Manual de Referencia

PL7 Micro/Junior/Pro

Descripción del programa PL7

TLX DR PL7 xx spa

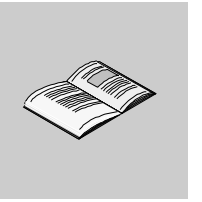
Documentos relacionados

Presentación

Este manual consta de dos tomos:

- Tomo 1: Descripción del programa PL7
  - Generalidades
  - Lenguaje de contactos
  - Lenguaje lista de instrucciones
  - Lenguaje literal estructurado
  - Lenguaje Grafcet
  - Bloques de función DFB
  - Módulos Funcionales
- Tomo 2: Descripción detallada de las Instrucciones y 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
  - Palabras reservadas
  - Conformidad con la norma CEI1131-3
  - Servidor OLE Automation
  - Rendimiento

Tabla de materias



Acerca de este 11

Parte I Descripción del programa PL7 13
Presentación 13

Capítulo 1 Presentación del programa PL7 15
Presentación 15
Presentación de los programas PL7 16
Presentación de los lenguajes PL7 17
Estructura de programa PL7 20
Módulos funcionales 22

Capítulo 2 Descripción de los objetos en lenguaje PL7 25
Presentación 25
Definición de los principales objetos booleanos 26
Definición de los principales objetos palabras 27
Direccionamiento de los objetos bits 29
Direccionamiento de los objetos de módulos de entradas/salidas del TSX 37 . . 31
Direccionamiento de los objetos de módulos de entradas/salidas en rack 34
Direccionamiento de los objetos de lenguaje de módulos remotos en el bus FIPIO 37
Direccionamiento de objetos de lenguaje referentes al bus AS-i 40
Direccionamiento de los objetos palabras 42
Regla de solapamientos 46
Objetos del bloque de función 47
Objetos PL7 del tipo tabla 50
Objetos indexados 53
Objetos Grafcet 56
Simbolización 57
Objetos presimbolizados 59

Capítulo 3 Memoria del usuario 61
Presentación 61
Estructura de la memoria de los autómatas Micro 62
Estructura de memoria de los autómatas Premium 64

Descripción de la memoria bits 66
Descripción de la memoria palabras 68
Características de la memoria de los autómatas TSX 37 70
Características de la memoria de los autómatas TSX/PCX 57 10/15/20/25/26/2873
Características de la memoria de los autómatas TSX/PCX 57 30/35/36 76
Características de la memoria de los autómatas TSX/PCX 57 453/4823 78

Capítulo 4 Modos de funcionamiento 81
Presentación 81
Tratamiento en el corte y rearranque del sector 82
Tratamiento del rearranque en caliente 84
Tratamiento del arranque en frío 86

Capítulo 5 Estructura del programa 91
Presentación 91
5.1 Descripción de las tareas. 92
Presentación 92
Presentación de la tarea maestra 93
Descripción de las secciones de los subprogramas 94
Presentación de la tarea rápida 98
Presentación de los tratamientos de sucesos 99
5.2 Estructura monotarea 101
Presentación 101
Estructura del programa monotarea 102
Ejecución cíclica 103
Ejecución periódica 105
Control del tiempo del ciclo 108
5.3 Estructura multitarea 109
Presentación 109
Estructura del programa de multitarea. 110
Desglose secuencial de las tareas en una estructura multitarea. 111
Asignación de las vías de entradas/salidas a las tareas maestra y rápida 112
Intercambios de entradas/salidas en los tratamientos de sucesos 113
5.4 Módulos funcionales 116
Estructuración en módulos funcionales 116

Parte II Descripción de los lenguajes PL7 117
Presentación 117

Capítulo 6 Lenguaje de contactos 119
Presentación 119
Presentación general del lenguaje de contactos 120
Estructura de una red de contactos 121
Etiqueta de una red de contactos 122
Comentario de una red de contactos. 123
Elementos gráficos del lenguaje de contactos. 124

Reglas de programación de una red de contactos 127
Regla de programación de los bloques de función 128
Reglas de programación de los bloques de operación 129
Ejecución de una red de contactos 130

Capítulo 7 Lenguaje lista de instrucciones 133
Presentación 133
Presentación general del lenguaje lista de instrucciones 134
Estructura de un programa de lista de instrucciones 135
Etiqueta de una frase en lenguaje de lista de instrucciones 136
Comentario de una frase en lenguaje de lista de instrucciones 137
Presentación de las instrucciones en lenguaje de lista de instrucciones 138
Regla de empleo de los paréntesis en el lenguaje de lista de instrucciones. . . 143
Descripción de las instrucciones MPS, MRD y MPP 145
Principios de programación de los bloques de función definidos con anterioridad 147
Reglas de ejecución de un programa en lista de instrucciones 149

Capítulo 8 Lenguaje literal estructurado 151
Presentación 151
Presentación del lenguaje literal estructurado 152
Estructura de un programa en lenguaje literal estructurado 153
Etiqueta de una frase en lenguaje literal estructurado 154
Comentario de una frase en lenguaje literal estructurado. 155
Instrucciones sobre objetos bits 156
Instrucciones aritméticas y lógicas 157
Instrucciones para las tablas y cadenas de caracteres 159
Instrucciones de conversiones numéricas 162
Instrucciones del programa e instrucciones específicas 163
Estructura de control condicional IF...THEN 165
Estructura de control condicional WHILE...END_WHILE 167
Estructura de control condicional REPEAT...END_REPEAT 168
Estructura de control condicional FOR...END_FOR 169
Instrucción de salida de bucle EXIT 170
Reglas de ejecución de un programa literal estructurado 171

Capítulo 9 Grafcet 175
Presentación 175
9.1 Presentación general del Grafcet 176
Presentación 176
Presentación del Grafcet. 177
Descripción de los símbolos gráficos del Grafcet 178
Descripción de los objetos específicos del Grafcet. 181
Posibilidades del Grafcet. 183
9.2 Regla de construcción del Grafcet 184
Presentación 184

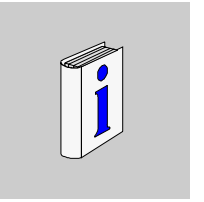
Representación del Grafcet 185
Aplicación de las divergencias y convergencias O 186
Aplicación de las divergencias y convergencias Y 187
Utilización de los reenvíos 188
Utilización de los enlaces orientados 191
Comentario Grafcet 192
9.3 Programación de las acciones y de las condiciones 193
Presentación 193
Programación de las acciones asociadas a las etapas 194
Programación de las acciones para la activación o la desactivación 196
Programación de las acciones continuas 197
Programación de las receptividades asociadas a las transiciones 198
Programación de las receptividades en lenguaje de contactos 199
Programación de las receptividades en lenguaje lista de instrucciones 200
Programación de las receptividades en lenguaje literal estructurado 201
9.4 Macro etapas 202
Presentación 202
Presentación de las macro etapas 203
Constitución de una macro etapa 204
Características de las macro etapas 205
9.5 Sección Grafcet 207
Presentación 207
Estructura de una sección Grafcet 208
Descripción del tratamiento preliminar 209
Posicionamiento previo del Grafcet 210
Inicialización del Grafcet 211
Reset del Grafcet 212
Inmovilización del Grafcet 213
Reset de las macroetapas 214
Funcionamiento del tratamiento secuencial 216
Descripción del tratamiento posterior 218

Capítulo 10 Bloques de función DFB 221
Presentación 221
Presentación de los bloques de función DFB 222
De qué manera se puede poner en marcha un bloque de función DFB 224
Definición de los objetos de los bloques de función tipo DFB 226
Definición de los parámetros DFB 229
Definición de las variables DFB 230
Regla de codificación de los tipos DFB 232
Creación de instancias del DFB 234
Regla de utilización de los DFB en un programa 235
Utilización de un DFB en un programa en lenguaje de contactos 237
Utilización de un DFB en un programa en lenguaje lista de instrucciones o literal 238
Ejecución de una instancia DFB 239

Ejemplo de programación del bloque de función DFB 240

Índice 243

Acerca de este



Presentación

Objeto	Este manual describe en su totalidad las instrucciones y objetos direccionables del lenguaje de programación de los autómatas programables Micro, Premium y Atrium.
Campo de aplicación	La actualización de esta publicación tiene en cuenta la funcionalidad de PL7 V4.4. Ello permite, sin embargo, poner en marcha las versiones anteriores de PL7.
Comentarios del usuario	Envíe sus comentarios a la dirección electrónica TECHCOMM@modicon.com

Descripción del programa PL7

I

Presentación

Objetivo de esta parte

Esta parte presenta el programa PL7. Describe las nociones elementales básicas imprescindibles para la programación de los autómatas Micro y Premium.

Contenido

Esta parte contiene los siguientes capítulos:

Capítulo	Nombre del capítulo	Página
1	Presentación del programa PL7	15
2	Descripción de los objetos en lenguaje PL7	25
3	Memoria del usuario	61
4	Modos de funcionamiento	81
5	Estructura del programa	91

Presentación del programa PL7

1

Presentación

Objeto del capítulo

Este capítulo presenta las características principales del programa PL7.

Contenido:

Este capítulo contiene los siguiente apartados:

Apartado	Página
Presentación de los programas PL7	16
Presentación de los lenguajes PL7	17
Estructura de programa PL7	20
Módulos funcionales	22

Presentación de los programas PL7

Generalidades

El diseño y la puesta en marcha de las aplicaciones para autómatas Micro y Premium se llevan a cabo con la ayuda de programas PL7.

Se proponen 3 tipos de programas PL7:
- PL7 Micro
- PL7 Junior
- PL7 Pro

Programas PL7

La tabla siguiente muestra las diferencias entre los 3 tipos de programas.

Servicios		**PL7 Micro**	**PL7 Junior**	**PL7 Pro**
Programación/Depuración/ Explotación		M	M/P	M/P
Bloques de función del usuario	Creación	-	-	P
	Uso	-	P	P
Pantallas de explotación	Creación	-	-	M/P
	Uso	-	M/P	M/P
Módulos funcionales		-	-	P
Bloque de función DFB para el diagnóstico		-	-	P

Leyenda:
M = autómatas Micro
P = autómatas Premium
- = no disponible

Convenciones de escritura

Continuación del documento:
- la notación PL7 o programa PL7 se utiliza para designar indistintamente los 3 tipos de programas PL7 Micro, PL7 Junior y PL7 Pro.
- la notación Premium se utiliza para designar indistintamente los procesadores TSX 57, PMX 57 y PCX 57.

Presentación de los lenguajes PL7

Generalidades

El programa PL7 propone 4 lenguajes de programación:
- lenguaje de contactos
- lista de instrucciones
- literal estructurado
- Grafcet

La tabla siguiente ofrece el posible uso de los lenguajes en función de los tipos de autómatas.

Lenguaje	Autómatas Micro	Autómatas Premium
Lenguaje de contactos	X	X
Lista de instrucciones	X	X
Literal estructurado	X	X
Grafcet	X (excepto las macro etapas)	X

Leyenda:
X = disponible
- = no disponible

Estos lenguajes pueden mezclarse en el seno de un misma aplicación. Una sección de programa puede escribirse en lenguaje de contactos, otra en literal...

Estos lenguajes ponen en marcha:
- los bloques de función predefinidos (Temporizaciones, Contadores...),
- las funciones tareas (analógica, comunicación, contaje...),
- las funciones específicas (gestión del tiempo, cadena de caracteres...).

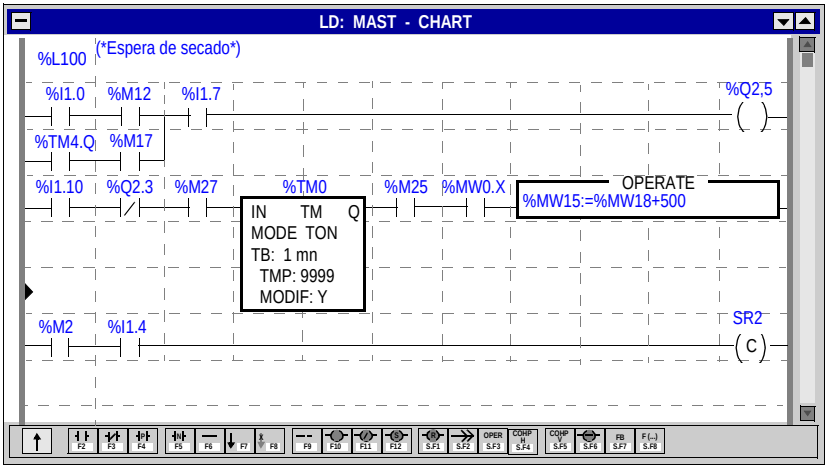
Los objetos del lenguaje se pueden simbolizar con la ayuda del editor de variables o en línea en los editores de programa.

El programa PL7 satisface la norma IEC 1131-3 (véase (Ver Manual di Referencia, tomo 2 y 3)).

Lenguaje de contactos

El lenguaje de contactos (LD) es un lenguaje gráfico. Permite la transcripción de esquemas de relés, se adapta al tratamiento combinatorio.

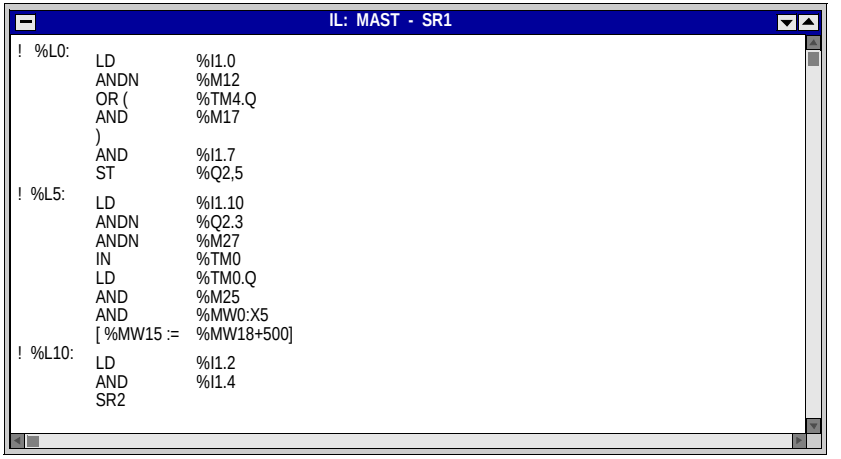
Ofrece los símbolos gráficos básicos: contactos, bobinas, bloques.
La escritura de cálculo numérico se puede realizar en el interior de los bloques de operaciones.
Ejemplo de red de contactos



Lenguaje lista de instrucciones

El lenguaje lista de instrucciones (IL) es un lenguaje "máquina" booleano que permite escribir tratamientos lógicos y numéricos.

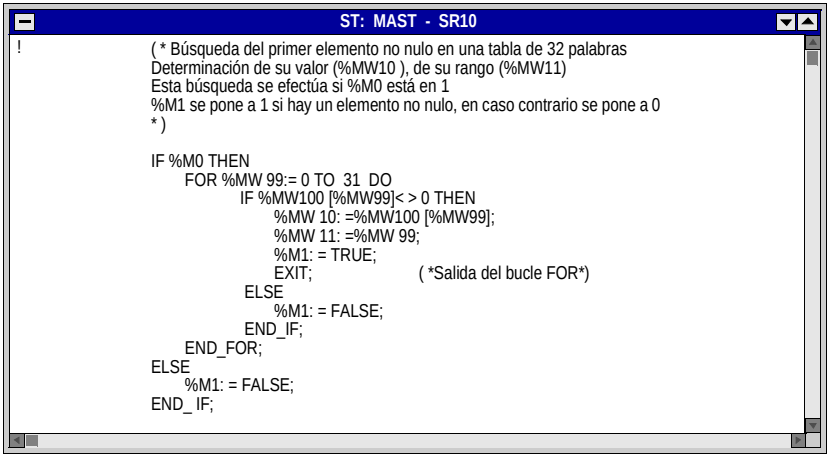
Ejemplo de programa en lenguaje lista de instrucciones



Lenguaje literal estructurado

El lenguaje literal estructurado (ST) es un lenguaje del tipo "informático" que permite la escritura estructurada de tratamientos lógicos y numéricos.

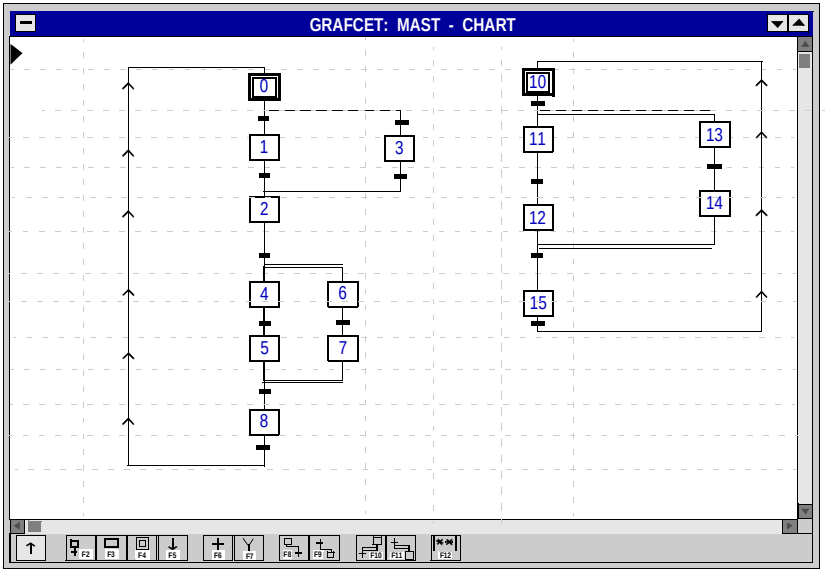
Ejemplo de programa en lenguaje literal estructurado



Lenguaje Grafcet

El lenguaje Grafcet permite representar gráficamente y de forma estructurada el funcionamiento de un automatismo secuencial.

Ejemplo de programa en lenguaje Grafcet.



Estructura de programa PL7

Generalidades

El programa PL7 propone dos tipos de estructura:

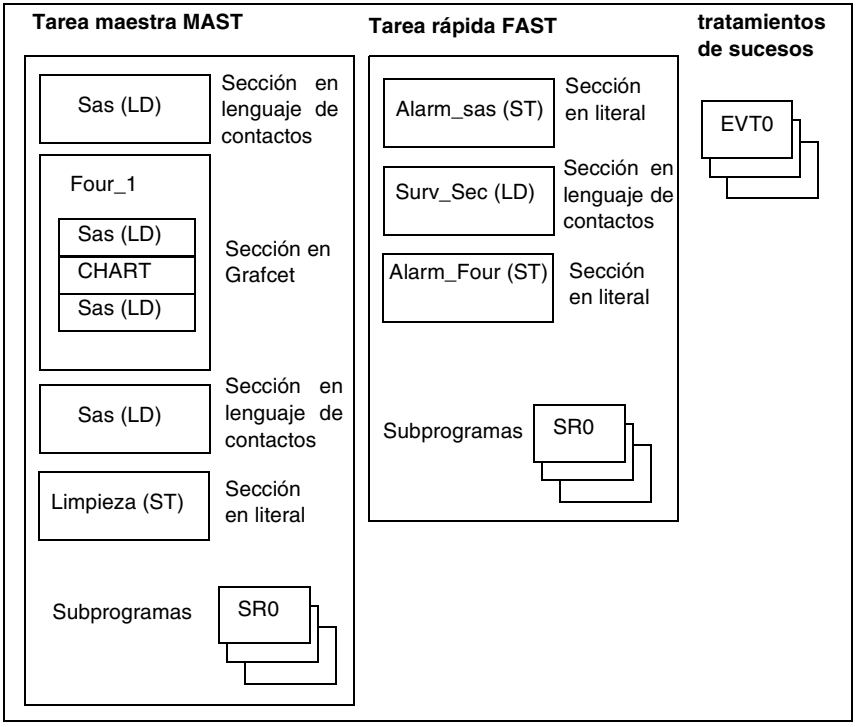
- **Monotarea:** es la estructura simplificada predeterminada, o se ejecuta una sola tarea maestra compuesta por un programa, constituida por varias secciones y subprogramas.
- **Multitarea:** esta estructura, mejor adaptada para las aplicaciones en tiempo real de altas prestaciones, está compuesta por un tarea maestra, una tarea rápida y tratamientos secuenciales prioritarios.

Principio

Las tareas maestras y rápidas de un programa PL7 están constituidas por varias partes llamadas secciones y por subprogramas.

Cada una de estas secciones puede programarse en el lenguaje adecuado al tratamiento que se vaya a realizar.

La ilustración siguiente muestra un ejemplo de corte de un programa PL7.



Este desglose en secciones permite crear un programa estructurado y generar o incorporar fácilmente los módulos del programa.

Los subprogramas pueden llamarse desde cualquier sección de la tarea a la que están vinculados o desde otros subprogramas de la misma tarea.

Módulos funcionales

Generalidades

El programa PL7 Pro permite estructurar una aplicación para el autómata Premium en módulos funcionales.

Un módulo funcional es un reagrupación de elementos de programa destinados a realizar una función de automatismo.

Independientemente de la estructura multitarea de los autómatas, se puede definir una estructura de árbol de directorios multiniveles de la aplicación del automatismo.

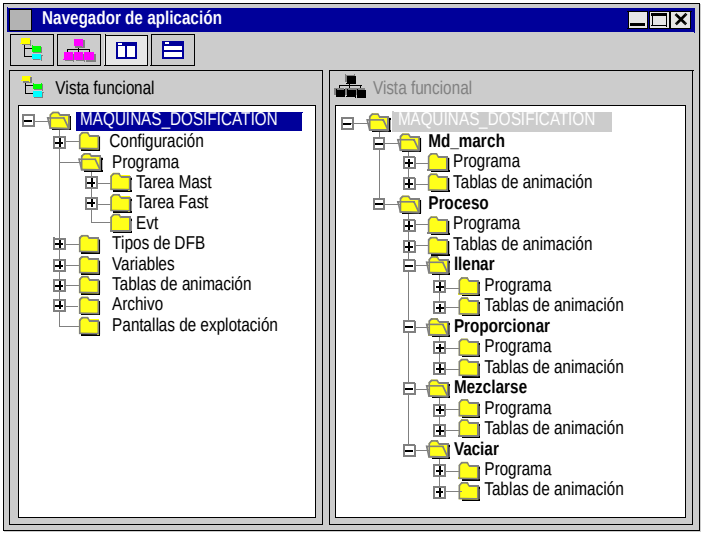
A cada nivel, se pueden vincular las secciones del programa escritas en lenguaje de contactos, literal, lista de instrucciones o Grafcet, así como las tablas de animación y las pantallas de explotación.

Vista funcional

La vista funcional en módulos permite tener un recorte por funciones coherentes delante del procedimiento que se deba dirigir.

La visualización estructural ofrece una vista del orden de ejecución de las secciones del programa del autómata.

La siguiente ilustración muestra las 2 vistas posibles de una aplicación.



Servicios asociados a la vista funcional

Los servicios de trabajo están disponibles en cualquiera de las dos vistas. Particularmente, para un solo comando, se puede forzar la ejecución o no de un módulo funcional.

En este caso, todas las secciones enlazadas al módulo funcional quedan automáticamente forzadas.

Exportación/ importación de los módulos funcionales

La estructura arborescente de directorios en módulos funcionales se puede exportar total o parcialmente.
En este caso, se exporta el conjunto de secciones del programa de diferentes niveles de módulos.

Descripción de los objetos en lenguaje PL7

2

Presentación

Objeto del capítulo

Este capítulo describe los objetos de los lenguajes PL7. Estos objetos se emplean como operandos en las instrucciones.

Contenido:

Este capítulo contiene los siguiente apartados:

Apartado	Página
Definición de los principales objetos booleanos	26
Definición de los principales objetos palabras	27
Direccionamiento de los objetos bits	29
Direccionamiento de los objetos de módulos de entradas/salidas del TSX 37	31
Direccionamiento de los objetos de módulos de entradas/salidas en rack	34
Direccionamiento de los objetos de lenguaje de módulos remotos en el bus FIPIO	37
Direccionamiento de objetos de lenguaje referentes al bus AS-i	40
Direccionamiento de los objetos palabras	42
Regla de solapamientos	46
Objetos del bloque de función	47
Objetos PL7 del tipo tabla	50
Objetos indexados	53
Objetos Grafcet	56
Simbolización	57
Objetos presimbolizados	59

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

Definición de los principales objetos palabras

Descripción La tabla siguiente describe los principales objetos palabras.

Palabras	Descripción	Ejemplos	Acceso por la escritura
Valores inmediatos	Son los valores algebraicos de formato homogéneo junto con los de las palabras estándar y de longitud doble (16 ó 32 bits), que permiten asignar valores a estas palabras.	2542	_
Entradas/ salidas	Son las "imágenes lógicas" de los valores eléctricos de entradas/salidas (ejemplo: entradas/salidas analógicas). Se almacenan en la memoria de datos y se actualizan en cada exploración de la tarea en la que están configuradas.	%IW23.5 %QW51.1	no sí
Internas	Se destinan a almacenar los valores actuales del programa. Se ordenan en el interior del espacio de Datos en una misma zona de memoria.	%MW10 %MD45	sí sí
Constantes	Almacenan las constantes o los mensajes alfanuméricos. El contenido sólo puede escribirse o modificarse mediante el terminal. Se almacenan en la misma zona que el programa, por lo tanto, pueden disponer de la ayuda de la memoria FLASH EPROM.	%KW30	si (sólo a través del terminal)
Sistema	Estas palabras garantizan varias funciones: ● algunas informan del estado del sistema (el tiempo de funcionamiento del sistema y aplicación...). ● otras permiten actuar sobre la aplicación (modos de funcionamiento...).	%SW5	según i
Bloques de función	Estas palabras corresponden a los parámetros o valores actuales de los bloques de función estándares o instancia de DFB.	%TM2.P	sí
Comunes	Se destinan a intercambiarse automáticamente en todas las estaciones conectadas a la red de comunicación.	%NW2.3	sí
Grafcet	Las palabras Grafcet permiten conocer el tiempo de actividad de las etapas.	%X5,T	sí

Formato de valores

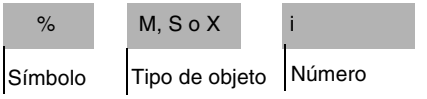
Los valores de las palabras pueden codificarse en los formatos siguientes:

Tipo	Tamaño	Ejemplo de valor	Límite inferior	Límite superior
Entero base 10	Longitud estándar	1506	-32768	+32767
	Longitud doble	578963	-2 147 483 648	2 147 483 647
Entero base 2	Longitud estándar	2#1000111011111011011	2#10...0	2#01...1
	Longitud doble	2#10001110111110110111111111011111011111	2#10...0	2#01...1
Entero base 16	Longitud estándar	16#AB20	16#0000	16#FFFF
	Longitud doble	16#5AC10	16#000000000	16#FFFFFFFF
Flotante		-1.32E12	-3.402824E+38 (1) 1.175494E-38 (1)	-1.175494E-38 (1) 3.402824E+38 (1)
Leyenda				
(1)	límites excluidos			

Direccionamiento de los objetos bits

Presentación

El direccionamiento de los bits internos, del sistema y de las etapas sigue las reglas siguientes:



Sintaxis

La tabla que viene a continuación describe los diferentes elementos que constituyen el direccionamiento.

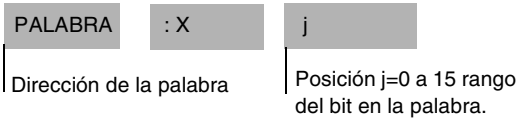
Familia	Elemento	Valores	Descripción
Símbolo	%	-	-
Tipo de objeto	M	-	**Bits internos** destinados a almacenar los estados intermedios actuales del programa. Se ordenan en el interior del espacio de datos en una misma zona de memoria.
	S	-	**Bits del sistema** (Ver Manual di Referencia, tomo 2), estos bits aseguran varias funciones: ● algunos dan a conocer el estado del sistema mediante la lectura de los bits %Si (rebasamiento del watchdog,...). ● otros permiten actuar sobre la aplicación (inicialización Grafcet, ...).
	X	-	**Bits de etapa**, los bits de etapas (Véase *Objetos Grafcet, p. 56*) proporcionan el estado de actividad de las etapas
Número	i	-	El valor máximo del número depende de la cantidad de objetos configurada.

Ejemplos:

- %M25 = bit interno número 25
- %S20 = bit del sistema número 20
- %X6 = bit de etapa número 6

Bits que se extraen de las palabras

El programa PL7 permite extraer uno de los 16 bits de las palabras de longitud estándar. La identificación de la palabra se completa entonces con el rango del bit extraído según la sintaxis que se muestra a continuación:



Ejemplos:

- %MW10:X4 = bit número 4 de la palabra interna %MW10
- %QW5.1:X10 = bit número 10 de la palabra de salida %QW5.1

Nota: La extracción de bits de las palabras también se puede llevar a cabo en las palabras indexadas.

Direccionamiento de los objetos de módulos de entradas/salidas del TSX 37

Presentación

El direccionamiento de los principales objetos bit y palabra de los módulos de entradas/salidas es del tipo geográfico. Es decir depende:

- del número (dirección) del rack,
- de la posición física del módulo en la caja,
- del número de la vía del módulo.

Ilustración

El direccionamiento se define de la siguiente manera:

Sintaxis La siguiente tabla describe los diferentes elementos que constituyen el direccionamiento.

Familia	Elemento	Valores	Descripción
Símbolo	%	-	-
Tipo de objeto	I Q	- -	Imagen de la entrada física del módulo, Imagen de la salida física del módulo, Estas informaciones se intercambian de forma implícita en cada ciclo de la tarea a la que están vinculadas.
	M	-	Variable interna Estas informaciones de lectura o de escritura se intercambian a pedido de la aplicación.
	K	-	Constante interna Estas informaciones de configuración únicamente están disponibles para lectura.
Formato (tamaño)	X	-	Booleano Para los objetos del tipo booleano, la X puede omitirse.
	W	16 bits	Longitud estándar
	D	32 bits	Longitud doble
	F	32 bits	Flotante. El formato flotante utilizado es el de la norma IEEE Std 754-1985 (equivalente a IEC 559).
Posición del módulo	x	0 a 8 0 a 10	TSX 37-10 TSX 37-21/22 **Nota:** un módulo de formato estándar (que ocupe 2 posiciones) se direcciona como 2 módulos de 1/2 formato superpuestos, (a continuación, véanse las explicaciones).
N° de vía	i	0 a 31 o MOD	Número de vía del módulo. MOD: vía reservada a la gestión del módulo y a los parámetros comunes a todas las vías.
Rango	r	0 a 127 o ERR	Posición del bit dentro de la palabra. ERR: indica un fallo de módulo o de vía.

Ejemplos La tabla siguiente presenta algunos ejemplos de direccionamiento de objetos.

Objeto	Descripción
%I1.5	Vía de entrada número 5 del módulo de entradas/salidas situado en la posición 1.
%MW2.0.3	Palabra de estado de rango 3 de la vía 0 del módulo de entradas/salidas situado en la posición 2.
%I5.MOD.ERR	Información de fallo del módulo de entradas/salidas situado en la posición 5.

En el caso de los módulos en formato estándar

Se direccionan como 2 módulos de formato superpuestos.

Por ejemplo, un módulo de 64 E/S que ocupa las posiciones 5 y 6, se contempla como 2 módulos de formato:

- un módulo de 32 entradas situado en la posición 5,
- un módulo de 32 entradas situado en la posición 6,

La tabla que se presenta a continuación describe la codificación Posición/Número de vía en función del módulo.

Módulo	**1/2 formato**			**Formato estándar**			
	4S	**8E**	**12E**	**28E/S**	**32E**	**32S**	**64E/S**
Número de vía	0 a 3	0 a 7	0 a 11	0 a 15 (E)	0 a 15 (E)	0 a 15 (S)	0 a 31 (E)
				0 a 11 (S)	0 a 15 (E)	0 a 15 (S)	0 a 31 (S)
Direccionamiento: Posición/Número de vía (x=posición)	x.0 a x.3	x.0 a x.7	x.0 a x.11	x.0 a x.15	x.0 a x.15	x.0 a x.15	x.0 a x.31
				(x+1).0 a (x+1).11	(x+1).0 a (x+1).15	(x+1).0 a (x+1).15	(x+1).0 a (x+1).31

Ejemplos

La tabla siguiente presenta dos ejemplos de direccionamiento de objetos de un módulo estándar 28E/S que ocupa las posiciones 3 y 4.

Objeto	**Descripción**
%I3.6	Vía de entrada número 6 del módulo.
%Q4.2	Vía de salida número 2 del módulo.

Direccionamiento de los objetos de módulos de entradas/salidas en rack

Presentación

El direccionamiento de los objetos principales de bit y palabra de los módulos de entradas/salidas es de tipo geográfico. Esto significa que depende:

- del número (dirección) del rack,
- de la posición física del módulo en el rack,
- del número de la vía del módulo.

Ilustración

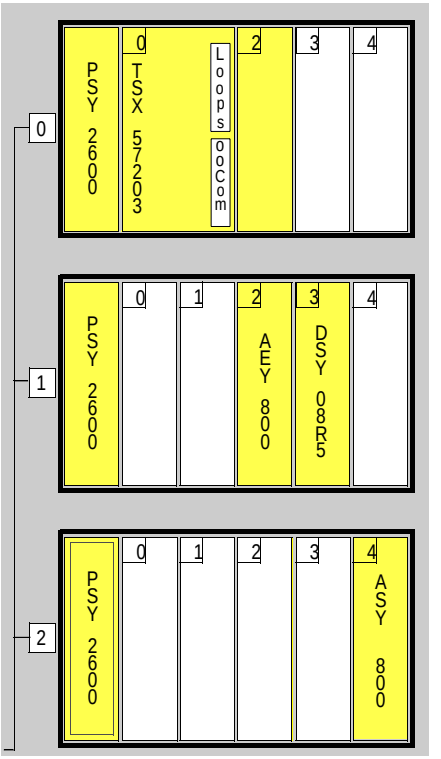
El direccionamiento se define como sigue:

%	I, Q, M, K	X, W, D, F	X	Y	□	i	□	r
Simbolo	Typo de objecto	Formato	Bastidor	Posición		N° via		Rango

Sintaxis En la siguiente tabla se describen los distintos elementos que componen el direccionamiento.

Familia	Elemento	Valores	Descripción
Símbolo	%	-	-
Tipo de objeto	I Q	- -	Imagen de la entrada física del módulo, Imagen de la salida física del módulo, Esta información se intercambia de forma automática en cada ciclo de la tarea a la que están asignadas.
	M	-	Variable interna Esta información sobre lectura o escritura se intercambia a petición de la aplicación.
	K	-	Constante interna Esta información de configuración está disponible en lectura únicamente.
Formato (tamaño)	X	-	Booleano Para los objetos de tipo booleano, este elemento se puede omitir.
	W	16 bits	Longitud simple.
	D	32 bits	Longitud doble.
	F	32 bits	Flotante. El formato flotante utilizado es el de la norma IEEE Std 754-1985 (equivalente IEC 559).
Dirección del rack	x	0 ó 1 0 a 7	TSX 5710/102/103/153, PMX 57102, PCX 571012). Otros procesadores.
Posición del módulo	y	00 a 14 (1)	Número de posición en el rack. Cuando el número de rack (x) es distinto de 0, la posición (y) se codifica con 2 dígitos: 00 a 14; por el contrario, si el número de rack (x) = 0, se eliminan los ceros no significativos (eliminación por la izquierda) de "y" ("x" no aparece e "y" tiene 1 dígito para los valores inferiores a 9).
N° de vía	i	0 a 127 o MOD	MOD: vía reservada para la gestión del módulo y los parámetros comunes a todas las vías.
Rango	r	0 a 127 o ERR	Posición del bit en la palabra. ERR: indica un error de módulo o de vía.
(1) : el número máximo de emplazamientos necesita 2 racks en la misma dirección.			

Ejemplos En la siguiente tabla se muestran algunos ejemplos de direccionamiento de objetos.

Objeto	Descripción	Ilustración
%MW2.0.3	Palabra de estado de rango 3 de la vía 0 del módulo de entradas TON situado en la posición 2 del rack 0.	
%MW103.0.3	Palabra de estado de rango 3 de la vía 0 del módulo de salidas TON situado en la posición 3 del rack 1.	
%I102.MOD.ERR	Información de fallo del módulo de entradas analógicas situado en la posición 2 del rack 1.	
%I204.3.ERR	Información de fallo de la vía 3 del módulo de salidas analógicas situado en la posición 4 del rack 2.	

Direccionamiento de los objetos de lenguaje de módulos remotos en el bus FIPIO

Presentación

El direccionamiento de los objetos principales de bit y palabra de los módulos remotos del bus FIPIO es de tipo geográfico. Esto significa que depende:

- del punto de conexión,
- del tipo de módulo (de base o de extensión),
- del número de la vía.

Ilustración

El direccionamiento se define como sigue:

%	I, Q, M, K	X, W, D, F	\	p.2.c	\	m	□	i	□	r
Simbolo	Typo de objecto	Formato		Dirección módulo/via y punto de conexión		N°de módulo		N° via		Rango

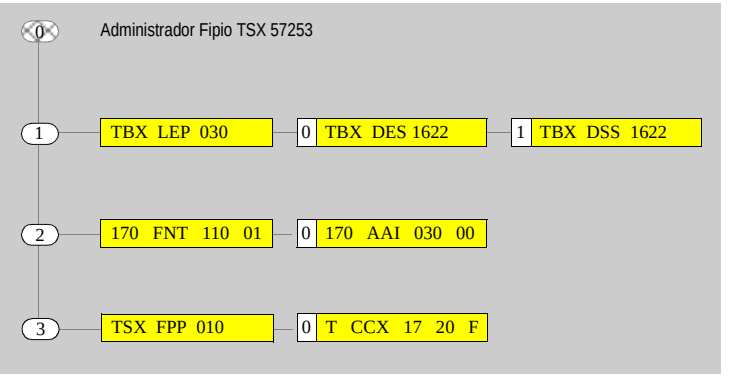
Sintaxis En la siguiente tabla se describen los distintos elementos que componen el direccionamiento.

Familia	Elemento	Valores	Significado
Símbolo	%	-	-
Tipo de objeto	I Q	- -	Imagen de la entrada física del módulo, Imagen de la salida física del módulo, Esta información se intercambia de forma automática en cada ciclo de la tarea a la que están asignadas.
	M	-	Variable interna Esta información sobre lectura o escritura se intercambia a petición de la aplicación.
	K	-	Constante interna Esta información de configuración está disponible en lectura únicamente.
Formato (tamaño)	X	-	Booleano Para los objetos de tipo booleano, la X se puede omitir.
	W	16 bits	Longitud simple.
	D	32 bits	Longitud doble.
	F	32 bits	Flotante. El formato flotante utilizado es el de la norma IEEE Std 754-1985 (equivalente IEC 559).
Dirección de módulo/vía y punto de conexión	p	0 ó 1	Número de posición del procesador en el rack.
	2	-	Número de vía del enlace FIPIO integrado en el procesador.
	c	1 a 127	Número del punto de conexión.
Posición del módulo	m	0 ó 1	0: módulo de base, 1: módulo de extensión.
N° de vía	i	0 a 127 o MOD	MOD: vía reservada para la gestión del módulo y los parámetros comunes a todas las vías.
Rango	r	0 a 255 o ERR	ERR: indica un error de módulo o de vía.

Ejemplos

En la siguiente tabla se muestran algunos ejemplos de direccionamiento de objetos.

Objeto	Significado
%MW\0.2.1\0.5.2	Palabra de estado de rango 2 del bit de imagen de la entrada 5 del módulo de base de entradas remotas situado en el punto de conexión 1 del bus FIPIO.
%I\0.2.1\0.7	Bit de imagen de la entrada 7 del módulo de base de entradas remotas situado en el punto de conexión 1 del bus FIPIO.
%Q\0.2.1\1.2	Bit de imagen de la salida 2 del módulo de extensión de salidas remotas situado en el punto de conexión 1 del bus FIPIO.
%I\0.2.2\0.MOD.ERR	Información de fallo del módulo Momentum situado en el punto de conexión 2 del bus FIPIO.
%Q\1.2.3\0.0.ERR	Información de fallo de la vía 0 del módulo CCX17 situado en el punto de conexión 3 del bus FIPIO.



Direccionamiento de objetos de lenguaje referentes al bus AS-i

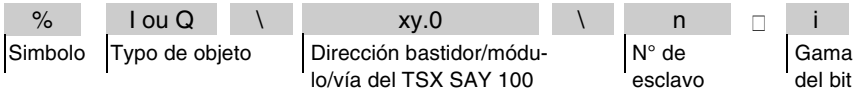
Presentación

El direccionamiento de los principales objetos de bit y de palabra asociados al bus AS-i es de tipo geográfico. Es decir que depende:

- del número (dirección) del rack en el que se ha posicionado el acoplador,
- de la posición física del acoplador en el rack,
- del número (dirección) del equipo esclavo en el bus AS-i.

Ilustración

El direccionamiento se define del modo siguiente:



Sintaxis

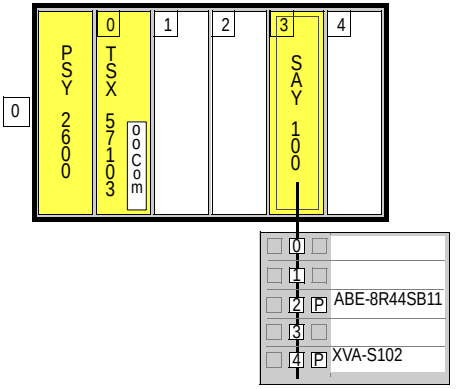
En la tabla siguiente se describen los distintos elementos que constituyen el direccionamiento.

Familia	Elemento	Valores	Descripción
Símbolo	%	-	-
Tipo de objeto	I Q	- -	Imagen de la entrada física del módulo, Imagen de la salida física del módulo, Estas informaciones se intercambian de manera automática en cada ciclo de la tarea a la que están asociadas.
Dirección del rack	x	0 ó 1 0 a 7	TSX 5710/102/103/153, PMX 57102, PCX 571012). Otros procesadores.
Posición del módulo	y	00 a 14 (1)	Número de posición en el rack. Cuando el número de rack (x) es distinto de 0, la posición (y) está codificada mediante 2 dígitos: 00 a 14; en cambio, si el número de rack (x) = 0, se eliminan los ceros no significativos (eliminación por la izquierda) de "y" ("x" no aparece e "y" es 1 dígito para los valores inferiores a 9).
Nº de vía	0	-	El acoplador TSX SAY 100 sólo dispone de una vía.
N° del esclavo	n	0 a 31	Dirección física del esclavo.
Rango	i	0 a 3	Posición del bit de imagen de la entrada o la salida.
(1) : El número máximo de emplazamientos requiere la utilización de un racks de extensión.			

Ejemplo

La tabla siguiente presenta algunos ejemplos de direccionamiento de objetos.

Objeto	Descripción
%I3.0\2.2	Entrada 2 del esclavo 2, el módulo TSX SAY 100 está posicionado en el emplazamiento 3 del rack 0.
%Q3.0\4.3	Salida 3 del esclavo 4, el módulo TSX SAY 100 está posicionado en el emplazamiento 3 del rack 0.



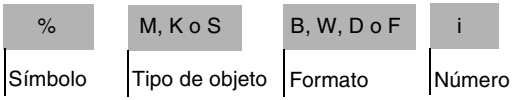
Direccionamiento de los objetos palabras

Presentación

El direccionamiento de palabras (salvo palabras de módulos de entradas/salidas y bloques de función) siguen una misma sintaxis que se describe a continuación.

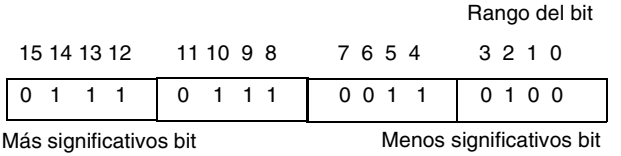
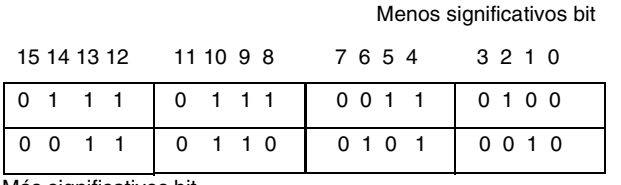
Ilustración

El direccionamiento de palabras internas, constantes y sistema sigue las reglas siguientes:



Sintaxis La tabla que viene a continuación describe los diferentes elementos que constituyen el direccionamiento.

Familia	Elemento	Valores	Descripción
Símbolo	%	-	-
Tipo de objeto	M	-	**Palabras internas** destinadas a almacenar los valores actuales del programa. Se ordenan en el interior del espacio de datos en una misma zona de memoria.
	K	-	**Palabras constantes** almacenan los valores constantes o los mensajes alfanuméricos. El contenido sólo puede escribirse o modificarse mediante el terminal. Se almacenan en la misma zona que el programa, por lo tanto, pueden disponer de la ayuda de la memoria FLASH EPROM.
	S	-	**Palabras de sistema** (Ver Manual di Referencia, tomo 2), estas palabras aseguran varias funciones: ● algunas informan del estado del sistema leyendo las palabras %SWi (el tiempo de funcionamiento del sistema y aplicación, ...). ● otras permiten actuar sobre la aplicación (modos de funcionamiento...).

Familia	Elemento	Valores	Descripción
Formato	B	8 bits	**Byte**, este formato se usa exclusivamente para las operaciones en la cadena de caracteres.
	W	16 bits	**Longitud estándar**: estas palabras de 16 bits pueden contener un valor algebraico comprendido entre –32 768 y 32 767, 
	D	32 bits	**Longitud doble**: estas palabras de 32 bits pueden contener un valor algebraico comprendido entre –2 147 483 648 y 2 147 483 647. Estas palabras se almacenan en la memoria en dos palabras de longitud estándar consecutivas. 
	F	32 bits	**Flotante**: el formato flotante que se usa es el de la norma IEEE Std 754-1985 (equivalente a IEC 559). La longitud de las palabras es de 32 bits, lo que corresponde a los números flotantes de precisión estándar. Ejemplos de valores flotantes: 1285.28 12.8528E2
Número	i	-	El valor máximo del número depende de la cantidad de objetos configurada.

Ejemplos:

- %MW15 = palabra interna de longitud estándar número 15
- %MF20 = palabra interna flotante número 20
- %KD26 = doble palabra constante número 26
- %SW30 = palabra de sistema número 30

Direccionamiento de las palabras en la red

El direccionamiento de palabras en la red se describe en el manual Tarea de comunicación.

Por otro lado, las redes usan los objetos específicos: las palabras comunes. Son objetos de palabras de longitud estándar (16 bits) comunes a todas las estaciones conectadas a la red de comunicaciones.

Direccionamiento: %NW{i.j}k

con: i = 0 a 127 número de red, j = 0 a 31 número de estación y k= 0 a 3 número de palabra

Regla de solapamientos

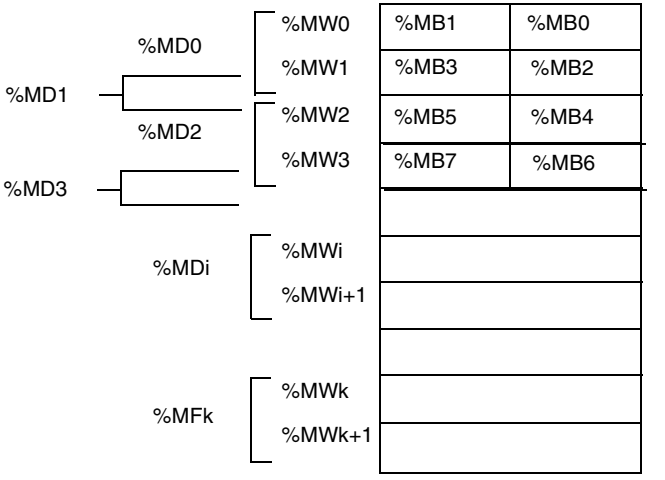
Principios

Los bytes, palabras estándar, doble longitud y flotantes se ordenan en el interior del espacio de datos en una misma zona de memoria.

Así, se produce solapamiento entre:
- la palabra de doble longitud %MDi y las palabras de longitud estándar %Mwi y %MWi+1 (la palabra %MWi contiene las menos significativas y la palabra %MWi+1 las más significativas de la palabra %MDi),
- la palabra de longitud estándar %MWi y los bytes %MBj y %MBj+1 (con j=2 x i),
- la flotante %MFk y las palabras de longitud estándar %MWk et %MWk+1.

Ilustración

Esta ilustración muestra el solapamiento de palabras internas.



Ejemplos

- %MD0 corresponde a %MW0 y %MW1 (véase la ilustración de más arriba).
- %MW3 corresponde a %MB7 y %MB6 (véase la ilustración de más arriba).
- %KD543 corresponde a %KW543 y %KW544.
- %MF10 corresponde a %MW10 y %MW11.

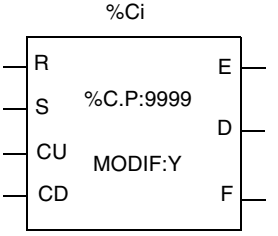
Objetos del bloque de función

Generalidades

Los bloques de función ponen en marcha los objetos bits y las palabras accesibles para el programa.

Ejemplo de bloque de función

La ilustración siguiente presenta un bloque de función contador/descontador.



Bloque contador/descontador

Objetos bits

Corresponden a las salidas de los bloques. Son bits accesibles a las instrucciones booleanas de prueba.

Objetos de palabras

Corresponden:

- a los parámetros de configuración del bloque, estos parámetros pueden ser accesibles (ejemplo: parámetro de preselección) o no (ejemplo: base de tiempo) por programa,
- a los valores actuales (ejemplo: %Ci.V valor de contaje en curso).

Lista de objetos de bloques de función accesibles a través del programa

La tabla que se ofrece a continuación describe el conjunto de los objetos de los bloques de función.

Bloques de funciones	Símbolo	Nº Máx. Micro	Nº Máx. Premium	Tipo de objetos	Descripción	Dirección	Acceso por la escritura
Temporizador	%TMi	64	255 (128 por defecto)	Palabra	Valor actual	%TMi.V	no
					Valor de preselección	%TMi.P	sí
				Bit	Salida de temporizador	%TMi.Q	no
Contador/ Descontador	%Ci	32	255 (64 por defecto)	Palabra	Valor actual	%Ci.V	no
					Valor de preselección	%Ci.P	sí
				Bit	Salida rebasamiento (vacía)	%Ci.E	no
					Salida preselección alcanzada	%Ci.D	no
					Salida rebasamiento (llena)	%Ci.F	no
Monoestable	%MNi	8	255 (32 por defecto)	Palabra	Valor actual	%MNi.V	no
					Valor de preselección	%MNi.P	sí
				Bit	Salida rebasamiento (vacía)	%MNi.R	no
Registro de palabra	%Ri	4	255 (4 por defecto)	Palabra	Acceso al registro	%Ri.I	sí
					Salida del registro	%Ri.O	sí
				Bit	Salida del registro llena	%Ri.F	no
					Salida del registro vacía	%Ri.E	no
Programador cíclico	%DRi	8	255 (8 por defecto)	Palabra	Número de paso en curso	%DRi.S	sí
					Estados del paso j	%DRi.Wj	no
					Tiempo de actividad del paso	%DRi.V	no
				Bit	Último paso definido actual	%DRi.F	no
Temporizador de la serie 7	%Ti	64	255 (0 por defecto)	Palabra	Valor actual	%Ti.V	no
					Valor de preselección	%Ti.P	sí
				Bit	Salida actual	%Ti.R	no
					Salida del temporizador de salida	%Ti.D	no

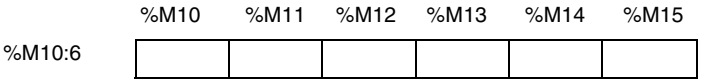
Nota: el número total de temporizadores %TMi+%Ti queda limitado a 64 para un Micro, y a 225 para un Premium.

Objetos PL7 del tipo tabla

Tablas de bits

Las tablas de bits son series de objetos adyacentes del mismo tipo y de longitud definida: L.

Ejemplo de tabla de bits: %M10:6



Esta tabla define los objetos bits que pueden convertirse en tabla de bits.

Tipo	Dirección	Ejemplo	Acceso con escritura
Bits de entradas TOR	%Ix.i:L	%I25.1:8	No
Bits de salidas TOR	%Qx.i:L	%Q34.0:16	Sí
Bits internos	%Mi:L	%M50:20	Sí
Bits Grafcet	%Xi:L, %Xj.i:L	%X50:30	No

Nota: Las longitudes máximas de las tablas dependen de los tipos de objeto
- **Para los bits de entradas/salidas TOR**: el tamaño máximo depende de la distribución modular (número de entradas/salidas del módulo)
- **Para los bits internos o Grafcet**: el tamaño máximo depende de las dimensiones definidas en la configuración.

Tablas de palabras

Las tablas de palabras son series de palabras adyacentes del mismo tipo y de longitud definida: L

Ejemplo de tabla de palabras: %KW10:5

	16 bits
%KW10	
%KW14	

Esta tabla define los objetos palabras que pueden convertirse en tabla de palabras.

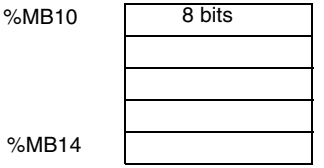
Tipo	**Formato**	**Dirección**	**Ejemplo**	**Acceso con escritura**
Palabras internas	Longitud estándar	%MWi:L	%MW50:20	Sí
	Longitud doble	%MDi:L	%MD30:10	Si
	Flotante	%MFi:L	%MF100:20	Si
Palabras constantes	Longitud estándar	%KWi:L	%KW50:20	No
	Longitud doble	%KDi:L	%KD30:10	No
	Flotante	%KFi:L	%KF100:20	No
Palabras Grafcet	Palabras Grafcet	%Xi.T:L, %Xj.i.T:L	%X12.T:8	No
Palabras del sistema	Palabras del sistema	%SWi:L	%SW50:4	Si

Nota: Las longitudes máximas de las tablas dependen de los tipos de objeto.
- **Para las palabras internas, constantes o Grafcet**: el tamaño máximo depende de las dimensiones definidas en la configuración.
- **Para las palabras del sistema**: sólo las palabras %SW50 a 53 pueden almacenarse en forma de tabla.

Cadenas de caracteres

Las cadenas de caracteres son series de bytes adyacentes del mismo tipo y de longitud definida: L

Ejemplo de cadena de caracteres: %MB10:5



Esta tabla define los objetos que pueden convertirse en cadena de caracteres.

Tipo	**Dirección**	**Ejemplo**	**Acceso con escritura**
Palabras internas	%MBi:L	%MB10:8	Si
Palabras constantes	%KBi:L	%KB20:6	Si

Nota: el índice i debe ser par.

Objetos indexados

Direccionamiento directo

El direccionamiento de los objetos se denomina directo cuando la dirección de dichos objetos es fija y está definida cuando se escribe el programa.

Ejemplo: %MW26 (palabra interna de dirección 26)

Direccionamiento indexado

En el direccionamiento indexado, la dirección directa del objeto se completa con un índice: a la dirección del objeto se le añade el contenido del índice.

El índice se define por:
- una palabra interna %MWi
- una palabra constante %KWi
- un valor inmediato

El número de "palabras índice" no tiene límite.

Este tipo de direccionamiento permite recorrer sucesivamente una serie de objetos de la misma naturaleza (palabras internas, palabras constantes...),: a la dirección del objeto se le añade el contenido del índice.

Ejemplo:
MW108[%MW2]: palabra de dirección directa 108 + contenido de la palabra %MW2. Si la palabra %MW2 tiene un contenido de valor 12, escribir %MW108[%MW2] equivale, por tanto, a escribir %MW120.

Descripción de los objetos indexables

La siguiente tabla define los objetos que pueden indexarse.

Tipo	Formato	Dirección	Ejemplo	Acceso con escritura
Bit de entradas	Booleano	%Ixy.i[índice]	%I21.3[%MW5]	No
Bit de salida	Booleano	%Qxy.i[índice]	%Q32.4[%MW5]	Sí
Bit interno	Booleano	%Mi[índice]	%M10[%MW5]	Si
Bit Grafcet	Booleano	%Mi[índice]	%X20[%MW5]	No
		%Xj.i[índice]	%X2.3[%MW5]	No
Palabras internas	Longitud estándar	%MWi[índice]	%MW30[%MW5]	Si
	Longitud doble	%MDi[índice]	%MD15[%MW5]	Sí
	Flotante	%MFi[índice]	%MF15[%MW5]	Sí
Palabras constante	Longitud estándar	%KWi[índice]	%KW50[%MW5]	No
	Longitud doble	%KDi[índice]	%KD50[%MW5]	No
	Flotante	%KFi[índice]	%KF50[%MW5]	No
Palabras Grafcet	Longitud estándar	%Xi .T[índice]	%X20 .T[%MW5]	No
		%Xj.i .T[índice]	%X2.3 .T[%MW5]	No
Tabla de palabras		%MWi[índice]:L	%MW50[%MW5]:10	Sí
		%MDi[índice]:L	%MD40[%MW5]:10	Sí
		%KWi[índice]:L	%KW70[%MW5]:20	No
		%KDi[índice]:L	%KD80[%MW5]:10	No

Nota: Los valores máximos de los índices dependen de los tipos de objetos indexados.

- **Para los bits de entradas/salidas TON**: 0<i+índice<m (siendo m el número máximo de entradas/salidas del módulo).
- **Para el resto de objetos** (excepto el objeto de doble longitud o el flotante): 0<i+índice<Nmáx, Nmáx = tamaño máximo que depende de las dimensiones definidas en la configuración.
  Para las palabras de doble longitud o las flotantes: 0<i+índice<Nmáx-1.

Indexación de las palabras dobles

La dirección real = dirección de la0 palabra doble indexada + 2 veces el contenido de la palabra índice.

Ejemplo: %MD6[%MW100]
Si %MW100=10, la palabra direccionada será 6 + 2 x 10 -->%MD26.

Rebasamiento del índice

Hay rebasamiento del índice desde que la dirección de un objeto indexado sobrepasa los límites de la zona incluyendo este mismo tipo de objeto, es decir, cuando:

- dirección del objeto + contenido del índice inferior en valor cero,
- dirección del objeto + contenido del índice superior en el límite del valor máximo configurado

En caso de que se rebase el índice, el sistema provoca el cambio al estado 1 del bit del sistema %S20 y la asignación del objeto se efectúa con un valor del índice igual a 0.

La tabla siguiente muestra las condiciones de puesta a 1 y reset del bit del sistema %S20.

Puesto al estado 1	**Reset**
• puesto a 1 por el sistema tras el rebasamiento del índice	• reset por parte del usuario, una vez modificado el índice

Objetos Grafcet

Objetos bits

La siguiente tabla recoge todos los objetos bits Grafcet disponibles y describe su función.

Tipo	Descripción
%Xi	estado de la etapa i del gráfico principal (Chart).
%XMj	estado de la macroetapa j del Grafcet.
%Xj.i	estado de la etapa i, de la macroetapa j del Grafcet.
%Xj.IN	estado de la etapa de entrada de la macroetapa.
%Xj.OUT	estado de la etapa de salida de la macroetapa.

Estos bits pasan a 1 cuando la etapa o la macro etapa es activa, a 0 cuando es inactiva.

Objetos palabras

La tabla siguiente recoge todos los objetos palabras Grafcet disponibles y describe su función.

Tipo	Descripción
%Xi.Ti	tiempo de actividad de la etapa i del Grafcet.
%Xj.i.T	tiempo de actividad de la etapa i de la macroetapa j del Grafcet.
%Xj.IN.T	tiempo de actividad de la etapa i de la macroetapa j que permiten que conozca el estado de la etapa i de la macroetapa j del Grafcet.
%Xj.OUT.T	tiempo de actividad de la etapa de entrada de la macroetapa.
%Xj.OUT	tiempo de actividad de la etapa de salida de la macroetapa.

Estas palabras se incrementan todas en 100 ms y toman un valor entre 0 y 9999.

Simbolización

Función

Los símbolos permiten direccionar los objetos del lenguaje PL7, por los nombres o mnemónicos personalizados.

Sintaxis

Un símbolo es una cadena con un máximo de 32 caracteres alfanuméricos, en la que el primer carácter es alfabético.

Un símbolo empieza por una letra mayúscula y las otras permanecen en minúsculas (por ejemplo: `Bruleur_1`).

En su introducción el símbolo puede escribirse en mayúsculas o en minúsculas (por ejemplo: `BRULEUR_1`), el programa cambia automáticamente el símbolo a su forma correcta.

Caracteres utilizables

La siguiente tabla ofrece los caracteres utilizables en la creación de los símbolos.

Tipo	Descripción
alfabéticos en mayúsculas	"de la A a la Z" y letras siguientes "ÀÁÂÃÄÅÆÇÈÉÊËÌÍÎÏDÑÒÓÔÕÖØÙÚÛÜYp"
alfabéticos en minúsculas	"de la a a la z" y letras acentuadas: àáâãäåæçèéêëìíîïñòóôõöØùúûüypßÿ
digitales	cifras de 0 a 9 (no pueden colocarse al comienzo del símbolo).
el carácter "_"	no puede colocarse ni al principio ni al final del símbolo.

El lenguaje se reserva algunas palabras que no pueden usarse como símbolos, véase (Ver Manual di Referencia, tomo 3).

Edición de símbolos

Los símbolos se definen y se asocian a los objetos del lenguaje a través del editor de variables, a cada uno de los símbolos puede asociarse un comentario de 508 caracteres.

Los símbolos y sus comentarios de almacenan en el disco del terminal y no en el autómata.

Objetos que pueden simbolizarse

Pueden simbolizarse todos los objetos PL7, excepto los objetos estructurados del tipo tabla y objetos indexados, aunque si el objeto de base o el índice está simbolizado, el símbolo se emplea en el objeto estructurado.

Ejemplos:

- si la palabra %MW0 tiene como símbolo "Temperatura", la tabla de palabras %MW0:12 se simboliza con `Temperatura: 12`,
- el símbolo de la palabra %MW10 es `Four_1`, la palabra indexada %MW0[%MW10] se simboliza por `Temperatura[Four_1]`.

Los objetos bits extraídos de las palabras, bits o palabras de bloques de función también pueden simbolizarse, pero si no ocurre esto, pueden heredar el símbolo del objeto de base.

Ejemplos:

- si el símbolo de la palabra %MW0 es `Etat_pompe` y si el bit extraído de la palabra %MW0:X1 no está simbolizado, hereda el símbolo de la palabra, %MW0:X1 tiene como símbolo: `Etat_pompe:X1`,
- si el símbolo del bloque de función %TM0 es `Tempo_four1` y si la salida %TM0.D no está simbolizada, hereda el símbolo del bloque, %TM0.D tiene como símbolo: `Tempo_four.D`.

Objetos únicamente simbólicos

Los parámetros de los bloques de función DFB son accesibles solamente bajo la forma de símbolos. Estos objetos se definen por la siguiente sintaxis:

Nom_DFB.Nom_paramètre

Los elementos tienen el significado y las características siguientes.

Elemento	Nº máximo de caracteres	Descripción
Nom_DFB	32	nombre que se aplica al bloque de función DFB utilizado.
Nom_paramètre	8	nombre que se aplica al parámetro de salidas o a la variable pública.

Ejemplo: `Control.Desviación` para la salida `Desviación` de la instancia DFB denominada `Control`.

Objetos presimbolizados

Función

Algunos módulos de función específica (ejemplo: conteo, comando de ejes…) permiten una simbolización automática de los objetos que están asociados al mismo.

Si se le da el símbolo genérico de la vía %CHxy.i del módulo, todos los símbolos de los objetos asociados a esta vía puede generarse automáticamente a pedido.

Sintaxis

Estos objetos se simbolizan con la siguiente sintaxis:

Prefijo_usuario - Sufijo_constructor

Los elementos tienen el significado y características siguientes:

Elemento	Nº de caracteres máximo	Descripción
Préfixe_utilisateur	12	símbolo genérico que el usuario da a la vía
Suffixe_constructeur	20	parte del símbolo que corresponde al objeto bit o palabra de la vía dada por el sistema

Nota: Además del símbolo, se genera automáticamente un comentario constructor que recuerda escasamente la función del objeto.

Ejemplo

Este ejemplo trata el caso de un módulo de conteo situado en el emplazamiento 3 de la caja del autómata.

Si el símbolo genérico (prefijo-usuario) que se ha atribuído a la vía 0 es `Conteo_piezas`, los siguientes símbolos se generarán automáticamente.

Variable	Tipo	Símbolo	Comentario
%CH3.0	CH		
%ID3.0	DWORD	`Conteo_piezas_cur_meas`	Medida actual del contador
%ID3.0.4	DWORD	`Conteo_piezas_capt`	Valor capturado del contador
%I3.0	EBOOL	`Conteo_piezas_enab_activ`	Validación activa
%I3.0.1	EBOOL	`Conteo_piezas_pres_done`	Preselección efectuada

Memoria del usuario

3

Presentación

Objeto del capítulo

Este capítulo describe la estructura de memoria de los autómatas Micro y Premium.

Contenido:

Este capítulo contiene los siguiente apartados:

Apartado	Página
Estructura de la memoria de los autómatas Micro	62
Estructura de memoria de los autómatas Premium	64
Descripción de la memoria bits	66
Descripción de la memoria de palabras	68
Características de la memoria de los autómatas TSX 37	70
Características de la memoria de los autómatas TSX/PCX 57 10/15/20/25/26/28	73
Características de la memoria de los autómatas TSX/PCX 57 30/35/36	76
Características de la memoria de los autómatas TSX/PCX 57 453/4823	78

Estructura de la memoria de los autómatas Micro

Generalidades

El espacio de memoria de los autómatas Micro que permanece accesible al usuario está dividido en dos conjuntos bien diferenciados:
- memoria de bits
- memoria de palabras

Memoria de bits

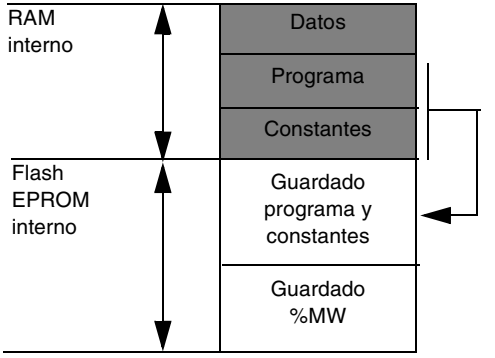
La memoria de bits está ubicada en la memoria RAM integrada en el módulo del procesador. Contiene la imagen de los 1.280 objetos bits.

Función de la memoria de palabras

La memoria de palabras (16 bits) soporta:
- **los datos**: los datos dinámicos de la aplicación y los datos del sistema,
- **el programa**: los descriptores y el código ejecutable de las tareas,
- **las constantes**: las palabras constantes, los valores iniciales y la configuración de las entradas/salidas.

Estructura sin la tarjeta de memoria de extensión

La memoria RAM interna del módulo procesador soporta los datos, el programa y las constantes.
El esquema que sigue describe la estructura de la memoria.

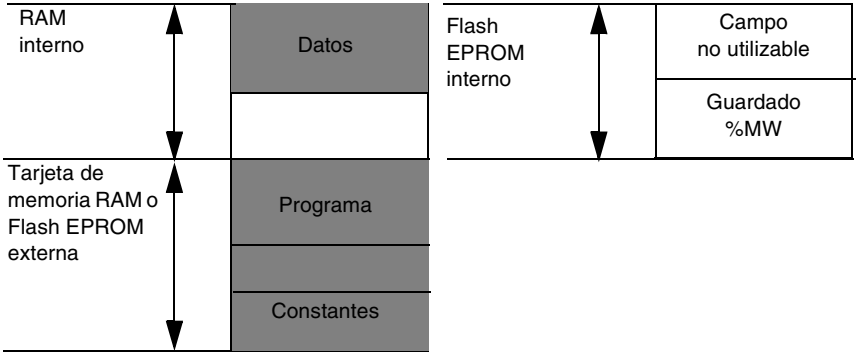


La memoria Flash EPROM integrada en el módulo procesador puede utilizarse para guardar:
- el programa de la aplicación (9 ó 15 Kpal. en función del procesador),
- de 1000 palabras internas %MWi.

Estructura con la tarjeta de memoria de extensión

La memoria RAM interna del módulo procesador soporta los datos.
La tarjeta de memoria de extensión es la encargada de soportar los programas y las constantes.

El esquema que sigue describe la estructura de la memoria.



Para guardar 1000 palabras internas %MWi puede utilizarse una memoria de 10/16 Kpal. Flash EPROM (según el procesador) integrada en el módulo procesador.

Guardado de la memoria

Las memorias RAM pueden alimentarse por pilas de Cadmio-níquel:
- que soporta el módulo procesador para la memoria bit y la RAM interna,
- insertada en una tarjeta para la tarjeta de memoria RAM.

La copia de la aplicación en la memoria FLASH EPROM interna requiere que el autómata no posea tarjeta PCMCIA y que, además, el tamaño de la aplicación sea inferior o igual a 9/15 Kpal. (dependiendo del procesador).

La transferencia de la aplicación desde la memoria FLASH EPROM interna hacia la memoria RAM se realiza automáticamente cuando existe una pérdida de la aplicación en modo RAM (fallo en el guardado o ausencia de batería).
Asimismo, se puede solicitar una transferencia manual a través de una terminal de programación.

Estructura de memoria de los autómatas Premium

Generalidades

En el espacio de memoria de los autómatas Premium, sólo interviene un conjunto. La memoria bits está integrada en la memoria de palabras (en la zona de datos), y está limitada a 4096 bits.

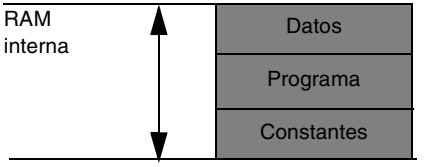
Función de la memoria palabras

la memoria de palabras (16 bits) soporta:

- **los datos**: datos dinámicos de la aplicación y los datos del sistema (éste reserva una zona de memoria RAM de 5 Kpalabras como mínimo)
- **el programa**: descriptores y código ejecutable de las tareas,
- **las constantes**: palabras constantes, valores iniciales y configuración de las entradas/salidas.

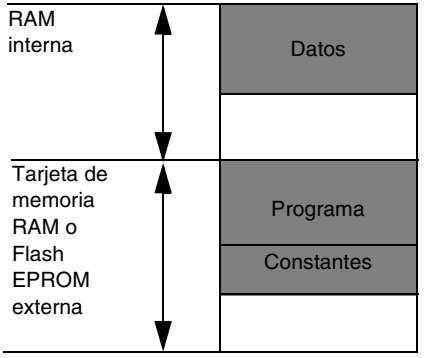
Estructura sin tarjeta de memoria extendida

Los programas, datos y constantes los soporta la memoria RAM interna en el módulo del procesador.
El siguiente esquema describe la estructura de la memoria.



Estructura con tarjeta de memoria extendida

Los datos se almacenan en la memoria RAM interna del módulo del procesador. Los programas y constantes se almacenan en la tarjeta de memoria extendida.
El siguiente esquema describe la estructura de la memoria.



Guardado de la memoria

La memoria bit y RAM interna se protegen con una pila de cadmio/níquel que contiene el módulo del procesador.
La tarjeta de memoria RAM interna se protege con una pila de cadmio/níquel.

Descripción de la memoria bits

Generalidades

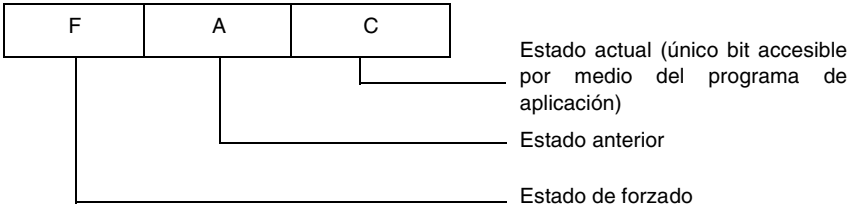
Para los autómatas Micro: esta memoria contiene 1280 objetos bits sea cual sea el tipo de autómata.

Para los autómatas Premium: esta memoria bits no existe y el contenido se encuentra en la memoria palabras en la zona de datos de la aplicación.

La codificación de los objetos bits PL7 permite la prueba de flanco ascendente o descendente sobre:
- los bits de entradas/salidas,
- los bits internos.

Funcionamiento

Cada objeto bit contenido en la memoria bits queda almacenado con la ayuda de 3 bits asignados del siguiente modo:



En la actualización de la memoria bits, el sistema asegura:

Fase	Descripción
1	La transferencia de la imagen del estado actual al estado anterior.
2	La reactualización del estado actual por el programa, el sistema o el terminal (en el caso de forzado de un bit).

Estado de forzado

En cuanto a un pedido de forzado por parte del terminal:
- el estado de forzado F pasa al estado 1
- el estado actual C pasa al estado:
  - 1 si se requiere el forzado a 1
  - 0 si se requiere el forzado a 0

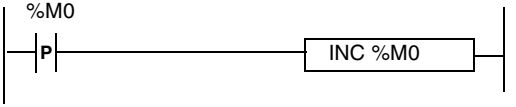
Estos estados ya no cambian hasta que:
- se suprima el forzado y el bit involucrado se actualice,
- se requiera el forzado inverso, solamente se modifica el estado actual.

Consejos de utilización de los flancos ascendentes o descendentes

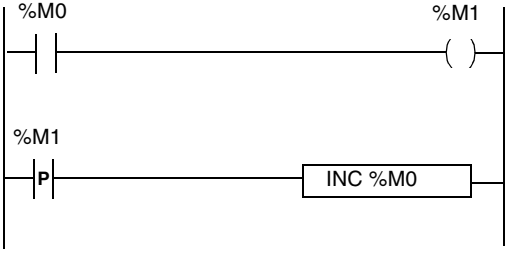
Las instrucciones contactos de flancos ascendentes o descendentes no funcionan correctamente si no se cumplen las reglas que siguen:

- en todos los casos, tratar para un mismo objeto:
  - el bit de entrada: se cambia el contacto de flanco en la tarea o el módulo de entrada,
  - bit de salida o interno: tratar la lectura y la escritura en el interior de una misma tarea.
- Cualquier objeto de bit probado en flanco debe escribirse una vez y sólo una utilizando las bobinas normal -( )- o inversa -(/)- (y/o equivalente en el lenguaje de lista de instrucciones). No se deben utilizar las bobinas -(S)- o -(R)-. Cuando una salida se declara en una lista de intercambio de un tratamiento de sucesos, provoca el intercambio del grupo de vías que tiene asociado, lo que perturba la gestión de flancos en la tarea que gestiona normalmente este grupo de vías.
- no llevar a cabo ni SET ni RESET en un objeto en el cual se pruebe el flanco, puesto que, incluso si el resultado de la ecuación condicional SET/RESET es igual a 0, la acción SET/RESET no se ejecuta, pero el registro del objeto se actualiza (pérdida del flanco).
- no se probará el flanco de entradas/salidas utilizado en una tarea de sucesos, en una tarea maestra o en una rápida
- para los bits internos: la detección de un flanco es independiente del ciclo de la tarea, se detecta un flanco sobre bit interno %Mi cuando su estado ha cambiado entre 2 lecturas.
  Este flanco permanece detectado mientras que ese bit interno no sea explorado en zona de acción.

Ejemplo: Así, en el ejemplo siguiente, si se activa a 1 el bit %M0 en una tabla de animación, el flanco persiste.



Para que se detecte el flanco sólo una vez, es necesario usar un bit interno intermediario. En este caso el historial de %M1 se actualiza, por lo tanto el flanco sólo está presente una vez.



Descripción de la memoria de palabras

Generalidades

Esta memoria de palabras de 16 bits está estructurada en tres espacios lógicos:
- Datos,
- programa y
- constantes.

El tamaño se define mediante la configuración.

Nota: los símbolos y los comentarios asociados a los objetos no quedan registrados en la memoria del autómata, aunque sí en la aplicación local (disco duro del terminal).

Memoria de datos de la aplicación

La memoria de datos consta de las áreas siguientes:

Tipo de palabras	Descripción
Sistema	Número fijo
Bloques de función	Corresponde a las palabras y entradas/salidas de estos bloques (valores actuales, de ajuste...). El número de cada tipo de bloque de función se determina mediante la configuración
Internos	Tamaño definido por el número declarado en la configuración.
Entradas/salidas	Corresponde a las palabras asociadas a cada módulo. La cantidad depende de los módulos configurados.
Comunes de red	Cuatro palabras por estación del autómata (disponibles sólo si el módulo de comunicación está presente y configurado en intercambio de palabras comunes).

Memoria del programa de aplicación

Esta área comprende el código del programa ejecutable, las informaciones gráficas (red de contactos) y los comentarios del programa.

Nota: Independientemente del tipo de lenguaje utilizado o el tamaño del área de programa, el número de componentes de programación (redes de contacto, frases literales...) es limitado. Antes de alcanzar el límite, aparecerá un mensaje de advertencia en PL7.

Memoria de constantes de aplicación

Esta área incluye los parámetros de los bloques de función y los módulos de entradas/salidas definidos en la configuración y las palabras constantes %KW.

Características de la memoria de los autómatas TSX 37

Tamaño de la memoria de bits

La tabla siguiente describe el reparto de memoria de los objetos bits.

Procesador TSX		**37 05/08/10**	**37 21/22**
Tamaño disponible en el procesador		1280	1280
Tipo de objetos	bits del sistema %Si	128	128
	bits de entradas/salidas %I/Qx.i	(1)	(1)
	bits internos %Mi	256	256
	bits de etapas %Xi	96	128
Leyenda			
(1)	depende de la configuración del equipo declarada (módulos de entradas/ salidas, equipo en el bus AS-i)		

Tamaño de la memoria de palabras

La tabla siguiente describe el reparto de memoria de los objetos palabras

Procesador TSX	**3705/08**	**3710**	**3721**			
Tarjeta de memoria	-	-	-	32 Kpalabras	64 Kpalabras	128 Kpalabras
RAM interna	9 Kpalabras	14 Kpalabras	20 Kpalabras	52 Kpalabras	84 Kpalabras	148 Kpalabras
Datos (%MWi)	0,5 Kpalabras (1)	0,5 Kpalabras (1)	0,5 Kpalabras (1)	17,5 Kpalabras	17,5 Kpalabras	17,5 Kpalabras
Programa 100% booleano						
• Lenguaje LD	1,6 Ki	3,8 Ki	6,6 Ki	13,7 Ki	28,5 Ki	58,2 Ki
• Lenguaje IL	2 Ki	4,9 Ki	8,4 Ki	17,5 Ki	36,3 Ki	74,2 Ki
• Lenguaje ST	1,3 Ki	3,3 Ki	5,6 Ki	11,7 Ki	24,2 Ki	49,5 Ki
Programa 90% booleano						
• Lenguaje LD	1,1 Ki	3,1 Ki	5,4 Ki	11,8 Ki	24,7 Ki	50,1 Ki
• Lenguaje IL	1,4 Ki	3,8 Ki	6,6 Ki	14,3 Ki	30,0 Ki	61 Ki
• Lenguaje ST	1,1 Ki	2,9 Ki	5,1 Ki	11,1 Ki	23,3 Ki	47,3 Ki
Programa 65% booleano						
• Lenguaje LD	0,9 Ki	2,2 Ki	4,0 Ki	9,1 Ki	18,9 Ki	38,4 Ki
• Lenguaje IL	1,0 Ki	2,5 Ki	4,6 Ki	10,3 Ki	21,3 Ki	43,4 Ki
• Lenguaje ST	1,0 Ki	2,5 Ki	4,6 Ki	10,3 Ki	21,3 Ki	43,4 Ki
Constantes (1)	128 palabras	128 palabras	128 palabras	256 palabras	512 palabras	512 palabras
Leyenda						
(1)	Tamaño predeterminado, puede ampliarse en detrimento del tamaño del programa de aplicación.					
Ki	Kinstrucciones (1024 instrucciones)					

La tabla siguiente describe el reparto de memoria de los objetos palabras

Procesador TSX	**37 22**			
Tarjeta de memoria	-	32 Kpalabras	64 Kpalabras	128 Kpalabras
RAM interna	20 Kpalabras	52 Kpalabras	84 Kpalabras	148 Kpalabras
Datos (%MWi)	0,5 Kpalabras (1)	17,5 Kpalabras	17,5 Kpalabras	17,5 Kpalabras

Procesador TSX	**37 22**			
Programa 100% booleano				
• Lenguaje LD	6,3 Ki	13,6 Ki	28,4 Ki	57,6 Ki
• Lenguaje IL	8,1 Ki	17,3 Ki	36,1 Ki	73,5 Ki
• Lenguaje ST	5,4 Ki	11,5 Ki	24,1 Ki	49 Ki
Programa 90% booleano				
• Lenguaje LD	5,2 Ki	11,6 Ki	24,5 Ki	50 Ki
• Lenguaje IL	6,3 Ki	14,2 Ki	29,8 Ki	60,8 Ki
• Lenguaje ST	4,9 Ki	11,0 Ki	23,2 Ki	47,2Ki
Programa 65% booleano				
• Lenguaje LD	3,9 Ki	8,9 Ki	18,8 Ki	38,4 Ki
• Lenguaje IL	4,4 Ki	10,1 Ki	21,2 Ki	43,4 Ki
• Lenguaje ST	4,4 Ki	10,1 Ki	21,2 Ki	43,4 Ki
Constantes (1)	128 palabras	256 palabras	512 palabras	512 palabras
Leyenda				
(1)	Tamaño predeterminado, puede ampliarse en detrimento del tamaño del programa de aplicación.			
Ki	Kinstrucciones (1024 instrucciones)			

Nota: el comando AP/Balance de memoria del programa PL7 permite conocer el reparto de la memoria de la aplicación en la memoria del autómata.

Características de la memoria de los autómatas TSX/PCX 57 10/15/20/25/26/28

Tamaño de la memoria de bits

Esta tabla describe el reparto de memoria de los objetos palabras de los autómatas TSX 57 1••, TSX 57 2••, TSX 57 2•23 y PCX 57 203.

Procesador		**TSX 57 103/153 y PCX 57 203**	**TSX57 203/ 253/2623/2823**
Tipo de objetos	bits del sistema %Si	128	128
	bits de entradas/salidas %I/Qx.i	(1)	(1)
	bits internos %Mi (Nº máx.)	3962	8056
	bits de etapas %Xi (Nº máx.)	1024	1024
Leyenda			
(1)	depende de la configuración del equipo declarada (módulos de entradas/ salidas, equipo en el bus AS-i)		

Tamaño de la memoria de palabras

La tabla siguiente describe el reparto de memoria de los objetos palabras de los autómatas TSX 57 1••, TSX 57 2••, TSX 57 2•23 y PCX 57 203.

Procesador	**TSX 57-103 - TSX 57 153**			**TSX-PCX 57 203/ 2623**	**TSX- 57 253/2823**	**TSX-PCX 57 203/ TSX 57 253/2623/ 2823**	**TSX-PCX 57 203/ TSX 57 253/2623/ 2823**	**TSX-PCX 57 203/ TSX 57 253/2623/ 2823**
Tarjeta de memoria	-	32K	64K	-	-	32K	64K	128K
RAM interna	32K	32K	32K	48K/64K	48K/64K	48K/64K	48K/64K	48K/64K
Datos (%MWi)	0,5 K (1)	26 K	26 K	1K (1)	1K (1)	30,5K	30,5K	30,5K
Programa 100% booleano								
• Lenguaje LD	8,8 Ki	12,3 Ki	26,9 Ki	15,5 Ki	22,8 Ki	12,3 Ki	26,6 Ki	56,5 Ki
• Lenguaje IL	11,2 Ki	15,6 Ki	34,3 Ki	19,7 Ki	29,1 Ki	15,6 Ki	33,9 Ki	71,6 Ki
• Lenguaje ST	7,6 Ki	10,5 Ki	22,9 Ki	13,1 Ki	19,4 Ki	10,4 Ki	22,6 Ki	47,8 Ki
Programa 90% booleano								
• Lenguaje LD	5,2 Ki	8,6 Ki	21,4 Ki	11,0 Ki	17,4 Ki	8,6 Ki	21,1 Ki	46,9 Ki
• Lenguaje IL	6,2 Ki	10,3 Ki	25,6 Ki	13,1 Ki	20,7 Ki	10,3 Ki	25,2 Ki	56,0 Ki
• Lenguaje ST	5,0 Ki	8,3Ki	20,5 Ki	10,5 Ki	16,6 Ki	8,3 Ki	20,2 Ki	44,9 Ki
Programa 65% booleano								
• Lenguaje LD	3,6 Ki	6,7 Ki	16,7 Ki	8,1 Ki	13,1 Ki	6,6 Ki	16,4 Ki	36,6 Ki
• Lenguaje IL	3,7 Ki	6,8 Ki	17,0 Ki	8,3 Ki	13,4 Ki	6,8 Ki	16,8 Ki	37,5 Ki
• Lenguaje ST	4,2 Ki	7,9 Ki	19,7 Ki	9,6 Ki	15,5 Ki	7,8 Ki	19,4 Ki	43,3 Ki
Constantes (1)	128 palabras	256 palabras	512 palabras	256 palabras	256 palaras	256 palabras	512 palabras	512 palabras
Leyenda								
(1)	Tamaño predeterminado, puede ampliarse en detrimento del tamaño del programa de aplicación.							
Ki	Kinstrucciones							
K	Kpalabras							

Nota:
- cuando esta tabla mencione para una característica 2 valores separados por un "/", se asociarán respectivamente a cada tipo de procesador (separados por un "/" en el encabezado de la tabla).
- el comando AP/Balance de memoria del programa PL7 permite conocer el reparto de la memoria de la aplicación en la memoria del autómata.

Características de la memoria de los autómatas TSX/PCX 57 30/35/36

Tamaño de la memoria de bits

Esta tabla describe el reparto de memoria de los objetos palabras de los autómatas TSX 57 3•3, TSX 57 3623 y PCX 57 353.

Procesador TSX/PCX		**57 303/353/3623**
Tipo de objetos	bits del sistema %Si	128
	bits de entradas/salidas %I/Qx.i	(1)
	bits internos %Mi (Nº máx)	16250
	bits de etapas %Xi (Nº máx.)	1024
Leyenda		
(1)	depende de la configuración del equipo declarada (módulos de entradas/ salidas, equipo en el bus AS-i)	

Tamaño de la memoria de palabras

Esta tabla describe el reparto de memoria de los objetos palabras de los autómatas TSX 57 3•3, TSX 57 3623, y PCX 57 353.

Procesador	**TSX 57 303/3623**	**TSX/ PCX57 353**	**TSX 57 303/ 3623 / TSX/ PCX57 353**	**TSX 57 303/ 3623 / TSX/ PCX57 353**	**TSX 57 303/ 3623 / TSX/ PCX57 353**	**TSX 57 303/ 3623 / TSX/ PCX57 353**	**TSX 57 303/ 3623 / TSX/ PCX57 353**
Tarjeta de memoria	-	-	32K	64K	128K	256K	384K
RAM interna	64K/80K	64K/80K	80K/96K	80K/96K	80K/96K	80K/96K	80K/96K
Datos (%MWi)	1K (1)	1K (1)	30,5K	30,5K	30,5K	30,5K	30,5K
Programa 100% booleano							
• Lenguaje LD	28,8 Ki	30,1 Ki	12,3 Ki	26,6 Ki	56,2 Ki	115,3 Ki	150,5 Ki
• Lenguaje IL	36,7 Ki	38,4 Ki	15,6 Ki	33,9 Ki	71,6 Ki	147,1 Ki	150,5 Ki
• Lenguaje ST	24,5 Ki	25,6 Ki	10,4 Ki	22,6 Ki	47,8 Ki	98,0 Ki	148,3 Ki
Programa 90% booleano							
• Lenguaje LD	22,6 Ki	23,8 Ki	8,6 Ki	21,1 Ki	46,9 Ki	98,4 Ki	149,9 Ki
• Lenguaje IL	27,1 Ki	28,4 Ki	10,3 Ki	25,2 Ki	56,0 Ki	117,5 Ki	157,6 Ki
• Lenguaje ST	21,7 Ki	22,7 Ki	8,3 Ki	20,2 Ki	44,9 Ki	94,2 Ki	142,9 Ki
Programa 65% booleano							
• Lenguaje LD	17,4 Ki	18,2 Ki	6,6 Ki	16,4 Ki	36,6 Ki	77,0 Ki	117,4 Ki
• Lenguaje IL	17,8 Ki	18,6 Ki	6,8 Ki	16,8 Ki	37,5 Ki	78,8 Ki	120,1 Ki
• Lenguaje ST	20,5 Ki	21,5 Ki	7,8 Ki	19,4 Ki	43,3 Ki	91,1 Ki	138,8 Ki
Constantes (1)	256 K	256 K	256 K	1024 K	1024 K	1024 K	1024 K
Leyenda							
(1)	Tamaño predeterminado, puede ampliarse en detrimento del tamaño del programa de aplicación.						
Ki	Kinstrucciones						
K	Kpalabras						

Nota:

- cuando esta tabla mencione para una característica 2 valores separados por un "/", se asociarán respectivamente a cada tipo de procesador (separados por un "/" en el encabezado de la tabla).
- el comando AP/Balance de memoria del programa PL7 permite conocer el reparto de la memoria de la aplicación en la memoria del autómata.

Características de la memoria de los autómatas TSX/PCX 57 453/4823

Tamaño de la memoria de bits

Esta tabla describe el reparto de la memoria de los objetos de los autómatas TSX 57 453/4823.

Procesador TSX		**57 453/4823**
Tipo de objetos	bits del sistema %Si	128
	bits de entradas/salidas %I/Qx.i	(1)
	bits internos %Mi (Nº máx.)	32634
	bits de etapas %Xi (Nº máx.)	1024
Leyenda		
(1)	depende de la configuración del equipo declarada (módulos de entradas/salidas, equipo en el bus AS-i)	

Tamaño de la memoria de palabras

La tabla siguiente describe el reparto de la memoria de los objetos de los autómatas TSX 57 453/4823.

Procesador	**TSX 57 453/4823**							
Tarjeta de memoria	-	32K	64K	128K	256K	384K (2)	512K (2)	2*480K (2)
RAM interna	96K	176K	176K	176K	176K	176K	176K	176K
Datos (%MWi)	1K (1)	30,5K	30,5K	30,5K	30,5K	30,5K	30,5K	30,5K
Programa 100% booleano								
• Lenguaje LD	37,5 Ki	12,3 Ki	26,6 Ki	56,2 Ki	115,3 Ki	150,5Ki	150,5 Ki	150,5 Ki
• Lenguaje IL	47,8 Ki	15,6 Ki	33,9 Ki	71,6 Ki	147,1 Ki	150,5 Ki	150,5 Ki	150,5 Ki
• Lenguaje ST	31,9 Ki	10,4 Ki	22,6 Ki	47,8 Ki	98,0 Ki	148,3 Ki	150,7 Ki	150,7 Ki
Programa 90% booleano								
• Lenguaje LD	30,2 Ki	8,6 Ki	21,1 Ki	46,9 Ki	98,4 Ki	149,9 Ki	157,6 Ki	157,6 Ki
• Lenguaje IL	36,0 Ki	10,3 Ki	25,2 Ki	56,0 Ki	117,5 Ki	157,6 Ki	157,6 Ki	157,6 Ki
• Lenguaje ST	28,9 Ki	8,3 Ki	20,2 Ki	44,9 Ki	94,2 Ki	142,9 Ki	157,8 Ki	157,8 Ki
Programa 65% booleano								
• Lenguaje LD	23,2 Ki	6,6 Ki	16,4 Ki	36,6 Ki	77,0 Ki	117,4 Ki	157,8 Ki	157,8 Ki
• Lenguaje IL	23,7 Ki	6,8 Ki	16,8 Ki	37,5 Ki	78,8 Ki	120,1 Ki	161,3 Ki	161,3 Ki
• Lenguaje ST	27,4 Ki	7,8 Ki	19,4 Ki	43,3 Ki	91,1 Ki	138,8 Ki	171,3 Ki	171,3 Ki
Constantes (1)	256 palabras	256 palabras	1024 palabras	1024 palabras	1024 palabras	1024 palabras	1024 palabras	1024 palabras
Leyenda								
(1)	Tamaño predeterminado, puede ampliarse en detrimento del tamaño del programa de aplicación.							
(2)	Se alcanza el número del elemento del programmation.							
Ki	Kinstrucciones							
K	Kpalabras							

Nota:

- cuando esta tabla mencione para una característica 2 valores separados por un "/", se asociarán respectivamente a cada tipo de procesador (separados por un "/" en el encabezado de la tabla).
- el comando AP/Balance de memoria del programa PL7 permite conocer el reparto de la memoria de la aplicación en la memoria del autómata.

Modos de funcionamiento

4

Presentación

Objeto del capítulo

Este capítulo trata del comportamiento del programa del usuario en el Rearranque en caliente y el arranque en frío.

Contenido:

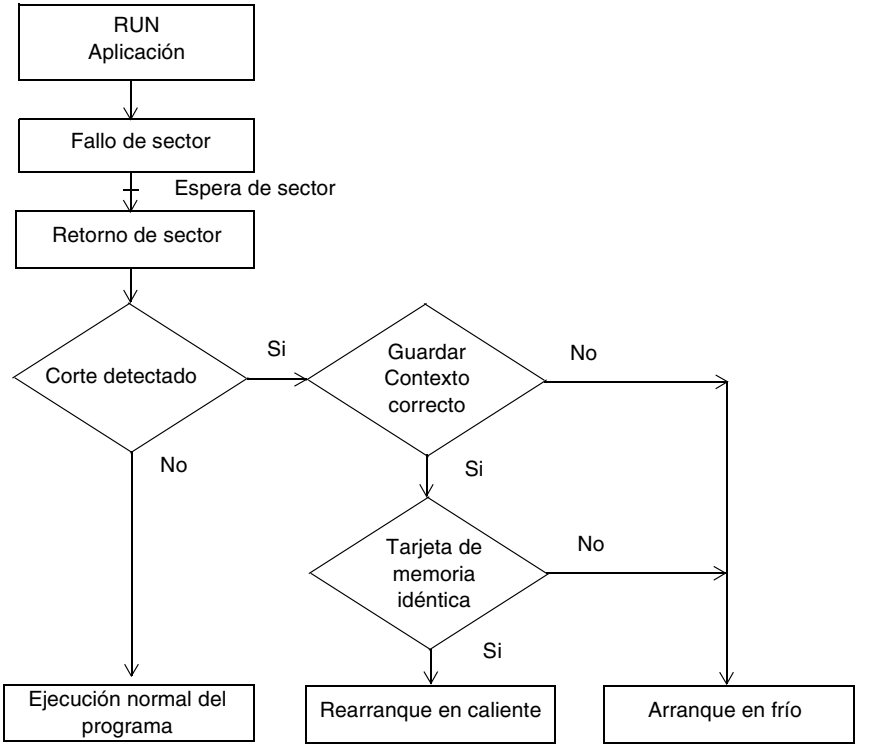
Este capítulo contiene los siguiente apartados:

Apartado	Página
Tratamiento en el corte y rearranque del sector	82
Tratamiento del rearranque en caliente	84
Tratamiento del arranque en frío	86

Tratamiento en el corte y rearranque del sector

Ilustración

La ilustración presenta los diferentes rearranques sectoriales detectadas por el sistema. Si la duración del corte es inferior al tiempo de filtrado de la alimentación (alrededor de 10 ms para las alimentaciones alternativas o de 1 ms para las alimentaciones continuas), el programa no puede verla, con lo cual se ejecutará con normalidad.



Funcionamiento

La tabla que se presenta a continuación describe las fases del tratamiento de los cortes sectoriales.

Fase	Descripción
1	En el momento del corte del sector, el sistema almacena el contexto de la aplicación y la hora del corte.
2	Sitúa todas las salidas en estado de reactivación (estado definido en la configuración).
3	En la reanudación del sector, el contexto guardado se compara al actual; lo que define el tipo de arranque que debe ejecutarse: ● si el contexto de la aplicación ha cambiado (pérdida de contexto del sistema o una nueva aplicación), el autómata efectúa una inicialización de la aplicación: arranque en frío, ● si el contexto de la aplicación es idéntico, el autómata efectúa un rearranque sin inicialización de los datos: rearranque en caliente

Corte de la alimentación en un rack, que no sea el rack 0

Todas las vías de ese rack quedan detectadas como error en el procesador, pero los otros racks no se alteran, los valores de las entradas durante el error no se actualizan en la memoria de la aplicación y se ponen a 0 en el caso de un módulo de entrada TON, a menos que hayan sido forzadas, en tal caso, se mantienen en el valor de forzado.

Si la duración del corte es inferior a 10 ms para las alimentaciones alternativas o a 1 ms para las alimentaciones continuas, el programa no puede detectarlo, con lo cual se ejecutará con normalidad.

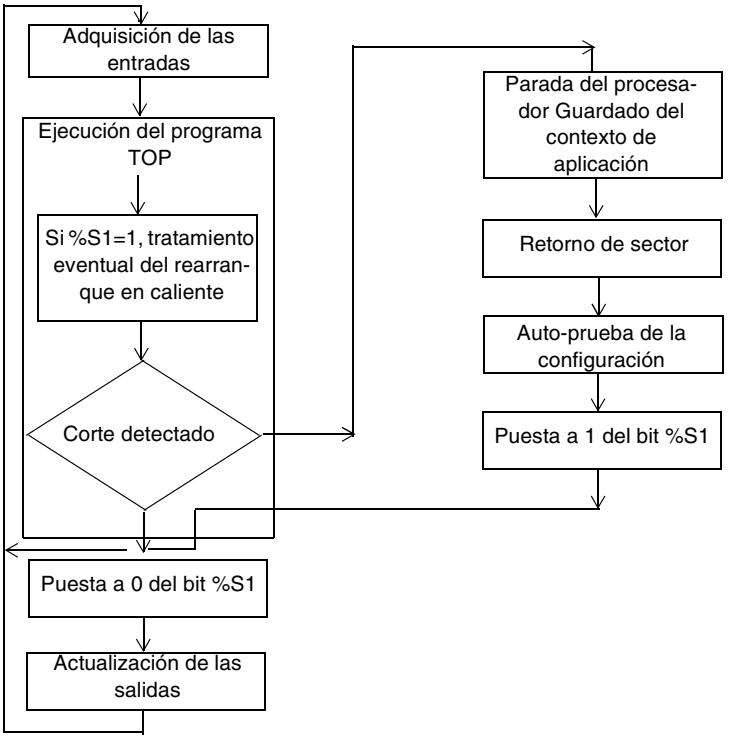
Tratamiento del rearranque en caliente

Causa de un rearranque en caliente

Un rearranque en caliente puede haber sido provocado:
- por una reanudación de sector sin pérdida de contexto,
- por una puesta a 1 por parte del programa del bit del sistema %S1,
- desde PL7 a través del terminal,
- por la acción sobre el botón RESET del módulo de alimentación del rack 0 (excepto en la estación equipada con un procesador PCX 57).

Ilustración

El esquema siguiente describe el funcionamiento de un rearranque en caliente.



Funcionamiento

La tabla que se presenta a continuación describe las fases de rearranque de la ejecución del programa en el rearranque en caliente.

Fase	Descripción
1	La ejecución del programa se reanuda a partir del elemento en el cual ha tenido lugar el corte del sector, sin actualización de las salidas.
2	Cuando termina el ciclo de rearranque, el sistema efectúa: ● la inicialización de las filas de mensajes y de sucesos, ● el envío de parámetros de configuración a todos los módulos de entradas/salidas TON y funciones, ● la desactivación de la tarea rápida y de los tratamientos de sucesos (hasta que termine el primer ciclo de la tarea maestra).
3	El sistema lleva a cabo un ciclo de rearranque durante el cual: ● Vuelve a tomar en cuenta el conjunto de los módulos de entradas, ● reinicia la tarea Master con el bit %S1 (reinicio en caliente) posicionado a 1, ● vuelve a poner en estado 0 los bits %S1 cuando termina este primer ciclo de la tarea Master, y ● reactiva la tarea rápida y los tratamientos de sucesos cuando finaliza este primer ciclo de la tarea Master.

Tratamiento a través del programa del rearranque en caliente

En caso de rearranque en caliente, si se desea un tratamiento particular delante de la aplicación, deberá escribirse el programa correspondiente en la prueba de %S1 a 1 al inicio del programa de la tarea maestra.

Evolución de las salidas

Desde la detección del corte del sector, las salidas se sitúan es posición de retorno:
- tanto si toman el valor de retorno,
- como si se mantiene el valor actual,

según la elección efectuada en la configuración.

En la reanudación del sector, las salidas se ponen a cero hasta que la tarea las actualice.

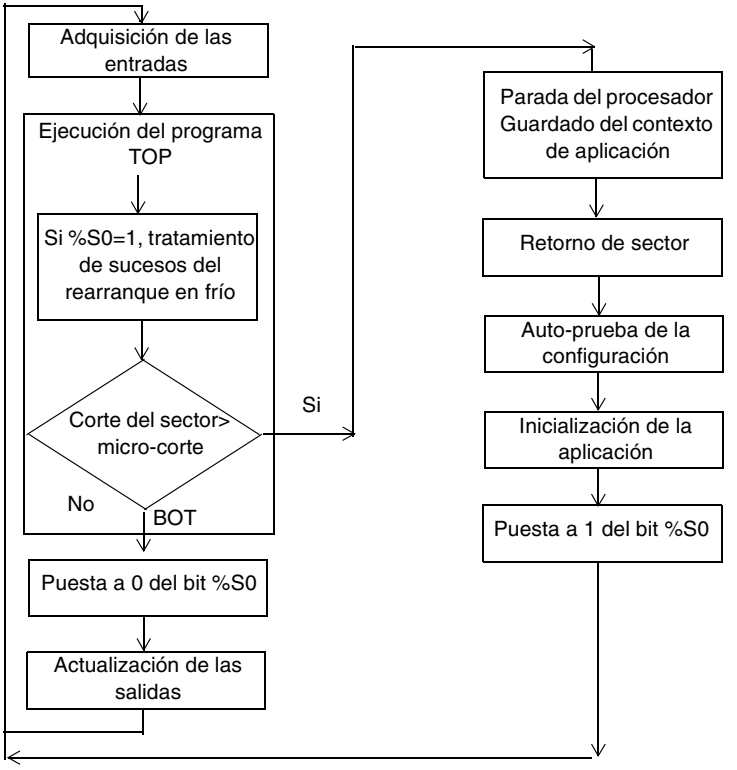
Tratamiento del arranque en frío

Causa de un arranque en frío

La tabla siguiente describe las diferentes causas posibles de un arranque en frío.

Causas	Características del arranque
Carga de una aplicación	Arranque en frío forzado en un STOP
Acción sobre el botón RESET del procesador	Arranque en frío forzado en STOP o en RUN según la definición de la configuración
Acción sobre el botón RESET del procesador después de un fallo con bloqueo	Arranque en frío forzado en STOP
Manipulación del prensor o inserción/ extracción de una tarjeta de memoria PCMCIA	Arranque en frío forzado en STOP o en RUN según la definición de la configuración
Inicialización desde PL7 Junior o PL7 Pro Forzado del bit del sistema %S0	Arranque en frío forzado en STOP o en RUN según la definición de la configuración, sin inicialización de los módulos de entradas/ salidas TON y función específica
Rearranque después de un corte de alimentación con pérdida del contexto	Arranque en frío forzado en STOP o en RUN según la definición de la configuración

Ilustración El esquema siguiente describe el funcionamiento de un rearranque en frío.



Funcionamiento

La tabla que se presenta a continuación describe las fases de reanudación de la ejecución del programa en el rearranque en frío.

Fase	Descripción
1	El arranque se efectúa en RUN o en STOP según el estado del parámetro `Arranque automático en RUN` definido en la configuración o si ésta se utiliza en función del estado de la entrada RUN/STOP. La ejecución del programa se reanuda al comienzo del ciclo.
2	El sistema efectúa: ● la puesta a cero de los bits, de la imagen de las E/S, y de las palabras internas (si la opción RAZ de los %MW en el rearranque en frío queda seleccionada en la pantalla Configuración del procesador). Si la opción RAZ de los %MW no está activa y si las palabras internas %MWi se guardan en la memoria interna Flash EPROM (TSX 37), estas últimas se restituyen en cuanto se produce un arranque en frío. ● la inicialización de los bits y palabras del sistema. ● la inicialización de los bloques de función a partir de los datos de configuración. ● la desactivación de las tareas, que no sean la tarea maestra, hasta que termine el primer ciclo de la tarea maestra. ● el posicionamiento del Grafcet en las etapas iniciales. ● la anulación de forzados. ● la inicialización de datos declarados en los DFB: ya sea a 0, ya sea al valor inicial declarado en el código, o con el valor guardado en el momento de la ejecución de la función SAVE ● la inicialización de las filas de mensaje y de sucesos ● el envío de parámetros de configuración a todos los módulos de entradas/salidas TON y módulos de función específica.
3	Para el primer ciclo de rearranque el sistema: ● relanza las tareas maestras con bit %S0 (rearraque en frío) definidas a 1, la palabra %SW10 (detectando rearranque en frío en la primera revolcución de una tarea) es definida a 0, ● reinicializa bit %S0 a 0 y reinicializa a 1 cada bit de palabra %SW10 al final de este primer ciclo de tareas maestras, ● activa la tarea rápida y los procesos de sucesos al fin de este primer ciclo de tareas maestras.

Tratamiento a través del programa de un arranque en frío

Para efectuar un tratamiento de aplicación después de un arranque en frío del autómata, existe la posibilidad de comprobar mediante el programa el bit %SW10:X0 (si %SW10:X0=0, se ha producido un rearrranque en frío).

Evolución de las salidas

Desde la detección del corte del sector, las salidas se sitúan es posición de retorno:
- tanto si toman el valor de retorno,
- como si se mantiene el valor en curso,

según la elección efectuada en la configuración.

En la reanudación del sector, las salidas se ponen a cero hasta que sean actualizadas por la tarea.

Estructura del programa

5

Presentación

Objeto del capítulo

Este capítulo describe las tareas y su ejecución en el autómata.

Contenido:

Este capítulo contiene las siguientes secciones:

Sección	Apartado	Página
5.1	Descripción de las tareas	92
5.2	Estructura monotarea	101
5.3	Estructura multitarea	109
5.4	Módulos funcionales	116

5.1 Descripción de las tareas

Presentación

Contenido de esta sección

Esta sección describe la función y el contenido de cada una de las tareas que pueden constituir un programa PL7.

Contenido

Esta sección contiene los siguientes apartados:

Apartado	Página
Presentación de la tarea maestra	93
Descripción de las secciones de los subprogramas	94
Presentación de la tarea rápida	98
Presentación de los tratamientos de sucesos	99

Presentación de la tarea maestra

Generalidades

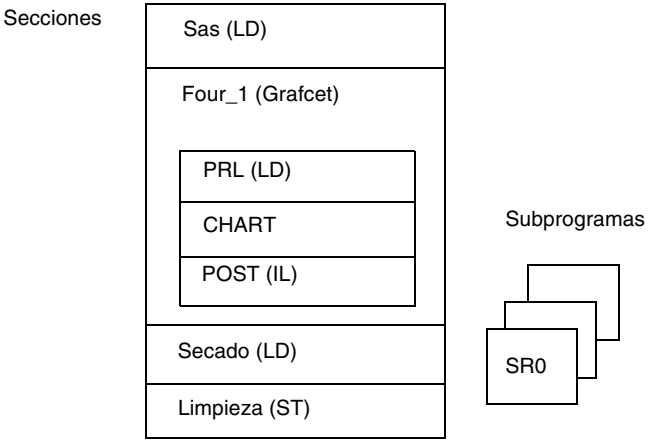
La tarea maestra equivale al programa principal, es obligatoria sea cual sea la estructura adoptada, monotarea o multitarea.

El programa de la tarea maestra (MAST) se compone de varios módulos de programas denominados secciones (Véase *Descripción de las secciones de los subprogramas, p. 94*), y de subprogramas.

La ejecución de la tarea maestra puede elegirse (en la configuración) cíclica (Véase *Ejecución cíclica, p. 103*) o periódica (Véase *Ejecución periódica, p. 105*).

Ilustración

La siguiente ilustración muestra un ejemplo de tarea maestra que consta de 4 secciones y 3 subprogramas.



Descripción de las secciones de los subprogramas

Presentación de las secciones

Las secciones son entidades autónomas de programación. Las etiquetas de identificación de las líneas de instrucciones, las redes de contactos... son propias de la sección (no es posible un salto del programa hacia otra sección).

Se programan ya sea en:
- lenguaje de contactos,
- lista de instrucciones,
- literal estructurado,
- Grafcet.

Las secciones se ejecutan en el mismo orden en que se han programado en la ventana del navegador (vista estructural).

Las secciones se vinculan a una tarea, una misma sección no puede pertenecer simultáneamente a varias tareas.

Presentación de los subprogramas

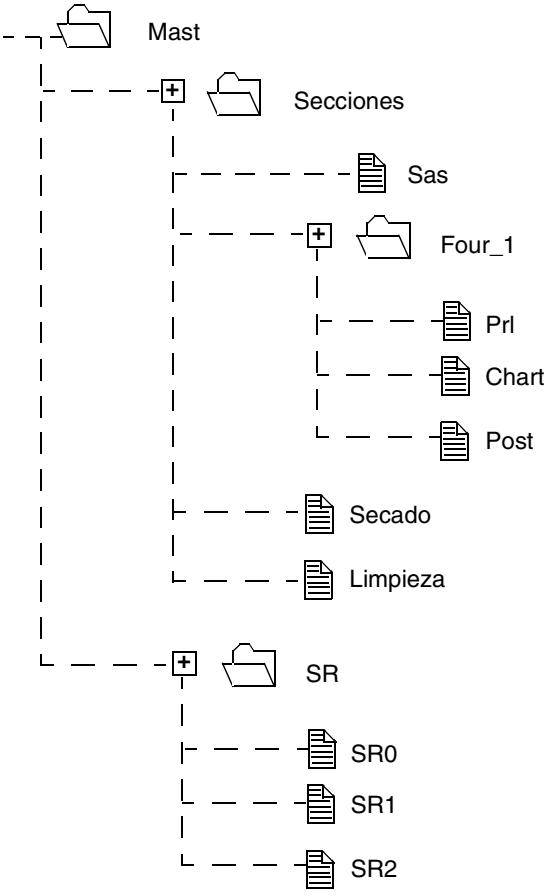
Los módulos de los subprogramas también se programan como entidades separadas ya sea en:
- lenguaje de contactos,
- lista de instrucciones,
- literal estructurado,

Las llamadas a los subprogramas se efectúan en las secciones o desde otro subprograma (con un máximo de 8 niveles de imbricación).

Los subprogramas también están vinculados a una tarea, un mismo subprograma no puede llamarse desde varias tareas.

Ejemplo

El esquema siguiente ofrece un ejemplo de la estructura de una tarea dividida en secciones y subprogramas.



Características de una sección

La tabla siguiente describe las características de una sección.

Característica	Descripción
Nombre	máximo 24 caracteres
Lenguaje	Lenguaje de contactos, Lista de instrucciones, Literal estructurado o Grafcet
Tarea	Maestra o rápida
Condición (opcional)	Objetos autorizados como condición: ● %M,%S,%X ● bits indexados, bits que se extraen de las palabras ● %I , %Q Todos estos objetos se pueden forzar desde el terminal, a excepción de los bits %S, bits indexados, bits extraídos, %Ixy.i.ERR, y %I xy.MOD.ERR. La condición deberá estar en estado 1 para que pueda ejecutarse la sección.
Comentario	máximo 250 caracteres.
Protección	Protección contra la escritura, protección contra lectura/escritura. La protección puede ser global o parcial.

Nota: en el arranque en frío, las condiciones de ejecución se ponen a 0, se inhiben todas las secciones a las que se asocia una condición.

Sección Grafcet

La tabla siguiente describe los elementos de programa de una sección Grafcet.

Tratamiento	Designación	Características
Preliminar	PRL	Programado en Lenguaje de contactos LD, Lista de instrucciones IL o Literal estructurado ST. Se ejecuta antes que el Grafcet.
Grafcet	CHART	En las páginas Grafcet, se programan las receptividades asociadas a las transiciones y a las acciones asociadas a las etapas o a las etapas de las macro etapas.
Posterior	POST	Programado en Lenguaje de contactos LD, Lista de instrucciones IL o Literal estructurado ST. Se ejecuta después del Grafcet.

Características de un subprograma

La tabla siguiente describe las características de un subprograma SRi.

Característica	Descripción
Número	0 a 253
Lenguaje	Lenguaje de contactos, Lista de instrucciones, Literal estructurado.
Tarea	Maestra o rápida
Comentario	máximo 250 caracteres

Presentación de la tarea rápida

Generalidades

Esta tarea, que tiene una prioridad mayor que la tarea maestra MAST, es periódica con el fin de dejar el tiempo a la tarea menos prioritaria a la hora de la ejecución.

Además, los tratamientos que se le asocian deben ser cortos para no castigar a la tarea maestra. De la misma manera que para la tarea maestra, el programa asociado consta de secciones y de subprogramas.

Periodo de la tarea rápida

El periodo para la tarea rápida FAST queda fijado en la configuración, entre 1 y 255 ms. Puede definirse superior a la tarea maestra MAST para adaptarse a los tratamientos periódicos lentos, pero prioritarios.

Sin embargo, el programa ejecutado debe ser corto para evitar el rebasamiento de las tareas con menos prioridad.

La tarea rápida se controla por un watchdog que permite detectar una duración anormal del programa de aplicación. En caso de rebasamiento, el bit del sistema %S11 se sitúa en 1 y el autómata declara la aplicación en fallo con bloqueo.

Control de la tarea rápida

La palabra del sistema %SW1 contiene el valor del periodo, se inicializa en el rearranque en frío a través del valor definido en la configuración, se puede cambiar a través del usuario, del programa o del terminal.

Los bits y palabras del sistema permiten controlar la ejecución de esta tarea:
- %S19: señala un rebasamiento del periodo, el sistema lo sitúa en 1, cuando el tiempo del ciclo es superior al del periodo de la tarea.
- %S31: permite validar o inhibir la tarea rápida, el sistema lo pone a 0 en el arranque en frío de la aplicación, en el momento en que termina el primer ciclo de la tarea maestra. Se pone a 1 ó a 0 para validar o inhibir la tarea rápida.

Visualización de los tiempos de ejecución de la tarea rápida

Las palabras del sistema que siguen permiten disponer de información sobre el tiempo del ciclo:
- %SW33 contiene el tiempo de ejecución del último ciclo,
- %SW34 contiene el tiempo de ejecución del ciclo más largo,
- %SW35 contiene el tiempo de ejecución del ciclo más corto.

Presentación de los tratamientos de sucesos

Generalidades

Los tratamientos secuenciales permiten reducir el tiempo de reacción del programa en los sucesos de comando que proceden de algunos módulos de funciones específicas.

Estos tratamientos se ejecutan con prioridad a todas las otras tareas. Por tanto, convienen a los tratamientos que reclaman plazos de reacción muy cortos con relación a la llegada del suceso.

El número de tratamientos de sucesos programables depende del tipo de procesador.

Tipo de autómata	Número de tratamientos	Designación
Micro TSX 37-05/08/10	8	EVT1 a EVT8
Micro TSX 37-21/22	16	EVT0 a EVT15
Premium TSX/PCX 57-1•	32	EVT0 a EVT31
Premium TSX/PCX 57-2•/3•/4•	64	EVT0 a EVT63

Funcionamiento

La aparición de un suceso desvía el programa de aplicación hacia el tratamiento que está asociado a la vía de entradas/salidas que ha provocado el suceso.

El sistema actualiza las entradas (%I, %IW, %ID) asociadas a la vía de E/S que ha desencadenado el suceso antes de la llamada del tratamiento de sucesos.

La asociación entre una vía y un número de suceso se lleva a cabo en la pantalla de configuración de las vías.

Sucesos de comando

Se trata de los sucesos externos asociados a las funciones específicas.

En los autómatas Micro, los tratamientos de sucesos pueden desencadenarse a través de:

- las entradas 0 a 3 del módulo de posición 1, en flanco ascendente o descendente,
- la vía o las vías de contaje de los módulos de contaje,
- las vías de contaje del módulo 1 (si éste está configurado como contador),
- la recepción de telegramas en un TSX 37-21/22 equipado con un módulo TSX FPP20.

En los autómatas Premium, los tratamientos de sucesos pueden desencadenarse a través de:

- les entradas de los módulos DEY 16 FK, DMY 28 FK, DMY 28 RFK
- las vías de módulos de contaje,
- las vías de módulos de comando de eje TSX CAY •,
- las vías de módulos de comando paso a paso TSX CFY •,
- las vías de comunicación "FPP20",
- ...

Gestión de los tratamientos de sucesos

Los tratamientos de sucesos pueden validarse en su totalidad o inhibirse mediante el programa de aplicación, a través del bit del sistema %S38.
Si uno o varios sucesos intervienen en el mismo momento en que se inhiben, los tratamientos asociados se pierden.

Dos instrucciones de lenguaje PL7, `MASKEVT()` y `UNMASKEVT()`, utilizadas en el programa de aplicación, permiten también enmascarar o desenmascarar los tratamientos de sucesos.
Si uno a varios sucesos intervienen en el mismo momento en que se enmascaran, el sistema los almacena y los tratamientos asociados se ejecutarán después del desenmascaramiento.

Prioridad de los tratamientos

Autómatas Micro TSX 37-05/08/10
Los 8 sucesos de comando posibles tienen todos el mismo nivel de prioridad; así, un tratamiento secuencial no puede ser interrumpido por otro.

Autómatas Micro TSX 37-21/22 o Premium
Hay 2 niveles de prioridad para los sucesos de comando: el suceso 0 (EVT0) tiene mayor prioridad que el resto de los sucesos.

5.2 Estructura monotarea

Presentación

Contenido de esta sección

Esta sección describe cómo se ejecuta una aplicación monotarea.

Contenido

Esta sección contiene los siguientes apartados:

Apartado	Página
Estructura del programa monotarea	102
Ejecución cíclica	103
Ejecución periódica	105
Control del tiempo del ciclo	108

Estructura del programa monotarea

Descripción

El programa de una aplicación monotarea se asocia a una única tarea del usuario, la tarea maestra MAST (véase *Presentación de la tarea maestra, p. 93*).

El programa asociado a la tarea maestra (MAST) consta de varias secciones y de subprogramas.

La ejecución de la tarea maestra se puede elegir (en la configuración):
- cíclica (Véase *Ejecución cíclica, p. 103*)
- o periódica (Véase *Ejecución periódica, p. 105*)

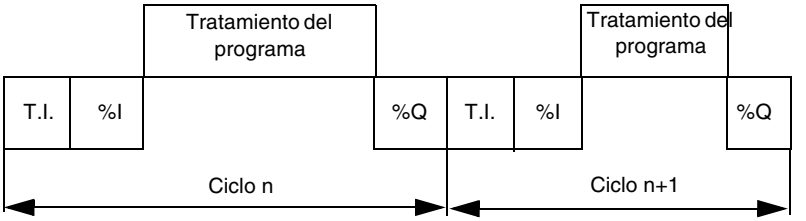
Ejecución cíclica

Descripción

Este tipo de funcionamiento corresponde a la ejecución normal del ciclo del autómata (por defecto).
Se trata de encadenar, unos detrás de otros, los ciclos de la tarea maestra (MAST).
Una vez efectuada la actualización de las salidas, el sistema realiza los tratamientos propios, luego encadena otro ciclo de la tarea.

Funcionamiento

El esquema siguiente muestra las fases de ejecución del ciclo del autómata.



Descripción de las distintas fases

La tabla que se ofrece a continuación describe las fases de funcionamiento.

Variable	Fase	Descripción
T.I.	Tratamiento interno	El sistema lleva a cabo implícitamente la vigilancia del autómata (gestiona los bits y palabras del sistema, actualiza los valores actuales del reloj-calendario, actualiza los indicadores de estado, detecta los cambios RUN/STOP, ...) y el tratamiento de las peticiones que provengan del terminal (modificaciones y animación). En el caso del Premium, el tratamiento interno se realiza en paralelo con los tratamientos de las entradas y de las salidas.
%I	Adquisición de las entradas	Escritura en memoria del estado de la información presente en las entradas de los módulos TON y función específica asociadas a la tarea,
-	Tratamiento del programa	Ejecución del programa de aplicación, escrito por el usuario,
%Q	Actualización de las salidas	Escritura de los bits o de las palabras asociadas a los módulos TON y función específica, incorporados a la tarea según el estado definido por el programa de aplicación.

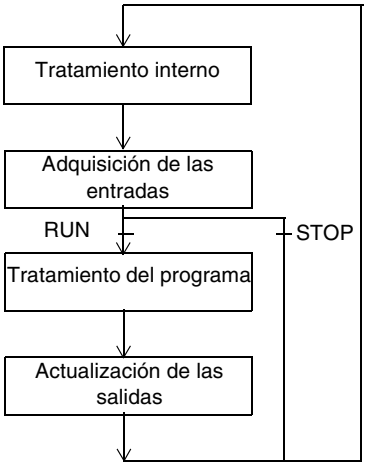
Modo de funcionamiento

Autómata en RUN, el procesador ejecuta según la orden, el tratamiento interno, la adquisición de las entradas, el tratamiento del programa de aplicación y la actualización de las salidas.
Autómata en STOP, el procesador efectúa:

- el tratamiento interno,
- la adquisición de las entradas,
- y según la configuración elegida:
  - modo retorno: las salidas se sitúan en posición de "retorno",
  - modo de mantenimiento: las salidas se mantienen en su último valor.

Ilustración

La siguiente ilustración muestra los ciclos de funcionamiento.



Control del ciclo

El control del ciclo se efectúa a través del watchdog (Véase *Control del tiempo del ciclo, p. 108*).

Ejecución periódica

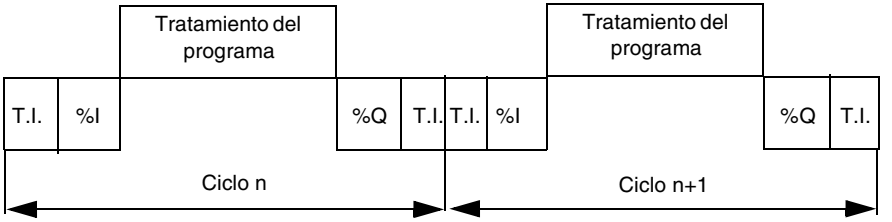
Descripción

En este modo de funcionamiento, la adquisición de las entradas, el tratamiento del programa de aplicación y la actualización de las salidas se efectúan periódicamente según un tiempo definido en la configuración (de 1 a 255 ms).

En el inicio del ciclo del autómata, un temporizador en el que el valor actual se inicializa en función del periodo definido en la configuración, empieza a descontar. El ciclo del autómata debe finalizar antes de que termine el temporizador, el cual reinicia un nuevo ciclo.

Funcionamiento

El esquema siguiente muestra las fases de ejecución del ciclo del autómata.



Descripción de las distintas fases

La tabla que se ofrece a continuación describe las fases de funcionamiento.

Variable	Fase	Descripción
T.I.	Tratamiento interno	El sistema lleva a cabo implícitamente la vigilancia del autómata (gestiona los bits y palabras del sistema, pone al día los valores actuales del reloj-calendario, actualiza los indicadores de estado, detecta los cambios RUN/STOP...) y el tratamiento de las peticiones que provengan del terminal (modificaciones y animación) En el caso del Premium, el tratamiento interno se realiza en paralelo con los tratamientos de las entradas y de las salidas.
%I	Adquisición de las entradas	Escritura en memoria del estado de las informaciones presentes en las entradas de los módulos TON y función específica asociados a la tarea,
-	Tratamiento del programa	Ejecución del programa de aplicación, escrito por el usuario,
%Q	Actualización de las salidas	Escritura de los bits o de las palabras asociadas a los módulos TON y función específica, incorporados a la tarea según el estado definido por el programa de aplicación.

Modo de funcionamiento

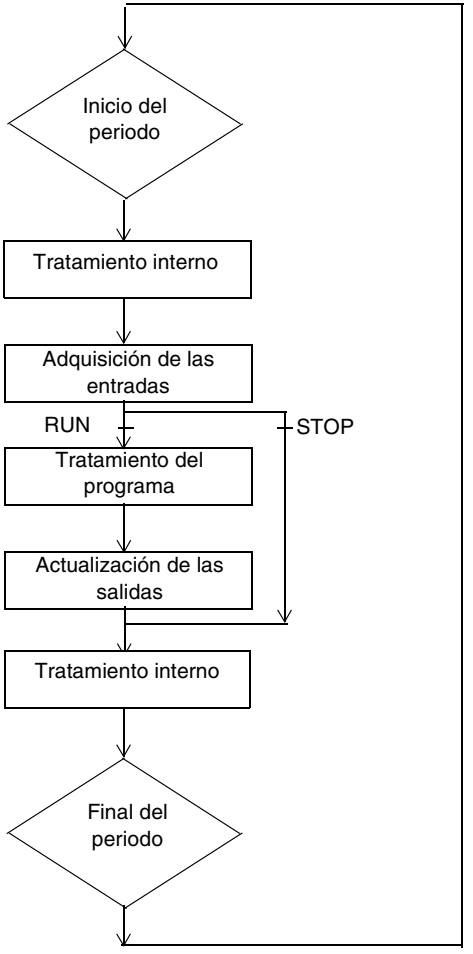
Autómata en RUN, el procesador ejecuta según la orden, el tratamiento interno, la adquisición de las entradas, el tratamiento del programa de aplicación y la actualización de las salidas.

- Si el periodo aún no ha finalizado, el procesador completa el ciclo de funcionamiento hasta que termine el periodo del tratamiento interno.
- Si el tiempo de funcionamiento fuera superior al que se deba cumplir en el periodo, el autómata indica un rebasamiento de periodo pasando al estado 1 el bit del sistema %S19 de la tarea, el tratamiento continua y se ejecuta en su totalidad (no obstante, no debe sobrepasar el tiempo límite del watchdog). El ciclo que sigue se encadena después de la escritura implícita de las salidas del ciclo en curso.

Autómata en STOP, el procesador realiza:

- el tratamiento interno,
- la adquisición de las entradas,
- y según la configuración elegida:
  - modo de retorno: las salidas se sitúan en posición de "retorno",
  - modo de mantenimiento: las salidas se mantienen en su último valor.

Ilustración La siguiente ilustración muestra los ciclos de funcionamiento.



Control del ciclo Se ejecutan dos controles:

- rebasamiento del periodo (Véase *Control del tiempo del ciclo, p. 108*),
- por watchdog (Véase *Control del tiempo del ciclo, p. 108*),

Control del tiempo del ciclo

Generalidades

La duración de la ejecución de la tarea maestra, en funcionamiento cíclico o periódico, se controla a través del autómata (watchdog) y no debe sobrepasar el valor definido en la configuración Tmáx (250 ms por defecto, 500 ms como máximo).

Watchdog del programa (funcionamiento periódico o cíclico)

En caso de rebasamiento, la aplicación entra en fallo, lo que provoca la parada inmediata del autómata.

- **en el Micro**, la puesta a 0 de la salida alarma %Q2.0, si se ha configurado,
- **en el Premium**, la puesta a 0 del relé de alarma en la alimentación

El bit %S11 permite dirigir la ejecución de esta tarea. Señala un rebasamiento del watchdog, el sistema lo sitúa en 1, cuando el tiempo del ciclo es superior al watchdog.

Nota: En el Premium, el valor del watchdog debe ser superior al periodo.

Control en el funcionamiento periódico

En funcionamiento periódico, un control adicional permite detectar un rebasamiento del periodo:

- %S19: señala un rebasamiento del periodo, el sistema lo sitúa en 1, cuando el tiempo del ciclo es superior al del periodo de la tarea.
- %SW0: esta palabra contiene el valor del periodo (en ms), se inicializa en el rearranque en frío a través del valor definido en la configuración, el usuario lo puede cambiar.

Aprovechamiento de los tiempos de ejecución de la tarea maestra

Las palabras del sistema que siguen permiten disponer de información sobre el tiempo del ciclo:

- %SW30 contiene el tiempo de ejecución del último ciclo.
- %SW31 contiene el tiempo de ejecución del ciclo más largo.
- %SW32 contiene el tiempo de ejecución del ciclo más corto.

Nota: A estas diferentes informaciones se puede acceder también desde el editor de configuración de forma explícita.

5.3 Estructura multitarea

Presentación

Objeto de esta sección

Esta sección describe cómo se ejecuta una aplicación multitarea.

Contenido

Esta sección contiene los siguientes apartados:

Apartado	Página
Estructura del programa de multitarea	110
Desglose secuencial de las tareas en una estructura multitarea	111
Asignación de las vías de entradas/salidas a las tareas maestra y rápida	112
Intercambios de entradas/salidas en los tratamientos de sucesos	113

Estructura del programa de multitarea

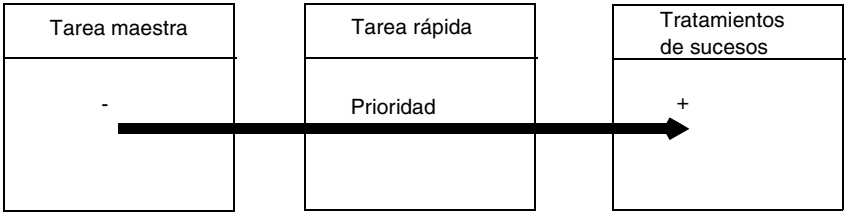
Descripción

La estructura en tareas de una aplicación semejante es la siguiente:

Tarea	Designación	Descripción
Maestra	MAST	Siempre presenta la que puede ser cíclica o periódica.
Rápida	FAST	Opcional que siempre es periódica.
De sucesos	EVTi	Llamadas por el sistema en el momento de la aparición de un suceso en un acoplador de entradas/salidas. Estos tratamientos son opcionales y sirven para las aplicaciones que precisan de tiempos de respuesta cortos para actuar sobre las entradas/salidas.

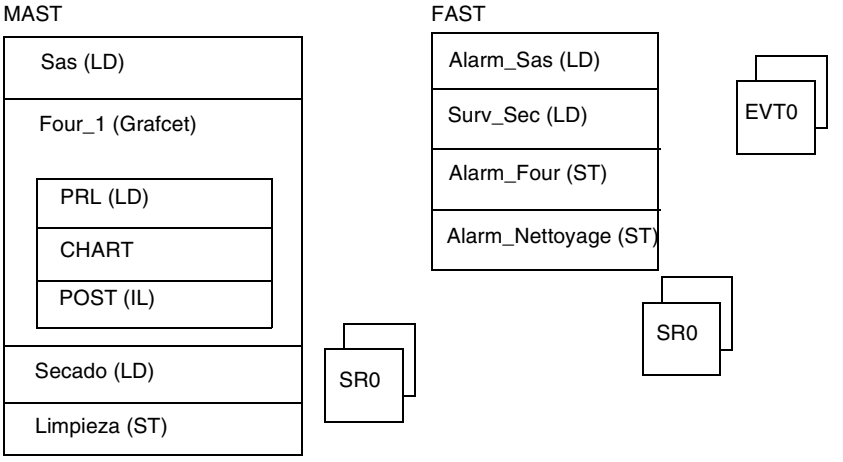
Ilustración

El esquema siguiente muestra las tareas de una estructura multitarea y su nivel de prioridad.



Ejemplo

El siguiente ejemplo presenta una estructura multitarea que consta de una tarea maestra MAST, de una tarea rápida FAST y de 2 tratamientos secuenciales EVTO y EVT1.



Desglose secuencial de las tareas en una estructura multitarea

Generalidades

La tarea maestra es activa por defecto.
La tarea rápida es activa por defecto si se ha programado.
El tratamiento de sucesos se activa en el momento de la aparición del suceso que se le ha asociado.

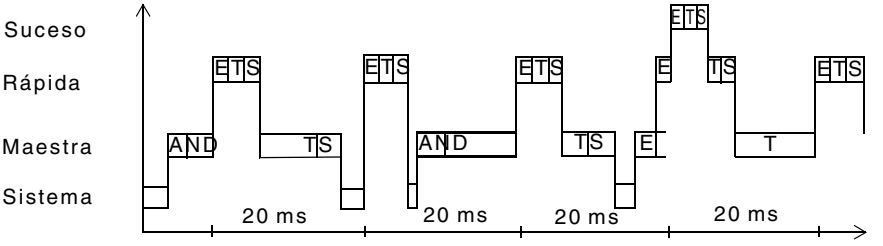
Funcionamiento

La tabla siguiente describe la ejecución de las tareas prioritarias.

Fase	Descripción
1	Llegada de un suceso o inicio del ciclo de la tarea rápida.
2	Parada de la ejecución de las tareas en curso menos prioritarias,
3	Ejecución de la tarea prioritaria.
4	La tarea interrumpida se reanuda cuando los tratamientos de la tarea prioritaria finalizan.

Descripción del desglose secuencial de las tareas

El diseño siguiente ilustra el desglose secuencial de las tareas de un tratamiento multitarea que incluye una tarea maestra cíclica, una tarea rápida con un periodo de 20 ms y un tratamiento de sucesos.



Leyenda:
E: adquisición de las entradas
T: tratamiento del programa
S: actualización de las salidas

Control de las tareas

La ejecución de las tareas rápidas y de sucesos puede controlarse a través del programa, usando los bits del sistema:

- %S30 permite activar o desactivar la tarea maestra MAST.
- %S31 permite activar o desactivar la tarea rápida FAST.
- %S38 permite activar o desactivar los tratamientos de sucesos EVTi.

Asignación de las vías de entradas/salidas a las tareas maestra y rápida

Generalidades

Además del programa de aplicación, las tareas maestra MAST y rápida FAST ejecutan funciones del sistema vinculadas a la gestión de las entradas/salidas implícitas que se les asocian.

La asociación de una vía o de un grupo de vías a una tarea se define en la pantalla de configuración del acoplador correspondiente; la tarea asociada predeterminada es la tarea MAST.

Módulos TON

La disposición de los módulos TON es de 8 vías sucesivas (vías 0 a 7, vías 8 a 15,...), las entradas/salidas pueden verse afectadas por grupos de 8 vías, independientemente de la tarea MAST o FAST.

Ejemplo: se puede asignar las vías de un módulo de 28 entradas/salidas de la forma siguiente:
- entradas 0 a 7 asociadas a la tarea MAST,
- entradas 8 a 15 asociadas a la tarea FAST,
- salidas 0 a 7 asociadas a la tarea MAST,
- salidas 8 a 15 asociadas a la tarea FAST.

Módulos de contaje

Cada vía de un módulo de contaje puede asociarse indistintamente a la tarea MAST o FAST.

Ejemplo: para un módulo de contaje de 2 vías se pueden asignar:
- la vía 0 a la tarea MAST,
- la vía 1 a la tarea FAST.

Módulos analógicos

Las vías de los módulos de entradas analógicas del Micro se asignan obligatoriamente a la tarea MAST. Por el contrario, se pueden vincular las vías o grupos de vías de las salidas analógicas, indistintamente de la tarea MAST o FAST, con una distribución de módulos de 2 vías.
Ejemplo: para un módulo de 4 salidas analógicas, se pueden asignar:
- las vías 0 y 1 a la tarea MAST y,
- las vías 2 y 3 a la tarea FAST.

Las vías de los módulos de entradas y salidas analógicas del Premium pueden vincularse a la tarea MAST o FAST. Esta asociación es individual para cada una de las vías de los módulos de entradas o de salidas analógicas aisladas (4 vías aisladas) y con una distribución de módulos de 4 vías para los módulos restantes.

Nota: Para obtener resultados óptimos, es preferible reagrupar las vías de un módulo en una misma tarea.

Intercambios de entradas/salidas en los tratamientos de sucesos

Generalidades

En cada tratamiento secuencial se pueden usar otras vías de entradas/salidas que no sean las propias del suceso.
De este modo, los intercambios se realizan implícitamente mediante el sistema antes (%I) y después (%Q) del tratamiento de aplicación.
Estos intercambios pueden circunscribirse a una vía (ejemplo, el módulo de contaje) o a un grupo de vías (módulo TON). En el segundo caso, si el tratamiento modifica, por ejemplo, las salidas 2 y 3 de un módulo TON, la imagen de salidas 0 a 7 se transferirá hacia el módulo.

Funcionamiento

La tabla siguiente describe los intercambios y los tratamientos efectuados.

Fase	Descripción
1	La aparición de un suceso desvía el programa de aplicación hacia el tratamiento que está asociado a la vía de entrada/salida que ha provocado del suceso.
2	Todas las entradas asociadas a la vía que ha provocado el suceso se obtienen automáticamente.
3	Se obtienen todas las entradas empleadas por el usuario en el tratamiento EVT i.
4	Se ejecuta el tratamiento secuencial. Deberá ser lo más breve posible.
5	Se actualizan todas las salidas empleadas por el usuario en el tratamiento EVT i. Las salidas asociadas a la vía que ha provocado el suceso también se deberán utilizar, para actualizarlas.

Regla de programación

Regla general:
Las entradas intercambiadas (y el grupo de vías asociadas), una vez ejecutado el tratamiento de sucesos, se actualizan (pérdida de los valores registrados, por lo tanto de los flancos), por ello, se deberá evitar comprobar los flancos en esas entradas de las tareas maestra (MAST) o rápida (FAST).

Si se trata de los módulos TOR TSX DEY16FK, TSX DMY28FK o TSX DMY28RFK:
La entrada que ha provocado el suceso no deberá comprobarse en el tratamiento de sucesos (el valor no se actualiza).
La prueba del flanco que ha provocado el suceso deberá efectuarse en la palabra de estado:

- %IWxy.i:X0 = 1 --> flanco ascendente,
- %IWxy.i:X0 = 1 --> flanco descendente.

En los autómatas Micro:

- los módulos de entradas analógicas que solamente pueden usarse en la tarea MAST, no deben intercambiarse en un tratamiento de sucesos.
- para cada tratamiento de sucesos, se pueden declarar como máximo los intercambios para 2 módulos de entrada (antes del tratamiento del suceso) y 2 módulos de salida (después del tratamiento del suceso).

Rendimiento

En los autómatas Premium, según el procesador utilizado, el número de intercambios usados no tiene límites:

Número de intercambios que pueden usarse en los tratamientos de sucesos mediante procesador	**P57-1•**	**P57-2• /3• /4•**
Entradas/salidas TON	32 intercambios	128 intercambios
Entradas/salidas analógicas	8 intercambios	16 intercambios
Otras funciones específicas	4 intercambios	16 intercambios

Para las entradas/salidas TON, un intercambio que incluye un grupo de 8 vías, se genera cuando se usan las entradas de un grupo de 8 vías (cualquier otro que no sea el grupo de vías que genera el suceso) y una vez que se han escrito las salidas de un grupo de 8 vías.
Para las entradas/salidas analógicas o de otra función específica, un intercambio se genera cuando se usan las entradas de una vía (distinta de la que ha generado el suceso y en el momento en que se han escrito las salidas de una vía).

Nota:
- Los intercambios de entradas/salidas de la tarea EVTi, que se han ejecutado por una vía (para algunos módulos analógicos y funciones específicas) o por un grupo de vías (para los módulos TON, y algunos módulos analógicos), si el tratamiento modifica, por ejemplo, las salidas 2 y 3 de un módulo TON, se transferirá hacia el módulo la imagen (memoria del autómata) de las salidas 0 a 7.
- Todo intercambio de una entrada/salida en una tarea de sucesos puede provocar la pérdida de la información de flanco frente a los tratamientos efectuados en esta vía (o grupo de vías), en la tarea donde se ha declarado: MAST o FAST.

Visualización del número de sucesos tratados

La palabra del sistema %SW48 ofrece el número de sucesos tratados.
Esta palabras se inicializa en 0 en el arranque en frío, después se incrementa por el sistema en el momento del inicio de un suceso.
Esta palabra la puede modificar el usuario.

El bit del sistema %S39 señala la pérdida del suceso.

5.4 Módulos funcionales

Estructuración en módulos funcionales

Generalidades

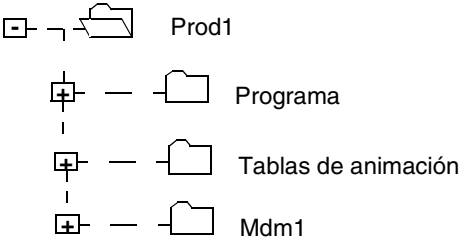
Un módulo funcional es una reagrupación de elementos de programa destinados a realizar una función de automatismo.

Estructura

Un módulo funcional se define por los atributos siguientes:
- nombre corto: 8 caracteres (ejemplo: TR371)
- nombre largo: 16 caracteres (ejemplo: Avance/Retroceso por BT371)
- una ficha descriptiva (sin limitación del número de caracteres) no almacenada en el autómata, aunque almacenada en el archivo .STX de la aplicación.

Ilustración

La siguiente ilustración muestra la composición de un módulo funcional:



Descripción de los elementos de un módulo funcional

La tabla describe la función de cada uno de los elementos:

Elemento	Composición
Programa	Uno o varios módulos de código: ● secciones ● sucesos ● macro etapas ● tablas de animación ● ...
Tablas de animación	Una o varias tablas de animaciones.
Mdm1	Módulos funcionales de nivel inferior, estos módulos asumen, en relación con la función principal, una o varias subfunciones de automatismo.

Límite de uso

Únicamente el producto PL7 PRO permite la puesta en marcha de módulos funcionales en los autómatas Premium.

Descripción de los lenguajes PL7

II

Presentación

Objeto de esta parte

Esta parte describe los lenguajes de programación de los autómatas Micro y Premium.

Contenido

Esta parte contiene los siguientes capítulos:

Capítulo	Nombre del capítulo	Página
6	Lenguaje de contactos	119
7	Lenguaje lista de instrucciones	133
8	Lenguaje literal estructurado	151
9	Grafcet	175
10	Bloques de función DFB	221

Lenguaje de contactos

6

Presentación

Contenido del capítulo

Este capítulo describe la programación en lenguaje de contactos.

Contenido:

Este capítulo contiene los siguiente apartados:

Apartado	Página
Presentación general del lenguaje de contactos	120
Estructura de una red de contactos	121
Etiqueta de una red de contactos	122
Comentario de una red de contactos	123
Elementos gráficos del lenguaje de contactos	124
Reglas de programación de una red de contactos	127
Regla de programación de los bloques de función	128
Reglas de programación de los bloques de operación	129
Ejecución de una red de contactos	130

Presentación general del lenguaje de contactos

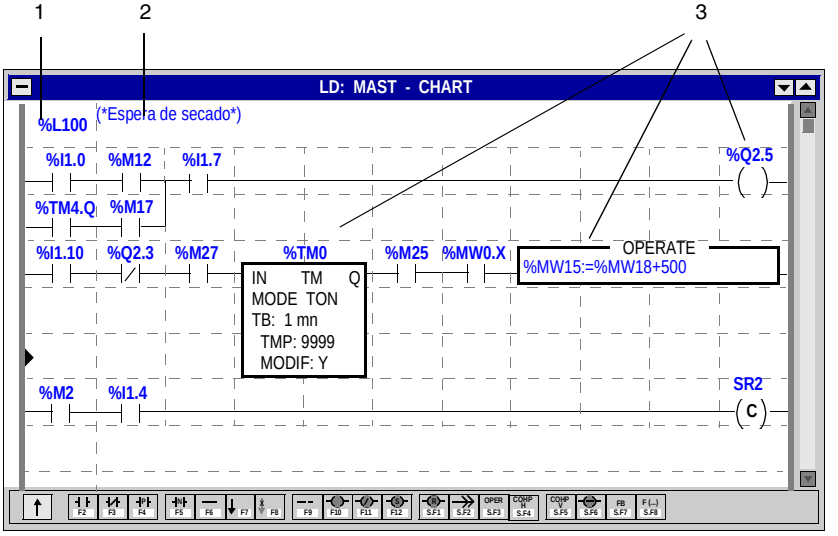
Generalidades

Una sección de programa escrita en lenguaje de contactos está constituida por una serie de redes de contactos ejecutados secuencialmente por el autómata.

La representación de una red de contactos es muy parecida a la de un esquema eléctrico.

Ilustración de una red de contactos

La próxima pantalla muestra una red de contactos PL7.



Composición de una red de contactos

Esta tabla describe los elementos que constituyen una red de contactos.

Variable	Elemento	Función
1	Etiqueta	Variable de una red de contactos (opcional).
2	Comentario	Información sobre una red de contactos (opcional).
3	Elementos gráficos	Representan: ● las entradas/salidas del autómata (botones-pulsadores, detectores, relés, indicadores...) ● las funciones de los automatismos (temporizadores, contadores...,), ● las operaciones aritméticas, lógicas y específicas, ● las variables internas del autómata.

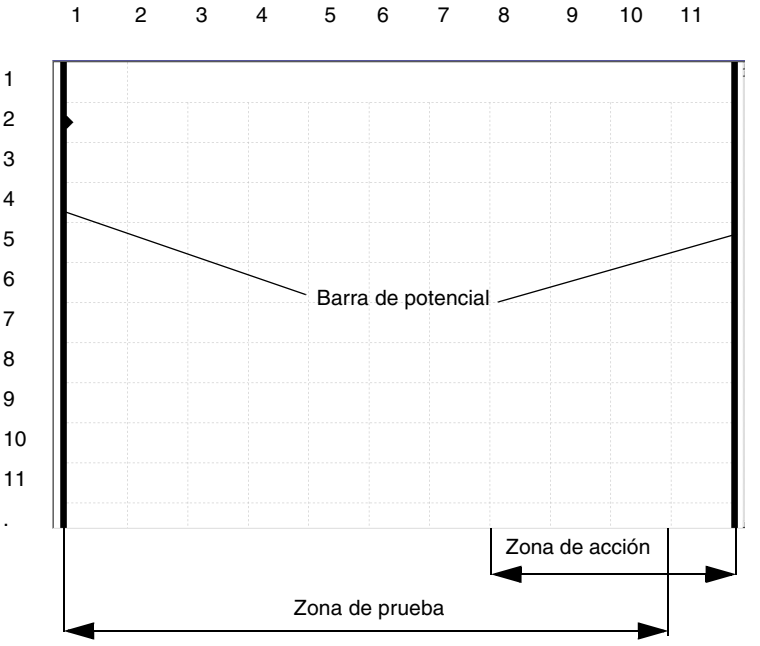
Estructura de una red de contactos

Introducción

Una red se sitúa entre dos barras de potencial. El sentido de circulación de la corriente se establece desde la barra de potencial izquierda hacia la barra de potencial derecha.

Ilustración

El siguiente diseño describe la estructura de una red de contactos.



Descripción de una red de contactos

Una red de contactos se compone de un conjunto de elementos gráficos colocados en una cuadrícula de:

- máximo 16 líneas y 11 columnas (para autómatas Premium),
- máximo 7 líneas y 11 columnas (para autómatas Micro).

Se reparte en dos zonas:

- la zona de prueba, en la que constan las condiciones necesarias para una acción
- la zona de acción, que aplica el resultado consecutivo a un encadenamiento de prueba.

Etiqueta de una red de contactos

Generalidades

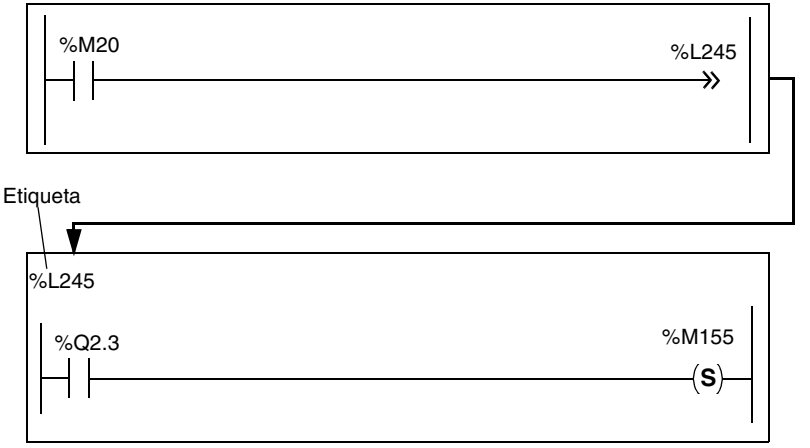
La etiqueta permite identificar una red en una entidad de programa (programa principal, subprograma,...). Es opcional.

Sintaxis

Esta etiqueta tiene la siguiente sintaxis: %Li con i comprendido entre 0 y 999. Se sitúa en la parte superior izquierda delante de la barra de potencial.

Ilustración

Las redes de contactos que se presentan a continuación ilustran el uso de una etiqueta.



Reglas

Una variable de etiqueta sólo puede asignarse a una única red en el seno de una misma entidad de programa.

Es necesario etiquetar una red con el fin de permitir una ramificación cuando se ha producido un salto de programa (véase la ilustración que sigue).

El orden de variables de las etiquetas es indistinto, (es el orden de introducción de las redes que el sistema tiene en cuenta durante el recuento).

Comentario de una red de contactos

Generalidades

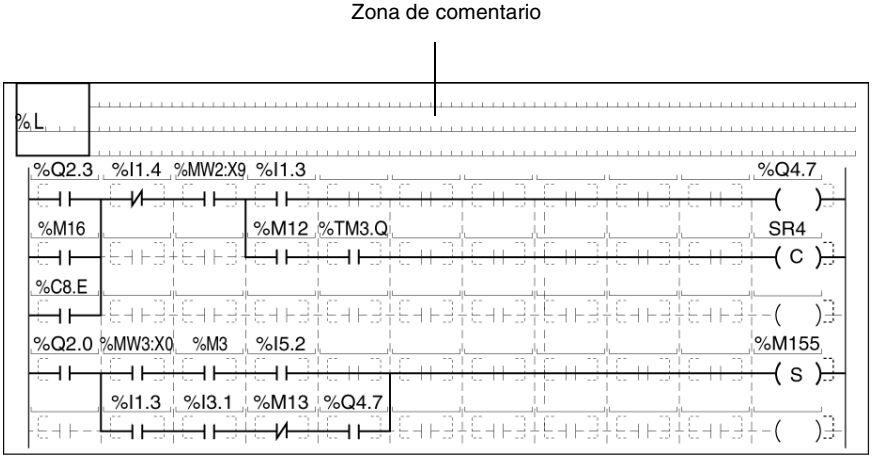
El comentario facilita la interpretación de la red a la que está asignado, aunque no es obligatorio.

Sintaxis

El comentario se integra en la red y comprende, como máximo, 222 caracteres alfanuméricos encuadrados de una y otra parte por los caracteres (* y *).

Ilustración

El siguiente diseño identifica la posición del comentario.



Reglas

los comentarios se muestran en la zona reservada en la parte superior de la red de contactos.

Si se suprimiera una red, el comentario que lleva asociado también se suprimiría.

Los comentarios se almacenan en el autómata y son accesibles para el usuario en todo momento. Según esto, consumen memoria de programa

Elementos gráficos del lenguaje de contactos

Generalidades

Los elementos gráficos son las instrucciones del lenguaje de contactos.

Contactos

Los elementos gráficos de contactos se programan en la zona de prueba y ocupan una celda (1 línea de alto y una columna de ancho).

Designación	Grafismo	Funciones
Contacto de cierre		Contacto de paso cuando el objeto bit que lo dirige está en el estado 1.
Contacto de apertura		Contacto de paso cuando el objeto bit que lo dirige está en el estado 0.
Contacto de detección del flanco ascendente		Flanco ascendente: detección del paso de 0 a 1 del objeto bit que lo controla.
Contacto de detección del flanco descendente		Flanco descendente: detección del paso de 1 a 0 del objeto bit que lo controla.

Elementos de unión

Los elementos gráficos de unión permiten vincular los elementos gráficos de prueba y de acción.

Designación	Grafismo	Funciones
Conexión horizontal		Permite vincular en serie los elementos gráficos de prueba y de acción entre las dos barras de potencial.
Conexión vertical de potencial		Permite vincular en paralelo los elementos gráficos de prueba y de acción.
Derivación de cortocircuito		Permite enlazar 2 objetos a través de varias conexiones.

Bobinas

Los elementos gráficos de bobinas se programan en la zona de acción y ocupan una celda (1 línea de alto y una columna de ancho).

Designación	Grafismo	Funciones
Bobina directa		El objeto bit asociado toma el valor del resultado de la zona de prueba.
Bobina inversa		El objeto bit asociado toma el valor inverso del resultado de la zona de prueba.
Bobina de conexión		El objeto bit asociado se sitúa en 1 cuando el resultado de la zona de prueba está en 1.
Bobina de desconexión		El objeto bit asociado se sitúa en 0 cuando el resultado de la zona de prueba está en 1.
Salto condicional a otra red (JUMP)	->>%Li	Permite una ramificación hacia una red etiquetada, hacia arriba ó hacia abajo. Los saltos sólo son efectivos en el seno de una misma entidad de programación (programa principal, subprograma...). La ejecución de un salto provoca: ● la parada de la exploración de la red en curso, ● la ejecución de la red etiquetada solicitada, ● la no exploración de la parte del programa situada entre la acción del salto y la red designada.
Bobina sostenida		Propuesta en lenguaje Grafcet, utilizada en el momento del sostenido de la programación de las receptividades asociadas a las transiciones provoca el paso a la etapa siguiente.
Bobina de llamada a un subprograma (CALL)		Permite una ramificación al inicio del subprograma cuando el resultado de la zona de prueba del subprograma está en 1. La ejecución de una llamada a un subprograma provoca: ● la parada de la exploración de la red en curso, ● la ejecución del subprograma, ● la recuperación de la exploración de la red interrumpida.
Regreso del subprograma	<RETURN>	Reservado a los subprogramas SR, permite el regreso al módulo que llama cuando el resultado de la zona de prueba está en 1.
Parada del programa	<HALT>	Provoca la parada de la ejecución del programa cuando el resultado de la zona de prueba está a 1.

Bloques de función estándar

Los elementos gráficos de los bloques de función DFB se programan en la zona de prueba y ocupan una dimensión de una altura máxima de 16 líneas y una anchura de 3 columnas.

Designación	Grafismo	Funciones
Bloques Temporizador, Contador, Monoestable, Registro, Programador cíclico		Cada uno de los bloques de funciones estándares usa entradas, salidas, entradas/ salidas que permiten enlazarles a los otros elementos gráficos.

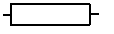
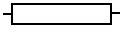
Bloques de función DFB

Los elementos gráficos de los bloques de función DFB se programan en la zona de prueba y ocupan una dimensión de una altura máxima de 16 líneas y una anchura de 3 columnas.

Designación	Grafismo	Funciones
Bloques programables		Cada uno de los bloques de funciones DFB usa entradas, salidas, entradas/salidas que permiten vincularlos a los otros elementos gráficos para los objetos de tipo bits o que se pueden asignar a objetos numéricos o tablas

Bloques de operación

Los elementos gráficos de los bloques de operación se programan en la zona de prueba y ocupan las dimensiones mencionadas más adelante.

Designación	Grafismo	Funciones
Bloque de comparación vertical		Permite comparar 2 operandos, según el resultado, la salida correspondiente pasa a 1. Dimensión: 2 columnas/4 líneas
Bloque de comparación horizontal		Permite comparar 2 operandos, la salida pasa a 1 cuando el resultado se puede verificar (un bloque puede contener hasta 4096 caracteres). Dimensión: 2 columnas/1 línea
Bloque Operación		Realiza las operaciones aritméticas, lógicas... recurre a la sintaxis del lenguaje literal estructurado. (Un bloque puede contener hasta 4096 caracteres). Dimensión: 4 columnas/1 línea

Reglas de programación de una red de contactos

Generalidades

La programación de una red de contactos se efectúa con la ayuda de los elementos gráficos, respetando las siguientes reglas de programación.

Reglas de programación

Los elementos gráficos simples de prueba y de acción ocupan cada uno una celda en el seno de una red.

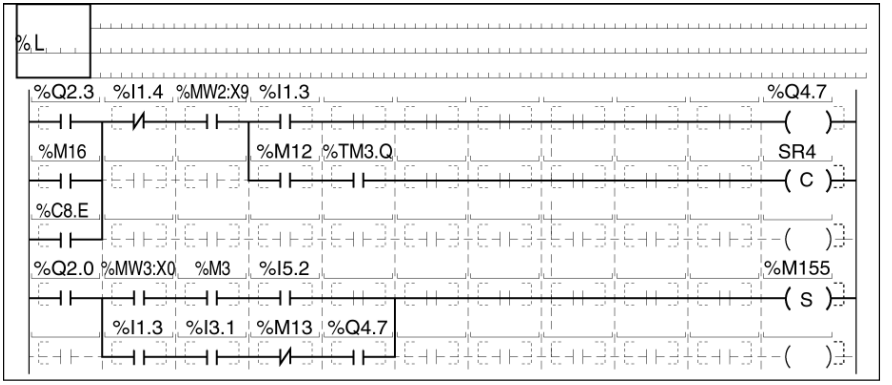
Cualquier línea de contactos empieza en la línea de potencial izquierda y debe terminar en la línea de potencial derecha.

Las pruebas se sitúan siempre en las columnas de la 1 a la 10.
Las acciones siempre se colocan en la columna 11.

El sentido de circulación de la corriente es el siguiente:
- para las conexiones horizontales, de la **izquierda hacia la derecha**,
- para las conexiones verticales, en ambos sentidos.

Ejemplo de red de contactos

La siguiente ventana muestra un ejemplo de red de contactos.



Regla de programación de los bloques de función

Generalidades

Los bloques de función estándar se colocan en la zona de prueba de las redes de contactos.

Reglas de programación de los bloques de función.

Sea cual sea el tipo de bloque de función utilizado, debe estar obligatoriamente vinculado en la entrada de la barra de potencial izquierda, directamente o a través de otros elementos gráficos.

- **salidas "en el aire":** no es necesario enlazar con otros elementos gráficos las salidas de los bloques de función,
- **salidas comprobables:** las salidas de los bloques de función son accesibles para el usuario en forma de objeto bit.

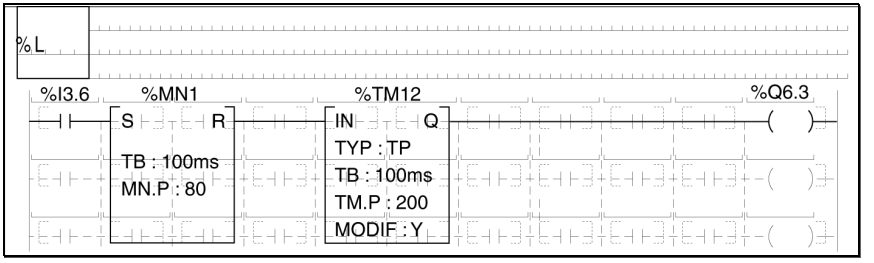
Las variables internas de los bloques y las salidas gráficas son objetos explotables a distancia desde otra parte del programa.

Las entradas no conectadas de los bloques de función estándar se ponen a 0.

Al igual que para los elementos gráficos del tipo de contactos, pueden efectuarse combinaciones de los bloques de función.

Ejemplo de una red de contactos

La ilustración siguiente muestra un ejemplo de una red de contactos que contiene 2 bloques de función.



Reglas de programación de los bloques de operación

Generalidades

Los bloques de comparación se sitúan en la zona de prueba y los bloques de operación se colocan en la zona de acción.

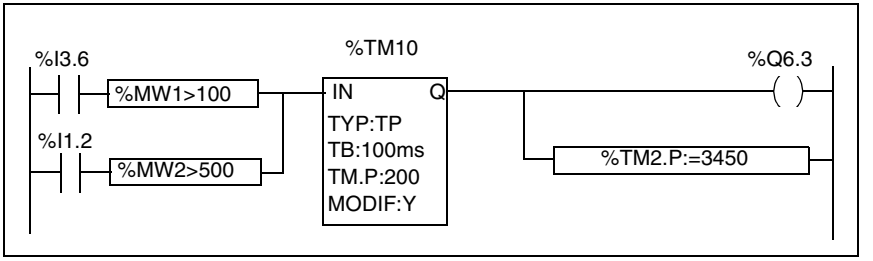
Reglas de programación de los bloques de operación

Sea cual sea el tipo de bloque de operación utilizado, debe estar obligatoriamente vinculado a la entrada de la barra de potencial izquierda, directamente o a través de otros elementos gráficos.

Al igual que para los elementos gráficos del tipo de contactos, pueden efectuarse combinaciones de los bloques de función y operación.

Ejemplo de bloques de operación

La ilustración siguiente muestra un ejemplo de una red de contactos que contiene 2 bloques de comparación y un bloque de operación.



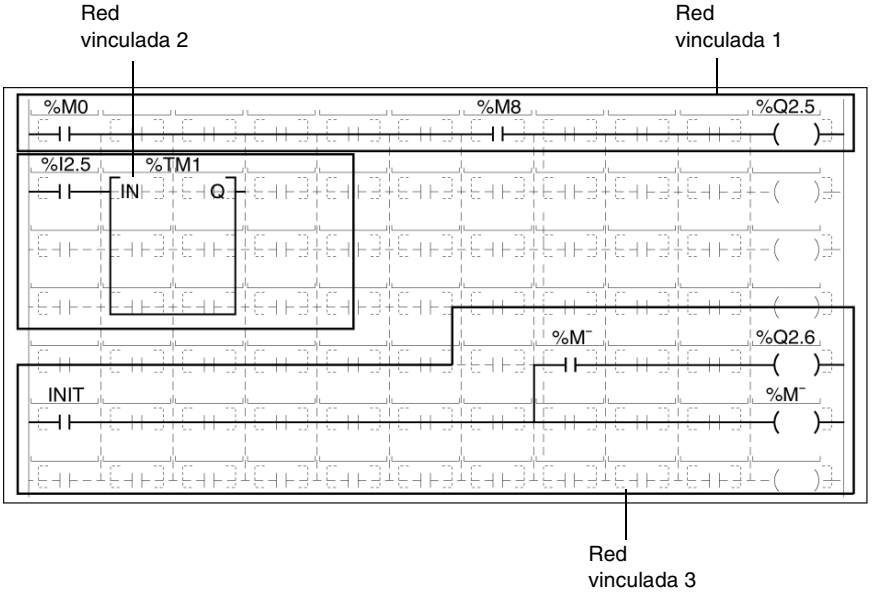
Ejecución de una red de contactos

Red vinculada

Una red vinculada contiene elementos gráficos, todos ellos enlazados entre sí por elementos de unión (fuera de la barra de potencial), aunque independientes de los otros elementos gráficos de la red (sin uniones verticales hacia arriba ó hacia abajo en el límite de la red vinculada).

Ilustración de redes vinculadas

La red de contactos que se muestra a continuación está constituida por 3 redes vinculadas.



Regla de ejecución de las redes vinculadas

La primera red vinculada evaluada es aquella en la cual la esquina izquierda está situada en la parte superior izquierda.

Una red vinculada se evalúa en el sentido de la ecuación: evaluación de la red de arriba abajo, línea a línea, y en cada línea de izquierda a derecha.

Si se diera el caso de que se encontrara una unión vertical de convergencia, la subred asociada se evalúa (según la misma lógica) antes de continuar con la evaluación de la red en la cual está englobada.

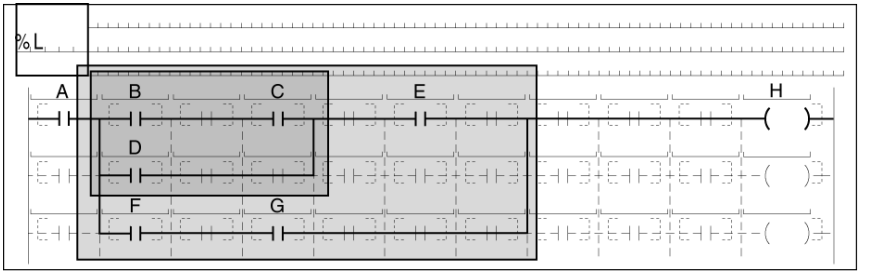
Ejecución de los elementos en una red vinculada

La siguiente tabla describe el orden de ejecución de los elementos en una red vinculada.

Fase	Descripción
1	El sistema evalúa el estado lógico de cada contacto, en función de: ● el valor actual de los objetos internos de aplicación, ● el estado de las entradas de los módulos de entradas/salidas adquirido al iniciarse el ciclo
2	El sistema ejecuta los tratamientos asociados a las funciones, a los bloques de funciones, y a los subprogramas,
3	El sistema actualiza los objetos bits asociados a las bobinas (la actualización de las salidas de los módulos de entradas/salidas se efectúa al final del ciclo),
4	El sistema deriva hacia otra red etiquetada del mismo módulo programa (salto a otra red ->>%Li), regreso al módulo de llamada <RETURN>, o parada del programa <HALT>,

Ejemplo 1: ilustración

El esquema siguiente muestra el orden de ejecución de los elementos gráficos.



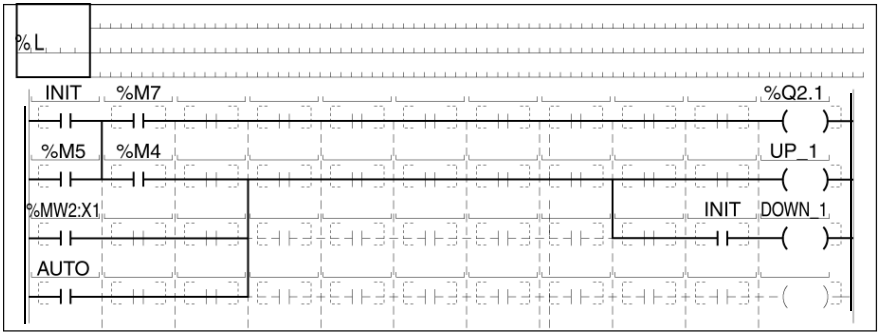
Ejemplo 1: funcionamiento

La tabla que viene a continuación describe la ejecución de los elementos gráficos en la red ilustrada anteriormente.

Fase	Descripción
1	Evaluación de la red hasta que encuentre el primer enlace vertical de convergencia: contactos A, B, C.
2	Evaluación de la primera subred: contacto D,
3	Continuación de la evaluación de la red hasta que se encuentre el segundo enlace vertical de convergencia: contacto E,
4	Evaluación de la segunda subred: contactos F y G,
5	Evaluación de la bobina H.

Ejemplo 2: ilustración

El esquema siguiente muestra el orden de ejecución de los elementos gráficos.



Ejemplo 2: funcionamiento

La tabla que viene a continuación describe la ejecución de los elementos gráficos en la red ilustrada anteriormente.

Fase	Descripción
1	bobina 1: INIT, %M5, %M7, %Q2.1,
2	bobina 2: %M4, %MW2:X1,AUTO, UP_1,
3	Bloque de operación

Lenguaje lista de instrucciones

7

Presentación

Generalidades Este capítulo describe las reglas de programación en lenguaje lista de instrucciones.

Contenido: Este capítulo contiene los siguiente apartados:

Apartado	Página
Presentación general del lenguaje lista de instrucciones	134
Estructura de un programa de lista de instrucciones	135
Etiqueta de una frase en lenguaje de lista de instrucciones	136
Comentario de una frase en lenguaje de lista de instrucciones	137
Presentación de las instrucciones en lenguaje de lista de instrucciones	138
Regla de empleo de los paréntesis en el lenguaje de lista de instrucciones.	143
Descripción de las instrucciones MPS, MRD y MPP	145
Principios de programación de los bloques de función definidos con anterioridad	147
Reglas de ejecución de un programa en lista de instrucciones	149

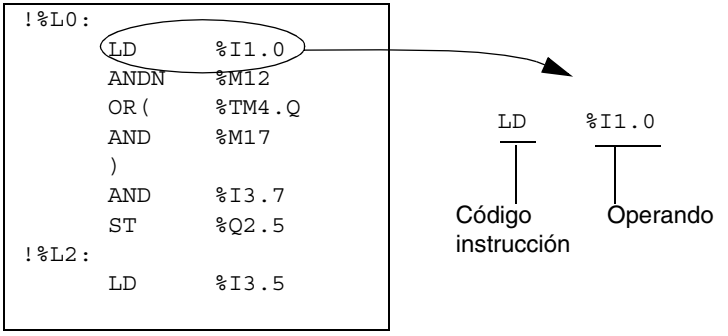
Presentación general del lenguaje lista de instrucciones

Generalidades

Una sección escrita en lenguaje de lista de instrucciones está constituida por una serie de instrucciones ejecutadas secuencialmente por el autómata.

Ilustración de un programa

La siguiente ilustración muestra un programa de lista de instrucciones PL7 y el detalle de una instrucción.



Composición de una instrucción

Esta tabla describe los componentes de una instrucción.

Elemento	Función
Código de instrucción	El código de instrucción determina la operación que debe ejecutarse. Hay dos tipos de códigos de instrucciones: ● **prueba**, en la cual figuran las condiciones necesarias para una acción (ej.: `LD`, `AND`, `OR`...), ● **acción**, que confirma el resultado consecutivo a un enlace de prueba. (ej.: `ST`, `STN`, `R`, ...).
Operando	Una instrucción actúa sobre un operando. Este operando puede ser: ● una entrada/salida del autómata (botones - pulsadores, detectores, relés, indicadores...), ● una función de automatismo (temporizadores, contadores...,), ● una operación aritmética y lógica o una operación de transferencia, ● una variable interna del autómata.

Estructura de un programa de lista de instrucciones

Generalidades

Al igual que en el lenguaje de contactos, las instrucciones se organizan en secuencia de instrucciones (equivalente a una red de contactos) llamadas frases.

Ejemplo de frase

La siguiente ilustración presenta una frase de lista de instrucciones PL7.

```
!(*Espera de secado*)  ———— 1
%L2:                   ———— 2
      LD      %I1.0
      AND     %M10     ———— 3
      ST      %Q2.5
```

Descripción de una frase

Cada una de las frases, que empieza con un signo de exclamación (que se establece automáticamente), incluye los elementos siguientes.

Variable	Elemento	Función
1	Comentario	Inicia una frase (opcional).
2	Etiqueta	Identifica una frase (opcional).
3	Instrucciones	De una a varias instrucciones de prueba, cuyo resultado se aplica a una o varias instrucciones de acción. Una instrucción ocupa como máximo una línea

Etiqueta de una frase en lenguaje de lista de instrucciones

Generalidades

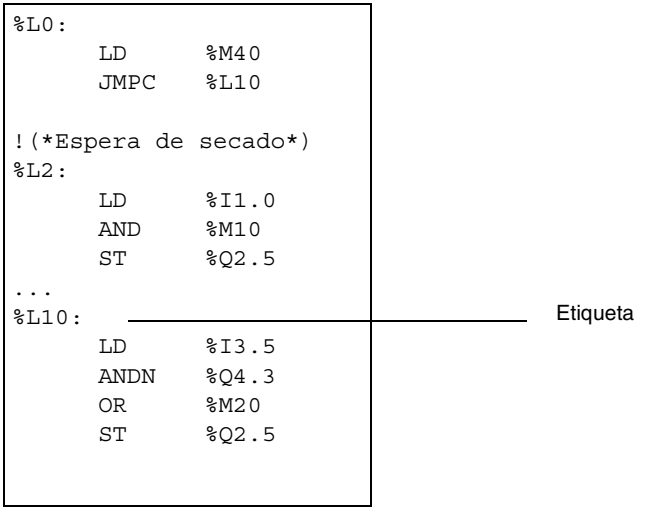
La etiqueta permite identificar una frase en una entidad de programa (programa principal, subprograma, ...). Es opcional.

Sintaxis

Esta etiqueta tiene la siguiente sintaxis: %Li con i comprendida entre 0 y 999. Se sitúa al inicio de una frase.

Ilustración

El programa que se muestra a continuación ilustra el uso de una etiqueta.



Reglas

Una misma etiqueta sólo puede asignarse a una sola frase en el seno de una misma entidad de programa.

Es necesario etiquetar una frase para permitir un enlace después de un salto de programa.

El orden de las variables de las etiquetas no importa. Lo que le sistema tienen en cuenta durante el recuento es el orden de introducción de las frases.

Comentario de una frase en lenguaje de lista de instrucciones

Generalidades

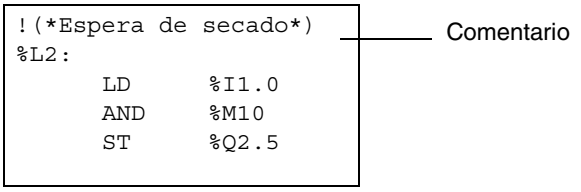
El comentario facilita la interpretación de una frase que le afecta. Es opcional.

Sintaxis

El comentario se puede integrar al principio de una frase y ocupará como máximo 3 líneas (o sea, 222 caracteres alfanuméricos), encuadrados por una parte y por la otra con los caracteres (* y *).

Ilustración

La siguiente ilustración identifica la posición del comentario en una frase.



Reglas

Los comentarios se muestran sólo a partir de la primera línea de la frase.

Si se suprimiera una frase, el comentario que llevase asociado también se suprimiría.

Los comentarios se almacenan en el autómata y el usuario podrá acceder a los mismos en todo momento. Según esto, consumen la memoria de programa.

Presentación de las instrucciones en lenguaje de lista de instrucciones

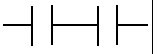
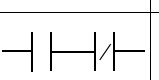
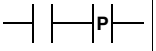
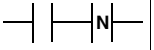
Generalidades

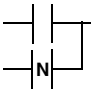
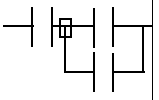
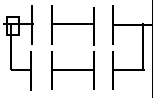
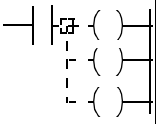
El lenguaje lista de instrucciones incluye las instrucciones:

- de prueba
- de acción
- en cuanto al bloque de función
- digitales

Instrucciones de prueba

La tabla siguiente describe las instrucciones de prueba del lenguaje de lista de instrucciones.

Designación	Grafismo equivalente	Funciones
LD		El resultado booleano es igual al estado del operando.
LDN		El resultado booleano es igual al estado inverso del operando.
LDR		El resultado booleano pasa a 1 cuando se detecta el paso de 0 a 1 del operando (flanco ascendente).
LDF		El resultado booleano pasa a 1 cuando se detecta el paso de 1a 0 del operando (flanco descendente).
AND		El resultado booleano es igual a la Y lógica entre el resultado booleano de la instrucción precedente y el estado del operando.
ANDN		El resultado booleano es igual a la Y lógica entre el resultado booleano de la instrucción precedente y el estado inverso del operando.
ANDR		El resultado booleano es igual a la Y lógica entre el resultado booleano de la instrucción precedente y la detección de un flanco ascendente del operando (1 = flanco ascendente).
ANDF		El resultado booleano es igual a la Y lógica entre el resultado booleano de la instrucción precedente y la detección de un flanco descendente del operando (1= flanco descendente).
OR		El resultado booleano es igual a la Ó lógica entre el resultado booleano de la instrucción precedente y el estado del operando.
ORN		El resultado booleano es igual a la Ó lógica entre el resultado booleano de la instrucción precedente y el estado inverso del operando.
ORR		El resultado booleano es igual a la Ó lógica entre el resultado booleano de la instrucción precedente y la detección de un flanco ascendente del operando (1= flanco ascendente).

Designación	Grafismo equivalente	Funciones
ORF		El resultado booleano es igual a la Y lógica entre el resultado booleano de la instrucción precedente, el estado del operando y la detección de un flanco descendente del operando (1= flanco descendente).
AND(		Y lógica (8 niveles de paréntesis)
OR(		O lógica (8 niveles de paréntesis)
XOR, XORN, XORR, XORF	-	O exclusivo
MPS MRD MPP		Orientación hacia las bobinas.
N	-	Negación

Instrucciones de acción

La siguiente tabla describe las instrucciones de prueba del lenguaje de lista de instrucciones.

Designación	Grafismo	Funciones
ST		El operando asociado toma el valor del resultado de la zona de prueba.
STN		El operando asociado toma el valor inverso del resultado de la zona de prueba.
S		El operando asociado se sitúa en 1 cuando el resultado de la zona de prueba está a 1.
R		El operando asociado se sitúa a 0 cuando el resultado de la zona de prueba está a 1.
JMP	-	Permite un enlace incondicional a una frase etiquetada, hacia arriba ó hacia abajo.
JMPC	-	Permite un enlace condicionado a un resultado booleno a 1, de una frase etiquetada arriba o abajo.
JMPCN	-	Permite un enlace condicionado a un resultado booleno a 0, de una frase etiquetada arriba o abajo.
SRn	-	Enlace al inicio de un subprograma.
RET	-	Regreso del subprograma.
RETC	-	Regreso del subprograma condicionado a un resultado booleao a 1.
RETCN	-	Regreso del subprograma condicionado a un resultado booleano a 0.
END	-	Fin del programa.
ENDC	-	Fin del programa condicionado a un resultado booleano a 1.
ENDCN	-	Fin del programa condicionado a un resultado booleano a 0.

Instrucciones en cuanto al bloque de función

La siguiente tabla describe las instrucciones de prueba del lenguaje de lista de instrucciones.

Designación	Grafismo	Funciones
Bloques Temporizador, Contador, Monoestable, Registro, Programador cíclico		Para cada uno de los bloques de función estándares, podemos encontrar las instrucciones que permiten dirigir el bloque. Una forma estructurada permite conectar directamente las entradas/salidas de los bloques.

Instrucciones digitales

La siguiente tabla describe las instrucciones de prueba del lenguaje de lista de instrucciones.

Designación	Instrucciones	Funciones
Elemento de prueba	LD[.....] AND[.....] OR[.....]	Permite comparar los 2 operandos, la salida pasa a 1 cuando el resultado se comprueba. **Ejemplo**: LD[%MW10<1000] Resultado a 1 cuando %MW10<1000.
Elemento de acción	[.....]	Ejecutan las operaciones aritméticas, lógicas... Usan la sintaxis del lenguaje literal estructurado. **Ejemplo**: [%MW10:=%MW0+100] El resultado de la operación %MW0+100 se sitúa en la palabra interna %MW10.

Regla de empleo de los paréntesis en el lenguaje de lista de instrucciones.

Generalidades

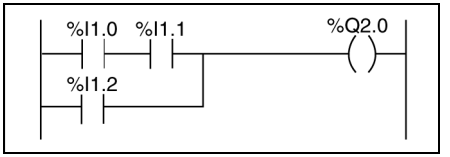
Las instrucciones AND y OR pueden usar paréntesis.
Estos paréntesis permiten elaborar esquemas de contactos de forma simple.

Principio

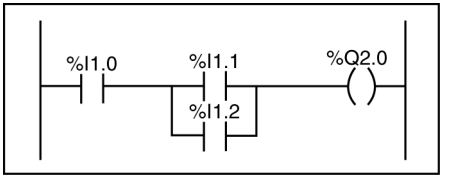
La obertura de paréntesis se asocia a la instrucción AND u OR.
El cierre de paréntesis es una instrucción, es obligatorio para cada paréntesis abierto.

Ejemplo: AND(

Los 2 programas siguientes ilustran el uso del paréntesis.



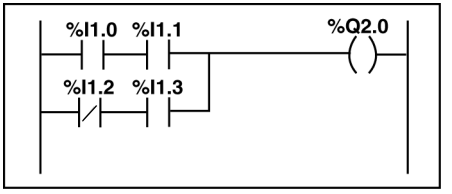

```
LD      %I1.0
AND     %I1.1
OR      %I1.2
ST      %Q2,0
```




```
LD      %I1.0
AND (   %I1.1
OR      %I1.2
)
ST      %Q2,0
```

Ejemplo: OR(

El programa siguiente ilustra el uso del paréntesis.




```
LD      %I1.0
AND     %I1.1
OR (N   %I1.2
AND     %I1.3
)
ST      %Q2.0
```

Asociación de paréntesis a los modificadores

Los "modificadores" siguientes pueden asociarse a los paréntesis.

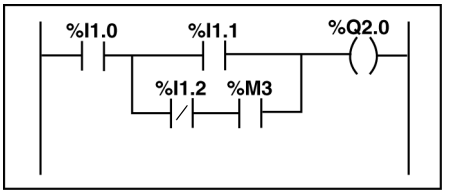
Código	Función	Ejemplo
N	Negación	AND(N
F	Flanco descendente (Falling edge)	AND(F
R	Flanco ascendente (Rising edge)	OR(R
[	Comparación	OR([%MW0>100]

Entrelazamiento de paréntesis

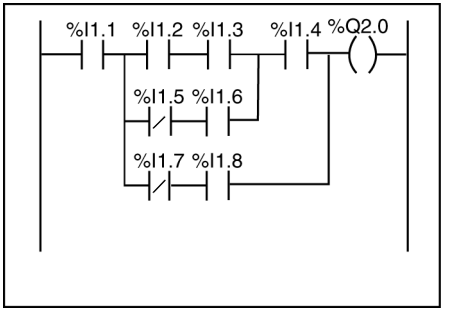
Se pueden entrelazar hasta 8 niveles de paréntesis.
Se deben seguir las siguientes reglas:

- Cada paréntesis abierto debe cerrarse obligatoriamente
- Las etiquetas `%Li:` no deben colocarse en las expresiones entre paréntesis, ni las instrucciones de salto `JMP` y de llamada a un subprograma `SRi`,
- Las instrucciones de asignación `ST`, `STN`, `S` y `R` no deben programarse entre paréntesis.

Ejemplo:
Los programas que se muestran a continuación ilustran el uso del entrelazamiento de los paréntesis.




```
LD      %I1.0
AND(    %I1.1
OR(N    %I1.2
AND     %M3
)
)
ST      %Q2.0
```




```
LD      %I1.1
AND(    %I1.2
AND     %I1.3
OR(N    %I1.5
AND     %I1.6
)
AND     %I1.4
OR(N    %I1.7
AND     %I1.8
)
)
ST      %Q2.0
```

Descripción de las instrucciones MPS, MRD y MPP

Generalidades

Los 3 tipos de instrucciones permiten tratar las desviaciones hacia las bobinas.

Estas instrucciones usan una memoria intermedia llamada pila que puede almacenar hasta 3 informaciones booleanas...

Nota: Estas instrucciones no pueden usarse en el seno de una expresión entre paréntesis

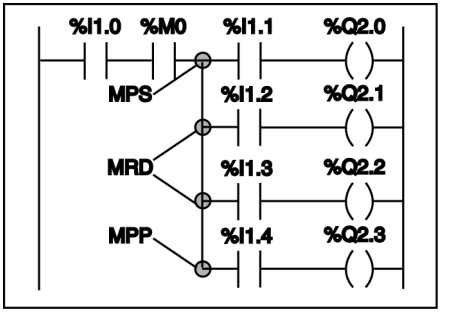
Función

La tabla siguiente describe la función de cada una de las instrucciones

Instrucción	Función
MPS (Memory PuSh)	Esta instrucción tiene por misión almacenar el resultado de la última instrucción de prueba en la parte superior de la pila y de desplazar los otros valores hacia el fondo de la misma.
MRD (Memory ReaD)	Esta instrucción lee la parte superior de la pila.
MPP (Memory PoP)	Esta instrucción tiene como misión leer, limpiar la parte superior de la pila y desplazar los otros valores hacia la parte superior de la pila.

Ejemplo 1

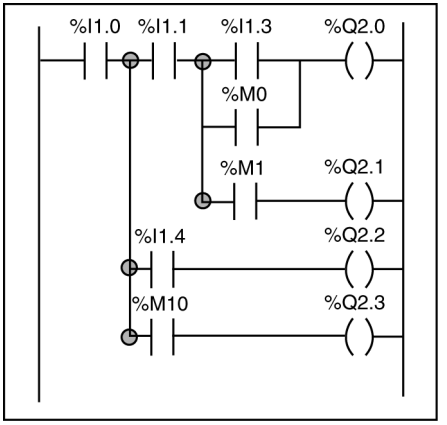
Este ejemplo ilustra el uso de las instrucciones MPS, MRD, y MPP.

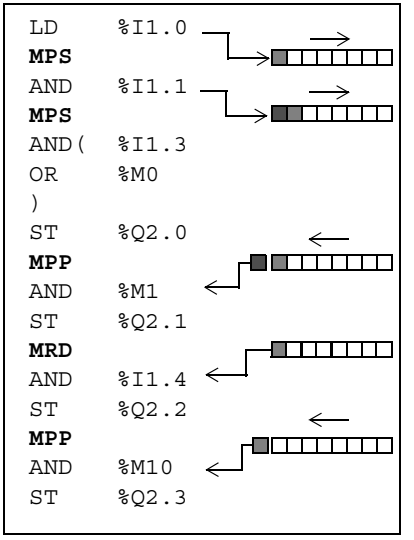



```
LD      %I1.0
AND     %M0
MPS
AND     %I1.1
ST      %Q2.0
MRD
AND     %I1.2
ST      %Q2.1
MRD
AND     %I1.3
ST      %Q2.2
MPP
AND     %I1.4
ST      %Q2.3
```

Ejemplo 2

Este ejemplo ilustra el funcionamiento de las instrucciones MPS, MRD, y MPP.





Principios de programación de los bloques de función definidos con anterioridad

Generalidades

Los bloques de funciones de automatismos se pueden programar de 2 formas distintas:

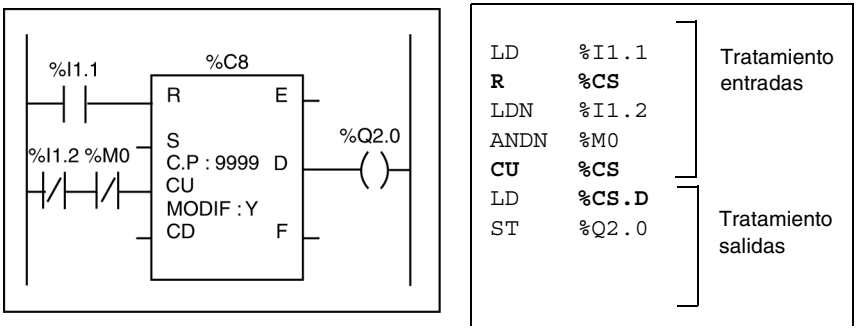
- con instrucciones específicas de cada bloque de función (ej.: `CU %Ci`), esta forma es la más sencilla y la más directa,
- con instrucciones de estructuración de bloque `BLK` ,`OUT_BLK`, `END_BLK`.

Principio de programación directa

La instrucciones dirigen las entradas de bloques (ej.: `CU`). Las salidas son accesibles bajo la forma de bit (ej.: `%C8.D`).

Ejemplo:

Este ejemplo ilustra la programación directa de un bloque de función contador.

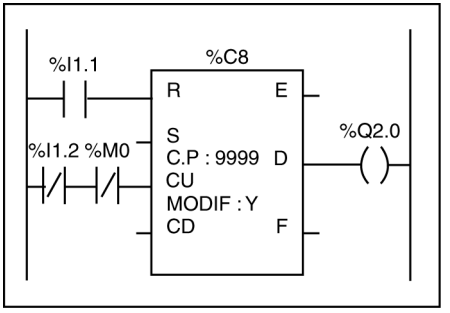


Principio de programación estructurado

Este tipo de programación usa una serie de instrucciones encuadradas por las instrucciones:

- BLK señala el inicio del bloque
- OUT_BLK permite conectar directamente las salidas del bloque
- END_BLK señala el final del bloque

Ejemplo:
Este ejemplo ilustra la programación estructurada de un bloque de función contador.




```
BLK    %C8
LD     %I1.1
R
LDN    %I1.2
ANDN   %M0
CU
OUT_BLK
LD     D
ST     %Q2.0
END_BLK
```

Tratamiento entradas

Tratamiento salidas

Nota: Este principio de programación estructurado que precisa de las instrucciones complementarias BLK, OUT_BLK y END_BLK pide volúmenes de memoria superiores en relación con la programación directa. Sin embargo, se puede usar, si se quiere mantener un parecido con los programas reversibles para nanoautómatas TSX 07.

Reglas de ejecución de un programa en lista de instrucciones

Principio

La ejecución de un programa en lista de instrucciones se efectúa de forma secuencial, instrucción por instrucción.

La primera instrucción de una secuencia de instrucciones siempre debe ser, bien una instrucción`LD` o bien una instrucción incondicional (ej.: `JMP`).

Cada instrucción (salvo `LD` y las instrucciones incondicionales) emplea el resultado booleano de la instrucción precedente.

Ejemplo 1

El programa que se muestra a continuación describe la ejecución completa de una frase.

`LD`	`%I1.1`	resultado = estado del bit %I1.1
`AND`	`%M0`	resultado = Y del resultado booleano que precede y del estado del bit %M0
`OR`	`%M10`	resultado = O del resultado booleano que precede y del estado del bit %M100
`ST`	`%Q2.0`	%Q2.0 toma el estado del resultado booleano que precede

Ejemplo 2

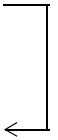
Los paréntesis permiten modificar la orden de tener en cuenta los resultados booleanos:

`LD`	`%I1.1`	resultado = estado del bit %I1.1
`AND`	`%M0`	resultado = Y del resultado booleano que precede y del estado del bit %M0
`OR(`	`%M10`	resultado = estado del bit %M10
`AND`	`%I1.2`	resultado = Y del resultado booleano que precede y del estado del bit %M10
`)`		
`ST`	`%Q2.0`	%Q2.0 toma el estado del resultado booleano que precede

Ejemplo 3

El escalonamiento de las instrucciones puede modificarse por las instrucciones de salto `JMP` de llamada a subprograma.

```
!       LD      %M0
        JMPC    %L10
!       LD      %I1.1
        AND     %M10
        ST      %Q2.0
!       %L10:
        LD      %I1.3
        AND     %M20
        ......
```



Salto a la etiqueta %L10 si %M0=1

Lenguaje literal estructurado

8

Presentación

Contenido del capítulo

Este capítulo describe las reglas de programación en lenguaje literal estructurado.

Contenido:

Este capítulo contiene los siguiente apartados:

Apartado	Página
Presentación del lenguaje literal estructurado	152
Estructura de un programa en lenguaje literal estructurado	153
Etiqueta de una frase en lenguaje literal estructurado	154
Comentario de una frase en lenguaje literal estructurado	155
Instrucciones sobre objetos bits	156
Instrucciones aritméticas y lógicas	157
Instrucciones para las tablas y cadenas de caracteres	159
Instrucciones de conversiones numéricas	162
Instrucciones del programa e instrucciones específicas	163
Estructura de control condicional IF...THEN	165
Estructura de control condicional WHILE...END_WHILE	167
Estructura de control condicional REPEAT...END_REPEAT	168
Estructura de control condicional FOR...END_FOR	169
Instrucción de salida de bucle EXIT	170
Reglas de ejecución de un programa literal estructurado	171

Presentación del lenguaje literal estructurado

Generalidades

El lenguaje literal estructurado es un lenguaje evolucionado del tipo algorítmico adaptado especialmente a la programación de funciones aritméticas complejas, manipulaciones de tablas y gestiones de mensajes.

Permite ejecutar programas para escritura de líneas de programación, compuestas de caracteres alfanuméricos.

Límite de uso

Este lenguaje se puede usar con los programas PL7 Micro, PL7 Junior y PL7 Pro en los autómatas Premium y Micro.
En la versión PL7 Pro, este lenguaje permite la creación de bloques de funciones usuario DFB en los autómatas Premium.

Ilustración de un programa

La siguiente ilustración muestra un programa en lenguaje estructurado PL7.

```
!     (* Búsqueda del primer elemento no nulo en una
      tabla de 32 palabras, determinación de su valor
      (%MW10), de su rango (%MW11). Esta búsqueda
      se efectúa si %M0 está en 1, %M1 se pone a 1 si
      hay un elemento no nulo, si no se pone a 0*)

      IF %M0 THEN
             FOR %MW99:=0 TO 31 DO
                  IF %MW100[%MW99]<>0 THEN
                       %MW10:=%MW100[%MW99];
                       %MW11:=%MW99;
                       %M1:=TRUE;
                       EXIT;    (*Salida del bucle*)
                  ELSE
                       %M1:=FALSE;
                  END_IF;
             END_FOR;
      ELSE
             %M1:=FALSE;
      END_IF;
```

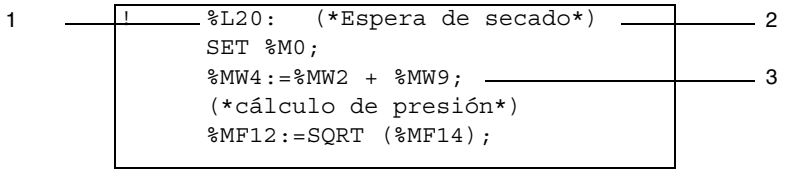
Estructura de un programa en lenguaje literal estructurado

Generalidades

Una sección de programa literal se organiza en frases.
Una frase literal equivale a una red de contactos en lenguaje de contactos.

Ejemplo de frase

La ilustración siguiente muestra una frase en lenguaje estructurado PL7.



Descripción de una frase

Cada una de las frases, que empieza con un signo de exclamación (que se establece automáticamente), incluye los elementos siguientes.

Variable	Elemento	Función
1	Etiqueta	Identifica una frase.
2	Comentario	Muestra una frase.
3	Instrucciones	De una a varias instrucciones separadas por ";".

Nota: Cada uno de estos elementos es opcional, es decir, que es posible tener una frase vacía, una frase compuesta sólo por comentarios ó únicamente de una etiqueta.

Etiqueta de una frase en lenguaje literal estructurado

Función

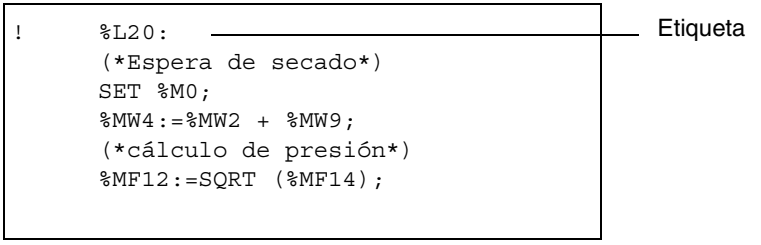
La etiqueta permite identificar una frase en una entidad de programa (programa principal, subprograma, ...). Es opcional.

Sintaxis

Esta etiqueta tiene la siguiente sintaxis: %Li: con i comprendida entre 0 y 999. Se sitúa al inicio de una frase.

Ilustración

El programa que se muestra a continuación ilustra el uso de una etiqueta



Reglas

Una misma etiqueta sólo puede asignarse a una sola frase en el seno de una misma entidad de programa.

Es necesario etiquetar una frase para permitir un enlace después de un salto de programa.

El orden de las variables de las etiquetas no importa, el sistema tiene en cuenta el orden de introducción de las frases durante el recuento.

Comentario de una frase en lenguaje literal estructurado

Función

El comentario facilita la interpretación de una frase a la que se asigna. Es opcional.

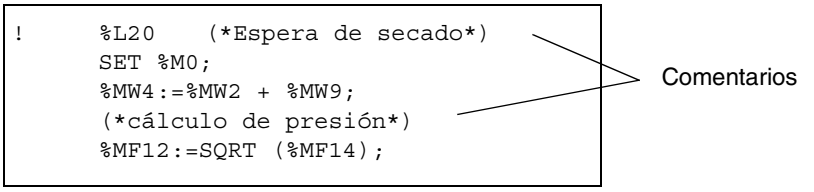
Sintaxis

El comentario puede integrarse en cualquier lugar de la frase, siendo ilimitado el número de comentarios por cada frase.

Un comentario se enmarca por una parte y por otra con los caracteres (* y *).

Ilustración

La siguiente ilustración identifica la posición del comentario en una frase.



Reglas

- En un comentario, todos los caracteres quedan autorizados.
- El número de caracteres se limita a 256 por comentario.
- Los comentarios imbricados están prohibidos.
- Un comentario puede contener varias líneas.

Los comentarios se almacenan en el autómata y son accesibles para el usuario en todo momento. Según esto, consumen la memoria de programa.

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

Instrucciones aritméticas y lógicas

Aritmética entera en palabras y en palabras dobles

Las instrucciones siguientes se aplican en los objetos palabras y palabras dobles.

Designación	Función
+, -, *, /	Suma, Resta, Multiplicación, División entera
REM	Resto de la división entera
SQRT	Raíz cuadrada entera
ABS	Valor absoluto
INC	Incremento
DEC	Disminución

Aritmética sobre flotantes

Las instrucciones siguientes se aplican a los objetos flotantes.

Designación	Función
+, -, *, /	Suma, Resta, Multiplicación, División
SQRT	Raíz cuadrada
ABS	Valor absoluto
TRUNC	Parte entera
LOG	Logaritmo en base 10
LN	Logaritmo neperiano
EXP	Exponencial natural
EXPT	Exponencial de un real por un real
COS	Coseno de un valor en radianes
SIN	Seno de un valor en radianes
TAN	Tangente de un valor en radianes
ACOS	Arcocoseno (resultado entre 0 y 2 p)
ASIN	Arcoseno (resultado entre –p/2 y +p/2)
ATAN	Arcotangente (resultado entre - p/2 y + p/2)
DEG_TO_RAD	Conversión de grados en radianes
RAD_TO_DEG	Conversión de radianes en grados

Instrucciones lógicas en palabras y en palabras dobles

Las instrucciones siguientes se aplican en los objetos palabras y palabras dobles.

Designación	Función
AND	Y lógica
OR	O lógica
XOR	O lógica exclusiva
NOT	Complemento lógico
SHL	Desplazamiento lógico a la izquierda
SHR	Desplazamiento lógico a la derecha
ROL	Desplazamiento lógico circular a la izquierda
ROR	Desplazamiento lógico circular a la derecha

Comparaciones numéricas en palabras, palabras dobles y flotantes

Las instrucciones siguientes se aplican en los objetos palabras, palabras dobles y flotantes.

Designación	Función
<	Estrictamente inferior a
>	Estrictamente superior a
<=	Inferior o igual a
>=	Superior o igual a
=	Igual a
<>	Diferente de

Instrucciones para las tablas y cadenas de caracteres

Instrucciones para las tablas de palabras y palabras dobles

Las instrucciones siguientes se aplican en las tablas de palabras y palabras dobles.

Designación	Función
Tabla: = Tabla	Asignación entre dos tablas
Tabla: = Palabra	Inicialización de una tabla
+, -, *, /, REM	Operaciones aritméticas entre tablas
+, -, *, /, REM	Operaciones aritméticas entre expresiones y tablas
SUM	Suma de los elementos de una tabla
EQUAL	Comparación entre dos tablas
NOT	Complemento lógico de una tabla
AND, OR, XOR	Operaciones lógicas entre dos tablas
AND, OR, XOR	Operaciones lógicas entre expresiones y tablas
FIND_EQW, FIND_EQD	Búsqueda del primer elemento igual a un valor
FIND_GTW, FIND_GTD	Búsqueda del primer elemento superior a un valor
FIND_LTW, FIND_LTD	Búsqueda del primer elemento inferior a un valor
MAX_ARW, MAX_ARD	Búsqueda del valor máximo en una tabla
MIN_ARW, MIN_ARD	Búsqueda del valor mínimo en una tabla
OCCUR_ARW, OCCUR_ARD	Número de ocurrencias de un valor en una tabla
SORT_ARW, SORT_ARD	Ordenar una tabla por orden creciente o decreciente
ROL_ARW, ROL_ARD	Diferencia circular a la izquierda de una tabla
ROR_ARW, ROR_ARD	Diferencia circular a la derecha de una tabla
FIND_EQWP,FIND_EQDP	Búsqueda del primer elemento igual a un valor desde una posición
LENGTH_ARW, LENGTH_ARD	Cálculo de la longitud de una tabla

Instrucciones para las tablas de flotantes

Las instrucciones siguientes se aplican a las tablas de flotantes.

Designación	Función
Tabla: = Tabla	Asignación entre dos tablas
Tabla: = Flotante	Inicialización de una tabla
SUM_ARR	Suma de los elementos de una tabla
EQUAL_ARR	Comparación entre dos tablas
FIND_EQR	Búsqueda del primer elemento igual a un valor
FIND_GTR	Búsqueda del primer elemento superior a un valor
FIND_LTR	Búsqueda del primer elemento inferior a un valor
MAX_ARR	Búsqueda del valor máximo en una tabla
MIN_ARR	Búsqueda del valor mínimo en una tabla
OCCUR_ARR	Número de ocurrencias de un valor en una tabla
SORT_ARR	Ordenar una tabla en orden creciente o decreciente
ROL_ARR	Diferencia circular a la izquierda de una tabla
ROR_ARR	Diferencia circular a la derecha de una tabla
LENGTH_ARR	Cálculo de la longitud de una tabla

Instrucciones sobre cadenas de caracteres

Las instrucciones siguientes se aplican a las cadenas de caracteres.

Designación	Función
STRING_TO_INT	Conversión ASCII í Binario (palabra de formato simple)
STRING_TO_DINT	Conversión ASCII í Binario (palabra de formato doble)
INT_TO_STRING	Conversión Binario í (palabra de formato simple) ASCII
DINT_TO_STRING	Conversión Binario í (palabra de formato doble) ASCII
STRING_TO_REAL	Conversión ASCII í Flotante
REAL_TO_STRING	Conversión Flotante í ASCII
<, >, <=, >=, =, <>	Comparación alfanumérica
FIND	Posición de una subcadena
EQUAL_STR	Posición del primer carácter diferente
LEN	Longitud de una cadena de caracteres
MID	Extracción de una subcadena
INSERT	Inserción de una subcadena
DELETE	Eliminación de una subcadena
CONCAT	Concatenación de dos cadenas
REPLACE	Reemplazo de una cadena
LEFT	Inicio de cadena
RIGHT	Final de cadena

Instrucciones de conversiones numéricas

Instrucciones de conversiones numéricas

Las instrucciones efectúan conversiones de bits, palabras, palabras dobles y flotantes.

Designación	Función
BCD_TO_INT	Conversión BCD í Binaria
INT_TO_BCD	Conversión Binaria í BCD
GRAY_TO_INT	Conversión Gray í Binaria
INT_TO_REAL	Conversión de un entero de formato simple en flotante
DINT_TO_REAL	Conversión de un entero de formato doble en flotante
REAL_TO_INT	Conversión de un flotante en entero de formato simple
REAL_TO_DINT	Conversión de un flotante en entero de formato doble
DBCD_TO_DINT	Conversión de un número BCD 32 bits en entero de 32 bits
DINT_TO_DBCD	Conversión de un entero de 32 bits en número BCD 32 bits
DBCD_TO_INT	Conversión de un número BCD 32 bits en entero de 16 bits
INT_TO_DBCD	Conversión de un entero de 16 bits en número BCD 32 bits
LW	Extracción de una palabra de peso menos significativo de una palabra doble
HW	Extracción de una palabra de peso más significativo de una palabra doble
CONCATW	Concatenación de 2 palabras simples

Instrucciones del programa e instrucciones específicas

Instrucciones del programa

Las instrucciones siguientes no actúan sobre objetos del lenguaje, sino sobre el desarrollo del programa.

Designación	Función
HALT	Parada de la ejecución del programa
JUMP	Salto a una etiqueta
SRi	Llamada del subprograma
RETURN	Regreso del subprograma
MASKEVT	Enmascaramiento de los sucesos en el autómata
UNMASKEVT	Desenmascaramiento de los sucesos en el autómata

Instrucciones de gestión del tiempo

Las instrucciones siguientes ejecutan operaciones sobre las fechas, las horas y sobre la duración.

Designación	Función
SCHEDULE	Función reloj-calendario
RRTC	Lectura de la fecha del sistema
WRTC	Actualización de la fecha del sistema
PTC	Lectura de la fecha y código de parada
ADD_TOD	Añadido de una duración a una hora del día
ADD_DT	Añadido de una duración a una fecha y hora
DELTA_TOD	Medición de la desviación entre horas del día
DELTA_D	Medición de la desviación entre fechas (sin hora)
DELTA_DT	Medición de la desviación entre fechas (con hora)
SUB_TOD	Volver en el tiempo a una hora
SUB_DT	Volver en el tiempo a una fecha y hora
DAY_OF_WEEK	Lectura del día actual de la semana
TRANS_TIME	Conversión de la duración en fecha
DATE_TO_STRING	Conversión de una fecha en cadena de caracteres
TOD_TO_STRING	Conversión de una hora en cadena de caracteres
DT_TO_STRING	Conversión de una fecha completa en cadena de caracteres
TIME_TO_STRING	Conversión de una duración en cadena de caracteres

Instrucciones "Orphée"

Las instrucciones siguientes son instrucciones específicas del lenguaje Orphée.

Designación	Función
WSHL_RBIT, DSHL_RBIT	Desplazamiento a la izquierda de la palabra con recuperación de bits desplazados
WSHR_RBIT, DSHR_RBIT	Desplazamiento a la derecha de la palabra con extensión de signo y recuperación de los bits desplazados
WSHRZ_C, DSHRZ_C	Desplazamiento a la derecha de la palabra con sustitución por 0 y recuperación de los bits desplazados
SCOUNT	Contaje/descontaje con señalización de rebasamiento
ROLW,ROLD	Diferencia circular izquierda
RORW,RORD	Diferencia circular derecha

Instrucciones de temporización

Esta instrucciones son funciones de temporización destinadas a utilizarse para la programación del código de las DFB.

Designación	Función
FTON	Temporización en la desactivación
FTOF	Temporización en la desactivación
FTP	Temporización de impulsión
FPULSOR	Generador de señales rectangulares

Estructura de control condicional IF...THEN

Función

Esta estructura de control ejecuta una o varias acciones si una condición es verdadera. En la forma general las condiciones pueden ser múltiples.

Forma simple

En la forma simple, la estructura de control tiene la sintaxis y el siguiente funcionamiento.

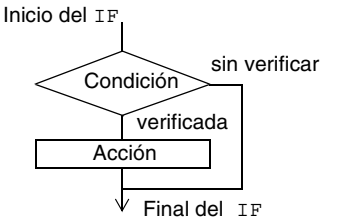
Sintaxis

```
IF condición THEN

    acciones;

END_IF;
```

Funcionamiento



Ejemplo:

```
! (*Acción condicional IF (forma sencilla)*)
  IF %M0 AND %M12 THEN
       RESET %M0;
       INC %MW4;
       %MW10:=%MW8+%MW9;
  END_IF;
```

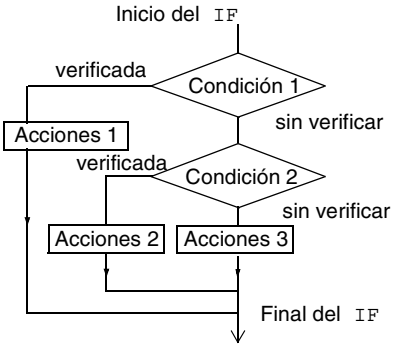
Forma general

En la forma general, la estructura de control tiene la sintaxis y el funcionamiento siguiente.

Sintaxis

```
IF condicional THEN
    acciones 1;
ELSEIF condición 2 THEN
    acciones 2;
ELSE
    acciones 3;
END_IF;
```

Funcionamiento



Ejemplo:

```
! (*Acción condicional IF (forma simple)*)
  IF %M0 AND %M1 THEN
       %MW5:=%MW3+%MW4;
       SET %M10;
  ELSEIF %M0 OR %M1 THEN
       %MW5:=%MW3-%MW4;
       SET %M11;
  ELSE
       RESET %M10;
       RESET %M11;
  END_IF;
```

Regla de programación

- Las condiciones pueden ser múltiples.
- Cada acción representa una lista de instrucciones.
- Algunas estructuras de control IF pueden ser imbricadas.
- El número de ELSIF no tiene límites.
- Existe como máximo una parte ELSE

Estructura de control condicional WHILE...END_WHILE

Función

Esta estructura de control ejecuta una acción repetitiva mientras se verifica una condición.

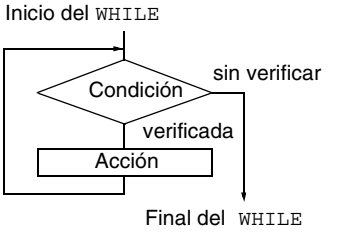
Descripción

La estructura de control tiene la sintaxis y el funcionamiento siguientes.

Sintaxis

```
WHILE condición DO
    acción;
END_WHILE;
```

Funcionamiento



Ejemplo:

```
! (*Acción reiterativa condicional WHILE*)
  WHILE %MW4<12 DO
       INC %MW4;
       SET %M25[%MW4];
  END_WHILE;
```

Regla de programación

- La condición puede ser múltiple.
- La acción representa una lista de instrucciones.
- La prueba de la condición se lleva a cabo antes de ejecutar la acción. Si en el momento de la primera evaluación de la condición, el valor resulta falso, la acción no se ejecuta jamás
- Algunas estructuras de control WHILE pueden ser imbricadas.

Nota: La instrucción EXIT (Véase *Función, p. 170*) permite detener la ejecución del bucle y continuar en la instrucción siguiente el END_WHILE.

Estructura de control condicional REPEAT...END_REPEAT

Función

Esta estructura de control ejecuta una acción repetitiva hasta que se verifique una condición.

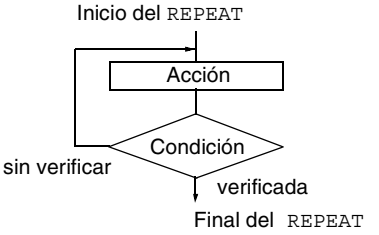
Descripción

La estructura de control tiene la sintaxis y el funcionamiento siguiente:

Sintaxis

```
REPEAT
    acción;
UNTIL condición END_REPEAT;
```

Funcionamiento



Ejemplo:

```
! (*Acción reiterativa condicional REPEAT*)
  REPEAT
       INC %MW4;
       SET %M25[%MW4];
  UNTIL %MW4>12 END_REPEAT;
```

Regla de programación

- La condición puede ser múltiple.
- La acción representa una lista de instrucciones.
- La prueba de la condición se lleva a cabo después de ejecutar la acción. Si en el momento de la primera evaluación de la condición, el valor resulta falso, la acción se ejecuta una vez.
- Algunas estructuras de control REPEAT pueden ser imbricadas.

Nota: La instrucción EXIT (Véase *Función, p. 170*) permite detener la ejecución del bucle y continuar en la instrucción siguiente el END_REPEAT.

Estructura de control condicional FOR...END_FOR

Función

Esta estructura de control realiza un tratamiento durante un cierto número de veces incrementando en 1 un índice en cada bucle.

Descripción

La estructura de control tiene la sintaxis y el funcionamiento siguientes:

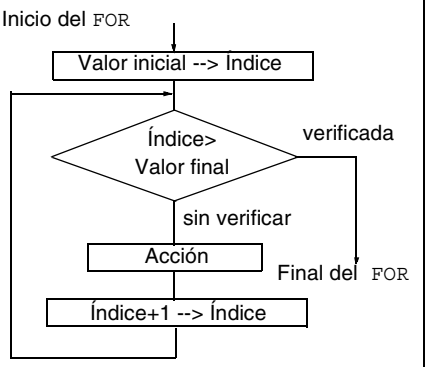
Sintaxis

```
FOR índice:=valor inicial TO
valor final DO

    acción;

END_FOR;
```

Funcionamiento



Ejemplo:

```
! (*Acción repetitiva FOR*)
  FOR %MW4=0 TO %MW23+12 DO
       SET %M25[%MW4];
  END_FOR;
```

Regla de programación

- Cuando el índice es estrictamente superior al valor final, la ejecución continua en la instrucción siguiente el END_FOR.
- El incremento del índice se lleva a cabo automáticamente y no de otra forma.
- La acción representa una lista de instrucciones.
- El valor inicial y el valor final son obligatoriamente expresiones digitales del tipo palabra.
- El índice es forzosamente un objeto del tipo palabra accesible en modo escritura.
- Algunas estructuras de control FOR pueden ser imbricadas.

Nota: La instrucción EXIT (Véase *Función, p. 170*) permite detener la ejecución del bucle y continuar en la instrucción siguiente el END_FOR.

Instrucción de salida de bucle EXIT

Función

Esta instrucción permite detener la ejecución del bucle y continuar en la instrucción siguiente la instrucción de final de bucle.

Regla de programación

- Esta instrucción únicamente puede utilizarse en las acciones de los 3 bucles WHILE, REPEAT o FOR.
- Esta instrucción se incorpora al bucle envolvente más cercano, es decir, que no detiene la ejecución de todos los bucles que le engloban.

Ejemplo

En este ejemplo, la instrucción EXIT permite detener el bucle REPEAT aunque en ningún caso el bucle WHILE.

```
! (*Instrucción de salida del bucle EXIT*)
  WHILE %MW1<124 DO
       %MW2:=0;
       %MW3:=%MW100[%MW1];
       REPEAT
             %MW500[%MW2]:=%MW3+%MW500[%MW2];
             IF(%MW500[%MW2]>32700) THEN
                   EXIT;
             END_IF;
             INC %MW2;
       UNTIL %MW2>25 END_REPEAT;
       INC %MW1;
  END_WHILE;
```

Reglas de ejecución de un programa literal estructurado

Generalidades

La ejecución de un programa literal se efectúa de forma secuencial, instrucción por instrucción respetando las estructuras de control.

En el caso de expresiones aritméticas o booleanas constituidas por varios operadores, las reglas de prioridad se definen entre los diferentes operadores.

Regla de prioridad de los operadores

La tabla siguiente muestra la prioridad para la evaluación de una expresión de más o menos prioridad.

Operador	Símbolo	Prioridad
Paréntesis	(expresión)	Más fuerte
Complemento lógico Inversión - en el operando + en el operando	NOT NOT - +	
Multiplicación División Módulo	* / REM	
Suma Resta	+ -	
Comparaciones	<,>,<=,>=	
Comparaciones de igualdad Comparaciones de desigualdad	= <>	
Y lógica Y booleana	AND AND	
O exclusiva lógica O exclusiva booleana	XOR XOR	
O lógica O booleana	OR OR	Menos significativo

Nota: Cuando exista conflicto entre dos operadores del mismo nivel de prioridad, el primer operador tendrá preferencia (la designación se efectúa de izquierda a derecha).

Ejemplo 1

En el siguiente ejemplo, el NOT se aplica al %MW3 después el resultado se multiplica por 25. La suma de %MW10 y %MW12 se calcula a continuación, después la Y lógica entre el resultado de la multiplicación y de la suma.

```
NOT %MW3 * 25 AND %MW10 + %MW12
```

Ejemplo 2

En este ejemplo, la multiplicación de %MW34 por 2 se efectúa en primer lugar, después el resultado se usa para efectuar el módulo.

```
%MW34 * 2 REM 6
```

Uso de los paréntesis

Los paréntesis se usan para modificar el orden de evaluación de los operadores, para que, por ejemplo, se lleve a cabo una suma antes que una multiplicación.

Los paréntesis se pueden imbricar y el nivel de imbricación no tiene límites.

Los paréntesis pueden utilizarse también para evitar una falsa interpretación del programa.

Ejemplo 1

En este ejemplo, la suma se lleva a cabo antes que la multiplicación:

```
(%MW10+%MW11)*%MW12
```

Ejemplo 2

Este ejemplo muestra que los paréntesis pueden utilizarse también para evitar una falsa interpretación del programa.

```
NOT %MW2 <> %MW4 + %MW6
```

Utilizando las reglas de prioridad de los operadores, la interpretación es la siguiente:

```
((NOT %MW2) <> (%MW4 + %MW6))
```

Aunque podría suponerse que la operación es la siguiente:

```
NOT (%MW2 <> (%MW4 + %MW6))
```

Los paréntesis permiten, por lo tanto, clarificar el programa.

Conversiones implícitas

Las conversiones implícitas se refieren a las palabras y las palabras dobles. Los operadores que se usan en las expresiones aritméticas, en las comparaciones y en el operador de asignación, efectúan las conversiones implícitas (en las que el usuario no interviene).

Para una instrucción del tipo: <operando 1> <operador> <operando 2>, los casos posibles de conversiones son:

Operando 1 de tipo	**Operando 2 de tipo**	**Conversión Operando 1**	**Conversión Operando 2**	**Operación en el tipo**
Palabra	Palabra	No	No	Palabra
Palabra	Palabra doble	Palabra doble	No	Palabra doble
Palabra doble	Palabra	No	Palabra doble	Palabra doble
Palabra doble	Palabra doble	No	No	Palabra doble

Para una asignación del tipo <operando izquierdo> := <operando derecho>, el operando de la izquierda es el que impone el tipo esperado para efectuar la operación, lo que quiere decir que el operando de la derecha debe convertirse, si es necesario, siguiendo la tabla:

Tipo de operando izquierda	**Tipo de operando derecha**	**Conversión operando de la derecha**
Palabra	Palabra	No
Palabra	Palabra doble	Palabra
Palabra doble	Palabra	palabra doble
Palabra doble	Palabra doble	No

Nota: Cualquier operación entre dos valores contiguos se efectúa en doble longitud.

Grafcet

9

Presentación

Contenido del capítulo

Este capítulo describe las reglas de programación en Grafcet.

Contenido:

Este capítulo contiene las siguientes secciones:

Sección	Apartado	Página
9.1	Presentación general del Grafcet	176
9.2	Regla de construcción del Grafcet	184
9.3	Programación de las acciones y de las condiciones	193
9.4	Macro etapas	202
9.5	Sección Grafcet	207

9.1 Presentación general del Grafcet

Presentación

Contenido de este subcapítulo

Este subcapítulo describe los elementos de base de un Grafcet

Contenido

Esta sección contiene los siguientes apartados:

Apartado	Página
Presentación del Grafcet	177
Descripción de los símbolos gráficos del Grafcet	178
Descripción de los objetos específicos del Grafcet	181
Posibilidades del Grafcet	183

Presentación del Grafcet

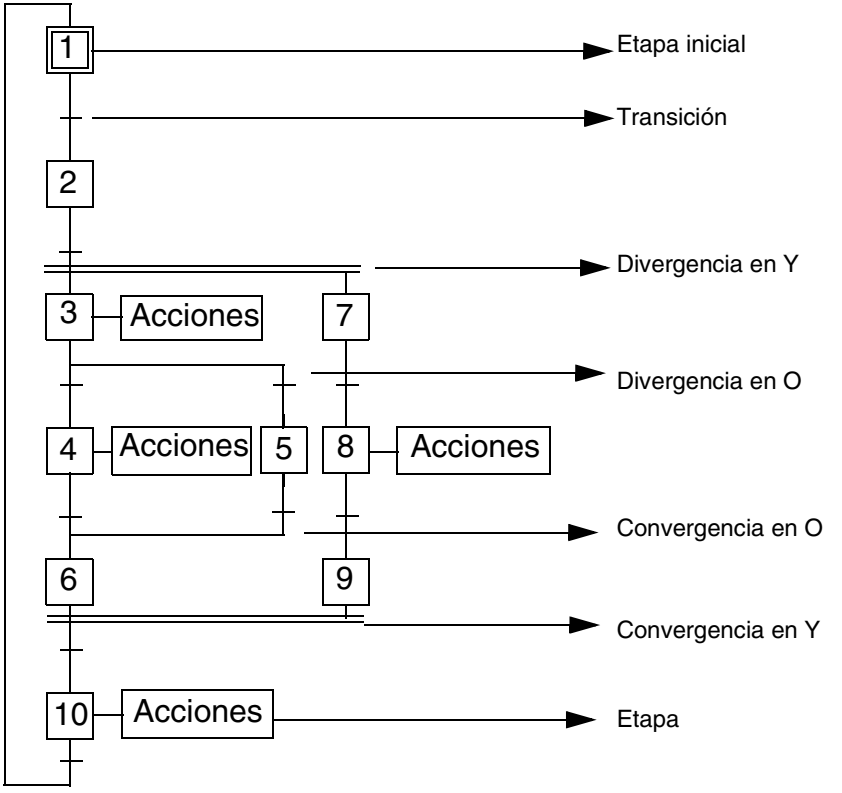
Generalidades

El lenguaje Grafcet se adecua al lenguaje "Diagrama funcional en secuencia" (SFC) de la norma IEC 1131-3.

El Grafcet permite representar gráficamente y de forma estructurada el funcionamiento de un automatismo secuencial.

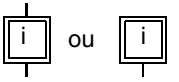
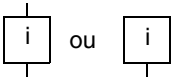
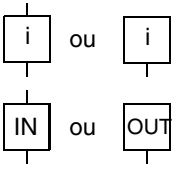
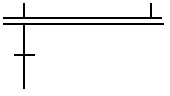
Presentación

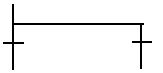
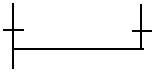
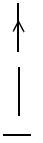
Esta descripción gráfica del comportamiento secuencial del automatismo y de las diferentes situaciones que de él derivan, se efectúa con la ayuda de símbolos gráficos simples.



Descripción de los símbolos gráficos del Grafcet

Descripción La tabla siguiente describe los elementos gráficos básicos del Grafcet.

Designación	Símbolo	Funciones
Etapas iniciales (		Simbolizan las etapas iniciales activas en el inicio del ciclo después de una inicialización o un rearranque en frío.
Etapas simples (		Simbolizan un estado estable del automatismo. El número máximo de etapas (las etapas iniciales incluidas) se puede configurar de: ● 1 a 96 para un TSX 37-10, ● 1 a 128 para un TSX 37-20, ● 1 a 250 para un TSX 57. Se puede configurar el número máximo de etapas activas simultáneamente.
Macro etapas		Simboliza una macro etapa: conjunto único de etapas y de transiciones. El número máximo de macro etapas se puede configurar de 0 a 63, únicamente para TSX 57.
Etapa de Macro etapas		Simbolizan las etapas de una macro etapa. El número máximo de etapas para cada macro etapa se puede configurar de 0 a 250, para TSX 57. Cada macro etapa implica una etapa IN y OUT.
Transiciones		Permiten pasar de una etapa a otra. Una receptividad asociada a esta transición permite definir las condiciones lógicas necesarias al alcanzar esta transición. El número máximo de transiciones es de 1024, y no se puede configurar. Se puede configurar el número máximo de transiciones validas simultáneamente.
Divergencias en Y		Transición de una etapa hacia varias etapas: permite activar simultáneamente 11 etapas como máximo.
Convergencias en Y		Transición de varias etapas hacia una sola: permite la desactivación simultánea de, como máximo, 11 etapas.

Designación	Símbolo	Funciones
Divergencias en O		Transición de una etapa hacia varias etapas: permite realizar un desvío hacia 11 etapas, como máximo.
Convergencias en O		Transición de varias etapas hacia una de sola: permite realizar un final de direccionamiento proveniente de 11 etapas como máximo.
Reenvíos de origen		"n" es el número de la etapa "de donde viene" (etapa de origen).
Reenvío de destino		"n" es el número de la etapa "a donde va" (etapa de destino).
Enlaces orientados hacia: ● alto ● bajo ● derecha o izquierda		Estos enlaces permiten efectuar un direccionamiento, un salto de etapas, una reanudación de etapas (secuencia).

Nota: El número máximo de etapas (etapas del gráfico principal + etapas de macro etapas) en la sección Grafcet no debe sobrepasar 1024 en TSX57.

Descripción de los objetos específicos del Grafcet

Generalidades

El Grafcet dispone de objetos bits asociados a las etapas, de bits de sistema específicos, de objetos palabras que indican el tiempo de actividad de las etapas y de las palabras de sistema específicas.

Objetos Grafcet

La tabla siguiente describe el conjunto de objetos asociados al Grafcet.

Designación		Descripción
Bits asociados a las etapas (1 = etapa activa)	%Xi	Estado de la etapa i del Grafcet principal
		(i de 0 a n) (n depende del procesador)
	%XMj	Estado de la macro etapa j (j de 0 a 63 para TSX/PMX/PCX 57)
	%Xj.i	Estado de la etapa i de la macro etapa j
	%Xj.IN	Estado de la etapa de entrada de la macro etapa j
	%Xj.OUT	Estado de la etapa de salida de la macro etapa j
Bits del sistema asociados al Grafcet	%S21	Provoca la inicialización del Grafcet
	%S22	Provoca el reset general del Grafcet
	%S23	Provoca la inmovilización del Grafcet
	%S24	Provoca el reset de las macro etapas en función de las palabras del sistema %SW22 a %SW25
	%S25	Puesta a 1 tras: • rebasamiento de las tablas (etapas/transición), • ejecución de un gráfico incorrecto (reenvío de destino en una etapa que no pertenece al gráfico).
Palabras asociadas a las etapas	%Xi.T	Tiempo de actividad de la etapa i del Grafcet principal
	%Xj.i.T	Tiempo de la actividad de la etapa i de la macro etapa j
	%Xj.IN.T	Tiempo de actividad de la etapa de entrada de la macro etapa j
	%Xj.OUT.T	Tiempo de actividad de la etapa de salida de la macro etapa j
Palabras de sistema asociadas al Grafcet	%SW20	Palabra que permite conocer por el ciclo actual el número de etapas activas, para activar y para desactivar.
	%SW21	Palabra que permite conocer por el ciclo actual el número de transiciones válidas, para validar o para invalidar.
	%SW22 à %SW25	Seguido de 4 palabras que permiten designar las macro etapas que se ponen en reset en la puesta a 1 del bit %S24.

Bits asociados a las etapas

Los bits asociados a la etapas %Xi, a las macro etapas %XMi, y a las etapas de las macro etapas %Xj.I, %Xj.IN y %Xj.OUT tienen las siguientes propiedades:

- Están en 1 cuando las etapas son activas.
- Pueden comprobarse en todas las tareas, pero sólo pueden escribirse en el tratamiento preliminar de la tarea maestra (preposicionamiento de los gráficos). Estas pruebas y acciones se programan tanto en lenguaje de contactos, como en lenguaje de lista de instrucciones, o en lenguaje literal.
- Se pueden indexar.

Tiempo de actividad

Las palabras tiempo de actividad de las etapas %Xi.T y de las etapas de las macro etapas %Xj.I , %Xj.IN y %Xj.OUT tienen las siguientes propiedades:

- Se incrementan todas en 100 ms y toman un valor de 0 a 9999.
- Incremento de la palabra: durante la actividad de la etapa asociada.
- En la desactivación de la etapa, el contenido está inmovilizado.
- En la activación de la etapa, el contenido se pone en reset y luego se incrementa.
- El número de palabras de tiempo de actividad no se puede configurar, se reserva una palabra para cada etapa.
- Estas palabras pueden indexarse.

Posibilidades del Grafcet

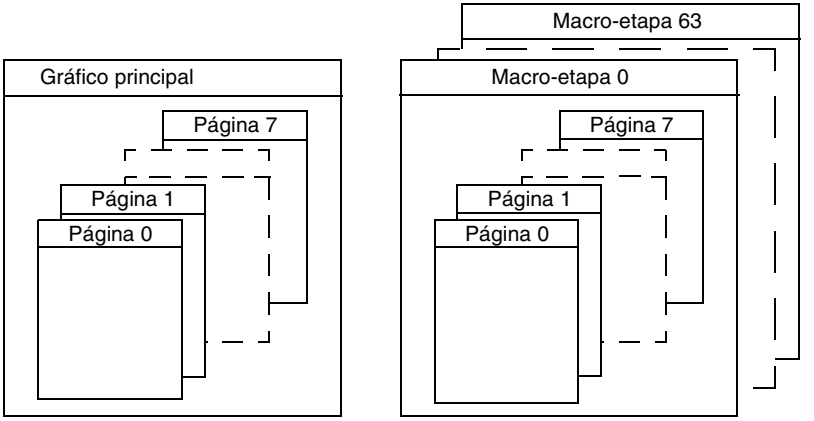
Generalidades

El tratamiento secuencial se estructura en:
- 1 subconjunto: Gráfico principal,
- 64 subconjuntos: Macro etapas

Estos subconjuntos, a su vez, quedan divididos en 8 páginas.

Ilustración

La ilustración siguiente describe la estructura general de la página del Grafcet.



Características

Dependen del procesador que se va a programar, están recogidas en la tabla que se ofrece a continuación.

Número	**TSX 37-10**		**TSX 37-20**		**TSX 57**	
	Por defecto	Máximo	Por defecto	Máximo	Por defecto	Máximo
Etapas del Gráfico principal	96	96	128	128	128	250
Macro etapas	0	0	0	0	8	64
Etapas de macro etapas	0	0	0	0	64	250
Total de las etapas	96	96	128	128	640	1024
Etapas activas simultáneamente	16	96	20	128	40	250
Transiciones válidas simultáneamente	20	192	24	256	48	400

El número de transiciones síncronas (o número de convergencias en Y) no debe sobrepasar 64, siendo el número total de transiciones de 1024, siempre.

9.2 Regla de construcción del Grafcet

Presentación

Contenido de este subcapítulo

Este subcapítulo describe las reglas de base para construir los gráficos del Grafcet.

Contenido

Esta sección contiene los siguientes apartados:

Apartado	Página
Representación del Grafcet	185
Aplicación de las divergencias y convergencias O	186
Aplicación de las divergencias y convergencias Y	187
Utilización de los reenvíos	188
Utilización de los enlaces orientados	191
Comentario Grafcet	192

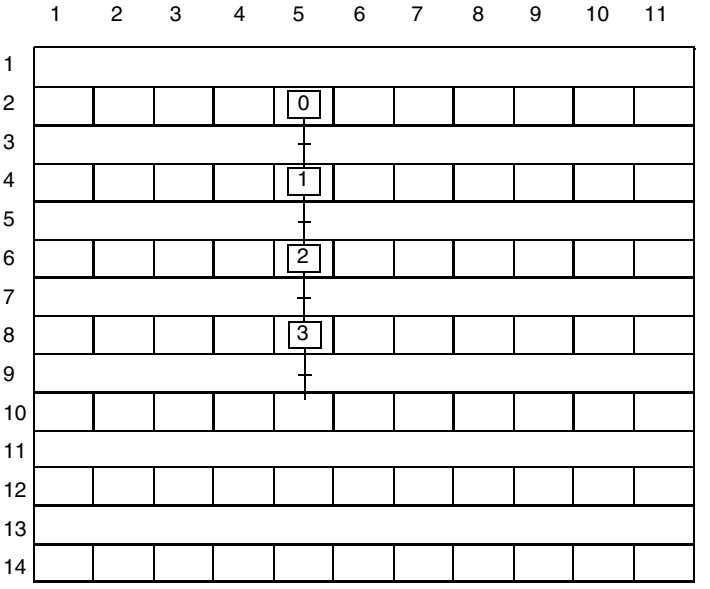
Representación del Grafcet

Generalidades

El gráfico principal y cada una de las macroetapas se programan en 8 páginas (página 0 a 7).
Una página Grafcet está compuesta de 14 líneas y 11 columnas que definen 154 celdas.
En cada celda, es posible introducir un elemento gráfico.

Ilustración

Los esquemas siguientes ilustran el reparto de una página Grafcet.



Reglas de escritura

- La primera línea permite introducir los reenvíos de origen.
- La última línea permite introducir los reenvíos de destino.
- Las líneas pares (de la 2 a la 12) son líneas de etapas (para las etapas reenvíos de destino).
- Las líneas impares (de la 3 a la 13) son las líneas de transiciones (para las transiciones y los reenvíos de origen).
- Cada etapa se identifica mediante un número diferente (0 a 127) en un orden indeterminado.
- Los gráficos diferentes pueden representarse en una misma página.

Aplicación de las divergencias y convergencias O

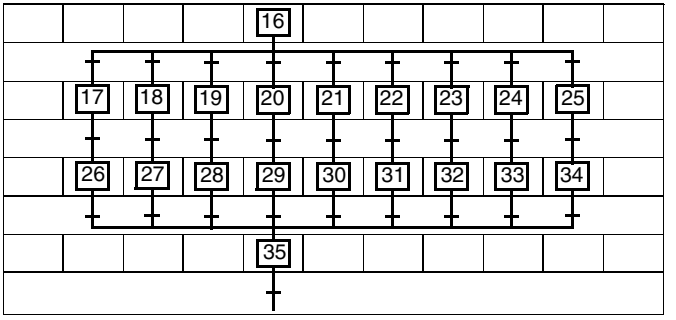
Función

Una divergencia O es un desvío de una etapa hacia varias etapas.

Una convergencia O aplica un final de desvío.

Ilustración

El esquema que puede verse a continuación presenta una divergencia O de una etapa hacia 9 etapas y una convergencia O.



Regla de uso

- El número de transiciones hacia arriba de un final de desvío (convergencia en O) o hacia abajo de un desvío (divergencia en O) no puede sobrepasar 11.
- Un desvío puede trazarse hacia la izquierda o hacia la derecha.
- Un desvío debe terminar, generalmente, con un final de desvío.
- Para evitar que se franqueen simultáneamente varias transiciones, las receptividades asociadas deben ser exclusivas.

Aplicación de las divergencias y convergencias Y

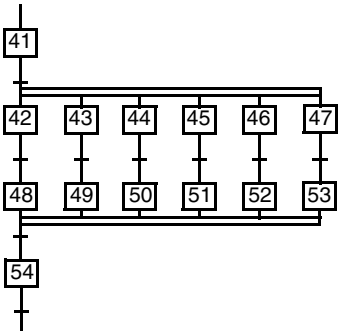
Función

Una divergencia Y permite activar simultáneamente varias etapas.

Una convergencia Y permite desactivar simultáneamente varias etapas.

Ilustración

El esquema siguiente presenta una divergencia y una convergencia Y de 6 etapas.



Reglas de uso

- El número de etapas hacia abajo de una activación simultánea (divergencia en Y) o hacia arriba de una desactivación simultánea (convergencia en Y) no debe sobrepasar 11.
- Una activación simultánea de etapas debe terminar generalmente por una desactivación simultánea de etapas.
- La activación simultánea siempre se representa desde la izquierda hacia la derecha.
- La desactivación simultánea se representa siempre desde la derecha hacia la izquierda.

Utilización de los reenvíos

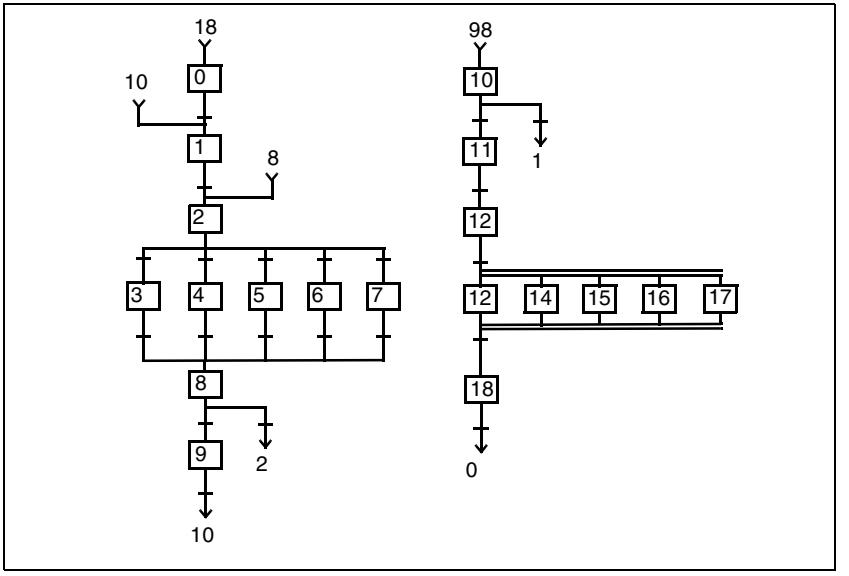
Función

Los reenvíos aseguran la continuidad de un Grafcet cuando el trazado directo de un enlace orientado no puede realizarse, tanto en el seno de una página, como entre dos páginas consecutivas o no.

Esta continuidad queda asegurada gracias a un reenvío de destino al que corresponde sistemáticamente un reenvío de origen.

Ejemplo

La siguiente ilustración muestra dos ejemplos de reenvíos.

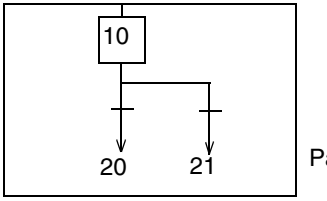
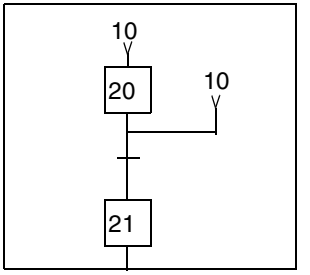
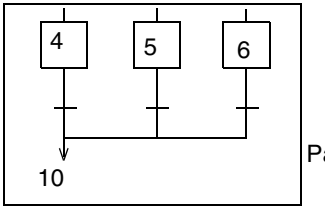
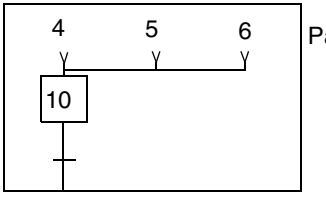


La tabla que se presenta a continuación aclara el uso de los reenvíos del ejemplo.

Uso	Ejemplo
El cierre de un gráfico se puede efectuar con la ayuda de reenvíos.	Cierre de la etapa 18 hacia la etapa 0.
Una reanudación de secuencia se puede efectuar con la ayuda de reenvíos.	Etapa 10 hacia etapa 1 o etapa 8 hacia etapa 2.
Uso de los reenvíos cuando una rama de gráfico es más larga que la página.	Etapa 9 hacia etapa 10.

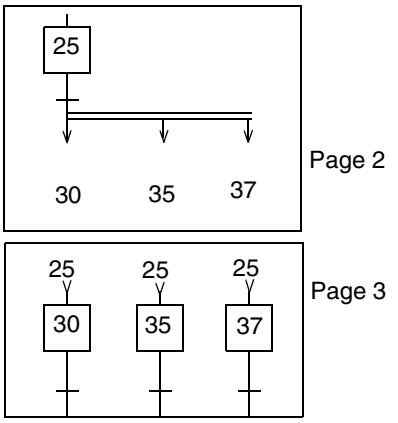
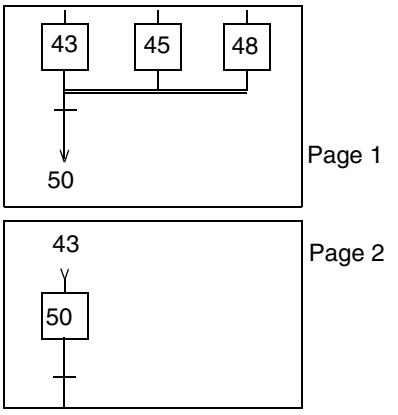
Reenvíos usados en las divergencias y convergencias O

La tabla siguiente ofrece las reglas de uso de los reenvíos en el caso de divergencia o convergencia O.

Regla	Ilustración
Para un desvío, las transiciones y los reenvíos de destino deben ser introducidos en la misma página.	 Page 1
Para el final de desvío, los reenvíos de origen deben introducirse en la misma página que la etapa de destino.	 Page 2
Para un final de desvío seguido de un reenvío de destino, debe haber tantos reenvíos de origen como de etapas antes del final del desvío.	 Page 1  Page 2

Reenvíos usados en las divergencias y convergencias Y

La tabla siguiente ofrece las reglas de uso de los reenvíos en el caso de divergencia o convergencia Y.

Regla	Ilustración
Para una activación simultánea de etapas, los reenvíos de destino deben estar en la misma página que la etapa y la transición de divergencia.	
Para una desactivación simultánea, las etapas y la transición de convergencia deben estar en la misma página que el reenvío de destino. Si varias etapas convergen en una sola transición, el reenvío de origen lleva el número de la etapa anterior que está más a la izquierda.	

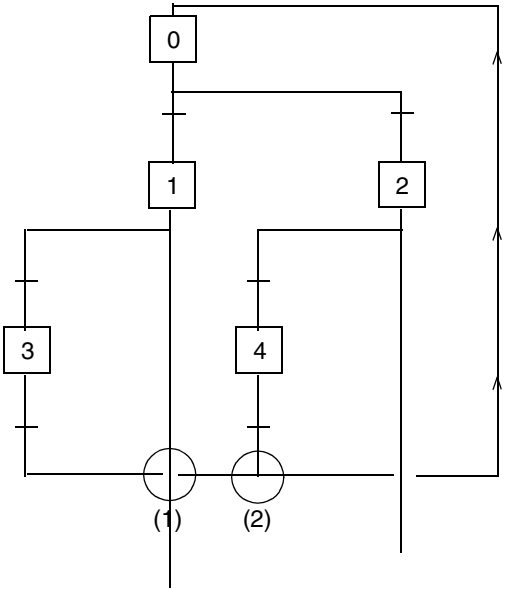
Utilización de los enlaces orientados

Función

Los enlaces orientados enlazan de nuevo una etapa a una transición o una transición a una etapa. Pueden ser verticales u horizontales.

Ilustración

El esquema siguiente presenta un ejemplo de utilización de un enlace orientado.



Reglas

Los enlaces orientados pueden:

- cruzarse (1), en este momento son de naturalezas diferentes,
- encontrarse (2), en este momento son de la misma naturaleza.

El cruce de un enlace con una activación o una desactivación simultánea de etapas es imposible.

Comentario Grafcet

Generalidades

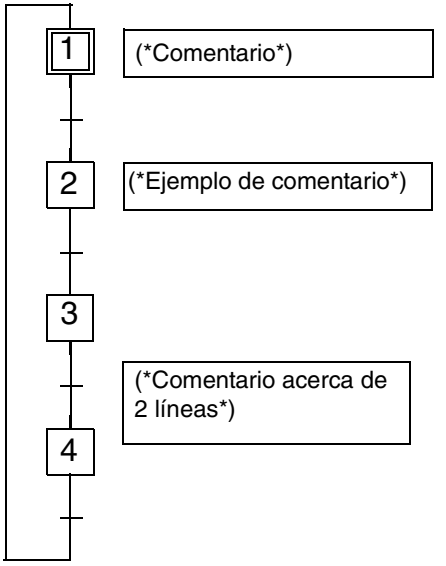
Los comentarios permiten ofrecer las informaciones de las etapas y las transiciones de un Grafcet. Son opcionales.

Sintaxis

El texto del comentario queda enmarcado por (* a la izquierda y *) a la derecha. Como mucho puede ser de 64 caracteres.

Ilustración

La siguiente ilustración presenta dos ejemplos de comentarios.



Reglas

- En una página Grafcet, es posible introducir un comentario en cualquier celda.
- Un comentario ocupa dos celdas contiguas, en dos líneas como máximo. Si la zona de visualización es demasiado pequeña, el comentario se interrumpe en la visualización, aunque, en el momento de imprimir la documentación, el comentario se presenta íntegro.
- El comentario introducido en una página Grafcet queda almacenado en las informaciones gráficas que incorpora el autómata. Según esto, consumen la memoria de programa.

9.3 Programación de las acciones y de las condiciones

Presentación

Contenido de este subcapítulo

Este subcapítulo describe las reglas de programación de las acciones y condiciones

Contenido

Esta sección contiene los siguientes apartados:

Apartado	Página
Programación de las acciones asociadas a las etapas	194
Programación de las acciones para la activación o la desactivación	196
Programación de las acciones continuas	197
Programación de las receptividades asociadas a las transiciones	198
Programación de las receptividades en lenguaje de contactos	199
Programación de las receptividades en lenguaje lista de instrucciones	200
Programación de las receptividades en lenguaje literal estructurado	201

Programación de las acciones asociadas a las etapas

Generalidades

Las acciones asociadas a las etapas describen las órdenes que hay que transmitir a la parte operativa (proceso que se automatizará) o a otros sistemas automatizados.

Las acciones que pueden programarse tanto en lenguaje de contactos, como en lenguaje de lista de instrucciones, o en lenguaje literal estructurado.
Estas acciones sólo se recuentan cuando la etapa a la que pertenecen está activada.

3 tipos de acciones

El programa PL7 autoriza tres tipos de acciones:

- **las acciones que hay que activar**: acciones ejecutadas una vez que la etapa a la que pertenecen pasa de un estado inactivo a uno de activo.
- **las acciones que hay que desactivar**: acciones ejecutadas una vez que la etapa a la que pertenecen pasa de un estado activo a uno de inactivo.
- **las acciones continuas**: estas acciones se ejecutan cuando la etapa a la que pertenecen está activada.

Nota: Una misma acción puede incluir varios elementos de programación (frases o redes de contactos).

Identificación de las acciones

Estas acciones se identifican tal como se muestra a continuación:

MAST - <nombre sección Grafcet> - CHART (ó MACROk)- PAGE n %Xi x
con
x = P1 para Activación, x = N1 Continua, x = P0 Desactivación
n = Número de la página
i = Número de la etapa

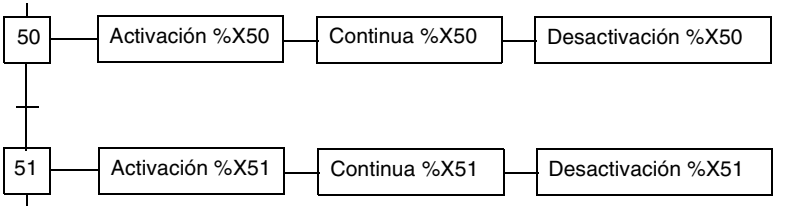
Ejemplo: MAST - Pintura - CHART - PAGE 0 %X1 P1 Acción para la activación de la etapa 1 de la página 0 de la sección Pintura

Reglas de uso

- Todas las acciones se considerarán como acciones memorizadas, por tanto: una acción controlada para el tiempo de duración de la etapa Xn deberá ponerse a acero durante la desactivación de la etapa Xn o en la activación de la etapa Xn+1.
  Una acción de efecto mantenido en varias etapas se sitúa en 1 durante la activación de la etapa Xn y se pondrá a cero en la desactivación de la etapa Xn+m.
- Todas las acciones pueden ser sometidas a condiciones lógicas, es decir, pueden ser condicionales.
- Las acciones sometidas a seguridades indirectas, deberán programarse en el tratamiento posterior (Véase *Descripción del tratamiento posterior, p. 218*) (tratamiento ejecutado en cada recuento)

Orden de ejecución de las acciones

Para el ejemplo que sigue, en una vuelta de ciclo, el orden de ejecución de las aciones es el siguiente. Cuando la etapa 51 está activada, las acciones se ejecutarán en el orden siguiente:
1. acciones para la desactivación de la etapa 50,
2. acciones para la activación de la etapa 51,
3. acciones continuas de la etapa 51.
Ejemplo:



Desde el momento de la activación de la etapa 51, las acciones continuas asociadas ya no se recontarán.

Programación de las acciones para la activación o la desactivación

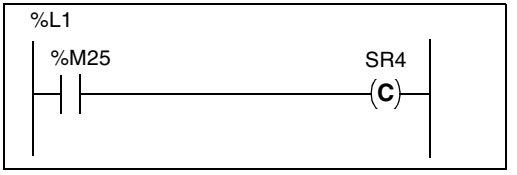
Reglas

Estas acciones se ejecutan una vez que la etapa a la que pertenecen pasa de un estado inactivo a uno de activo.

Estas acciones funcionan a través de impulsos y se ejecutan en **una sola etapa de recuento**. Permiten llamar a un subprograma, incrementar un contador, etc.

Ejemplo 1

En este ejemplo, esta acción llama a un subprograma



Ejemplo 2

En este ejemplo, esta acción incrementa la palabra %MW10 y pone a 0 las palabras %MW0 y %MW25.

```
%L1:
INC %MW10;%MW0:=0;%MW25:=0;
```

Programación de las acciones continuas

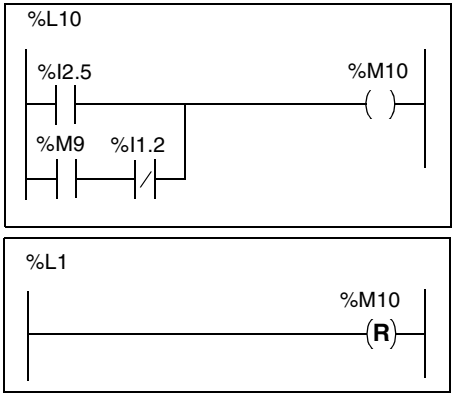
Reglas

Estas acciones se ejecutan cuando la etapa a la que pertenecen está activada.
Pueden ser:

- **Acciones condicionales**: la acción se ejecuta si se satisface una condición,
- **Acciones temporizadas**: se trata de un caso particular, el tiempo interviene como condición lógica. Este control se puede llevar a cabo simplemente comprobando el tiempo de actividad asociado a la etapa.

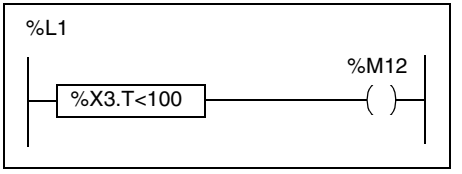
Ejemplo de acción condicional

En este ejemplo, el bit %M10 se controla en la entrada %I2.5 o en el bit interno %M9 y en la entrada %I1.2.
Mientras que la etapa 2 es activa y las condiciones están presentes, %M10 se sitúa en 1.
El último estado leído para desactivarse se almacena porque las acciones asociadas ya no se examinan. Por tanto, es imprescindible situar a 0 el bit %M10, por ejemplo, en la acción de desactivar la etapa.
Ilustración del ejemplo.



Ejemplo de acción temporizada

En el ejemplo, el bit %M12 se controla mientras el tiempo de actividad de la etapa 3 sea inferior a 10 segundos (base de tiempo: 100 ms).



Del mismo modo, estas acciones pueden ser incondicionales.

Programación de las receptividades asociadas a las transiciones

Generalidades

Una receptividad asociada a una transición permite definir las condiciones lógicas necesarias para superar esta transición.

El número máximo de transiciones es de 1024, y no se puede configurar.
El número máximo de transiciones validas simultáneas se puede configurar.

Reglas

- A cada transición se asocia una receptividad que puede programarse en lenguaje de contactos, en lenguaje lista de instrucciones o en lenguaje literal.
- Una receptividad sólo se examina si la transición a la que pertenece es válida.
- Una receptividad corresponde a una red de contactos o a una lista de instrucciones o a una expresión literal, incluyendo una serie de pruebas sobre bits y/o en la palabra.
- Una receptividad sin programar siempre es una receptividad falsa.

Identificación de la receptividad

Las receptividades se identifican tal como se muestra a continuación:

`MAST - <nombre sección Grafcet> - CHART(ó MACROk) - PAGE n %X(i) --> % X(j)` con:
n = Número de la página
i = Número de la etapa anterior
i = Número de la etapa posterior

Ejemplo: `MAST - Pintura - CHART - PAGE 0 %X(0) --> %X(1)`
Receptividad asociada a la transición entre la etapa 0 y la etapa 1 de la página 0 del gráfico de la sección Pintura.

Nota: Cuando se produce una activación simultánea o una desactivación de etapas, la identificación mostrada es la de la etapa de la columna situada más a la izquierda.

Receptividad usando el tiempo de actividad

En algunas aplicaciones, las acciones se dirigen sin control de información de retorno (fin de carrera, detector,...). La duración de la etapa queda condicionada por un tiempo, el lenguaje PL7 permite utilizar el tiempo activado asociado a cada etapa.

Ejemplo: `! X3.T>=150`
Esta receptividad programada en lenguaje literal estructurado permite permanecer en la etapa 3 durante 15 segundos.

Programación de las receptividades en lenguaje de contactos

Reglas de programación

La receptividad asociada a la transición se programa bajo la forma de una red de contactos que incluye una zona de prueba y una zona de acción.

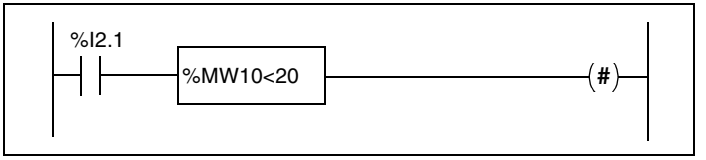
La estructura de la red de contactos es parecida a la de una red programada en un módulo de programa.

Únicamente se podrán utilizar los elementos siguientes:

- **elementos gráficos de prueba**: contactos (%Mi, %I, %Q, %TMi.D ...), bloques de comparaciones,
- **elementos gráficos de acción**: únicamente la bobina "sostenido" (las otras bobinas no son significativas en este caso).

Ejemplo

Este ejemplo ilustra la programación de una receptividad en lenguaje de contactos.



Programación de las receptividades en lenguaje lista de instrucciones

Reglas de programación

La receptividad asociada a la transición se programa bajo la forma de una lista de instrucciones incluyendo sólo las instrucciones de prueba.

La lista de instrucciones admitidas para la escritura de una receptividad difiere de una lista de instrucciones clásica en:

- la estructura general: no hay etiqueta (%L).
- la lista de instrucciones:
  - no hay instrucciones de acciones (objetos bits, palabras o bloques de funciones),
  - no hay salto, ni llamada de subprograma.

Ejemplo

Este ejemplo ilustra la programación de una receptividad en lenguaje lista de instrucciones.

```
!           LD    %I2.1
            AND   [%MW10<20]
```

Programación de las receptividades en lenguaje literal estructurado

Reglas de programación

La receptividad asociada a la transición se programa bajo la forma de una expresión booleana, de una expresión aritmética o de una asociación de las dos.

La expresión admitida para escribir una receptividad difiere de una línea de programación en lenguaje literal en:

- la estructura general:
  - no hay etiqueta (%L).
  - no hay frase de acción, ni frases condicionales ni frases repetidas.
- la lista de instrucciones:
  - no hay acción sobre objeto bit,
  - no hay salto, ni llamada de subprograma,
  - no hay transferencia, ni tampoco instrucción de acción sobre bloques.

Ejemplo

Este ejemplo ilustra la programación de una receptividad en lenguaje literal estructurado.

```
! %I2.1 AND [%MW10<20]
```

9.4 Macro etapas

Presentación

Contenido de esta sección

Esta sección describe la programación de las macro etapas.

Contenido

Esta sección contiene los siguientes apartados:

Apartado	Página
Presentación de las macro etapas	203
Constitución de una macro etapa	204
Características de las macro etapas	205

Presentación de las macro etapas

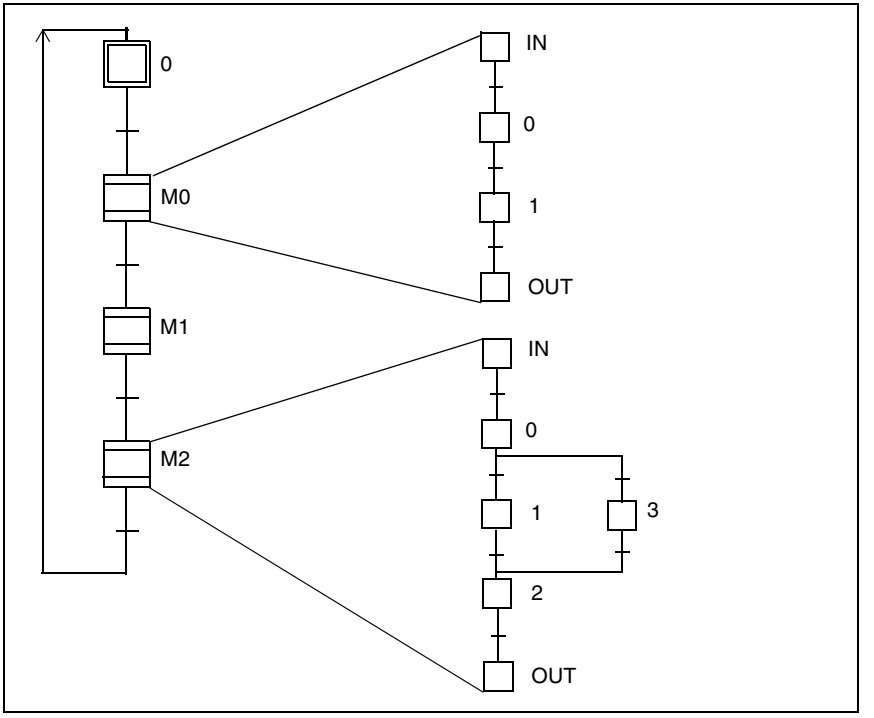
Generalidades

Una macro etapa es una representación condensada, única, de un conjunto de etapas y de transiciones.
Una macro etapa se introduce en un gráfico como una etapa y respeta las reglas de evolución.

Macro representación

Un Grafcet de primer nivel que describe el encadenamiento de secuencias permite explicar mejor la estructuración de la parte comando.
Cada secuencia está asociada a un símbolo particular de la etapa: la macro etapa.

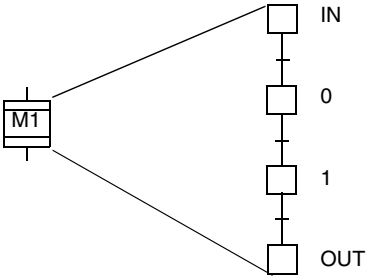
Esta noción de "macro representación" permite jerarquizar el análisis. Cada nivel puede completarse, modificarse, sin involucrar a los otros niveles.
Las macro etapas están disponibles para los autómatas TSX57.
La figura siguiente muestra un Grafcet constituido por 3 macro etapas.



Constitución de una macro etapa

Descripción

La simbolización gráfica de una macro etapa se distingue de una etapa por dos trazos horizontales.
La ilustración siguiente muestra una macro etapa y su expansión.

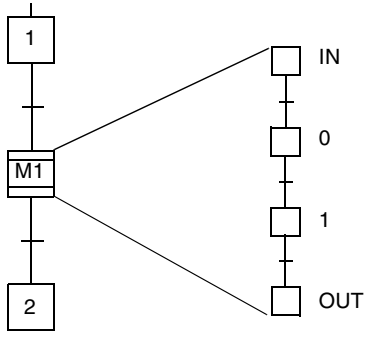


La expansión de una macro etapa se caracteriza por 2 etapas específicas:
- una etapa de entrada que responde a las mismas reglas que las otras etapas,
- una etapa de salida que no puede disponer de acciones asociadas.

Evolución

Cuando la macro etapa es activa, la evolución de la macro etapa respeta las reglas generales de la evolución de un Grafcet).

Ejemplo:



La macro etapa M1 se activa cuando la etapa 1 está activada y su receptividad hacia abajo es verdadera.
Se desactiva cuando la etapa de salida está activa y la receptividad M1>2 es verdadera. En ese momento, la etapa 2 queda activada.

Características de las macro etapas

Características generales

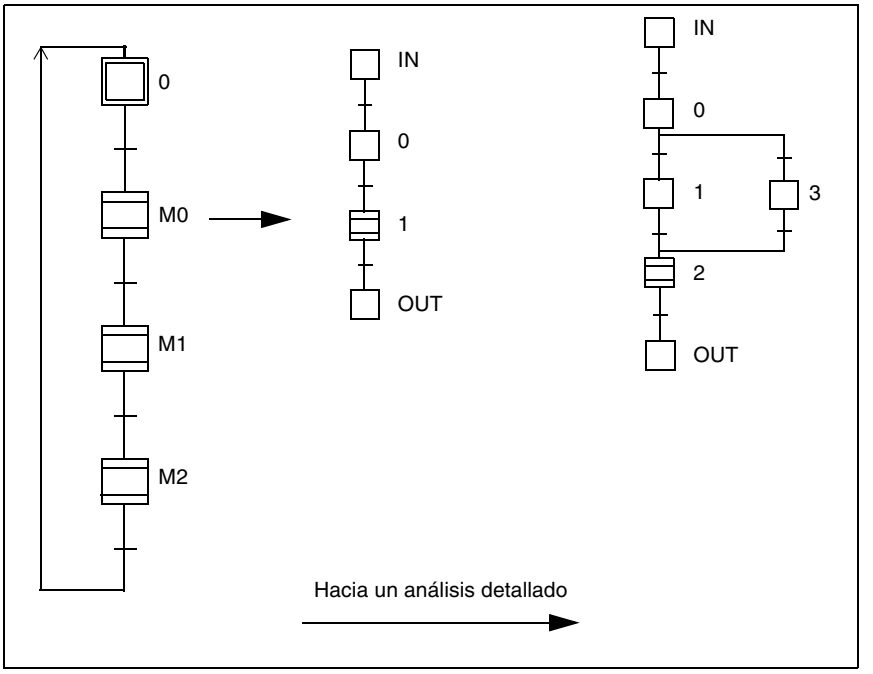
El lenguaje Grafcet PL7 autoriza la programación de 64 macro etapas M0 a M63.

La expansión de una macro etapa, compuesta de una o varias secuencias, se programa como mucho a 8 páginas y comprende un máximo de 250 etapas, además de las etapas IN y OUT.

Una macro etapa puede contener una o varias macro etapas. Esta jerarquía es posible hasta un total de 64 niveles.

Ilustración

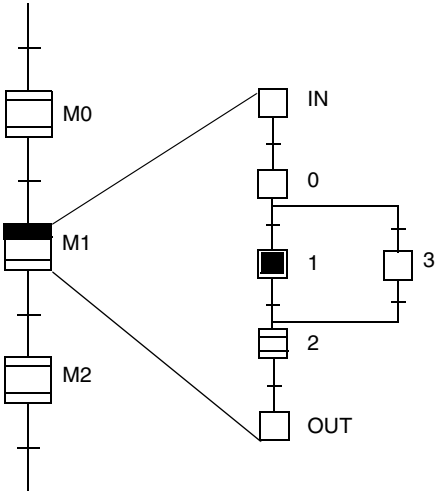
El análisis de una aplicación puede estructurarse de manera que pueda garantizar un enfoque global más detallado de las diferentes operaciones a ejecutar.



Etapas iniciales

La expansión de una macro etapa puede contener una o varias etapas iniciales. Estas etapas iniciales se activan al conectar o cuando se produce una inicialización por programa. En ese momento la macro etapa queda reflejada en estado activo.

En el ejemplo que sigue, la etapa inicial 1 de la expansión queda activada cuando se produce una inicialización del programa, la macro etapa pasa entonces al estado activo.



9.5 Sección Grafcet

Presentación

Contenido de este subcapítulo

Este subcapítulo presenta la constitución de una sección Grafcet. Describe la aplicación y las reglas de programación de cada tratamiento.

Contenido

Esta sección contiene los siguientes apartados:

Apartado	Página
Estructura de una sección Grafcet	208
Descripción del tratamiento preliminar	209
Posicionamiento previo del Grafcet	210
Inicialización del Grafcet	211
Reset del Grafcet	212
Inmovilización del Grafcet	213
Reset de las macroetapas	214
Funcionamiento del tratamiento secuencial	216
Descripción del tratamiento posterior	218

Estructura de una sección Grafcet

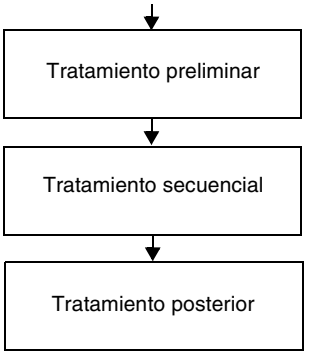
Composición de una sección

Una sección de programa escrito en Grafcet implica tres tratamientos consecutivos:
- el tratamiento preliminar,
- el tratamiento secuencial,
- el tratamiento posterior.

La sección Grafcet se programa en la tarea MAST.

Ilustración

El siguiente esquema ilustra el orden de recuento de los tratamientos.



Función de los tratamientos

La siguiente tabla describe la función de cada uno de los tratamientos y el lenguaje con el que pueden programarse.

Tratamiento	Función	Lenguaje
Preliminar	Permite tratar: ● las inicializaciones en el rearranque del sector o en fallo, ● las inicializaciones en el rearranque del sector o en fallo, ● la lógica de entrada.	Lenguaje de contactos, lista de instrucciones ó literal.
Secuencial	Permite tratar la estructura secuencial de la aplicación y permite el acceso al tratamiento de las receptividades y de las acciones directamente asociadas a las etapas.	Grafcet
Posterior	Permite tratar: ● la lógica de salida, ● la supervisión y las seguridades indirectas específicas a las salidas.	Lenguaje de contactos, lista de instrucciones o literal.

Nota: Las macroetapas se ejecutan con su orden de recuento en el tratamiento secuencial.

Descripción del tratamiento preliminar

Generalidades

Introducido en lenguaje de contactos, en lenguaje de lista de instrucciones o en lenguaje literal, el tratamiento preliminar se recuenta en su totalidad de arriba hacia abajo.

Ejecutado antes de los tratamientos secuencial y posterior, permite tratar todos los sucesos que tienen influencia sobre los últimos:

- gestión de reanudaciones de sector y reinicializaciones,
- puesta a cero o posicionamiento previo de los gráficos

Únicamente hace falta actuar sobre los bits asociados a las etapas en el tratamiento preliminar (puesta a 0 ó a 1 de los bits etapas %Xi ó %Xi.j para las instrucciones `SET` y `RESET`).

Bits del sistema

Las operaciones de posicionamiento previo, inicialización, inmovilización... se efectúan con la ayuda de los bits del sistema %S21 a %S24.

Los bits del sistema asociados al Grafcet se clasifican por orden de prioridad (%S21 a %S24), cuando algunos de ellos se ponen simultáneamente a 1 en el tratamiento preliminar, se tratan uno a uno en orden creciente (uno solo es efectivo para todo el recuento).

Estos bits son efectivos al inicio del tratamiento secuencial.

Tratamiento de las reanudaciones en frío

Con una nueva aplicación, o en una pérdida de contexto del sistema, éste efectúa un arranque en frío.

El sistema pone el bit %S21 a 1 antes de la llamada del tratamiento preliminar y el Grafcet se sitúa en las etapas iniciales.

Si se desea un tratamiento particular con respecto a la aplicación en caso de arranque en frío, se dispone de la posibilidad de probar %S0 que se mantiene en 1 durante el primer ciclo de la tarea maestra (MAST).

Tratamiento de las reanudaciones en caliente.

Después de un corte de alimentación en la red sin cambio de aplicación, el sistema efectúa una reanudación en caliente, se reinicia en el estado que precedía al corte del sector.

Si se desea un tratamiento particular con respecto a la aplicación en caso de reanudación en caliente, se dispone de la posibilidad de probar %S1 en el tratamiento preliminar, y de llamar al programa correspondiente.

Posicionamiento previo del Grafcet

Función

El posicionamiento previo del Grafcet puede emplearse cuando se pasa de un funcionamiento en marcha normal a marcha específica o cuando aparece un incidente (ejemplo: fallo que provoca una marcha reducida).

Esta operación interviene en el desarrollo normal del ciclo de aplicación, por tanto, deberá llevarse a cabo con precaución.

Posicionamiento previo del Grafcet

El posicionamiento previo puede referirse al conjunto o a una parte del tratamiento secuencial:

- usando las instrucciones `SET`, `RESET`,
- para la puesta a cero general (%S22) más tarde, en el ciclo siguiente, posicionamiento a 1 de las etapas.

Nota: Si se produce una puesta a cero de una etapa, no se ejecutan las acciones para su desactivación.

Ejemplo

En este ejemplo la puesta a 1 del bit %M20 provoca el posicionamiento previo de las etapas %X12 a 1, de las etapas %X10 y %X11 a 0.

Inicialización del Grafcet

Función

La inicialización del Grafcet se efectúa por medio del bit del sistema %S21. Normalmente en el estado 0, el paso al estado 1 de %S21 provoca:

- la desactivación de las etapas activas,
- la activación de las etapas iniciales.

Inicialización del Grafcet

La tabla que se muestra a continuación muestra las diferentes posibilidades de la puesta a 1 y a 0 del bit del sistema %S21.

Puesto al estado 1	Reset
• Con la puesta al estado 1 de %S0 • Con el programa del usuario • Con el terminal (con depuración o tabla de animación)	• Con el sistema al inicio del tratamiento • Con el programa del usuario • Con el terminal (con depuración o tabla de animación)

Regla de uso

Cuando se administra con el programa del usuario, %S21 deberá situarse a 0 ó 1 en el tratamiento preliminar.

Reset del Grafcet

Función

El reset del Grafcet se lleva a cabo a través del bit del sistema %S22.

Normalmente en el estado 0, la puesta al estado 1 de %S22 provoca la desactivación de las etapas activas del conjunto del tratamiento secuencial.

Nota: La función RESET_XIT permite reinicializar por programa el tiempo de actividad de todas las etapas del tratamiento secuencial (Ver Manual di Referencia, tomo 2).

Reset del Grafcet

La tabla que se muestra a continuación muestra las diferentes posibilidades de la puesto a 1 y a 0 del bit del sistema %S22.

Puesto en el estado 1.	Reset
• Con el programa del usuario • Con el terminal (con depuración o tabla de animación)	• Mediante el sistema al final del tratamiento secuencial

Regla de uso

- este bit debe colocarse a 1 en el tratamiento preliminar,
- la puesta a 0 de %S22 se realiza a través del sistema, por lo que resulta del todo inútil ponerlo en Reser mediante el programa o con el terminal.

Para reiniciar el tratamiento secuencial en una situación dada, se debe prever, según la aplicación, un procedimiento de inicialización o de preposicionamiento del Grafcet.

Inmovilización del Grafcet

Función

La inmovilización del Grafcet se efectúa por medio del bit del sistema %S23.

Normalmente en el estado 0, la puesta a 1 de %S23 provoca la conservación del estado de los Grafcet. Independientemente del valor de las receptividades hacia abajo de las etapas activas, los Grafcet no cambian. La inmovilización se mantiene mientras el bit %S23 esté en 1.

Inmovilización del Grafcet

La tabla que se muestra a continuación muestra las diferentes posibilidades de la puesta a 1 y reset del bit del sistema %S23.

Puesto al estado 1	Reset
• Con el programa del usuario • Con el terminal (con depuración o tabla de animación)	• Con el programa del usuario • Con el terminal (con depuración o tabla de animación)

Regla de uso

- administrado por el programa del usuario, este bit se situará en 1 ó 0 en el tratamiento preliminar,
- el bit %S23 asociado a los bits %S21 y %S22 permite llevar a cabo una inmovilización del tratamiento secuencial al estado inicial o al estado 0. Del mismo modo, el Grafcet puede preposicionarse una vez haya sido inmovilizado por %S23.

Reset de las macroetapas

Función

La inmovilización del Grafcet se efectúa por medio del bit del sistema %S24.

Normalmente en el estado 0, el paso al estado 1 del %S24 provoca el reset de las macroetapas escogidas en una tabla de 4 palabras del sistema (%SW22 a %SW25).

Nota: La función RESET_XIT permite reinicializar por programa el tiempo de actividad de las etapas de macroetapa (Ver Manual di Referencia, tomo 2).

Reser de las macroetapas

La tabla que se muestra a continuación muestra las diferentes posibilidades de la puesta a 1 y reset del bit del sistema %S24.

Puesto al estado 1	Puesto de nuevo al estado 0
• Con el programa del usuario	• Con el sistema al inicio del tratamiento

Reglas de uso

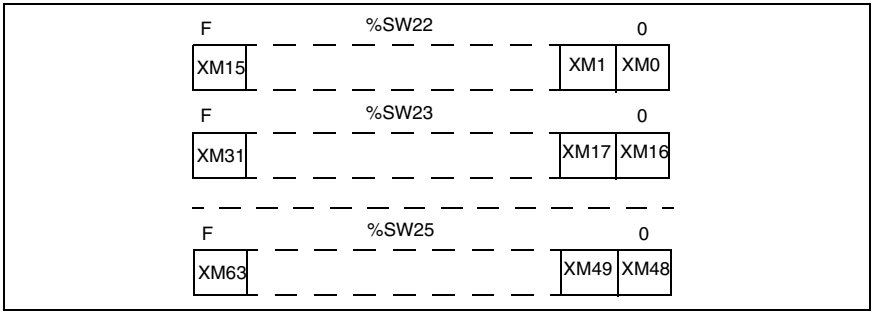
- este bit debe colocarse a 1 solamente en el tratamiento preliminar,
- la puesta a 0 de %S24 se realiza a través del sistema; está prohibido ponerlo a reset mediante el programa o con el terminal.

Tabla de palabras %SW22 a %SW25

A cada bit de esta tabla le corresponde una macroetapa. Su uso es el siguiente:

- carga de la tabla de palabras %SW22 a %SW25 (bit para poner a 1 cuando la macroetapa correspondiente no deba ponerse a reset),
- validación con %S24.

La siguiente ilustración presenta la codificación de las palabras %SW22 a %SW25.



Ejemplo:

```
! IF %I4.2 AND %T3.D THEN
%SW22:=16#AF8F;
%SW23:=16#F3FF;
%SW24:=16#FFEF;
%SW25:=16#FFFF;
SET %S24
```

Estas cuatro palabras se inicializan en 16#FFFF si %S21 = 1.

Funcionamiento del tratamiento secuencial

Generalidades

Este tratamiento permite la programación de la estructura secuencial de aplicación.
El tratamiento secuencial comprende:
- el gráfico principal organizado en 8 páginas,
- hasta 64 macro etapas de 8 páginas cada una.

En el gráfico principal, algunos Grafcet no afines pueden programarse y tener lugar simultáneamente.

La evolución del Grafcet se efectúa en 3 grandes fases.

Fase 1

La tabla siguiente describe las operaciones realizadas en la primera fase.

Fase	Descripción
1	Evaluación de las receptividades de las transiciones validadas.
2	Petición de desactivación de las etapas arriba asociadas.
3	Petición de activación de las etapas abajo referidas

Fase 2

La fase 2 corresponde a la evolución de la situación del Grafcet en función de las transiciones franqueadas:

Fase	Descripción
1	Desactivación de las etapas hacia arriba de las transiciones franqueadas.
2	Activación de las etapas hacia abajo de las transiciones franqueadas.
3	Invalidación de las transiciones franqueadas.
4	Validación de las transiciones hacia abajo de las nuevas etapas activadas. **Resultado:** El sistema actualiza dos tablas dedicadas respectivamente a la actividad de las etapas y a la validez de las transiciones: ● la tabla de actividad de las etapas memoriza, para el ciclo en curso, las etapas activas, las etapas para activar y las etapas que se van desactivar, ● la tabla de validez de las transiciones almacena, para el ciclo en curso, las transiciones situadas hacia abajo de las etapas a que se refiere la tabla precedente

Fase 3

Las acciones asociadas a las etapas activas se ejecutan en el siguiente orden:

Fase	Descripción
1	Acciones para la desactivación de las etapas que deben desactivarse.
2	Acciones para la activación de las etapas que deben activarse.
3	Acciones continuas de las etapas activas.

Rebasamiento de las capacidades

El número de elementos de la tabla de actividad de las etapas y de la tabla de validez de las transiciones se puede configurar.
El rebasamiento de la capacidad de una o de otra comporta:

- el paso a STOP del autómata (se detiene la ejecución de la aplicación),
- el paso a 1 del bit sistema %S26 (rebasamiento de la capacidad de una de las dos tablas),
- el parpadeo del indicador ERR del autómata.

El sistema pone a disposición del usuario dos palabras del sistema:

- %SW20: palabra que permite conocer por el ciclo actual, el número de etapas activas, que se van a activar y desactivar
- %SW21: palabra que permite conocer por el ciclo actual, el número de transiciones válidas, para validar ó para invalidar.

Diagnóstico

En caso de fallo de bloqueo, las palabras del sistema %SW125 a %SW127 permiten determinar la naturaleza del fallo.

%SW125	%SW126	%SW127	
DEF7	0	= 0	Rebasamiento de la tabla de las etapas (etapas/transiciones)
DEF7	= 0	0	Rebasamiento de la tabla de las transiciones
DEFE	N° etapa	Nº de la macro etapa ó 64 para el gráfico principal	Ejecución del gráfico incorrecto (problema de transición con reenvío de destino sin resolver).

Descripción del tratamiento posterior

Generalidades

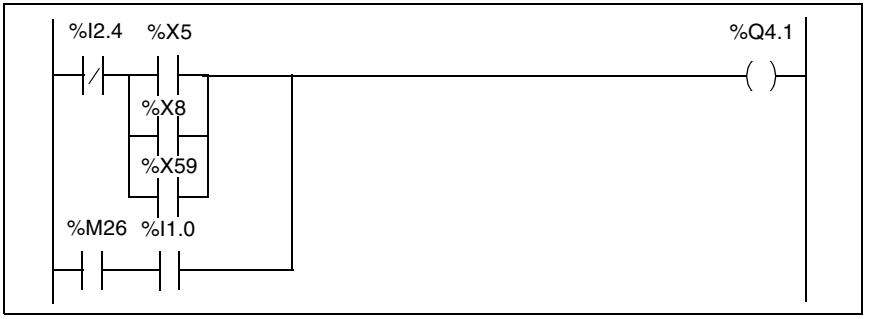
Introducido en lenguaje de contactos, en lenguaje de lista de instrucciones o en lenguaje literal, el tratamiento posterior se recuenta de arriba hacia abajo.
Este tratamiento es el último que se ha ejecutado antes de la activación de las salidas y permite programar la lógica de salida.

Acciones asociadas al Grafcet

El tratamiento posterior permite completar las consignas emitidas por el tratamiento secuencial integrando a la ecuación de una salida los modos de marcha y de parada y las seguridades indirectas específicas de la acción.
Permite, del mismo modo, tratar una salida activada varias veces durante el tratamiento secuencial.

Generalmente, se recomienda programar las aciones actuando directamente sobre el proceso en el tratamiento posterior.

Ejemplo:



- I2.4 = seguridad indirecta específica al control de la salida %Q4.1.
- %M26 = bit interno resultado de la lógica de entrada que trata los modos de marcha y de parada.
- %I1.0 = botón pulsador.

La salida %Q4.1 queda activada por las etapas 5, 8 y 59 del tratamiento secuencial.

Acciones independientes del Grafcet

El tratamiento posterior permite también programar las salidas independientes del tratamiento secuencial.

Control de la ejecución del Grafcet

Puede ser necesario controlar el proceso correcto del Grafcet comprobando el tiempo de actividad de algunas etapas. La comprobación de este tiempo se lleva a cabo comparándolo bien un valor mínimo, o bien un valor máximo determinado por el usuario.

El tratamiento de la falla se deja a criterio del usuario (señalización, procedimiento particular de funcionamiento, edición de mensaje).

Ejemplo:

```
! IF (%X2.T > 100 AND %X2) THEN SET %Q4.0 ;END_IF ;
```

Bloques de función DFB

10

Presentación

Contenido del capítulo

Este capítulo describe la programación de los bloques de función del usuario DFB.

Contenido:

Este capítulo contiene los siguiente apartados:

Apartado	Página
Presentación de los bloques de función DFB	222
De qué manera se puede poner en marcha un bloque de función DFB	224
Definición de los objetos de los bloques de función tipo DFB	226
Definición de los parámetros DFB	229
Definición de las variables DFB	230
Regla de codificación de los tipos DFB	232
Creación de instancias del DFB	234
Regla de utilización de los DFB en un programa	235
Utilización de un DFB en un programa en lenguaje de contactos	237
Utilización de un DFB en un programa en lenguaje lista de instrucciones o literal	238
Ejecución de una instancia DFB	239
Ejemplo de programación del bloque de función DFB	240

Presentación de los bloques de función DFB

Función

El programa PL7-Pro ofrece al usuario la posibilidad de crear sus propios bloques de función que respondan a las especificaciones de sus aplicaciones.

Estos bloques de función del usuario permiten estructurar una aplicación. Se emplearán en el momento en que una secuencia de programa repita varias reanudaciones de la aplicación o para inmovilizar una programación estándar (ejemplo: algoritmo de comando de un motor que incluye las seguridades locales).

Pueden transmitirse al conjunto de los programadores y utilizarse en la misma aplicación o en todas las demás aplicaciones (función exportación/importación).

Ejemplos de uso

El empleo de un bloque de función DFB en una aplicación permite:
- simplificar el diseño y el aprovechamiento del programa,
- aumentar la legibilidad del programa,
- facilitar su depuración (todas las variables introducidas por el bloque de función DFB se identifican en la interfaz),
- disminuir el volumen de códigos generado (el código correspondiente al DFB sólo se carga una vez, sea cual fuere el número de llamadas al DFB en el programa).

Comparación con la sección

En relación con la sección, permiten:
- parametrizar más fácilmente el tratamiento,
- utilizar variables internas propias al DFB, es decir, independientes de la aplicación,
- poderse comprobar con independencia de la aplicación.

Ofrecen en lenguaje de contactos una visualización gráfica del bloque, lo que facilita la programación y la depuración.
Además, los bloques de función DFB se aprovechan de los datos remanentes.

Campo de uso

La tabla que aparece a continuación describe el ámbito de aplicación de los DFB.

Función	**Ámbito**
Autómatas para los cuales los DFB son utilizables.	Premium
Programa de creación de los DFB	PL7 Pro
Programas con los cuales los DFB son utilizables.	PL7 Pro o un PL7 Junior
Lenguaje de programación para la creación del código de los DFB.	lenguaje literal estructurado y lenguaje de contactos
Lenguajes de programación con los cuales los DFB son utilizables.	lenguaje de contactos, literal estructurado y de lista de instrucciones

De qué manera se puede poner en marcha un bloque de función DFB

Procedimiento

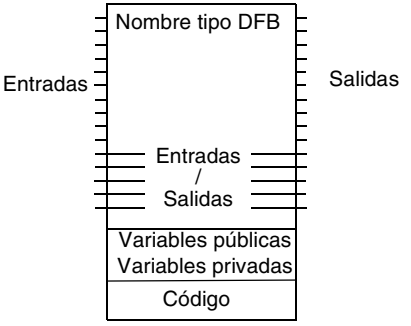
La puesta en marcha de un bloque de función DFB se lleva a cabo en 3 etapas principales:

Etapa	Acción
1	Diseño del DFB modelo (llamado: Tipo DFB).
2	Creación de una imagen de este bloque llamada instancia para cada uso de la aplicación.
3	Uso de la instancia en el programa PL7.

Diseño del tipo DFB

Se trata de definir y codificar todos los elementos que constituyen el DFB patrón, con la ayuda del editor de DFB.

La siguiente ilustración muestra la composición de un DFB patrón.



Un bloque de función Tipo DFB consta de:

- un nombre,
- parámetros:
  - entradas,
  - salidas,
  - entradas/salidas,
- de variables:
  - variables públicas,
  - variables privadas,
- un código en lenguaje literal estructurado o en lenguaje de contactos,
- un comentario,
- de una ficha descriptiva.

Creación de una instancia DFB

Una vez concebido el Tipo DFB, el usuario define una instancia del DFB con la ayuda del editor de variables o en el momento de llamar a la función en el editor de programa.

Uso de los DFB

Esta instancia del bloque se emplea a continuación como un bloque de función estándar en lenguaje de contactos, o como una función elemental en lenguaje literal estructurado o en lista de instrucciones.

Se puede programar en diferentes tareas (excepto en las tareas secuenciales) y secciones de la aplicación.

Definición de los objetos de los bloques de función tipo DFB

Características generales de los objetos DFB

Estos objetos son datos internos al DFB, son puramente simbólicos (sin direccionamiento en forma de variable).

Los DFB emplean 2 tipos de objetos:
- los parámetros
- las variables

Sintaxis

Para cada parámetro o variable utilizada, el diseñador del bloque de función Tipo DFB define:
- un nombre de 8 caracteres como máximo (se pueden emplear las letras no acentuadas, las cifras, el carácter "_"; el primer carácter debe ser una letra; se prohiben las palabras clave y los símbolos),
- un tipo de objeto (véase la tabla siguiente),
- un comentario opcional de 80 caracteres como máximo,
- un valor inicial (excepto para los parámetros Entradas/Salidas).

Tipo de objetos

La tabla que se ofrece a continuación describe la lista de los diferentes tipos de objetos posibles cuando se declaran los parámetros y las variables del tipo DFB.

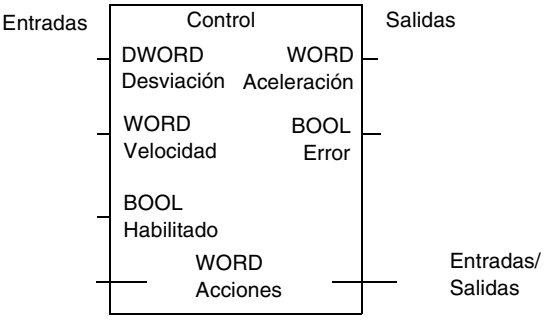
Acción sobre...	**Tipo**	**Nombre**	**Ejemplos**
Bits	BOOL	Booleano	El tipo BOOL no administra los flancos. Si la gestión del flanco no resulta útil en el tratamiento, es mejor utilizar el tipo BOOL Ejemplo de objeto del tipo BOOL del lenguaje PL7: %MWi:Xj que no administra los flancos, pero que ocupa menos memoria que el tipo EBOOL.
	EBOOL	Booleano extendido	El tipo EBOOL administra los flancos, por lo tanto, pueden ejecutarse con este tipo de parámetro o de variable las instrucciones sobre el flanco del tipo RE y FE. si, en el momento de usarlo, se desea asociar un tipo EBOOL a un parámetro de entradas/salidas, deberá ser del tipo EBOOL en el DFB. Ejemplo de objeto del tipo EBOOL del lenguaje PL7: %Mi,%Ixy.i,%Qxy.i.
Palabras	WORD	Entero 16 bits	Ejemplo de objeto del tipo WORD del lenguaje PL7: %MWi, %KWi,
	DWORD	Entero 32 bits	Ejemplo de objeto del tipo DWORD del lenguaje PL7: %MDi, %KDi,
	REAL	Real	Ejemplo de objeto del tipo REAL del lenguaje PL7: %MFi, %KFi
Tablas	AR_X	Tabla de bits	Ejemplo de objeto del tipo AR_X del lenguaje PL7: %Mi:L, %Ix.i:L
	AR_W	Tabla de entero 16 bits	Ejemplo de objeto del tipo AR_W del lenguaje PL7: %MWi:L, %KWi:L
	AR_D	Tabla de entero 32 bits	Ejemplo de objeto del tipo AR_D del lenguaje PL7: %MDi:L, %KDi:L
	AR_R	Tabla de reales	Ejemplo de objeto del tipo AR_R del lenguaje PL7: %MFi:L, %KFi:L
	STRING	Cadena de caracteres	Ejemplo de objeto del tipo STRING del lenguaje PL7: %MBi, %KBi

Nota:
- Caso de las tablas: la extensión de la tabla se deberá mencionar obligatoriamente por los parámetros de salidas y las variables públicas y privadas, por el contrario no hace falta definirlos para los parámetros de entradas y los parámetros de entradas/salidas.
- Los valores iniciales pueden definirse para las entradas (si no son del tipo tabla), para las salidas y para las variables públicas y privadas.

Definición de los parámetros DFB

Ilustración

La siguiente ilustración presenta dos ejemplos de parámetros



Descripción de los parámetros

La siguiente tabla describe la función de cada uno de los tipos de parámetros.

Parámetro	Número máximo	Función
Entradas	15 (1)	Estos son los datos para introducir en el DFB por el programa de aplicación. Estos parámetros sólo de lectura no pueden modificarse con el código del DFB.
Salidas	15 (2)	Estos son los datos elaborados por el DFB con destino al programa de aplicación.
Entradas/ Salidas	15	Estos son los parámetros de entradas que pueden modificarse con el código del DFB.

Leyenda:
(1) Número de entradas + Número de entradas/salidas inferior o igual a 15
(2) Número de salidas + Número de entradas/salidas inferior o igual a 15

Nota:
- Cualquier bloque de DFB deberá tener por lo menos una entrada booleana.
- La modificación de la interfaz de un DFB (variables públicas o parámetros) solamente es posible si no se trata de una instancia y no se utiliza en la aplicación.

Definición de las variables DFB

Descripción de las variables

La siguiente tabla describe la función de cada uno de los tipos de variables.

Variable	Número máximo	Función
Pública	100	Variables internas utilizadas en el tratamiento y a las que el usuario puede acceder a través de los ajustes o mediante el programa de aplicación, aparte del código DFB (en calidad de variable pública de instancia DFB, véase a continuación: Acceso a las variables públicas.
Privada	100	Variables internas en el código del bloque de función, estas variables se calculan y se trabajan en el propio interior del DFB, aunque no tienen ninguna conexión con el exterior del DFB. Esta variables son útiles para la programación del bloque, pero no tienen interés para el usuario del bloque (por ejemplo: variable intermedia de reenvío de una expresión combinatoria a otra, resultado de un cálculo intermedio...).

Nota: La modificación de la interfaz de un DFB (variables públicas o parámetros) solamente es posible si no se trata de una instancia y no se utiliza en la aplicación.

Acceso a las variables públicas

Únicamente son accesibles, en calidad de objetos del programa de aplicación, los parámetros de salidas y las variables públicas, aparte del cuerpo del bloque de función.
La sintaxis es la siguiente:

Nombre_DFB.Nombre_parámetro

Donde **Nombre_DFB** es el nombre que se da a la instancia del DFB utilizado (32 caracteres como máximo)

y **Nombre_parámetro** es el nombre que se da al parámetro de salidas o a la variable pública (8 caracteres como máximo).

Ejemplo: `Control.Desvío` para la salida `Desvío` de la instancia DFB denominada `Control`.

Guardado y restitución de las variables públicas

Las variables públicas, modificadas mediante programa o por ajuste, pueden guardarse en el mismo lugar que los valores de inicialización (definidos en las instancias DFB) poniendo a 1 del bit el sistema %S94.
El reemplazo sólo tendrá lugar si se obtiene la autorización en cada variable del tipo DFB.

Estos valores guardados se aplicarán de nuevo al poner a 1 el bit del sistema %S95 o cuando se reinicialice el autómata.

La inhibición de la función «**Save/Restore**» global para todos los bloques de función DFB es posible (cuadro de diálogo **Propiedades del tipo DFB**).

Regla de codificación de los tipos DFB

Generalidades

El código define el tratamiento que efectuará el bloque DFB en función de los parámetros definidos.
El código del bloque de función DFB se programa en lenguaje literal o en lenguaje de contactos.
En el caso del lenguaje literal, el DFB está formado de una sola frase literal de longitud ilimitada.

Reglas de programación

Se permiten todas las instrucciones y funciones avanzadas del lenguaje, excepto:

- la llamada a los bloques de función estándares,
- la llamada a los otros bloques de función DFB,
- ramificación a una etiqueta `JUMP`,
- la llamada al subprograma,
- la instrucción `HALT`,
- las instrucciones que usan las variables de los módulos de entradas/salidas (ej.: `READ_STS`, `SMOVE`...).

El código aprovecha los parámetros y las variables del DFB definidos por el usuario.

El código del bloque de función DFB no puede usar ni los objetos de entradas/salidas (%I,%Q...), ni los objetos globales de la aplicación (%MW,%KW...) excepto los bits y palabras del sistema %S y %SW.

Nota: no pueden usarse etiquetas

Funciones específicas

La tabla siguiente describe las funciones específicamente adaptadas para utilizarse en el código.

Funciones	**Función**
`FTON, FTOF, FTP, FPULSOR`	Estas funciones de temporización se usarán en lugar de los bloques de función de temporización estándar.
`LW, HW, COCATW`	Estas instrucciones permiten manipular las palabras y las dobles palabras.
`LENGTH_ARW, LENGTH_ARD, LENGTH_ARR`	Estas instrucciones permiten calcular las longitudes de la tabla.

Ejemplo de código

El siguiente programa ofrece un ejemplo de código literal.

```
CHR_200:=CHR_100;
CHR_114:=CHR_104;
CHR_116:=CHR_106;
RESET ARRANQUE;
(*Se incrementa 80 veces CHR_100*)
FOR CHR_102:=1 TO 80 DO
            INC CHR_100;
            WHILE((CHR_104-CHR_114)<100)DO
            IF(CHR_104>400) THEN
EXIT;
            END_IF;
            INC CHR_104;
            REPEAT
            IF(CHR_106>300) THEN
EXIT;
            END_IF;
            INC CHR_106;
            UNTIL ((CHR_100-CHR_116)>100)
            END_REPEAT;
            END_WHILE;
            (* Se produce el bucle durante CHR_106)
            IF (CHR_106=CHR_116)
            THEN EXIT;
            ELSE
               CHR_114:=CHR_104;
               CHR_116:=CHR_106;
            END_IF;
            INC CHR_200;
END_FOR;
```

Creación de instancias del DFB

Generalidades

Una instancia DFB es una copia del Tipo DFB:
- aprovecha el código del Tipo DFB, (no se da la duplicación del código),
- crea una zona de datos específicos de esta instancia, que es una copia de los parámetros y de las variables del Tipo DFB. Esta zona se sitúa en el espacio datos de aplicación.

Cada instancia DFB se identifica por un nombre de 32 caracteres, como máximo, definido por el usuario.
Los caracteres permitidos son idénticos a los autorizados para los símbolos, o sea que se autorizan:
- las letras sin acentuar,
- las cifras,
- el carácter "_".

El primer carácter debe ser una letra; las palabras clave y los símbolos quedan prohibidos.

Reglas

Es posible crear tantas instancias como sean necesarias (únicamente con el límite de la memoria del autómata) a partir de un mismo tipo de DFB.

Los valores iniciales de las variables públicas definidas por los bloques de función del Tipo DFB pueden modificarse para cada instancia.

Regla de utilización de los DFB en un programa

Generalidades

Las instancias de DFB se pueden usar en todos los lenguajes (lenguaje de contactos, literal y lista de instrucciones) y en todas las partes de aplicación: secciones, subprogramas, módulo Grafcet, (excepto en las tareas secuenciales).

Reglas generales de uso

Las reglas definidas a continuación se respetarán independientemente del lenguaje utilizado:

- se darán a conocer todos los parámetros de entradas del tipo tabla, así como los parámetros de entradas/salidas
- los parámetros de entradas sin conectar guardan el valor de la llamada anterior o el valor de inicialización si nunca se ha llamado al bloque con esta entrada reseñada o conectada.
- todos los objetos vinculados a los parámetros de entradas, de salidas y de entradas/salidas serán obligatoriamente del mismo tipo que los definidos en el momento de la creación del Tipo DFB (por ejemplo: si el tipo WORD queda definido por el parámetro de entrada "velocidad", de ningún modo puede llegar a afectar las dobles palabras %MDi, %KDi).
  Únicamente pueden mezclarse los tipos BOOL y EBOOL para los parámetros de entradas o de salidas (nunca para los de entradas/salidas).
  Ejemplo: el parámetro de entrada "Validación" puede definirse como BOOL y puede asociarse a un bit interno %Mi del tipo EBOOL, por el contrario, en el código interno del tipo DFB el parámetro de entrada tendrá la propiedad de un tipo BOOL, no sabe administrar los flancos.

Asignación de los parámetros

La siguiente tabla resume las diferentes posibilidades de asignación de los parámetros en los diferentes lenguajes de programación:

Parámetro	Tipo	Asignación de parámetro	Asignación
Entradas	Booleano	Conectado (1)	opcional (2)
	Digital	Objeto o expresión	opcional
	Tabla	Objeto	obligatorio
Entradas/ Salidas	Booleano	Objeto	obligatorio
	Digital	Objeto	obligatorio
	Tabla	Objeto	obligatorio
Salidas	Booleano	Conectado (1)	opcional
	Digital	Objeto	opcional
	Tabla	Objeto	opcional

(1) conectado en lenguaje de contactos, u objeto en lenguaje booleano o literal
(2) en lenguaje de contactos todo bloque DFB debe tener al menos una entrada booleana (binaria) conectada.

Utilización de un DFB en un programa en lenguaje de contactos

Principio

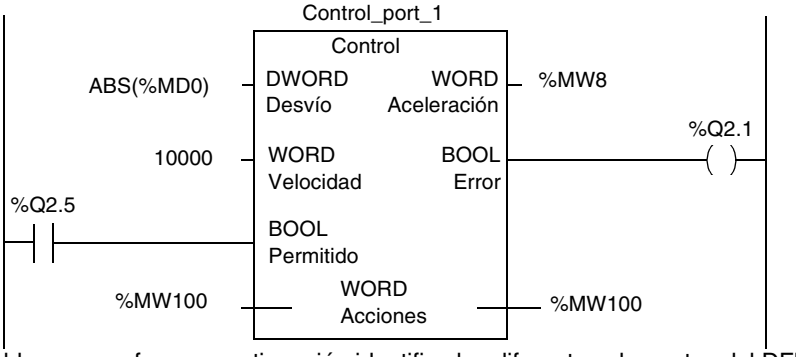
Existen dos posibilidades para llamar a un bloque de función DFB:

- una llamada textual en un bloque de operación, la sintaxis y las obligaciones sobre los parámetros son idénticas a las del lenguaje literal.
- una llamada gráfica, véase el ejemplo que se ofrece a continuación.

Los bloques funcionales DFB gráficos tienen las entradas/salidas asignado directamente por los objetos o las expresiones, estos objetos o expresiones ocupan una celda de la red gráfica.
2 bloques funcionales DFB conectados en serie deberán estar separados por 2 columnas, como mínimo

Ejemplo

La siguiente ilustración presenta un ejemplo simple de programación de un DFB.



La tabla que se ofrece a continuación identifica los diferentes elementos del DFB.

Variable	Función
1	Nombre del DFB
2	Nombre del Tipo DFB
3	Parámetro efectivo de la primera entrada
4	Parámetros de entradas (nombre y tipo)
5	Parámetros de salidas (nombre y tipo)
6	Parámetros de entradas/salidas (nombre y tipo)

Nota:

- Un bloque de función DFB deberá tener por lo menos una entrada booleana conectada.
- Las entradas, salidas o entradas/salidas digitales del bloque no están conectadas. A estas patillas quedan agrupados los objetos mencionados en la celda frente a la patilla.

Utilización de un DFB en un programa en lenguaje lista de instrucciones o literal

Generalidades

La llamada del bloque de función DFB constituye una acción, que puede situarse tanto en una frase, como en todas las otras acciones del lenguaje.

Sintaxis general

La sintaxis de programación de los DFB es como sigue:

```
Nom_DFB (E1,...,En,ES1,...,ESn,S1,...,Sn)
```

La siguiente tabla describe la función de los parámetros de las instrucciones.

Parámetros	Función
E1, ..., En	Expresiones (excepto para los objetos del tipo BOOL/EBOOL), objetos o valores inmediatos que sirven de parámetros efectivos a los parámetros de entradas.
ES1, ..., ESn	Parámetros efectivos que corresponden a las entradas/salidas; siempre son objetos de lenguajes en lectura/escritura.
S1, ..., Sn	Los parámetros efectivos que corresponden a las salidas; siempre son objetos de lenguajes en lectura/escritura.

Sintaxis literal

La instrucción en lenguaje literal presenta la siguiente sintaxis:
Nombre_DFB (E1,...,En,ES1,...,ESn,S1,...,Sn);

Ejemplo: Con_pernos(%I2.0,%MD10,%I2.1,%Q1.0);

Sintaxis en lista de instrucciones

La instrucción en lenguaje lista de instrucciones presenta la siguiente sintaxis:
[Nombre_DFB (E1,...,En,ES1,...,ESn,S1,...,Sn)]

Ejemplo:

```
LD TRUE
 [Con_pernos(%I2.0,%MD10,%I2.1,%Q1.0)]
```

Ejecución de una instancia DFB

Funcionamiento

La ejecución de una instancia DFB se efectúa en el siguiente orden:

Etapa	Acción
1	Carga de los parámetros de entradas y de entradas/salidas con la ayuda de los parámetros efectivos. Cualquier entrada que se deje libre, en la inicialización o en el rearranque en frío, toma el valor de inicialización definido en el tipo DFB, a continuación toma el valor actual del parámetro.
2	Paso según el valor de los parámetros de entradas (excepto para el tipo de tabla).
3	Paso según la dirección de los parámetros de entradas/salidas.
4	Ejecución del código literal.
5	Escritura de los parámetros de salidas.

Herramientas de depuración.

El programa PL7 ofrece diversas herramientas de depuración del programa PL7 y de los DFB:

- tablas de animación: todos los parámetros y variables públicas se muestran y se animan en tiempo real, los objetos deseados se pueden modificar y forzar,
- punto de parada, paso a paso y diagnóstico del programa,
- pantallas de explotación: para la depuración unitaria.

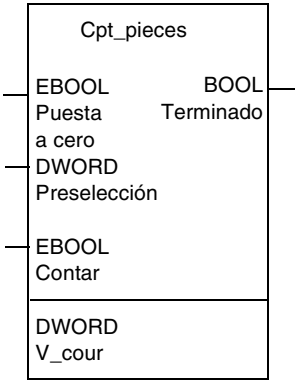
Ejemplo de programación del bloque de función DFB

Generalidades

Este ejemplo se ofrece sólo a título didáctico, el DFB que se va a programar es un contador.

Características del tipo DFB

El contador se configura a partir del siguiente Tipo DFB:



La tabla siguiente describe las características del Tipo DFB que se va a programar.

Características	**Valores**
Nombre	**Cpt_pieces**
Entradas	• **Puesta a cero**: puesta a cero del contador • **Presel.**: valor de preselección del contador • **Count**: entrada de contaje
Salidas	**Terminado**: salida del valor preseleccionado alcanzada
Variable pública	**V_cour**: Valor actual incrementado por la entrada **Contar**

Funcionamiento del contador

La tabla siguiente describe el funcionamiento que debe tener el contador.

Fase	Descripción
1	Este bloque cuenta los flancos ascendentes en la entrada **Contar**.
2	El resultado se coloca en la variable **V_cour**, este valor se pone en reset mediante un flanco ascendente en la entrada **Puesta a cero**.
3	El contaje se efectúa hasta el valor de preselección. Cuando este valor alcance la salida **Terminado** se pone a 1, se pondrá en reset en el flanco ascendente en la entrada **Puesta a cero**.

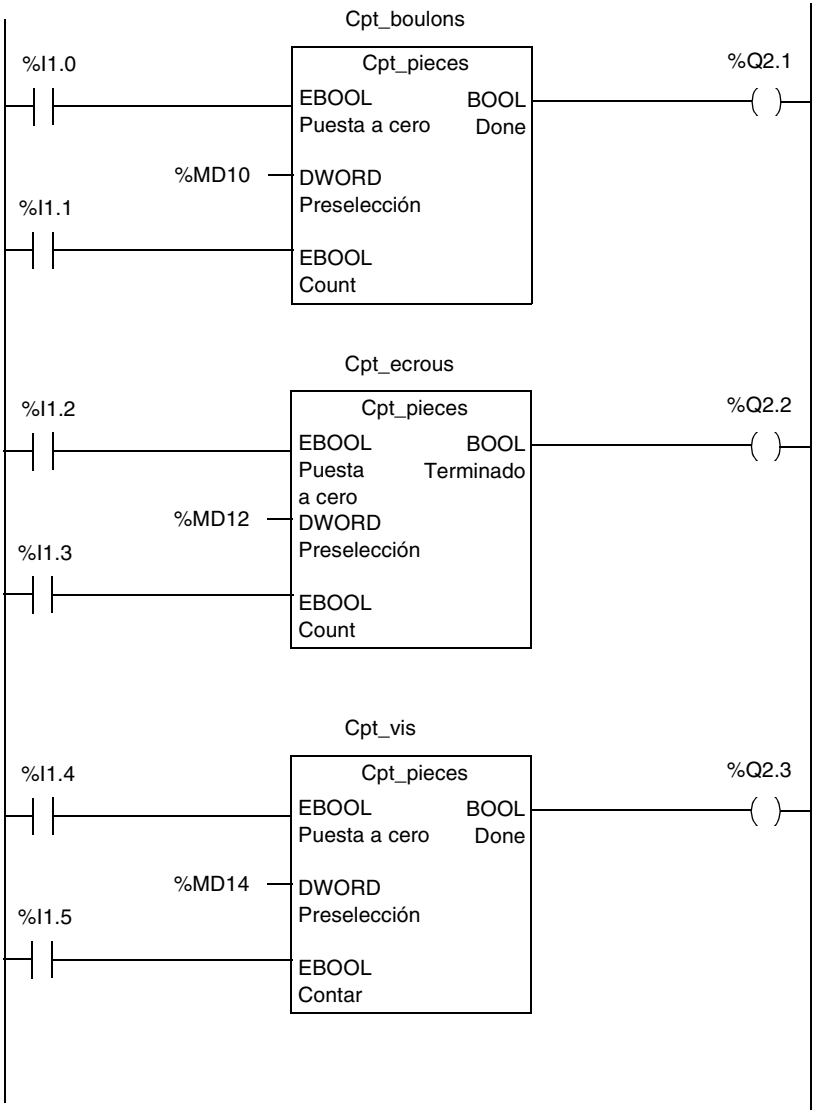
Código del DFB

La programación del código del Tipo DFB se facilita a continuación.

```
!(*Programación del DFB Cpt_pieces*)
IF RE Puesta a cero THEN
      V_cour=0;
END_IF;
IF RE Contar THEN
      V_cour=V_cour+1;
END_IF;
IF(V_cour>=Preselección) THEN
      SET Terminado;
ELSE
      RESET Terminado;
END_IF;
```

Ejemplo de uso

En este ejemplo el Tipo DFB creado, se utiliza 3 veces (3 instancias DFB) para el contaje de 3 tipos de piezas.
En el momento en que se alcanza el número de piezas programado (en las palabras %MD10, %MD12 y %MD14), la salida del contador controla la parada del sistema de aprovisionamiento de piezas correspondiente.
El programa siguiente emplea las 3 instancias del tipo DFB Cpt_pieces: Cpt_boulons, Cpt_ecrous y Cpt_vis.



Índice

A

Acción, 194
Acción continua, 197
Acción para la activación, 196
Activación, 196
Arranque en frío, 86

B

Bloque de función DFB, 222

C

Comentario, 192
  Grafcet, 192
  Lista de instrucciones, 137
  literal, 155
  Red de contactos, 123
Convergencia O, 186
Convergencia Y, 187
Convergencias en Y, 179, 180
Corte del sector, 82

D

Desvío, 186
DFB, 222
Direccionamiento
  Bus AS-i, 40
  Bus FIPIO, 37
  E/S Micro, 31
  Módulos en rack, 34
  Momentum, 37
  TBX, 37
Divergencia O, 186
Divergencia Y, 187
Divergencias en Y, 179, 180

E

Ejecución
  Cíclica, 103
  Periódica, 105
  Red de contactos, 130
Ejecución de un programa literal, 171
Elementos gráficos, 124
Enlace orientado, 191
Etapa de entrada, 204
Etapa de salida, 204
Etiqueta
  Lista de instrucciones, 136
  literal, 154
  Red de contactos, 122
EXIT, 170

F

FOR...END_FOR, 169

Frase
Lista de instrucciones, 135
literal, 153

G

Grafcet, 177

I

IF...THEN, 165
Inicialización del Grafcet, 211
Inmovilización del Grafcet, 213
Instancia DFB, 234
Instrucción
aritmética, 157
cadena de caracteres, 159
conversión, 162
Gestión del tiempo, 163
lógica, 157
objeto bits, 156
programa, 163
tablas, 159
Instrucciones
Lista de instrucciones, 138

L

Lenguaje
Literal estructurado, 152
Lenguaje de contactos, 120
Lenguajes
PL7, 17
Lista de instrucciones, 134
Literal estructurado, 152

M

Macro etapa, 203
Memoria
Bits, 66
Micro, 70
palabras, 68
Premium, 64, 73, 76, 78
TSX 37, 62, 70
TSX 57, 73, 76, 78
TSX Micro, 62
TSX57, 64
Módulo funcional, 22, 116
Monotarea, 102
Multitarea, 20, 110, 111

O

Objet
Bloque de función, 47
Objeto
Bit, 29
Booleano, 26
DFB, 226
Indexado, 53
Palabra, 27, 42
Objeto en el lenguaje PL7, 25
Objetos Grafcet, 56, 181

P

Página Grafcet, 183, 185
Parámetro
DFB, 229
PL7, 16
Posicionamiento previo del Grafcet, 210
Presimbolización, 59
Programa PL7, 16
Programación
Red de contactos, 127
Puesta cero de las macro etapas, 214

R

Reanudación del sector, 82
Rearranque en caliente, 84
Receptividad, 198
Red de contactos, 121

Reenvío de destino, 188
Reenvío de origen, 188
REPEAT...END_REPEAT, 168
Reset del Grafcet, 212

S

Sección, 20, 94
Sección Grafcet, 208
Simbolización, 57
Símbolos Grafcet, 178
Subprograma, 20, 94

T

Tabla, 50
Tarea, 20
    Maestra, 93
    Rápida, 98
Tratamiento
    De sucesos, 99, 113
    posterior, 218
Tratamiento preliminar, 209
Tratamiento secuencial, 216

V

Variable
    DFB, 230

W

WHILE...END, 167