10:=%KD0[%MW20]]
```

Lenguaje literal estructurado
Ejemplo 3:

```
IF %I3.3 THEN
  %MW10:=100;
END_IF;
```

Sintaxis

Operador y sintaxis

Operador	**Sintaxis**
:=	Op1:=Op2

Operandos

Tipo	**Operando 1 (Op1)**	**Operando 2 (Op2)**
Palabras indexables	%MW	%MW,%KW,%Xi.T
Palabras no indexables	%QW,%SW,%NW,%BLK	Val.inm.,%IW,%QW,%SW,%NW, %BLK,Expr.num.
Palabras dobles indexables	%MD	%MD,%KD
Palabras dobles no indexables	%QD,%SD	Val.inm.,%ID,%QD,%SD, Expr. numérica

Nota: Las conversiones palabra <--> palabra doble se efectúan de forma implícita; en la asignación palabra doble --> palabra, si el valor de l apalabra doble no puede incluirse en la palabra, el bit %S18 se sitúa en 1.
Es posible realizar asignaciones múltiples. Ejemplo: `%MW0:=%MW2:=%MW4`
Atención: en el ejemplo `%MD14:=%MW10:=%MD12`, no se obtiene obligatoriamente `%MD14:=%MD12`, ya que en la asignación a %MW10, se pierden pesos más significativos de la palabra doble debido a la conversión palabra doble-palabra simple.

Instrucciones aritméticas en enteros

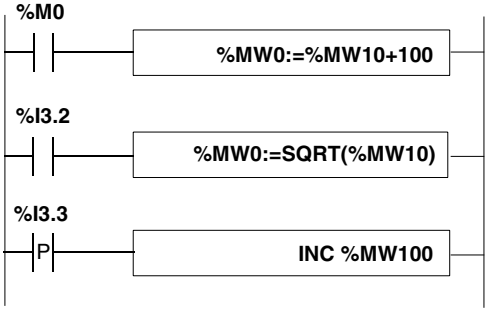
Generalidades

Estas instrucciones permiten efectuar una operación aritmética entre dos operandos o en un operando.
Lista de instrucciones:

+	suma de dos operandos	**SQRT**	raíz cuadrada de un operando
-	resta de dos operandos	**INC**	incremento de un operando
*****	multiplicación de dos operandos	**DEC**	decremento de un operando
/	división de dos operandos	**ABS**	valor absoluto de un operando
REM	resto de la división de 2 operandos		

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD %M0
[%MW0:=%MW10+100]

LD %I3.2
[%MW0:=SQRT(%MW10)]

LD  %I3.3
[INC %MW100]
```

Lenguaje literal estructurado

```
IF %M0 THEN
  %MW0:=%MW10+100;
END_IF;
IF %I3.2 THEN
  %MW0:=SQRT(%MW10);
END_IF;
IF RE %I1.3 THEN
  INC %MW100;
END_IF
```

Sintaxis

Operador y sintaxis

Operador	**Sintaxis**
+,**-**,*****,**/**,**REM**	Op1:=Op2 Operador Op3
SQRT,**ABS**	Op1:=Operador(Op2)
INC,**DEC**	Operador Op1

Operandos

Tipo	**Operando 1 (Op1)**	**Operando 2 (Op2)**
Palabras indexables	%MW	%MW,%KW,%Xi.T
Palabras no indexables	%QW,%SW,%NW,%BLK	Val.inm.,%IW,%QW,%SW,%NW, %BLK,Expr.num.
Palabras dobles indexables	%MD	%MD,%KD
Palabras dobles no indexables	%QD,%SD	Val.inm.,%ID,%QD,%SD, Expr. numérica

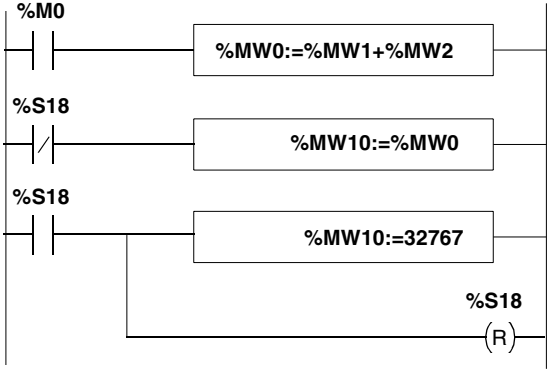
Nota: Las operaciones INC y DEC no se pueden utilizar en expresiones numéricas.

Reglas de utilización

- **Suma: rebasamiento de capacidad durante la operación**
  En caso de que el resultado supere los límites:
  - -32768 o +32767 para un operando de longitud simple,
  - -2.147.483.648 o +2.147.483.647 para un operando de longitud doble.

  El bit %S18 (rebasamiento) pasa al estado 1. Por lo tanto, el resultado no es significativo.
  La gestión del bit %S18 se lleva a cabo mediante programa de usuario:

Ejemplo en el lenguaje de contactos:



Ejemplo en lenguaje lista de instrucciones:

```
LD    %M0
[%MW0:=%MW1+%MW2]
LDN   %S18
[%MW10:=%MW0]
LD    %S18
[%MW10:=32767]
R     %S18]
```

Ejemplo en lenguaje literal estructurado:

```
IF %M0 THEN
   %M0:=%MW1+%MW2;
END_IF;
IF %S18 THEN
   %MW10:=32767;RESET %S18;
ELSE
   %MW10:=%MW0;
END_IF;
```

En caso de que %MW1 =23241 y %MW2=21853, el resultado real (45094) no pueda expresarse en una palabra de 16 bits, el bit %S18 pasa al estado 1 y el resultado obtenido (-20442) es erróneo. En este ejemplo, cuando el resultado es superior a 32767, su valor se fija a 32767.

- **Multiplicación:**
  Rebasamiento de capacidad durante la operación.
  En caso de que el resultado supere la capacidad de la palabra de colocación, el bit %S18 (rebasamiento) pasa al estado 1 y el resultado no es significativo.
- **División/resto de la división:**
  División por 0.
  En caso de que el divisor sea igual a 0, la división es imposible y el bit de sistema %S18 pasa al estado 1, por lo que el resultado será erróneo.
  Rebasamiento de capacidad durante la operación.
- **Extracción de la raíz cuadrada:**
  La extracción de la raíz cuadrada sólo se efectúa con valores positivos. El resultado es por lo tanto siempre positivo. En caso de que el operando de la raíz cuadrada sea negativo, el bit de sistema %S18 pasa al estado 1 y el resultado es erróneo.

Nota:
- Cuando el resultado de una operación no es un entero (caso de una división o de una raíz cuadrada), el resultado se trunca (redondeo al entero inferior más próximo).
- El signo del resto de la división (REM) es el del numerador.
- La gestión del bit de sistema %S18 corre a cargo del programa de usuario. El autómata lo pone a 1 y el programa debe volverlo a poner a cero para poder utilizarlo de nuevo (véase el ejemplo anterior).

Instrucciones lógicas

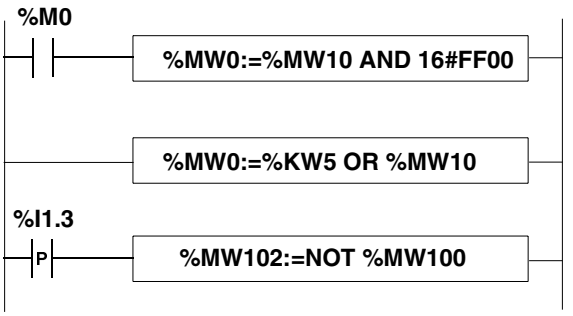
Generalidades

Las instrucciones asociadas permiten efectuar una operación lógica entre dos operandos o en un operando.
Lista de instrucciones:

AND	Y (bit a bit) entre dos operandos
OR	O lógico (bit a bit) entre dos operandos
XOR	O exclusivo (bit a bit) entre dos operandos
NOT	complemento lógico (bit a bit) de un operando

Estructura

Lenguaje de contactos:



Lenguaje lista de instrucciones:

```
LD    %M0
[%MW0:=%MW10 AND 16#FF00]

LD TRUE
[%MW0:=%KW5 OR %MW10]

LD %I1.3
[%MW102:=NOT%MW100]
```

Lenguaje literal estructurado:

```
IF %M0 THEN
  %MW0:=%MW10 AND 16#FF00;
END_IF;
%MW0:=%KW5 OR %MW10;
IF %I1.3 THEN
  %MW102:=NOT %MW100;
END_IF;
```

Sintaxis

Operador y sintaxis

Operador	**Sintaxis**
AND,**OR**,**XOR**	Op1:=Op2 Operador Op3
NOT	Op1:=NOT Op2

Operandos

Tipo	**Operando 1 (Op1)**	**Operandos 2 y 3 (Op2 y Op3)**
Palabras indexables	%MW	%MW,%KW,%Xi.T
Palabras no indexables	%QW,%SW,%NW	Val.inm.,%IW,%QW,%SW,%NW, %BLK,Expr.num.
Palabras dobles indexables	%MD	%MD,%KD
Palabras dobles no indexables	%QD,%SD	Val.inm.,%ID,%QD,%SD, Expr. numérica

Expresiones numéricas

Generalidades

La expresión numérica se compone de varios operandos numéricos y de operadores aritméticos y lógicos, descritos anteriormente.
El número de operadores y de operandos de una expresión aritmética no está limitado.
Ejemplo:

```
%MW25*3-SQRT(%MW10)+%KW8*(%MW15 + %MW18)AND16#FF
```

Reglas de aplicación

- Los operandos de una misma expresión numérica pueden ser tanto de longitud simple como doble:
  Ejemplo:
  `%MW6*%MW15+SQRT(%DW6)/(%MW149[%MW8])+%KD29)AND16#FF`
- Un operando o una operación con un solo operando pueden ir precedidos del signo + o - (por defecto, signo +)
  Ejemplo:
  `SQRT(%MW5)*-%MW9`
- Todos los objetos palabras pueden utilizarse dentro de una expresión aritmética. Es posible indexar algunas palabras.

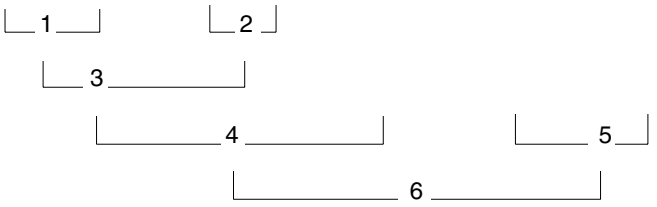
Prioridad de ejecución de las instrucciones

En la expresión numérica se respeta la prioridad de las distintas instrucciones. La ejecución se efectúa en el orden que se indica a continuación:
Orden de ejecución:

Rango	Instrucción
1	Instrucción a un operando
2	*,/,REM
3	+,-
4	<,>,<=,>=
5	=,<>
6	AND
7	XOR
8	OR

Ejemplo:
La ejecución de las instrucciones anteriores se efectúa según el orden de la numeración:

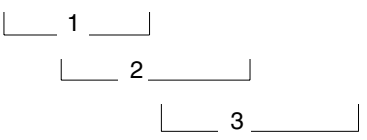
```
SQRT (%MW3) + %MW5 * 7 AND %MW8 OR %MW5 XOR %MW10
```



Paréntesis

Los paréntesis permiten modificar el orden de evolución de las prioridades. Se recomienda utilizarlos para estructurar las expresiones numéricas.
En el ejemplo se indica el orden de ejecución de los paréntesis

```
((%MW5 AND %MW6) + %MW7) * %MW8
```



1.5 Instrucciones de programa

Presentación

Objeto de este apartado

En este apartado se describen las instrucciones de programa del lenguaje PL7

Contenido

Esta sección contiene los siguientes apartados:

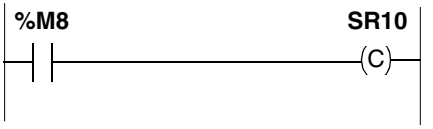
Llamada a un subprograma

Generalidades

La instrucción de llamada a un subprograma permite llamar a un módulo de subprograma situado en la misma tarea.

Estructura

Lenguaje de contactos:



Lenguaje lista de instrucciones:

```
LD  %M8
SR10
```

Lenguaje literal estructurado:

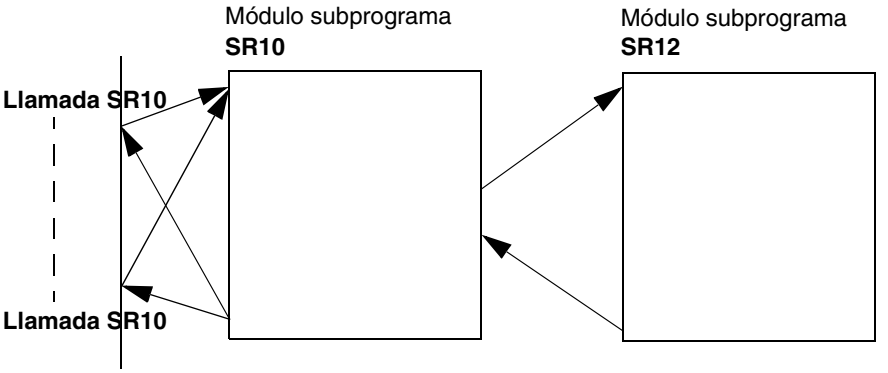
```
IF %M8 THEN
   SR10;
END_IF;
```

Reglas

- La llamada a un subprograma sólo puede realizarse si el módulo de subprograma se ha creado previamente.
- El retorno de un subprograma se realiza en la acción que sigue inmediatamente a la instrucción de llamada del subprograma.
- Un subprograma puede llamar a otro subprograma; el número de llamadas en cascada está limitado a 8.
- Los subprogramas se asignan a una tarea; sólo pueden llamarse desde la misma tarea.

Principio

Principio de ejecución de subprogramas:



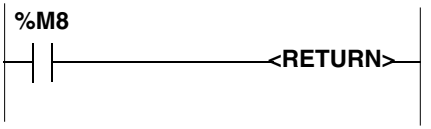
Retorno de subprograma

Generalidades

La instrucción de retorno de subprograma está reservada para los módulos de subprograma y permite volver al módulo que llama si el resultado booleano de la instrucción de prueba anterior está en 1.

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD  %M8
RETC
```

Lenguaje literal estructurado

```
IF %M8 THEN
   RETURN;
END_IF;
```

El lenguaje lista de instrucciones incluye las instrucciones adicionales siguientes:

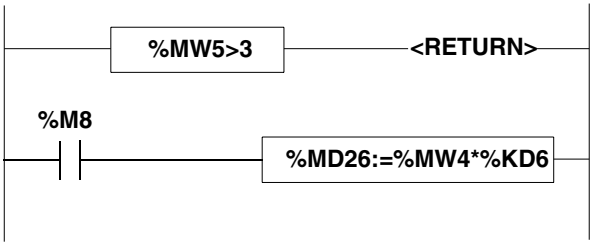
- **RETCN** : retorno de subprograma si el resultado booleano de la instrucción de prueba anterior está en 0.
- **RET** : retorno de subprograma incondicional.

Reglas de utilización

La instrucción de retorno de subprograma está implícita al final de cada subprograma, pero puede utilizarse para volver al módulo que llama antes de que finalice el subprograma.

Ejemplos

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD    [%MW5>3]
RETC
LD    %M8
[%MD26:=%MW4*%KD6]
```

Lenguaje literal estructurado

```
IF (%M5>3) THEN
   RETURN;
END_IF;
IF %M8 THEN
   %MD26:=%MW4*%KD6;
END_IF;
```

Salto en el programa

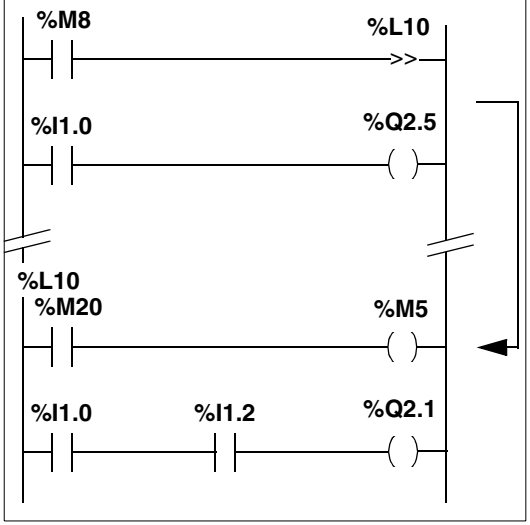
Generalidades

Las instrucciones de salto permiten conectarse a una línea de programación identificada por una etiqueta %Li:

- **JMP** : salto de programa incondicional
- **JMPC** : salto de programa si el resultado booleano de la instrucción de prueba anterior está en 1.
- **JMPCN** : salto de programa si el resultado booleano de la instrucción de prueba anterior está en 0. %Li representa la etiqueta de la línea en la que se realiza la conexión (i identifica de 1 a 999 con 256 etiquetas como máximo)

Estructura JUMPC

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD      %M8
JMPC    %L10
LD      %I1.0
ST      %Q2.5

----------
%L10:
LD     %M20
ST     %M5
LD      %I1.0
AND    %I1.2
ST     %Q2.1
```

Salto de etiqueta %L10, si %M8 está a 1

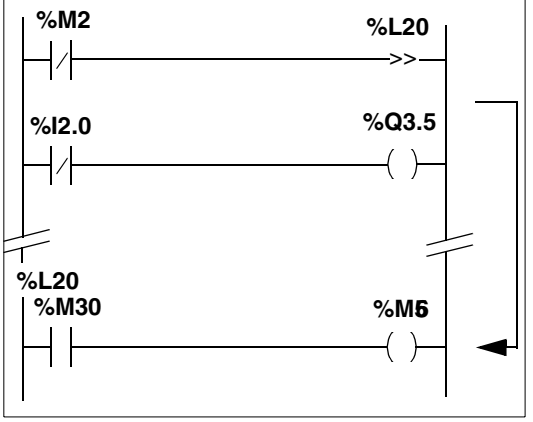
Lenguaje literal estructurado

```
IF %M8 THEN
   JUMP %L10;
END_IF;
%Q2.5:=%I1.0;

------
%L10:
  %M5:=%M20;
  %Q2.1:=%I1.0 AND %I1.2;
```

Salto a la etiqueta %L10 si %M8 está a 1

JUMPCN

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD      %M2
JMPCN   %L20
LDN     %I2.0
ST      %Q3.5

----------
%L20:
LD     %M30
ST     %M6
```

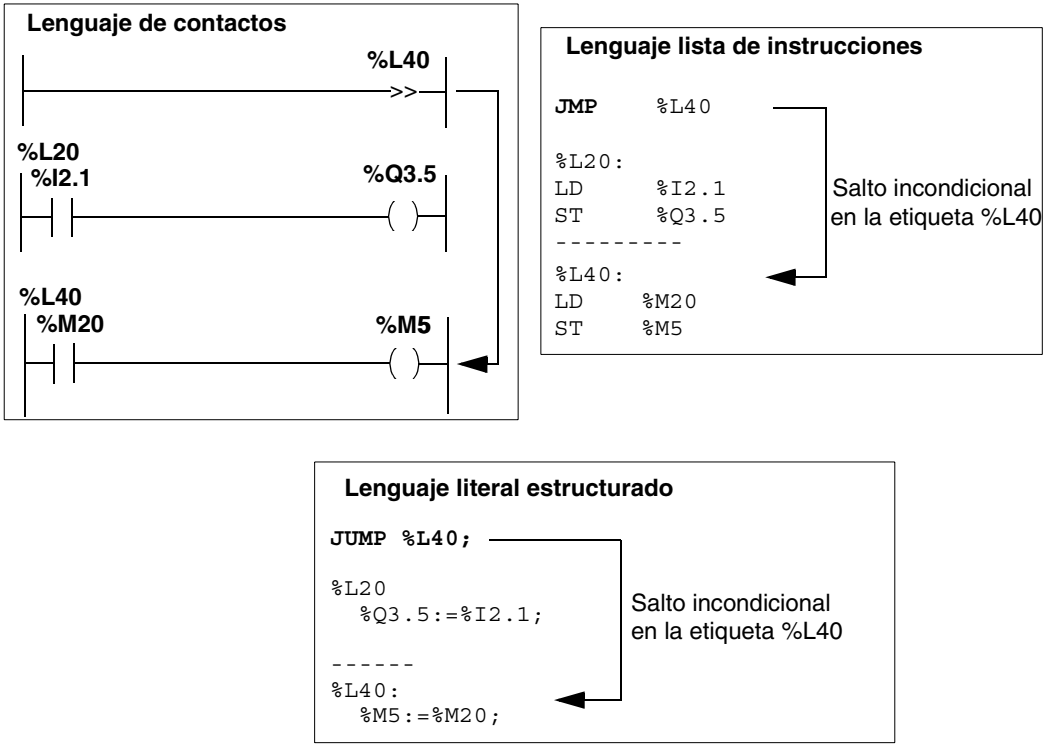
Salto de etiqueta %L20, si %M2 está a 0

Lenguaje literal estructurado

```
IF NOT %M2 THEN
   JUMP %L20;
END_IF;
%Q3.5:=NOT%I2.0;

------
%L20:
  %M6:=%M30;
```

Salto de etiqueta %L10 si %M8 está a 1

JMP



Reglas

- Un salto de programa se efectúa en el interior de una misma entidad de programación (módulo principal de una tarea maestra MAIN, subprograma %SRi,...)
- Un salto de programa se realiza hacia una línea de programación situada antes o después

En caso de salto hacia adelante, es necesario prestar atención al tiempo de ejecución del programa: éste se prolonga y puede conllevar un rebasamiento del período de la tarea que incluye el salto hacia adelante.

Instrucciones de fin de programa

Generalidades

Las instrucciones END, ENDC y ENDCN permiten definir el final de ejecución del ciclo del programa:

- **END**: fin de programa incondicional
- **ENDC**: fin de programa si el resultado booleano de la instrucción de prueba anterior está a 1
- **ENDCN**: fin de programa si el resultado booleano de la instrucción de prueba anterior está a 0.

Nota: Las instrucciones **END**, **ENDC** y **ENDCN** no deben utilizarse en las secciones de programa de los autómatas **Premium** y **Micro**. Es necesario reemplazarlas respectivamente con las instrucciones JMP, JMPC y JMPCN con un salto hacia una etiqueta al final del programa. El software de programación PL7 no ha realizado ningún control de conformidad

Reglas

Por defecto (modo normal), cuando se activa el final del programa, hay una actualización de las salidas y del paso al ciclo siguiente.
Si la exploración es periódica, las salidas se actualizarán, esperarán el final del período y pasarán al siguiente ciclo.

Nota: Estas instrucciones sólo pueden utilizarse en lenguaje de la lista de instrucciones en la tarea maestra.

Ejemplo

Lenguaje de la lista de instrucciones

Ejemplo 1:

```
LD   %M1
ST   %Q2.1
LD   %M2
ST   %Q2.2
-------
END
```

Ejemplo 2:

```
LD   %M1
ST   %Q2.1
LD   %M2
ST   %Q2.2
------
LD   %I2.2
ENDC
LD   %M2
ST   %Q2.2
------
END
```

- Si %I1.2 = 1, hay final de exploración del programa
- Si %I1.2 = 0, La exploración continua hasta la próxima instrucción END

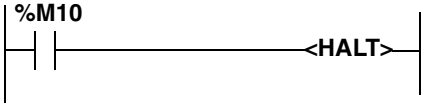
Parada del programa

Generalidades

La instrucción HALT de un programa de aplicación permite detener su ejecución (parada de todas las tareas), lo que tiene como efecto la fijación de los objetos variables del programa.
Para ejecutarse de nuevo, el programa así creado deberá inicializarse (mediante el comando INIT de PL7). Por lo tanto, las instrucciones que siguen a la instrucción HALT no se ejecutarán.

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD   %M10
HALT
```

Lenguaje literal estructurado

```
IF %M10 THEN
    HALT;
END_IF;
```

Instrucciones de enmascaramiento/desenmascaramiento de sucesos

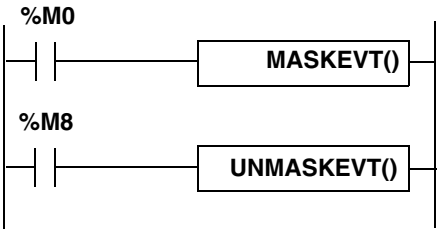
Generalidades

Estas instrucciones permiten enmascarar o desenmascarar el conjunto de los sucesos que llevan a cabo la activación de las tareas de sucesos.

- **MASKEVT**: enmascaramiento global de sucesos. El autómata almacena los sucesos, pero las tareas de sucesos asociadas permanecen inactivas mientras la operación de enmascaramiento sea válida (hasta la siguiente instrucción UNMASKEVT).
- **UNMASKEVT**: desenmascaramiento global de sucesos. Se tratan los sucesos almacenados durante el período de enmascaramiento. El mecanismo de tratamiento de sucesos es operativo hasta la siguiente instrucción MASKEVT.

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD    %M0
[MASKEVT()]

LD   %M8
[UNMASKEVT()]
```

Lenguaje literal estructurado

```
IF %M0 THEN
   MASKEVT();
END_IF;
IF %M8 THEN
   UNMASKEVT();
END_IF;
```

Instrucciones NOP

Generalidades

La instrucción **NOP** no lleva a cabo ninguna acción. Permite "reservar" líneas en un programa para poder escribir después instrucciones sin modificar los números de línea.

Instrucciones avanzadas

2

Presentación

Contenido de este capítulo

Este capítulo describe las instrucciones avanzadas del lenguaje PL7.

Contenido:

Este capítulo contiene las siguientes secciones:

2.1 Presentación de las instrucciones avanzadas

Presentación de las instrucciones avanzadas

Generalidades

Las instrucciones que se describen en este capítulo responden a requisitos de programación avanzada.
Tienen el mismo efecto independientemente del lenguaje utilizado. Sólo varía la sintaxis.

Son las siguientes:
- bien instrucciones de base del programa,
- o bien funciones que se consideran extensiones del programa.

Las instrucciones de tipo función extendida permiten enriquecer el programa de base mediante instrucciones específicas de programación.
- Operaciones en cadenas de caracteres, tablas de palabras, etc.
- Funciones específicas: Comunicación, Regulación, Diálogo del operador, etc.

Familias de instrucciones

Incluyen las siguientes familias:
- Cadenas de caracteres
- Tablas de enteros
- Gestión de fechas, horas y duraciones
- Conversiones
- Tablas de bits
- Funciones "Orphée"

Las familias siguientes se describen en las funciones específicas correspondientes:
- Comunicación
- Regulación
- Diálogo del operador
- Comando de movimiento

Nota: Las instrucciones de tipo función implican una ocupación adicional de la memoria de la aplicación (únicamente si se utilizan realmente en el programa). El programador debe tener en cuenta esta ocupación de memoria para cada función, independientemente del número de utilizaciones, todo ello de acuerdo con el tamaño máximo de la memoria del autómata.

2.2 Bloques de función predefinidos avanzados

Presentación

Objeto de este apartado

En este apartado se describen los bloques de función avanzados predefinidos del lenguaje PL7

Contenido

Esta sección contiene los siguientes apartados:

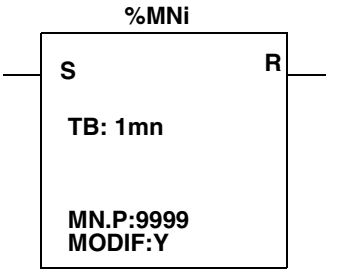
Presentación del bloque de función monoestable

Generalidades

El bloque de función monoestable permite elaborar un impulso de duración precisa. Dicha duración se puede programar y modificar o no por terminal

Figura

Representación gráfica del bloque de función monoestable



Características

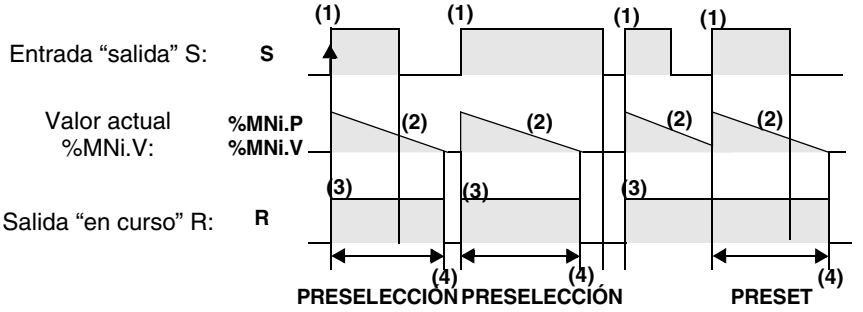
Características del bloque de función monoestable

Característica	Variable	Valor
Número	%MNi	0 a 7 para un TSX 37, 0 a 254 para un TSX 57
Base de tiempo	TB	1mn, 1s, 100ms, 10ms (1mn por defecto)
Valor actual	%MNi.V	Palabra que disminuye de %MNi.P hacia 0 al terminar el temporizador. Puede leerse y probarse, pero no escribirse.
Valor de preselección	%MNi.P	$0 \leq \%MNi.P \leq 9999$. Palabra que puede leerse, probarse y escribirse. La duración del impulso (PRESET) es igual a: %MNi.P x TB
Modificación MODIF	Y/N	• Y: posibilidad de modificar el valor de preselección en ajuste. • N: sin acceso al ajuste.
Entrada "Inicio" (o instrucción)	S (Start)	En el flanco ascendente %MNi.V = %MNi.P y a continuación %MNi.V disminuye hacia 0
Salida "Monoestable"	R (Running)	El bit asociado %MNi.R está en 1 si %MNi.V > 0 ("transcurso en curso" monoestable) %MNi.R = 0 si %MNi.V = 0

Funcionamiento del bloque de función monoestable

Generalidades El bloque de función monoestable permite elaborar un impulso de duración precisa.

Figura El cronograma muestra el funcionamiento del monoestable



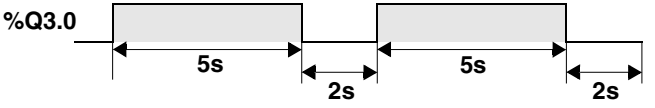
Funcionamiento Descripción del funcionamiento del monoestable

Fase	Descripción
1	Desde que aparece un flanco ascendente en la entrada S del monoestable, el valor actual %MNi.V toma el valor de preselección %MNi.P.
2	El valor actual %MNi.V disminuye hacia 0 de una unidad en cada impulso de la base de tiempo TB.
3	El bit de salida %MNi.R (Running) asociado a la salida R pasa al estado 1 desde el momento en el que el valor actual %MNi.V es distinto de 0.
4	Cuando el valor actual %MNi.V = 0, el bit de salida %MNi.R vuelve al estado 0.

Configuración y programación de los bloques de función monoestable

Ejemplo

Parpadeo en períodos cíclicos variables: el valor de preselección de cada monoestable define la duración de cada impulso.

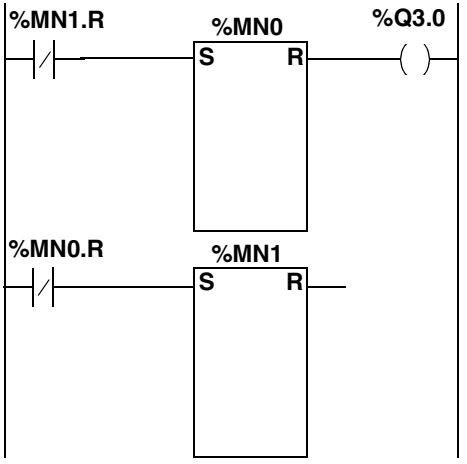


Configuración

A continuación se indican los parámetros que deben introducirse en el editor de variables:

- TB: 1mn, 1s, 100ms, 10ms o 1ms (100ms en este ejemplo)
- %MNi.P: 0 a 9999 (%MN0.P = 50 y %MN1.P = 20 en este ejemplo)
- MODIF: Y o N

Programación **Lenguaje de contactos**



Lenguaje lista de instrucciones

```
LDN %MN1.R
S %MN0
LD %MN0.R
ST %Q3.0
LDN %MN0.R
S %MN1
```

Lenguaje literal estructurado

```
%M0:=NOT %MN1.R;
IF RE %M0 THEN
 START %MN0;
END_IF;
%Q3.0:=%MN0.R;
%M1:=NOT %MN0.R;
IF RE %M1 THEN
 START %MN1;
END_IF;
```

En el ejemplo anterior, la salida %Q3.0 se pone en el estado 1 durante 5s (%MN0.P) y vuelve al estado 0 durante 2s (%MN1.P).

Observaciones

- En el lenguaje literal estructurado, la instrucción `START%MNi` permite lanzar la ejecución del bloque de función monoestable. Dicha instrucción fuerza un flanco ascendente en la entrada S del bloque, lo que provoca la reinicialización del bloque de función. La utilización de dicha instrucción debe por lo tanto ser por impulsos.
- La función monoestable puede también llevarse a cabo por el bloque de función %TMi en modo TP (Véase *Funcionamiento del bloque de función del temporizador %TMi en modo TP, p. 39*).

Casos específicos

- **Incidencia de un "rearranque en frío"**: (%S0 = 1) provoca la carga del valor de preselección %MNi.P en el valor actual %MNi.V; al perderse el valor de preselección eventualmente modificado por el terminal, la salida %MNi.R vuelve a 0.
- **Incidencia de un "rearranque en caliente"**: (%S1) no tiene ninguna incidencia en el valor actual del monoestable (%MNi.V).
- **Incidencia de un paso a stop, desactivación de la tarea y punto de parada**: el paso a stop del autómata no fija el valor actual. Ocurre lo mismo cuando la tarea en curso se desactiva o cuando se ejecuta un punto de parada.
- **Incidencia de un salto de programa**: El hecho de no explorar la red donde está programado el bloque monoestable no fija el valor actual %MNi.V, que continúa disminuyendo hacia 0. De igual forma, el bit %MNi.R asociado a la salida del bloque monoestable conserva su funcionamiento normal y puede por lo tanto probarse en otra red. Por el contrario, las bobinas directamente "conectadas" a la salida del bloque (ej. %Q3.0) no se activan, puesto que el autómata no las explora.
- **Prueba del bit %MNi.R**: este bit puede cambiar de estado en el transcurso de un ciclo.

Presentación del bloque de función Registro

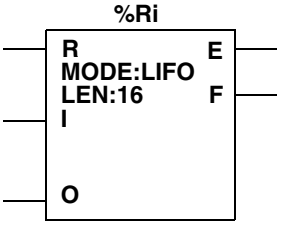
Generalidades

Un registro es un bloque de memoria que permite almacenar hasta 255 palabras de 16 bits de dos formas distintas:

- cola de espera (primero en entrar, primero en salir), denominada FIFO (First In, First Out)
- pila (último en entrar, primero en salir) denominada pila LIFO (Last In, First Out)

Figura

La representación gráfica del bloque de función de registro es la siguiente:



Características Lista de las características del bloque de función Registro:

Característica	Variable	Valor
Número	%Ri	0 a 3 para un TSX 37, 0 a 254 para un TSX 57
Modo	FIFO LIFO	Cola de espera Pila (elección por defecto)
Longitud	LEN	Número de palabras de 16 bits (1< LEN< 255) que forman el bloque de memoria de registro.
Palabra de entrada	%Ri.I	Palabra de acceso al registro. Puede leerse, probarse y escribirse.
Palabra de salida	%Ri.O	En el flanco ascendente, provoca la colocación de una palabra de información en la palabra %Ri.O.
Entrada (o instrucción) "Almacenamiento"	I (In)	En el flanco ascendente, provoca el almacenamiento del contenido de la palabra %Ri.I
Entrada (o instrucción) "Vaciado"	O (Out)	En el flanco ascendente, provoca la colocación de una palabra de información en la palabra %Ri.O.
Entrada (o instrucción) "Reset"	R (Reset)	En el estado 1, inicializa el registro
Salida "Vacío"	E (Empty)	El bit %Ri.E asociado indica que el registro está vacío. Se puede probar.
Salida "Completo"	F (Full)	El bit %Ri.F asociado indica que el registro está completo. Se puede probar.

Nota: Cuando las entradas I y O se activan simultáneamente, el almacenamiento tiene lugar antes que el vaciado.

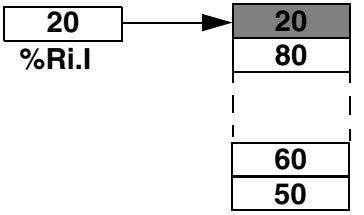
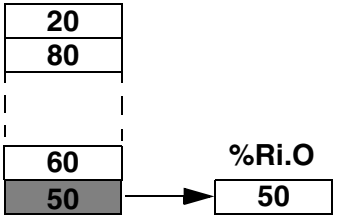
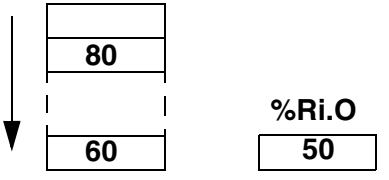
Funcionamiento del bloque de función Registro en modo FIFO

Generalidades

En el modo FIFO (First In - First Out), la primera información que entra en la pila del registro es la primera que sale.

Funcionamiento

Esta tabla describe el funcionamiento del modo FIFO

Etapa	Descripción
1	En un flanco ascendente de la entrada I o una activación de la instrucción I, el contenido de la palabra de entrada %Ri.I previamente cargada se almacena en la parte superior de la pila. Cuando la pila está completa, la carga no puede realizarse y el bit de sistema %S18 pasa a 1. 
2	en el flanco ascendente de la entrada O o la activación de la instrucción O, la palabra de información más baja de la cola se guarda en la palabra de salida %Ri.O. 
3	Desde el momento en el que la palabra se transfiere a Ri.O, el contenido del registro se desplaza de un paso hacia abajo. Cuando el registro está vacío (salida E=1), no puede realizarse el vaciado, la palabra de salida %Ri.O no evoluciona y conserva su valor. La pila puede reinicializarse en cualquier momento (estado 1 en la entrada R o activación de la instrucción R). 

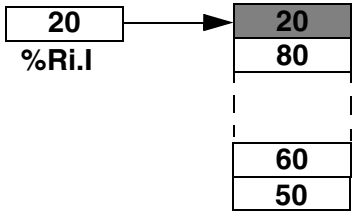
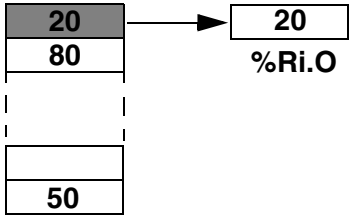
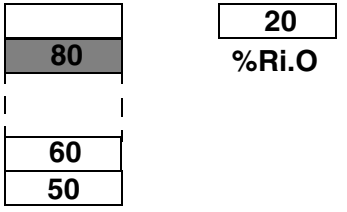
Funcionamiento del bloque de función Registro en modo LIFO

Generalidades

En el modo LIFO (Last In - First Out), la última información que entra en la pila del registro es la primera que sale.

Funcionamiento

Esta tabla describe el funcionamiento del modo LIFO

Etapa	Descripción
1	En un flanco ascendente de la entrada I o una activación de la instrucción I, el contenido de la palabra de entrada %Ri.I previamente cargada se almacena en la parte superior de la pila. Cuando la pila está completa, la carga no puede realizarse y el bit de sistema %S18 pasa a 1. 
2	En el flanco ascendente de la entrada O o la activación de la instrucción O, la palabra de información más alta de la pila (última información en entrar) se guarda en la palabra de salida %Ri.O. 
3	Desde que la palabra se transfiere a Ri.O, la palabra siguiente del registro está disponible. Cuando el registro está vacío (salida E=1), no puede realizarse el vaciado, la palabra de salida %Ri.O no evoluciona y conserva su valor. La pila puede reinicializarse en cualquier momento (estado 1 en la entrada R o activación de la instrucción R). 

Programación y configuración del bloque de función Registro

Ejemplo

En el ejemplo siguiente se muestra la carga de %R2.I por la palabra %MW34 a petición de la entrada %I1.2, si el registro R2 no está completo (%R2.F=0). %M1 realiza la petición de entrada en el registro. La entrada %I1.3 realiza la petición de salida y la colocación de %R2.O en %MW20 se efectúa si el registro no está vacío (%R2.E=0).

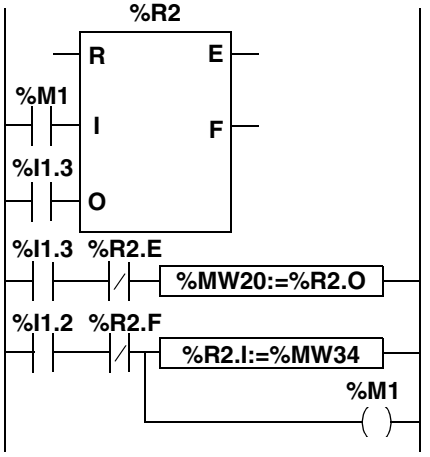
Configuración

A continuación se indican los parámetros que deben introducirse en el editor de configuración:

- Número: 1 a 4 para un TSX 37, 1 a 255 para un TSX 57
- Longitud: 1 a 255

El modo de funcionamiento (FIFO o LIFO) debe introducirse en el editor de variables.

Programación **Lenguaje de contactos**



Lenguaje lista de instrucciones

```
LD %M1
I %R2
LD %I1.3
O %R2
LD %I.3
ANDN %R2.E
[%MW20:=%R2.O]
LD %I.2
ANDN %R2.F
[%R2.I:=%MW34]
ST %M1
```

Lenguaje literal estructurado

```
IF RE %M1 THEN
 PUT %R2;
END_IF;
IF RE %I1.3 THEN
 GET %R2;
END_IF;
IF (%I1.3 AND NOT %R2.E) THEN
 %MW20:=%R2.O;
END_IF;
%M1:=%I1.2 AND NOT %R2.F;
IF %M1 THEN
 %R2.I:=%MW34;
END_IF;
```

Observación

En el lenguaje literal estructurado, 3 instrucciones permiten programar los bloques de función de registro:

- **RESET** %Ri: inicialización del registro
- **PUT** %Ri: provoca el almacenamiento del contenido de la palabra %R.I en el registro.
- **GET** %Ri: provoca la colocación de una palabra de información en la palabra %Ri.O.

Las instrucciones PUT y GET realizan un flanco ascendente en las entradas I y O respectivamente del bloque de función. La utilización de estas instrucciones debe por lo tanto ser por impulsos.

Casos específicos

- **Incidencia de un "rearranque en frío"**: (%S0=1) provoca la inicialización del contenido del registro. El bit de salida %Ri.E asociado a la salida E se pone en 1.
- **Incidencia de un "rearranque en caliente"**: (%S1=1) no tiene ninguna incidencia sobre el contenido del registro, como tampoco en el estado de los bits de salida.
- **En la puesta a 0** (entrada R o instrucción R)
  - En el lenguaje de contactos, los historiales de las entradas I y O se actualizan con los valores conectados.
  - En el lenguaje lista de instrucciones, los historiales de las entradas I y O no se actualizan: cada una conserva los valores que tenía antes de la llamada.
  - En el lenguaje literal estructurado, los historiales de las entradas I y O se actualizan con 0.

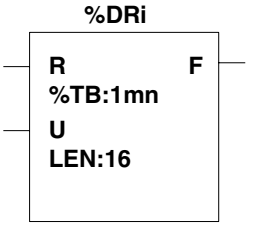
Presentación del bloque de función Programador cíclico (Drum)

Generalidades

Con un principio de funcionamiento similar al del programador de levas, el programador cíclico cambia de paso en función de sucesos exteriores. A cada paso, el punto alto de una leva da una orden utilizada por el automatismo. En el caso del programador cíclico, estos puntos altos se simbolizan por el estado 1 en cada paso y se asignan a bits de salida %Qi.j o internos %Mi denominados bits de órdenes.

Figura

Representación gráfica del bloque de función Programador cíclico (Drum)



Características Lista de las características del bloque de función Programador cíclico:

Característica	Variable	Valor
Número	%DRi	0 a 7 para un TSX 37, 0 a 254 para un TSX 57
Número de pasos	LEN	1 a 16 (16 por defecto).
Base de tiempo	TB	1mn, 1s, 100ms, 10ms (1mn por defecto).
Tiempo de envoltura o duración del paso en curso	%DRi.V	$0 \leq \%DRi.V \leq 9999$. Palabra que puede ponerse de nuevo a cero en cada cambio de paso. Puede leerse y probarse, pero no escribirse. La duración es igual a %DRi.V x TB.
Número del paso en curso	%DRi.S	$0 \leq \%Di.S \leq 15$. Palabra que puede leerse y probarse. Sólo puede escribirse a partir de un valor inmediato.
Entrada "retorno al paso 0"	R (RESET)	En el estado 1, inicializa el programador al paso 0.
Entrada "Avance"	U (UP)	En el flanco ascendente, provoca el avance de un paso del programador y la actualización de los bits de órdenes.
Salida	F (FULL)	Indica que el último paso definido está en curso. El bit %DRi.F asociado puede probarse (%DRi.F=1 si %DRi.S=número de pasos configurados -1).
Estado de un paso	%DRi.Wj	Palabra de 16 bits que define los estados del paso j del programador i. Puede leerse y probarse, pero no escribirse.
Bits de órdenes	%DRi.Wj	Salidas o bits internos asociados al paso (16 bits de órdenes)

Nota: El bit %S18 pasa a 1 si se escribe un paso no configurado.

Funcionamiento del bloque de función Programador cíclico (Drum)

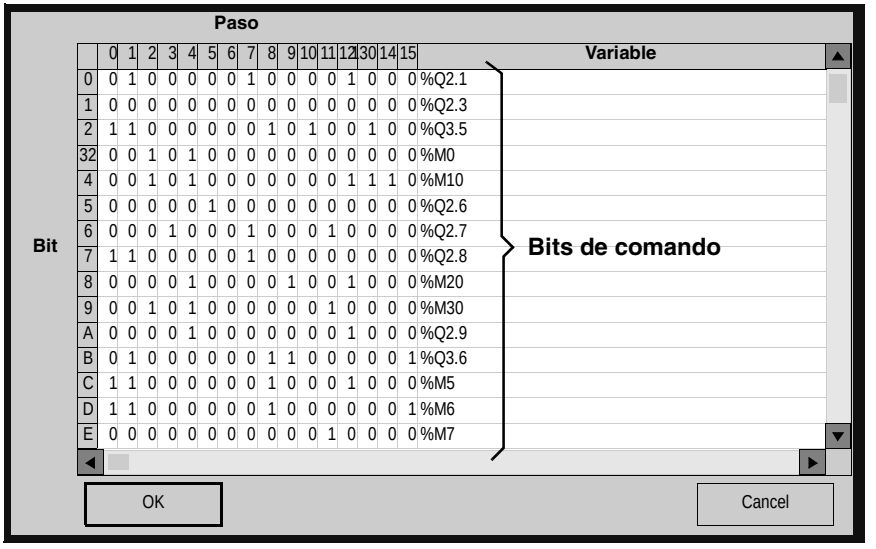
Generalidades

El programador cíclico se compone:

- de una matriz de datos constantes (levas) organizada en columnas: en paso de 0 a N-1 (N es el número de paso configurado); cada columna presenta los estados del paso en forma de 16 informaciones binarias con variables de 0 a F;
- de una lista de bits de órdenes (1 por línea) correspondientes a salidas %Qxy.i o a bits internos %Mi. Durante el paso en curso, los bits de órdenes toman los estados binarios definidos para dicho paso.

Figura

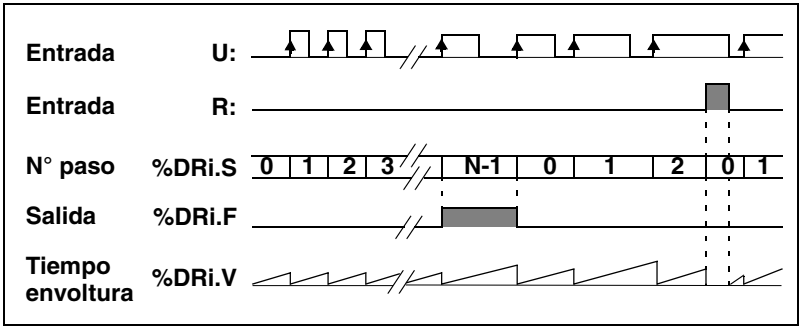
En la siguiente tabla se resumen las características principales del programador cíclico (programador configurado con 16 pasos)



Bit	Paso 0	1	2	3	4	5	6	7	8	9	10	11	12	30	14	15	Variable
0	0	1	0	0	0	0	0	1	0	0	0	0	1	0	0	0	%Q2.1
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	%Q2.3
2	1	1	0	0	0	0	0	0	1	0	1	0	0	1	0	0	%Q3.5
32	0	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	%M0
4	0	0	1	0	1	0	0	0	0	0	0	0	1	1	1	0	%M10
5	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	%Q2.6
6	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	0	%Q2.7
7	1	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	%Q2.8
8	0	0	0	0	1	0	0	0	0	1	0	0	1	0	0	0	%M20
9	0	0	1	0	1	0	0	0	0	0	0	1	0	0	0	0	%M30
A	0	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	%Q2.9
B	0	1	0	0	0	0	0	0	1	1	0	0	0	0	0	1	%Q3.6
C	1	1	0	0	0	0	0	0	1	0	0	0	1	0	0	0	%M5
D	1	1	0	0	0	0	0	0	1	0	0	0	0	0	0	1	%M6
E	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	%M7

En el ejemplo, para el paso 1, los bits de órdenes %Q2.1;%Q3.5 ;%Q2.8;%Q3.6;%M5 y %M6 se ponen en el estado 1 y el resto se ponen a 0.

Diagrama de funcionamiento



El número del paso en curso aumenta en cada flanco ascendente de la entrada U (o activación de la instrucción U). Dicho número puede modificarse por programa.

Programación y configuración del bloque de función Programador cíclico (Drum)

Ejemplo

En el ejemplo, las 5 primeras salidas %Q2.0 a %Q2.4 se activan una tras otra cada vez que la entrada %I1.1 se pone a 1. La entrada I1.0 reinicializa las salidas en el paso 0.

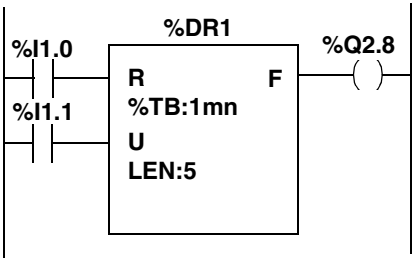
Configuración

La siguiente información se define en el editor de variables:

- número de paso: (LEN:5),
- base de tiempo (TB: 1mn)
- estados de las salidas (bits de órdenes) para cada paso del programador.

		Paso:					
		0	1	2	3	4	
	0 :	1	0	0	0	0	%Q2.0
	1 :	0	1	0	0	0	%Q2.1
Bits:	2 :	0	0	1	0	0	%Q2.2
	3 :	0	0	0	1	0	%Q2.3
	4 :	0	0	0	0	1	%Q2.4

Programación **Lenguaje de contactos**



Lenguaje lista de instrucciones

```
LD %I1.0
R %DR1
LD %I1.1
U %DR1
LD %DR1.F
ST %Q2.8
```

Lenguaje literal estructurado

```
IF %I1.0 THEN
  RESET %DR1;
END_IF;
IF RE %I1.1 THEN
  UP %DR1;
END_IF;
%Q2.8:=%DR1.F;
```

Observaciones

En el lenguaje literal estructurado, 2 instrucciones permiten programar los bloques de función del programador cíclico:

- RESET %DRi: inicializa el programador en el paso 0,
- UP %DRi: provoca el avance de un paso del programador y la actualización de los bits de órdenes. Esta instrucción realiza un flanco ascendente en la entrada U del bloque de función, por lo que su utilización debe ser por impulsos.

Nota: En la puesta a 0 (entrada R, instrucción R o instrucción RESET):

- En el lenguaje de contactos, el historial de la entrada U se actualiza con los valores conectados.
- En el lenguaje lista de instrucciones, el historial de la entrada U no se actualiza; conserva el valor que tenía antes de la llamada.
- En el lenguaje literal estructurado, el historial de U se actualiza con 0.

Casos específicos

- **Incidencia de un "rearranque en frío":** (%S0=1) provoca la reinicialización del programador en el paso 0 (con actualización de los bits de órdenes).
- **Incidencia de un "rearranque en caliente": "**(%S1=1) provoca la actualización de los bits de órdenes, según el paso en curso.
- **Incidencia de un salto de programa, desactivación de la tarea y punto de parada:** el hecho de no explorar el programador cíclico no provoca la puesta a 0 de los bits de órdenes.
- **La actualización de los bits de órdenes :**sólo se efectúa en un cambio de paso o en un rearranque en frío o en caliente.

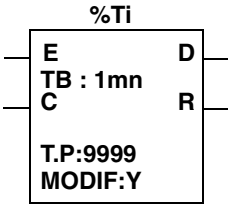
Presentación del bloque de función temporizador (Timer) serie 7

Generalidades

Este bloque de función temporizador compatible con los bloques de la serie 7 PL7-2/3 permite controlar de forma temporizada acciones específicas. El valor de este retardo se puede programar y modificar o no por terminal.

Figura

Representación gráfica del bloque de función temporizador serie 7



Características Características del bloque de función temporizador serie 7

Característica	Variable	Valor
Número	%Ti	0 a 63 para un TSX 37, 0 a 254 para un TSX 57
Base de tiempo	TB	1mn, 1s, 100ms, 10ms (1mn por defecto)
Valor actual	%Ti.V	Palabra que disminuye de %Ti.P hacia 0 al terminar el temporizador. Puede leerse y probarse, pero no escribirse.
Valor de preselección	%Ti.P	$0 \leq \%Ti.P \leq 9999$. Palabra que puede leerse, probarse y escribirse. Pasa al valor 9999 por defecto. La duración es igual a %Ti.P x TB.
Modificación MODIF	Y/N	• Y : posibilidad de modificar el valor de preselección en ajuste. • N : sin acceso al ajuste.
Entrada "Activación"	E(Enable)	En el estado 0, reinicializa el temporizador %Ti.V = %Ti.P.
Entrada "Control"	C(Control)	En el estado 0, inmoviliza el valor actual %Ti.V.
Salida "Temporizador transcurrido"	D(Done)	El bit asociado %Ti.D = 1, si el temporizador transcurrido %Ti.V = 0.
Salida "Temporizador en curso"	R(Running)	El bit asociado %Ti.R = 1, si el temporizador %Ti.P > %Ti.V > 0 y si la entrada C está en el estado 1.

Nota: Los bloques de función %Ti no se pueden programar en lista de instrucciones, por el contrario, es posible acceder a los objetos de los bloques %Ti (%Ti.V, %Ti.P, %Ti.D y %Ti.R). El número total de %Tmi + %Ti debe ser inferior a 64 en el TSX 37 y a 255 en el TSX 57.

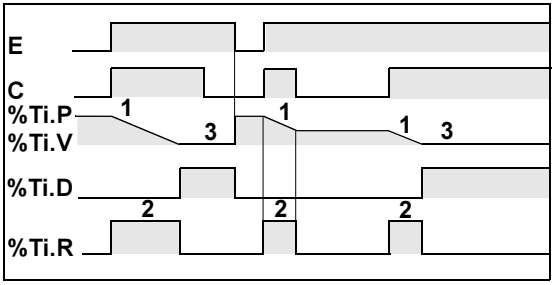
Funcionamiento del bloque de función temporizador (Timer) serie 7

Generalidades

El temporizador evoluciona cuando sus 2 entradas (E y C) se encuentran en el estado 1. Funciona como un descontador.

Figura

Diagrama de funcionamiento del temporizador serie 7



E	0	0	1	1
C	0	1	0	1
%Ti.V %Ti.P	%Ti.V = %Ti.P	%Ti.V = %Ti.P	%Ti.V inmovilizada	%Ti.V disminución de %Ti.P -> 0
%Ti.D	0	1	0	1 si Tempo transcurrida
%Ti.R	0	1	0	1 si Tempo en curso

Funcionamiento

Descripción del funcionamiento

Fase	Descripción
1	El valor actual %Ti.V disminuye de la preselección %Ti.P hacia 0 de una unidad en cada impulso de la base de tiempo TB.
2	el bit de salida %Ti.R (Temporizador en curso) asociado a la salida R se encuentra en el estado 1, el bit de salida %Ti.D (Temporizador transcurrido) asociado a la salida D se encuentra en el estado 0,
3	cuando el valor actual %Ti.V=0, %Ti.D pasa al estado 1 y %Ti.R vuelve al estado 0.

Instrucciones

En el lenguaje literal estructurado, 3 instrucciones permiten programar los bloques de función temporizador %Ti

- PRESET %Ti: reinicializa el temporizador
- START %Ti: provoca el transcurso del temporizador
- STOP %Ti: inmoviliza el valor actual del temporizador

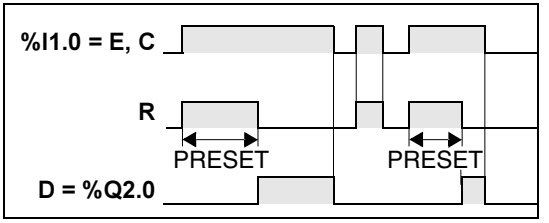
Programación del temporizador serie 7 en modo "Retardo en la conexión"

Generalidades

Según su programación, el bloque de función "Temporizador" puede realizar distintas funciones. En esta parte se describe la función de "retardo en la conexión"

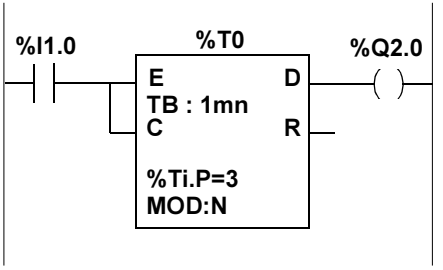
Figura

Diagrama de funcionamiento de la función de retardo en la conexión



Programación

Programación en el lenguaje de contactos



Programación en el lenguaje literal estructurado

```
IF %I1.0 THEN
  START %T0;
ELSE
  PRESET %T0;
END_IF;
%Q2.0:=%T0.D;
```

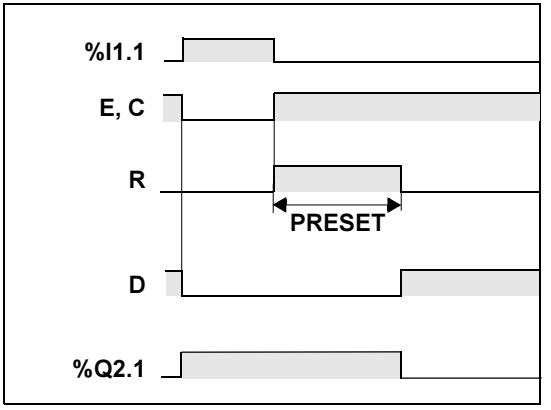
Programación del temporizador serie 7 en modo "Retardo en la desconexión"

Generalidades

Según su programación, el bloque de función "Temporizador" puede realizar distintas funciones. En esta parte se describe la función de "retardo en la desconexión"

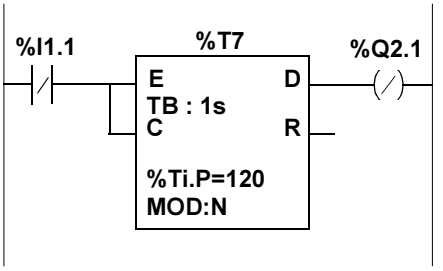
Figura

Diagrama de funcionamiento de la función de retardo en la desconexión



Programación

Programación en el lenguaje de contactos



Programación en el lenguaje literal estructurado

```
IF %I1.1 THEN
  PRESET %T7;
ELSE
  START %T7;
END_IF;
%Q2.1:=NOT%T7.D;
```

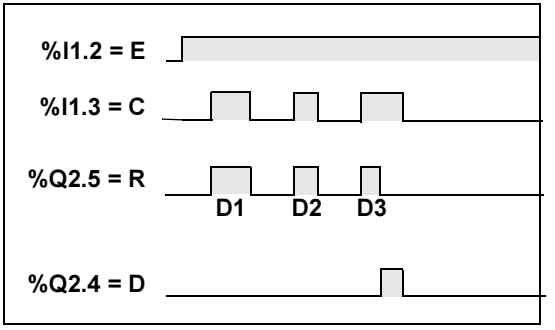
Programación del temporizador serie 7 en modo "Retardo acumulado en la conexión"

Generalidades

Según su programación, el bloque de función "Temporizador" puede realizar distintas funciones. En esta parte se describe la función de "retardo acumulado en la conexión".

Figura

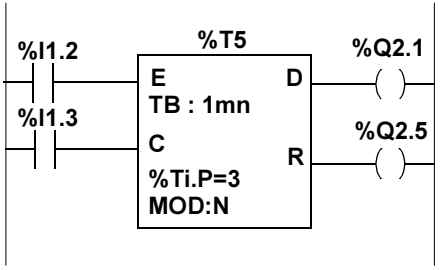
Diagrama de funcionamiento de la función de retardo acumulado en la conexión



PRESET = D1+D2+D3

Programación

Programación en el lenguaje de contactos



Programación en el lenguaje literal estructurado

```
IF %I1.2 THEN
IF %I1.3 THEN
   START %T5;
ELSE
   STOP %T5;
END_IF;
ELSE
   PRESET %T5;
END_IF;
%Q2.4:=%T5.D;
%Q2.5:=%T5.R;
```

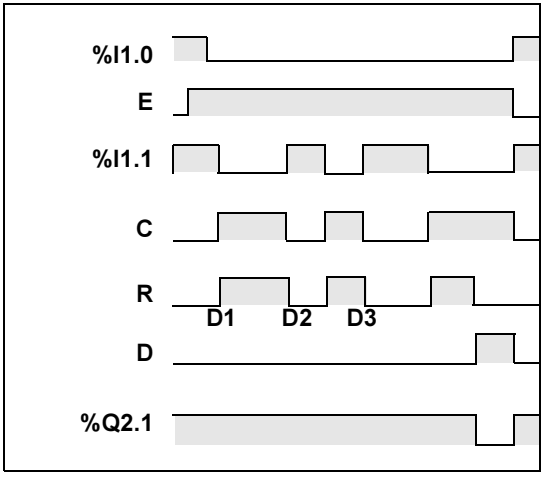
Programación del temporizador serie 7 en modo "Retardo acumulado en la desconexión"

Generalidades

Según su programación, el bloque de función "Temporizador" puede realizar distintas funciones. En esta parte se describe la función de "retardo acumulado en la desconexión"

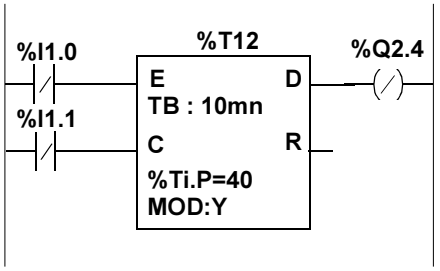
Figura

Diagrama de funcionamiento de la función de retardo acumulado en la desconexión



Programación

Programación en el lenguaje de contactos



Programación en el lenguaje literal estructurado

```
IF %I1.0 THEN
   PRESET %T12;
ELSE
 IF %I1.1 THEN
   STOP %T12;
ELSE
  START %T12;
END_IF;
END_IF;
%Q2.4:=NOT %T12.D;
```

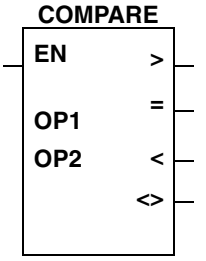
Presentación del bloque de operación comparador vertical

Generalidades

El bloque comparador vertical permite efectuar una comparación entre 2 operandos (OP). Estos 2 operandos son de tipo palabra de 16 bits eventualmente indexados o valor inmediato.
El número de bloques de comparador vertical no está limitado ni numerado.

Figura

Representación gráfica del bloque de operación comparador vertical



Características

Características del bloque de operación comparador vertical

Característica	Variable	Valor
Entrada de comando	EN	En el estado 1 provoca la comparación de los dos operandos.
Salida "Superior"	>	Está en el estado 1 si el contenido de OP1 es superior al de OP2.
Salida "Igual"	=	Está en el estado 1 si el contenido de OP1 es igual al de OP2.
Salida "Inferior"	<	Está en el estado 1 si el contenido de OP1 es inferior al de OP2.
Salida "Diferente"	<>	Está en el estado 1 si el contenido de OP1 es diferente del de OP2.
Operando número 1	OP1	Este operando es un objeto palabra de longitud simple (puede indexarse).
Operando número 2	OP2	Este operando es un objeto palabra de longitud simple (puede indexarse).

Funcionamiento del bloque de operación comparador vertical

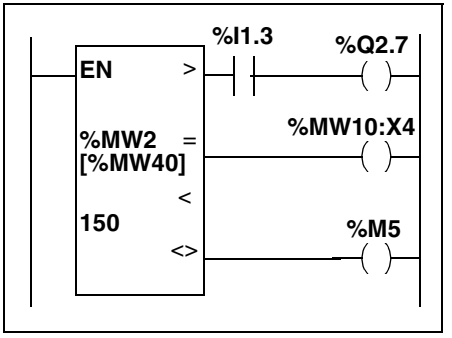
Funcionamiento

La puesta a 1 de la entrada de comando provoca la comparación de los dos operandos; las cuatro salidas se activan en función del resultado de la comparación. La puesta a 0 de la entrada de comando provoca la puesta a cero de las salidas activadas.

Ejemplo

El siguiente programa muestra la comparación de la palabra %MW2 indexada por la palabra %MW40 y del valor inmediato 150. En caso de que el contenido de %MW2[%MW40] sea superior a 150 y si %I1.3 = 1, se controla la bobina %Q2.7. Si el contenido es igual a 150 ¡, se controla la bobina %MW10:X4. La bobina %M5 sólo se controla si el contenido es diferente de 150 (< o >).

Lenguaje de contactos



Nota: Este bloque de función no existe en el lenguaje lista de instrucciones ni en lenguaje literal estructurado. Utilizar las operaciones de comparación >, <, =, <>.

Casos específicos

- **Incidencia de un "rearranque en frío":** al provocar (%S0) un reset del operando OP1 y eventualmente de OP2 (si OP2 es una palabra interna), las salidas se activan en función de la comparación con los nuevos valores.
- **Incidencia de un "rearranque en caliente":** (%S1) no tiene ninguna incidencia sobre el bloque de comparación.

2.3 Instrucciones de desplazamiento

Instrucciones de desplazamiento

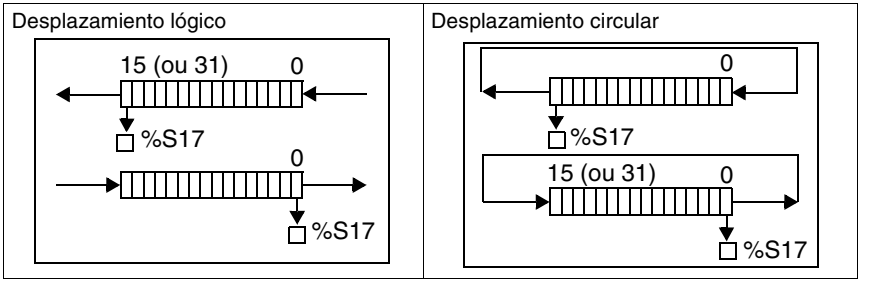
Generalidades

Las instrucciones de desplazamiento consisten en desplazar los bits de un operando palabra o palabra doble de un determinado número de posiciones hacia la derecha o hacia la izquierda. Existen dos tipos de desplazamiento:

- **el desplazamiento lógico:**
  - SHL(op2,i) desplazamiento lógico hacia la izquierda de i posiciones.
  - SHR(op2,i) desplazamiento lógico hacia la derecha de i posiciones.
- **El desplazamiento circular**
  - ROL(op2,i) desplazamiento circular hacia la izquierda de i posiciones.
  - ROR(op2,i) desplazamiento circular hacia la derecha de i posiciones.

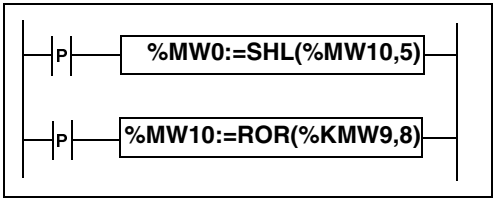
Si el operando que se va a desplazar es un operando de longitud simple, la variable i estará comprendida entre 1 y 16. Si el operando que se va a desplazar es un operando de longitud doble, la variable i estará comprendida entre 1 y 32. El estado del último bit salido se almacena en el bit %S17.

Figura de los dos tipos de desplazamiento



Estructura

Lenguaje de contactos:



Lenguaje lista de instrucciones:

```
LDR %I1.1
[%MW0:=SHL(%MW10,5)]
```

Lenguaje literal estructurado:

```
IF RE%I1.2 THEN
 %MW10:=ROR(%KW9,8);
END_IF;
```

Sintaxis

Operadores: SHL, SHR, ROL, ROR
Operandos:

Tipo	Operando 1 (Op1)	Operando 2 (Op2)
Palabras indexables	%MW	%MW, %KW, %Xi.T
Palabras no indexables	%QW, %SW, %NW, %BLK	Val.imm., %IW, %QW, %SW, %NW, %BLK, Expr. num.
Palabras dobles indexables	%MD	%MD, %KD
Palabras dobles no indexables	%QD, %SD	Val.imm., %ID, %QD, %SD, Expr. num.

Sintaxis: Op1:=Operador(Op2,i)

2.4 Instrucciones en flotante

Presentación

Objeto de este apartado

En este apartado se describen las instrucciones en flotante del lenguaje PL7

Contenido

Esta sección contiene los siguientes apartados:

Instrucciones en flotante

Generalidades

El programa PL7 permite efectuar operaciones en objetos flotantes.

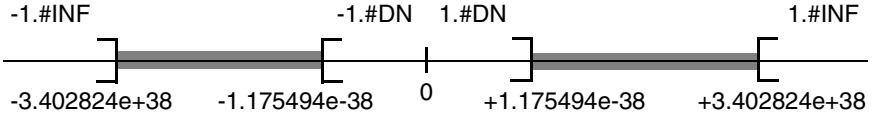
El formato utilizado es el de la norma IEEE STD 734-1985 (equivalencia IEC 559). La longitud de las palabras es de 32 bits, lo que corresponde a números flotantes de simple precisión.

Los valores flotantes se pueden representar con o sin exponente; deben llevar siempre un punto (coma flotante).

Ejemplos de valores flotantes:
sin exponente: 1285.28
con exponente: 1.28528e3

Los valores flotantes están comprendidos entre -3.402824e+38 y -1.175494e-38 ; 1.175494e-38 y 3.402824e+38 (valores en gris en el esquema). Llevan también el valor 0 anotado como 0.0

.



Cuando el resultado de un cálculo está comprendido entre -1.175494e-38 y 1.175494e-38, se redondea a 0. Un valor comprendido entre estos límites no puede introducirse como flotante si se ha introducido en otro formato; en flotante, aparecerá el símbolo 1.#DN o - 1.#DN.

Cuando el resultado de un cálculo es:
- inferior a -3.402824e+38, aparece el símbolo -1.#INF (para - infinito)
- superior a +3.402824e+38, aparece el símbolo 1.#INF (para + infinito)

Cuando el resultado de una operación es indefinido (por ejemplo, raíz cuadrada de número negativo), aparece el símbolo 1.#NAN o -1.#NAN.

El bit de sistema %S18 se sitúa en 1 cuando el resultado no se encuentra entre los límites válidos.

Los bits de la palabra de estado %SW17 indican la causa de un fallo en una operación flotante:

Diferentes bits de la palabra SW17:

%SW17:X0	operación no válida, el resultado no es un número (1.#NAN o -1.#NAN)
%SW17:X1	operando sin normalizar (comprendido entre -1.175494e-38 y 1.175494e-38), el resultado se redondea a 0.
%SW17:X2	división por 0, el resultado es infinito (-1.#INF o 1.#INF)
%SW17:X3	resultado superior en valor absoluto a +3.402824e+38, el resultado es infinito (-1.#INF o 1.#INF)
%SW17:X4	resultado inferior a 1.175494e-38, el resultado es 0.
%SW17:X5	imprecisión del resultado

El sistema vuelve a poner a 0 esta palabra en el arranque en frío y el programa también para utilizarla de nuevo.

La precisión de la representación es de 2-24. Para visualizar un número flotante, es inútil mostrar más de 6 cifras después de la coma.

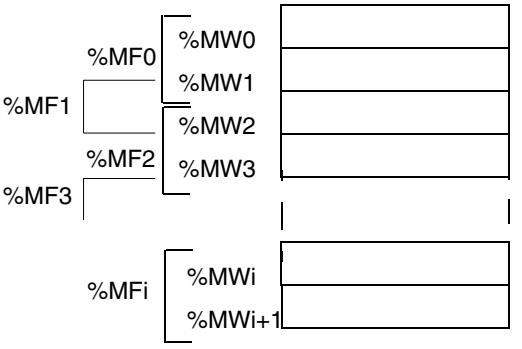
Nota:
- el valor "1285" se interpreta como valor entero; para tenerse en cuenta como valor flotante, debe escribirse: "1285.0"
- las instrucciones de conversión Entero <--> Flotante permiten pasar de un formato a otro.

Direccionamiento de los objetos flotantes

Abreviaturas	Direccionamiento completo	Tipo de flotante	Acceso	Forma indexada
Val.inm.	-	Valores inmediatos	R	-
%MF	%MFi	flotante interno	R/W	%MFi[index]
%KF	%KFi	constante flotante	R	%KFi[index]

Posibilidad de solapamiento entre objetos:
Las palabras simples, de doble longitud y flotantes se guardan en el interior del espacio dado en una misma zona de memoria. Así, la palabra flotante %MFi corresponde a las palabras de longitud simple %MWi y %MWi+1 (la palabra %MWi contiene las menos significativas y la palabra %MWi+1 las más significativas de la palabra %MFi).

Figura:



Ejemplo:
%MF0 corresponde a %MW0 y %MW. %KF543 corresponde a %KW543 y %KW544.

Instrucciones de comparación en flotante

Generalidades

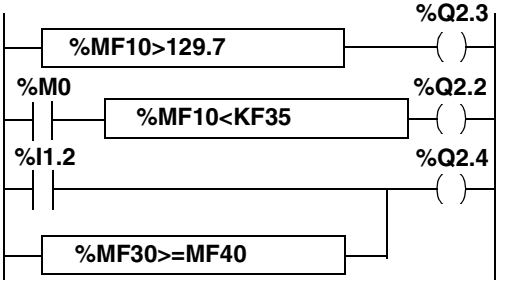
Las instrucciones de comparación permiten comparar dos operandos.

>	prueba si el operando 1 es superior al operando 2,
>=	prueba si el operando 1 es superior o igual al operando 2,
<	prueba si el operando 1 es inferior al operando 2,
<=	prueba si el operando 1 es inferior o igual al operando 2,
=	prueba si el operando 1 es igual al operando 2,
<>	prueba si el operando 1 es diferente del operando 2,

El resultado está en 1 cuando la comparación solicitada es verdadera.

Estructura

Lenguaje de contactos



Los bloques de comparación se programan en la zona de prueba.

Lenguaje lista de instrucciones

```
LD [%MF10>129.7]
ST %Q2.3
LD %M0
AND [%MF10<KF35]
ST %Q2.2
LD %I1.2
OR [%MF30>=MF40]
ST %Q2.4
```

La comparación se efectúa entre corchetes que figuran a continuación de las instrucciones LD, AND y OR.

Lenguaje literal estructurado

```
%Q2.3:=%MF10>129.7;
%Q2.2:=(%MF20<%KF35) AND %M0;
%Q2.4:=(%MF30>=%MF40) OR %I1.2;
```

Sintaxis

Operadores: `>, >=, <, <=, =, <>`
Operandos:

Tipo	Operandos 1 y 2 (Op1 y Op2)
Flotantes indexables	%MF,%KF
Flotantes no indexables	Valor inmediato flotante, Expresión numérica flotante

Nota: En el lenguaje lista de instrucciones, las instrucciones de comparación se pueden utilizar entre paréntesis.

Instrucciones de asignación en flotante

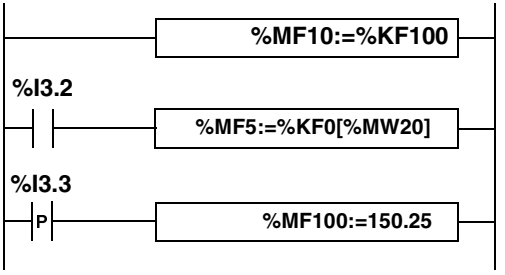
Generalidades

Se pueden llevar a cabo las siguientes operaciones de asignación en flotantes:

- flotante (indexado) -> flotante (indexado). Véase ej. 1
- valor flotante inmediato -> flotante (indexado). Véase ej. 2

Estructura

Lenguaje de contactos:



Lenguaje lista de instrucciones:
Ej. 1

```
LD TRUE
[%MF10:=%KF10]

LD %I3.2
[%MF5:=%KF0[%MW20]]
```

Ej. 2

```
LDR %I3.3
[%MF100:=150.25]
```

Lenguaje literal estructurado:
Ej. 1

```
%MF10:=%KF10;
IF %I3.2 THEN
  %MF5:=%KF0[%MW20];
END_IF;
```

Ej. 2

```
IF RE %I1.3 THEN
  %MF100:=150.25;
END_IF;
```

Sintaxis

Operadores: `:=`
Operandos:

Tipo	Operando 1 (Op1)	Operando 2 (Op2)
Flotantes indexables	%MF	%MF, %KF
Flotantes no indexables		Valor inmediato flotante, Expresión numérica flotante

Sintaxis: Op1:=Op2

Nota: Es posible realizar asignaciones múltiples. Ejemplo: `%MF0:=%MF2:=%MF4`

Instrucciones aritméticas en flotante

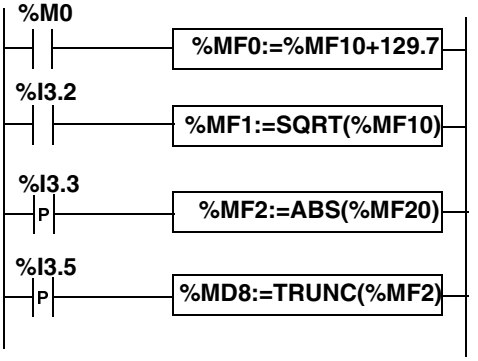
Generalidades

Estas instrucciones permiten efectuar una operación aritmética entre dos operandos o en un operando.

+	suma de dos operandos	SQRT	raíz cuadrada de un operando
-	resta de dos operandos	ABS	valor absoluto de un operando
*	multiplicación de dos operandos	TRUNC	parte entera de un valor flotante
/	división de dos operandos		

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD %M0
[%MF0:=%MF10+129.7]

LD %I3.2
[%MF1:=SQRT(%MF10)]

LDR %I3.3
[%MF2:=ABS(%MF20)]

LDR %I3.5
[%MD8:=TRUNC(%MF2)]
```

Lenguaje literal estructurado

```
IF %M0 THEN
 %MF0:=%MF10+129.7;
END_IF;
IF %I3.2 THEN
 %MF1:=SQRT(%MF10);
END_IF;
IF %I3.3 THEN
 %MF2:=ABS(%MF20);
END_IF;
IF %I3.5 THEN
 %MD8:=TRUNC(%MF2);
END_IF
```

Sintaxis

Operadores y sintaxis de las instrucciones aritméticas en flotante

Operadores	Sintaxis
+, - *, /	Op1:=Op2 Operador Op3
SQRT, ABS, TRUNC	Op1:=Operador(Op2)

Nota: Cuando se realiza una suma o una resta entre dos números flotantes, los dos operandos deben respetar la condición $\mathrm{Op1} > \mathrm{Op2} \times 2^{-24}$, con Op1>Op2. Si no se respeta esta condición, el resultado es igual al operando 1 (Op1). Este comportamiento no tiene grandes consecuencias cuando se trata de una operación aislada, ya que el error resultante es de poca importancia (2^{-24}), aunque tiene consecuencias inesperadas en caso de que el cálculo sea iterativo. Ejemplo: tomemos la instrucción **%MF2:= %MF2 + %MF0** repetida indefinidamente. Si las condiciones iniciales son %MF.0 = 1.0 y %MW2= 0, observamos un bloqueo del valor de %MF2 a 16777216.
Por tanto, se recomienda programar los cálculos iterativos con sumo cuidado. Si, pese a todo, deseamos programar este tipo de cálculo, la aplicación del cliente deberá encargarse de gestionar los errores de truncamiento.

Operandos de las instrucciones aritméticas en flotante:

Tipo	Operando 1 (Op1)	Operando 2 (Op2)
Palabras indexables	%MF (1)	%MF, %KF
Palabras no indexables	-	Valor inmediato flotante, expresión numérica flotante

(1) %MD en el caso de la instrucción TRUNC

Reglas de utilización

- Las operaciones con flotantes y enteros no se pueden mezclar directamente. Las operaciones de conversión (Véase *Instrucciones de conversión numérica, p. 145*) llevan a cabo la conversión a uno de los dos formatos.
- El bit de sistema %S18 se gestiona de igual forma que las operaciones con enteros (Véase *Instrucciones aritméticas en enteros, p. 62*), la palabra %SW17 (Véase *Instrucciones en flotante, p. 125*) indica la causa del fallo.

Instrucciones logarítmicas y exponenciales

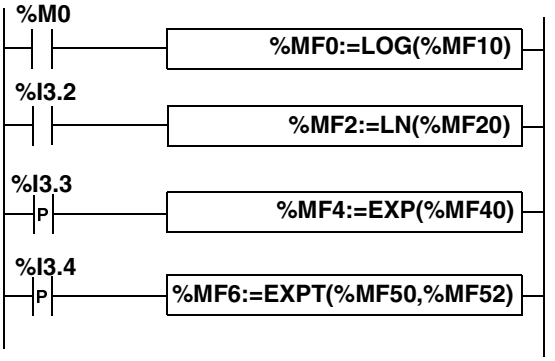
Generalidades

Estas instrucciones permiten realizar operaciones logarítmicas y exponenciales.

LOG	logaritmo de base 10
LN	logaritmo neperiano
EXP	exponencial natural
EXPT	potenciación de un real por un real

Estructura

Lenguaje de contactos



Lenguaje de la lista de instrucciones

```
LD %M0
[%MF0:=LOG(%MF10]

LD %I3.2
[%MF2:=LN(%MF20)]

LDR %I3.3
[%MF4:=EXP(%MF40)]

LDR %I3.4
[%MF6:=EXPT(%MF50,%MF52)]
```

Lenguaje Literal estructurado

```
IF %M0 THEN
 %MF0:=LOG(%MF10);
END_IF;
IF %I3.2 THEN
 %MF2:=LN(%MF20);
END_IF;
IF %I3.3 THEN
 %MF4:=EXP(%MF40);
END_IF;
IF %I3.4 THEN
 %MF6:=EXPT(%MF50,%MF52);
END_IF;
```

Sintaxis

Operadores y sintaxis de las instrucciones logarítmicas y exponenciales

Operadores	**Sintaxis**
LOG, EXP, LN	Op1:=Operador(Op2)
EXPT	Op1:=Operador (Op2,Op3)

Operandos de instrucciones logarítmicas y exponenciales

Tipo	**Operando 1 (Op1)**	**Operando 2 (Op2)**	**Operando 3 (Op3)**
Palabras indexables	%MF	%MF, %KF	%MF
Palabras no indexables	-	Valor inmediato flotante Expr. num. flotante	Valor inmediato flotante

Reglas de uso

- cuando el operando de la función es un valor no válido (ejemplo: logaritmo de un número negativo), éste da un resultado indeterminado o infinito y pone el bit %S18 a 1, la palabra %SW17 indica la causa del fallo (Generalidades (Véase *Instrucciones en flotante, p. 125*)).
- en el caso de las funciones logarítmicas, para los valores próximos a 1,0 (comprendidos entre 0,99 y 1,0 o entre 1,0 y 1,01), el resultado será igual a 0, los bits %S18 y %SW17:X5 se colocan a 1.

Instrucciones trigonométricas

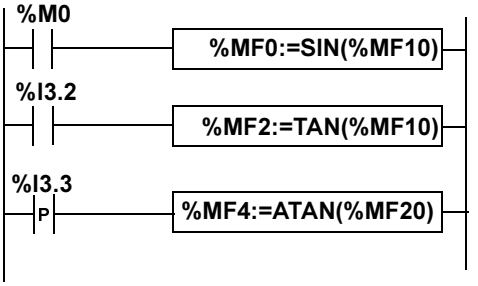
Generalidades

Estas instrucciones permiten realizar operaciones trigonométricas.

SIN	seno de un ángulo expresado en radianes	**ASIN**	arco seno (resultado entre $-\frac{\pi}{2}$ y $\frac{\pi}{2}$)
COS	coseno de un ángulo expresado en radianes	**ACOS**	arco coseno (resultado entre 0 y π)
TAN	tangente de un ángulo expresado en radianes	**ATAN**	arco tangente (resultado entre $-\frac{\pi}{2}$ y $\frac{\pi}{2}$)

Estructura

Lenguaje de contactos



Lenguaje de la lista de instrucciones

```
LD %M0
[%MF0:=SIN(%MF10)]

LD %I3.2
[%MF2:=TAN(%MF10)]

LDR %I3.3
[%MF4:=ATAN(%MF20)]
```

Lenguaje Literal estructurado

```
IF %M0 THEN
 %MF0:=SIN(%MF10);
END_IF;
IF %I3.2 THEN
 %MF2:=TAN(%MF10);
END_IF;
IF %I3.3 THEN
 %MF4:=ATAN(%MF20);
END_IF;
```

Sintaxis

Operadores y sintaxis de las instrucciones de operaciones trigonométricas:

Operadores	Sintaxis
SIN, COS, TAN, ASIN, ACOS, ATAN	Op1:=Operador(Op2)

Operandos de instrucciones de operaciones trigonométricas:

Tipo	Operando 1 (Op1)	Operando 2 (Op2)
Palabras indexables	%MF	%MF, %KF
Palabras no indexables	-	Valor inmediato flotante Expr. num. flotante

Reglas de uso

- cuando el operando de la función es un valor no válido (ejemplo: arco coseno de un número superior a 1), éste da un resultado indeterminado o infinito y pone el bit %S18 a 1, la palabra %SW17 (Véase *Instrucciones en flotante, p. 125*) indica la causa del fallo.
- las funciones SIN/COS/TAN admiten como parámetro un ángulo entre -4096π y 4096π, pero la precisión decrece progresivamente cuando se trata de ángulos que se encuentran fuera del intervalo comprendido entre -2π y $+2\pi$ debido a que el módulo 2π efectúa una imprecisión en el parámetro antes de realizar cualquier operación.
- Para los valores 0<Op2<0.01 y 0.999<Op2<1.0 de ASIN, el bit %S18 y el bit %SW17:X5 pasan a 1, lo que significa que la medida es imprecisa.

Instrucciones de conversión

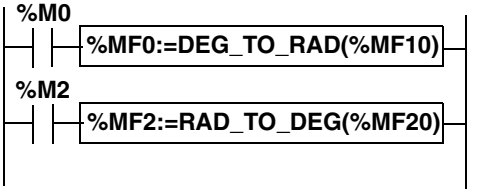
Generalidades

Estas instrucciones permiten llevar a cabo operaciones de conversión.

DEG_TO_RAD	conversión de grados a radianes; el resultado es el valor del ángulo comprendido entre 0 y 2π
RAD_TO_DEG	coseno de un ángulo expresado en radianes, el resultado es el valor del ángulo comprendido entre 0 y 360 grados

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD %M0
[%MF0:=DEG_TO_RAD(%MF10)]

LD %M2
[%MF2:=RAD_TO_DEG(%MF20)]
```

Lenguaje literal estructurado

```
IF %M0 THEN
 %MF0:=DEG_TO_RAD(%MF10);
END_IF;
IF %M2 THEN
 %MF2:=RAD_TO_DEG(%MF20);
END_IF;
```

Sintaxis

Operadores y sintaxis de las instrucciones de conversión:

Operadores	Sintaxis
DEG_TO_RAD **RAD_TO_DEG**	Op1:=Operador(Op2)

Operandos de las instrucciones de conversión:

Tipo	Operando 1 (Op1)	Operando 2 (Op2)
Palabras indexables	%MF	%MF, %KF
Palabras no indexables	-	Valor inm. flotante Expr. num. flotante

Reglas de utilización

El ángulo que se va a convertir debe estar comprendido entre -737280.0 y +737280.0 (para las conversiones DEG_TO_RAD) o entre -4096π y 4096π (para las conversiones RAD_TO_DEG).
Para los valores no comprendidos entre estos límites, el resultado mostrado será + 1.#NAN, los bits %S18 y %SW17:X0 se sitúan en 1.

Redondeo de un valor flotante con formato ASCII

Generalidades

La función ROUND proporciona el valor aproximado de un número flotante representado por una cadena de caracteres.

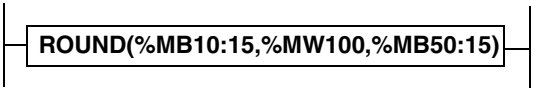
Esta función utiliza 3 parámetros:

ROUND(cadena 1, Long, Cadena 2)

- **Cadena 1:** Tablas de bytes que componen la cadena de caracteres de origen
- **Long:** Palabra que proporciona la posición, en la cadena de caracteres, a partir de la cual se efectúa el redondeo (la posición se calcula contando el número de caracteres después de la coma, ésta incluida).
- **Cadena 2:** Tablas de bytes que componen la cadena de caracteres del resultado

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD TRUE
[ROUND(%MB10:15,%MW100,%MB50:15)]
```

Lenguaje literal estructurado

```
ROUND(%MB10:15,%MW100,%MB50:15);
```

Ejemplos

Ejemplos de redondeo de valores flotantes ASCII

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
%MB10:15	"-"	"1"	"."	"2"	"3"	"4"	"5"	"6"	"7"	"0"	"e"	"+"	"2"	"6"	$00

%MW100 = 4

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
%MB50:15	"-"	"1"	"."	"2"	"3"	"4"	"5"	"0"	"0"	"0"	"e"	"+"	"2"	"6"	$00

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
%MB10:15	"-"	"1"	"."	"1"	"3"	"5"	"4"	"9"	"4"	"2"	"e"	"-"	"3"	"0"	$00

%MW100 = 2

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14
%MB50:15	"+"	"1"	"."	"1"	"0"	"0"	"0"	"0"	"0"	"0"	"e"	"-"	"3"	"0"	$00

Sintaxis

Operadores y sintaxis de las instrucciones de conversión:

Operadores	**Sintaxis**
ROUND	Op(cadena 1,Long, cadena 2)

Operandos de las instrucciones de conversión:

Tipo	**Operando 1 (Op1)**	**Operando 2 (Op2)**
Tablas de bytes	%MB:15	-
Palabras no indexables	-	%MW

Reglas de utilización

- La longitud de las cadenas de caracteres de origen y del resultado debe estar comprendida entre 15 y 255 bytes. En caso contrario, el bit %S15 se sitúa en 1.
- El parámetro de longitud Long debe estar comprendido entre 0 y 8. En caso contrario, el bit %S20 (rebasamiento de índice) se sitúa en 1. Caso particular: para L=0 o L=8, no se efectúa el redondeo (cadena de origen = cadena del resultado)
- Cuando el último carácter diferente de 0 es > a 5, el carácter anterior aumenta.

2.5 Instrucciones de conversión numérica

Presentación

Objeto de este apartado

En este apartado se describen las instrucciones en flotante del lenguaje PL7

Contenido

Esta sección contiene los siguientes apartados:

Instrucciones de conversión BCD <-> Binaria

Generalidades

Se proponen seis instrucciones de conversión.

Lista de instrucciones:

BCD_TO_INT	conversión de un número BCD de 16 bits en entero de 16 bits
INT_TO_BCD	conversión de un entero de 16 bits en número BCD de 16 bits
DBCD_TO_DINT	conversión de un número BCD de 32 bits en entero de 32 bits
DINT_TO_DBCD	conversión de un entero de 32 bits en número BCD de 32 bits
DBCD_TO_INT	conversión de un número BCD de 32 bits en entero de 16 bits
INT_TO_DBCD	conversión de un entero de 16 bits en número BCD de 32 bits

Recapitulación sobre el código BCD

El código BCD (Binary Coded Decimal), que significa decimal codificado binario, permite representar una cifra decimal de 0 a 9 con un conjunto de 4 bits. Así, un objeto palabra de 16 bits puede contener un número expresado con 4 cifras (0 < N < 9999).

Equivalencia entre decimal y BCD:

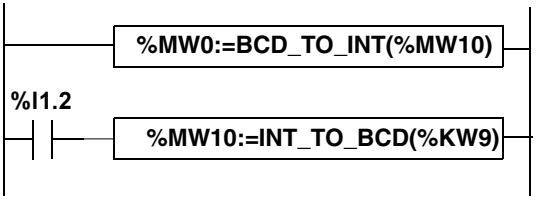
Decimal	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001

Ejemplos de codificación BCD:
- Palabra %MW5 que expresa el valor BCD "2450" correspondiente al valor binario: 0010 0100 0101 0000
- Palabra %MW12 que expresa el valor decimal "2450" correspondiente al valor binario : 0000 1001 1001 0010

El paso de la palabra %MW5 a la palabra %MW12 se efectúa a través de la instrucción BCD_TO_INT.
El paso de la palabra %MW12 a la palabra %MW5 se efectúa a través de la instrucción INT_TO_BCD.

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD TRUE
[%MW0:=BCD_TO_INT(%MW10)]

LD I1.2
[%MW10:=INT_TO_BCD(%KW9)]
```

Lenguaje literal estructurado

```
%MW0:=BCD_TO_INT(%MW10);
IF %I1.2 THEN
 %MW10:=INT_TO_BCD(%KW9);
END_IF;
```

Sintaxis

Operadores y sintaxis (conversión de un número de 16 bits):

Operadores	Sintaxis
BCD_TO_INT	Op1=operador(Op2)
INT_TO_BCD	
INT_TO_DBCD	

Operandos (conversión de un número de 16 bits):

Tipo	Operando 1 (Op1)	Operando 2 (Op2)
Palabras indexables	%MW	%MW,%KW,%Xi.T
Palabras no indexables	%QW,%SW,%NW, %BLK	Val. inm.,%IW,%SW%NW,%BLK,Expr. num
Palabras dobles indexables	%MD	-
Palabras dobles no indexables	%QD,%SD	-

Operadores y sintaxis (conversión de un número de 32 bits):

Operadores	Sintaxis
DBCD_TO_DINT	Op1=operador(Op2)
DINT_TO_DBCD	
DBCD_TO_INT	

Operandos (conversión de un número de 32 bits):

Tipo	Operando 1 (Op1)	Operando 2 (Op2)
Palabras indexables	%MW	%MW,%KW,%Xi.T
Palabras no indexables	%QW,%SW,%NW, %BLK	-
Palabras dobles indexables	%MD	%MD,%KD
Palabras dobles no indexables	%QD,%SD	Val. inm.,%ID,%QD%SD,Expr. num

Ejemplo de aplicaciones

La instrucción BCD_TO_INT se utiliza para tratar un valor de consigna presente en la entrada del autómata en ruedas codificadoras codificadas BCD.
La instrucción INT_TO_BCD se utiliza para mostrar valores numéricos (ej: resultado de un cálculo, valor actual de bloque de función) en visualizadores codificados BCD.

Reglas de utilización

- Conversión BCD->Binario
  Las instrucciones de conversión BCD-->Binario comprueban que el operador de conversión se realice efectivamente en un valor codificado en BCD. En caso de que el valor no sea un valor BCD, el bit de sistema %S18 se pone a 1 y el resultado muestra el valor del primer cuarteto del fallo.
  Ej: `BCD_TO_INT(%MW2)` con %MW2=4660 da como resultado 1234. Por el contrario, %MW2=242 (16#00F2) provoca la puesta a 1 de %S18 y el resultado es 15.
  Para la instrucción DBCD_TO_INT, si el número BCD es superior a 32767, el bit de sistema %S18 se pone a 1 y el valor -1 se guarda en el resultado.
- Conversión Binario->BCD
  Cuando el último carácter diferente de 0 es > a 5, el carácter anterior aumenta.
  La instrucción INT_TO_BCD (o DINT_TO_BCD) comprueba que el operador de conversión se realiza efectivamente en un valor comprendido entre 0 y 9999 (o 0 y 9999 9999). En caso contrario, el bit de sistema %S18 se pone a 1 y el resultado muestra el valor del parámetro de entrada.
  Ej: `INT_TO_BCD (%MW2)` con %MW2=2478 da como resultado 9336. Por el contrario, %MW2=10004 provoca la puesta a 1 de %S18 y el resultado es 10004.
  Para la instrucción INT_TO_DBCD, si el parámetro de entrada es negativo, el bit de sistema %S18 se pone a 1 y el resultado muestra el valor del parámetro de entrada.

Instrucciones de conversión Entero <-> Flotante

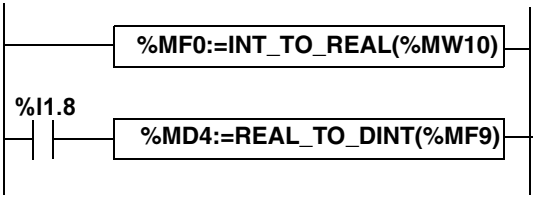
Generalidades

Se proponen cuatro instrucciones de conversión.
Lista de las instrucciones de conversión entero<-> flotante:

INT_TO_REAL	conversión de una palabra entera --> flotante
DINT_TO_REAL	conversión de una palabra doble entera --> flotante
REAL_TO_INT	conversión flotante --> palabra entera (el resultado es el valor algebraico más cercano)
REAL_TO_DINT	conversión flotante --> palabra doble entera (el resultado es el valor algebraico más cercano)

Estructura

Lenguaje de contactos



Lenguaje de la lista de instrucciones

```
LD TRUE
[%MF0:=INT_TO_REAL(%MW10)]

LD I1.8
[%MD4:=REAL_TO_DINT(%MF9)]
```

Lenguaje Literal estructurado

```
%MF0:=INT_TO_REAL(%MW10);
IF %I1.8 THEN
 %MD4:=REAL_TO_DINT(%MF9);
END_IF;
```

Sintaxis

Operadores y sintaxis (conversión de una palabra entera --> flotante):

Operadores	Sintaxis
INT_TO_REAL	Op1=INT_TO_REAL(Op2)

Operandos (conversión de una palabra entera --> flotante):

Tipo	Operando 1 (Op1)	Operando 2 (Op2)
Palabras indexables	-	%MW,%KW,%Xi.T
Palabras no indexables	-	Val. imm.,%IW,%QW,%SW%NW,%BLK,Expr. num
Palabras flotantes indexables	%MF	-

Ejemplo: conversión de una palabra entera --> flotante: 147 --> 1.47e+02

Operadores y sintaxis (conversión de una palabra doble entera --> flotante):

Operadores	Sintaxis
DINT_TO_REAL	Op1=DINT_TO_REAL(Op2)

Operandos (conversión de una palabra doble entera --> flotante):

Tipo	Operando 1 (Op1)	Operando 2 (Op2)
Palabras indexables	-	%MD,%KD
Palabras no indexables	-	Val. imm.,%ID,%QD%SD,Expr. num
Palabras flotantes indexables	%MF	-

Ejemplo:conversión de una palabra doble entera --> flotante: 68905000 --> 6.8905e+07

Operadores y sintaxis (conversión flotante --> palabra entera o palabra doble entera):

Operadores	Sintaxis
REAL_TO_INT	Op1=Operador(Op2)
REAL_TO_DINT	

Operandos (conversión flotante --> palabra entera o palabra doble entera):

Tipo	Operando 1 (Op1)	Operando 2 (Op2)
Palabras indexables	%MW	-
Palabras no indexables	%QW,%NW,%BLK	-
Palabras dobles indexables	%MD	-
Palabras dobles no indexables	%QD	-
Palabras flotantes indexables	-	%MF,%KF
Palabras flotantes no indexables	-	Valor inmediato flotante

Ejemplo:
conversión flotante --> palabra entera: 5978.6 --> 5979
conversión flotante --> palabra doble entera: -1235978.6 --> -1235979

Nota: Si durante una conversión real a entero (o real a palabra doble entera) el valor flotante se encuentra fuera de los límites de la palabra (o de la palabra doble), el bit %S18 se coloca a 1.

Precisión de redondeo

La norma IEEE 754 define 4 modos de redondear para las operaciones con flotantes.
El modo utilizado en las siguientes instrucciones es el modo "redondear al valor más cercano":
"si los valores más cercanos que se pueden representar son iguales a la distancia del resultado teórico, el valor suministrado será aquel cuyo bit menos significativo sea igual a 0".
En algunos casos, el resultado del redondeo puede, por lo tanto, tomar un valor predeterminado o un valor superior.

Por ejemplo:
Redondeo del valor 10,5 -> 10
Redondeo del valor 11,5 -> 12

Instrucciones de conversión Gray <-> Entero

Generalidades

La instrucción GRAY_TO_INT convierte una palabra de código Gray en entero (código binario puro).

Recapitulación sobre el código Gray

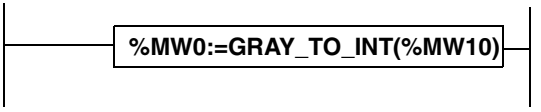
El código Gray o "binario reflejado" permite codificar un valor numérico en curso de evolución en una serie de configuraciones binarias que se diferencian entre sí por el cambio de estado de un solo y único bit.
Este código permite por ejemplo evitar la variación siguiente: en binario puro, el paso del valor 0111 a 1000 puede generar valores aleatorios comprendidos entre 0 y 1000; los bits no cambian de valor de forma perfectamente simultánea.

Equivalencia entre decimal, BCD y Gray:

Decimal	0	1	2	3	4	5	6	7	8	9
BCD	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001
Gray	0000	0001	0011	0010	0110	0111	0101	0100	1100	1101

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD TRUE
[%MW0:=GRAY_TO_INT(%MW10)]
```

Lenguaje literal estructurado

```
%MW0:=GRAY_TO_INT(%MW10);
```

Sintaxis

Operadores y sintaxis:

Operadores	**Sintaxis**
GRAY_TO_INT	Op1=GRAY_TO_INT(Op2)

Operandos:

Tipo	**Operando 1 (Op1)**	**Operando 2 (Op2)**
Palabras indexables	%MW	%MW,%KW,%Xi.T
Palabras no indexables	%QW,%SW,%NW, %BLK	Val.inm.,%IW,%QW,%SW,%NW,%BLK, Expr. num.

Instrucciones de conversión palabra <--> palabra doble

Generalidades

Las instrucciones que se describen a continuación son útiles en el caso de objetos puramente simbólicos (caso de los bloques de función DFB).
En el caso de los objetos direccionables, gracias a los mecanismos de solapamiento (ejemplo: palabra doble %MD0 formada de las palabras %MW0 y %MW1) no es necesario utilizar estas instrucciones.

Lista de instrucciones de conversión palabra<-> palabra doble:

LW	Instrucciones de extracción de la palabra de peso menos significativo de una palabra doble
HW	Instrucciones de extracción de la palabra de peso más significativo de una palabra doble
CONCATW	Instrucciones de concatenación de 2 palabras

Sintaxis

Operadores y sintaxis de las instrucciones de extracción de la palabra de peso menos significativo de una palabra doble:

Operadores	Sintaxis
LW	Op1=LW(Op2)

Operandos de las instrucciones de extracción de la palabra de peso menos significativo de una palabra doble:

Op1	Palabra de longitud simple (tipo Word)
Op2	Palabra de longitud doble (tipo DWord)

Ejemplo:

```
Presión_cubeta:=LW(Parámetro_1)
```

si `Parámetro_1=16#FFFF1234`, `Presión_cubeta=16#1234`

Operadores y sintaxis de las instrucciones de extracción de la palabra de peso más significativo de una palabra doble:

Operadores	Sintaxis
HW	Op1=HW(Op2)

Operandos de las instrucciones de extracción de la palabra de peso más significativo de una palabra doble:

Op1	Palabra de longitud simple (tipo Word)
Op2	Palabra de longitud doble (tipo DWord)

Ejemplo:

```
Presión_cubeta:=HW(Parámetro_1)
si Parámetro_1=16#FFFF1234, Presión_cubeta=16#FFFF
```

Operadores y sintaxis de las instrucciones de concatenación de 2 palabras simples y transferencia a una palabra doble:

Operadores	Sintaxis
CONCATW	Op1=CONCATW(Op2, Op3)

Operandos de las instrucciones de concatenación de 2 palabras simples y transferencia a una palabra doble:

Op1	Palabra de longitud doble (tipo DWord)
Op2	Palabra de longitud simple (tipo Word)

Op3	Palabra de longitud simple (tipo Word)

Ejemplo:

```
Presión_cubeta:=CONCATW(Parámetro_1,Parámetro_2)
```

si `Parámetro_1=16#1234`, `Parámetro_1=16#FFFF`,
`Presión_cubeta=16#FFFF1234`

2.6 Instrucciones sobre tablas de palabras

Presentación

Objeto de este apartado

En este apartado se describen las instrucciones de tablas de palabras del lenguaje PL7

Contenido

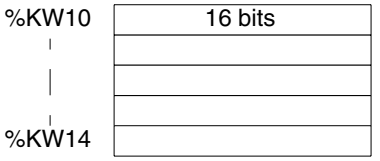
Esta sección contiene los siguientes apartados:

Instrucciones sobre tablas de palabras

Generalidades

El programa PL7 permite efectuar operaciones en tablas:
- de palabras
- de palabras dobles
- de palabras flotantes

Las tablas de palabras son series de palabras adyacentes del mismo tipo y de longitud definida: L
Ejemplo de tabla de palabras: %KW10:5



Características de las tablas de palabras

Tipo	Formato	Dirección máxima	Tamaño	Acceso de escritura
Palabras internas	Longitud simple	%MWi:L	i+L<=Nmax (1)	Sí
	Longitud doble	%MWDi:L	i+L<=Nmax-1 (1)	Sí
	Flotante	%MFi:L	i+L<=Nmax-1 (1)	Sí
Palabras constantes	Longitud simple	%KWi:L	i+L<=Nmax (1)	No
	Longitud doble	%KWDi:L	i+L<=Nmax-1 (1)	No
	Flotante	%KFi:L	i+L<=Nmax-1 (1)	No
Palabras de sistema	Longitud simple	%SW50:4 (2)	-	Sí

(1) Nmax = número máximo de palabras definido en la configuración del programa
(2) Únicamente las palabras %SW50 a %SW53 pueden direccionarse en forma de tablas.

Reglas generales sobre las operaciones de tablas

- las operaciones en tablas sólo se efectúan en tablas que contengan objetos del mismo tipo
- las operaciones en tablas sólo se efectúan en 2 tablas como máximo
- si en una operación las tablas tienen tamaños diferentes, la tabla del resultado corresponderá al mínimo de las 2 tablas operandos
- el usuario debe evitar efectuar operaciones en tablas con solapamiento (por ejemplo: `%MW100[20]:=%MW90[20]+%KW100[20]`)
- la operación en 2 tablas se efectúa en cada elemento de mismo rango de las 2 tablas y el resultado se transfiere al elemento de mismo rango de la tabla del resultado
- si en una operación entre 2 elementos, el bit de sistema %S18 se sitúa en 1, el resultado para dicha operación será erróneo, pero la operación para los elementos siguientes se efectuará correctamente
- cuando uno de los operandos sea una expresión numérica, ésta debe escribirse entre paréntesis
- el rango de una palabra en una tabla corresponde a su posición en esta última; la primera posición corresponde al rango 0

Instrucciones aritméticas en tablas

Generalidades

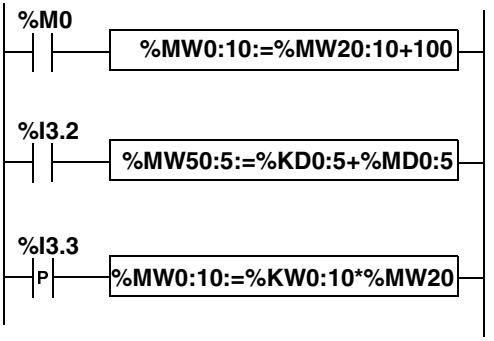
Estas instrucciones permiten efectuar una operación aritmética entre dos operandos de tipo tablas de palabras (o palabra y tabla de palabras).

Lista de instrucciones

+ : suma	***** : multiplicación
- : resta	**/** : división
REM : resto de la división	-

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD %M0
[%MW0:10:=%MW20:10+100]

LD %I3.2
[%MD50:5:=%KD0:5+%MD0:5]
```

Lenguaje literal estructurado

```
IF RE %I3.3 THEN
 %MW0:10:=%KW0:10*%MW20;
END_IF;
```

Sintaxis

Operadores y sintaxis de las instrucciones aritméticas en tablas:

Operadores	**Sintaxis**
+,-,*,/,REM	Op1:=Op2 Operador Op3

Operandos de las instrucciones aritméticas en tablas de palabras:

Tipo	**Operando 1 (Op1)**	**Operandos 2 y 3 (Op2 y 3)**
Tablas de palabras indexables	%MW:L	%MW:L,%KW:L,%Xi.T:L
Palabras indexables	-	%MW,%KW,%Xi.T
Palabras no indexables	-	Val.imm.,%IW,%QW,%SW,%NW, %BLK,Expr. num.

Operandos de las instrucciones aritméticas en tablas de palabras dobles:

Tipo	**Operando 1 (Op1)**	**Operandos 2 y 3 (Op2 y 3)**
Tablas de palabras indexables	%MD:L	%MD:L,%KD:L
Palabras dobles indexables	-	%MD,%KD
Palabras dobles no indexables	-	Val.inm.,%ID,%QD,Expr. numérica

Instrucciones lógicas en tablas

Generalidades

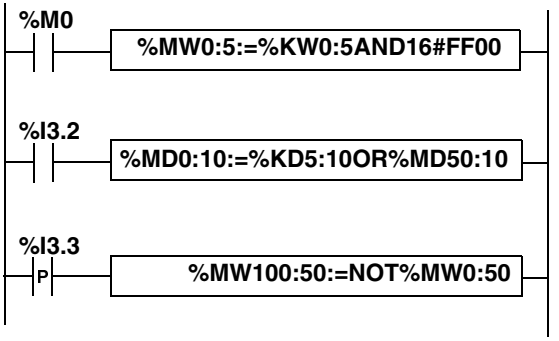
Estas instrucciones permiten efectuar una operación aritmética entre dos operandos de tipo tablas de palabras (o palabra y tabla de palabras).

Lista de instrucciones

AND: Y (bit a bit)	**XOR**: O exclusivo (bit a bit)
OR: O lógico (bit a bit)	**NOT**: complemento lógico (bit a bit) de una tabla (1 solo operando)

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD %M0
[%MW0:5:=%KW0:5 AND 16#FF00]
```

Lenguaje literal estructurado

```
IF %I3.2 THEN
 %MD0:10:=%KD5:10 OR %MD50:10;
END_IF;
IF RE %I3.3 THEN
 %MW100:50 := NOT %MW0:50;
END_IF;
```

Sintaxis

Operadores y sintaxis de las instrucciones aritméticas en tablas:

Operadores	Sintaxis
AND,OR,XOR	Op1:=Op2 Operador Op3
NOT	Op1:=NOT Op2

Operandos de las instrucciones lógicas en tablas de palabras:

Tipo	Operando 1 (Op1)	Operandos 2 y 3 (Op2 y 3)
Tablas de palabras indexables	%MW:L	%MW:L,%KW:L,%Xi.T:L
Palabras indexables	-	%MW,%KW,%Xi.T
Palabras no indexables	-	Val.inm.,%IW,%QW,%SW,%NW,%BLK, Expr. num.

Operandos de las instrucciones lógicas en tablas de palabras dobles:

Tipo	Operando 1 (Op1)	Operandos 2 y 3 (Op2 y 3)
Tablas de palabras indexables	%MD:L	%MD:L,%KD:L
Palabras dobles indexables	-	%MD,%KD,%SD
Palabras dobles no indexables	-	Val.inm.,%ID,%QD,Expr. numérica

Funciones de suma en tablas

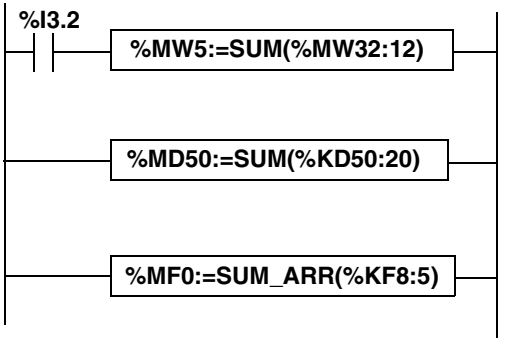
Generalidades

Las funciones SUM y SUM_ARR efectúan la suma de todos los elementos de una tabla de palabras:

- si la tabla está formada por palabras de formato simple, el resultado se proporciona con la forma de una palabra de formato simple (función SUM)
- si la tabla está formada por palabras dobles, el resultado se proporciona con la forma de una palabra doble (función SUM)
- si la tabla está formada por palabras flotantes, el resultado se proporciona con la forma de una palabra flotante (función SUM_ARR)

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD %I3.2
[%MW5:=SUM(%MW32:12]
```

Lenguaje literal estructurado

```
%MD50:=SUM(%KD50:20)

%MF0:=SUM_ARR(%KF8:5)
```

Sintaxis

Sintaxis de las instrucciones de suma en tablas:

Res:=SUM(Tab)
Res:=SUM_ARR(Tab)

Parámetros de las instrucciones de suma en tablas

Tipo	**Resultado (res)**	**Tabla (Tab)**
Tablas de palabras indexables	-	%MW:L,%KW:L,%Xi.T:L
Palabras indexables	%MW	-
Palabras no indexables	%QW,%SW,%NW	-
Tablas de palabras dobles indexables	-	%MD:L,%KD:L
Palabras dobles indexables	%MD	-
Palabras dobles no indexables	%QD,%SD	-
Tablas de flotantes indexables	-	%MF:L,%KF:L
Palabras flotantes indexables	%MF	-

Nota: el bit %S18 pasa a 1 cuando el resultado no se encuentra dentro de los límites del formato de palabra o palabra doble según el operando de la tabla.

Ejemplo

```
%MW5:=SUM(%MW30:4)
```

con %MW30=10, %MW31=20, %MW32=30, %MW33=40
%MW5=10+20+30+40=100

Funciones de comparación de tablas

Generalidades

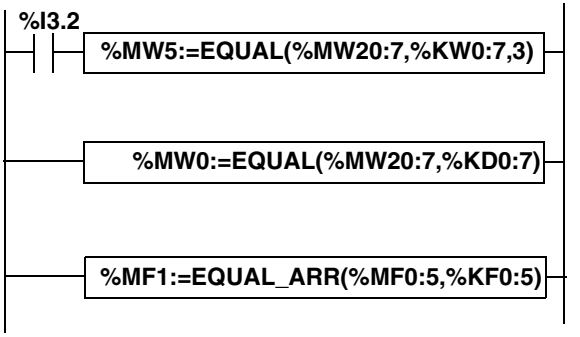
Las funciones EQUAL (en entero) y EQUAL _ARR (en flotante) efectúan la comparación de 2 tablas elemento por elemento.

Si aparece una diferencia, el rango de los primeros elementos diferentes se muestra en forma de palabra, de lo contrario, el valor mostrado es igual a -1.

El tercer parámetro proporciona el rango a partir del cual se inicia la comparación (ejemplo: 0 para comenzar al principio). Este tercer parámetro es opcional (no es posible con al función EQUAL_ARR); cuando se omite, la comparación se efectúa en la totalidad de la tabla.

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD %I3.2
[%MW5:=EQUAL(%MD20:7,KD0:7,3)]
```

Lenguaje literal estructurado

```
%MW0:=EQUAL(%MD20:7,%KD0:7)

%MW1:=EQUAL_ARR(%MF0:5,%KF0:5)
```

Sintaxis

Sintaxis de las instrucciones de comparación de tablas:

Res:=EQUAL(Tab1,Tab2,rang)
Res:=EQUAL_ARR(Tab1,Tab2)

Parámetros de las instrucciones de comparación de tablas

Tipo	Resultado (Res)	Tabla (Tab)	Rango
Tablas de palabras	-	%MW:L,%KW:L,%Xi.T:L	-
Palabras indexables	%MW	-	%MW,%KW,%Xi.T
Palabras no indexables	%QW,%SW,%NW	-	Val.inm.,%QW,%IW,%SW,%NW, Expr. num.
Tablas de palabras dobles	-	%MD:L,%KD:L	-
Palabras dobles indexables	%MD	-	%MD,%KD
Palabras dobles no indexables	%QD,%SD	-	Val.inm.,%QD,%ID,%SD,Expr. num.
Tablas de flotantes	-	%MF:L,%KF:L	-
Palabras flotantes	%MF	-	-

Nota:
- las tablas deben tener obligatoriamente la misma longitud
- si el parámetro de rango es superior al tamaño de las tablas, el resultado es igual a dicho rango.

Ejemplo

```
%MW5:=EQUAL(%MW30:4,%KW0:4,1)
```

Comparación de las 2 tablas:

Rango	Tabla de palabras	Tablas de constantes	Diferencia
0	%MW30=10	%KW0=20	Ignorado (rango<1)
1	%MW31=20	%KW1=20	=
2	%MW32=30	%KW2=30	=
3	%MW33=40	%KW3=60	Diferente

La palabra %MW5 vale 3 (primer rango diferente)

Funciones de búsqueda en tablas

Generalidades

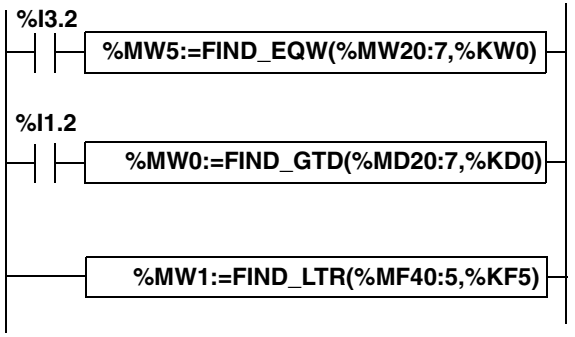
Se proponen 11 funciones de búsqueda:

- **FIND_EQW**: búsqueda de la posición en una tabla de palabras del primer elemento igual a un valor determinado
- **FIND_GTW**: búsqueda de la posición en una tabla de palabras del primer elemento superior a un valor determinado
- **FIND_LTW**: búsqueda de la posición en una tabla de palabras del primer elemento inferior a un valor determinado
- **FIND_EQD**: búsqueda de la posición en una tabla de palabras dobles del primer elemento igual a un valor determinado
- **FIND_GTD**: búsqueda de la posición en una tabla de palabras dobles del primer elemento superior a un valor determinado
- **FIND_LTD**: búsqueda de la posición en una tabla de palabras dobles del primer elemento inferior a un valor determinado
- **FIND_EQR**: búsqueda de la posición en una tabla de flotantes del primer elemento igual a un valor determinado
- **FIND_GTR**: búsqueda de la posición en una tabla de flotantes del primer elemento superior a un valor determinado
- **FIND_LTR**: búsqueda de la posición en una tabla de flotantes del primer elemento inferior a un valor determinado
- **FIND_EQWP**: búsqueda de la posición en una tabla de palabras del primer elemento igual a un valor determinado desde un rango
- **FIND_EQDP**: búsqueda de la posición en una tabla de palabras dobles del primer elemento igual a un valor determinado desde un rango

El resultado de estas instrucciones es igual al rango del primer elemento encontrado o a -1 si la búsqueda es infructuosa.

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD %I3.2
[%MW5:=FIND_EQW(%MW20:7,Kw0)]
```

Lenguaje literal estructurado

```
IF %I1.2 THEN
 %MW0:=FIND_GTD(%MD20:7,%KD0);
END_IF;

%MW1:=FIND_LTR(%MF40:5,%KF5);

%MW9:=FIND_EQWP(%MW30:8,%KF5,%MW4);
```

Sintaxis

Sintaxis de las instrucciones de búsqueda en tablas:

Función	Sintaxis
FIND_EQW	Res:=Función(Tab,Val)
FIND_GTW	
FIND_LTW	
FIND_EQD	
FIND_GTD	
FIND_LTD	
FIND_EQR	
FIND_GTR	
FIND_LTR	
FIND_EQWP	Res:=Función(Tab,Val,rang)
FIND_EQDP	

Parámetros de las instrucciones de búsqueda en tablas de palabras (FIND_EQW,FIND_GTW,FIND_LTW,FIND_EQWP)

Tipo	Resultado (Res)	Tabla (Tab)	Valor (val), rango
Tablas de palabras indexables	-	%MW:L,%KW:L,%Xi.T:L	-
Palabras indexables	%MW	-	%MW,%KW,%Xi.T
Palabras no indexables	%QW,%SW,%NW	-	Val.inm.,%QW,%IW,%SW,%NW,Expr. num.

Parámetros de las instrucciones de búsqueda en tablas de palabras dobles (FIND_EQD,FIND_GTD,FIND_LTD,FIND_EQDP)

Tipo	Resultado (Res)	Tabla (Tab)	Valor (val)
Tablas de palabras indexables	-	%MD:L,%KD:L,%Xi.T:L	-
Palabras dobles indexables	%MW	-	%MD,%KD
Palabras dobles no indexables	%QW,%SW,%NW	-	Val.inm.,%QD,%ID,%SD,Expr. num.

Nota: Para el rango, véase la tabla de palabras (igual que para FIND_EQWP)

Parámetros de las instrucciones de búsqueda en tablas de palabras flotantes (FIND_EQR,FIND_GTR,FIND_LTR)

Tipo	Resultado (Res)	Tabla (Tab)	Valor (val)
Tablas de flotantes	-	%MF:L,%KF:L	-
Palabras flotantes indexables	%MW	-	%MF,%KF
Palabras flotantes no indexables	%QW,%SW,%NW	-	Val.inm.,Expr. num.

Ejemplo

```
%MW5:=FIND_EQW(%MW30:4,%KW0)
```

Búsqueda de la posición de la primera palabra =%KW0=30 en la tabla:

Rango	Tabla de palabras	Resultado
0	%MW30=10	-
1	%MW31=20	-
2	%MW32=30	%MW5=2 (valor del rango)
3	%MW33=40	-

Funciones de búsqueda de valores máximos y mínimos en tablas

Generalidades

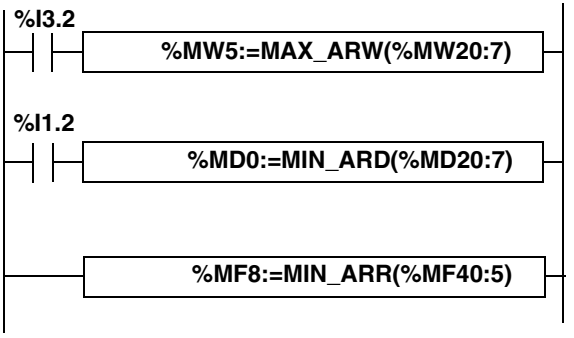
Se proponen 6 funciones de búsqueda:

- **MAX_ARW**: búsqueda del valor máximo en una tabla de palabras
- **MIN_ARW**: búsqueda del valor mínimo en una tabla de palabras
- **MAX_ARD**: búsqueda del valor máximo en una tabla de palabras dobles
- **MIN_ARD**: búsqueda del valor mínimo en una tabla de palabras dobles
- **MAX_ARR**: búsqueda del valor máximo en una tabla de flotantes
- **MIN_ARR**: búsqueda del valor mínimo en una tabla de flotantes

El resultado de estas instrucciones es igual al valor máximo (o mínimo) encontrado en la tabla.

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD %I3.2
[%MW5:=MAX_ARW(%MW20:7)]
```

Lenguaje literal estructurado

```
IF %I1.2 THEN
 %MD0:=MIN_ARD(%MD20:7);
END_IF;
%MF8:=MIN_ARR(%MF40:5);
```

Sintaxis

Sintaxis de las instrucciones de búsqueda de valores máximos y mínimos en tablas:

Función	Sintaxis
MAX_ARW	Res:=Función(Tab)
MIN_ARW	
MAX_ARD	
MIN_ARD	
MAX_ARR	
MIN_ARR	

Parámetros de las instrucciones de búsqueda de valores máximos y mínimos en tablas:

Tipo	Resultado (Res)	Tabla (Tab)
Tablas de palabras indexables	-	%MW:L,%KW:L,%Xi.T:L
Palabras indexables	%MW	-
Palabras no indexables	%QW,%SW,%NW	-
Tablas de palabras dobles indexables	-	%MD:L,%KD:L
Palabras dobles indexables	%MD	-
Palabras dobles no indexables	%QD,%SD	-
Tablas de flotantes	-	%MF:L,%KF:L
Palabras flotantes indexables	%MF	-

Número de ocurrencias de un valor en una tabla

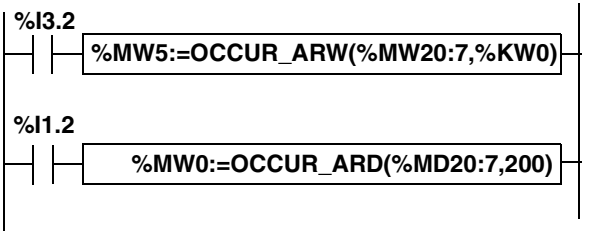
Generalidades

Se proponen 3 funciones de búsqueda:

- **OCCUR_ARW**: busca en una tabla de palabras el número de elementos iguales a un valor determinado
- **OCCUR_ARD**: busca en una tabla de palabras dobles el número de elementos iguales a un valor determinado
- **OCCUR_ARR**: busca en una tabla de flotantes el número de elementos iguales a un valor determinado

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD %I3.2
[%MW5:=OCCUR_ARW(%MW20:7,%KW0)]
```

Lenguaje literal estructurado

```
IF %I1.2 THEN
 %MW0:=OCCUR_ARD(%MD20:7,200);
END_IF;
```

Sintaxis

Sintaxis de las instrucciones de búsqueda de valores máximos y mínimos en tablas:

Función	Sintaxis
OCCUR_ARW	Res:=Función(Tab,Val)
OCCUR_ARD	
OCCUR_ARR	

Parámetros de las instrucciones de búsqueda de valores máximos y mínimos en tablas:

Tipo	Resultado (Res)	Tabla (Tab)	Valor (Val)
Tablas de palabras indexables	-	%MW:L,%KW:L,%Xi .T:L	-
Palabras indexables	%MW	-	%MW,%KW,%Xi.T
Palabras no indexables	%QW,%SW,%NW	-	Val.inm.,%QW,%IW, %SW,%NW,Expr. num.
Tablas de palabras dobles indexables	-	%MD:L,%KD:L	-
Palabras dobles indexables	%MW	-	%MD,%KD
Palabras dobles no indexables	%QW,%SW,%NW	-	Val.inm.,%QD,%ID,% SD,Expr. num.
Tablas de flotantes	-	%MF:L,%KF:L	-
Palabras flotantes indexables	%MF	-	%MF,%KF
Palabras flotantes no indexables	%QW,%SW,%NW	-	Val. inm., Expr. num.

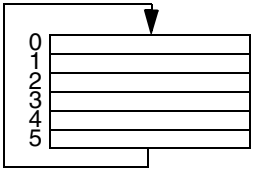
Función de desplazamiento circular en una tabla

Generalidades

Se proponen 6 funciones de desplazamiento:

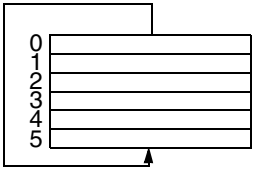
- **ROL_ARW**: realiza el desplazamiento circular de n posiciones de arriba hacia abajo de los elementos de la tabla de palabras
- **ROL_ARD**: realiza el desplazamiento circular de n posiciones de arriba hacia abajo de los elementos de la tabla de palabras dobles
- **ROL_ARR**: realiza el desplazamiento circular de n posiciones de arriba hacia abajo de los elementos de la tabla de flotantes

Figura de las funciones ROL_



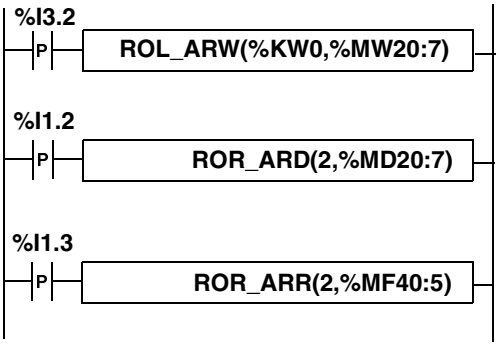
- **ROL_ARW**: realiza el desplazamiento circular de n posiciones de abajo hacia arriba de los elementos de la tabla de palabras
- **ROR_ARD**: realiza el desplazamiento circular de n posiciones de abajo hacia arriba de los elementos de la tabla de palabras dobles
- **ROR_ARR**: realiza el desplazamiento circular de n posiciones de abajo hacia arriba de los elementos de la tabla de flotantes

Figura de las funciones ROR_



Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LDR %I3.2
[ROL_ARW(%KW0,%MW0)]
```

Lenguaje literal estructurado

```
IF RE %I1.2 THEN
 ROR_ARD(2,%MD20:7);
END_IF;
IF RE %I1.3 THEN
 ROR_ARR(2,%MF40:5);
END_IF;
```

Sintaxis

Sintaxis de las instrucciones de desplazamiento circular en tablas de palabras **ROL_ARW** y **ROR_ARW**

Función	Sintaxis
ROL_ARW	Función(n,Tab)
ROR_ARW	

Parámetros de las instrucciones de desplazamiento circular en tablas de palabras **ROL_ARW** y **ROR_ARW**

Tipo	Número de posiciones (n)	Tabla (Tab)
Tablas de palabras indexables	-	%MW:L
Palabras indexables	%MW,%KW,%Xi.T	-
Palabras no indexables	Val.inm.,%QW,%IW,%SW, %NW,Expr.num.	-

Sintaxis de las instrucciones de desplazamiento circular en tablas de palabras dobles **ROL_ARD** y **ROR_ARD**

Función	Sintaxis
ROL_ARD	Función(n,Tab)
ROR_ARD	

Parámetros de las instrucciones de desplazamiento circular en tablas de palabras dobles **ROL_ARD** y **ROR_ARD**

Tipo	Número de posiciones (n)	Tabla (Tab)
Tablas de palabras indexables	-	%MD:L
Palabras indexables	%MW,%KW,%Xi.T	-
Palabras no indexables	Val.inm.,%QW,%IW,%SW, %NW,Expr.num.	-

Sintaxis de las instrucciones de desplazamiento circular en tablas de flotantes **ROL_ARR** y **ROR_ARR**

Función	Sintaxis
ROL_ARR	Función(n,Tab)
ROR_ARR	

Parámetros de las instrucciones de desplazamiento circular en tablas de flotantes: **ROL_ARR** y **ROR_ARR**:

Tipo	**Número de posiciones (n)**	**Tabla (Tab)**
Tablas de palabras indexables	-	%MF:L
Palabras indexables	%MW,%KW,%Xi.T	-
Palabras no indexables	Val.inm.,%QW,%IW,%SW, %NW,Expr.num.	-

Nota: si el valor de n es negativo o nulo, no se efectúa ningún desplazamiento.

Función de clasificación en tabla

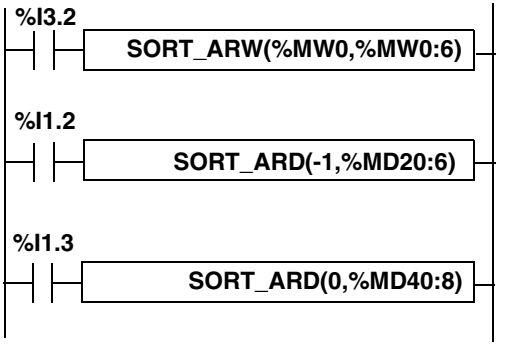
Generalidades

Se proponen 3 funciones de clasificación:

- **SORT_ARW**: realiza las clasificaciones por orden ascendente o descendente de los elementos de la tabla de palabras y ordena el resultado en la misma tabla.
- **SORT_ARD**: realiza las clasificaciones por orden ascendente o descendente de los elementos de la tabla de palabras dobles y ordena el resultado en la misma tabla.
- **SORT_ARR**: realiza las clasificaciones por orden ascendente o descendente de los elementos de la tabla de flotantes y ordena el resultado en la misma tabla.

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD %I3.2
[SORT_ARW(%MW20,%MW0:6)]
```

Lenguaje literal estructurado

```
IF %I1.2 THEN
 SORT_ARD(-1,%MD20:6);
END_IF;
IF %I1.3 THEN
 SORT_ARR(0,%MF40:8);
END_IF;
```

Sintaxis

Sintaxis de las funciones de clasificación en tablas:

Función	Sintaxis
SORT_ARW	Función(sentido,Tab)
SORT_ARD	
SORT_ARR	

- el parámetro "sentido" proporciona el orden de clasificación: sentido > 0, la clasificación se efectúa en orden ascendente, sentido < 0, la clasificación se efectúa en orden descendente
- el resultado (tabla ordenada) se devuelve al parámetro Tab (tabla para clasificar).

Parámetros de las funciones de clasificación en tablas:

Tipo	Sentido de la clasificación	Tabla (Tab)
Tablas de palabras (SORT_ARW)	-	%MW:L
Tablas de palabras dobles (SORT_ARD)	-	%MD:L
Tablas de flotantes (SORT_ARR)	-	%MF:L
Palabras indexables	%MW,%KW	-
Palabras no indexables	Val.inm.,%QW,%IW,%SW, %NW,Expr.num.	-

Función de cálculo de la longitud de tablas

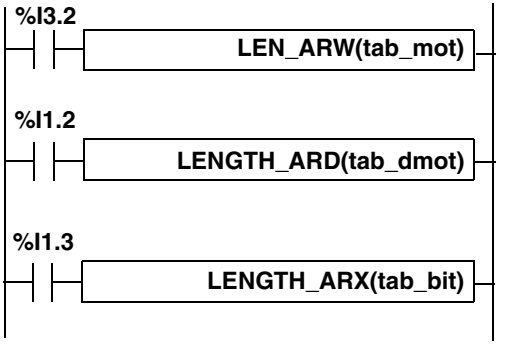
Generalidades

Se proponen 4 funciones de cálculo de la longitud de las tablas. Estas funciones resultan especialmente útiles para programar bloques de función DFB cuando las longitudes de las tablas no se han definido de forma explícita:

- **LENGTH_ARW**: calcula la longitud de una tabla de palabras en número de elementos
- **LENGTH_ARD**: calcula la longitud de una tabla de palabras dobles en número de elementos
- **LENGTH_ARR**: calcula la longitud de una tabla de flotantes en número de elementos
- **LENGTH_ARX**: calcula la longitud de una tabla de bits en número de elementos

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD %I3.2
[LENGTH_ARW(tab_mot)]
```

Lenguaje literal estructurado

```
IF %I1.2 THEN
 LENGTH_ARD(tab_dpalabra);
END_IF;
IF %I1.3 THEN
 LENGTH_ARX(tab_bit);
END_IF;
```

Sintaxis

Sintaxis de las funciones de cálculo de la longitud de tablas:

Función	Sintaxis
LENGTH_ARW	Result=Función(Tab)
LENGTH_ARD	
LENGTH_ARR	
LENGTH_ARX	

Parámetros de las funciones de cálculo de la longitud de tablas:

Tipo	Tabla (Tab)	Resultado (Result)
Tablas (LENGTH_ARW)	palabra	-
Tablas (LENGTH_ARD)	palabra doble	-
Tablas (LENGTH_ARR)	flotante	-
Tablas (LENGTH_ARX)	bit	-
Palabras indexables	-	%MW
Palabras no indexables	-	%QW,%SW,NW

Nota: los parámetros de tabla son objetos puramente simbólicos.

2.7 Instrucciones de cadenas de caracteres

Presentación

Objeto de este apartado

En este apartado se describen las instrucciones de cadenas de caracteres del lenguaje PL7

Contenido

Esta sección contiene los siguientes apartados:

Formato de una cadena de caracteres o tabla de caracteres

Generalidades

- Una tabla de caracteres se compone de una serie de bytes en la que se puede guardar una cadena de caracteres. El tamaño de la tabla permite especificar la longitud máxima que puede tener la cadena de caracteres (255 como máximo). Ejemplo: **%MB4:6 representa una tabla de 6 bytes que contiene una cadena de 6 caracteres como máximo.**
- El primer byte de inicio de una tabla debe ser par (no se puede introducir una tabla de bytes que comience por un byte impar, ej: %MB5:6).
- Las tablas de bytes utilizan la misma zona de memoria que las palabras %MW, %MD, por lo que existe el riesgo de solapamiento (("Regla de solapamientos" - Manual de referencia Tomo 1)).
- El término cadena de caracteres representa el conjunto de los caracteres comprendidos entre el principio de la tabla y el primer terminador de cadena encontrado.
- El carácter NUL (código hexa 00) se denomina Terminador de cadena. Se simboliza como **Ø** en el resto del capítulo.
- La longitud de una cadena de caracteres viene por lo tanto determinada, bien por el número de caracteres antes del terminador de la cadena, bien por el tamaño de la tabla si no se detecta ningún terminador.

Ejemplos:

La siguiente tabla (de 12 elementos) contiene la cadena de caracteres 'ABCDE' (de longitud 5):

'A'	**'B'**	**'C'**	**'D'**	**'E'**	**Ø**	**'J'**	**'K'**	**'L'**	**'M'**	**'N'**	**'O'**

La siguiente tabla (de 10 elementos) contiene la cadena de caracteres 'ABCDEJKLMN' (de longitud 10):

'A'	**'B'**	**'C'**	**'D'**	**'E'**	**'J'**	**'K'**	**'L'**	**'M'**	**'N'**

Nota: El bit de sistema %S15 se sitúa en 1 en los siguientes casos:

- Cuando al escribir una cadena de caracteres en una tabla, aquella es más larga que el tamaño de esta última (imposibilidad de escribir el terminador de cadena Ø)
- Al intentar acceder a un carácter que no se encuentra en la cadena considerada
- Incoherencia de los parámetros: La longitud que se va a eliminar es nula (función DELETE), la longitud que se va a extraer es nula (función MID), la longitud que se va a sustituir es nula (función REPLACE), búsqueda de una subcadena más larga que la cadena (función FIND).

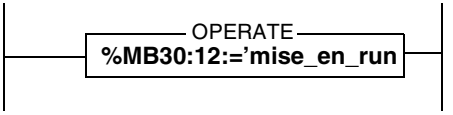
Asignación de una cadena de caracteres

Generalidades

Permite transferir una cadena de caracteres a una tabla de bytes de longitud L.

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD TRUE
[%MB30:12:='mise_en_run']
```

Lenguaje literal estructurado

```
%MB30:12:='mise_en_run';
```

Ejemplo

Transferencia de la cadena de caracteres ‘mise_en_run’ a la tabla de bytes de longitud 12

%MB	*30*	*31*	*32*	*33*	*34*	*35*	*36*	*37*	*38*	*39*	*40*	*41*
	‘m’	**‘i’**	**‘s’**	**‘e’**	**‘_’**	**‘e’**	**‘n’**	**‘_’**	**‘r’**	**‘u’**	**‘n’**	**Ø**

Sintaxis

Operadores de asignación de cadena de caracteres

Op1:=Op2

Operandos de asignación de cadena de caracteres

Tipo	Operando 1 (Op1)	Operando 2 (Op2)
Tablas de bytes	%MB:L	%MB:L,KB:L,Valor inmediato

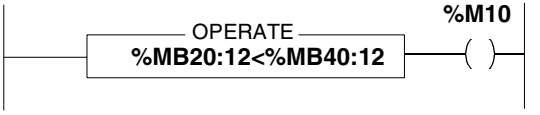
Comparaciones alfanuméricas

Generalidades

Estos operadores permiten comparar dos cadenas de caracteres incluidas en las tablas de bytes que se han pasado a parámetros. La comparación se efectúa carácter por carácter.
El resultado es un bit que vale 1 si las dos cadenas cumplen la condición establecida por el operador, carácter por carácter; en caso contrario, el bit vale 0.
La tabla de códigos ASCII (ISO 646) determina el orden de los caracteres. Por ejemplo, la cadena 'Z' es más larga que la cadena 'AZ', que a su vez es más larga que la cadena 'ABC'.

Estructura

Lenguaje de contactos



Nota: Los bloques de comparación se programan en la zona de prueba.

Lenguaje lista de instrucciones

```
LD [%MB20:12<%MB40:12]
ST %M10
```

Nota: La comparación se efectúa entre corchetes que figuran a continuación de las instrucciones LD, AND y OR.

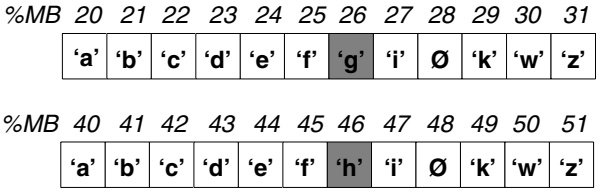
Lenguaje literal estructurado

```
%MB10<%MB40:12;
```

Ejemplo

Ejemplo: `%MB20:12<%MB40:12` ==> SÍ El resultado vale 1

Figura



Los elementos situados después del terminador no se tienen en cuenta.

Sintaxis

Operadores de comparaciones alfanuméricas

Operadores	**Sintaxis**
<, >, <=, >=, =, <>	Op1 Operador Op2

Operandos de comparaciones alfanuméricas

Tipo	**Operando 1 (Op1) y Operando 2 (Op2)**
Tablas de bytes	%MB:L, %KB:L, valor inmediato

Funciones de conversión Numérico <---> ASCII

Generalidades

Estas funciones permiten convertir un valor numérico (o flotante) en una cadena de caracteres codificada en ASCII o a la inversa.
El resultado de la conversión debe transferirse a un objeto PL7 mediante una operación de asignación: tabla de bytes, palabra simple o de doble longitud, flotante.

Lista de las funciones de conversiones Numérico <---> ASCII posibles

Operadores	Descripción
INT_TO_STRING	Conversión Binario -->ASCII (palabras)
DINT_TO_STRING	Conversión Binario -->ASCII (palabras dobles)
STRING_TO_INT	Conversión ASCII -->Binario (palabras simples)
STRING_TO_DINT	Conversión ASCII -->Binario (palabras dobles)
REAL_TO_STRING	Conversión Flotante-->ASCII
STRING_TO_REAL	Conversión ASCII-->Flotante

Recapitulación sobre el formato flotante (Véase *Instrucciones en flotante, p. 125*)

Recapitulación sobre el código ASCII:
El conjunto de los 256 caracteres alfanuméricos y de control se puede codificar en 8 bits. Dicho código, denominado ASCII (American Standard Code for Information Interchange), es compatible con la noción de bytes. Cualquier tabla de n bytes puede por lo tanto estar formada por n códigos ASCII que definen n caracteres.

Conversión binario-->ASCII

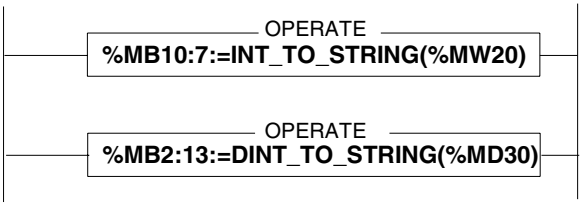
Generalidades

Estas funciones permiten convertir un valor numérico (palabra simple o de doble longitud) en una cadena de caracteres codificada en ASCII.
Cada cifra, así como el signo del valor pasado a parámetro, se codifica en ASCII en un elemento de la tabla de bytes del resultado.

- Función **INT_TO_STRING**: Puesto que el contenido de una palabra de longitud simple puede estar comprendido entre -32768 y +32767, es decir, 5 cifras más el signo, el resultado será una tabla de 6 caracteres más el terminador. El signo '+' o '-' se sitúa en el primer carácter, las unidades en el sexto carácter, las decenas en el quinto, y así sucesivamente.
- Función **DINT_TO_STRING**: Puesto que el contenido de una palabra de longitud doble puede estar comprendido entre -2147483648 y +2147483647, es decir, 10 cifras más el signo, el resultado será una tabla de 12 caracteres más el terminador. El signo '+' o '-' se sitúa en el primer carácter, la unidad en el duodécimo carácter, las decenas en el decimoprimer carácter, y así sucesivamente. El segundo carácter es siempre '0'.

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD TRUE
[%MB10:7:=INT_TO_STRING(%MW20)]
```

Lenguaje literal estructurado

```
%MB2:13:=DINT_TO_STRING(%MD30);
```

Ejemplos

Conversión Binario --->ASCII
`%MB10:7:=INT_TO_STRING(%MW20)` con %MW20 = - 3782 en decimal
==> El resultado se guarda en la tabla de 7 bytes según %MB10:

Figura

%MB	*10*	*11*	*12*	*13*	*14*	*15*	*16*
	'_'	**'0'**	**'3'**	**'7'**	**'8'**	**'2'**	**Ø**

Ejemplo: `%MB2:13:=DINT_TO_STRING(%MD30)`
con %MD30 = - 234701084

Figura

%MB	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*
	'-'	**'0'**	**'0'**	**'2'**	**'3'**	**'4'**	**'7'**	**'0'**	**'1'**	**'0'**	**'8'**	**'4'**	**Ø**

Sintaxis

Operadores de conversión Binario-->ASCII

Sintaxis
Result:=**INT_TO_STRING** (valor)

Operandos de conversión Binario-->ASCII

Tipo	**Result (resultado)**	**valor**
Tablas de 6 bytes + terminador	%MB:7	-
Palabras indexables	-	%MW,%KW,%Xi.T
Palabras no indexables	-	%IW,%QW,%SW,%NW,Val inm.,Expr. num.

Operadores de conversión Binario-->ASCII (palabras dobles)

Sintaxis
Result:=**DINT_TO_STRING** (valor)

Operandos de conversión Binario-->ASCII (palabras dobles)

Tipo	**Result (resultado)**	**valor**
Tablas de 12 bytes + terminador	%MB:13	-
Palabras indexables	-	%MD,%KD
Palabras no indexables	-	%ID,%QD,%SD,Val inm.,Expr. num.

Conversión ASCII-->binario

Generalidades

Estas funciones permiten convertir en binario una cadena de caracteres que represente un valor numérico (resultado transferido a una palabra simple o de longitud doble). Cada uno de los elementos de la tabla pasado a parámetro representa el código ASCII de un carácter. Los caracteres permitidos son las cifras y los caracteres '+' y '-'.

- Función **STRING_TO_INT**: convierte una cadena de 6 caracteres que representa un valor numérico comprendido entre -32768 y +32767. El primer carácter debe representar el signo y los caracteres según el valor: el segundo, las decenas de millares; ... el sexto carácter, las unidades. El valor debe situarse a la derecha de la cadena.
- Función **STRING_TO_DINT**: convierte una cadena de 12 caracteres que representan un valor numérico comprendido entre -2147483648 y +2147483647. el primer carácter debe representar el signo y los caracteres según el valor: el segundo es el carácter '0' ; el tercero, los millares de millones;... el duodécimo, las unidades. El valor debe situarse a la derecha de la cadena.

Estructura

Lenguaje de contactos

Lenguaje lista de instrucciones

```
LD TRUE
[%MW13:=STRING_TO_INT(%MB20:7)]
```

Lenguaje literal estructurado

```
%MD2:=STRING_TO_DINT(%MB30:13);
```

Ejemplos

Ejemplo: `%MW13:=STRING_TO_INT(%MB20:7)` , con

%MB	*20*	*21*	*22*	*23*	*24*	*25*	*26*
	'-'	**'0'**	**'2'**	**'3'**	**'4'**	**'7'**	**Ø**

El resultado de %MW13 = -2347 en decimal

Sintaxis

Operadores de conversión ASCII-->Binario

Sintaxis
Result:=**STRING_TO_INT** (cadena)

Operandos de conversión ASCII-->Binario

Tipo	**Result (resultado)**	**valor**
Palabras indexables	%MW	-
Palabras no indexables	%QW,%SW,%NW	-
Tablas de 6 bytes + terminador	-	%MB:7,%KB:7,Val. inm.

Nota: El bit %S18 se sitúa en 1 si el valor descrito por la cadena no está comprendido entre -32768 y +32767 o si uno de los 6 caracteres es erróneo.

Operadores de conversión ASCII-->Binario (palabras dobles)

Sintaxis
Result:=**DINT_TO_STRING** (cadena)

Operandos de conversión ASCII-->Binario (palabras dobles)

Tipo	**Result (resultado)**	**valor**
Palabras indexables	%MD	-
Palabras no indexables	%QD,%SD	-
Tablas de 12 bytes + terminador	-	%MB:13,%KB:13,Val. inm.

Nota: El bit %S18 se sitúa en 1 si el valor descrito por la cadena no está comprendido entre -2147483648 y +2147483647 o si uno de los 12 caracteres es erróneo.

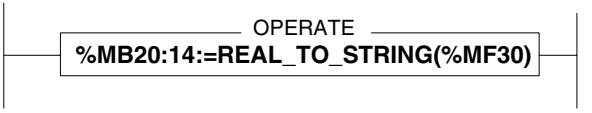
Conversión Flotante-->ASCII

Generalidades

Esta función permite convertir un valor numérico real contenido en una palabra de tipo flotante en una cadena de caracteres codificada en ASCII. El resultado se transfiere a una tabla de 13 bytes + el terminador.
Cada cifra del valor, así como los caracteres '+', '-', '.', 'e' y 'E' se codifican en ASCII en un elemento de la tabla del resultado.
El signo del valor se encuentra en el primer carácter, la coma (.) en el tercero, el exponente 'e' en el décimo, el signo del exponente en el decimoprimero.

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD TRUE
[%MB20:14:=REAL_TO_STRING(%MF30)]
```

Lenguaje literal estructurado

```
%MB20:14:=REAL_TO_STRING(%MF30);
```

Ejemplos

Ejemplo: `%MB20:14:=REAL_TO_STRING(%MF30)` con %MF30=- 3.234718e+26 ===> resultado:

%MB	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*	*33*
	'-'	**'3'**	**'.'**	**'2'**	**'3'**	**'4'**	**'7'**	**'1'**	**'8'**	**'e'**	**'+'**	**'2'**	**'6'**	**Ø**

Sintaxis

Operadores de conversión Flotante-->ASCII

Sintaxis
Result:=**REAL_TO_STRING** (valor)

Operandos de conversión Flotante-->ASCII

Tipo	**Result (resultado)**	**valor**
Tablas de 13 bytes + terminador	%MB14	-
Palabras indexables	-	%MF,%KF
Palabras no indexables	-	Val. inm.,Expr. num.

Nota: El bit %S18 se sitúa en 1 si el valor flotante pasado a parámetro no está comprendido entre -3.402824e+38 y -1.175494e-38 ó +1.175494e-38 y +3.402824e+38. En tal caso, el valor del resultado es erróneo.

Conversión ASCII-->Flotante

Generalidades

Esta función permite convertir en flotante una cadena de caracteres que represente un valor numérico real (resultado transferido a una palabra de tipo flotante).
Cada uno de los elementos de la tabla pasado a parámetro representa el código ASCII de un carácter. Los caracteres permitidos son las cifras y los caracteres '+', '-', '.', 'e' y 'E'. No se utiliza el terminador de cadena para determinar el final de la misma, lo que significa que los 13 caracteres de la tabla deben ser correctos.
El signo del valor debe encontrarse en el primer carácter, la coma (.) en el tercero, el exponente 'e' en el décimo, el signo del exponente en el decimoprimero. Por ejemplo, el valor 3.12 debe indicarse con la forma '+3.120000e+00'.

Estructura

Lenguaje de contactos

Lenguaje lista de instrucciones

```
LD TRUE
[%MF18:=STRING_TO_REAL(%MF20:13)]
```

Lenguaje literal estructurado

```
%MB18:=STRING_TO_REAL(%MB20:13);
```

Ejemplos

Ejemplo: `%MF18:=STRING_TO_REAL(%MB20:13)` con

%MB	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*	*32*
	'-'	**'3'**	**'.'**	**'2'**	**'3'**	**'4'**	**'7'**	**'1'**	**'8'**	**'e'**	**'+'**	**'2'**	**'6'**

===> resultado: %MF18 = -3.234718e+26

Sintaxis

Operadores de conversión ASCII-->Flotante

Sintaxis
Result:=**STRING_TO_REAL** (cadena)

Operandos de conversión ASCII-->Flotante

Tipo	**Result (resultado)**	**valor**
Palabras indexables	%MF	-
Tablas de 13 bytes	-	%MB:13,%KB:13,Valor inmediato

Nota: El bit %S18 se sitúa en 1:
- si el valor descrito por la cadena no está comprendido entre -3.402824e+38 y -1.175494e-38
- si el valor descrito por la cadena no está comprendido entre +1.175494e-38 y +3.402824e+38
- si uno de los 13 caracteres es erróneo.

Concatenación de dos cadenas

Generalidades

Estas instrucciones llevan a cabo la concatenación de dos cadenas de caracteres definidas en parámetros. El resultado es una tabla de bytes que contiene una cadena de caracteres.

Estructura

Lenguaje de contactos

Lenguaje lista de instrucciones

```
LD TRUE
[%MB30:14:=CONCAT(%MB4:6,%MB14:9)]
```

Lenguaje literal estructurado

```
%MB30:14:=CONCAT(%MB4:6,%MB14:9);
```

Ejemplos

Ejemplo: `%MB30:14:=CONCAT(%MB4:6,%MB19:9)`

%MB	*4*	*5*	*6*	*7*	*8*	*9*
	'i'	**'n'**	**'c'**	**'o'**	**'n'**	**Ø**

%MB	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*
	't'	**'e'**	**'s'**	**'t'**	**'a'**	**'b'**	**'l'**	**'e'**	**Ø**

%MB	*30*	*31*	*32*	*33*	*34*	*35*	*36*	*37*	*38*	*39*	*40*	*41*	*42*	*43*
	'i'	**'n'**	**'c'**	**'o'**	**'n'**	**'t'**	**'e'**	**'s'**	**'t'**	**'a'**	**'b'**	**'l'**	**'e'**	**Ø**

Sintaxis

Operadores de concatenación de cadenas

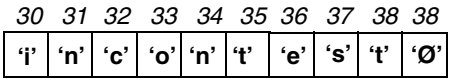
Sintaxis
Result:=**CONCAT**(cadena1, cadena 2)

Operandos de concatenación de cadenas

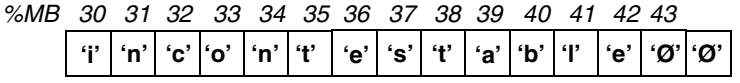
Tipo	**Result (resultado)**	**Cadenas 1 y 2**
Tablas de bytes	%MB:L	%MB:L,%KB;L,Valor inmediato

Nota:

- Si la tabla de resultado es demasiado corta, se trunca y el bit de sistema %S15 se sitúa en 1. `%MB30:10:=CONCAT(%MB4:6, %MB14:9)`



- Si la tabla de resultado es demasiado larga, la cadena se completa con caracteres terminadores 'Ø'. `%MB30:15:=CONCAT(%MB4:6,%MB14:9)`



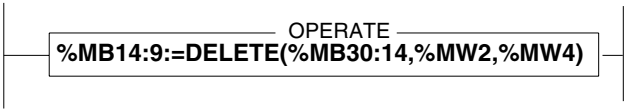
Eliminación de una subcadena de caracteres

Generalidades

Elimina un número determinado de caracteres (zona de longitud L), a partir de una rango determinado (posición del primer carácter que se debe eliminar) en la cadena definida como parámetro. El resultado es una tabla de bytes que contiene una cadena de caracteres.

Estructura

Lenguaje de contactos



Lenguaje de lista de instrucciones

```
LD TRUE
[%MB14:9:=DELETE(%MB30:14,%MW2,%MW4)]
```

Lenguaje literal estructurado

```
%MB14:9:=DELETE(%MB30:14,%MW2,%MW4);
```

Ejemplos

Ejemplo: `%MB14:9:=DELETE(%MB30:14,%MW2,%MW4)`
con %MW2 = 5 (5 caracteres para eliminar) %MW4 = 3 (posición = 3)

%MB	*30*	*31*	*32*	*33*	*34*	*35*	*36*	*37*	*38*	*39*	*40*	*41*	*42*	*43*
	'i'	**'n'**	**'c'**	**'o'**	**'n'**	**'t'**	**'e'**	**'s'**	**'t'**	**'a'**	**'b'**	**'l'**	**'e'**	**Ø**

%MB	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*
	'i'	**'n'**	**'s'**	**'t'**	**'a'**	**'b'**	**'l'**	**'e'**	**Ø**

Sintaxis

Operador de eliminación de una subcadena de caracteres

Sintaxis
Result:=**DELETE**(cadena1, long, pos)

Operandos de eliminación de una subcadena de caracteres

Tipo	**Result (resultado)**	**Cadena**	**Long (longitud), Pos (posición)**
Tablas de bytes	%MB:L	%MB:L,%KB;L,Val. inmediato	-
Palabras indexables	-	-	%MW,%KW,%Xi.T
Palabra no indexable	-	-	%IW,%QW,%SW,%NW, Val.inm.,Expr.num.

Nota: Posibilidad de solapamiento entre los parámetros según los índices de los objetos PL7:
- Tabla que contiene la cadena de origen.
- Tabla que contiene la cadena de resultado.
- Palabra que contiene la longitud que se va a eliminar.
- Palabra que contiene la posición del primer carácter que se va a eliminar.

Una longitud o una posición negativa se interpreta como igual a 0. El parámetro de posición se inicia en el valor 1 correspondiente a la primera posición en la cadena de caracteres.

Inserción de una subcadena de caracteres

Generalidades

Inserción de la subcadena de caracteres definida por el segundo parámetro (cadena2) en la cadena de caracteres definida por el primer parámetro (cadena1). La inserción se realiza en la primera cadena, después del carácter situado en la posición determinada por el parámetro de posición (Pos).
El resultado de la inserción es una nueva cadena de caracteres transferida a una tabla de bytes.

Estructura

Lenguaje de contactos

Lenguaje lista de instrucciones

```
LD TRUE
[%MB2:14:=INSERT(%MB20:9,%MB30:6,%MW40)]
```

Lenguaje literal estructurado

```
%MB2:14:=INSERT(%MB20:9,%MB30:6,%MW40);
```

Ejemplos

Ejemplo: `%MB2:14:=INSERT(%MB20:9,%MB30:6,%MW40)`
con %MW40=posición 2

%MB	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*
	'i'	**'n'**	**'s'**	**'t'**	**'a'**	**'b'**	**'l'**	**'e'**	**Ø**

%MB	*30*	*31*	*32*	*33*	*34*	*35*
	'c'	**'o'**	**'n'**	**'t'**	**'e'**	**Ø**

%MB	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
	'i'	**'n'**	**'c'**	**'o'**	**'n'**	**'t'**	**'e'**	**'s'**	**'t'**	**'a'**	**'b'**	**'l'**	**'e'**	**Ø**

Sintaxis

Operadores de inserción de una subcadena de caracteres

Sintaxis
Result:=INSERT (cadena1, cadena2, pos)

Operandos de inserción de una subcadena de caracteres

Tipo	**Result (resultado)**	**Cadenas 1 y 2**	**Pos (posición)**
Tablas de bytes	%MB:L	%MB:L,%KB;L	-
Palabras indexables	-	-	%MW,%KW,%Xi.T
Palabra no indexable	-	-	%IW,%QW,%SW,%NW, Val.inm.,Expr.num.

Nota:
- El parámetro de posición se inicia en el valor 1 correspondiente a la primera posición de la cadena de caracteres.
- No es posible realizar una inserción al principio de una cadena. Para ello se debe utilizar la función CONCAT.
- Si la tabla es demasiado larga, se completa con caracteres de tipo terminador.
- Palabra que contiene la posición del carácter que se debe eliminar.
- El bit de sistema %S15 se sitúa en 1 en los siguientes casos:
  - El valor del parámetro de posición es negativo o igual a 0. En tal caso, se interpreta como si fuera igual a 0 y la tabla de resultado contendrá una cadena vacía (compuesta de terminadores).
  - La tabla de resultado es demasiado corta, por lo que se trunca.

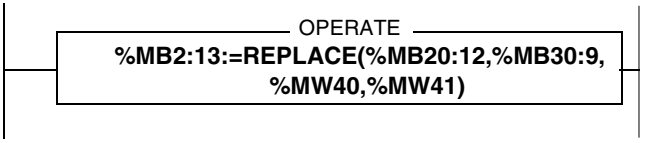
Sustitución de una subcadena de caracteres

Generalidades

Sustituye un segmento de un cadena de caracteres definida en la tabla de origen (cadena1) por una subcadena de caracteres definida en la tabla de sustitución (cadena2). La sustitución que se va a llevar a cabo se define a través de los parámetros de posición (pos.) y de longitud (long.). Esta longitud corresponde a la longitud de la cadena que desaparece y no a la longitud de la subcadena que la sustituye.

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD TRUE
[%MB2:13:=REPLACE(%MB20:12,%MB30:9,%MW40,%MW41)]
```

Lenguaje literal estructurado

```
%MB2:13:=REPLACE(%MB20:12,%MB30:12,%MW40,%MW41);
```

Ejemplos

Ejemplo: `%MB2:13:=REPLACE(%MB20:12,%MB30:12,%MW40,%MW41)` con `%MW40=3` (longitud=3) y `%MW41=9` (posición 9)

	%MB	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*
Chaîne 1		**'m'**	**'i'**	**'s'**	**'e'**	**'_'**	**'e'**	**'n'**	**'_'**	**'r'**	**'u'**	**'n'**	**Ø**

	%MB	*30*	*31*	*32*	*33*	*34*	*35*	*36*	*37*	*38*
Chaîne 2		**'s'**	**'t'**	**'o'**	**'p'**	**Ø**	**'r'**	**'u'**	**'n'**	**Ø**

%MB	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*
	'm'	**'i'**	**'s'**	**'e'**	**'_'**	**'e'**	**'n'**	**'_'**	**'s'**	**'t'**	**'o'**	**'p'**	**Ø**

Sintaxis

Operadores de sustitución de una subcadena de caracteres

Sintaxis
Result:=REPLACE (cadena1, cadena2, long., pos.)

Operandos de sustitución de una subcadena de caracteres

Tipo	**Result (resultado)**	**Cadenas 1 y 2**	**Long (longitud), Pos (posición)**
Tablas de bytes	%MB:L	%MB:L,%KB;L	-
Palabras indexables	-	-	%MW,%KW,%Xi.T
Palabra no indexable	-	-	%IW,%QW,%SW,%NW, Val.inm.,Expr.num.

Nota:
- El parámetro de posición se inicia en el valor 1 correspondiente a la primera posición de la cadena de caracteres.
- Si la tabla de salida es demasiado larga, la cadena se completa con caracteres de tipo terminador.

El bit de sistema %S15 se sitúa en 1 en los siguientes casos:
- Si el valor del parámetro de posición es negativo o igual a 0. En tal caso, se interpreta como si fuera igual a 0 y la tabla de resultado contendrá una cadena vacía (compuesta de terminadores).
- Si la posición pasada a parámetro es superior o igual a la longitud de la cadena de origen, la tabla del resultado contendrá una cadena vacía (compuesta de terminadores).
- Si la tabla de resultado es demasiado corta, se trunca.
- Palabra que contiene la posición del carácter que se debe eliminar.
- Si la posición del primer terminador de la cadena es inferior o igual a la posición del primer carácter que se debe sustituir, la tabla de salida es una copia de la tabla de origen hasta el terminador, que se completa con caracteres terminadores.

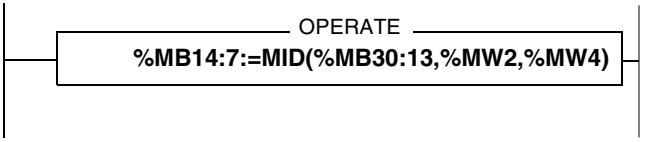
Extracción de una subcadena de caracteres

Generalidades

Extracción de un número determinado de caracteres en una cadena de origen pasada a parámetro (cadena).
El rango del primer carácter que debe extraerse viene determinado por el parámetro de posición (pos), y el número de caracteres que debe extraerse por el parámetro de longitud (long.). La cadena extraída se guarda en una tabla de bytes (result.).

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD TRUE
[%MB14:7:=MID(%MB30:13,%MW2,%MW4)]
```

Lenguaje literal estructurado

```
%MB14:7:=MID(%MB30:13,%MW2,%MW4);
```

Ejemplos

Ejemplo: `%MB14:7:=MID(%MB30:13,%MW2,%MW4)`
con `%MW2=4` (longitud) y `%MW4=9` (posición)

%MB	*30*	*31*	*32*	*33*	*34*	35	36	37	38	39	40	41	42
	'm'	**'i'**	**'s'**	**'e'**	**'_'**	**'e'**	**'n'**	**'_'**	**'s'**	**'t'**	**'o'**	**'p'**	**Ø**

Résultat :

%MB	*14*	*15*	*16*	*17*	*18*	*19*	*20*
	's'	**'t'**	**'o'**	**'p'**	**Ø**	**Ø**	**Ø**

Sintaxis

Operadores de extracción de una subcadena de caracteres

Sintaxis
Result:=MID (cadena, long., pos.)

Operandos de extracción de una subcadena de caracteres

Tipo	**Result (resultado)**	**Cadena**	**Long (longitud), Pos (posición)**
Tablas de bytes	%MB:L,Val. inm.	%MB:L,%KB;L	-
Palabras indexables	-	-	%MW,%KW,%Xi.T
Palabra no indexable	-	-	%IW,%QW,%SW,%NW, Val.inm.,Expr.num.

Nota:
- El parámetro de posición se inicia en el valor 1 correspondiente a la primera posición de la cadena de caracteres.
- Si la tabla de salida es demasiado larga, la cadena se completa con caracteres de tipo terminador.
- Si la longitud pasada a parámetro es superior al tamaño de la cadena de origen, la tabla del resultado contendrá esta última.
- Si se alcanza el último elemento de la tabla o el terminador de cadena antes de haber extraído el número de caracteres definido por el parámetro de longitud, la extracción se detiene en los mismos.

El bit de sistema %S15 se sitúa en 1 en los siguientes casos:
- Si el valor del parámetro de longitud que se debe extraer es negativo o nulo. En este caso se interpreta como si fuera igual a 0 y la tabla del resultado contendrá una cadena vacía (compuesta de terminadores).
- Si el valor del parámetro de posición del principio de la extracción es nulo, superior o igual a la longitud de la tabla, o bien superior o igual a la posición del primer terminador. En tal caso, la tabla del resultado contendrá una cadena vacía (compuesta de terminadores).
- Si la tabla del resultado es demasiado corta, se trunca.

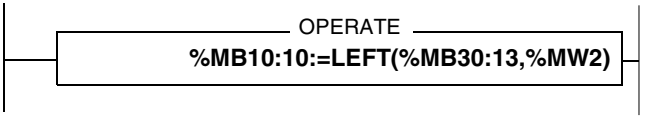
Extracción de caracteres

Generalidades

Extracción de un número determinado de caracteres situados en el extremo izquierdo (LEFT) o extremo derecho (RIGHT) de una cadena de origen pasada a parámetro (cadena).
El número de caracteres que debe extraerse viene determinado por el parámetro de longitud (long.). La cadena extraída se guarda en una tabla de bytes (result.).

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD TRUE
[%MB10:10:=LEFT(%MB30:13,%MW2)]
```

Lenguaje literal estructurado

```
%MB10:10:=LEFT(%MB30:13,%MW2);
```

Ejemplos

Ejemplo: `%MB10:10:=LEFT(%MB30:13,%MW2)`
con `%MW2=8` (longitud)

%MB	*30*	*31*	*32*	*33*	*34*	*35*	*36*	*37*	*38*	*39*	*40*	*41*	*42*
	'm'	**'i'**	**'s'**	**'e'**	**'_'**	**'e'**	**'n'**	**'_'**	**'s'**	**'t'**	**'o'**	**'p'**	**Ø**

Résultat :

%MB	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*
	'm'	**'i'**	**'s'**	**'e'**	**'_'**	**'e'**	**'n'**	**'_'**	**Ø**	**Ø**

Sintaxis

Operadores de extracción de caracteres

Sintaxis
Result:=LEFT (cadena, long)
Result:=RIGHT (cadena, long)

Operandos de extracción de caracteres

Tipo	**Result (resultado)**	**Cadena**	**Long (longitud), Pos (posición)**
Tablas de bytes	%MB:L	%MB:L,%KB;L,Val. inm.	-
Palabras indexables	-	-	%MW,%KW,%Xi.T
Palabra no indexable	-	-	%IW,%QW,%SW,%NW, Val.inm.,Expr.num.

Nota:
- Si la tabla de salida es demasiado larga, la cadena del resultado se completa con caracteres de tipo terminador.
- Si la longitud pasada a parámetro es superior al tamaño de la cadena de origen, la tabla del resultado contendrá esta última.

El bit de sistema %S15 se sitúa en 1 en los siguientes casos:
- Si el valor del parámetro de longitud que se debe extraer es negativo o nulo. En tal caso, la tabla del resultado contendrá una cadena vacía (compuesta de terminadores).
- Si el valor del parámetro de posición del principio de la extracción es nulo, superior o igual a la longitud de la tabla, o bien superior o igual a la posición del primer terminador. En tal caso, la tabla del resultado contendrá una cadena vacía (compuesta de terminadores).
- Si la tabla del resultado es demasiado corta, se trunca.

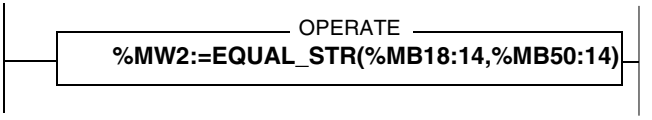
Comparación de dos cadenas de caracteres

Generalidades

Comparación de dos cadenas de caracteres El resultado es una palabra que contiene la posición del primer carácter diferente.
En caso de equivalencia perfecta entre las dos cadenas de caracteres, el resultado es -1.

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD TRUE
[%MW2:=EQUAL_STR(%MB18:14,%MB50:14)]
```

Lenguaje literal estructurado

```
%MW2:=EQUAL_STR(%MB18:14,%MB50:14);
```

Ejemplos

Ejemplo: `%MW2:=EQUAL_STR(%MB18:14,%MB50:14)`
con

%MB	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*
	'a'	**'b'**	**'c'**	**'d'**	**'e'**	**'f'**	**'g'**	**'h'**	**'i'**	**'p'**	**'w'**	**'x'**	**'y'**	**'z'**

Résultat :

%MB	*50*	*51*	*52*	*53*	*54*	*55*	*56*	*57*	*58*	*59*	*60*	*61*	*62*	*63*
	'a'	**'b'**	**'c'**	**'d'**	**'?'**	**'f'**	**'g'**	**'h'**	**'Ø'**	**'v'**	**'w'**	**'x'**	**'y'**	**'z'**

==> MW2:= 5

Sintaxis

Operadores de comparación de dos cadenas de caracteres

Sintaxis
Result:=EQUAL_STR (cadena, cadena2)

Operandos de comparación de dos cadenas de caracteres

Tipo	**Result (resultado)**	**Cadenas 1 y 2**
Tablas de bytes	-	%MB:L,%KB;L,Val. inm.
Palabras indexables	%MW	-
Palabra no indexable	%QW,%SW,%NW	-

Nota:
- Una longitud o una posición negativa se interpreta como si fuera igual a 0.
- Las letras en mayúsculas son diferentes de las letras en minúsculas.

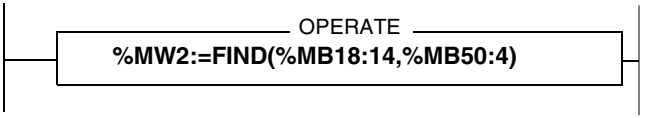
Búsqueda de una subcadena de caracteres

Generalidades

Búsqueda de la subcadena de caracteres definida por el segundo parámetro en la cadena de caracteres definida por el primer parámetro.
El resultado es una palabra que contiene la posición, en la primera cadena, del principio de la subcadena buscada.
Si la búsqueda es infructuosa, el resultado es -1.

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD TRUE
[%MW2:=FIND(%MB18:14,%MB50:4)]
```

Lenguaje literal estructurado

```
%MW2:=FIND(%MB18:14,%MB50:4);
```

Ejemplos

Ejemplo: `%MW2:=FIND(%MB18:14,%MB50:4)` con:

%MB	*18*	*19*	*20*	*21*	*22*	*23*	*24*	*25*	*26*	*27*	*28*	*29*	*30*	*31*
	‘a’	‘b’	‘c’	‘d’	‘e’	‘f’	‘g’	‘h’	‘i’	Ø	‘w’	‘x’	‘y’	‘z’

%MB	*50*	*51*	*52*	*53*
	‘f’	‘g’	‘h’	Ø

==> MW2:= 6 Indica que el principio de la cadena buscada se sitúa a partir del sexto carácter.

Sintaxis

Operadores de búsqueda de subcadenas de caracteres

Sintaxis
Result:=FIND (cadena1, cadena2)

Operandos de búsqueda de subcadenas de caracteres

Tipo	**Result (resultado)**	**Cadenas 1 y 2**
Palabras indexables	%MW	-
Palabra no indexable	%QW,%SW,%NW	-
Tablas de bytes	-	%MB:L,%KB;L,Val. inm.

Nota: Una longitud o una posición negativa se interpreta como si fuera igual a 0.

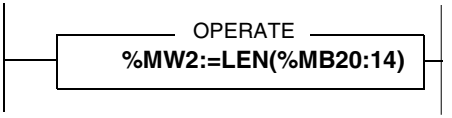
Longitud de una cadena de caracteres

Generalidades

Esta función devuelve la longitud de la cadena de caracteres pasada a parámetros, es decir, el número de caracteres situados antes del terminador.

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

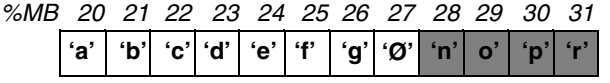
```
LD TRUE
[%MW2:=LEN(%MB20:14)]
```

Lenguaje literal estructurado

```
%MW2:=LEN(%MB20:14);
```

Ejemplos

Ejemplo: `%MW2:=LEN(%MB20:14` con:))



==> `MW2:= 7`

Sintaxis

Operador de una longitud de cadena de caracteres

Sintaxis
Result:=LEN (cadena)

Operandos de una longitud de cadena de caracteres

Tipo	**Result (resultado)**	**Cadenas 1 y 2**
Palabras indexables	%MW	-
Palabra no indexable	%QW,%SW,%NW	-
Tablas de bytes	-	%MB:L,%KB;L,Val. inm.

Nota: Si no se encuentra ningún terminador, esta función devuelve el tamaño de la tabla como se indica en: "Formato de una cadena de caracteres o tabla de caracteres" (Véase *Formato de una cadena de caracteres o tabla de caracteres, p. 187*).

2.8 Instrucciones de gestión del tiempo: fechas, horas, duraciones

Presentación

Objeto de este apartado

Este apartado describe las instrucciones de gestión del tiempo: fechas, horas, duraciones del lenguaje PL7

Contenido

Esta sección contiene los siguientes apartados:

Formato de los parámetros de las instrucciones de gestión del tiempo

Generalidades

Los parámetros de Fecha, Hora y Duración que utilizan estas instrucciones corresponden a los tipos de formato definidos por la norma IEC1131-3.

Formato Duración (tipo TIME)

Este formato permite codificar duraciones expresadas en décimas de segundo y corresponde al formato TIME de la norma.
Estos valores tienen el formato: **sssssssss.d**

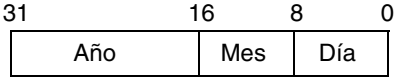
Lo cual proporciona, por ejemplo: 3674.3 , para 1 hora, 1 minuto, 14 segundos y 3 décimas
El valor se codifica en 32 bits (una palabra doble) y los límites se establecen a [0, 4294967295] décimas de segundo, lo que representa aproximadamente 13 años y 7 meses.

Nota: Sólo están autorizados los valores comprendidos en el intervalo [00:00:00, 23:59:59].

Formato Fecha (tipo DATE)

Este formato permite codificar el año, el mes y el día. Corresponde al formato DATE de la norma.
El valor tiene el formato: **yyyy-mm-dd**

Lo cual proporciona, por ejemplo: 1984-06-25
El valor se codifica en BCD con 32 bits (una palabra doble) y 3 campos:



Año: 4 dígitos
Mes: 2 dígitos
Día: 2 dígitos

Ejemplo expresado en hexadecimal:

19H	84H	06H	25H

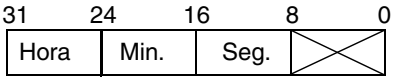
= 1984-06-25

Nota: Sólo están permitidos los valores comprendidos en el intervalo [1990-01-01, 2099-12-31].

Formato Hora del día (tipo TOD)

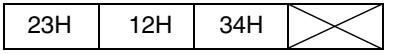
Este formato permite codificar la hora, los minutos y los segundos. Corresponde al formato TIME_OF_DAY de la norma.
El valor tiene el formato: **hh:mm:ss**

Lo cual proporciona, por ejemplo: 23:12:34
El valor se codifica en BCD con 32 bits (una palabra doble) y 3 campos:) :



Horas: 2 dígitos (palabras de peso significativo)
Minutos: 2 dígitos (palabras de peso significativo)
Segundos: 2 dígitos (palabras de peso menos significativo)

Ejemplo expresado en hexadecimal:



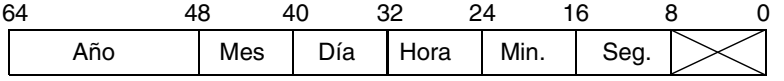
= 23:12:34

Nota: Sólo están autorizados los valores comprendidos en el intervalo [00:00:00, 23:59:59].

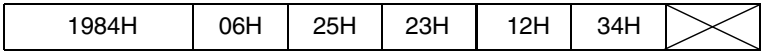
Formato Fecha y hora (tipo DT)

Este formato permite codificar el año, el mes, el día, la hora, los minutos y los segundos. Corresponde al formato DATE_AND_TIME de la norma.
El valor tiene el formato: **yyyy-mm-dd-hh:mm:ss**

Lo cual proporciona, por ejemplo: 1984-06-25-23:12:34
El valor se codifica en BCD con 64 bits (una tabla de palabras de longitud 4):



Ejemplo expresado en hexadecimal:

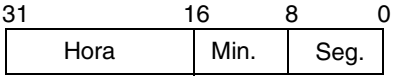


Nota: Sólo están permitidos los valores comprendidos en el intervalo [1990-01-01-00:00:00, 2099-12-31-23:59:59].

Formato Hora, Minuto, Segundo (tipo HMS)

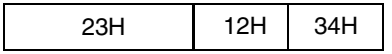
Este formato, utilizado exclusivamente por la función TRANS_TIME, permite codificar las horas, los minutos y los segundos.
El valor tiene el formato: **hh:mm:ss**

Lo cual proporciona, por ejemplo: 23:12:34
El valor se codifica en BCD con 32 bits (una palabra doble) y 3 campos:



Horas: 4 dígitos (palabras de peso significativo)
Minutos: 2 dígitos (palabras de peso menos significativo)
Segundos: 2 dígitos (palabras de peso menos significativo)

Ejemplo expresado en hexadecimal:



= 23:12:34

Utilización de los bits y las palabras de sistema

Bit de sistema %S17

El bit de sistema **%S17** se sitúa en los siguientes casos:
- Resultado de una operación fuera del intervalo de valores permitido
- Un parámetro de entrada no se puede interpretar y es incoherente con el formato deseado (DATE, DT o TOD)
- Operación en formato Hora del día (TOD) que conlleva un cambio de día
- Conflicto de acceso al reloj-calendario

Bit de sistema %S15

El bit de sistema **%S15** se pone a 1 al escribir una cadena en una tabla cuando aquella es más larga que el tamaño de esta última.

Palabras de sistema

Las palabras de sistema:
- **%SD18**: contador de tiempo absoluto; permite asimismo efectuar cálculos de duración (incrementado cada 1/10 de segundo por el sistema)
- **%SW49** a **%SW53** (Véase *Descripción de las palabras de sistema %SW48 a %SW59, p. 332*) se pueden utilizar también para visualizar fechas

Función de reloj-calendario

Generalidades

Esta función permite activar acciones en horarios y fechas predefinidos o calculados.
Pone a 1 el parámetro de salida OUT si la fecha establecida por el reloj del autómata durante la llamada de la función forma parte del periodo programado en los parámetros de entrada.

Sintaxis

Operador de la función de reloj-calendario

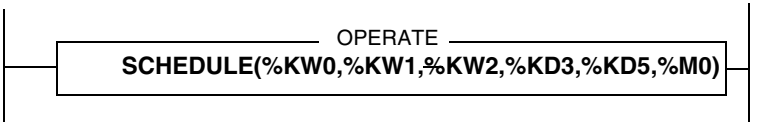
SCHEDULE (DBEG, DEND, WEEK, HBEG, HEND, OUT)

Características de los parámetros

Salida	**OUT**	Bit que contiene el resultado de las comparaciones realizadas por la función de reloj-calendario: a 1 durante los periodos definidos por los parámetros.
Fecha de inicio	**DBEG**	Palabra que codifica la fecha de inicio del periodo (mes-día) en BCD (valores límite: de 01-01 a 12-31)
Fecha de finalización	**DEND**	Palabra que codifica la fecha de finalización del periodo (mes-día) en BCD (valores límite: de 01-01 a 12-31)
Día de la semana	**WEEK**	Palabra que codifica los días de la semana tomados en cuenta en el periodo definido por los parámetros DBEG y DEND. Los 7 bits menos significativos representan los siete días de la semana: bit 6 = lunes, bit 5 = martes,... bit 0 = domingo.
Hora de inicio	**HBEG**	Palabra doble que codifica la hora de inicio del periodo del día (horas-minutos-segundos) en BCD de formato de hora del día (tipo: TOD). Valores límite: 00:00:00, 23:59:59
Hora de finalización	**HEND**	Palabra doble que codifica la hora de finalización del periodo del día (horas-minutos-segundos) en BCD de formato de hora del día (tipo: TOD). Valores límite: 00:00:00, 23:59:59

Estructura

Lenguaje de contactos



Lenguaje de lista de instrucciones

```
LD TRUE
[SCHEDULE(%KW0,%KW1,%KW2,%KD3,%KD5,%M0)]
```

Lenguaje literal estructurado

```
SCHEDULE(%KW0,%KW1,%KW2,%KD3,%KD5,%M0);
```

Ejemplos

Ejemplo: Programación de dos intervalos horarios no continuos

SCHEDULE	(16#0501, 16#1031, 2#0000000001111100, 16"08300000, 16#12000000, %M0);	(*fecha de inicio: 1 de mayo*) (*fecha de finalización: 31 de octubre*) (*de lunes a viernes*) (*hora de inicio: 8 h 30*) (*hora de finalización: 18 h*) (*resultado en %M0*)
SCHEDULE	(16#0501, 16#1031, 2#0000000001111100, 16"14000000, 16#18000000, %M1);	(*fecha de inicio: 1 de mayo*) (*fecha de finalización: 31 de octubre*) (*de lunes a viernes*) (*hora de inicio: 14 h*) (*hora de finalización: 18 h*) (*resultado en %M1*)
	%Q0.0:=%M0 OR %M1;	

Operandos

Operandos de la función de reloj-calendario

Tipo	DBEG,DEND,WEEK	HBEG,HEND	OUT
Palabras indexables	%MW,%KW,%Xi.T	-	-
Palabras no indexables	%IW,%QW,%SW,%NW,Val.imm.,expr.num.	-	-
Palabras dobles indexables	-	%MD,%KD	-
Palabras dobles no indexables	-	%ID,%QD,Val.imm., Expr.num.	-
Bits	-	-	%I,%Q,%M,%S, %BLK,%*:Xk,%X

Nota:

- Los dos parámetros DBEG y DEND establecen un intervalo de días durante el año; este intervalo puede ser de hasta dos años naturales. Ejemplo: del 10 de octubre al 7 de abril. El 29 de febrero puede utilizarse en este periodo, aunque se ignorará en los años no bisiestos.
- Los dos parámetros HBEG y HEND establecen un intervalo horario durante el día; este intervalo puede ser de hasta dos días. Ejemplo: de 22 h a 6 h 10 min 20 s.
- Si una de las fechas DBEG y DEND o una de las horas HBEG y HEND no es correcta, es decir, no corresponde a una fecha o una hora real, la salida OUT se pondrá a 0 y el bit %S17 se pondrá a 1.
- Si el autómata de destino no posee un reloj interno (como es el caso de TSX37-10), la salida se pondrá a 0 y el bit de sistema %S17 se pondrá a 1.
- Es posible reducir la carga del procesador del autómata cuando la precisión no resulta esencial durante la modulación de la llamada de la función SCHEDULE por parte del bit de sistema %S6 o %S7.
- Para un intervalo horario programado de hasta dos días, por ejemplo, de 15 h (día 1) a 8 h (día 2), las condiciones se validarán de nuevo de 15 h a 24 h el segundo día. Si únicamente desea efectuar este intervalo horario una vez a la semana, se recomienda utilizar dos veces la función SCHEDULE con un intervalo horario de 15 h a 24 h (día 1) y de 0 h a 8 h (día 2).

Función Reloj-calendario de red

Generalidades

La función R_NTPC permite recuperar la fecha y hora en un servidor NTP conforme a dos formatos:

- un formato de visualización,
- un formato para realizar cálculos.

Nota: Esta función requiere la conexión a una red Ethernet que permita el acceso a un servidor NTP.

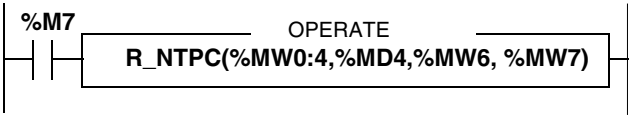
Sintaxis

Operador de la función Reloj-calendario de red

Sintaxis
R_NTPC(N_DT,SEC,MSEC,Status)

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD %M7
[R_NTPC(%MW0:4,%MD4,%MW6,%MW7)]
```

Lenguaje literal estructurado

```
IF %M7 THEN
  R_NTPC(%MW0:4,%MD4,%MW6,%MW7);
END_IF;
```

Operandos

Operandos de la función Reloj-calendario de red

Parámetros	Tipo	Descripción
N_DT	Tabla de cuatro palabras (%MW)	Palabra 1: ● byte 0: reservado ● byte 1: segundos Palabra 2: ● byte 0: minutos ● byte 1: hora Palabra 3: ● byte 0: día ● byte 1: mes Palabra 4: ● byte 0 y 1: año
SEC	DWORD (%MD)	Fecha y la hora que se han convertido en segundos desde el 1 de enero de 1980.
MSEC	WORD (%MW)	Valores de los milisegundos de la hora.
Estado	WORD (%MW)	La variable Status indica la validez del resultado de la función R_NTPC. Su valor es: ● 0 si el resultado no es válido, ● 1 si el resultado es válido y si la precisión de éste es inferior a 10 ms.

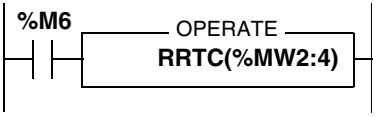
Lectura de la fecha del sistema

Generalidades

Lectura de la fecha del sistema (Real Time Clock) y transferencia al objeto dado en parámetro en formato Fecha y hora (DT).

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD %M6
[RRTC(%MW2:4)]
```

Lenguaje literal estructurado

```
IF %M6 THEN
 RRTC(%MW2:4);
END_IF;
```

Ejemplos

Ejemplo: `RRTC(%MW2:4)`
El resultado se transfiere a la tabla de palabras internas de longitud 4: %MW2 a %MW5.

Sintaxis

Operador de lectura de la fecha del sistema

Sintaxis
RRTC(fecha)

Operandos de lectura de la fecha del sistema

Tipo	Fecha
Tabla de 4 palabras con formato de fecha y hora	%MW:4

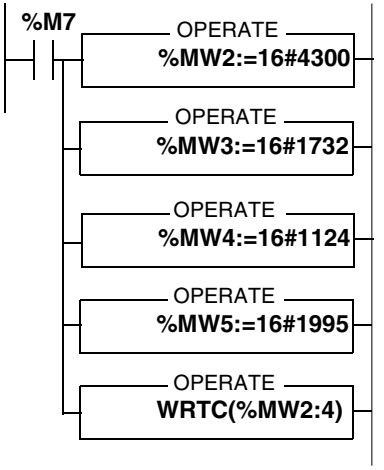
Actualización de la fecha del sistema

Generalidades

Actualización de la fecha del sistema (Real Time Clock) y transferencia al objeto dado en parámetro en formato Fecha y hora (DT).

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD %M7
[%MW2:=16#4300]
[%MW3:=16#1732]
[%MW4:=16#1124]
[%MW5:=16#1995]
[WRTC(%MW2:4)]
```

Lenguaje literal estructurado

```
IF %M7 THEN
 %MW2:=16#4300;
 %MW3:=16#1732;
 %MW4:=16#1124;
 %MW5:=16#1995;
 WRTC(%MW2:4);
END_IF;
```

Ejemplos

Ejemplo: La nueva fecha se carga en una tabla de palabras internas de longitud 4 %MW2:4 y a continuación se envía al sistema a través de la función WRTC.

Sintaxis

Operador de actualización de la fecha del sistema

Sintaxis
WRTC(fecha)

Operandos de actualización de la fecha del sistema

Tipo	**Fecha**
Tabla de 4 palabras	%MW:4,%KW:4 con formato de fecha y hora

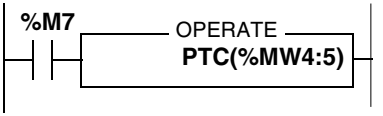
Lectura de fecha y código de parada

Generalidades

Lectura de la fecha de la última parada del autómata y del código que especifica la causa de la misma (en la 5ª palabra, equivale a %SW58 (Véase *Descripción de las palabras de sistema %SW48 a %SW59, p. 332*))

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD %M7
[PTC(%MW4:5)]
```

Lenguaje literal estructurado

```
IF %M7 THEN
  PTC(%MW4:5);
END_IF;
```

Ejemplos

Ejemplo: `PTC(%MW4:5)`
El resultado se transfiere a la tabla de palabras internas de longitud 5: %MW4 a %MW8

Sintaxis

Operador de lectura de fecha y código de parada

Sintaxis
PTC (fecha)

Operandos de lectura de fecha y código de parada

Tipo	Fecha
Tabla de 5 palabras con formato de fecha y hora	%MW:5

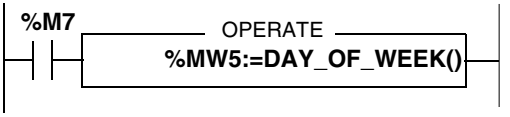
Lectura del día de la semana

Generalidades

Esta función proporciona como resultado el día actual de la semana con formato de una cifra de 1 a 7 transferida a una palabra (1 = Lunes, 2 = Martes,3 = Miércoles, 4 = Jueves, 5 = Viernes, 6 = Sábado, 7 = Domingo).

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD %M7
[%MW5:=DAY_OF_WEEK()]
```

Lenguaje literal estructurado

```
IF %M7 THEN
  %MW5:=DAY_OF_WEEK();
END_IF;
```

Ejemplos

Ejemplo: `%MW5:=DAY_OF_WEEK()`
%MW5:=4 corresponde al jueves

Sintaxis

Operador de lectura del día de la semana

Sintaxis
Result:=**DAY_OF_WEEK()**

Operandos de lectura del día de la semana

Tipo	**Result (resultado)**
Palabras indexables	%MW
Palabras no indexables	%QW,%SW,%NW

Nota: Si la función no puede actualizar el resultado debido a un error de acceso al reloj-calendario, el resultado devuelto es 0 y el bit de sistema %S17 se sitúa en 1.

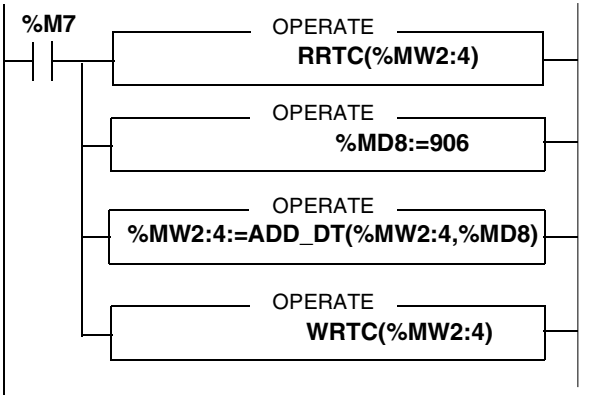
Suma / Resta de una duración a una fecha

Generalidades

Suma o resta de una duración (en décimas de segundo) (In2) a una fecha de origen (In1). El resultado es una fecha nueva que se transfiere a una tabla de 4 palabras.
ADD_DT () = Suma de una duración
SUB_DT () = Resta de una duración

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD %M7
[RRTC(%MW2:4)]
[%MD8:=906]
[%MW2:4:=ADD_DT(%MW2:4,%MD8)]
[WRTC(%MW2:4)]
```

Lenguaje literal estructurado

```
IF %M7 THEN
 RRTC(%MW2:4);
 %MD8:=906;
 %MW2:4:=ADD_DT(%MW2:4,%MD8);
 WRTC(%MW2:4);
END_IF;
```

Ejemplos

Ejemplo: `%MW2:4:=ADD_DT(%MW2:4,%MD8)`
%MW2:4:= Fecha de origen
%MD8:=906 (906 décimas de segundo redondeadas a 1 min. 31s)
%MW2:4:= Fecha nueva

Sintaxis

Operadores de suma/resta de una duración a una fecha

Sintaxis
Result:=**ADD_DT**(In1, In2)
Result:=**SUB_DT**(In1, In2)

Operandos de suma/resta de una duración a una fecha

Tipo	**Result (resultado)**	**In1 (fecha de origen)**	**In2 (duración)**
Tablas de 4 palabras con formato de fecha y hora	%MW4	%MW4:4,%KW:4	-
Palabras dobles indexables	-	-	%MD,%KD
Palabras dobles no indexables	-	-	%ID,%QD,Val.inm., Expr.num.

Nota:
- El principio del redondeo se aplica al parámetro "duración" (expresado en décimas de segundo) para poder realizar la suma o la resta a la fecha (precisión al segundo).
  - ssssssssss.0 a ssssssssss.4 redondeado a ssssssssss.0
  - ssssssssss.5 a ssssssssss.9 redondeado a ssssssssss.0 +1.0
- Es necesario prever la gestión de los años bisiestos en la aplicación.
- Si el resultado de la operación está fuera del intervalo de valores permitidos, el bit de sistema %S17 se sitúa en 1 y el valor del resultado es igual al límite mínimo (para SUB_DT) o permanece bloqueado en el máximo (para ADD_DT).
- Si el parámetro de entrada "fecha de origen" no se puede interpretar o es incoherente con el formato DT (DATE_AND_TIME), el bit de sistema %S17 se sitúa en 1 y el valor del resultado es igual a 0001-01-01-00:00:00.

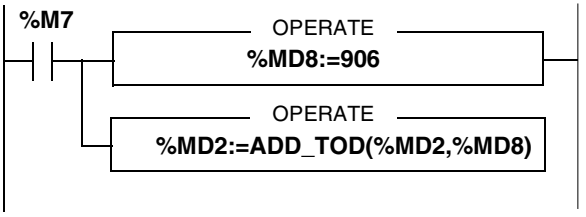
Suma / Resta de una duración a una hora del día

Generalidades

Suma o resta de una duración a una hora del día El resultado es una hora del día nueva que se transfiere a una palabra doble.
ADD_TOD () = Suma de una duración
SUB_TOD () = Resta de una duración

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD %M7
[%MD8:=906]
[%MD2:=ADD_TOD(%MD2,%MD8)]
```

Lenguaje literal estructurado

```
IF %M7 THEN
 %MD8:=906;
 %MD2:=ADD_TOD(%MD2,%MD8);
END_IF;
```

Ejemplos

Ejemplo: `%MD2:=ADD_TOD(%MD2,%MD8)`
%MD2:= Hora de origen (ej: 12:30:00)
%MD8:= 906 (906 décimas de segundo redondeadas a 1 min. 31s)
%MD2:= Hora nueva (ej: 13:31:31)

Sintaxis

Operadores de suma/resta de una duración a una hora del día

Sintaxis
Result:=**ADD_TOD**(In1, In2)
Result:=**SUB_TOD**(In1, In2)

Operandos de suma/resta de una duración a una hora del día

Tipo	**Result (resultado)**	**In1 (hora de origen) e In2 (duración)**
Palabras dobles indexables	%MD	%MD,%KD
Palabras dobles no indexables	%QD	%ID,%QD,Val.inm.,Expr.num.

result e **In1** estén en formato TOD, **In2** está en formato de duración.

Nota:
- El principio del redondeo se aplica al parámetro "duración" (expresado en décimas de segundo) para poder realizar la suma o la resta a la fecha (precisión al segundo).
  - sssssssss.0 a sssssssss.4 redondeado a ssssssss.0
  - sssssssss.5 a sssssssss.9 redondeado a ssssssss.0 +1.0
- Se produce un cambio de día si el resultado de la operación está fuera del intervalo de valores permitidos. En tal caso, el bit de sistema %S17 se sitúa en 1 y el valor del resultado se puede interpretar con un módulo 24:00:00.
- Si el parámetro de entrada "hora del día" no se puede interpretar en formato TOD, el bit de sistema %S17 se sitúa en 1 y el resultado es igual a 00:00:00.

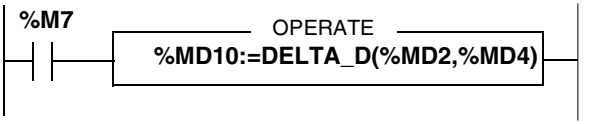
Diferencia entre dos fechas (sin hora)

Generalidades

Calcula la diferencia de tiempo entre dos fechas. El resultado, en valor absoluto, se transfiere a una palabra doble.

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD %M7
[%MD10:=DELTA_D(%MD2,%MD4)]
```

Lenguaje literal estructurado

```
IF %M7 THEN
 %MD10:=DELTA_D(%MD2,%MD4);
END_IF;
```

Ejemplos

```
%MD10:=DELTA_D(%MD2,%MD4)
```

%MD2:= Fecha número1 (ej: 1994-05-01)
%MD4:= Fecha número2 (ej: 1994-04-05)
==> %MD10:= 22464000 (==> diferencia = 26 días)

Sintaxis

Operador de diferencia entre dos fechas (sin hora)

Sintaxis
Result:=**DELTA_D**(Fecha1, Fecha2)

Operandos de diferencia entre dos fechas (sin hora)

Tipo	**Result (resultado)**	**Fechas 1 y 2**
Palabras dobles indexables	%MD	%MD,%KD
Palabras dobles no indexables	%QD	%ID,%QD,Val.inm.,Expr.num.

result está en formato TIME, **Fecha 1** y **2** están en formato DATE.
El formato TIME está definido con una precisión en décimas de segundos. El formato DATE se define con una precisión en días. La diferencia de tiempo calculada será múltiplo de 864000 (= 1día = 24 h x 60 mn x 60 s x 10 décimas).

Nota:
- Se produce un rebasamiento si el resultado supera el valor máximo permitido para una duración (TIME). En ese caso, el resultado es igual a 0 y el bit de sistema %S18 se sitúa en 1.
- Si alguno de los parámetros de entrada no se puede interpretar y es incoherente con el formato DATE, el bit de sistema %S17 se sitúa en 1 y el resultado es igual a 0.

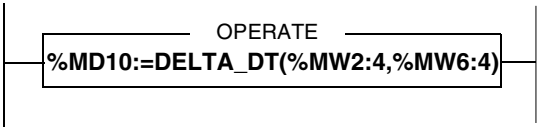
Diferencia entre dos fechas (con hora)

Generalidades

Calcula la diferencia de tiempo entre dos fechas. El resultado, en valor absoluto, se transfiere a una palabra doble.

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD TRUE
[%MD10:=DELTA_DT(%MW2:4,%MW6:4)]
```

Lenguaje literal estructurado

```
%MD10:=DELTA_DT(%MW2:4,%MW6:4);
```

Ejemplos

```
%MD10:=DELTA_DT(%MW2:4,%MW6:4)
```

%MW2:4:= Fecha número1 (ej: 1994-05-01-12:00:00)
%MW6:4:= Fecha número2 (ej: 1994-05-01-12-01-30)
==> %MD10:= 900 (==> diferencia = 1 minuto y 30 segundos)

Sintaxis

Operador de diferencia entre dos fechas (con hora)

Sintaxis
Result:=**DELTA_DT**(Fecha1, Fecha2)

Operandos de diferencia entre dos fechas (con hora)

Tipo	**Result (resultado)**	**Fechas 1 y 2**
Palabras dobles indexables	%MD	-
Palabras dobles no indexables	%QD	-
Tabla de 4 palabras con formato DT	-	%MW:4,%KW:4

result está en formato TIME, **Fecha 1** y **2** están en formato DT.
El formato TIME está definido con una precisión en décimas de segundos. El formato DT se define con una precisión en segundos. La diferencia de tiempo calculada será múltiplo de 10.

Nota:
- Se produce un rebasamiento si el resultado supera el valor máximo permitido para una duración (TIME). En ese caso, el resultado es igual a 0 y el bit de sistema %S18 se sitúa en 1.
- Si alguno de los parámetros de entrada no se puede interpretar y es incoherente con el formato DT, el bit de sistema %S17 se sitúa en 1 y el resultado es igual a 0.

Diferencia entre dos horas

Generalidades

Calcula la diferencia de tiempo entre dos horas del día. El resultado se transfiere a una palabra doble en valor absoluto que proporciona una duración.

Estructura

Lenguaje de contactos

Lenguaje lista de instrucciones

```
LD TRUE
[%MD10:=DELTA_TOD(%MD2,%MD4)]
```

Lenguaje literal estructurado

```
%MD10:=DELTA_TOD(%MD2,%MD4);
```

Ejemplos

```
%MD10:=DELTA_TOD(%MD2,%MD4)
```

%MD2:= Hora1 (ej: 02:30:00)
%MD4:= Hora2 (ej: 02 41 00)
==> %MD10:= 6600 (==> diferencia = 11 minutos)

Sintaxis

Operador de diferencia entre dos horas

Sintaxis
Result:=**DELTA_TOD**(Fecha1, Fecha2)

Operandos de diferencia entre dos horas

Tipo	**Result (resultado)**	**Horas 1 y 2**
Palabras dobles indexables	%MD	%MD,%KD
Palabras dobles no indexables	%QD	%ID,%QD,Valor inmediato, Expr. numérica

result está en formato TIME, **Hora 1** y **2** están en formato TOD.

El formato TIME está definido con una precisión en décimas de segundos. El formato TOD se define con una precisión en segundos. La diferencia de tiempo calculada será múltiplo de 10.

Nota: Si alguno de los parámetros de entrada no se puede interpretar y es incoherente con el formato TOD, el bit de sistema %S17 se sitúa en 1 y el resultado es igual a 0.

Conversión de una fecha en cadena de caracteres

Generalidades

Esta instrucción convierte una fecha en una cadena de caracteres (sin hora) con formato: YYYY-MM-DD (10 caracteres). Dicha cadena termina con el carácter terminador Ø. Cada uno de los caracteres Y,M,D simboliza una cifra.

Estructura

Lenguaje de contactos

Lenguaje lista de instrucciones

```
LD TRUE
[%MB2:11=DATE_TO_STRING(%MD40)]
```

Lenguaje literal estructurado

```
%MB2:11=DATE_TO_STRING(%MD40);
```

Ejemplos

```
%MB2:11=DATE_TO_STRING(%MD40)
```

%MD40:= Fecha (ej: 1998-12-27)

%MB	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*
	'1'	**'9'**	**'9'**	**'8'**	**'-'**	**'1'**	**'2'**	**'-'**	**'2'**	**'7'**	**Ø**

Sintaxis

Operador de conversión de una fecha en cadena

Sintaxis
Result:=**DATE_TO_STRING**(Fecha)

Operandos de conversión de una fecha en cadena

Tipo	**Result (resultado)**	**Fecha**
Tablas de 11 bytes	%MB:11	-
Palabras dobles indexables	-	%MD,%KD
Palabras dobles no indexables	-	%ID,%QD,Valor inmediato, Expr. numérica

Nota:

- Si el parámetro de entrada (fecha) no se puede interpretar y es incoherente con el formato DATE, el bit de sistema %S17 se sitúa en 1 y la función devuelve la cadena `**** - ** - **`.
- Si la cadena de salida es demasiado corta, se trunca y el bit de sistema %S15 se sitúa en 1.

```
%MB2:8 := DATE_TO_STRING(%MD40)
```

==> %MB

2	*3*	*4*	*5*	*6*	*7*	*8*	*9*
'1'	**'9'**	**'9'**	**'8'**	**'-'**	**'1'**	**'2'**	**'-'**

==> %S15 = 1

- Si la cadena de salida es demasiado larga, se completa con caracteres de tipo terminador Ø.

```
%MB2:12 := DATE_TO_STRING(%MD40)
```

==> %MB

2	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*
'1'	**'9'**	**'9'**	**'8'**	**'-'**	**'1'**	**'2'**	**'-'**	**'2'**	**'7'**	**Ø**	**Ø**

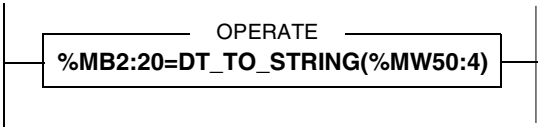
Conversión de una fecha completa en cadena de caracteres

Generalidades

Esta instrucción convierte una fecha completa (con hora) en una cadena de caracteres con formato: YYYY-MM-DD-HH:MM:SS (19 caracteres). Dicha cadena termina con el carácter terminador Ø. Cada uno de los caracteres Y,M,D,H,M,S simboliza una cifra.

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD TRUE
[%MB2:20=DT_TO_STRING(%MW50:4)]
```

Lenguaje literal estructurado

```
%MB2:20=DT_TO_STRING(%MW50:4);
```

Ejemplos

```
%MB2:20=DT_TO_STRING(%MW50:4)
```

%M50:4:= Fecha y hora (tipo DT) (ej: 1998-12-27-23:14:37)

==>

%MB	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22
	'1'	'9'	'9'	'8'	'-'	'1'	'2'	'-'	'2'	'7'	'-'	'2'	'3'	':'	'1'	'4'	':'	'3'	'7'	Ø	Ø

Sintaxis

Operador de conversión de una fecha completa en cadena

Sintaxis
Result:=DT_TO_STRING(Fecha)

Operandos de conversión de una fecha completa en cadena

Tipo	**Result (resultado)**	**Fecha**
Tablas de 20 bytes	%MB:20	-
Tabla de 4 palabras con formato DT	-	%MW:4,%KW:4

Nota:

- Si el parámetro de entrada (fecha) no se puede interpretar y es incoherente con el formato DT (DATE_AND_TIME), el bit de sistema %S17 se sitúa en 1 y la función devuelve la cadena `****-**-**:**:**`.
- Si la cadena de salida es demasiado corta, se trunca y el bit de sistema %S15 se sitúa en 1.

```
%MB2:8:=DT_TO_STRING(%MW50:4)
```

==> %MB

2	*3*	*4*	*5*	*6*	*7*	*8*	*9*
'1'	**'9'**	**'9'**	**'8'**	**'-'**	**'1'**	**'2'**	**'-'**

==> %S15 = 1

- Si la cadena de salida es demasiado larga, se completa con caracteres de tipo terminador Ø.

```
%MB2:21:=DT_TO_STRING(%MD50:4)
```

==>
%MB

2	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*	*18*	*19*	*20*	*21*	*22*
'1'	**'9'**	**'9'**	**'8'**	**'-'**	**'1'**	**'2'**	**'-'**	**'2'**	**'7'**	**'-'**	**'2'**	**'3'**	**':'**	**'1'**	**'4'**	**':'**	**'3'**	**'7'**	**Ø**	**Ø**

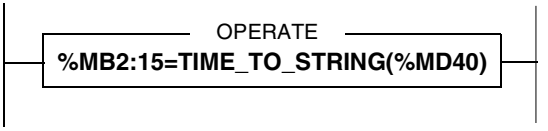
Conversión de una duración en cadena de caracteres

Generalidades

Esta instrucción convierte una duración (con formato TIME) en una cadena de caracteres. El formato del resultado se descompone en horas, minutos, segundos y décimas en 15 caracteres: HHHHHH:MM:SS.D. Dicha cadena termina con el carácter terminador Ø. Cada uno de los caracteres H,M,S,D simboliza una cifra. La duración máxima es de 119304 horas, 38 minutos, 49 segundos y 5 décimas.

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD TRUE
[%MB2:15=TIME_TO_STRING(%MD40)]
```

Lenguaje literal estructurado

```
%MB2:15=TIME_TO_STRING(%MD40);
```

Ejemplos

```
%MB2:15=TIME_TO_STRING(%MD40)
```

%MD40:= 27556330.3 (formato TIME)

%MB	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*
	'0'	**'0'**	**'7'**	**'6'**	**'5'**	**'4'**	**':'**	**'3'**	**'2'**	**':'**	**'1'**	**'0'**	**'.'**	**'3'**	**Ø**

Sintaxis

Operador de conversión de una duración en cadena

Sintaxis
Result:=**TIME_TO_STRING**(Duración)

Operandos de conversión de una duración en cadena

Tipo	**Result (resultado)**	**Duración**
Tablas de 15 bytes	%MB:15	-
Palabras dobles indexables	-	%MD,%KD
Palabras dobles no indexables	-	%ID,%QD,Valor inmediato, Expr. numérica

Duración está en formato TIME

Nota:

- Si la cadena de salida es demasiado corta, se trunca y el bit de sistema %S15 se sitúa en 1.

```
%MB2:8:=TIME_TO_STRING(%MD40)
```

==> %MB

2	*3*	*4*	*5*	*6*	*7*	*8*	*9*
'0'	**'0'**	**'7'**	**'6'**	**'5'**	**'4'**	**':'**	**'3'**

==> %S15 = 1

- Si la cadena de salida es demasiado larga, se completa con caracteres de tipo terminador Ø.

```
%MB2:16:=TIME_TO_STRING(%MD40)
```

==>

%MB

2	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*	*16*	*17*
'0'	**'0'**	**'7'**	**'6'**	**'5'**	**'4'**	**':'**	**'3'**	**'2'**	**':'**	**'1'**	**'0'**	**'.'**	**'3'**	**Ø**	**Ø**

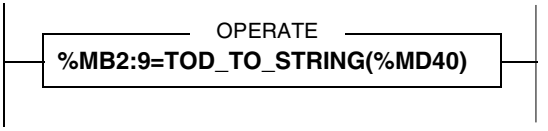
Conversión de una hora del día en cadena de caracteres

Generalidades

Esta instrucción convierte una hora del día (con formato TOD - TIME_OF_DAY) en una cadena de caracteres con formato HH:MM:SS en 8 caracteres más un carácter terminador Ø. Cada uno de los caracteres H,M,S simboliza una cifra.

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD TRUE
[%MB2:9=TOD_TO_STRING(%MD40)]
```

Lenguaje literal estructurado

```
%MB2:9=TOD_TO_STRING(%MD40);
```

Ejemplos

```
%MB2:9=TOD_TO_STRING(%MD40)
%MD40:= 23:12:27 (formato TOD)
```

%MB	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*
	'2'	**'3'**	**':'**	**'1'**	**'2'**	**':'**	**'2'**	**'7'**	**Ø**

Sintaxis

Operador de conversión de una hora del día en cadena

Sintaxis
Result:=TOD_TO_STRING(Duración)

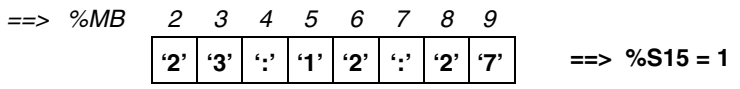
Operandos de conversión de una hora del día en cadena

Tipo	**Result (resultado)**	**Hora**
Tablas de 9 bytes	%MB:9	-
Palabras dobles indexables	-	%MD,%KD
Palabras dobles no indexables	-	%ID,%QD,Valor inmediato, Expr. numérica

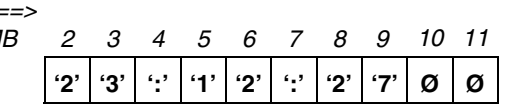
Hora está en formato TOD

Nota:

- Si la cadena de salida es demasiado corta, se trunca y el bit de sistema %S15 se sitúa en 1.
  %MB2:8 := TOD_TO_STRING (%MD40) (con %MD40 := 23:12:27)



- Si la cadena de salida es demasiado larga, se completa con caracteres de tipo terminador Ø.
  %MB2:10 := TOD_TO_STRING (%MD40) (con %MD40 := 23:12:27)



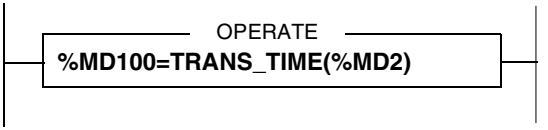
Conversión de una duración en HHHH:MM:SS

Generalidades

Esta instrucción convierte una duración (con formato TIME) en número de horas-minutos-segundos, HHHH:MM:SS. Valores límite [0000:00:00 , 9999:59:59].

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD TRUE
[%MD100=TRANS_TIME(%MD2)]
```

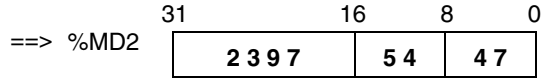
Lenguaje literal estructurado

```
%MD100=TRANS_TIME(%MD2);
```

Ejemplos

```
%MD100=TRANS_TIME(%MD2)
```

con %MD2:= 36324873 décimas de segundo



valores expresados en hexadecimal

Sintaxis

Operador de conversión de una duración en HHHH:MM:SS

Sintaxis
Result:=**TRANS_TIME**(Duración)

Operandos de conversión de una duración en HHHH:MM:SS

Tipo	**Result (resultado)**	**Duración**
Palabras dobles indexables	%MD	%MD,%KD
Palabras dobles no indexables	%QD	%ID,%QD,Valor inmediato, Expr. numérica

Result está en formato HMS
Duración está en formato TIME

Nota:
- El principio del redondeo se aplica al parámetro "duración" (expresado en décimas de segundo) para poder realizar la conversión (precisión al segundo).
  - ssssssss.0 a ssssssss.4 redondeado a ssssssss.0
  - ssssssss.5 a ssssssss.9 redondeado a ssssssss.0 + 1.0
- La duración máxima convertida puede alcanzar 10000 horas. Esto significa que si el valor de la duración (TIME) pasada en parámetro es superior o igual a 360000000, no se puede convertir. El bit de sistema %S15 se sitúa en 1 y el resultado es igual a 0000:00:00.

2.9 Instrucciones sobre tabla de bits

Presentación

Objeto de este apartado

En este apartado se describen las instrucciones de tablas de bits del lenguaje PL7

Contenido

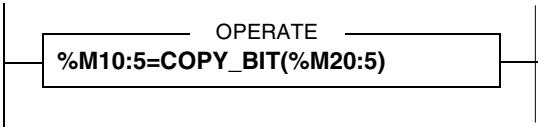
Esta sección contiene los siguientes apartados:

Copia de una tabla de bits a una tabla de bits

Generalidades Esta función efectúa la copia bit a bit de una tabla de bits en otra tabla de bits.

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD TRUE
[%M10:5=COPY_BIT(%M20:5)]
```

Lenguaje literal estructurado

```
%M10:5=COPY_BIT(%M20:5);
```

Sintaxis Operador de copia de tabla de bits

Sintaxis
Result:=**COPY_BIT**(Tab)

Operandos de copia de tabla de bits

Tipo	Result (resultado)	Tab (tabla)
Tabla de bits	%M:L,%Q:L,%I:L	%M:L,%Q:L,%I:L,%Xi:L

Nota:
- Las tablas pueden ser de distinto tamaño. En tal caso, la tabla del resultado contiene el resultado de la función ejecutada en una longitud equivalente al tamaño más pequeño de las tablas, y el resto de la tabla del resultado no se modifica.
- Cuidado con los solapamientos entre la tabla de entrada y la tabla del resultado.

Instrucciones lógicas en tablas de bits

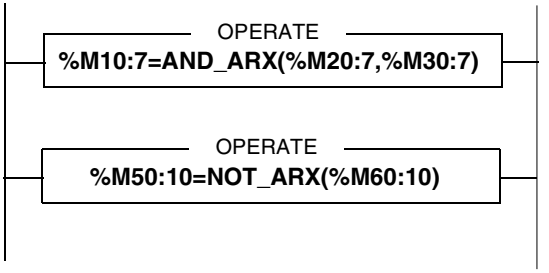
Generalidades

Las funciones asociadas permiten llevar a cabo una operación lógica bit a bit entre dos tablas de bits y guarda el resultado en otra tabla de bits.

- **AND_ARX**: Y lógico (bit a bit).
- **OR_ARX**: O lógico (bit a bit)
- **XOR_ARX**: O exclusivo (bit a bit)
- **NOT_ARX**: complemento lógico (bit a bit) de una tabla

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD TRUE
[%M10:7=AND_ARX(%M20:7,%M30:7)]

LD TRUE
[%M50:10=NOT_ARX(%M60:10)]
```

Lenguaje literal estructurado

```
%M10:7=AND_ARX(%M20:7,%M30:7);
%M50:10=NOT_ARX(%M60:10);
```

Sintaxis

Operadores de instrucciones lógicas en tablas de bits

Sintaxis
Result:=**AND_ARX**(Tab 1, Tab 2)
Result:=**OR_ARX**(Tab 1, Tab 2)
Result:=**XOR_ARX**(Tab 1, Tab 2)
Result:=**NOT_ARX**(Tab 1)

Operandos de instrucciones lógicas en tablas de bits

Tipo	**Result (resultado)**	**Tab 1 y Tab 2 (tabla)**
Tabla de bits	%M:L,%Q:L,%I:L	%M:L,%Q:L,%I:L,%Xi:L

Nota:
- Las tablas pueden ser de distinto tamaño. En tal caso, la tabla del resultado contiene el resultado de la función ejecutada en una longitud equivalente al tamaño más pequeño de las tablas, y el resto de la tabla del resultado no se modifica.
- Posibilidad de solapamiento entre la tabla de entrada y la tabla del resultado.

Copia de una tabla de bits a una tabla de palabras

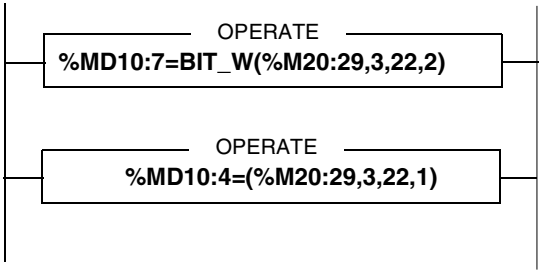
Generalidades

La función copia los bits de una tabla o parte de una tabla de bits en una tabla de palabras (o palabras dobles).
En la tabla de bits, la extracción se efectúa a partir de un determinado rango (brow) para un número de bits (nbit).
En la tabla de palabras (o palabras dobles), la copia se realiza a partir del rango (wrow o drow) comenzando por la palabra menos significativa.

- **BIT_W**: Copia de una tabla de bits a una tabla de palabras.
- **BIT_D**: Copia de una tabla de bits a una tabla de palabras dobles.

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

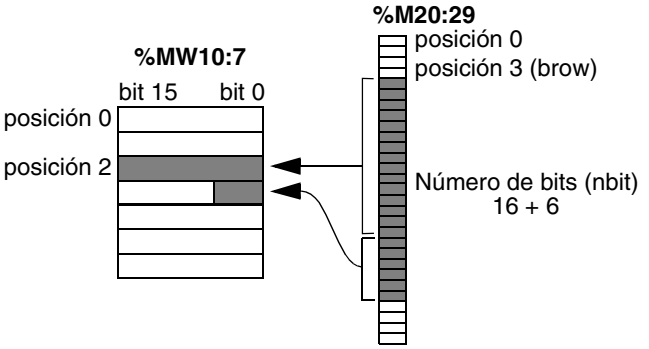
```
LD TRUE
[%MD10:7=BIT_W(%M20:29,3,22,2)]

LD TRUE
[%MD10:4=(%M20:29,3,22,1)]
```

Lenguaje literal estructurado

```
%MD10:7=BIT_W(%M20:29,3,22,2);
%MD10:4=(%M20:29,3,22,1);
```

Ejemplo

```
%MD10:7=BIT_W(%M20:29,3,22,2);
```



Sintaxis

Operadores de copia de una tabla de bits a una tabla de palabras

Sintaxis
Result:=**BIT_W**(Tab, brow, nbit, wrow)
Result:=**BIT_D**(Tab, brow, nbit, drow)

Operandos de copia de una tabla de bits a una tabla de palabras

Tipo	**Result (resultado)**	**Tab (tabla)**	**brow - nbit wrow o drow**
Tablas de palabras	%MW:L	-	-
Tablas de palabras dobles	%MD:L	-	-
Tabla de bits	-	%M:L,%Q:L,%I:L,%Xi:L	-
Palabras indexables	-	-	%MW,%KW,%Xi.T
Palabras no indexables	-	-	%IW,%QW,%SW, %NW, Valor inm., Expr. num.

Nota:
- Si el número de bits que se va a tratar es superior al número de bits restantes en la tabla a partir del rango (brow), la función ejecuta la copia hasta el último elemento de la tabla.
- Si el número de bits que se va a copiar es superior al número de bits que componen las palabras restantes de la tabla del resultado, la función detiene la copia en el último elemento de la tabla de palabras (o palabras dobles).
- Un valor negativo en los parámetros brow, nbit, wrow o drow se interpretará como nulo.

Copia de una tabla de palabras en una tabla de bits

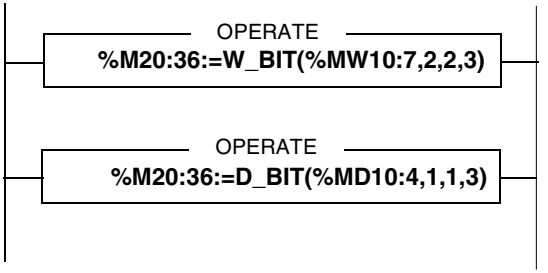
Generalidades

La función copia los bits de una tabla o parte de una tabla de palabras (o palabras dobles) en una tabla de bits.
En la tabla de palabras (o palabras dobles), la extracción se efectúa a partir de la palabra de rango (wrow o drow) para un número de palabras (nwd).
En la tabla de bits, la copia se realiza a partir del rango (brow) comenzando por el bit de peso menos significativo de cada palabra.

- **W_BIT** : Copia de una tabla de palabras en una tabla de bits.
- **D_BIT** : Copia de una tabla de palabras dobles en una tabla de bits.

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

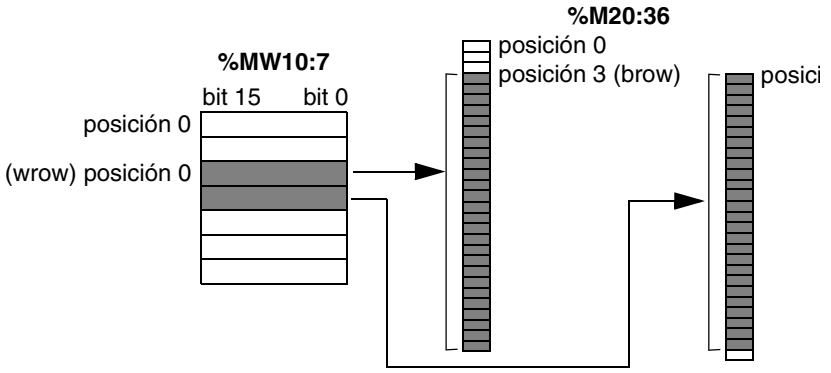
```
LD TRUE
[%M20:36:=W_BIT(%MW10:7,2,2,3)]

LD TRUE
[%M20:36:=D_BIT(%MD10:4,1,1,3)]
```

Lenguaje literal estructurado

```
%M20:36:=W_BIT(%MW10:7,2,2,3);
%M20:36:=D_BIT(%MD10:4,1,1,3);
```

Ejemplo

```
%M20:36:=W_BIT(%MW10:7,2,2,3);
```



Sintaxis

Operadores de copia de una tabla de palabras en una tabla de bits

Sintaxis
Result:=**W_BIT**(Tab, wrow, nwd, brow)
Result:=**D_BIT**(Tab, drow, nwd, brow)

Operandos de copia de una tabla de palabras en una tabla de bits

Tipo	**Result (resultado)**	**Tab (tabla)**	**wrow o drow nwd -brow**
Tablas de bits	%M:L,%Q:L,%I:L	-	-
Tablas de palabras	-	%MW:L,%KW:L	-
Tabla de palabras dobles	-	%MD:L,%KD:L	-
Palabras indexables	-	-	%MW,%KW,%Xi.T
Palabras no indexables	-	-	%IW,%QW,%SW,%NW, Valor inm.,Expr. num.

Nota:

- Si el número de bits que se va a tratar es superior al número de bits restantes en la tabla a partir del rango (wrow), la función ejecuta la copia hasta el último elemento de la tabla.
- Si el número de bits que se va a copiar es superior al número de bits que componen las palabras restantes de la tabla del resultado, la función detiene la copia en el último elemento de la tabla de palabras (o palabras dobles).
- Si el número de bits que se va a copiar es superior al número de bits restantes en la tabla del resultado, la función detiene la copia en el último elemento de la tabla.
- Un valor negativo en los parámetros brow, nbit, wrow o drow se interpretará como nulo.

2.10 Funciones "Orphée": desplazamientos, contador

Presentación

Objeto de este apartado

Este apartado describe las funciones "Orphée": desplazamientos, contador del lenguaje PL7

Contenido

Esta sección contiene los siguientes apartados:

Desplazamientos de palabras con recuperación de los bits desplazados

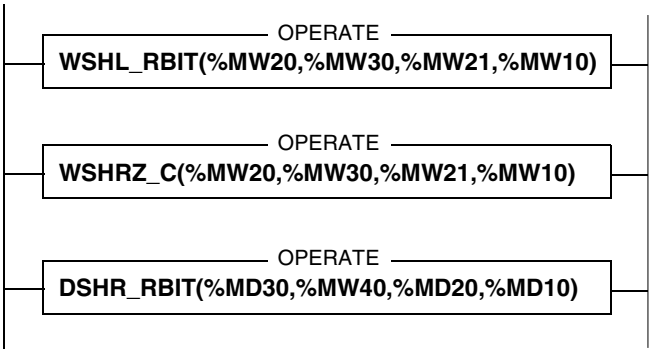
Generalidades

Las funciones realizan un número (nbit) de desplazamientos aritméticos a la izquierda o la derecha en una palabra o palabra doble (a).
Después del desplazamiento, el valor se guarda en (resu) y los bits desplazados en (rest).

- **WSHL_RBIT**: Desplazamiento a la izquierda en una palabra con recuperación de los bits desplazados.
- **DSHL_RBIT**: Desplazamiento a la izquierda en una palabra doble con recuperación de los bits desplazados.
- **WSHRZ_C**: Desplazamiento hacia la derecha en una palabra con relleno mediante 0 y recuperación de los bits desplazados.
- **DSHRZ_C**: Desplazamiento hacia la derecha en una palabra doble con relleno mediante 0 y recuperación de los bits desplazados.
- **WSHR_RBIT**: Desplazamiento hacia la derecha en una palabra con extensión de signo y recuperación de los bits desplazados.
- **DSHR_RBIT**: Desplazamiento hacia la derecha en una palabra doble con extensión de signo y recuperación de los bits desplazados.

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

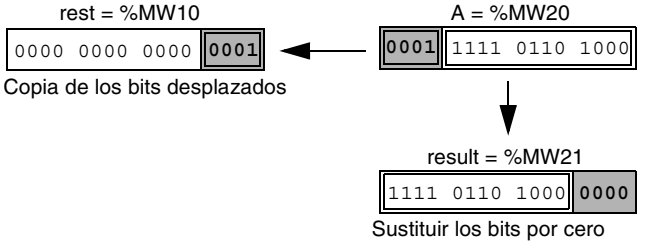
```
LD TRUE
[WSHL_RBIT(%MW20,%MW30,%MW21,%MW10)]

LD TRUE
[WSHRZ_C(%MW20,%MW30,%MW21,%MW10)]

LD TRUE
[DSHR_RBIT(%MD30,%MW40,%MD20,%MD10)]
```

Lenguaje literal estructurado

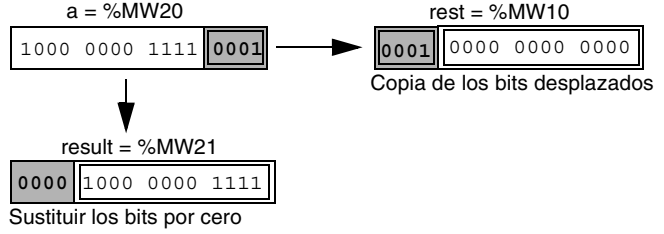
```
WSHL_RBIT(%MW20,%MW30,%MW21,%MW10);

WSHRZ_C(%MW20,%MW30,%MW21,%MW10);

DSHR_RBIT(%MD30,%MW40,%MD20,%MD10);
```

Ejemplos

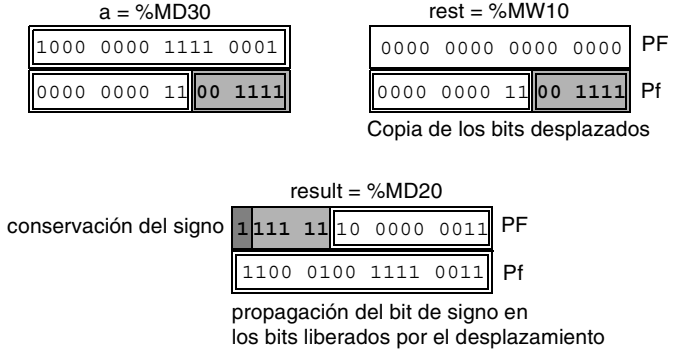
`WSHL_RBIT(%MW20,%MW30,%MW21,%MW10)` con %MW30 = 4



`WSHRZ_C(%MW20,%MW30,%MW21,%MW10)` con %MW30 = 4



`DSHR_RBIT(%MD30,%MW40,%MD20,%MD10)` con %MW40 = 6



Sintaxis

Operadores de desplazamiento en palabras con recuperación de bits desplazados

Sintaxis
WSHL_BIT(a, nbit, resu, rest)
WSHRZ_C(a, nbit, resu, rest)
WSHR_RBIT(a, nbit, resu, rest)

Operandos de desplazamiento en palabras con recuperación de bits desplazados

Tipo	**a**	**nbit**	**resu, rest**
Palabras indexables	%MW,%KW	%MW,%KW,%Xi.T	%MW
Palabras no indexables	%IW,%QW,%SW, %NW,Valor inm., Expresión num.	%IW,%QW,%SW, %NW,Valor inm., Expresión num.	%QW,%SW,%NW

Operadores de desplazamiento en palabras dobles con recuperación de bits desplazados

Sintaxis
DSHL_BIT(a, nbit, resu, rest)
DSHRZ_C(a, nbit, resu, rest)
DSHR_RBIT(a, nbit, resu, rest)

Operandos de desplazamiento en palabras dobles con recuperación de bits desplazados

Tipo	**a**	**nbit**	**resu, rest**
Palabras dobles indexables	%MD,%KD	-	%MD
Palabras dobles no indexables	%ID,%QD,%SD, Valor inmediato, Expresión num.	-	%QD,%SD
Palabras indexables	-	%MW,%KW,%Xi.T	-
Palabras no indexables	-	%IW,%QW,%SW, %NW,Valor inm., Expresión num.	-

Nota: Si el parámetro (nbit) no está comprendido entre 1 y 16 para los desplazamientos en palabras, o entre 1 y 32 para los desplazamientos en palabras dobles, las salidas (resu) y (rest) no son significativas y el bit de sistema %S18 se sitúa en 1.

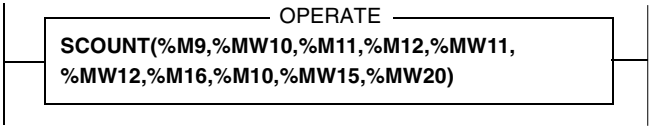
Contaje/descontaje con señalización de rebasamiento

Generalidades

La función efectúa un contaje/descontaje con señalización de rebasamiento. Esta función sólo se ejecuta si la entrada de validación (en) se encuentra en estado 1. Dos entradas separadas (cu y cd) permiten contar y descontar sucesos. La salida (Qmín) se sitúa en 1 desde el momento en el que se alcanza el umbral mínimo (mín); la salida (Qmáx) se sitúa en 1 desde el momento en el que se alcanza el umbral máximo (máx).
El valor inicial del contaje viene fijado por el parámetro (pv) y el valor actual del contaje viene determinado por el parámetro (cv).
Una palabra de 16 bits (mwd) permite almacenar el estado de las entradas cu y cd (bit 0 para el almacenamiento de cu y bit 1 para el almacenamiento de cd).

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD TRUE
[SCOUNT(%M9,%MW10,%M11,%M12,%MW11,%MW12,%M16,%M10,%MW15,%MW2
0)]
```

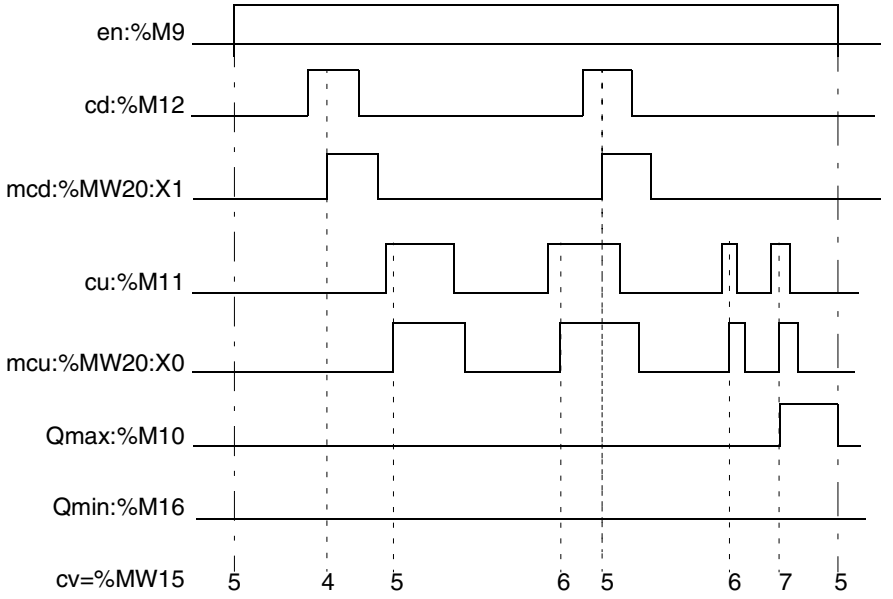
Lenguaje literal estructurado

```
SCOUNT(%M9,%MW10,%M11,%M12,%MW11,%MW12,%M16,%M10,%MW15,%MW20
);
```

Ejemplos

```
SCOUNT(%M9,%MW10,%M11,%M12,%MW11,%MW12,%M16,%M10,%MW15,%MW20
)
```

con %MW10 (pv) = 5, %MW11 (mín) = 0, %MW12 (máx) = 7



Sintaxis

Operadores de contaje/descontaje con señalización de rebasamiento

Sintaxis
SCOUNT(en, pv, cu, cd, mín, máx, Qmín, Qmáx, cv, mwd)

Operandos de contaje/descontaje con señalización de rebasamiento

Tipo	**en, cu, cd**	**Qmín, Qmáx**	**pv, mín, máx**	**cv,mwd**
Bits	%I,%Q,%M,%S, %BLK,%.:Xk	%I,%Q,%M	-	-
Palabras indexables	-	-	%MW,%KW,%Xi.T	%MW
Palabras no indexables	-	-	%IW,%QW,%SW, %NW,Valor inm., Expresión num.	%QW,%SW, %NW

Nota:

- Si (en) = 0, la función ya no se valida y en cada acceso se obtiene:
  Qmín = Qmáx = 0
  mcu = mcd = 0 cv = pv
- Si máx > mín, entonces:
  cv >= máx ---> Qmáx = 1 y Qmín = 0
  mín < cv < máx ---> Qmáx = Qmín = 0
  cv <= mín ---> Qmáx = 0 y Qmín = 1
- Si máx < mín, entonces:
  máx <= cv <= mín ---> Qmáx = 1 y Qmín = 0
  cv < máx ---> Qmáx = 0 y Qmín = 1
  cv > mín ---> Qmáx = 1 y Qmín = 0
- Si máx = mín, entonces:
  cv < mín y máx ---> Qmáx = 0 y Qmín = 1
  cv >= mín y máx ---> Qmáx = 1 y Qmín = 0
- La modificación del parámetro (pv) con (en) en estado 1 no tiene ninguna incidencia sobre el funcionamiento.
- Un valor negativo para los parámetros (pv) y (mín) se interpreta como un valor nulo.
- Un valor inferior a 1 para el parámetro (máx) se interpreta como igual a 1.

Desplazamientos circulares

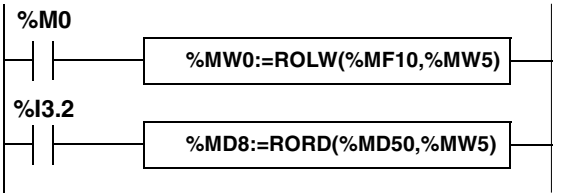
Generalidades

Las funciones realizan desplazamientos circulares a la derecha o a la izquierda en una palabra o palabra doble.

- **ROLW**: desplazamiento circular hacia la izquierda en una palabra con un número de desplazamientos calculado
- **RORW**: desplazamiento circular hacia la derecha en una palabra con un número de desplazamientos calculado
- **ROLD**: desplazamiento circular hacia la izquierda en una palabra doble con un número de desplazamientos calculado
- **RORD**: desplazamiento circular hacia la derecha en una palabra doble con un número de desplazamientos calculado

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD %M0
[%MW0:=ROLW(%MW10,%MW5)]

LD %I3.2
[%MD10:=RORD(%MD100,%MW5)]
```

Lenguaje literal estructurado

```
IF %M0 THEN
 %MW0:=ROLW(%MW10,%MW5);
END_IF;
IF %I3.2 THEN
 %MD8:=RORD(%MD100,%MW5);
END_IF
```

Sintaxis

Operadores de desplazamientos circulares

Operadores	**Sintaxis**
ROLW, RORW, ROLD, RORD	Op1:=Operador(Op2,n)

Operandos de desplazamientos circulares en palabra **ROLW**, **RORW**

Tipo	**Operando 1 (Op1)**	**Operando 2 (Op2)**	**Número de posición (n)**
Palabras indexables	%MW	%MW,%KW,%Xi.T	%MW,%KW,%Xi.T
Palabras no indexables	-	Val.inm.,%IW,%QW,%SW,%NW,%BLK, Expr.num.	Val.inm.,%IW,%QW, %SW,%NW,%BLK, Expr.num.

Operandos de desplazamientos circulares en palabra doble **ROLD**, **RORD**

Tipo	**Operando 1 (Op1)**	**Operando 2 (Op2)**	**Número de posición (n)**
Palabras indexables	%MD	%MD,%KD	%MW,%KW,%Xi.T
Palabras no indexables	%QD,%SD	Val.inm.,%ID,%QD,%SD, Expr.num.	Val.inm.,%IW,%QW, %SW,%NW,%BLK, Expr.num.

Nota: Se utilizan preferentemente las instrucciones de base ROL y ROR (cuando el número de desplazamientos es estático), ya que estas instrucciones tienen mejores resultados.

2.11 Funciones de temporización

Presentación

Objeto de este apartado

En este apartado se describen las funciones de temporización del lenguaje PL7

Contenido

Esta sección contiene los siguientes apartados:

Funciones de temporización

Generalidades

Las funciones de temporización, a diferencia de los bloques de función predefinidos, no tienen límite en cuanto al número y se pueden utilizar en el código de los bloques de función DFB.
Se proponen 4 funciones de temporización:

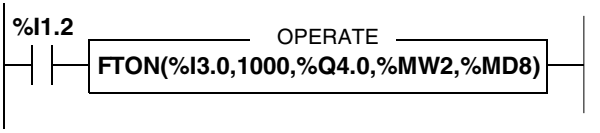
- **FTON**: Temporización de conexión.
- **FTOF**: Temporización de desconexión.
- **FTP**: Temporización de impulso: Temporización
- **FPULSOR**: señal rectangular: Temporización

Función temporización de conexión

Generalidades Esta función permite gestionar retardos en la conexión, que se pueden programar.

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD I1.2
[FTON(%I3.0,1000,%Q4.0,%MW2,%MD8)]
```

Lenguaje literal estructurado

```
IF %I1.2 THEN
 FTON(%I3.0,1000,%Q4.0,%MW2,%MD8);
END_IF;
```

Sintaxis

Operadores de la función de temporización de conexión FTON

Sintaxis
FTON(EN,PT,Q,ET,PRIV)

Operandos de la función de temporización de conexión FTON

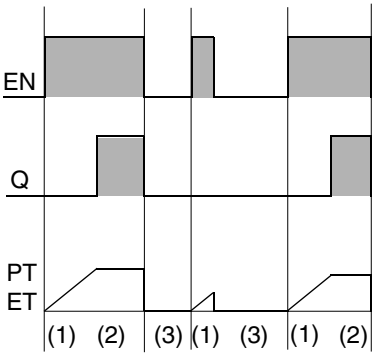
Tipo	**EN**	**PT**	**Q**	**ET**	**PRIV**
Palabras indexables	-	%MW,%KW, %Xi.T	-	%MW	-
Palabras no indexables	-	%IW,%QW, %SW,%NW, Valor inmediato, Expresión numérica	-	%IW,%QW	-
Palabras dobles indexables	-	-	-	-	%MD
Bits	%I,%Q,%M, %S,%BLK, %*:Xk,%X		%I,%Q,%M %S,%*:Xk, %X	-	-

Características Características de la función de temporización de conexión FTON

Característica	Variable	Valor
Entrada "Activación"	EN	En el flanco ascendente inicia la temporización
Valor de preselección	PT	Palabra de entrada que determina la duración de la temporización (en centésimas de segundo). Permite definir una duración máxima de 5 min y 27 s con una precisión de 10 ms. (1)
Salida "Temporizador"	Q	Salida en 1 al final de la temporización.
Valor actual	ET	Palabra de salida que aumenta de 0 a PT al terminar el temporizador.
Variable de cálculo	PRIV	Palabra doble para el almacenamiento de los estados internos. Debe asociarse a esta palabra doble una variable de la aplicación exclusivamente reservada a tal efecto.

Nota: (1) se tiene en cuenta una modificación de esta palabra durante la temporización.

Funcionamiento Descripción del funcionamiento de la función de temporización de conexión FTON

Etapa	Acción	Descripción	Figura
1	Flanco ascendente en la entrada EN	El temporizador se inicia: su valor actual ET aumenta de 0 hacia PT (centésimas de segundo).	
2	El valor actual ha alcanzado PT	El bit de salida Q pasa a 1 y a continuación permanece en 1 mientras la entrada EN esté en 1.	
3	la entrada EN está en 0	El temporizador se detiene aunque estuviera en curso de evolución: ET toma el valor 0.	

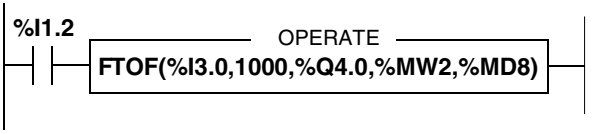
Función temporización de desconexión

Generalidades

Esta función permite gestionar retardos en la desconexión, que se pueden programar.

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD I1.2
[FTOF(%I3.0,1000,%Q4.0,%MW2,%MD8)]
```

Lenguaje literal estructurado

```
IF %I1.2 THEN
 FTOF(%I3.0,1000,%Q4.0,%MW2,%MD8);
END_IF;
```

Sintaxis

Operadores de la función de temporización de desconexión FTOF

Sintaxis
FTOF(EN,PT,Q,ET,PRIV)

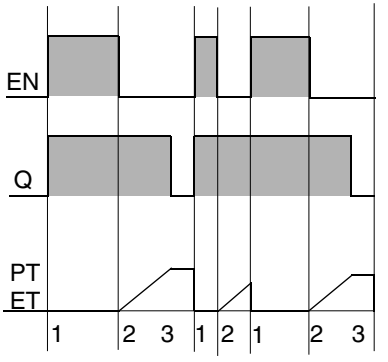
Operandos de la función de temporización de desconexión FTOF: idénticas a FTON (Véase *Función temporización de conexión, p. 277*)

Características Características de la función de temporización de desconexión FTOF

Característica	Variable	Valor
Entrada "Activación"	EN	En el flanco descendente inicia la temporización
Valor de preselección	PT	Palabra de entrada que determina la duración de la temporización (en centésimas de segundo). Permite definir una duración máxima de 5 min y 27 s con una precisión de 10 ms. (1)
Salida "Temporizador"	Q	Salida puesta en 1 en el flanco ascendente de EN y en 0 al final de la temporización.
Valor actual	ET	Palabra de salida que aumenta de 0 a PT al terminar el temporizador.
Variable de cálculo	PRIV	Palabra doble para el almacenamiento de los estados internos. Debe asociarse a esta palabra doble una variable de la aplicación exclusivamente reservada a tal efecto.

Nota: (1) se tiene en cuenta una modificación de esta palabra durante la temporización.

Funcionamiento Descripción del funcionamiento de la función de temporización de desconexión FTOF

Etapa	Acción	Descripción	Figura
1	Flanco ascendente en la entrada EN	El valor actual ET toma el valor 0 (aunque el temporizador esté en curso de evolución) y el bit de salida Q pasa a 1 (o permanece en 1).	
2	En el flanco descendente de la entrada EN	el temporizador se inicia y a continuación el valor actual aumenta desde 0 hacia PT (centésimas de segundo).	
3	Cuando el valor actual ha alcanzado PT.	El bit de salida Q vuelve a 0.	

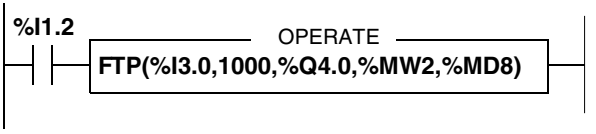
Función temporización de impulso

Generalidades

Esta función permite elaborar un impulso de duración precisa, que se puede programar.

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD I1.2
[FTP(%I3.0,1000,%Q4.0,%MW2,%MD8)]
```

Lenguaje literal estructurado

```
IF %I1.2 THEN
 FTP(%I3.0,1000,%Q4.0,%MW2,%MD8);
END_IF;
```

Sintaxis

Operadores de la función de temporización de impulso FTP

Sintaxis
FTP(EN,PT,Q,ET,PRIV)

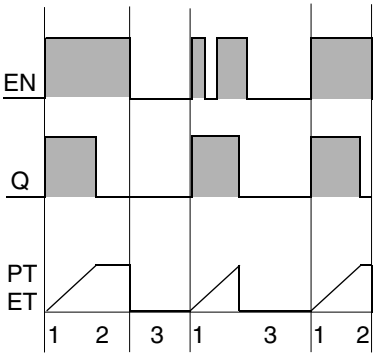
Operandos de la función de temporización de impulso FTP: idénticos a FTON (Véase *Función temporización de conexión, p. 277*)

Características Características de la función de temporización de impulso FTP

Característica	Variable	Valor
Entrada "Activación"	EN	En el flanco descendente inicia la temporización
Valor de preselección	PT	Palabra de entrada que determina la duración de la temporización (en centésimas de segundo). Permite definir una duración máxima de 5 min y 27 s con una precisión de 10 ms. (1)
Salida "Temporizador"	Q	Salida en 1 al final de la temporización.
Valor actual	ET	Palabra de salida que aumenta de 0 a PT al terminar el temporizador.
Variable de cálculo	PRIV	Palabra doble para el almacenamiento de los estados internos. Debe asociarse a esta palabra doble una variable de la aplicación exclusivamente reservada a tal efecto.

Nota: (1) se tiene en cuenta una modificación de esta palabra durante la temporización.

Funcionamiento Descripción del funcionamiento de la función de temporización de impulso FTP

Etapa	Acción	Descripción	Figura
1	Flanco ascendente en la entrada EN	El temporizador se inicia (si no está ya en curso de evolución) y el valor actual ET aumenta desde 0 hacia PT (centésimas de segundo). El bit de salida Q pasa a 1.	 Este monoestable no se puede reactivar.
2	Cuando el valor actual ha alcanzado PT.	El bit de salida Q vuelve a 0.	
3	La entrada EN y la salida Q están en 0	PT toma el valor 0.	

Función generador de señal rectangular

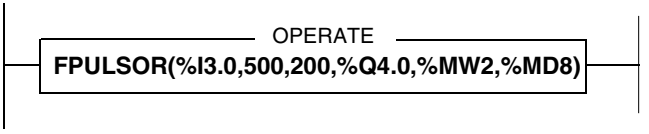
Generalidades

Esta función permite generar una señal rectangular periódica de la que se puede hacer variar el ancho de impulso a 1 y el impulso a 0 mediante programa a través de 2 temporizadores:

- **TON**: temporización en la subida (para el impulso en 1).
- **TOFF**: temporización en la caída (para el impulso en 0).

Estructura

Lenguaje de contactos



Lenguaje lista de instrucciones

```
LD TRUE
[FPULSOR(%I3.0,500,200,%Q4.0,%MW2,%MD8)]
```

Lenguaje literal estructurado

```
IF %I1.2 THEN
 FPULSOR(%I3.0,500,200,%Q4.0,%MW2,%MD8);
END_IF;
```

Sintaxis

Operadores de la función generador de señal rectangular FPULSOR

Sintaxis
FPULSOR(EN,TON,TOFF,Q,ET,PRIV)

Operandos de la función generador de señal rectangular FPULSOR

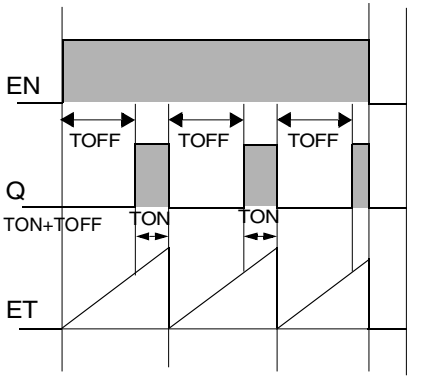
Tipo	**EN**	**TON,TOFF**	**Q**	**ET**	**PRIV**
Palabras indexables	-	%MW,%KW, %Xi.T	-	%MW	-
Palabras no indexables	-	-	-	%IW,%QW	-
Palabras dobles indexables	-	-	-	-	%MD
Bits	%BLK,%*:Xk, %X	%I,%Q,%M, %S	%S,%*:Xk,%X	%I,%Q,%M	-

Características Características de la función generador de señal rectangular FPULSOR:

Característica	Variable	Valor
Entrada "Activación"	EN	En el flanco ascendente se inicia la generación de la señal rectangular
Valor de preselección (impulso en 1)	TON	Palabra de entrada que determina la duración (en centésimas de segundo) del impulso en 1. Permite definir una duración máxima de 5 min y 27 s con una precisión de 10 ms. (1)
Valor de preselección (impulso en 0)	TOFF	Palabra de entrada que determina la duración (en centésimas de segundo) del impulso en 0. Permite definir una duración máxima de 5 min y 27 s con una precisión de 10 ms. (1)
Salida de señal rectangular	Q	Salida de impulso a 0 en la duración TOFF, a 1 en la duración TON.
Valor actual	ET	Palabra de salida que aumenta de 0 a TON+TOFF al terminar el temporizador.
Variable de cálculo	PRIV	Palabra doble para el almacenamiento de los estados internos. Debe asociarse a esta palabra doble una variable de la aplicación exclusivamente reservada a tal efecto.

Nota: (1) se tiene en cuenta una modificación de estas palabras durante la temporización. La suma TOFF+TON tiene una duración máxima de 5 min y 27 s.

Funcionamiento Descripción del funcionamiento de la función generador de señal rectangular FPULSOR:

Etapa	Acción	Descripción	Figura
1	Flanco ascendente en la entrada EN	la generación de la señal rectangular se inicia: (si la señal no está ya en curso de evolución) su valor actual ET aumenta desde 0 hacia TON+TOFF (centésimas de segundo).	
2	Mientras no termine la temporización TOFF	El bit de salida Q permanece en 0.	
3	TOFF se ha terminado, TON se inicia	El bit de salida Q pasa a 1 hasta el final de TON y el generador realiza un bucle en (2) y (3)	
4	EN pasa a 0	TON y TOFF vuelven a 0, el bit de salida Q pasa a 0	

2.12 Funciones de archivado de datos

Presentación

Objeto de este apartado

En este apartado se describen las funciones de archivado de datos del lenguaje PL7

Contenido

Esta sección contiene los siguientes apartados:

Funciones de archivado de datos

Presentación

Estas funciones permiten archivar los datos por programa en una zona dedicada de las tarjetas de memoria de la aplicación.

Ejemplo de aplicación

- almacenamiento automático de datos (consignas de estado, historiales, ...) de la aplicación en la tarjeta de memoria de la aplicación situada en el emplazamiento de memoria del procesador del autómata.
- guardado de las validaciones de producción en esta tarjeta de memoria.

Diferentes funciones

6 funciones permiten el archivado y la restitución de los datos.

Las funciones siguientes se aplican indistintamente a las tarjetas de memoria PCMCIA Tipo I (tarjetas de memoria situadas en el emplazamiento 0 del procesador) y Tipo III (tarjetas de memoria situadas en el emplazamiento 1 del procesador).

- **SET_PCM_EXT**: para inicializar en un valor todo o parte de la zona de archivado de la tarjeta de memoria,
- **WRITE_PCM_EXT**: para escribir los datos en la zona de archivado de la tarjeta de memoria,
- **READ_PCM_EXT**: para leer los datos en la zona de archivado de la tarjeta de memoria.

Nota: Estas funciones requieren:
- PL7 V4.2 o superior,
- una versión de OS del autómata (SV) igual o superior a 5.2.

Las funciones siguientes se aplican únicamente para las tarjetas de memoria PCMCIA Tipo I (tarjetas de memoria situadas en el emplazamiento 0 del procesador).

- **SET_PCMCIA**: para inicializar en un valor todo o parte de la zona de archivado de la tarjeta de memoria,
- **WRITE_PCMCIA**: para escribir los datos en la zona de archivado de la tarjeta de memoria,
- **READ_PCMCIA**: para leer los datos almacenados en la zona de archivado de la tarjeta de memoria.

Nota: el acceso a los datos alamacenados en la zona de archivado de una tarjeta de memoria sólo es posible desde la aplicación residente en el autómata mediante estas 6 funciones de base. En ningún caso, una estación remota puede acceder a esta zona directamente a través de una red o bus de comunicación.

Inicialización de la zona de archivado

Presentación

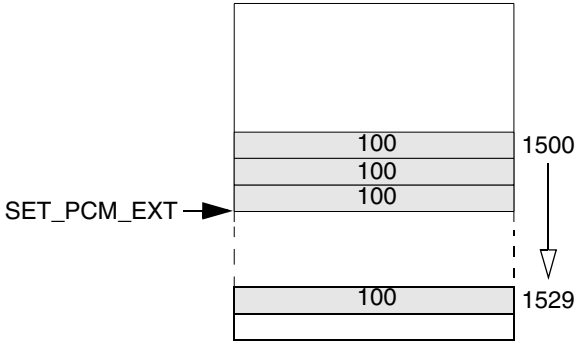
La función **SET_PCM_EXT** permite inicializar en el valor deseado todo o parte de la zona de archivado de una tarjeta de memoria.

Esta función utiliza 5 parámetros:

- **SLOT**: número de la vía donde está insertada la tarjeta de memoria PCMCIA:
  - 0 para una tarjeta situada en el emplazamiento 0 del procesador (tarjeta PCMCIA Tipo I),
  - 1 para una tarjeta situada en el emplazamiento 1 del procesador (tarjeta PCMCIA Tipo III).
- **DEST**: dirección de la zona de archivado a partir de la cual se efectúa la Inicialización
- **NUM**: número de palabras a inicializar
- **VAL**: valor de la Inicialización
- **CR**: código que devuelve el resultado de la ejecución del comando de Inicialización

Ejemplo

Representación de la tarjeta de memoria de la aplicación:

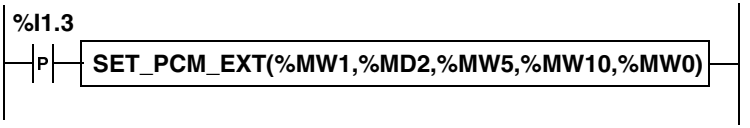


En este ejemplo:

- **SLOT** = %MW1 , %MW1 que contiene el valor 1
- **DEST** = %MD2 , %MD2 que contiene el valor 1500
- **NUM** = %MW5, %MW5 que contiene el valor 30
- **VAL** = %MW10, %MW10 que contiene el valor 100

Estructura

Lenguaje de contactos:



Lenguaje lista de instrucciones

```
LDR   %I1.3
[SET_PCM_EXT(%MW1,%MD2,%MW5,%MW10,%MW0)]
```

Lenguaje literal estructurado:

```
IF RE %I1.3 THEN
  SET_PCM_EXT(%MW1,%MD2,%MW5,%MW10,%MW0);
END_IF;
```

Sintaxis

Sintaxis de función:

SET_PCM_EXT (SLOT,DEST,NUM,VAL,CR)

Parámetros:

Tipo	SLOT	DEST	NUM	VAL	CR
Palabras indexables	%MW, Val inm.	-	%MW, Val inm.	%MW, Val inm.	%MW
Palabras no indexables	-	-	-	-	%QW,%SW, %NW
Palabras dobles indexables	-	%MD,Val inm.	-	-	-
Palabras dobles no indexables	-	%QD,%SD	-	-	-

Codificación del parámetro **estado** devuelto después del comando de inicialización:

Valor (en hexadecimal)	Significado
0000	inicialización correctamente efectuada
0201	no hay zona de archivos en la tarjeta de memoria
0202	fallo de la tarjeta de memoria
0204	tarjeta de memoria protegida contra escritura
0241	DEST < 0
0242	DEST + NUM - 1 -> la dirección más alta de la tarjeta
0401	NUM = 0 o negativo
0402	número de alojamiento incorrecto (diferente de 0 ó 1)
0501	Función sin apoyo

Inicialización de la zona de archivado

Presentación

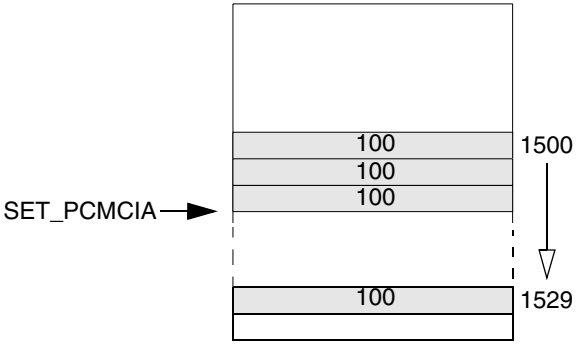
La función **SET_PCMCIA** permite inicializar en el valor deseado todo o parte de la zona de archivado de la tarjeta de memoria de la aplicación (PCMCIA Tipo I).

Esta función utiliza 4 parámetros:

- **DEST**: dirección de la zona de archivado a partir de la cual se efectúa la Inicialización
- **NUM**: número de palabras a inicializar
- **VAL**: valor de la Inicialización
- **CR**: código que devuelve el resultado de la ejecución del comando de Inicialización

Ejemplo

Representación de la tarjeta de memoria de la aplicación:

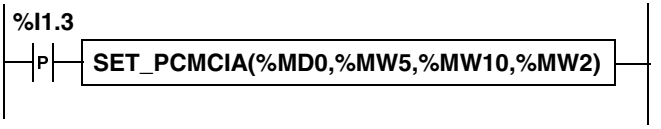


En este ejemplo:

- **DEST** = %MD0 , %MD0 que contiene el valor 1500
- **NUM** = %MW5, %MW5 que contiene el valor 30
- **VAL** = %MW10, %MW10 que contiene el valor 100

Estructura

Lenguaje de contactos :



Lenguaje lista de instrucciones

```
LDR   %I1.3
[SET_PCMCIA(%MD0,%MW5,%MW10,%MW2)]
```

Lenguaje literal estructurado:

```
IF RE %I1.3 THEN
  SET_PCMCIA(%MD0,%MW5,%MW10,%MW2);
END_IF;
```

Sintaxis

Sintaxis de función:

SET_PCMCIA (DEST,NUM,VAL,CR)

Parámetros:

Tipo	**DEST**	**NUM**	**VAL**	**CR**
Palabras indexables	-	%MW,Val inm.	%MW,Val inm.	%MW
Palabras no indexables	-	-	-	%QW,%SW, %NW
Palabras dobles indexables	%MD,Val inm.	-	-	-
Palabras dobles no indexables	%QD,%SD	-	-	-

Codificación del parámetro **CR** devuelto después del comando de inicialización:

Valor (en hexadecimal)	**Significado**
0000	inicialización correctamente efectuada
0201	no hay zona de archivos en la tarjeta de memoria
0202	fallo de la tarjeta de memoria
0204	tarjeta de memoria protegida contra escritura
0241	DEST negativo
0242	EST + NUM - 1 -> la dirección más alta de la tarjeta de memoria
0401	NUM = 0 o negativo

Ecritura de los datos en la zona de archivado extendida

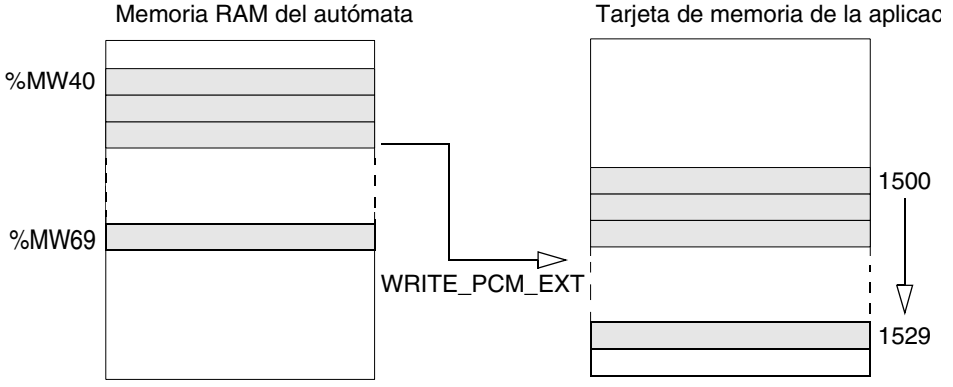
Presentación

La función **WRITE_PCM_EXT** permite transferir datos de la memoria RAM del autómata (palabras %MW) a la zona de archivado de una tarjeta de memoria. Esta función utiliza 5 parámetros:

- **SLOT**: número de la vía donde está insertada la tarjeta de memoria PCMCIA:
  - 0 para una tarjeta situada en el emplazamiento 0 del procesador (tarjeta PCMCIA Tipo I),
  - 1 para una tarjeta situada en el emplazamiento 1 del procesador (tarjeta PCMCIA Tipo III).
- **DEST**: dirección de la zona de archivado a partir de la cual se almacenarán los datos
- **NUM**: número de palabras a almacenar
- **EMIS**: palabra que contiene la dirección de inicio de la zona a transferir a la tarjeta de memoria
- **CR**: código que devuelve el resultado del comando de escritura.

Ejemplo

Ilustración:

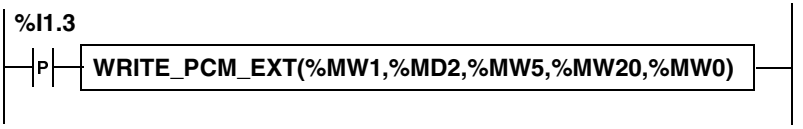


En este ejemplo:

- **SLOT** = %MW1, %MW1 que contiene el valor 1
- **DEST** = %MD2, %MD2 que contiene el valor 1500
- **NUM** = %MW5, %MW5 que contiene el valor 30
- **EMIS** = %MW20, %MW20 que contiene el valor 40

Estructura

Lenguaje de contactos:



Lenguaje lista de instrucciones:

```
LDR   %I1.3
[WRITE_PCM_EXT(%MW1,%MD2,%MW5,%MW20,%MW0)]
```

Lenguaje literal estructurado:

```
IF RE %I1.3 THEN
  WRITE_PCM_EXT(%MW1,%MD2,%MW5,%MW20,%MW0);
END_IF;
```

Sintaxis

Sintaxis de función:

WRITE_PCM_EXT (SLOT,DEST,NUM,VAL,CR)

Parámetros:

Tipo	SLOT	DEST	NUM	EMIS	CR
Palabras indexables	%MW, Val inm.	-	%MW, Val inm.	%MW, Val inm.	%MW
Palabras no indexables	-	-	-	-	%QW,%SW, %NW
Palabras dobles indexables	-	%MD,Val inm.	-	-	-
Palabras dobles no indexables	-	%QD,%SD	-	-	-

Codificación del parámetro **estado** devuelto después del comando de escritura:

Valor (en hexadecimal)	Significado
0000	escritura correctamente efectuada
0102	EMIS + NUM - 1 -> número máximo de %MW declarado en el autómata
0104	ninguna aplicación válida o ninguna %MW en el autómata
0201	no hay zona de archivos en la tarjeta de memoria
0202	fallo de la tarjeta de memoria
0204	tarjeta de memoria protegida contra escritura
0241	DEST < 0
0242	DEST + NUM - 1 -> la dirección más alta de la tarjeta de memoria
0401	NUM = 0
0402	número de alojamiento incorrecto (diferente de 0 ó 1)
0501	Función sin apoyo

Escritura de los datos en la zona de archivado

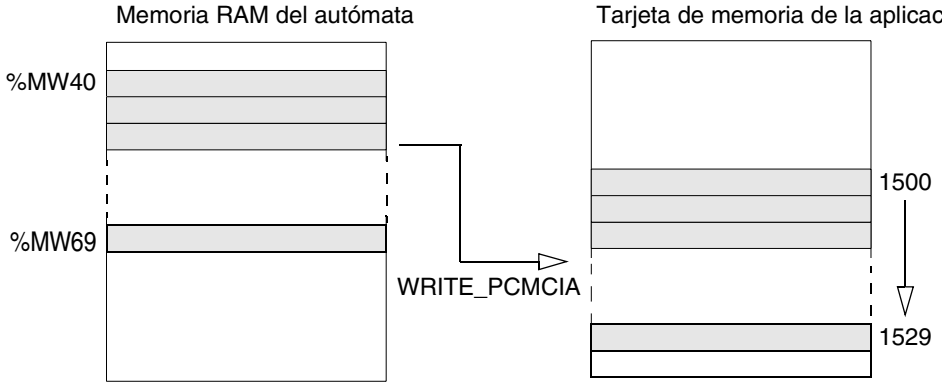
Presentación

La función **WRITE_PCMCIA** permite transferir datos de la memoria RAM del autómata (palabras %MW) a la zona de archivado de la tarjeta de memoria de la aplicación (PCMCIA Tipo 1).
Esta función utiliza 4 parámetros:

- **DEST**: dirección de la zona de archivado a partir de la cual se almacenarán los datos
- **NUM**: número de palabras a almacenar
- **EMIS**: palabra que contiene la dirección de inicio de la zona a transferir en la tarjeta de memoria
- **CR**: código que devuelve el resultado del comando de escritura.

Ejemplo

Ilustración:

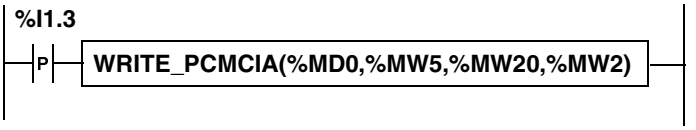


En este ejemplo:

- **DEST** = %MD0, %MD0 que contiene el valor 1500
- **NUM** = %MW5, %MW5 que contiene el valor 30
- **EMIS** = %MW20, %MW20 que contiene el valor 40

Estructura

Lenguaje de contactos:



Lenguaje lista de instrucciones:

```
LDR   %I1.3
[WRITE_PCMCIA(%MD0,%MW5,%MW20,%MW2)]
```

Lenguaje literal estructurado:

```
IF RE %I1.3 THEN
  WRITE_PCMCIA(%MD0,%MW5,%MW20,%MW2);
END_IF;
```

Sintaxis

Sintaxis de función:

WRITE_PCMCIA (DEST,NUM,EMIS,CR)

Parámetros:

Tipo	**DEST**	**NUM**	**EMIS**	**CR**
Palabras indexables	-	%MW,Val inm.	%MW,Val inm.	%MW
Palabras no indexables	-	-	-	%QW,%SW, %NW
Palabras dobles indexables	%MD,Val inm.	-	-	-
Palabras dobles no indexables	%QD,%SD	-	-	-

Codificación del parámetro **CR** devuelto después del comando de escritura:

Valor (en hexadecimal)	**Significado**
0000	escritura correctamente efectuada
0102	EMIS + NUM - 1 -> número máximo de %MW declarado en el autómata
0104	ninguna aplicación válida o ninguna %MW en el autómata
0201	no hay zona de archivos en la tarjeta de memoria
0202	fallo de la tarjeta de memoria
0204	tarjeta de memoria protegida contra escritura
0241	DEST < 0
0242	DEST + NUM - 1 -> la dirección más alta de la tarjeta de memoria
0401	NUM = 0

Lectura de los datos en la zona de guardado extendida

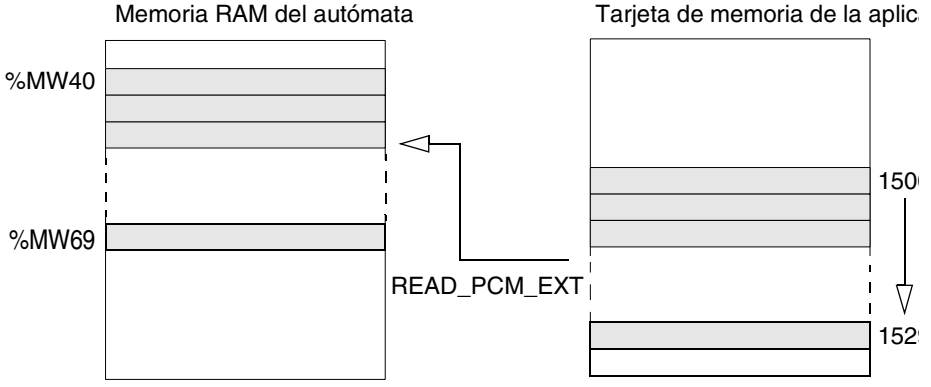
Presentación

La función **READ_PCM_EXT** permite transferir datos de la zona de archivado de la tarjeta de memoria de la aplicación a la memoria RAM del autómata (palabras %MW).
Esta función utiliza 5 parámetros:

- **SLOT**: número de la vía donde está insertada la tarjeta de memoria PCMCIA:
  - 0 para una tarjeta situada en el emplazamiento 0 del procesador (tarjeta PCMCIA Tipo I),
  - 1 para una tarjeta situada en el emplazamiento 1 del procesador (tarjeta PCMCIA Tipo III).
- **SRC**: dirección de la zona de archivado en la que se almacenan los datos a leer
- **NUM**: número de palabras a leer
- **RCPT**: palabra que contiene la dirección de inicio de la zona transferida por la tarjeta de memoria
- **CR**: código que devuelve el resultado de la ejecución del comando de lectura

Ejemplo

Ilustración:

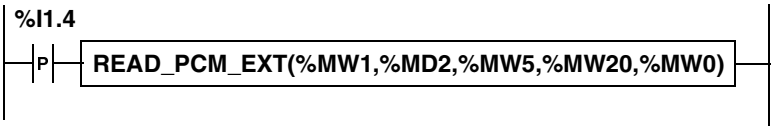


En este ejemplo:

- **SLOT** = %MW1, %MW1 que contiene el valor 1
- **SRC** = %MD2, %MD2 que contiene el valor 1500
- **NUM** = %MW5, %MW5 que contiene el valor 30
- **RCPT** = %MW20, %MW20 que contiene el valor 40

Estructura

Lenguaje de contactos:



Lenguaje lista de instrucciones:

```
LDR   %I1.4
[READ_PCM_EXT(%MW1,%MD2,%MW5,%MW20,%MW0)]
```

Lenguaje literal estructurado:

```
IF RE %I1.4 THEN
  READ_PCM_EXT(%MW1,%MD2,%MW5,%MW20,%MW0);
END_IF;
```

Sintaxis

Sintaxis de función:

READ_PCM_EXT (SLOT,SRC,NUM,RCPT,CR)

Parámetros:

Tipo	SLOT	SRC	NUM	RCPT	CR
Palabras indexables	%MW, Val inm.	-	%MW, Val inm.	%MW, Val inm.	%MW
Palabras no indexables	-	-	-	-	%QW,%SW, %NW
Palabras dobles indexables	-	%MD,Val inm.	-	-	-
Palabras dobles no indexables	-	%QD,%SD	-	-	-

Codificación del parámetro **CR** devuelto después del comando de escritura:

Valor (en hexadecimal)	Significado
0000	lectura correctamente efectuada
0102	SRC + NUM -1 -> número máximo de %MW declarado en el autómata
0104	ninguna aplicación válida o ninguna %MW en el autómata
0201	no hay zona de archivos en la tarjeta de memoria
0202	fallo de la tarjeta de memoria
0204	tarjeta de memoria protegida contra escritura
0241	SRC < 0
0242	SRC + NUM -1 -> la dirección más alta de la tarjeta de memoria
0401	NUM = 0
0402	número de alojamiento incorrecto (diferente de 0 ó 1)
0501	Función sin apoyo

Lectura de los datos en la zona de archivado

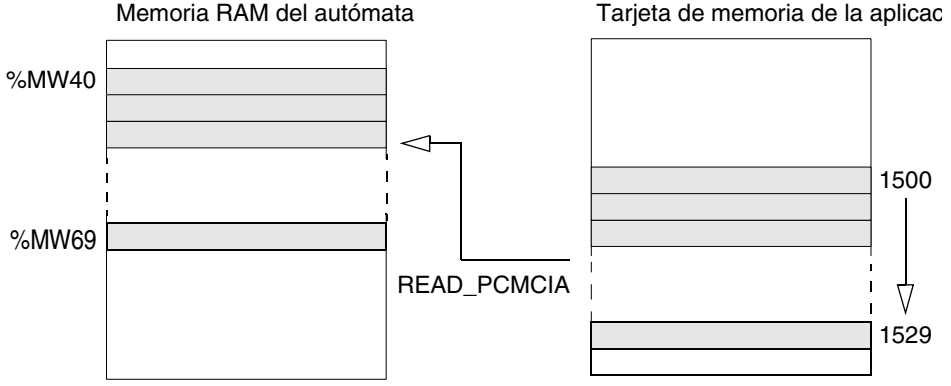
Presentación

La función **READ_PCMCIA** permite transferir datos de la zona de archivado de la tarjeta de memoria de la aplicación (PCMCIA Tipo I) a la memoria RAM del autómata (palabras %MW).
Esta función utiliza 4 parámetros:

- **SRC**: dirección de la zona de archivado en la que están almacenados los datos a leer
- **NUM**: número de palabras a leer
- **RCPT**: palabra que contiene la dirección de inicio de la zona transferida por la tarjeta de memoria
- **CR**: código que devuelve el resultado de la ejecución del comando de lectura

Ejemplo

Ilustración:

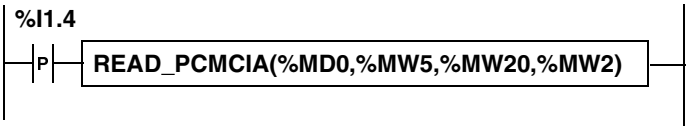


En este ejemplo:

- **SRC** = %MD0, %MD0 que contiene el valor 1500
- **NUM** = %MW5, %MW5 que contiene el valor 30
- **RCPT** = %MW20, %MW20 que contiene el valor 40

Estructura

Lenguaje de contactos:



Lenguaje lista de instrucciones:

```
LDR   %I1.4
[READ_PCMCIA(%MD0,%MW5,%MW20,%MW2)]
```

Lenguaje literal estructurado:

```
IF RE %I1.4 THEN
  READ_PCMCIA(%MD0,%MW5,%MW20,%MW2);
END_IF;
```

Sintaxis

Sintaxis de función:

READ_PCMCIA (SRC,NUM,RCPT,CR)

Parámetros:

Tipo	**SRC**	**NUM**	**RCPT**	**CR**
Palabras indexables	-	%MW,Val inm.	%MW,Val inm.	%MW
Palabras no indexables	-	-	-	%QW,%SW, %NW
Palabras dobles indexables	%MD,Val inm.	-	-	-
Palabras dobles no indexables	%QD,%SD	-	-	-

Codificación del parámetro **CR** devuelto después del comando de escritura:

Valor (en hexadecimal)	**Significado**
0000	lectura correctamente efectuada
0102	RCPT + NUM - 1 -> número máximo de %MW declarado en el autómata
0104	ninguna aplicación válida o ninguna %MW en el autómata
0201	no hay zona de archivos en la tarjeta de memoria
0202	fallo de la tarjeta de memoria
0204	tarjeta de memoria protegida contra escritura
0241	SRC < 0
0242	RCPT + NUM - 1 > la dirección más alta de la tarjeta de memoria
0401	NUM= 0

2.13 Funciones Grafcet

Función puesta a cero de los tiempos de actividades de etapas

Generalidades

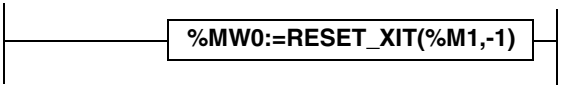
Esta función reinicia todos los tiempos de actividades del tratamiento secuencial "Chart" o de una macroetapa.
Esta función es válida en el autómata Premium/Atrium (versión de programa superior o igual a V3.0).

Ésta dispone de los siguientes parámetros de entrada y salida:

Tipo	Parámetros	Función
Entrada	En	Condiciones de activación de la función
	Núm	Número del módulo Grafcet que se va a reiniciar. Tiene un valor de: ● -1 si el módulo es el tratamiento secuencial "Chart", ● o el número de la macroetapa implicada.
Salida	Resultado	Confirmación de la ejecución de la función.

Estructura

Lenguaje de contactos



Lenguaje de lista de instrucciones

```
LD verdadero
[%MW0:=RESET_XIT(%M1,-1)]
```

Lenguaje literal estructurado

```
%MW0:=RESET_XIT(%M1,-1);
```

Sintaxis

Operador:

Sintaxis
Result:=RESET_XIT(En,Num)

Operandos:

Tipo	**Resultado**	**Condición de validación (En)**	**Número de módulo Grafcet (Núm)**
Bits	-	%M	-
Palabras	%MW	-	%MW, %KW, Valor inmediato

Resultado

Codificación del resultado del parámetro tras la ejecución de la instrucción:

Valor (hexadecimal)	**Significado**
0000	Operación correcta
FFFF	Parámetro de entrada fuera de los límites: la macro etapa no existe en la aplicación.
FFFA	El autómata es de tipo MICRO

Objetos de sistema

3

Presentación

Contenido En este capítulo se describen todos los bits de sistema y palabras de sistema del lenguaje PL7

Contenido: Este capítulo contiene las siguientes secciones:

3.1 Bits de sistema

Presentación

Objeto de este apartado

Este capítulo describe los bits de sistema del lenguaje PL7.

Contenido

Esta sección contiene los siguientes apartados:

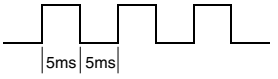
Presentación de los bits de sistema

Generalidades

Los autómatas TSX 37 y TSX 57 disponen de bits de sistema %Si que indican los estados del autómata o que permiten actuar sobre el funcionamiento de éste. Dichos bits pueden probarse en el programa del usuario con el fin de detectar cualquier suceso de funcionamiento que conlleve un procedimiento de tratamiento particular. Algunos de ellos deben volver a su estado inicial o normal por programa. No obstante, los bits de sistema que vuelven a su estado inicial o normal a través del sistema, no deben hacerlo a través del programa ni del terminal.

Descripción de los bits de sistema %S0 a %S7

Descripción detallada Descripción de los bits de sistema %S0 a %S7

Bit	Función	Descripción	Estado inicial	TSX37	TSX57
%S0	Arranque en frío	Normalmente en estado 0; vuelve al estado 1 mediante: ● restablecimiento de corriente con pérdida de datos (fallo de batería), ● programa de usuario, ● terminal, ● cambio de cartucho, ● pulsación en el botón RESET. Este bit pasa a 1 en el primer ciclo completo. Vuelve a 0 antes del ciclo siguiente. (Funcionamiento (("Modos de functionamientos" - Manual de referencia Tomo 1)))	0	SÍ	SÍ
%S1	Rearranque en caliente	Normalmente en estado 0; vuelve al estado 1 mediante: ● restablecimiento de corriente guardando los datos, ● programa de usuario, ● terminal. El sistema vuelve a ponerlo a 0 al final del primer ciclo completo y antes de la actualización de las salidas. (Funcionamiento (("Modos de functionamientos" - Manual de referencia Tomo 1)))	0	SÍ	SÍ
%S4	Base de tiempo de 10 ms	Bit cuyo cambio de estado depende de un reloj interno. Es asíncrono respecto al ciclo del autómata. Diagrama:  	-	SÍ	SÍ
%S5	Base de tiempo de 100 ms	Igual que %S4	-	SÍ	SÍ
%S6	Base de tiempo de 1 s	Igual que %S4	-	SÍ	SÍ
%S7	Base de tiempo de 1 mn	Igual que %S4	-	SÍ	SÍ

Descripción de los bits de sistema %S8 a %S16

Descripción detallada Descripción de los bits de sistema %S8 a %S16

Bit	Función	Descripción	Estado inicial	TSX37	TSX57
%S8	Prueba del cableado	Normalmente en estado 1, este bit se utiliza para realizar la prueba de cableado, cableado cuando el autómata TSX 37 se encuentra en estado "no configurado" ● estado 1: las salidas están forzadas a 0, ● estado 0: un terminal de ajuste puede modificar las salidas.	1	SÍ	NO
%S9	Puesta a 0 de las salidas	Normalmente el programa de las salidas o el terminal puede establecer el estado 0 a estado 1: ● estado 1: puesta a 0 de todas la salidas TON y analógicas, independientemente del modo de retorno que se configura para cada módulo, ● estado 0: las salidas se actualizan normalmente.,	0	SÍ	NO
%S9	Paso en modo retorno de las salidas en todos los buses	Normalmente el programa o el terminal puede establecer el estado 0 a estado 1: ● estado 1: paso en modo retorno (0 ó 1) en función de la selección de configuración de todas las salidas TON, Analógicas ..., ● estado 0: las salidas se actualizan normalmente.,	0	NO	SÍ
%S10	Fallo E/S	Normalmente el estado 1 pasa al estado 0 cuando se detecta un fallo de E/S de un módulo en rack o de un módulo remoto (FIPIO) (configuración no conforme, fallo de intercambio, fallo de hardware). El bit %S10 pasa a 1 cuando desaparece el error.	1	SÍ	SÍ
%S11	Rebasamiento del watchdog	Normalmente el sistema pasa el estado 0 a estado 1 cuando el tiempo de ejecución de una tarea es superior al tiempo de ejecución máximo (watchdog) que aparece en la configuración. El rebasamiento del watchdog provoca el paso en STOP del autómata y la aplicación se detiene en estado de error (el indicador ERR parpadea).	0	SÍ	SÍ
%S13	Primer ciclo después de la puesta en RUN	Normalmente el sistema pasa el estado 0 a estado 1 durante el primer ciclo después de la puesta en RUN del autómata.	-	SÍ	SÍ

Bit	Función	Descripción	Estado inicial	TSX37	TSX57
%S15	Fallo de la cadena de caracteres	Normalmente el estado 0, pasa a estado 1 cuando la zona de destino de una transferencia de cadena de caracteres no tiene el tamaño suficiente para acoger a esta cadena de caracteres, el usuario deberá volver a poner este bit a 0. Cada cadena gestiona su propio bit %S15.	0	SÍ	SÍ
%S16	Fallo de E/S tarea	Normalmente el sistema pasa el estado 0 a estado 1 cuando se produce un fallo en un módulo de E/S en rack o en estado remoto en un FIPIO configurado en la tarea, el usuario deberá volver a poner a 1 este bit. Cada tarea gestiona su propio bit %S16.	1	SÍ	SÍ

Descripción de los bits de sistema %S17 a %S20

Descripción detallada Descripción de los bits de sistema %S17 a %S20

Bit	Función	Descripción	Estado inicial	TSX37	TSX57
%S17	Bit por desplazamiento o informe aritmético	Normalmente en estado 0; vuelve al estado 1 mediante el sistema: • cuando una operación de desplazamiento contiene el estado del último bit, • cuando se produce un rebasamiento aritmético no firmado (fechas). El usuario debe volver a poner el bit a 0.	0	SÍ	SÍ
%S18	Rebasamiento o error aritmético	Normalmente en estado 0. Pasa al estado 1 en caso de rebasamiento de capacidad en una operación de 16 por alguno de los motivos siguientes: • resultado superior a + 32767 o inferior a - 32768, en longitud simple, • resultado superior a +2.147.483.647 o inferior a - 2.147.483.648, en longitud doble, • resultado superior a +3.402824E+38 o inferior a - 3.402824E+38, en flotante (versión de programa > 1.0), • rebasamiento de capacidad en DCB, • división por 0, • raíz de un número negativo, • forzado a un paso inexistente en un programador cíclico, • apilado de un registro completo, desapilado de un registro vacío. Debe probarse mediante el programa del usuario después de cada operación en la que exista riesgo de rebasamiento; si es el caso, el usuario debe volver a ponerlo a 0. Cada tarea gestiona su propio bit %S18.	0	SÍ	SÍ
%S19	Rebasamiento del período de tarea (exploración periódica)	Normalmente en estado 0. El sistema pone este bit en estado 1 en caso de rebasamiento del período de ejecución (tiempo de ejecución de tarea superior al período definido por el usuario en configuración o programado en la palabra %SW asociada a la tarea). El usuario debe volver a poner el bit en estado 0. Cada tarea gestiona su propio bit %S19.	0	SÍ	SÍ

Bit	Función	Descripción	Estado inicial	TSX37	TSX57
%S20	Rebasamiento de índice	Normalmente en estado 0. Pasa al estado 1 cuando la dirección del objeto indexado es inferior a 0 o supera el número de objetos declarado en la configuración. Debe probarse mediante el programa del usuario después de cada operación en la que exista riesgo de rebasamiento; si es el caso, vuelve a 0. Cada tarea gestiona su propio bit %S20.	0	SÍ	SÍ

Descripción de los bits de sistema %S21 a %S26

Descripción detallada Bits de sistema %S21 a %S26 asociados al Grafcet

Bit	Función	Descripción	Estado inicial	TSX37	TSX57
%S21	Inicialización	El usuario gestiona este bit para inicializar el Grafcet (preferentemente, puesta a 1 en el tratamiento preliminar). El sistema vuelve a ponerlo a 0 después de la inicialización del Grafcet (al final del tratamiento preliminar, cuando se evalúa el nuevo estado del Grafcet). La inicialización del Grafcet consiste en desactivar todas las etapas activas y en activar las etapas iniciales. En un arranque en frío, el sistema sitúa este bit en 1 durante el tratamiento preliminar.	0	SÍ	SÍ
%S22	Reset del Grafcet	Normalmente en estado 0. Este bit sólo puede ponerse en estado 1 mediante programa en el tratamiento preliminar. En estado 1, provoca la desactivación de todas las etapas del Grafcet. El sistema vuelve a ponerlo a 0 después de la validación al final del tratamiento preliminar.	0	SÍ	SÍ
%S23	Fijación del Grafcet	Normalmente en estado 0. La puesta en estado 1 de %S23 conserva el estado de los Grafcet. Los Grafcet no cambian cualquiera que sea el valor de las receptividades descendentes de las etapas activas. La inmovilización se mantiene mientras el bit %S23 esté en 1. El programa del usuario gestiona este bit; sólo pasa a 1 o a 0 en el tratamiento preliminar.	0	SÍ	SÍ
%S24	Reset de las macroetapas	Normalmente en estado 0. La puesta a 1 de %S24 provoca la puesta a cero de las macroetapas elegidas en una tabla de 4 palabras de sistema %SW22 a %SW25. El sistema vuelve a ponerlo a 0 después de la validación al final del tratamiento preliminar.	0	NO	SÍ
%S26	Rebasamiento de las tablas (etapas/ transiciones)	Normalmente en estado 0. El sistema de tablas pone este bit en estado 1 cuando se superan las posibilidades de activación (etapas o transiciones) o cuando se ejecuta un gráfico incorrecto (reenvío de destino en una etapa que no pertenece al gráfico). El rebasamiento provoca el paso a STOP del autómata. Este bit vuelve al estado 0 cuando se inicializa el terminal.	0	SÍ	SÍ

Descripción de los bits de sistema %S30 a %S59

Descripción detallada Descripción de los bits de sistema %S30 a %S59

Bit	Función	Descripción	Estado inicial	TSX37	TSX57
%S30	Activación/ desactivación de la tarea maestra	Normalmente en estado 1. La puesta a cero por el usuario provoca la desactivación de la tarea maestra.	1	SÍ	SÍ
%S31	Activación de la tarea rápida	Normalmente en estado 1. La puesta a 0 por el usuario provoca la desactivación de la tarea rápida.	1	SÍ	SÍ
%S38	Validación/ inhibición de los sucesos	Normalmente en estado 1. La puesta a 0 por el usuario provoca la inhibición de los sucesos.	1	SÍ	SÍ
%S39	Saturación en el tratamiento de los sucesos	El sistema pone a 1 este bit para indicar que uno o varios sucesos no se pueden tratar debido a la saturación de las colas de espera. El usuario debe volver a poner el bit en estado 0.	0	SÍ	SÍ
%S40 a **%S47**	Fallo de E/S (racks) estáticas desactivadas	Los bits %S40 a %S47 están asignados respectivamente a los racks 0 a 7 y normalmente se encuentran en el estado 1. Cada uno de estos bits pasa a 0 cuando se produce un fallo en las entradas/salidas del rack correspondiente. El bit vuelve al estado 1 cuando desaparece el fallo.	1	NO	SÍ
%S49	Reactivación de las salidas	Normalmente en estado 0. El usuario puede poner este bit en 1 para solicitar una activación cada 10s a partir de la aparición del fallo de las salidas estáticas activadas por sobreintensidad o cortocircuito.	0	SÍ	NO
%S50	Actualización de la fecha y la hora por las palabras %SW50 a 53	Normalmente en estado 0. Se puede poner en 1 o en 0 mediante programa o terminal. ● en estado 0: acceso a la fecha y la hora mediante lectura por las palabras del sistema %SW50 a 53, ● en estado 1: actualización de la fecha y la hora mediante escritura de las palabras de sistema %SW50 a 53.	0	SÍ	SÍ
%S51	Pérdida de la hora del reloj-calendario	Este bit, gestionado por el sistema, indica en el estado 1 que el reloj-calendario no está presente o que sus palabras de sistema no son significativas. En tal caso, debe efectuarse una puesta en hora del reloj-calendario.	0	SÍ	SÍ

Bit	Función	Descripción	Estado inicial	TSX37	TSX57
%S59	Actualización de la fecha y la hora por la palabra %SW59	Normalmente en estado 0. Se puede poner en 1 o en 0 mediante programa o terminal. ● en estado 0: el sistema no gestiona la palabra %SW59, ● en estado 1: el sistema gestiona los flancos en la palabra %SW59 para ajustar la fecha y la hora actuales (por incrementos).	0	SÍ	SÍ

Descripción de los bits de sistema de %S60 a %S69

Descripción detallada Descripción de los bits de sistema de %S60 a %S69

Bit	Función	Descripción	Estado inicial	TSX37	TSX57
%S60	Comando de conmutación voluntaria	Bit de sistema que controla la conmutación voluntaria en caso de puesta en marcha de una arquitectura redundante (véase su utilización en el manual "Warm Standby Premium"). El usuario o la aplicación pueden volver a poner el bit a 0.	0	NO	SÍ
%S66	Gestión del indicador de la batería	Normalmente en estado 0. Se puede poner en 1 o en 0 mediante el programa o el terminal. Permite encender o no el indicador de la batería en caso de ausencia o fallo de la pila de guardado: ● Estado 0: el indicador de la batería se enciende cuando no está la pila de salvaguarda o ésta presenta algún fallo. ● Estado 1: el indicador de la batería siempre está apagado. En un arranque en frío, el sistema pone %S66 a 0.	0	SÍ	NO
%S67	Estado de la pila de la tarjeta de memoria	Este bit permite controlar el estado de funcionamiento de la pila de salvaguarda de la tarjeta de memoria RAM: ● Estado 0: pila presente y en funcionamiento. ● Estado 1: pila ausente o fuera de servicio.	-	SÍ	SÍ
%S68	Estado de la pila del procesador	Este bit permite controlar el estado de funcionamiento de la pila de salvaguarda de los datos y del programa en la memoria RAM: ● Estado 0: pila presente y en funcionamiento. ● Estado 1: pila ausente o fuera de servicio.	-	SÍ	SÍ
%S69	Visualización de los datos de usuario en los visualizadores del autómata	Normalmente en estado 0. Se puede poner en 1 o en 0 mediante el programa o el terminal: ● Estado 0: modo "Word" de los visualizadores desactivado. ● Estado 1: modo "Word" de los visualizadores activado.	0	SÍ	NO

Descripción de los bits de sistema %S70 a %S92

Descripción detallada Descripción de los bits de sistema %S70 a %S92

Bit	Función	Descripción	Estado inicial	TSX37	TSX57
%S70	Actualización de los datos en el bus AS-i o en el enlace TSX Nano	El sistema pone este bit a 1 en cada fin de ciclo del enlace TSX Nano o de ciclo de interrogación del bus AS-i. Cuando se conecta, este bit indica que todos los datos se actualizan al menos una vez y que, por lo tanto, son significativos. El usuario vuelve a poner este bit a 0.	0	SÍ	SÍ
%S73	Paso a modo protegido en el bus AS-i	Normalmente en el estado 0, el usuario pone este bit a 1 para pasar a modo protegido en el bus AS-i. Previamente, el bit %S74 deberá estar en el estado 1. Este bit sólo se utilizará en una prueba de cableado y no se aplicará al autómata.	0	SÍ	NO
%S74	Salvaguarda de la configuración presente en el bus AS-i	Normalmente en el estado 0, el usuario pone este bit a 1 para que se guarde la configuración presente en el bus AS-i. Este bit sólo se utilizará en una prueba de cableado y no se aplicará al autómata.	0	SÍ	NO
%S75	Prueba de la pila de la tarjeta de memoria Data Archiving	Normalmente a 1 (pila OK), este bit pasa a 0 cuando la pila está gastada o defectuosa.	1	NO	SÍ
%S80	Puesta a cero del contador de mensajes	Normalmente en el estado 0, el usuario puede poner este bit a 1 para volver a poner a cero los contadores de mensajes %SW80 a %SW86.	0	SÍ	SÍ
%S90	Actualización de las palabras comunes	Actualiza todos los segundos. El programa o el terminal puede poner este bit a 0.	0	SÍ	SÍ
%S92	Paso a modo de medida de función de comunicación	Normalmente en el estado 0, el usuario puede poner este bit a 1 para colocar las funciones de comunicación en modo de medida de rendimiento. El parámetro de tiempo de espera de las funciones de comunicación muestra, por lo tanto, el tiempo de intercambio de ida y vuelta en decenas de ms (si este tiempo <10 s; en caso contrario, no es significativo).	0	SÍ	SÍ

Descripción de los bits de sistema %S94 a %S99

Descripción detallada Descripción de los bits de sistema %S94 a %S99

Bit	Función	Descripción	Estado inicial	TSX37	TSX57
%S94	Guardado de los ajustes DFB	Normalmente en estado 0. El usuario puede poner en 1 este bit para guardar los valores de ajuste de los bloques de función del usuario.	0	NO	SÍ
%S95	Restablecimiento de los ajustes DFB	Normalmente en estado 0. El usuario puede poner en 1 este bit para restablecer los valores de ajuste de los bloques de función del usuario.	0	NO	SÍ
%S96	Validación del guardado del programa de aplicación	• en estado 0: guardado del programa de aplicación no válido, • en estado 1: guardado del programa de aplicación válido. Este bit puede leerse en cualquier momento (mediante programa o ajuste) y concretamente después de un arranque en frío o un rearranque en caliente. Es significativo respecto a una copia de seguridad de la aplicación realizada mediante PL7 en la memoria Flash EPROM interna.	-	SÍ	NO
%S97	Validación del guardado de %MW	• en estado 0: guardado de los %MW no válidos, • en estado 1: guardado de los %MW válidos. Este bit puede leerse en cualquier momento (mediante programa o ajuste) y concretamente después de un arranque en frío o un rearranque en caliente.	-	SÍ	NO
%S98	Conexión remota del pulsador del acoplador TSX SAZ 10	Normalmente en estado 0. El usuario gestiona este bit: • en estado 0: pulsador del acoplador TSX SAZ 10 activo, • en estado 1: pulsador del acoplador TSX SAZ 10 sustituido por una entrada TON (palabra %SW98 (Véase *Descripción de las palabras de sistema %SW98 a %SW109, p. 344*)).	0	SÍ	NO
%S99	Conexión remota del pulsador del bloque de visualización	Normalmente en estado 0. El usuario gestiona este bit: • en estado 0: pulsador del bloque de visualización centralizada activo, • en estado 1: pulsador del bloque de visualización centralizada sustituido por una entrada TON (palabra %SW99 (Véase *Descripción de las palabras de sistema %SW98 a %SW109, p. 344*)).	0	SÍ	NO

Descripción de los bits de sistema %S100 a %S119

Descripción detallada Descripción de los bits de sistema %S100 a %S119

Bit	Función	Descripción	Estado inicial	TSX37	TSX57
%S100	Protocolo en la toma del terminal	Situado en 0 o en 1 por el sistema según el estado del puente INL/DPT en la toma de la consola. ● si falta el puente (%S100=0), se utiliza el protocolo UNI-TELWAY maestro, ● si el puente está presente (%S100=1), se utiliza el protocolo indicado por la configuración de la aplicación.	-	SÍ	SÍ
%S101	Buffer de diagnóstico configurado	El sistema pone a 1 este bit cuando está configurada la opción de diagnóstico; se reserva entonces un buffer de diagnóstico destinado al almacenamiento de los errores procedentes de los DFB.	-	SÍ	SÍ
%S102	Buffer de diagnóstico lleno	El sistema pone a 1 este bit cuando el buffer que recibe los errores de los bloques de función de diagnóstico está lleno.	-	SÍ	SÍ
%S118	Fallo general de E/S FIPIO	Normalmente en 1. El sistema pone a 0 este bit cuando aparece un fallo en un equipo conectado al bus FIPIO. Cuando desaparece el fallo, el sistema vuelve a poner el bit a 1.	1	SÍ	SÍ
%S119	Fallo general de E/S en rack	Normalmente en 1. El sistema pone a 0 este bit cuando aparece un fallo en un módulo de E/S instalado en alguno de los racks. Cuando desaparece el fallo, el sistema vuelve a poner el bit a 1.	1	SÍ	SÍ

3.2 Palabras de sistema

Presentación

Objeto de este apartado

Este apartado describe las palabras de sistema del lenguaje PL7.

Contenido

Esta sección contiene los siguientes apartados:

Descripción de las palabras de sistema %SW0 a %SW11

Descripción detallada Descripción de las palabras de sistema %SW0 a %SW11

Palabras	Función	Descripción	Gestión
%SW0	Período de exploración de la tarea maestra	Permite modificar el período de la tarea maestra definida en la configuración mediante el programa de usuario o el terminal. El período se expresa en ms (1..255 ms). %SW0=0 en funcionamiento cíclico. En arranque en frío: toma el valor definido por configuración.	Usuario
%SW1	Período de exploración de la tarea rápida	Permite modificar el período de la tarea rápida definida en la configuración mediante el programa de usuario o el terminal. El período se expresa en ms (1..255 ms). En arranque en frío: toma el valor definido por configuración.	Usuario
%SW8	Control de adquisición de las entradas de las tareas	Normalmente en estado 0. Se puede poner en 1 o en 0 mediante programa o terminal. Permite inhibir la fase de adquisición de las entradas de cada tarea. ● %SW8:X0 =1 asignado a la tarea MAST: ya no se controlan las tareas relativas a esta tarea. ● %SW8:X1 =1 asignado a la tarea FAST: ya no se controlan las tareas relativas a esta tarea.	Usuario
%SW9	Control de la actualización de las salidas de las tareas	Normalmente en estado 0. Se puede poner en 1 o en 0 mediante programa o terminal. Permite inhibir la fase de actualización de las salidas de cada tarea. ● %SW9:X0 =1 asignado a la tarea MAST: ya no se controlan las tareas relativas a esta tarea. ● %SW9:X1 =1 asignado a la tarea FAST: ya no se controlan las tareas relativas a esta tarea.	Usuario
%SW10	Primer ciclo después de un arranque en frío	El valor 0 del bit de la tarea en curso significa que ejecuta su primer ciclo después del arranque en frío. ● %SW10:X0: asignado a la tarea maestra MAST ● %SW10:X1: asignado a la tarea rápida FAST	Sistema
%SW11	Duración del watchdog	Permite leer la duración del watchdog definida en la configuración. Se expresa en ms (10...500 ms).	Sistema

AVISO

Relativo a las palabras de sistema %SW8 y %SW9:

Atención: las salidas de los módulos conectados al bus X pasan automáticamente al modo de retorno, las salidas de los equipos conectados al bus FIPIO se mantienen en el estado que precede a la puesta en 1 del bit.

Si no se respetan estas precauciones pueden producirse graves lesiones o daños materiales

Descripción de las palabras de sistema %SW12 a %SW18

Descripción detallada Descripción de las palabras de sistema %SW12 a %SW18

Palabras	Función	Descripción	Gestión
%SW12	Dirección UNI-TELWAY de la toma del terminal	Dirección UNI_TELWAY de la toma del terminal (en modo esclavo) definida en la configuración y cargada en esta palabra en un arranque en frío.	Sistema
%SW13	Dirección principal de la estación	Indica, para la red principal: ● el número de la estación (byte menos significativo) de 0 a 127 ● el número de la red (byte más significativo) de 0 a 63 (valor de los micro interruptores de la tarjeta PCMCIA)	Sistema
%SW17	Estado de fallo en operación flotante	Cuando se detecta un fallo en una operación aritmética flotante, el bit %S18 pasa a 1 y el estado de fallo de %SW17 se actualiza según la codificación siguiente: ● %SW17:X0 = operación no válida; el resultado no es un número. ● %SW17:X1 = operando sin normalizar; el resultado es correcto. ● %SW17:X2 = división por 0; el resultado es el infinito. ● %SW17:X2 = desbordamiento; el resultado es el infinito. ● %SW17:X4 = rebasamiento; el resultado es 0. ● %SW17:X5 = imprecisión del resultado. El sistema vuelve a poner a 0 esta palabra en el arranque en frío y el programa también para utilizarla de nuevo.	Sistema Usuario
%SW18	Contador de tiempo absoluto	Esta palabra doble permite efectuar cálculos de duración. Se incrementa cada 1/10º de segundo por el sistema (incluso con el autómata en STOP). Puede leerse y escribirse mediante programa de usuario o terminal.	Sistema Usuario

Descripción de las palabras de sistema %SW20 a %SW25

Descripción detallada Descripción de las palabras de sistema %SW20 a %SW25 (asociadas al Grafcet)

Palabras	Función	Descripción	Gestión
%SW20	Nivel de actividad del Grafcet	Esta palabra contiene el número de etapas activas, por activar y por desactivar respecto al ciclo actual. El sistema la actualiza en cada evolución del gráfico.	Sistema
%SW21	Tabla de validación de las transiciones Grafcet	Esta palabra contiene el número de transiciones válidas, por validar y por invalidar respecto al ciclo actual. El sistema la actualiza en cada evolución del gráfico.	Sistema
%SW22 a %SW25	Tabla de puesta a 0 de macroetapa	A cada bit de esta tabla corresponde una macroetapa con %SW22:X0 para XM0 ....%SW25:X16 para XM63. Las macroetapas cuyo bit asociado en la tabla esté en 0 volverán a 0 cuando el bit %S24 pase a 1.	Usuario

Descripción de las palabras de sistema %SW30 a %SW35

Descripción detallada Descripción de las palabras de sistema %SW30 a %SW35

Palabras	Función	Descripción	Gestión
%SW30	Tiempo de ejecución de la tarea maestra	Indica el tiempo de ejecución del último ciclo de la tarea maestra (en ms).	Sistema
%SW31	Tiempo de ejecución máximo de la tarea maestra	Indica el tiempo de ejecución más largo de tarea maestra desde el último arranque en frío (en ms).	Sistema
%SW32	Tiempo de ejecución mínimo de la tarea maestra	Indica el tiempo de ejecución más corto de tarea maestra desde el último arranque en frío (en ms).	Sistema
%SW33	Tiempo de ejecución de la tarea rápida	Indica el tiempo de ejecución del último ciclo de la tarea rápida (en ms).	Sistema
%SW34	Tiempo de ejecución máximo de la tarea rápida	Indica el tiempo de ejecución más largo de tarea rápida desde el último arranque en frío (en ms).	Sistema
%SW35	Tiempo de ejecución mínimo de la tarea rápida	Indica el tiempo de ejecución más corto de tarea rápida desde el último arranque en frío (en ms).	Sistema

Nota: Precisión sobre el tiempo de ejecución : es el tiempo transcurrido entre el principio (adquisición de las entradas) y el final (actualización de las salidas) de un ciclo de exploración. Este tiempo incluye el tratamiento de las tareas de sucesos y de la tarea rápida, así como el tratamiento de las peticiones de la consola.

Descripción de las palabras de sistema %SW48 a %SW59

Descripción detallada Descripción de las palabras de sistema %SW48 a %SW59

Palabras	Función	Descripción	Gestión
%SW48	Número de sucesos	Indica el número de sucesos tratados desde el último arranque en frío (en ms). Dicha palabra puede escribirse por programa o por terminal.	Sistema Usuario
%SW49 **%SW50** **%SW51** **%SW52** **%SW53**	Función reloj-calendario (1)	Palabras de sistema que contienen la fecha y la hora actuales (en BCD): ● %SW49: día de la semana (desde 1 para lunes hasta 7 para domingo). ● %SW50: segundos (SS00) ● %SW51: horas y minutos (HHMM) ● %SW52: mes y día (MMDD) ● %SW53: año (AAAA) El sistema gestiona estas palabras cuando el bit %S50 está en 0. Estas palabras pueden escribirse mediante programa de usuario o terminal cuando el bit %S50 está en 1 (Véase *Descripción de los bits de sistema %S30 a %S59, p. 320*).	Sistema Usuario
%SW54 **%SW55** **%SW56** **%SW57** **%SW58**	Función reloj-calendario (1)	Palabras de sistema que contienen la fecha y la hora del último fallo de corriente o de parada del autómata (en BCD): ● %SW54: segundos (00SS) ● %SW55: horas y minutos (HHMM) ● %SW56: mes y día (MMDD) ● %SW57: año (AAAA) ● %SW58: el byte más significativo contiene el día de la semana (desde 1 para lunes hasta 7 para domingo).	Sistema
%SW58	Código de la última parada	El byte menos significativo contiene el código de la última parada: ● 1 = paso de RUN a STOP mediante el terminal ● 2 = parada por fallo de programa (rebasamiento de la tarea del autómata) ● 4 = corte de corriente ● 5 = parada por fallo de hardware ● 6 = parada por instrucción HALT	Sistema

Palabras	Función	Descripción	Gestión
%SW59	Ajuste de la fecha actual	Contiene dos series de 8 bits para ajustar la fecha actual. La acción se realiza siempre en al flanco ascendente del bit. El bit %S59 valida esta palabra. Figura:  	Usuario

Nota: (1) Únicamente en los autómatas TSX 37-21/22 y TSX 57

Descripción de las palabras de sistema %SW60 a %SW62

Descripción detallada Descripción de las palabras de sistema %SW60 a %SW61 específicas del diagnóstico de Warm Standby Premium

Palabras	Función	Descripción	Gestión
%SW60	Diagnóstico del autómata redundante	Diagnóstico específico de la redundancia de un autómata local (Warm Standby Premium) Significado de los diferentes bits de la palabra %SW60: • %SW60:X0=1 indica que el autómata se encuentra en estado 'Normal' • %SW60:X0=1 indica que el autómata se encuentra en estado 'de emergencia' • %SW60:X3=0 indica un fallo de entradas/salidas en FIPIO en el autómata Normal; es la imagen del bit %S118 • %SW60:X4=0 indica un fallo de entradas/salidas en rack; es la imagen del bit %S119 • %SW60:X5=1 indica un fallo detectado mediante autopruebas en al menos uno de los TSX ETY 210 • %SW60:X7=1 indica un fallo grave de la red FIPIO, por ejemplo, un cortocircuito o un bloque de terminales desconectado • %SW60:X8=1 indica un fallo en el módulo TSX ETY 110 utilizado para el enlace entre autómatas • %SW60:X9=1 indica un fallo en la comunicación entre autómatas, no es posible recuperar el diagnóstico del autómata doble • %SW60:X10 es un bit reservado • %SW60:X11=1 indica que el autómata de emergencia no puede pasar a autómata normal; esta información sólo se genera en el autómata normal, no es relevante en el autómata de emergencia • %SW60:X12=0 indica que el autómata es la estación A • %SW60:X12=1 indica que el autómata es la estación B • %SW60:X13=1 indica el modo Run del autómata • %SW60:X14=1 indica el modo Stop del autómata • %SW60:X15=1 indica el modo Halt del autómata	Sistema

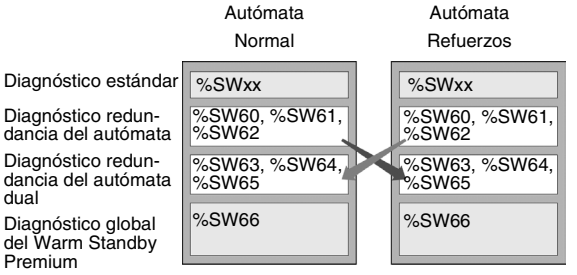
Palabras	Función	Descripción	Gestión
%SW61	Diagnóstico del autómata redundante	Significado de los diferentes bits de la palabra %SW61: ● %SW61:X0=1 indica un problema de intercambio de la Base de Datos a través del enlace Ethway entre autómatas; esta información sólo se genera para el autómata normal en Run ● %SW60:X0=1 indica que el autómata se encuentra en estado 'de emergencia' ● %SW61:X1=1 indica un problema de comunicación entre el módulo TSX ETY 210 cliente TCP-IP de otro equipo. Esta información sólo se genera para el autómata normal en Run. Cuando el bit pasa a 1, se provoca una conmutación si el autómata de emergencia puede pasar a normal ● %SW60:X4=0 indica un fallo de entradas/salidas en rack; es la imagen del bit %S119 ● %SW60:X5=1 indica un fallo detectado mediante autopruebas en al menos uno de los TSX ETY 210 ● %SW61:X2 es un bit reservado ● %SW61:X3 es un bit reservado ● %SW61:X4=1 indica un primer intercambio correcto de la Base de Datos ● %SW61:X5=1 indica que la función de redundancia ha puesto el procesador en Stop; el diagnóstico se obtiene en la palabra %MWp.14.2 ● %SW61:X6=1 es un bit reservado ● %SW61:X7=0 indica un problema de configuración o de funcionamiento de la función de redundancia; el diagnóstico se obtiene en la palabra %MWp.14.2 ● %SW61:X7=1 indica que la función de redundancia se ha configurado correctamente ● %SW61:X8 a %SW61:X15 son bits reservados p: designa el emplazamiento del procesador	Sistema

Palabras	Función	Descripción	Gestión
%SW61	Diagnóstico del autómata redundante	Significado de los diferentes bits de la palabra %SW61: ● %SW61:X0=1 indica un problema de intercambio de la Base de Datos a través del enlace Ethway entre autómatas; esta información sólo se genera para el autómata normal en Run ● %SW60:X0=1 indica que el autómata se encuentra en estado 'de emergencia' ● %SW61:X1=1 indica un problema de comunicación entre el módulo TSX ETY 210 cliente TCP-IP de otro equipo. Esta información sólo se genera para el autómata normal en Run. Cuando el bit pasa a 1, se provoca una conmutación si el autómata de emergencia puede pasar a normal ● %SW60:X4=0 indica un fallo de entradas/salidas en rack; es la imagen del bit %S119 ● %SW60:X5=1 indica un fallo detectado mediante autopruebas en al menos uno de los TSX ETY 210 ● %SW61:X2 es un bit reservado ● %SW61:X3 es un bit reservado ● %SW61:X4=1 indica un primer intercambio correcto de la Base de Datos ● %SW61:X5=1 indica que la función de redundancia ha puesto el procesador en Stop; el diagnóstico se obtiene en la palabra %MWp.14.2 ● %SW61:X6=1 es un bit reservado ● %SW61:X7=0 indica un problema de configuración o de funcionamiento de la función de redundancia; el diagnóstico se obtiene en la palabra %MWp.14.2 ● %SW61:X7=1 indica que la función de redundancia se ha configurado correctamente ● %SW61:X8 a %SW61:X15 son bits reservados p: designa el emplazamiento del procesador	Sistema

Palabras	Función	Descripción	Gestión
%SW62	Visualización de la función de árbitro de bus y productor / consumidor del bus FIPIO.	El byte menos significativo indica el estado de la función de productor / consumidor. El byte más significativo indica el estado de la función de árbitro de bus (BA). Valor del byte: ● 16#00: la función no existe (sin aplicación FIPIO), ● 16#07: la función está en curso de STOP BA (la orden de STOP BA se envía, el comando no ha terminado). ● 16#0F: la función está en curso de RUN BA (la orden de RUN BA se envía, el comando no ha terminado). ● 16#70: la función se inicializa pero no está operativa (en STOP BA), ● 16#F0: la función está en curso de ejecución normal (en RUN BA).	Sistema

Descripción de las palabras de sistema %SW63 a %SW65

Descripción detallada Descripción de las palabras de sistema %SW63 a %SW65 específicas del diagnóstico de Warm Standby Premium

Palabras	Función	Descripción	Gestión
%SW63 a %SW65	Intercambio de las palabras de diagnóstico entre autómatas	El diagnóstico del autómata doble está disponible en las palabras %SW63 a %SW65. Las palabras %SW63, %SW64 y %SW65 del autómata normal contienen respectivamente las palabras %SW60, %SW61 y %SW62 del autómata de emergencia. Asimismo, las palabras %SW63, %SW64 y %SW65 del autómata de emergencia contienen respectivamente las palabras %SW60, %SW61 y %SW62 del autómata normal. Figura  El intercambio de palabras se lleva a cabo a través del enlace Ethway entre autómatas (módulo TSX ETY 110)	Sistema

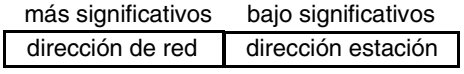
Descripción de las palabras de sistema %SW66 a %SW69

Descripción detallada Descripción de la palabra de sistema %SW66 específica del diagnóstico de Warm Standby Premium

Palabras	Función	Descripción	Gestión
%SW66	Diagnóstico general de la arquitectura Warm Standby Premium	En cada uno de los autómatas se elabora un diagnóstico general de la arquitectura Warm Standby Premium, a partir de los diagnósticos de redundancia de los dos autómatas. Dicho diagnóstico se guarda en %SW66, cuyos bits tienen el significado que se indica a continuación:	Sistema
	• %SW66:X0=0 indica un funcionamiento degradado de Warm Standby Premium • %SW66:X0=1 indica un funcionamiento nominal de Warm Standby Premium • %SW60:X1=1 indica que el autómata A es el autómata normal • %SW66:X2=1 indica que el autómata B es el autómata normal • %SW66:X3=1 indica un fallo de comunicación entre autómatas **Información relativa al autómata A** • %SW66:X4=1 indica un fallo grave de la red FIPIO en el autómata A • %SW66:X5=1 indica que el autómata A está en STOP • %SW66:X6=1 indica que el autómata A está en Halt • %SW66:X7=1 indica un fallo de comunicación Ethernet TCP-IP del autómata A (módulo TSX ETY 210 o función cliente) • %SW66:X8=1 indica un fallo en al menos uno de los módulos en rack del autómata A • %SW66:X9=1 indica un fallo en al menos uno de los equipos FIPIO del autómata A **Información relativa al autómata B** • %SW66:X10=1 indica un fallo grave de la red FIPIO en el autómata B • %SW66:X11=1 indica que el autómata B está en STOP • %SW66:X12=1 indica que el autómata B está en Halt • %SW66:X13=1 indica un fallo de comunicación Ethernet TCP-IP del autómata B (módulo TSX ETY 210 o función cliente) • %SW66:X14=1 indica un fallo en al menos uno de los módulos en rack del autómata B • %SW66:X15=1 indica un fallo en al menos uno de los equipos FIPIO del autómata B		

Nota: La información de %SW66:X4 a %SW66:X15 no es significativa si existe un fallo de comunicación entre autómatas (%SW66:X3=1)

Descripción de las palabras de sistema %SW67 a %SW69 específicas del diagnóstico de Warm Standby Premium

Palabras	Función	Descripción	Gestión
%SW67	Dirección de red y de estación del autómata doble	Esta palabra contiene la dirección de red y de estación del autómata doble, lo que permite establecer la comunicación entre autómatas. Dicha palabra debe visualizarse en hexadecimal para interpretarse como sigue:  	Sistema
%SW68 **%SW69**	Base de tiempo utilizada por los EF Tempo.	Estas palabras contienen una base de tiempo utilizada por los EF Tempo. Se transfiere desde el autómata normal hacia el autómata de emergencia para la actualización y la sincronización.	Sistema

Descripción de las palabras de sistema %SW80 a %SW89

Descripción detallada Descripción de las palabras de sistema %SW80 a %SW89

Palabras	Función	Descripción	Gestión
%SW80 **%SW81** **%SW82** **%SW83** **%SW84** **%SW85** **%SW86**	Gestión de mensajes y telegramas	• %SW80: nº de mensajes emitidos por el sistema hacia la toma de terminal. • %SW81: nº de mensajes recibidos por el sistema desde la toma de terminal. • %SW82: nº de mensajes emitidos por el sistema hacia el acoplador PCMCIA. • %SW83: nº de mensajes recibidos por el sistema desde el acoplador PCMCIA. • %SW84: nº de telegramas emitidos por el sistema. • %SW85: nº de telegramas recibidos por el sistema. • %SW86: nº de mensajes rechazados por el sistema.	Sistema Usuario
%SW87 **%SW88** **%SW89**	Gestión de los flujos de comunicación (1)	• %SW87: número de peticiones tratadas por el servidor síncrono por ciclo de la tarea maestra (MAST). • %SW88: número de peticiones tratadas por el servidor asíncrono por ciclo de la tarea maestra (MAST). • %SW89: número de peticiones tratadas por funciones del servidor (inmediato) por ciclo de la tarea maestra (MAST).	Sistema

Nota: (1) Únicamente en autómatas TSX/PCX/PMX 57

Descripción de las palabras de sistema %SW96 y %SW97

Descripción detallada Estas palabras sólo existen en el TSX 37
Descripción de las palabras de sistema %SW96 y %SW97

Palabras	Función	Descripción	Gestión

%SW96	Comando o diagnóstico de la función de guardado/ restablecimiento del programa de aplicación y de %MW	• bit 0: petición de transferencia hacia la zona de guardado. Este bit está activo en el flanco ascendente. El sistema vuelve a ponerlo a 0 desde el restablecimiento de la validación del flanco ascendente.
		• bit 1: cuando el bit tiene el valor 1, significa que la función de guardado ha terminado. El bit vuelve a 0 desde la validación del flanco ascendente en el bit 0.
		• bit 2: informe del guardado:
		  • 0 -> guardado sin error,
		  • 1 -> error durante el guardado.
		• bits 3 a 5: reservados.
		• bit 6: validación del guardado del programa de aplicación (igual que %S96).
		• bits 8 a 15: este byte sólo es significativo si el bit de confirmación está en 1 (bit 2 = 1, error al guardar):
		  • 1 -> número de %MW para guardar superior al número de %MW configurado
		  • 2 -> número de %MW para guardar superior a 1000 o inferior a 0,
		  • 3 -> número de %MW para restablecer superior al número de %MW configurado,
		  • 4 -> tamaño de la aplicación en la RAM interna superior a 15 Kpalabras (se recuerda que el guardado de las %MW está siempre asociado a un guardado del programa de aplicación en la Flash EPROM interna),
		  • 5 -> servicio prohibido en RUN,
		  • 6 -> presencia de un cartucho de copia de seguridad en el autómata,
		  • 7 -> fallo de escritura en la Flash EPROM.
			Sistema Usuario

Palabras	Función	Descripción	Gestión
%SW97	Número de %MW para guardar	Permite programar el número de %MW que se van a guardar. Cuando la palabra está comprendida entre 1 y 1000, los primeros 1 a 1000 %MW se transfieren a la Flash EPROM interna. Cuando la palabra vale 0, el programa de la aplicación contenido en la RAM interna es el único que se transfiere a la Flash EPROM interna. **Se borra un eventual guardado de %MW.** En un arranque en frío, esta palabra se inicializa a -1 si la Flash EPROM interna no contiene ningún guardado de %MW. De lo contrario, se inicializa al valor del número de palabras guardadas.	Usuario

Descripción de las palabras de sistema %SW98 a %SW109

Descripción detallada Descripción de las palabras de sistema %SW98 a %SW109

Palabras	Función	Descripción	Gestión
%SW98	Dirección geográfica del módulo/vía de la entrada TON (2)	Cuando el bit %S98 = 1, la palabra indica la dirección geográfica de la entrada TON (módulo/vía), en sustitución del pulsador del acoplador TSX SAZ 10 : más significativos bajo significativos número de módulo | número de vía	Usuario
%SW99	Dirección de la entrada TON (2)	Cuando el bit %S99 = 1, la palabra indica la dirección geográfica de la entrada TON (módulo/vía), en sustitución del pulsador del bloque de visualización centralizada. más significativos bajo significativos número de módulo | número de vía	Usuario
%SW108	Contador de vías forzadas	Contabiliza las vías forzadas a 0 o a 1 en la aplicación. Las vías se actualizan mediante forzado o cancelación de forzado.	Sistema
%SW109	Contador de vías analógicas forzadas	Contabiliza las vías analógicas forzadas a 0.	Sistema

Nota: (2) Únicamente en TSX 37

Descripción de la palabra de sistema %SW116

Descripción detallada Descripción de la palabra de sistema %SW116 - FIPIO

Palabras	Función	Descripción	Gestión
%SW116	Fallo de E/S FIPIO en la tarea	Normalmente en 0. Cada bit de esta palabra es significativa de un estado de intercambio FIPIO **en la tarea** en la que se prueba. El usuario debe volver a poner esta palabra a 0. Detalle de los bits de la palabra %SW116: ● x0 = 1 error de intercambio explícito (la variable no se intercambia en el bus), ● x1 = 1 tiempo de espera en un intercambio explícito (sin respuesta al final del tiempo de espera), ● x2 = 1 número máximo de intercambios explícitos simultáneos alcanzado, ● x3 = 1 una trama es incorrecta, ● x4 = 1 la longitud de una trama recibida es superior a la longitud declarada, ● x5 = reservado a 0, ● x6 = 1 una trama no es válida o un agente se inicializa, ● x7 = 1 ausencia de equipo configurado, ● x8 = 1 fallo de vía (al menos una vía de un equipo indica un fallo), ● x9 a x14 = reservado a 0, ● x15 = fallo general (O de los bits 3, 4, 6, 7, 8).	Sistema Usuario

Descripción de las palabras de sistema de %SW124 a %SW127

Descripción detallada Descripción de las palabras de sistema de %SW124 a %SW127

Palabras	Función	Descripción	Gestión
%SW124	Tipo de fallo de la CPU	El sistema escribe en esta palabra el último tipo de fallo de CPU encontrado (estos códigos se intercambian en un arranque en frío): ● 16#30 : fallo del código del sistema ● 16#60 a 64: rebasamiento de pila ● 16#90 : fallo del sistema de interrupción: IT imprevista ● 16#53 : fallo de tiempo de espera en los intercambios de E/S	Sistema

%SW125	Tipo de fallo de bloqueo	El sistema escribe en esta palabra el último tipo de fallo de bloqueo encontrado:
		● 16#DEB0: rebasamiento de watchdog
		● 16#2258 : ejecución de la instrucción HALT
		● 16#DEF8: ejecución de un instrucción JMP a una etiqueta sin definir
		● 16#2XXX: ejecución de un instrucción CALL a un subprograma sin definir
		● 16#0XXX: ejecución de una función desconocida
		● 16#DEFE: el programa Grafcet incluye reenvíos a etapas sin definir.
		● 16#DEFF: flotante no implementado
		● División entre 0:
		  ● con enteros, (16#DEF0 --> %SW125), (1-->%S18) y (0-->%SW17),
		  ● con flotantes (16#DE87 --> %SW125), (1-->%S18) y (4-->%SW17).
		● 16#DEF1: error de transferencia de cadena de caracteres (1-->%S15)
		● Rebasamiento de capacidad (overflow):
		  ● con enteros, (16#DEF2 --> %SW125), (1-->%S18) y (0-->%SW17),
		  ● con flotantes (16#DE87 --> %SW125), (1-->%S18) y (8-->%SW17).
		● 16#DEF3: rebasamiento de índice (1-->%S20)
			Sistema

%SW126 **%SW127**	Dirección de la instrucción del fallo de bloqueo	Dirección de la instrucción que ha generado el fallo de bloqueo. ● %SW126 contiene el offset de esta dirección ● %SW127 contiene la base de esta dirección	Sistema

Descripción de las palabras de sistema %SW128 a %SW143

Descripción detallada Descripción de las palabras de sistema %SW128 a %SW143 - FIPIO

Palabras	Función	Descripción	Gestión
%SW128 a %SW143	Fallo del punto de conexión FIPIO	Cada bit de este grupo indica el estado de un equipo conectado al bus FIPIO. Normalmente en 1. El estado 0 de uno de estos bits indica la aparición de un fallo en este punto de conexión. Para un punto de conexión no configurado, el bit correspondiente siempre es 1.	Sistema

Tabla de correspondencia entre los bits de las palabras y la dirección de un punto de conexión

	x0	x1	x2	x3	x4	x5	x6	x7	x8	x9	x10	x11	x12	x13	x14	x15
%SW128 :	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
%SW129 :	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
%SW130 :	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
%SW131 :	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
%SW132 :	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
%SW133 :	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
%SW134 :	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
%SW135 :	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
%SW136 :	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
%SW137 :	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
%SW138 :	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
%SW139 :	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
%SW140 :	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
%SW141 :	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
%SW142 :	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
%SW143 :	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Descripción de las palabras de sistema %SW144 a %SW146

Descripción detallada Descripción de las palabras de sistema %SW144 a %SW146 - FIPIO

Palabras	Función	Descripción	Gestión
%SW144	Modo de marcha de la función de árbitro de bus FIPIO	Esta palabra de sistema permite detener e iniciar la función de árbitro de bus y la función de productor/consumidor. Permite modificar el modo de arranque (automático y manual) del bus en caso de parada. • x0 = 1 función de productor/consumidor en RUN • x0 = 0 función de productor/consumidor en STOP (no se intercambia ninguna variable en el bus) • x1 = 1 el árbitro de bus está en RUN • x1 = 0 el árbitro de bus está en STOP (no se realiza ninguna exploración de variables ni de mensajes en el bus) • x2 = 1 arranque automático en caso de parada automática del bus • x2 = 0 arranque manual en caso de parada automática del bus • x3 = reservado en 1 • x4 a x15 reservados en 0	Usuario
%SW145	Modificación de los parámetros del árbitro de bus FIPIO	El usuario pone a 1 los bits y a continuación el sistema a 0 cuando se efectúa la inicialización. • x0 = 1 modificación de la prioridad del árbitro de bus: el byte más significativo de esta palabra de sistema contiene el valor de la prioridad del árbitro de bus que se aplicará a éste • x1 a x2 están reservados • x3 a x7 reservados en 0 • x8 a x15: este byte contiene el valor que se aplicará al bus, según el valor del bit x0 La modificación de estos parámetros puede realizarse cuando el árbitro de bus está en RUN, pero la validación por parte de la aplicación requiere la detención y el posterior arranque de ésta.	Usuario Sistema

Palabras	Función	Descripción	Gestión
%SW146	Visualización de la función de árbitro de bus FIPIO	El byte menos significativo indica el estado de la función de productor/consumidor. El byte más significativo indica el estado de la función de árbitro de bus. Valor del byte: ● 16#00: la función no existe (sin aplicación FIPIO) ● 16#07: la función está en curso de STOP (la orden de STOP se envía, el comando no ha terminado) ● 16#0F: la función está en curso de RUN (la orden de RUN se envía, el comando no ha terminado) ● 16#70: la función se inicializa pero no está operativa (en STOP) ● 16#F0: la función está en curso de ejecución normal (en RUN)	Sistema

AVISO

Relativo a las palabras %SW144 y %SW145

La modificación de estas palabras de sistema puede implicar la parada de la estación de autómata.

Si no se respetan estas precauciones pueden producirse graves lesiones o daños materiales

Descripción de las palabras de sistema %SW147 a %SW152

Descripción detallada Descripción de las palabras de sistema %SW147 a %SW152

Palabras	Función	Descripción	Gestión
%SW147	Tiempo del ciclo de la red MAST	Un valor distinto de cero indica, en ms, el valor del tiempo del ciclo de la red (TCR-MAST) de la tarea MAST.	Sistema
%SW148	Valor del tiempo de ciclo de red de la tarea FAST.	Un valor distinto de cero indica, en ms, el valor del tiempo del ciclo de la red (TCR-FAST) de la tarea FAST.	Sistema
%SW149		Reservado a 0.	Sistema
%SW150	Número de tramas emitidas	Esta palabra indica el número de tramas emitidas por el administrador de la vía FIPIO.	Sistema
%SW151	Número de tramas recibidas	Esta palabra indica el número de tramas recibidas por el administrador de la vía FIPIO.	Sistema
%SW152	Número de mensajes restablecidos	Esta palabra indica el número de restablecimientos de mensajes efectuados por el administrador de la vía FIPIO.	Sistema

Descripción de la palabra de sistema %SW153

Descripción detallada Descripción de la palabra de sistema %SW153 - FIPIO

Palabras	Función	Descripción	Gestión
%SW153	lista de los fallos del administrador de la vía FIPIO.	El sistema pone a 1 cada bit y el usuario los vuelve a poner a 0. Véase la lista siguiente.	Usuario Sistema

Descripción de los bits

- X0 = fallo de rebasamiento de la estación: corresponde a una pérdida de símbolo MAC en la recepción, vinculada a una reacción demasiado lenta del receptor.
- X1 = fallo de rechazo de mensaje: indica un mensaje con confirmación denegada o sin confirmación. MAC en recepción,
- X0 = fallo de underrun de la estación: corresponde a la incapacidad de la estación para respetar la velocidad de emisión en la red.
- X4 = fallo de capa física: corresponde a una ausencia prolongada de transmisión en la capa física.
- X5 = fallo de ausencia de eco: corresponde a un fallo para el cual el emisor está en curso de emisión, con una corriente de emisión comprendida en el rango de funcionamiento y detección simultánea de ausencia de señal en la misma vía.
- X6 = fallo de conversación: corresponde a un fallo para el cual el emisor dispone del control de la línea desde un tiempo superior al límite máximo de funcionamiento definido. Este fallo puede deberse, por ejemplo, a un deterioro del modulador o a una capa de enlace de datos defectuosa.
- X7 = fallo de hipocorriente: corresponde a un fallo en el cual el emisor genera en su línea, cuando se solicita, una corriente inferior al límite mínimo de funcionamiento definido. Este fallo se debe, por ejemplo, al aumento de la impedancia de línea (línea abierta...).
- X8 = fallo de trama agujereada: indica la recepción de un silencio en el cuerpo de una trama después de identificar a un delimitador de principio de trama y antes de identificar a un delimitador de final de trama. La aparición de un silencio en condiciones normales de funcionamiento tiene lugar después de identificar un delimitador de final de trama.
- X9 = fallo de trama CRC en la recepción: indica una diferencia de valor entre el CRC calculado en la trama que se recibe normalmente y el CRC contenido en la misma.
- X10 = fallo de codificación de trama en la recepción: indica la recepción de determinados símbolos, pertenecientes exclusivamente a las secuencias de delimitación de principio y fin de la trama, en el cuerpo de ésta.
- X11 = fallo de longitud de la trama recibida: el número de bytes recibidos para el cuerpo de una trama es superior a 256 bytes.
- X12 = recepción de una trama de tipo desconocido: en el cuerpo de una trama, el primer byte identifica el tipo de trama de enlace. En el protocolo de enlace de la norma WORLDFIP se define un determinado número de tipos de tramas. La presencia de cualquier otro código en una trama corresponde a un fallo de tipo de trama desconocido.
- X13 = recepción de una trama truncada: un fragmento de trama se caracteriza por el reconocimiento de una secuencia de símbolos del delimitador de final de trama cuando la estación de destino esperaba recibir un delimitador de principio de trama.
- X14 = no se utiliza, valor no significativo
- X15 = no se utiliza, valor no significativo

Descripción de la palabra de sistema %SW154

Descripción detallada

Descripción de la palabra de sistema %SW154 - FIPIO

Palabras	Función	Descripción	Gestión
%SW154	lista de los fallos del administrador de la vía FIPIO.	El sistema pone a 1 cada bit y el usuario los vuelve a poner a 0. Véase la lista siguiente.	Usuario Sistema

Descripción de los bits

- X0 = tiempo de espera de secuencia aperiódica: indica un rebasamiento de la ventana de mensajes o de variables aperiódicas en un ciclo elemental del macrociclo.
- X1 = rechazo de petición de mensajería: indica una saturación de la cola de espera de mensajes; el árbitro de bus ya no puede momentáneamente almacenar y a continuación satisfacer una petición.
- X2 = rechazo de comando de actualización urgente: indica una saturación de la cola de espera de peticiones de intercambio de variables aperiódicas urgentes; el árbitro de bus ya no puede momentáneamente almacenar ni satisfacer la petición.
- X3 = rechazo de comando de actualización no urgente: indica una saturación de la cola de espera de peticiones de intercambio de variables aperiódicas no urgentes; el árbitro de bus ya no puede momentáneamente almacenar ni satisfacer la petición.
- X4 = fallo de silencio: el árbitro de bus no ha detectado ninguna actividad en el bus durante un período superior al tiempo normalizado WorldFip.
- X5 = colisión en la red en la emisión de identificador: indica una actividad en la red durante los períodos teóricos de silencio. Entre una emisión y la espera de una respuesta por parte del árbitro de bus, no debe circular nada en el bus. Si el árbitro de bus detecta una actividad, genera un fallo de colisión (por ejemplo, cuando varios árbitros están activos al mismo tiempo en el bus).
- X6 = fallo de overrun del árbitro de bus: indica un conflicto de acceso a la memoria de la estación del árbitro de bus.
- X7 = no se utiliza, valor no significativo
- x8 a x15 = reservados a 0.

Descripción de las palabras de sistema %SW155 a %SW162

Descripción detallada Descripción de las palabras de sistema %SW155 a %SW162

Palabras	Función	Descripción	Gestión
%SW155	Número de intercambios explícitos	Número de intercambios explícitos en curso de tratamiento	Sistema
%SW160		Resultado del último registro (función de diagnóstico).	Sistema
%SW161		Resultado de la última cancelación de registro (función de diagnóstico).	Sistema
%SW162		Número de errores en curso en el buffer de diagnóstico.	Sistema

Índice

Symbols

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

W

X