

Unity Pro

Lenguajes y estructura del programa Manual de referencia

04/2009

Schneider Electric no asume ninguna responsabilidad ante los posibles errores que aparezcan en este documento. Si tiene alguna sugerencia para llevar a cabo mejoras o modificaciones o si ha encontrado errores en esta publicación, le rogamos que nos lo notifique.

Queda prohibido reproducir cualquier parte de este documento bajo ninguna forma o medio posible, ya sea electrónico, mecánico o fotocopia, sin autorización previa de Schneider Electric.

Deberán tenerse en cuenta todas las normas de seguridad nacionales, regionales y locales pertinentes a la hora de instalar y utilizar este producto. Por razones de seguridad y para garantizar que se siguen los consejos de la documentación del sistema, las reparaciones sólo podrá realizarlas el fabricante.

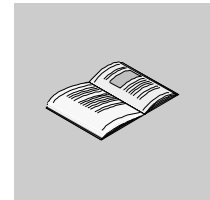
Cuando se utilicen dispositivos para aplicaciones con requisitos técnicos de seguridad, siga las instrucciones pertinentes.

Si no se utiliza el software de Schneider Electric o un software compatible con nuestros productos de hardware, pueden sufrirse daños o lesiones o provocar un funcionamiento inadecuado del dispositivo.

Si no se tiene en cuenta esta información se pueden causar daños personales o en el equipo.

© 2009 Schneider Electric. Reservados todos los derechos.

Tabla de materias



	Información de seguridad	11
	Acerca de este libro	13
Parte I	Presentación general de Unity Pro	15
Capítulo 1	Presentación	17
	Funciones de Unity Pro	18
	Interfase de usuario	23
	Explorador de proyectos	25
	Formatos de archivo de proyecto y de aplicación de usuario	27
	Configurador	31
	Editor de datos	35
	Editor de programas	43
	Diagrama de bloques de funciones (FBD)	46
	Lenguaje del diagrama de Ladder (LD)	48
	Información general sobre el lenguaje de secuencias SFC	50
	Lista de instrucciones IL	54
	Texto estructurado ST	55
	Simulador del PLC	57
	Exportación/importación	59
	Documentación de usuario	60
	Servicios de depuración	61
	Visualizador de diagnósticos	67
	Ventana de usuario	68
Parte II	Estructura de la aplicación	71
Capítulo 2	Descripción de las funciones disponibles en cada tipo de autómatas	73
	Funciones disponibles en los diferentes tipos de PLC	73
Capítulo 3	aplicación, estructura del programa	75
3.1	Descripción de las tareas y de los procesamientos	76
	Presentación de la tarea maestra	77
	Presentación de la tarea rápida	78
	Presentación de las tareas auxiliares	79
	Descripción general del procesamiento de eventos	81

3.2	Descripción de secciones y subrutinas	82
	Descripción de las secciones.	83
	Descripción de secciones de SFC.	85
	Descripción de las subrutinas	86
3.3	Ejecución monotarea	87
	Descripción del ciclo de las tareas maestras.	88
	Monotarea: Ejecución cíclica	90
	Ejecución periódica	91
	Control del tiempo de ciclo.	93
	Ejecución de las secciones de Quantum con entradas/salidas descentralizadas	94
3.4	Ejecución multitarea.	96
	Estructura del software multitarea	97
	Desglose secuencial de las tareas en una estructura multitarea.	99
	Control de tareas	101
	Asignación de los canales de entradas/salidas a las tareas maestra, rápida y auxiliares	104
	Gestión de los tratamientos de sucesos	106
	Ejecución del procesamiento de eventos de tipo TIMER	108
	Intercambios de entradas/salidas en el procesamiento de eventos	112
	Programación de procesamiento de eventos	113
Capítulo 4	Estructura de la memoria de la aplicación	115
4.1	Estructura de memoria de los autómatas Premium, Atrium y Modicon M340	116
	Estructura de memoria de los PLC Modicon M340	117
	Estructura de memoria de los PLC Premium y Atrium	121
	Descripción detallada de las zonas de memoria	124
4.2	Estructura de memoria de los autómatas Quantum	125
	Estructura de memoria de los autómatas Quantum	126
	Descripción detallada de las zonas de memoria	129
Capítulo 5	Modalidades de funcionamiento	131
5.1	Modalidades de funcionamiento de los autómatas Modicon M340	132
	Procesamiento en caso de corte y restablecimiento de la alimentación para los PLC Modicon M340	133
	Procesamiento en arranque en frío para PLC Modicon M340.	136
	Procesamiento del reinicio en caliente en los PLC Modicon M340.	141
	Inicio automático en modalidad RUN para PLC Modicon M340.	145
5.2	Modalidades de funcionamiento de los autómatas Premium, Quantum .	146
	Procesamiento en caso de corte y restablecimiento de la alimentación para PLC Premium/Quantum.	147
	Procesamiento del arranque en frío para PLC Quantum y Premium . . .	150
	Procesamiento del reinicio en caliente para PLC Quantum y Premium .	155
	Inicio automático en modalidad RUN para Premium/Quantum	158
5.3	Modalidad HALT del autómata	159
	Modalidad HALT del PLC	159

Capítulo 6	Objetos de sistema	161
6.1	Bits de sistema	162
	Introducción de bits de sistema	163
	Descripción de los bits de sistema %S0 a %S7	164
	Descripción de los bits de sistema %S9 a %S13	167
	Descripción de los bits de sistema de %S15 a %S21	169
	Descripción de los bits de sistema de %S30 a %S123	172
	Descripción de los bits de sistema de %S60 a %S79	176
	Descripción de los bits de sistema de %S80 a %S96	181
	Descripción de los bits de sistema de %S100 a %S122	185
6.2	Palabras de sistema	187
	Descripción de las palabras de sistema %SW0 a %SW11	188
	Descripción de las palabras de sistema %SW12 a %SW29	192
	Descripción de las palabras de sistema de %SW30 a %SW47	198
	Descripción de las palabras de sistema %SW48 a %SW59	200
	Descripción de las palabras de sistema de %SW70 a %SW100	203
	Descripción de las palabras de sistema de %SW108 a %SW116	214
	Descripción de las palabras de sistema %SW124 a %SW127	215
6.3	Palabras de sistema específicas de Atrium/Premium	217
	Descripción de las palabras de sistema %SW60 a %SW65	218
	Descripción de las palabras de sistema de %SW128 a %SW143	223
	Descripción de las palabras de sistema de %SW144 a %SW146	224
	Descripción de las palabras de sistema de %SW147 a %SW152	226
	Descripción de la palabra de sistema %SW153	227
	Descripción de la palabra de sistema %SW154	229
	Descripción de las palabras de sistema Premium/Atrium de %SW155 a %SW167	230
6.4	Palabras de sistema específicas de Quantum	231
	Descripción de las palabras de sistema Quantum %SW60 a %SW123	232
	Descripción de las palabras de sistema Quantum de %SW98 a %SW100	236
	Descripción de las palabras de sistema Quantum de %SW110 a %SW179	237
	Descripción de las palabras de sistema Quantum de %SW180 a %SW640	240
6.5	Palabras de sistema específicas de Modicon M340	248
	Descripción de las palabras de sistema: %SW146 y %SW147, %SW160 a %SW167	248
Parte III	Descripción de los datos	249
Capítulo 7	Presentación general de los datos	251
	General	252
	Descripción general de las familias de tipos de datos	253
	Descripción general de instancias de datos	255
	Descripción general de referencias de datos	257
	Reglas sintácticas de los nombres de tipos/instancias	258

Capítulo 8	Tipos de datos	259
8.1	Tipos de datos elementales (EDT) con formato Binario	260
	Descripción general de tipos de datos con formato binario	261
	Tipos booleanos	263
	Tipos enteros	268
	El tipo Time	270
8.2	Tipos de datos elementales (EDT) con formato BCD	271
	Descripción general de tipos de datos con formato BCD	272
	El tipo Date	274
	El tipo Time of Day (TOD)	275
	El tipo Date and Time (DT)	276
8.3	Tipos de datos elementales (EDT) con formato Real	278
	Presentación del tipo de datos REAL	278
8.4	Tipo de datos elementales (EDT) con formato de cadena de caracteres	281
	Descripción general de tipos de datos con formato de cadena de caracteres	281
8.5	Tipos de datos elementales (EDT) con formato de cadena de bits	284
	Descripción general de los tipos de datos con formato de cadena de bits	285
	Tipos de cadena de bits	286
8.6	Tipos de datos derivados (DDT/IODDT)	288
	Matrices	289
	Estructuras	292
	Descripción general de la familia de tipos de datos derivados (DDT)	293
	DDT: normas de asignación	296
	Descripción general de los tipos de datos derivados de entradas/salidas (IODDT)	299
8.7	Tipos de datos de bloques de función (DFB\EFB)	301
	Descripción general de las familias de tipos de datos de bloques de funciones	302
	Características de los tipos de datos de bloques de funciones (EFB\DFB)	304
	Características de elementos que pertenecen a bloques de funciones	306
8.8	Tipos de datos genéricos (GDT)	309
	Descripción general de los tipos de datos genéricos	309
8.9	Tipos de datos pertenecientes a las gráficas de funciones secuenciales (SFC)	311
	Descripción general de los tipos de datos de la familia de gráficas de funciones secuenciales	311
8.10	Compatibilidad entre los tipos de datos	313
	Compatibilidad entre tipos de datos	313
Capítulo 9	Instancias de datos	317
	Instancias de tipos de datos	318
	Atributos de instancias de datos	322
	Instancias de datos con direccionamiento directo	324

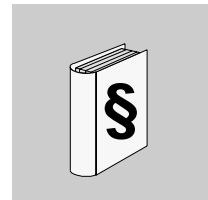
Capítulo 10	Referencias de datos	331
	Referencias de instancias de datos por valores	332
	Referencias de instancias de datos por nombre	334
	Referencias de instancias de datos por direcciones	337
	Reglas de denominación de los datos	341
Parte IV	Lenguajes de programación	343
Capítulo 11	función, lenguaje de bloques FBD	345
	Generalidades sobre el lenguaje de bloques de funciones FBD	346
	Funciones elementales, bloques de funciones elementales, bloques de funciones derivados y procedimientos (FFB)	348
	Llamadas de subrutina	358
	Elementos de control	359
	Conexión	360
	Objeto de texto	362
	Secuencia de ejecución de FFB	363
	Modificación de la secuencia de ejecución	366
	Configuración de bucles	371
Capítulo 12	Diagrama de contactos (LD)	373
	Generalidades sobre el lenguaje de diagrama de contactos LD	374
	Contactos	377
	Bobinas	379
	Funciones elementales, bloques de función elementales, bloques de función derivados y procedimientos (FFB)	381
	Elementos de control	392
	Bloques de funcionamiento y de comparación	393
	Conexiones	396
	Objeto de texto	400
	Detección de flancos	401
	Secuencia de ejecución y flujo de señal	411
	Configuración de bucles	413
	Modificación de la secuencia de ejecución	415
Capítulo 13	SFC, lenguaje de ejecución secuencial	421
13.1	Generalidades sobre el lenguaje de ejecución secuencial SFC	422
	Generalidades sobre el lenguaje de ejecución secuencial SFC	423
	Reglas de conexión	427
13.2	Paso y paso de macro	428
	Paso	429
	Pasos de macro y secciones de macro	433
13.3	Acción y sección de acción	438
	Acción	439
	Sección de acción	441
	Descriptor	442

13.4	Transición y sección de transición	445
	Transición	446
	Sección de transición	448
13.5	Salto	450
	Salto	450
13.6	Conexión	451
	Conexión	451
13.7	Bifurcaciones y conjunciones	453
	Bifurcaciones alternativas y conjunciones alternativas	454
	Bifurcación simultánea y conjunción simultánea	456
13.8	Objeto de texto	458
	Objeto de texto	458
13.9	Single-Token	459
	Secuencia de ejecución en Single-Token	460
	Cadena alternativa	461
	Salto de cadena y bucles de cadena	462
	Cadenas simultáneas	465
	Selección asimétrica de cadenas simultáneas	467
13.10	Multi-Token	471
	Secuencia de ejecución en Multi-Token	472
	Cadena alternativa	474
	Cadenas simultáneas	477
	Salto a una cadena simultánea	481
	Salto desde una cadena simultánea	483
Capítulo 14	Lista de instrucciones (IL)	487
14.1	Generalidades sobre la lista de instrucciones IL	488
	Generalidades sobre la lista de instrucciones IL	489
	Operandos	492
	Modificador	494
	Operadores	496
	Llamada de subrutina	507
	Etiquetas y saltos	508
	Comentario	510
14.2	Llamada de funciones elementales, módulos de función elementales, módulos de función derivados y procedimientos	511
	Llamada de funciones elementales	512
	Llamada de bloques de funciones elementales y bloques de funciones derivados	517
	Llamada de procedimientos	529
Capítulo 15	Texto estructurado (ST)	537
15.1	Generalidades sobre el texto estructurado ST	538
	Generalidades sobre el texto estructurado (ST)	539
	Operandos	542
	Operadores	544

15.2	Instrucciones	549
	Instrucciones	550
	Asignación	551
	Instrucción de selección IF...THEN...END_IF	554
	Instrucción de selección ELSE	556
	Instrucción de selección ELSIF...THEN	557
	Instrucción de selección CASE...OF...END_CASE	559
	Instrucción de repetición FOR...TO...BY...DO...END_FOR	560
	Repetición de la instrucción WHILE...DO...END_WHILE	563
	Instrucción de repetición REPEAT...UNTIL...END_REPEAT	564
	Instrucción de repetición EXIT	565
	Llamada de subrutina	566
	RETURN	567
	Instrucción vacía	568
	Etiquetas y saltos	569
	Comentario	570
15.3	Llamada de funciones elementales, módulos de función elementales, módulos de función derivados y procedimientos	571
	Llamada de funciones elementales	572
	Llamada de bloques de funciones elementales y bloques de funciones derivados	577
	Procedimientos	586
Parte V	Bloques de funciones del usuario (DFB)	593
Capítulo 16	Presentación de los bloques de funciones del usuario (DFB)	595
	Introducción a los bloques de funciones del usuario	596
	Implementación de un bloque de función DFB	598
Capítulo 17	Descripción de los bloques de funciones del usuario (DFB)	601
	Definición de datos internos de bloques de funciones DFB	602
	Parámetros de DFB	605
	Variables de DFB	609
	Sección de código de DFB	611
Capítulo 18	Instancia de los bloques de funciones del usuario (DFB)	613
	Creación de una instancia de DFB	614
	Ejecución de una instancia de DFB	616
	Ejemplo de programación de un bloque de función derivado (DFB)	617

Capítulo 19	Utilización de los DFB a partir de los diferentes lenguajes de programación	621
	Reglas de uso de los DFB en un programa	622
	Utilización de los IODDT en un DFB	626
	Utilización de un DFB en un programa en lenguaje de contactos	629
	Utilización de un DFB en un programa en literal estructurado	631
	Utilización de un DFB en un programa en lista de instrucciones	634
	Utilización de un DFB en un programa en lenguaje de bloques funcionales	638
Capítulo 20	usuario, DFB de diagnóstico	641
	Presentación de los DFB de diagnóstico de usuario	641
Apéndices		643
Apéndice A	Códigos y valores de error de EFB	645
	Tabla de códigos de error de la librería base	646
	Tabla de códigos de error de la librería de diagnóstico	648
	Tablas de códigos de error de la librería de comunicación	649
	Tablas de códigos de error para la librería de gestión de E/S	654
	Tabla de códigos de error de la librería CONT_CTL	665
	Tabla de códigos de error de la biblioteca de movimiento	675
	Tablas de códigos de error de la biblioteca obsoleta	679
	Errores comunes de coma flotante	691
Apéndice B	Conformidad con IEC	693
B.1	Información general relativa a la norma IEC 61131-3	694
	Información general relativa a la conformidad con la norma IEC 61131-3	694
B.2	Tablas de conformidad con las normas IEC	696
	Elementos comunes	697
	Elementos del lenguaje IL	710
	Elementos de lenguaje ST	712
	Elementos gráficos comunes	714
	Elementos del lenguaje LD	715
	Parámetros dependientes del tipo de implementación	716
	Condiciones de error	720
B.3	Extensiones de la norma IEC 61131-3	722
	Extensiones de la norma IEC 61131-3, 2ª edición	722
B.4	Sintaxis de lenguajes textuales	724
	Sintaxis de lenguaje textual	724
Glosario		725
Índice		753

Información de seguridad



Información importante

AVISO

Lea atentamente estas instrucciones y observe el equipo para familiarizarse con el dispositivo antes de instalarlo, utilizarlo o realizar su mantenimiento. Los mensajes especiales que se ofrecen a continuación pueden aparecer a lo largo de la documentación o en el equipo para advertir de peligros potenciales o para ofrecer información que aclara o simplifica los distintos procedimientos.



La inclusión de este icono en una etiqueta de peligro o advertencia indica un riesgo de descarga eléctrica, que puede provocar lesiones si no se siguen las instrucciones.



Éste es el icono de alerta de seguridad. Se utiliza para advertir de posibles riesgos de lesiones. Observe todos los mensajes que siguen a este icono para evitar posibles lesiones o incluso la muerte.



PELIGRO

PELIGRO indica una situación inminente de peligro que, si no se evita, **provocará** lesiones graves o incluso la muerte.



ADVERTENCIA

ADVERTENCIA indica una situación potencialmente peligrosa que, si no se evita, **puede provocar la** muerte o lesiones graves.

AVISO

AVISO indica una situación potencialmente peligrosa que, si no se evita, puede provocar lesiones leves o moderadas.

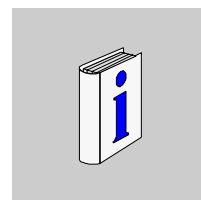
AVISO

AVISO , utilizado sin el símbolo de alerta de seguridad, indica una situación potencialmente peligrosa que, si no se evita, puede provocar daños en el equipo.

TENGA EN CUENTA

La instalación, manejo, puesta en servicio y mantenimiento de equipos eléctricos deberán ser realizados sólo por personal cualificado. Schneider Electric no se hace responsable de ninguna de las consecuencias del uso de este material.

Acerca de este libro



Presentación

Objeto

Este manual describe los elementos necesarios para la programación de los PLC Premium, Atrium y Quantum mediante el taller de programación de Unity Pro.

Campo de aplicación

Esta documentación es válida para el software Unity Pro v4.1.

Información relativa al producto

ADVERTENCIA

FUNCIONAMIENTO INESPERADO DEL EQUIPO

La aplicación de este producto requiere experiencia en el diseño y la programación de sistemas de control. Sólo las personas con dicha experiencia deberían tener permiso para programar, instalar, alterar y aplicar este producto.

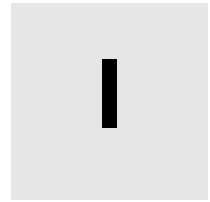
Siga todos los estándares y códigos de seguridad nacionales y locales.

Si no se siguen estas instrucciones pueden producirse lesiones personales graves o mortales o daños en el equipo.

Comentarios del usuario

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

Presentación general de Unity Pro



Presentación

1

Visión general

En este capítulo se describe la estructura y el comportamiento en general de un proyecto creado con Unity Pro.

Contenido de este capítulo

Este capítulo contiene los siguiente apartados:

Apartado	Página
Funciones de Unity Pro	18
Interfase de usuario	23
Explorador de proyectos	25
Formatos de archivo de proyecto y de aplicación de usuario	27
Configurador	31
Editor de datos	35
Editor de programas	43
Diagrama de bloques de funciones (FBD)	46
Lenguaje del diagrama de Ladder (LD)	48
Información general sobre el lenguaje de secuencias SFC	50
Lista de instrucciones IL	54
Texto estructurado ST	55
Simulador del PLC	57
Exportación/importación	59
Documentación de usuario	60
Servicios de depuración	61
Visualizador de diagnósticos	67
Ventana de usuario	68

Funciones de Unity Pro

Plataformas de hardware

Unity Pro admite las plataformas de hardware siguientes:

- Modicon M340
- Premium
- Atrium
- Quantum

Lenguajes de programación

Unity Pro proporciona los lenguajes de programación siguientes para crear el programa de usuario:

- Diagrama de bloques funcionales (FBD)
- Lenguaje de diagrama de contactos (LD)
- Lista de instrucciones (IL)
- Texto estructurado (ST)
- Control secuencial (SFC).

Todos estos lenguajes de programación pueden utilizarse juntos en el mismo proyecto.

Todos ellos cumplen la norma IEC 61131-3.

Bibliotecas de bloques

Entre los bloques incluidos en el paquete de las completas bibliotecas de bloques Unity Pro se encuentran desde bloques para realizar simples operaciones booleanas, pasando por bloques para operaciones con cadenas de caracteres y matrices, hasta bloques para controlar complejos bucles de control.

Para obtener una vista general más sencilla, los distintos bloques se organizan en bibliotecas que a su vez se desglosan en familias.

Estos bloques pueden utilizarse en los lenguajes de programación FBD, LD, IL y ST.

Elementos de un programa

Un programa puede estar formado por los siguientes elementos:

- Una tarea maestro (MAST)
- Una tarea rápida (FAST)
- De una a cuatro tareas auxiliares (no disponibles para Modicon M340)
- Secciones, asignadas a una de las tareas definidas
- Secciones para procesar los eventos controlados de temporizador (Timerx)
- Secciones para procesar los eventos controlados de hardware (EVTx)
- Secciones de subrutina (SR)

Paquetes de software

Se encuentran disponibles los siguientes paquetes de software:

- Unity Pro S
- Unity Pro M
- Unity Pro L
- Unity Pro XL
- Unity Pro XLS
- Unity Developers Edition (UDE).

Campo de aplicación del rendimiento

En la tabla siguiente se muestran las características principales de los paquetes de software individuales:

	Unity Pro S	Unity Pro M	Unity Pro L	Unity Pro XL	Unity Pro XLS
Lenguajes de programación					
Diagrama de bloques funcionales (FBD)	+	+	+	+	+
Lenguaje de diagrama de contactos (LD)	+	+	+	+	+
Lista de instrucciones (IL)	+	+	+	+	+(2)
Texto estructurado (ST)	+	+	+	+	+(2)
Lenguaje secuencial (SFC)	+	+	+	+	+(2)
Bibliotecas (1)					
Biblioteca estándar	+	+	+	+	+(2)
Biblioteca de controles	+	+	+	+	+(2)
Biblioteca de comunicación	+	+	+	+	+(2)
Biblioteca de diagnóstico	+	+	+	+	+(2)
Biblioteca de gestión de E/S	+	+	+	+	+(2)
Biblioteca del sistema	+	+	+	+	+(2)
Biblioteca de dirección de control de movimiento	-	+	+	+	+(2)
Biblioteca de TCP abierta	-	opcional	opcional	opcional	opcional (2)
Biblioteca obsoleta	+	+	+	+	+(2)

	Unity Pro S	Unity Pro M	Unity Pro L	Unity Pro XL	Unity Pro XLS
Biblioteca MFB	+	+	+	+	+(2)
Biblioteca de seguridad	-	-	-	-	+
Biblioteca de gestión de ficheros de tarjeta de memoria	+	+	+	+	+(2)
Información general					
Creación y uso de estructuras de datos (DDT)	+	+	+	+	+(2)
Creación y uso de bloques funcionales derivados (DFB)	+	+	+	+	+(2)
Navegador de proyectos con vista funcional o estructural	+	+	+	+	+
Gestión de derechos de acceso	+	+	+	+	+
Pantalla de operario	+	+	+	+	+
Visualizador de diagnósticos	+	+	+	+	+
Diagnósticos de sistema	+	+	+	+	+
Diagnósticos de proyecto	+	+	+	+	+(2)
Convertidor de aplicaciones	-	Convertidor PL7	Convertidor PL7 Convertidor de Concept	Convertidor PL7 Convertidor de Concept	Convertidor PL7 Convertidor de Concept
Gestión multiestación	-	-	-	-	-
Plataformas admitidas					
Modicon M340	BMX P34 1000 BMX P34 20••	BMX P34 1000 BMX P34 20••	BMX P34 1000 BMX P34 20••	BMX P34 1000 BMX P34 20••	BMX P34 1000 BMX P34 20••

	Unity Pro S	Unity Pro M	Unity Pro L	Unity Pro XL	Unity Pro XLS
Premium	-	P57 0244M P57 CA 0244M P57 CD 0244M P57 104M P57 154M P57 1634M P57 204M P57 254M P57 2634M H57 24M	Todas las CPU excepto: P57 554M P57 5634M	Todas las CPU	Todas las CPU
Quantum	-	-	140 CPU 311 10, 140 CPU 434 12 U/A* 140 CPU 534 14 U/A* * Actualización con el SO de Unity	CPU 311 10 CPU 534 14 U/A CPU 651 50 CPU 652 60 CPU 651 60 CPU 671 60	CPU 311 10 CPU 434 12 U/A CPU 534 14 U/A CPU 651 50 CPU 651 60 CPU 652 60 CPU 671 60 CPU 651 60 S CPU 671 60 S
Atrium	-	PCI 57 204	Todas las CPU	Todas las CPU	Todas las CPU
Simulador	+	+	+	+	+
Apertura					
Hipervínculos	+	+	+	+	+
Unity Pro Server (para OFS, UDE, UAG)	-	-	-	+	+
Componentes de software incluidos en el paquete de software					
Documentación como la ayuda contextual y el PDF	+	+	+	+	+
Herramienta del cargador del SO + Firmware HW	+	+	+	+	+
Unity Loader	+	+	+	+	+

+ = disponible

+ (1) = La disponibilidad de los bloques depende de las plataformas (véase *Unity Pro, Estándar, Librería de bloques*) de hardware.

+ (2) = Disponible en todos los PLC excepto en las plataformas CPU 651 60 S, CPU 671 60 S.

- = no disponible

Convención sobre nomenclatura

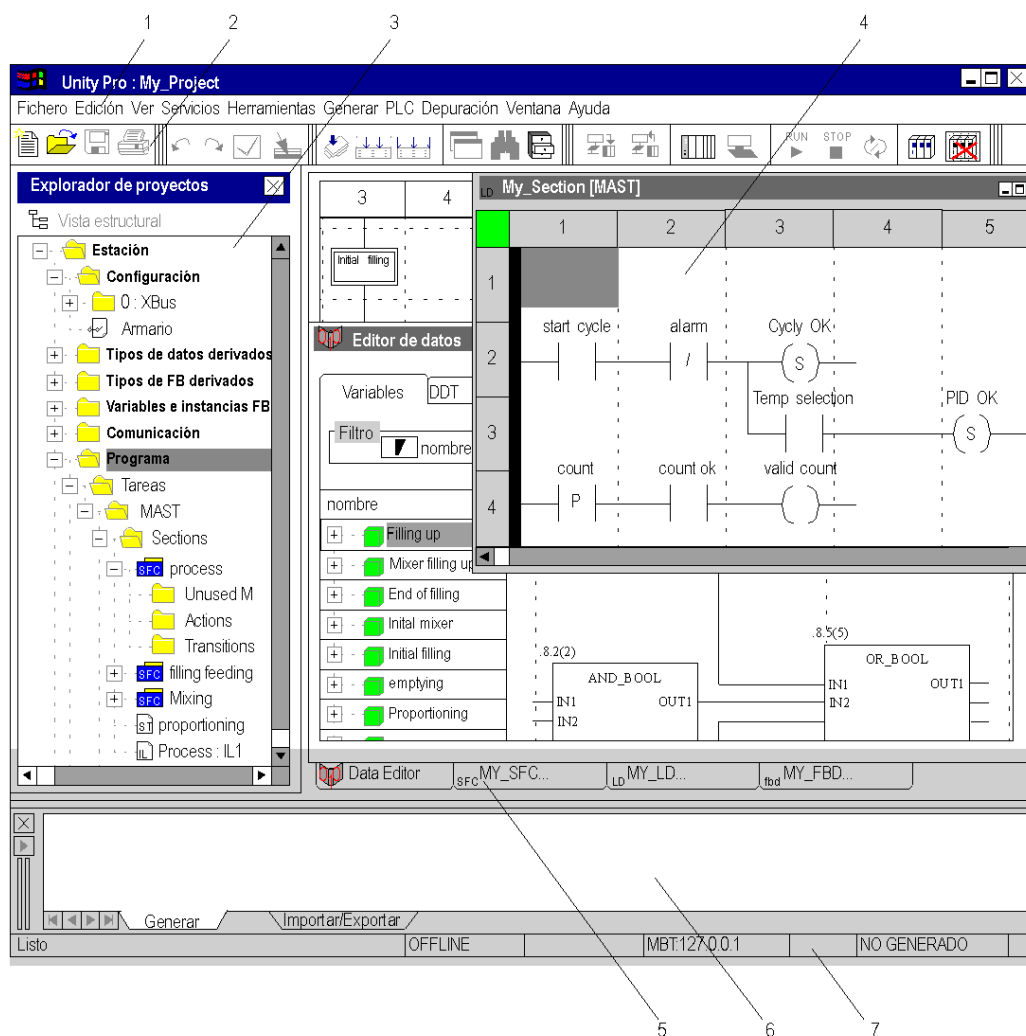
En la documentación siguiente, "Unity Pro" se utiliza como denominación general de "Unity Pro S", "Unity Pro M", "Unity Pro L", "Unity Pro XL" y "Unity Pro XLS".

Interfase de usuario

Vista general

La interfase de usuario está compuesta por varias ventanas y barras de herramientas que se pueden organizar de forma libre.

Interfase de usuario



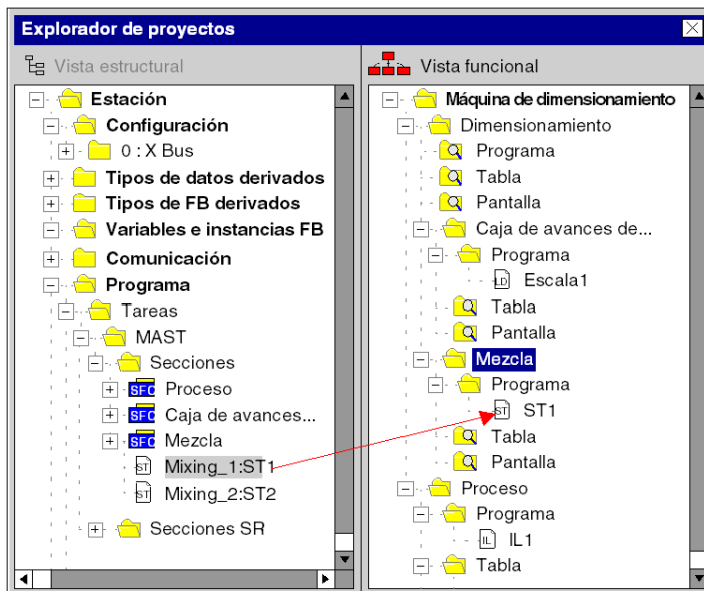
Leyenda:

Número	Descripción
1	Barra de menús (<i>véase Unity Pro, Modalidades de funcionamiento,)</i>
2	Barra de herramientas (<i>véase Unity Pro, Modalidades de funcionamiento,)</i>
3	Explorador de proyectos (<i>véase Unity Pro, Modalidades de funcionamiento,)</i>
4	Ventana de editor (editores de lenguajes de programación, editor de datos, etc.)
5	Fichas para el acceso directo a la ventana del editor
6	Ventana de resultados (<i>véase Unity Pro, Modalidades de funcionamiento,)</i> (proporciona información acerca de los errores producidos, del seguimiento de señales, de las funciones de importación, etc.).
7	Barra de estado (<i>véase Unity Pro, Modalidades de funcionamiento,)</i>

Explorador de proyectos

Introducción

El explorador de proyectos muestra todos los parámetros del proyecto. La visualización se puede efectuar como una vista estructural (topológica) o una vista funcional.



Vista estructural

En la vista estructural, el explorador de proyectos ofrece, entre otras, las siguientes prestaciones:

- Crear y eliminar elementos
- El símbolo de la sección indica el lenguaje de programación de la sección y si ésta está protegida (si hay una sección vacía, el símbolo aparece en gris).
- Visualización de las propiedades de los elementos
- Creación de directorios del usuario
- Inicio de los diversos editores
- Inicio de la función de importación/exportación

Vista funcional

En la vista funcional, el explorador de proyectos ofrece, entre otras, las siguientes prestaciones:

- Creación de módulos funcionales
- Insertar secciones, tablas de animación, etc. mediante la función Arrastrar y soltar desde la vista estructural
- Creación de secciones
- Visualización de las propiedades de los elementos
- Inicio de los diversos editores
- El símbolo de la sección muestra el lenguaje de programación de la sección y otros atributos.

Formatos de archivo de proyecto y de aplicación de usuario

Introducción

Unity Pro gestiona tres tipos de archivos para almacenar proyectos y aplicaciones de usuario. Cada tipo de archivo puede utilizarse según las necesidades específicas.

Los tipos de archivo se pueden identificar por su extensión:

- *.STU: Archivo Unity Pro.
- *.STA: Archivo de aplicación archivada de Unity Pro.
- *.XEF: Archivo de intercambio de aplicación Unity Pro.

Archivo STU

Este tipo de archivo se utiliza para tareas de trabajo diarias. Este formato se utiliza de manera predeterminada al abrir o guardar un proyecto de usuario.

La tabla siguiente presenta las ventajas y los inconvenientes de los archivos *STU* :

Ventajas	Inconvenientes
• El proyecto se puede guardar en cualquier fase (coherente o incoherente) mediante el comando predeterminado.	• No resulta útil al transferir un proyecto debido al tamaño muy grande del archivo.
• El almacenamiento y la apertura del proyecto son rápidos, ya que toda la base de datos interna está presente en el archivo.	• No es compatible cuando se actualiza Unity Pro de una versión a otra.

Archivo STA

Este tipo de archivo se utiliza para archivar proyectos y sólo se puede crear cuando el proyecto se ha generado. Este tipo de archivo permite la compatibilidad entre distintas versiones de Unity Pro.

Existen dos maneras de crear un archivo **STA** :

- el **archivo STA** se puede crear **manualmente** accediendo al menú **Archivo** → **Archivar** de la ventana principal de Unity Pro.
- el **archivo STA** se crea **de forma automática** cada vez que se guarda el proyecto como archivo **STU** si se encuentra en estado **Generado** .

NOTA: El archivo STA creado automáticamente se guarda automáticamente en el mismo directorio y con el mismo nombre de archivo que el archivo de proyecto STU, excepto que se añade un **sufijo “.Auto”** al nombre del archivo. Si ya existe un archivo STA automático, **se sobrescribe** sin confirmar nada.

NOTA: Si el proyecto está en estado **Generado** al guardar un archivo STU mediante un Unity Pro Server también se crea un archivo STA.

Para abrir un archivo *STA* se debe acceder al menú **Archivo** → **Abrir** de la ventana principal de Unity Pro.

NOTA: En la ventana del menú **Abrir** el tipo de archivo seleccionado debe ser *Archivo de aplicaciones archivadas (STA) de Unity Pro*.

- Para obtener más información sobre la creación de un archivo *STA* consulte el Manual de instalación de Unity Pro (véase *Unity Pro, Manual de instalación*):
Crear un archivo de aplicación archivada de Unity Pro (véase *Unity Pro, Manual de instalación*).
- Para obtener más información sobre la apertura de un archivo *STA* consulte el Manual de instalación de Unity Pro (véase *Unity Pro, Manual de instalación*):
Restauración de un archivo de aplicación archivada de Unity Pro (véase *Unity Pro, Manual de instalación*).

La tabla siguiente presenta las ventajas y los inconvenientes de los archivos *STA* :

Ventajas	Inconvenientes
● Almacenamiento rápido de proyecto.	● Sólo se puede crear cuando el proyecto se ha generado.
● Los proyectos se pueden compartir a través de correo electrónico o soportes de memoria de tamaño reducido.	● La apertura del proyecto es larga, ya que el archivo del proyecto se reconstruye antes de la operación.
● Posibilidad de conectar en Modalidad online idéntica con el PLC después de abrir el proyecto en una versión nueva de Unity Pro. Para obtener más información, consulte Conexión/Desconexión (véase <i>Unity Pro, Modalidades de funcionamiento</i>,) en el manual Modalidades de funcionamiento (véase <i>Unity Pro, Modalidades de funcionamiento</i>,) .	
● Permitir modificaciones en línea con el PLC sin ninguna descarga previa en el mismo.	
● El archivo <i>STA</i> generado es compatible con todas las versiones de Unity Pro. NOTA: Para poder cargar un archivo <i>STA</i> creado con otra versión de Unity Pro, todas las funciones usadas en la aplicación deben admitirse en la versión actual.	

Archivo XEF

Este tipo de archivo se utiliza para exportar proyectos en un formato de origen *XML* y se puede crear en cualquier fase de un proyecto.

Para exportar un archivo *XEF* se debe acceder al menú **Archivo** → **Exportar proyecto** de la ventana principal de Unity Pro.

Para importar un archivo *XEF* se debe acceder al menú **Archivo** → **Abrir** de la ventana principal de Unity Pro.

NOTA: En la ventana del menú **Abrir** el tipo de archivo seleccionado debe ser *Archivo de intercambio de aplicación (XEF) de Unity Pro*.

Para obtener más información sobre la creación de un archivo *XEF* consulte el Manual de instalación de Unity Pro (véase *Unity Pro, Manual de instalación*): Crear un archivo de intercambio de aplicación Unity Pro (véase *Unity Pro, Manual de instalación*).

Para obtener más información sobre la restauración de un archivo *XEF* consulte el Manual de instalación de Unity Pro (véase *Unity Pro, Manual de instalación*): Restauración de un archivo de intercambio de aplicación Unity Pro (véase *Unity Pro, Manual de instalación*).

La tabla siguiente presenta las ventajas y los inconvenientes de los archivos *XEF* :

Ventajas	Inconvenientes
El formato de origen <i>XML</i> garantiza la compatibilidad del proyecto con cualquier versión de Unity Pro.	Tamaño medio.
	La apertura del proyecto tarda un tiempo mientras éste se importa antes de la operación.
	La generación del proyecto es obligatoria para volver a ensamblar el código binario del proyecto.
	El funcionamiento con el PLC requiere volver a crear el proyecto y realizar una descarga en el procesador.
	La conexión con el PLC en Modalidad online idéntica con un archivo <i>XEF</i> no es posible. Para obtener más información, consulte Conexión/Desconexión (véase <i>Unity Pro, Modalidades de funcionamiento</i>,) en el manual Modalidades de funcionamiento (véase <i>Unity Pro, Modalidades de funcionamiento</i>,) .

Información importante

Los archivos *STU* no son compatibles entre versiones de Unity Pro. Para usar un proyecto con una versión distinta de Unity Pro, los usuarios deben almacenar los:

- Archivos de aplicación archivada de Unity Pro (*STA*):
Con el archivo *STA* es posible volver a usar el proyecto creado actual con la nueva versión de Unity Pro instalada en el ordenador.

- Archivos de intercambio de aplicación Unity Pro (*XEF*):
El archivo *XEF* debe usarse si el proyecto se ha creado.

Comparación de tipos de archivo

La tabla siguiente ofrece un resumen de los tres tipos de archivo:

Tipos de archivo	<i>STU</i>	<i>STA</i>	<i>XEF</i>
Aplicaciones binarias	Sí	Sí	No
Aplicaciones de origen	Sí	Sí	Sí
Base de datos interna	Sí	No	No
Comparación del tamaño de archivo	10, consulte (1)	0,03, consulte (1)	3
Comparación del tamaño de almacenamiento	10	1,6	6
Comparación del tiempo de apertura	1	10	10
Conexión con el PLC en Modalidad online idéntica	Posible	Posible	No es posible, consulte (2)
Copia de seguridad de archivo	Posible	Posible, consulte (3)	Posible

(1): Archivos comprimidos.

(2): Es necesario cargar el proyecto antes en el PLC.

(3): El proyecto sólo se puede guardar si se ha generado.

NOTA: Los valores de la tabla representan un porcentaje entre tipos de archivo, donde el valor de *STU* es la referencia.

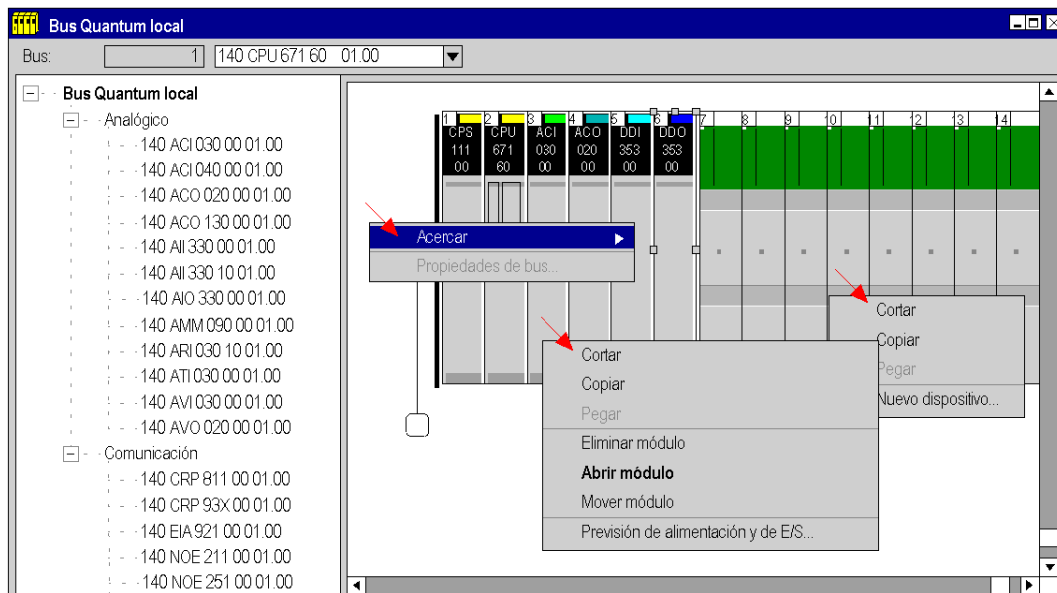
Configurador

Ventana del configurador

La ventana del configurador está dividida en dos ventanas:

- Ventana de catálogo
Desde esta ventana, es posible seleccionar un módulo e insertarlo en la representación gráfica de la configuración del PLC directamente mediante la función Arrastrar y soltar.
- Representación gráfica de la configuración del PLC

Representación de la ventana del configurador



En función de la posición del puntero del ratón, se abrirá uno de los siguientes menús contextuales:

- Si coloca el puntero sobre el fondo, podrá:
 - Modificar la unidad de la CPU y
 - seleccionar diversos factores de zoom.
- Si coloca el puntero sobre un módulo, podrá:
 - Acceder a las funciones de edición (borrar, copiar, mover),
 - abrir la configuración del módulo para definir los parámetros específicos del módulo y
 - visualizar las propiedades de E/S y la corriente total.
- Si coloca el puntero sobre un slot vacío, podrá:

- Insertar un módulo del catálogo e
- insertar un módulo copiado previamente, incluyendo sus propiedades definidas.

Configuración del módulo

La ventana de configuración del módulo (se ejecuta desde el menú contextual del módulo o haciendo doble clic en el módulo) sirve para configurar el módulo. Entre sus funciones se encuentran, por ejemplo, la selección de canal, la selección de la función del canal seleccionado, la asignación de memoria de señal (sólo Quantum), etc.

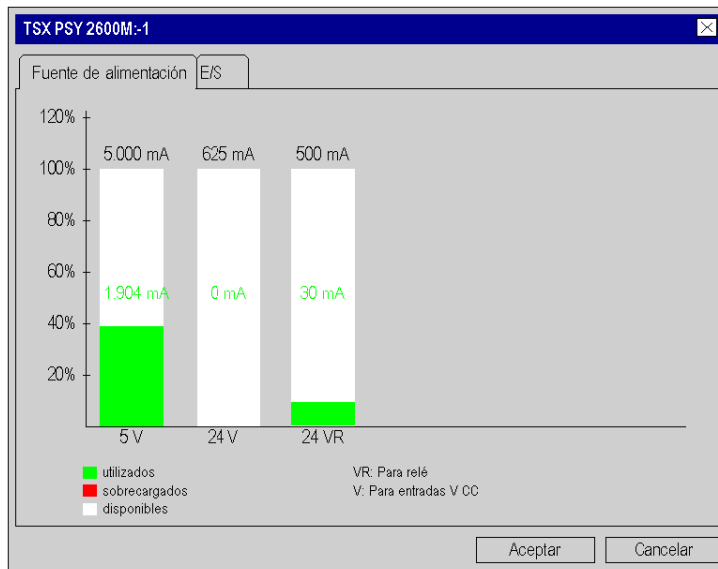
Ventana de configuración para un módulo de E/S Premium

	Simbolo	Área	Escala	Filtro
0		+/-10 V	%...	0
1		Termo K	1/10 °C	0
2		+/-10 V	%...	0
3		Pt100 IEC/DIN	1/10 °C	0

Propiedades del módulo

La ventana de propiedades del módulo (se ejecuta por medio del menú contextual del módulo) muestra las propiedades del módulo, p. ej., el consumo de alimentación, la cantidad de puntos de E/S (sólo Premium), etc.

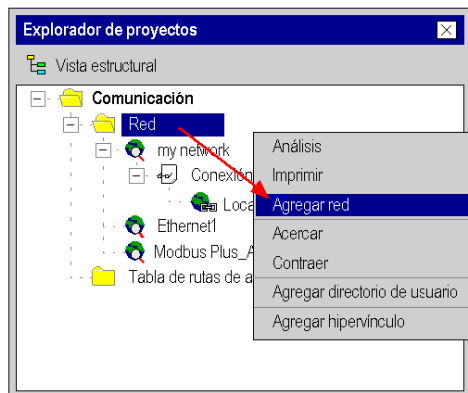
En la ficha de la fuente de alimentación de la ventana de propiedades del módulo, se muestra la corriente total del bastidor.



Configuración de red

La configuración de red se ejecuta a través de la carpeta de comunicaciones.

Configuración de red



Las ventanas de la configuración de red permiten, entre otras cosas:

- Crear redes
- Analizar una red
- Imprimir la configuración de red

Ventana para configurar una red.

Tras la configuración, la red se asigna a un módulo de comunicaciones.

Editor de datos

Introducción

El editor de datos ofrece las siguientes prestaciones:

- Declaración de instancias de variables
- Definición de tipos de datos derivados (DDT)
- Declaración de instancias de bloques de función elementales y derivados (EFB/DFB)
- Definición de parámetros de bloques de función derivados (DFB)

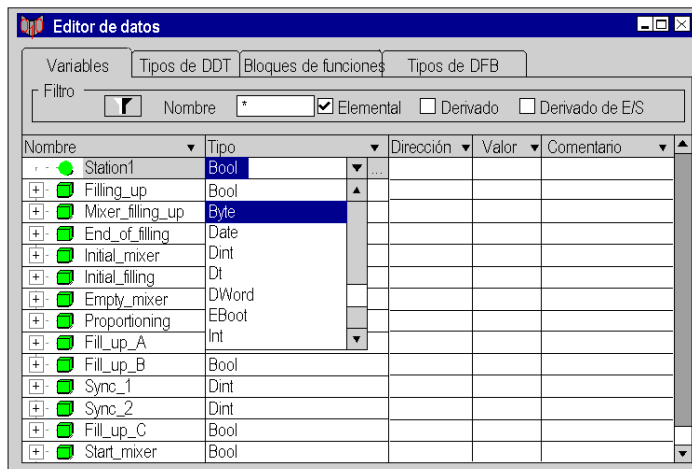
Las siguientes funciones están disponibles en todas las fichas del editor de datos:

- Copiar, cortar, pegar
- Expandir/minimizar datos estructurados
- Ordenar por tipo, símbolo, dirección, etc.
- Filtrado
- Insertar, eliminar y modificar la posición de las columnas
- Utilizar la función Arrastrar y soltar entre el editor de datos y los editores de programas
- Deshacer (undo) la última modificación
- Exportación/importación

Variables

La ficha **Variables** sirve para declarar variables.

Ficha Variables:



Están disponibles las siguientes funciones:

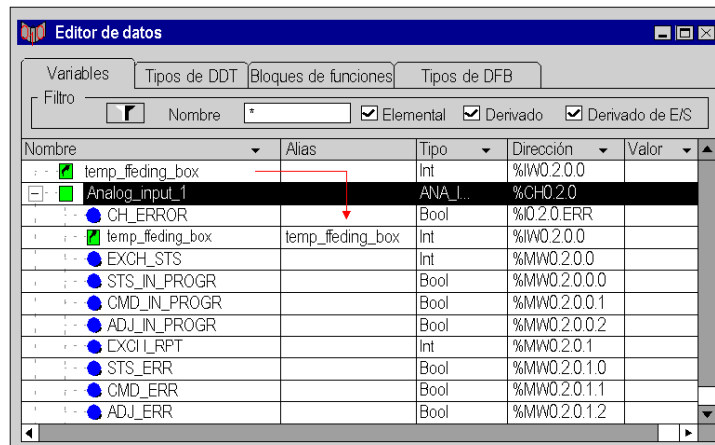
- Definición de un símbolo para las variables
- Asignación del tipo de datos

- Cuadro de selección propio para tipos de datos derivados
- Asignación de una dirección
- Simbolización automática de variables de E/S
- Asignación de un valor inicial
- Introducción de un comentario
- Visualización de todas las propiedades de una variable en un cuadro de propiedades aparte

Tipos de datos dependientes del hardware (IO DDT)

Los IO DDT sirven para asignar la estructura de E/S completa de un módulo a una única variable.

Asignación de IO DDT:



Están disponibles las siguientes funciones:

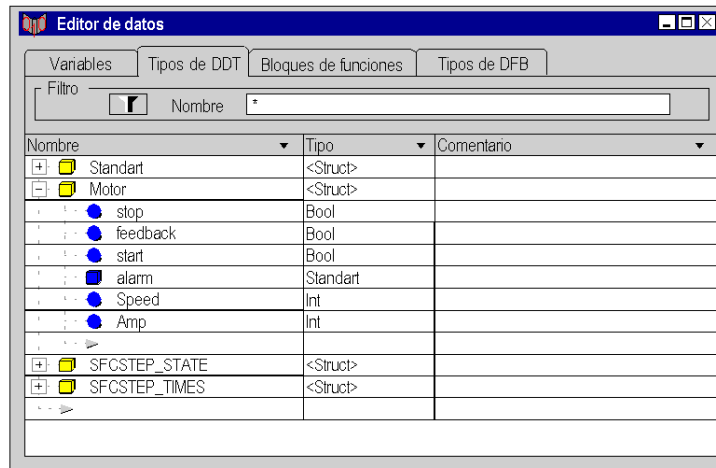
- Mediante los IO DDT es posible asignar estructuras de E/S completas a una única variable.
- Tras introducir la dirección de la variable, a todos los elementos de la estructura se les asigna automáticamente el bit de E/S o la palabra de E/S correctos.
- Gracias a la posibilidad de asignar la dirección a posteriori, es posible crear de forma rápida y sencilla módulos estándar cuyas direcciones no se conocerán hasta un momento posterior.
- Para todos los elementos de la estructura de E/S se puede predeterminar un alias.

Tipos de datos derivados (DDT)

La ficha **Tipos de DDT** sirve para definir tipos de datos derivados (DDT).

Un tipo de datos derivados es la definición de una estructura o de un campo de cualquier tipo de datos ya definido (elemental o derivado).

Ficha Tipos de DDT:



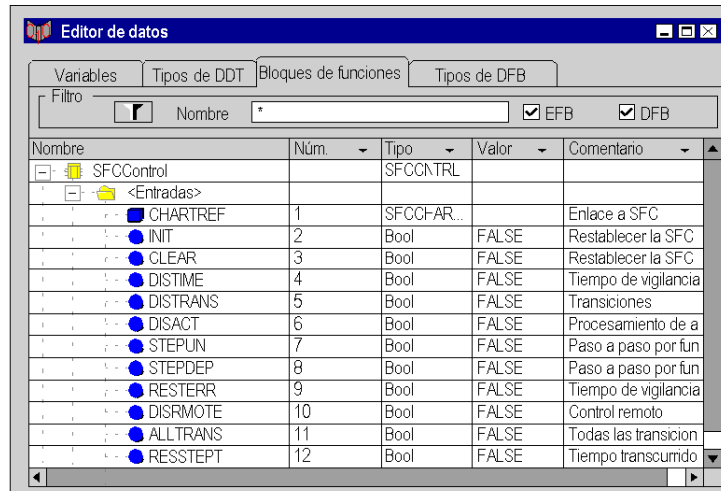
Están disponibles las siguientes funciones:

- Definición de DDT intercalados (8 niveles como máx.)
- Definición de campos (matrices) con hasta 6 dimensiones
- Asignación de un valor inicial
- Asignación de una dirección
- Introducción de un comentario
- Análisis del tipo de datos derivados
- Asignación del tipo de datos derivados a una biblioteca
- Visualización de todas las propiedades de un tipo de datos derivados en un cuadro de diálogo de propiedades aparte

Bloques de función

La ficha **Bloques de funciones** sirve para la declaración de instancias de bloques de funciones elementales y derivados (EFB/DFB).

Ficha **Bloques de funciones:**



Están disponibles las siguientes funciones:

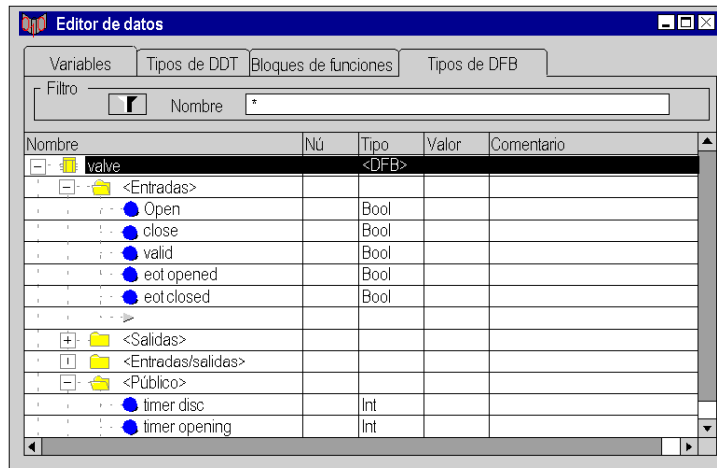
- Visualización de los bloques de función utilizados en el proyecto
- Definición de un símbolo para los bloques de función utilizados en el proyecto
- Aceptación automática de los símbolos definidos en el proyecto
- Introducción de un comentario para los bloques de función
- Visualización de todos los parámetros (entradas/salidas) de los bloques de función
- Asignación de un valor iniciar a las entradas/salidas del bloque de función

Tipos de DFB

La ficha **Tipos de DFB** sirve para definir los parámetros de los bloques de funciones derivados (DFB).

La lógica DFB se genera directamente en una o varias secciones de los lenguajes de programación FBD, LD, IL o ST.

Ficha Tipos de DFB:



Están disponibles las siguientes funciones:

- Definición del nombre del DFB
- Definición de todos los parámetros del DFB, como:
 - Entradas
 - Salidas
 - VAR_IN_OUT (entradas/salidas combinadas)
 - Variables privadas
 - Variables públicas
- Asignación del tipo de datos a los parámetros del DFB
- Cuadro de selección propio para tipos de datos derivados
- Asignación de un valor inicial
- Anidamiento de DFB
- Utilización de varias secciones en un DFB
- Introducción de un comentario para DFB y parámetros de DFB
- Análisis de los DFB definidos
- Gestión de versiones
- Asignación de los DFB definidos a una biblioteca

Utilización de los datos

Las instancias y los tipos de datos generados con el editor de datos se pueden insertar en los editores de los lenguajes de programación en función del contexto.

Están disponibles las siguientes funciones:

- Acceso a todos los editores de lenguajes de programación
- Sólo se muestran los datos compatibles

- La visualización de las funciones, los bloques de función, los procedimientos y los tipos de datos derivados se clasifican de acuerdo con la biblioteca de procedencia.
- La declaración de instancias se puede realizar durante la programación.

Cuadro de diálogo para la selección de datos:



Modificación online

Es posible modificar el tipo de una variable o una instancia de bloque de función (FB) declarada en una aplicación o en un bloque de función derivado (DFB) directamente en modalidad online (véase *Unity Pro, Modalidades de funcionamiento*,). Esto significa que no es necesario para detener la aplicación para realizar ese tipo de modificación.

Estas operaciones se pueden efectuar en el editor de datos o en el editor de propiedades, de la misma forma como se haría en la modalidad offline.

! ATENCIÓN

Riesgo de comportamiento inesperado de la aplicación

Al cambiar el tipo de una variable, el nuevo valor de la variable que se modificará depende de su tipo:

- En el caso de una **variable no ubicada**, la variable se establece en el valor inicial, si existe alguno. De lo contrario, se establece en el valor predeterminado.
- En el caso de una **variable ubicada**, la variable se reinicia con el valor inicial, si existe alguno. De lo contrario, no se modifica el valor binario actual.

Antes de aplicar el cambio del tipo de variable, compruebe el impacto del nuevo valor de la variable al ejecutar la aplicación.

Si no se siguen estas instrucciones pueden producirse lesiones personales o daños en el equipo.

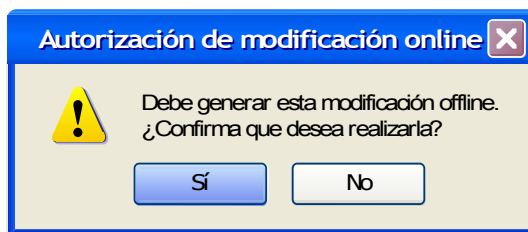
NOTA: No es posible modificar el tipo de una variable declarada en tipos de datos derivados (DDT) en modalidad online (véase *Unity Pro, Modalidades de funcionamiento*,). La aplicación se debe cambiar a modalidad offline (véase *Unity Pro, Modalidades de funcionamiento*,) para generar esa modificación.

Restricciones sobre modificaciones online

En los casos siguientes, no se permite la modificación de tipo online de una variable o de un Bloque de función (FB):

- Si la variable se usa como datos globales de red, no se permite la modificación del tipo online.
- Si el FB actual no se puede eliminar en línea o si no se puede agregar un FB nuevo en línea, no se permite la modificación de tipo online de este FB. De hecho, algunos Bloques de funciones elementales (EFB) como los Bloques de funciones estándar (SFB) no pueden agregarse ni eliminarse online. Como consecuencia, no es posible cambiar el tipo de una instancia EFB a una instancia SFB ni viceversa.

En ambos casos, se muestra el cuadro de diálogo siguiente:



NOTA: Debido a estas limitaciones, si un bloque de función derivado (DFB) contiene al menos una instancia de SFB, no es posible agregar ni eliminar una instancia de este DFB en modalidad online (*véase Unity Pro, Modalidades de funcionamiento,*).

Editor de programas

Introducción

Un programa puede estar compuesto por:

- **Tareas**, que se ejecutan de forma cíclica o periódica.

Las tareas están formadas por:

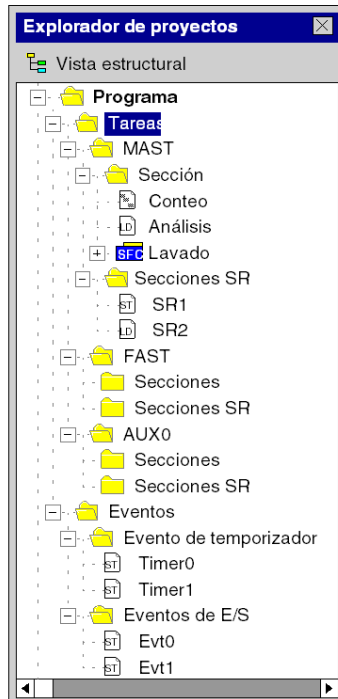
- Secciones
- Subrutinas

- **Procesamientos de eventos**, que tienen prioridad sobre todas las demás tareas.

Los procesamientos de eventos están formadas por:

- Secciones para el procesamiento de eventos controlados por tiempo
- Secciones para el procesamiento de eventos controlados por hardware

Ejemplo de un programa:



Tareas

Unity Pro admite varias tareas al mismo tiempo (multitasking).

Las tareas se ejecutan de forma independiente y paralela, el PLC se encargará de controlar las prioridades de ejecución. Las tareas se adaptan a las diversas necesidades, de forma que constituyen un potente instrumento para estructurar el proyecto.

Un proyecto multitarea puede estar compuesto por:

- Una tarea maestro (MAST)
La tarea maestra se ejecuta de forma cíclica o periódica.
Es la parte principal del programa y se ejecuta de forma secuencial.
- Una tarea rápida (FAST)
La tarea rápida se ejecuta de forma periódica. Tiene un nivel de prioridad superior al de la tarea maestra. La tarea rápida está ideada para procesamientos periódicos que duran un breve espacio de tiempo.
- De una a cuatro tareas AUX
Las tareas AUX se ejecutan de forma periódica. Están ideadas para procesamientos más lentos y tienen el nivel de prioridad más bajo.

El proyecto también se puede estructurar como una única tarea. En tal caso, sólo estará activa la tarea maestra.

Procesamiento de eventos

El procesamiento de eventos se realiza en las denominadas "secciones de eventos". Estas secciones de eventos tienen prioridad sobre las secciones de todas las demás tareas. Por lo tanto, son ideales para procesamientos que requieren tiempos de respuesta muy cortos ante la aparición de eventos.

Están disponibles los siguientes tipos de sección para el procesamiento de eventos:

- Sección para el procesamiento de eventos controlados por tiempo (sección Timerx)
- Sección para el procesamiento de eventos controlados por hardware (sección Evtx)

Son compatibles los siguientes lenguajes de programación:

- FBD (lenguaje de bloques de función)
- LD (lenguaje de esquema de contactos)
- IL (lista de instrucciones)
- ST (texto estructurado)

Secciones

Las secciones son unidades de programa autónomas en las que se crea la lógica del proyecto.

Las secciones se ejecutan en el mismo orden en el que se representan en el explorador de proyectos (vista estructural). Las secciones están ligadas a una tarea.

Una misma sección no puede pertenecer simultáneamente a varias tareas.

Son compatibles los siguientes lenguajes de programación:

- FBD (lenguaje de bloques de función)
- LD (lenguaje de esquema de contactos)
- SFC (lenguaje de ejecución secuencial)
- IL (lista de instrucciones)
- ST (texto estructurado)

Subrutinas

Las subrutinas se crean como unidades independientes en secciones de subrutina.

La llamada de subrutinas se realiza desde las secciones o desde otra subrutina.

Es posible configurar un anidamiento de hasta 8 niveles.

Una subrutina no se puede llamar a sí misma (no es recursiva).

Las subrutinas están asignadas a una tarea. No es posible llamar la misma subrutina desde varias tareas.

Son compatibles los siguientes lenguajes de programación:

- FBD (lenguaje de bloques de función)
- LD (lenguaje de esquema de contactos)
- IL (lista de instrucciones)
- ST (texto estructurado)

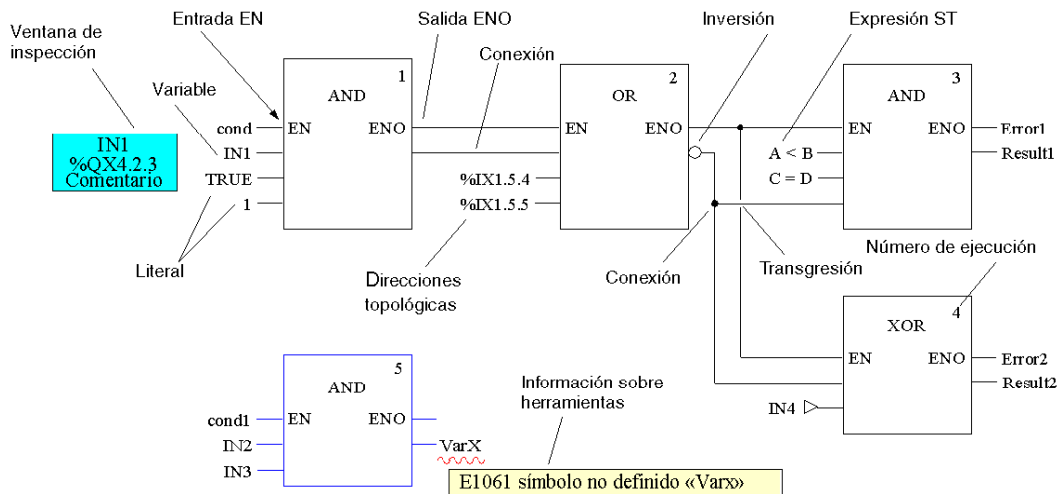
Diagrama de bloques de funciones (FBD)

Introducción

El Editor FBD permite la programación gráfica de bloques de funciones según CEI 61131-3.

Representación

Representación de una sección FBD:



Objetos

Los objetos del lenguaje de programación FBD (diagrama de bloques de funciones) sirven de ayuda para dividir una sección en una cantidad de

- funciones elementales (EF),
- bloques de funciones elementales (EFB),
- bloques de funciones derivadas (DFB),
- procedimientos,
- llamadas de subrutina,
- saltos,
- vínculos,
- parámetros reales,
- objetos de texto para comentarios de la lógica.

Propiedades

Las secciones FBD disponen de un reticulado detrás de ellas. Una unidad de reticulado está compuesta por 10 coordenadas. Una unidad de reticulado es la distancia mínima posible entre dos objetos de una sección FBD.

El lenguaje de programación FBD no está basado en celdas; los objetos están alineados con las coordenadas.

Una sección FBD puede configurarse con un número de celdas (coordenadas del reticulado horizontal y coordenadas del reticulado vertical).

El programa puede introducirse con el ratón o el teclado.

Ayudas de entrada

El editor FBD ofrece las ayudas de entrada siguientes:

- Barras de herramientas para un acceso rápido y fácil a los objetos deseados
- Comprobación sintáctica y semántica a medida que se escribe el programa
 - Visualización en color azul de los bloques de funciones y funciones incorrectos
- Marcación con una línea roja ondulada de las palabras desconocidas (por ejemplo, variables no declaradas) o de los tipos de datos inadecuados
- Breve descripción de errores en la información rápida (información sobre herramientas).
- La información sobre variables y pines se puede mostrar en la información rápida (información sobre herramientas)
 - Tipo, nombre, dirección y comentario de una variable o expresión
 - Tipo, nombre y comentario de un pin FFB
- Visualización tabular de FFB
- Los parámetros reales pueden introducirse y visualizarse como símbolos o direcciones topológicas
- Distintos factores de zoom
- Seguimiento de conexiones
- Optimización de rutas de conexión
- Visualización de ventanas de inspección

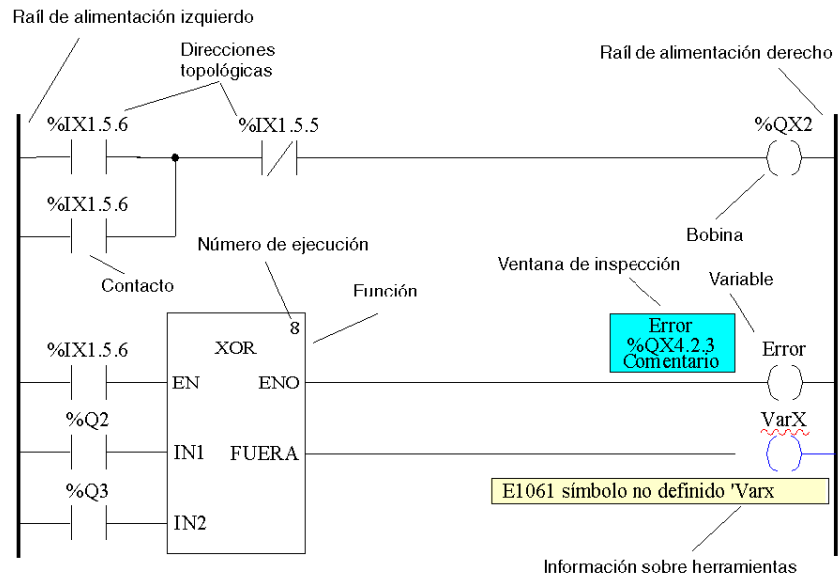
Lenguaje del diagrama de Ladder (LD)

Introducción

El editor LD permite la programación gráfica de diagramas de Ladder según la norma CEI 61131-3.

Representación

Representación de una sección LD:



Objetos

Los objetos del lenguaje de programación LD ayudan a dividir una sección en un número de:

- Contactos
- Bobinas
- Funciones elementales (EF)
- Bloques de funciones elementales (EFB)
- Bloques de funciones derivadas (DFB)
- Procedimientos
- Elementos de control
- Bloques de funcionamiento y comparación que representan una ampliación de la norma CEI 61131-3
- Llamadas de subrutina
- Saltos

- Vínculos
- Parámetros reales
- Objetos de texto para comentarios de la lógica

Propiedades

Las secciones LD tienen una cuadrícula de fondo que divide la sección en líneas y columnas.

LD es un lenguaje de programación orientado a celdas, es decir, sólo se puede colocar un objeto en cada celda.

Las secciones LD pueden tener un tamaño de 11 a 64 columnas y de 17 a 2000 líneas.

El programa puede introducirse con el ratón o el teclado.

Ayudas de entrada

El editor LD ofrece las ayudas de entrada siguientes:

- Los objetos se pueden seleccionar en la barra de herramientas, el menú o directamente con las teclas de método abreviado.
- Comprobación sintáctica y semántica a medida que se escribe el programa.
 - Visualización de los objetos incorrectos en color azul
 - Marcación con una línea roja ondulada de las palabras desconocidas (por ejemplo, variables no declaradas) o de los tipos de datos inadecuados.
 - Breve descripción de errores en la información rápida (información sobre herramientas).
- La información sobre las variables y elementos de una sección LD, que pueden conectarse a una variable (pines, contactos, bobinas, bloques de funcionamiento y comparación), se puede mostrar en la información rápida (información sobre herramientas)
 - tipo, nombre, dirección y comentario de una variable o expresión
 - tipo, nombre y comentario de pines FFB, contactos, etc.
- Visualización tabular de FFB
- Los parámetros reales pueden introducirse y visualizarse como símbolos o direcciones topológicas
- Distintos factores de zoom
- Seguimiento de conexiones FFB
- Optimización de rutas de conexiones FFB
- Visualización de ventanas de inspección

Información general sobre el lenguaje de secuencias SFC

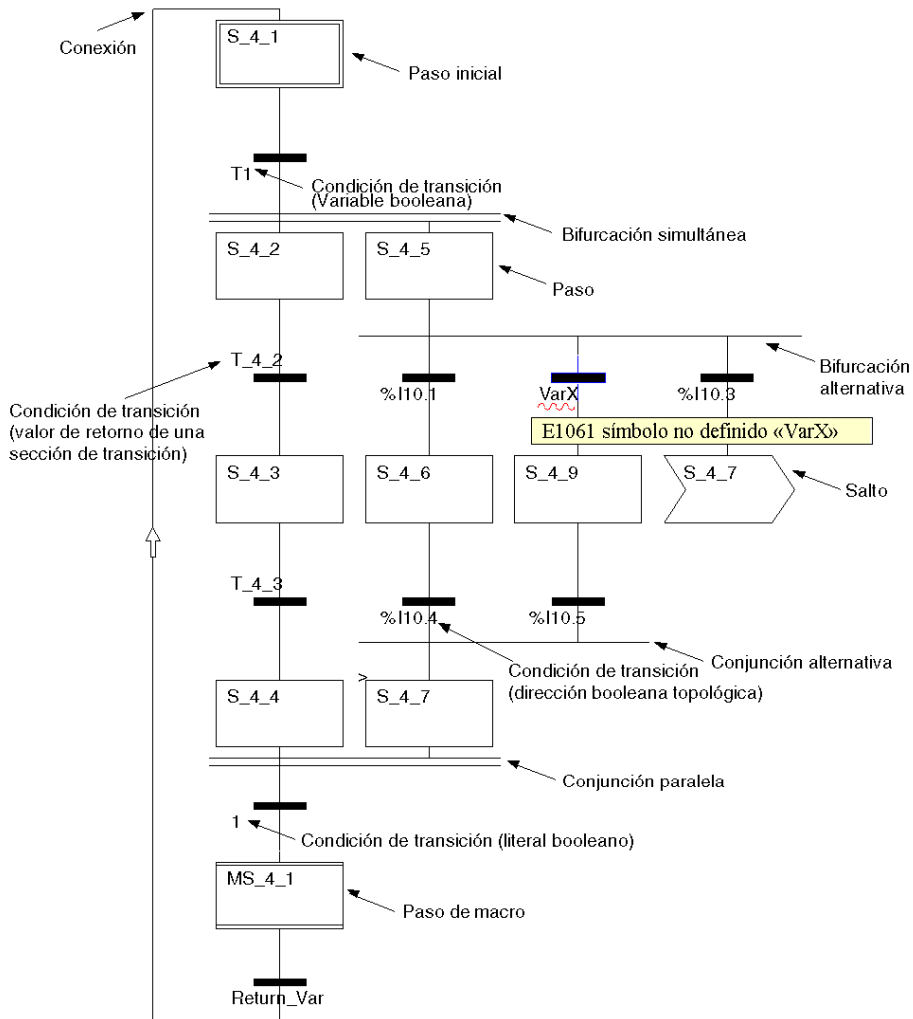
Introducción

El lenguaje de secuencias SFC (Gráfica de función secuencial), que cumple con la norma IEC 61131-3, se describe en esta sección.

Las restricciones de conformidad con la IEC pueden eliminarse mediante procedimientos de activación explícitos. Así, pueden realizarse funciones como token múltiple, varios pasos iniciales, saltos a las cadenas paralelas o desde éstas, etc.

Representación

Representación de una sección SFC:



Objetos

Una sección SFC proporciona los objetos siguientes para crear un programa:

- Pasos
- Pasos de macro (secuencias de subpasos integrada)
- Transiciones (condiciones de transición)

- Secciones de transición
- Secciones de acción
- Saltos
- Vínculos
- Secuencias alternativas
- Secuencias paralelas
- Objetos de texto para comentarios de la lógica

Propiedades

El editor SFC tiene un reticulado de fondo que divide la sección en 200 filas y 32 columnas.

El programa puede introducirse con el ratón o el teclado.

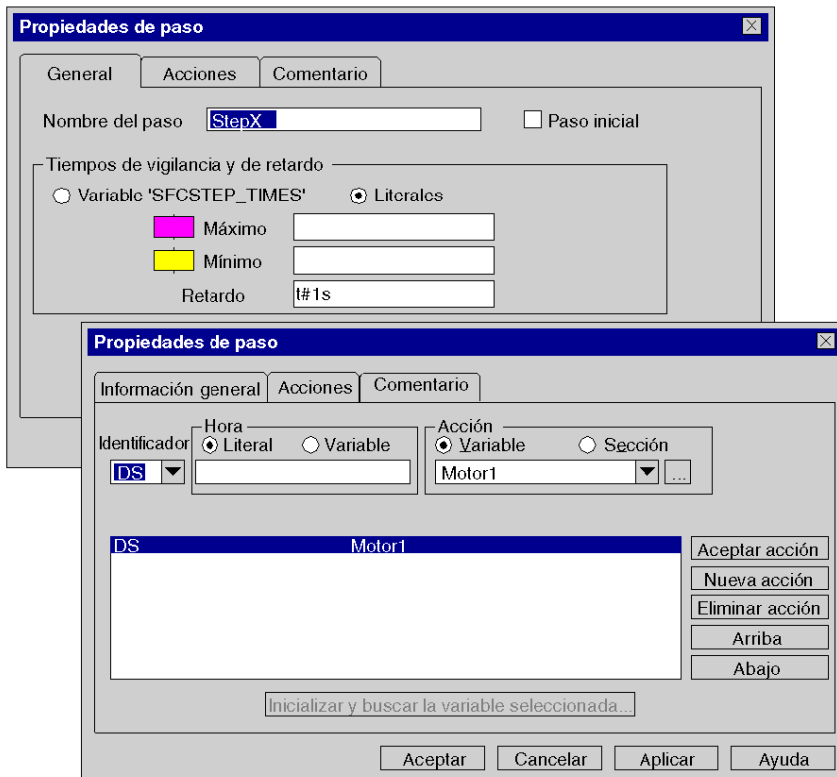
Ayudas de entrada

El editor SFC ofrece las ayudas de entrada siguientes:

- Barras de herramientas para un acceso rápido y fácil a los objetos deseados
- Numeración automática de pasos
- Acceso directo a acciones y condiciones de transición
- Comprobación sintáctica y semántica a medida que se escribe el programa
 - Visualización de los objetos incorrectos en color azul
 - Marcación con una línea roja ondulada de las palabras desconocidas (por ejemplo, variables no declaradas) o de los tipos de datos inadecuados.
 - Breve descripción de errores en la información rápida (información sobre herramientas).
- La información sobre variables y transiciones se puede mostrar en la información rápida (información sobre herramientas)
 - Tipo, nombre, dirección y comentario de una variable o expresión
 - Tipo, nombre y comentario de transiciones
- Distintos factores de zoom
- Mostrar/ocultar las acciones asignadas
- Seguimiento de conexiones
- Optimización de rutas de conexión

Propiedades de paso

Propiedades de paso:



Las propiedades de paso se definen mediante un cuadro de diálogo que ofrece las funciones siguientes:

- Definición de pasos iniciales
- Definición de tiempos de diagnóstico
- Comentarios de pasos
- Asignación de acciones y sus identificadores

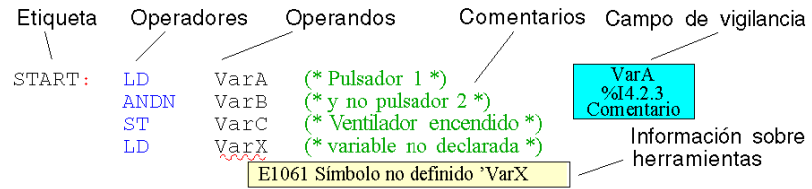
Lista de instrucciones IL

Introducción

El Editor IL permite la programación de listas de instrucciones según CEI 61131-3.

Representación

Representación de una sección IL



Objetos

Una lista de instrucciones está compuesta por una secuencia de instrucciones.

Cada instrucción comienza en una fila nueva y está compuesta por:

- Un operador
- En ocasiones, un modificador
- Si es necesario, uno o más operandos
- Eventualmente, una marca como destino de salto
- En ocasiones, un comentario para comentar la lógica

Asistentes de entrada

El editor IL ofrece, entre otros, los siguientes asistentes de entrada:

- Al crear el programa se realiza una comprobación sintáctica y semántica.
 - Las palabras clave y los comentarios se representan con colores.
 - Las palabras desconocidas (p. ej., las variables no declaradas) o los tipos de datos inadecuados se marcan con una línea ondulada roja.
 - El error se describe brevemente en la información sobre herramientas (Tooltip).
- Visualización de funciones y módulos de función en forma de tabla
- Asistente de entrada para funciones y módulos de función
- Los operandos se pueden indicar y visualizar como símbolos o como direcciones topológicas.
- Visualización de los campos de vigilancia.

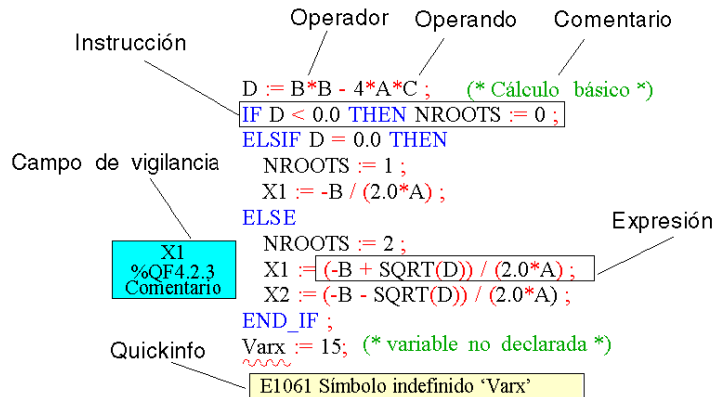
Texto estructurado ST

Introducción

El Editor ST permite la programación en texto estructurado según CEI 61131-3.

Representación

Representación de una sección ST



Objetos

El lenguaje de programación ST trabaja con las denominadas "expresiones".

Las expresiones son construcciones compuestas por operadores y operandos que devuelven un valor durante la ejecución.

Los operadores son símbolos para las operaciones que se van a ejecutar.

Los operadores se aplican a los operandos. Los operandos son, por ejemplo, variables, literales, salidas/entradas de funciones y de módulos de función, etc.

Las instrucciones sirven para estructurar y controlar las expresiones.

Asistentes de entrada

El editor ST ofrece, entre otros, los siguientes asistentes de entrada:

- Al crear el programa se realiza una comprobación sintáctica y semántica.
 - Las palabras clave y los comentarios se representan con colores.
 - Las palabras desconocidas (p. ej., las variables no declaradas) o los tipos de datos inadecuados se marcan con una línea ondulada roja.
- El error se describe brevemente en la información sobre herramientas (Tooltip).
- Visualización de funciones y módulos de función en forma de tabla

- Asistente de entrada para funciones y módulos de función
- Los operandos se pueden indicar y visualizar como símbolos o como direcciones topológicas.
- Visualización de los campos de vigilancia.

Simulador del PLC

Presentación

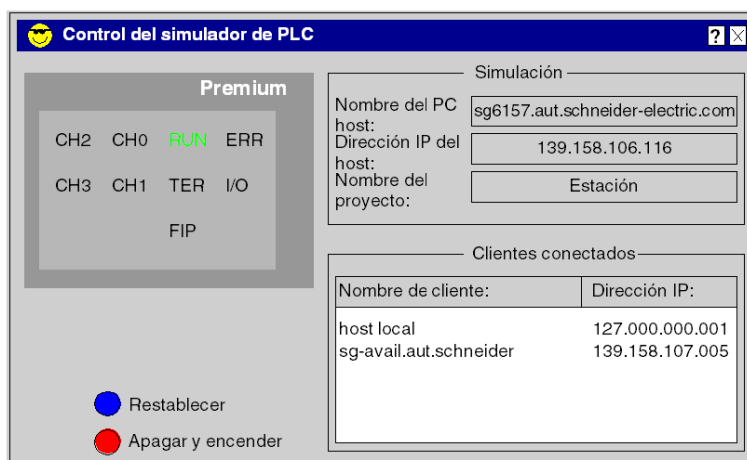
El simulador del PLC permite realizar la búsqueda de errores en el proyecto sin tener que establecer una conexión con un PLC real.

Todas las tareas del proyecto (Mast, Fast, AUX y eventos) que se ejecutan en un PLC real también están disponibles en el simulador. La diferencia con un PLC real consiste en la ausencia de módulos de E/S y redes de comunicaciones (como, p. ej., ETHWAY, Fipio y Modbus Plus), así como en un comportamiento en tiempo real no determinístico.

Naturalmente, en el simulador del PLC están disponibles todas las funciones de depuración y animación, de puntos de parada, de forzado de variables, etc.

Representación

Representación de un cuadro de diálogo:



Estructura del simulador

El panel del simulador ofrece la siguiente información:

- tipo del PLC simulado;
- estado actual del PLC simulado;
- nombre del proyecto cargado;
- dirección IP y nombre DNS del PC host del simulador y todos los PC clientes conectados;
- cuadro de diálogo para simular eventos de E/S;
- botón **Restablecer** para restablecer el PLC simulado (inicio en frío simulado);

- botón **Conexión/desconexión de tensión** para la simulación de un reinicio en caliente;
- menú contextual (botón derecho del ratón) para controlar el simulador.

Exportación/importación

Introducción

Las funciones de exportación e importación permiten utilizar en otros proyectos los datos ya creados. Además, con el formato de exportación/importación XML resulta muy sencillo transferir datos a un software externo u obtener sus datos.

Exportación

Se pueden exportar los siguientes objetos:

- Proyectos completos, incluida la configuración
- Secciones de todos los lenguajes de programación
- Secciones de subrutinas de todos los lenguajes de programación
- Bloques de función derivados (DFB)
- Tipos de datos derivados (DDT)
- Declaraciones de variables
- Ventana de usuario

Importación

Lógicamente, todos los objetos que se pueden exportar también se pueden importar.

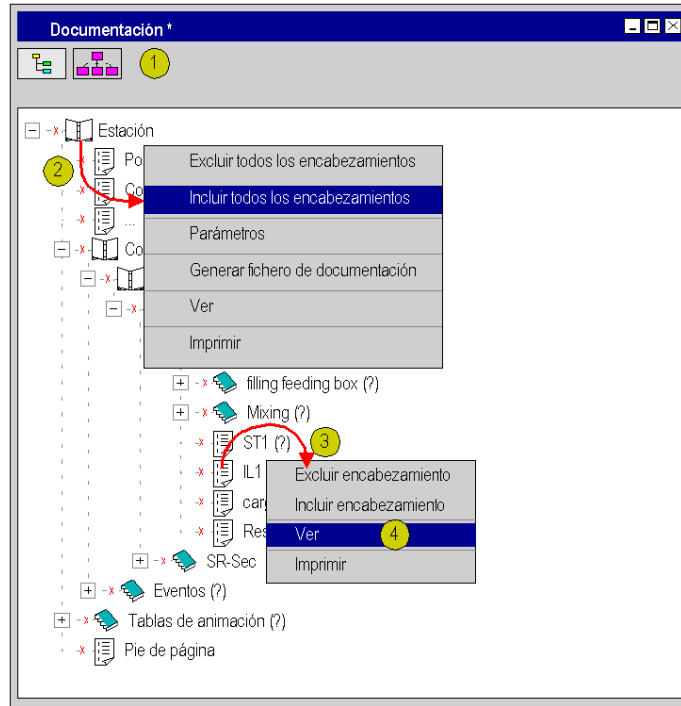
Existen dos variantes de importación:

- **Importación directa**
Importa el objeto tal como se exportó.
- **Importación con asistente**
Los asistentes permiten modificar los nombres de las variables, las secciones o las unidades funcionales. Además, se puede modificar la asignación de las direcciones.

Documentación de usuario

Documentación de usuario

Prestaciones de la documentación de usuario



Las siguientes funciones sirven para documentar el proyecto:

- Impresión de todo el proyecto (2) o ciertas partes (3)
- Elección entre las vistas estructural o funcional (1)
- Adaptación de los resultados (pies de página, información general, etc.)
- Impresión local para los editores de lenguajes de programación, el editor de configuración etc.
- Tipo de letra especial (negrita) para las palabras clave
- Formato de papel a elegir
- Vista preliminar (4)
- Almacenamiento de la documentación

Servicios de depuración

Localización de errores en la aplicación de usuario

Para optimizar la localización de errores en el proyecto, se ofrecen las siguientes prestaciones:

- establecimiento de puntos de parada (break points) en los editores de lenguajes de programación
- ejecución por pasos (step-by-step) del programa, paso a paso por instrucciones (step into), paso a paso para salir (step out) y paso a paso por función (step over)
- memoria de llamadas para obtener la ruta completa del programa
- control de entradas y salidas

Modalidad online

Si el PC está unido al PLC y se ha establecido la conexión, se considera que están en modalidad online.

La modalidad online se utiliza para localizar errores (depuración), para la animación y para modificar el programa en el PLC.

Si se debe establecer la conexión, tendrá lugar una comparación entre el proyecto del PC y el del PLC automáticamente.

Esta comparación puede tener los siguientes resultados:

- **Distintos proyectos en el PC y el PLC**

En este caso, la modalidad online sólo estará disponible de forma limitada. Sólo es posible utilizar comandos del PLC (p. ej., inicio, parada), servicios de diagnóstico y supervisión de variables. No es posible modificar la lógica del programa ni la configuración en el PLC. Sin embargo, sí son posibles las funciones de descarga y carga, que se ejecutan en una modalidad ilimitada (mismo proyecto en el PC y en el PLC).

- **Mismos proyectos en el PC y el PLC**

Existen dos posibilidades:

- **ONLINE SAME, BUILT**

La última generación del proyecto en el PC se ha cargado en el PLC y después no se han efectuado modificaciones, es decir, el proyecto es absolutamente idéntico en el PC y en el PLC.

En este caso, todas las funciones de animación estarán disponibles de forma ilimitada.

- **ONLINE EQUAL, NOT BUILT**

La última generación del proyecto en el PC se ha cargado en el PLC; sin embargo, se han efectuado modificaciones a posteriori.

En este caso, las funciones de animación sólo estarán disponibles en las partes no modificadas del proyecto.

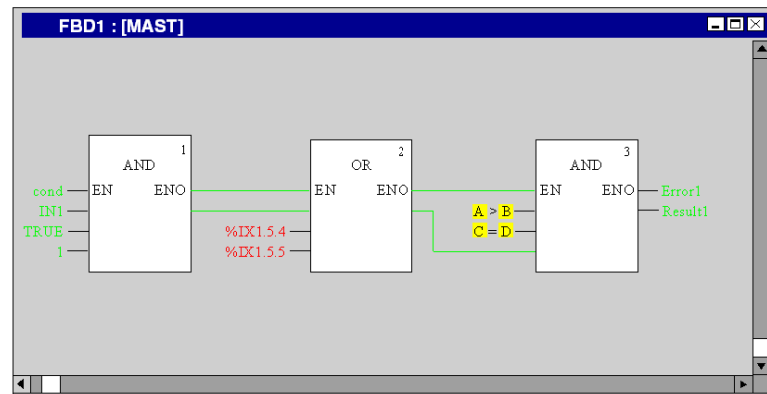
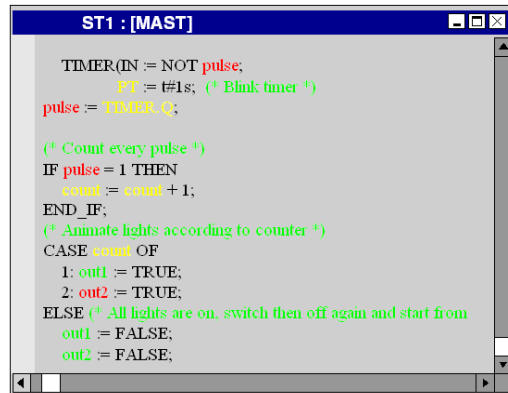
Animación

Existen distintas posibilidades para la animación de variables:

- **Animación de secciones**

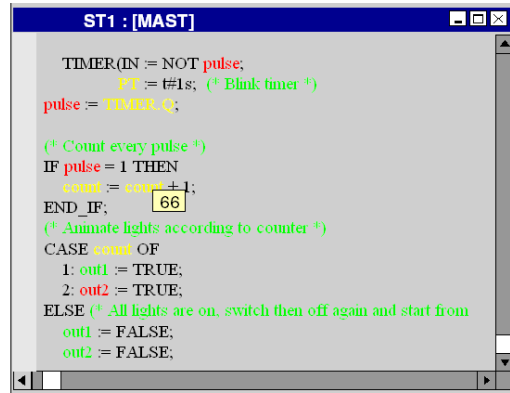
Se pueden animar todos los lenguajes de programación (FBD, LD, SFC, IL y ST).

La animación de las variables y de las conexiones tiene lugar directamente en la sección.



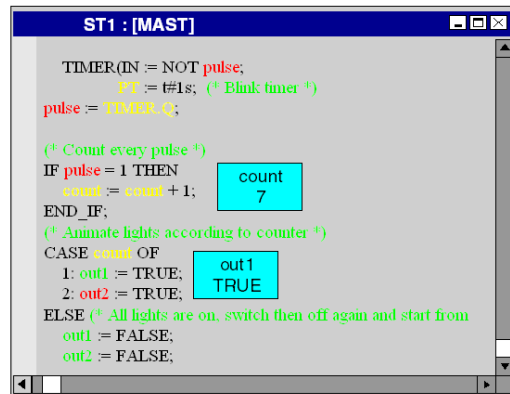
- **Información sobre herramientas**

Al desplazar el puntero del ratón sobre una variable, aparece una breve información con el valor de la variable.



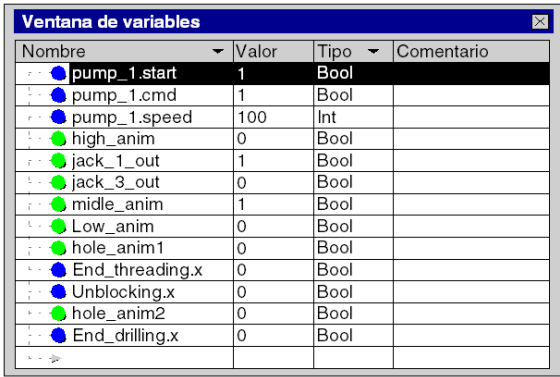
- **Ventana de inspección**

Por cada variable se puede crear una ventana de inspección. Esta ventana muestra el valor de las variables, sus direcciones y comentarios (si existen). Esta función está disponible en todos los lenguajes de programación.



- **Ventana de variables**

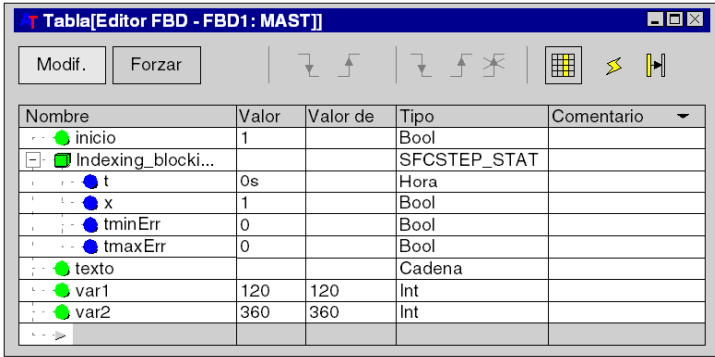
Esta ventana muestra todas las variables utilizadas en la sección actual.



Nombre	Valor	Tipo	Comentario
pump_1.start	1	Bool	
pump_1.cmd	1	Bool	
pump_1.speed	100	Int	
high_anim	0	Bool	
jack_1_out	1	Bool	
jack_3_out	0	Bool	
midle_anim	1	Bool	
Low_anim	0	Bool	
hole_anim1	0	Bool	
End_threading.x	0	Bool	
Unblocking.x	0	Bool	
hole_anim2	0	Bool	
End_drilling.x	0	Bool	

● **Tabla de animación**

En las tablas de animación se pueden mostrar, modificar o forzar los valores de todas las variables del proyecto. Los valores se pueden modificar de uno e uno o varios a la vez.



Nombre	Valor	Valor de	Tipo	Comentario
inicio	1		Bool	
Indexing_blocki...			SFCSTEP_STAT	
t	0s		Hora	
x	1		Bool	
tminErr	0		Bool	
tmaxErr	0		Bool	
texto			Cadena	
var1	120	120	Int	
var2	360	360	Int	

Punto de observación

Los puntos de observación permiten visualizar los datos del PLC en el punto exacto en que se crearon (1) y no al final del ciclo.

Las tablas de animación pueden sincronizarse con el punto de observación (2).

Un contador (3) indica la frecuencia con que se ha actualizado el punto de observación.

Sección ST con punto de observación:

Punto de observación

341

ST (Sección): My_ST [MAST]

```

if pump_1.start
then pump_1.cmd := true;
else pump_1.cmd := false; pump_1.speed := 0;
end_if;
if pump_1.cmd then pump_1.speed := pump_1.speed + 1; end_if;
if pump_1.speed > 100 then pump_1.speed := 100; end_if;
(* analizando y procesando con animación *)
high_alarm := not jack_1_out and not jack_3_out;

```

Tabla[Editor FBD - My_ST: MAST]

Nombre	Valor	Tipo	Comentario
inicio		Bool	
Indexing_blocki...		SFCST...	
t		Hora	
x		Bool	
tminErr		Bool	

Punto de parada

Los puntos de parada permiten detener la ejecución del proyecto en un punto cualquiera.

Sección ST con punto de parada:

ST1: [MAST]

```

TIMER(IN := NOT pulso;
PT := t#1s; (* Temporizador parpadeando *)
pulso := TIMER.Q;

(* Contar cada pulso *)
IF pulso = 1 THEN
  contar := contar + 1;
END_IF;

(* La animación se ilumina según el contador *)
CASE contar OF
  1: out1 := TRUE;
  2: out2 := TRUE;
ELSE (* Todas las luces están encendidas, apáguelas de nuevo
y comienza a partir de
  out1 := FALSE;

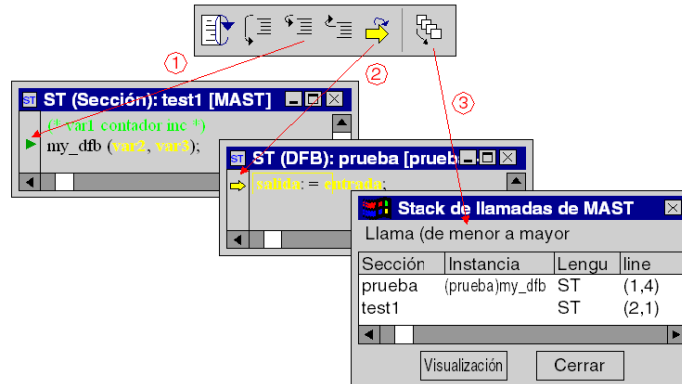
```

Punto de parada

Modalidad paso a paso

La modalidad paso a paso permite ejecutar el programa paso a paso. Las funciones paso a paso están disponibles cuando el proyecto se ha detenido al alcanzar el punto de parada o ya se encuentra en modalidad paso a paso.

Sección ST en la modalidad paso a paso:



La modalidad paso a paso dispone de las siguientes funciones:

- Ejecución paso a paso (step-by-step) del programa
- Paso a paso por instrucciones (step into) (1)
- Paso a paso para salir de salida (step out)
- Paso a paso por función (step over)
- Visualización del paso ejecutado en ese momento (2)
- Memoria de llamadas (3)

La ejecución múltiple de la función "paso a paso por instrucciones" (step into) permite que la memoria de llamadas muestre la ruta completa a partir del primer punto de parada (break point).

NOTA: La ejecución del programa PLC en la modalidad paso a paso y el acceso (StepIn) a una sección protegida contra lectura/escritura puede impedir que el programa se pueda leer y ocasiona la salida de la sección. El usuario debe cambiar la modalidad del PLC a "Parada" para volver al estado inicial.

Marcadores

Los marcadores permiten marcar fragmentos de código para poder localizarlos más fácilmente.

Visualizador de diagnósticos

Descripción

Unity Pro dispone de un diagnóstico de sistema y de proyectos.

En caso de que se produzcan errores, éstos aparecen en la ventana de diagnóstico. Para solucionar el error, desde la ventana de diagnóstico es posible abrir la sección que provocó el error.

Confirmación: 0 ▲	Mensaje ▲	Error ▲	Símbolo ▲	Rango ▲
Confirmado	Error en batería de búfer	Sistema...	%S68	0
Borrado	Error en batería de búfer	Sistema...	%S68	0
Borrado	Error en batería de búfer	Sistema...	%S68	0
Borrado	Desborde del índice	Sistema...	%S20 (MAST)	0

Alarma de sistema	Error en batería de búfer	28/01/2002 21:10:51

Bastidor del dispositivo fallido:	0
Slot del dispositivo fallido:	0

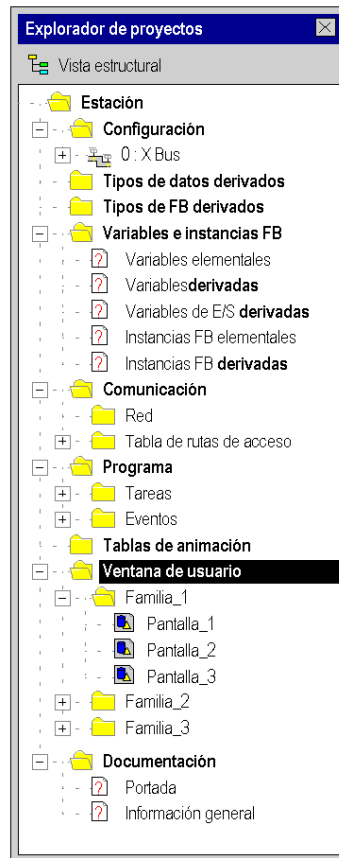
Ventana de usuario

Introducción

Las ventanas de usuario sirven para visualizar el proceso de automatización.

El editor de ventanas de usuario permite crear, modificar y gestionar ventanas de usuario de forma sencilla.

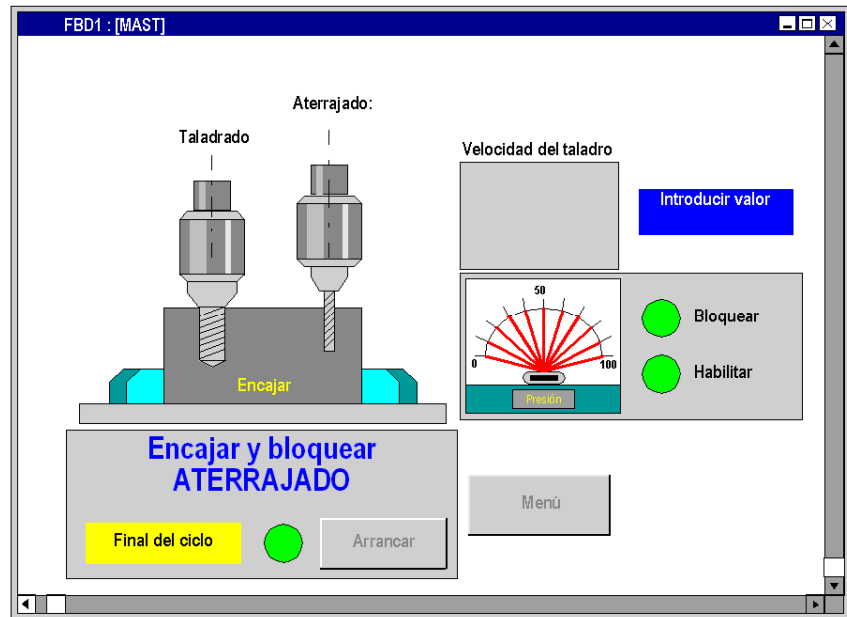
La creación y el acceso a las ventanas de usuario tiene lugar a través del explorador de proyectos.



Editor de ventanas de usuario

Las ventanas de usuario están compuestas por gran cantidad de información (variables dinámicas, vistas generales, textos descriptivos, etc.) y permiten vigilar y modificar rápida y fácilmente las variables de automatización.

Ventana de usuario



El editor de ventanas de usuario ofrece las siguientes prestaciones:

- Amplias funciones de visualización
 - Elementos geométricos
Línea, rectángulo, elipse, curva, polígono, mapa de bits, texto
 - Elementos de control
Botón, casilla de verificación, barra de movimiento, navegación por la pantalla, hipervínculo, campo de entrada, control numérico
 - Elementos de animación
Diagrama de barras, diagrama de tendencias, cuadro de diálogo, aparecer, desaparecer, colores intermitentes, animación de variables
- Creación de una biblioteca para la gestión de objetos gráficos
- Copiado de objetos
- Creación de una lista con todas las variables utilizadas en la ventana de usuario
- Creación de mensajes que se vayan a utilizar en las ventanas de usuario
- Acceso directo desde las ventanas de usuario a la tabla de animación o a la tabla de referencias cruzadas para una o varias variables

- Información sobre herramientas (Quickinfo) que ofrecen información sobre las variables
- Gestión de ventanas de usuario en familias
- Importación/exportación de ventanas de usuario individuales o familias enteras

Estructura de la aplicación



Objeto

En esta sección, se describen las estructuras del programa de la aplicación y la memoria asociados a cada tipo de PLC.

Contenido de esta parte

Esta parte contiene los siguientes capítulos:

Capítulo	Nombre del capítulo	Página
2	Descripción de las funciones disponibles en cada tipo de autómata	73
3	aplicación, estructura del programa	75
4	Estructura de la memoria de la aplicación	115
5	Modalidades de funcionamiento	131
6	Objetos de sistema	161

Descripción de las funciones disponibles en cada tipo de autómata



Funciones disponibles en los diferentes tipos de PLC

Lenguajes de programación

Los lenguajes siguientes están disponibles para las plataformas Modicon M340, Premium, Atrium y Quantum:

- LD
- FBD
- ST
- IL
- SFC

NOTA: Solo están disponibles los lenguajes LD y FBD en los PLC de seguridad Quantum.

Tareas y procesos

En la siguiente tabla se describen las tareas y procesos disponibles.

Plataformas	Modicon M340		Premium: TSX			Atrium: TSX	Quantum: 140 CPU		
Procesadores	P34 1000	P34 20••	P57 0244 P57 1••	P57 2•• P57 3•• P57 4•• H57 24M H57 44M	P57 5•• P57 6634	PCI 57 204/354	31•••• 43•••• 53••••	651•• 652 60 671 60	651 60S 671 60S
Tarea maestra Cíclica o periódica	X	X	X	X	X	X	X	X	X
Tarea rápida Periódica	X	X	X	X	X	X	X	X	-
Tareas auxiliares Periódica	-	-	-	-	4	-	-	4	-
Tamaño máximo de una sección			64 Kb					16 Mb	-

Plataformas	Modicon M340		Premium: TSX			Atrium: TSX	Quantum: 140 CPU		
Tratamiento de sucesos de tipo E/S	32	64	32	64	128	64	64	128	-
Tratamiento de sucesos de tipo Timer	16	32	-	-	32	-	16	32	-
Procesos de eventos totales de tipo E/S y Timer	32	64	32	64	128	64	64	128	-

X o valor tareas o procesos disponibles (el valor es el número máximo)

- tareas o procesos no disponibles.

aplicación, estructura del programa



Objeto

Este capítulo describe la estructura y la ejecución de los programas realizados con el programa Unity Pro.

Contenido de este capítulo

Este capítulo contiene las siguientes secciones:

Sección	Apartado	Página
3.1	Descripción de las tareas y de los procesamientos	76
3.2	Descripción de secciones y subrutinas	82
3.3	Ejecución monotarea	87
3.4	Ejecución multitarea	96

3.1 Descripción de las tareas y de los procesamientos

Objeto

Esta sección describe las tareas y los tratamientos que componen el programa de aplicación.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Presentación de la tarea maestra	77
Presentación de la tarea rápida	78
Presentación de las tareas auxiliares	79
Descripción general del procesamiento de eventos	81

Presentación de la tarea maestra

Generalidades

La tarea maestra representa la tarea principal del programa de aplicación. Es obligatoria y se crea de forma predeterminada.

Estructura

La tarea maestra (MAST) se compone de secciones y subrutinas.

Cada sección de la tarea maestra está programada en los siguientes lenguajes: LD, FBD, IL, ST o SFC.

Las subrutinas están programadas en LD, FBD, IL o ST y se llaman en las secciones de tareas.

NOTA: SFC sólo se puede utilizar en las secciones de la tarea maestra. El número de secciones programadas en SFC es ilimitado.

Ejecución

La ejecución de la tarea maestra se puede elegir:

- cíclica (sección predeterminada)
- o periódica (1 a 255 ms)

Control

La tarea maestra se puede controlar mediante programa, bits y palabras de sistema.

Objetos de sistema	Descripción
%SW0	Periodo de tareas.
%S30	Activación de la tarea maestra.
%S11	Error del watchdog.
%S19	Desborde de periodo.
%SW27	Número de ms transcurridos en el sistema durante el último ciclo MAST.
%SW28	Tiempo máximo de administración del sistema (en ms) de Modicon M340.
%SW29	Tiempo mínimo de administración del sistema (en ms) de Modicon M340.
%SW30	Tiempo de ejecución (en ms) del último ciclo.
%SW31	Tiempo de ejecución (en ms) del ciclo más largo.
%SW32	Tiempo de ejecución (en ms) del ciclo más corto.

Presentación de la tarea rápida

Generalidades

La tarea rápida está destinada a los procesamientos de corta duración y periódicos.

Estructura

La tarea rápida (FAST) se compone de secciones y de subrutinas.

Cada sección de la tarea rápida se programa en uno de los lenguajes: LD, FBD, IL o ST.

El lenguaje SFC no se puede utilizar en las secciones de una tarea rápida.

Las subrutinas se programan en los lenguajes LD, FBD, IL o ST y se requieren en las secciones de la tarea.

Ejecución

La ejecución de la tarea rápida es periódica.

Tiene más prioridad que la tarea maestra.

El periodo de la tarea rápida (FAST) queda fijado en la configuración entre 1 y 255 ms.

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

Control

La tarea rápida se puede controlar a través del programa mediante bits y palabras del sistema.

Objetos del sistema	Descripción
%SW1	Duración de la tarea.
%S31	Activación de la tarea rápida.
%S11	Fallo de watchdog.
%S19	Rebasamiento del periodo.
%SW33	Tiempo de ejecución (en ms) del último ciclo.
%SW34	Tiempo de ejecución (en ms) del ciclo más largo.
%SW35	Tiempo de ejecución (en ms) del ciclo más corto.

Presentación de las tareas auxiliares

Generalidades

Las tareas auxiliares están pensadas para las tareas cuyo tratamiento es más lento. Son las tareas de menor prioridad.

Se pueden programar hasta 4 tareas auxiliares (AUX0, AUX1, AUX2 o AUX3) en los autómatas Premium TSX P57 5•• y Quantum 140 CPU 6••••. Las tareas auxiliares no están disponibles en los autómatas Modicon M340.

Estructura

Las tareas auxiliares (AUX) se componen de secciones y subrutinas.

Cada sección de la tarea auxiliar está programada en uno de los lenguajes siguientes: LD, FBD, IL o ST.

El lenguaje SFC no se puede utilizar en las secciones de una tarea auxiliar.

Se puede programar un máximo de 64 subrutinas en el lenguaje LD, FBD, IL o ST. Dichas subrutinas se llaman en las secciones de tareas.

Ejecución

La ejecución de las tareas auxiliares periódica .

Son las de menor prioridad.

El periodo de las tareas auxiliares puede fijarse entre 10 ms y 2.55 s.

Control

Las tareas auxiliares se pueden controlar mediante programa, bits o palabras de sistema.

Objetos de sistema	Descripción
%SW2	Periodo de la tarea auxiliar 0.
%SW3	Periodo de la tarea auxiliar 1.
%SW4	Periodo de la tarea auxiliar 2.
%SW5	Periodo de la tarea auxiliar 3.
%S32	Activación de la tarea auxiliar 0.
%S33	Activación de la tarea auxiliar 1.
%S34	Activación de la tarea auxiliar 2.
%S35	Activación de la tarea auxiliar 3.
%S11	Error del watchdog
%S19	Desborde de periodo.

Objetos de sistema	Descripción
%SW36	Tiempo de ejecución (en ms) del último ciclo de la tarea auxiliar 0.
%SW39	Tiempo de ejecución (en ms) del último ciclo de la tarea auxiliar 1.
%SW42	Tiempo de ejecución (en ms) del último ciclo de la tarea auxiliar 2.
%SW45	Tiempo de ejecución (en ms) del último ciclo de la tarea auxiliar 3.
%SW37	Tiempo de ejecución (en ms) del ciclo más largo de la tarea auxiliar 0.
%SW40	Tiempo de ejecución (en ms) del ciclo más largo de la tarea auxiliar 1.
%SW43	Tiempo de ejecución (en ms) del ciclo más largo de la tarea auxiliar 2.
%SW46	Tiempo de ejecución (en ms) del ciclo más largo de la tarea auxiliar 3.
%SW38	Tiempo de ejecución (en ms) del ciclo más corto de la tarea auxiliar 0.
%SW41	Tiempo de ejecución (en ms) del ciclo más corto de la tarea auxiliar 1.
%SW44	Tiempo de ejecución (en ms) del ciclo más corto de la tarea auxiliar 2.
%SW47	Tiempo de ejecución (en ms) del ciclo más corto de la tarea auxiliar 3.

Descripción general del procesamiento de eventos

Generalidades

El procesamiento de eventos se utiliza para reducir el tiempo de respuesta del programa de aplicación a eventos:

- procedentes de módulos de entradas/salidas,
- procedentes de temporizadores de evento.

Estas tareas de procesamiento se ejecutan con prioridad sobre todas las demás tareas. Por tanto, son idóneas para procesar tareas que requieren un tiempo de respuesta muy corto en relación con el evento.

El número de tareas procesadoras de eventos (*véase página 73*) que se pueden programar depende del tipo de procesador.

Estructura

Una tarea procesadora de eventos es uniseccional, y se compone de una sola sección (incondicional).

Está programada en lenguaje LD, FBD, IL o ST.

Se ofrecen dos tipos de evento:

- Evento de E/S: para eventos procedentes de módulos de entradas/salidas,
- Evento de TEMPORIZADOR: para eventos procedentes de temporizadores de evento.

Ejecución

La ejecución de una tarea procesadora de eventos es asíncrona.

Cuando se produce un evento, el programa de aplicación es redirigido a la tarea de procesamiento asociada a canal de entrada/salida o al temporizador de evento que originó el evento.

Control

Las palabras y bits de sistema siguientes se pueden utilizar para controlar las tareas procesadoras de eventos durante la ejecución del programa.

Objetos de sistema	Descripción
%S38	Activación del procesamiento de eventos.
%S39	Saturación del snack de gestión de llamadas a eventos.
%SW48	Número de tareas procesadoras de eventos ejecutadas.
%SW75	Número de eventos de tipo timer en cola.

3.2 Descripción de secciones y subrutinas

Objeto

En esta sección, se describen las secciones y subrutinas que conforman una tarea.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Descripción de las secciones	83
Descripción de secciones de SFC	85
Descripción de las subrutinas	86

Descripción de las secciones

Descripción general 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 (LD)
- Lenguaje de bloques funcionales (FBD)
- Lista de instrucciones (IL)
- Literal estructurado (ST)
- Diagrama funcional en secuencia (SFC)

con la condición de que el lenguaje se admita en la tarea.

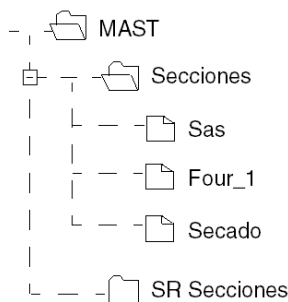
Las secciones se ejecutan en el mismo orden en que se han programado en la ventana del navegador (vista estructural).

Se puede asociar una condición de ejecución a una o varias secciones en las tareas maestra, rápida y auxiliares, pero no así en los procesamiento de eventos.

Las secciones están conectadas a una tarea. Una misma sección no puede pertenecer simultáneamente a varias tareas.

Ejemplo

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



Características de una sección

La tabla siguiente describe las características de una sección.

Característica	Descripción
Nombre	32 caracteres como máximo (los acentos se pueden utilizar, pero no así los espacios).
Lenguaje	LD, FBD, IL, ST o SFC
Tarea o tratamiento	Maestra, rápida, auxiliares, de sucesos
Condición (opcional)	Se puede utilizar una variable bit de tipo BOOL o EBOOL para condicionar la ejecución de la sección.
Comentario	máximo 256 caracteres
Protección	Protección contra la escritura, protección contra lectura/escritura.

Descripción de secciones de SFC

Generalidades

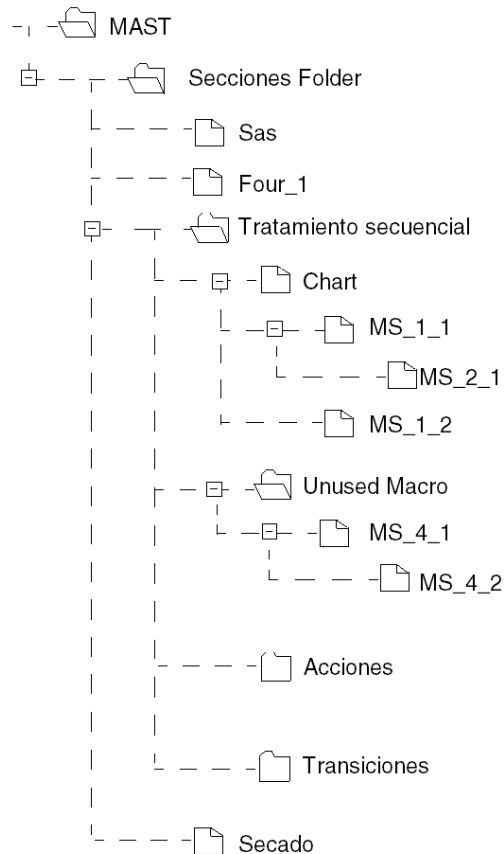
Las secciones en lenguaje de diagrama funcional en secuencia se componen:

- de un gráfico principal (Chart) programado en SFC
- de macro etapas (MS) programadas en SFC
- de acciones y transiciones programadas en LD, FBD , ST o IL

Las secciones SFC se pueden programar únicamente en la tarea maestra (véase la descripción detallada de las secciones SFC).

Ejemplo

La siguiente ilustración muestra un ejemplo de composición de una sección SFC y las llamadas de las macro etapas utilizadas a partir del gráfico (Chart).



Descripción de las subrutinas

Descripción general de subrutinas

Las subrutinas se programan como entidades independientes, en:

- lenguaje de contactos LD,
- lenguaje de bloques funcionales FBD,
- lista de instrucciones IL,
- Literal estructurado (ST).

Las llamadas de subrutinas se realizan desde las secciones o desde otra subrutina.

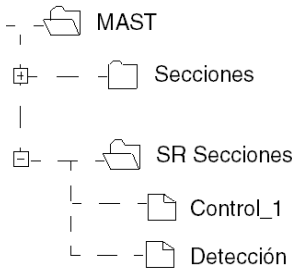
El número máximo de intercalados es 8.

Una subrutina no se puede llamar a sí misma (no es recursiva).

Las subrutinas también están conectadas a una tarea. No es posible llamar la misma subrutina desde varias tareas.

Ejemplo

El siguiente diagrama muestra una tarea estructurada en secciones y subrutinas.



Características de una subrutina

La tabla siguiente describe las características de una subrutina.

Característica	Descripción
Nombre	32 caracteres como máximo (los acentos se pueden utilizar, pero no así los espacios).
Lenguaje	LD, FBD, IL o ST.
Tarea	Maestra, rápida o auxiliar
Comentario	máximo 512 caracteres

3.3 Ejecución monotarea

Objeto

Esta sección describe el funcionamiento de una aplicación monotarea.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Descripción del ciclo de las tareas maestras	88
Monotarea: Ejecución cíclica	90
Ejecución periódica	91
Control del tiempo de ciclo	93
Ejecución de las secciones de Quantum con entradas/salidas descentralizadas	94

Descripción del ciclo de las tareas maestras

Generalidades

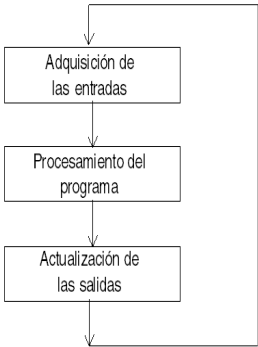
El programa de una aplicación monotarea se asocia a una única tarea del usuario, la tarea maestra (*véase página 77*).

La ejecución de la tarea maestra se puede elegir:

- cíclica
- periódica

Figura

La siguiente figura muestra el ciclo de funcionamiento.



Descripción de las distintas fases

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

Fase	Descripción
Adquisición de las entradas	Escritura en memoria del estado de la información presente en las entradas de los módulos TON y de función específica asociadas a la tarea. Estos valores se pueden modificar por los valores de forzado.
Procesamiento del programa	Ejecución del programa de aplicación, escrito por el usuario.
Actualización de las salidas	Escritura de los bits o de las palabras de salidas asociadas a los módulos TON y de función específica, incorporados a la tarea según el estado definido mediante el programa de aplicación. Al igual que para las entradas, la escritura de las salidas se puede modificar por los valores de forzado.

NOTA: Durante las fases de adquisición de las entradas y de actualización de las salidas, el sistema lleva a cabo también implícitamente la supervisión del autómata (gestiona los bits y palabras del sistema, actualiza los valores actuales del reloj de tiempo real, actualiza los indicadores de estado de los indicadores LED y las pantallas LCD (excepto en Modicon M340), detecta los cambios RUN/STOP, etc.) y el procesamiento de las peticiones procedentes del terminal (modificaciones y animación).

Modo de funcionamiento

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

Autómata en STOP, el procesador realiza:

- El procesamiento interno
- La adquisición de las entradas (1)
- 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.

(1) caso de los autómatas Premium, Atrium y Quantum, la adquisición de las entradas se inhibe cuando el autómata está en STOP.

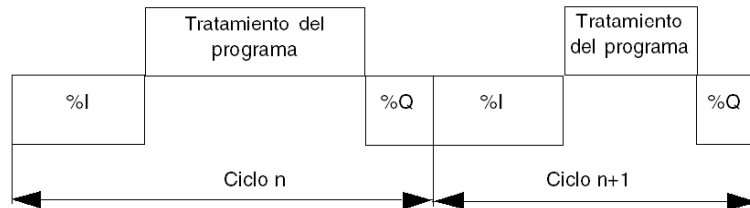
Monotarea: Ejecución cíclica

Generalidades

La tarea maestra funciona como se indica a continuación. Se ofrece una descripción de la ejecución cíclica de la tarea maestra en una operación monotarea.

Funcionamiento

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



%I Lectura de las entradas

%Q Escritura de las salidas

Descripción

Este tipo de operación consiste en secuenciar los ciclos de la tarea uno tras otro. Una vez actualizadas las salidas, el sistema lleva a cabo su propio procesamiento específico y, a continuación, inicia otro ciclo de la tarea, sin detenerse.

Control del ciclo

El ciclo se controla mediante el watchdog (*véase página 93*).

Ejecución periódica

Descripción

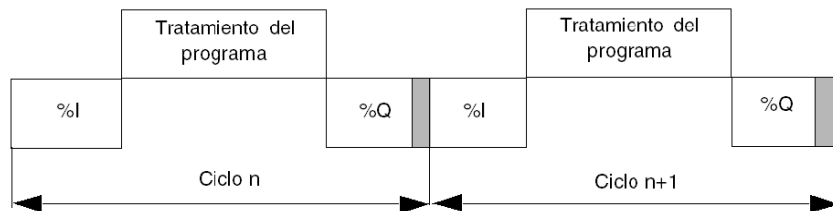
En este modo de funcionamiento, la adquisición de las entradas, el tratamiento del programa de aplicación y la actualización de las salidas se efectúan periódicamente según un tiempo definido de 1 a 255 ms.

En el inicio del ciclo del autómata, una temporización en la que el valor actual se inicializa en función del periodo definido, empieza a descontar.

El ciclo del autómata debe finalizar antes de que termine dicha temporización, la cual reinicia un nuevo ciclo.

Funcionamiento

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



%I Lectura de las entradas

%Q Escritura de las salidas

Modo de funcionamiento

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.

Control del ciclo

Se ejecutan dos controles:

- rebasamiento del periodo (*véase página 93*), y
- por watchdog (*véase página 93*).

Control del tiempo de ciclo

Generalidades

El periodo de ejecución de la tarea maestra, en operación cíclica o periódica, se controla mediante el autómata (watchdog) y no debe sobrepasar el valor definido en la configuración de Tmax (1500 ms de forma predeterminada, 1.5 s como máximo).

Watchdog del software (operación periódica o cíclica)

Si se produce desborde del watchdog, se declara un error en la aplicación, que provoca la detención inmediata del autómata (estado PAUSA).

El bit %S11 indica un desborde de watchdog. El sistema lo ajusta a 1 cuando el tiempo de ciclo sobrepasa al watchdog.

La palabra %SW11 contiene el valor del watchdog en ms. Dicho valor no lo puede modificar el programa.

NOTA:

- La reactivación de la tarea requiere la conexión del terminal para analizar la causa del error, corregirlo, reinicializar el autómata y cambiarlo a EJECUTAR.
- No se puede salir de PAUSA cambiando a DETENER. Para ello, debe reinicializar la aplicación para garantizar la coherencia de los datos.

Control en operación periódica

En operación periódica, un control adicional permite detectar un desborde de periodo. Los desbordes de periodo no hacen que el autómata se detenga si se mantienen por debajo del valor del watchdog.

El bit %S19 indica un desborde de periodo. El sistema lo ajusta a 1 cuando el tiempo de ciclo sobrepasa al tiempo del watchdog.

La palabra %SW0 contiene el valor del periodo (en ms). Se inicializa en un reinicio en frío mediante el valor definido. Es modificable por el usuario.

Explotación de los tiempos de ejecución de la tarea maestra

Las palabras de sistema siguientes pueden usarse para obtener información sobre el tiempo de 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: Estos diversos elementos de información también son accesibles de forma explícita desde el editor de configuración.

Ejecución de las secciones de Quantum con entradas/salidas descentralizadas

Generalidades

Los PLC Quantum poseen un sistema de gestión de secciones específico que se aplica a las estaciones de entradas/salidas descentralizadas.

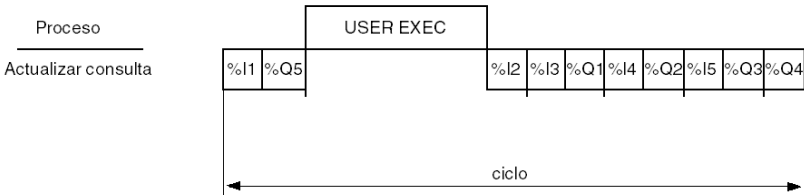
Estas estaciones se utilizan con los módulos RIO siguientes:

- 140 CRA 931 00
- 140 CRA 932 00

Este sistema permite actualizar las entradas/salidas descentralizadas en las secciones, por lo que garantiza unos mejores tiempos de reacción (sin esperar todo el ciclo de la tarea para actualizar las entradas/salidas).

Funcionamiento

En el diagrama siguiente se muestran las fases de E/S cuando se asocian 5 estaciones a secciones de tarea de cliente.



%Ii entradas de la estación n.º i

%Qi salidas de la estación n.º i

i número de estación

Descripción

Fase	Descripción
1	Petición de actualización: • las entradas de la primera estación (i=1) • las salidas de la última estación (i=5)
2	Procesamiento del programa
3	• Actualización de las entradas de la primera estación (i=1) • Petición de actualización de las entradas de la segunda estación (i=2)
4	Petición de actualización: • las entradas de la tercera estación (i=3) • las salidas de la primera estación (i=1)
5	Petición de actualización: • las entradas de la cuarta estación (i=4) • las salidas de la segunda estación (i=2)

Fase	Descripción
6	Petición de actualización: ● las entradas de la última estación ($i=5$) ● las salidas de la tercera estación ($i=3$)
7	Petición de actualización de las salidas de la cuarta estación ($i=4$)

3.4 Ejecución multitarea

Objeto

Esta sección describe el funcionamiento de una aplicación multitarea.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Estructura del software multitarea	97
Desglose secuencial de las tareas en una estructura multitarea	99
Control de tareas	101
Asignación de los canales de entradas/salidas a las tareas maestra, rápida y auxiliares	104
Gestión de los tratamientos de sucesos	106
Ejecución del procesamiento de eventos de tipo TIMER	108
Intercambios de entradas/salidas en el procesamiento de eventos	112
Programación de procesamiento de eventos	113

Estructura del software multitarea

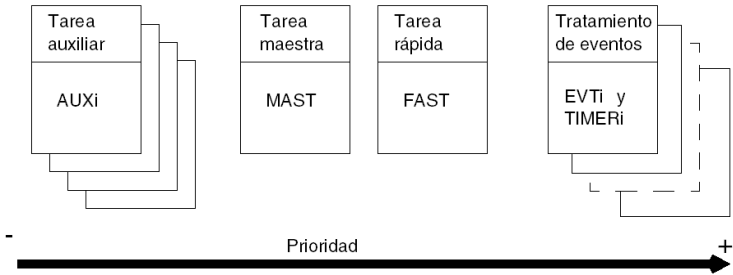
Tareas y procesamientos

La estructura de tareas de este tipo de aplicación es la siguiente:

Tarea/procesa miento	Designación	Descripción
Maestra	MAST	Siempre presente, puede ser cíclica o periódica.
Rápida	FAST	Opcional, siempre periódica.
Auxiliar	AUX de 0 a 3	Opcional, siempre periódica.
Evento	EVTi y TIMERi (véase <i>págin a 106</i>)	Llamada por el sistema cuando se produce un evento en un módulo de entradas/salidas o activada mediante el temporizador de eventos. Estos tipos de procesamientos son opcionales y pueden emplearlos aplicaciones que necesitan actuar sobre las entradas/salidas con un tiempo de respuesta reducido.

Figura

El siguiente diagrama muestra las tareas en una estructura multitarea y su nivel de prioridad:



Descripción

La tarea maestra (MAST) sigue siendo la base de la aplicación. Las otras tareas varían en función del tipo de PLC (véase *página 73*).

Se establecen niveles de prioridad para cada tarea con el fin de dar prioridad a determinados tipos de procesamiento.

El procesamiento de eventos se puede activar de forma asíncrona con respecto a tareas periódicas a través de una orden generada por eventos externos. Se procesa como una prioridad y requiere la detención de cualquier procesamiento en curso.

Precauciones

Tareas múltiples: reglas de oro

ATENCIÓN

Tareas múltiples: reglas de oro

El uso compartido de entradas/salidas entre diferentes tareas puede producir un comportamiento inesperado de la aplicación.

Se recomienda especialmente que asocie cada salida o cada entrada a una sola tarea.

Si no se siguen estas instrucciones pueden producirse lesiones personales o daños en el equipo.

Desglose secuencial de las tareas en una estructura multitarea

Generalidades

- La tarea maestra está activa por defecto.
- Las tareas rápidas y auxiliares están activas por defecto si están programadas.
- El tratamiento de sucesos se activa en el momento de la aparición del suceso que se le ha asociado.

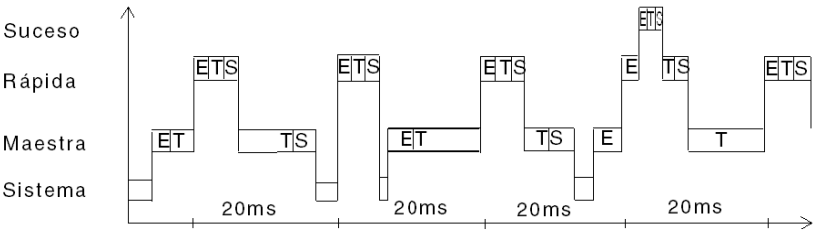
Funcionamiento

En la tabla siguiente se describe la ejecución de las tareas prioritarias (este funcionamiento se muestra también en el esquema siguiente).

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 esquema siguiente ilustra el desglose secuencial de las tareas de un procesamiento multitarea que incluye una tarea maestra cíclica, una tarea rápida con un periodo de 20 ms y un procesamiento de eventos.



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 controlar la activación o desactivación de la tarea maestra MAST.
- %S31 permite controlar la activación o desactivación de la tarea rápida FAST.
- %S32 a %S35 permiten activar o no las tareas auxiliares AUX0 a AUX3.
- %S38 permite activar o desactivar los tratamientos de sucesos EVTi.

NOTA: Las funciones elementales MASKEVT y UNMASKEVT permiten también el enmascaramiento y desenmascaramiento global de los eventos por parte del programa.

Control de tareas

Operación cíclica y periódica

En una operación multitarea, la tarea de mayor prioridad se utilizará en modo periódico para dejar tiempo suficiente para que se ejecuten las tareas de menor prioridad.

Por ello, sólo la tarea de prioridad inferior debe utilizarse en modo cíclico. Así pues, al elegir el modo operativo cíclico para la tarea maestra se excluye el uso de tareas auxiliares.

Medición de las duraciones de tareas

La duración de las tareas se mide de forma continua. Dicha medición representa la duración entre el inicio y el final de ejecución de la tarea.

Incluye el tiempo empleado por las tareas de mayor prioridad que pudieran interrumpir la ejecución de las tarea objeto de medición.

Las palabras de sistema siguientes indican los tiempos de ciclo actual, máximo y mínimo de cada tarea (valor en ms)

Medición de tiempos	MAST	FAST	AUX0	AUX1	AUX2	AUX3
Corriente	%SW30	%SW33	%SW36	%SW39	%SW42	%SW45
Máximo	%SW31	%SW34	%SW37	%SW40	%SW43	%SW46
Mínimo	%SW32	%SW35	%SW38	%SW41	%SW44	%SW47

NOTA: Los tiempos máximo y mínimo se toman de los tiempo medidos desde el último reinicio en frío.

Periodos de tareas

Los periodos de tareas se definen en las propiedades de tarea. Pueden ser modificados mediante las palabras de sistema siguientes.

Palabras de sistema	Tarea	Valores	Valores predeterminados	Observaciones
%SW0	MAST	0..255 ms	Cíclica	0 = operación cíclica
%SW1	FAST	1..255 ms	5 ms	-
%SW2	AUX0	10 ms..2.55 s	100 ms	Los valores del periodo se expresan en 10 ms.
%SW3	AUX1	10 ms..2.55 s	200 ms	
%SW4	AUX2	10 ms..2.55 s	300 ms	
%SW5	AUX3	10 ms..2.55 s	400 ms	

Cuando el tiempo del ciclo de la tarea sobrepasa el periodo, el sistema ajusta el bit de sistema %S19 de la tarea a 1 y continua con el siguiente ciclo.

NOTA: Los valores de los periodos no dependen de la prioridad de las tareas. Se puede definir el periodo de una tarea rápida que sea mayor que el de la tarea maestra.

Watchdog

La ejecución de cada tarea se controla mediante un watchdog configurable utilizando las propiedades de tarea.

La siguiente tabla ofrece el rango de valores watchdog de cada una de las tareas:

Tareas	Valores watchdog (mín...máx) (ms)	Valor watchdog predeterminado (ms)	Palabra de sistema asociada
MAST	10..1500	250	%SW11
FAST	10..500	100	-
AUX0	100..5000	2000	-
AUX1	100..5000	2000	-
AUX2	100..5000	2000	-
AUX3	100..5000	2000	-

Si se produce desborde del watchdog, se declara un error en la aplicación, que provoca la detención inmediata del autómatas (estado PAUSA).

La palabra %SW11 contiene el valor watchdog de la tarea maestra en ms. Dicho valor no lo puede modificar el programa.

El bit %S11 indica un desborde de watchdog. El sistema lo ajusta a 1 cuando el tiempo de ciclo sobrepasa al watchdog.

NOTA:

- La reactivación de la tarea requiere la conexión del terminal para analizar la causa del error, corregirlo, reinicializar el autómatas y cambiarlo a EJECUTAR.
- No se puede salir de PAUSA cambiando a DETENER. Para ello, debe reinicializar la aplicación para garantizar la coherencia de los datos.

Control de las tareas

Durante la ejecución del programa de aplicación, se puede activar o inhibir una tarea utilizando los bits de sistema siguientes:

Bits de sistema	Tarea
%S30	MAST
%S31	FAST
%S32	AUX0

Bits de sistema	Tarea
%S33	AUX1
%S34	AUX2
%S35	AUX3

La tarea está activa cuando el bit de sistema asociado está ajustado en 1. El sistema comprueba estos bits al final de la tarea maestra.

Cuando se inhibe una tarea, se siguen leyendo las entradas y escribiendo las salidas.

Al iniciar el programa de aplicación, la tarea maestra sólo está activa durante el primer ciclo de ejecución. Al final del primer ciclo automáticamente se activan las demás tareas salvo que una de ellas esté inhibida (bit de sistema asociado ajustado a 0) por el programa.

Controles sobre las fases de lectura de entradas y escritura de salidas

Los bits de las palabras de sistema siguientes se pueden utilizar (sólo cuando el autómata está en la modalidad EJECUTAR) para inhibir las fases de lectura de entrada y escritura de salidas.

Inhibición de fases...	MAST	FAST	AUX0	AUX1	AUX2	AUX3
lectura de entradas	%SW8.0	%SW8.1	%SW8.2	%SW8.3	%SW8.4	%SW8.5
escritura de salidas	%SW9.0	%SW9.1	%SW9.2	%SW9.3	%SW9.4	%SW9.5

NOTA: De forma predeterminada, las fases de lectura de entradas y escritura de salidas están activas (bits de las palabras de sistema %SW8 y %SW9 ajustados a 0).

En Quantum, las entradas/salidas que se distribuyen mediante el bus DIO no se asignan por las palabras %SW8 y %SW9.

Asignación de los canales de entradas/salidas a las tareas maestra, rápida y auxiliares

Generalidades

Cada tarea garantiza la escritura y la lectura de las entradas/salidas que tiene asignadas.

La asociación de un canal, de un grupo de canales o de un módulo de entradas/salidas a una tarea se define en la pantalla de configuración del módulo correspondiente.

La tarea asociada por defecto es la tarea MAST.

Lectura de las entradas y escritura de las salidas en Premium

Todos los canales de entradas/salidas de los módulos en bastidores se pueden asociar a una tarea (MAST, FAST o AUX 0..3).

Caso de las entradas\salidas locales y remotas (bus X):

En cada ciclo de la tarea, las entradas se leen al principio de la tarea y las salidas se escriben al final de la tarea.

Caso de las entradas\salidas remotas en el bus Fipio:

En modalidad controlada, la actualización de las entradas/salidas se correlaciona con el período de la tarea. El sistema garantiza la actualización de las entradas/salidas en un solo período. Sólo se actualizan las entradas/salidas asociadas a dicha tarea.

En esta modalidad, el período de la tarea del autómata (MAST, FAST o AUX) debe ser superior o igual al tiempo de ciclo de red.

En modalidad libre, no se impone ningún límite al período de la tarea. El período de la tarea del autómata (MAST, FAST o AUX) puede ser inferior al tiempo de ciclo de red. En tal caso, la tarea se puede ejecutar sin actualizar las entradas/salidas. La selección de esta modalidad ofrece la posibilidad de tener las duraciones de tarea más cortas posibles en el caso de aplicaciones en las que la rapidez es fundamental.

Lectura de las entradas y escritura de las salidas en Quantum

Caso de las entradas\salidas locales:

Cada módulo o grupo de módulos de entradas/salidas se puede asociar únicamente a una tarea (MAST, FAST o AUX 0..3).

Caso de las entradas\salidas descentralizadas:

Las estaciones de entradas/salidas remotas se pueden asociar únicamente a la tarea maestra (MAST). La asignación se efectúa en las secciones, con 1 estación de entradas remotas y 1 estación de salidas remotas por sección.

Caso de las entradas\salidas distribuidas:

Las estaciones de entradas/salidas distribuidas se pueden asociar únicamente a la tarea maestra (MAST).

Las entradas se leen al principio de la tarea maestra y las salidas se escriben al final de dicha tarea.

Ejemplo en Premium

La disposición de los módulos TON Premium es de 8 canales sucesivos (canales 0 a 7, 8 a 15, etc.), las entradas/salidas pueden asignarse por grupos de 8 canales, independientemente de la tarea MAST, AUXi o FAST.

Ejemplo: se puede asignar los canales de un módulo de 28 entradas/salidas de la forma siguiente:

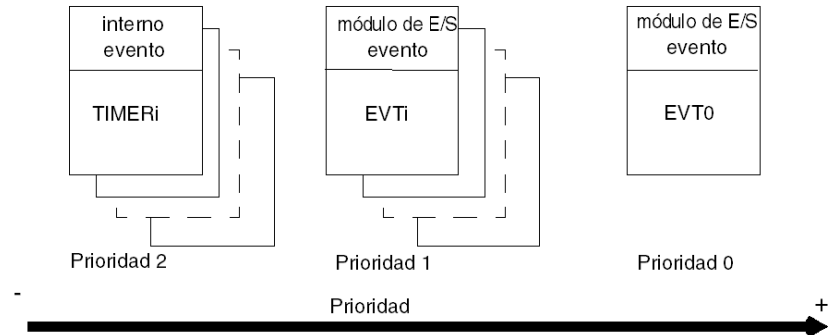
- Entradas 0 a 7 asociadas a la tarea MAST.
- Entradas 8 a 15 asociadas a la tarea FAST.
- Salidas 0 a 7 asociadas a la tarea MAST.
- Salidas 8 a 15 asociadas a la tarea AUX0.

Gestión de los tratamientos de sucesos

General

Los tratamientos de sucesos tienen prioridad sobre las tareas.

En la figura siguiente se describen los 3 niveles de prioridad definidos:



Gestión de las prioridades

- El tratamiento de sucesos EVT0 es el de mayor prioridad. Puede por sí solo interrumpir los demás tratamientos de sucesos.
- Los tratamientos de sucesos EVTi iniciados por módulos de entradas/salidas (prioridad 1) tienen prioridad sobre los tratamientos de sucesos TIMERi iniciados por temporizadores (prioridad 2).
- **En los autómatas Modicon M340, Premium y Atrium:** los tipos de procesamiento de eventos con prioridad 1 se memorizan y procesan por orden de llegada.
- **En los autómatas Quantum:** se determina la prioridad de los tipos de procesamiento de prioridad 1:
 - por la posición del módulo de entradas/salidas en el rack,
 - por la posición de la vía en el módulo.

El módulo con el número de posición más bajo tiene la prioridad más alta.
- Los tratamientos de sucesos iniciados por temporizador tienen la prioridad 2. La prioridad de tratamiento se determina por el número de temporizador más bajo.

Control

El programa de aplicación puede validar o inhibir de forma global los distintos tipos de procesamientos de eventos utilizando el bit de sistema %S38. Si se producen uno o más eventos mientras están inhibidos, se perderá el procesamiento asociado.

Dos funciones elementales del lenguaje, `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.

Ejecución del procesamiento de eventos de tipo TIMER

Descripción

Los procesamientos de eventos de tipo TIMER son procesamientos iniciados mediante la función ITCNTRL (véase *Unity Pro, Sistema, Librería de bloques*). Esta función de temporización activa de forma periódica el procesamiento de eventos cada vez que se alcanza el valor de preselección.

Referencia

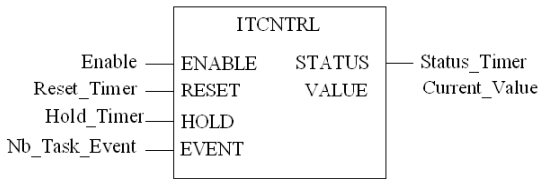
Los siguientes parámetros se seleccionan en las propiedades de procesamiento de eventos.

Parámetro	Valor	Valor predeterminado	Función
Base de tiempo	1 ms, 10 ms, 100 ms, 1 s	10 ms	Base de tiempo del temporizador. Nota: la base de tiempo de 1 ms se debe utilizar con precaución, ya que existe el riesgo de que se produzca un desborde si la frecuencia de inicio de los procesamientos es demasiado intensa.
Preajuste	1 a 1023	10	Valor de preselección del temporizador. La temporización elaborada es igual a: preselección x base de tiempo.
Fase	0 a 1023	0	Valor de desplazamiento temporal entre la transición de STOP a RUN del PLC y el primer reinicio del temporizador desde 0. El valor temporal es igual a: fase x base de tiempo.

NOTA: La fase debe ser inferior a la preselección del evento de tipo TIMER.

Función ITCNTRL

Representación en FBD:



La tabla siguiente describe los parámetros de entrada:

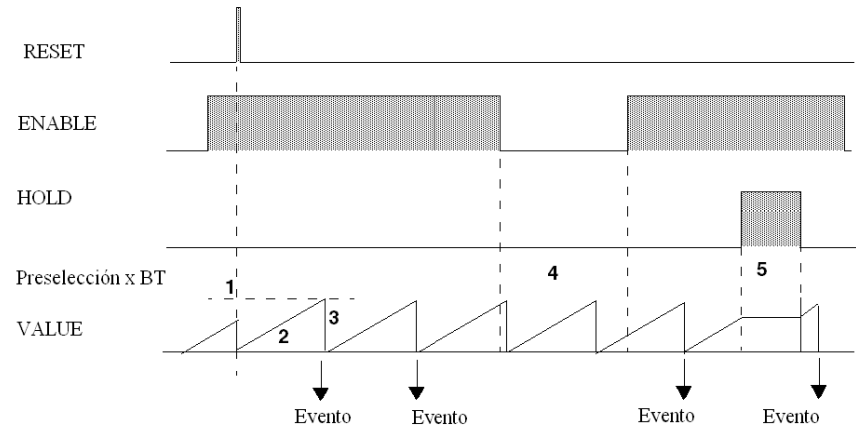
Parámetro	Tipo	Comentario
Enable	BOOL	Entrada de validación.
Reset_Timer	BOOL	En 1, reinicializa el temporizador.
Hold_Timer	BOOL	En 1, inmoviliza el incremento del temporizador.
Nb_Task_Event	BYTE	Byte de entrada que determina el número del procesamiento de eventos que se va a activar.

En la tabla siguiente se describen los parámetros de salida:

Parámetro	Tipo	Comentario
Status_Timer	WORD	Palabra de estado.
Current_Value	TIME	Valor actual del temporizador.

Gráfico de tendencias de funcionamiento normal

Gráfico de tendencias.



Funcionamiento normal

En la siguiente tabla se describe el inicio de los procesamientos de eventos de tipo TIMER (consulte el gráfico de tendencias que aparece más arriba).

Fase	Descripción
1	En el caso de que se reciba un flanco ascendente en la entrada <code>RESET</code> , el temporizador se pone a 0.
2	El valor actual <code>VALUE</code> del temporizador aumenta de 0 al valor de preselección de una unidad a cada impulso de la base de tiempo.
3	Se emite un evento cuando el valor actual alcanza el valor de preselección, el temporizador se pone a 0 y, a continuación, se vuelve a activar. El procesamiento de eventos asociado se inicia si el evento no está enmascarado. Puede ser diferido si se está ejecutando un procesamiento de eventos de igual o superior prioridad.
4	Cuando la entrada <code>ENABLE</code> está a 0, los eventos dejan de emitirse. Los procesamientos de eventos de tipo TIMER no se inician.
5	Cuando la entrada <code>HOLD</code> está a 1, el temporizador permanece fijo, el valor actual no aumenta más hasta que se vuelve a poner a 0.

Sincronización del procesamiento de eventos

El parámetro Phase permite iniciar los procesamientos de eventos de tipo TIMER diferentes del intervalo de tiempo constante.

Este parámetro define un desplazamiento temporal con un origen de tiempo absoluto, que es el último paso de STOP a RUN del PLC.

Condición de funcionamiento:

- Los procesamientos de eventos deben presentar los mismos valores de base de tiempo y de preselección.
- Las entradas `RESET` y `HOLD` no se deben posicionar en 1.

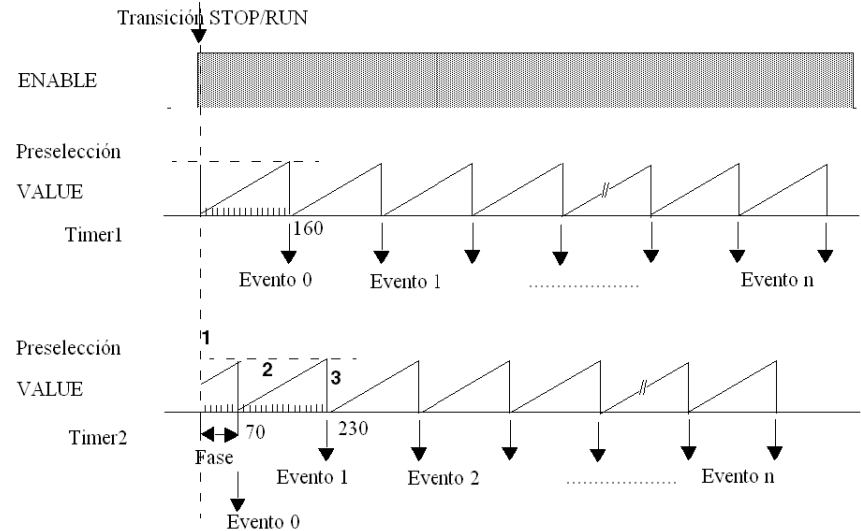
Ejemplo: dos procesamientos de eventos, Timer1 y Timer2, que se deben ejecutar con un intervalo de 70 ms.

El primer procesamiento, Timer1, se puede definir mediante una fase igual a 0 y el segundo, Timer2, mediante una fase de 70 ms (fase de 7 y base de tiempo de 10 ms).

Todo evento iniciado mediante el temporizador asociado al procesamiento Timer1 irá seguido, tras 70 ms, de un evento procedente del temporizador asociado al procesamiento Timer2.

Gráfico de tendencias: Transición de STOP a RUN

Gráfico de tendencias del ejemplo descrito anteriormente con un mismo valor de preselección de 16 (160 ms) para Timer1 y Timer2.



Funcionamiento después de que el PLC haya pasado de STOP a RUN

En la siguiente tabla se describe el funcionamiento tras el paso de STOP a RUN del PLC (consulte el gráfico de tendencias que se muestra más arriba):

Fase	Descripción
1	En el caso de una transición de STOP a RUN del PLC, la temporización se inicia de forma que el valor de preselección se alcance tras un periodo de tiempo igual a fase x base de tiempo, momento en que se emitirá el primer evento.
2	El valor actual VALUE del temporizador aumenta de 0 al valor de preselección de una unidad a cada impulso de la base de tiempo.
3	Se emite un evento cuando el valor actual alcanza el valor de preselección, el temporizador se pone a 0 y, a continuación, se vuelve a activar. El procesamiento de eventos asociado se inicia si el evento no está enmascarado. Puede ser diferido si se está ejecutando un procesamiento de eventos de prioridad igual o superior.

Intercambios de entradas/salidas en el procesamiento de eventos

Generalidades

En cada tratamiento de sucesos se pueden usar otras vías de entradas/salidas que no sean las propias del suceso.

Al igual que para las tareas, los intercambios se realizan implícitamente mediante el sistema antes (%I) y después (%Q) del tratamiento de aplicación.

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 al tratamiento de sucesos se obtienen automáticamente.
3	Se ejecuta el tratamiento de sucesos. Deberá ser lo más breve posible.
4	Se actualizan todas las salidas asociadas al tratamiento de sucesos.

Caso de Premium/Atrium

Las entradas adquiridas y las salidas que se actualizan son las siguientes:

- las entradas asociadas al canal que ha provocado el evento y
- las entradas y salidas utilizadas en el procesamiento de eventos.

NOTA: Estos intercambios pueden ser relativos:

- a un canal (ejemplo de módulo de conteo) o
- a un grupo de canales (módulo binario). En este 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.

Caso de Quantum

Las entradas adquiridas y las salidas que se actualizan se seleccionan en la configuración. Sólo se pueden elegir entradas/salidas locales.

Regla de programación

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), rápida (FAST) o auxiliar (AUXi).

Programación de procesamiento de eventos

Procedimiento

En la siguiente tabla se resumen los pasos principales de la programación de procesamiento de eventos.

Paso	Acción
1	Fase de configuración (para eventos iniciados por módulos de entrada/salida) En modalidad offline, en el editor de configuración, seleccione Procesamiento de eventos (EVT) y el número de procesamiento de evento para el canal del módulo de entrada/salida correspondiente.
2	Fase de desenmascaramiento La tarea que se puede interrumpir debe en particular: ● Activar el procesamiento de eventos en el nivel de sistema: establecer el bit %S38 en 1 (valor predeterminado). ● Desenmascarar eventos con la instrucción UNMASKEVT (activa de manera predeterminada). ● Desenmascarar los eventos correspondientes en el nivel de canal (en el caso de eventos activados por módulo de entrada/salida) estableciendo en 1 los objetos de lenguaje implícitos del módulo de entrada/salida. De manera predeterminada, los eventos están enmascarados. ● Comprobar que el stack de eventos en el nivel de sistema no está saturado (el bit %S39 debe estar en 0).
3	Fase de creación de programación de eventos El programa debe: ● Determinar el origen de los eventos en función de la palabra de estado de evento asociada con el módulo de entrada/salida si el módulo puede generar diferentes eventos. ● Llevar a cabo el procesamiento reflejo asociado con el evento. El proceso debe ser lo más breve posible. ● Escribir las salidas reflejas correspondientes. Nota: La palabra de estado de eventos se establece automáticamente en 0.

Ilustración del desenmascaramiento de eventos

Esta ilustración muestra el desenmascaramiento de eventos en la tarea MAST.

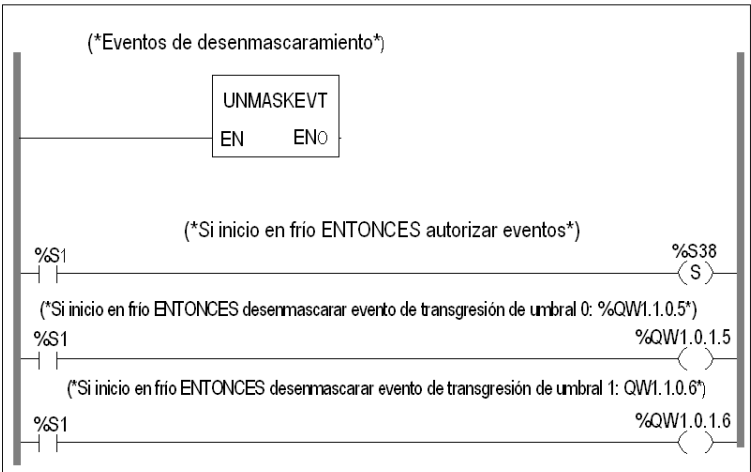
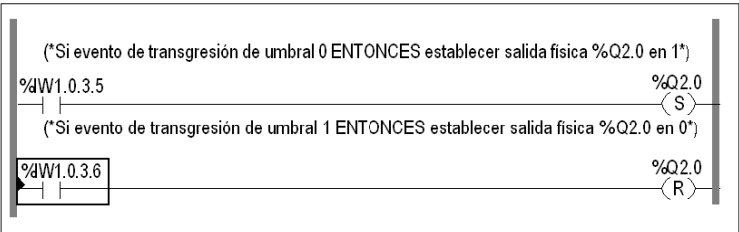


Ilustración de los contenidos del procesamiento de eventos

Esta ilustración muestra los contenidos posibles del procesamiento de eventos (funcionamiento y prueba de bits).



Estructura de la memoria de la aplicación



Objeto

Este capítulo describe la estructura de la memoria de la aplicación de los autómatas Premium, Atrium y Quantum.

Contenido de este capítulo

Este capítulo contiene las siguientes secciones:

Sección	Apartado	Página
4.1	Estructura de memoria de los autómatas Premium, Atrium y Modicon M340	116
4.2	Estructura de memoria de los autómatas Quantum	125

4.1 Estructura de memoria de los autómatas Premium, Atrium y Modicon M340

Objeto

En esta sección se describe la estructura de memoria y se ofrece información detallada de las zonas de memoria de los autómatas Premium, Atrium y Modicon M340.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Estructura de memoria de los PLC Modicon M340	117
Estructura de memoria de los PLC Premium y Atrium	121
Descripción detallada de las zonas de memoria	124

Estructura de memoria de los PLC Modicon M340

Descripción general

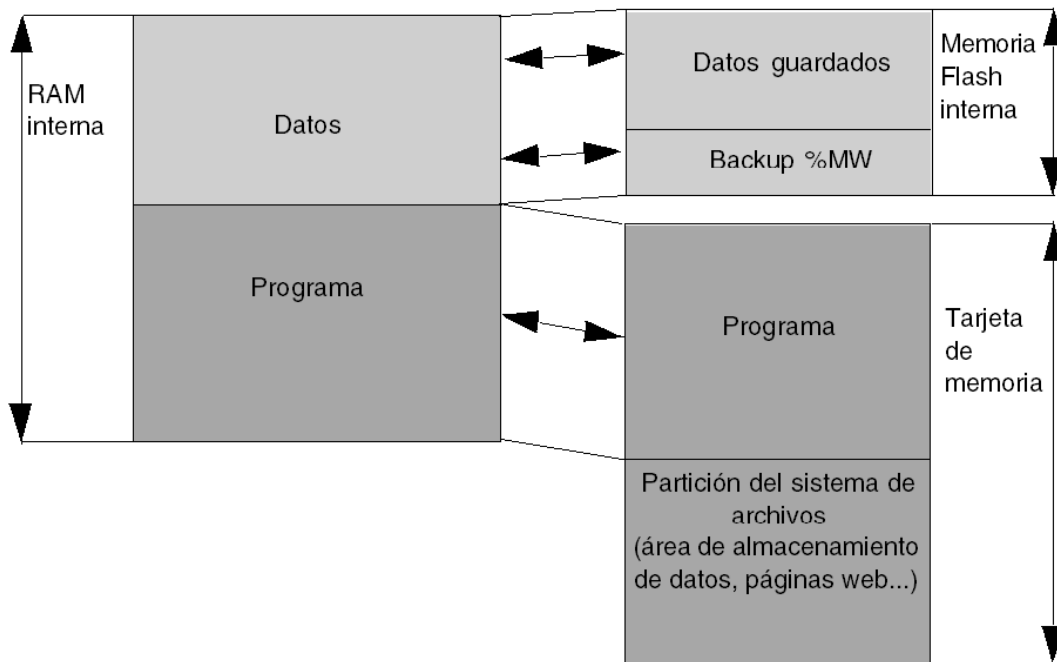
La memoria del PLC contiene:

- **datos localizados de la aplicación,**
- **datos sin localizar de la aplicación y**
- **el programa:** descriptores y código ejecutable de las tareas, palabras constantes, valores iniciales y configuración de entradas/salidas.

Estructura

Los datos y el programa son compatibles con la RAM interna del módulo del procesador.

En el diagrama siguiente se describe la estructura de la memoria.



Backup del programa

Si la tarjeta de memoria se encuentra presente, funciona correctamente y no está protegida contra escritura, el programa se guardará en dicha tarjeta:

- De forma automática, después de:
 - una descarga,
 - una modificación en línea o
 - un flanco ascendente del bit de sistema %S66 en el programa del proyecto.
- De forma manual:
 - con el comando **PLC → Backup del proyecto → Guardar backup**
 - en una tabla de animación, ajustando el bit de sistema %S66.

ADVERTENCIA

EXTRACCIÓN ANTICIPADA DE LA TARJETA DE MEMORIA

La interrupción de un procedimiento de almacenamiento de la aplicación debida a una extracción anticipada o brusca de la tarjeta de memoria puede provocar la pérdida de la aplicación guardada. El bit %S65 (*véase página 176*) permite gestionar una extracción correcta (consulte el bit %65 de la página de ayuda en el capítulo de bits de sistema)

Si no se siguen estas instrucciones pueden producirse lesiones personales graves o mortales o daños en el equipo.

La tarjeta de memoria utiliza tecnología Flash, por lo que no necesita batería.

Restauración del programa

Si la tarjeta de memoria se encuentra presente y funciona correctamente, el programa se copiará de dicha tarjeta de memoria del PLC a la memoria interna:

- De forma automática después de
 - apagar y encender.
- De forma manual, con el comando Unity Pro **PLC → Backup del proyecto → Restaurar backup**

NOTA: Al insertar la tarjeta de memoria en la modalidad de ejecución o de detención, debe realizar un ciclo de apagado y encendido para restaurar el proyecto en el PLC.

Datos guardados

Los datos ubicados, los no ubicados y el búfer de diagnóstico se guardan automáticamente en la memoria interna Flash cuando se desconecta la alimentación. Se inician en caliente.

Save_Param

La función `SAVE_PARAM` realiza el ajuste de los parámetros inicial y actual en la RAM interna (como en otros PLC). En este caso, el contenido de la RAM interna y de la tarjeta de memoria es distinto (%S96 = 0 y el indicador luminoso CARDERR está encendido). Al iniciar en frío (después de restaurar la aplicación), el parámetro actual se reemplaza por los últimos valores iniciales ajustados, sólo si se ha realizado una función de almacenamiento en la tarjeta de memoria (Guardar backup o flanco ascendente %S66).

Almacenamiento del valor actual

En un flanco ascendente %S94, los valores actuales reemplazan a los valores iniciales en la memoria interna. El contenido de la RAM interna y de la tarjeta de memoria es distinto (%S96 = 0 y el indicador luminoso CARDERR está encendido). Al iniciar en frío, los valores actuales se reemplazan por los valores iniciales más recientes, sólo si se ha realizado una función de almacenamiento en la tarjeta de memoria (Guardar backup o flanco ascendente %S66).

Eliminar archivos

Hay dos formas de eliminar todos los archivos de la tarjeta de memoria:

- Formatear la tarjeta de memoria (elimina todos los archivos de la partición del sistema de archivos)
- Borrado del contenido del directorio \DataStorage\ (elimina sólo los archivos agregados por el usuario)

Las dos acciones se realizan con %SW93 (véase página 203).

La palabra del sistema %SW93 sólo puede usarse después de descargar una aplicación predeterminada en el PLC.

ATENCIÓN

TARJETA DE MEMORIA DEFECTUOSA

No formatee la tarjeta de memoria con una herramienta que no sea de Schneider. La tarjeta de memoria necesita una estructura para contener programas y datos. El formateo con otra herramienta destruye esta estructura.

Si no se siguen estas instrucciones pueden producirse lesiones personales o daños en el equipo.

Backup %MW

Los valores de %MWi pueden guardarse en la memoria Flash interna mediante %SW96 (véase página 203). Estos valores se restaurarán al iniciarse en frío, incluida la descarga de aplicaciones, si la opción **Inicializarse %MW con inicio en frío** está desactivada en la pantalla de configuración (véase *Unity Pro, Modalidades de funcionamiento*,) del procesador.

En las palabras %MW, los valores se pueden guardar y restaurar con un reinicio en frío o una descarga si la opción Restablecer de %MW con reinicio en frío no está marcada en la pantalla de configuración del procesador. Con la palabra %SW96, es posible gestionar las palabras internas %MW de acción de la memoria (guardar, eliminar) y la información sobre los estados de las acciones de las palabras internas %MW.

Características de las tarjetas de memoria

Existen dos tipos de tarjeta de memoria:

- **Aplicación:** estas tarjetas contienen el programa de aplicación y páginas web.
- **Aplicación + almacenamiento de archivos:** estas tarjetas contienen el programa de aplicación, archivos de datos de los EFB de gestión de archivos de la tarjeta de memoria y páginas web.

Estructura de memoria de los PLC Premium y Atrium

Generalidades

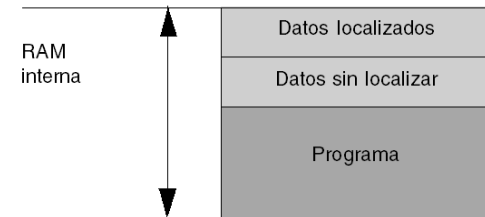
La memoria del autómata contiene:

- **Los datos localizados** de la aplicación,
- **los datos sin localizar** de la aplicación y
- **el programa**: descriptores y código ejecutable de las tareas, palabras constantes, valores iniciales y configuración de las entradas/salidas.

Estructura sin tarjeta de extensión de memoria

La memoria RAM interna del módulo del procesador contiene los datos y el programa.

El diagrama siguiente describe la estructura de la memoria.

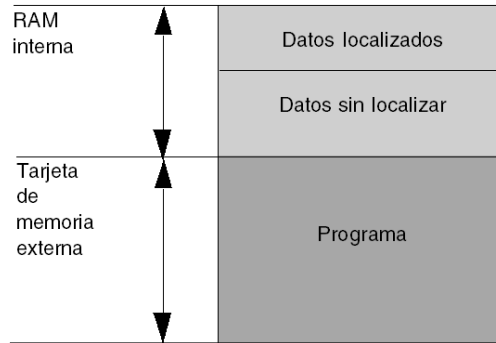


Estructura con tarjeta de ampliación de memoria

Los datos se almacenan en la memoria RAM interna del módulo del procesador.

El programa se guarda en la tarjeta de memoria de extensión.

El diagrama siguiente describe la estructura de la memoria.



Almacenamiento de la memoria

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

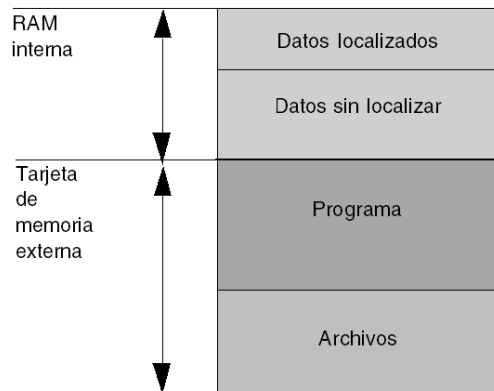
Las tarjetas de memoria RAM están protegidas por una pila de cadmio/níquel.

Especificaciones de las tarjetas de memoria

Existen tres tipos de tarjetas de memoria:

- **De aplicación:** estas tarjetas contienen el programa de la aplicación. Son de tecnología RAM o Flash Eprom.
- **De aplicación y almacenamiento de archivos:** estas tarjetas contienen, además del programa, una zona que permite almacenar/restablecer los datos mediante el programa. Son de tecnología RAM o Flash Eprom
- **De almacenamiento de archivos:** estas tarjetas permiten almacenar/restablecer datos mediante el programa. Estas tarjetas son de tecnología SRAM.

El esquema siguiente describe la estructura de memoria con tarjeta de tipo aplicación y almacenamiento de archivos.



NOTA: En el caso de los procesadores con dos emplazamientos para tarjeta de memoria, el emplazamiento inferior está reservado para la función de almacenamiento de archivos.

Descripción detallada de las zonas de memoria

Datos de usuario

Esta zona contiene los datos localizados y los datos sin localizar de la aplicación.

- datos localizados:
 - datos booleanos %M, %S y numéricos %MW, %SW
 - datos asociados a los módulos %I, %Q, %IW, %QW,%KW....
- datos sin localizar:
 - datos booleanos y numéricos (instancias)
 - Instancias de EFB y DFB

Programa de usuario y constantes

Esta zona contiene los códigos ejecutables y las constantes de la aplicación.

- códigos ejecutables:
 - código de programa
 - código asociado a los EF, EFB y a la gestión de los módulos de E/S
 - código asociado a los DFB
- constantes:
 - palabras constantes KW
 - constantes asociadas a las entradas/salidas
 - valores iniciales de los datos

Esta zona contiene también la información necesaria para descargar la aplicación: Códigos gráficos, símbolos, etc.

Otra información

También se almacena en la memoria otro tipo de información relativa a la configuración y a la estructura de la aplicación (en la zona de datos o de programa en función del tipo de información).

- Configuración: otros datos relativos a la configuración (configuración de hardware, configuración de software).
- Sistema: datos utilizados por el sistema operativo (pila de las tareas, etc.).
- Diagnóstico: información relativa al diagnóstico del proceso o del sistema, búfer de diagnóstico.

4.2 Estructura de memoria de los autómatas Quantum

Finalidad de esta sección

En esta sección se describe la estructura de memoria y se ofrece información detallada de las zonas de memoria de los autómatas Quantum.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Estructura de memoria de los autómatas Quantum	126
Descripción detallada de las zonas de memoria	129

Estructura de memoria de los autómatas Quantum

Generalidades

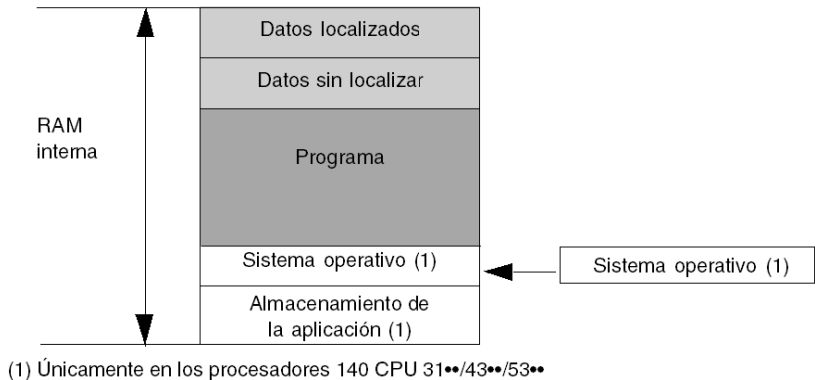
La memoria del autómata contiene:

- **datos localizados de la aplicación** (memoria de señal),
- **los datos sin localizar** de la aplicación y
- **el programa**: descriptores y código ejecutable de las tareas, valores iniciales y configuración de las entradas/salidas.

Estructura sin tarjeta de ampliación de memoria

La memoria RAM interna del módulo del procesador contiene los datos y el programa.

El diagrama siguiente describe la estructura de la memoria.



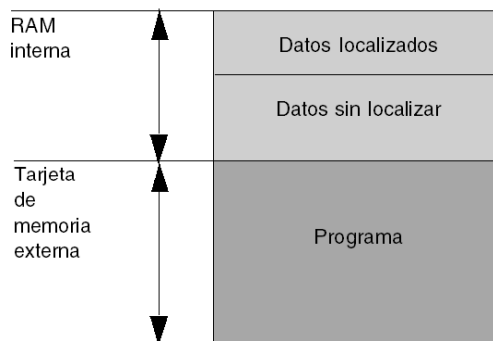
Estructura con tarjeta de ampliación de memoria

Los procesadores Quantum 140 CPU 6••• pueden contar con una tarjeta de ampliación de memoria.

Los datos se almacenan en la memoria RAM interna del módulo del procesador.

El programa se guarda en la tarjeta de memoria de extensión.

El diagrama siguiente describe la estructura de la memoria.



Almacenamiento de la memoria

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

Las tarjetas de memoria RAM están protegidas por una pila de cadmio/níquel.

Arranque con la aplicación guardada en la memoria de almacenamiento

En la tabla siguiente, se describen los diferentes resultados obtenidos según el estado del PLC o el conmutador MEM del PLC (véase *Quantum con Unity Pro, Hardware, Manual de referencia*), al tiempo que se indica si la casilla de ejecución automática está o no seleccionada.

Estado del PLC:	Conmutador MEM del PLC ¹	Ejecución automática en aplic. ²	Resultados
NONCONF	Start u Off	Des	Arranque en frío: la aplicación se carga desde la memoria de almacenamiento a la RAM del PLC. El PLC permanece en posición STOP.
NONCONF	Start u Off	On	Arranque en frío: la aplicación se carga desde la memoria de almacenamiento a la RAM del PLC. El PLC permanece en posición RUN.
NONCONF	Mem Prt o Stop	No aplicable	No se descarga ninguna aplicación. Encendido del PLC en estado NONCONF.
Configurado	Start u Off	Off	Arranque en frío: la aplicación se carga desde la memoria de almacenamiento a la RAM del PLC. El PLC permanece en posición STOP.
1 Start y Stop sólo son válidos para los modelos 434 y 534, y Off sólo es válido para el modelo 311. Mem Prt es válido en todos los modelos.			
2 La ejecución automática de la aplicación hace referencia a la aplicación que se ha cargado.			

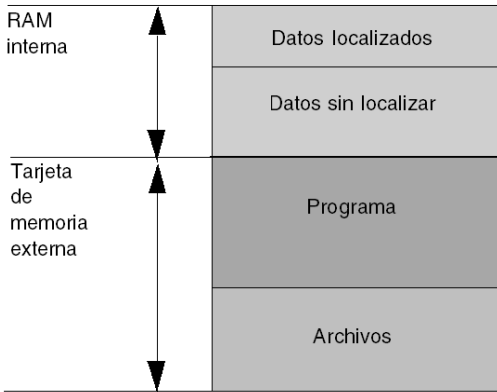
Estado del PLC:	Conmutador MEM del PLC ¹	Ejecución automática en aplic. ²	Resultados
Configurado	Start u Off	On	Arranque en frío: la aplicación se carga desde la memoria de almacenamiento a la RAM del PLC. El PLC permanece en posición RUN.
Configurado	Mem Prt o Stop	Irrelevante	Arranque en caliente: no se descarga ninguna aplicación. El PLC se enciende con el estado anterior.
1 Start y Stop sólo son válidos para los modelos 434 y 534, y Off sólo es válido para el modelo 311. Mem Prt es válido en todos los modelos.			
2 La ejecución automática de la aplicación hace referencia a la aplicación que se ha cargado.			

Especificaciones de las tarjetas de memoria

Existen tres tipos de tarjetas de memoria:

- **De aplicación:** estas tarjetas contienen el programa de la aplicación. Son de tecnología RAM o Flash Eprom
- **De aplicación y almacenamiento de archivos:** estas tarjetas contienen, además del programa, una zona que permite almacenar/restablecer los datos mediante el programa. Son de tecnología RAM o Flash Eprom
- **De almacenamiento de archivos:** estas tarjetas permiten almacenar/restablecer datos mediante el programa. Estas tarjetas son de tecnología SRAM.

El esquema siguiente describe la estructura de memoria con tarjeta de tipo aplicación y almacenamiento de archivos.



NOTA: En el caso de los procesadores con dos emplazamientos para tarjeta de memoria, el emplazamiento inferior está reservado para la función de almacenamiento de archivos.

Descripción detallada de las zonas de memoria

Datos sin localizar

Esta zona contiene los datos sin localizar:

- Datos booleanos y numéricos
- EFB y DFB

Datos localizados

Esta zona contiene los datos localizados (State Ram).

Dirección	Dirección de los objetos	Utilización de los datos
0xxxxx	%Qr.m.c.d,%Mi	Bits de módulo de salidas y bits internos.
1xxxxx	%Ir.m.c.d, %Ii	Bits de módulos de entradas.
3xxxxx	%IW.r.m.c.d, %IWi	Palabras de registro de entrada de los módulos de entradas/salidas.
4xxxxx	%QWr.m.c.d, %MWi	Palabras de salida de los módulos de entradas/salidas y palabras internas.

Programa de usuario

Esta zona contiene los códigos ejecutables de la aplicación.

- Código del programa
- Código asociado a los EF, EFB y a la gestión de los módulos de E/S
- Código asociado a los DFB
- Valores iniciales de las variables

Esta zona contiene también la información necesaria para descargar la aplicación: Códigos gráficos, símbolos, etc.

Sistema operativo

En el caso de los procesadores 140 CPU 31••/41••/51••, esta zona contiene el sistema operativo para el procesamiento de la aplicación. Este sistema operativo se transfiere desde una memoria interna EPROM a la memoria interna RAM durante la conexión.

Almacenamiento de la aplicación

Los procesadores 140 CPU 31••/41••/51•• presentan una zona de memoria Flash EPROM de 1435KB que permite almacenar el programa y los valores iniciales de las variables.

La aplicación que se encuentra en esta zona se transfiere de forma automática a la memoria RAM interna durante la conexión del procesador del autómata (si el conmutador PLC MEM está apagado en la parte frontal del procesador del autómata).

Otra información

También se almacena en la memoria otro tipo de información relativa a la configuración y a la estructura de la aplicación (en la zona de datos o de programa en función del tipo de información).

- Configuración: otros datos relativos a la configuración (configuración de hardware, configuración de software).
- Sistema: datos utilizados por el sistema operativo (pila de las tareas, etc.).
- Diagnóstico: información relativa al diagnóstico del proceso o del sistema, búfer de diagnóstico.

Modalidades de funcionamiento



Objeto

En este capítulo se describen los modos de funcionamiento del autómata en caso de corte y restablecimiento de la corriente, las incidencias en el programa de aplicación y la actualización de las entradas/salidas.

Contenido de este capítulo

Este capítulo contiene las siguientes secciones:

Sección	Apartado	Página
5.1	Modalidades de funcionamiento de los autómatas Modicon M340	132
5.2	Modalidades de funcionamiento de los autómatas Premium, Quantum	146
5.3	Modalidad HALT del autómata	159

5.1 **Modalidades de funcionamiento de los autómatas Modicon M340**

Finalidad de esta sección

En esta sección se describen las modalidades de funcionamiento de los autómatas Modicon M340.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Procesamiento en caso de corte y restablecimiento de la alimentación para los PLC Modicon M340	133
Procesamiento en arranque en frío para PLC Modicon M340	136
Procesamiento del reinicio en caliente en los PLC Modicon M340	141
Inicio automático en modalidad RUN para PLC Modicon M340	145

Procesamiento en caso de corte y restablecimiento de la alimentación para los PLC Modicon M340

General

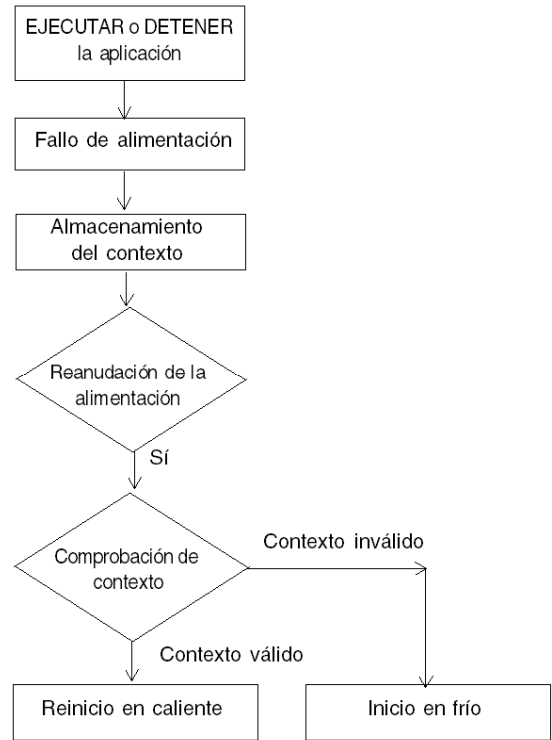
Si la duración del corte es inferior al tiempo de filtrado de la alimentación, el programa no lo ve y se ejecuta normalmente. En caso contrario, se produce una interrupción del programa y se activa el procesamiento de restablecimiento de la alimentación.

Tiempo de filtrado:

PLC	Alimentación alterna	Alimentación continua
BMX CPS 2000 BMX CPS 3500	10 ms	-
BMX CPS 2010 BMX CPS 3020	-	1 ms

Ilustración

La ilustración siguiente muestra las distintas etapas del apagado y encendido.



Operación

La tabla describe las fases del tratamiento de los cortes de alimentación.

Fase	Descripción
1	Cuando se produce un corte de la alimentación, el sistema guarda en la memoria Flash interna el contexto de la aplicación, los valores de las variables de la aplicación y el estado del sistema.

Fase	Descripción
2	El sistema sitúa todas las salidas en estado de retorno (estado definido en la configuración).
3	Cuando se restablece la alimentación, se realizan diversas acciones y comprobaciones para verificar si está disponible el reinicio en caliente: ● Restauración del contexto de la aplicación desde la memoria Flash interna, ● Verificación con la tarjeta de memoria (presencia, disponibilidad de la aplicación), ● Comprobación de que el contexto de la aplicación es idéntico al de la tarjeta de memoria. Si todas las comprobaciones son correctas, se efectuará un reinicio en caliente (<i>véase página 141</i>); de lo contrario, se realizará un arranque en frío (<i>véase página 136</i>).

Procesamiento en arranque en frío para PLC Modicon M340

Causa de un arranque en frío

En la tabla siguiente se describen las diferentes causas de un inicio en frío.

Causas	Características del inicio
Carga de una aplicación	Inicio en frío forzado en STOP
Restaurar la aplicación de la tarjeta de memoria si difiere de la que hay en la memoria RAM interna	Inicio en frío forzado en STOP o en RUN, según la definición de la configuración
Restaurar aplicación de tarjeta de memoria, con comandos de Unity Pro PLC → Backup del proyecto →	Inicio en frío forzado en STOP o en RUN, según la definición de la configuración
Pulsación del botón RESET de la fuente de alimentación	Inicio en frío forzado en STOP o en RUN, según la definición de la configuración
Pulsación del botón RESET de la fuente de alimentación durante menos de 500 ms tras una desconexión	Inicio en frío forzado en STOP o en RUN, según la definición de la configuración
Pulsación del botón RESET de la fuente de alimentación tras un error del procesador, salvo que se trate de un error del watchdog	Inicio en frío forzado en STOP. El inicio en la modalidad RUN, según está definido en la configuración, no se tiene en cuenta
Inicialización desde Unity Pro Forzado del bit de sistema %S0	Inicio en STOP o en RUN (conservando la modalidad operativa en curso en el momento de la desconexión), inicialización únicamente de la aplicación
Restablecimiento después de un corte de alimentación con pérdida del contexto	Inicio en frío forzado en STOP o en RUN, según la definición de la configuración

ATENCIÓN

PÉRDIDA DE DATOS POR LA TRANSFERENCIA DE UNA APLICACIÓN

La carga o transferencia de una aplicación en el PLC normalmente conlleva la inicialización de variables no localizadas.

Para guardar las variables ubicadas:

- Evite la inicialización de `%MWi` desmarcando **Inicializar %MWi con inicio en frío** en la **pantalla de configuración** de la CPU.

Es necesario asignar a los datos una dirección topológica si el proceso requiere conservar los valores actuales de estos datos al transferir la aplicación.

Si no se siguen estas instrucciones pueden producirse lesiones personales o daños en el equipo.

** ATENCIÓN****PÉRDIDA DE DATOS POR LA TRANSFERENCIA DE UNA APLICACIÓN**

No pulse el botón RESET en la fuente de alimentación. En caso contrario, %MWi se restablece y se cargan los valores iniciales.

Si no se siguen estas instrucciones pueden producirse lesiones personales o daños en el equipo.

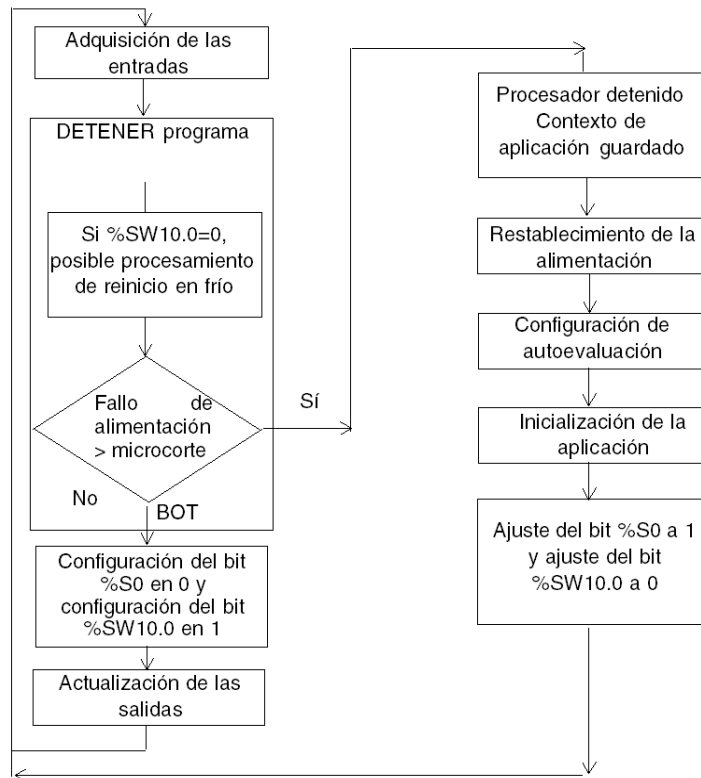
** ATENCIÓN****RIESGO DE PÉRDIDA DE UNA APLICACIÓN**

Si no hay ninguna tarjeta de memoria en el PLC durante un reinicio en frío, se pierde la aplicación.

Si no se siguen estas instrucciones pueden producirse lesiones personales o daños en el equipo.

Figura

El diagrama siguiente describe el funcionamiento de un reinicio en frío.



Operación

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

Fase	Descripción
1	El inicio se efectúa en RUN o en STOP, según el estado del parámetro <i>Inicio automático en RUN</i> definido en la configuración o si éste 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 lo siguiente: ● Desactiva las tareas, que no sean la tarea maestra, hasta que termine el primer ciclo de la tarea maestra. ● Inicializa los datos (bits, imagen de E/S, palabras, etc.) con los valores iniciales definidos en el editor de datos (valor en 0 si no se ha definido ningún valor inicial). Para las palabras %MW, los valores pueden recuperarse en un reinicio en frío, si las dos condiciones son válidas: ● La opción Inicializar %MW en reinicio en frío (véase <i>Unity Pro, Modalidades de funcionamiento</i>,) está desmarcada en la pantalla de configuración del procesador, ● la memoria flash interna tiene una copia de seguridad válida (consulte %SW96 (véase página 203)). Nota: si el número de palabras de %MW supera el tamaño del (consulte la estructura de la memoria de los PLC M340 (véase página 117)) durante la operación de almacenamiento, las palabras restantes se establecen en 0. ● Inicializa los bloques de funciones elementales a partir de los datos iniciales. ● Inicializa los datos declarados en los DFB: en 0 o en el valor inicial declarado en el tipo de DFB. ● Inicializa los bits y palabras de sistema. ● Posiciona los gráficos en los pasos iniciales. ● Cancela los forzados que haya. ● Inicializa las filas de mensajes y de eventos. ● Envía los parámetros de configuración a todos los módulos de entradas/salidas binarias y específicos de la aplicación.
3	En este primer ciclo de reinicio, el sistema efectúa lo siguiente: ● Reinicia la tarea maestra con los bits %S0 (reinicio en frío) y %S13 (primer ciclo en RUN) en la posición 1, la palabra %SW10 (detección de un reinicio en frío en el primer ciclo de una tarea) se pone a 0. ● Pone a 0 los bits %S0 y %S13, y pone a 1 cada bit de la palabra %SW10, cuando finaliza el primer ciclo de la tarea maestra. ● Activa la tarea rápida y los tratamientos de eventos cuando finaliza este primer ciclo de la tarea maestra.

Procesamiento por programa de un inicio en frío

Se recomienda comprobar el bit %SW10.0 para detectar un inicio en frío y comenzar un procesamiento específico para dicho inicio en frío.

NOTA: Se puede probar el bit %S0 si se ha seleccionado el parámetro `Inicio automático` en `RUN`. Si éste no es el caso, el PLC se inicia en `STOP`, el bit %S0 se pone a 1 en el primer ciclo de reinicio, pero el programa no lo detecta debido a que éste no se ejecuta.

Cambios de las salidas

En cuanto se detecta el corte de corriente, las salidas se colocan en posición de retorno:

- pueden tomar el valor de retorno o
- conservar el valor actual,

según la elección efectuada en la configuración.

Cuando se restablece la corriente, las salidas se ponen a cero hasta que la tarea las actualice.

Procesamiento del reinicio en caliente en los PLC Modicon M340

Causa de un reinicio en caliente

Un reinicio en caliente puede estar causado por un restablecimiento de la alimentación sin pérdida de contexto.

ATENCIÓN

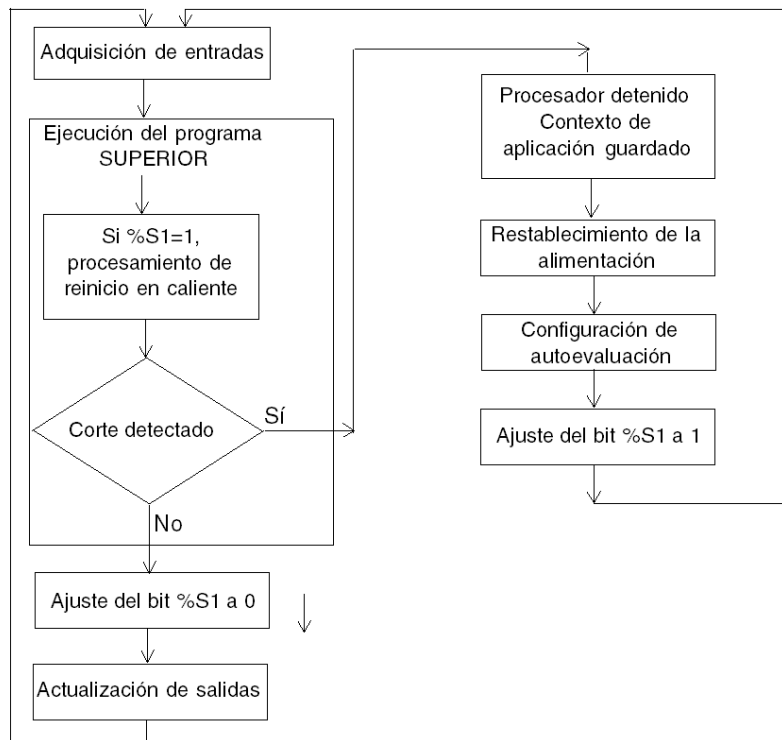
RIESGO DE PÉRDIDA DE UNA APLICACIÓN

Si no hay ninguna tarjeta de memoria en el PLC durante un reinicio en caliente, se pierde la aplicación.

Si no se siguen estas instrucciones pueden producirse lesiones personales o daños en el equipo.

Ilustración

En el esquema siguiente se describe el funcionamiento de un reinicio en caliente.



Funcionamiento

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

Fase	Descripción
1	La ejecución del programa no se reanuda a partir del elemento en el cual ha tenido lugar el corte de corriente. El resto del programa se descarta durante el inicio en caliente. Cada tarea se reiniciará desde el principio.
2	Cuando termina el ciclo de reinicio, el sistema lleva a cabo lo siguiente: ● Restaura el valor de las variables de la aplicación ● Establece el bit %S1 en 1. ● La inicialización de las filas de mensajes y de eventos. ● El envío de parámetros de configuración a todos los módulos de entradas/salidas TON y funciones específicas. ● La desactivación de la tarea rápida y de los tratamientos de eventos (hasta que termine el primer ciclo de la tarea maestra).
3	El sistema lleva a cabo un ciclo de reinicio en el que: ● Reinicia la tarea maestra desde el principio del ciclo. ● Vuelve a establecer en 0 los bits %S1 cuando termina este primer ciclo de la tarea maestro. ● Reactiva la tarea rápida y los tratamientos de eventos cuando finaliza este primer ciclo de la tarea maestra.

Procesamiento por programa de un reinicio en caliente

En caso de reinicio en caliente, si se desea que la aplicación se proteja de una determinada forma, deberá escribirse el programa correspondiente para que pruebe que %S1 está establecido en 1 al inicio del programa de la tarea maestra.

Funciones específicas del inicio en caliente SFC

La CPU no considera el inicio en caliente del PLC M340 como un inicio en caliente de verdad. El intérprete SFC no depende de las tareas.

SFC publica un área de memoria "ws_data" en el SO que contiene datos específicos de la sección SFC que se guardarán cuando se produzca un fallo de alimentación. Al principio del procesamiento de cadenas, los pasos que actualmente están activos se guardan en "ws_data" y el procesamiento se marca como que se indica en la "sección más importante". Al final del procesamiento de cadenas, se desmarca la "sección más importante".

Si se produce un fallo de alimentación en la "sección más importante", éste podría detectarse si este estado está activo al principio (cuando la exploración se interrumpe y la tarea MAST se reinicia desde el principio). En este caso, es posible que el área de trabajo no sea coherente y se restablece a partir de los datos guardados.

Se utiliza información adicional de SFCSTEP_STATE del área de datos localizada para reconstruir el estado del equipo.

Cuando se produce un fallo de alimentación:

- Durante la primera exploración %S1 =1 MAST se ejecuta, pero las tareas FAST y EVENT no se ejecutan.

Cuando se restablece la alimentación:

- Borra las cadenas, anula el registro de los diagnósticos y mantiene las acciones definidas.
- Establece los pasos desde el área guardada.
- Establece los tiempos de paso de SFCSTEP_STATE.
- Restaura el tiempo transcurrido para las acciones temporizadas.

NOTA: El intérprete de SFC es independiente; si la transición es válida, la cadena SFC evoluciona mientras %S1 sea cierto.

Cambios de las salidas

En cuanto se detecta un fallo de alimentación, las salidas se colocan en posición de retorno:

- Pueden tomar el valor de retorno.
- O bien conservar el valor actual.

Según la elección efectuada en la configuración.

Cuando se restablece la alimentación, las salidas están en posición de retorno hasta que las actualiza la tarea.

Inicio automático en modalidad RUN para PLC Modicon M340

Descripción

Inicio automático en RUN es una opción de configuración del procesador. Esta opción fuerza al PLC para que se inicie en la modalidad RUN tras un reinicio en frío (*véase página 136*), salvo cuando se ha cargado una aplicación en este.

En Modicon M340, esta opción no se tiene en cuenta cuando se presiona el botón RESET de alimentación tras un error del procesador, salvo que se trate de un error del watchdog.

PELIGRO

RIESGO DE INICIO DE PROCESO IMPREVISTO

Las acciones siguientes activarán el inicio automático en modalidad RUN:

- La restauración de la aplicación desde la tarjeta de memoria.
- El uso no intencionado o descuidado del botón de reinicio.

Para evitar un reinicio imprevisto en la modalidad RUN, utilice:

- La entrada RUN/STOP en Modicon M340.

Si no se siguen estas instrucciones provocará lesiones graves o incluso la muerte.

5.2 Modalidades de funcionamiento de los autómatas Premium, Quantum

Finalidad de esta sección

En esta sección se describen las modalidades de funcionamiento de los autómatas Premium y Quantum.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Procesamiento en caso de corte y restablecimiento de la alimentación para PLC Premium/Quantum	147
Procesamiento del arranque en frío para PLC Quantum y Premium	150
Procesamiento del reinicio en caliente para PLC Quantum y Premium	155
Inicio automático en modalidad RUN para Premium/Quantum	158

Procesamiento en caso de corte y restablecimiento de la alimentación para PLC Premium/Quantum

General

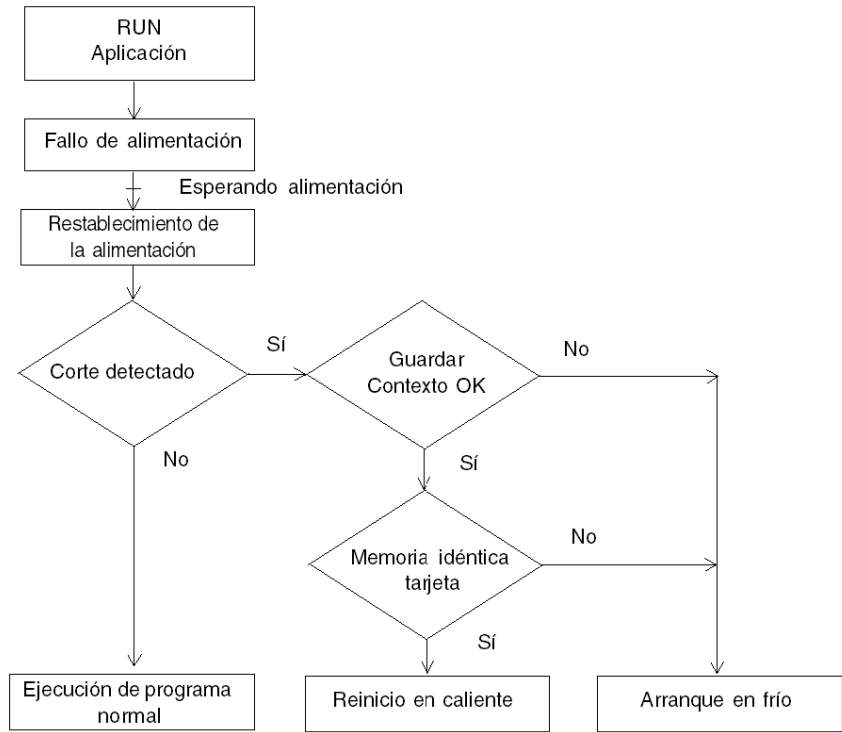
Si la duración del corte es inferior al tiempo de filtrado de la alimentación, el programa no lo ve y se ejecuta normalmente. En caso contrario, se produce una interrupción del programa y el tratamiento del restablecimiento de la alimentación.

Tiempo de filtrado:

PLC	Alimentación alterna	Alimentación continua
Premium	10 ms	1 ms
Atrium	30 ms	-
Quantum	10 ms	1 ms

Ilustración

La figura presenta los diferentes restablecimientos de alimentación detectados por el sistema.



Funcionamiento

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

Fase	Descripción
1	En el momento del corte de la alimentación, el sistema almacena el contexto de la aplicación y la hora del corte.

Fase	Descripción
2	Sitúa todas las salidas en estado de reactivación (estado definido en la configuración).
3	Cuando se restablece la alimentación, 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 re arranque sin inicialización de los datos: re arranque en caliente

Corte de la alimentación en un bastidor distinto del bastidor 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 al tiempo de filtrado, el programa no lo ve y se ejecuta normalmente.

Procesamiento del arranque en frío para PLC Quantum y Premium

Causa de un arranque en frío

En la tabla siguiente se describen las diferentes causas de un inicio en frío.

Causas	Características del inicio
Carga de una aplicación	Inicio en frío forzado en STOP
Pulsación del botón RESET del procesador (Premium)	Inicio en frío forzado en STOP o en RUN, según la definición de la configuración
Pulsación del botón RESET del procesador después de un fallo del procesador o del sistema (Premium).	Inicio en frío forzado en STOP
Manipulación de la tapa prensil o inserción/extracción de una tarjeta de memoria PCMCIA	Inicio en frío forzado en STOP o en RUN, según la definición de la configuración
Inicialización desde Unity Pro Forzado del bit de sistema %S0	Inicio en STOP o en RUN (conserva la modalidad de funcionamiento en curso) sin inicialización de los módulos de entradas/salidas TON ni de función específica
Restablecimiento después de un corte de alimentación con pérdida del contexto	Inicio en frío forzado en STOP o en RUN, según la definición de la configuración

ATENCIÓN

PÉRDIDA DE DATOS POR LA TRANSFERENCIA DE UNA APLICACIÓN

La carga o transferencia de una aplicación en el PLC normalmente conlleva la inicialización de variables no localizadas.

Para guardar variables ubicadas con PLC Premium y Quantum:

- Guardar y restaurar %M y %MW haciendo clic en **PLC → Transferencia de datos**.

Para PLC Premium:

- Evite la inicialización de %MW desmarcando **Inicializar %MWi con inicio en frío** en la **pantalla de configuración** de la CPU.

Para PLC Quantum:

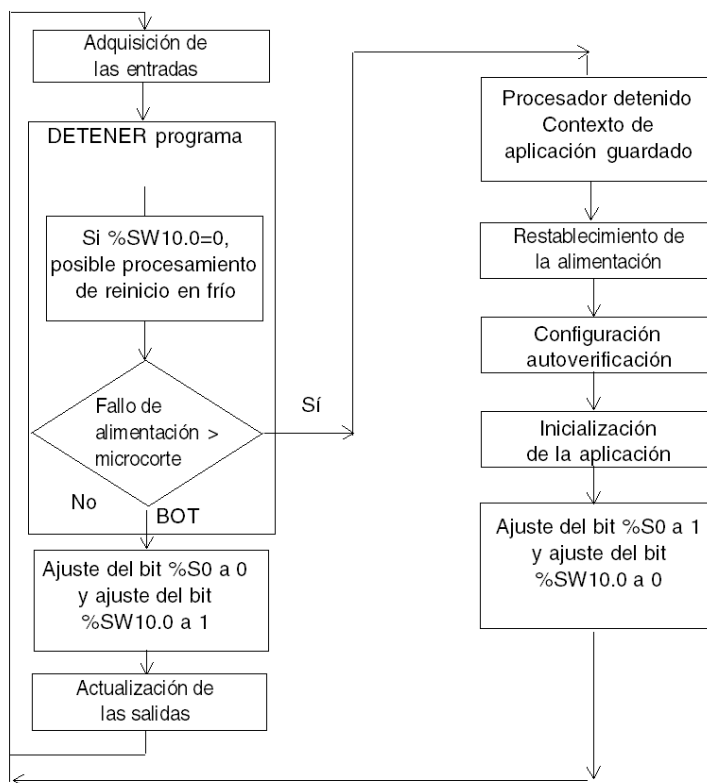
- Evite la inicialización de %MW desmarcando **Restablecer %MWi** en la **pantalla de configuración** de la CPU.

Es necesario asignar a los datos una dirección topológica si el proceso requiere conservar los valores actuales de estos datos al transferir la aplicación.

Si no se siguen estas instrucciones pueden producirse lesiones personales o daños en el equipo.

Figura

El diagrama siguiente describe el funcionamiento de un reinicio en frío.



Operación

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

Fase	Descripción
1	El inicio se efectúa en RUN o en STOP según el estado del parámetro <i>Inicio automático en RUN</i> definido en la configuración o, si éste 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 lo siguiente: ● La inicialización de datos (bits, imagen de E/S, palabras, etc.) con los valores iniciales definidos en el editor de datos (valor en 0 si no se ha definido ningún valor inicial). En el caso de las palabras %MW, estos valores pueden conservarse en un arranque en frío si no se ha seleccionado la opción de restablecimiento de %MW en caso de reinicio en frío en la pantalla de configuración del procesador. ● La inicialización de los bloques de funciones elementales a partir de los datos iniciales. ● La inicialización de los datos declarados en los DFB: en 0 o en el valor inicial declarado en el tipo de DFB. ● La inicialización de los bits y palabras de sistema. ● La desactivación de las tareas, que no sean la tarea maestra, hasta que termine el primer ciclo de la tarea maestra. ● El posicionamiento de los gráficos en las etapas iniciales. ● La cancelación de forzados. ● La inicialización de las filas de mensaje y de eventos. ● El envío de parámetros de configuración a todos los módulos de entradas/salidas binarias y módulos de función específica.
3	En este primer ciclo de reinicio, el sistema efectúa lo siguiente: ● Reinicia la tarea maestra con los bits %S0 (reinicio en frío) y %S13 (primer ciclo en RUN) en la posición 1; la palabra %SW10 (detección de un reinicio en frío en el primer ciclo de una tarea) se pone a 0. ● Restablece a 0 los bits %S0 y %S13, y pone a 1 cada bit de la palabra %SW10, cuando finaliza el primer ciclo de la tarea maestra. ● Activa la tarea rápida y los procesamientos de eventos cuando finaliza este primer ciclo de la tarea maestra.

Procesamiento por programa de un inicio en frío

Se recomienda comprobar el bit %SW10.0 para detectar un inicio en frío y comenzar un procesamiento específico para dicho inicio en frío.

NOTA: Es posible probar el bit %S0, si el parámetro *Inicio automático en RUN* se ha seleccionado. Si éste no es el caso, el PLC se inicia en STOP, el bit %S0 se pone a 1 en el primer ciclo de reinicio, pero el programa no lo detecta debido a que éste no se ejecuta.

Evolución de las salidas, caso de Premium y Atrium

En cuanto se detecta el corte de corriente, las salidas se colocan en posición de retorno:

- pueden tomar el valor de retorno o
- conservar el valor actual,

según la elección efectuada en la configuración.

Cuando se restablece la corriente, las salidas se ponen a cero hasta que la tarea las actualice.

Evolución de las salidas, caso de Quantum.

En cuanto se detecta el corte de corriente,

- las salidas locales se ponen a cero y
- las salidas de los bastidores de extensión descentralizados o distribuidos pasan a la posición de retorno.

Cuando se restablece la corriente, las salidas se ponen a cero hasta que la tarea las actualice.

ATENCIÓN

El comportamiento de las salidas forzadas se ha modificado entre Modsoft/NxT/Concept y Unity Pro.

Con Modsoft/NxT/Concept, no es posible forzar las salidas si el interruptor de protección de la memoria del procesador Quantum está en posición "CON".

Con Unity Pro, es posible forzar las salidas si el interruptor de protección de la memoria del procesador Quantum está en posición "CON".

Con Modsoft/NxT/Concept, las salidas forzadas permanecen en el estado correspondiente tras un reinicio en frío.

Con Unity Pro, las salidas forzadas pierden el estado correspondiente tras un inicio en frío.

Si no se siguen estas instrucciones pueden producirse lesiones personales o daños en el equipo.

140 CPU 31••/41••/51•• de Quantum

Estos procesadores cuentan con una zona de memoria Flash EPROM de 1.435 KB que permite guardar el programa y los valores iniciales de las variables.

Cuando se restablece la corriente, se puede seleccionar la modalidad de funcionamiento deseado mediante el conmutador PLC MEM en la parte delantera del procesador del PLC. Consulte la información detallada sobre el funcionamiento de este conmutador en el manual de Quantum (*véase Quantum con Unity Pro, Hardware, Manual de referencia*).

- **Posición Des:** La aplicación que se encuentra en esta zona se transfiere de forma automática a la memoria RAM interna tras la conexión a la alimentación del procesador del PLC: reinicio en frío de la aplicación.

Posición Con: La aplicación que se encuentra en esta zona no se transfiere a la memoria RAM interna: reinicio en caliente de la aplicación.

Procesamiento del reinicio en caliente para PLC Quantum y Premium

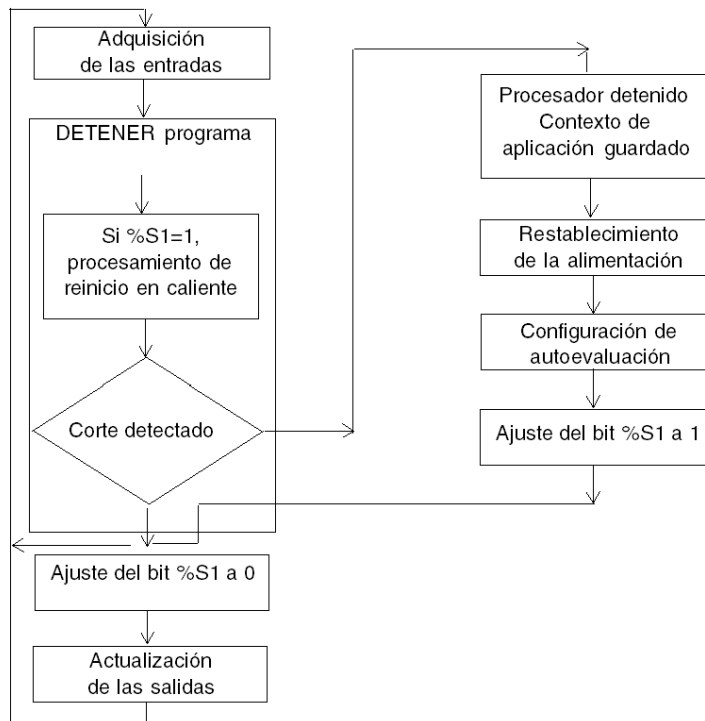
Causa de un reinicio en caliente

Un reinicio en caliente puede haber sido provocado:

- por un restablecimiento de la alimentación sin pérdida de contexto,
- por una puesta a 1 por parte del programa del bit del sistema %S1,
- por Unity Pro desde el terminal o
- mediante el botón RESET del módulo de fuente de alimentación del bastidor 0 (en PLC Premium).

Ilustración

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



Operación

En la tabla siguiente se describen las fases de reinicio de la ejecución del programa en el reinicio en caliente.

Fase	Descripción
1	La ejecución del programa se reanuda a partir del elemento en el cual ha tenido lugar el corte de corriente, sin actualización de las salidas.
2	Cuando termina el ciclo de reinicio, el sistema realiza: ● La inicialización de las filas de mensaje y de eventos. ● El envío de parámetros de configuración a todos los módulos de entradas/salidas binarias y funciones específicas. ● La desactivación de la tarea rápida y del procesamiento de eventos (hasta que termine el primer ciclo de la tarea maestra).
3	El sistema lleva a cabo un ciclo de reinicio en el que: ● Vuelve a confirmar todos los módulos de entradas. ● Reinicia la tarea maestra con el bit %S1 (reinicio en caliente) puesto a 1. ● Vuelve a poner en estado 0 el bit %S1 cuando termina este primer ciclo de la tarea maestra. ● Reactiva la tarea rápida, las tareas auxiliares y el procesamiento de eventos cuando finaliza este primer ciclo de la tarea maestra.

Procesamiento por programa de un reinicio en caliente

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

En autómatas Quantum, el conmutador colocado en la parte delantera del procesador permite configurar las modalidades operativas. Si desea más información, consulte la documentación de Quantum (*véase Quantum con Unity Pro, Hardware, Manual de referencia*).

Evolución de las salidas, caso de Premium y Atrium

En cuanto se detecta un fallo de alimentación, las salidas se colocan en posición de retorno:

- pueden tomar el valor de retorno o
- conservar el valor actual,

según la elección efectuada en la configuración.

Cuando se restablece la alimentación, las salidas están en posición de retorno hasta que las actualiza la tarea.

Evolución de las salidas, caso de Quantum

En cuanto se detecta el corte de corriente,

- las salidas locales se ponen a cero y
- las salidas de los bastidores de extensión descentralizados o distribuidos pasan a la posición de retorno.

Cuando se restablece la alimentación, las salidas están en posición de retorno hasta que las actualiza la tarea.

Evolución de las salidas, caso de bastidores de ampliación

Si hay un corte de corriente en el bastidor donde se encuentra la CPU

- Estado de retorno en cuanto se detecta pérdida de CPU
- Estado de seguridad durante la configuración de E/S
- Estado calculado por la CPU después de la primera ejecución de la tarea que ha provocado este corte

Cuando se restablece la alimentación, las salidas están en posición de retorno hasta que las actualiza la tarea.

Inicio automático en modalidad RUN para Premium/Quantum

Descripción

Inicio automático en RUN es una opción de configuración del procesador. Esta opción fuerza al PLC para que se inicie en la modalidad RUN tras un reinicio en frío (véase *página 150*), salvo cuando se ha cargado una aplicación en este.

En PLC Quantum, el inicio automático en modalidad RUN también depende de la posición del interruptor en el panel frontal del procesador. Para obtener más información, consulte la documentación de Quantum (véase *Quantum con Unity Pro, Hardware, Manual de referencia*).

PELIGRO

RIESGO DE INICIO DE PROCESO IMPREVISTO

Las acciones siguientes activarán la opción “Inicio automático en modalidad RUN”:

- Si introduce la tarjeta PCMCIA cuando el PLC está encendido (Premium, Quantum).
- Si sustituye el procesador mientras está encendido (Premium, Quantum).
- El uso no intencionado o descuidado del botón de reinicio.
- Si la batería resulta ser defectuosa en caso de un corte de alimentación (Premium, Quantum).

Para evitar un reinicio imprevisto en la modalidad RUN:

- Se recomienda encarecidamente que utilice la entrada RUN/STOP en los PLC Premium, o bien, el interruptor ubicado en la parte frontal del panel del procesador para PLC Quantum.
- Igualmente, se recomienda encarecidamente **no** utilizar entradas memorizadas como entrada RUN/STOP del PLC.

Si no se siguen estas instrucciones provocará lesiones graves o incluso la muerte.

5.3 Modalidad HALT del autómeta

Modalidad HALT del PLC

Presentación

El autómeta pasa a modalidad HALT en los casos siguientes:

- Uso de la instrucción HALT
- Desborde del watchdog
- Error de ejecución del programa (división por cero, desborde...) si el bit %S78 (*véase página 176*) se establece en 1.

Precauciones

Atención: Cuando el autómeta está en la modalidad HALT, todas las tareas se detienen (*véase Unity Pro, Modalidades de funcionamiento, .*). Es necesario comprobar el comportamiento de las E/S asociadas.

Objetos de sistema



Objeto

Este capítulo describe los bits y las palabras de sistema del lenguaje Unity Pro.

Nota: los símbolos asociados a cada objeto de bit o de palabra del sistema a los que se hace referencia en las tablas descriptivas de dichos objetos no están incluidos de serie en el programa, se pueden introducir a través del editor de datos.

Se proponen para homogeneizar su denominación en las diferentes aplicaciones.

Contenido de este capítulo

Este capítulo contiene las siguientes secciones:

Sección	Apartado	Página
6.1	Bits de sistema	162
6.2	Palabras de sistema	187
6.3	Palabras de sistema específicas de Atrium/Premium	217
6.4	Palabras de sistema específicas de Quantum	231
6.5	Palabras de sistema específicas de Modicon M340	248

6.1 Bits de sistema

Objeto

Este capítulo describe los bits del sistema.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Introducción de bits de sistema	163
Descripción de los bits de sistema %S0 a %S7	164
Descripción de los bits de sistema %S9 a %S13	167
Descripción de los bits de sistema de %S15 a %S21	169
Descripción de los bits de sistema de %S30 a %S123	172
Descripción de los bits de sistema de %S60 a %S79	176
Descripción de los bits de sistema de %S80 a %S96	181
Descripción de los bits de sistema de %S100 a %S122	185

Introducción de bits de sistema

General

Los autómatas Modicon M340, Premium, Atrium y Quantum utilizan bits de sistema %Si que indican los estados del autómata o que permiten controlar el funcionamiento de éste.

Dichos bits pueden probarse en el programa del usuario con el fin de detectar cualquier evolución de funcionamiento que conlleve un procedimiento de procesamiento establecido.

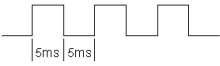
Algunos de estos bits deben volver a su estado inicial o normal por programa. No obstante, los bits de sistema que vuelven a su estado inicial o normal a través del sistema, no deben hacerlo a través del programa ni del terminal

Descripción de los bits de sistema %S0 a %S7

Descripción detallada

Descripción de los bits de sistema %S0 a %S7:

Bit Símbolo	Función	Descripción	Estado inicial	Aplicac ión Modico n M340	Premiu m Atrium	Quantu m
%S0 COLDSTAR T	Arranque en frío	Normalmente en 0, puede definir este bit a 1 mediante: ● el restablecimiento de la alimentación con pérdida de datos (fallo de la batería) ● el programa de usuario ● el terminal ● un cambio de cartucho Este bit se define en 1 durante el primer ciclo completo de restauración del PLC en modalidad RUN o STOP. El sistema lo restablece en 0 antes del ciclo siguiente. Para detectar el primer ciclo que se está ejecutando después de un arranque en frío, consulte %SW10. En la modalidad de seguridad, este bit no está disponible en PLC de seguridad Quantum. %S0 no se define siempre durante la primera exploración del PLC. Si es necesario establecer una señal para cada inicio del PLC, se deberá utilizar %S21 en su lugar. Para Premium y Quantum, Procesamiento del arranque en frío para PLC Quantum y Premium (véase página 152) Para Modicon M340, Procesamiento del arranque en frío para PLC Modicon M340 (véase página 139)	1 (1 ciclo)	Sí	Sí	Sí
%S1 WARMSTAR T	Reinicio en caliente	Normalmente en 0, puede definir este bit a 1 mediante: ● el restablecimiento de la alimentación con datos guardados ● el programa de usuario ● el terminal El sistema lo restablece a 0 al final del primer ciclo completo y antes de actualizar las salidas. Este bit no está disponible en PLC de seguridad Quantum. %S1 no se define siempre durante la primera exploración del PLC. Si es necesario establecer una señal para cada inicio del PLC, se deberá utilizar %S21 en su lugar.	0	Sí	Sí	Sí (excepto para PLC de seguri dad)

Bit Símbolo	Función	Descripción	Estado inicial	Aplicac ión Modico n M340	Premiu m Atrium	Quantu m
%S4 TB10MS	Base de tiempo 10 ms	Un temporizador interno regula el cambio de estado de este bit. Es asíncrono en relación con el ciclo del PLC. Gráfico:  Este bit no está disponible en PLC de seguridad Quantum.	-	Sí	Sí	Sí (excepto para PLC de seguri dad)
%S5 TB100MS	Base de tiempo 100 ms	Ídem %S4	-	Sí	Sí	Sí (excepto para PLC de seguri dad)
%S6 TB1SEC	Base de tiempo 1 s	Ídem %S4	-	Sí	Sí	Sí (excepto para PLC de seguri dad)
%S7 TB1MIN	Base de tiempo 1 min	Ídem %S4	-	Sí	Sí	Sí (excepto para PLC de seguri dad)

Descripción de los bits de sistema %S9 a %S13

Descripción detallada

Descripción de los bits de sistema %S9 a %S13:

Bit Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%S9 OUTDIS	Sitúa en posición de retorno las salidas de todos los buses.	Normalmente está en estado 0, y el programa o el terminal establecen este bit en 1: ● Establecido en 1: establece el bit a 0 o mantiene el valor actual según la configuración elegida (bus X, Fipio, AS-i, etc.). ● Establecido en 0: las salidas se actualizan normalmente. Nota: en Modicon M340, las entradas/salidas distribuidas mediante el bus CANopen no se ven afectadas por el bit %S9. Nota: el bit de sistema actúa directamente en las salidas físicas y no en los bits de imagen de las salidas. Nota: el explorador de E/S Ethernet no se ve afectado por el bit %S9.	0	Sí (1)	Sí	NO
%S10 IOERR	Fallo de entrada/salida	Normalmente en estado 1, el sistema vuelve a establecerlo en 0 cuando falla un módulo de E/S en bastidor o cuando se detecta un dispositivo en Fipio (por ejemplo, una configuración incompatible o fallos de intercambio o de hardware, etc.). El sistema vuelve a establecer el bit %S10 en 1 en el momento en que desaparece el fallo.	1	Sí	Sí	Sí

ATENCIÓN

%S10 para PLC Quantum

En Quantum, los errores de comunicación de los módulos (NOM, NOE, NWM, CRA, CRP) y de los módulos MMS no se comunican en los bits %S10 y %S16.

Es responsabilidad del usuario utilizar estos bits de sistema de forma correcta.

Si no se siguen estas instrucciones pueden producirse lesiones personales o daños en el equipo.

Bit Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%S11 WDG	Desborde de watchdog	Normalmente está en estado 0, y el sistema establece este bit en 1 tan pronto como el tiempo de ejecución de la tarea sobrepasa el tiempo de ejecución máximo (es decir, el watchdog) establecido en las propiedades de tarea.	0	SÍ	SÍ	SÍ
%S12 PLCRUNNING	PLC en modalidad RUN	El sistema establece este bit en 1 cuando el PLC está en modalidad RUN. El sistema lo establece en 0 tan pronto como el PLC no está en modalidad RUN (STOP, INIT, etc.).	0	SÍ	SÍ	SÍ
%S13 1RSTSCANRUN	Primer ciclo después de la puesta en RUN	La conmutación del PLC de la modalidad STOP a RUN (incluso después de un arranque en frío con arranque automático en ejecución) se indica al establecer el bit de sistema %S13 en 1. Este bit vuelve a ponerse a 0 al final del primer ciclo de la tarea MAST en la modalidad de ejecución.	-	SÍ	SÍ	SÍ

Descripción de los bits de sistema de %S15 a %S21

Descripción detallada

Descripción de los bits de sistema de %S15 a %S21:

Bit Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%S15 STRINGERROR	Fallo de cadena de caracteres	Normalmente en estado 0, este bit pasa al estado 1 cuando el área de destino de una transferencia de cadena de caracteres no tiene el tamaño suficiente (incluido el número de caracteres y el carácter de fin de cadena de caracteres) para recibirla. La aplicación se detiene debido a un error si el bit %S78 se ha establecido en 1. La aplicación debe volver a poner el bit en 0. Este bit no está disponible en PLC de seguridad Quantum.	0	SÍ	SÍ	SÍ (excepto para PLC de seguridad)
%S16 IOERRTSK	Fallo de salidas/entradas de tarea	Normalmente en estado 1, el sistema vuelve a establecer este bit en 0 cuando falla un módulo de E/S en bastidor o en un dispositivo Fipio configurado en la tarea. El usuario debe volver a establecer el bit en 1.	1	SÍ	SÍ	SÍ

ATENCIÓN

%S16 para PLC Quantum

En Quantum, los errores de comunicación de los módulos (NOM, NOE, NWM, CRA, CRP) y de los módulos MMS no se comunican en los bits %S10 y %S16.

Es responsabilidad del usuario utilizar estos bits de sistema de forma correcta.

Si no se siguen estas instrucciones pueden producirse lesiones personales o daños en el equipo.

Bit Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%S17 CARRY	Salida de desplazamiento circular	Normalmente está en estado 0. Durante una operación de desplazamiento circular, este bit adopta el estado del bit saliente.	0	Sí	Sí	Sí
%S18 OVERFLOW	Desborde o error aritmético	Normalmente en estado 0, este bit pasa a 1 en caso de desborde de la capacidad si: ● el resultado es superior a +32.767 o inferior a -32.768, en longitud simple; ● el resultado es superior a +65.535, en un número entero sin signo; ● el resultado es superior a +2.147.483.647 o inferior a -2.147.483.648, en longitud doble; ● el resultado es superior a +4.294.967.296, en un número entero en longitud doble sin signo; ● los valores reales sobrepasan los límites; ● hay una división entre 0; ● hay una raíz de un número negativo; ● se fuerza un paso inexistente en un programador cíclico; ● hay un apilamiento de un registro completo, vaciado de un registro vacío. Sólo hay un caso en el que los PLC Modicon M340 no aumentan el bit %S18 cuando los valores reales superan los límites. Esto sólo sucede si se utilizan operandos no normalizados o algunas operaciones que generan resultados no normalizados (transgresión por debajo de rango gradual). Debe comprobarse mediante el programa del usuario después de cada operación en la que exista riesgo de desborde; si es el caso, el usuario debe volver a ponerlo a 0. Cuando el bit %S18 pasa a 1, la aplicación se detiene debido a un error si el bit %S78 se ha establecido en 1.	0	Sí	Sí	Sí

Bit Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%S19 OVERRUN	Desborde del período de tarea (exploración periódica)	Normalmente en estado 0, el sistema pone este bit en estado 1 en caso de desborde del período de ejecución (tiempo de ejecución de tarea superior al período definido por el usuario en la configuración o programado en la palabra %SW asociada a la tarea). El usuario debe volver a establecer el bit en 0. Cada tarea gestiona su propio bit %S19.	0	Sí	Sí	Sí
%S20 INDEXOVF	Desborde del índice	Normalmente en estado 0, este bit pasa a estado 1 cuando la dirección del objeto de índice sea menor que 0 o supere el número de objetos declarados en la configuración. En este caso, ocurre lo mismo que si el índice fuera igual a 0. Debe comprobarse mediante el programa del usuario después de cada operación en la que exista riesgo de desborde; si es el caso, vuelve a 0. Cuando el bit %S20 pasa a 1, la aplicación se detiene debido a un error si el bit %S78 se ha establecido en 1. Este bit no está disponible en PLC de seguridad Quantum.	0	Sí	Sí	Sí (excepto para PLC de seguridad)
%S21 1RSTTASKRUN	Primer ciclo de tarea	El bit %S21, que se comprueba en una tarea (Mast, Fast, Aux0, Aux1, Aux2 o Aux3), indica el primer ciclo de dicha tarea, incluso después de un arranque en frío con arranque automático en ejecución y un arranque en caliente. %S21 se pone a 1 al comienzo del ciclo y a 0 al final del ciclo. Nota: Es importante saber que el bit %S21 no tiene el mismo significado en PL7 y en Unity Pro.	0	Sí	Sí	Sí

Descripción de los bits de sistema de %S30 a %S123

Descripción detallada

Descripción de los bits de sistema de %S30 a %S123:

Bit Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%S30 MASTACT	Activación/d esactivación de la tarea maestra	Normalmente puesta a 1. La tarea maestra se desactiva cuando el usuario pone el bit a 0. El sistema considera este bit al final de cada ciclo de tarea MAST. Este bit no está disponible en PLC de seguridad Quantum.	1	Sí	Sí	Sí (excepto para PLC de seguridad)
%S31 FASTACT	Activación/d esactivación de la tarea rápida	Normalmente se define en 1 cuando el usuario crea la tarea. La tarea se desactiva cuando el usuario define el bit en 0. Este bit no está disponible en PLC de seguridad Quantum.	1	Sí	Sí	Sí (excepto para PLC de seguridad)
%S32 AUX0ACT	Activación/d esactivación de la tarea auxiliar 0	Normalmente se define en 1 cuando el usuario crea la tarea. La tarea auxiliar se desactiva cuando el usuario define el bit en 0. Este bit no está disponible en PLC de seguridad Quantum.	0	NO	Sí	Sí (excepto para PLC de seguridad)
%S33 AUX1ACT	Activación/d esactivación de la tarea auxiliar 1	Normalmente se define en 1 cuando el usuario crea la tarea. La tarea auxiliar se desactiva cuando el usuario define el bit en 0. Este bit no está disponible en PLC de seguridad Quantum.	0	NO	Sí	Sí (excepto para PLC de seguridad)
%S34 AUX2ACT	Activación/d esactivación de la tarea auxiliar 2	Normalmente se define en 1 cuando el usuario crea la tarea. La tarea auxiliar se desactiva cuando el usuario define el bit en 0. Este bit no está disponible en PLC de seguridad Quantum.	0	NO	Sí	Sí (excepto para PLC de seguridad)

Bit Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%S35 AUX3ACT	Activación/desactivación de la tarea auxiliar 3	Normalmente se define en 1 cuando el usuario crea la tarea. La tarea auxiliar se desactiva cuando el usuario define el bit en 0. Este bit no está disponible en PLC de seguridad Quantum.	0	NO	Sí	Sí (excepto para PLC de seguridad)
%S38 ACTIVEVT	Habilitación/inhibición de eventos	Normalmente puesta a 1. Los eventos se inhiben cuando el usuario pone el bit a 0. Este bit no está disponible en PLC de seguridad Quantum.	1	Sí	Sí	Sí (excepto para PLC de seguridad)
%S39 EVTOVR	Saturación durante el procesamiento de eventos	El sistema define este bit en 1 para indicar que no se pueden procesar uno o más eventos tras la saturación de las filas. El usuario debe restablecer este bit en 0. Este bit no está disponible en PLC de seguridad Quantum.	0	Sí	Sí	Sí (excepto para PLC de seguridad)
%S40 RACK0ERR	Fallo de entrada/salida del bastidor 0	El bit %S40 se asigna al bastidor 0. Normalmente en 1, el sistema define este bit en 0 cuando aparece un fallo en la entrada/salida del bastidor. En este caso: - El bit %S10 se define en 0. - El indicador de E/S se enciende. - El bit del módulo %I.r.m.c.Err se define en 1. Este bit se restablece en 1 cuando desaparece el fallo.	1	Sí	Sí	NO
%S41 RACK1ERR	Fallo de entrada/salida del bastidor 1	Igual que %S40 para el bastidor 1.	1	Sí	Sí	NO
%S42 RACK2ERR	Fallo de entrada/salida del bastidor 2	Igual que %S40 para el bastidor 2.	1	Sí	Sí	NO

Bit Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%S43 RACK3ERR	Fallo de entrada/salida del bastidor 3	Igual que %S40 para el bastidor 3.	1	SÍ	SÍ	NO
%S44 RACK4ERR	Fallo de entrada/salida del bastidor 4	Igual que %S40 para el bastidor 4.	1	SÍ	SÍ	NO
%S45 RACK5ERR	Fallo de entrada/salida del bastidor 5	Igual que %S40 para el bastidor 5.	1	SÍ	SÍ	NO
%S46 RACK6ERR	Fallo de entrada/salida del bastidor 6	Igual que %S40 para el bastidor 6.	1	SÍ	SÍ	NO
%S47 RACK7ERR	Fallo de entrada/salida del bastidor 7	Igual que %S40 para el bastidor 7.	1	SÍ	SÍ	NO
%S50 RTCWRITE	Actualización de hora y fecha mediante palabras de %SW50 a %SW53	Normalmente definida en 0, el programa o el terminal definen este bit en 1. ● Establecido en 0: actualice las palabras de sistema de %SW50 a %SW53 por la fecha y hora especificadas en el reloj de tiempo real del PLC. ● Establecido en 1: las palabras de sistema %SW50 a %SW53 ya no se actualizan, por lo que puede modificarlas. ● El cambio de 1 a 0 permite actualizar el reloj de tiempo real con los valores introducidos en palabras de %SW50 a %SW53.	0	SÍ	SÍ	SÍ
%S51 RTCERR	Pérdida de tiempo en el reloj de tiempo real	Este bit gestionado por el sistema definido en 1 indica que falta el reloj de tiempo real o que las palabras de sistema (de %SW50 a %SW53) no tienen significado. En este caso debe poner el reloj en la hora correcta.	–	SÍ	SÍ	SÍ

Bit Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%S59 RTCTUNING	Actualización incremental de hora y fecha mediante la palabra %SW59	Normalmente en estado 0, el programa o el terminal pueden definir este bit en 1 o 0. ● Establecido en 0: el sistema no gestiona la palabra %SW59. ● Establecido en 1: el sistema gestiona los bordes de la palabra %SW59 para ajustar la fecha y hora actual (por incrementos).	0	Sí	Sí	Sí
%S123	Bit de sistema	Este bit lo utiliza el sistema y no puede utilizarlo la aplicación de usuario.	—	NO	NO	NO

Descripción de los bits de sistema de %S60 a %S79

Descripción detallada

Descripción de los bits de sistema de %S60 a %S79:

Bit Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%S60 BACKUPCHVOV	Comando de conmutación voluntaria	Este bit controla la conmutación voluntaria en caso de puesta en marcha de una arquitectura redundante. El usuario o la aplicación pueden volver a poner el bit a 0. Sólo disponible para Premium. De forma predeterminada, este bit está a 0; si este bit está puesto a 1, no pasa nada, no se produce ninguna conmutación.	0	NO	SÍ	NO
%S65 CARDIS	Deshabilitar tarjeta	Es necesario generar un flanco ascendente en el bit %S65 antes de extraer la tarjeta para asegurar la coherencia de la información. De hecho, en la detección de flanco ascendente, terminarán todos los accesos actuales (lectura y escritura de archivos, almacenamiento de la aplicación) y el LED de acceso de la tarjeta se apaga (la luz CARDERR no varía). A continuación, se puede extraer la tarjeta; el LED CARDERR está encendido. Inserción de la tarjeta: el LED de acceso está encendido y el LED CARDERR muestra el estado (%S65 no varía). El usuario tendrá que restablecer %S65 a 0 para que se realice la detección de flancos posteriormente. Si se ha generado un flanco ascendente en el bit %S65 y no se ha extraído la tarjeta, el restablecimiento a 0 del bit no permite el acceso a la tarjeta.	0	SÍ	NO	NO

Bit Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%S66 LEDBATT APPLIBCK	Copia de seguridad de la aplicación	El usuario pone este bit a 1 para iniciar una operación de copia de seguridad (transferencia de la aplicación de la memoria RAM a la tarjeta). El sistema detectará el borde ascendente para iniciar la operación de copia de seguridad. El sistema lee el estado de este bit cada segundo. La operación de copia de seguridad sólo se produce si la aplicación en la memoria RAM es distinta de la que hay en la tarjeta. Una vez finalizada la operación de copia de seguridad, el sistema pone este bit a 0. Advertencia: antes de realizar una nueva operación de copia de seguridad poniendo el bit %S66 a 1, debe probar que el sistema ha puesto el bit %S66 a 0 (lo que significa que la operación de copia de seguridad anterior ha terminado). Nunca utilice %S66 si está puesto a 1. Esto puede provocar una pérdida de datos. El bit %S66 resulta especialmente útil cuando se han sustituido el valor de inicialización %S94 y los parámetros guardados.	0	Sí	NO	NO

Bit Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%S67 PCMCIABAT0	Estado de la batería de la tarjeta de memoria de la aplicación	Este bit permite supervisar el estado de la batería principal cuando la tarjeta de memoria se encuentra en la ranura PCMCIA superior. Esto afecta a los PLC Atrium, Premium y Quantum (CPU 140 CPU 671 60/60S, 140 CPU 651 60/60S, 140 CPU 652 60 y 140 CPU 651 50): ● Establecido en 1: la batería principal de tensión es baja. Se mantiene la aplicación, aunque es necesario sustituir la batería según el procedimiento de mantenimiento predictivo (véase <i>Premium y Atrium con Unity Pro, Procesadores, bastidores y módulos de alimentación, Manual de instalación</i>). ● Establecido en 0: la batería principal de tensión es suficiente (siempre se mantiene la aplicación). ● Bit %S67 compatible con Unity versión ≥ 2.02. NOTA: Con PCMCIA "azules" (PV ≥ 04), el bit %S67 no se establece en 1 cuando la batería principal está ausente, mientras que con PCMCIA "verdes" (PV < 04), el bit %S67 se establece en 1 en la misma condición.	-	NO	Sí	Sí
%S68 PLCBAT	Estado de la batería del procesador	Este bit permite controlar el estado de funcionamiento de la batería de copia de seguridad del almacenamiento de datos y del programa en la memoria RAM: ● Establecido en 0: batería presente y en funcionamiento. ● Establecido en 1: batería ausente o fuera de servicio.	-	NO	Sí	Sí

Bit Símbolo	Función	Descripción	Estado o inicial	Modicon M340	Premium Atrium	Quantum
%S75 PCMCIABAT1	Estado de la batería de la tarjeta de memoria de almacenamiento de datos	Este bit es compatible con la versión 2.02 o posterior de Unity Pro. Permite supervisar el estado de la batería principal cuando la tarjeta de memoria se encuentra en la ranura PCMCIA inferior. Los procesadores Premium siguientes admiten %S75: TSX P57 4••, TSX P57 5•• y TSX P57 6••. NOTA: Para todos los demás procesadores Premium, %S75 sólo muestra un nivel de batería bajo cuando la batería ya se encuentra en un nivel crítico. Los procesadores Quantum siguientes admiten %S75: 140 CPU 671 60/60S*, 140 CPU 651 60/60S*, 140 CPU 652 60 y 140 CPU 651 50. %S75: ● se establece en 1 cuando la tensión de la batería principal es baja. Se mantiene la aplicación, aunque es necesario sustituir la batería según el procedimiento de mantenimiento predictivo (véase <i>Premium y Atrium con Unity Pro, Procesadores, bastidores y módulos de alimentación, Manual de instalación</i>). ● Se establece en 0 cuando la tensión de la batería principal es suficiente (siempre se mantiene la aplicación). * Los datos almacenados en una tarjeta de memoria en la ranura B no se procesan en proyectos de seguridad.	-	NO	Sí	Sí

Bit Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%S76 DIAGBUFFCONF	Búfer de diagnóstico configurado	El sistema pone a 1 este bit cuando está configurada la opción de diagnóstico. Se reserva entonces un búfer de diagnóstico destinado al almacenamiento de los errores procedentes de los DFB de diagnóstico. Este bit es de sólo lectura.	0	SÍ	SÍ	SÍ
%S77 DIAGBUFFFULL	Búfer de diagnóstico lleno	El sistema establece este bit en 1 cuando el búfer que recibe los errores de los bloques de funciones de diagnóstico está lleno. Este bit es de sólo lectura.	0	SÍ	SÍ	SÍ
%S78 HALTIFERROR	Parada en caso de error	Normalmente en 0. El usuario puede poner a 1 este bit para programar la parada del PLC por un fallo de la aplicación: %S15, %S18 y %20. En PLC de seguridad Quantum, el estado de parada se sustituye por el estado de error cuando se encuentra en la modalidad segura. Tenga en cuenta también que %S15 y %20 no están disponibles.	0	SÍ	SÍ	SÍ
%S79 MBFBCTRL	Control de bit forzado de Modbus	Este bit cambia el comportamiento del servidor Modbus de Quantum en lo que se refiere a bits forzados: ● A 0 (valor predeterminado), gestión estándar: el valor de bit cambia incluso si se fuerza el bit. ● Si el usuario lo pone en 1: la petición de escritura de bits en bits forzados no cambia su valor. Si no se produce ningún error en la respuesta de la petición. Al igual que otros accesos, el bit del historial está siempre actualizado, sea cual sea el estado de forzado.	0	NO	NO	SÍ

Descripción de los bits de sistema de %S80 a %S96

Descripción detallada

Descripción de los bits de sistema de %S80 a %S96:

Bit Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%S80 RSTMSGCNT	Puesta a cero de los contadores de mensajes	Normalmente en 0. El usuario puede poner a 1 este bit para volver a poner en cero los contadores de mensajes de %SW80 a %SW86.	0	SÍ	SÍ	SÍ
%S82	Ajuste de lectura MB+PCM CIA	Este bit se utiliza para cambiar la modalidad de intercambio de solicitud de MB+MBP100 PCMCIA. De manera predeterminada (valor 0), el sistema envía una solicitud a la tarjeta y leerá una respuesta en el siguiente ciclo Mast. Esta modalidad está recomendada para una duración corta de Mast. Cuando se establece en 1, el sistema envía una solicitud a la tarjeta y espera una respuesta. Esta modalidad está recomendada para una duración larga de Mast.	0	NO	SÍ	NO
%S90 COMRFSH	Actualización de las palabras comunes	Normalmente en estado 0. Pasa a 1 cuando se reciben las palabras comunes procedentes de otra estación de la red. Este bit puede ponerse a 0 mediante el programa o el terminal para comprobar el ciclo de intercambio de palabras comunes.	0	NO	SÍ	NO
%S91 LCKASYNREQ	Bloqueo de solicitud asíncrona	Si este bit se pone a 1, las solicitudes de comunicación asíncrona tratadas en la tarea de supervisión se ejecutan íntegramente sin interrumpir las tareas restantes, como las tareas MAST o FAST. De esta forma se garantiza una lectura o una escritura coherente de los datos. Recapitulación: el servidor de solicitud de la tarea de supervisión se direcciona por la puerta 7 (X-Way).	0	NO	SÍ	NO

Bit Símbolo	Función	Descripción	Estado o inicial	Modicon M340	Premium Atrium	Quantum
%S92 EXCHGTIME	Modalidad de medida de la función de comunicación	Normalmente en estado 0. El usuario puede poner a 1 este bit para situar las funciones de comunicación en la modalidad de medida de rendimiento. El parámetro de time-out (<i>véase Unity Pro, Comunicación, Biblioteca de bloques</i>) de las funciones de comunicación (en la tabla de gestión) muestra el tiempo de intercambio de ida y vuelta en milisegundos. Nota: Las funciones de comunicación se ejecutan con una base de tiempo de 100 ms.	0	Sí	Sí	NO

Bit Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%S94 SAVECURRVAL	Almacenamiento de los valores de ajustes	Normalmente en 0. El usuario puede poner en 1 este bit para reemplazar los valores iniciales de las variables declaradas con un atributo "Save" (por ejemplo: variables de DFB) por los valores actuales. Para Modicon M340, en un flanco ascendente %S94 el contenido de la RAM interna y de la tarjeta de memoria es diferente (%S96 = 0 y el LED CARDERR está encendido). Al iniciar en frío, los valores actuales se reemplazan por los valores iniciales más recientes, sólo si se ha realizado una función de almacenamiento en la tarjeta de memoria (Guardar backup o flanco ascendente %S66). El sistema vuelve a poner a 0 el bit %S94 cuando finaliza la sustitución. Nota: Este bit se debe utilizar con precaución: no ponga este bit a 1 de forma continua y utilice sólo la tarea maestra. Este bit no está disponible en PLC de seguridad Quantum. Cuando se utiliza con una memoria Flash PCMCIA TSX MFP (o TSX MCP), el almacenamiento de los valores de ajustes no está disponible.	0	Sí	Sí	Sí (excepto para PLC de seguridad)
%S96 BACKUPPROGOK	Programa de copia de seguridad listo	El sistema ajusta este bit a 0 ó a 1. - Lo establece en 0 cuando falta la tarjeta o está inservible (formato incorrecto, tipo no reconocido, etc.), o el contenido de la tarjeta no es coherente con la memoria RAM interna de aplicaciones. - Lo establece en 1 cuando la tarjeta está correcta y la aplicación es coherente con la memoria RAM interna de aplicaciones de la CPU.	-	Sí	NO	NO

ATENCIÓN

FALLO DE CARGA

El bit %S94 no debe ponerse a 1 durante una carga.

Si el bit %S94 se pone a 1, la carga puede fallar.

Si no se siguen estas instrucciones pueden producirse lesiones personales o daños en el equipo.

ATENCIÓN

PÉRDIDA DE DATOS

No se debe usar el bit %S94 con la memoria Flash PCMCIA TSX MFP ni TSX MCP. La función de este bit de sistema no está disponible con este tipo de memoria.

Si no se siguen estas instrucciones pueden producirse lesiones personales o daños en el equipo.

Descripción de los bits de sistema de %S100 a %S122

Descripción detallada

Descripción de los bits de sistema de %S100 a %S122:

Bit Símbolo	Función	Descripción	Estado inicial	Modico n M340	Premiu m Atrium	Quantu m
%S100 PROTTERINL	Protocolo en la toma del terminal	Situado en 0 o en 1 por el sistema según el estado del derivador INL/DPT en la toma de la consola. - Si falta el derivador (%S100=0), se utiliza el protocolo Uni-Telway maestro. - Si el derivador está presente (%S100=1), se utiliza el protocolo indicado por la configuración de la aplicación.	-	NO	SÍ	NO
%S118 REMIOERR	Fallo general de E/S Fipio	Normalmente en 1. El sistema pone a 0 este bit cuando aparece un fallo en un equipo conectado al bus de entradas/salidas remotas RIO y Fipio. Cuando desaparece el fallo, el sistema vuelve a poner el bit a 1.	-	SÍ	SÍ	SÍ
%S119 LOCIOERR	Fallo general de E/S en bastidor	Normalmente en 1. El sistema pone a 0 este bit cuando aparece un fallo en un módulo de E/S instalado en alguno de los bastidores. Cuando desaparece el fallo, el sistema vuelve a poner el bit a 1.	-	SÍ	SÍ	SÍ
%S120 DIOERRPLC	Fallo de bus DIO (CPU)	Normalmente en 1. El sistema pone a 0 este bit cuando aparece un fallo en un equipo conectado al bus DIO gestionado por la conexión Modbus Plus integrada en la CPU. Este bit no está disponible en PLC de seguridad Quantum.	-	NO	NO	SÍ (excepto para PLC de seguridad)

Bit Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%S121 DIOERRNOM1	Fallo del bus DIO (NOM n.º 1)	Normalmente en 1. El sistema pone a 0 este bit cuando aparece un fallo en un equipo conectado al bus DIO gestionado por el primer módulo 140 NOM 2••. Este bit no está disponible en PLC de seguridad Quantum.	-	NO	NO	SÍ (excepto para PLC de seguridad)
%S122 DIOERRNOM2	Fallo del bus DIO (NOM n.º 2)	Normalmente en 1. El sistema pone a 0 este bit cuando aparece un fallo en un equipo conectado al bus DIO gestionado por el segundo módulo 140 NOM 2••. Este bit no está disponible en PLC de seguridad Quantum.	-	NO	NO	SÍ (excepto para PLC de seguridad)

6.2 Palabras de sistema

Objeto

En este capítulo se describen las palabras de sistema Modicon M340, Atrium, Premium y Quantum.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Descripción de las palabras de sistema %SW0 a %SW11	188
Descripción de las palabras de sistema %SW12 a %SW29	192
Descripción de las palabras de sistema de %SW30 a %SW47	198
Descripción de las palabras de sistema %SW48 a %SW59	200
Descripción de las palabras de sistema de %SW70 a %SW100	203
Descripción de las palabras de sistema de %SW108 a %SW116	214
Descripción de las palabras de sistema %SW124 a %SW127	215

Descripción de las palabras de sistema %SW0 a %SW11

Descripción detallada

Descripción de las palabras de sistema %SW0 a %SW11.

Palabra Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%SW0 MASTERPERIOD	Periodo de exploración de la tarea maestra	Esta palabra se utiliza para modificar el periodo de la tarea maestra mediante el programa del usuario o el terminal. El periodo se expresa en ms (1 - 255 ms) %SW0=0 en funcionamiento cíclico. En un reinicio en frío: toma el valor definido por la configuración. Esta palabra no está disponible en PLC de seguridad Quantum.	0	SÍ	SÍ	SÍ (excepto para PLC de seguridad)
%SW1 FASTPERIOD	Periodo de exploración de la tarea rápida (FAST)	Esta palabra se utiliza para modificar el periodo de la tarea rápida mediante el programa del usuario o el terminal. El periodo se expresa en milisegundos (1 - 255 ms). En un reinicio en frío, toma el valor definido por la configuración. Esta palabra no está disponible en PLC de seguridad Quantum.	0	SÍ	SÍ	SÍ (excepto para PLC de seguridad)
%SW2 AUX0PERIOD %SW3 AUX1PERIOD %SW4 AUX2PERIOD %SW5 AUX3PERIOD	Periodo de la exploración de la tarea auxiliar.	Esta palabra se utiliza para modificar el periodo de las tareas definidas en la configuración, mediante el programa del usuario o el terminal. El periodo se expresa en décimas de ms (de 10 ms a 2,55 s). (1) únicamente en PLC 140 CPU 6•y TSX 57 5. Estas palabras no están disponibles en PLC de seguridad Quantum.	0	NO	SÍ (1)	SÍ (1) (excepto para PLC de seguridad)
%SW6 %SW7	Dirección IP	Proporciona la dirección IP del puerto Ethernet de la CPU. No se tiene en cuenta la modificación. Es 0, si la CPU no dispone de una conexión Ethernet.	-	SÍ	NO	NO

Palabra Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%SW8 TSKINHIBIN	Adquisición de monitorización de entrada de tarea	Normalmente en estado 0. El programa o el terminal pueden definir este bit en 1 ó 0. Impide la fase de adquisición de entrada de cada tarea: ● %SW8.0 = 1 impide la adquisición de entradas relativas a la tarea MAST. ● %SW8.1 = 1 impide la adquisición de entradas relativas a la tarea FAST. ● %SW8.2 a 5 = 1 impide la adquisición de entradas relativas a las tareas AUX 0...3. (1) Nota: En Modicon M340, las entradas/salidas distribuidas mediante el bus CANopen no se ven afectadas por la palabra %SW8. (2) Nota: En Quantum, las entradas/salidas distribuidas mediante el bus DIO no se ven afectadas por la palabra %SW8. Esta palabra no está disponible en PLC de seguridad Quantum.	0	Sí (1)	Sí	Sí (2) (excepto para PLC de seguridad)

Palabra Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%SW9 TSKINHIBOUT	Monitorización de actualización de salida de tarea	Normalmente en estado 0. El programa o el terminal pueden definir este bit en 1 ó 0. Impide la fase de actualización de cada tarea. ● %SW9.0 = 1 asignada a la tarea MAST; las salidas relativas a esta tarea ya no se gestionan. ● %SW9.1 = 1 asignada a la tarea FAST; las salidas relativas a esta tarea ya no se gestionan. ● %SW9.2 a 5 = 1 asignada a las tareas AUX 0...3; las salidas relativas a estas tareas ya no se gestionan. (3) Nota: En Modicon M340, las entradas/salidas distribuidas mediante el bus CANopen no se ven afectadas por la palabra %SW9. (4) Nota: En Quantum, las entradas/salidas distribuidas mediante el bus DIO no se ven afectadas por la palabra %SW9. Esta palabra no está disponible en PLC de seguridad Quantum.	0	Sí (3)	Sí	Sí (4) (excepto para PLC de seguridad)

⚠ ATENCIÓN

PELIGRO DE COMPORTAMIENTO IMPREVISTO

En Premium/Atrium:

Las salidas de los módulos ubicadas en el bus X cambian automáticamente a la modalidad configurada (retorno o mantenimiento). En el bus Fipio, algunos dispositivos no admiten la modalidad de retorno. Sólo admiten la modalidad de mantenimiento.

En Quantum:

Todas las salidas y el bastidor local o remoto (RIO) se mantienen en el estado anterior al cambio a 1 del bit %SW9 correspondiente a la tarea.

Las entradas/salidas distribuidas (DIO) no se asignan por la palabra del sistema %SW9.

Si no se siguen estas instrucciones pueden producirse lesiones personales o daños en el equipo.

Palabra Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%SW10 TSKINIT	Primer ciclo tras el arranque en frío	Si el valor del bit de la tarea actual es 0, la tarea realiza su primer ciclo después del arranque en frío. ● %SW10 . 0: asignada a la tarea MAST. ● %SW10 . 1: asignada a la tarea FAST. ● %SW10 . 2 a 5: asignadas a las tareas AUX 0...3. Esta palabra no está disponible en PLC de seguridad Quantum.	0	Sí	Sí	Sí (excepto para PLC de seguridad)
%SW11 WDGVALUE	Duración del watchdog	Lee la duración del watchdog. La duración se expresa en milisegundos (10 - 1.500 ms). Esta palabra no se puede modificar.	-	Sí	Sí	Sí

Descripción de las palabras de sistema %SW12 a %SW29

Descripción detallada

Descripción de las palabras de sistema %SW12 a %SW29:

Palabra Símbolo	Funcionam iento	Descripción	Estado inicial	Modico n M340	Premiu m Atrium	Quantu m
%SW12 UTWPORTADDR	Dirección del puerto serie del procesador	Para Premium: dirección Uni-Telway del puerto de terminal (en modalidad de esclavo) definida en la configuración y cargada en esta palabra en un arranque en frío. El sistema no tiene en cuenta la modificación del valor de esta palabra. Para Modicon M340: proporciona la dirección del esclavo de Modbus del puerto serie de la CPU. No se tiene en cuenta la modificación. Es 0, si la CPU no dispone de una conexión de puerto serie.	-	Sí	Sí	NO (consulte %SW12 a continua ción)
%SW12 APMODE	Modalidad del procesador de la aplicación	Únicamente para PLC de seguridad Quantum, esta palabra indica la modalidad de servicio del procesador de la aplicación del módulo CPU. 16#A501 = modalidad de mantenimiento 16#5AFE = modalidad segura Cualquier otro valor se interpreta como un error. Nota: En un sistema de seguridad Hot Standby, esta palabra se intercambia desde el PLC primario hasta el PLC Standby para informar al PLC Standby de la modalidad segura o de mantenimiento.	16#A501	NO	NO	Sí Sólo en PLC de segurida d
%SW13 XWAYNETWADDR	Dirección principal de la estación	Esta palabra indica lo siguiente para la red principal (Fipway o Ethway): - el número de la estación (byte de menor valor) de 0 a 127, - el número de la red (byte de mayor valor) de 0 a 63, (valor de los microinterruptores de la tarjeta PCMCIA).	254 (16#00FE)	NO	Sí	NO (consulte %SW13 a continua ción)

Palabra Símbolo	Funcionamiento	Descripción	Estado inicial	Modico n M340	Premiu m Atrium	Quantu m
%SW13 INTELMODE	Modalidad del procesador Intel	Únicamente para PLC de seguridad Quantum, esta palabra indica la modalidad de servicio del procesador Intel Pentium del módulo CPU. ● 16#501A = modalidad de mantenimiento ● 16#5AFE = modalidad segura Cualquier otro valor se interpreta como un error. Nota: En un sistema de seguridad Hot Standby, esta palabra se intercambia desde el PLC primario hasta el PLC Standby para informar al PLC Standby de la modalidad segura o de mantenimiento.	-	NO	NO	Sí Sólo en PLC de seguridad
%SW14 OSCOMMVERS	Versión comercial del procesador del PLC	Esta palabra contiene la versión de sistema operativo (SO) del procesador del PLC. Ejemplo: 16#0135 Versión: 01 Número de emisión: 35	-	Sí	Sí	Sí
%SW15 OSCOMMPATCH	Versión del parche del procesador del PLC	Esta palabra contiene la versión comercial del parche para el procesador del PLC. La codificación se lleva a cabo en el byte de menor valor de la palabra. Codificación: 0 = sin parche, 1 = A, 2 = B... Ejemplo: 16#0003 corresponde al parche C.	-	Sí	Sí	Sí
%SW16 OSINTVERS	Número de versión del firmware	Esta palabra contiene el número de versión en formato hexadecimal del firmware del procesador del PLC. Ejemplo: 16#0011 Versión: 2.1 N.º de versión: 17	-	Sí	Sí	Sí

Palabra Símbolo	Funcionamiento	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%SW17 FLOATSTAT	Estado de fallo en operación flotante	Cuando se detecta un fallo en una operación aritmética flotante, el bit %S18 pasa a 1 y el estado de error de %SW17 se actualiza según la codificación siguiente: ● %SW17.0 = operación inválida / el resultado no es un número; ● %SW17.1 =operando no normalizado / el resultado es aceptable (flag no gestionado por Modicon M340); ● %SW17.2 = división entre 0 / el resultado es infinito; ● %SW17.3 = desborde / el resultado es infinito; ● %SW17.4 = transgresión por debajo de rango / el resultado es 0; ● De %SW17.5 a 15 = no se utiliza. El sistema, así como el programa a fin de reutilizarla, vuelve a poner esta palabra a 0 en el arranque en frío. Esta palabra no está disponible en PLC de seguridad Quantum.	0	Sí	Sí	Sí (excepto para PLC de seguridad)

Palabra Símbolo	Funcionam iento	Descripción	Estado inicial	Modico n M340	Premiu m Atrium	Quantu m
%SD18: %SW18 y %SW19 100MSCOUNTER	Contador de tiempo absoluto	%SW18 representa los bytes de menor valor y %SW19 los bytes de mayor valor de la palabra doble %SD18, que el sistema incrementa cada décima de segundo. La aplicación puede leer o escribir estas palabras para realizar cálculos de duración. %SD18 se incrementa sistemáticamente, incluso en la modalidad STOP y estados equivalentes. Sin embargo, no se tienen en cuenta las veces que el PLC está apagado, puesto que la función no está vinculada al administrador de tiempo real, sino sólo al reloj en tiempo real. Para PLC de seguridad Quantum, sabiendo que los dos procesadores deben procesar exactamente los mismos datos, el valor de %SD18 se actualiza al comienzo de la tarea MAST y, a continuación, se congela durante la ejecución de la aplicación.	0	Sí	Sí	Sí

Palabra Símbolo	Funcionamiento	Descripción	Estado inicial	Modico n M340	Premiu m Atrium	Quantu m
%SD20: %SW20 y %SW21 MSCOUNTER	Contador de tiempo absoluto	Para PLC Quantum y M340, el sistema incrementa %SD20 cada milésima de segundo (incluso cuando el PLC está en STOP, %SD20 deja de incrementarse si el PLC está apagado). %SD20 puede leerse mediante el programa de usuario o el terminal. %SD20 se restablece en los arranques en frío. %SD20 no se restablece en los arranques en caliente. Para los PLC Premium TSX P57 1•4M/2•4M/3•4M/C024M/024M y TSX PCI57 204M/354M, el sistema incrementa %SD20 5 veces cada 5 milésimas de segundo. Para el resto de PLC Premium, %SD20 se establece según el contador de tiempo a 1 ms, como los PLC Quantum y M340. Para PLC de seguridad Quantum, sabiendo que los dos procesadores deben procesar exactamente los mismos datos, el valor de %SD18 se actualiza al comienzo de la tarea MAST y, a continuación, se congela durante la ejecución de la aplicación.	0	Sí	Sí	Sí
%SW23	Valor del conmutador rotativo	El byte de menor valor contiene el conmutador rotativo del procesador Ethernet. Puede leerse mediante el programa de usuario o el terminal.	-	Sí	NO	NO

Palabra Símbolo	Funcionam iento	Descripción	Estado inicial	Modico n M340	Premiu m Atrium	Quantu m
%SW26	Número de solicitudes procesadas	Esta palabra de sistema permite verificar en el lado del servidor el número de solicitudes procesadas por PLC por ciclo.	-	Sí	Sí	Sí
%SW27 %SW28 %SW29	Duración de la administraci ón del sistema	● %SW27 es la última duración de la administración del sistema. ● %SW28 contiene la duración máxima de la administración del sistema. ● %SW29 contiene la duración mínima de la administración del sistema. La duración de la administración del sistema depende de la configuración (número de E/S...) y de las peticiones de ciclo actuales (comunicación, diagnósticos). Duración de la administración del sistema = duración del ciclo Mast – duración de ejecución del código de usuario. Pueden leerse y escribirse mediante el programa de usuario o el terminal.	-	Sí	NO	NO

Descripción de las palabras de sistema de %SW30 a %SW47

Descripción detallada

Descripción de las palabras de sistema de %SW30 a %SW35:

Palabra Símbolo	Función	Descripción	Estado inicial	Modico n M340	Premiu m	Quantu m
%SW30 MASTCURRTIME	Tiempo de ejecución de la tarea maestra	Esta palabra indica el tiempo de ejecución del último ciclo de la tarea maestra (en ms).	-	SÍ	SÍ	SÍ
%SW31 MASTMAXTIME	Tiempo de ejecución máximo de la tarea maestra	Esta palabra indica el tiempo de ejecución más largo de tarea maestra desde el último arranque en frío (en ms).	-	SÍ	SÍ	SÍ
%SW32 MASTMINTIME	Tiempo de ejecución mínimo de la tarea maestra	Esta palabra indica el tiempo de ejecución más corto de tarea maestra desde el último arranque en frío (en ms).	-	SÍ	SÍ	SÍ
%SW33 FASTCURRTIME	Tiempo de ejecución de la tarea rápida	Esta palabra indica el tiempo de ejecución del último ciclo de la tarea rápida (en ms). Esta palabra no está disponible en PLC de seguridad Quantum.	-	SÍ	SÍ	SÍ (excepto para PLC de seguridad)
%SW34 FASTMAXTIME	Tiempo de ejecución máximo de la tarea rápida	Esta palabra indica el tiempo de ejecución más largo de la tarea rápida desde el último arranque en frío (en ms). Esta palabra no está disponible en PLC de seguridad Quantum.	-	SÍ	SÍ	YES (excepto para PLC de seguridad)
%SW35 FASTMINTIME	Tiempo de ejecución mínimo de la tarea rápida	Esta palabra indica el tiempo de ejecución más corto de la tarea rápida desde el último arranque en frío (en ms). Esta palabra no está disponible en PLC de seguridad Quantum.	-	SÍ	SÍ	SÍ (excepto para PLC de seguridad)

NOTA: Precisión sobre el tiempo de ejecución; es el tiempo transcurrido entre el principio (adquisición de las entradas) y el final (actualización de las salidas) de un ciclo de exploración. Este tiempo incluye el tratamiento de las tareas de sucesos y de la tarea rápida, así como el tratamiento de las peticiones de la consola.

Descripción de las palabras de sistema de %SW36 a %SW47.

Palabra Símbolo	Función	Descripción	Estado inicial	Modico n M340	Quantu m	Premium
%SW36 AUX0CURRTIME %SW39 AUX1CURRTIME %SW42 AUX2CURRTIME %SW45 AUX3CURRTIME	Tiempo de ejecución de las tareas auxiliares	Estas palabras indican el tiempo de ejecución del último ciclo de las tareas AUX 0...3 (en ms). (1) solamente en los PLC 140 CPU 6** y TSX P57 5**. Estas palabras no están disponibles en PLC de seguridad Quantum.	-	NO	Sí (1)	Sí (1) (excepto para PLC de seguridad)
%SW37 AUX0MAXTIME %SW40 AUX1MAXTIME %SW43 AUX2MAXTIME %SW46 AUX3MAXTIME	Tiempo de ejecución máx. de las tareas auxiliares	Estas palabras indican el tiempo de ejecución más largo de las tareas AUX 0...3 desde el último arranque en frío (en ms). (1) únicamente en los PLC 140 CPU 6** y TSX 57 5**. Estas palabras no están disponibles en PLC de seguridad Quantum.	-	NO	Sí (1)	Sí (1) (excepto para PLC de seguridad)
%SW38 AUX0MINTIME %SW41 AUX1MINTIME %SW44 AUX2MINTIME %SW47 AUX3MINTIME	Tiempo de ejecución mín. de las tareas auxiliares	Estas palabras indican el tiempo de ejecución más corto de las tareas AUX 0...3 desde el último arranque en frío (en ms). (1) únicamente en los PLC 140 CPU 6** y TSX 57 5**. Estas palabras no están disponibles en PLC de seguridad Quantum.	-	NO	Sí (1)	Sí (1) (excepto para PLC de seguridad)

Descripción de las palabras de sistema %SW48 a %SW59

Descripción detallada

Descripción de las palabras de sistema de %SW48 a %SW59.

Palabra Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%SW48 IOEVTNB	Número de eventos	Esta palabra indica el número de eventos procesados desde el último arranque en frío (en ms). El programa o el terminal pueden escribir esta palabra. Esta palabra no está disponible en PLC de seguridad Quantum.	0	Sí	Sí	Sí (excepto para PLC de seguridad)
%SW49 DAYOFWEEK %SW50 SEC %SW51 HOURMIN %SW52 MONTHDAY %SW53 YEAR	Función del reloj de tiempo real	Palabras del sistema que contienen la fecha y hora actuales (en BCD): ● %SW49: día de la semana: ● 1 = lunes ● 2 = martes ● 3 = miércoles ● 4 = jueves ● 5 = viernes ● 6 = sábado ● 7 = domingo ● %SW50: segundos (16#SS00), ● %SW51: horas y minutos (16#HHMM), ● %SW52: mes y día (16#MMDD), ● %SW53: año (16#AAAA). El sistema gestiona estas palabras cuando el bit %S50 se define en 0. El usuario del programa o el terminal pueden escribir estas palabras cuando el bit %S50 se define en 1.	-	Sí	Sí	Sí

Palabra Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%SW54 STOPSEC %SW55 STOPHM %SW56 STOPMD %SW57 STOPYEAR %SW58 STOPDAY	Función del reloj de tiempo real en la última parada	Las palabras del sistema contienen la fecha y la hora del último fallo de alimentación o detención del PLC (en formato decimal codificado en binario): ● %SW54: segundos (00SS), ● %SW55: horas y minutos (HHMM), ● %SW56: mes y día (MMDD), ● %SW57: año (AAAA), ● %SW58: el byte más significativo contiene el día de la semana (de 1 para lunes a 7 para domingo) y el byte menos significativo contiene el código de la última detención: ● 1 = cambio de RUN a STOP por el terminal o la entrada exclusiva ● 2 = detención por el watchdog (tarea del PLC o desborde del SFC) ● 4 = corte de alimentación u operación de bloqueo de la tarjeta de memoria ● 5 = detención por fallo de hardware ● 6 = detención por fallo de software. Los detalles del tipo de fallo de software se guardan en %SW125.	-	Sí	Sí	Sí

Palabra Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%SW59 ADJDATETIME	Ajuste de la fecha actual	Contiene dos series de 8 bits para ajustar la fecha actual. La acción se ejecuta siempre en el flanco ascendente del bit. Esta palabra se activa por el bit %S59=1. En la ilustración siguiente, los bits de la columna de la izquierda aumentan el valor y los bits de la columna de la derecha reducen el valor: ↗ + Bits 0 1 2 3 4 5 6 7 ↘ - 8 9 10 11 12 13 14 15 Tipo de valor Día de la semana Segundos Minutos Horas Días Meses Años Siglos	0	Sí	Sí	Sí

Descripción de las palabras de sistema de %SW70 a %SW100

Descripción detallada

Descripción de las palabras de sistema de %SW70 a %SW100.

Palabra Símbolo	Función	Descripción	Estado inicial	Modico n M340	Premiu m Atrium	Quantu m
%SW70 WEEKOFYEAR	Función del reloj de tiempo real	Palabra de sistema que contiene el número de la semana del año: 1 a 52 (en BCD).	-	Sí	Sí	Sí
%SW71 KEY_SWITCH	Posición de los conmutadores en el panel frontal de Quantum	Esta palabra proporciona la imagen de las posiciones de los conmutadores en el panel frontal del procesador Quantum. El sistema actualiza esta palabra automáticamente. ● Conmutador %SW71.0 = 1 en posición "Memoria protegida" ● Conmutador %SW71.1 = 1 en posición "PARADA" ● Conmutador %SW71.2 = 1 en posición "INICIO" ● Conmutador %SW71.8 = 1 en posición "MEM" ● Conmutador %SW71.9 = 1 en posición "ASCII" ● Conmutador %SW71.10 = 1 en posición "RTU" ● De %SW71.3 a 7 y de 11 a 15 no se utilizan.	0	NO	NO	Sí
%SW75 TIMEREVTNB	Contador de eventos de tipo Temporizador	Esta palabra contiene el número de eventos de tipo temporizador en cola. (1): no disponible en los siguientes procesadores: TSX 57 1•/2•/3•/4•/5•. Esta palabra no está disponible en PLC de seguridad Quantum.	0	Sí	Sí (1)	Sí (excepto para PLC de seguridad)
%SW76 DLASTREG	Función de diagnóstico: registro	Resultado del último registro: ● = 0 si el registro ha sido correcto, ● = 1 si el búfer de diagnóstico no se ha configurado, ● = 2 si el búfer de diagnóstico está lleno.	0	Sí	Sí	Sí

Palabra Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%SW77 DLASTDEREG	Función de diagnóstico: sin registro	Resultado de la última anulación de registro: ● = 0 si la anulación del registro ha sido correcta, ● = 1 si el búfer de diagnóstico no se ha configurado, ● = 21 si el identificador del error no es válido, ● = 22 si el error no se ha registrado.	0	Sí	Sí	Sí
%SW78 DNBERRBUF	Función de diagnóstico: número de errores	Número de errores actualmente en el búfer de diagnóstico.	0	Sí	Sí	Sí
%SW80 MSGCNT0 %SW81 MSGCNT1	Gestión de mensajes	El sistema actualiza estas palabras, que también se pueden restablecer mediante %S80. ● %SW80: número de mensajes enviados por el sistema al puerto del terminal (puerto serie Modbus en Modicon M340, puerto Uni-Telway en Premium). ● %SW81: número de mensajes recibidos por el sistema del puerto del terminal (puerto serie Modbus en Modicon M340, puerto Uni-Telway en Premium).	0	Sí	Sí	Sí
%SW82 MSGCNT2 %SW83 MSGCNT3	Gestión de mensajes	El sistema actualiza estas palabras, que también se pueden restablecer mediante %S80. ● %SW82: número de mensajes enviados por el sistema al módulo PCMCIA. ● %SW83: número de mensajes recibidos por el sistema del módulo PCMCIA.	0	NO	Sí	NO

Palabra Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%SW84 MSGCNT4 %SW85 MSGCNT5	Premium: gestión de telegramas Modicon M340: Gestión de mensajes	El sistema actualiza estas palabras, que también se pueden restablecer mediante %S80. Para Premium: ● %SW84: número de telegramas enviados por el sistema. ● %SW85: número de telegramas recibidos por el sistema. Para Modicon M340: ● %SW84: número de mensajes enviados al puerto USB. ● %SW85: número de mensajes recibidos por el puerto USB.	0	Sí	Sí	NO
%SW86 MSGCNT6	Gestión de mensajes	El sistema actualiza esta palabra, que también se pueden restablecer mediante %S80. Para Premium: ● Número de mensajes rechazados por el sistema. Para Modicon M340: ● Número de mensajes rechazados por el sistema, por ejemplo, no procesados por falta de recursos. Si Modbus Server rechaza el mensaje, entonces éste se corresponde con un mensaje de excepción de Modbus, enviado por la CPU al cliente Modbus remoto.	0	Sí	Sí	NO
%SW87 MSTSERVCNT	gestión del flujo de comunicación	Número de solicitudes procesadas por el servidor síncrono por ciclo de tarea maestra (MAST). Las solicitudes procesadas pueden proceder de todos los puertos de comunicación (con acceso al Modbus/UNI-TE del servidor, cada uno tiene sus propias limitaciones). Esto significa que las solicitudes de otros clientes y los EF de comunicación (como el explorador de E/S, el HMI conectado, etc.) deberían contarse.	0	Sí	Sí	Sí

Palabra Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%SW88 ASNSERV CNT %SW89 APPSERV CNT	Premium: gestión del flujo de comunicación Modicon M340: solicitudes HTTP y FTP recibidas por segundo por el servidor web y el servidor FTP del procesador	Para Premium: ● %SW88: número de solicitudes procesadas por el servidor asíncrono por ciclo de tarea maestra (MAST). ● %SW89: número de solicitudes procesadas por las funciones del servidor (inmediatamente) por ciclo de tarea maestra (MAST). Para Modicon M340: ● %SW88: número de solicitudes HTTP recibidas por segundo por el servidor web del procesador. ● %SW89: número de solicitudes FTP recibidas por segundo por el servidor FTP.	0	Sí	Sí	NO

Palabra Símbolo	Función	Descripción	Estado inicial	Modico n M340	Premiu m Atrium	Quantu m
%SW90 MAXREQNB	Número máximo de solicitudes procesadas por ciclo de tarea maestra	Esta palabra se utiliza para establecer un número máximo de solicitudes (todos los protocolos incluidos: UNITE, Modbus, etc.) que puede procesar el servidor del PLC por ciclo de tarea maestra. (No afecta a las solicitudes que envía el PLC como cliente.) Este número de solicitudes debe incluirse entre un mínimo y un máximo (definido como N+4) según el modelo. Para la gama M340: ● BMX P34 10••/20••/: N = 8 (mínimo 2, máximo 8 + 4 = 12). Para la gama Premium: ● TSX 57 0•: N = 4 (mínimo 2, máximo 4 + 4 = 9). ● TSX 57 1•: N = 4 (mínimo 2, máximo 4 + 4 = 8). ● TSX 57 2•: N = 8 (mínimo 2, máximo 8 + 4 = 12). ● TSX 57 3•: N = 12 (mínimo 2, máximo 12 + 4 = 16). ● TSX 57 4•: N = 16 (mínimo 2, máximo 16 + 4 = 20). ● TSX 57 5•: N = 16 (mínimo 2, máximo 16 + 4 = 20). Para la gama Quantum: ● 140 CPU 31••/43••/53••/: N = 10 (mínimo 5, máximo 10 + 4 = 14). ● 140 CPU 6••: N = 20 (mínimo 5, máximo 20 + 4 = 24).	N:	Sí	Sí	Sí

Palabra Símbolo	Función	Descripción	Estado inicial	Modico n M340	Premiu m Atrium	Quantu m
Continuación %SW90 MAXREQNB	Número máximo de solicitudes procesadas por ciclo de tarea maestra	El sistema inicializa la palabra con N (valor predeterminado). Si se introduce el valor 0, será el valor N el que se tendrá en cuenta. Si se introduce un valor entre 1 y el mínimo, se tendrá en cuenta el valor mínimo. Si se introduce un valor superior al máximo, se tendrá en cuenta el valor máximo. El número de solicitudes que se debe procesar por ciclo debe incluir las solicitudes procedentes de todos los puertos de comunicación (con acceso al servidor). Esto significa que también deberían incluirse las solicitudes procedentes de otros clientes y, a continuación, los EF de comunicación, como el explorador de E/S, la HMI conectada, y así sucesivamente.	N:	Sí	Sí	Sí
%SW91-92	Velocidades de mensajes de bloques de funciones	• %SW91: número de mensajes de bloques de funciones enviados por segundo. • %SW92: número de mensajes de bloques de funciones recibidos por segundo. Puede leerse mediante el programa de usuario o el terminal. Estos contadores no incluyen otras solicitudes salientes procedentes del explorador de E/S, por ejemplo.	0	Sí	Sí	NO

Palabra Símbolo	Función	Descripción	Estado inicial	Modico n M340	Premiu m Atrium	Quantu m
%SW93	Comando y estado del formato del sistema de archivos de la tarjeta de memoria	Puede leerse y editarse mediante el programa de usuario o el terminal. El cliente utiliza esta palabra para aplicar formato a la tarjeta de memoria o para limpiarla: La operación de formato elimina las páginas web. Para recuperarlas, realice una de las dos acciones siguientes. ● Utilice el FTP. ● Antes de realizar el formato, guarde las páginas web mediante el FTP. ● Después de realizar el formato, vuelva a cargar las páginas web mediante el FTP. ● Vuelva a instalar el sistema operativo de firmware del procesador. La operación de limpieza eliminar el contenido del directorio de almacenamiento de datos. Sólo se puede aplicar formato o limpiar en modalidad de detención: ● %SW93.0 = 1 un flanco ascendente inicia la operación de formato. ● %SW93.1 ofrece el estado del sistema de archivos tras la solicitud petición de operación de formato o limpieza: ● %SW93.1 = 0 sistema de archivos no válido o comando en curso. ● %SW93.1 = 1 sistema de archivos válido. ● %SW93.2 = 1 un flanco ascendente inicia la operación de limpieza.	0	Sí	NO	NO

Palabra Símbolo	Función	Descripción	Estado inicial	Modico n M340	Premiu m Atrium	Quantu m
%SW94 %SW95	Firma de modificación de la aplicación	Estas dos palabras contienen un valor de 32 bits que cambia con cada modificación de la aplicación, salvo si: ● Actualiza información de Upload. ● Sustituye el valor inicial del valor por el valor actual. ● Guarda el comando del parámetro. Pueden leerse mediante el programa de usuario o el terminal.	-	Sí	NO	NO

Palabra Símbolo	Función	Descripción	Estado inicial	Modico n M340	Premiu m Atrium	Quantu m
%SW96 CMDDIAGSAVE REST	Comando y diagnóstico de guardar y restaurar	Esta palabra se utiliza para copiar o eliminar el valor actual de %MW en la memoria flash interna (<i>véase página 117</i>) o procedente de ella y proporcionar el estado de la acción. Puede leerse mediante el programa de usuario o el terminal: ● %SW96.0: solicitud para copiar el valor actual de %MW en la memoria flash interna. Puesto a 1 por el usuario para solicitar guardar, y a 0 por el sistema cuando la operación de almacenamiento está en curso. Debe detener el procesador antes de realizar la copia a través de %SW 96.0. ● El sistema establece el valor %SW96.1 en 1 cuando finaliza el almacenamiento, y en 0 cuando la operación de almacenamiento está en curso. ● %SW96.2 = 1 indica un error en una operación de almacenamiento o restauración (consulte %SW96.8 a 15 para obtener una definición de los códigos de error). ● %SW96.3 = 1 indica que hay una operación de restauración en curso. ● %SW96.4 puede establecerse en 1 por parte del usuario para eliminar el área %MW en la memoria flash interna. ● %SW96.7 = 1 indica que la memoria interna contiene una copia de seguridad %MW válida. ● %SW96.8 a 15 son códigos de error cuando %SW96.2 se establece en 1: ● %SW96.9 = 1 indica que el número guardado %MW es inferior al número configurado; ● %SW96.8 = 1 y %SW96.9 = 1 indican que el número %MW guardado es superior al número configurado; ● %SW96.8 = 1, %SW96.9 = 1 y %SW96.10 = 1 indican un error	-	Sí	NO	NO
35006147 04/2009						211

Palabra Símbolo	Función	Descripción	Estado inicial	Modico n M340	Premiu m Atrium	Quantu m
%SW97 CARDSTS	Estado de tarjeta	Puede leerse mediante el programa de usuario o el terminal. Indica el estado de la tarjeta. %SW97: 0000 = no hay ningún error. 0001 = la copia de seguridad de la aplicación o la escritura de archivo se ha enviado a una tarjeta con protección de escritura. 0002 = no se reconoce la tarjeta o se ha dañado la copia de seguridad de la aplicación. 0003 = copia de seguridad de la aplicación solicitada, aunque no hay ninguna tarjeta disponible. 0004 = error de acceso a la tarjeta, por ejemplo, después de eliminar una tarjeta de modo incorrecto. 0005 = no existe ningún sistema de archivos en la tarjeta o el sistema de archivos no es compatible. Utilice %SW93.0 para formatear la tarjeta.	-	Sí	NO	NO
%SW99 ¹ INPUTADR/SW AP ¹	Gestión de redundancia de comunicaciones (1)	NOTA: Esta palabra se utiliza para el módulo Quantum y Premium, pero su función es distinta. Palabra utilizada para gestionar la redundancia de los módulos de red. Cuando se detecta un programa en un módulo de comunicación utilizado para acceder a un número de red x (X-WAY), es posible cambiar a otro módulo de comunicación (conectado a la misma red) introduciendo el número de red en la palabra %SW99. El sistema restablece %SW99 a 0.	0	NO	Sí ¹	NO

Palabra Símbolo	Función	Descripción	Estado inicial	Modico n M340	Premiu m Atrium	Quantu m
%SW99 ² CRA_COMPAT_ HIGH ²	Registro de estado alto de compatibilidad CRA	NOTA: Esta palabra se utiliza para el módulo Quantum y Premium, pero su función es distinta. La palabra aumenta cada vez que se lleva a cabo una modificación CCOTF en un PLC. Cuando se inserta un módulo en la estación RIO, el bit correspondiente está en 1 e indica que el módulo está conectado en la estación y es compatible con CCOTF.	0	NO	NO	Si ²
%SW100 CCOTF_COUNT	Registro de estado de conteo CCOTF	Palabra utilizada para gestionar la compatibilidad de CCOTF cuando se inserta un módulo nuevo. %SW100 = XXYY donde: ● XX se incrementa cada vez que se realiza una configuración de E/S en estado RUN en una estación RIO. ● YY se incrementa cada vez que se realiza una configuración de E/S en estado RUN en un bastidor local.	0	NO	NO	Sí

Descripción de las palabras de sistema de %SW108 a %SW116

Descripción detallada

Descripción de las palabras de sistema %SW108 a %SW116.

Palabra Símbolo	Función	Descripción	Estado inicial	Modico n M340	Quantu m	Premiu m Atrium
%SW108 FORCEDIOIM	Número de bits de módulos de E/S forzados	Esta palabra de sistema cuenta el número de bits de módulos de entradas/salidas forzados. La palabra se incrementa durante un forzado o disminuye en una cancelación de forzado.	0	Sí	Sí	Sí
%SW109 Æ	Número de canales analógicos forzados	Esta palabra del sistema cuenta el número de canales analógicos forzados. La palabra se incrementa durante un forzado o disminuye en una cancelación de forzado.	0	Sí	NO	Sí
%SW116 Æ	Fallo de E/S Fipio	Normalmente en 0. Cada bit de esta palabra es significativo de un estado de intercambio Fipio en la tarea en la que se prueba. El usuario debe volver a poner esta palabra a 0. Detalle de los bits de la palabra %SW116: ● %SW116.0 = 1 error de intercambio explícito (la variable no se intercambia en el bus) ● %SW116.1 = 1 timeout en un intercambio explícito (sin respuesta al final del timeout) ● %SW116.2 = 1 número máximo de intercambios explícitos simultáneos alcanzado ● %SW116.3 = 1 una trama es incorrecta ● %SW116.4 = 1 la longitud de una trama recibida es superior a la longitud declarada ● %SW116.5 = reservado a 0 ● %SW116.6 = 1 una trama no es válida o un agente se inicializa ● %SW116.7 = 1 ausencia de equipo configurado ● %SW116.8 = 1 fallo de canal (al menos un canal de un equipo indica un fallo) ● %SW116.9 a 15 = reservado a 0	-	NO	NO	Sí

Descripción de las palabras de sistema %SW124 a %SW127

Descripción detallada

Descripción de las palabras de sistema de %SW124 a %SW127.

Palabra Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%SW124 CPUERR	Error del sistema o del tipo de procesador	El sistema registra el último tipo de fallo del sistema detectado en esta palabra (estos códigos no se modifican en un reinicio en frío): ● 16#30: fallo de código del sistema, ● 16#53: fallo de time out durante intercambios de E/S, ● 16#60 a 64: desborde de stack, ● 16#65: El periodo de ejecución de tarea rápida es demasiado bajo ● 16#90: fallo del conmutador del sistema: IT impredecible.	-	Sí	Sí	Sí

Palabra Símbolo	Función	Descripción	Estado inicial	Modicon M340	Premium Atrium	Quantum
%SW125 BLKERRTYPE	Último fallo detectado	Esta palabra indica el código del último fallo detectado: Los siguientes códigos de error hacen que el PLC se detenga si %S78 se establece en 1. %S15, %S18 y %S20 están siempre activados con independencia de %S78: ● 16#2258: ejecución de la instrucción HALT, ● 16#DE87: error de cálculo en números de coma flotante (%S18, estos errores se incluyen en la palabra %SW17), ● 16#DEB0: desborde de watchdog (%S11), ● 16#DEF0: división entre 0 (%S18) ● 16#DEF1: error de transferencia de cadena de caracteres (%S15), ● 16#DEF2: error aritmético; %S18, ● 16#DEF3: desborde de índice (%S20). Nota: Los siguientes códigos 16#8xF4, 16#9xF4 y 16#DEF7 indican un error en la gráfica de función secuencial (SFC).	-	Sí	Sí	Sí
%SW126 ERRADDR0 %SW127 ERRADDR1	Dirección de instrucción de error de bloqueo	Dirección de la instrucción que ha generado el error de bloqueo de la aplicación. Para procesadores de 16 bits, TSX P57 1●/2●: ● %SW126 contiene el offset de esta dirección; ● %SW127 contiene el número de segmento de esta dirección. Para procesadores de 32 bits: ● %SW126 contiene la palabra menos significativa de esta dirección; ● %SW127 contiene la palabra más significativa de esta dirección.	0	Sí	Sí	Sí

6.3 Palabras de sistema específicas de Atrium/Premium

Objeto

En esta sección se describen las palabras de sistema %SW128 a %SW167 en el caso de autómatas Premium y Atrium.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Descripción de las palabras de sistema %SW60 a %SW65	218
Descripción de las palabras de sistema de %SW128 a %SW143	223
Descripción de las palabras de sistema de %SW144 a %SW146	224
Descripción de las palabras de sistema de %SW147 a %SW152	226
Descripción de la palabra de sistema %SW153	227
Descripción de la palabra de sistema %SW154	229
Descripción de las palabras de sistema Premium/Atrium de %SW155 a %SW167	230

Descripción de las palabras de sistema %SW60 a %SW65

Descripción detallada

Descripción de las palabras de sistema de %SW60 a %SW65 en Hot Standby de Premium y Atrium.

Palabra Símbolo	Función	Descripción	Estado o inicial	Premium	Atrium
%SW60 HSB_CMD	Registro de comando Hot Standby Premium	Significado de los diferentes bits de la palabra %SW60: ● %SW60.1 ● =0 define el autómata A al modo Fuera de línea. ● =1 define el autómata A al modo RUN. ● %SW60.2 ● =0 define el autómata B al modo Fuera de línea. ● =1 define el autómata B al modo RUN. ● Discrepancia en la versión de SO %SW60.4 ● =0 Si hay discrepancia de las versiones del SO con el autómata primario, standby pasa a la modalidad Fuera de línea. ● =1 Si hay discrepancia de las versiones del SO con el autómata primario, standby permanece en modalidad Standby. Discrepancia del SO del Firmware. Esto está relacionado con la versión del SO del procesador principal, la versión del SO del coprocesador incorporado, la versión del SO del ETY controlado y permite que un sistema Hot Standby funcione con versiones diferentes del SO que se esté ejecutando en el primario y el Standby.	0	Sí	NO

Palabra Símbolo	Función	Descripción	Estado inicial	Premium	Atrium
%SW61 HSB_STS	Registro de estado Hot Standby Premium	Significado de los diferentes bits de la palabra %SW61.0 a %SW61.6: ● %SW61.0 y %SW61.1 Estado del autómata local. ● %SW61.1=0 y %SW61.0=1: modo Fuera de línea. ● %SW61.1=1 y %SW61.0=0: Modalidad primaria. ● %SW61.1=1 y %SW61.0=1: Modalidad Standby. ● %SW61.2 y %SW61.3 Estado del autómata remoto. ● %SW61.3=0 y %SW61.2=1: modo Fuera de línea. ● %SW61.3=1 y %SW61.2=0: Modalidad primaria. ● %SW61.3=1 y %SW61.2=1: Modalidad Standby. ● %SW61.3=0 y %SW61.2=0: el autómata remoto no es accesible (apagado, sin comunicación). ● %SW61.4 está ajustado=1: cuando se detecta una discrepancia de lógica entre los controladores primario y Standby. ● %SW61.5 está ajustado a 0 ó 1, según la dirección MAC del coprocesador Ethernet: ● =0 el autómata con la dirección MAC más baja se convierte en el autómata A primario. ● =1 el autómata con la dirección MAC más alta se convierte en el autómata B. ● %SW61.6: este bit indica si la conexión de sinc. CPU entre los dos autómatas es válida: ● %SW61.6=0: la conexión de sinc. CPU es válida. El contenido del bit 5 es significativo. ● %SW61.6=1: la conexión de sync CPU no es válida. En este caso, el contenido del bit 5 no es significativo porque no puede realizarse la comparación de las dos direcciones MAC.	0	Sí	NO

Palabra Símbolo	Función	Descripción	Estado inicial	Premium	Atrium
%SW61 HSB_STS	Registro de estado Hot Standby Premium	Significado de los diferentes bits de la palabra %SW61.7 a %SW61.9: ● %SW61.7: este bit indica si hay una discrepancia de la versión del SO del procesador principal entre el primario y el standby: ● =0: no hay discrepancia en la versión del SO del firmware: ● =1: Discrepancia de las versiones del SO. Si la discrepancia de la versión del SO no está permitida en el registro de comando (bit 4 = 0), el sistema no funcionará como redundante en cuanto se señale el fallo. ● %SW61.8: este bit indica si hay una discrepancia de la versión del SO del procesador entre el primario y el standby: ● =0: no hay discrepancia de la versión del SO del coprocesador. ● =1: Discrepancia de la versión del SO del coprocesador. Si la discrepancia de la versión del SO no está permitida en el registro de comando (bit 4 = 0), el sistema no funcionará como redundante en cuanto se señale el fallo. ● %SW61.9: este bit indica si hay al menos un módulo ETY que no tenga la versión mínima: ● =0: todos los módulos ETY tienen la versión mínima. ● =1: al menos un módulo ETY no tiene la versión mínima". En este caso, no podrá iniciarse ningún autómata primario.	0	Sí	NO

Palabra Símbolo	Función	Descripción	Estado inicial	Premium	Atrium
%SW61 HSB_STS	Registro de estado Hot Standby Premium	Significado de los diferentes bits de la palabra %SW61.10 y %SW61.15: ● %SW61.10: este bit indica si hay una discrepancia de la versión del SO del ETY controlado entre el primario y el standby: ● =0: discrepancia de la versión del SO del ETY controlado. ● =1: discrepancia de la versión del SO del ETY controlado. Si la discrepancia de la versión del SO no está permitida en el registro de comando (bit 4 = 0), el sistema no funcionará como redundante en cuanto se señale el fallo. ● %SW61.15: Si %SW 61.15 se define = 1, este ajuste indica que el dispositivo del coprocesador Ethernet se ha configurado y funciona correctamente.	0	Sí	NO
%SW62 HSBY_REVERSE0 %SW63 HSBY_REVERSE1 %SW64 HSBY_REVERSE2 %SW65 HSBY_REVERSE3	Palabra de transferencia en Premium	Estas cuatro palabras son registros inversos reservados para el proceso de transferencia inverso. Estos cuatro registros inversos pueden escribirse en el programa de aplicación (primera sección) del controlador Standby y se transfieren en cada exploración del controlador Primario.	0	Sí	NO

Descripción de las palabras de sistema de %SW128 a %SW143

Descripción detallada

Descripción de las palabras de sistema %SW128 a %SW143:

Palabra Símbolo	Función	Descripción	Estado inicial
%SW128...143 ERRORCNXi con i=0 a 15	Fallo del punto de conexión Fipio	Cada bit de este grupo indica el estado de un equipo conectado al bus Fipio. Normalmente en 1. El estado 0 de uno de estos bits indica la aparición de un fallo en este punto de conexión. Para un punto de conexión no configurado, el bit correspondiente siempre es 1.	0

Tabla de correspondencia entre los bits de las palabras y la dirección de un punto de conexión:

	Bit 0	Bit 1	Bit 2	Bit 3	Bit 4	Bit 5	Bit 6	Bit 7	Bit 8	Bit 9	Bit 10	Bit 11	Bit 12	Bit 13	Bit 14	Bit 15
%SW128	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
%SW129	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
%SW130	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
%SW131	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63
%SW132	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79
%SW133	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95
%SW134	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111
%SW135	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127
%SW136	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143
%SW137	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159
%SW138	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175
%SW139	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191
%SW140	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207
%SW141	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223
%SW142	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239
%SW143	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255

Descripción de las palabras de sistema de %SW144 a %SW146

Descripción detallada

Descripción de las palabras de sistema %SW144 a %SW146.

Palabra Símbolo	Función	Descripción	Estado inicial
%SW144 BAOPMOD	Modalidad de servicio de la función de árbitro del bus Fipio	Esta palabra de sistema permite detener e iniciar la función de árbitro de bus y la función de productor/consumidor. Permite modificar la modalidad de arranque (automática y manual) del bus en caso de parada. ● %SW144.0 ● = 1: función de productor/consumidor en RUN. ● = 0: función de productor/consumidor en STOP (no se intercambia ninguna variable en el bus). ● %SW144.1 ● = 1: el árbitro de bus está en RUN 0. ● = 0: el árbitro de bus está en STOP (no se realiza ninguna exploración de variables ni mensajes en el bus). ● %SW144.2 ● = 1: arranque automático en caso de parada automática del bus. ● = 0: arranque manual en caso de parada automática del bus. ● %SW144.3 a 15 reservadas, %SW144.3 = 1, %SW144.4 a 15 = 0.	0
%SW145 BAPARAM	Modificación de los parámetros del árbitro del bus Fipio	El usuario pone a 1 los bits y a continuación el sistema a 0 cuando se efectúa la inicialización. ● %SW145.0 = 1: modificación de la prioridad del árbitro de bus; el byte más significativo de esta palabra de sistema contiene el valor de la prioridad del árbitro de bus que se aplicará a éste. ● %SW145.1 y %SW145.2 están reservadas. ● %SW145.3 a %SW145.7 reservadas a 0. ● %SW145.8 a %SW145.15: este byte contiene el valor que se aplicará al bus, según el valor del bit 0. La modificación de estos parámetros puede realizarse cuando el árbitro de bus está en RUN, pero la validación por parte de la aplicación necesita detener y, a continuación, arrancar ésta.	0
%SW146 BASTATUS	Visualización de la función de árbitro del bus Fipio	El byte menos significativo indica el estado de la función de productor/consumidor. El byte más significativo indica el estado de la función de árbitro de bus. Valor del byte: ● 16#00: la función no existe (no hay aplicación Fipio). ● 16#70: la función se inicializa pero no está operativa (en STOP). ● 16#F0: la función está en curso de ejecución normal (en RUN).	0

** ATENCIÓN****Relativo a las palabras %SW144 y %SW145**

La modificación de estas palabras de sistema puede implicar la parada de la estación del PLC.

Si no se siguen estas instrucciones pueden producirse lesiones personales o daños en el equipo.

Descripción de las palabras de sistema de %SW147 a %SW152

Descripción detallada

Descripción de las palabras de sistema %SW147 a %SW152:

Palabra Símbolo	Función	Descripción	Estado inicial
%SW147 TCRMAST	Tiempo del ciclo de la red MAST	Un valor distinto de cero indica, en ms, el valor del tiempo del ciclo de la red (TCRMAST) de la tarea MAST.	0
%SW148 TCRFAST	Tiempo del ciclo de la red FAST	Un valor distinto de cero indica, en ms, el valor del tiempo del ciclo de la red (TCRFAST) de la primera tarea FAST.	0
%SW150 NBFRSENT	Número de bloques de datos emitidos	Esta palabra indica el número de tramas emitidas por el administrador de la vía Fipio.	0
%SW151 NBFRREC	Número de tramas recibidas	Esta palabra indica el número de tramas recibidas por el administrador de la vía Fipio.	0
%SW152 NBRESENTMSG	Número de mensajes restablecidos	Esta palabra indica el número de restablecimientos de mensajes efectuados por el administrador de la vía Fipio.	0

Descripción de la palabra de sistema %SW153

Descripción detallada

Descripción de la palabra de sistema %SW153:

Palabra Símbolo	Función	Descripción	Estado inicial
%SW153 FipioERRO	Lista de los fallos del administrador de la vía Fipio.	El sistema pone a 1 cada bit y el usuario los vuelve a poner a 0. Véase la lista siguiente.	0

Descripción de los bits

- bit 0 = "fallo de rebasamiento de la estación": corresponde a una pérdida de símbolo MAC en la recepción, vinculada a una reacción demasiado lenta del receptor.
- bit 1 = "fallo de rechazo de mensaje": indica un mensaje con confirmación denegada o sin confirmación MAC en recepción,
- bit 2 = "fallo de rechazo de interrupción de variable".
- bit 3 = "fallo de underrun de la estación": corresponde a la incapacidad de la estación para respetar la velocidad de emisión en la red.
- bit 4 = "fallo de capa física": corresponde a una ausencia prolongada de transmisión en la capa física.
- bit 5 = "fallo de ausencia de eco": corresponde a un fallo para el cual el emisor está en curso de emisión, con una corriente de emisión comprendida en el rango de funcionamiento y detección simultánea de ausencia de señal en la misma vía.
- bit 6 = "fallo de conversación": corresponde a un fallo para el cual el emisor dispone del control de la línea desde un tiempo superior al límite máximo de funcionamiento definido. Este fallo puede deberse, por ejemplo, a un deterioro del modulador o a una capa de enlace de datos defectuosa.
- bit 7 = "fallo de hipocorriente": corresponde a un fallo en el cual el emisor genera en su línea, cuando se solicita, una corriente inferior al límite mínimo de funcionamiento definido. Este fallo se debe al aumento de la impedancia de línea (línea abierta...).
- bit 8 = "fallo de bloque de datos fragmentado": indica la recepción de un silencio en el cuerpo de una trama después de identificar a un delimitador de principio de trama y antes de identificar a un delimitador de final de trama. La aparición de un silencio en condiciones normales de funcionamiento tiene lugar después de identificar un delimitador de final de trama.
- bit 9 = "fallo de trama CRC en la recepción": indica una diferencia de valor entre el CRC calculado en la trama que se recibe normalmente y el CRC contenido en la misma.

- bit 10 = "fallo de codificación de trama en la recepción": indica la recepción de determinados símbolos, pertenecientes exclusivamente a las secuencias de delimitación de principio y fin de la trama, en el cuerpo de ésta.
- bit 11 = "fallo de longitud de la trama recibida": el número de bytes recibidos para el cuerpo de una trama es superior a 256 bytes.
- bit 12 = "recepción de una trama de tipo desconocido": en el cuerpo de una trama, el primer byte identifica el tipo de trama de enlace. En el protocolo de enlace de la norma WorldFip se define un determinado número de tipos de tramas. La presencia de cualquier otro código en una trama corresponde a un fallo de tipo de trama desconocido.
- bit 13 = "recepción de una trama truncada": un fragmento de trama se caracteriza por el reconocimiento de una secuencia de símbolos del delimitador de final de trama cuando la estación de destino esperaba recibir un delimitador de principio de trama.
- bit 14 = "no se utiliza, valor no significativo".
- bit 15 = "no se utiliza, valor no significativo"

Descripción de la palabra de sistema %SW154

Descripción detallada

Descripción de la palabra de sistema %SW154:

Palabra Símbolo	Función	Descripción	Estado inicial
%SW154 FipioERR1	Lista de los fallos del administrador de la vía Fipio.	El sistema pone cada bit a 1 y el usuario los vuelve a poner a 0. Véase la lista siguiente.	0

Descripción de los bits

- bit 0 = "timeout de secuencia aperiódica": indica un rebasamiento de la ventana de mensajes o de variables aperiódicas en un ciclo elemental del macrociclo.
- bit 1 = "rechazo de solicitud de mensaje": indica una saturación de la cola de espera de mensajes; el árbitro de bus ya no puede momentáneamente almacenar y a continuación satisfacer una petición.
- bit 2 = "rechazo de comando de actualización urgente": indica una saturación de la cola de espera de peticiones de intercambio de variables aperiódicas urgentes; el árbitro de bus ya no puede momentáneamente almacenar ni satisfacer la petición.
- bit 3 = "rechazo de comando de actualización no urgente": indica una saturación de la cola de espera de peticiones de intercambio de variables aperiódicas no urgentes; el árbitro de bus ya no puede momentáneamente almacenar ni satisfacer la petición.
- bit 4 = "fallo de silencio": el árbitro de bus no ha detectado ninguna actividad en el bus durante un período superior al tiempo normalizado WorldFip.
- bit 5 = "colisión en la red en la emisión de identificador": indica una actividad en la red durante los períodos teóricos de silencio. Entre una emisión y la espera de una respuesta por parte del árbitro de bus, no debe circular nada en el bus. Si el árbitro de bus detecta una actividad, genera un fallo de colisión (por ejemplo, cuando varios árbitros están activos al mismo tiempo en el bus).
- bit 6 = "fallo de overrun del árbitro de bus": indica un conflicto de acceso a la memoria de la estación del árbitro de bus.
- bit 7 = "no se utiliza, valor no significativo".
- bit 8 a bit 15 = reservado a 0.

Descripción de las palabras de sistema Premium/Atrium de %SW155 a %SW167

Descripción detallada

Descripción de las palabras de sistema %SW155 a %SW167:

Palabra Símbolo	Función	Descripción	Estado inicial
%SW155 NBEXPLFIP	Número de intercambios explícitos en Fipio	Número de intercambios explícitos que se procesan en Fipio, efectuados mediante instrucciones (READ_STS, REA_PARAM, etc.). Tiene también en cuenta los intercambios explícitos efectuados mediante solicitudes (READ_IO_OBJECT, WRITE_IO_OBJECT, etc.) Nota: El número de intercambios explícitos es siempre inferior a 24.	0
%SW160 à %SW167 PREMRACK0 a PREMRACK7	Estado de funcionamiento de los módulos del autómata	Las palabras %SW160 a %SW167 se asocian respectivamente a los bastidores 0 a 7. Los bits de 0 a 15 de cada una de estas palabras se asocian a los módulos situados en las posiciones de 0 a 15 de estos bastidores. El bit está a 0 si el módulo presenta un fallo y a 1 si el módulo funciona correctamente. Ejemplo: %SW163.5 =0 El módulo situado en el emplazamiento 5 del bastidor 3 presenta un fallo.	0

6.4 Palabras de sistema específicas de Quantum

Objeto de esta sección

En esta sección se describen las palabras de sistema %SW60 a %SW640 en el caso de PLC Quantum.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Descripción de las palabras de sistema Quantum %SW60 a %SW123	232
Descripción de las palabras de sistema Quantum de %SW98 a %SW100	236
Descripción de las palabras de sistema Quantum de %SW110 a %SW179	237
Descripción de las palabras de sistema Quantum de %SW180 a %SW640	240

Descripción de las palabras de sistema Quantum %SW60 a %SW123

Descripción detallada

Descripción de las palabras de sistema %SW60 a %SW123.

Palabra Símbolo	Función	Descripción	Estado inicial
%SW60 HSB_CMD	Registro de comando Hot Standby Quantum	Significado de los diferentes bits de la palabra %SW60: ● %SW60.0=1 invalida los comandos introducidos en la pantalla (teclado). ● %SW60.1 ● = 0 establece el PLC A en la modalidad OFFLINE. ● = 1 establece el PLC A en la modalidad RUN. ● %SW60.2 ● = 0 establece el PLC B en la modalidad OFFLINE. ● = 1 establece el PLC B en la modalidad RUN. Nota: Si los bits %SW60.1 y %SW60.2 se ponen a 0 de forma simultánea, se produce una conmutación: ● el controlador primario pasa a RUN offline y ● el controlador Standby funciona como RUN primario. ● %SW60.3=0 establece el PLC Standby en la modalidad offline si las aplicaciones son diferentes. ● %SW60.4 ● =0 autoriza a actualizar el firmware sólo después de detener la aplicación. ● =1 autoriza a actualizar el firmware sin detener la aplicación. ● %SW60.5=1 solicitud de transferencia de aplicación de Standby al primario. ● %SW60.8 ● =0 la dirección cambia al puerto 1 Modbus en el primer intercambio. ● =1 ninguna dirección cambia al puerto 1 Modbus en el primer intercambio. ● %SW60.9 ● =0 la dirección cambia al puerto 2 Modbus en el primer intercambio. ● =1 ninguna dirección cambia al puerto 2 Modbus en el primer intercambio. ● %SW60.10 ● =0 la dirección cambia al puerto 3 Modbus en el primer intercambio. ● =1 ninguna dirección cambia al puerto 3 Modbus en el primer intercambio.	0

Palabra Símbolo	Función	Descripción	Estado inicial
%SW61 HSB_STS	Registro de estado de Quantum	Significado de los diferentes bits de la palabra %SW61: ● Bits de modalidad de servicio %SW61.0 y %SW61.1 del PLC ● %SW61.1=0, %SW61.0=1: modalidad offline. ● %SW61.1=1, %SW61.0=0: modalidad primaria. ● %SW61.1=1, %SW61.0=1: modalidad secundaria (Standby). ● Bits de modalidad de servicio %SW61.2 y %SW61.3 del otro PLC ● %SW61.3=0, %SW61.2=1: modalidad offline. ● %SW61.3=1, %SW61.2=0: modalidad primaria. ● %SW61.3=1, %SW61.2=1: modalidad secundaria (Standby). ● %SW61.3=0, %SW61.2=0: el PLC remoto no es accesible (desconectado, sin comunicación). ● %SW61.4=0 las aplicaciones son idénticas en ambos PLC. ● %SW61.5 ● =0 el PLC se utiliza como unidad A. ● =1 el PLC se utiliza como unidad B. ● %SW61.7 ● =0 Misma versión del SO del PLC. ● =1 Diferente versión del PLC. ● %SW61.8 ● =0 Misma versión del SO del coprocesador. ● =1 Diferente versión del coprocesador. ● %SW61.12 ● =0 La información proporcionada por el bit 13 no es revelante ● =1 La información proporcionada por el bit 13 es válida ● %SW61.13 ● =0 Dirección NOE definida como IP ● =1 Dirección NOE definida como IP + 1 ● %SW61.15 ● =0 Hot Standby no activado ● =1 Hot Standby activado	0

Palabra Símbolo	Función	Descripción	Estado inicial
%SW62 HSBY_REVERSE0 %SW63 HSBY_REVERSE1 %SW64 HSBY_REVERSE2 %SW65 HSBY_REVERSE3	Palabra de transferen cia	El usuario de la primera sección de la tarea maestra puede modificar estas 4 palabras. Se transfieren automáticamente desde el procesador Standby y actualizan el PLC primario. Se pueden leer en el PLC primario y usarse como parámetros primarios de la aplicación.	0
%SW123	Palabra de sistema que utiliza el sistema	Esta palabra la utiliza el sistema y no puede utilizarla la aplicación de usuario.	—

Descripción de las palabras de sistema Quantum de %SW98 a %SW100

Descripción detallada

Descripción de las palabras de sistema %SW98 a %SW100.

Palabra Símbolo	Función	Descripción	Estado inicial
%SW98 CRA_COMPAT_LOW	Registro de estado bajo de compatibilidad CRA	Significado de los diferentes bits de la palabra %SW98: ● %SW98.0 no se utiliza y está establecido en 0 de forma predeterminada. ● %SW98.1 a %SW98.15 ● =0 establece la estación 2 en 16 no es compatible. ● =1 establece la estación 2 en 16 es compatible.	0
%SW99 CRA_COMPAT_HIGH	Registro de estado alto de compatibilidad CRA	Significado de los diferentes bits de la palabra %SW99: ● %SW99.0 a %SW99.15 ● =0 establece la estación 17 en 32 no es compatible. ● =1 establece la estación 17 en 32 es compatible.	0
%SW100 CCOTF_COUNT	Registro de estado de conteo CCOTF	Significado de los diferentes bits de la palabra %SW100: ● XYY ● XX se incrementa cada vez que se realiza una configuración de E/S en estado RUN en una estación RIO, ● YY se incrementa cada vez que se realiza una configuración de E/S en estado RUN en un bastidor local.	0

Descripción de las palabras de sistema Quantum de %SW110 a %SW179

Descripción detallada

Descripción de las palabras de sistema de %SW110 a %SW179. Estas palabras están activas en los PLC Quantum 140 CPU 6***.

Palabra Símbolo	Función	Descripción	Estado inicial
%SW110	Número de área de memoria no restringida para %M	La palabra del sistema proporciona información sobre el tamaño del área de memoria no restringida para %M.	0
%SW111	Número de área de memoria no restringida para %MW	La palabra del sistema proporciona información sobre el tamaño del área de memoria no restringida para %MW.	0
%SW128 NB_P502_CNX	Número de conexiones abiertas	El byte de mayor valor de esta palabra indica el número de conexiones TCP abiertas en el puerto 502 de la conexión Ethernet TCP/IP.	0
%SW129 NB_DENIED_CNX	Número de conexiones denegadas	Esta palabra indica el número de conexiones TCP denegadas del puerto 502 de la conexión Ethernet TCP/IP.	0
%SW130 NB_P502_REF	Número de mensajes rechazados	Esta palabra indica el número de mensajes TCP denegados en el puerto 502 de la conexión Ethernet TCP/IP.	0
%SW132 y %SW133 NB_SENT_MSG	Número de mensajes enviados	Esta palabra doble %SDW132 indica el número de mensajes enviados al puerto 502 de la conexión Ethernet TCP/IP.	0
%SW134 y %SW135 NB_RCV_MSG	Número de mensajes recibidos	Esta palabra doble %SDW134 indica el número de mensajes recibidos en el puerto 502 de la conexión Ethernet TCP/IP.	0
%SW136 NB_IOS_CNX	Número de equipos explorados	Esta palabra indica el número de equipos explorados en el puerto 502 de la conexión Ethernet TCP/IP.	0
%SW137 NB_IOS_MSG	Número de mensajes de exploración de E/S recibidos	Esta palabra indica el número de mensajes recibidos por segundo desde el servicio de exploración de E/S en el puerto 502 de la conexión Ethernet TCP/IP.	0
%SW138 GLBD_ERROR	Error de coherencia de datos globales	Error de coherencia de datos globales.	0
%SW139 BW_GLBD_IOS	Carga del servicio de exploración de E/S y de datos globales	El byte de menor valor de esta palabra mide el porcentaje de carga relativa a la exploración de E/S. El byte de mayor de valor de esta palabra mide el porcentaje de carga relativa a los datos globales.	0

Palabra Símbolo	Función	Descripción	Estado inicial
%SW140 BW_OTHER_MSG	Carga del servicio de mensajería y otros servicios	El byte de menor valor de esta palabra mide el porcentaje de carga relativa al servicio de mensajería. El byte de mayor valor de esta palabra mide el porcentaje de carga relativa a otros servicios.	0
%SW141 y %SW142 IP_ADDR	Dirección IP	Esta palabra doble %SDW141 recibe la dirección IP de la conexión Ethernet.	0
%SW143 y %SW144 IP_NETMASK	Máscara de subred IP	Esta palabra doble %SDW143 recibe la máscara de subred de la conexión Ethernet.	0
%SW145 y %SW146 IP_GATEWAY	Dirección predeterminada de la pasarela de Ethernet	Esta palabra doble %SDW145 recibe la dirección predeterminada de la pasarela de Ethernet.	0
De %SW147 a %SW149 MAC_ADDR1 a 3	Direcciones MAC	Las palabras %SW147, %SW148 y %SW149 codifican las direcciones MAC 1, MAC 2 y MAC 3 respectivamente.	0
%SW150	Versión del coprocesador	Esta palabra de código de la versión del coprocesador para los PLC 140 CPU 671-60. La versión se muestra en formato hexadecimal.	0
%SW151 BOARD_STS	Estado de la conexión Ethernet	Esta palabra codifica el estado de la conexión Ethernet. ● Bit 0 = 0 si se detiene la conexión Ethernet. ● Bit 1 = 0. ● Bit 2: 0 = modalidad semidúplex, 1 = dúplex completo. ● Bit 3 = 0. ● Bit 4 a 11: = 7 para Quantum, = 6 para Hot Standby Quantum. ● Bit 12: 0 = conexión de 10 Mbits, 1 = conexión de 100 Mbits. ● Bit 13: 0 = conexión 10/100Base-TX (par trenzado). ● Bit 14: 0. ● Bit 15: 0 = conexión Ethernet inactiva, 1 = conexión Ethernet activa.	0

Palabra Símbolo	Función	Descripción	Estado inicial
De %SW160 a %SW167 REFRESH_IO	Estado de funcionamiento de los equipos mediante exploración de E/S	Los bits de las palabras de %SW160 a %SW167 se asocian a los dispositivos que han sido sometidos a exploración de E/S. El bit se establece en 0 si el dispositivo presenta un fallo y en 1 si el dispositivo funciona correctamente. %SW160.0: dispositivo n.º 1. %SW160.1: dispositivo n.º 2. %SW167.15: dispositivo n.º 128. Nota: Estas palabras de sistema sólo están disponibles en coprocesadores Quantum y no están disponibles en módulos NOE.	-
De %SW168 a %SW171 VALID_GD	Estado de funcionamiento de los datos globales	Los bits de las palabras de %SW168 a %SW171 se asocian a los datos globales. El bit está en 0 si el equipo presenta un fallo y en 1 si el equipo funciona correctamente. %SW168.0: dispositivo n.º 1. %SW168.1: dispositivo n.º 2. %SW171.15: dispositivo n.º 64.	-

Descripción de las palabras de sistema Quantum de %SW180 a %SW640

Descripción detallada

Descripción de las palabras de sistema de %SW180 a %SW640.

Palabra Símbolo	Función	Descripción	Estado inicial
De %SW180 a %SW339 IOHEALTHij i=1..32, j=1..5	Estado de funcionamiento de los módulos del PLC.	Las palabras de %SW180 a %SW339 están asociadas a las estaciones de PLC: cinco palabras por estación correspondientes a los bastidores de 1 a 5 de cada estación. %SW180: estado de funcionamiento de los módulos del bastidor 1 de la estación 1. %SW181: estado de funcionamiento de los módulos del bastidor 2 de la estación 1. %SW185: estado de funcionamiento de los módulos del bastidor 1 de la estación 2. %SW186: estado de funcionamiento de los módulos del bastidor 2 de la estación 2. Los bits de 0 a 15 de cada una de estas palabras se asocian a los módulos situados en las posiciones de 16 a 1 de estos bastidores. El bit está en 0 si el módulo presenta un fallo y en 1 si el módulo funciona correctamente. Ejemplo: %SW185.5 = 0. El módulo situado en el slot 11 del bastidor 1 de la estación 2 presenta un fallo. Nota: Los módulos 140 XBE 100 00 (<i>véase Quantum con Unity Pro, Hardware, Manual de referencia</i>) requieren una gestión particular. Estas palabras no están disponibles en PLC de seguridad.	0
%SW340 MB+DIOSLOT	Número de slot del procesador con conexión Modbus Plus.	Número de slot del procesador que integra la conexión Modbus Plus para la conexión a la primera red DIO. El número de slot está codificado de 0 a 15. Esta palabra no está disponible en PLC de seguridad Quantum.	-

Palabra Símbolo	Función	Descripción	Estado inicial
De %SW341 a %SW404 MB+IOHEALTHi i=1..64	Estado de funcionamiento de los módulos de estaciones distribuidas de la primera red DIO.	Las palabras de %SW341 a %SW404 están asociadas a las estaciones distribuidas (DIO): 64 palabras asociadas a las 64 estaciones DIO de la primera red. %SW341: estado de funcionamiento de los módulos de la estación 1. %SW342: estado de funcionamiento de los módulos de la estación 2. %SW404: estado de funcionamiento de los módulos de la estación 64. Los bits de 0 a 15 de cada una de estas palabras se asocian a los módulos situados en las posiciones de 16 a 1 de estas estaciones. El bit está en 0 si el módulo presenta un fallo y en 1 si el módulo funciona correctamente. Ejemplo: %SW362.5 = 0 El módulo situado en el slot 11 de la estación 22 de la primera red DIO presenta un fallo. Nota: Para los módulos 140 CRA 2** **, el valor de este bit no resulta significativo, ya que se fija siempre en 0. Estas palabras no están disponibles en PLC de seguridad.	-
%SW405 NOM1DIOSLOT	Número de slot del primer módulo de interfase de red DIO.	Número de slot del módulo 140 NAME 2** para la conexión a la segunda red DIO. El número de slot está codificado de 0 a 15. Esta palabra no está disponible en PLC de seguridad Quantum.	-

Palabra Símbolo	Función	Descripción	Estado inicial
De %SW406 a %SW469 NOM1DIOHEALTHi i=1..64	Estado de funcionamiento de los módulos de estaciones distribuidas de la segunda red DIO.	Las palabras de %SW406 a %SW469 están asociadas a las estaciones distribuidas (DIO): 64 palabras asociadas a las 64 estaciones DIO de la segunda red. %SW406: estado de funcionamiento de los módulos de la estación 1. %SW407: estado de funcionamiento de los módulos de la estación 2. %SW469: estado de funcionamiento de los módulos de la estación 64. Los bits de 0 a 15 de cada una de estas palabras se asocian a los módulos situados en las posiciones de 16 a 1 de estas estaciones. El bit está en 0 si el módulo presenta un fallo y en 1 si el módulo funciona correctamente. Ejemplo: %SW412.5 = 0 El módulo situado en el slot 11 de la estación 7 de la segunda red DIO presenta un fallo. Nota: Para los módulos 140 CRA 2** ***, el valor de este bit no resulta significativo, ya que se fija siempre en 0. Estas palabras no están disponibles en PLC de seguridad.	-
%SW470 NOM2DIOSLOT	Número de slot del segundo módulo de interfase de red DIO.	Número de slot del módulo 140 NOM 2** para la conexión a la tercera red DIO. El número de slot está codificado de 0 a 15. Esta palabra no está disponible en PLC de seguridad Quantum.	-

Palabra Símbolo	Función	Descripción	Estado inicial
De %SW471 a %SW534 NOM2DIOHEALTHi i=1..64	Estado de funcionamiento de los módulos de estaciones distribuidas de la tercera red DIO.	Las palabras de %SW471 a %SW534 están asociadas a las estaciones distribuidas (DIO): 64 palabras asociadas a las 64 estaciones DIO de la tercera red. %SW471: estado de funcionamiento de los módulos de la estación 1. %SW472: estado de funcionamiento de los módulos de la estación 2. %SW534: estado de funcionamiento de los módulos de la estación 64. Los bits de 0 a 15 de cada una de estas palabras se asocian a los módulos situados en las posiciones de 16 a 1 de estas estaciones. El bit está en 0 si el módulo presenta un fallo y en 1 si el módulo funciona correctamente. Ejemplo: %SW520.5 = 0 El módulo situado en el slot 11 de la estación 86 de la tercera red DIO presenta un fallo. Nota: Para los módulos 140 CRA 2** **, el valor de este bit no resulta significativo y se fija siempre en 0. Estas palabras no están disponibles en PLC de seguridad.	-

Palabra Símbolo	Función	Descripción	Estado inicial
%SW535 RIOERRSTAT	Error RIO en el inicio.	Esta palabra almacena el código de error de inicio. Se establece siempre en 0 cuando el sistema está en funcionamiento; en caso de error, el autómata no se inicia, pero genera un código de estado de parada. 01: longitud de asignación de las E/S. 02: número de vínculo de E/S remotas. 03: número de estaciones en la asignación de las E/S. 04: suma de control de asignación de las E/S. 10: longitud del descriptor de estación. 11: número de estación de E/S. 12: tiempo de autonomía de la estación. 13: número de puerto ASCII. 14: número de módulos de la estación. 15: estación ya configurada. 16: puerto ya configurado. 17: más de 1.024 puntos de salida. 18: más de 1.024 puntos de entrada. 20: dirección del slot del módulo. 21: dirección del bastidor del módulo. 22: número de bytes de salida. 23: número de bytes de entrada. 25: primer número de referencia. 26: segundo número de referencia. 28: bits internos fuera del rango de 16 bits. 30: módulo de salida impar sin pareja. 31: módulo de entrada impar sin pareja. 32: referencia de módulo impar sin pareja. 33: referencia 1x después del registro 3x. 34: referencia del módulo de prueba ya utilizado. 35: el módulo 3x no es de prueba. 36: el módulo 4x no es de prueba.	-

Palabra Símbolo	Función	Descripción	Estado inicial
%SW536 CAERRCNT0 %SW537 CAERRCNT1 %SW538 CAERRCNT2	Estado de la comunicación en el cable A	Las palabras de %SW536 a %SW538 son palabras de error de comunicación en el cable A. ● %SW536: ● byte más significativo: cuenta los errores de trama; ● byte menos significativo: cuenta los desbordes del receptor DMA. ● %SW537: ● byte más significativo: cuenta los errores de recepción; ● byte menos significativo: cuenta las recepciones de las estaciones incorrectas. ● %SW538: ● %SW538.15 = 1, trama corta, ● %SW538.14 = 1, sin final de trama, ● %SW538.3 = 1, error CRC, ● %SW538.2 = 1, error de alineación, ● %SW538.1 = 1, error de desborde, ● %SW538.13 a 4 y 0 no se utilizan.	-
%SW539 CBERRCNT0 %SW540 CBERRCNT1 a %SW541 CBERRCNT2	Estado de la comunicación en el cable B	Las palabras de %SW539 a %SW541 son palabras de error de comunicación en el cable A. ● %SW539: ● byte más significativo: cuenta los errores de trama; ● byte menos significativo: cuenta los desbordes del receptor DMA. ● %SW540: ● byte más significativo: cuenta los errores de recepción; ● byte menos significativo: cuenta las recepciones de las estaciones incorrectas. ● %SW541: ● %SW541.15 = 1, trama corta, ● %SW541.14 = 1, sin final de trama, ● %SW541.3 = 1, error CRC, ● %SW541.2 = 1, error de alineación, ● %SW541.1 = 1, error de desborde, ● %SW541.13 a 4 y el bit 0 no se utilizan.	-

Palabra Símbolo	Función	Descripción	Estado o inicial
%SW542 GLOBERRCNT0 %SW543 GLOBERRCNT1 %SW544 GLOBERRCNT2	Estado de comunicación global	Las palabras de %SW542 a %SW544 son palabras de error de comunicación global. ● %SW542: muestra el estado de la comunicación global. ● %SW542.15 = 1, comunicación en funcionamiento correcto. ● %SW542.14 = 1, comunicación en el cable A en funcionamiento correcto. ● %SW542.13 = 1, comunicación en el cable B en funcionamiento correcto. ● %SW542.11 a 8 = contador de las comunicaciones perdidas. ● %SW542.7 a 0 = contador totalizador de nuevo intento. ● %SW543: es el contador totalizador global de los errores para el cable A: ● byte más significativo: cuenta los errores detectados; ● byte menos significativo: cuenta las «faltas de respuesta». ● %SW544: es el contador totalizador global de los errores para el cable B: ● byte más significativo: cuenta los errores detectados; ● byte menos significativo: cuenta las «faltas de respuesta».	-

Palabra Símbolo	Función	Descripción	Estado inicial
De %SW545 a %SW640 MODUNHEALTHi IOERRCNTi IORETRYi (i=1..32)	Estado de las estaciones descentralizadas	Las palabras de %SW545 a %SW640 permiten describir el estado de las estaciones descentralizadas. Se utilizan tres palabras de estado para cada estación. • %SW545: muestra el estado de la comunicación global de la estación 1 • %SW545.15 = 1, comunicación en funcionamiento correcto, • %SW545.14 = 1, comunicación en el cable A en funcionamiento correcto, • %SW545.13 = 1, comunicación en el cable B en funcionamiento correcto, • %SW545.11 a 8 = contador de las comunicaciones perdidas, • %SW545.7 a 0 = contador totalizador de nuevo intento. • %SW546: es el contador totalizador global de los errores para el cable A de la estación 1: • byte más significativo: cuenta los errores detectados; • byte menos significativo: cuenta las «faltas de respuesta». • %SW547: es el contador totalizador global de los errores para el cable B de la estación 1: • byte más significativo: cuenta los errores detectados; • byte menos significativo: cuenta las «faltas de respuesta». Las palabras: De %SW548 a 550 están asignadas a la estación 2, De %SW551 a 553 están asignadas a la estación 3, ... De %SW638 a 640 están asignadas a la estación 32.	-
De %SW545 a %SW547 MODUNHEALTH1 IOERRCNT1 IORETRY1	Estado de la estación local	Para los PLC en los que la estación 1 está reservada a las entradas/salidas locales, las palabras de estado de %SW545 a %SW547 se utilizan de la forma siguiente. • %SW545: estado de la estación local, • %SW545.15 = 1, todos los módulos funcionan correctamente. • %SW545.14 a 8 = no se utilizan, siempre a 0, • %SW545.7 a 0 = número de veces en las que el módulo estaba defectuoso; el contador vuelve a 255, • %SW546: se utiliza como contador de los errores del bus de entradas/salidas de 16 bits, • %SW547: se utiliza como contador de repetición del bus de entradas/salidas de 16 bits.	-

6.5 Palabras de sistema específicas de Modicon M340

Descripción de las palabras de sistema: %SW146 y %SW147, %SW160 a %SW167

Descripción detallada

Descripción de las palabras de sistema %SW146 y %SW147, y %SW160 a %SW167:

Palabra Símbolo	Función	Descripción	Estado inicial
%SW146 y %SW147	Modicon M340	Estas dos palabras de sistema contienen el número de serie único de la tarjeta SD (32 bits). Si no hay una tarjeta SD o hay una no reconocida, las dos palabras de sistema se establecen en 0. Esta información se puede utilizar para proteger una aplicación (<i>véase Modicon M340 con Unity Pro, Procesadores, bastidores y módulos de fuente de alimentación, Manual de configuración</i>) contra la duplicación.	-
%SW160 a %SW167 PREMRACK0 a PREMRACK7	Error en bastidor 0 a 7 de Premium y Modicon M340	Las palabras %SW160 a %SW167 se asocian respectivamente a los bastidores 0 a 7. Los bits de 0 a 15 de cada una de estas palabras se asocian a los módulos situados en las posiciones de 0 a 15 de estos bastidores. El bit está a 0 si el módulo presenta un fallo y a 1 si el módulo funciona correctamente. Ejemplo: %SW163.5 = 0. El módulo ubicado en la posición 5 del bastidor 3 presenta un fallo. En caso de semibastidores, dos semibastidores forman un bastidor normal completo, identificado únicamente por un conmutador.	-

Descripción de los datos



Objeto

En esta parte se describen los diferentes tipos de datos que se pueden utilizar en un proyecto y la forma de hacerlo.

Contenido de esta parte

Esta parte contiene los siguientes capítulos:

Capítulo	Nombre del capítulo	Página
7	Presentación general de los datos	251
8	Tipos de datos	259
9	Instancias de datos	317
10	Referencias de datos	331

Presentación general de los datos



Objeto

En este capítulo se presenta de forma muy general:

- los diferentes tipos de datos,
- las instancias de datos y
- las referencias de datos.

Contenido de este capítulo

Este capítulo contiene los siguiente apartados:

Apartado	Página
General	252
Descripción general de las familias de tipos de datos	253
Descripción general de instancias de datos	255
Descripción general de referencias de datos	257
Reglas sintácticas de los nombres de tipos/instancias	258

General

Introducción

Un **dato** designa a un objeto en el que se pueden iniciar instancias, por ejemplo:

- una variable,
- un bloque de función.

Los datos se definen en tres fases. que son:

- la fase **tipos de datos**, en la que se precisa:
 - su categoría,
 - su formato.
- la fase **instancias de datos**, en la que se definen su emplazamiento en la memoria y su propiedad, que puede ser:
 - localizada o
 - sin localizar.
- la fase **referencias de datos**, en la que se define su medio de acceso:
 - por valor inmediato,
 - por nombre,
 - por dirección.

Figura

A continuación se indican las tres fases que caracterizan a los datos:



Iniciar una instancia en un dato consiste en asignarle un emplazamiento en la memoria según su tipo.

Referenciar un dato consiste en definirle una referencia (nombre, dirección, etc.) que permita localizarlo en la memoria.

Descripción general de las familias de tipos de datos

Introducción

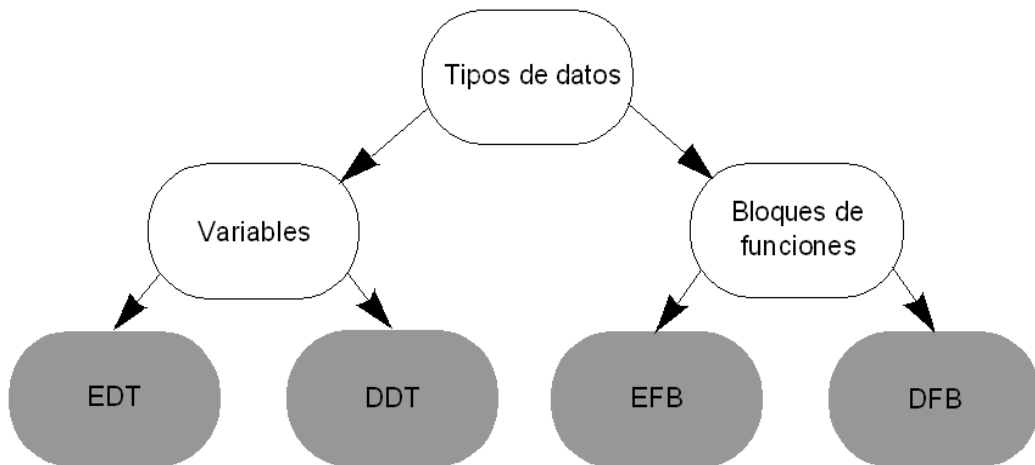
Un **tipo de dato** es una información de software que especifica, para un dato:

- su estructura,
- su formato,
- una lista de sus atributos y
- su comportamiento.

Todas las instancias del tipo de datos comparten estas propiedades.

Figura

Las familias de tipos de datos se clasifican en diferentes categorías (gris oscuro):



Definiciones

Familias de tipos de datos y sus definiciones.

Familia	Definición
EDT	Tipos de datos elementales (Elementary data types), por ejemplo: ● Bool ● Int ● Byte ● Word ● DWord ● etc.
Matrices	Tipos de datos derivados (Derived data types), por ejemplo: ● Matrices que contienen elementos del mismo tipo: ● Matriz de Bool (matriz de EDT) ● Matriz de tablas (matriz de DDT) ● Matriz de estructuras (matriz de DDT) ● Estructuras que contienen elementos de distintos tipos: ● Estructura de Bool, Word, etc. (estructura de EDT) ● Estructura de matrices, estructura de estructuras, estructura de matrices/estructuras (estructura de DDT) ● Estructura de Bool, estructura de matriz, etc. (estructura de EDT y DDT) ● Estructura relativa a los datos de entradas/salidas (estructura de IODDT) ● Estructura que contiene variables que restablecen las propiedades de estado de una acción o transición de una gráfica de función secuencial
EFB	Bloques de funciones elementales escritos en lenguaje C. Entre ellos, se incluyen: ● Variables de entradas ● Variables internas ● Variables de salidas ● Un algoritmo de procesamiento
DFB	Bloques de funciones derivados escritos en lenguajes de automatización (Literal estructurado, Lista de instrucciones, etc.). Entre ellos, se incluyen: ● Variables de entradas ● Variables internas ● Variables de salidas ● Un algoritmo de procesamiento

Descripción general de instancias de datos

Introducción

Una **instancia de datos** es una entidad funcional individual que posee todas las características del tipo de datos del que depende.

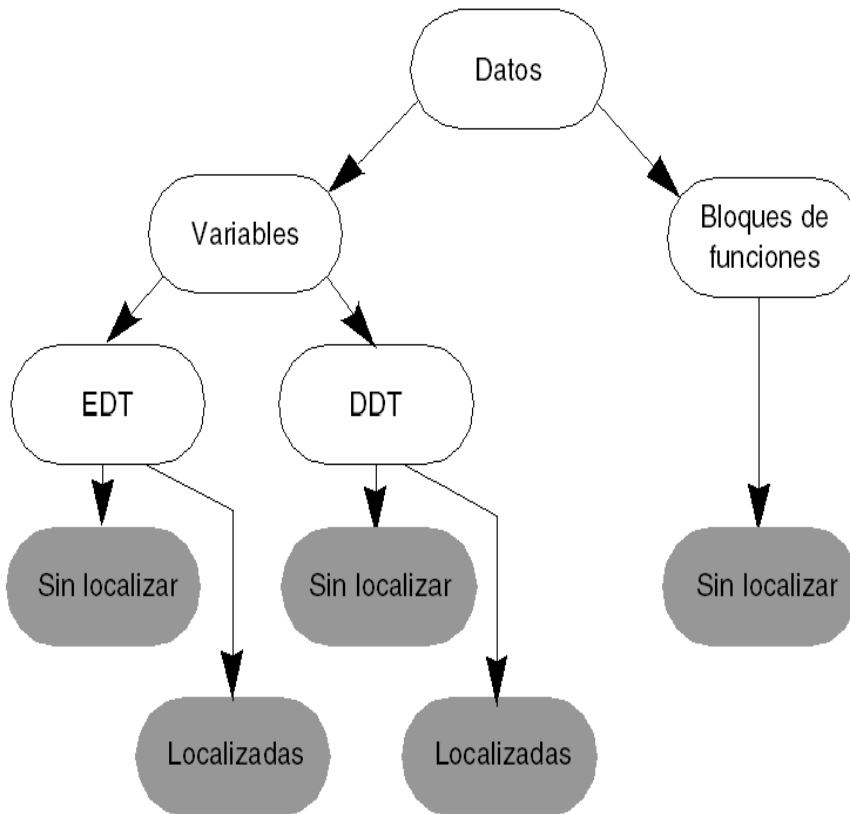
Se puede vincular una o varias instancias a un tipo de datos.

La instancia de datos puede tener una asignación de memoria:

- no localizada o
- localizada

Figura

Asignación de memoria de las instancias (gris oscuro) pertenecientes a los diferentes tipos.



Definiciones

Definición de las asignaciones de memoria de las instancias de datos.

Instancia de datos	Definición
Sin localizar	El sistema asigna automáticamente el emplazamiento de memoria de la instancia y puede cambiar cada vez que se genera la aplicación. La instancia tiene como dirección un nombre (símbolo) que elige el usuario.
Localizadas	El emplazamiento de la memoria de la instancia es fijo, está predefinido y no cambia nunca. La instancia tiene como dirección un nombre (símbolo) que elige el usuario y una dirección topológica que define el fabricante, o bien únicamente la dirección topológica del fabricante.

Descripción general de referencias de datos

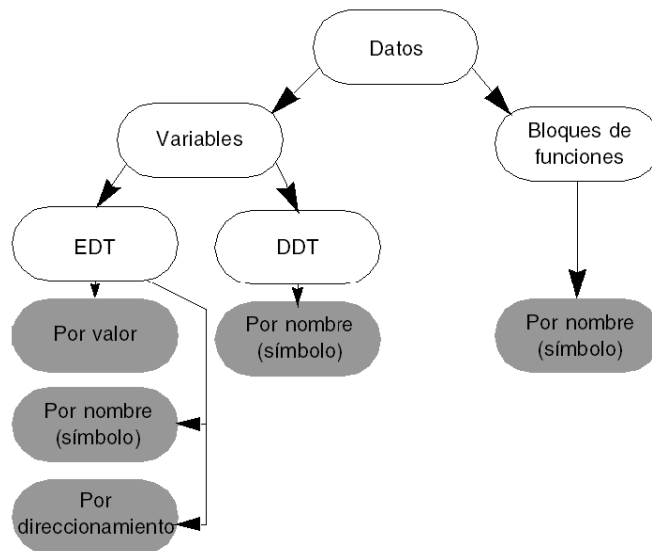
Introducción

Una **referencia de datos** permite al usuario acceder a la instancia de dicho dato por:

- valor inmediato, verdadero únicamente para los datos de tipo EDT,
- direccionamiento, verdadero únicamente para los datos de tipo EDT, y
- nombre (símbolo), verdadero para todos los tipos de datos EDT, DDT, EFB, DFB, así como los objetos SFC.

Figura

Referencias de datos posibles según el tipo de datos (gris oscuro).



Reglas sintácticas de los nombres de tipos/instancias

Introducción

La sintaxis de los nombres de tipos y de variables se puede llevar a cabo con o sin la utilización del establecimiento extendido de caracteres. La selección se efectúa en la ficha **Extensiones de lenguaje** de la opción **Herramientas->Ajustes del proyecto**.

- Si la opción **Permitir establecimiento extendido de caracteres** está seleccionada, la aplicación cumple la norma CEI.
- Si la opción **Permitir establecimiento extendido de caracteres** no está seleccionada, el usuario tiene cierta flexibilidad, pero la aplicación no cumple la norma CEI.

El establecimiento extendido de caracteres que se emplea para los nombres introducidos en la aplicación incluye:

- Los bloques de funciones del usuario DFB (bloque de función derivada) o los DDT (tipos de datos derivados),
- los elementos internos que componen un tipo de datos de bloque de función DFB/EFB o un tipo de datos derivados (DDT), y
- las instancias de datos y

Si la casilla "Permitir establecimiento..." está seleccionada

Los nombres introducidos son cadenas compuestas de caracteres alfanuméricos, del carácter Underscore.

Las reglas son las siguientes:

- El primer carácter del nombre es un carácter alfabético o es el carácter Underscore, y
- no puede haber dos caracteres Underscore consecutivos.

Si la casilla "Permitir establecimiento..." no está seleccionada

Los nombres introducidos son cadenas compuestas de caracteres alfanuméricos, del carácter Underscore.

Están permitidos caracteres adicionales, como:

- Los caracteres correspondientes a los códigos ASCII de 192 a 223 (excepto el código 215) y
- Los caracteres correspondientes a los códigos ASCII de 224 a 255 (excepto el código 247) y

Las reglas son las siguientes:

- El primer carácter del nombre es un carácter **alfanumérico** o es el carácter Underscore, y
- los caracteres Underscore **pueden ser** consecutivos.

Tipos de datos

8

Objeto

En este capítulo se describen todos los tipos de datos que se pueden utilizar en una aplicación.

Contenido de este capítulo

Este capítulo contiene las siguientes secciones:

Sección	Apartado	Página
8.1	Tipos de datos elementales (EDT) con formato Binario	260
8.2	Tipos de datos elementales (EDT) con formato BCD	271
8.3	Tipos de datos elementales (EDT) con formato Real	278
8.4	Tipo de datos elementales (EDT) con formato de cadena de caracteres	281
8.5	Tipos de datos elementales (EDT) con formato de cadena de bits	284
8.6	Tipos de datos derivados (DDT/IODDT)	288
8.7	Tipos de datos de bloques de función (DFB\EFB)	301
8.8	Tipos de datos genéricos (GDT)	309
8.9	Tipos de datos pertenecientes a las gráficas de funciones secuenciales (SFC)	311
8.10	Compatibilidad entre los tipos de datos	313

8.1 Tipos de datos elementales (EDT) con formato Binario

Objeto

En esta sección, se describe el tipo de datos con formato Binario, que son:

- tipos booleanos,
- tipos enteros y
- tipos Time.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Descripción general de tipos de datos con formato binario	261
Tipos booleanos	263
Tipos enteros	268
El tipo Time	270

Descripción general de tipos de datos con formato binario

Introducción

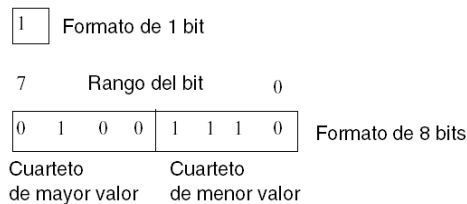
Los tipos de datos con formato binario pertenecen a la familia de datos elementales EDT (Elementary data type), que agrupa tipos de datos **denominados simples** y no compuestos (matrices, estructuras o bloques de funciones).

Recapitulación sobre el formato binario

Un dato con formato binario se compone de uno o varios bits, cada uno de los cuales está representado por una de las cifras de la base 2, es decir, **0 ó 1**.

La escala del dato depende del número de bits que la componen.

Ejemplo:



Un dato puede ser:

- Con signo, en cuyo caso el bit de rango más alto es el bit con signo:
 - 0 indica un valor positivo, y
 - 1 indica un valor negativo.

El rango de valores es el siguiente:

$$[-2^{\langle Bits - 1 \rangle}, 2^{\langle Bits - 1 \rangle} - 1]$$

- Sin signo, en cuyo caso todos los bits representan el valor
- El rango de valores es el siguiente:

$$[0, 2^{Bits} - 1]$$

Bits = número de bits (formato).

Tipos de datos con formato binario

Lista de los tipos de datos:

Tipo	Designación	Formato (bits)	Valor predeterminado
BOOL	Booleano	8	0=(False)
EBOOL	Booleano con detección de flancos y forzado	8	0=(False)
INT	Entero	16	0
DINT	Entero doble	32	0
UINT	Entero sin signo	16	0
UDINT	Entero doble sin signo	32	0
TIME	Entero doble sin signo	32	T=0s

Tipos booleanos

Presentación

Existen dos tipos de booleanos, que son:

- El tipo BOOL, que contiene únicamente el valor FALSE (=0) o TRUE (=1), y
- el tipo EBOOL, que contiene el valor FALSE (=0) o TRUE (=1), pero también incluye información relativa a la gestión de los flancos (ascendentes o descendentes) y el forzado.

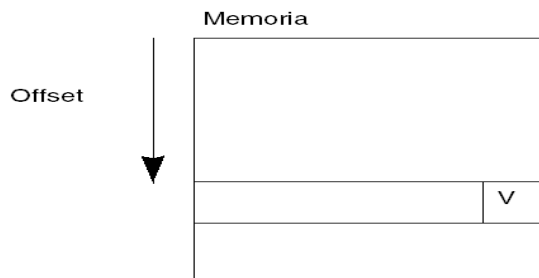
Principio del tipo BOOL

Este tipo ocupa un byte en la memoria, pero el valor se guarda solamente en un bit.

El valor predeterminado de este tipo es FALSE (=0).

Se puede acceder a él a través de una dirección que contenga el offset en el byte correspondiente:

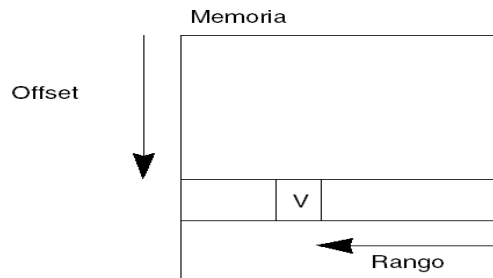
Direccionamiento:



En el caso **del bit extraído de la palabra**, se puede acceder a él mediante una dirección que contenga la información siguiente:

- Un offset en el byte correspondiente.
- El rango que define la posición en la palabra.

Direccionamiento:



Principio del tipo EBOOL

Este tipo ocupa un byte en la memoria, que incluye:

- El bit para el valor (V).
- El bit de registro (H) para la gestión de los flancos (ascendentes o descendentes). En cada cambio del estado del objeto, el valor se copia en este bit.
- El bit que contiene el estado de forzado (F). Igual a 0 si el objeto no se ha forzado e igual a 1 si el objeto se ha forzado.

El valor predeterminado de los bits asociados al tipo EBOOL es FALSE (=0).

Se puede acceder a él a través de una dirección que especifique el offset en el byte correspondiente.

Direccionamiento:

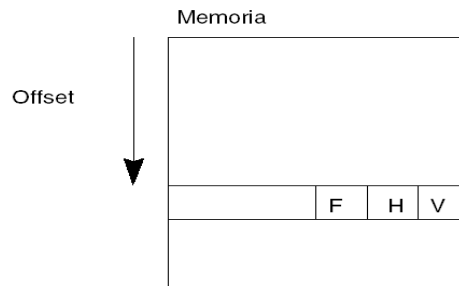


Gráfico de tendencias de registro

El gráfico de tendencias siguiente presenta el principio de los estados de los bits (valor e historial) asociados al tipo EBOLL.

Los flancos ascendentes del bit de valor (1, 4) se copian en el bit de registro del ciclo de PLC siguiente (2, 5). Los flancos descendentes del bit de valor (2, 7) se copian en el bit de registro del ciclo de PLC siguiente (3, 8).

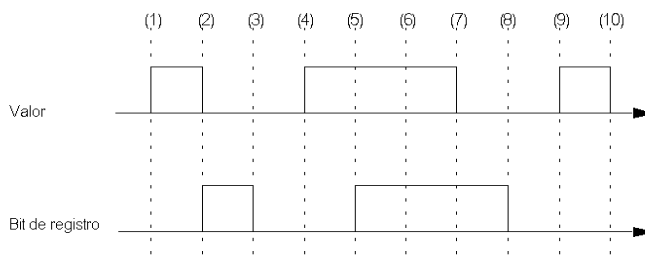
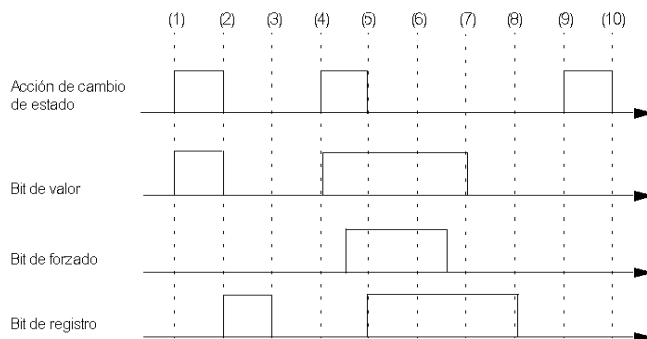


Gráfico de tendencias y forzado

El gráfico de tendencias siguiente presenta el principio de los estados de los bits (valor, historial y forzado) asociados al tipo EBOLL.

Los flancos ascendentes del bit de valor (1, 4) se copian en el bit de registro del ciclo de PLC siguiente (2, 5). Los flancos descendentes del bit de valor (2, 7) se copian en el bit de registro del ciclo de PLC siguiente (3, 8). Entre (4 y 5) el bit de forzado es igual a 1; los bits de valor y registro permanecen en 1.



Variables de PLC que pertenecen a los tipos booleanos

Lista de variables

Variable	Tipo
Bit interno	EBOOL
Bit de sistema	BOOL
Bit extraído de la palabra	BOOL
Entradas %I	
Bit de error de módulo	BOOL
Bit de error de canal	BOOL
Bit de entrada	EBOOL
Salidas %Q	
Bit de salida	EBOOL

Compatibilidad entre BOOL y EBOOL

Las operaciones permitidas entre estos dos tipos de variables son las siguientes:

- La copia de valores.
- La copia de direcciones.

Copia entre tipos

	Destino BOOL	Destino EBOOL
Fuente BOOL	Sí	Sí
Fuente EBOOL	Sí	Sí

Compatibilidad entre los parámetros de las funciones elementales (EF)

Parámetro efectivo (externo a la EF)	Parámetro formal BOOL (interno a la EF)	Parámetro formal EBOOL (interno a la EF)
BOOL	Sí	No
EBOOL	In -> Sí In-Out -> No Out -> Sí	Sí

Compatibilidad entre los parámetros de los bloques de funciones (EFB\DFB)

Parámetro efectivo (externo al FB)	Parámetro formal BOOL (interno al FB)	Parámetro formal EBOOL (interno al FB)
BOOL	Sí	In -> Sí In-Out -> No Out -> Sí
EBOOL	In -> Sí In-Out -> No Out -> Sí	Sí

Compatibilidad entre variables de tabla

	Destino ARRAY[i..j] OF BOOL	Destino ARRAY[i..j] OF EBOOL
Fuente ARRAY[i..j] OF BOOL	Sí	No
Fuente ARRAY[i..j] OF EBOOL	No	Sí

Compatibilidad entre variables estáticas

	Direccionamiento directo BOOL (%MW:xi)	Direccionamiento directo EBOOL (%Mi)
Variable declarada BOOL (Var:BOOL)	Sí	No
Variable declarada EBOOL (Var:EBOOL)	No	Sí

Compatibilidad

El tipo de datos EBOOL sigue estas reglas:

- Una variable de tipo EBOOL no puede emitirse como parámetro de entrada/salida de tipo BOOL.
- Las matrices de EBOOL no pueden emitirse como parámetros de tipo ANY de un FFB.
- Las matrices de BOOL y de EBOOL no son compatibles para la instrucción de asignación (regla idéntica para los parámetros de FFB).
- En Quantum:
 - Las variables localizadas de tipo EBOOL no pueden emitirse como parámetros de entradas/salidas de tipo EBOOL.
 - Las matrices de EBOOL no pueden emitirse como parámetros de un DFB.

Tipos enteros

Presentación

Los tipos Enteros permiten representar un valor en diferentes bases. que son:

- La base 10 (decimal) de forma predeterminada, en cuyo caso el valor llevará o no signo en función del tipo de entero
- La base 2 (binaria), en cuyo caso el valor no tiene signo y el prefijo es **2#**
- La base 8 (octal), en cuyo caso el valor no tiene signo y el prefijo es **8#**
- La base 16 (hexadecimal), en cuyo caso el valor no tiene signo y el prefijo es **16#**

NOTA: en la representación decimal, si el tipo elegido tiene signo, el valor puede ir precedido de el signo + o -, siendo el signo + opcional.

Tipo Entero (INT)

Tipo con signo y formato de 16 bits.

En la tabla se indica el rango de cada base.

Base	de...	a...
Decimal	-32768	32767
Binaria	2#1000000000000000	2#0111111111111111
Octal	8#100000	8#077777
Hexadecimal	16#8000	16#7FFF

Tipo Entero doble (DINT)

Tipo con signo y formato de 32 bits.

En la tabla se indica el rango de cada base.

Base	de...	a...
Decimal	-2147483648	2147483647
Binaria	2#10000000000000000000000000000000	2#01111111111111111111111111111111
Octal	8#200000000000	8#1777777777
Hexadecimal	16#80000000	16#7FFFFFFF

Tipo Entero sin signo (UINT)

Tipo sin signo y formato de 16 bits.

En la tabla se indica el rango de cada base.

Base	de...	a...
Decimal	0	65535
Binaria	2#0	2#1111111111111111
Octal	8#0	8#177777
Hexadecimal	16#0	16#FFFF

Tipo Entero doble sin signo (UDINT)

Tipo sin signo y formato de 32 bits.

En la tabla se indica el rango de cada base.

Base	de...	a...
Decimal	0	4294967295
Binaria	2#0	2#11111111111111111111111111111111
Octal	8#0	8#3777777777
Hexadecimal	16#0	16#FFFFFFFF

El tipo Time

Presentación

El tipo Time **T#** o **TIME#** se representa mediante un tipo entero doble sin signo (UDINT) (véase página 268).

Indica una duración en milisegundos que, aproximadamente, representa una duración máxima de 49 días.

Las unidades de tiempo permitidas para representar el valor son:

- días (**D**),
- horas (**H**),
- minutos (**M**),
- segundos (**S**) y
- milisegundos (**MS**).

Introducción de un valor

En esta tabla, se muestran dos posibles modos de introducción del valor máximo del tipo **Time**, según las unidades de tiempo permitidas.

Diagrama	Comentario
T#4294967295MS	Valor en milisegundos
T#4294967S_295MS	Valor en segundos/milisegundos
T#71582M_47S_295MS	Valor en minutos/segundos/milisegundos
T#1193H_2M_47S_295MS	Valor en horas/minutos/segundos/milisegundos
T#49D_17H_2M_47S_295MS	Valor en días/horas/minutos/segundos/milisegundos

8.2 Tipos de datos elementales (EDT) con formato BCD

Objeto

Esta sección describe los tipos de datos con formato BCD (Binary Coded Decimal), que son:

- el tipo Date,
- el tipo Time of Day (TOD) y
- el tipo Date and Time (DT).

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Descripción general de tipos de datos con formato BCD	272
El tipo Date	274
El tipo Time of Day (TOD)	275
El tipo Date and Time (DT)	276

Descripción general de tipos de datos con formato BCD

Introducción

Los tipos de datos con formato BCD pertenecen a la familia de datos elementales EDT (Elementary data type), que agrupa tipos de datos **denominados simples** y no compuestos (matrices, estructuras o bloques de funciones).

Recapitulación sobre el formato BCD

El formato Decimal codificado Binario (Binary coded Decimal) permite representar las cifras decimales **comprendidas entre 0 y 9** mediante un conjunto de cuatro bits (cuarteto).

En este formato, los cuatro bits que permiten codificar las cifras decimales tienen un rango de sus combinaciones inutilizado.

Tabla de correspondencias:

Decimal	Binario
0	0000
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000
9	1001
	1010 (inutilizado)
	1011 (inutilizado)
	1100 (inutilizado)
	1101 (inutilizado)
	1110 (inutilizado)
	1111 (inutilizado)

Ejemplo de codificación en un formato de 16 bits:

Valor decimal	2	4	5	0
2450				
Valor binario	0010	0100	0101	0000

Ejemplo de codificación en un formato de 32 bits:

Valor decimal 78993016	7	8	9	9	3	0	1	6
Valor binario	0111	1000	1001	1001	0011	0000	0001	0110

Tipos de datos con formato BCD

Existen tres tipos de datos:

Tipo	Designación	Escala (bits)	Valor predeterminado
DATE	Fecha	32	D#1990-01-01
TIME_OF_DAY	Hora del día	32	TOD#00:00:00
DATE_AND_TIME	Fecha y hora	64	DT#1990-01-01-00:00:00

El tipo Date

Presentación

El tipo **Date**, codificado en un formato de 32 bits, contiene la siguiente información:

- El año codificado en un campo de 16 bits (cuatro cuartetos de mayor valor),
- el mes codificado en un campo de 8 bits (dos cuartetos) y
- el día codificado en un campo de 8 bits (dos cuartetos de menor valor).

Representación de la fecha 2001-09-20 con formato BCD:

Año (2001)	Mes (09)	Día (20)
0010 0000 0000 0001	0000 1001	0010 0000

Reglas de sintaxis

La introducción del tipo **Date** es la siguiente: **D#**<Año>-<Mes>-<Día>

En la siguiente tabla, se indican los límites inferior y superior de cada campo.

Campo	Límites	Comentario
Año	[1990,2099]	
Mes	[01,12]	El 0 situado a la izquierda aparece siempre y se puede omitir al introducir los datos
Día	[01,31]	Para los meses 01\03\05\07\08\10\12
	[01,30]	Para los meses 04\06\09\11
	[01,29]	Para el mes 02 (años bisiestos)
	[01,28]	Para el mes 02 (años no bisiestos)

Ejemplo:

Introducción	Comentarios
D# 2001-1-1	El 0 de la izquierda del mes y el día se puede omitir
d# 1990-02-02	El prefijo puede ir en minúsculas

El tipo Time of Day (TOD)

Presentación

El tipo **Time of Day** , codificado en un formato de 32 bits, contiene la siguiente información:

- La hora codificada en un campo de 8 bits (dos cuartetos de mayor valor),
- los minutos codificados en un campo de 8 bits (dos cuartetos), y
- los segundos codificados en un campo de 8 bits (dos cuartetos).

NOTA: Los ocho bits de menor valor no se utilizan.

Representación con formato BCD de la hora del día 13:25:47:

Hora (13)	Minutos (25)	Segundos (47)	Byte de menor valor
0001 0011	0010 0101	0100 0111	Inutilizados

Reglas de sintaxis

La introducción del tipo Time of Day es la siguiente:

TOD#<Hora>:<Minutos>:<Segundos>

En la siguiente tabla, se indican los límites inferior y superior de cada campo.

Campo	Límites	Comentario
Hora	[00,23]	El 0 situado a la izquierda aparece siempre y se puede omitir al introducir los datos
Minuto	[00,59]	El 0 situado a la izquierda aparece siempre y se puede omitir al introducir los datos
Segundo	[00,59]	El 0 situado a la izquierda aparece siempre y se puede omitir al introducir los datos

Ejemplo:

Introducción	Comentario
TOD# 1:59:0	Los 0 de la izquierda de las horas y los segundos se puede omitir
tod# 23:10:59	El prefijo puede ir en minúsculas
Tod# 0:0:0	El prefijo puede ser combinado (minúsculas\mayúsculas)

El tipo Date and Time (DT)

Presentación

El tipo **Date and Time**, codificado en un formato de 64 bits, contiene la siguiente información:

- El año codificado en un campo de 16 bits (cuatro cuartetos de mayor valor),
- el mes codificado en un campo de 8 bits (dos cuartetos) y
- el día codificado en un campo de 8 bits (dos cuartetos),
- la hora codificada en un campo de 8 bits (dos cuartetos),
- los minutos codificados en un campo de 8 bits (dos cuartetos), y
- los segundos codificados en un campo de 8 bits (dos cuartetos).

NOTA: Los ocho bits de menor valor no se utilizan.

Ejemplo: Representación de la fecha y la hora 2000-09-20:13:25:47 con formato BCD:

Año (2000)	Mes (09)	Día (20)	Hora (13)	Minuto (25)	Segundos (47)	Byte de menor valor
0010 0000 0000 0000	0000 1001	0010 0000	0001 0011	0010 0101	0100 0111	Inutilizados

Reglas de sintaxis

La introducción del tipo **Date and Time** es la siguiente:

DT#<Año>-<Mes>-<Día>-<Hora>:<Minutos>:<Segundos>

En la siguiente tabla, se indican los límites inferior y superior de cada campo.

Campo	Límites	Comentario
Año	[1990,2099]	
Mes	[01,12]	El 0 situado a la izquierda aparece siempre y se puede omitir al introducir los datos
Día	[01,31]	Para los meses 01\03\05\07\08\10\12
	[01,30]	Para los meses 04\06\09\11
	[01,29]	Para el mes 02 (años bisiestos)
	[01,28]	Para el mes 02 (años no bisiestos)
Hora	[00,23]	El 0 situado a la izquierda aparece siempre y se puede omitir al introducir los datos
Minuto	[00,59]	El 0 situado a la izquierda aparece siempre y se puede omitir al introducir los datos
Segundo	[00,59]	El 0 situado a la izquierda aparece siempre y se puede omitir al introducir los datos

Ejemplo:

Introducción	Comentario
DT #2000-1-10-0:40:0	El 0 de la izquierda de los meses\horas\segundos se puede omitir
dt #1999-12-31-23:59:59	El prefijo puede ir en minúsculas
Dt #1990-10-2-12:02:30	El prefijo puede ser combinado (minúsculas\mayúsculas)

8.3 Tipos de datos elementales (EDT) con formato Real

Presentación del tipo de datos REAL

Introducción

Los tipos de datos con formato binario pertenecen a la familia de datos elementales EDT (Elementary data type), que agrupa tipos de datos **denominados simples** y no compuestos (matrices, estructuras o bloques de funciones).

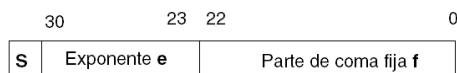
Recapitulación sobre el formato REAL

El formato REAL (coma flotante en el estándar ANSI/IEEE) se codifica en formato de 32 bits que corresponde a los números de coma flotante de un solo decimal.

Los 32 bits que representan el valor de coma flotante están organizados en tres campos distintos, que son:

- **S**, el bit de signo que puede tener el valor:
 - 0, para un número de coma flotante positivo.
 - 1, para un número de coma flotante negativo.
- **e**, el exponente codificado en un campo de 8 bits.
- **f**, la parte de coma fija codificada en un campo de 23 bits.

Representación:



El valor de la parte de coma fija (mantisa) está entre [0, 1] y se calcula mediante la fórmula siguiente:

$$f = \sum_{i=0}^{22} (2^{i-23} \cdot b_i)$$

b_i representa el valor del bit (0 ó 1) del rango correspondiente (i = 0 a 22)

Tipos de números que se pueden representar

Son los números:

- Normalizado
- No normalizado
- De valores infinitos
- Con valores +0 y -0

Esta tabla recoge los valores de los distintos campos según el tipo de número.

e	f	S	Tipo de número
[0, 255]	[0, 1]	0 ó 1	Normalizado
0	[0, 1]	0 ó 1	No normalizado DEN
255	0	0	+ infinito (INF)
255	0	1	- infinito (-INF)
255	[0,1] y bit 22 = 0	0 ó 1	SNAN
255	[0,1] y bit 22 = 0	0 ó 1	QNAN
0	0	0	+0
0	0	1	-0

NOTA: El estándar IEC 559 define dos clases de NAN (no un número): QNAN y SNAN.

- QNAN: es un NAN cuyo bit 22 está puesto a 1.
- SNAN: es un NAN cuyo bit 22 está puesto a 0.

Se comportan de la siguiente forma:

- QNAN no producir errores cuando aparezcan en operandos de una función o una expresión.
- SNAN producir un error cuando aparezca en operandos de una función o una expresión aritmética (consulte %SW17 (véase página 192) y %S18 (véase página 169)).

Esta tabla recoge la fórmula de cálculo del valor V del número de coma flotante:

Tipo de número de coma flotante	Valor V
Normalizado	$(-1)^S \times 2^{(e-127)} \times (1+f)$
No normalizado (DEN)	$(-1)^S \times 2^{-126} \times f$

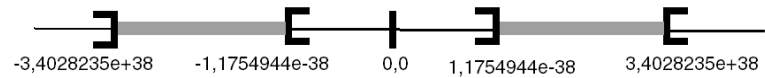
NOTA: Un número real entre -1,1754944e-38 y 1,1754944e-38 es un **DEN** no normalizado. Cuando un operando es un **DEN**, el resultado no está garantizado. Los bits %SW17 (véase página 192) y %S18 (véase página 169) sólo aumentan para M340. Los PLC Modicon M340 pueden emplear los operandos no normalizados, pero, debido al formato, sufren una pérdida de precisión. La transgresión por debajo de rango se señala en función de la operación sólo si el resultado es 0 (transgresión total) o cuando el resultado no es normalizado (transgresión gradual, con pérdida de precisión).

El tipo REAL

Presentación:

Tipo	Escala (bits)	Valor predeterminado
REAL	32	0,0

Rango de valores (zonas atenuadas):



Si el resultado de un cálculo es:

- Un número entre -1,1754944e-38 y 1,1754944e-38, es un **DEN**.
- Menor que -3,4028234e+38, aparece el símbolo **-INF** (para -infinito).
- Mayor que +3,4028234e+38, aparece el símbolo **INF** (para +infinito).
- Indefinido (raíz cuadrada de un número negativo), aparecerá el símbolo **NAN**.

Ejemplos

Representación el número de coma flotante -5.934113e-18.

Signo S	Exponente e	Parte de coma fija f
1	01000101	10110101110111000011101

Ejemplo de una entrada equivalente:

La entrada...	es igual a...	y...
+0,456	0,456	0,456
-1.32e12	-132E10	-.132e+13
1.0E+6	1.000.000.	1.e6

8.4 Tipo de datos elementales (EDT) con formato de cadena de caracteres

Descripción general de tipos de datos con formato de cadena de caracteres

Introducción

El tipo de datos con formato de cadena de caracteres pertenece a la familia de datos elementales EDT (Elementary data type), que incluye los tipos de datos **simples** y no derivados (tablas, estructuras o bloques de funciones).

Tipo de cadena de caracteres

El formato de cadena de caracteres permite representar una cadena de caracteres ASCII, en la que cada carácter está codificado con un formato de 8 bits.

Las características del tipo de cadena de caracteres son las siguientes:

- 16 caracteres predeterminados en la cadena (carácter de final de cadena excluido).
- Una cadena se compone de caracteres ASCII comprendidos entre 16#20 y 16#FF (representación hexadecimal).
- En una cadena vacía, el carácter de final de cadena (código ASCII "ZERO") es el primero de la cadena.
- El tamaño máximo de una cadena es de 65.535 caracteres.

El tamaño de la cadena de caracteres se puede optimizar a la hora de definir el tipo mediante el comando **STRING[<tamaño>]**, siendo <tamaño> un entero sin signo UINT que puede definir una cadena de 1 a 65.535 caracteres ASCII.

NOTA: Los caracteres ASCII 0 a 127 son comunes a todos los idiomas, pero los caracteres 128 a 255 dependen del idioma. Asegúrese de que el idioma de Unity Pro es el mismo que el del SO. Si son distintos, puede que la comunicación CHAR MODE se vea perturbada y que no se garantice un envío correcto de caracteres superiores a 127. En concreto, si el carácter "Parada al recibir" es superior a 127, no se tendrá en cuenta.

Reglas de sintaxis

La introducción va precedida y termina con el carácter `""` (código ASCII 16#27).

El signo \$ (dólar) es un carácter especial que, seguido de determinadas letras, indica:

- \$L o \$l, ir a la línea siguiente (avance de línea).
- \$N o \$n, ir al principio de la línea siguiente (línea nueva).
- \$P o \$p, ir a la página siguiente.
- \$R o \$r, retorno de carro.

- \$T o \$t, tabulación (Tab).
- \$\$, representa el carácter \$ en una cadena.
- \$', representa el carácter comilla en una cadena.

El usuario puede emplear la sintaxis \$nn para mostrar caracteres que no se deben imprimir en una variable STRING. Puede ser, por ejemplo, un retorno de carro (código ASCII 16#0D).

Ejemplos

Ejemplos de introducción:

Tipo	Entrada	Contenido de la cadena • representa el carácter final de la cadena * representa los bytes vacíos
STRING	'ABCD'	ABDC•***** (16 caracteres)
STRING[4]	'jean'	jean•
STRING[10]	'It's jean'	It's jean•*
STRING[5]	''	•*****
STRING[5]	'\$'	'•*****
STRING[5]	'el número'	el n.º•
STRING[13]	'0123456789'	0123456789•***
STRING[5]	'\$R\$L'	<cr><lf>•***
STRING[5]	'\$\$1.00'	\$1.00•

Declaración de variables de tipo STRING

Es posible declarar una variable de tipo STRING de dos maneras diferentes:

- STRING y
- STRING[<Número de elementos>]

El comportamiento es diferente en función del uso:

Tipo	Declaración de variables	Parámetro de entrada de FFB	Parámetro de salida de EF	Parámetro de salida de FB
STRING	Tamaño fijo: 16 caracteres	El tamaño es igual al tamaño real del parámetro de entrada.	El tamaño es igual al tamaño real del parámetro de entrada.	Tamaño fijo de 16 caracteres
STRING[<n>]	Tamaño fijo: n caracteres	El tamaño es igual al tamaño real del parámetro de entrada con límite de n caracteres.	EF escribe un máximo de n caracteres.	FB escribe un máximo de n caracteres.

Cadenas y pin ANY

Cuando se utiliza una variable de tipo `STRING` como parámetro de tipo `ANY`, se recomienda comprobar que el tamaño de la variable es inferior al tamaño máximo declarado.

Ejemplo:

Utilización de `STRING` en la función `SEL` (selector).

```
String1: STRING[8]
```

```
String2: STRING[4]
```

```
String3: STRING[4]
```

```
String1:= 'AAAAAAA'
```

```
String3:= 'CC'
```

Caso 1:

```
String2:= 'BBBB'
```

(* el tamaño de la cadena es igual que el tamaño máximo declarado *)

```
String1:= SEL(FALSE, String2, String3);
```

(* el resultado será: 'BBBBAAAA' *)

Caso 2:

```
String2:= 'BBB'
```

(* el tamaño de la cadena es inferior al tamaño máximo declarado*)

```
String1:= SEL(FALSE, String2, String3);
```

(* el resultado será: 'BBB' *)

8.5 Tipos de datos elementales (EDT) con formato de cadena de bits

Objeto

En esta sección, se describe el tipo de datos con formato de cadena de bits. que son:

- Tipo Byte
- Tipo Word
- Tipo Dword

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Descripción general de los tipos de datos con formato de cadena de bits	285
Tipos de cadena de bits	286

Descripción general de los tipos de datos con formato de cadena de bits

Introducción

Los tipos de datos con formato de cadena de bits pertenecen a la familia de datos elementales EDT (Elementary data type), que reagrupa los tipos de datos **denominados simples** y no compuestos (matrices, estructura o bloque de funciones).

Recapitulación sobre el formato de cadena de bits

La particularidad de este formato es que el conjunto de los bits que lo componen no representa un valor numérico, sino una combinación de bits separados.

Los datos que pertenecen a los tipos de este formato se pueden representar en tres bases: que son:

- Hexadecimal (16#)
- Octal (8#)
- Binaria (2#)

Tipos de datos con formato de cadena de bits

Existen tres tipos de datos:

Tipo	Escala (bits)	Valor predeterminado
BYTE	8	0
WORD	16	0
DWORD	32	0

Tipos de cadena de bits

Tipo Byte

El tipo Byte está codificado en un formato de 8 bits.

En la siguiente tabla, se indican los límites inferior y superior de las bases que lo representan.

Base	Límite inferior	Límite superior
Hexadecimal	16#0	16#FF
Octal	8#0	8#377
Binaria	2#0	2#1111111

Ejemplos de representación:

Contenido del dato	Representación en una de las bases
00001000	16#8
00110011	8#63
00110011	2#110011

Tipo Word

El tipo Word está codificado en un formato de 16 bits.

En la siguiente tabla, se indican los límites inferior y superior de las bases que lo representan.

Base	Límite inferior	Límite superior
Hexadecimal	16#0	16#FFFF
Octal	8#0	8#177777
Binaria	2#0	2#1111111111111111

Ejemplos de representación:

Contenido del dato	Representación en una de las bases
0000000011010011	16#D3
1010101010101010	8#125252
0000000011010011	2#11010011

Tipo Dword

El tipo Dword está codificado en un formato de 32 bits.

En la siguiente tabla, se indican los límites inferior y superior de las bases que lo representan.

Base	Límite inferior	Límite superior
Hexadecimal	16#0	16#FFFFFFFF
Octal	8#0	8#3777777777
Binaria	2#0	2#11111111111111111111111111111111

Ejemplos de representación:

Contenido de los datos	Representación en una de las bases
00000000000010101101110011011110	16#ADCDE
00000000000000010000000000000000	8#200000
00000000000010101011110011011110	2#10101011110011011110

8.6 Tipos de datos derivados (DDT/IODDT)

Objeto

Esta sección describe los tipos de datos derivados, que son:

- Tablas (DDT)
- Estructuras
 - Estructuras relativas a los datos de entradas\salidas (IODDT) y
 - estructuras relativas a otros datos (DDT).

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Matrices	289
Estructuras	292
Descripción general de la familia de tipos de datos derivados (DDT)	293
DDT: normas de asignación	296
Descripción general de los tipos de datos derivados de entradas/salidas (IODDT)	299

Matrices

¿Qué es una matriz?

Es un elemento de datos que contiene un **conjunto** de datos **del mismo tipo**, como por ejemplo:

- Datos elementales (EDT),
por ejemplo:
 - Un grupo de palabras BOOL
 - Un grupo de palabras de valor entero UINT
 - etc.
- Datos derivados (DDT),
por ejemplo:
 - Un grupo de tablas WORD
 - Un grupo de estructuras
 - etc.

Características

Una matriz se caracteriza por dos parámetros:

- Uno que define su organización (dimensiones de matriz).
- Otro que define el tipo de datos que contiene.

NOTA: La organización más compleja es la matriz con **seis dimensiones**.

La sintaxis que incluye estos dos parámetros es la siguiente:

ARRAY[<dimensiones de matriz>]OF <tipo de datos>

<dimensión 1>,<dimensión 2>,<dimensión n>

<limite minimo>..

<limite minimo> **estrictamente menor** que <limite máximo>

Definición e instancias de una matriz

Definición de un tipo de matriz:

```
X: ARRAY[1..10] OF BOOL
```

Instancias de una matriz

```
Tab_1: X
Tab_2: ARRAY[1..10] OF BOOL
```

Las instancias Tab_1 y Tab_2 son del mismo tipo y tienen la misma dimensión; la única diferencia entre ambas se contempla durante la instancia:

- El tipo Tab_1 adopta el nombre X.
- Es necesario definir el tipo Tab_2 (tabla sin nombre).

NOTA: resulta útil dar un nombre al tipo de manera que, al llevar a cabo cualquier modificación, sólo será necesario efectuar esta acción una vez; de lo contrario, deberán efectuarse tantas modificaciones como instancias haya.

Ejemplos

Esta tabla presenta las instancias de matrices de diferentes dimensiones:

Entrada	Comentarios
Tab_1: ARRAY[1..2] OF BOOL	Matriz de una dimensión con dos palabras booleanas.
Tab_2: ARRAY[-10..20] OF WORD	Matriz de una dimensión con 31 estructuras de tipo WORD (estructura definida por el usuario).
Tab_3: ARRAY[1..10, 1..20] OF INT	Matrices de dos dimensiones con enteros 10 x 20.
Tab_4: ARRAY[0..2, -1..1, 201..300, 0..1] OF REAL	Matrices de cuatro dimensiones con reales 3 x 3 x 100 x 2.

 ADVERTENCIA

ÍNDICE DE MATRIZ IRRECONOCIBLE

Muchas funciones (READ_VAR y WRITE_VAR, por ejemplo) no reconocen el índice de una matriz de palabras que empieza por un número diferente de 0. En caso de utilizar este índice, las funciones tomarán en cuenta el número de palabras de la matriz, pero no el índice de inicio establecido en la definición de la matriz.

Si no se siguen estas instrucciones pueden producirse lesiones personales graves o mortales o daños en el equipo.

Acceso a un elemento de datos en las matrices Tab_1 y Tab_3:

```
Tab_1[2]
```

```
;Para acceder al segundo elemento
```

```
Tab_3[4][18]
```

```
;Para acceder al elemento decimocctavo de la cuarta submatriz
```

Reglas de asignación entre matrices

Debemos distinguir las cuatro matrices siguientes:

```
Tab_1:ARRAY[1..10] OF INT
```

```
Tab_2:ARRAY[1..10] OF INT
```

```
Tab_3:ARRAY[1..11] OF INT
```

```
Tab_4:ARRAY[101..110] OF INT
```

```
Tab_1:=Tab_2; Asignación autorizada
```

```
Tab_1:=Tab_3; Asignación rechazada (no es compatible con IEC)
```

```
Tab_1:=Tab_4; Asignación rechazada (no es compatible con IEC)
```

Estructuras

¿Qué es una estructura?

Es un dato que contiene un **conjunto** de datos **de distinto tipo**, tales como:

- Un conjunto de BOOL, WORD, UINT, etc., (estructura EDT).
- Un conjunto de matrices (estructura de DDT).
- Un conjunto de REAL, DWORD, matrices, etc., (estructura de EDT y DDT).

NOTA: Es posible llevar a cabo estructuras intercaladas (DDT intercaladas) en ocho niveles. **No se autorizan las estructuras (DDT) recursivas.**

Características

Una estructura se compone de datos, cada uno de los cuales se caracteriza por:

- Un tipo.
- Un nombre, que permite identificarlo.
- Un comentario (opcional) que describe su función.

Definición de un tipo de estructura:

```
IDENT
    Apellido: STRING[12]
    Nombre:  STRING[16]
    Edad:   UINT

;La estructura de tipo IDENT contiene un dato de tipo UINT y dos
datos de tipo STRING
```

Definición de dos instancias de datos de la estructura de tipo IDENT:

```
Persona_1: IDENT
Persona_2: IDENT

;Las instancias Persona_1 y Persona_2 son de tipo Estructura IDENT
```

Acceso a un dato de una estructura

Acceso a un dato de la instancia Persona_1 de tipo IDENT:

```
Persona_1.Nombre ;Permite acceder al nombre de la Persona_1

Persona_1.Edad   ;Permite acceder a la edad de la Persona_1
```

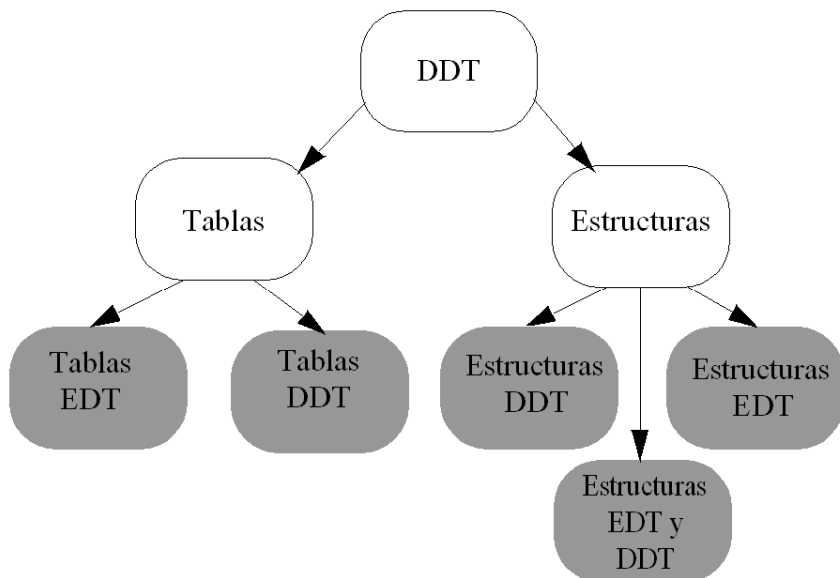
Descripción general de la familia de tipos de datos derivados (DDT)

Introducción

La familia DDT (tipos de datos derivados) incluye tipos de datos "**derivados**" como:

- tablas,
- estructuras.

Ilustración:



Características

Un elemento de datos perteneciente a la familia DDT está compuesto de:

- El nombre de tipo (*véase página 258*) (máximo 32 caracteres) definido por el usuario (no es obligatorio para las tablas pero se recomienda). (*véase página 289*)
- El tipo (estructura o tabla).
- Un comentario opcional (un máximo de 1024 caracteres). Los caracteres autorizados corresponden a los códigos ASCII 32 a 255.

- La descripción (en el caso de una estructura) de los elementos:
 - el nombre del elemento (*véase página 258*) (32 caracteres máximo),
 - el tipo de elemento,
 - un comentario opcional (un máximo de 1.024 caracteres).describiendo su función. Los caracteres autorizados corresponden a los códigos ASCII 32 a 255.
- Información del tipo:
 - número de la versión del tipo,
 - fecha de la última modificación del código o de las variables internas o de las variables de la interfaz,
 - un archivo descriptivo opcional (32.767 caracteres), que describa el bloque de funciones y sus diferentes modificaciones.

NOTA: El tamaño total de la tabla o de la estructura no debe superar 64 Kbytes.

Ejemplos

Definición de tipos

```
COORD
    X: INT
    Y: INT
;Estructura de tipo COORD

SEGMENT
    Origen: COORD
    Destino: COORD
;Estructura de tipo SEGMENT que contiene 2 estructuras
de tipo COORD

OUTLINE: ARRAY[0..99] OF SEGMENT
;Tabla de tipo OUTLINE que contiene 100 estructuras de
tipo SEGMENT

DRAW
    Color: INT
    Ancla: COORD
    Modelo: ARRAY[0..15,0..15] OF WORD
    Contorno: OUTLINE
;Estructura de tipo DRAW
```

Acceso a los datos de una instancia de estructura de tipo DRAW

```
Cartoon: DRAW
;Instancia de estrucutra de tipo DRAW

Cartoon.Modelo[15,15]
;Acceso al último elemento de datos de la tabla Modelo de
la estructura Cartoon

Cartoon.Contorno[0].Origen.X
;Acceso al elemento de datos X de la estructura COORD que
pertenece a la primera estructura SEGMENT de la tabla
Contorno.
```

DDT: normas de asignación

Introducción

Los DDT se almacenan en la memoria del PLC en función del orden en el que se introducen sus elementos.

No obstante, se deben considerar las siguientes reglas.

Principio de Premium y Quantum

El principio de almacenamiento de Premium y Quantum es el siguiente:

- Los elementos se almacenan en el orden en el que se introducen en la estructura.
- El elemento básico es el byte (ajuste de datos en los bytes de memoria).
- Cada elemento cuenta con una norma de alineación:
 - La de los tipos `BOOL` y `BYTE` se pueden ajustar bien en bytes pares o bien en impares.
 - Todos los demás tipos elementales se ajustan en bytes pares.
 - Las estructuras y las tablas se alinean en función de la norma de alineación de los tipos `BOOL` y `BYTE` si contienen únicamente elementos `BOOL` y `BYTE`; en caso contrario, se alinean con los bytes pares de la memoria.

ADVERTENCIA

RIESGO DE INCOMPATIBILIDAD TRAS LA CONVERSIÓN DE CONCEPT

Con la aplicación de programación **Concept**, las estructuras de datos no gestionan ningún cambio de desplazamiento (cada elemento se coloca uno detrás del otro en la memoria, sea cual sea su tipo). Por tanto, se recomienda comprobarlo todo, en particular la coherencia de los datos al utilizar **DDT** ubicados en la memoria de señal (riesgo de cambios) o funciones para comunicarse con otros dispositivos (transferencias con un tamaño distinto a los programados en **Concept**).

Si no se siguen estas instrucciones pueden producirse lesiones personales graves o mortales o daños en el equipo.

Principio de Modicon M340

El principio de almacenamiento de PLC Modicon M340 es el siguiente:

- Los elementos se almacenan en el orden en el que se introducen en la estructura.
- El elemento básico es el byte.
- Una norma de alineación y función del elemento:
 - Los de tipo `BOOL` y `BYTE` se alinean en bytes pares o impares.
 - Los de tipo `INT`, `WORD` y `UINT` se alinean en bytes pares.

- Los de tipo DINT, UDINT, REAL, TIME, DATE, TOD, DT y DWORD se alinean en palabras dobles.
- Las estructuras y las tablas se alinean según las normas de sus elementos.

ADVERTENCIA

Intercambios incorrectos entre un proyecto de Modicon M340 y un proyecto Premium o Quantum.

Compruebe si la estructura de los datos intercambiados dispone de las mismas alineaciones en los dos proyectos.

De lo contrario, los datos no se intercambiarán correctamente.

Si no se siguen estas instrucciones pueden producirse lesiones personales graves o mortales o daños en el equipo.

NOTA: Es posible que la alineación de datos no se mantenga igual cuando el proyecto se transfiera del simulador de Unity Pro a un PLC M340. Conviene comprobar la estructura de los datos del proyecto.

NOTA: Unity Pro (véase *Unity Pro, Modalidades de funcionamiento*,) indica dónde parece diferir la alineación. Compruebe las instancias correspondientes en el editor de datos. Consulte la página Ajustes del proyecto (véase *Unity Pro, Modalidades de funcionamiento*,) para saber cómo habilitar esta opción.

Ejemplos

En la tabla siguiente se proporcionan algunos ejemplos de estructuras de datos. En los siguientes ejemplos, los DDT de la estructura se direccionan a %MWi. El primer byte de la palabra corresponde a los ocho bits de menor valor y el segundo byte de la palabra corresponde a los ocho bits de mayor valor.

En todas las estructuras siguientes, la primera variable se asigna a la dirección %MW100:

Primera dirección de memoria		Descripción de la estructura
Modicon M340	Premium	Para_PWM1
%MW100 (primer byte)	%MW100 (primer byte)	t_period: TIME
%MW102 (primer byte)	%MW102 (primer byte)	t_min: TIME
%MW104 (primer byte)	%MW104 (primer byte)	in_max: REAL
		Mode_TOTALIZER
%MW100 (primer byte)	%MW100 (primer byte)	hold: BOOL
%MW100 (segundo byte)	%MW100 (segundo byte)	rst: BOOL

Primera dirección de memoria		Descripción de la estructura
		Info_TOTALIZER
%MW100 (primer byte)	%MW100 (primer byte)	outc: REAL
%MW102 (primer byte)	%MW102 (primer byte)	cter: UINT
%MW103 (primer byte)	%MW103 (primer byte)	done: BOOL
%MW103 (segundo byte)	%MW103 (segundo byte)	Reservado para la alineación

En la tabla siguiente, se proporcionan dos ejemplos de estructuras de datos con matrices:

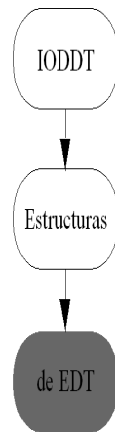
Primera dirección de memoria		Descripción de la estructura
Modicon M340	Premium	EHC105_Out
%MW100 (primer byte)	%MW100 (primer byte)	Quit: BYTE
%MW100 (segundo byte)	%MW100 (segundo byte)	Control: ARRAY [1..5] OF BYTE
%MW104 (primer byte)	%MW103 (primer byte)	Final: ARRAY [1..5] OF DINT
		CPCfg_ex
%MW100 (primer byte)	%MW100 (primer byte)	Profile_type: INT
%MW101 (primer byte)	%MW101 (primer byte)	Interp_type: INT
%MW102 (primer byte)	%MW102 (primer byte)	Nb_of_coords: INT
%MW103 (primer byte)	%MW103 (primer byte)	Nb_of_points: INT
%MW104 (primer byte)	%MW104 (primer byte)	reserved: ARRAY [0..4] OF BYTE
%MW106 (segundo byte)	%MW106 (segundo byte)	Reservado para la alineación de la variable Master_offset en bytes pares
%MW108 (primer byte)	%MW107 (primer byte)	Master_offset: DINT
%MW110 (primer byte)	%MW109 (primer byte)	Follower_offset: INT
%MW111 (palabra completa)	-	Reservado para la alineación

Descripción general de los tipos de datos derivados de entradas/salidas (IODDT)

Presentación

Los tipos de datos derivados de entradas/salidas IODDT (Input Output Derived Data Type) **están predefinidos por el fabricante** y contienen objetos de lenguaje de la familia EDT pertenecientes al canal de un módulo de función específica.

Figura:



Los tipos IODDT son estructuras cuyo tamaño (número de elementos que los componen) depende del canal o del módulo de entradas/salidas que representan.

Un módulo de entradas/salidas determinado puede tener más de un IODDT.

La diferencia con una estructura clásica es que:

- la estructura IODDT está predefinida por el fabricante y
- los elementos que componen la estructura IODDT no tienen una asignación de memoria contigua, sino una dirección específica en el módulo.

Ejemplos

Estructura IODDT para un canal de entrada/salida de un módulo analógico

```

ANA_IN_GEN      ;Estructura de tipo ANA_IN_GEN
Value:INT;Valor de la entrada
Err:  BOOL;Error de canal
  
```

Acceso a datos de una instancia de tipo ANA_IN_GEN:

```
Cistern_Level: ANA_IN_GEN  
;Instancia de tipo ANA_IN_GEN que corresponde, por ejemplo,  
a un captador de nivel de cubeta
```

```
Cistern_Level.Value;Lectura del valor de entrada del canal  
Cistern_Level.Err ;Lectura del bit de error del canal
```

Acceso mediante direccionamiento directo:

Si se trata del canal 0 del módulo 2 del bastidor 0, tenemos:

```
Cistern_Level      corresponde a %CH0.2.0  
Cistern_Level.Value corresponde a %IW0.2.0.0  
Cistern_Level_Err  corresponde a %IO.2.0.ERR
```

8.7 Tipos de datos de bloques de función (DFB\EFB)

Objeto

En esta sección, se describen los tipos de datos de bloques de función. que son:

- Bloques de funciones del usuario (DFB)
- Bloques de funciones elementales (EFB)

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Descripción general de las familias de tipos de datos de bloques de funciones	302
Características de los tipos de datos de bloques de funciones (EFB\DFB)	304
Características de elementos que pertenecen a bloques de funciones	306

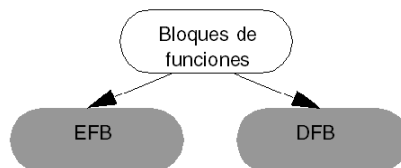
Descripción general de las familias de tipos de datos de bloques de funciones

Introducción

Las familias de tipos de datos de bloques de funciones son:

- La familia de tipo bloque de función elemental (EFB) (véase página 253) y
- la familia de tipo bloque de función de usuario (DFB) (véase página 253).

Figura:

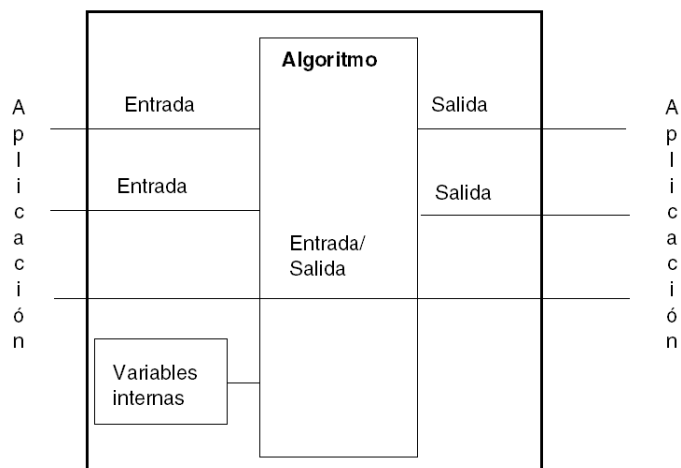


Los bloques de funciones son entidades que contienen:

- variables de entradas y de salidas que sirven de interfaz con la aplicación,
- un algoritmo de procesamiento que utiliza las variables de entradas y ofrece información de las variables de salidas, y
- variables internas privadas y públicas utilizadas por el algoritmo de procesamiento.

Figura

Bloque de funciones:



Bloque de función del usuario (DFB)

Los tipos de bloques de funciones del usuario (Derived Function Blocks) los desarrolla el usuario con uno o varios lenguajes (en función del número de secciones). Estos lenguajes son:

- Lenguaje de contactos,
- lenguaje literal estructurado,
- lenguaje lista de instrucciones y
- lenguaje de bloques funcionales FBD.

Un tipo de DFB puede tener una o varias instancias, cada una de las cuales se identifica mediante un nombre (símbolo) y posee los tipos de datos de DFB.

Bloque de funciones elemental (EFB)

Los bloques de funciones elementales (EFB) los proporciona el fabricante y están programados en lenguaje C.

El usuario puede crear sus propios EFB, para lo que debe disponer de la herramienta de software opcional "**SDKC**".

Un tipo de EFB puede tener una o varias instancias, cada una de las cuales se identifica mediante un nombre (símbolo) y posee los datos del tipo de EFB.

Características de los tipos de datos de bloques de funciones (EFB\DFB)

Definición del tipo

El tipo de un bloque de funciones EFB o DFB se define mediante:

- El nombre del tipo (*véase página 258*), definido por el usuario para los DFB.
- Un comentario opcional. Los caracteres autorizados corresponden a los códigos ASCII 32 a 255.
- Los datos de interfaz con la aplicación:
 - Las entradas, a las que no se puede acceder en lectura/escritura desde la aplicación, pero que lee el código del bloque de funciones.
 - Las entradas/salidas, a las que no se puede acceder en lectura/escritura desde la aplicación, pero que lee y escribe el código del bloque de funciones.
 - Las salidas, a las que se puede acceder en lectura desde la aplicación y que lee y escribe el código del bloque de funciones.
- Los datos internos:
 - Públicos, a los que se puede acceder en lectura/escritura desde la aplicación, y que lee y escribe el código del bloque de funciones.
 - Privados, a los que no se puede acceder desde la aplicación y que lee y escribe el código del bloque de funciones.
- El código:
 - Para los DFB, es el usuario quien los escribe en lenguaje de automatismo (literal estructurado, lista de instrucciones, lenguaje de contactos, lenguaje de bloques funcionales) y está estructurado en una sola sección si la opción IEC está activa, o bien puede estructurarse en varias secciones si esta opción está inactiva.
 - Para los EFB, se escribe en lenguaje C.
- Información como, por ejemplo:
 - El número de versión del tipo,
 - La fecha de la última modificación del código, de las variables internas o de las variables de interfaces y
 - una ficha descriptiva facultativa (32767 caracteres) que describe la función del bloque y sus diferentes modificaciones.

Características

En la tabla se indican las características de los elementos que componen un tipo:

Elemento	EFB	DFB
Nombre	32 caracteres	32 caracteres
Comentario	1024 caracteres	1024 caracteres
Datos de entradas	32 máximo	32 máximo
Datos de entradas/salidas	32 máximo	32 máximo

Elemento	EFB	DFB
Datos de salidas	32 máximo	32 máximo
Número de interfaces (Entradas + Salidas + Entradas/Salidas)	32 máximo (2)	32 máximo (2)
Datos públicos	Sin límites (1)	Sin límites (1)
Datos privados	Sin límites (1)	Sin límites (1)
Lenguaje de programación	Lenguaje C	Lenguaje: ● Literal estructurado, ● lista de instrucciones, ● de contactos o ● de bloques funcionales.
Sección		Una sección se define mediante: ● Un nombre (32 caracteres como máximo), ● una condición de validación, ● un comentario (256 caracteres como máximo) y ● una protección: ● Sin, ● lectura o ● lectura/escritura. Una sección no puede acceder a las variables declaradas en la aplicación, excepto: ● Las palabras dobles de sistema %SDi, ● las palabras de sistema %SWi y ● los bits del sistema %Si.

(1): El tamaño de la memoria del PLC es la única limitación.

(2): No se tiene en cuenta la entrada EN ni la salida ENO.

Características de elementos que pertenecen a bloques de funciones

Definición de elemento

Cada elemento (datos de interfase o internos) se define mediante lo siguiente:

- Un nombre (*véase página 258*) (de 32 caracteres como máximo), definido por el usuario.
- Un tipo, que puede pertenecer a las familias siguientes:
 - Tipos de datos elementales (EDT)
 - Tipos de datos derivados (DDT)
 - Tipos de datos de bloques de funciones (EFB/DFB)
- Un comentario opcional (de 1.024 caracteres como máximo). Los caracteres autorizados corresponden a los códigos ASCII 32 a 255.
- Un valor inicial
- Un derecho de acceso del programa de aplicación (secciones de la aplicación o sección perteneciente a los DFB; consulte “Definición del tipo de bloques de funciones (interfase y variables internas)” (*véase página 304*).
- Un derecho de acceso de las solicitudes de comunicación
- Un flag de copia de seguridad de variables públicas.

Tipos de datos autorizados para un elemento perteneciente a un DFB

A continuación se detallan los tipos de datos autorizados:

Elemento de DFB	Tipos de EDT	Tipos de DDT				ANY...	Tipos de bloques de funciones
		IODDT	Tablas sin nombre	ANY_ARRAY	Otros		
Datos de entrada	Sí	Sin	Sí	Sí	Sí	Sí (2)	No
Datos de entrada/salida	Sí (1)	Sí	Sí	Sí	Sí	Sí (2)	No
Datos de salida	Sí	No	Sí	No	Sí	Sí (2) (3)	No
Datos públicos	Sí	No	Sí	No	Sí	No	No
Datos privados	Sí	No	Sí	No	Sí	No	Sí

(1): no autorizado para los datos estáticos de tipo EBOOL utilizados en los PLC Quantum.

(2): no autorizado para los datos de tipo BOOL y EBOOL.

(3): debe completarse durante la ejecución del DFB, y no se utiliza fuera del DFB.

Tipos de datos autorizados para un elemento perteneciente a un EFB

A continuación se detallan los tipos de datos autorizados:

Elemento de EFB	Tipos de EDT	Tipos de DDT				ANY...	Tipos de bloques de funciones
		IODDT	Tablas sin nombre	ANY_ARRAY	Otros		
Datos de entrada	Sí	No	No	Sí	Sí	Sí (1)	No
Datos de entrada/salida	Sí	Sí	No	Sí	Sí	Sí (1)	No
Datos de salida	Sí	No	No	No	Sí	Sí (1) (2)	No
Datos públicos	Sí	No	No	No	Sí	No	No
Datos privados	Sí	No	No	No	Sí	No	Sí

(1): no autorizado para los datos de tipo BOOL y EBOOL.

(2): debe completarse durante la ejecución del EFB, y no se utiliza fuera del EFB.

Valores iniciales para un elemento perteneciente a un DFB

Esta tabla especifica si los valores iniciales pueden introducirse desde la definición de tipo DFB o la instancia DFB:

Elemento de DFB	Desde el tipo DFB	Desde la instancia DFB
Datos de entrada (que no son del tipo ANY...)	Sí	Sí
Datos de entrada (del tipo ANY...)	No	No
Datos de entrada/salida	No	No
Datos de salida (que no son del tipo ANY...)	Sí	Sí
Datos de salida (del tipo ANY...)	No	No
Datos públicos	Sí	Sí
Datos privados	Sí	No

Valores iniciales para un elemento perteneciente a un EFB

Esta tabla especifica si los valores iniciales pueden introducirse desde la definición de tipo EFB o la instancia EFB:

Elemento de EFB	Desde el tipo EFB	Desde la instancia DFB
Datos de entrada (que no son del tipo ANY..., consulte generic data types (véase página 309))	Sí	Sí
Datos de entrada (del tipo ANY...)	No	No
Datos de entrada/salida	No	No
Datos de salida (que no son del tipo ANY...)	Sí	Sí
Datos de salida (del tipo ANY...)	No	No
Datos públicos	Sí	Sí
Datos privados	Sí	No

** ADVERTENCIA**

ÍNDICE DE UNA MATRIZ NO RECONOCIDA POR EFB Y DFB

Los EFB y DFB no reconocen el índice de una matriz de palabras que empieza por un número distinto de 0. Si se utiliza este índice, los EFB y DFB tendrán en cuenta el número de palabras de la matriz, pero no el índice de inicio establecido en la definición de la matriz.

Si no se siguen estas instrucciones pueden producirse lesiones personales graves o mortales o daños en el equipo.

8.8 Tipos de datos genéricos (GDT)

Descripción general de los tipos de datos genéricos

Presentación

Los tipos de datos genéricos son conjuntos de tipos de datos clásicos (EDT, DDT) que tienen por objeto determinar la compatibilidad entre estos tipos de datos clásicos.

Dichos conjuntos se identifican por el prefijo "ANY_ARRAY", pero estos prefijos no pueden utilizarse en ningún caso para instanciar los datos.

Sus campos de utilización hacen referencia a las familias del tipos de datos de bloques de funciones (EFB/DFB) y funciones elementales (EF) para definir los tipos de datos compatibles con sus interfaces para:

- entradas,
- entradas/salidas y
- salidas.

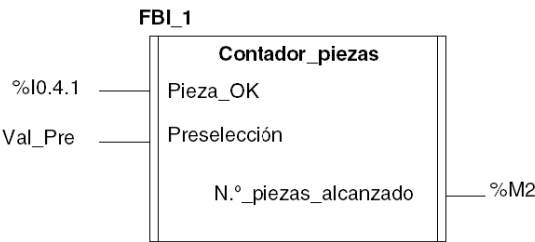
Tipos de datos genéricos disponibles

Los tipos de datos genéricos disponibles en Unity Pro son los siguientes:

- ANY_ARRAY_WORD
- ANY_ARRAY_UINT
- ANY_ARRAY_UDINT
- ANY_ARRAY_TOD
- ANY_ARRAY_TIME
- ANY_ARRAY_STRING
- ANY_ARRAY_REAL
- ANY_ARRAY_INT
- ANY_ARRAY_EBOOL
- ANY_ARRAY_DWORD
- ANY_ARRAY_DT
- ANY_ARRAY_DINT
- ANY_ARRAY_DATE
- ANY_ARRAY_BYTE
- ANY_ARRAY_BOOL

Ejemplo

Se da el siguiente DFB:



El parámetro de entrada **Preajuste** puede definirse como GDT.

NOTA: Los objetos permitidos para los diferentes parámetros se definen en esta matriz (véase página 603).

8.9 Tipos de datos pertenecientes a las gráficas de funciones secuenciales (SFC)

Descripción general de los tipos de datos de la familia de gráficas de funciones secuenciales

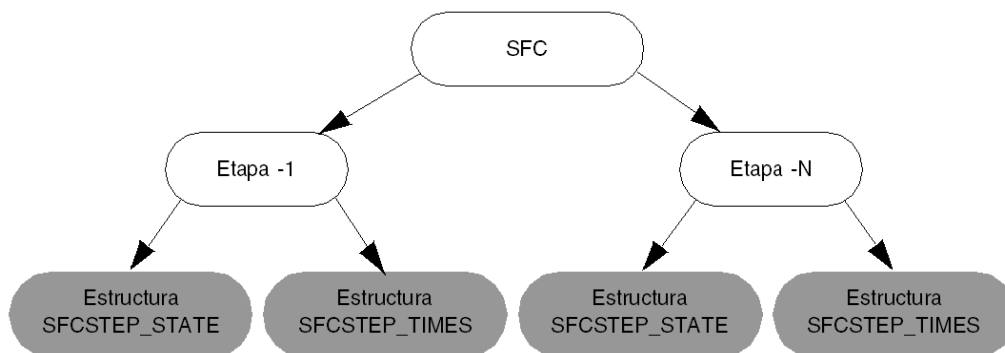
Introducción

La familia de los tipos de datos de diagramas funcionales en secuencia SFC (Sequential function chart) agrupa tipos de datos **denominados compuestos**, tales como estructuras que restablecen las propiedades y el estado del gráfico (Chart) y las acciones que lo componen.

Cada etapa está representada por dos estructuras, que son:

- La estructura **SFCSTEP_STATE** y
- la estructura **SFCSTEP_TIMES**.

Figura:



NOTA: Los dos tipos de estructuras **SFCSTEP_STATE** y **SFCSTEP_TIMES** están también vinculadas a cada macroetapa del diagrama funcional en secuencia.

Definición de la estructura de tipo **SFCSTEP_STATE**

Esta estructura reúne todos los datos relativos al estado de la etapa o de la macroetapa.

Estos datos son:

- **x**: dato elemental (EDT) de tipo BOOL que contiene el valor TRUE cuando la etapa está activa.
- **t**: tipo de dato elemental (EDT) TIME que contiene el tiempo de actividad del paso. Al estar inactivo, el valor del paso se mantiene hasta la siguiente activación.
- **tminErr**: dato elemental (EDT) de tipo BOOL que contiene el valor TRUE si el tiempo de actividad de la etapa es inferior al tiempo de actividad mínimo programado.
- **tmaxErr**: dato elemental (EDT) de tipo BOOL que contiene el valor TRUE si el tiempo de actividad de la etapa es superior al tiempo de actividad máximo programado.

Se puede acceder a estos datos a partir de la aplicación en sólo lectura.

Definición de la estructura de tipo SFCSTEP_TIMES

Esta estructura reúne todos los datos relativos a los parámetros del tiempo de ejecución de la etapa o de la macroetapa.

Estos datos son:

- **delay**: dato elemental (EDT) de tipo TIME que define el tiempo de retardo de exploración de la transición situado hacia abajo de la etapa activa.
- **tmin**: tipo de dato elemental (EDT) TIME que contiene el valor mínimo durante el que se debe ejecutar el paso. Si no se respeta este valor, el tmin.Err de datos pasa al valor TRUE.
- **tmax**: tipo de dato elemental (EDT) TIME que contiene el valor máximo durante el que se debe ejecutar el paso. Si no se respeta este valor, el tmax.Err de datos pasa al valor TRUE.

Sólo se puede acceder a estos datos a partir del editor del SFC.

Sintaxis de acceso a un dato de la estructura SFCSTEP_STATE

Los nombres de instancias de esta estructura corresponden a los nombres de las etapas o macroetapas del diagrama funcional en secuencia

Sintaxis	Comentario
Nombre_Etapa.x	Permite conocer el estado de la etapa (activa/inactiva)
Nombre_Etapa.t	Permite conocer el tiempo de activación en curso o total de la etapa
Nombre_Etapa.tminErr	Permite conocer si el tiempo mínimo de activación de la etapa es inferior al tiempo programado en Nombre-Etapa.tmin
Nombre_Etapa.tmaxErr	Permite conocer si el tiempo máximo de activación de la etapa es superior al tiempo programado en Nombre-Etapa.tmax

8.10 Compatibilidad entre los tipos de datos

Compatibilidad entre tipos de datos

Introducción

A continuación se presentan las diferentes reglas de compatibilidad entre tipos **en el interior** de cada una de las siguientes familias:

- la familia de tipos de datos elementales (EDT),
- la familia de tipos de datos derivados (DDT),
- la familia de tipos de datos genéricos (GDT).

Familia de tipos de datos elementales (EDT)

La familia de tipos de datos elementales (EDT) contiene subfamilias, a saber:

- la subfamilia de tipos de datos con formato binario,
- la subfamilia de tipos de datos con formato BCD,
- la subfamilia de tipos de datos con formato real,
- la subfamilia de tipos de datos con formato de cadena de caracteres,
- la subfamilia de tipos de datos con formato de cadena de bits.

No hay compatibilidad entre dos tipos de datos, cualesquiera que sean, aunque pertenezcan a la misma subfamilia.

Familia de tipos de datos derivados (DDT)

La familia de tipos de datos derivados (DDT) contiene subfamilias, a saber:

- la subfamilia de tipo matrices,
- la subfamilia de tipo estructuras:
 - estructuras relativas a los datos de entradas/salidas (IODDT),
 - estructuras relativas a los demás datos.

Reglas relativas a las estructuras:

Dos estructuras son compatibles si sus elementos:

- tienen los mismos nombres,
- son del mismo tipo,
- están organizados siguiendo el mismo orden.

Se dan cuatro tipos de estructuras:

```
ELEMENT_1
  My_Element: INT
  Other_Element: BOOL
;Estructura de tipo ELEMENT_1

ELEMENT_2
  My_Element: INT
  Other_Element: BOOL
;Estructura de tipo ELEMENT_2

ELEMENT_3
  Elemento: INT
  Other_Element: BOOL
;Estructura de tipo ELEMENT_3

ELEMENT_4
  Other_Element: BOOL
  My_Element: INT
;Estructura de tipo ELEMENT_4
```

Compatibilidad entre los tipos de estructuras

Tipos	ELEMENT_1	ELEMENT_2	ELEMENT_3	ELEMENT_4
ELEMENT_1		SÍ	NO	NO
ELEMENT_2	SÍ		NO	NO
ELEMENT_3	NO	NO		NO
ELEMENT_4	NO	NO	NO	

Reglas relativas a las tablas

Dos tablas son compatibles si:

- las dimensiones y su organización son idénticas,
- cada dimensión correspondiente es del mismo tipo.

Se dan cinco tipos de tablas:

```
TAB_1: ARRAY[10..20]OF INT
;Tabla unidimensional de tipo TAB_1

TAB_2: ARRAY[20..30]OF INT
;Tabla unidimensional de tipo TAB_2

TAB_3: ARRAY[20..30]OF INT
;Tabla unidimensional de tipo TAB_3

TAB_4: ARRAY[20..30]OF TAB_1
;Tabla unidimensional de tipo TAB_4

TAB_5: ARRAY[20..30,10..20]OF INT
;Tabla de bidimensional de tipo TAB_5
```

Compatibilidad entre los tipos de tablas:

El tipo...	y el tipo...	son...
TAB_1	TAB_2	incompatibles
TAB_2	TAB_3	compatibles
TAB_4	TAB_5	compatibles
TAB_4[25]	TAB_5[28]	compatibles

Familia de tipos de datos genéricos (GDT)

La familia de tipos de datos genéricos (GDT) se compone de conjuntos organizados de forma jerárquica que contienen tipos de datos pertenecientes a las familias:

- tipos de datos elementales (EDT),
- tipos de datos derivados (DDT).

Reglas:

Un tipo de datos clásico es compatible con los tipos de datos genéricos que le son jerárquicos.

Un tipo de datos genérico es compatible con los tipos de datos genéricos que le son jerárquicos.

Ejemplo:

El tipo INT es compatible con los tipos ANY_INT, ANY_NUM o ANY_MAGNITUDE.

El tipo INT no es compatible con los tipos ANY_BIT ni ANY_REAL.

El tipo genérico ANY_INT es compatible con el tipo ANY_NUM.

El tipo genérico ANY_INT no es compatible con el tipo ANY_REAL.

Contenido del capítulo

Este capítulo describe las instancias de datos y sus características.

Estas instancias pueden ser:

- Instancias de datos sin localizar,
- instancias de datos localizados e
- instancias de datos con direccionamiento directo.

Contenido de este capítulo

Este capítulo contiene los siguiente apartados:

Apartado	Página
Instancias de tipos de datos	318
Atributos de instancias de datos	322
Instancias de datos con direccionamiento directo	324

Instancias de tipos de datos

Introducción

¿Qué es una instancia de tipos de datos? (véase página 255)

La instancia de un tipo de datos se identifica mediante:

- **un nombre (símbolo)**, en cuyo caso se dice que el dato está **sin localizar**, ya que su asignación de memoria no se define, sino que el sistema la efectúa automáticamente
- **un nombre (símbolo) y una dirección topológica** que define el fabricante, en cuyo caso se dice que el dato está **localizado**, ya que se conoce su asignación de memoria, y
- **una dirección topológica** que define el fabricante, en cuyo caso se dice que el dato tiene **direccionamiento directo**, ya que se conoce su asignación de memoria.

Instancias de datos sin localizar

Las instancias de datos sin localizar se gestionan desde el sistema operativo del autómatas sin que el usuario conozca su ubicación física.

Las instancias de datos sin localizar se definen a partir de tipos de datos que pueden pertenecer a la familia:

- Tipos de datos elementales (EDT),
- Tipos de datos derivados (DDT)
- Tipos de datos de bloques de funciones (EFB/DFB)

Ejemplos:

```
Var_1: BOOL
;Instancia de la familia EDT de tipo booleano con asignación de
memoria de 1 byte

Var_2: UDINT
;Instancia de la familia EDT de tipo entero doble sin signo con
asignación de memoria de 4 bytes

Var_3: ARRAY[1..10]OF INT
;Instancia de la familia DDT de tipo tabla con asignación de memoria
de 20 bytes

COORD
  X: INIC
  Y: INIC
Var_4: COORD
;Instancia de la familia DDT de estructura tipo COORD con asignación
de memoria de 4 bytes
```

Instancias de datos localizados

Las instancias de datos localizadas poseen un emplazamiento predefinido en la memoria del autómatas, que el usuario conoce.

Las instancias de datos localizadas se definen a partir de tipos de datos que pueden pertenecer a la familia:

- Tipos de datos elementales (EDT),
- Tipos de datos derivados (DDT)
- Tipos de datos de gráficas de funciones secuenciales(SFC)

NOTA: En Premium/Atrium las instancias de de datos localizados de tipo doble (%MD<i>, %KD<i>) o flotantes (%MF<i>, %KF<i>) deben localizarse mediante un tipo entero (%MW<i>, %KW<i>). Únicamente, los objetos de E/S permiten localizar las instancias de tipo (%MD<i>, %KD<i>, %QD, %ID, %MF<i>, %KF<i>, %QF, %IF) utilizando su dirección topológica (ejemplo %MD0.6.0.11, %MF0.6.0.31).

NOTA: En Modicon M340, las instancias de datos localizados de tipo doble (%MD<i>, %KD<i>) o flotantes (%MF<i>, %KF<i>) no están disponibles.

NOTA: En Modicon M340, el valor de índice (i) debe ser par (*véase página 296*) en las instancias de datos localizados de tipo doble (%MW y %KW).

Ejemplos:

```

Var_1 : EBOOL AT %M100
;Instancia de la familia EDT de tipo booleano (con asignación de
memoria de 1 byte) predefinida en %M100.

Var_2: BOOL AT %I2.1.0.ERR
;Instancia de la familia EDT de tipo booleano (con asignación de
memoria de 1 byte) predefinida en %I2.1.0.ERR.

Var_3: INT AT %MW10
;Instancia de la familia EDT de tipo entero (con asignación de
memoria de 2 bytes) predefinida en %MW10.

Var_4: : DINT AT %MW1
;Prohibida en Modicon M340. Las instancias de datos localizados de
tipo doble deben tener una dirección topológica par (%MW2,
%MW10.....).

Var_5: WORD AT %MW10
;Instancia de la familia EDT de tipo WORD (con asignación de memoria
de 2 bytes) predefinida en %MW10.

Var_6: ARRAY[1..10]OF INT AT %MW50
;Instancia de la familia EDT de tipo tabla (con asignación de memoria
de 20 bytes) predefinida en %MW50.

COORD
  X: INIC
  Y: INIC

Var_7: COORD AT %MW20
;Instancia de la familia DDT de estructura tipo COORD (con asignación
de memoria de 4 bytes) predefinida a partir %NW20

Var_8: DINT AT %MD0.6.0.11
;Instancia de la familia EDT de tipo DINT (asignación de memoria de
4 bytes) predefinida a partir de la dirección topológica del objeto
de E/S del módulo específico de la aplicación.

Var_9: REAL AT %MF0.6.0.31
;Instancia de la familia EDT de tipo REAL (asignación de memoria de
4 bytes) predefinida a partir de la dirección topológica del objeto
de E/S del módulo específico de la aplicación.

```


NOTA: Las instancias de tipos de datos de diagrama funcional en secuencia (SFC) se crean en el momento de insertarse en el programa de aplicación con un nombre predeterminado que el usuario puede modificar.

Instancias de datos con direccionamiento directo

Las instancias de datos con direccionamiento directo poseen un emplazamiento predefinido en la memoria del autómatas o en un módulo de función específica, que el usuario conoce.

Las instancias de datos de direccionamiento directo se definen a partir de tipos pertenecientes a la familia del tipo de datos elementales (EDT).

Ejemplos de instancias de datos con direccionamiento directo:

Internas	Constantes	Sistema	Entradas/Salidas	Red
%Mi		%Si	%Q, %I	
%MWi	%KWi	%SWi	%QW, %IW	%NW
%MDi (1)	%KDi (1)		%QD, %ID	
%MFi (1)	%KFi (1)		%QF, %IF	
Leyenda				
(1) No disponible en Modicon M340				

Atributos de instancias de datos

Presentación

Los atributos de una instancia de datos es información que la define.

Dicha información es la siguiente:

- Su nombre (*véase página 258*) (excepto para las instancias de datos de direccionamiento directo (*véase página 324*)).
- Su dirección topológica (excepto para las instancias de tipos de datos sin localizar).
- Su tipo de datos, que puede pertenecer a la familia:
 - Tipo de datos elementales (EDT)
 - Tipos de datos derivados (DDT)
 - Tipos de datos de bloques de funciones (EFB/DFB)
 - Tipos de datos de gráfico de funciones secuencial (SFC)
- Un comentario descriptivo opcional (con un máximo de 1.024 caracteres). Los caracteres autorizados corresponden a los códigos ASCII de 32 a 255.

Nombre de una instancia de datos

Se trata del símbolo (32 caracteres como máximo) elegido por el usuario y que permite diferenciar la instancia. Debe ser único.

Algunos nombres no se pueden utilizar, por ejemplo:

- Palabras clave utilizadas en los lenguajes textuales,
- nombres de sección de programa,
- nombres de tipos de datos predefinidos o elegidos por el usuario (estructuras, matrices),
- nombres de tipos de datos DFB/EFB predefinidos o elegidos por el usuario y
- nombres de funciones elementales (EF) predefinidas o elegidas por el usuario.

Nombre de instancias que pertenecen a la familia SFC

Los nombres de las instancias se introducen implícitamente mientras que el usuario dibuja su gráfico de función secuencial. Son nombres predeterminados que facilita el fabricante y que el usuario puede modificar.

Nombres predeterminados facilitados por el fabricante:

Objeto SFC	Nombre
Etapas	S_<nombre sección>_<n.º de etapa>
Etapas de macroetapa	S_<nombre sección>_<n.º de macroetapa>_<n.º de etapa>
Macroetapa	MS_<nombre sección>_<n.º de etapa>

Objeto SFC	Nombre
Macroetapa intercalada	MS_<nombre sección>_<n.º de macroetapa>_<n.º de etapa>
Etapas de entrada de la macroetapa	S_IN<nombre de sección>_<n.º de macroetapa>
Etapas de salida de la macroetapa	S_OUT<nombre de sección>_<n.º de macroetapa>
Transición	T_<nombre sección>_<n.º transición>
Transición de macroetapa	T_<nombre sección>_<n.º de macroetapa>_<n.º transición>

Nombres de instancias que pertenecen a la familia de bloques de funciones

Los nombres de las instancias se introducen implícitamente mientras que el usuario introduce las instancias en las secciones del programa de aplicación. Son nombres predeterminados que facilita el fabricante y que **el usuario puede modificar**.

Sintaxis de los nombres predeterminados facilitados por el fabricante:

```
FB_<nombre del tipo de bloque de funciones>_<n.º de instancia>
```

NOTA: El nombre de la instancia no incluye el de la sección en la que se utiliza, ya que se puede emplear en diferentes secciones de la aplicación.

Acceso a un elemento de una instancia de la familia DDT

La sintaxis de acceso es la siguiente:

```
Caso de los datos de tipo estructuras
<Nombre de la instancia>.<Nombre del elemento>
```

```
Caso de los datos de tipo matrices
<Nombre de la instancia>[Índice del elemento]
```

Regla:

El tamaño máximo de la sintaxis de acceso es de 1024 caracteres como máximo, y los límites posibles de un tipo de datos derivados son los siguientes:

- 10 niveles de intercalación (matrices/estructuras),
- 6 dimensiones por matriz y
- 4 dígitos (cifras) para definir el índice del elemento de una matriz.

Instancias de datos con direccionamiento directo

Presentación

¿Qué es una instancia de datos de direccionamiento directo? (véase página 321)

Sintaxis de acceso

La sintaxis de una instancia de datos de direccionamiento directo se define mediante el símbolo % seguido de un **prefijo de localización de memoria** y, en determinados casos, de información adicional.

El prefijo de localización de memoria puede ser:

- **M**, para las variables internas
- **K**, para las constantes (Premium y Modicon M340)
- **S**, para las variables de sistema
- **N**, para las variables de redes
- **I**, para las variables de entradas
- **Q**, para las variables de salidas

Caso de las variables internas %M

Sintaxis de acceso:

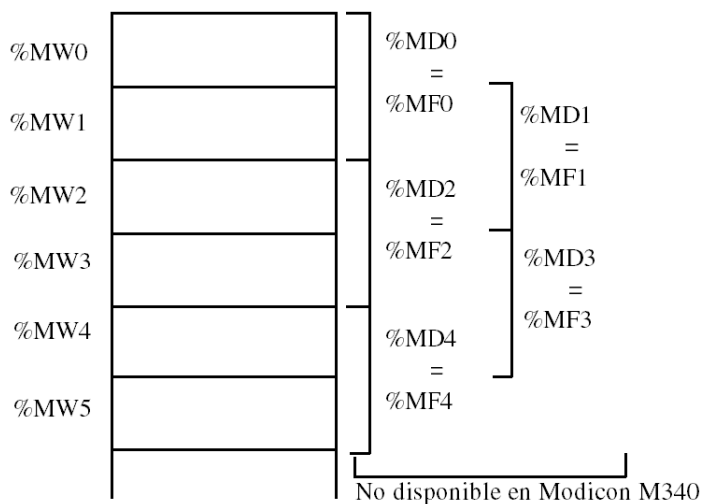
	Sintaxis	Formato	Ejemplo	Derecho de acceso al programa
Bit	%M<i> o %MX<i>	3 bits (EBOOL)	%M1	L/E
Palabra	%MW<i>	16 bits (INT)	%MW10	L/E
Bit extraído de palabra	%MW<i>.<j>	1 bit (BOOL)	%MW15.5	L/E
Palabra doble	%MD<i> (1)	32 bits (DINT)	%MD8	L/E
Real (flotante)	%MF<i> (1)	32 bits (REAL)	%MF15	L/E
Leyenda				
(1): No disponible en Modicon M340.				

<i> representa el número de la instancia (empieza en 0 para Premium y en 1 para Quantum).

En Modicon M340, la instancia de tipo doble (palabra doble) o flotante (real) debe localizarse mediante un %MW de tipo entero. El índice <i> del %MW debe ser par.

NOTA: Los datos %M<i> o %MX<i> detectan los flancos y gestionan el forzado.

Organización de la memoria:



NOTA: La modificación de %MW<i> conlleva las modificaciones de %MD<i> y %MF<i> correspondientes.

Constantes %K

Sintaxis de acceso:

	Sintaxis	Formato	Derecho de acceso al programa
Constante de palabra	%KW<i>	16 bits (INT)	L
Constante de palabra doble	%KD<i> (1)	32 bits (DINT)	L
Constante Real (flotante)	%KF<i> (1)	32 bits (REAL)	L
Leyenda			
(1): No disponible en Modicon M340.			

<i> representa el número de la instancia.

NOTA: La organización de la memoria es idéntica a la de las variables internas. Debe tenerse en cuenta que estos datos no están disponibles en los PLC Quantum.

Caso de las constantes %I

Sintaxis de acceso:

	Sintaxis	Formato	Derecho de acceso al programa
Constante bit	%I<i>	3 bits (EBOOL)	L
Constante de palabra	%IW<i>	16 bits (INT)	L

<i> representa el número de la instancia.

NOTA: Estos datos sólo están disponibles en los PLC Quantum y Momentum.

Caso de las variables de sistema %S

Sintaxis de acceso:

	Sintaxis	Formato	Derecho de acceso al programa
Bit	%S<i> o %SX<i>	1 bit (BOOL)	L/E o L
Palabra	%SW<i>	32 bits (INT)	L/E o L

<i> representa el número de la instancia.

NOTA: La organización de la memoria es idéntica a la de las variables internas. Los datos %S<i> o %SX<i> no son de detección de flancos y no gestionan el forzado.

Caso de las variables de redes %N

Estas variables contienen información que se debe intercambiar entre varios programas de aplicación a través de la red de comunicación.

Sintaxis de acceso:

	Sintaxis	Formato	Derecho de acceso al programa
Palabra común	%NW<n>.<s>.<d>	16 bits (INT)	L/E o L
Bit extraído de palabra	%NW<n>.<s>.<d>.<j>	1 bit (BOOL)	L/E o L

<n> representa el número de red.

<s> representa el número de la estación.

<d> representa el número del dato.

<j> representa el rango del bit en la palabra.

Caso de las variables de entradas/salidas

Estas variables están contenidas en los módulos de función específica.

Sintaxis de acceso:

	Sintaxis	Ejemplo	Derecho de acceso al programa
Estructura de entradas/salidas (IODDT)	%CH<@mod>.<c>	%CH4.3.2	L
Entradas %I			
Bit de error del módulo de tipo BOOL	%I<@mod>.MOD.ERR	%I4.2.MOD.ERR	L
Bit de error del canal de tipo BOOL	%I<@mod>.<c>.ERR	%I4.2.3.ERR	L
Bit de tipo BOOL o EBOOL	%I<@mod>.<c>	%I4.2.3	L
	%I<@mod>.<c>.<d>	%I4.2.3.1	L
Palabra de tipo INT	%IW<@mod>.<c>	%IW4.2.3	L
	%IW<@mod>.<c>.<d>	%IW4.2.3.1	L
Palabra doble de tipo DINT	%ID<@mod>.<c>	%ID4.2.3	L
	%ID<@mod>.<c>.<d>	%ID4.2.3.2	L
Real (flotante) de tipo REAL	%IF<@mod>.<c>	%IF4.2.3	L
	%IF<@mod>.<c>.<d>	%IF4.2.3.2	L
Salidas %Q			
Bit de tipo EBOOL	%Q<@mod>.<c>	%Q4.2.3	L/E
	%Q<@mod>.<c>.<d>	%Q4.2.30.1	L/E
Palabra de tipo INT	%QW<@mod>.<c>	%QW4.2.3	L/E
	%QW<@mod>.<c>.<d>	%QW4.2.3.1	L/E
Palabra doble de tipo DINT	%QD<@mod>.<c>	%QD4.2.3	L/E
	%QD<@mod>.<c>.<d>	%QD4.2.3.2	L/E
Real (flotante) de tipo REAL	%QF<@mod>.<c>	%QF4.2.3	L/E
	%QF<@mod>.<c>.<d>	%QF4.2.3.2	L/E
Variables %M (Premium)			
Palabra de tipo INT	%MW<@mod>.<c>	%MW4.2.3	L/E
	%MW<@mod>.<c>.<d>	%MW4.2.3.1	L/E
Palabra doble de tipo DINT	%MD<@mod>.<c>	%MD4.2.3	L/E
	%MD<@mod>.<c>.<d>	%MD4.2.3.2	L/E
Real (flotante) de tipo REAL	%MF<@mod>.<c>	%MF4.2.3	L/E
	%MF<@mod>.<c>.<d>	%MF4.2.3.2	L/E
Constantes %K (Modicon M340 y Premium)			

	Sintaxis	Ejemplo	Derecho de acceso al programa
Palabra de tipo INT	%KW<@mod>.<c>	%KW4.2.3	L
	%KW<@mod>.<c>.<d>	%KW4.2.3.1	L
Palabra doble de tipo DINT	%KD<@mod>.<c>	%KD4.2.3	L
	%KD<@mod>.<c>.<d>	%KD4.2.3.12	L
Real (flotante) de tipo REAL	%KF<@mod>.<c>	%KF4.2.3	L
	%KF<@mod>.<c>.<d>	%KF4.2.3.12	L

<@mod = \.<e>\<r>.<m>

 número de bus (omitido en caso de estación local).

<e> número del punto de conexión del dispositivo (omitido en caso de estación local, el punto de conexión se denomina también «drop» para los usuarios de autómatas Quantum).

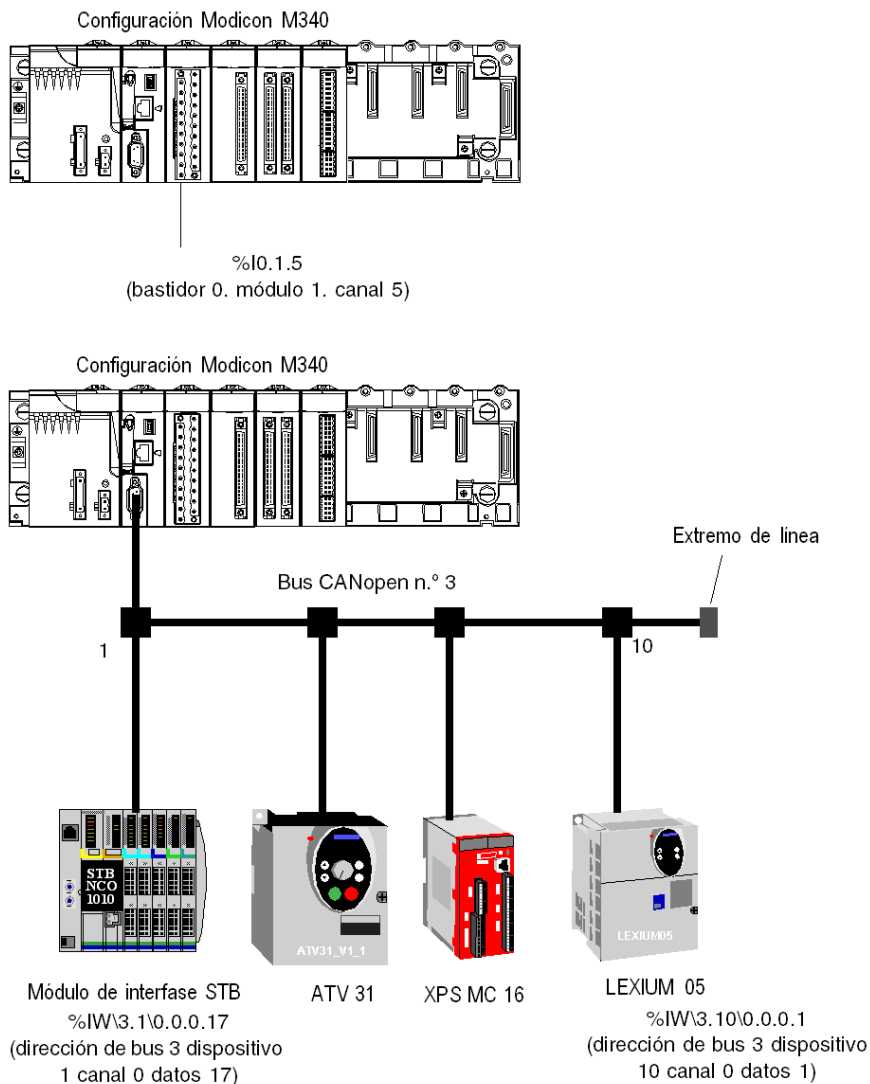
<r> número de bastidor.

<m> emplazamiento del módulo

<c> número de canal (de 0 a 999) o palabra reservada MOD.

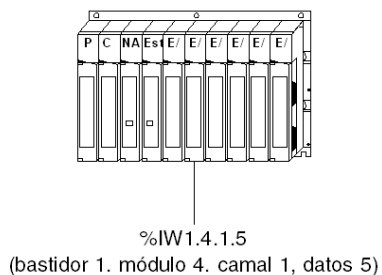
<d> número del dato (de 0 a 999) o palabra reservada ERR (opcional si el valor es 0). En Modicon M340 <d> siempre es par.

Ejemplos: estación local y estación en bus para los PLC Modicon M340.

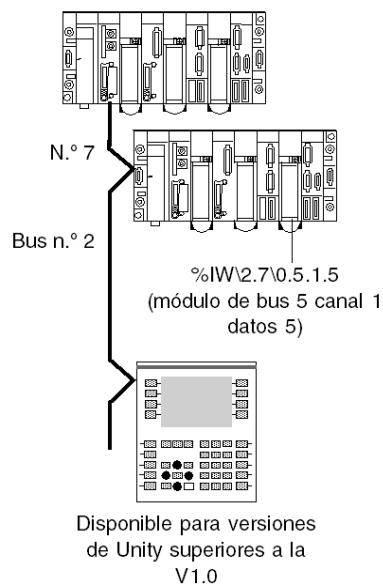
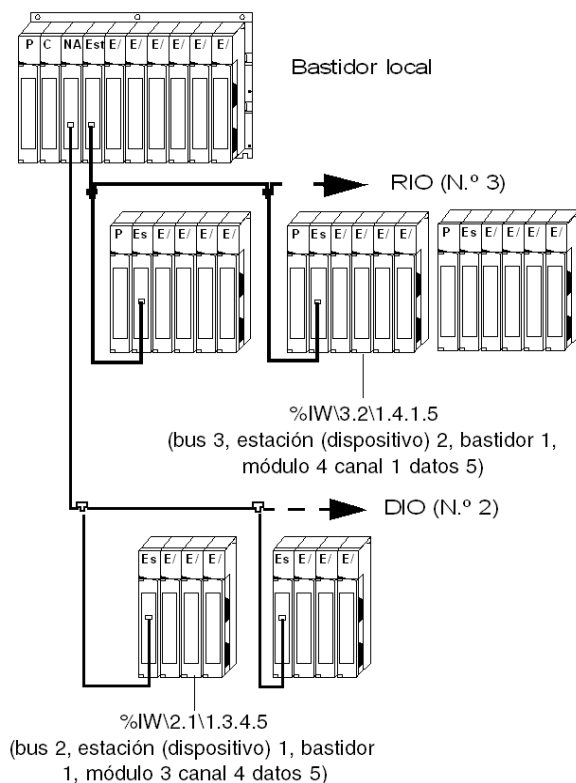
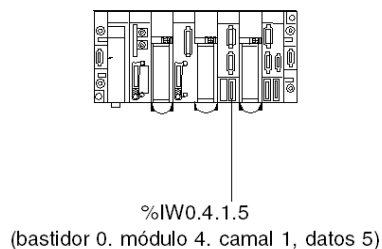


Ejemplos: estación local y estación en bus para los PLC Quantum y Premium.

Ejemplo de Quantum



Ejemplo de Premium



Contenido del capítulo

Este capítulo describe las referencias de instancias de datos.

Estas referencias pueden ser:

- Referencias por valores,
- referencias por nombres o
- referencias por direcciones.

Contenido de este capítulo

Este capítulo contiene los siguiente apartados:

Apartado	Página
Referencias de instancias de datos por valores	332
Referencias de instancias de datos por nombre	334
Referencias de instancias de datos por direcciones	337
Reglas de denominación de los datos	341

Referencias de instancias de datos por valores

Introducción

¿Qué es una referencia de instancia de datos? (véase *página 257*)

Presentación

Una referencia de instancia de datos por valor es una instancia que no tiene nombre (símbolo) ni dirección topológica.

Se trata de un **valor inmediato** que se puede asignar a una instancia de tipo de datos pertenecientes a la familia EDT.

La norma IEC 1131 permite los valores inmediatos en instancias de datos de tipo:

- Booleano
 - BOOL
 - EBOOL
- Entero
 - INT
 - UINT
 - DINT
 - UDINT
 - TIME
- Reales
 - REAL
- Fecha y hora
 - DATE
 - DATE AND TIME
 - TIME OF DAY
- Cadenas de caracteres
 - STRING

El software de programación amplía la norma añadiendo los **tipos de cadenas de bits**.

- BYTE
- WORD
- DWORD

Ejemplos de valores inmediatos

En la tabla se asocian valores inmediatos con tipos de instancias

Valor inmediato	Tipo de instancia
'Soy una cadena de caracteres'	STRING
T#1s	TIME
D#2000-01-01	DATE
TOD#12:25:23	TIME_OF_DAY
DT#2000-01-01-12:25:23	DATE_AND_TIME
16#FFF0	WORD
UINT#16#9AF (valor tipificado)	UINT
DWORD#16#FFFF (valor tipificado)	DWORD

Referencias de instancias de datos por nombre

Introducción

¿Qué es una referencia de instancia de datos? (véase página 257)

Referencias de instancias de la familia EDT

El usuario elige un nombre (símbolo) que permite acceder a la instancia del dato:

```
Estado_Válvula: BOOL

Umbral_Alto: EBOOL AT %M10

Contenido_Tolva: UINT

Temperatura_Horno: INT AT %MW100

Valor_Codificador: WORD
```

Referencias de instancias de la familia DDT

Caso de las matrices:

El usuario elige un nombre (símbolo) que permite acceder a la instancia del dato:

```
Dados dos tipos de matrices:

Gama_Colores ARRAY[1..15]OF STRING
Vehículos ARRAY[1..100]OF Gama_Colores
;

Vehículo: Vehículos
Nombre de instancia de la matriz de tipo Vehículos

Vehículo[11, 5]
Acceso al vehículo 11 de color correspondiente al 5º elemento de
la matriz Gama_Colores
```

Caso de las estructuras:

El usuario elige un nombre (símbolo) que permite acceder a la instancia del dato:

Dadas las dos estructuras:

ADRESSE

Calle: STRING[20]
Código_Postal: UDINT
Ciudad: STRING: [20]

IDENT

Nombre: STRING[15]
Nombre: STRING[15]
Edad: UINT
Fecha_Nacimiento: DATE
Localidad: ADRESSE

Persona_1:IDENT

;Nombre de instancia de la estructura de tipo IDENT

Persona_1.Edad

Acceso a la edad de la Persona_1

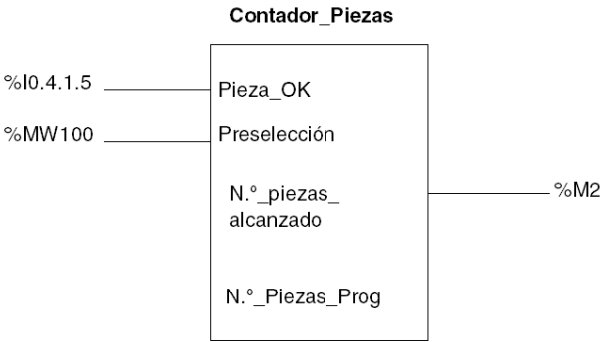
Persona_1.Localidad.Ciudad

;Acceso a la localidad donde reside la Persona_1

Referencias de instancias de las familias DFB\EFB

El usuario elige un nombre (símbolo) que permite acceder a la instancia del dato.

Dado el tipo de DFB:



Contador_Tornillos: Contador_Piezas
Nombre de instancia del bloque de tipo Contador_Piezas

Contador_Tornillos.N.º_Piezas_Prog
Acceso a la variable pública N.º_Piezas_Prog

Contador_Tornillos.N.º_piezas_alcanzado
Acceso a la interfaz de salida N.º_piezas_alcanzado

Referencias de instancias de datos por direcciones

Introducción

¿Qué es una referencia de instancia de datos? (véase página 257)

Presentación

Una referencia de instancia de datos por direcciones sólo es posible en determinadas instancias de datos que pertenecen a la **familia EDT**. Estas instancias son:

- variables internas (%M<i>, %MW<i>, %MD<i>, %MF<i>)
- constantes (%KW<i>, %KD<i>, %KF<i>)
- entradas/salidas (%I<dirección>, %Q<dirección>)

NOTA: Las instancias %MD<i>, %MF<i>, %KD<i> y %KF<i> no están disponibles en Modicon M340.

Referencia por direccionamiento directo

El direccionamiento se considera directo cuando la dirección de la instancia es fija, es decir, cuando se define al escribir el programa.

Ejemplos:

%M1

Acceso al primer bit de la memoria

%MW12

Acceso a la duodécima palabra de la memoria

%MD4

Acceso a la cuarta palabra doble de la memoria

%KF100

;Acceso a la centésima palabra de coma flotante de la memoria

%Q0.4.0.5

Acceso al quinto bit del módulo de salida en la posición 4 del bastidor 0

Referencias por dirección indexada

El direccionamiento se considera indexado cuando la dirección de la instancia se completa con un índice.

El índice se define mediante:

- un valor perteneciente a un tipo entero
- una expresión aritmética compuesta de tipos enteros

Una variable indexada siempre tiene una equivalencia no indexada:

```
%MW<i>[<índice>] <=> %MW<j>
```

Las reglas para calcular <j> son las siguientes:

Objeto<i>[índice]	Objeto<j>
%M<i>[índice]	<j>=<i> + <índice>
%MW<i>[índice]	<j>=<i> + <índice>
%KW<i>[índice]	<j>=<i> + <índice>
%MD<i>[índice]	<j>=<i> + (<índice> x 2)
%KD<i>[índice]	<j>=<i> + (<índice> x 2)
%MF<i>[índice]	<j>=<i> + (<índice> x 2)
%KF<i>[índice]	<j>=<i> + (<índice> x 2)

Ejemplos:

```
%MD6[10] <=> %MD26  
  
%MW10[My_Var+8] <=> %MW20 (con My_Var=2)
```

Durante la compilación del programa, un control comprueba que:

- el índice no sea negativo
- el índice no supere el espacio de memoria asignado para cada uno de estos tres tipos de datos

Bits de extracción de palabra

Se puede extraer uno de los 16 bits de las palabras simples (%MW, %SW; %KW, %IW, %QW).

La dirección de la instancia se completa con el rango del bit extraído (<j>).

```
WORD<i> . <j>
```

Ejemplos:

```
%MW10.4
Bit n.º 4 de la palabra %MW10

%SW8.4
Bit n.º 4 de la palabra de sistema %SW8

%KW100.14
Bit n.º 14 de la constante KW100

%QW0.5.1.0.10
Bit n.º 10 de la palabra 0 del canal 1 del módulo de salida
5 del bastidor 0
```

Tablas de bits y de palabras

Se trata de una serie de objetos adyacentes (bits o palabras) de mismo tipo y de longitud definida.

```
OBJECT<i> :L
```

Presentación de las tablas de bits:

Tipo	Dirección	Acceso de escritura
Bits de entrada de E/S binarias	%Ix.i:L	No
Bits de salida de E/S binarias	%Qx.i:L	Sí
Bits internos	%Mi:L	Sí

Presentación de las tablas de palabras:

Tipo	Dirección	Acceso de escritura
Palabras internas	%MWi:L %MDi:L %MFi:L	Sí
Palabras constantes	%KWi:L %KDi:L %KFi:L	No
Palabras de sistema	%SW50:4	Sí

Ejemplos:

%M2:65

Define una tabla EBOOL de %M2 a %M66

%MW125:30

Define una tabla INT de %MW125 a %MW 154

Reglas de denominación de los datos

Introducción

En una aplicación, el usuario elige un nombre para:

- definir un tipo de datos,
- instanciar un dato (símbolo) o
- identificar una sección.

Se han definido algunas reglas para evitar que se produzcan conflictos. Para ello, es preciso diferenciar los distintos ámbitos de aplicación de los datos.

¿Qué es un ámbito?

Se trata de un espacio de la aplicación a partir del cual se puede o no acceder a una variable, tal como:

- El ámbito de aplicación, que incluye:
 - Las diferentes tareas de la aplicación y
 - las secciones que las componen.
- Los ámbitos por tipos de datos, tales como:
 - Las estructuras/matrices para la familia DDT y
 - los EFB/DFB para la familia de bloques de funciones.

Reglas

La tabla indica si se puede o no utilizar un **nombre** que ya existe en la aplicación para elementos de nueva creación:

Contenido de la aplicación -> Elementos nuevos (a continuación)	Sección	SR	DDT/IO DDT	Tipo de FB	Instancias de FB	EF	Variable
Sección	No	No	Sí	Sí	Sí	Sí	Sí
SR	No	No	Sí	Sí	No	(1)	No
DDT/IODDT	No	No	No	No (4)	No	No (4)	No
Tipo de FB	Sí	Sí	No	No	(3)	No	(3)
Instancias de FB	No	No	No	Sí	No	Sí	No
EF	Sí	(2)	No	No	No	No	No
Variable	Sí	No	Sí	Sí	No	(1)	No

(1): Una instancia perteneciente al ámbito de la aplicación no puede tener el mismo nombre que una EF.

(2): Una instancia perteneciente al ámbito del tipo (variable interna) puede tener el mismo nombre que una EF. El EF en cuestión no se puede emplear en este tipo.

(3): No se permite la creación o importación de EFB/DFB con el mismo nombre que el de una instancia ya existente.

(4): Un elemento DDT/IODDT puede tener el mismo nombre que una FB/EF; sin embargo, no es recomendable, ya que la FB/EF no debe usarse en la aplicación.

NOTA: A continuación, se proporcionan consideraciones adicionales a las reglas de la tabla, que especifican lo siguiente:

- Dentro de un tipo, una instancia (variable interna) no puede tener el mismo nombre que el del tipo de objeto al que pertenece.
- No hay conflicto entre el nombre de una instancia perteneciente a una sección de la aplicación y el nombre de una instancia perteneciente a una sección de un DFB.
- No hay conflicto entre el nombre de una sección perteneciente a una tarea y el nombre de una sección perteneciente a un DFB.

Lenguajes de programación



Contenido de esta sección

En esta sección se describe la sintaxis de los lenguajes de programación disponibles.

Contenido de esta parte

Esta parte contiene los siguientes capítulos:

Capítulo	Nombre del capítulo	Página
11	función, lenguaje de bloques FBD	345
12	Diagrama de contactos (LD)	373
13	SFC, lenguaje de ejecución secuencial	421
14	Lista de instrucciones (IL)	487
15	Texto estructurado (ST)	537

Vista general

En este capítulo, se describe el lenguaje de bloques de función FBD según CEI 61131.

Contenido de este capítulo

Este capítulo contiene los siguiente apartados:

Apartado	Página
Generalidades sobre el lenguaje de bloques de funciones FBD	346
Funciones elementales, bloques de funciones elementales, bloques de funciones derivados y procedimientos (FFB)	348
Llamadas de subrutina	358
Elementos de control	359
Conexión	360
Objeto de texto	362
Secuencia de ejecución de FFB	363
Modificación de la secuencia de ejecución	366
Configuración de bucles	371

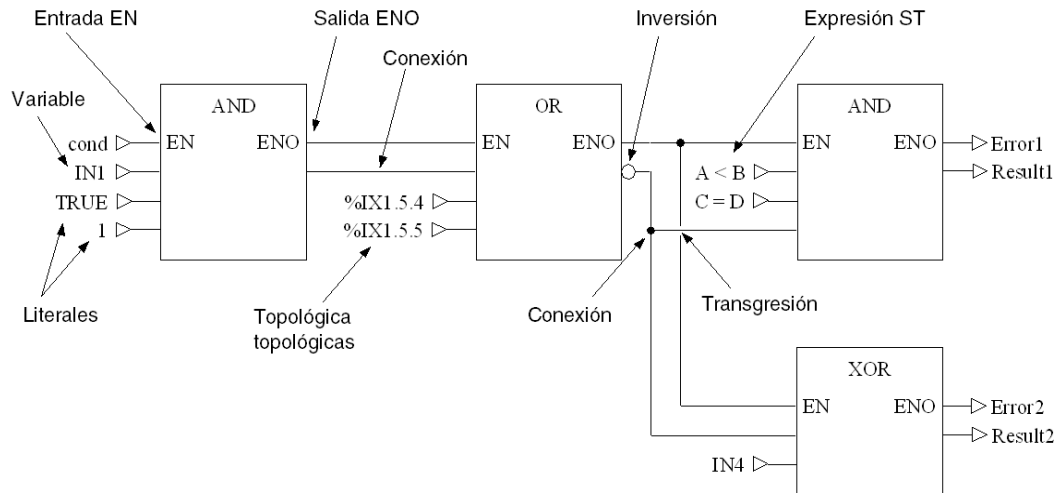
Generalidades sobre el lenguaje de bloques de funciones FBD

Introducción

El editor FBD permite la programación gráfica de bloques de funciones según CEI 61131-3.

Representación de una sección FBD

Representación:



Objetos

Los objetos del lenguaje de programación FBD (diagrama de bloques de funciones) sirven de ayuda para dividir una sección en una cantidad de:

- EF y EFB (funciones elementales (véase página 348) y bloques de funciones elementales (véase página 348))
- DFB (bloques de funciones derivados) (véase página 349),
- Procedimientos (véase página 350) y
- Elementos de control (véase página 359)

Estos objetos, denominados FFB en conjunto, pueden conectarse entre sí mediante:

- Conexiones (véase página 360) o
- Parámetros actuales (véase página 350)

La lógica de la sección se puede comentar por medio de objetos de texto (consulte "Objeto de texto, página 362").

Tamaño de la sección

Una sección FBD está compuesta por una ventana con una sola página.

Esta página está colocada sobre una rejilla. Una unidad de rejilla está compuesta por 10 puntos de retícula. Una unidad de retícula es la distancia mínima posible entre dos objetos de una sección FBD.

El lenguaje de programación FBD no está basado en celdas; los objetos están alineados con las coordenadas.

Una sección FBD puede configurarse con un número de celdas (coordenadas del reticulado horizontal y coordenadas del reticulado vertical).

Conformidad CEI

La descripción de la conformidad del lenguaje de programación FBD con la norma CEI figura en "Conformidad CEI (*véase página 693*)".

Funciones elementales, bloques de funciones elementales, bloques de funciones derivados y procedimientos (FFB)

Introducción

FFB es el término genérico que incluye:

- Función elemental (EF) (véase página 348)
- Bloque de función elemental (EFB) (véase página 348)
- Bloque de función derivado (DFB) (véase página 349)
- Procedimiento (véase página 350)

Función elemental

Las funciones elementales (EF) no tienen estados internos. Si los valores de entrada son los mismos, el valor de salida es el de cada ejecución de la función. Por ejemplo, la suma de dos valores siempre da el mismo resultado.

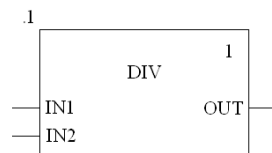
Una función elemental se representa de forma gráfica por medio de una trama con varias entradas y una salida. En él, las entradas siempre aparecen a la izquierda y la salida a la derecha.

El nombre de la función, es decir, el tipo de función, aparece centrado dentro de la trama.

A la derecha del tipo de función se indica el número de ejecución (véase página 363) de la función.

Encima de la trama aparece el contador de función. El contador de función es el número correlativo de la función dentro de la sección actual. Los contadores de función no se pueden modificar.

Función elemental



En el caso de algunas funciones elementales, se puede ampliar la cantidad de entradas.

Bloque de función elemental

Los bloques de funciones elementales (EFB) tienen estados internos. Si los valores de entrada son los mismos, el valor de la salida puede ser diferente cada vez que se ejecuta la función, por ejemplo, el valor de salida se incrementa para un contador.

Un bloque de función elemental se representa de forma gráfica por medio de una trama con varias entradas y salidas. En él, las entradas siempre aparecen a la izquierda y las salidas a la derecha.

Los bloques de funciones pueden disponer de varias salidas.

El nombre del bloque de función, es decir, el tipo de bloque de función, aparece centrado dentro de la trama.

A la derecha del tipo de bloque de función se indica el número de ejecución (*véase página 363*) del bloque de función.

El nombre de la instancia aparece por encima de la trama.

El nombre de instancia sirve para denominar de forma unívoca los distintos bloques de funciones de un proyecto.

El nombre de instancia se genera automáticamente y tiene la siguiente estructura:

FBI_n

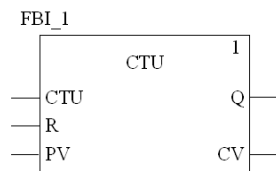
FBI = instancia del bloque de función

n = número correlativo del bloque de función en el proyecto

Este nombre generado automáticamente se puede modificar para mejorar la visión general. El nombre de instancia (32 caracteres como máximo) debe ser inequívoco dentro de un mismo proyecto; no se diferencia entre mayúsculas y minúsculas. El nombre de instancia debe cumplir la nomenclatura general.

NOTA: Según la norma CEI 61131-3, sólo se permite una letra como primer carácter de los nombres de instancias. Si desea utilizar cifras como primer carácter, deberá habilitar esa opción de forma explícita.

Bloque de función elemental

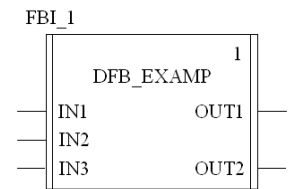


DFB

Los bloques de funciones derivados (DFB) presentan las mismas propiedades que los bloques de funciones elementales. Sin embargo, el usuario los crea en los lenguajes de programación FBD, LD, IL o ST.

La única diferencia con respecto a los bloques de funciones elementales es que los bloques de funciones derivados se representan gráficamente por medio de una trama con líneas verticales dobles.

Bloque de función derivado



Procedimiento

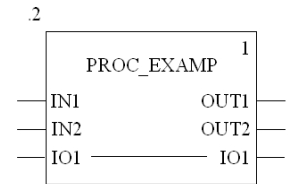
Los procedimientos son funciones desde el punto de vista técnico.

La única diferencia con respecto a las funciones elementales es que los procedimientos pueden tener más de una salida y admiten el tipo de datos VAR_IN_OUT.

Los procedimientos son una ampliación de la norma CEI 61131-3 y se deben habilitar de forma explícita.

Los procedimientos no se diferencian visualmente de las funciones elementales.

Procedimiento

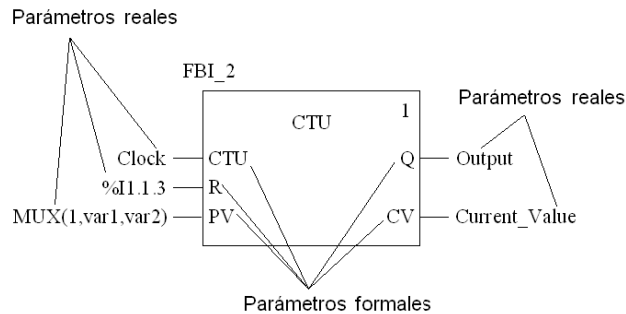


Parámetros

Para transferir valores al FFB o aplicarlos desde él, es necesario utilizar entradas y salidas. A éstas se les llama parámetros formales.

A los parámetros formales se vinculan objetos que contienen los estados actuales del proceso. Se conocen como parámetros reales.

Parámetros reales y formales:



Los valores del proceso se transfieren al FFB a través de los parámetros actuales durante el tiempo de ejecución del programa y se vuelven a emitir después del procesamiento.

A las entradas de FFB sólo se puede vincular un único objeto (parámetro real) del siguiente tipo:

- Variable
- Dirección
- Literal
- Expresión ST (véase página 539)

Las expresiones ST de las entradas de FFB representan una ampliación de la norma CEI 61131-3 y se deben habilitar de forma explícita.

- Enlace

A las salidas de FFB se pueden vincular las siguientes combinaciones de objetos (parámetros reales):

- Una variable
- Una variable y una o varias conexiones (aunque no en el caso de salidas VAR_IN_OUT (véase página 357))
- Una dirección
- Una dirección y una o varias conexiones (aunque no en el caso de salidas VAR_IN_OUT (véase página 357))
- Una o varias conexiones (aunque no en el caso de salidas VAR_IN_OUT (véase página 357))

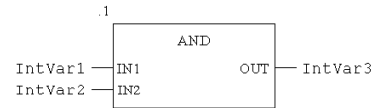
El tipo de datos del objeto que se va a enlazar debe coincidir con el tipo de datos de la entrada/salida de FFB. Si todos los parámetros reales están compuestos por literales, se elegirá un tipo de datos adecuado para el bloque de función.

Excepción: en el caso de entradas/salidas de FFB genéricas del tipo de datos ANY_BIT se pueden vincular objetos del tipo de datos INT o DINT (UINT ni UDINT).

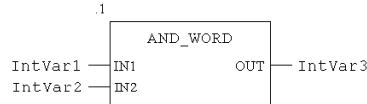
Se trata de una ampliación de la norma CEI 61131-3 y se debe habilitar de forma explícita.

Ejemplo:

Se admite:



No se admite:



(En este caso se debe utilizar `AND_INT`).

En principio, no es necesario asignar un parámetro real a cada parámetro formal. Pero esto no es válido para pins negados. A estos hay que asignarles siempre un parámetro real. También algunos tipos de parámetros formales lo requieren obligatoriamente. En la siguiente tabla encontrará estos tipos de parámetros formales.

Tabla de tipos de parámetros formales:

Tipo de parámetro	EDT	STRING	ARRAY	ANY_ARRAY	IODDT	STRUCT	FB	ANY
EFB: entrada	-	+	+	+	/	+	/	+
EFB: VAR_IN_OUT	+	+	+	+	+	+	/	+
EFB: salida	-	-	+	+	+	-	/	+
DFB: entrada	-	+	+	+	/	+	/	+
DFB: VAR_IN_OUT	+	+	+	+	+	+	/	+
DFB: salida	-	-	+	/	/	-	/	+
EF: entrada	-	-	+	+	+	+	+	+
EF: VAR_IN_OUT	+	+	+	+	+	+	/	+
EF: salida	-	-	-	-	-	-	/	-
Procedimiento: entrada	-	-	+	+	+	+	+	+
Procedimiento: VAR_IN_OUT	+	+	+	+	+	+	/	+
Procedimiento: salida	-	-	-	-	-	-	/	+
+ Parámetro real requerido obligatoriamente								
- Parámetro real no requerido obligatoriamente								
/ No es aplicable								

Los FFB que utilizan en las entradas parámetros reales que todavía no contienen ninguna asignación de valor trabajan con los valores iniciales de estos parámetros reales.

Si no se asigna un valor a un parámetro formal, durante la ejecución del bloque de función se utilizará el valor inicial. Si no se ha definido ningún valor inicial, se aplicará el valor predeterminado (0).

Si un parámetro formal no tiene asignado ningún valor y se han realizado varias instancias del bloque de función/DFB, las instancias que se ejecuten a partir de ese momento trabajarán con el valor antiguo.

Variables públicas

Además de las entradas y salidas, algunos bloques de funciones también disponen de las denominadas variables públicas.

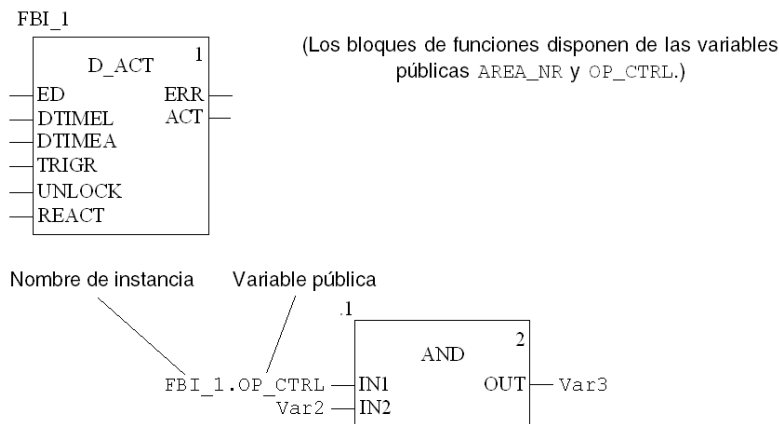
Estas variables sirven para transmitir valores estáticos (valores no influidos por el proceso) al bloque de función. Se utilizan para la parametrización del bloque de función.

Las variables públicas son una ampliación de la norma CEI 61131-3.

La asignación de valores a las variables públicas se realiza mediante sus valores iniciales.

La lectura de los valores de las variables públicas se realiza a través del nombre de instancia del bloque de función y a través del nombre de la variable pública.

Ejemplo:



Variables privadas

Además de las entradas, las salidas y las variables públicas, algunos bloques de funciones también disponen de las denominadas variables privadas.

Al igual que las variables públicas, las privadas se utilizan para transferir valores estadísticos (valores no influidos por el proceso) al bloque de función.

El programa de usuario no puede acceder a las variables privadas. Sólo se puede acceder a este tipo de variables mediante la tabla de animación.

NOTA: Los DFB intercalados se declaran como variables privadas del DFB principal. Por tanto, tampoco se puede acceder a sus variables a través de la programación, sino a través de la tabla de animación.

Las variables privadas constituyen un suplemento de la normativa CEI 61131-3.

Notas sobre la programación

Tenga en cuenta las siguientes indicaciones sobre la programación:

- Los FFB sólo se ejecutan si la entrada EN es 1 o si la entrada EN está oculta (consulte también *EN* y *ENO*, *página 355*).
- Las entradas y salidas booleanas se pueden invertir.
- Si se utilizan variables VAR_IN_OUT (véase *página 357*), se imponen condiciones especiales.
- Las instancias de bloques de funciones/DFB se pueden llamar varias veces (consulte también *Llamada múltiple de una instancia del bloque de función*, *página 354*).

Llamada múltiple de una instancia del bloque de función

Las instancias de bloques de funciones/DFB se pueden llamar varias veces, excepto las instancias de EFB de comunicaciones y de bloques de funciones/DFB con una salida ANY y ninguna entrada ANY, que sólo se pueden llamar una vez.

La llamada múltiple de una misma instancia de bloque de función/DFB resulta conveniente en los siguientes casos:

- Cuando el bloque de función/DFB no posee ningún valor interno o los valores internos no son necesarios para continuar el procesamiento.
En este caso, la llamada múltiple de una misma instancia de bloque de función/DFB permite ahorrar espacio en memoria, ya que el código del bloque de función/DFB sólo se carga una vez.
El bloque de función/DFB se procesa, por así decirlo, como una función.
- Cuando el bloque de función/DFB tiene valores internos y éstos se deben modificar en varios puntos del programa, por ejemplo, si el valor de un contador se debe aumentar en diversos puntos del programa.
En este caso, con la llamada múltiple de la misma instancia del bloque de función/DFB no es necesario guardar los resultados intermedios para continuar el procesamiento en otro lugar del programa.

EN y ENO

En todos los FFB se puede configurar una entrada **EN** y una salida **ENO**.

Si el valor de **EN** es "0", al llamar el FFB no se ejecutarán los algoritmos definidos por dicho FFB, y **ENO** se ajustará a "0".

Si el valor de **EN** es "1", al llamar el FFB se ejecutarán los algoritmos definidos por dicho FFB. Una vez que se han ejecutado los algoritmos correctamente, el valor de **ENO** pasa a "1". Si se produce un error al ejecutar estos algoritmos, **ENO** pasa a "0".

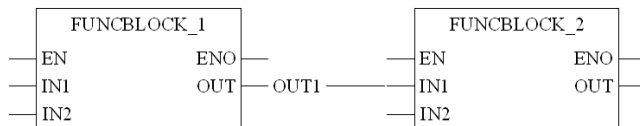
Si **ENO** pasa a "0" (porque **EN** = 0 o por un error durante la ejecución):

- Bloques de funciones
 - Manipulación de **EN/ENO** con bloques de funciones que (sólo) tienen un vínculo como parámetro de salida:



Si el valor de **EN** en **FUNCBLOCK_1** se establece en "0", el enlace en la salida **OUT** de **FUNCBLOCK_1** mantiene el estado que tenía en el último ciclo que se ejecutó correctamente.

- Gestión de **EN/ENO** con bloques de funciones que tienen una variable y un vínculo como parámetros de salida:



Si el valor de **EN** en **FUNCBLOCK_1** se establece en "0", el enlace en la salida **OUT** de **FUNCBLOCK_1** mantiene el estado que tenía en el último ciclo que se ejecutó correctamente. La variable **OUT1** en el mismo pin puede mantener su estado anterior o puede modificarse desde fuera sin influir en el enlace. La variable y el enlace se guardan de forma independiente entre sí.

- Funciones/procedimientos
Según la definición de CEI 61131-3, las salidas de las funciones desactivadas (entrada **EN** ajustada a "0") no están definidas (esto es aplicable a los procedimientos).
A continuación se presenta una explicación de los estados de salida en este caso:

- Gestión de EN/ENO en las funciones o procedimientos que poseen (sólo) una conexión como parámetro de salida:



Si el valor de EN en FUNC_PROC_1 se establece en "0", el valor del enlace en la salida de OUT de FUNC_PROC_1 dependerá del ajuste del proyecto

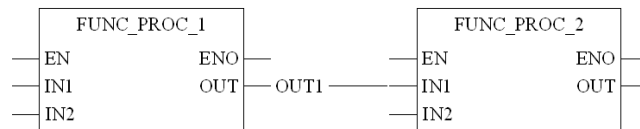
Mantener conexiones de salida en EF desactivadas disponible a partir de Unity Pro 4.1.

Si se establece este ajuste del proyecto en "0", el valor del enlace se establecerá en "0".

Si se establece este ajuste del proyecto en "1", el vínculo mantendrá el valor que tenía en el último ciclo ejecutado correctamente.

Consulte *Mantener conexiones de salida en EF desactivadas* (véase *Unity Pro, Modalidades de funcionamiento*,).

- Gestión de EN/ENO en las funciones o procedimientos que poseen una variable y una conexión como parámetro de salida:



Si el valor de EN en FUNC_PROC_1 se establece en "0", el valor del enlace en la salida de OUT de FUNC_PROC_1 dependerá del ajuste del proyecto

Mantener conexiones de salida en EF desactivadas disponible a partir de Unity Pro 4.1.

Si se establece este ajuste del proyecto en "0", el valor del enlace se establecerá en "0".

Si se establece este ajuste del proyecto en "1", el vínculo mantendrá el valor que tenía en el último ciclo ejecutado correctamente.

Consulte *Mantener conexiones de salida en EF desactivadas* (véase *Unity Pro, Modalidades de funcionamiento*,).

La variable OUT1 en el mismo pin puede mantener su estado anterior o puede modificarse desde fuera sin influir en el enlace. La variable y el enlace se guardan de forma independiente entre sí.

El comportamiento en la salida de los FFB no depende de si los FFB se han llamado sin EN/ENO o con EN = 1.

NOTA: Para bloques de funciones deshabilitados (EN = 0) con una función de tiempo interna (por ejemplo, bloques de funciones DELAY), el tiempo sigue avanzando, ya que se calcula con la ayuda de un reloj del sistema, y por tanto es independiente del ciclo de programa y de la apertura del bloque.

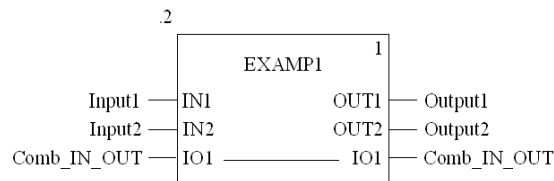
Variable VAR_IN_OUT

A menudo, los FFB se utilizan para leer una variable en la entrada (variables de entrada), procesarla y devolver los valores modificados de **esta** variable (variables de salida).

Este caso específico de variable de entrada/salida también se denomina variable VAR_IN_OUT.

El enlace de las variables de entrada y salida se representa en el FFB por medio de una línea.

Variable VAR_IN_OUT



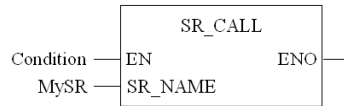
Tenga en cuenta las siguientes particularidades cuando utilice los FFB con variables VAR_IN_OUT:

- Hay que asignar obligatoriamente una variable a todas las entradas VAR_IN_OUT.
- Los enlaces gráficos sólo permiten conectar salidas VAR_IN_OUT a entradas VAR_IN_OUT.
- Sólo es posible enlazar una conexión gráfica a una salida/entrada VAR_IN_OUT.
- En el caso de las salidas VAR_IN_OUT, no es posible combinar una variable/dirección con conexiones gráficas.
- No puede haber literales ni constantes vinculadas a entradas/salidas VAR_IN_OUT.
- No se puede utilizar ninguna negación en las entradas/salidas VAR_IN_OUT.
- Es posible vincular a la entrada VAR_IN_OUT y a la salida VAR_IN_OUT distintas variables o componentes de variables. En tal caso, el valor de la variable o de su componente en la entrada se copiará en la variable o componente de variable en la salida.

Llamadas de subrutina

Llamada de subrutina

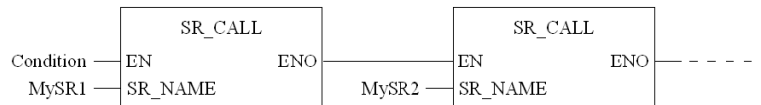
En FBD, las subrutinas se llaman mediante los bloques siguientes:



Si el estado de **EN** es 1, se llama la subrutina correspondiente (Nombre de las variables en **SR_Name**).

La salida **ENO** no sirve en este tipo de bloque para mostrar el estado de error. La salida **ENO** es siempre 1 en este tipo de bloque y sirve para llamar varias subrutinas simultáneamente.

Con la siguiente construcción, es posible llamar varias subrutinas simultáneamente.



La subrutina que se vaya a llamar debe encontrarse en la misma tarea que la sección FBD invocante.

También es posible llamar subrutinas ubicadas dentro de otras subrutinas.

Las llamadas de subprograma son una ampliación de la norma CEI 61131-3 y se deben habilitar de forma explícita.

En las secciones de acción SFC sólo se admiten llamadas de subrutina si está habilitada la modalidad Multi-Token.

Elementos de control

Introducción

Los elementos de control sirven para ejecutar saltos dentro de una sección FBD y para el retorno a la rutina principal desde una subrutina (SRx) o desde un bloque de función derivado (DFB).

Elementos de control

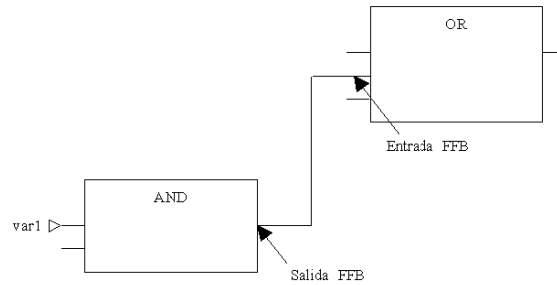
Se encuentran disponibles los siguientes elementos de control.

Designación	Representación	Descripción
Salto		Si el estado de la conexión izquierda es 1, se ejecutará un salto a la etiqueta (en la sección actual). Para generar un salto condicional, el objeto de salto se conecta a una salida FFB booleana. Para generar un salto incondicional, se asigna el valor 1 al objeto de salto a través de la función AND.
Etiqueta	LABEL :	Las etiquetas (destinos de salto) se representan como texto con dos puntos al final. El texto está limitado a 32 caracteres y debe ser unívoco en toda la sección. El texto debe cumplir las convenciones de nomenclatura general. Las etiquetas de salto sólo se pueden colocar entre los dos primeros puntos de retícula del borde izquierdo de la sección. Nota: Las etiquetas de salto no pueden "cortar" ninguna red, es decir, la línea imaginaria que une la etiqueta de salto con el borde derecho de la sección no puede estar cortada por ningún objeto. Esto también se aplica a las conexiones.
Retorno		Los objetos RETURN no pueden usarse en el programa principal. - En un DFB, un objeto RETURN fuerza el retorno al programa que llamó al DFB. - El resto de la sección DFB que contiene el objeto RETURN no se ejecuta. - Las siguientes secciones del DFB no se ejecutan. El programa que llamó al DFB se ejecutará después de volver del DFB. Si otro DFB llama al DFB, el DFB llamado se ejecutará después de volver. - En un SR, un objeto RETURN fuerza el retorno al programa que llamó al SR. - El resto de la sección SR que contiene el objeto RETURN no se ejecuta. El programa que llamó al SR se ejecutará después de volver del SR.

Conexión

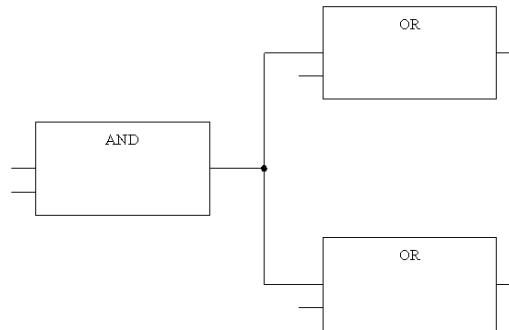
Descripción

Las conexiones son vínculos verticales y horizontales entre los FFB.

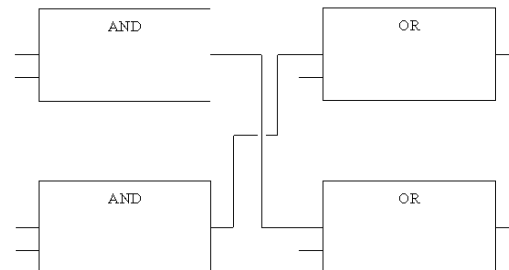


Representación

Los puntos de conexión se identifican por medio de un círculo relleno.



Los cruces se representan con una conexión "interrumpida".



Notas sobre la programación

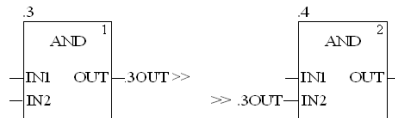
Tenga en cuenta las siguientes indicaciones sobre la programación:

- Se pueden utilizar conexiones para todos los tipos de datos.
- Los tipos de datos de las entradas/salidas que se vayan conectar deben coincidir.
- Se pueden establecer varias conexiones con una salida de FFB. No obstante, sólo puede haber una con una entrada FFB.
- Sólo se pueden conectar entradas y salidas entre sí. No es posible interconectar varias salidas a la vez. Esto quiere decir que en FBD no se puede realizar ninguna operación OR por medio de conexiones. Se debe utilizar siempre una función OR.
- Está permitido que las conexiones se solapen con otros objetos.
- Con conexiones no se pueden configurar bucles, ya que en este caso no se puede determinar de forma inequívoca la secuencia de ejecución en la sección. Los bucles se deben resolver con parámetros actuales (consulte "*Configuración de bucles, página 371*").
- Para evitar el cruce de conexiones, las conexiones también se pueden representar en forma de conectores.

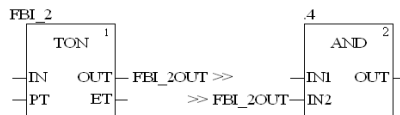
El origen y el destino de la conexión se identifican con un nombre unívoco dentro de la sección.

El nombre del conector tiene la siguiente estructura en función del tipo de objeto de origen de la conexión:

- En el caso de funciones: "Contador de función/parámetro formal" del origen de la conexión



- En el caso de bloques de función: "Nombre de instancia/parámetro formal" del origen de la conexión



Objeto de texto

Descripción

El texto puede ubicarse como objetos de texto mediante el lenguaje de bloques de funciones FBD. El tamaño de estos objetos depende de la longitud del texto. En función del tamaño del texto, es posible ampliar el tamaño del objeto en más unidades de rejilla en dirección vertical y horizontal. Los objetos de texto no se solapan con los FFB; sin embargo, pueden solaparse con las conexiones.

Secuencia de ejecución de FFB

Introducción

La secuencia de ejecución está determinada en primer lugar por la posición de los FFB dentro de la sección (ejecución de izquierda a derecha y de arriba a abajo). Si, a continuación, los FFB se conectan a una red por medio de conexiones gráficas, la secuencia de ejecución estará determinada por el flujo de datos.

La secuencia de ejecución se indica mediante el número de ejecución (número situado en la esquina superior derecha del marco del FFB).

Secuencia de ejecución de redes

Para la secuencia de ejecución de redes se aplican las siguientes reglas:

- La ejecución de una sección se realiza red a red en función de las conexiones de FFB de arriba a abajo.
- No es posible configurar bucles mediante conexiones, ya que en este caso no se puede determinar de forma inequívoca la secuencia de ejecución en la sección. Los bucles se deben resolver con parámetros actuales (consulte "*Configuración de bucles, página 371*").
- La secuencia de ejecución de las redes que no están conectadas entre sí mediante conexiones está determinada por la secuencia gráfica (de derecha a izquierda y de arriba a abajo). Esta secuencia de ejecución se puede modificar (consulte "*Modificación de la secuencia de ejecución, página 366*").
- El cálculo de una red finaliza completamente antes de que comience el cálculo de otra red que utilice salidas de la red precedente.
- Ningún elemento de una red se considera calculado mientras no se haya calculado el estado de todas las entradas de dicho elemento.
- El cálculo de una red sólo se considera finalizado cuando se han calculado todas las salidas de dicha red.

Flujo de señal dentro de una red

Para la secuencia de ejecución dentro de una red se aplican las siguientes reglas:

- El cálculo de un FFB no se inicia hasta que no se hayan calculado todos los elementos (salidas FFB, etc.) que estén conectados a sus entradas.
- La secuencia de ejecución de los FFB que estén conectados a varias salidas del mismo FFB es de arriba a abajo.
- La secuencia de ejecución de los FFB no está influida por su posición dentro de la red.

Esto no es válido cuando varios FFB están conectados a la misma salida del FFB "invocante". En este caso, la secuencia de ejecución está determinada por la secuencia gráfica (de arriba a abajo).

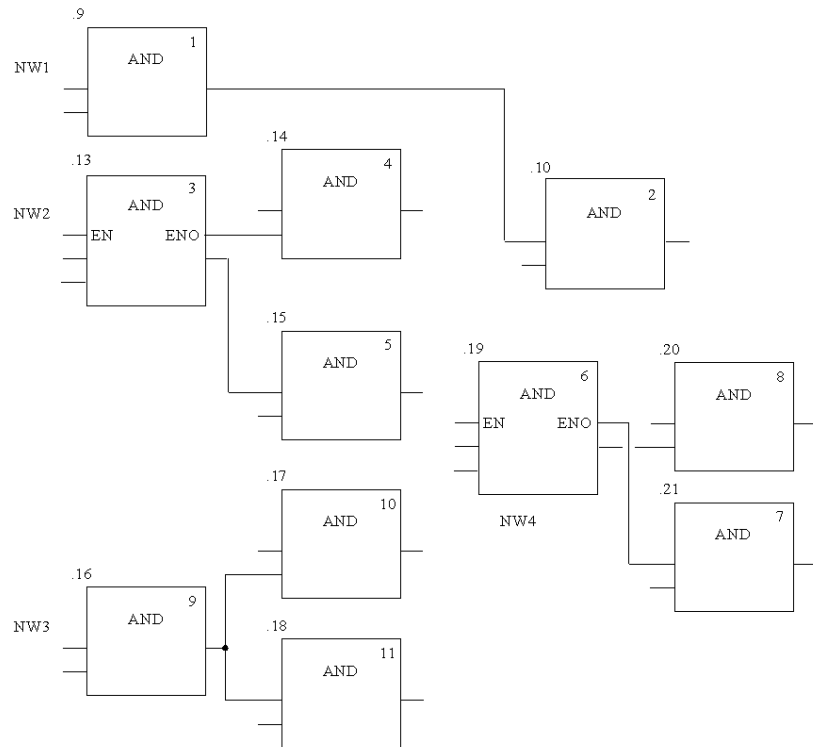
Prioridades

Prioridades a la hora de determinar el flujo de señal dentro de una sección.

Prioridad	Regla	Descripción
1	Conexión	Las conexiones tienen el nivel de prioridad más alto a la hora de determinar el flujo de señal dentro de una sección FBD.
2	Definición de usuario	Modificación de la secuencia de ejecución por parte del usuario.
3	Red a red	El cálculo de una red finaliza completamente antes de que comience el cálculo de la red siguiente.
4	Secuencia de las salidas	Los FFB conectados a las salidas del mismo FFB "invocante" se calculan de arriba abajo.
5	Circuito a circuito	Nivel de prioridad más bajo. (Sólo es válido si no se aplica ninguna de las demás reglas).

Ejemplo

Ejemplo de la secuencia de ejecución de objetos en una sección FBD.



Modificación de la secuencia de ejecución

Introducción

La secuencia de ejecución de las redes y de los objetos incluidos dentro de una red está definida por una serie de reglas (*véase página 364*).

En la mayoría de los casos resulta recomendable modificar la secuencia de ejecución propuesta por el sistema.

Existen las siguientes posibilidades para determinar/modificar la secuencia de ejecución de las redes:

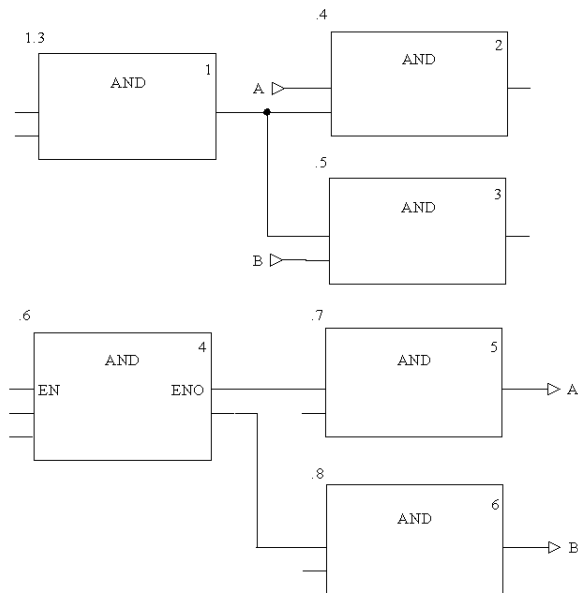
- Utilización de conexiones en lugar de parámetros actuales
- Posición de las redes
- Determinación explícita de la secuencia de ejecución

Existen las siguientes posibilidades para determinar/modificar la secuencia de ejecución de las redes:

- Posición de los FFB

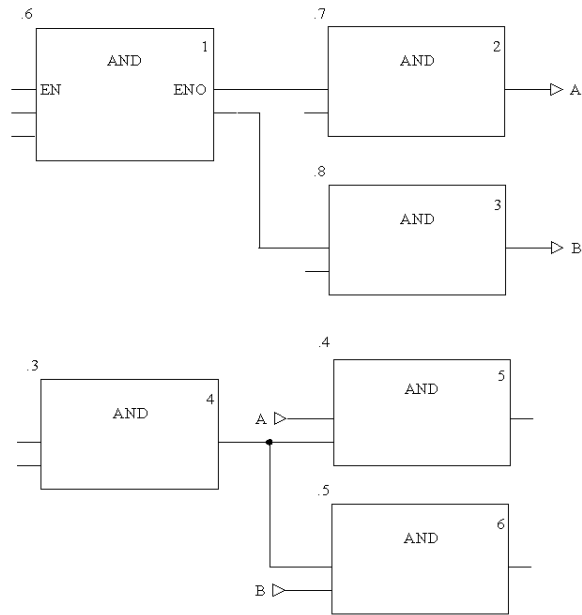
Situación de salida

En la figura siguiente se muestran dos redes cuya secuencia de ejecución está determinada únicamente por su posición dentro de la sección, con independencia de que los bloques .4/.5 y .7/.8 requieran otra secuencia de ejecución.



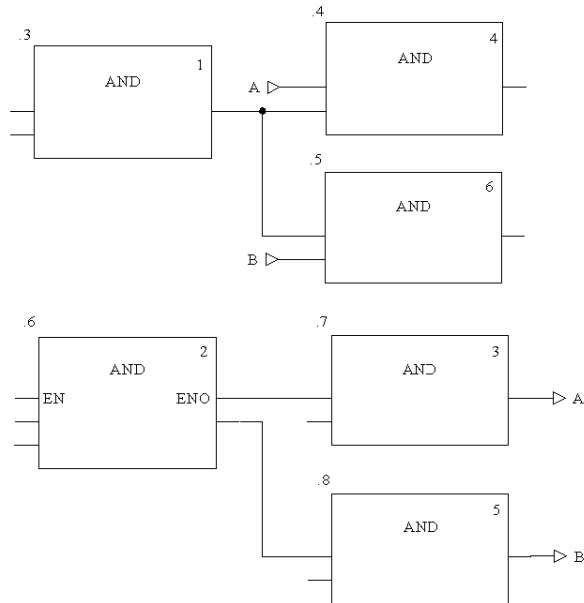
Posición de las redes

La secuencia de ejecución adecuada se puede conseguir modificando la posición de las redes de la sección (consulte también *Situación de salida*, página 366).



Determinación explícita

La secuencia de ejecución adecuada se puede conseguir modificando explícitamente la secuencia de ejecución de un FFB. En el caso de los FFB cuya secuencia de ejecución se haya modificado explícitamente, el número de ejecución se representa en un campo negro (consulte también *Situación de salida*, página 366).



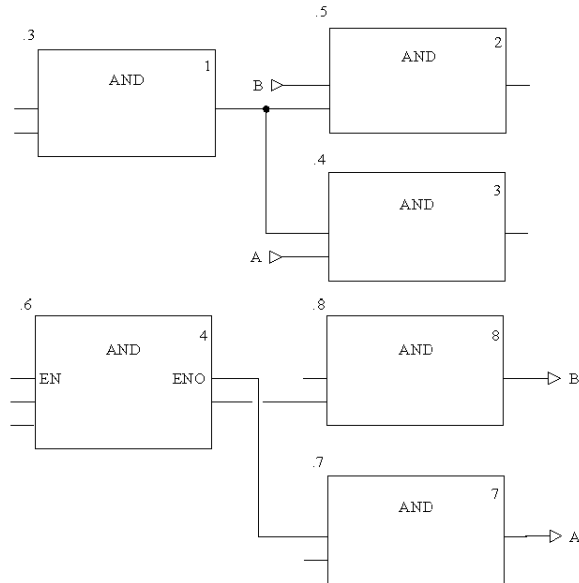
NOTA: Sólo se permite una única referencia a una instancia; por ejemplo, la instancia ".7" sólo se puede referenciar una vez.

Posiciones de los FFB

La posición de los FFB sólo influye sobre la secuencia de ejecución si hay varios FFB conectados a la misma salida del FFB "invocante" (consulte también *Situación de salida*, página 366).

En la primera red, se han intercambiado las posiciones de los bloques .4 y .5. En este caso (origen común de ambas entradas de bloque) también se intercambia la secuencia de ejecución de ambos bloques (procesamiento de arriba a abajo).

En la segunda red, se han intercambiado las posiciones de los bloques . 7 y . 8. En este caso (origen distinto de las entradas de bloque), la secuencia de ejecución de ambos bloques no se intercambia (procesamiento en la secuencia de las salidas de bloque invocantes).

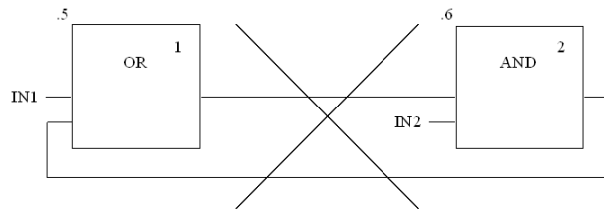


Configuración de bucles

Bucles no permitidos

No es posible configurar bucles exclusivamente a base de conexiones ya que de este modo no se puede determinar el flujo de señal de forma unívoca (la salida de un FFB es la entrada del FFB siguiente y la salida de éste es, a su vez, la entrada del primero).

Bucle no permitido a través de conexiones



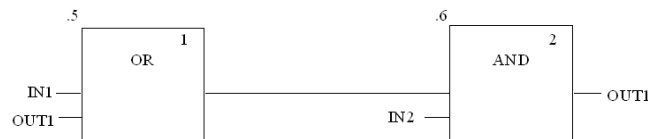
Generación a través de un parámetro real

Una lógica de este tipo se debe resolver con ayuda de una variable de realimentación para que el flujo de señal se pueda determinar de forma unívoca.

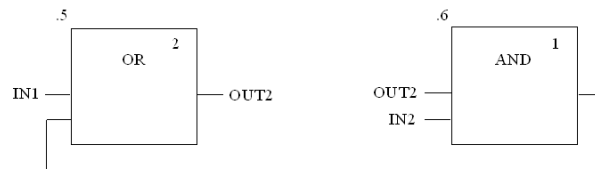
Las variables de realimentación se deben inicializar. El valor inicial se utiliza durante la primera ejecución de la lógica. Una vez ejecutado, el valor inicial se sustituye por el valor actual.

Tenga en cuenta en ambos casos la secuencia de ejecución (número entre paréntesis detrás del nombre de instancia) de los dos bloques.

Bucle resuelto a través de un parámetro actual: caso 1



Bucle resuelto a través de un parámetro actual: caso 2



Vista general

En este capítulo, se describe el lenguaje de diagrama de contactos LD según CEI 611311.

Contenido de este capítulo

Este capítulo contiene los siguiente apartados:

Apartado	Página
Generalidades sobre el lenguaje de diagrama de contactos LD	374
Contactos	377
Bobinas	379
Funciones elementales, bloques de función elementales, bloques de función derivados y procedimientos (FFB)	381
Elementos de control	392
Bloques de funcionamiento y de comparación	393
Conexiones	396
Objeto de texto	400
Detección de flancos	401
Secuencia de ejecución y flujo de señal	411
Configuración de bucles	413
Modificación de la secuencia de ejecución	415

Generalidades sobre el lenguaje de diagrama de contactos LD

Introducción

En esta sección se describe el esquema de contactos (Ladder Diagramm) LD según CEI 61131-3.

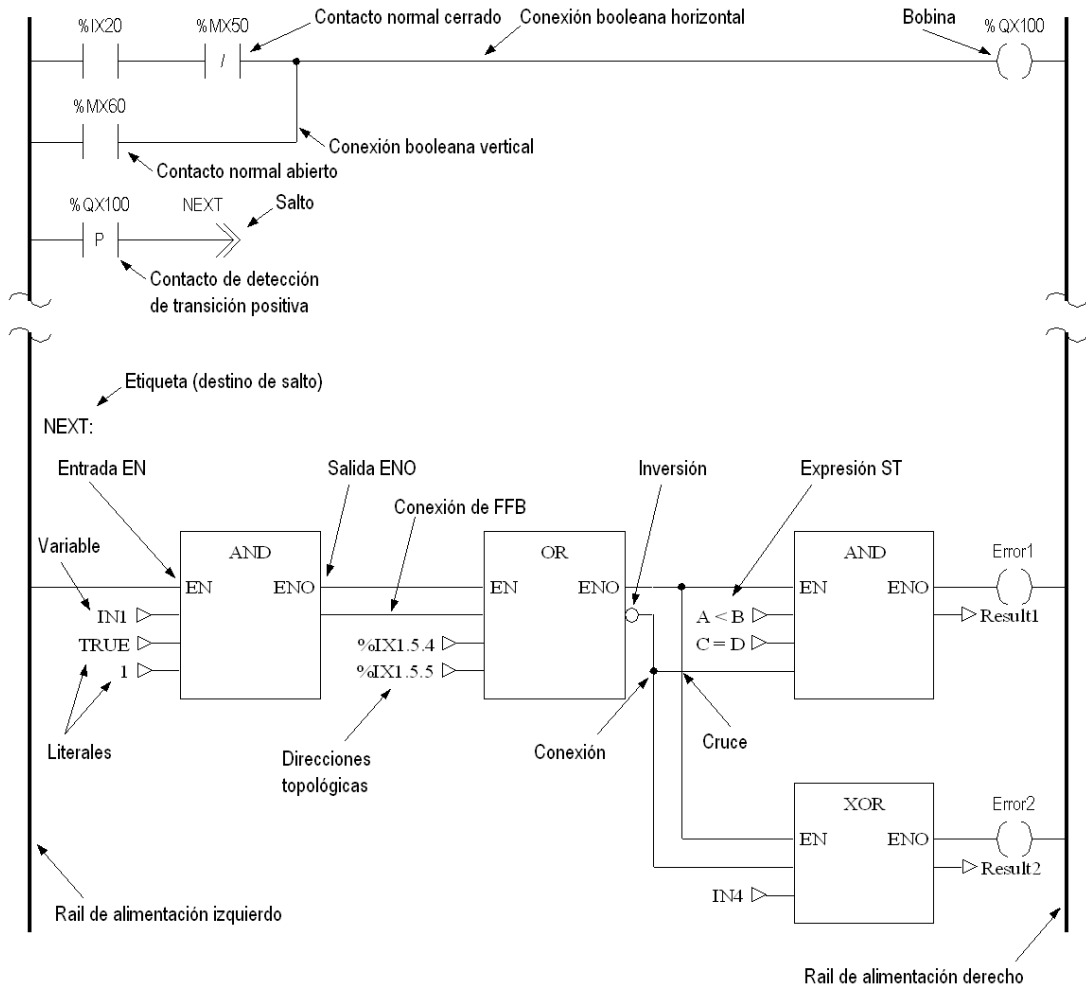
La estructura de una sección LD corresponde a un circuito de corriente para conmutadores de relé.

En el lado izquierdo del editor LD se encuentra el denominado "rail de alimentación izquierdo". Este rail de alimentación izquierdo corresponde a la fase (conductor L) de un circuito de corriente. Al igual que en un circuito de corriente, en la programación LD sólo se "editan" los objetos LD que estén conectados a una fuente de alimentación, es decir, que estén conectados al rail de alimentación izquierdo. El rail de alimentación derecho equivale al conductor neutro. A él se conectan directa o indirectamente todas las bobinas y salidas de FFB, y de esta forma se establece un flujo de corriente.

Un grupo de objetos conectados entre sí que no poseen ninguna conexión a otros objetos (excepto al rail de alimentación) se denomina red o circuito de corriente.

Representación de una sección LD

Representación:



Objetos

Los objetos del lenguaje de programación LD ofrecen ayuda para dividir una sección en una cantidad de:

- Contactos (véase página 377)
- Bobinas (véase página 379)

- EF y EFB (funciones elementales (*véase página 381*) y bloques de funciones elementales (*véase página 382*))
- DFB (bloques de funciones derivados (*véase página 383*))
- Procedimientos (*véase página 383*)
- Elementos de control (*véase página 392*) y
- Bloques de operación y comparación (*véase página 393*), que representan una ampliación de la norma CEI 61131-3.

Estos objetos se pueden conectar entre sí por medio de:

- Conexiones (*véase página 396*) o
- Parámetros actuales (*véase página 384*) (sólo FFB)

La lógica de la sección se puede comentar por medio de objetos de texto (consulte "*Objeto de texto, página 400*").

Tamaño de la sección

Una sección LD está compuesta por una ventana de una sola página.

Esta página tiene una rejilla de fondo que divide la sección en filas y columnas.

Para las secciones LD se puede definir un ancho de 11-64 columnas y 17-2.000 filas.

El lenguaje de programación LD está basado en celdas, es decir, en cada celda se puede colocar un único objeto.

Secuencia de procesamiento

La secuencia de procesamiento de los diferentes objetos en una sección LD está determinada por el flujo de datos dentro de la sección. Las redes conectadas al rail de alimentación izquierdo se procesarán de arriba a abajo (conexión con el rail de alimentación izquierdo). Las redes independientes entre sí dentro de la sección se procesarán siguiendo la secuencia de ubicación (de arriba a abajo) (consulte "*Secuencia de ejecución y flujo de señal, página 411*").

Conformidad CEI

Para obtener una descripción de la conformidad del lenguaje de programación LD con la norma CEI, consulte "Conformidad CEI (*véase página 693*)".

Contactos

Introducción

Un contacto es un elemento de LD que transfiere un estado de la conexión horizontal al extremo derecho. Este estado es el resultado de una operación AND booleana sobre el estado de la conexión horizontal del extremo izquierdo con el estado del parámetro booleano real correspondiente.

Un contacto no modifica el valor del parámetro real correspondiente.

Los contactos ocupan una celda.

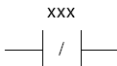
Como parámetros reales, se admiten:

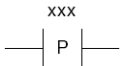
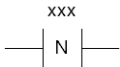
- Variables booleanas
- Constantes booleanas
- Direcciones booleanas (direcciones topológicas o direcciones simbólicas)
- Expresión ST (*véase página 539*) que un resultado booleano (p.ej. `VarA OR VarB`)

Las expresiones ST como parámetros reales de contactos representan una ampliación de la norma CEI 61131-3 y se deben habilitar de forma explícita.

Tipos de contactos

Están disponibles los siguientes contactos:

Designación	Representación	Descripción
Normalmente abierto		En el caso de los contactos normalmente abiertos, el estado de la conexión de la izquierda se transfiere a la conexión de la derecha si el estado del parámetro booleano real correspondiente (indicado mediante xxx) está activo. De lo contrario, el estado de la conexión derecha está inactivo.
Normalmente cerrado		En el caso de los contactos normalmente cerrados, el estado de la conexión de la izquierda se transfiere a la conexión de la derecha si el estado del parámetro booleano real correspondiente (indicado mediante xxx) está inactivo. De lo contrario, el estado de la conexión derecha está inactivo.

Designación	Representación	Descripción
Contacto para detectar transiciones positivas		En el caso de estos contactos, la conexión derecha de un ciclo de programa está activa si la transferencia del parámetro real correspondiente (indicado mediante xxx) pasa de inactivo a activo y el estado de la conexión izquierda está activo al mismo tiempo. De lo contrario, el estado de la conexión derecha es 0. Consulte también <i>Detección de flancos</i> , página 401.
Contacto para detectar transiciones negativas		En el caso de estos contactos, la conexión derecha de un ciclo de programa está activa si la transferencia del parámetro real correspondiente (indicado mediante xxx) pasa de activo a inactivo y el estado de la conexión izquierda está activo al mismo tiempo. De lo contrario, el estado de la conexión derecha es 0. Consulte también <i>Detección de flancos</i> , página 401.

Bobinas

Introducción

Una bobina es un elemento de LD que transfiere el estado de la conexión horizontal del extremo izquierdo, sin modificar, a la conexión horizontal del extremo derecho. El estado se almacena en el parámetro booleano real correspondiente.

Normalmente, las bobinas siguen a contactos o FFB, pero también pueden ir seguidas de contactos.

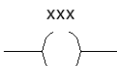
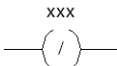
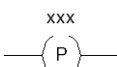
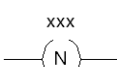
Las bobinas ocupan una celda.

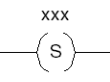
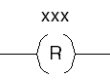
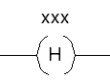
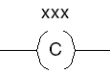
Como parámetros reales, se admiten:

- Variables booleanas
- Direcciones booleanas (direcciones topológicas o direcciones simbólicas)

Tipos de bobinas

Se encuentran disponibles las siguientes bobinas:

Designación	Representación	Descripción
Bobina		En el caso de las bobinas, el estado de la conexión de la izquierda se transfiere al parámetro booleano real correspondiente (indicado mediante xxx) y la conexión de la derecha.
Bobina negada		En el caso de estas bobinas, el estado de la conexión de la izquierda se copia en la conexión de la derecha. El estado invertido de la conexión de la izquierda se copia al parámetro booleano real correspondiente (indicado mediante xxx). Si la conexión de la izquierda está inactiva, la conexión de la derecha también estará inactiva y el parámetro booleano real correspondiente estará activo.
Bobina para detectar transiciones positivas		En el caso de estas bobinas, el estado de la conexión de la izquierda se copia en la conexión de la derecha. El parámetro real correspondiente de tipo de dato EBOOL (indicado mediante xxx) es 1 para un ciclo de programa si se realiza una transición de la conexión de la izquierda de 0 a 1. Consulte también <i>Detección de flancos, página 401</i> .
Bobina para detectar transiciones negativas		En el caso de estas bobinas, el estado de la conexión de la izquierda se copia en la conexión de la derecha. El parámetro booleano real correspondiente (indicado mediante xxx) es 1 para un ciclo de programa si se realiza una transición de la conexión de la izquierda de 1 a 0. Consulte también <i>Detección de flancos, página 401</i> .

Designación	Representación	Descripción
Bobina de ajuste		En el caso de estas bobinas, el estado de la conexión de la izquierda se copia en la conexión de la derecha. El parámetro booleano real correspondiente (indicado mediante xxx) está activo si la conexión de la izquierda está activa. Si no, permanece sin cambios. El parámetro booleano real correspondiente puede restablecerse mediante la bobina de ajuste. Consulte también <i>Detección de flancos, página 401</i>.
Bobina de restablecimiento		En el caso de estas bobinas, el estado de la conexión de la izquierda se copia en la conexión de la derecha. El parámetro booleano real correspondiente (indicado mediante xxx) está inactivo si la conexión de la izquierda está activa. Si no, permanece sin cambios. El parámetro booleano real correspondiente puede establecerse mediante la bobina de ajuste. Consulte también <i>Detección de flancos, página 401</i>.
Bobina de detención		En el caso de estas bobinas, si el estado de la conexión de la izquierda es 1, la ejecución del programa se detiene de forma inmediata. (En las bobinas de detención, el estado de la conexión de la izquierda no se copia en la conexión de la derecha.)
Bobina de llamada		En el caso de estas bobinas, el estado de la conexión de la izquierda se copia en la conexión de la derecha. Si el estado de la conexión de la izquierda está activo, se produce una llamada al subprograma correspondiente (indicado mediante xxx). La subrutina a la que se va a llamar debe encontrarse en la misma tarea que la sección LD invocante. También es posible llamar subrutinas ubicadas dentro de otras subrutinas. Las subrutinas son una ampliación de la norma CEI 61131-3 y se deben habilitar de forma explícita. En las secciones de acción SFC, sólo se admiten bobinas de llamada (llamadas de subrutina) si está habilitada la modalidad Multi-Token.

Funciones elementales, bloques de función elementales, bloques de función derivados y procedimientos (FFB)

Introducción

FFB es el término genérico que incluye:

- Función elemental (EF) *(véase página 381)*
- Bloque de función elemental (EFB) *(véase página 382)*
- Bloque de función derivado (DFB) *(véase página 383)*
- Procedimiento *(véase página 383)*

Los FFB ocupan un ancho de 1 a 3 columnas (en función de la longitud de los nombres de los parámetros formales) y una longitud de 2 a 33 líneas (en función de la cantidad de filas de parámetros formales).

Función elemental

Las funciones no tienen estados internos. Si los valores de entrada son los mismos, el valor de salida es el de cada ejecución de la función. Por ejemplo, la suma de dos valores siempre da el mismo resultado.

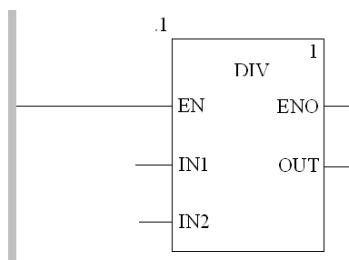
Una función elemental se representa de forma gráfica por medio de una trama con varias entradas y una salida. En él, las entradas siempre aparecen a la izquierda y la salida a la derecha.

El nombre de la función, es decir, el tipo de función, aparece centrado dentro de la trama.

A la derecha del tipo de función se indica el número de ejecución *(véase página 411)* de la función.

Encima de la trama aparece el contador de función. El contador de función es el número correlativo de la función dentro de la sección actual. Los contadores de función no se pueden modificar.

Función elemental



En el caso de algunas funciones elementales, se puede ampliar la cantidad de entradas.

Bloque de función elemental

Los bloques de funciones elementales tienen estados internos. Si los valores de entrada son los mismos, el valor de la salida puede ser diferente cada vez que se ejecuta la función, por ejemplo, el valor de salida se incrementa para un contador.

Un bloque de función elemental se representa de forma gráfica por medio de una trama con varias entradas y salidas. En él, las entradas siempre aparecen a la izquierda y las salidas a la derecha. El nombre del bloque de función, es decir, el tipo de bloque de función, aparece centrado dentro de la trama. El nombre de la instancia aparece por encima de la trama.

Los bloques de funciones pueden disponer de varias salidas.

El nombre del bloque de función, es decir, el tipo de bloque de función, aparece centrado dentro de la trama.

A la derecha del tipo de bloque de función se indica el número de ejecución (*véase página 411*) del bloque de función.

El nombre de la instancia aparece por encima de la trama.

El nombre de instancia sirve para denominar de forma unívoca los distintos bloques de funciones de un proyecto.

El nombre de instancia se genera automáticamente y tiene la siguiente estructura:

`FBI_n`

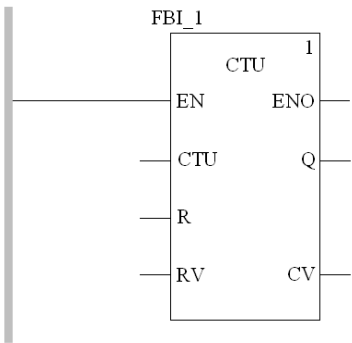
`FBI` = Instancia del bloque de función

`n` = Número correlativo del bloque de función en el proyecto

Este nombre generado automáticamente se puede modificar para mejorar la visión general. El nombre de instancia (32 caracteres como máximo) debe ser inequívoco dentro de un mismo proyecto; no se diferencia entre mayúsculas y minúsculas. El nombre de instancia debe cumplir la nomenclatura general.

NOTA: Según la norma CEI 61131-3, sólo se permite una letra como primer carácter de los nombres de instancias. Si desea utilizar cifras como primer carácter, deberá habilitar esa opción de forma explícita.

Bloque de función elemental

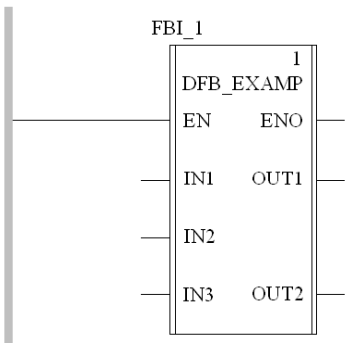


DFB

Los bloques de funciones derivados (DFB) presentan las mismas propiedades que los bloques de funciones elementales. Sin embargo, el usuario los crea en los lenguajes de programación FBD, LD, IL o ST.

La única diferencia con respecto a los bloques de funciones elementales es que los bloques de funciones derivados se representan gráficamente por medio de una trama con líneas verticales dobles.

Bloque de función derivado



Procedimiento

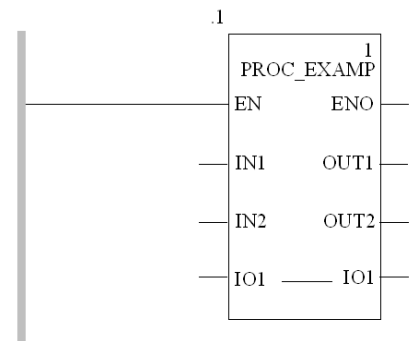
Los procedimientos son funciones desde el punto de vista técnico.

La única diferencia con respecto a las funciones elementales es que los procedimientos pueden tener más de una salida y admiten el tipo de datos VAR_IN_OUT.

Los procedimientos no se diferencian visualmente de las funciones elementales.

Los procedimientos son una ampliación de la norma CEI 61131-3 y se deben habilitar de forma explícita.

Procedimiento

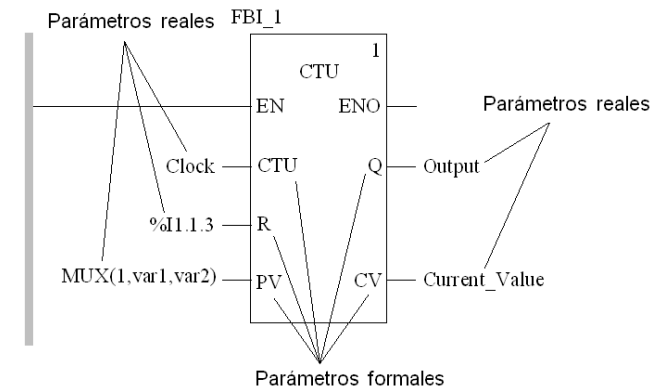


Parámetros

Para transferir valores al FFB o aplicarlos desde él, es necesario utilizar entradas y salidas. A éstas se les llama parámetros formales.

A los parámetros formales se vinculan objetos que contienen los estados actuales del proceso. Se conocen como parámetros reales.

Parámetros reales y formales:



Los valores del proceso se transfieren al FFB a través de los parámetros actuales durante el tiempo de ejecución del programa y se vuelven a emitir después del procesamiento.

A las entradas de FFB sólo se puede vincular un único objeto (parámetro real) del siguiente tipo:

- Contacto
- Variable
- Dirección
- Literal
- Expresión ST

Las expresiones ST de las entradas de FFB representan una ampliación de la norma CEI 61131-3 y se deben habilitar de forma explícita.

- Enlace

A las salidas de FFB se pueden vincular las siguientes combinaciones de objetos (parámetros reales):

- Una o más bobinas
- Uno o más contactos
- Una variable
- Una variable y una o varias conexiones (aunque no en el caso de salidas VAR_IN_OUT (véase página 390))
- Una dirección
- Una dirección y una o varias conexiones (aunque no en el caso de salidas VAR_IN_OUT (véase página 390))
- Una o varias conexiones (aunque no en el caso de salidas VAR_IN_OUT (véase página 390))

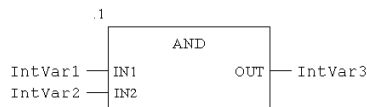
El tipo de datos del objeto que se va a enlazar debe coincidir con el tipo de datos de la entrada/salida de FFB. Si todos los parámetros reales están compuestos por literales, se elegirá un tipo de datos adecuado para el bloque de función.

Excepción: en el caso de entradas/salidas de FFB genéricas del tipo de datos ANY_BIT se pueden vincular objetos del tipo de datos INT o DINT (UINT ni UDINT).

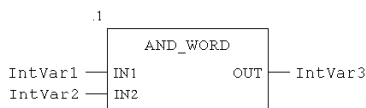
Se trata de una ampliación de la norma CEI 61131-3 y se debe habilitar de forma explícita.

Ejemplo:

Se admite:



No se admite:



(En este caso se debe utilizar `AND_INT`).

En principio, no es necesario asignar un parámetro real a cada parámetro formal. Pero esto no es válido para pins negados. A estos hay que asignarles siempre un parámetro real. También algunos tipos de parámetros formales lo requieren obligatoriamente. En la siguiente tabla encontrará estos tipos de parámetros formales.

Tabla de tipos de parámetros formales:

Tipo de parámetro	EDT	STRING	ARRAY	ANY_ARRAY	IODDT	STRUCT	FB	ANY
EFB: entrada	-	+	+	+	/	+	/	+
DFB: salida	-	-	+	/	/	-	/	+
EFB: VAR_IN_OUT	+	+	+	+	+	+	/	+
DFB: entrada	-	+	+	+	/	+	/	+
DFB: VAR_IN_OUT	+	+	+	+	+	+	/	+
EFB: salida	-	-	+	+	+	-	/	+
EF: entrada	-	-	+	+	+	+	+	+
EF: VAR_IN_OUT	+	+	+	+	+	+	/	+
EF: salida	-	-	-	-	-	-	/	-
Procedimiento: entrada	-	-	+	+	+	+	+	+
Procedimiento: VAR_IN_OUT	+	+	+	+	+	+	/	+
Procedimiento: salida	-	-	-	-	-	-	/	+
+ Parámetro real requerido obligatoriamente								
- Parámetro real no requerido obligatoriamente								
/ No es aplicable								

Los FFB que utilizan en las entradas parámetros reales que todavía no contienen ninguna asignación de valor trabajan con los valores iniciales de estos parámetros reales.

Si no se asigna un valor a un parámetro formal, durante la ejecución del bloque de función se utilizará el valor inicial. Si no se ha definido ningún valor inicial, se aplicará el valor predeterminado (0).

Si un parámetro formal no tiene asignado ningún valor y se han realizado varias instancias del bloque de función/DFB, las instancias que se ejecuten a partir de ese momento trabajarán con el valor antiguo.

Variables públicas

Además de las entradas y salidas, algunos bloques de funciones también disponen de variables públicas.

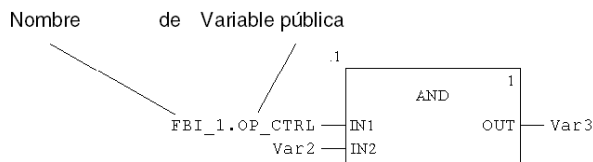
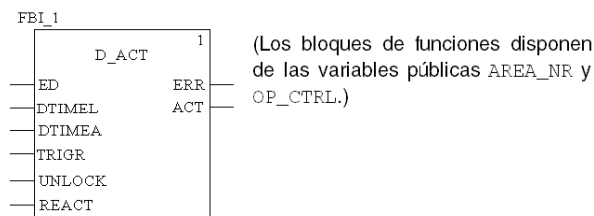
Estas variables sirven para transmitir valores estáticos (valores no influidos por el proceso) al bloque de función. Se utilizan para la parametrización del bloque de función.

Las variables públicas son una ampliación de la norma CEI 61131-3.

La asignación de valores a las variables públicas se realiza mediante sus valores iniciales.

La lectura de los valores de las variables públicas se realiza a través del nombre de instancia del bloque de función y a través del nombre de la variable pública.

Ejemplo:



Variables privadas

Además de las entradas, las salidas y las variables públicas, algunos bloques de funciones también disponen de las denominadas variables privadas.

Al igual que las variables públicas, las privadas se utilizan para transferir valores estadísticos (valores no influidos por el proceso) al bloque de función.

El programa de usuario no puede acceder a las variables privadas. Sólo se puede acceder a este tipo de variables mediante la tabla de animación.

NOTA: Los DFB intercalados se declaran como variables privadas del DFB principal. Por tanto, tampoco se puede acceder a sus variables a través de la programación, sino a través de la tabla de animación.

Las variables privadas constituyen un suplemento de la normativa CEI 61131-3.

Notas sobre la programación

Tenga en cuenta las siguientes indicaciones sobre la programación:

- Los FFB sólo se editan si se han conectado de forma directa o indirecta al riel de alimentación izquierdo.
- Si el FFB se va a ejecutar de forma condicional, se puede previncular la entrada EN por medio de contactos u otros FFB (consulte también *EN* y *ENO*, página 388).
- Las entradas y salidas booleanas se pueden invertir.
- Si se utilizan variables VAR_IN_OUT (véase página 390), se imponen condiciones especiales.
- Las instancias de bloques de funciones/DFB se pueden llamar varias veces (consulte también *Llamada múltiple de una instancia del bloque de función*, página 388).

Llamada múltiple de una instancia del bloque de función

Las instancias de bloques de funciones/DFB se pueden llamar varias veces, excepto las instancias de EFB de comunicaciones y de bloques de funciones/DFB con una salida ANY y ninguna entrada ANY, que sólo se pueden llamar una vez.

La llamada múltiple de una misma instancia de bloque de función/DFB resulta conveniente en los siguientes casos:

- Cuando el bloque de función/DFB no posee ningún valor interno o los valores internos no son necesarios para continuar el procesamiento.
En este caso, la llamada múltiple de una misma instancia de bloque de función/DFB permite ahorrar espacio en memoria, ya que el código del bloque de función/DFB sólo se carga una vez.
El bloque de función/DFB se procesa, por así decirlo, como una función.
- Cuando el bloque de función/DFB tiene valores internos y éstos se deben modificar en varios puntos del programa, por ejemplo, si el valor de un contador se debe aumentar en diversos puntos del programa.
En este caso, con la llamada múltiple de la misma instancia del bloque de función/DFB no es necesario guardar los resultados intermedios para continuar el procesamiento en otro lugar del programa.

EN y ENO

En todos los FFB se puede configurar una entrada EN y una salida ENO.

Si el valor de EN es "0", al llamar el FFB no se ejecutarán los algoritmos definidos por dicho FFB, y ENO se ajustará a "0".

Si el valor de EN es "1", al llamar el FFB se ejecutarán los algoritmos definidos por dicho FFB. Una vez que se han ejecutado los algoritmos correctamente, el valor de ENO pasa a "1". Si se produce un error al ejecutar estos algoritmos, ENO pasa a "0".

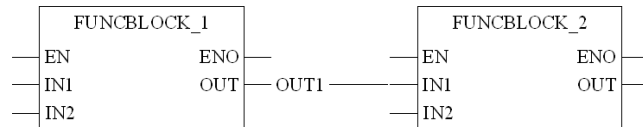
Si ENO pasa a "0" (porque EN = 0 o por un error durante la ejecución):

- Bloques de funciones
 - Manipulación de EN/ENO con bloques de funciones que (sólo) tienen un vínculo como parámetro de salida:



Si el valor de EN en FUNCBLOCK_1 se establece en "0", el enlace en la salida OUT de FUNCBLOCK_1 mantiene el estado que tenía en el último ciclo que se ejecutó correctamente.

- Gestión de EN/ENO con bloques de funciones que tienen una variable y un vínculo como parámetros de salida:



Si el valor de EN en FUNCBLOCK_1 se establece en "0", el enlace en la salida OUT de FUNCBLOCK_1 mantiene el estado que tenía en el último ciclo que se ejecutó correctamente. La variable OUT1 en el mismo pin puede mantener su estado anterior o puede modificarse desde fuera sin influir en el enlace. La variable y el enlace se guardan de forma independiente entre sí.

- Funciones/procedimientos

Según la definición de CEI 61131-3, las salidas de las funciones desactivadas (entrada EN ajustada a "0") no están definidas (esto es aplicable a los procedimientos).

A continuación se presenta una explicación de los estados de salida en este caso:

- Gestión de EN/ENO en las funciones o procedimientos que poseen (sólo) una conexión como parámetro de salida:



Si el valor de EN en FUNC_PROC_1 se establece en "0", el valor del enlace en la salida de OUT de FUNC_PROC_1 dependerá del ajuste del proyecto

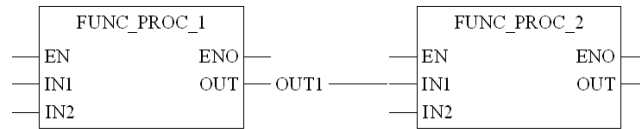
Mantener conexiones de salida en EF desactivadas disponible a partir de Unity Pro 4.1.

Si se establece este ajuste del proyecto en "0", el valor del enlace se establecerá en "0".

Si se establece este ajuste del proyecto en "1", el vínculo mantendrá el valor que tenía en el último ciclo ejecutado correctamente.

Para obtener información detallada, consulte el apartado *Mantener conexiones de salida en EF desactivadas* (véase *Unity Pro, Modalidades de funcionamiento*,).

- Gestión de EN/ENO en las funciones o procedimientos que poseen una variable y una conexión como parámetro de salida:



Si el valor de EN en FUNC_PROC_1 se establece en "0", el valor del enlace en la salida de OUT de FUNC_PROC_1 dependerá del ajuste del proyecto

Mantener conexiones de salida en EF desactivadas disponible a partir de Unity Pro 4.1.

Si se establece este ajuste del proyecto en "0", el valor del enlace se establecerá en "0".

Si se establece este ajuste del proyecto en "1", el vínculo mantendrá el valor que tenía en el último ciclo ejecutado correctamente.

Para obtener información detallada, consulte el apartado *Mantener conexiones de salida en EF desactivadas* (véase *Unity Pro, Modalidades de funcionamiento*,).

La variable OUT1 en el mismo pin puede mantener su estado anterior o puede modificarse desde fuera sin influir en el enlace. La variable y el enlace se guardan de forma independiente entre sí.

El comportamiento en la salida de los FFB no depende de si los FFB se han llamado sin EN/ENO o con EN = 1.

NOTA: Para bloques de funciones deshabilitados (EN = 0) con una función de tiempo interna (por ejemplo, bloques de funciones DELAY), el tiempo sigue avanzando, ya que se calcula con la ayuda de un reloj del sistema, y por tanto es independiente del ciclo de programa y de la apertura del bloque.

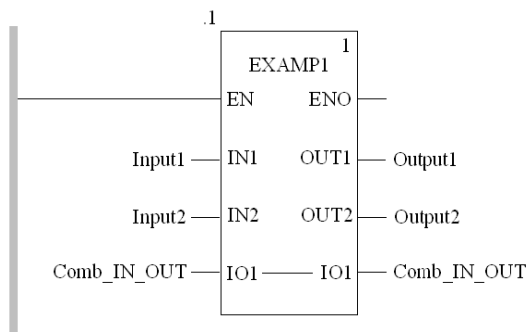
Variable VAR_IN_OUT

A menudo, los FFB se utilizan para leer una variable en la entrada (variables de entrada), procesarla y devolver los valores modificados de **esta** variable (variables de salida).

Este caso específico de variable de entrada/salida también se denomina variable VAR_IN_OUT.

El enlace de las variables de entrada y salida se representa en el FFB por medio de una línea.

Variable VAR_IN_OUT



Tenga en cuenta las siguientes particularidades cuando utilice los FFB con variables VAR_IN_OUT:

- Hay que asignar obligatoriamente una variable a todas las entradas VAR_IN_OUT.
- Los enlaces gráficos sólo permiten conectar salidas VAR_IN_OUT a entradas VAR_IN_OUT.
- Sólo es posible enlazar una conexión gráfica a una salida/entrada VAR_IN_OUT.
- En el caso de las salidas VAR_IN_OUT no es posible combinar una variable/dirección con conexiones gráficas.
- No puede haber literales ni constantes vinculadas a entradas/salidas VAR_IN_OUT.
- No se puede utilizar ninguna negación en las entradas/salidas VAR_IN_OUT.
- Es posible vincular a la entrada VAR_IN_OUT y a la salida VAR_IN_OUT distintas variables o componentes de variables. En tal caso, el valor de la variable o de su componente en la entrada se copiará en la variable o componente de variable en la salida.

Elementos de control

Introducción

Los elementos de control sirven para ejecutar saltos dentro de una sección LD y para el retorno al programa principal desde una subrutina (SRx) o desde un bloque de función derivado (DFB).

Los elementos de control ocupan una celda.

Elementos de control

Se encuentran disponibles los siguientes elementos de control.

Designación	Representación	Descripción
Salto		Si el estado de la conexión izquierda es 1, se ejecutará un salto a la etiqueta (en la sección actual). Para generar un salto incondicional, el objeto de salto se coloca directamente en la barra de alimentación izquierda. Para generar un salto condicional, el objeto de salto se coloca al final de una fila de contactos.
Etiqueta		Las etiquetas (destinos de salto) se representan como texto con dos puntos al final. El texto está limitado a 32 caracteres y debe ser unívoco en toda la sección. El texto debe cumplir las convenciones de nomenclatura general. Las etiquetas de salto sólo se pueden colocar en la primera celda situada inmediatamente junto a la barra de alimentación izquierda. Nota: Las etiquetas de salto no pueden "cortar" ninguna red, es decir, la línea imaginaria que une la etiqueta de salto con el borde derecho de la sección no puede estar cortada por ningún objeto. Esto también es válido para las conexiones booleanas y las conexiones FFB.
Retorno		Los objetos <code>RETURN</code> no pueden usarse en el programa principal. - En un DFB, un objeto <code>RETURN</code> fuerza el retorno al programa que llamó al DFB. - El resto de la sección DFB que contiene el objeto <code>RETURN</code> no se ejecuta. - Las siguientes secciones del DFB no se ejecutan. El programa que llamó al DFB se ejecutará después de volver del DFB. Si otro DFB llama al DFB, el DFB llamado se ejecutará después de volver. - En un SR, un objeto <code>RETURN</code> fuerza el retorno al programa que llamó al SR. - El resto de la sección SR que contiene el objeto <code>RETURN</code> no se ejecuta. El programa que llamó al SR se ejecutará después de volver del SR.

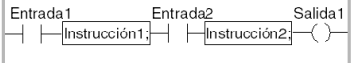
Bloques de funcionamiento y de comparación

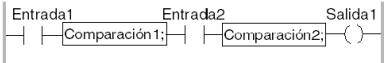
Introducción

Además de los objetos definidos en la norma CEI 61131-3, existen otros bloques para ejecutar instrucciones ST (*véase página 539*) y expresiones ST (*véase página 539*) y para operaciones de comparación sencillas. Estos bloques están disponibles exclusivamente en el lenguaje de programación LD.

Objetos

Se encuentran disponibles los siguientes objetos:

Designación	Representación	Descripción
Bloque de funcionamiento		Si el estado de la conexión izquierda es 1, se ejecutará la instrucción ST incluida en el bloque. Se admiten todas las instrucciones ST (<i>véase página 539</i>) excepto las instrucciones de control: ● (RETURN, ● JUMP, ● IF, ● CASE, ● FOR ● etc.). En el caso de los bloques de operación, el estado de la conexión izquierda se transmite a la conexión derecha independientemente del resultado de la instrucción ST. Un bloque puede contener hasta 4.096 caracteres. Si no se pueden representar todos los caracteres, se representa el inicio de la secuencia de caracteres seguido de puntos suspensivos (...). Un bloque de operación ocupa 1 fila y 4 columnas. Ejemplo:  En el ejemplo, Instrucción1 se ejecuta si Entrada1=1. Instrucción2 se ejecuta si Entrada1=1 y Entrada2=1 (el resultado de Instrucción1 no influye en la ejecución de Instrucción2). Salida1 será 1 cuando Entrada1 = 1 y Entrada2 = 1 (los resultados de Instrucción1 e Instrucción2 no influyen en el estado de Salida1).

Designación	Representación	Descripción
Bloque de comparación horizontal		Los bloques de comparación horizontal sirven para ejecutar una expresión de comparación (<, >, <=, >=, =, <>) en el lenguaje de programación ST. (Nota: Se puede conseguir la misma funcionalidad mediante las expresiones ST (véase página 539)). El bloque de comparación realiza un AND de su pin de entrada de la izquierda y el resultado de su condición de comparación asigna el resultado de este AND a su pin de salida incondicionalmente. Por ejemplo, si el estado de la conexión izquierda es 1 y el resultado de la comparación es 1, el estado de la conexión derecha será 1. Un bloque de comparación horizontal puede contener hasta 4.096 caracteres. Si no se pueden representar todos los caracteres, se representa el inicio de la secuencia de caracteres seguido de puntos suspensivos (...). Un bloque de comparación horizontal ocupa una fila y dos columnas. Ejemplo:  En el ejemplo, Comparación1 se ejecuta si Entrada1=1. Comparación2 se ejecuta si Entrada1=1, Entrada2=1, el resultado de Comparación1=1. Salida1 es 1 si Entrada1=1, Entrada2=1, el resultado de Comparación1=1 y el resultado de Comparación2=1.

Conexiones

Descripción

Las conexiones son vínculos entre objetos LD (contactos, bobinas, FFB, etc.).

Se distingue entre dos tipos de vínculos:

- Conexiones booleanas

Las conexiones booleanas están compuestas por uno o más segmentos que conectan objetos booleanos (contactos, bobinas) entre sí.

Las conexiones booleanas pueden ser de dos tipos:

- Conexiones booleanas horizontales

Las conexiones booleanas horizontales permiten la conexión en serie de contactos y bobinas.

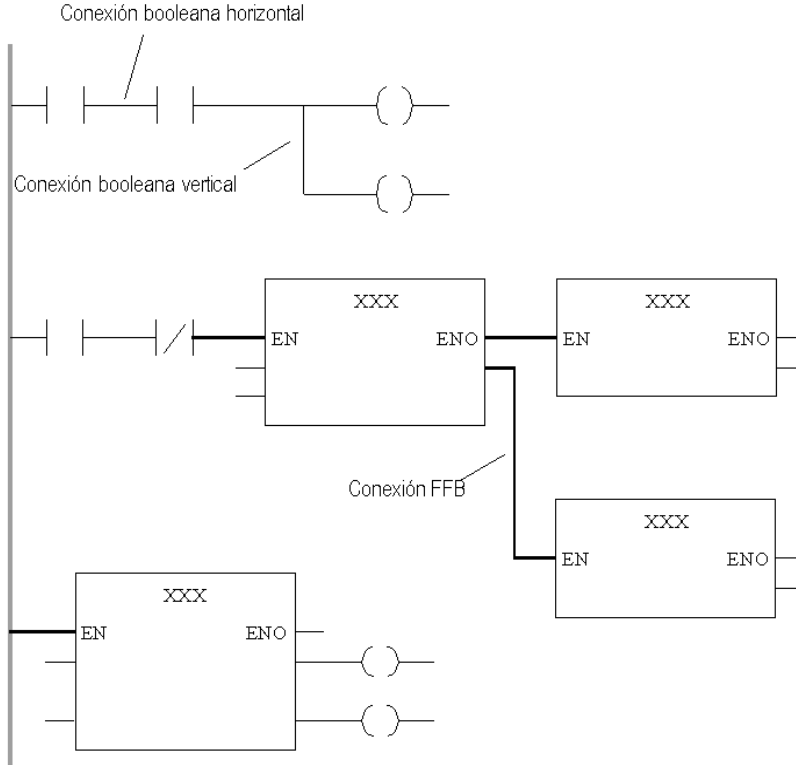
- Conexiones booleanas verticales

Las conexiones booleanas verticales permiten la conexión en paralelo de contactos y bobinas.

- Conexiones FFB

Las conexiones FFB están compuestas por una combinación de segmentos horizontales y verticales que conectan entradas/salidas FFB a otros objetos.

Conexiones



Notas generales sobre la programación

Tenga en cuenta las siguientes indicaciones generales sobre la programación:

- Los tipos de datos de las entradas/salidas que se vayan conectar deben coincidir.
- No se admiten las conexiones entre parámetros de longitud variable (por ejemplo, `ANY_ARRAY_INT`).
- Se pueden establecer varias conexiones a una salida (parte derecha de un contacto/una bobina, salida FFB). Sin embargo, sólo se puede establecer una única conexión a una entrada (parte izquierda de un contacto/una bobina, entrada FFB).

- Las entradas, bobinas y contactos no vinculados de FFB están ocupados de forma predeterminada con el valor "0".
- Con conexiones no se pueden configurar bucles, ya que en este caso no se puede determinar de forma inequívoca la secuencia de ejecución en la sección. Los bucles se deben resolver con parámetros actuales (consulte "*Bucles no permitidos, página 413*").

Notas sobre la programación de conexiones booleanas

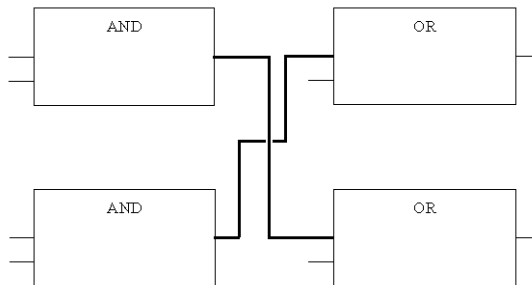
Notas sobre la programación de conexiones booleanas.

- **No** está permitido que las conexiones booleanas se solapen con otros objetos.
- En el caso de las conexiones booleanas, el flujo de señal (flujo de corriente) va de izquierda a derecha. Por este motivo no se permiten las conexiones en sentido inverso.
- Si dos conexiones booleanas se cruzan, se establecerá automáticamente un vínculo entre ambas conexiones. Dado que el cruce de conexiones booleanas no es posible, los vínculos no se marcan de ninguna forma especial.

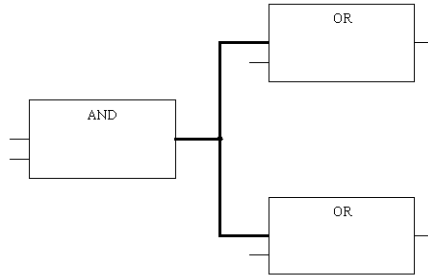
Notas sobre la programación de conexiones FFB

Notas sobre la programación de conexiones FFB.

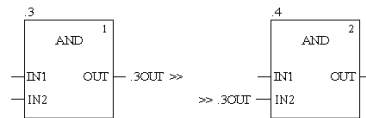
- Al menos una parte de una conexión FFB debe estar conectada a una entrada o salida FFB.
- Para diferenciarlas de las conexiones booleanas, las conexiones FFB se representan con un espesor de línea doble.
- En el caso de las conexiones FFB, el flujo de señal (flujo de corriente) va de la salida FFB a la entrada FFB, independientemente del sentido. Por este motivo, se permiten las conexiones en sentido inverso.
- Sólo se pueden conectar entradas FFB y salidas FFB entre sí. No es posible interconectar varias salidas FFB a la vez. Esto quiere decir que en LD no se puede realizar ninguna operación OR por medio de conexiones FFB.
- Está permitido que las conexiones FFB se solapen con otros objetos.
- Se permite el cruce de conexiones FFB. Los cruces se representan con una conexión "interrumpida".



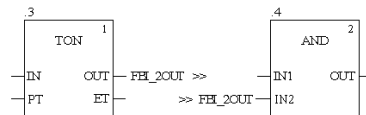
- En el caso de las conexiones FFB, los puntos de conexión entre varias conexiones FFB se marcan con un círculo relleno.



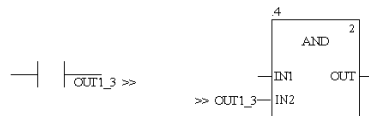
- Para evitar el cruce de conexiones, las conexiones FFB también se pueden representar en forma de conectores. El origen y el destino de la conexión FFB se identifican con un nombre unívoco dentro de la sección. El nombre del conector tiene la siguiente estructura en función del tipo de objeto de origen de la conexión:
 - En el caso de funciones: "Contador de función/parámetro formal" del origen de la conexión



- En el caso de bloques de función: "Nombre de instancia/parámetro formal" del origen de la conexión



- En el caso de contactos: "OUT1_número correlativo"



Conexiones verticales

La «conexión vertical» representa un caso especial de conexión. La conexión vertical sirve como OR lógico. En este tipo de conexiones OR son posibles 32 entradas (contactos) y 64 salidas (bobinas, conexiones).

Objeto de texto

Introducción

El texto puede ubicarse como objetos de texto en el lenguaje de diagrama de contactos (LD). El tamaño de estos objetos depende de la longitud del texto. En función del tamaño del texto, es posible ampliar el tamaño del objeto en más unidades de rejilla en dirección vertical y horizontal. Los objetos de texto pueden solaparse con otros objetos.

Detección de flancos

Introducción

En el reconocimiento de flancos se controla la transición de un bit de 0 -> 1 (flanco positivo) o de 1 -> 0 (flanco negativo).

Para ello, se compara el valor del bit en el ciclo anterior con su valor en el ciclo actual. En este caso, no sólo será necesario el valor actual sino también el anterior.

En el reconocimiento de flancos son necesarios dos bits (el valor actual y el anterior) en lugar de uno.

Puesto que el tipo de datos `BOOL` sólo proporciona un bit (valor actual), para el reconocimiento de flancos hay otro tipo de datos, `EBOOL` (`BOOL` ampliado). Además del reconocimiento de flancos, el tipo de datos `EBOOL` ofrece la posibilidad de forzar. Así, se debe almacenar también si el forzado de bits está activo o no.

El tipo de datos `EBOOL` almacena los siguientes datos:

- el valor actual del bit en *Bit de valor*
- el valor anterior del bit en *bit de registro*
(al comienzo de cada ciclo se copia el contenido del bit de valor en el bit de registro)
- Información sobre si el forzado del bit está activo en *bit de forzado*
(0 = forzado inactivo, 1 = forzado activo).

Restricciones para EBOOL

 ATENCIÓN FUNCIONAMIENTO INESPERADO DEL EQUIPO Para realizar una buena detección de flancos, <code>%M</code> debe actualizarse en cada ciclo de tarea. Al realizar una escritura unívoca, el flanco será infinito. Si no se siguen estas instrucciones pueden producirse lesiones personales o daños en el equipo.

Utilizando una variable `EBOOL` para que los contactos reconozcan flancos positivos (P) o negativos (N) o con una EF denominada RE o FE, deberá seguir las restricciones que se describen a continuación.

EBOOL con %M no escrito dentro de programa

Una variable `EBOOL` con una dirección `%M`, que no se escribe dentro del programa sino directamente, por ejemplo, mediante una tabla de animación, una pantalla de operador o un HMI, no funcionará de la manera esperada. El flanco es `TRUE` infinitamente porque el `%M` sólo se escribe una vez.

NOTA: Para evitar este problema, el %M tiene que escribirse al final de la tarea para actualizar la información del valor anterior.

El valor anterior sólo se actualiza cuando se escribe el bit %M, de manera que si escribe el bit sólo una vez, la detección de flanco será infinita.

Valor anterior	Valor actual	Detección de flanco	Descripción
0	0	0	estado 0 (antes de escribir el bit)
0	1	1	Escribir 1 en el bit (por ejemplo, mediante tabla de animación).
0	1	1	Si no vuelve a escribir, el flanco permanece infinitamente.
1	1	0	Vuelva a escribir 1 en el bit, el valor anterior se actualiza y la detección de flanco se establece en 0.

EBOOL con %M escrito dentro de programa

Para una variable EBOOL con una dirección %M, que se escribe dentro del programa, tiene que seguir las restricciones que se describen a continuación:

- No utilice el bit con una bobina SET o RESET. En este caso el valor anterior no se actualiza. De esta manera puede efectuar un flanco infinito.
- No escriba el bit de forma condicional. Una lógica simple como
`IF NOT %M1 THEN %M1 := TRUE; END_IF` lleva a un flanco infinito, porque sólo se escribe una vez.

EBOOL con %I

Para una variable EBOOL con una dirección %I debe seguir las restricciones que se describen a continuación:

- Al utilizar multitarea, la prueba de flanco %I debe efectuarse en la tarea en que se actualiza. Debe evitarse el uso de la detección de flanco de un %I programado en una tarea de mayor prioridad.

Ejemplo: Si tiene una tarea rápida, que actualiza un %I, no utilice una detección de flanco en la tarea MAST. Dependiendo de la planificación, puede detectar el flanco o no.

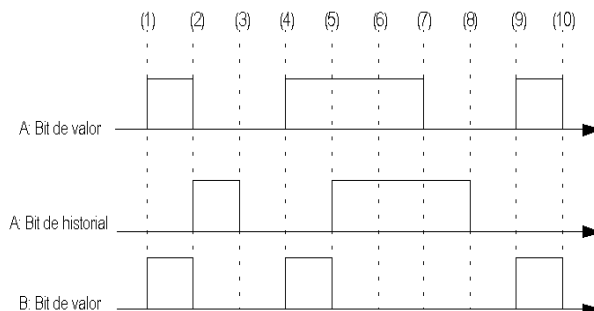
Detección de flancos positivos

Para detectar los flancos positivos se utiliza un contacto para el reconocimiento de flancos positivos. En el caso de este contacto, la conexión derecha es igual a 1 durante un ciclo del programa siempre que se produzca una transición de 0 a 1 del parámetro real correspondiente (A) y, al mismo tiempo, el estado de la conexión izquierda sea 1. De lo contrario, el estado de la conexión derecha es 0.

En el ejemplo se detecta un flanco positivo de la variable A y, por lo tanto, se establece B durante un ciclo.



Siempre que el bit de valor de A sea igual a 1 y el bit de registro sea igual a 0, se establece B en 1 durante un ciclo (ciclos 1, 4 y 9).



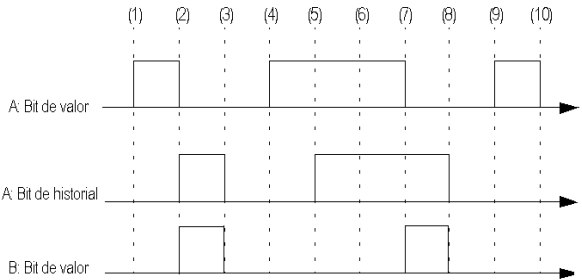
Detección de flancos negativos

Para detectar los flancos negativos se utiliza un contacto para el reconocimiento de flancos negativos. En el caso de este contacto, la conexión derecha es igual a 1 durante un ciclo del programa siempre que se produzca una transición de 1 a 0 del parámetro real correspondiente (A) y, al mismo tiempo, el estado de la conexión izquierda sea 1. De lo contrario, el estado de la conexión derecha es 0.

En el ejemplo se detecta un flanco negativo de la variable A y, por lo tanto, se establece B durante un ciclo.



Siempre que el bit de valor de A sea igual a 0 y el bit de registro sea igual a 1, se establece B en 1 durante un ciclo (ciclos 2 y 8).



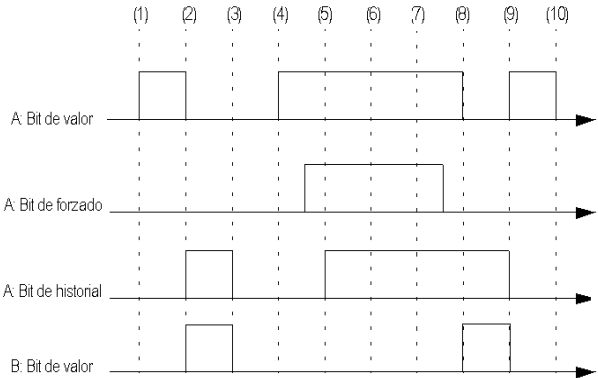
Forzado de bits

Al forzar bits, se sobrescribe el valor de la variable averiguado por lógica con el valor de forzado.

En el ejemplo se detecta un flanco negativo de la variable A y, por lo tanto, se establece B durante un ciclo.



Siempre que el bit de valor o bit de forzado de A sea igual a 0 y el bit de registro sea igual a 1, se establece B en 1 durante un ciclo (ciclos 1 y 8).



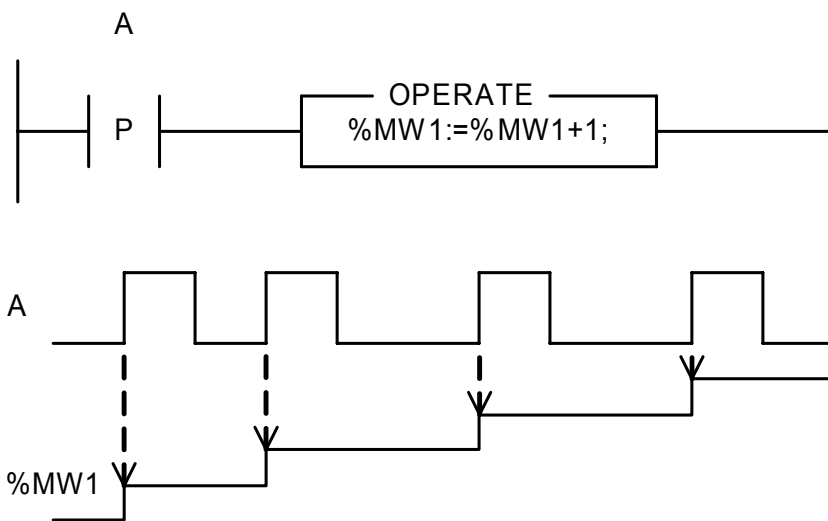
Utilización de variables BOOL y EBOOL

El comportamiento del reconocimiento de flanco con los tipos de variable `BOOL` o `EBOOL` puede ser diferente:

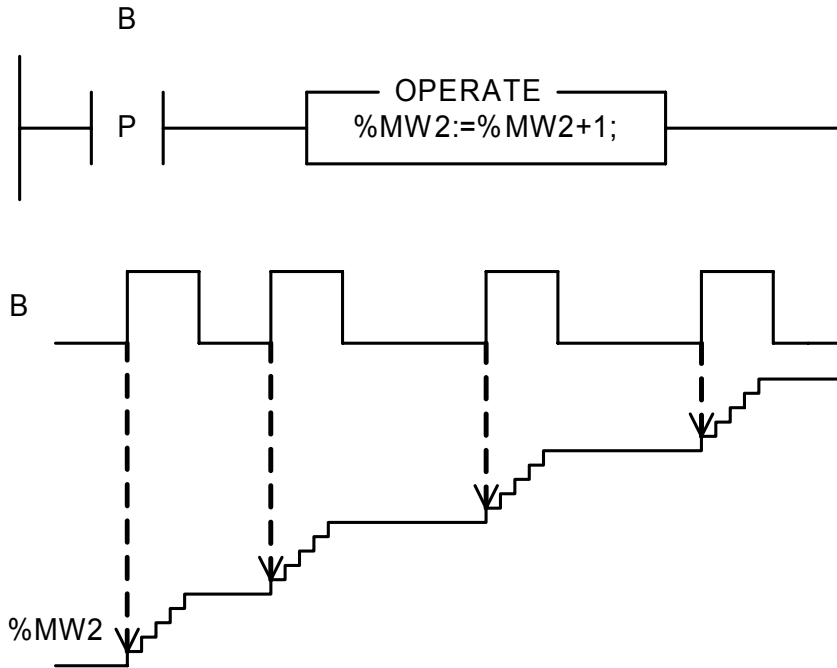
- Cuando se utiliza una variable `BOOL`, el sistema gestiona el registro permitiendo la detección de flanco durante la ejecución de contacto.
- Cuando se utiliza una variable `EBOOL`, el bit de registro se actualiza durante la ejecución de bobina.

En los ejemplos siguientes se muestra el comportamiento distinto en función del tipo de variable.

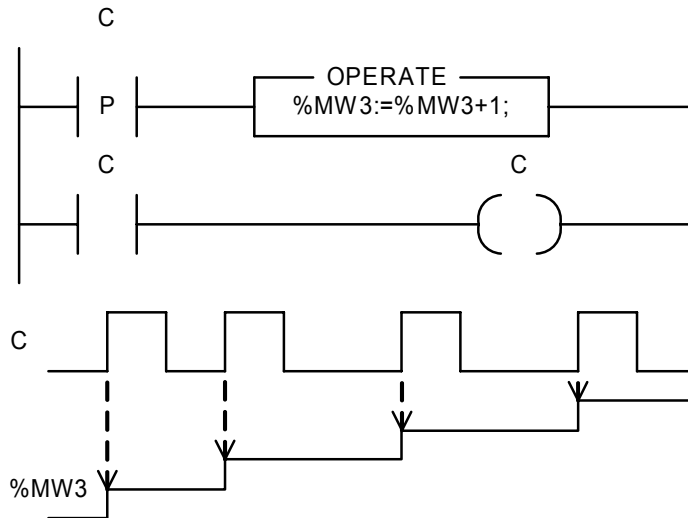
La variable `A` se define como `BOOL`, cuando `A` se establece en 1, `%MW1` se incrementa en 1.



La variable B se define como EBOOL, el comportamiento es distinto cuando se compara con la variable A. Mientras B está establecido en 1, %MW2 se incrementa en 1 porque el bit de registro no está actualizado.



La variable **C** se define como **EBOOL**, el comportamiento es idéntico al de la variable **A**. El bit de registro se actualiza.

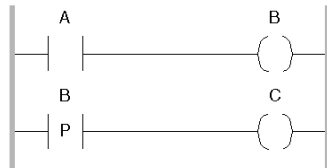


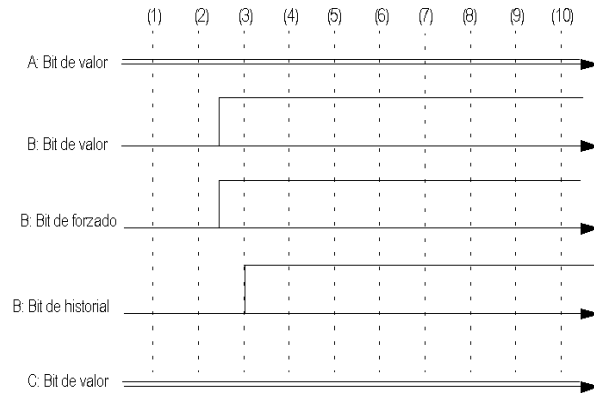
Al forzar las bobinas se puede producir la pérdida de la detección de flancos

Al forzar las bobinas se puede producir la pérdida de la detección de flancos.

En el ejemplo, si **A** es igual a 1, **B** debe ser igual a 1 y, en el caso de un flanco ascendente de **A**, la bobina **B** debe establecerse durante un ciclo.

En este ejemplo, la variable **B** se asigna a la bobina primero y, a continuación, al contacto para el reconocimiento de flancos positivos.





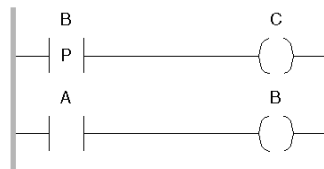
Al comienzo del segundo ciclo, el bit de valor de B es igual a 0. Al forzar B durante este ciclo, el bit de forzado y el bit de valor se establecen en 1. Al procesar la primera línea de la lógica durante el tercer ciclo, el bit de registro de la bobina (B) también se establecerá en 1.

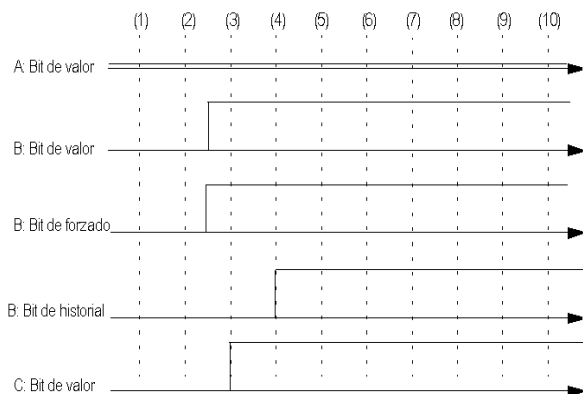
Problema:

En el reconocimiento de flancos (comparación del bit de valor y el bit de registro), no se puede detectar ningún flanco de la segunda línea de la lógica ya que, debido a la actualización, el bit de valor y el bit de registro de la línea 1 de B son siempre idénticos.

Solución:

En este ejemplo, la variable B tiene asignado el contacto para el reconocimiento de flancos positivos primero y, a continuación, la bobina.





Al comienzo del segundo ciclo, el bit de valor de **B** es igual a 0. Al forzar **B** durante este ciclo, el bit de forzado y el bit de valor se establecen en 1. Al procesar la primera línea de la lógica durante el tercer ciclo, el bit de registro de la conexión (**B**) permanecerá en 0.

El reconocimiento de flancos detecta la diferencia entre el bit de valor y el bit de registro y establece la bobina (**C**) en 1 durante un ciclo.

La utilización de bobinas de establecimiento o restablecimiento puede provocar la pérdida de la detección de flancos

En el caso de variables **EBOOL**, la utilización de bobinas de establecimiento o restablecimiento puede provocar la pérdida del reconocimiento del flanco.

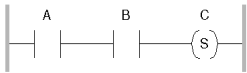
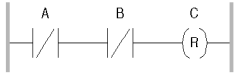
La variable de la bobina de establecimiento o restablecimiento (en el ejemplo, la variable **C**) siempre se ve afectada por el valor de la conexión izquierda.

Si el valor de la conexión izquierda es 1, el bit de valor (en el ejemplo, la variable **C**) se copia en el bit de registro y el bit de valor se establece en 1.

Si el valor de la conexión izquierda es 0, el bit de valor (en el ejemplo, la variable **C**) se copia en el bit de registro y el bit de valor no cambia.

Esto significa que, independientemente del valor que tenga la conexión izquierda de la bobina de establecimiento o restablecimiento, el bit de registro siempre se actualiza.

En el ejemplo se detecta un flanco positivo de la variable C y por lo tanto se establece en D durante un ciclo.

Línea de código	Comportamiento en LD	Equivalencia en ST
1	Situación de salida: C = 0, Bit de registro = 0  A = 1, B = 1, C = 1, Bit de registro = 0	<pre>IF A AND B THEN C := 1; ELSE C := C; END_IF;</pre>
2	 A = 1, B = 1, C = 1, Registro = 1	<pre>IF NOT(A) AND NOT(B) THEN C := 0; ELSE C := C; END_IF;</pre>
3	 C = 1, Registro = 1 D = 0, ya que el bit de valor y el bit de registro de C son idénticos. El flanco ascendente de C, ejecutado en la línea de código 1, no es reconocido por el código en la línea 2, ya que provoca una actualización del bit de registro. (Si la condición no es verdadera, al valor actual de C se le volverá a asignar C. Consulte la instrucción ELSE en la línea de código 2 del ejemplo ST).	-

Secuencia de ejecución y flujo de señal

Secuencia de ejecución de redes

Para la secuencia de ejecución de redes se aplican las siguientes reglas:

- La ejecución de una sección se realiza red a red en función de las conexiones de objetos de arriba a abajo.
- No es posible configurar bucles mediante conexiones, ya que en este caso no se puede determinar de forma inequívoca la secuencia de ejecución en la sección. Los bucles se deben resolver con parámetros actuales (consulte "*Configuración de bucles, página 413*").
- La secuencia de ejecución de las redes que sólo están conectadas entre sí mediante barra de alimentación izquierda está determinada por la secuencia gráfica (de arriba a abajo) en la que dichas redes están conectadas a la barra de alimentación izquierda. Esto no es aplicable si la secuencia se modifica mediante elementos de control.
- El cálculo de una red finaliza completamente antes de que comience el cálculo de la red siguiente.
- Ningún elemento de una red se considera calculado mientras no se haya calculado el estado de todas las entradas de dicho elemento.
- El cálculo de una red sólo se considera finalizado cuando se han calculado todas las salidas de dicha red. Esto también es válido si la red contiene uno o más elementos de control.

Flujo de señal dentro de una red

Para el flujo de señal dentro de una red (circuito eléctrico) se aplican las siguientes reglas:

- El flujo de señal con conexiones booleanas va:
 - De izquierda a derecha en caso de conexiones booleanas horizontales.
 - De arriba a abajo en caso de conexiones booleanas verticales.
- En el caso de las conexiones FFB, el flujo de señal va de la salida FFB a la entrada FFB, independientemente del sentido.
- El cálculo de un FFB no se inicia hasta que no se hayan calculado todos los elementos (salidas FFB, etc.) que estén conectados a sus entradas.
- La secuencia de ejecución de los FFB que estén conectados a varias salidas del mismo FFB es de arriba a abajo.
- La secuencia de ejecuciones de los objetos no está influida por su posición dentro de la red.
- La secuencia de ejecución de FFB se muestra como número de ejecución sobre el FFB.

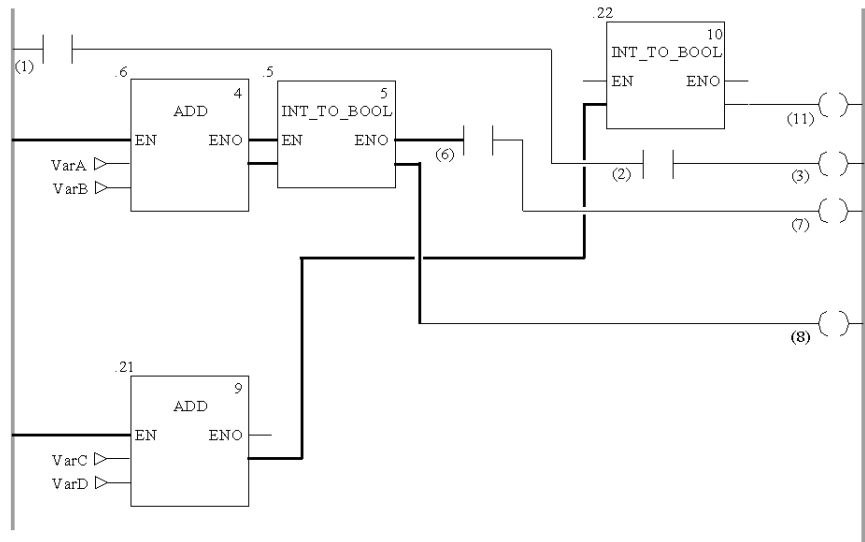
Prioridades

Prioridades a la hora de determinar el flujo de señal dentro de una sección.

Prioridad	Regla	Descripción
1	Conexión	Las conexiones tienen el nivel de prioridad más alto a la hora de determinar el flujo de señal dentro de una sección LD.
2	Red a red	El cálculo de una red finaliza completamente antes de que comience el cálculo de la red siguiente.
3	Secuencia de las salidas	Las salidas del mismo bloque de función o las salidas de las conexiones verticales se calculan de arriba a abajo.
4	Circuito a circuito	Nivel de prioridad más bajo. La secuencia de ejecución de las redes que sólo están conectadas entre sí mediante barra de alimentación izquierda está determinada por la secuencia gráfica (de arriba a abajo) en la que dichas redes están conectadas a la barra de alimentación izquierda. (Esto sólo es válido si no se aplica ninguna de las demás reglas).

Ejemplo

Ejemplo de la secuencia de ejecución de objetos en una sección LD



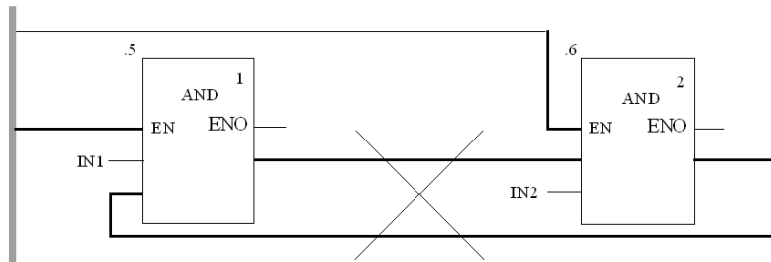
NOTA: Los números de ejecución de contactos y bobinas no se muestran. Sólo se incluyen en el gráfico con fines explicativos.

Configuración de bucles

Bucles no permitidos

No es posible configurar bucles exclusivamente a base de conexiones ya que de este modo no se puede determinar el flujo de señal de forma unívoca (la salida de un FFB es la entrada del FFB siguiente y la salida de éste es, a su vez, la entrada del primero).

Bucle no permitido a través de conexiones



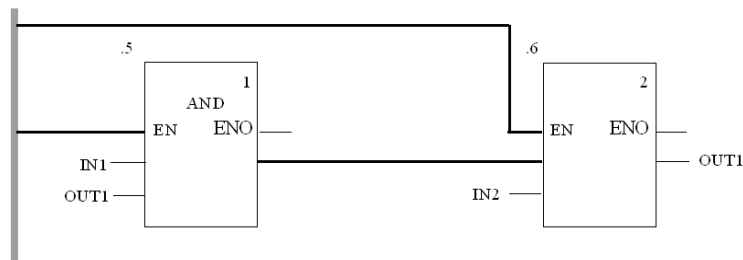
Generación a través de un parámetro real

Una lógica de este tipo se debe resolver con ayuda de una variable de realimentación para que el flujo de señal se pueda determinar de forma unívoca.

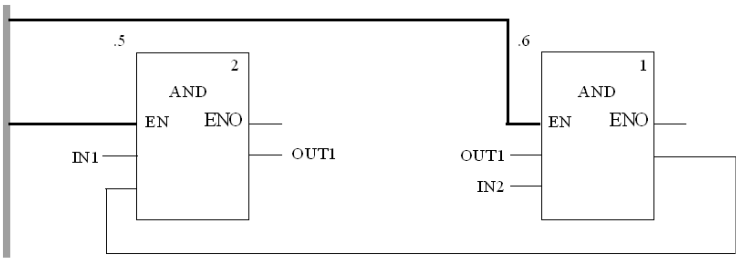
Las variables de realimentación se deben inicializar. El valor inicial se utiliza durante la primera ejecución de la lógica. Una vez ejecutado, el valor inicial se sustituye por el valor actual.

Tenga en cuenta en ambos casos la secuencia de ejecución (número entre paréntesis detrás del nombre de instancia) de los dos bloques.

Bucle resuelto a través de un parámetro actual: caso 1



Bucle resuelto a través de un parámetro actual: caso 2



Modificación de la secuencia de ejecución

Introducción

La secuencia de ejecución de las redes y de los objetos incluidos dentro de una red está definida por una serie de reglas (*véase página 411*).

En la mayoría de los casos resulta recomendable modificar la secuencia de ejecución propuesta por el sistema.

Existen las siguientes posibilidades para determinar/modificar la secuencia de ejecución de las redes:

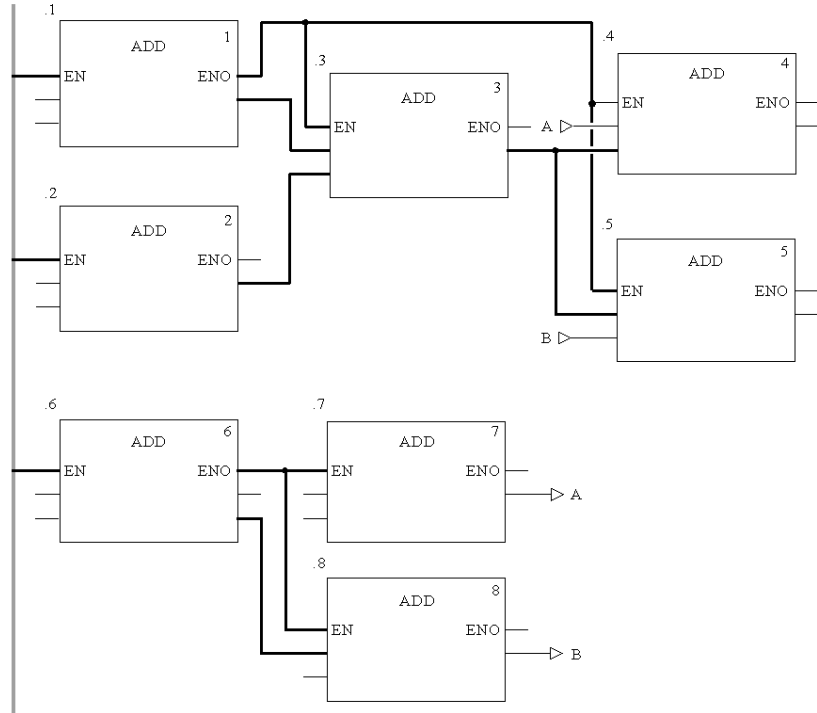
- Utilización de conexiones en lugar de parámetros actuales
- Posición de las redes

Existen las siguientes posibilidades para determinar/modificar la secuencia de ejecución dentro de las redes:

- Posición de los objetos

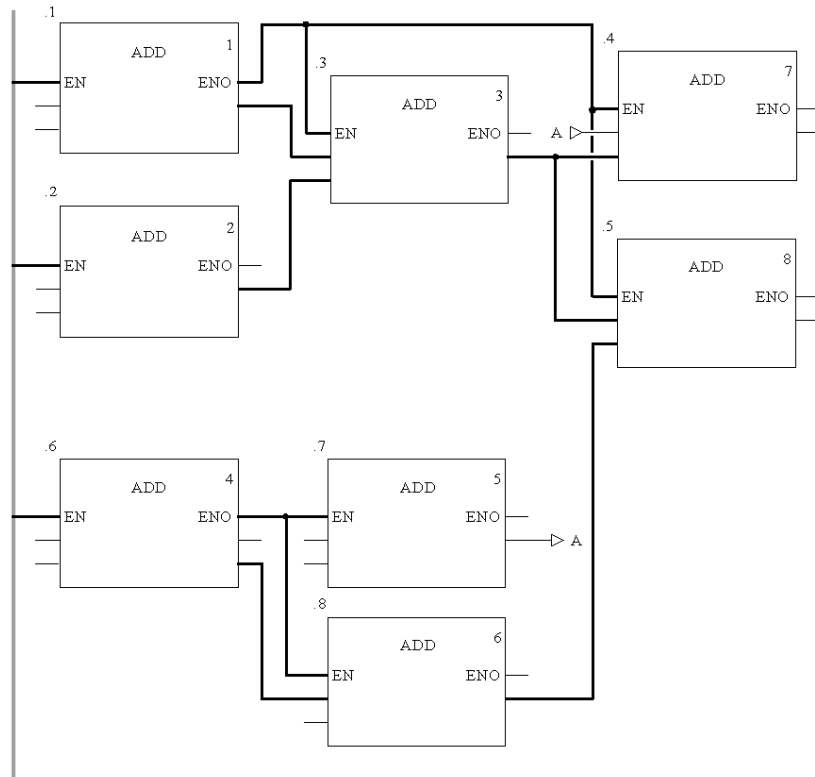
Situación de salida

En la figura siguiente se muestran dos redes cuya secuencia de ejecución está determinada únicamente por su posición dentro de la sección, con independencia de que los bloques .4/.5 y .7/.8 requieran otra secuencia de ejecución.



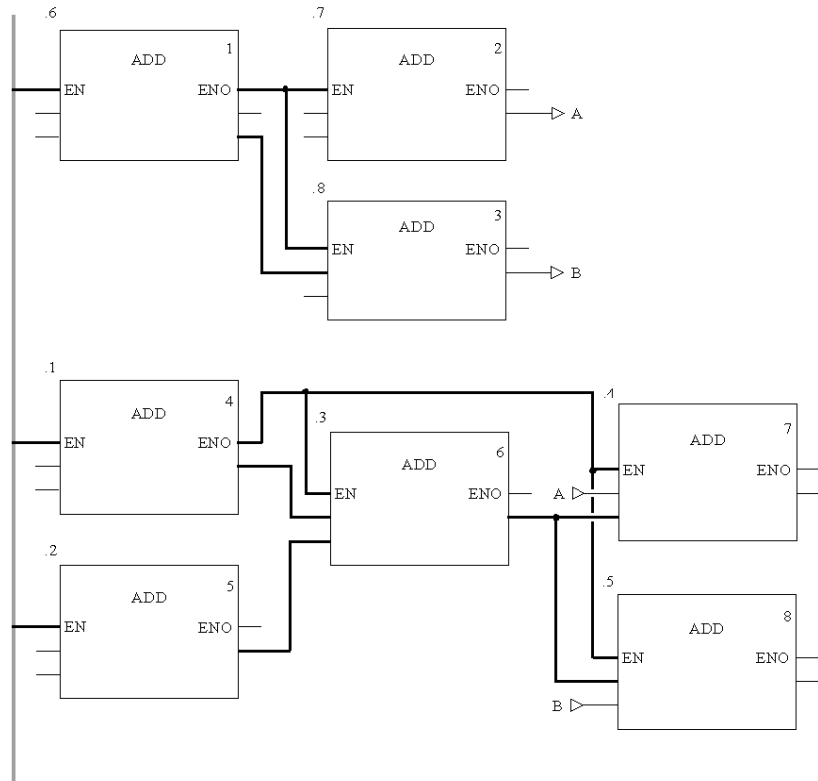
Conexión en lugar de parámetros actuales

Utilizando una conexión en lugar de una variable, ambas redes se ejecutan en la secuencia correcta (consulte también "*Situación de salida, página 416*").



Posición de las redes

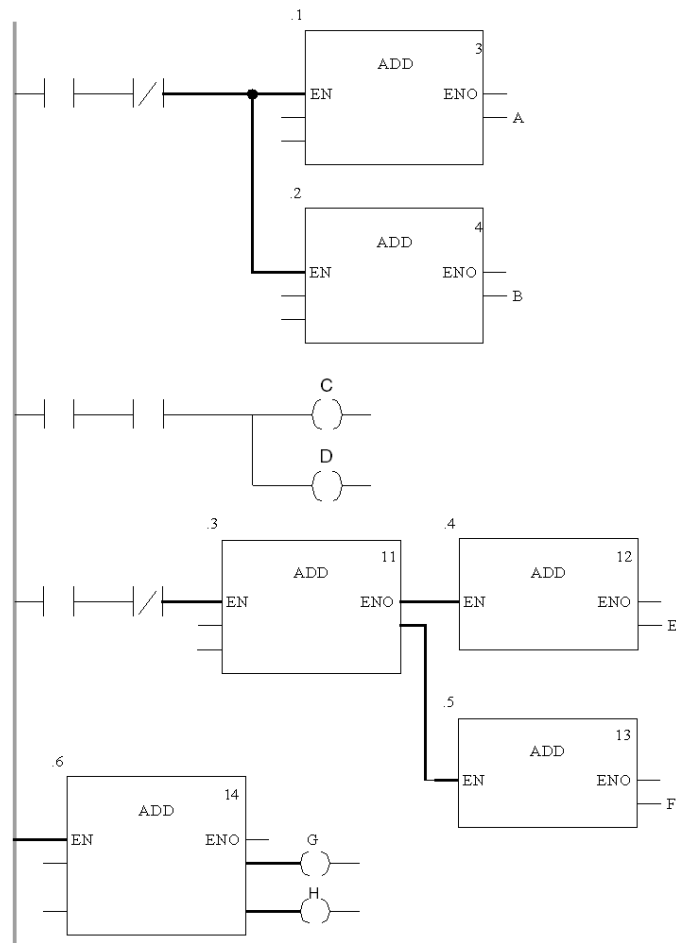
La secuencia de ejecución adecuada se puede conseguir modificando la posición de las redes (consulte también "*Situación de salida, página 416*").



Posición de los objetos

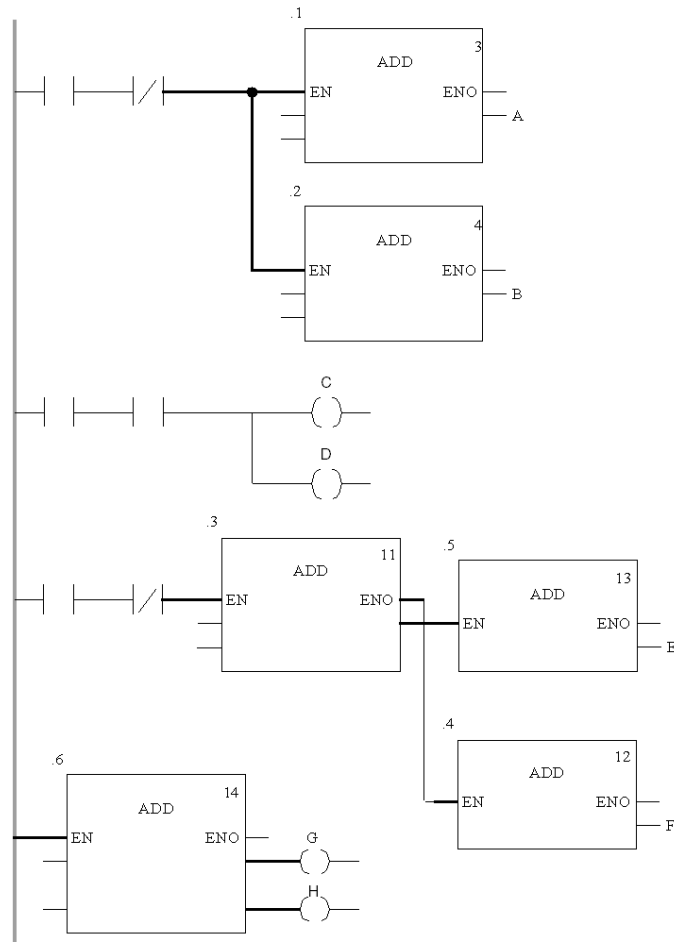
La posición de los objetos sólo influye sobre la secuencia de ejecución si varias entradas (conexión izquierda de contactos/bobinas, entradas FFB) están conectadas a la misma salida del objeto "invocante" (conexión derecha de contactos/bobinas, salidas FFB) (consulte también *Situación de salida, página 416*).

Situación de salida



En la primera red, se han intercambiado las posiciones de los bloques .1 y .2. En este caso (origen común de ambas entradas de bloque) también se intercambia la secuencia de ejecución de ambos bloques (procesamiento de arriba a abajo). Lo mismo es válido para el intercambio de las bobinas C y D en la segunda red.

En la tercera red, se han intercambiado las posiciones de los bloques .4 y .5. En este caso (origen distinto de las entradas de bloque), la secuencia de ejecución de ambos bloques no se intercambia (procesamiento en la secuencia de las salidas de bloque invocantes). Lo mismo es válido para el intercambio de las bobinas G y H en la última red.



SFC, lenguaje de ejecución secuencial

13

Vista general

En este capítulo, se describe el lenguaje de ejecución secuencial SFC según CEI 611311.

Contenido de este capítulo

Este capítulo contiene las siguientes secciones:

Sección	Apartado	Página
13.1	Generalidades sobre el lenguaje de ejecución secuencial SFC	422
13.2	Paso y paso de macro	428
13.3	Acción y sección de acción	438
13.4	Transición y sección de transición	445
13.5	Salto	450
13.6	Conexión	451
13.7	Bifurcaciones y conjunciones	453
13.8	Objeto de texto	458
13.9	Single-Token	459
13.10	Multi-Token	471

13.1 Generalidades sobre el lenguaje de ejecución secuencial SFC

Vista general

En esta sección se ofrece una vista general sobre el lenguaje de ejecución secuencial SFC.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Generalidades sobre el lenguaje de ejecución secuencial SFC	423
Reglas de conexión	427

Generalidades sobre el lenguaje de ejecución secuencial SFC

Introducción

El lenguaje de ejecución secuencial SFC (Gráfica de función secuencial), que cumple con la norma CEI 61131-3, se describe en esta sección.

Estructura de un control de secuencia

En Unity Pro, un control de secuencia conforme a CEI está compuesto por secciones SFC (nivel superior), secciones de transición y secciones de acción.

Estas secciones SFC sólo se admiten en la tarea master del proyecto. En las demás tareas o DFB no se pueden utilizar secciones SFC.

Cada sección SFC contiene en Single-Token exactamente una red SFC (cadena secuencial).

En Multi-Token, una sección SFC puede contener una o varias redes SFC independientes.

Objetos

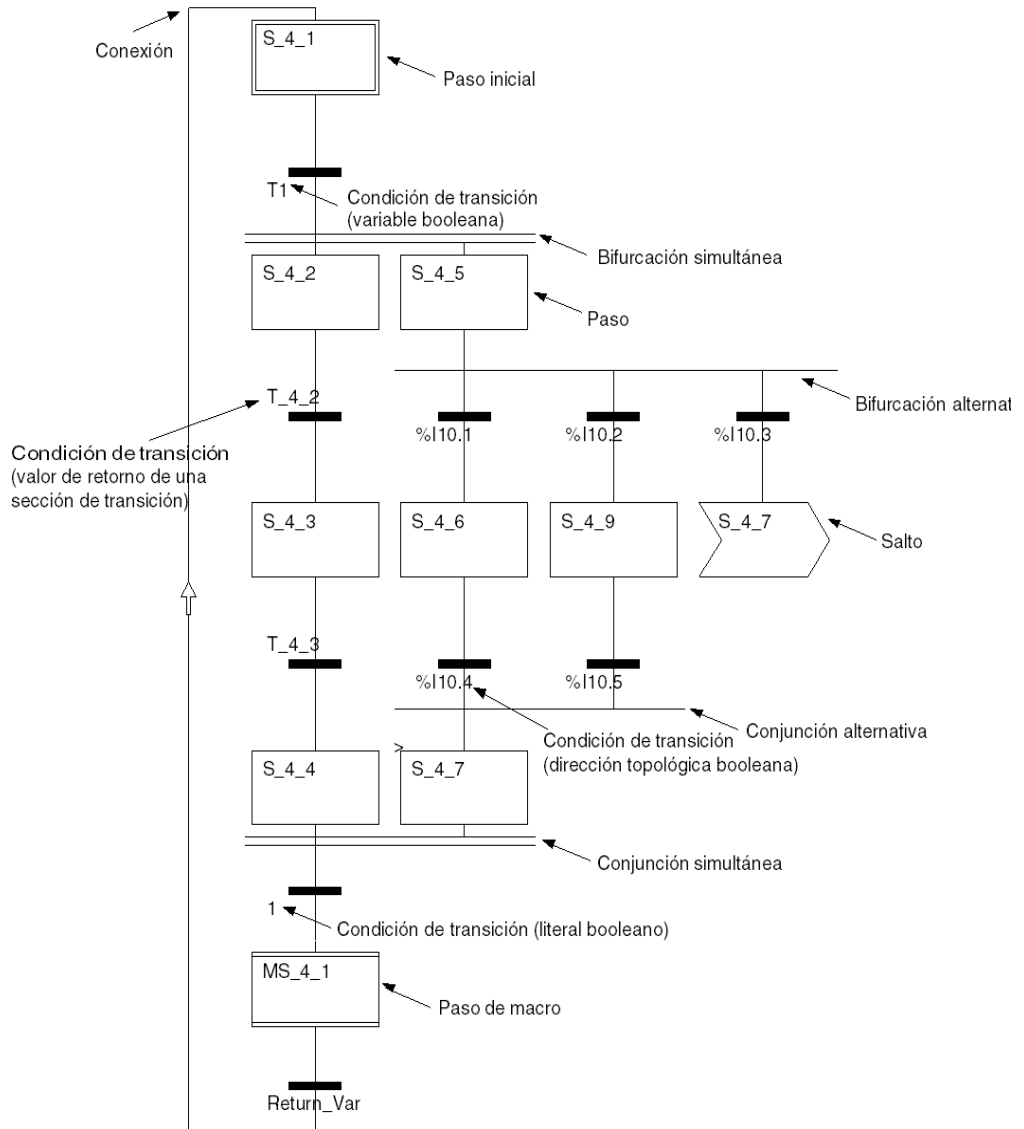
Una sección SFC ofrece los siguientes objetos para la creación del programa:

- Paso (*véase página 429*)
- Paso macro (subcadena de pasos intercalada) (*véase página 433*)
- Transición (condición de transición) (*véase página 446*)
- Salto (*véase página 450*)
- Conexión (*véase página 451*)
- Bifurcación alternativa (*véase página 454*)
- Conjunción alternativa (*véase página 454*)
- Bifurcación simultánea (*véase página 456*)
- Conjunción simultánea (*véase página 456*)

La lógica de la sección se puede comentar por medio de objetos de texto (consulte "*Objeto de texto, página 458*").

Representación de una sección SFC

Representación:



Estructura de una sección SFC

Una sección SFC es una "máquina de estado", es decir, el estado se representa mediante el paso activo, y las transiciones reproducen el comportamiento de conexión/modificación. Los pasos y transiciones se vinculan unos con otros mediante conexiones direccionales. No es posible vincular dos pasos directamente en ningún caso; siempre están separados por una transición. Las secuencias de los estados de señal activos tienen lugar a lo largo de las conexiones direccionales, disparadas por medio de la conexión de una transición. La dirección de la secuencia de una cadena sigue las conexiones direccionales y transcurre desde la parte interior del paso previo hasta la parte superior del paso siguiente. Las bifurcaciones se procesan de izquierda a derecha.

A cada paso le corresponden cero o más acciones. A cada transición le corresponde una condición de transición.

La última transición de la cadena está ligada siempre a otro paso de la cadena (a través de una conexión gráfica o un símbolo de salto), de forma que exista un círculo cerrado. Las cadenas de pasos se ejecutan de forma cíclica.

Variable SFCCHART_STATE

Si se crea una sección SFC, se le asigna automáticamente una variable del tipo de datos `SFCCHART_STATE`. La variable generada siempre tendrá el nombre de la sección SFC a la que pertenezca.

Esta variable sirve para asignar los bloques de control SFC a la sección SFC que se vaya a controlar.

Regla de token

El comportamiento de una red SFC está condicionado considerablemente por la elección de la cantidad de tokens, es decir, de la cantidad de pasos activos.

Es posible un comportamiento unívoco con un solo token (Single-Token). (Bifurcaciones simultáneas cada una con un token activo [paso] por bifurcación como un sólo token). Esto corresponde a una cadena de pasos tal y como se describe en la normativa CEI 61131-3).

Una cadena de pasos con una cantidad máxima de pasos activos definida por el usuario (Multi-Token) aumenta el grado de libertad. Las limitaciones para garantizar la univocidad e impedir el bloqueo se suavizan/anulan, y el usuario debe asegurarlas por sí mismo. Las cadenas de pasos con Multi-Token no cumplen la norma CEI 61131-3.

Tamaño de la sección

- Una sección SFC está compuesta por una ventana con una sola página.
- Por motivos de rendimiento, es sumamente recomendable crear menos de 100 secciones SFC en un proyecto (las secciones macro no se cuentan).

- La ventana tiene una rejilla lógica de fondo con 200 líneas y 32 columnas.
- Todos los pasos, transiciones y saltos requieren una celda.
- Las bifurcaciones y las conjunciones no necesitan disponer de su propia celda, sino que se insertan en la celda del paso o de la transición correspondiente.
- Por cada sección SFC (incluyendo todas sus secciones de macro) se puede ubicar un máximo de 1.024 pasos.
- Por cada sección SFC (incluyendo todas sus secciones de macro) puede haber un máximo de 100 pasos activos (Multi-Token).
- Por cada sección SFC se puede activar al mismo tiempo un máximo de 64 pasos (Multi-Token) manualmente.
- Se puede asignar un máximo de 20 acciones a cada paso SFC.
- Son 8 los niveles de anidamiento de macros, es decir, paso de macro a paso de macro.

Conformidad CEI

La descripción de la conformidad del lenguaje de programación SFC con la norma CEI figura en "Conformidad CEI (*véase página 693*)".

Reglas de conexión

Reglas de conexión

La tabla indica qué salidas y entradas de objetos se pueden conectar entre sí.

De salida de objeto de	A entrada de objeto de
Paso	Transición
	Bifurcación alternativa
	Conjunción simultánea
Transición	Paso
	Salto
	Bifurcación simultánea
	Conjunción alternativa
Bifurcación alternativa	Transición
Conjunción alternativa	Paso
	Salto
	Bifurcación simultánea
	Conjunción alternativa
Bifurcación simultánea	Paso
	Salto
	Conjunción alternativa (sólo con Multi-Token (véase página 471))
Conjunción simultánea	Transición
	Bifurcación alternativa (sólo con Multi-Token (véase página 471))
	Conjunción alternativa

13.2 Paso y paso de macro

Vista general

En este apartado se describen los objetos de paso y de paso de macro del lenguaje de ejecución secuencial SFC.

Contenido de esta sección

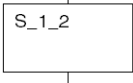
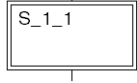
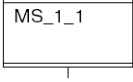
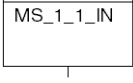
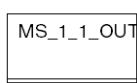
Esta sección contiene los siguientes apartados:

Apartado	Página
Paso	429
Pasos de macro y secciones de macro	433

Paso

Tipos de pasos

Existen los siguientes tipos de pasos.

Tipo	Representación	Descripción
Paso "normal"		Un paso se activa cuando el paso anterior se convierte en inactivo (debe transcurrir un tiempo de retardo definido) y se cumple la transición preconectada. Normalmente, un paso se convierte en inactivo cuando transcurre el tiempo de retardo definido y se cumple la condición postconectada. En el caso de las conjunciones simultáneas, todos los pasos anteriores deben cumplir estas condiciones. A cada paso le corresponden cero o más acciones. Los pasos sin acción se denominan pasos de espera.
Paso inicial		El estado inicial de una cadena de ejecución está caracterizado por el paso inicial. Tras la inicialización del proyecto o de la cadena de ejecución, el paso inicial está activo. A los pasos iniciales no se les suele asignar ninguna acción. En Single-Token (conforme a IEC 61131-3), sólo se admite un único paso inicial por secuencia. En Multi-Token se admite una cantidad definible de pasos iniciales (de 0 a 100).
Paso de macro		Consulte " <i>Paso de macro, página 433</i> ".
Paso de entrada		Consulte " <i>Paso de entrada, página 433</i> ".
Paso de salida		Consulte " <i>Paso de salida, página 434</i> ".

Nombres de paso

Al generar un paso, se le asigna un número recomendado. Este número tiene la estructura S_{i_j} , donde i es el número actual (interno) de la sección, y j es el número de paso actual (interno) de la sección actual.

Estos números se pueden modificar para mejorar la visión general. Los nombres de paso (máx. 32 caracteres) no deben repetirse en todo el proyecto, es decir, no puede haber ningún otro paso, variable, sección, etc. con el mismo nombre. No se distingue entre mayúsculas y minúsculas. El nombre del paso debe cumplir la nomenclatura general.

Tiempos de paso

A cada paso se le puede asignar un tiempo de vigilancia mínimo, otro máximo y un tiempo de retardo:

- **Tiempo de vigilancia mínimo**

El tiempo de vigilancia mínimo indica el tiempo de duración que, por lo general, debe estar activo un paso como mínimo. Si el paso se convierte en inactivo antes de que transcurra este tiempo, se emitirá un mensaje de error. En la modalidad de animación, además, el error se identificará con un cambio de color (amarillo) del objeto del paso.

Si no se indica ningún tiempo de vigilancia mínimo o se indica un tiempo de vigilancia mínimo de 0, no se vigilará el paso.

El estado de error se mantiene hasta que se activa el paso de nuevo.

- **Tiempo de vigilancia máximo**

El tiempo de vigilancia máximo indica el tiempo de duración que, por lo general, debe estar como máximo activo un paso. Si el paso permanece activo después de que transcurra este tiempo, se emitirá un mensaje de error. En la modalidad de animación, además, el error se identificará con un cambio de color (rosa) del objeto del paso.

Si no se indica ningún tiempo de vigilancia máximo o se indica un tiempo de vigilancia máximo de 0, no se vigilará el paso.

El estado de error se mantiene hasta que el paso se convierte en inactivo.

- **Tiempo de retardo**

El tiempo de retardo (tiempo de duración del paso) establece el tiempo mínimo durante el que debe estar activo un paso.

NOTA: Los tiempos indicados son válidos para el paso, pero no para las acciones asignadas a él. Para éstas se pueden definir tiempos propios.

Determinación de los tiempos de paso

Al definir/calcular los tiempos, tenga en cuenta la siguiente fórmula:

Tiempo de retardo < tiempo de vigilancia mínimo < tiempo de
vigilancia máximo

Hay dos posibilidades de asignar los valores definidos a un paso:

- Introducirlo como literal de duración
- Utilizar la estructura de datos `SFCSTEP_TIMES`

Variable SFCSTEP_TIMES

A los pasos se les puede asignar una variable del tipo de datos `SFCSTEP_TIMES`. Los elementos de esta estructura de datos se pueden leer y escribir (lectura/escritura).

La estructura de datos se gestiona del mismo modo que cualquier otra, es decir, que se puede emplear en declaraciones de variables y, por lo tanto, es posible acceder a la estructura de datos completa (p. ej. como parámetro FFB).

Elementos de la estructura de datos:

Nombre del elemento	Tipo de datos	Descripción
"VarName".delay	TIME	Tiempo de retardo
"VarName".min	TIME	Tiempo de vigilancia mínimo
"VarName".max	TIME	Tiempo de vigilancia máximo

Variable SFCSTEP_STATE

A cada paso se le asigna de forma implícita una variable del tipo de datos `SFCSTEP_STATE`. Esta variable de paso tiene el nombre del paso asignado. Los elementos de esta estructura de datos sólo se pueden leer (sólo lectura).

Puede ver las variables `SFCSTEP_STATE` en el **Editor de datos**. El comentario para una variable `SFCSTEP_STATE` es el comentario introducido como una propiedad del paso. Consulte el capítulo sobre la definición de las propiedades de los pasos (*véase Unity Pro, Modalidades de funcionamiento,) en el manual de modalidades de servicio de Unity Pro.*

La estructura de datos no se puede utilizar en declaraciones de variables. Por este motivo, no es posible acceder como un entero (por ejemplo, como parámetro FFB).

Elementos de la estructura de datos:

Nombre del elemento	Tipo de datos	Descripción
"StepName".t	TIME	Tiempo de duración actual en el paso. Si se desactiva el paso, el valor de este elemento se mantendrá hasta que se vuelva a activar el paso.
"StepName".x	BOOL	1: Paso activo 0: Paso inactivo

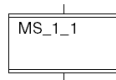
Nombre del elemento	Tipo de datos	Descripción
"StepName".tminErr	BOOL	Este elemento es una ampliación de la norma IEC 61131-3. 1: Transgresión por debajo del tiempo de vigilancia mínimo 0: Sin transgresión por debajo del tiempo de vigilancia mínimo El elemento se restablece automáticamente en los siguientes casos: ● Cuando el paso se vuelve a activar. ● Cuando se restablece el control de secuencia. ● Si el botón de comando Restablecer errores de tiempos está activado
"StepName".tmaxErr	BOOL	Este elemento es una ampliación de la norma IEC 61131-3. 1: Rebasamiento del tiempo de vigilancia máximo 0: Sin rebasamiento del tiempo de vigilancia máximo El elemento se restablece automáticamente en los siguientes casos: ● Cuando se abandona el paso. ● Cuando se restablece el control de secuencia. ● Si el botón de comando Restablecer errores de tiempos está activado

Pasos de macro y secciones de macro

Paso de macro

Los pasos de macro sirven para llamar a secciones de macro y así estructurar jerárquicamente los controles de secuencia.

Representación de un paso de macro:



Los pasos de macro tienen las siguientes propiedades:

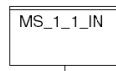
- Los pasos de macro se pueden ubicar en secciones de "control de secuencia" y en secciones de macro.
- No hay límite para el número de pasos de macro.
- La profundidad de intercalado, es decir, un paso de macro dentro de otro, es de 8 niveles.
- A cada paso de macro se le asigna de forma implícita una variable del tipo de datos `SFCSTEP_STATE`, consulte la "*Variable SFCSTEP_STATE, página 431*".
- A los pasos de macro se les puede asignar una variable del tipo de datos `SFCSTEP_TIMES`; consulte la "*Variable SFCSTEP_TIMES, página 431*".
- A los pasos de macro no se les puede asignar NINGUNA acción.
- Cada paso de macro se puede sustituir por la cadena secuencial contenida en la sección de macro asignada.

Los pasos de macro son una ampliación de la norma CEI 61131-3 y se deben habilitar de forma explícita.

Paso de entrada

Cada sección de macro comienza con un paso de entrada.

Representación de un paso de entrada:



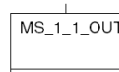
Los pasos de entrada tienen las siguientes propiedades:

- El editor SFC coloca automáticamente los pasos de entrada en las secciones de macro.
- En cada sección de macro se puede situar únicamente un paso de entrada.
- Un paso de entrada no se puede eliminar, copiar o insertar manualmente.
- Cada paso de entrada tiene asignada de forma implícita una variable del tipo de datos `SFCSTEP_STATE`; consulte la "*Variable SFCSTEP_STATE, página 431*".
- A los pasos de entrada se les puede asignar una variable del tipo de datos `SFCSTEP_TIMES`; consulte la "*Variable SFCSTEP_TIMES, página 431*".
- A los pasos de entrada se les puede asignar acciones.

Paso de salida

Cada sección de macro termina con un paso de salida.

Representación de un paso de salida:



Los pasos de salida tienen las siguientes propiedades:

- El editor SFC coloca automáticamente los pasos de salida en las secciones de macro.
- En cada sección de macro se puede situar únicamente un paso de salida.
- Un paso de salida no se puede eliminar, copiar o insertar manualmente.
- A los pasos de salida no se les puede asignar NINGUNA acción.
- A los pasos de salida sólo se les puede asignar un tiempo de retardo. No es posible asignar tiempos de vigilancia (consulte *Tiempos de paso, página 430*).

Sección de macro

Una sección de macro está compuesta por una única cadena secuencial que, en principio, dispone de los mismos elementos que una sección de "control de secuencia" (p. ej. pasos, paso o pasos iniciales, pasos de macro, transiciones, bifurcación, conjunciones, etc.).

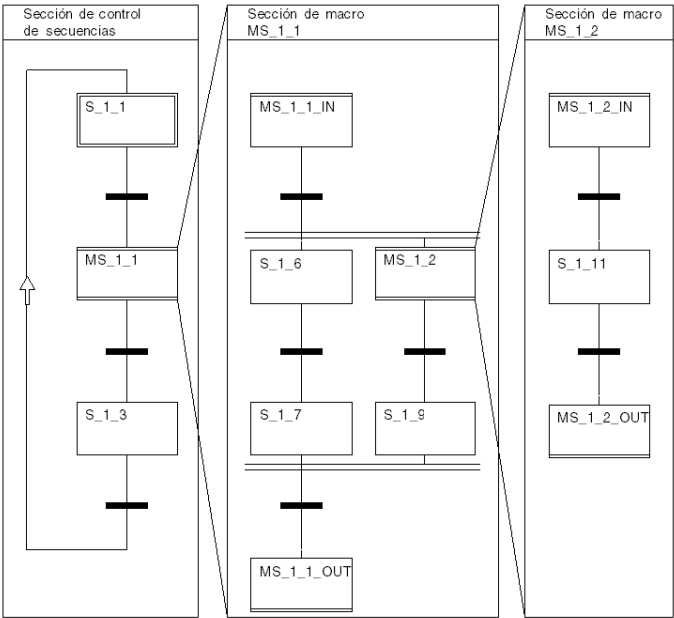
Además, cada sección de macro contiene en su comienzo un paso de entrada y, en el final, un paso de salida.

Cada paso de macro se puede sustituir por la cadena secuencial contenida en la sección de macro asignada.

Por esta razón, las secciones de macro pueden contener 0, 1 o más pasos iniciales (consulte también "*Tipos de pasos, página 429*").

- Single-Token
 - Se utilizarán
0 pasos iniciales en las secciones de macro cuando ya exista un paso inicial en la sección inmediatamente superior o inferior.
 - Se utilizará
1 paso inicial en las secciones de macro cuando no exista ningún paso inicial en la sección inmediatamente superior o inferior.
- Multi-Token
Por cada sección (incluidas todas sus secciones de macro) se puede utilizar un máximo de 100 pasos iniciales.

Uso de secciones de macro:



El nombre de estas secciones de macro es exactamente el mismo que el nombre del paso de macro invocante. Si se cambia el nombre del paso de macro, el nombre de la sección de macro asignada cambiará automáticamente.

Una sección de macro sólo se puede utilizar una vez.

Procesamiento de pasos de macro

Procesamiento de pasos de macro:

Fase	Descripción
1	Un paso de macro se activa cuando la condición de transición previa pasa a VERDADERA. Al mismo tiempo se activa el paso de entrada de la sección de macro.
2	Se procesa la cadena secuencial de la sección de macro. El paso de macro permanece activo mientras haya activo al menos un paso de la sección de macro.

Fase	Descripción
3	Si el paso de salida de la sección de macro se activa, se habilitará la siguiente transición que siga al paso de macro.
4	El paso de macro se desactivará cuando el paso de salida esté activo y, en consecuencia, esté habilitada la siguiente condición de transición y la condición de transición pase a VERDADERA. Al mismo tiempo, el paso de salida de la sección de macro se desactivará.

Nombres de paso

Al generar un paso, se le asigna un número recomendado.

Significado de los números recomendados:

Tipo de paso	Número recomendado	Descripción
Paso de macro	MS_i_j	MS = paso de macro i = número correlativo (interno) de la sección actual j = número de paso de macro correlativo (interno) en la sección actual
Paso de entrada	MS_k_l_IN	MS = paso de macro k = número correlativo (interno) de la sección invocante l = número de paso de macro correlativo (interno) en la sección invocante IN = paso de entrada
Paso de salida	MS_k_l_OUT	MS = paso de macro k = número correlativo (interno) de la sección invocante l = número de paso de macro correlativo (interno) en la sección invocante OUT = paso de salida
Paso "normal" (dentro de una sección de macro)	S_k_m	S = paso k = número correlativo (interno) de la sección invocante m = número de paso correlativo (interno) en la sección invocante

Estos números se pueden modificar para mejorar la visión general. Los nombres de paso (máximo 28 caracteres para los nombres de paso de macro, máximo 32 caracteres para los nombres de paso) no deben repetirse en todo el proyecto, es decir, no puede haber ningún otro paso, variable, sección (excepto el nombre de la sección de macro asignada al paso de macro), etc. con el mismo nombre. No se distingue entre mayúsculas y minúsculas. El nombre del paso debe cumplir las convenciones de nomenclatura estándar.

Si se cambia el nombre del paso de macro, el nombre de la sección de macro asignada y los pasos que contenga cambiarán automáticamente.

Por ejemplo, si se cambia el nombre de `MS_1_1` a `MyStep`, los nombres de pasos de la sección de macro cambiarán a `MyStep_IN`, `MyStep_1`, ..., `MyStep_n`, `MyStep_OUT`.

13.3 Acción y sección de acción

Vista general

En este apartado se describen las acciones y secciones de acción del lenguaje de ejecución secuencial SFC.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Acción	439
Sección de acción	441
Descriptor	442

Acción

Introducción

Las acciones tienen las siguientes propiedades:

- Una acción puede ser una variable booleana (variable de acción *(véase página 439)*) o una sección (sección de acción *(véase página 441)*) de los lenguajes de programación FBD, LD, IL o ST.
- Un paso se puede asociar a ninguna o a varias acciones. Un paso al que no se asigna ninguna acción tiene una función de espera, es decir, espera hasta que la transición asignada se haya completado.
- Si en un paso se han asignado varias acciones, éstas se procesarán en el orden en que aparezcan en el cuadro de lista de acciones.
Excepción: Independientemente de su posición en el cuadro de lista de acciones, las acciones con el descriptor *(véase página 442)* P1 siempre se procesarán las primeras y las acciones con el descriptor P0, las últimas.
- El control de acciones se expresa mediante descriptores *(véase página 442)*.
- A cada paso se le puede asignar un máximo de 20 acciones.
- Cualquier variable de acción asignada a una acción también se puede utilizar en acciones de otros pasos.
- La variable de acción también se puede utilizar para leer y escribir en cualquier otra sección del proyecto (asignación múltiple).
- Las acciones que tengan asignado un descriptor con duración sólo pueden estar activas una única vez.
- Como variables de acción sólo se admiten variables y direcciones booleanas o elementos booleanos de variables de elementos múltiples.
- Las acciones tienen nombres unívocos.
El nombre de cada acción es bien el nombre de la variable de acción o bien el nombre de la sección de acción.

Variable de acción

Como variable de acción se admiten:

- Direcciones del tipo de datos `BOOL`
Se puede asociar una acción a una salida de hardware mediante una dirección. En tal caso, la acción se utiliza como señal de habilitación de una transición, como señal de entrada en otra sección y como señal de salida para el hardware.
- Variable simple o elemento de una variable de elementos múltiples del tipo de datos `BOOL`
La acción se puede utilizar como señal de entrada en otra sección con ayuda de una variable.
 - Unlocated Variable
En el caso de las Unlocated Variables, la acción se puede utilizar como señal de habilitación de una transición y como señal de entrada en otra sección.
 - Located Variable

En este caso, la acción se puede utilizar como señal de habilitación de una transición, como señal de entrada en otra sección y como señal de salida para el hardware.

Nombres de acción

Si se utiliza como acción una dirección o una variable, para el nombre de la acción se utilizará su designación (p. ej. %Q10.4, Variable1).

Si se utiliza como acción una sección de acción, como nombre de acción se utilizará el nombre de dicha sección.

Los nombres de acción (máx. 32 caracteres) no deben repetirse en todo el proyecto, es decir, no puede haber ninguna otra transición, variable, sección etc. con el mismo nombre. No se distingue entre mayúsculas y minúsculas. El nombre de la acción debe cumplir la nomenclatura general.

Sección de acción

Introducción

Es posible crear una sección de acción para cada acción. Se trata de una sección que contiene la lógica de la acción y que se encuentra conectada automáticamente con la acción.

Nombre de la sección de acción

El nombre de la sección de acción es siempre idéntico al nombre de la acción asignada; consulte "*Nombres de acción, página 440*".

Lenguajes de programación

En las secciones de acción se puede utilizar los siguientes lenguajes de programación: FBD, LD, IL y ST.

Propiedades de las secciones de acción

Los secciones de acción tienen las siguientes propiedades:

- Las secciones de acción pueden contener un número cualquiera de salidas.
- Las llamadas de subrutina son posibles en las secciones de acción únicamente si está habilitada la modalidad Multi- Token.

Nota: Las subrutinas llamadas **no** se encuentran sometidas al controlador de la cadena de secuencia, es decir,

- el descriptor asignado a la sección de acción invocante no tiene repercusiones en la subrutina,
- la subrutina permanece inactiva incluso si se desactiva el paso invocante.
- En las secciones de acción no se pueden utilizar funciones, bloques de función ni procedimientos de diagnóstico.
- Las secciones de acción pueden contener un número cualquiera de redes.
- Las secciones de acción pertenecen a la sección SFC, en la que se definen y pueden ser asignadas dentro de la misma (incluyendo todas sus secciones de macro) a cualquier acción.
- Las acciones de acción que tengan asignado un descriptor con duración sólo pueden estar activas una única vez.
- Las secciones de acción pertenecen a la sección SFC en la que se han definido. Si se borra la sección SFC correspondiente, se eliminarán automáticamente todas las secciones de acción de dicha sección SFC.
- Las secciones de acción únicamente pueden ser llamadas por acciones.

Descriptor

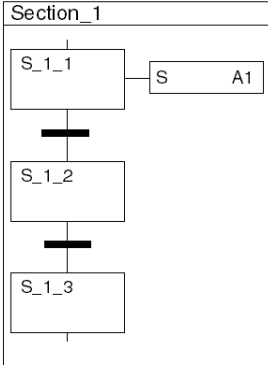
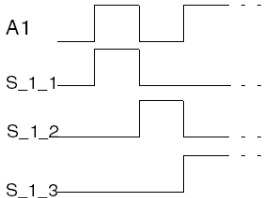
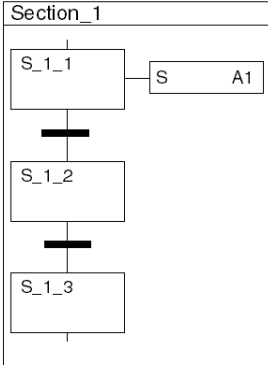
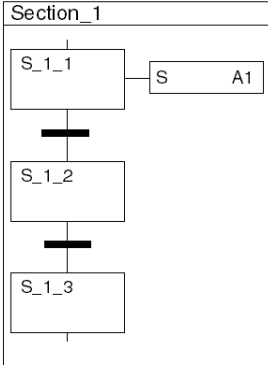
Introducción

En todas las uniones de una acción en un paso, debe establecerse un descriptor que defina el control de dicha acción.

Descriptores disponibles

Los siguientes descriptores están disponibles:

Descriptor	Significado	Descripción
N / Ninguno	Sin guardar	Si el paso se activa, la acción se establece en 1. Si el paso se desactiva, la acción se establece en 0.
R	Restablecer prioritaria	Se restablece la acción que se haya establecido con el descriptor S en otro paso. Además, puede impedirse el periodo de actividad de cualquier acción. Nota: Los descriptores se declaran sin almacenamiento de forma automática. Eso significa que su valor se restablece en 0 cuando el programa se detiene y se reinicia, p. ej.: por un corte en la alimentación. En caso de que necesite una salida con almacenamiento, utilice el bloque de funciones RS o SR de la biblioteca de módulos estándar.

Descriptor	Significado	Descripción				
S	Establecer (guardada)	La acción guardada queda activa cuando el paso correspondiente se desactiva. La acción se desactiva sólo cuando se restablezca con el descriptor R en otro paso de la cadena de ejecución actual, lo que significa que la acción no puede restablecerse desde otra sección; p. ej.: <table><tr><th>Section_1</th><th>Section_2</th></tr><tr><td></td><td><pre>IF S2.x= true THEN A1:= false; END_IF;</pre></td></tr></table>  Nota: Los descriptores se declaran sin almacenamiento de forma automática. Eso significa que su valor se restablece en 0 cuando el programa se detiene y se reinicia, p. ej.: por un corte en la alimentación. En caso de que necesite una salida con almacenamiento, utilice el bloque de funciones RS o SR de la biblioteca de módulos estándar. Nota: Se admite un máximo de 100 acciones con el descriptor S por sección SFC.	Section_1	Section_2		<pre>IF S2.x= true THEN A1:= false; END_IF;</pre>
Section_1	Section_2					
	<pre>IF S2.x= true THEN A1:= false; END_IF;</pre>					
I	Con límite temporal	Si el paso se activa, la acción también se activa. Una vez transcurrida la duración que haya definido para la acción, ésta se volverá a establecer en 0 aunque el paso siga activo. Si el paso se desactiva, la acción siempre se establece en 0. Nota: Para este descriptor, además es necesario definir una duración del tipo TIME.				
D	Con tiempo de retardo	Si el paso se activa, se inicia el temporizador interno y, una vez transcurrida la duración que se haya definido de forma manual para la acción, ésta se establece en 1. Siempre que el paso se desactive, la acción también se desactivará. Si el paso se desactiva antes de que haya transcurrido el tiempo interno, la acción no se activa. Nota: Para este descriptor, además es necesario definir una duración del tipo TIME.				

Descriptor	Significado	Descripción
P	Impulso	Si el paso se activa, la acción se establece en 1 y permanece así durante un ciclo de programa, independientemente de si el paso permanece activo o no.
DS	Guardada y con retardo	Si el paso se activa, se inicia el temporizador interno y, una vez transcurrida la duración que haya definido, la acción se activa. La acción se desactiva sólo cuando se restablezca con el descriptor R en otro paso. Si el paso se desactiva antes de que haya transcurrido el tiempo interno, la acción no se activa. Nota: Para este descriptor, además es necesario definir una duración del tipo <code>TIME</code> .
P1	Impulso (flanco ascendente)	Si el paso se activa (flanco 0->1), la acción se establece en 1 y permanece así durante un ciclo de programa, independientemente de si el paso permanece activo o no. Nota: Las acciones con el descriptor P1 siempre se procesan las primeras, independientemente de su posición en el cuadro de lista de acciones. Consulte también <i>Acción, página 439</i> .
P0	Impulso (flanco descendente)	Si el paso se desactiva (flanco 1->0), la acción se establece en 1 y permanece así durante un ciclo de programa. Nota: Las acciones con el descriptor P0 siempre se procesan las últimas, independientemente de su posición en el cuadro de lista de acciones. Consulte también <i>Acción, página 439</i> .

13.4 Transición y sección de transición

Vista general

En este apartado se describen los objetos de transición y las secciones de transición del lenguaje de ejecución secuencial SFC.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Transición	446
Sección de transición	448

Transición

Introducción

Una transición indica la condición gracias a la cual el control pasa de uno o varios pasos, que anteceden a la transición, a uno o varios pasos sucesores a lo largo de la correspondiente conexión.

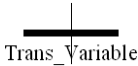
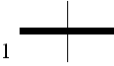
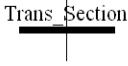
Condición de transición

Cada transición tiene asignada una condición de transición del tipo de datos `BOOL`.

Como condición de transición se admiten:

- Una dirección (entrada o salida)
- Una variable (entrada o salida)
- Un literal
- Una sección de transición (*véase página 448*)

El tipo de condición de transición determina la posición del nombre.

Condición de transición	Posición del nombre
• Dirección • Variable	
• Literal	
• Sección de transición	

Nombres de transición

Si se utiliza como condición de transición una dirección o una variable, como nombre de transición se indicará su designación (p. ej. `%I10.4`, `Variable1`).

Si se utiliza como condición de transición una sección de transición, como nombre de transición se indicará el nombre de sección.

Los nombres de transición (máx. 32 caracteres) no deben repetirse en todo el proyecto, es decir, no puede haber ninguna otra transición, variable, sección (excepto la sección de transición asignada) etc. con el mismo nombre. No se distingue entre mayúsculas y minúsculas. El nombre de la transición debe cumplir la nomenclatura general.

Habilitación de una transición

Se habilita una transición si sus pasos inmediatamente anteriores están activos. Las transiciones, cuyos pasos antecesores inmediatos no están activos, normalmente no se evaluarán.

NOTA: Si no se ha definido ninguna condición de transición, dicha transición no se activará nunca.

Disparo de una transición

El disparo de una transición se efectúa si se ha habilitado una transición y se cumple la condición de transición correspondiente.

El disparo de una transición provoca la desactivación (restablecimiento) de todos los pasos inmediatamente anteriores relacionados con la transición, seguida de la activación de todos los pasos inmediatamente posteriores.

Tiempo de disparo de una transición

En teoría, el tiempo de disparo (tiempo de conmutación) de una transición se puede considerar lo más corto posible, pero jamás puede ser cero. El tiempo de disparo de una transición tiene como mínimo la duración de un ciclo de programa.

Sección de transición

Introducción

Es posible crear una sección de transición para cada transición. Se trata de una sección que contiene la lógica de la condición de transición y que se encuentra conectada automáticamente con la transición.

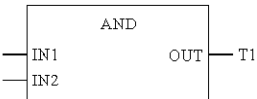
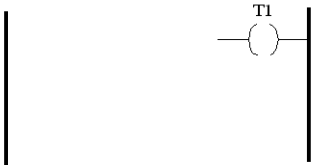
Nombre de la sección de transición

El nombre de la sección de transición es siempre idéntico al nombre de la transición asignada; consulte "*Nombres de transición, página 446*".

Lenguajes de programación

En las secciones de transición se pueden utilizar los siguientes lenguajes de programación: FBD, LD, IL y ST.

Redes recomendadas de las secciones de transición

Lenguaje	Red recomendada	Descripción
FBD		La red recomendada contiene un bloque AND con 2 entradas cuya salida booleana está conectada con una variable que tiene el nombre de la sección de transición. El bloque recomendado se puede conectar o eliminar.
LD		La red recomendada contiene una bobina conectada con una variable que tiene el nombre de la sección de transición. La bobina recomendada se puede conectar o eliminar.

Lenguaje	Red recomendada	Descripción
IL	-	La red recomendada está vacía. Como contenido, únicamente se puede generar una lógica booleana. La asignación del resultado de la lógica a la salida (la variable de transición) tiene lugar automáticamente, es decir, no se admite la instrucción de memoria ST. Ejemplo: <pre>LD A AND B</pre>
ST	-	La red recomendada está vacía. Como contenido, únicamente se puede generar una lógica booleana en forma de una expresión (imbricada). La asignación del resultado de la lógica a la salida (la variable de transición) tiene lugar automáticamente, es decir, no se admite la instrucción de asignación :=. La expresión no concluye con punto y coma (;). Ejemplo: <pre>A AND B O A AND (WORD_TO_BOOL (B))</pre>

Propiedades de las secciones de transición

Las secciones de transición presentan las siguientes propiedades:

- Las secciones de transición sólo tienen una salida (la variable de transición) y su tipo de datos es `BOOL`. El nombre de estas variables es exactamente el mismo que el nombre de la sección de transición.
- La variable de transición sólo se puede utilizar una vez con función de escritura.
- La variable de transición se puede leer desde cualquier punto del proyecto.
- Sólo se pueden utilizar funciones, pero no se podrán usar ni bloques de función ni procedimientos.
- En LD únicamente se puede utilizar una sola bobina.
- Sólo hay una red, es decir, todas las funciones utilizadas se unen unas con otras de manera directa o indirecta.
- Las secciones de transición sólo se pueden utilizar una vez.
- Las secciones de transición pertenecen a la sección SFC en la que se hayan definido. Si se borra la sección SFC correspondiente, se eliminarán automáticamente todas las secciones de transición de dicha sección SFC.
- Sólo se puede acceder a las secciones de transición desde otras transiciones.

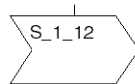
13.5 Salto

Salto

Generalidades

Los saltos se utilizan para representar conexiones direccionales que no se han dibujado en toda su longitud.

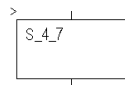
Representación de un salto



Propiedades de los saltos

Los saltos presentan las siguientes propiedades:

- Varios saltos pueden tener como objetivo el mismo paso.
- Según CEI 61131-3, los saltos para introducirse en una cadena simultánea (véase página 456) o para salir fuera de una cadena simultánea no son posibles. No obstante, si hubieran de utilizarse, deberán habilitarse de forma explícita.
- Dentro de los saltos, se distingue entre saltos de cadena (véase página 462) y bucles de cadena (véase página 462).
- El destino del salto se identifica mediante el símbolo de destino de salto (>).



Nombre de saltos

Los saltos no tienen nombres propios en sentido estricto. En su lugar, dentro del símbolo del salto aparece el nombre del paso de destino (objetivo del salto).

13.6 Conexión

Conexión

Introducción

Las conexiones unen pasos y transiciones, transiciones y pasos, etc.

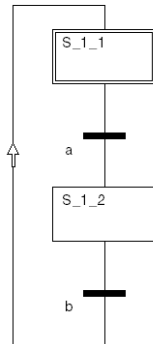
Propiedades de las conexiones

Las conexiones tienen las siguientes propiedades:

- Las conexiones entre objetos del mismo tipo (paso con paso, transición con transición, etc.) no son posibles.
- Es posible establecer conexiones entre:
 - Salidas de objetos no conectadas y
 - Entradas de pasos conectadas o no conectadas
(es decir, las entradas de pasos se pueden conectar de forma múltiple)
- No es posible solapar conexiones y otros objetos SFC (paso, transición, salto etc.).
- Es posible solapar conexiones y conexiones.
- El cruce de conexiones es posible, y se indica mediante una conexión "interrumpida".



- Las conexiones están compuestas por segmentos verticales y horizontales.
- El flujo general de la señal en una cadena secuencial va de arriba a abajo. Sin embargo, para poder configurar bucles, es posible que las conexiones con un paso vayan de abajo a arriba. Esto es válido para conexiones de transiciones, bifurcaciones en paralelo o conjunciones alternativas con un paso. En estos casos, el sentido de la conexión aparece representado por una flecha.



- Dentro de las conexiones, se distingue entre saltos de cadena (véase página 462) y bucles de cadena (véase página 462).

13.7 Bifurcaciones y conjunciones

Vista general

En este apartado se describen los objetos de bifurcación y conjunción del lenguaje de ejecución secuencial SFC.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Bifurcaciones alternativas y conjunciones alternativas	454
Bifurcación simultánea y conjunción simultánea	456

Bifurcaciones alternativas y conjunciones alternativas

Introducción

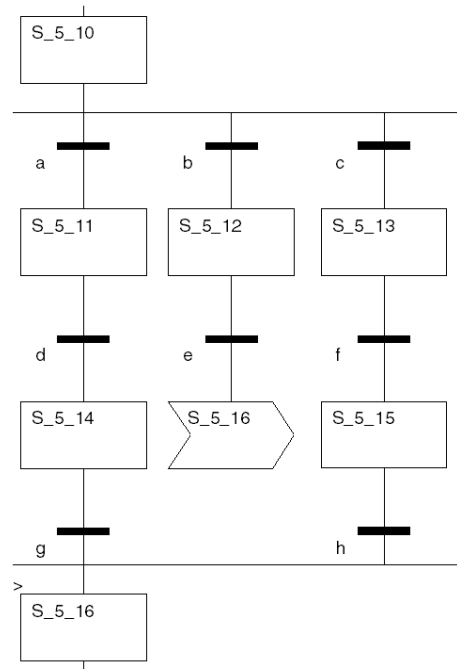
La bifurcación alternativa ofrece la posibilidad de programar bifurcaciones condicionales en el flujo de control de la estructura del SFC.

En las bifurcaciones alternativas, un paso va seguido de tantas transiciones por debajo de la línea horizontal como secuencias distintas existan.

Todas las derivaciones alternativas se unen mediante conjunciones alternativas o saltos (*véase página 450*) formando un solo nudo, en el que se procesarán después.

Ejemplo de una cadena alternativa

Ejemplo de una cadena alternativa



Propiedades de una cadena alternativa

Las propiedades de una cadena alternativa dependerán de si el control de secuencia se ejecuta en Single-Token o Multi-Token.

Consulte:

- Propiedades de una cadena alternativa en Single-Token (*véase página 461*)
- Propiedades de una cadena alternativa en Multi-Token (*véase página 474*)

Bifurcación simultánea y conjunción simultánea

Introducción

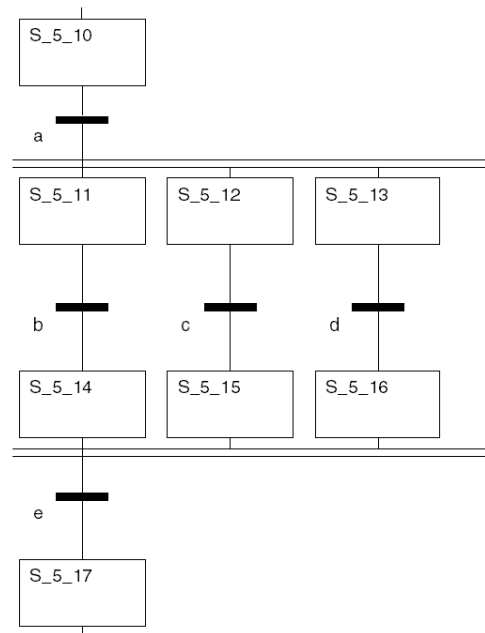
En las bifurcaciones simultáneas, al conectar una única transición, se activan de forma paralela varios (máx. 32) pasos (derivaciones). La ejecución tiene lugar de izquierda a derecha. Tras esta activación conjunta, se procesa cada una de las derivaciones de forma independiente.

Todas las bifurcaciones simultáneas se agrupan mediante una conjunción simultánea según CEI 61131-1. La transición se evalúa después de una conjunción simultánea si se han activado todos los pasos previos de la conjunción.

Si una bifurcación simultánea se agrupa mediante una conjunción alternativa, esto sólo es posible en la modalidad Multi-Token (*véase página 477*).

Ejemplo de una cadena simultánea

Ejemplo de una cadena simultánea



Propiedades de una cadena simultánea

Consulte

- Propiedades de una cadena simultánea en Single-Token (*véase página 461*)
- Propiedades de una cadena simultánea en Multi-Token (*véase página 474*)

13.8 Objeto de texto

Objeto de texto

Introducción

El texto puede ubicarse como objetos de texto mediante el lenguaje de ejecución secuencial SFC. El tamaño de estos objetos depende de la longitud del texto. Este objeto de texto es, al menos, del tamaño de una celda y puede agrandarse tanto vertical como horizontalmente cubriendo otras celdas, según el tamaño del texto. Los objetos de texto pueden solaparse con otros objetos SFC.

13.9 Single-Token

Vista general

En esta sección se describe la modalidad de servicio "Single-Token" para los controles de secuencia.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Secuencia de ejecución en Single-Token	460
Cadena alternativa	461
Salto de cadena y bucles de cadena	462
Cadenas simultáneas	465
Selección asimétrica de cadenas simultáneas	467

Secuencia de ejecución en Single-Token

Descripción

En Single-Token rigen las siguientes reglas:

- La situación original se define mediante el paso inicial. El control de secuencia contiene únicamente un paso inicial.
- En la cadena secuencial nunca habrá más de un paso activo. La única excepción es la constituida por las bifurcaciones simultáneas, en las que hay un paso activo por bifurcación.
- Las secuencias de los estados de señal activos tienen lugar a lo largo de las conexiones direccionales, activadas por medio de la conexión de una o varias transiciones. La dirección de la secuencia de una cadena sigue las conexiones direccionales y transcurre desde la parte interior del paso previo hasta la parte superior del paso siguiente.
- Se habilita una transición si los pasos inmediatamente anteriores están activos. Las transiciones cuyos pasos inmediatamente anteriores no están activos normalmente no se evalúan.
- Una transición se activa si se ha habilitado una transición y se cumple la condición de transición correspondiente.
- La activación de una transición provoca la desactivación (restablecimiento) de todos los pasos inmediatamente anteriores relacionados con la transición, seguida de la activación de todos los pasos inmediatamente posteriores.
- Si se cumplen varias condiciones de transición en una línea de pasos consecutivos, se conectará progresivamente un paso por ciclo.
- Los pasos no se pueden activar o desactivar desde otra sección que no sea SFC.
- Es posible utilizar pasos de macro.
- En las bifurcaciones alternativas nunca habrá más de una bifurcación activa. El resultado de las condiciones de transición de aquellas transiciones que sigan a la bifurcación alternativa determinará la bifurcación que se ejecute. Si se cumple una condición de transición, ya no se procesarán las transiciones restantes. Se activará la bifurcación de la transición realizada. La prioridad para las bifurcaciones va de izquierda a derecha. Todas las bifurcaciones alternativas se agruparán al final por medio de una conjunción alternativa o un salto.
- En las bifurcaciones simultáneas, al conectar una única transición, se activarán varios pasos (bifurcaciones). Tras esta activación conjunta, se procesa cada una de las bifurcaciones de forma independiente. Todas las bifurcaciones simultáneas se agruparán al final por medio de una conjunción simultánea global. Los saltos para introducirse en una bifurcación simultánea o para salir de ella no son posibles.

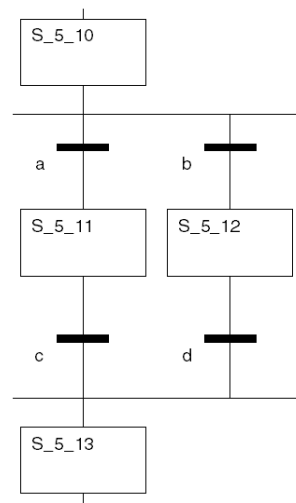
Cadena alternativa

Cadena alternativa

De acuerdo con CEI 61131-3, no es posible conectar más de una transición al mismo tiempo (seleccionar 1 de n). El resultado de las condiciones de transición de aquellas transiciones que sigan a la bifurcación alternativa determinará la ramificación que se ejecute. Las transiciones de la bifurcación se procesarán de izquierda a derecha. Si se cumple una condición de transición, ya no se procesarán las transiciones restantes. Se activará la bifurcación de la transición realizada. De esta forma resulta una prioridad de izquierda a derecha para las bifurcaciones.

Si no conmuta ninguna de las transiciones, permanecerá el paso que está ajustado en ese momento.

Cadena alternativa



Si...	Entonces...
S_5_10 está activo y la condición de transición a es verdadera (independientemente de b),	tendrá lugar una secuencia de S_5_10 a S_5_11.
S_5_10 está activo, y la condición de transición b es verdadera y a es falsa,	tendrá lugar una secuencia de S_5_10 a S_5_12.

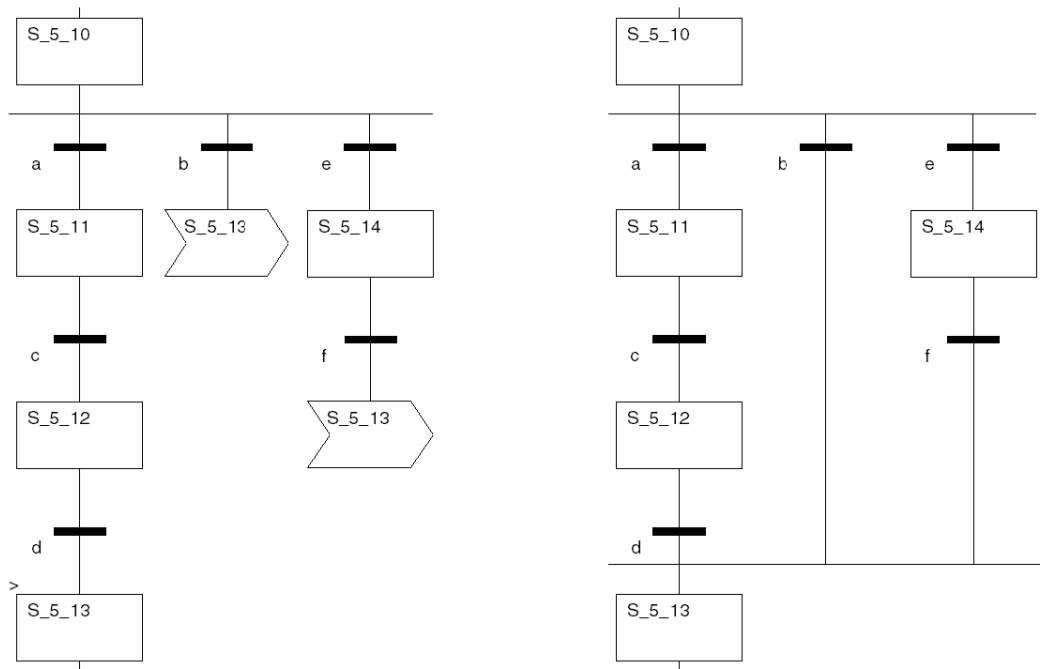
Saltos de cadena y bucles de cadena

Salto de cadena

Un salto de cadena es un caso especial de bifurcación alternativa, en el que se saltan ciertos pasos de la cadena.

Un salto de cadena puede realizarse con saltos o con conexiones.

Salto de cadena

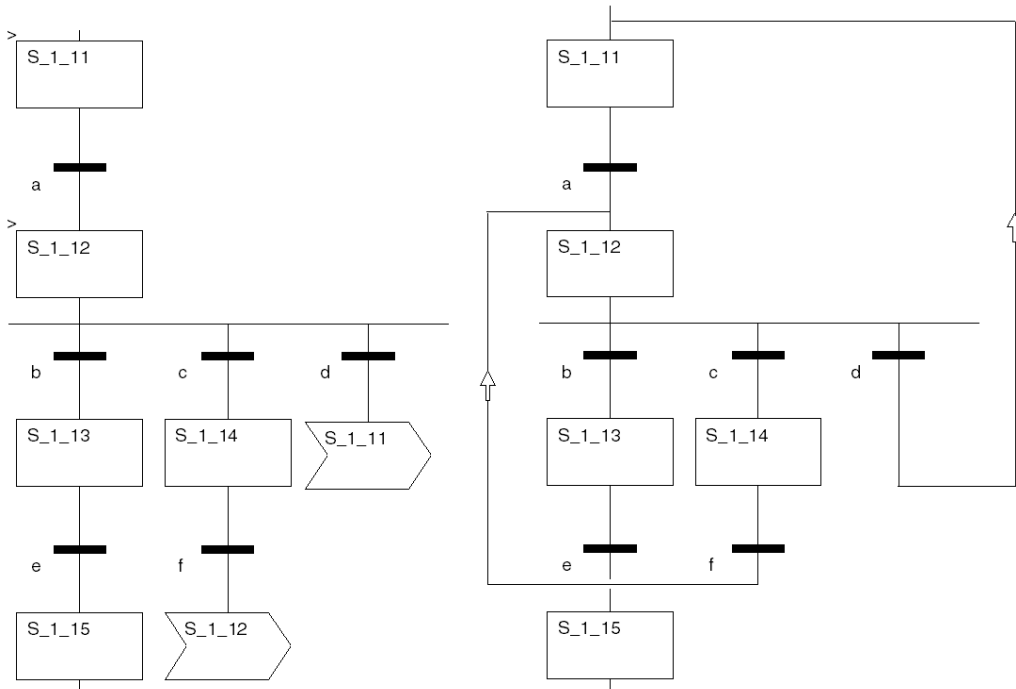


Si...	Entonces...
La condición de transición a es verdadera,	tendrá lugar una secuencia de s_5_10 a s_5_11, s_5_12 y s_5_13.
La condición de transición b es verdadera,	tendrá lugar un salto directo de s_5_10 a s_5_13.
La condición de transición e es verdadera,	tendrá lugar una secuencia de s_5_10 a s_5_14 y s_5_13.

Bucle de cadena

Un bucle de cadena es un caso especial de bifurcación alternativa, en la que una o varias derivaciones vuelven a un paso previo.

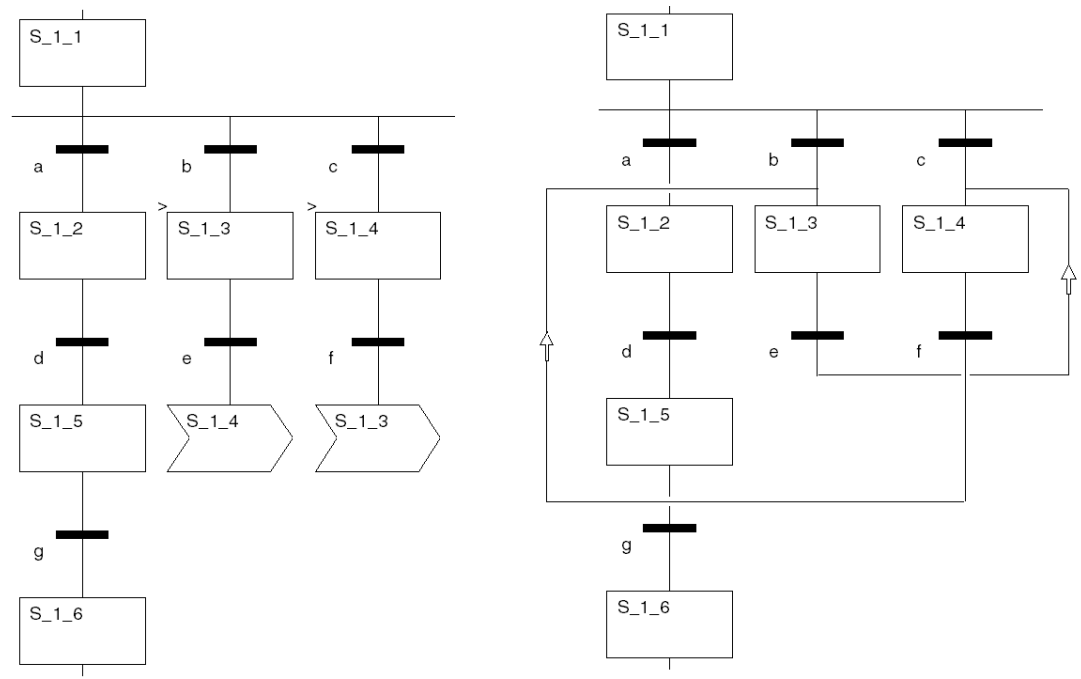
Un bucle de cadena puede realizarse con saltos o con conexiones.
Bucle de secuencia:



Si...	Entonces...
La condición de transición a es verdadera,	tendrá lugar una secuencia de S_1_11 a S_1_12 .
La condición de transición b es verdadera,	tendrá lugar una secuencia de S_1_12 a S_1_13 .
La condición de transición b es falsa y c es verdadera,	tendrá lugar una secuencia de S_1_12 a S_1_14 .
La condición de transición f es verdadera,	tendrá lugar un salto hacia atrás de S_1_14 a S_1_12 .
El bucle hacia atrás de S_1_12 a S_1_12 a través de las condiciones de transición c y f se repetirá hasta que la condición de transición b pase a verdadera, o c a falsa y d a verdadera.	
Las condiciones de transición b y c son falsas y d es verdadera,	tendrá lugar un salto directo hacia atrás de S_1_12 a S_1_11 .
El bucle de S_1_11 a S_1_12 y hacia atrás hasta S_1_11 a través de las condiciones de transición a y d se repetirá hasta que la condición de transición b o c sea verdadera.	

No está permitido que haya bucles de cadena sin fin dentro de una cadena alternativa.

Bucle de cadena sin fin



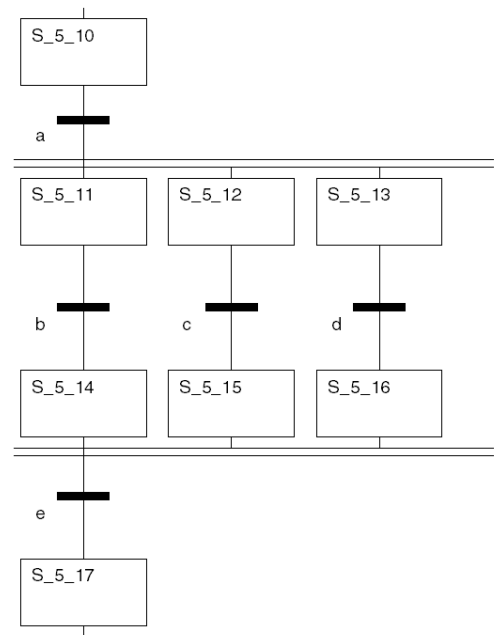
Si...	Entonces...
La condición de transición b es verdadera,	tendrá lugar una secuencia de S_1_1 a S_1_3 .
La condición de transición e es verdadera,	tendrá lugar un salto a S_1_4 .
La condición de transición f es verdadera,	tendrá lugar un salto a S_1_3 .
El bucle de S_1_3 a través de la condición de transición e , a S_1_4 a través de la condición de transición f y salto hacia atrás a S_1_3 se repetirá sin fin.	

Cadenas simultáneas

Cadenas simultáneas

En las bifurcaciones simultáneas, al conectar una única transición, se activan de forma paralela varios (máx. 32) pasos (bifurcaciones). Esto es válido tanto en Single-Token como en Multi-Token.

Procesamiento de cadenas simultáneas

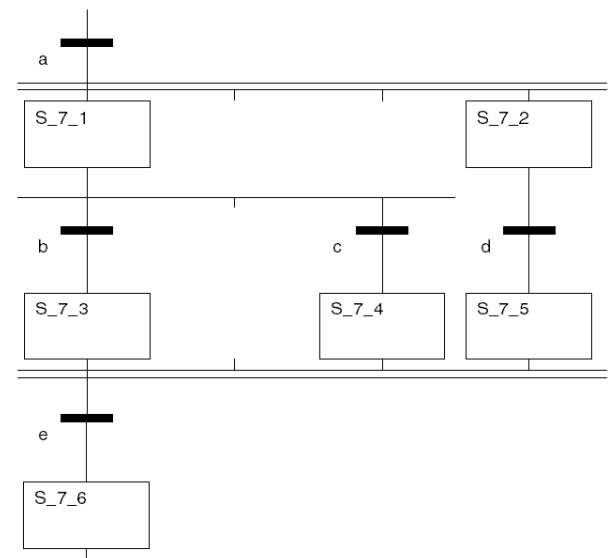


Si...	Entonces...
S_5_10 se encuentra activo y la condición de transición a, que pertenece a la transición conjunta, también es verdadera,	tendrá lugar una secuencia de S_5_10 a S_5_11, S_5_12 y S_5_13.
Los pasos S_5_11, S_5_12 y S_5_13 están activos,	las secuencias se procesarán de forma independiente.
S_5_14, S_5_15 y S_5_16 están activos al mismo tiempo y la condición de transición e, que pertenece a la transición conjunta, es verdadera,	tendrá lugar una secuencia de S_5_14, S_5_15 y S_5_16 a S_5_17.

Utilización de una bifurcación alternativa en una cadena simultánea

Si se utiliza una bifurcación alternativa en una cadena simultánea, esto provocará en Single-Token un bloqueo de la cadena.

Utilización de una bifurcación alternativa en una cadena simultánea



Si...	Entonces...
La condición de transición a es verdadera,	tendrá lugar una secuencia a S_7_1 y S_7_2.
Los pasos S_7_1 y S_7_2 están activados,	las secuencias se procesarán de forma independiente.
La condición de transición d es verdadera,	tendrá lugar una secuencia a S_7_5.
La condición de transición b es verdadera y c es falsa,	tendrá lugar una secuencia a S_7_3.
Si S_7_3, S_7_4 y S_7_5 están conectados mediante una conjunción simultánea, no puede tener lugar ninguna secuencia a S_7_6, ya que S_7_3 y S_7_4 nunca pueden estar activos al mismo tiempo. (O bien S_7_3 está activado mediante una condición de transición b o bien S_7_4 está activado a través de c, nunca los dos al mismo tiempo). Por esta razón, S_7_3, S_7_4 y S_7_5 nunca pueden estar activos al mismo tiempo. La cadena se bloqueará. El mismo problema surge cuando al entrar en la bifurcación alternativa, la condición de transición b es falsa y c es verdadera.	

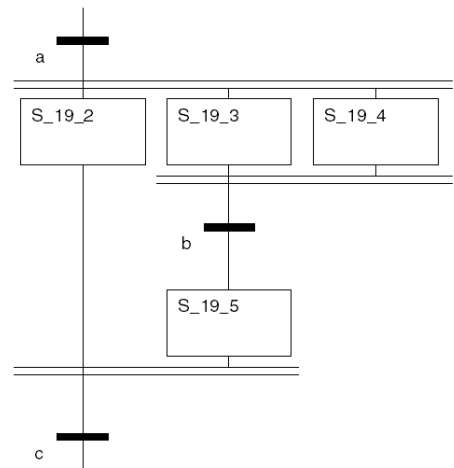
Selección asimétrica de cadenas simultáneas

Introducción

Según CEI 61131-3, una bifurcación simultánea siempre debe cerrarse con una conjunción simultánea. Sin embargo, el número de bifurcaciones simultáneas no tiene que coincidir necesariamente con el número de conjunciones simultáneas.

Cantidad mayor de ejecuciones conjuntas

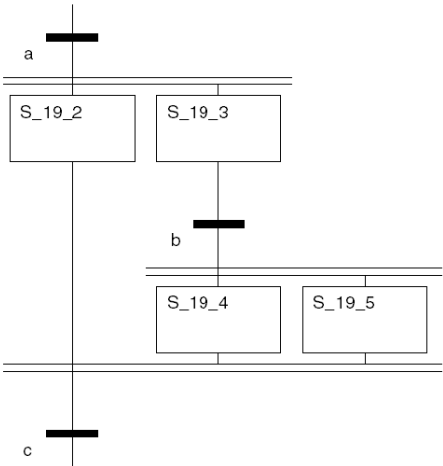
Cadena con 1 bifurcación simultánea y 2 conjunciones simultáneas



Si...	Entonces...
La condición de transición a es verdadera,	tendrá lugar una secuencia a S_19_2, S_19_3 y S_19_4.
Los pasos S_19_2, S_19_3 y S_19_4 están activados,	las secuencias se procesarán de forma independiente.
La condición de transición b es verdadera,	tendrá lugar una secuencia a S_19_5.
Los pasos S_19_2 y S_19_5 están activos y la condición de transición c es verdadera,	se abandonará la cadena simultánea.

Mayor cantidad de bifurcaciones

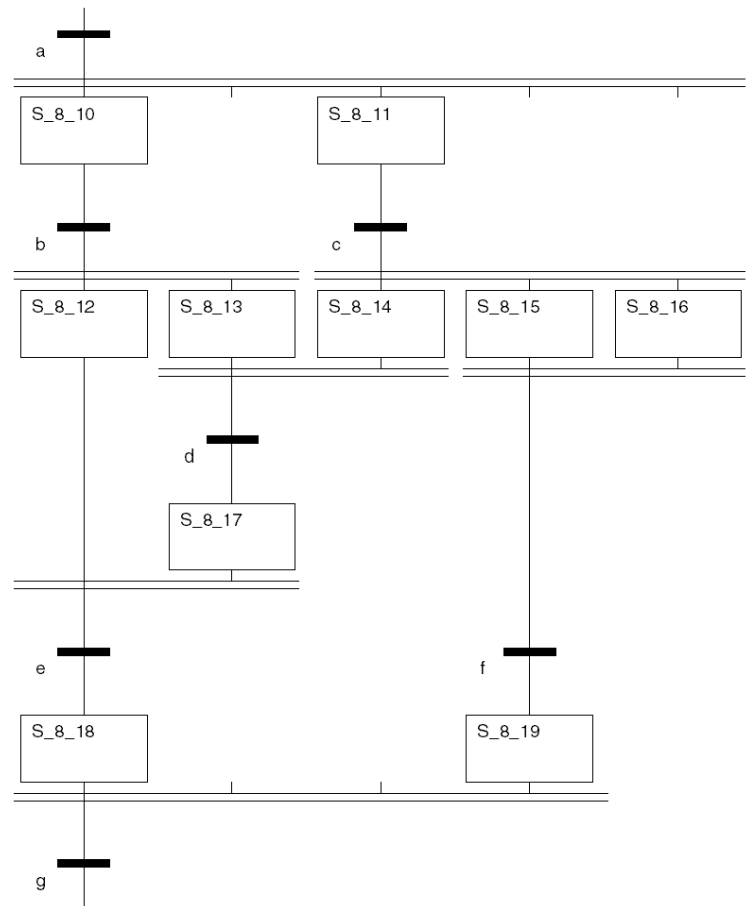
Cadena con 2 bifurcaciones simultáneas y 1 conjunción simultánea



Si...	Entonces...
La condición de transición a es verdadera,	tendrá lugar una secuencia a S_19_2, S_19_3.
Los pasos S_19_2 y S_19_3 están activados,	las secuencias se procesarán de forma independiente.
La condición de transición b es verdadera,	tendrá lugar una secuencia a S_19_4, S_19_5.
Los pasos S_19_4 y S_19_5 están activados,	las secuencias se procesarán de forma independiente.
Los pasos S_19_2, S_19_4 y S_19_5 están activos y la condición de transición c es verdadera,	se abandonará la cadena simultánea.

Cadenas simultáneas intercaladas

Cadenas simultáneas intercaladas



Si...	Entonces...
La condición de transición a es verdadera,	tendrá lugar una secuencia a S_8_10 y S_8_11.
La condición de transición b es verdadera,	tendrá lugar una secuencia a S_8_12 y S_8_13.
La condición de transición c es verdadera,	tendrá lugar una secuencia a S_8_14, S_8_15 y S_8_16.
Los pasos S_8_13 y S_8_14 están activos y la condición de transición d es verdadera,	tendrá lugar una secuencia a S_8_17.

Si...	Entonces...
Los pasos S_8_12 y S_8_17 están activos y la condición de transición e es verdadera,	tendrá lugar una secuencia a S_8_18.
...	...

13.10 Multi-Token

Vista general

En esta sección se describe la modalidad de servicio "Multi-Token" para los controles de secuencia.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Secuencia de ejecución en Multi-Token	472
Cadena alternativa	474
Cadenas simultáneas	477
Salto a una cadena simultánea	481
Salto desde una cadena simultánea	483

Secuencia de ejecución en Multi-Token

Descripción

En Multi-Token rigen las siguientes reglas:

- La situación de salida se define por medio de un cantidad definible de pasos iniciales (0 a 100).
- En la cadena secuencial puede haber activa cualquier cantidad de pasos al mismo tiempo.
- Las secuencias de los estados de señal activos tienen lugar a lo largo de las conexiones direccionales, disparadas por medio de la conexión de una o varias transiciones. La dirección de la secuencia de una cadena sigue las conexiones direccionales y transcurre desde la parte interior del paso previo hasta la parte superior del paso siguiente.
- Se habilita una transición si sus pasos inmediatamente anteriores están activos. Las transiciones, cuyos pasos inmediatamente anteriores no están activos, no se evaluarán.
- El disparo de una transición se efectúa si se ha habilitado una transición y se cumple la condición de transición correspondiente.
- El disparo de una transición provoca la desactivación (restablecimiento) de todos los pasos inmediatamente anteriores relacionados con la transición, seguida de la activación de todos los pasos inmediatamente posteriores.
- Si se cumplen varias condiciones de transición en una línea de pasos consecutivos, se conectará progresivamente un paso por ciclo.
- Los pasos y pasos de macro se pueden activar o desactivar desde otra sección que no sea SFC o por medio de operaciones de usuario.
- Si un paso activo se activa y desactiva al mismo tiempo, el paso permanecerá activo.
- Es posible utilizar pasos de macro. Asimismo, las secciones de paso de macro también pueden contener pasos iniciales.
- En las bifurcaciones alternativas puede haber varias derivaciones activas. El resultado de las condiciones de transición de aquellas transiciones que sigan a la bifurcación alternativa determinará las derivaciones que se ejecutarán. Las transiciones de la bifurcación se procesarán en paralelo. Se activarán las derivaciones que cumplan la transición. Las derivaciones alternativas no deben agruparse al final por medio de una conjunción alternativa o un salto.
- Si se configuran saltos para entrar en una derivación simultánea o para salir de una derivación simultánea, esto puede habilitarse por medio de una opción. En este caso, las derivaciones simultáneas no deben agruparse al final por medio de una conjunción simultánea.
- En la sección de acción se pueden utilizar llamadas a subrutinas.
- Es posible generar múltiples tokens mediante:
 - Pasos iniciales múltiples
 - Bifurcación simultánea o alternativa no finalizada

- Saltos en relación con cadenas alternativas y simultáneas
- Activación de pasos a través del bloque de control SFC `SETSTEP` desde una sección que no es SFC o a través de comandos de control SFC.
- Los tokens se pueden finalizar a través de:
 - Coincidencia de dos o más tokens en un paso.
 - Desactivación de pasos a través del bloque de control SFC `RESETSTEP` desde una sección que no es SFC o a través de comandos de control SFC.

Cadena alternativa

Cadena alternativa

En Multi-Token, el usuario puede predeterminar el comportamiento para la evaluación de las condiciones de transición en bifurcaciones alternativas.

Es posible indicar los siguientes valores predeterminados:

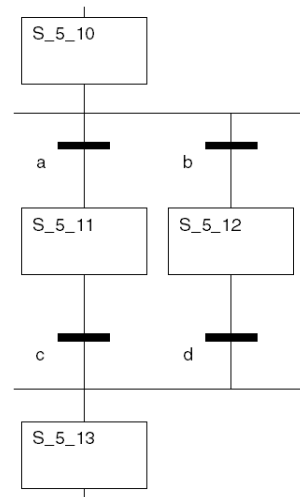
- Procesamiento de izquierda a derecha con parada después de la primera transición activa (seleccionar 1 de n). Esto corresponde al comportamiento de cadenas alternativas en Single-Token (*véase página 461*).
- Procesamiento paralelo de la bifurcación alternativa (seleccionar x de n)

Seleccionar x de n

En Multi-Token se pueden conectar varias transiciones en paralelo (seleccionar x de n). El resultado de las condiciones de transición de aquellas transiciones que sigan a la bifurcación alternativa determinará las derivaciones que se ejecutarán. Se procesarán todas las transiciones de la bifurcación. Se activarán todas las derivaciones que cumplan la transición.

Si no conmuta ninguna de las transiciones, permanecerá el paso que está ajustado en ese momento.

Seleccionar x de n

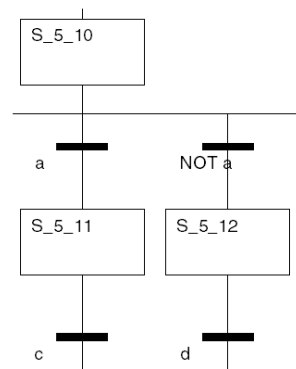


Si...	Entonces...
S_5_10 está activo, y la condición de transición a es verdadera y b es falsa,	tendrá lugar una secuencia de S_5_10 a S_5_11.

S_5_10 está activo, y la condición de transición a es falsa y b es verdadera,		tendrá lugar una secuencia de S_5_10 a S_5_12.	
S_5_10 está activo, y las condiciones de transición a y b son verdaderas,		tendrá lugar una secuencia de S_5_10 a S_5_11 y S_5_12.	
Al activar en paralelo las dos derivaciones alternativas, se generará un segundo token. Los dos tokens ahora se ejecutarán en paralelo, es decir S_5_11 y S_5_12 estarán activos al mismo tiempo.			
Token 1 (S_5_11)		Token 2 (S_5_12)	
Si...	Entonces...	Si...	Entonces...
La condición de transición c es verdadera,	tendrá lugar una secuencia de S_5_11 a S_5_13.	La condición de transición d es verdadera,	tendrá lugar una secuencia de S_5_12 a S_5_13.
Si S_5_13 todavía está activo (token 1) al activar la condición de transición c, entonces se finalizará el token 2 y la cadena se volverá a procesar como Single-Token. Si S_5_13 ya no está activo (token 1), entonces se volverá a activar a través del token 2 y ambos tokens continuarán ejecutándose en paralelo (Multi-Token).			

Si en esta modalidad de servicio se van a conectar bifurcaciones alternativas sólo exclusivamente, esto debe definirse de forma explícita mediante la lógica de transición.

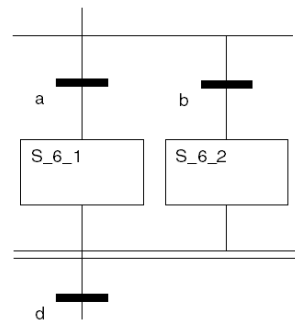
Ejemplo:



Salir de una bifurcación alternativa a través de una conjunción simultánea

Si se cierra una bifurcación alternativa a través de una conjunción simultánea, se puede llegar a bloquear la cadena.

Salir de una bifurcación alternativa a través de una conjunción simultánea



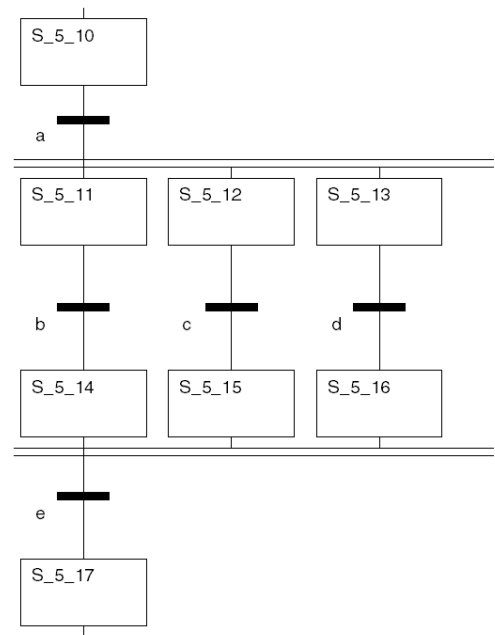
Si...	Entonces...
La condición de transición a es verdadera y b es falsa,	tendrá lugar una secuencia a S_6_1.
Como S_6_1 y S_6_2 se encuentran conectados mediante una conjunción simultánea, no se podrá abandonar la bifurcación, ya que S_6_1 y S_6_2 no pueden estar activos al mismo tiempo. (O bien S_6_1 está activado mediante una condición de transición a o bien S_6_2 está activado a través de b). Por esta razón, S_6_1 y S_6_2 pueden estar activos al mismo tiempo. La cadena se bloqueará. Para eliminar el bloque, se puede utilizar, por ejemplo, un segundo token posterior cuya secuencia se realice a través de la transición b.	

Cadenas simultáneas

Cadenas simultáneas

En las bifurcaciones simultáneas, al conectar una única transición, se activan de forma paralela varios (máx. 32) pasos (bifurcaciones). Esto es válido tanto en Single-Token como en Multi-Token.

Procesamiento de cadenas simultáneas

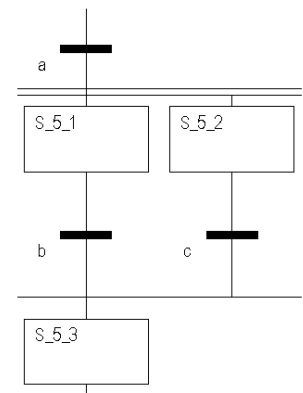


Si...	Entonces...
S_5_10 se encuentra activo y la condición de transición a, que pertenece a la transición conjunta, también es verdadera,	tendrá lugar una secuencia de S_5_10 a S_5_11, S_5_12 y S_5_13.
Los pasos S_5_11, S_5_12 y S_5_13 están activos,	las secuencias se procesarán de forma independiente.
S_5_14, S_5_15 y S_5_16 están activos al mismo tiempo y la condición de transición e, que pertenece a la transición conjunta, es verdadera,	tendrá lugar una secuencia de S_5_14, S_5_15 y S_5_16 a S_5_17.

Salir de una bifurcación simultánea mediante una conjunción alternativa

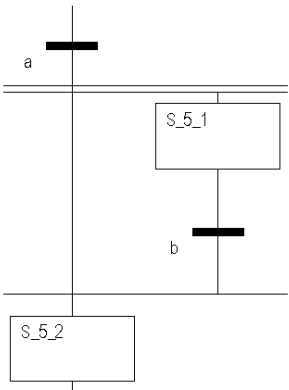
En Multi-Token, para salir de una bifurcación simultánea se puede utilizar una conjunción alternativa en lugar de una conjunción simultánea.

Salir de una cadena simultánea mediante una bifurcación alternativa (variante 1)



Si...		Entonces...	
La condición de transición a es verdadera,		tendrá lugar una secuencia a s_5_1 y s_5_2.	
Los pasos s_5_1 y s_5_2 están activados,		las secuencias se procesarán de forma independiente.	
La condición de transición b es verdadera y c es falsa,		tendrá lugar una secuencia a s_5_3.	
Con la secuencia a través de la conjunción alternativa para salir de la cadena simultánea, se generará un segundo token. Los dos tokens se ejecutarán en paralelo, es decir, ahora s_5_2 y s_5_3 estarán activos al mismo tiempo.			
Token 1 (S_5_3)		Token 2 (S_5_2)	
Si...	Entonces...	Si...	Entonces...
El paso s_5_3 está activo.		El paso s_5_2 está activo.	
		La condición de transición c es verdadera,	tendrá lugar una secuencia a s_5_3.
Si s_5_3 todavía está activo (token 1), se finalizará el token 2 y la cadena se volverá a procesar como Single-Token. Si s_5_3 ya no está activo (token 1), entonces se volverá a activar a través del token 2 y ambos tokens continuarán ejecutándose en paralelo (Multi-Token).			

Salir de una cadena simultánea mediante una bifurcación alternativa (variante 2)

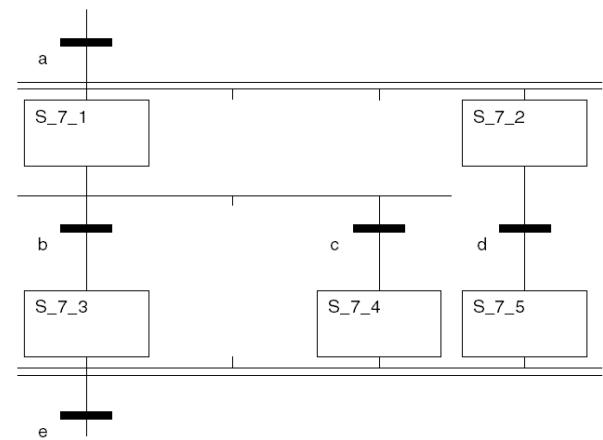


Si...		Entonces...	
La condición de transición a es verdadera,		tendrá lugar una secuencia a s_5_1 y s_5_2.	
Con la secuencia a través de la conjunción alternativa para salir de la cadena simultánea, se generará un segundo token. Los dos tokens ahora se ejecutarán en paralelo, es decir s_5_1 y s_5_2 estarán activos al mismo tiempo.			
Token 1 (S_5_2)		Token 2 (S_5_1)	
Si...	Entonces...	Si...	Entonces...
El paso s_5_2 está activo.		El paso s_5_1 está activo.	
		La condición de transición b es verdadera,	tendrá lugar una secuencia a s_5_2.
Si s_5_2 todavía está activo (token 1), se finalizará el token 2 y la cadena se volverá a procesar como Single-Token. Si s_5_2 ya no está activo (token 1), entonces se volverá a activar a través del token 2 y ambos tokens continuarán ejecutándose en paralelo (Multi-Token).			

Utilización de una bifurcación alternativa en una cadena simultánea

Si se utiliza una única bifurcación alternativa en una cadena simultánea, se puede llegar a bloquear la cadena.

Utilización de una bifurcación alternativa en una cadena simultánea



Si...	Entonces...
La condición de transición a es verdadera,	tendrá lugar una secuencia a S_7_1 y S_7_2.
Los pasos S_7_1 y S_7_2 están activados,	las secuencias se procesarán de forma independiente.
La condición de transición d es verdadera,	tendrá lugar una secuencia a S_7_5.
La condición de transición b es verdadera,	tendrá lugar una secuencia a S_7_3.
Como S_7_3, S_7_4 y S_7_5 se encuentran conectados mediante una conjunción simultánea, no se podrá abandonar la cadena simultánea, ya que S_7_3 y S_7_4 no pueden estar activos al mismo tiempo. (O bien S_7_3 está activado mediante una condición de transición b o bien S_7_4 está activado a través de c). Por esta razón, S_7_3, S_7_4 y S_7_5 no se activan al mismo tiempo. La cadena se bloqueará. Para desactivar el bloqueo se puede utilizar un segundo token posterior cuya secuencia se realiza a través de la transición c.	

Salto a una cadena simultánea

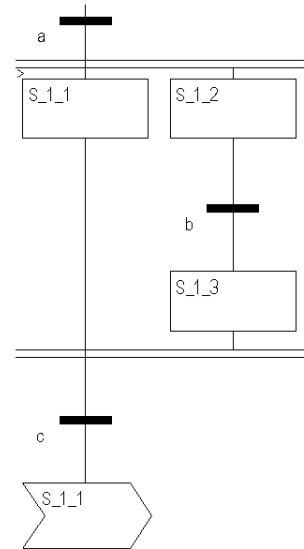
Descripción

En Multi-Token se puede habilitar la posibilidad de saltar a una cadena simultánea o saltar fuera de ella.

Un salto en una cadena simultánea no activa todas las derivaciones. Como la transición después de una conjunción simultánea no se evalúa hasta que se han establecido todos los pasos previos directos de la transición, la cadena simultánea ya no se podrá abandonar, la cadena se bloqueará.

Salto en una cadena simultánea

Salto en una cadena simultánea



Si...	Entonces...
La condición de transición a es verdadera,	tendrá lugar una secuencia a s_1_1 y s_1_2.
Los pasos s_1_1 y s_1_2 están activados,	las secuencias se procesarán de forma independiente.
s_1_2 está activo y la condición de transición b es verdadera,	tendrá lugar una secuencia de s_1_2 a s_1_3.
s_1_1 y s_1_3 están activos y es verdadera la condición de transición c, que pertenece a la transición conjunta,	tendrá lugar una secuencia de s_1_1 y s_1_3 para saltar a s_1_1.

Si...	Entonces...
S_1_1 se activa a través del salto,	sólo se activará la derivación de S_1_1. La derivación de S_1_2 no se activará.
Como S_1_1 y S_1_3 ahora no se pueden activar al mismo tiempo, la cadena no se podrá conectar progresivamente. La cadena se bloqueará. Para desactivar el bloqueo se puede utilizar un segundo token posterior que vuelva a activar el paso S_1_2.	

Salto desde una cadena simultánea

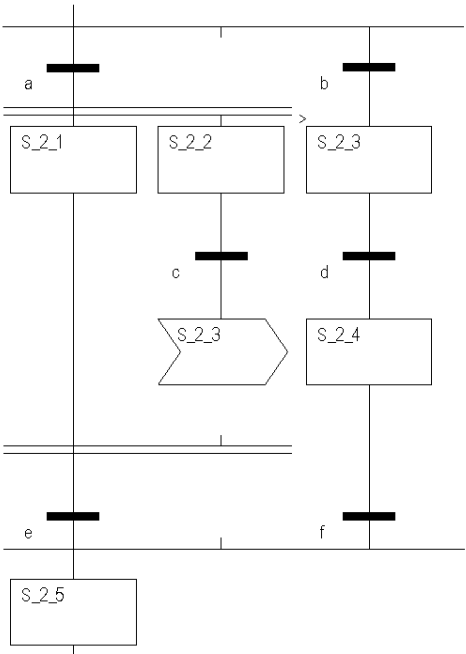
Introducción

En Multi-Token se puede habilitar la posibilidad de saltar a una cadena simultánea o saltar fuera de ella.

En todos los casos se generarán tokens adicionales.

Salto desde una cadena simultánea

Salto desde una cadena simultánea

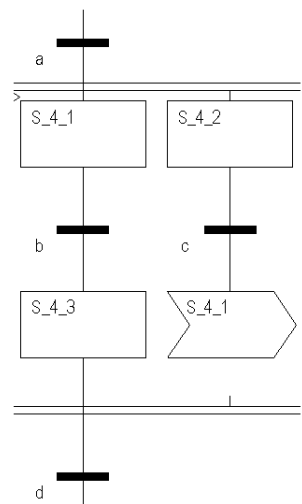


Si...		Entonces...	
La condición de transición a es verdadera y b es falsa,		tendrá lugar una secuencia a s_2_1 y s_2_2.	
Los pasos s_2_1 y s_2_2 están activados,		las secuencias se procesarán de forma independiente.	
La condición de transición c es verdadera,		tendrá lugar un salto a s_2_3.	
Con el salto desde la cadena simultánea se generará un segundo token. Los dos tokens se ejecutarán en paralelo, es decir, ahora s_2_1 y s_2_3 estarán activos al mismo tiempo.			
Token 1 (S_2_1)		Token 2 (S_2_3)	
Si...	Entonces...	Si...	Entonces...

La condición de transición e es verdadera,	tendrá lugar una secuencia a S_2_5.	La condición de transición d es verdadera,	tendrá lugar una secuencia a S_2_4.
		La condición de transición f es verdadera,	tendrá lugar una secuencia a S_2_5.
Si S_2_5 todavía está activo (token 1) al activar la condición de transición e, entonces se finalizará el token 2 y la cadena se volverá a procesar como Single-Token. Si S_2_5 ya no está activo (token 1), entonces se volverá a activar a través del token 2 y ambos tokens continuarán ejecutándose en paralelo (Multi-Token).			

Salto entre dos derivaciones de una cadena simultánea

Salto entre dos derivaciones de una cadena simultánea

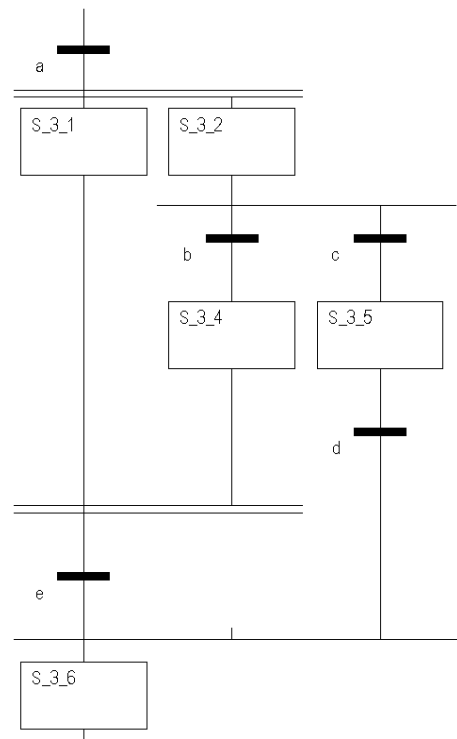


Si...		Entonces...	
La condición de transición a es verdadera,		tendrá lugar una secuencia a S_4_1 y S_4_2.	
Los pasos S_4_1 y S_4_2 están activados,		las secuencias se procesarán de forma independiente.	
La condición de transición b es verdadera,		tendrá lugar una secuencia a S_4_3.	
La condición de transición c es verdadera,		tendrá lugar un salto a S_4_1.	
Con el salto desde una derivación de cadena simultánea se generará un segundo token. Los dos tokens se ejecutarán en paralelo, es decir, ahora S_4_3 y S_4_1 estarán activos al mismo tiempo.			
Token 1 (S_4_3)		Token 2 (S_4_1)	
Si...	Entonces...	Si...	Entonces...
El paso S_4_3 se procesa		El paso S_4_1 se procesa	

	La condición de transición b es verdadera,	tendrá lugar una secuencia a S_4_3 .
Si en la activación mediante el token 2, el paso S_4_3 todavía está activo (token 1), el token 2 se finalizará y la cadena volverá a procesarse como Single-Token. Si en la activación mediante el token 2, el paso S_4_3 ya no está activo (token 1), se volverá a activar mediante el token 2 y los dos tokens seguirán ejecutándose en paralelo (Multi-Token). En ambos casos, si la condición de transición d es verdadera, se abandonará la cadena simultánea.		

Salir de una cadena simultánea mediante bifurcación alternativa

Salir de una cadena simultánea mediante una bifurcación alternativa



Si...	Entonces...
La condición de transición a es verdadera,	tendrá lugar una secuencia a S_3_1 y S_3_2 .
Los pasos S_3_1 y S_3_2 están activados,	las secuencias se procesarán de forma independiente.
La condición de transición b es falsa y c es verdadera,	tendrá lugar una secuencia a S_3_5 .
Con la secuencia a través de la bifurcación alternativa para salir de la cadena simultánea, se generará un segundo token. Los dos tokens se ejecutarán en paralelo, es decir, ahora S_3_1 y S_3_5 estarán activos al mismo tiempo.	

Token 1 (S_3_1)		Token 2 (S_3_5)	
Si...	Entonces...	Si...	Entonces...
Como s_3_4 no puede activarse, s_3_1 (token 1) permanecerá activo.		La condición de transición d es verdadera,	tendrá lugar una secuencia a s_3_6.
Si la condición de transición a es verdadera, tendrá lugar una secuencia a s_3_1 y s_3_2. De esta forma se finalizará el token 2 y la cadena volverá a procesarse como Single-Token.			
La condición de transición a es verdadera, tendrá lugar una secuencia a s_3_1 y s_3_2.			
		La condición de transición b es verdadera y c es falsa,	tendrá lugar una secuencia a s_3_4.
Como s_3_4 no puede activarse, s_3_1 (token 1) permanecerá activo hasta que se realice una secuencia a través de s_3_2 (token 2) y la transición b. Si s_4_4 ya no está activo (token 1), entonces se volverá a activar a través del token 2 y ambos tokens continuarán ejecutándose en paralelo (Multi-Token). (La ejecución conjunta de los dos tokens también puede tener lugar en s_4_3.)			

Lista de instrucciones (IL)

14

Vista general

En este capítulo, se describe el lenguaje de programación Lista de instrucciones (IL) conforme a CEI 61131.

Contenido de este capítulo

Este capítulo contiene las siguientes secciones:

Sección	Apartado	Página
14.1	Generalidades sobre la lista de instrucciones IL	488
14.2	Llamada de funciones elementales, módulos de función elementales, módulos de función derivados y procedimientos	511

14.1 Generalidades sobre la lista de instrucciones IL

Vista general

Esta sección ofrece una visión general sobre la lista de instrucciones IL.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Generalidades sobre la lista de instrucciones IL	489
Operandos	492
Modificador	494
Operadores	496
Llamada de subrutina	507
Etiquetas y saltos	508
Comentario	510

Generalidades sobre la lista de instrucciones IL

Introducción

Con ayuda del lenguaje de programación Lista de instrucciones (IL) se puede, por ejemplo, llamar bloques de funciones y funciones de forma condicional o incondicional, efectuar asignaciones y realizar saltos dentro de la sección de forma condicional o incondicional.

Instrucciones

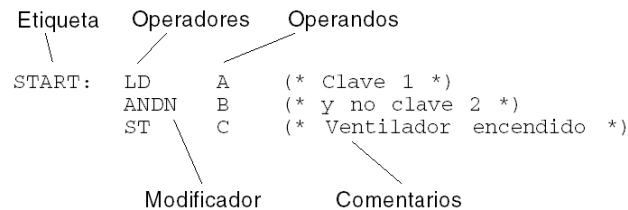
Una lista de instrucciones está compuesta por una secuencia de instrucciones.

Cada instrucción comienza en una fila nueva y está compuesta por:

- Un operador (*véase página 496*)
- En ocasiones, un modificador (*véase página 494*) y
- si fuese necesario, uno o varios operandos (*véase página 492*)

En caso de que se utilicen varios operandos, éstos irán separados por comas. Es posible que la instrucción esté precedida de una etiqueta (*véase página 508*). Dicha etiqueta irá seguida de dos puntos (:). La instrucción puede ir acompañada de un comentario (*véase página 510*).

Ejemplo:



Estructura del lenguaje de programación

IL es un lenguaje basado en un acumulador; es decir, cada instrucción utiliza o modifica el contenido actual del acumulador (un tipo de memoria intermedia interna). La norma CEI 61131 denomina a este acumulador "Resultado".

Por este motivo, una lista de instrucciones debe comenzar siempre por el operando LD ("Comando Cargar en el acumulador").

Ejemplo de una suma

Comando	Significado
LD 10	El valor 10 se carga en el acumulador.

Comando	Significado
ADD 25	Se suma 25 al contenido del acumulador.
ST A	El resultado se guarda en la variable A. El contenido de la variable A y el acumulador es ahora 35. Cualquier instrucción adicional funcionará con el contenido "35" del acumulador si no empieza por LD.

Las operaciones de comparación se refieren siempre al acumulador. El resultado booleano de la comparación se ubica en el acumulador y por ello es el contenido actual del acumulador.

Ejemplo de una comparación

Comando	Significado
LD B	El valor B se carga en el acumulador.
GT 10	El contenido del acumulador se compara con 10.
ST A	El resultado de la comparación se guarda en la variable A. Si B es menor o igual a 10, el valor de la variable A y el contenido del acumulador será 0 (FALSE). Si B es mayor que 10, el valor de la variable A y el contenido del acumulador será 1 (TRUE).

Tamaño de la sección

La longitud de una línea de instrucciones está limitada a 300 caracteres.

La longitud de una sección IL no está limitada dentro del entorno de programación.

La longitud de una sección IL sólo está limitada por el tamaño de la memoria del PLC.

Sintaxis

Los identificadores y palabras clave no distinguen entre mayúsculas y minúsculas.

Los espacios en blanco y los tabuladores no ejercen ninguna influencia sobre la sintaxis y se pueden utilizar libremente.

Excepción: no se admiten espacios en blanco ni tabuladores en:

- Palabras clave
- Literales
- Valores
- Identificadores
- Variables
- Combinaciones de limitadores [por ejemplo, (* para comentarios)]

Secuencia de ejecución

La ejecución de las instrucciones se realiza fila a fila de arriba a abajo. Esta secuencia se puede modificar mediante paréntesis.

Si, por ejemplo, A, B, C y D tienen los valores 1, 2, 3 y 4, y se calculan de la siguiente forma:

```
LD A
ADD B
SUB C
MUL C
ST E
```

el resultado en E será 0.

Si se realiza la operación:

```
LD A
ADD B
SUB (
LD C
MUL D
)
ST E
```

el resultado en E será 9.

Comportamiento ante errores

Las siguientes condiciones se consideran errores durante la ejecución de una expresión:

- Intento de división entre 0.
- Los operandos no poseen el tipo de datos adecuado para la operación.
- El resultado de una operación numérica sobrepasa el rango de valores del tipo de datos.

Conformidad CEI

Para obtener una descripción de conformidad CEI del lenguaje de programación IL, consulte Conformidad CEI (*véase página 693*).

Operandos

Introducción

Los operadores se aplican a los operandos.

Un operando puede ser:

- Una dirección
- Un literal
- Una variable
- Una variable de elementos múltiples
- Un elemento de una variable de elementos múltiples
- Una salida de EFB/DFB
- Una llamada de EFB/DFB.

Tipos de datos

El operando y el contenido del acumulador actual deben ser del mismo tipo de datos. Si se procesan operandos de tipos de datos distintos, se debe realizar antes una conversión de tipos.

En el ejemplo siguiente, la variable entera `i1` se convierte en una variable real antes de sumarse con la variable real `r4`.

```
LD i1
INT_TO_REAL
ADD r4
ST r3
```

Como excepción a esta regla, las variables del tipo de datos `TIME` se pueden multiplicar o dividir por las variables de los tipos de datos `INT`, `DINT`, `UINT` o `UDINT`.

Operaciones permitidas:

- LD timeVar1
DIV dintVar1
ST timeVar2
- LD timeVar1
MUL intVar1
ST timeVar2
- LD timeVar1
MUL 10
ST timeVar2

La norma CEI 61131-3 clasifica esta función como efecto "no deseado".

Utilización directa de direcciones

Las direcciones se pueden utilizar directamente (sin una declaración previa). En este caso, el tipo de datos de la dirección se asigna directamente. Esta asignación se realiza mediante el "prefijo de magnitud".

En la tabla siguiente, se indican los diversos prefijos de magnitud.

Prefijo de magnitud/símbolo	Ejemplo	Tipo de datos
Sin prefijo	%I10, %CH203.MOD, %CH203.MOD.ERR	BOOL
X	%MX20	BOOL
B	%QB102.3	BYTE
W	%KW43	INT
D	%QD100	DINT
F	%MF100	REAL

Utilización de otros tipos de datos

Si desea asignar a una dirección otros tipos de datos distintos a los tipos de datos predeterminados, deberá hacerlo mediante una declaración explícita. Esta declaración de variables se puede realizar de forma cómoda a través del editor de variables. Los tipos de datos de una dirección no se pueden declarar directamente en una sección ST (p. ej. declaración AT %MW1: UINT; no permitida).

Las siguientes variables están declaradas en el editor de variables:

```
UnlocV1: ARRAY [1..10] OF INT;  
LocV1:   ARRAY [1..10] OF INT AT %MW100;  
LocV2:   TIME AT %MW100;
```

Las siguientes llamadas son sintácticamente correctas:

```
%MW200 := 5;  
LD LocV1[%MW200]  
ST UnlocV1[2]  
  
LD t#3s  
ST LocV2
```

Acceso a variables de campo

Para el acceso a variables de campo (ARRAY), en la indicación de índice sólo se admiten literales y variables de tipo INT, DINT, UINT y UDINT.

El índice de un elemento ARRAY puede ser negativo si el límite inferior del rango es negativo.

Ejemplo: Guardado de una variable de campo

```
LD var1[i]  
ST var2.otto[4]
```

Modificador

Introducción

Los modificadores influyen en la ejecución del operador (consulte "*Operadores, página 496*").

Tabla de modificadores

Tabla de modificadores

Modificador	Se utiliza con operandos de tipo de datos	Descripción
N	BOOL, BYTE, WORD, DWORD	El modificador N se utiliza para invertir el valor de un operando por cada bit. Ejemplo: En el ejemplo, C es 1, si A es 1 y B es 0. <pre>LD A ANDN B ST C</pre>
C	BOOL	El modificador C se utiliza para ejecutar la instrucción correspondiente cuando el valor del acumulador es 1 (TRUE). Ejemplo: En el ejemplo, el salto a START sólo se realiza cuando A es 1 (TRUE) y B es 1 (TRUE). <pre>LD A AND B JMPC START</pre>

Modificador	Se utiliza con operandos de tipo de datos	Descripción
CN	BOOL	Si los modificadores C y N se combinan, la instrucción correspondiente se ejecutará únicamente cuando el valor del acumulador sea un 0 booleano (FALSE). Ejemplo: En el ejemplo, el salto a START sólo se realiza cuando A es 0 (FALSE) y B es 0 (FALSE). <pre>LD A AND B JMPCN START</pre>
(	Todos	El modificador paréntesis izquierdo (se utiliza para restablecer la evaluación del operando hasta que aparezca el operador paréntesis derecho). La cantidad de operaciones con paréntesis derecho debe ser igual a la cantidad de modificadores de paréntesis izquierdo. Los paréntesis pueden estar intercalados. Ejemplo: En el ejemplo, E es 1 si C o D son 1, y A y B son 1. <pre>LD A AND B AND (C OR D) ST E</pre> El ejemplo también puede programarse de la siguiente manera: <pre>LD A AND B AND (LD C OR D) ST E</pre>

Operadores

Introducción

Un operador es un símbolo para:

- Una operación aritmética que se va a ejecutar
- Una operación lógica ejecutable
- La llamada a un bloque de función elemental, un DFB o una subrutina

Los operadores son genéricos, es decir, se adaptan automáticamente al tipo de datos de los operandos.

Operadores de carga y de memoria

Operadores de carga y de memoria del lenguaje de programación IL:

Operador	Modificador	Significado	Operandos	Descripción
LD	N (sólo con operandos de los tipos de datos BOOL, BYTE, WORD o DWORD)	Carga el valor del operando en el acumulador.	Literal, variable, dirección directa con cualquier tipo de datos	Con LD se carga el valor de un operando en el acumulador. El ancho de datos del acumulador se adecua automáticamente al tipo de datos del operando. Esto también se aplica a los tipos de datos derivados. Ejemplo: En este ejemplo, el valor de A se carga en el acumulador, se suma con B y el resultado se guarda en E . LD A ADD B ST E
ST	N (sólo con operandos de los tipos de datos BOOL, BYTE, WORD o DWORD)	Guarda el valor del acumulador en el operando.	Variable, dirección directa con cualquier tipo de datos	Con ST se guarda el valor actual del acumulador en el operando. El tipo de datos del operando deberá coincidir con el "tipo de datos" del acumulador. Ejemplo: En este ejemplo, el valor de A se carga en el acumulador, se suma con B y el resultado se guarda en E . LD A ADD B ST E Dependiendo de si a ST le sigue o no un operador LD , se seguirá calculando con el resultado "antiguo". Ejemplo: En este ejemplo, el valor de A se carga en el acumulador, se suma con B y el resultado se guarda en E . A continuación, al valor de E (contenido actual del acumulador) se le resta el valor de B y el resultado se guarda en C . LD A ADD B ST E SUB B ST C

Operadores de establecimiento y restablecimiento

Operadores de establecimiento y restablecimiento del lenguaje de programación IL:

Operador	Modificador	Significado	Operandos	Descripción
S	-	El operando se ajusta a 1 cuando el contenido del acumuladores 1.	Variable, dirección directa del tipo de datos <code>BOOL</code>	Con <code>S</code>, el operando se ajusta a 1 si el contenido actual del acumulador es un 1 booleano. Ejemplo: En este ejemplo, el valor de <code>A</code> se carga en el acumulador. Si el contenido del acumulador (valor de <code>A</code>) es 1, entonces <code>OUT</code> se establece en 1. <pre>LD A S OUT</pre> En la mayoría de los casos, este operador se utiliza junto con el operador de restablecimiento <code>R</code>. Ejemplo: En este ejemplo se muestra un biestable RS (restablecer dominante), que se controla por medio de las dos variables booleanas <code>A</code> y <code>C</code>. <pre>LD A S OUT LD C R OUT</pre>
R	-	El operando se ajusta a 0 cuando el contenido del acumuladores 1.	Variable, dirección directa del tipo de datos <code>BOOL</code>	Con <code>R</code>, el operando se ajusta a 0 si el contenido actual del acumulador es un 1 booleano. Ejemplo: En este ejemplo, el valor de <code>A</code> se carga en el acumulador. Si el contenido del acumulador (valor de <code>A</code>) es 1, entonces <code>OUT</code> se establece en 0. <pre>LD A R OUT</pre> En la mayoría de los casos, este operador se utiliza junto con el operador de establecimiento <code>S</code>. Ejemplo: En este ejemplo se muestra un biestable SR (establecer dominante), que se controla por medio de las dos variables booleanas <code>A</code> y <code>C</code>. <pre>LD A R OUT LD C S OUT</pre>

Operadores lógicos

Operadores lógicos del lenguaje de programación IL:

Operador	Modificador	Significado	Operandos	Descripción
AND	N, N (, (	AND lógico	Literal, variable, dirección directa de los tipos de datos BOOL, BYTE, WORD o DWORD	Con AND tiene lugar una operación lógica "Y" entre el contenido del acumulador y el operando. Con los tipos de datos BYTE, WORD y DWORD, esta operación se lleva a cabo por bits. Ejemplo: En el ejemplo, D será 1 cuando A, B y C sean 1. <pre>LD A AND B AND C ST D</pre>
OR	N, N (, (	OR lógico	Literal, variable, dirección directa de los tipos de datos BOOL, BYTE, WORD o DWORD	Con OR tiene lugar una operación lógica "O" entre el contenido del acumulador y el operando. Con los tipos de datos BYTE, WORD y DWORD, esta operación se lleva a cabo por bits. Ejemplo: En el ejemplo, D será 1 cuando A o B sean 1 y C sea 1. <pre>LD A OR B OR C ST D</pre>

Operador	Modificador	Significado	Operandos	Descripción
XOR	N, N (, (	OR exclusivo lógico	Literal, variable, dirección directa de los tipos de datos BOOL, BYTE, WORD o DWORD	Con XOR tiene lugar una operación de tipo "OR exclusivo" entre el acumulador y el operando. Si se vinculan más de dos operandos, el resultado será 1 en caso de que haya una cantidad impar de estados 1, y 0 si hay una cantidad par de estados 1. Con los tipos de datos BYTE, WORD y DWORD, esta operación se lleva a cabo por bits. Ejemplo: En el ejemplo, D es 1 si A o B es 1. Si A y B tienen el mismo estado (ambos 0 o 1), entonces D es 0. LD A XOR B ST D Si se vinculan más de dos operandos, el resultado será 1 en caso de que haya una cantidad impar de estados 1, y 0 si hay una cantidad par de estados 1. Ejemplo: En el ejemplo, F es 1 si los operandos 1 o 3 son 1. F es 0 si los operandos 0, 2 o 4 son 1. LD A XOR B XOR C XOR D XOR E ST F
NOT	-	Negación lógica (complemento)	Contenido del acumulador de los tipos de datos BOOL , BYTE , WORD o DWORD	Con NOT se invierte el contenido del acumulador por bits. Ejemplo: En el ejemplo, B será 1 cuando A sea 0 y B será 0 cuando A sea 1. LD A NOT ST B

Operadores aritméticos

Operadores aritméticos del lenguaje de programación IL:

Operador	Modificador	Significado	Operandos	Descripción
ADD	(	Adición	Literal, variable, dirección directa de los tipos de datos INT, DINT, UINT, UDINT, REAL o TIME	Con ADD se suma el valor del operando al valor del contenido del acumulador. Ejemplo: El ejemplo corresponde a la fórmula $D = A + B + C$. LD A ADD B ADD C ST D
SUB	(	Sustracción	Literal, variable, dirección directa de los tipos de datos INT, DINT, UINT, UDINT, REAL o TIME	Con SUB se resta el valor del operando al contenido del acumulador. Ejemplo: El ejemplo corresponde a la fórmula $D = A - B - C$. LD A SUB B SUB C ST D
MUL	(	Multiplicación	Literal, variable, dirección directa de los tipos de datos INT, DINT, UINT, UDINT o REAL	Con MUL se multiplica el contenido del acumulador por el valor del operando. Ejemplo: El ejemplo corresponde a la fórmula $D = A * B * C$. LD A MUL B MUL C ST D Nota: Para las multiplicaciones con el tipo de datos Time está disponible la función MULTIME de la biblioteca obsoleta.

Operador	Modificador	Significado	Operandos	Descripción
DIV	(	División	Literal, variable, dirección directa de los tipos de datos INT, DINT, UINT, UDINT o REAL	Con DIV se divide el contenido del acumulador entre el valor del operando. Ejemplo: El ejemplo corresponde a la fórmula $D = A / B / C$. LD A DIV B DIV C ST D Nota: Para las divisiones con el tipo de datos Time está disponible la función DIVTIME de la biblioteca obsoleta.
MOD	(	División de módulo	Literal, variable, dirección directa de los tipos de datos INT, DINT, UINT o UDINT	Con MOD, el valor del primer operando se divide entre el valor del segundo operando, y el resto de la división (módulo) se emite como resultado. Ejemplo: En el siguiente ejemplo: ● C será 1 cuando A sea 7 y B sea 2. ● C será 1 cuando A sea 7 y B sea -2. ● C será -1 cuando A sea -7 y B sea 2. ● C será -1 cuando A sea -7 y B sea -2. LD A MOD B ST C

Operadores de comparación

Operadores de comparación del lenguaje de programación IL:

Operador	Modificador	Significado	Operandos	Descripción
GT	(	Comparación: >	Literal, variable, dirección directa de los tipos de datos BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT o TOD	Con GT se compara el contenido del acumulador con el contenido del operando. Si el contenido del acumulador es mayor que el contenido del operando, el resultado será un 1 booleano. Si el contenido del acumulador es menor o igual que el contenido del operando, el resultado será un 0 booleano. Ejemplo: En el ejemplo, el valor de D será 1 cuando A sea mayor que 10; de lo contrario, el valor de D será 0. LD A GT 10 ST D
GE	(	Comparación: >=	Literal, variable, dirección directa de los tipos de datos BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT o TOD	Con GE se compara el contenido del acumulador con el contenido del operando. Si el contenido del acumulador es mayor o igual que el contenido del operando, el resultado será un 1 booleano. Si el contenido del acumulador es menor que el contenido del operando, el resultado será un 0 booleano. Ejemplo: En el ejemplo, el valor de D será 1 cuando A sea mayor o igual que 10; de lo contrario, el valor de D será 0. LD A GE 10 ST D
EQ	(	Comparación: =	Literal, variable, dirección directa de los tipos de datos BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT o TOD	Con EQ se compara el contenido del acumulador con el contenido del operando. Si el contenido del acumulador es igual que el contenido del operando, el resultado será un 1 booleano. Si el contenido del acumulador no es igual que el contenido del operando, el resultado será un 0 booleano. Ejemplo: En el ejemplo, el valor de D será 1 cuando A sea igual que 10; de lo contrario, el valor de D será 0. LD A EQ 10 ST D

Operador	Modificador	Significado	Operandos	Descripción
NE	(	Comparación: <>	Literal, variable, dirección directa de los tipos de datos BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT o TOD	Con NE se compara el contenido del acumulador con el contenido del operando. Si el contenido del acumulador no es igual que el contenido del operando, el resultado será un 1 booleano. Si el contenido del acumulador es igual que el contenido del operando, el resultado será un 0 booleano. Ejemplo: En el ejemplo, el valor de D será 1 cuando A no sea igual que 10; de lo contrario, el valor de D será 0. LD A NE 10 ST D
LE	(	Comparación: <=	Literal, variable, dirección directa de los tipos de datos BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT o TOD	Con LE se compara el contenido del acumulador con el contenido del operando. Si el contenido del acumulador es menor o igual que el contenido del operando, el resultado será un 1 booleano. Si el contenido del acumulador es mayor que el contenido del operando, el resultado será un 0 booleano. Ejemplo: En el ejemplo, el valor de D será 1 cuando A sea menor o igual que 10; de lo contrario, el valor de D será 0. LD A LE 10 ST D
LT	(	Comparación: <	Literal, variable, dirección directa de los tipos de datos BOOL, BYTE, WORD, DWORD, STRING, INT, DINT, UINT, UDINT, REAL, TIME, DATE, DT o TOD	Con LT se compara el contenido del acumulador con el contenido del operando. Si el contenido del acumulador es menor que el contenido del operando, el resultado será un 1 booleano. Si el contenido del acumulador es mayor o igual que el contenido del operando, el resultado será un 0 booleano. Ejemplo: En el ejemplo, el valor de D será 1 cuando A sea menor que 10; de lo contrario, el valor de D será 0. LD A LT 10 ST D

Operadores de llamada

Operadores de llamada del lenguaje de programación IL:

Operador	Modificador	Significado	Operandos	Descripción
CAL	C, CN (sólo si el contenido del acumulador es de tipo BOOL)	Llamada de un bloque de función, DFB o subrutina	Nombre de instancia del bloque de función, DFB o subrutina	Con CAL se llama un bloque de función, un DFB o una subrutina de forma condicional o incondicional. Consulte también " <i>Llamada de bloques de funciones elementales y bloques de funciones derivados, página 517</i> " y " <i>Llamada de subrutina, página 507</i> ".
FUNCTIONNAME	-	Ejecución de una función	Literal, variable, dirección directa (el tipo de datos depende de la función)	Con el nombre de función se ejecuta una función concreta. Consulte también " <i>Llamada de funciones elementales, página 512</i> ".
PROCEDURENAME	-	Ejecución de un procedimiento	Literal, variable, dirección directa (el tipo de datos depende del procedimiento)	Con el nombre de procedimiento se ejecuta un procedimiento concreto. Consulte también " <i>Llamada de procedimientos, página 529</i> ".

Operadores de estructuración

Operadores de estructuración del lenguaje de programación IL:

Operador	Modificador	Significado	Operandos	Descripción
JMP	C, CN (sólo si el contenido del acumulador es de tipo BOOL)	Salto a la etiqueta	LABEL	Con JMP se ejecuta un salto condicional o incondicional a una etiqueta. <i>Consulte también "Etiquetas y saltos, página 508".</i>
RET	C, CN (sólo si el contenido del acumulador es de tipo BOOL)	Retorno a la unidad organizativa del programa inmediatamente superior	-	Los operadores RETURN pueden usarse en DFB (bloques de funciones derivados) y en SR (subrutinas). Los operadores RETURN no pueden usarse en el programa principal. - En un DFB, un operador RETURN fuerza el retorno al programa que llamó al DFB. - El resto de la sección DFB que contiene el operador RETURN no se ejecuta. - Las siguientes secciones del DFB no se ejecutan. El programa que llamó al DFB se ejecutará después de volver del DFB. Si otro DFB llama al DFB, el DFB llamado se ejecutará después de volver. - En un SR, un operador RETURN fuerza el retorno al programa que llamó al SR. - El resto de la sección SR que contiene el operador RETURN no se ejecuta. El programa que llamó al SR se ejecutará después de volver del SR.
)	-	Procesamiento de operaciones retenidas	-	Con el paréntesis derecho) se inicia el procesamiento del operador restablecido. La cantidad de operaciones con paréntesis derecho debe ser igual a la cantidad de modificadores de paréntesis izquierdo. Los paréntesis pueden estar intercalados. Ejemplo: En el ejemplo, E será 1 cuando C y/o D sean 1 y A y B sean 1. <pre>LD A AND B AND (C OR D) ST E</pre>

Llamada de subrutina

Llamada de subrutina

La llamada de una subrutina está formada por el operador `CAL`, seguido del nombre de la sección de subrutina y una lista de parámetros vacía (opcional).

Las llamadas de subrutinas no devuelven ningún valor.

El subprograma invocante debe encontrarse en la misma tarea que la sección IL invocante.

También es posible llamar subrutinas ubicadas dentro de otras subrutinas.

P. ej.

```
ST A  
CAL NombreSubrutina ()  
LD B
```

O bien

```
ST A  
CAL NombreSubrutina  
LD B
```

Las llamadas de subrutina son una ampliación de la norma CEI 61131-3 y se deben habilitar de forma explícita.

En las secciones de acción SFC sólo se admiten llamadas de subrutina si está habilitada la modalidad Multi-Token.

Etiquetas y saltos

Introducción

Las etiquetas sirven como punto de destino de los saltos.

Propiedades de las etiquetas:

Propiedades de las etiquetas:

- Las etiquetas deben ser siempre el primer elemento de una fila.
- Las marcas deben ser unívocas en toda la sección sin que se diferencie entre mayúsculas y minúsculas.
- Las etiquetas pueden tener una extensión máxima de 32 caracteres.
- Las etiquetas deben cumplir la nomenclatura CEI.
- Las etiquetas se separan con dos puntos : de las instrucciones siguientes.
- Las etiquetas sólo pueden aparecer al principio de una "expresión"; de lo contrario, podría haber un valor indefinido en la batería.

Ejemplo:

```
start: LD A
      AND B
      OR C
      ST D
      JMP start
```

Propiedades de los saltos:

Propiedades de los saltos

- Las operaciones `JMP` ejecutan de forma condicional o incondicional un salto hasta una etiqueta.
- `JMP` puede utilizarse con los modificadores `C` y `CN` (sólo cuando el contenido actual del acumulador es del tipo de datos `BOOL`).
- Los saltos se pueden realizar dentro de secciones de un programa y de un DFB.
- Los saltos sólo se pueden realizar dentro de la sección actual.

Los destinos del salto posibles son:

- La primera instrucción `LD` de una llamada de EFB o de DFB con asignación de parámetros de entrada (consulte `start2`),
- Una instrucción `LD` "normal" (consulte `start1`),
- Una instrucción `CAL`, que no funciona con asignación de parámetros de entrada (consulte `start3`),
- Una instrucción `JMP` (consulte `start4`),
- El final de una lista de instrucciones (consulte `start5`).

Ejemplo

```
start2: LD A
        ST counter.CU
        LD B
        ST counter.R
        LD C
        ST counter.PV
        CAL counter
        JMPCN start4
start1: LD A
        AND B
        OR C
        ST D
        JMPC start3
        LD A
        ADD E
        JMP start5
start3: CAL counter (
        CU:=A
        R:=B
        PV:=C )
        JMP start1
        LD A
        OR B
        OR C
        ST D
start4: JMPC start1
        LD C
        OR B
start5: ST A
```

Comentario

Descripción

En el editor IL, los comentarios comienzan con la cadena (* y terminan con *) .
Entre estas dos cadenas se puede introducir cualquier comentario.

Según la normativa CEI 61131-3, no se puede intercalar comentarios. Sin embargo, si éstos se intercalan, deben habilitarse de forma explícita.

14.2 Llamada de funciones elementales, módulos de función elementales, módulos de función derivados y procedimientos

Vista general

Llamada de funciones elementales, módulos de función elementales, módulos de función derivados y procedimientos en el lenguaje de programación IL.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Llamada de funciones elementales	512
Llamada de bloques de funciones elementales y bloques de funciones derivados	517
Llamada de procedimientos	529

Llamada de funciones elementales

Aplicación de las funciones

Las funciones elementales están disponibles en forma de bibliotecas. La lógica de las funciones se define en el lenguaje de programación C y no se puede modificar en el editor IL.

Las funciones no tienen estados internos. Si los valores de entrada son los mismos, el valor de salida es el mismo en cada ejecución de la función. Por ejemplo, la suma de dos valores siempre da el mismo resultado. En el caso de algunas funciones elementales, se puede ampliar la cantidad de entradas.

Las funciones elementales tienen un único valor de retorno (salida).

Parámetros

Para transferir valores a una función o aplicarlos desde ella, es necesario utilizar varias entradas y una salida. A éstas se les llama parámetros formales.

Los estados de proceso actuales se transmiten a los parámetros formales. A éstos se les llama parámetros actuales.

Como parámetro actual para las entradas de la función se puede utilizar:

- Variable
- Dirección
- Literal

Como parámetro actual para las salidas de la función se puede utilizar:

- Variable
- Dirección

El tipo de datos del parámetro actual debe coincidir con el tipo de datos del parámetro formal. Las únicas excepciones son los parámetros formales genéricos, cuyo tipo de datos está condicionado por el parámetro actual.

En el caso de los parámetros formales genéricos `ANY_BIT`, se pueden emplear parámetros reales de tipos de datos `INT` o `DINT` (pero no `UINT` ni `UDINT`).

Se trata de una ampliación de la norma CEI 61131-3 y se debe habilitar de forma explícita.

Ejemplo:

Se admite:

```
AND (AnyBitParam := IntVar1, AnyBitParam2 := IntVar2)
```

No se admite:

```
AND_WORD (WordParam1 := IntVar1, WordParam2 := IntVar2)
```

(En este caso se debe utilizar `AND_INT`).

```
AND_ARRAY_WORD (ArrayInt, ...)
```


(En este caso se debe realizar una conversión de tipo explícita a través de `INT_ARR_TO_WORD_ARR ( . . . )`).

En el caso de las llamadas formales, no es necesario en principio asignar un valor a todos los parámetros formales. En la tabla encontrará los tipos de parámetros formales que sí lo requieren obligatoriamente.

Tipo de parámetro	EDT	STRING	ARRAY	ANY_ARRAY	IODDT	STRUCT	FB	ANY
Entrada	-	-	+	+	+	+	+	+
VAR_IN_OUT	+	+	+	+	+	+	/	+
Salida	-	-	-	-	-	-	/	-
+ Parámetro actual requerido obligatoriamente								
- Parámetro actual no requerido obligatoriamente								
/ No aplicable								

Si no se asigna un valor a un parámetro formal, durante la ejecución de la función se utilizará el valor inicial. Si no se ha definido ningún valor inicial, se aplicará el valor predeterminado (0).

Notas sobre la programación

Tenga en cuenta las siguientes indicaciones sobre la programación:

- Las funciones sólo se ejecutan si la entrada `EN` es 1 o si la entrada `EN` no se utiliza (consulte también "*EN y ENO, página 516*").
- Todas las funciones genéricas están cargadas. Es decir, las funciones se pueden llamar con o sin la indicación del tipo de datos.

P. ej.

```
LD i1
ADD i2
ST i3
es igual que
LD i1
ADD_INT i2
ST i3
```

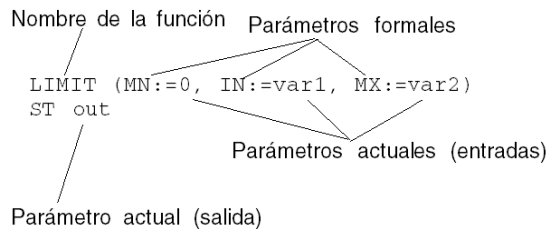
- En IL, al contrario que en ST, las funciones no son intercalables.
- Hay dos formas de llamar una función:
 - Llamada formal (llamada de una función con nombres de parámetros formales)
 - Llamada informal (llamada de una función sin nombres de parámetros formales)

Llamada formal

Con este tipo de llamada (llamada con nombres de parámetros formales), las funciones se llaman por medio de una secuencia de instrucciones compuesta por el nombre de la función seguido de una lista entre paréntesis con asignaciones de valores (parámetros actuales) a los parámetros formales. La secuencia en la que se enumeran los parámetros formales **no es significativa**. La lista de los parámetros actuales se puede cambiar de línea automáticamente después de cada coma. Tras la ejecución de la función, el resultado se carga en el acumulador y se puede guardar con ST.

Con este tipo de llamada se puede utilizar EN y ENO.

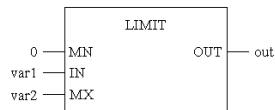
Llamada de una función con nombres de parámetros formales



O bien

```
LIMIT (
MN:=0,
IN:=var1,
MX:=var2
)
ST out
```

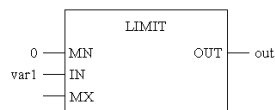
Llamada de la misma función en FBD



Con la llamada informal no es necesario asignar un valor a todos los parámetros formales (consulte también "*Parámetros, página 512*").

```
LIMIT (MN:=0, IN:=var1)
ST out
```

Llamada de la misma función en FBD

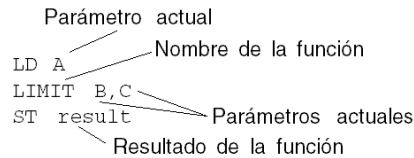


Llamada informal

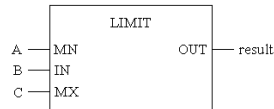
Con este tipo de llamada (llamada sin nombres de parámetros formales), las funciones se llaman por medio de una secuencia de instrucciones compuesta por la carga del primer parámetro actual en el acumulador, el nombre de la función y una lista opcional de parámetros actuales. La secuencia en la que se enumeran los parámetros actuales es **significativa**. La lista de los parámetros actuales no se puede cambiar de línea. Tras la ejecución de la función, el resultado se carga en el acumulador y se puede guardar con `ST`.

Con este tipo de llamada **no** es posible utilizar `EN` y `ENO`.

Llamada de una función con nombres de parámetros formales



Llamada de la misma función en FBD



NOTA: Tenga en cuenta que con las llamadas informales, la lista de parámetros actuales **no** se puede escribir entre paréntesis. En este caso, la norma CEI 61133-3 requiere que se omitan los paréntesis para indicar que el primer parámetro real no forma parte de la lista.

Llamada informal **inválida** de una función:

```
LD A
LIMIT (B,C)
```

Si el valor que se va a procesar (primer parámetro actual) ya se encuentra en el acumulador, se puede suprimir la instrucción de carga.

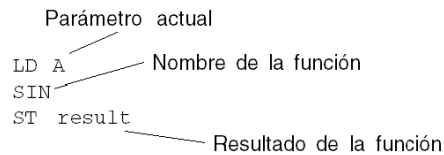
```
LIMIT B,C
ST result
```

Si va a continuar procesando el resultado inmediatamente, puede suprimir la instrucción de almacenamiento.

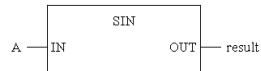
```
LD A
LIMIT_REAL B,C
MUL E
```

Si la función que se va a ejecutar sólo tiene una entrada, el nombre de la función no irá seguido por ninguna lista de parámetros actuales.

Llamada de una función con un parámetro real:



Llamada de la misma función en FBD



EN y ENO

En todas las funciones se puede configurar una entrada **EN** y una salida **ENO**.

Si el valor de **EN** es "0", al llamar la función no se ejecutarán los algoritmos definidos por dicha función, y **ENO** se ajustará a "0".

Si el valor de **EN** es "1", al llamar la función se ejecutarán los algoritmos definidos por dicha función. Si no hay problemas en la ejecución de estos algoritmos, el valor de **ENO** será "1". Si se produce un error, **ENO** será "0".

Si **ENO** está ajustado a "0" (porque **EN**=0 o debido a un error en la ejecución), la salida de la función se ajustará a "0".

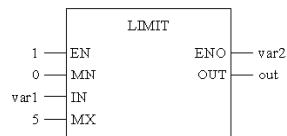
El comportamiento de salida de la función no depende de si dicha función se ha llamado sin **EN/ENO** o con **EN=1**.

Si va a utilizar **EN/ENO**, la llamada de la función deberá ser formal.

LIMIT (**EN:=1**, MN:=0, IN:=var1, MX:=5, **ENO=>var2**)

ST out

Llamada de la misma función en FBD



Llamada de bloques de funciones elementales y bloques de funciones derivados

Bloque de función elemental

Los bloques de funciones elementales tienen estados internos. Si las entradas tienen los mismos valores, el valor de la salida puede ser otro durante las operaciones individuales. Por ejemplo, con un contador, el valor de la salida se incrementa.

Los bloques de funciones pueden disponer de varios valores de retorno (salidas).

Bloque de función derivado

Los bloques de funciones derivados (DFB) presentan las mismas propiedades que los bloques de funciones elementales. Sin embargo, el usuario los crea en los lenguajes de programación FBD, LD, IL o ST.

Parámetro

Para transferir valores a un bloque de función o aplicarlos desde él, es necesario utilizar entradas y salidas. A éstas se les llama parámetros formales.

Los estados de proceso actuales se transmiten a los parámetros formales. A éstos se les llama parámetros actuales.

Como parámetro actual para las entradas del bloque de función se puede utilizar:

- Variable
- Dirección
- Literal

Como parámetro actual para las salidas del bloque de función se puede utilizar:

- Variable
- Dirección

El tipo de datos del parámetro actual debe coincidir con el tipo de datos del parámetro formal. Las únicas excepciones son los parámetros formales genéricos, cuyo tipo de datos está condicionado por el parámetro actual.

Excepción:

En el caso de los parámetros formales genéricos del tipo de datos `ANY_BIT` se pueden utilizar parámetros actuales de los tipos de datos `INT` y `DINT` (ni `UINT` ni `UDINT`).

Se trata de una ampliación de la norma CEI 61131-3 y se debe habilitar de forma explícita.

Ejemplo:

Se admite:

```
AND (AnyBitParam := IntVar1, AnyBitParam2 := IntVar2)
```

No se admite:

AND_WORD (WordParam1 := IntVar1, WordParam2 := IntVar2)

(En este caso se debe utilizar AND_INT).

AND_ARRAY_WORD (ArrayInt, ...)

(En este caso se debe realizar una conversión de tipo explícita a través de INT_ARR_TO_WORD_ARR (...)).

En principio, no es necesario asignar un valor a todos los parámetros formales. En la tabla encontrará los tipos de parámetros formales que sí lo requieren obligatoriamente.

Tipo de parámetro	EDT	STRING	ARRAY	ANY_ARRAY	IODDT	STRUCT	FB	ANY
EFB: entrada	-	+	+	+	/	+	/	+
EFB: VAR_IN_OUT	+	+	+	+	+	+	/	+
EFB: salida	-	-	+	+	+	-	/	+
DFB: entrada	-	+	+	+	/	+	/	+
DFB: VAR_IN_OUT	+	+	+	+	+	+	/	+
DFB: salida	-	-	+	/	/	-	/	+
+ Parámetro actual requerido obligatoriamente								
- Parámetro actual no requerido obligatoriamente								
/ No aplicable								

Si no se asigna un valor a un parámetro formal, durante la ejecución del bloque de función se utilizará el valor inicial. Si no se ha definido ningún valor inicial, se aplicará el valor predeterminado (0).

Si un parámetro formal no tiene asignado ningún valor y el bloque de función/DFB se ha instanciado varias veces, las instancias que se ejecuten a partir de ese momento trabajarán con el valor antiguo.

Variables públicas

Además de las entradas y salidas, algunos bloques de funciones también disponen de las denominadas variables públicas.

Estas variables sirven para transmitir valores estáticos (valores no influidos por el proceso) al bloque de función. Se utilizan para la parametrización del bloque de función.

Las Public Variables son una ampliación de la norma CEI 61131-3.

La asignación de valores a las Public Variables se realiza a través de sus valores iniciales o a través de instrucciones de carga y almacenamiento.

Ejemplo:

```

      Nombre de instancia      Variable pública
      /      \
LD 1      /
ST D_ACT1.OP_CTRL

```

(D_ACT1 es una instancia del bloque de función D_ACT y dispone de las variables públicas AREA_NR y OP_CTRL).

La lectura de los valores de las Public Variables se realiza a través del nombre de instancia del bloque de función y a través del nombre de la Public Variable.

Ejemplo:

```

      Nombre de instancia      Variable pública
      /      \
LD D_ACT1.OP_CTRL
ST Var1

```

Variables privadas

Además de las entradas, las salidas y las variables públicas, algunos bloques de funciones también disponen de las denominadas variables privadas.

Al igual que las variables públicas, las privadas se utilizan para transferir valores estadísticos (valores no influidos por el proceso) al bloque de función.

El programa usuario no puede acceder a las variables privadas. Sólo se puede acceder a este tipo de variables mediante la tabla de animación.

NOTA: Los DFB intercalados se declaran como variables privadas del DFB principal. Por tanto, tampoco se puede acceder a sus variables a través de la programación, sino a través de la tabla de animación.

Las variables privadas constituyen un suplemento a la normativa CEI 61131-3.

Notas sobre la programación

Tenga en cuenta las siguientes indicaciones sobre la programación:

- Las funciones sólo se ejecutan si la entrada **EN** es 1 o si la entrada **EN** no se utiliza (consulte también "*EN y ENO, página 526*").
- La asignación de variables a salidas del tipo **ANY** o **ARRAY** se debe realizar mediante el **=>** operador (consulte también "*Forma formal de CAL con lista de parámetros de entrada, página 520*").

No es posible realizar una asignación fuera de la llamada de bloque de función.

La instrucción

```
My_Var := My_SAH.OUT
```

es **inválida**, la salida **OUT** del bloque de función **SAH** es del tipo **ANY**.

La instrucción

```
Cal My_SAH (OUT=>My_Var)
```

es, por el contrario, **válida**.

- Si se utilizan variables VAR_IN_OUT (véase página 526), se imponen condiciones especiales.
- La utilización de los bloques de funciones consta de dos partes:
 - Declaración (véase página 520)
 - Llamada del bloque de función
- Hay cuatro formas de llamar un bloque de función:
 - Forma formal de CAL con lista de los parámetros de entrada (véase página 520) (llamada con nombres de parámetros formales)
En este caso es posible asignar variables a las salidas mediante el operador =>.
 - Forma informal de CAL con lista de los parámetros de entrada (véase página 522) (llamada sin nombres de parámetros formales)
 - CAL y carga/almacenado (véase página 523) de los parámetros de entrada
 - Uso de los operadores de entrada (véase página 524)
- Las instancias de bloques de funciones/DBF se pueden ejecutar varias veces, excepto las instancias de EFB de comunicaciones que sólo se pueden ejecutar una única vez (consulte "Llamada múltiple de una instancia de bloque de función, página 525").

Declaración

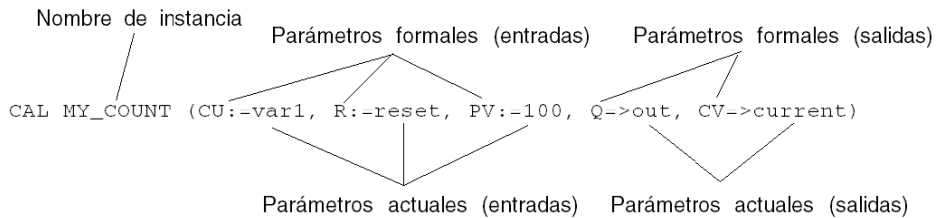
Antes de llamar un bloque de función es necesario declararlo primero en el editor de variables.

Forma formal de CAL con lista de parámetros de entrada

Con este tipo de llamada (llamada con nombres de parámetros formales), los bloques de funciones se llaman por medio de una instrucción formada por la instrucción **CAL** seguida del nombre de instancia del bloque de funciones y una lista entre paréntesis de asignaciones de parámetros actuales a los parámetros formales. La asignación de los parámetros formales de entrada se realiza mediante la asignación **: =**, y la asignación de los parámetros formales de salida, mediante la asignación **=>**. La secuencia en la que se enumeran los parámetros formales de entrada y de salida **no es significativa**. La lista de los parámetros actuales se puede cambiar de línea automáticamente después de cada coma.

Con este tipo de llamada se puede utilizar **EN** y **ENO**.

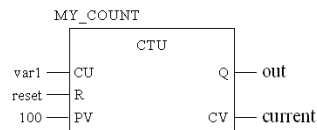
Llamada de un bloque de función en forma formal de CAL con lista de los parámetros de entrada:



O bien

```
CAL MY_COUNT (CU:=var1,
               R:=reset,
               PV:=100,
               Q=>out,
               CV=>current)
```

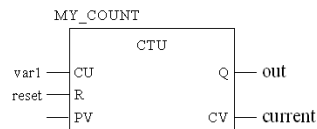
Llamada del mismo bloque de función en FBD:



No es necesario asignar un valor a todos los parámetros formales (consulte también "*Parámetro, página 517*").

```
CAL MY_COUNT (CU:=var1, R:=reset, Q=>out, CV=>current)
```

Llamada del mismo bloque de función en FBD:



El valor de una salida de bloque de función se puede almacenar cargando la salida del bloque de función (nombre de instancia del bloque de función y parámetro formal separado por un punto) y guardándola a continuación.

Carga y almacenamiento de las salidas del bloque de función:

Nombre de instancia Parámetro formal (salida)

```
LD MY_COUNT.Q
ST out
LD MY_COUNT.CV
ST current
```

Parámetro actual (salida)

Forma informal de CAL con lista de parámetros de entrada

Con este tipo de llamada (llamada sin nombres de parámetros formales), los bloques de funciones se llaman por medio de una instrucción formada por la instrucción **CAL** seguida del nombre de instancia del bloque de función y una lista entre paréntesis de los parámetros actuales de las entradas y las salidas. La secuencia en la que se enumeran los parámetros actuales en una llamada de bloque de función **es significativa**. La lista de los parámetros actuales no se puede cambiar de línea.

Con este tipo de llamada **no** es posible utilizar **EN** y **ENO**.

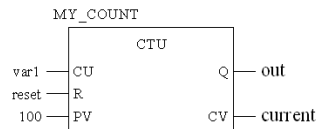
Llamada de un bloque de función en forma informal de CAL con lista de los parámetros de entrada:

Nombre de instancia Parámetros actuales (entradas)

```
CAL MY_COUNT (var1, reset, 100, out, current)
```

Parámetros actuales (salidas)

Llamada del mismo bloque de función en FBD:



Con la llamada informal tampoco es necesario asignar un valor a todos los parámetros formales (consulte también "*Parámetro, página 517*").

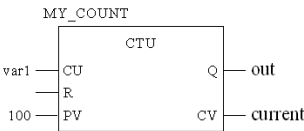
Se trata de una ampliación de la norma CEI 61131-3 y se debe habilitar de forma explícita.

Para excluir un parámetro se utiliza un campo de parámetros vacío.

Llamada con campo de parámetros vacío

```
CAL MY_COUNT (var1, , 100, out, current)
```

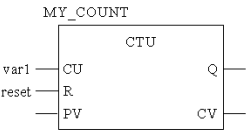
Llamada del mismo bloque de función en FBD:



Si se suprimen los parámetros formales al final, no se tiene que utilizar un campo de parámetros vacío.

MY_COUNT (var1, reset)

Llamada de los mismos bloques de funciones en FBD



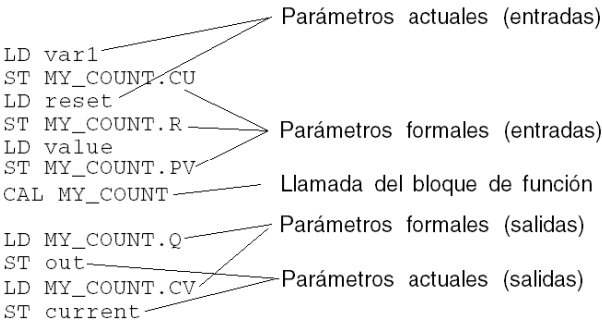
CAL y carga/almacenamiento de los parámetros de entrada

Los bloques de funciones se pueden llamar mediante una lista de instrucciones compuesta por la carga de los parámetros actuales, seguida por el almacenamiento en los parámetros formales y la instrucción **CAL**. La secuencia en la que se cargan y se almacenan los parámetros **no es significativa**.

Entre la primera instrucción de carga del parámetro actual y la llamada del bloque de función sólo pueden aparecer instrucciones de carga y almacenamiento para el bloque de función que se vaya a parametrizar en ese momento. Todas las demás instrucciones no se admiten en esta posición.

No es necesario asignar un valor a todos los parámetros formales (consulte también "*Parámetro, página 517*").

CAL con carga/almacenamiento de los parámetros de entrada:



Uso de los operadores de entrada

Los bloques de funciones se pueden llamar mediante una lista de instrucciones compuesta por la carga de los parámetros actuales, el almacenamiento en los parámetros formales y un operador de entrada. La secuencia en la que se cargan y se almacenan los parámetros **no es significativa**.

Entre la primera instrucción de carga del parámetro actual y el operador de entrada del bloque de función sólo pueden aparecer instrucciones de carga y almacenamiento para el bloque de función que se vaya a parametrizar en ese momento. Todas las demás instrucciones no se admiten en esta posición.

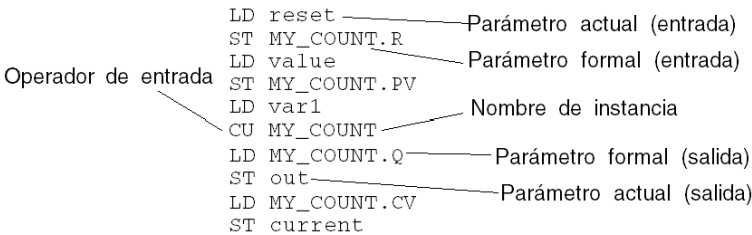
Con este tipo de llamada **no** es posible utilizar **EN** y **ENO**.

No es necesario asignar un valor a todos los parámetros formales (consulte también "*Parámetro, página 517*").

En la tabla siguiente figuran los operadores de entrada posibles para los distintos bloques de funciones. No hay disponible ningún otro operador de entrada.

Operador de entrada	Tipo de FB
S1, R	SR
S, R1	RS
CLK	R_TRIG
CLK	F_TRIG
CU = R - PV	CTU_INT, CTU_DINT, CTU_UINT, CTU_UDINT
CD, LD, PV	CTD_INT, CTD_DINT, CTD_UINT, CTD_UDINT
CU, CD, R, LD, PV	CTUD_INT, CTUD_DINT, CTUD_UINT, CTUD_UDINT
IN, PT	TP
IN, PT	TON
IN, PT	TOF

Uso de los operadores de entrada:



Llamada de un bloque de función sin entradas

Aunque el bloque de función no tenga ninguna entrada o no sea necesario parametrizar sus entradas, hay que llamar el bloque de función antes de poder utilizar sus salidas. De lo contrario, se transmitirán los valores iniciales de las salidas, es decir "0".

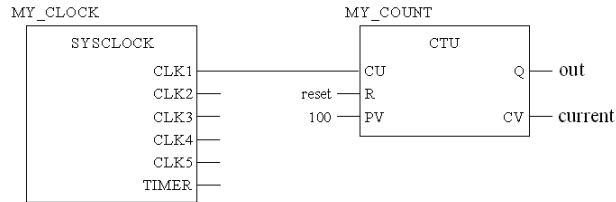
P. ej.

Llamada de los bloques de funciones en IL

CAL MY_CLOCK ()

```
CAL MY_COUNT (CU:=MY_CLOCK.CLK1, R:=reset, PV:=100)
LD MY_COUNT.Q
ST out
LD MY_COUNT.CV
ST current
```

Llamada del mismo bloque de función en FBD:



Llamada múltiple de una instancia de bloque de función

Las instancias de bloques de funciones/DBF se pueden llamar varias veces, excepto las instancias de EFB de comunicaciones, que sólo se pueden llamar una única vez.

La llamada múltiple de una misma instancia de bloque de función/DFB resulta conveniente en los siguientes casos:

- Cuando el bloque de función/DFB no posee ningún valor interno o los valores internos no son necesarios para continuar el procesamiento.
En este caso, la llamada múltiple de una misma instancia de bloque de función/DFB permite ahorrar espacio en memoria, ya que el código del bloque de función/DFB sólo se carga una única vez.
El bloque de función/DFB se procesa, por así decirlo, como una función.
- Cuando el bloque de función/DFB tiene valores internos y éstos se deben modificar en varios puntos del programa, por ejemplo, si el valor de un contador se debe aumentar en diversos puntos del programa.
En este caso, con la llamada múltiple de la misma instancia de bloque de función/DFB no es necesario guardar los resultados intermedios para continuar el procesamiento en otro lugar del programa.

EN y ENO

En todos los bloques de funciones/DFB se puede configurar una entrada **EN** y una salida **ENO**.

Si el valor de **EN** es "0", al llamar el bloque de función/DFB no se ejecutarán los algoritmos definidos por dicho bloque de función/DFB, y **ENO** se establecerá en "0".

Si el valor de **EN** es "1", al llamar el bloque de función/DFB se ejecutarán los algoritmos definidos por dicho bloque de función/DFB. Si no hay problemas en la ejecución de estos algoritmos, el valor de **ENO** será "1". Si se produce un error, **ENO** será "0".

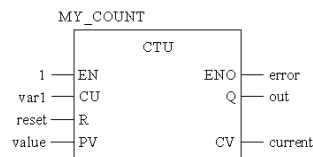
Si **ENO** se ajusta a "0" (debido a que **EN** es igual a 0 o a un error producido durante la ejecución), las salidas del bloque de función/DFB conservan el estado que tenían en el último ciclo ejecutado correctamente.

El comportamiento de salida de los bloques de funciones/DFB no depende de si dichos bloques de funciones/DFB se han llamado sin **EN/ENO** o con **EN=1**.

Si se va a utilizar **EN/ENO**, la llamada del bloque de función deberá ser formal. La asignación de una variable a **ENO** debe realizarse con el operador **=>**.

```
CAL MY_COUNT (EN:=1, CU:=var1, R:=reset, PV:=value,
              ENO=>error, Q=>out, CV=>current) ;
```

Llamada del mismo bloque de función en FBD:



Variable VAR_IN_OUT

A menudo, los bloques de funciones se utilizan para leer una variable en la entrada (variables de entrada), procesarla y volver a emitir los valores modificados de la misma variable (variables de salida). Este caso específico de variable de entrada/salida también se denomina variable **VAR_IN_OUT**.

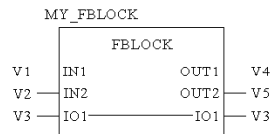
Tenga en cuenta las siguientes particularidades al utilizar bloques de funciones/DFB con variables **VAR_IN_OUT**:

- Hay que asignar obligatoriamente una variable a todas las entradas **VAR_IN_OUT**.
- No se puede asignar ningún literal ni ninguna constante a las entradas **VAR_IN_OUT**.
- No es posible asignar **ningún** valor a las salidas **VAR_IN_OUT**.
- Las variables **VAR_IN_OUT** **no** se pueden utilizar fuera de la llamada de bloque.

Llamada de un bloque de función con variable **VAR_IN_OUT** en IL

```
CAL MY_FBLOCK(IN1:=V1, IN2:=V2, IO1:=V3,  
              OUT1=>V4, OUT2=>V5)
```

Llamada del mismo bloque de función en FBD:



Las variables VAR_IN_OUT **no** se pueden utilizar fuera de la llamada de un bloque de función.

Por este motivo las siguientes llamadas de bloque de función son **inválidas**:

Llamada **inválida**, ejemplo 1.

LD V1	Carga de la variable V1 en el acumulador.
CAL InOutFB	Llamada de un bloque de función con parámetros VAR_IN_OUT. El acumulador está cargado ahora con la referencia a un parámetro VAR_IN_OUT.
AND V2	Operación AND del contenido del acumulador con la variable V2. Errores: No se puede realizar la operación porque no es posible acceder al parámetro VAR_IN_OUT (contenido del acumulador) fuera de la llamada del bloque de función.

Llamada **inválida**, ejemplo 2.

LD V1	Carga de la variable V1 en el acumulador.
AND InOutFB.inout	Operación AND del contenido del acumulador con la referencia a un parámetro VAR_IN_OUT. Errores: No se puede realizar la operación porque no es posible acceder al parámetro VAR_IN_OUT fuera de la llamada del bloque de función.

Por el contrario, las siguientes llamadas del bloque de función son **válidas**:

Llamada **válida**, ejemplo 1.

CAL InOutFB (IN1:=V1,inout:=V2	Llamada de un bloque de función con parámetro VAR_IN_OUT y asignación de los parámetros actuales dentro de la llamada del bloque de función.

Llamada **válida**, ejemplo 2.

LD V1	Carga de la variable V1 en el acumulador.
ST InOutFB.IN1	Asignación del contenido del acumulador al parámetro IN1 del bloque de función IN1.
CAL InOutFB(inout:=V2)	Llamada del bloque de función con asignación del parámetro actual (V2) al parámetro VAR_IN_OUT.

Llamada de procedimientos

Procedimiento

Los procedimientos están disponibles en forma de bibliotecas. La lógica de los procedimientos se define en el lenguaje de programación C y no se puede modificar en el editor IL.

Los procedimientos, al igual que las funciones, no tienen estados internos. Si los valores de entrada son los mismos, el valor de salida es el mismo en cada ejecución del procedimiento. Por ejemplo, la suma de dos valores siempre da el mismo resultado.

Al contrario de lo que sucede con las funciones, los procedimientos no emiten valores de retorno y admiten variables `VAR_IN_OUT`.

Los procedimientos son una ampliación de la norma IEC 61131-3 y se deben habilitar de forma explícita.

Parámetro

Para transferir valores a un procedimiento o aplicarlos desde él, es necesario utilizar entradas y salidas. A éstas se les llama parámetros formales.

Los estados de proceso actuales se transmiten a los parámetros formales. A éstos se les llama parámetros actuales.

Como parámetro actual para las entradas de un procedimiento se puede utilizar:

- Variable
- Dirección
- Literal

Como parámetro actual para las salidas de un procedimiento se puede utilizar:

- Variable
- Dirección

El tipo de datos del parámetro actual debe coincidir con el tipo de datos del parámetro formal. Las únicas excepciones son los parámetros formales genéricos, cuyo tipo de datos está condicionado por el parámetro actual.

En el caso de los parámetros formales genéricos `ANY_BIT`, se pueden emplear los parámetros reales de tipos de datos `INT` o `DINT` (pero no `UINT` ni `UDINT`).

Se trata de una ampliación de la norma CEI 61131-3 y se debe habilitar de forma explícita.

Ejemplo:

Se admite:

```
AND (AnyBitParam := IntVar1, AnyBitParam2 := IntVar2)
```

No se admite:

```
AND_WORD (WordParam1 := IntVar1, WordParam2 := IntVar2)
```

(En este caso se debe utilizar `AND_INT`).

`AND_ARRAY_WORD (ArrayInt, ...)`

(En este caso se debe realizar una conversión de tipo explícita a través de

`INT_ARR_TO_WORD_ARR (...)`).

En el caso de las llamadas formales, no es necesario en principio asignar un valor a todos los parámetros formales. En la tabla encontrará los tipos de parámetros formales que sí lo requieren obligatoriamente.

Tipo de parámetro	EDT	STRING	ARRAY	ANY_ARRAY	IODDT	STRUCT	FB	ANY
Entrada	-	-	+	+	+	+	+	+
VAR_IN_OUT	+	+	+	+	+	+	/	+
Salida	-	-	-	-	-	-	/	+
+ Parámetro actual requerido obligatoriamente								
- Parámetro actual no requerido obligatoriamente								
/ No aplicable								

Si no se asigna un valor a un parámetro formal, durante la ejecución del bloque de función se utilizará el valor inicial. Si no se ha definido ningún valor inicial, se aplicará el valor predeterminado (0).

Notas sobre la programación

Tenga en cuenta las siguientes indicaciones sobre la programación:

- Los procedimientos sólo se ejecutarán si la entrada `EN` es igual a 1 o si la entrada `EN` no se utiliza (consulte también "*EN y ENO, página 533*").
- Si se utilizan variables `VAR_IN_OUT` (véase *página 534*), se imponen condiciones especiales.
- Hay dos formas de llamar los procedimientos:
 - Llamada formal (llamada de una función con nombres de parámetros formales).
En este caso es posible asignar variables a las salidas mediante el operador `=>` (llamada de un bloque de función de forma abreviada).
 - Llamada informal (llamada de una función sin nombres de parámetros formales)

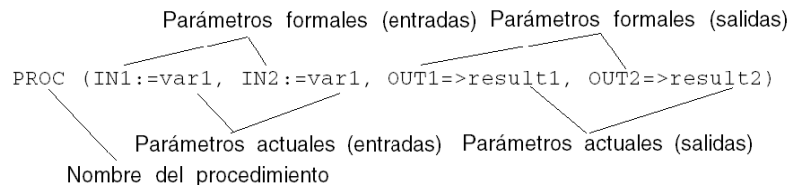
Llamada formal

Con esta forma de llamada (llamada con nombres de parámetros formales), los procedimientos se llaman por medio de una secuencia de instrucciones compuesta por una instrucción `CAL` opcional seguida del nombre del procedimiento y de una lista entre paréntesis de asignaciones de parámetros actuales a los parámetros formales. La asignación de los parámetros formales de entrada se realiza mediante la asignación `:=`, y la asignación de los parámetros formales de salida, mediante la asignación `=>`. La secuencia en la que se enumeran los parámetros formales de entrada y de salida **no es significativa**.

La lista de los parámetros actuales se puede cambiar de línea automáticamente después de cada coma.

Con este tipo de llamada se puede utilizar `EN` y `ENO`.

Llamada de un procedimiento con nombres de parámetros formales



O bien

```
CAL PROC (IN1:=var1, IN2:=var1, OUT1=>result1, OUT2=>result2)
```

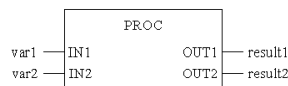
O bien

```
PROC (IN1:=var1,
      IN2:=var1,
      OUT1=>result1,
      OUT2=>result2)
```

o

```
CAL PROC (IN1:=var1,
          IN2:=var1,
          OUT1=>result1,
          OUT2=>result2)
```

Llamada del mismo procedimiento en FBD



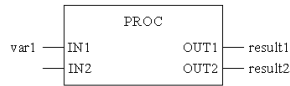
Con la llamada informal no es necesario asignar un valor a todos los parámetros formales (consulte también "*Parámetro, página 529*").

```
PROC (IN1:=var1, OUT1=>result1, OUT2=>result2)
```

O

```
CAL PROC (IN1:=var1, OUT1=>result1, OUT2=>result2)
```

Llamada del mismo procedimiento en FBD

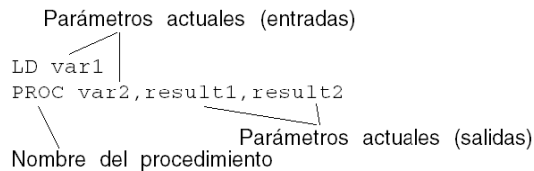


Llamada informal sin instrucción CAL

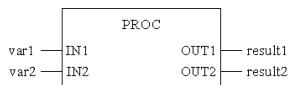
Con este tipo de llamada (llamada sin nombres de parámetros formales), los procedimientos se llaman por medio de una secuencia de instrucciones compuesta por la carga del primer parámetro actual en el acumulador, el nombre del procedimiento y una lista de los parámetros actuales de las entradas y las salidas. La secuencia en la que se enumeran los parámetros actuales es **significativa**. La lista de los parámetros actuales no se puede cambiar de línea.

Con este tipo de llamada **no** es posible utilizar EN y ENO.

Llamada de un procedimiento con nombres de parámetros formales



Llamada del mismo procedimiento en FBD



NOTA: Tenga en cuenta que con las llamadas informales, la lista de parámetros actuales **no** se puede escribir entre paréntesis. En este caso, la norma CEI 61133-3 requiere que se omitan los paréntesis para indicar que el primer parámetro real no forma parte de la lista.

Llamada informal **no válida** de un procedimiento:

```
LD A
LIMIT (B,C)
```

Si el valor que se va a procesar (primer parámetro actual) ya se encuentra en el acumulador, se puede suprimir la instrucción de carga.

```
EXAMP1 var2,result1,result2
```

Llamada informal con instrucción CAL

Con este tipo de llamada, los procedimientos se llaman por medio de una secuencia de instrucciones compuesta por la instrucción `CAL`, seguida del nombre del procedimiento y una lista entre paréntesis de los parámetros actuales de las entradas y las salidas. La secuencia en la que se enumeran los parámetros actuales es **significativa**. La lista de los parámetros actuales no se puede cambiar de línea.

Con este tipo de llamada **no** es posible utilizar `EN` y `ENO`.

Llamada de un procedimiento con nombres de parámetros formales con instrucción `CAL`:

Parámetros actuales (entradas)

```
CAL PROC (var1,var2,result1,result2)
```

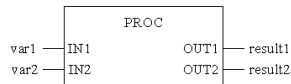
Nombre del procedimiento

Parámetros actuales (salidas)

O bien

```
CAL PROC (var1,
          var2,
          result1,
          result2)
```

Llamada del mismo procedimiento en FBD



NOTA: Al contrario de lo que sucede con las llamadas informales sin instrucción `CAL`, en las llamadas informales con instrucción `CAL`, el valor que se va a procesar (primer parámetro real) no se carga de forma explícita en el acumulador, sino que forma parte de la lista de los parámetros reales. Por esta razón, en las llamadas informales con instrucción `CAL`, la lista de los parámetros reales se debe poner entre paréntesis.

EN y ENO

En todos los procedimientos se puede configurar una entrada `EN` y una salida `ENO`.

Si el valor de `EN` es "0", al llamar la función no se ejecutarán los algoritmos definidos por dicho procedimiento y `ENO` se ajustará a "0".

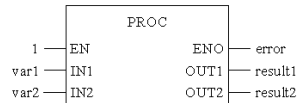
Si el valor de `EN` es "1", al llamar el procedimiento se ejecutarán los algoritmos con los que se haya definido dicho procedimiento. Si no hay problemas en la ejecución de estos algoritmos, el valor de `ENO` se convertirá en "1". Si se produce un error, `ENO` se convertirá en "0".

Si ENO está ajustado a "0" (porque EN=0 o debido a un error en la ejecución), las salidas del procedimiento se ajustarán a "0".

Si va a utilizar EN o ENO, la llamada del procedimiento deberá ser formal. La asignación de una variable a ENO debe realizarse con el operador =>.

```
PROC (EN:=1, IN1:=var1, IN2:=var2,
      ENO=>error, OUT1=>result1, OUT2=>result2) ;
```

Llamada del mismo procedimiento en FBD



Variable VAR_IN_OUT

A menudo, los procedimientos se utilizan para leer una variable en la entrada (variables de entrada), procesarla y volver a emitir los valores modificados de la misma variable (variables de salida). Este caso específico de variable de entrada/salida también se denomina variable VAR_IN_OUT.

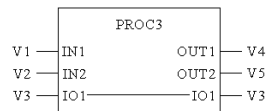
Tenga en cuenta las siguientes particularidades cuando utilice procedimientos con variables VAR_IN_OUT:

- Hay que asignar obligatoriamente una variable a todas las entradas VAR_IN_OUT.
- No se puede asignar ningún literal ni ninguna constante a las entradas VAR_IN_OUT.
- No es posible asignar **ningún** valor a las salidas VAR_IN_OUT.
- Las variables VAR_IN_OUT **no** se pueden utilizar fuera de la llamada de procedimientos.

Llamada de un procedimiento con variable VAR_IN_OUT en IL:

```
PROC3 (IN1:=V1, IN2:=V2, IO1:=V3,
      OUT1:=V4, OUT2:=V5) ;
```

Llamada del mismo procedimiento en FBD



Las variables VAR_IN_OUT **no** se pueden utilizar fuera de la llamada de procedimientos.

Por este motivo las siguientes llamadas de procedimientos son **inválidas**:

Llamada **inválida**, ejemplo 1:

LD V1	Carga de la variable V1 en el acumulador.
CAL InOutProc	Llamada de un procedimiento con parámetros VAR_IN_OUT. El acumulador está cargado ahora con la referencia a un parámetro VAR_IN_OUT.
AND V2	Operación AND del contenido del acumulador con la variable V2. Errores: No se puede realizar la operación porque no es posible acceder al parámetro VAR_IN_OUT (contenido del acumulador) fuera de la llamada de procedimientos.

Llamada **inválida**, ejemplo 2:

LD V1	Carga de la variable V1 en el acumulador.
AND InOutProc.inout	Operación AND del contenido del acumulador con la referencia a un parámetro VAR_IN_OUT. Error: No se puede realizar la operación porque no es posible acceder al parámetro VAR_IN_OUT fuera de la llamada de procedimientos.

Llamada **inválida**, ejemplo 3:

LD V1	Carga de la variable V1 en el acumulador.
InOutFB V2	Llamada del procedimiento con asignación del parámetro actual (V2) al parámetro VAR_IN_OUT. Errores: No se puede realizar la operación porque en esta forma de llamada de procedimientos, sólo el parámetro VAR_IN_OUT estaría disponible para otro uso en el acumulador.

Por el contrario, las siguientes llamadas de procedimientos son **válidas**:

Llamada **válida**, ejemplo 1:

CAL InOutProc (IN1:=V1,inout:=V2)	Llamada de un procedimiento con parámetro VAR_IN_OUT y asignación formal de los parámetros actuales dentro de la llamada de procedimientos.

Llamada **válida**, ejemplo 2:

InOutProc (IN1:=V1,inout:=V2)	Llamada de un procedimiento con parámetro VAR_IN_OUT y asignación formal de los parámetros actuales dentro de la llamada de procedimientos.

Llamada **válida**, ejemplo 3:

<code>CAL InOutProc (V1,V2)</code>	Llamada de un procedimiento con parámetro <code>VAR_IN_OUT</code> y asignación formal de los parámetros actuales dentro de la llamada de procedimientos.

Vista general

En este capítulo, se describe el lenguaje de programación de texto estructurado ST conforme a la norma CEI 61131.

Contenido de este capítulo

Este capítulo contiene las siguientes secciones:

Sección	Apartado	Página
15.1	Generalidades sobre el texto estructurado ST	538
15.2	Instrucciones	549
15.3	Llamada de funciones elementales, módulos de función elementales, módulos de función derivados y procedimientos	571

15.1 Generalidades sobre el texto estructurado ST

Vista general

En esta sección se ofrece una vista general sobre el texto Estructurado ST.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Generalidades sobre el texto estructurado (ST)	539
Operandos	542
Operadores	544

Generalidades sobre el texto estructurado (ST)

Introducción

El lenguaje de programación de texto estructurado (ST) permite, por ejemplo, llamar bloques de función, ejecutar funciones, efectuar asignaciones, ejecutar instrucciones de forma condicional y repetir instrucciones.

Expresión

El lenguaje de programación ST trabaja con las denominadas "expresiones".

Las expresiones son construcciones compuestas por operadores y operandos que devuelven un valor durante la ejecución.

Operador

Los operadores son símbolos para las operaciones que se van a ejecutar.

Operando

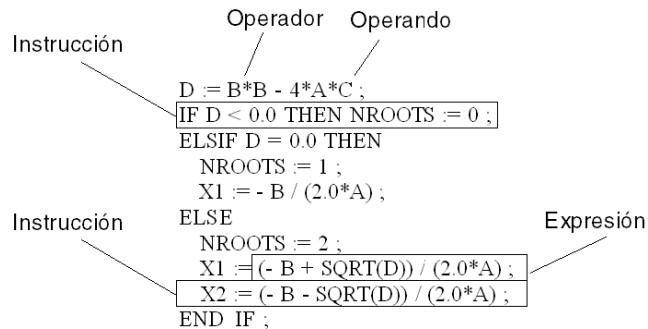
Los operadores se aplican a los operandos. Los operandos son, por ejemplo, variables, literales, salidas/entradas FFB, etc.

Instrucciones

Las instrucciones sirven para asignar a los parámetros actuales los valores devueltos por las expresiones y para estructurar y controlar las expresiones.

Representación de una sección ST

Representación de una sección ST:



Tamaño de la sección

La longitud de una línea de instrucciones está limitada a 300 caracteres.

La longitud de una sección ST no está limitada dentro del entorno de programación. La longitud de una sección ST sólo está limitada por el tamaño de la memoria del PLC.

Sintaxis

Para la indicación de los identificadores y de las palabras clave no se diferencia entre mayúsculas y minúsculas.

Excepción: no se admiten espacios en blanco ni tabuladores en:

- palabras clave
- Literales
- Valores
- Identificadores
- Variables y
- combinaciones de limitadores [por ejemplo, (* para comentarios)].

Secuencia de ejecución

La evaluación de una expresión está formada por la aplicación de los operadores sobre los operandos en el mismo orden en que se haya definido la jerarquía de los operadores (consulte "Tabla de operadores (*véase página 544*)"). El operador de mayor jerarquía en una expresión será el que se ejecute en primer lugar, a continuación, se ejecutará el operador de siguiente jerarquía y así sucesivamente hasta que se complete la evaluación. Los operadores del mismo rango se ejecutarán de izquierda a derecha, tal y como están escritos en la expresión. Esta secuencia se puede modificar mediante paréntesis.

Si, por ejemplo, A, B, C y D tienen los valores 1, 2, 3 y 4 y se calculan tal y como se indica a continuación:

$A+B-C*D$

entonces, el resultado será -9.

Si se realiza la operación:

$(A+B-C)*D$

entonces, el resultado será 0.

Si un operador posee dos operandos, primero se ejecutará el operando de la izquierda. Por ejemplo, en la expresión:

$SIN(A)*COS(B)$

se calculará primero la expresión $SIN(A)$, luego se calculará la expresión $COS(B)$ y, por último, se calculará el producto.

Comportamiento ante errores

Las siguientes condiciones se consideran errores durante la ejecución de una expresión:

- Intento de división entre 0.
- Los operandos no poseen el tipo de datos adecuado para la operación.
- El resultado de una operación numérica sobrepasa el rango de valores del tipo de datos.

Si se produce un error durante la ejecución de la operación, se activará el bit de sistema (%S) correspondiente (si el PLC utilizado lo admite).

Conformidad CEI

Para obtener una descripción de la conformidad del lenguaje de programación ST con la norma CEI, consulte "Conformidad CEI (*véase página 693*)".

Operandos

Introducción

Un operando puede ser:

- Una dirección
- Un literal
- Una variable
- Una variable de elementos múltiples
- Un elemento de una variable de elementos múltiples
- Una llamada de función
- Una salida FFB

Tipos de datos

Los tipos de datos de los operandos que se vayan a procesar en una instrucción deben ser idénticos. Si se procesan operandos de tipos de datos distintos, se debe realizar antes una conversión de tipos.

En el ejemplo siguiente, la variable entera `i1` se convierte en una variable real antes de sumarse con la variable real `r4`.

```
r3 := r4 + SIN(INT_TO_REAL(i1)) ;
```

Como excepción a esta regla, las variables del tipo de datos `TIME` se pueden multiplicar o dividir por las variables de los tipos de datos `INT`, `DINT`, `UINT` o `UDINT`.

Operaciones permitidas:

- `timeVar1 := timeVar2 / dintVar1;`
- `timeVar1 := timeVar2 * intVar1;`
- `timeVar := 10 * time#10s;`

La norma CEI 61131-3 clasifica esta función como efecto "no deseado".

Utilización directa de direcciones

Las direcciones se pueden utilizar directamente (sin una declaración previa). En este caso, el tipo de datos de la dirección se asigna directamente. Esta asignación se realiza mediante el "prefijo de magnitud".

En la tabla siguiente se indican los diversos prefijos de magnitud.

Prefijo de magnitud/símbolo	Ejemplo	Tipo de datos
Sin prefijo	<code>%I10</code> , <code>%CH203.MOD</code> , <code>%CH203.MOD.ERR</code>	BOOL
X	<code>%MX20</code>	BOOL
B	<code>%QB102.3</code>	BYTE
W	<code>%KW43</code>	INT

Prefijo de magnitud/símbolo	Ejemplo	Tipo de datos
D	%QD100	DINT
F	%MF100	REAL

Utilización de otros tipos de datos

Si desea asignar a una dirección otros tipos de datos distintos a los tipos de datos predeterminados, deberá hacerlo mediante una declaración explícita. Esta declaración de variables se puede realizar de forma cómoda a través del editor de variables. Los tipos de datos de una dirección no se pueden declarar directamente en una sección ST (p. ej. declaración AT %MW1: UINT; no permitida).

Las siguientes variables están declaradas en el editor de variables:

```
UnlocV1: ARRAY [1..10] OF INT;  
LocV1:   ARRAY [1..10] OF INT AT %MW100;  
LocV2:   TIME AT %MW100;
```

Las siguientes llamadas son sintácticamente correctas:

```
%MW200 := 5;  
UnlocV1[2] := LocV1[%MW200];  
LocV2      := t#3s;
```

Acceso a variables de campo

Para el acceso a variables de campo (ARRAY), en la indicación de índice sólo se admiten literales y variables de los tipos de datos INT, UINT, DINT y UDINT.

El índice de un elemento ARRAY puede ser negativo si el límite inferior del rango es negativo.

Ejemplo: Utilización de variables de campo

```
var1[i] := 8 ;  
var2.otto[4] := var3 ;  
var4[1+i+j*5] := 4 ;
```

Operadores

Introducción

Un operador es un símbolo para:

- Una operación aritmética ejecutable
- Una operación lógica ejecutable
- Un procesamiento de función (llamada)

Los operadores son genéricos, es decir, se adaptan automáticamente al tipo de datos de los operandos.

Tabla de operadores

Los operadores se ejecutan según su jerarquía, consulte también "*Secuencia de ejecución, página 540*".

Operadores del lenguaje de programación ST

Operador	Significado	Jerarquía	Operandos posibles	Descripción
()	Paréntesis	1 (mayor)	Expresión	Los paréntesis se utilizan para modificar la secuencia de ejecución de los operadores. Ejemplo: Si los operandos A, B, C y D tienen los valores 1, 2, 3 y 4 respectivamente, entonces $A+B-C*D$ tiene como resultado -9. Pero, $(A+B-C)*D$ tiene como resultado 0.
FUNCNAME (lista de parámetros reales)	Procesamiento de función (llamada)	2	Expresión, literal, variable, dirección (todos los tipos de datos)	El procesamiento de función se utiliza para ejecutar funciones (consulte " <i>Llamada de funciones elementales, página 572</i> ").
-	Negación	3	Expresión, literal, variable, dirección de los tipos de datos INT, DINT, UINT, UDINT o REAL	La negación - origina una inversión del signo para el valor del operando. Ejemplo: En el ejemplo, OUT es -4 si IN1 es 4. $OUT := - IN1 ;$
NOT	Complemento	3	Expresión, literal, variable, dirección de los tipos de datos BOOL, BYTE, WORD o DWORD	NOT provoca una inversión del operando por bits. Ejemplo: En el ejemplo, OUT es 0011001100 si IN1 es 1100110011. $OUT := NOT IN1 ;$

Operador	Significado	Jerarquía	Operandos posibles	Descripción
**	Potenciación	4	Expresión, literal, variable, dirección de los tipos de datos REAL (base) e INT, DINT, UINT, UDINT o REAL (exponente)	En la potenciación **, el valor del primer operando (base) se eleva a la potencia del valor del segundo operando (exponente). Ejemplo: En el ejemplo, OUT es 625,0 si IN1 es 5,0 e IN2 es 4,0. OUT := IN1 ** IN2 ;
*	Multiplicación	5	Expresión, literal, variable, dirección de los tipos de datos INT, DINT, UINT, UDINT o REAL	En la multiplicación *, el valor del primer operando se multiplica por el valor del segundo operando. Ejemplo: En el ejemplo, OUT es 20,0 si IN1 es 5,0 e IN2 es 4,0. OUT := IN1 * IN2 ; Nota: Para las multiplicaciones con el tipo de datos Time está disponible la función MULTIME de la biblioteca obsoleta.
/	División	5	Expresión, literal, variable, dirección de los tipos de datos INT, DINT, UINT, UDINT o REAL	En la división /, el valor del primer operando se divide entre el valor del segundo operando. Ejemplo: En el ejemplo, OUT es 4,0 si IN1 es 20,0 e IN2 es 5,0. OUT := IN1 / IN2 ; Nota: Para las divisiones con el tipo de datos Time está disponible la función DIVTIME de la biblioteca obsoleta.
MOD	Módulo	5	Expresión, literal, variable, dirección de los tipos de datos INT, DINT, UINT o UDINT	En MOD, el valor del primer operando se divide entre el valor del segundo operando, y el resto de la división (módulo) se emite como resultado. Ejemplo: En el siguiente ejemplo: ● OUT será 1 cuando IN1 sea 7 y B sea 2. ● OUT será 1 cuando IN1 sea 7 y B sea -2. ● OUT será -1 cuando IN1 sea -7 y B sea 2. ● OUT será -1 cuando IN1 sea -7 y B sea -2. OUT := IN1 MOD IN2 ;
+	Adición	6	Expresión, literal, variable, dirección de los tipos de datos INT, DINT, UINT, UDINT, REAL o TIME	En la adición +, el valor del primer operando se suma al valor del segundo operando. Ejemplo: En el siguiente ejemplo: OUT será 9 si IN1 es 7 e IN2 es 2. OUT := IN1 + IN2 ;

Operador	Significado	Jerarquía	Operandos posibles	Descripción
-	Sustracción	6	Expresión, literal, variable, dirección de los tipos de datos INT, DINT, UINT, UDINT, REAL o TIME	En la sustracción -, el valor del segundo operando se resta al del primer operando. Ejemplo: En el ejemplo, OUT es 6 si IN1 es 10 e IN2 es 4. OUT := IN1 - IN2 ;
<	Menor que	7	Expresión, literal, variable, dirección de los tipos de datos BOOL, BYTE, INT, DINT, UINT, UDINT, REAL, TIME, WORD, DWORD, STRING, DT, DATE o TOD	Con la comparación < se compara el valor del primer operando con el valor del segundo operando. Si el valor del primer operando es menor que el valor del segundo, el resultado será un 1 booleano. Si el valor del primer operando es mayor o igual que el valor del segundo, el resultado será un 0 booleano. Ejemplo: En el ejemplo, OUT es 1 si IN1 es menor que 10; de lo contrario, será 0. OUT := IN1 < 10 ;
>	Mayor que	7	Expresión, literal, variable, dirección de los tipos de datos BOOL, BYTE, INT, DINT, UINT, UDINT, REAL, TIME, WORD, DWORD, STRING, DT, DATE o TOD	Con la comparación > se compara el valor del primer operando con el valor del segundo operando. Si el valor del primer operando es mayor que el valor del segundo, el resultado será un 1 booleano. Si el valor del primer operando es menor o igual que el valor del segundo, el resultado será un 0 booleano. Ejemplo: En el ejemplo, OUT es 1 si IN1 es mayor que 10, y si IN1 es menor que 0. OUT := IN1 > 10 ;
<=	Menor o igual que	7	Expresión, literal, variable, dirección de los tipos de datos BOOL, BYTE, INT, DINT, UINT, UDINT, REAL, TIME, WORD, DWORD, STRING, DT, DATE o TOD	Con la comparación <= se compara el valor del primer operando con el valor del segundo operando. Si el valor del primer operando es menor o igual que el valor del segundo, el resultado será un 1 booleano. Si el valor del primer operando es mayor que el valor del segundo, el resultado será un 0 booleano. Ejemplo: En el ejemplo, OUT es 1 si IN1 es menor o igual que 10; de lo contrario, será 0. OUT := IN1 <= 10 ;

Operador	Significado	Jerarquía	Operandos posibles	Descripción
>=	Mayor o igual que	7	Expresión, literal, variable, dirección de los tipos de datos BOOL, BYTE, INT, DINT, UINT, UDINT, REAL, TIME, WORD, DWORD, STRING, DT, DATE o TOD	Con la comparación >= se compara el valor del primer operando con el valor del segundo operando. Si el valor del primer operando es mayor o igual que el valor del segundo, el resultado será un 1 booleano. Si el valor del primer operando es menor que el valor del segundo, el resultado será un 0 booleano. Ejemplo: En el ejemplo, OUT es 1 si IN1 es mayor o igual que 10; en caso contrario, será 0. OUT := IN1 >= 10 ;
=	Igualdad	8	Expresión, literal, variable, dirección de los tipos de datos BOOL, BYTE, INT, DINT, UINT, UDINT, REAL, TIME, WORD, DWORD, STRING, DT, DATE o TOD	Con la comparación = se compara el valor del primer operando con el valor del segundo operando. Si el valor del primer operando es igual que el valor del segundo, el resultado será un 1 booleano. Si el valor del primer operando no es igual al valor del segundo, el resultado será un 0 booleano. Ejemplo: En el ejemplo, OUT es 1 si IN1 es igual a 10; de lo contrario, será 0. OUT := IN1 = 10 ;
<>	Desigualdad	8	Expresión, literal, variable, dirección de los tipos de datos BOOL, BYTE, INT, DINT, UINT, UDINT, REAL, TIME, WORD, DWORD, STRING, DT, DATE o TOD	Con la comparación <> se compara el valor del primer operando con el valor del segundo operando. Si el valor del primer operando no es igual al valor del segundo, el resultado será un 1 booleano. Si el valor del primer operando es igual que el valor del segundo, el resultado será un 0 booleano. Ejemplo: En el ejemplo, OUT es 1 si IN1 no es igual a 10; de lo contrario, será 0. OUT := IN1 <> 10 ;
&	AND lógico	9	Expresión, literal, variable, dirección de los tipos de datos BOOL, BYTE, WORD o DWORD	En el caso de & tiene lugar una operación AND lógica entre los operandos. Con los tipos de datos BYTE, WORD y DWORD, esta operación se lleva a cabo por bits. Ejemplo: En los ejemplos, OUT es 1 si IN1, IN2 e IN3 son 1. OUT := IN1 & IN2 & IN3 ;

Operador	Significado	Jerarquía	Operandos posibles	Descripción
AND	AND lógico	9	Expresión, literal, variable, dirección de los tipos de datos BOOL, BYTE, WORD o DWORD	En el caso de AND, tiene lugar una conexión de AND lógica entre los operandos. Con los tipos de datos BYTE, WORD y DWORD, esta operación se lleva a cabo por bits. Ejemplo: En los ejemplos, OUT es 1 si IN1, IN2 e IN3 son 1. OUT := IN1 AND IN2 AND IN3 ;
XOR	OR exclusivo lógico	10	Expresión, literal, variable, dirección de los tipos de datos BOOL, BYTE, WORD o DWORD	En el caso de XOR tiene lugar una operación OR exclusiva lógica entre los operandos. Con los tipos de datos BYTE, WORD y DWORD, esta operación se lleva a cabo por bits. Ejemplo: En el ejemplo, OUT será 1 si IN1 e IN2 son distintos. Si IN1 e IN2 tienen el mismo estado (ambos son 0 ó 1), OUT será 0. OUT := IN1 XOR IN2 ; Si se vinculan más de dos operandos, el resultado será 1 en caso de que haya una cantidad impar de estados 1, y 0 si hay una cantidad par de estados 1. Ejemplo: En el ejemplo, OUT es 1 si 1 ó 3 operandos son 1. OUT es 0 si 0, 2 ó 4 operandos son 1. OUT := IN1 XOR IN2 XOR IN3 XOR IN4 ;
OR	OR lógico	11 (menor)	Expresión, literal, variable, dirección de los tipos de datos BOOL, BYTE, WORD o DWORD	En el caso de OR tiene lugar una operación OR lógica entre los operandos. Con los tipos de datos BYTE, WORD y DWORD, esta operación se lleva a cabo por bits. Ejemplo: En el ejemplo, OUT es 1 si IN1, IN2 o IN3 es 1. OUT := IN1 OR IN2 OR IN3 ;

15.2 Instrucciones

Vista general

En esta sección, se describen las instrucciones del lenguaje de programación de texto estructurado ST.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Instrucciones	550
Asignación	551
Instrucción de selección IF...THEN...END_IF	554
Instrucción de selección ELSE	556
Instrucción de selección ELSIF...THEN	557
Instrucción de selección CASE...OF...END_CASE	559
Instrucción de repetición FOR...TO...BY...DO...END_FOR	560
Repetición de la instrucción WHILE...DO...END_WHILE	563
Instrucción de repetición REPEAT...UNTIL...END_REPEAT	564
Instrucción de repetición EXIT	565
Llamada de subrutina	566
RETURN	567
Instrucción vacía	568
Etiquetas y saltos	569
Comentario	570

Instrucciones

Descripción

Las instrucciones son los "comandos" del lenguaje de programación ST.
Las instrucciones se deben cerrar mediante el símbolo del punto y coma.
En una línea puede haber varias instrucciones (separadas por punto y coma).
Un punto y coma solo representa una instrucción vacía (*véase página 568*).

Asignación

Introducción

La asignación reemplaza el valor actual de una variable de elemento único o de elementos múltiples por el resultado de la evaluación de una expresión.

Una asignación está compuesta por una especificación de variables en la parte izquierda, seguida de un operador de asignación `:=`, seguido de la expresión que se va a evaluar.

Ambas variables (parte izquierda y derecha del operador de asignación) deben tener el mismo tipo de datos.

Las matrices (arrays) constituyen un caso especial. Si se habilita explícitamente, es posible asignar dos matrices de longitudes distintas.

Asignación del valor de una variable a otra variable

Las asignaciones se utilizan para asignar el valor de una variable a otra variable.

La instrucción

```
A := B ;
```

se utiliza, por ejemplo, para reemplazar el valor de la variable `A` por el valor actual de la variable `B`. Si `A` y `B` presentan un tipo de datos elemental, el valor único de `B` se transferirá a `A`. Si `A` y `B` presentan un tipo de datos derivado, los valores de todos los elementos de `B` se transferirán a `A`.

Asignación del valor de un literal a una variable

Las asignaciones se utilizan para asignar un literal a una variable.

La instrucción

```
C := 25 ;
```

se utiliza, por ejemplo, para asignar el valor 25 a la variable `C`.

Asignación del valor de una operación a una variable

Las asignaciones se utilizan para asignar a una variable un valor que es el resultado de una operación.

La instrucción

```
X := (A+B-C) * D ;
```

se utiliza, por ejemplo, para asignar a la variable `X` el resultado de la operación $(A+B-C) * D$.

Asignación del valor de un FFB a una variable

Las asignaciones se utilizan para asignar a una variable un valor proporcionado por una función o por un bloque de función.

La instrucción

```
B := MOD(C,A) ;
```

se utiliza, por ejemplo, para llamar la función MOD (módulo) y asignar el resultado del cálculo a la variable B.

La instrucción

```
A := MY_TON.Q ;
```

se utiliza, por ejemplo, para asignar a la variable A el valor de la salida Q del bloque de función MY_TON (instancia del bloque de función TON). (No se trata de una llamada de bloque de función.)

Asignaciones múltiples

Las asignaciones múltiples son una ampliación de la norma CEI 61131-3 y se deben habilitar de forma explícita.

Aunque estén habilitadas, las asignaciones múltiples NO están permitidas en los siguientes casos:

- En la lista de parámetros de una llamada de bloque de función
- En la lista de elementos para inicializar variables estructuradas

La instrucción

```
X := Y := Z
```

está permitida.

Las instrucciones

```
FB(in1 := 1, In2 := In3 := 2) ;
```

y

```
strucVar := (comp1 := 1, comp2 := comp3 := 2) ;
```

no están permitidas.

Asignaciones entre matrices y variables WORD-/DWORD

Las asignaciones entre matrices y variables WORD/DWORD sólo son posibles si antes se ha efectuado una conversión de tipo, por ejemplo:

```
%Q3.0:16 := INT_TO_AR_BOOL(%MW20) ;
```

Están disponibles las siguientes funciones de conversión (biblioteca general, familia Array):

- MOVE_BOOL_AREBOOL
- MOVE_WORD_ARWORD
- MOVE_DWORD_ARDWORD

- MOVE_INT_ARINT
- MOVE_DINT_ARDINT
- MOVE_REAL_ARREAL

Instrucción de selección IF...THEN...END_IF

Descripción

La instrucción IF determina que una instrucción o un grupo de instrucciones se ejecute sólo si la expresión booleana correspondiente tiene el valor 1 (verdadero). Si la condición es 0 (falso), la instrucción o el grupo de instrucciones no se ejecutará.

La instrucción THEN marca el final de la condición y el principio de la instrucción o instrucciones.

La instrucción END_IF señala el final de la instrucción o instrucciones.

NOTA: Se puede intercalar una cantidad cualquiera de instrucciones IF...THEN...END_IF para generar instrucciones de selección complejas.

Ejemplo de IF...THEN...END_IF

La condición se puede expresar mediante una variable booleana.

Si FLAG es 1, se ejecutan las instrucciones; si FLAG es 0, no se ejecutan las instrucciones.

```
IF FLAG THEN
    C:=SIN(A) * COS(B) ;
    B:=C - A ;
END_IF ;
```

La condición también se puede expresar mediante una operación de la que se obtenga un resultado booleano.

Si A es mayor que B, se ejecutan las instrucciones; si A es menor o igual que B, no se ejecutan las instrucciones.

```
IF A>B THEN
    C:=SIN(A) * COS(B) ;
    B:=C - A ;
END_IF ;
```

Ejemplo de IF NOT...THEN...END_IF

Con NOT se puede invertir la condición (ejecución de ambas instrucciones si 0).

```
IF NOT FLAG THEN
    C:=SIN_REAL(A) * COS_REAL(B) ;
    B:=C - A ;
END_IF ;
```

Consulte también

ELSE (véase página 556)

ELSIF (*véase página 557*)

Instrucción de selección ELSE

Descripción

La instrucción **ELSE** sigue siempre a una instrucción **IF . . . THEN**, **ELSIF . . . THEN** o **CASE**.

Si la instrucción **ELSE** sigue a **IF** o **ELSIF**, la instrucción o el grupo de instrucciones se ejecuta sólo cuando las expresiones booleanas correspondientes de la instrucción **IF** y **ELSIF** tienen el valor 0 (falso). Si la condición de la instrucción **IF** o **ELSIF** es 1 (verdadero), la instrucción o el grupo de instrucciones no se ejecuta.

Si la instrucción **ELSE** sigue a **CASE**, la instrucción o el grupo de instrucciones sólo se ejecuta cuando ninguna marca contiene el valor del selector. En el caso de que una marca contenga el valor del selector, la instrucción o el grupo de instrucciones no se ejecuta.

NOTA: Se puede intercalar una cantidad cualquiera de instrucciones **IF . . . THEN . . . ELSE . . . END_IF** para generar instrucciones de selección complejas.

Ejemplo de ELSE

```

IF A>B THEN
    C:=SIN(A) * COS(B) ;
    B:=C - A ;
ELSE
    C:=A + B ;
    B:=C * A ;
END_IF ;
    
```

Consulte también

IF (véase página 554)
ELSIF (véase página 557)
CASE (véase página 559)

Instrucción de selección ELSIF...THEN

Descripción

La instrucción **ELSIF** sigue siempre a una instrucción **IF . . . THEN**. La instrucción **ELSIF** determina que una instrucción o un grupo de instrucciones sólo se ejecuta si la expresión booleana correspondiente de la instrucción **IF** tiene el valor 0 (falso) y la expresión booleana correspondiente de la instrucción **ELSIF** tiene el valor 1 (verdadero). Si la condición de la instrucción **IF** es 1 (verdadero) o la condición de la instrucción **ELSIF** es 0 (falso), la instrucción o el grupo de instrucciones no se ejecuta.

La instrucción **THEN** marca el final de la condición o condiciones **ELSIF** y el principio de la instrucción o instrucciones.

NOTA: Se puede intercalar una cantidad cualquiera de instrucciones **IF . . . THEN . . . ELSIF . . . THEN . . . END_IF** para generar instrucciones de selección complejas.

Ejemplo de ELSIF . . . THEN

```
IF A>B THEN
    C:=SIN(A) * COS(B) ;
    B:=SUB(C,A) ;
ELSIF A=B THEN
    C:=ADD(A,B) ;
    B:=MUL(C,A) ;
END_IF ;
```

Ejemplo de instrucciones intercaladas

```
IF A>B THEN
    IF B=C THEN
        C:=SIN(A) * COS(B) ;
    ELSE
        B:=SUB(C,A) ;
    END_IF ;
ELSIF A=B THEN
    C:=ADD(A,B) ;
    B:=MUL(C,A) ;
ELSE
    C:=DIV(A,B) ;
END_IF ;
```

Consulte también

IF (véase página 554)

ELSE (*véase página 556*)

Instrucción de selección CASE...OF...END_CASE

Descripción

La instrucción **CASE** está compuesta por una expresión del tipo de datos **INT** (el "selector") y una lista de grupos de instrucciones. Cada grupo está provisto de una marca que está compuesta por uno o más números enteros (**INT**, **DINT**, **UINT**, **UDINT**) o rangos de valores de enteros. Se ejecuta el primer grupo de instrucciones cuya marca contenga el valor calculado del selector. De lo contrario, no se ejecuta ninguna de las instrucciones.

La instrucción **OF** señala el principio de las marcas.

Dentro de la instrucción **CASE** se puede incluir una instrucción **ELSE** cuyas instrucciones se ejecuten si ninguna marca contiene el valor del selector.

La instrucción **END_CASE** marca el final de la instrucción o instrucciones.

Ejemplo de CASE...OF...END_CASE

Ejemplo de CASE...OF...END_CASE

```

Selector
  |
  v
CASE SELECT OF
1, 5:  C:=SIN(A) * COS(B) ;
2:     B:=C - A ;
6..10: C:=C * A ;
ELSE
  B:=C * A ;
  C:=A / B ;
END_CASE ;
  
```

Consulte también

ELSE (véase página 556)

Instrucción de repetición FOR...TO...BY...DO...END_FOR

Descripción

La instrucción **FOR** se utiliza cuando se puede determinar de antemano la cantidad de repeticiones. De lo contrario, se utilizan las instrucciones **WHILE** (véase página 563) o **REPEAT** (véase página 564).

La instrucción **FOR** repite una secuencia de instrucciones hasta la instrucción **END_FOR**. La cantidad de repeticiones se determina mediante el valor inicial, el valor final y la variable de control.

La variable de control, el valor inicial y el valor final deben tener el mismo tipo de datos (**DINT** o **INT**).

La variable de control, el valor inicial y el valor final se pueden modificar mediante una de las instrucciones repetidas. Se trata de una ampliación de la norma CEI 61131-3.

La instrucción **FOR** incrementa el valor de las variables de control desde un valor inicial hasta un valor final. El valor de incremento tiene el valor predeterminado 1. Si desea utilizar otro valor, puede indicar explícitamente el valor del incremento (variable o constante). El valor de las variables de control se verifica antes de cada nuevo ciclo del bucle. Si éste se encuentra fuera del rango del valor inicial y el valor final, se abandonará el bucle.

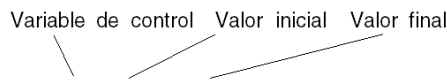
Antes del primer ciclo del bucle, se comprueba si el incremento de la variable de control se acerca al valor final partiendo del valor inicial. Si este no es el caso (p. ej. $\text{valor inicial} \leq \text{valor final}$ e incremento negativo), entonces no se procesa el bucle. Fuera del bucle, el valor de la variable de control es indefinido.

La instrucción **DO** marca el final de la definición de repetición y el principio de la instrucción o instrucciones.

La repetición se puede terminar antes de tiempo mediante la instrucción **EXIT**. La instrucción **END_FOR** marca el final de la instrucción o instrucciones.

Ejemplo: FOR con incremento 1

FOR con incremento 1



```

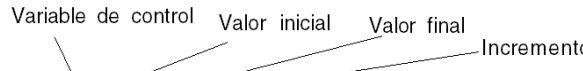
FOR i := 1 TO 50 DO
    C := C * COS(B) ;
END_FOR ;
  
```


FOR con incremento distinto a 1

Si desea utilizar un incremento distinto a 1, puede definirlo mediante BY. El incremento, el valor inicial, el valor final y la variable de control deben tener el mismo tipo de datos (DINT o INT). El signo de la expresión BY define el criterio de la dirección de procesamiento (progresivo, regresivo). Si esta expresión es positiva, entonces el bucle se ejecuta de forma progresiva; si es negativa, el bucle se ejecuta de forma regresiva.

Ejemplo: Conteo progresivo en dos pasos

Conteo progresivo en dos pasos



```
FOR i:= 1 TO 10 BY 2 DO (* BY > 0 : bucle progresivo *)
  C:= C * COS(B) ; (* la instrucción se ejecuta 5 veces *)
END_FOR ;
```

Ejemplo: Conteo regresivo

Conteo regresivo

```
FOR i:= 10 TO 1 BY -1 DO (* BY < 0 : Backwards.loop *)
  C:= C * COS(B) ; (* La instrucción se ejecuta 10 x *)
END_FOR ;
```

Ejemplo: Bucles "únicos"

Los bucles del ejemplo se ejecutan exactamente una sola vez ya que el valor inicial es igual que el valor final. En este caso no tiene importancia si el incremento es positivo o negativo.

```
FOR i:= 10 TO 10 DO (* Bucle único *)
  C:= C * COS(B) ;
END_FOR ;
```

O bien

```
FOR i:= 10 TO 10 BY -1 DO (* Bucle único *)
  C:= C * COS(B) ;
END_FOR ;
```

Ejemplo: Bucles críticos

Si en el ejemplo, el incremento j es > 0 , entonces se ejecutarán las instrucciones.

Si $j < 0$, no se ejecutarán las instrucciones, ya que el valor inicial de situación $<$ sólo admite un incremento del valor final de ≥ 0 .

Si $j = 0$, se ejecutarán las instrucciones y se producirá un bucle sin fin, ya que con un incremento de 0 jamás se alcanzará el valor final.

```
FOR i:= 1 TO 10 BY j DO  
    C:= C * COS (B) ;  
END_FOR ;
```

Repetición de la instrucción WHILE...DO...END_WHILE

Descripción

La instrucción `WHILE` provoca la ejecución repetitiva de una secuencia de instrucciones hasta que la expresión booleana correspondiente sea 0 (falso). Si la expresión es falsa desde el principio, el grupo de instrucciones no se ejecuta en absoluto.

La instrucción `DO` marca el final de la definición de repetición y el principio de la instrucción o instrucciones.

La repetición se puede terminar antes de tiempo mediante la instrucción `EXIT`.

La instrucción `END_WHILE` señala el final de la instrucción o instrucciones.

En los casos siguientes, `WHILE` no puede utilizarse, ya que puede crear un bucle infinito que conllevaría un bloqueo del programa:

- `WHILE` no se puede utilizar para efectuar una sincronización entre procesos; por ejemplo, como "bucle en espera" con una condición final externa determinada.
- `WHILE` puede no utilizarse en un algoritmo, ya que la finalización de la condición de final de bucle o la ejecución de una instrucción `EXIT` no se pueden garantizar.

Ejemplo de WHILE...DO...END_WHILE

```
x := 1; WHILE x <= 100 DO x := x + 4; END_WHILE ;
```

Consulte también

`EXIT` (véase página 565)

Instrucción de repetición REPEAT...UNTIL...END_REPEAT

Descripción

La instrucción `REPEAT` provoca la ejecución repetitiva de una secuencia de instrucciones (al menos una vez) hasta que la condición booleana correspondiente sea 1 (verdadero).

La instrucción `UNTIL` marca la condición final.

La repetición se puede terminar antes de tiempo mediante la instrucción `EXIT`.

La instrucción `END_REPEAT` marca el final de la instrucción o instrucciones.

En los casos siguientes, `REPEAT` podría no utilizarse, ya que puede crear un bucle sin fin que conllevaría un bloqueo del programa:

- `REPEAT` no se puede utilizar para efectuar una sincronización entre procesos, por ejemplo, como "bucle de espera" con una condición final externa determinada.
- `REPEAT` puede no utilizarse en un algoritmo, ya que la finalización de la condición de final de bucle o la ejecución de una instrucción `EXIT` no se pueden garantizar.

Ejemplo REPEAT...UNTIL...END_REPEAT

```
x := -1
REPEAT
    x := x + 2
UNTIL x >= 101
END_REPEAT ;
```

Consulte también

`EXIT` (véase *página 565*)

Instrucción de repetición EXIT

Descripción

La instrucción **EXIT** se emplea para finalizar instrucciones de repetición (**FOR**, **WHILE**, **REPEAT**) antes de que se dé la condición de final.

Si la instrucción **EXIT** se encuentra dentro de una repetición imbricada, se abandonará el bucle interno (en el que se encuentra **EXIT**). A continuación se ejecutará la primera instrucción después del final del bucle (**END_FOR**, **END_WHILE** o **END_REPEAT**).

Ejemplo de EXIT

Si **FLAG** tiene el valor 0, **SUM** será 15 después de la ejecución de las instrucciones.

Si **FLAG** tiene el valor 1, **SUM** será 6 después de la ejecución de las instrucciones.

```
SUM := 0 ;
FOR I := 1 TO 3 DO
  FOR J := 1 TO 2 DO
    IF FLAG=1 THEN EXIT;
  END_IF ;
  SUM := SUM + J ;
END_FOR ;
SUM := SUM + I ;
END_FOR
```

Consulte también

CASE (véase página 559)

WHILE (véase página 563)

REPEAT (véase página 564)

Llamada de subrutina

Llamada de subrutina

La llamada de una subrutina está compuesta por el nombre de la sección de la subrutina y una lista de parámetros vacía.

Las llamadas de subrutinas no devuelven ningún valor.

La subrutina invocante debe encontrarse en la misma tarea que la sección ST invocante.

También es posible llamar subrutinas ubicadas dentro de otras subrutinas.

P. ej.

```
nombre_de_la_subrutina () ;
```

Las llamadas de subrutina son una ampliación de la norma CEI 61131-3 y se deben habilitar de forma explícita.

En las secciones de acción SFC sólo se admiten llamadas de subrutina si está habilitada la modalidad Multi-Token.

RETURN

Descripción

Las instrucciones `RETURN` pueden usarse en DFB (bloques de funciones derivados) y en SR (subrutinas).

Las instrucciones `RETURN` no pueden usarse en el programa principal.

- En un DFB, una instrucción `RETURN` fuerza el retorno al programa que llamó al DFB.
 - El resto de la sección DFB que contiene la instrucción `RETURN` no se ejecuta.
 - Las siguientes secciones del DFB no se ejecutan.

El programa que llamó al DFB se ejecutará después de volver del DFB.

Si otro DFB llama al DFB, el DFB llamado se ejecutará después de volver.

- En un SR, una instrucción `RETURN` fuerza el retorno al programa que llamó al SR.
 - El resto de la sección SR que contiene la instrucción `RETURN` no se ejecuta.

El programa que llamó al SR se ejecutará después de volver del SR.

Instrucción vacía

Descripción

Un punto y coma solo ; representa una instrucción vacía.

P. ej.

```
IF x THEN ; ELSE ..
```

En este ejemplo, la instrucción que sigue a `THEN` es una instrucción vacía. Esto significa que, si la condición de `IF` es 1, el programa abandonará inmediatamente la instrucción `IF`.

Etiquetas y saltos

Introducción

Las etiquetas sirven como punto de destino de los saltos.

Los saltos y las etiquetas en ST son una ampliación de la norma CEI 61131-3 y se deben habilitar de forma explícita.

Propiedades de las etiquetas

Propiedades de las etiquetas:

- Las etiquetas deben ser siempre el primer elemento de una fila.
- Las etiquetas sólo pueden encontrarse delante de instrucciones de primer orden (no en bucles).
- Las etiquetas deben ser unívocas en toda la sección sin que se diferencie entre mayúsculas y minúsculas.
- Las etiquetas pueden tener una extensión máxima de 32 caracteres.
- Las etiquetas deben cumplir la nomenclatura general.
- Las etiquetas se separan con dos puntos : de las instrucciones siguientes.

Propiedades de los saltos

Propiedades de los saltos

- Los saltos se pueden realizar dentro de secciones de un programa y de un DFB.
- Los saltos sólo se pueden realizar dentro de la sección actual.

Ejemplo

```
IF var1 THEN
    JMP START;
:
:
START: ...
```

Comentario

Descripción

En el editor ST, los comentarios comienzan con la cadena de caracteres (* y terminan con *) . Entre estas dos cadenas se puede introducir cualquier comentario. Los comentarios se pueden introducir en cualquier posición en el editor ST excepto en palabras clave, literales, identificadores y variables.

Según la normativa CEI 61131-3, no se puede intercalar comentarios. Sin embargo, si éstos se intercalan, deben habilitarse de forma explícita.

15.3 Llamada de funciones elementales, módulos de función elementales, módulos de función derivados y procedimientos

Vista general

Llamada de funciones elementales, módulos de función elementales, módulos de función derivados y procedimientos en el lenguaje de programación ST.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Llamada de funciones elementales	572
Llamada de bloques de funciones elementales y bloques de funciones derivados	577
Procedimientos	586

Llamada de funciones elementales

Funciones elementales

Las funciones elementales están disponibles en forma de bibliotecas. La lógica de las funciones se define en el lenguaje de programación C y no se puede modificar en el editor ST.

Las funciones no tienen estados internos. Si los valores de las entradas son idénticos, el valor de salida será el mismo en todas las ejecuciones de la función. Por ejemplo, la suma de dos valores dará el mismo resultado en cada ejecución.

Ciertas funciones elementales se pueden ampliar a más de dos entradas.

Las funciones elementales tienen un único valor de retorno (salida).

Parámetros

Para transferir valores a una función o aplicarlos desde ella, es necesario utilizar varias entradas y una salida. A éstas se les llama parámetros formales.

Los estados de proceso actuales se transmiten a los parámetros formales. A éstos se les llama parámetros actuales.

Como parámetro actual para las entradas de la función se puede utilizar:

- Variable
 - Dirección
 - Literal
 - Expresión ST

Como parámetro actual para las salidas de la función se puede utilizar:

- Variable
- Dirección

El tipo de datos del parámetro actual debe coincidir con el tipo de datos del parámetro formal. Las únicas excepciones son los parámetros formales genéricos, cuyo tipo de datos está condicionado por el parámetro actual.

En el caso de los parámetros formales genéricos `ANY_BIT`, se pueden emplear parámetros reales de tipos de datos `INT` o `DINT` (pero no `UINT` ni `UDINT`).

Se trata de una ampliación de la norma CEI 61131-3 y se debe habilitar de forma explícita.

Ejemplo:

Se admite:

```
AND (AnyBitParam := IntVar1, AnyBitParam2 := IntVar2);
```

No se admite:

```
AND_WORD (WordParam1 := IntVar1, WordParam2 := IntVar2);
```

(En este caso se debe utilizar `AND_INT`).

```
AND_ARRAY_WORD (ArrayInt, ...);
```

(En este caso se debe realizar una conversión de tipo explícita a través de `INT_ARR_TO_WORD_ARR (...);`).

En principio, no es necesario asignar un valor a todos los parámetros formales. En la tabla encontrará los tipos de parámetros formales que sí lo requieren obligatoriamente.

Tipo de parámetro	EDT	STRING	ARRAY	ANY_ARRAY	IODDT	STRUCT	FB	ANY
Entrada	-	-	+	+	+	+	+	+
VAR_IN_OUT	+	+	+	+	+	+	/	+
Salida	-	-	-	-	-	-	/	-
+ Parámetro actual requerido obligatoriamente								
- Parámetro actual no requerido obligatoriamente								
/ No aplicable								

Si no se asigna un valor a un parámetro formal, durante la ejecución del bloque de función se utilizará el valor inicial. Si no se ha definido ningún valor inicial, se aplicará el valor predeterminado (0).

Notas sobre la programación

Tenga en cuenta las siguientes indicaciones sobre la programación:

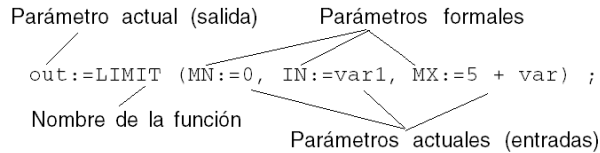
- Todas las funciones genéricas están cargadas. Es decir, las funciones se pueden llamar con o sin la indicación del tipo de datos.
P. ej.
`i1 := ADD (i2, 3);`
es idéntico a
`i1 := ADD_INT (i2, 3);`
- Las funciones se pueden intercalar (consulte también *Intercalar funciones, página 575*).
- Las funciones sólo se ejecutan si la entrada `EN` es 1 o si la entrada `EN` no se utiliza (consulte también "*EN y ENO, página 576*").
- Hay dos formas de llamar una función:
 - Llamada formal (llamada de una función con nombres de parámetros formales)
 - Llamada informal (llamada de una función sin nombres de parámetros formales)

Llamada formal

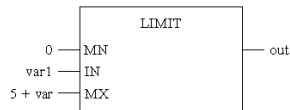
Con este tipo de llamada (llamada con nombres de parámetros formales), la llamada está compuesta por el parámetro actual de la salida seguido de la instrucción de asignación `:=`, el nombre de la función y una lista entre paréntesis con las asignaciones de valores (parámetros actuales) a los parámetros formales. La secuencia en la que se enumeran los parámetros formales en la llamada de función **no es significativa**.

Con este tipo de llamada se puede utilizar `EN` y `ENO`.

Llamada de una función con nombres de parámetros formales



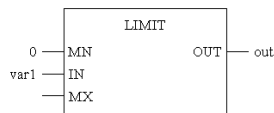
Llamada de la misma función en FBD



Con la llamada informal no es necesario asignar un valor a todos los parámetros formales (consulte también "*Parámetros, página 572*").

`out:=LIMIT (MN:=0, IN:=var1) ;`

Llamada de la misma función en FBD

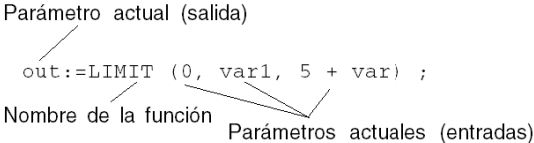


Llamada informal

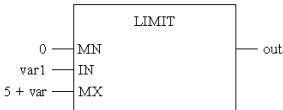
Con este tipo de llamada (llamada sin nombres de parámetros formales), la llamada está compuesta por el parámetro actual de la salida seguido del símbolo de la instrucción de asignación `:=`, el nombre de la función y una lista entre paréntesis de los parámetros actuales de las entradas. La secuencia en la que se enumeran los parámetros actuales en una llamada de función **es significativa**.

Con este tipo de llamada **no** se puede utilizar `EN` y `ENO`.

Llamada de una función sin nombres de parámetros formales



Llamada de la misma función en FBD



Con la llamada informal tampoco es necesario asignar un valor a todos los parámetros formales (consulte también "*Parámetros, página 572*").

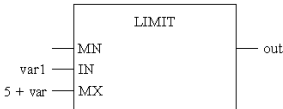
Se trata de una ampliación de la norma IEC 61131-3 y se debe habilitar de forma explícita.

Para excluir un parámetro se utiliza un campo de parámetros vacío.

Llamada con campo de parámetros vacío

`out:=LIMIT ( ,var1, 5 + var) ;`

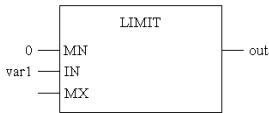
Llamada de la misma función en FBD



Si se suprimen los parámetros formales al final, no se tiene que utilizar un campo de parámetros vacío.

`out:=LIMIT (0, var1) ;`

Llamada de la misma función en FBD



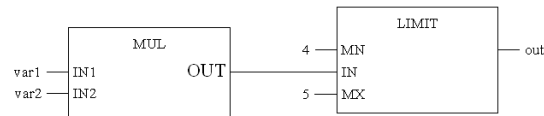
Intercalar funciones

La llamada de una función puede contener la llamada de otras funciones. La profundidad de intercalado no está limitada.

Llamada de una función intercalada

`out:=LIMIT (MN:=4, IN:=MUL(IN1:=var1, IN2:=var2), MX:=5) ;`

Llamada de la misma función en FBD



Las funciones que emiten valores del tipo de datos `ANY_ARRAY` **no** se pueden utilizar **dentro** de una llamada de función.

Intercalado **no admitido** con `ANY_ARRAY`:

```

      ANY_ARRAY
      /
out:=LIMIT (MN:=4, IN:=EXAMP(IN1:=var1, IN2:=var2), MX:=5) ;

```

Como valor de retorno de la función invocante o como parámetro de las funciones intercaladas se pueden utilizar los tipos de datos `ANY_ARRAY`.

Intercalado **admitido** con `ANY_ARRAY`:

```

ANY_ARRAY      ANY_ARRAY      ANY_ARRAY
/              /              /
out:=EXAMP(MN:=4, IN:=EXAMP(IN1:=var1, IN2:=var2), MX:=var3)

```

EN y ENO

En todas las funciones se puede configurar una entrada `EN` y una salida `ENO`.

Si el valor de `EN` es "0", al llamar la función no se ejecutarán los algoritmos definidos por dicha función, y `ENO` se ajustará a "0".

Si el valor de `EN` es "1", al llamar la función se ejecutarán los algoritmos definidos por dicha función. Si no hay problemas en la ejecución de estos algoritmos, el valor de `ENO` se ajustará a "1". Si se produce un error, `ENO` se ajustará a "0".

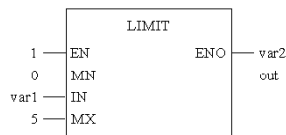
Si `ENO` está ajustado a "0" (porque `EN=0` o debido a un error en la ejecución), la salida de la función se ajustará a "0".

El comportamiento de salida de la función no depende de si dicha función se ha llamado sin `EN/ENO` o con `EN=1`.

Si va a utilizar `EN/ENO`, la llamada de la función deberá ser formal.

```
out:=LIMIT (EN:=1, MN:=0, IN:=var1, MX:=5, ENO=>var2) ;
```

Llamada de la misma función en FBD



Llamada de bloques de funciones elementales y bloques de funciones derivados

Bloque de función elemental

Los bloques de funciones elementales tienen estados internos. Si las entradas tienen los mismos valores, el valor de la salida puede ser otro durante las operaciones individuales. Por ejemplo, con un contador, el valor de la salida se incrementa.

Los bloques de funciones pueden disponer de varios valores de retorno (salidas).

Bloque de función derivado

Los bloques de funciones derivados (DFB) presentan las mismas propiedades que los bloques de funciones elementales. Sin embargo, el usuario los crea en los lenguajes de programación FBD, LD, IL o ST.

Parámetro

Para transferir valores a un bloque de función o aplicarlos desde él, es necesario utilizar entradas y salidas. A éstas se les llama parámetros formales.

Los estados de proceso actuales se transmiten a los parámetros formales. A éstos se les llama parámetros actuales.

Como parámetro actual para las entradas del bloque de función se puede utilizar:

- Variable
- Dirección
- Literal

Como parámetro actual para las salidas del bloque de función se puede utilizar:

- Variable
- Dirección

El tipo de datos del parámetro actual debe coincidir con el tipo de datos del parámetro formal. Las únicas excepciones son los parámetros formales genéricos, cuyo tipo de datos está condicionado por el parámetro actual.

En el caso de los parámetros formales genéricos del tipo de datos `ANY_BIT` se pueden utilizar parámetros reales de los tipos de datos `INT` y `DINT` (ni `UINT` ni `UDINT`).

Se trata de una ampliación de la norma CEI 61131-3 y se debe habilitar de forma explícita.

Ejemplo:

Se admite:

```
AND (AnyBitParam := IntVar1, AnyBitParam2 := IntVar2);
```

No se admite:

```
AND_WORD (WordParam1 := IntVar1, WordParam2 := IntVar2);
```

(En este caso se debe utilizar AND_INT).

```
AND_ARRAY_WORD (ArrayInt, ...);
```

(En este caso se debe realizar una conversión de tipo explícita a través de
INT_ARR_TO_WORD_ARR (...);.)

En principio, no es necesario asignar un valor a todos los parámetros formales. En la tabla encontrará los tipos de parámetros formales que sí lo requieren obligatoriamente.

Tipo de parámetro	EDT	STRING	ARRAY	ANY_ARRAY	IODDT	STRUCT	FB	ANY
EFB: entrada	-	+	+	+	/	+	/	+
EFB: VAR_IN_OUT	+	+	+	+	+	+	/	+
EFB: salida	-	-	+	+	+	-	/	+
DFB: entrada	-	+	+	+	/	+	/	+
DFB: VAR_IN_OUT	+	+	+	+	+	+	/	+
DFB: salida	-	-	+	/	/	-	/	+
+ Parámetro actual requerido obligatoriamente								
- Parámetro actual no requerido obligatoriamente								
/ No aplicable								

Si no se asigna un valor a un parámetro formal, durante la ejecución del bloque de función se utilizará el valor inicial. Si no se ha definido ningún valor inicial, se aplicará el valor predeterminado (0).

Si un parámetro formal no tiene asignado ningún valor y el bloque de función/DFB se ha instanciado varias veces, las instancias que se ejecuten a partir de ese momento trabajarán con el valor antiguo.

Variables públicas

Además de las entradas y salidas, algunos bloques de funciones también disponen de las denominadas variables públicas.

Estas variables sirven para transmitir valores estáticos (valores no influidos por el proceso) al bloque de función. Se utilizan para la parametrización del bloque de función.

Las Public Variables son una ampliación de la norma CEI 61131-3.

La asignación de valores a las Public Variables se realiza mediante sus valores iniciales o a través de asignaciones.

Ejemplo:

Nombre de instancia	Variable pública	
/	/	
D_ACT1.OP_CTRL := 1 ;		(D_ACT1 es una instancia del bloque de función D_ACT y dispone de las variables públicas AREA_NR y OP_CTRL).

La lectura de los valores de las Public Variables se realiza a través del nombre de instancia del bloque de función y a través del nombre de la Public Variable.

Ejemplo:

Nombre de instancia	Variable pública
/	/
Var1 := D_ACT1.OP_CTRL ;	

Variables privadas

Además de las entradas, las salidas y las variables públicas, algunos bloques de funciones también disponen de las denominadas variables privadas.

Al igual que las variables públicas, las privadas se utilizan para transferir valores estadísticos (valores no influidos por el proceso) al bloque de función.

El programa usuario no puede acceder a las variables privadas. Sólo se puede acceder a este tipo de variables mediante la tabla de animación.

NOTA: Los DFB intercalados se declaran como variables privadas del DFB principal. Por tanto, tampoco se puede acceder a sus variables a través de la programación, sino a través de la tabla de animación.

Las variables privadas constituyen un suplemento a la normativa CEI 61131-3.

Notas sobre la programación

Tenga en cuenta las siguientes indicaciones sobre la programación:

- Los bloques de funciones sólo se ejecutan si la entrada **EN** es 1 o si la entrada **EN** no se utiliza (consulte también "*EN y ENO, página 584*").
- La asignación de variables a salidas del tipo **ANY** o **ARRAY** se debe realizar mediante el **=>** operador (consulte también "*Llamada formal, página 580*"). No es posible realizar una asignación fuera de la llamada de bloque de función.

La instrucción

```
My_Var := My_SAH.OUT;
```

es **inválida**, la salida **OUT** del bloque de función **SAH** es del tipo **ANY**.

La instrucción

```
Cal My_SAH (OUT=>My_Var);
```

es, por el contrario, **válida**.

- Si se utilizan variables **VAR_IN_OUT** (véase *página 584*), se imponen condiciones especiales.
- La utilización de los bloques de funciones en ST consta de dos partes:

- Declaración (véase página 580)
- Llamada del bloque de función
- Hay dos formas de llamar un bloque de función:
 - Llamada formal (véase página 580) (llamar a una función con nombres de parámetros formales)
En este caso es posible asignar variables a las salidas mediante el operador `=>`.
 - Llamada informal (véase página 581) (llamar sin nombres de parámetros formales)
- Las instancias de bloques de funciones/DBF se pueden ejecutar varias veces, excepto las instancias de EFB de comunicaciones que sólo se pueden ejecutar una única vez (consulte "Llamada múltiple de una instancia de bloque de función, página 583").

Declaración

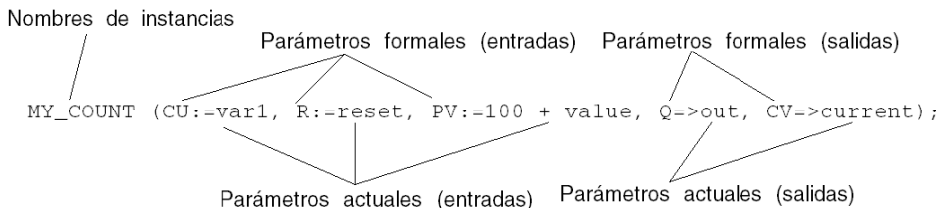
Antes de llamar un bloque de función es necesario declararlo primero en el editor de variables.

Llamada formal

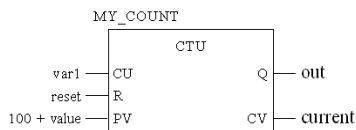
Con la llamada formal (llamada con nombres de parámetros formales), los bloques de funciones se llaman por medio de una instrucción formada por el nombre de instancia del bloque de funciones, seguido por una lista entre paréntesis de asignaciones de parámetros actuales a los parámetros formales. Asigne los parámetros formales de entrada mediante la asignación `:=`, y la asignación de los parámetros formales de salida, mediante la asignación `=>`. La secuencia en la que se enumeran los parámetros formales de entrada y de salida **no es significativa**.

Con este tipo de llamada se puede utilizar `EN` y `ENO`.

Llamada de un bloque de funciones con nombres de parámetros formales:



Llamada del mismo bloque de función en FBD:



La asignación del valor de una salida de bloque de función se realiza mediante la introducción del nombre del parámetro real, seguido de la instrucción de asignación `:= y`, después, del nombre de la instancia del bloque de función, así como mediante la carga del parámetro formal de la salida del bloque de función (separado por un punto).

P. ej.

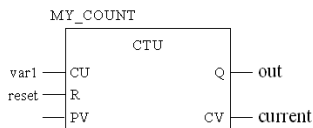
```
MY_COUNT (CU:=var1, R:=reset, PV:=100 + value);
Q := MY_COUNT.out ;
CV := MY_COUNT.current ;
```

NOTA: Los DDT de matriz de tipo no se pueden asignar de este modo. Sin embargo, se pueden asignar los DDT de estructura de tipo.

No es necesario asignar un valor a todos los parámetros formales (consulte también "*Parámetro, página 577*").

```
MY_COUNT (CU:=var1, R:=reset, Q=>out, CV=>current);
```

Llamada del mismo bloque de función en FBD

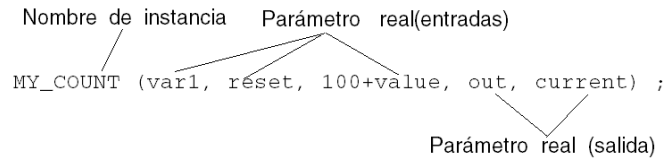


Llamada informal

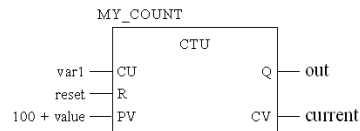
Con la llamada informal (llamada sin nombres de parámetros formales), los bloques de funciones se llaman por medio de una instrucción formada por el nombre de instancia del bloque de funciones, seguido por una lista entre paréntesis de parámetros actuales de las entradas y salidas. La secuencia en la que se enumeran los parámetros actuales en una llamada de bloque de función **es significativa**.

Con este tipo de llamada **no** es posible utilizar **EN** y **ENO**.

Llamada de un bloque de función sin parámetros formales:



Llamada del mismo bloque de función en FBD



Con la llamada informal tampoco es necesario asignar un valor a todos los parámetros formales (consulte también "*Parámetro, página 577*"). Esto no se aplica a las variables VAR_IN_OUT, para los parámetros de entrada con longitudes dinámicas y salidas del tipo ANY. Siempre se le debe asignar una variable.

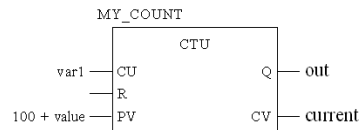
Se trata de una ampliación de la norma CEI 61131-3 y se debe habilitar de forma explícita.

Para excluir un parámetro se utiliza un campo de parámetros vacío.

Llamada con campo de parámetros vacío:

MY_COUNT (var1, , 100 + value, out, current) ;

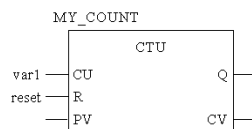
Llamada del mismo bloque de función en FBD



Si se suprimen los parámetros formales al final, no se tiene que utilizar un campo de parámetros vacío.

MY_COUNT (var1, reset) ;

Llamada del mismo bloque de función en FBD



Llamada de un bloque de función sin entradas

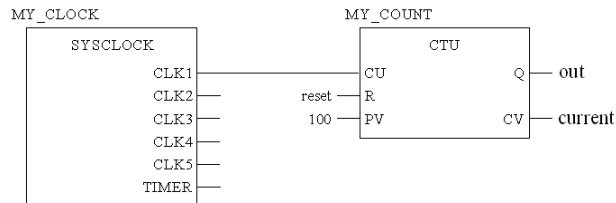
Aunque el bloque de función no tenga ninguna entrada o no sea necesario parametrizar sus entradas, hay que llamar el bloque de función antes de poder utilizar sus salidas. De lo contrario, se transmitirán los valores iniciales de las salidas, es decir "0".

P. ej.

Llamada de los bloques de funciones en ST:

```
MY_CLOCK ( ) ;
MY_COUNT (CU:=MY_CLOCK.CLK1, R:=reset, PV:=100,
           Q=>out, CV=>current) ;
```

Llamada de los mismos bloques de funciones en FBD



Llamada múltiple de una instancia de bloque de función

Las instancias de bloques de funciones/DBF se pueden llamar varias veces, excepto las instancias de EFB de comunicaciones, que sólo se pueden llamar una única vez.

La llamada múltiple de una misma instancia de bloque de función/DFB resulta conveniente en los siguientes casos:

- Cuando el bloque de función/DFB no posee ningún valor interno o los valores internos no son necesarios para continuar el procesamiento.
En este caso, la llamada múltiple de una misma instancia de bloque de función/DFB permite ahorrar espacio en memoria, ya que el código del bloque de función/DFB sólo se carga una única vez.
El bloque de función/DFB se procesa, por así decirlo, como una función.
- Cuando el bloque de función/DFB tiene valores internos y éstos se deben modificar en varios puntos del programa, por ejemplo, si el valor de un contador se debe aumentar en diversos puntos del programa.
En este caso, con la llamada múltiple de la misma instancia de bloque de función/DFB no es necesario guardar los resultados intermedios para continuar el procesamiento en otro lugar del programa.

EN y ENO

En todos los bloques de funciones/DFB se puede configurar una entrada **EN** y una salida **ENO**.

Si el valor de **EN** es "0", al llamar el bloque de función/DFB no se ejecutarán los algoritmos definidos por dicho bloque de función/DFB, y **ENO** se ajustará a "0".

Si el valor de **EN** es "1", al llamar el bloque de función/DFB se ejecutarán los algoritmos definidos por dicho bloque de función/DFB. Si no hay problemas en la ejecución de estos algoritmos, el valor de **ENO** será "1". Si se produce un error, **ENO** será "0".

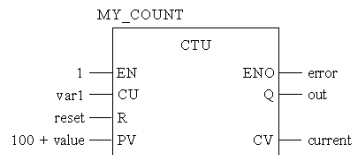
Si **ENO** se ajusta a "0" (debido a que **EN** es igual a 0 o a un error producido durante la ejecución), las salidas del bloque de función/DFB conservan el estado que tenían en el último ciclo ejecutado correctamente.

El comportamiento de salida de los bloques de funciones/DFB no depende de si dichos bloques de funciones/DFB se han llamado sin **EN/ENO** o con **EN=1**.

Si se va a utilizar **EN/ENO**, la llamada del bloque de función deberá ser formal. La asignación de una variable a **ENO** debe realizarse con el operador **=>**.

```
MY_COUNT (EN:=1, CU:=var1, R:=reset, PV:=100 + value,
          ENO=>error, Q=>out, CV=>current) ;
```

Llamada del mismo bloque de función en FBD



Variable VAR_IN_OUT

A menudo, los bloques de funciones se utilizan para leer una variable en la entrada (variables de entrada), procesarla y volver a emitir los valores modificados de la misma variable (variables de salida). Este caso específico de variable de entrada/salida también se denomina variable **VAR_IN_OUT**.

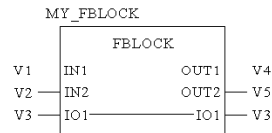
Tenga en cuenta las siguientes particularidades al utilizar bloques de funciones/DFB con variables **VAR_IN_OUT**:

- Hay que asignar obligatoriamente una variable a todas las entradas **VAR_IN_OUT**.
- No se puede asignar ningún literal ni ninguna constante a las entradas **VAR_IN_OUT**.
- No es posible asignar **ningún** valor a las salidas **VAR_IN_OUT**.
- Las variables **VAR_IN_OUT** **no** se pueden utilizar fuera de la llamada de un bloque de función.

Llamada de un bloque de función con variable **VAR_IN_OUT** en ST:


```
MY_FBLOCK(IN1:=V1, IN2:=V2, IO1:=V3, OUT1=>V4, OUT2=>V5);
```

Llamada del mismo bloque de función en FBD:



Las variables `VAR_IN_OUT` **no** se pueden utilizar fuera de la llamada de un bloque de función.

Por este motivo, las siguientes llamadas de bloque de función son **inválidas**:

Llamada **inválida**, ejemplo 1.

<code>InOutFB.inout := V1;</code>	Asignación de las variables <code>V1</code> a un parámetro <code>VAR_IN_OUT</code> . Errores: No se puede realizar la operación porque no es posible acceder al parámetro <code>VAR_IN_OUT</code> fuera de la llamada de bloque de función.

Llamada **inválida**, ejemplo 2.

<code>V1 := InOutFB.inout;</code>	Asignación de un parámetro <code>VAR_IN_OUT</code> a la variable <code>V1</code> . Errores: No se puede realizar la operación porque no es posible acceder al parámetro <code>VAR_IN_OUT</code> fuera de la llamada de bloque de función.

Por el contrario, las siguientes llamadas del bloque de función son **válidas**:

Llamada **válida**, ejemplo 1.

<code>InOutFB (inout:=V1);</code>	Llamada de un bloque de función con parámetro <code>VAR_IN_OUT</code> y con asignación formal de los parámetros actuales dentro de la llamada del bloque de función.

Llamada **válida**, ejemplo 2.

<code>InOutFB (V1);</code>	Llamada de un bloque de función con parámetro <code>VAR_IN_OUT</code> y con asignación informal de los parámetros actuales dentro de la llamada del bloque de función.

Procedimientos

Procedimiento

Los procedimientos están disponibles en forma de bibliotecas. La lógica de los procedimientos se define en el lenguaje de programación C y no se puede modificar en el editor ST.

Los procedimientos, al igual que las funciones, no tienen estados internos. Si los valores de las entradas son idénticos, el valor de la salida será el mismo en todas las ejecuciones del procedimiento. Por ejemplo, la suma de dos valores dará el mismo resultado en cada ejecución.

Al contrario de lo que sucede con las funciones, los procedimientos no emiten valores de retorno y admiten variables `VAR_IN_OUT`.

Los procedimientos son una ampliación de la norma IEC 61131-3 y se deben habilitar de forma explícita.

Parámetro

Para transferir valores a un procedimiento o aplicarlos desde él, es necesario utilizar entradas y salidas. A éstas se les llama parámetros formales.

Los estados de proceso actuales se transmiten a los parámetros formales. A éstos se les llama parámetros actuales.

Como parámetro actual para las entradas de un procedimiento se puede utilizar:

- Variable
- Dirección
- Literal
- Expresión ST

Como parámetro actual para las salidas de un procedimiento se puede utilizar:

- Variable
- Dirección

El tipo de datos del parámetro actual debe coincidir con el tipo de datos del parámetro formal. Las únicas excepciones son los parámetros formales genéricos, cuyo tipo de datos está condicionado por el parámetro actual.

En el caso de los parámetros formales genéricos `ANY_BIT`, se pueden emplear los parámetros reales de tipos de datos `INT` o `DINT` (pero no `UINT` ni `UDINT`).

Se trata de una ampliación de la norma CEI 61131-3 y se debe habilitar de forma explícita.

Ejemplo:

Se admite:

```
AND (AnyBitParam := IntVar1, AnyBitParam2 := IntVar2);
```

No se admite:

```
AND_WORD (WordParam1 := IntVar1, WordParam2 := IntVar2);  
(En este caso se debe utilizar AND_INT).
```

```
AND_ARRAY_WORD (ArrayInt, ...);  
(En este caso se debe realizar una conversión de tipo explícita a través de  
INT_ARR_TO_WORD_ARR (...)).
```

En principio, no es necesario asignar un valor a todos los parámetros formales. En la tabla encontrará los tipos de parámetros formales que sí lo requieren obligatoriamente.

Tipo de parámetro	EDT	STRING	ARRAY	ANY_ARRAY	IODDT	STRUCT	FB	ANY
Entrada	-	-	+	+	+	+	+	+
VAR_IN_OUT	+	+	+	+	+	+	/	+
Salida	-	-	-	-	-	-	/	+
+ Parámetro actual requerido obligatoriamente								
- Parámetro actual no requerido obligatoriamente								
/ No aplicable								

Si no se asigna un valor a un parámetro formal, durante la ejecución del bloque de función se utilizará el valor inicial. Si no se ha definido ningún valor inicial, se aplicará el valor predeterminado (0).

Notas sobre la programación

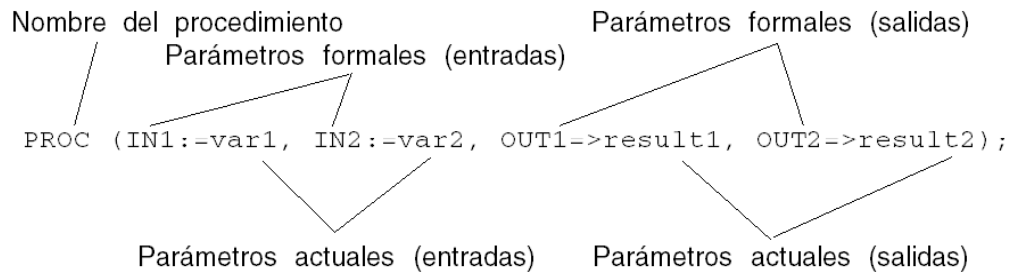
- Tenga en cuenta las siguientes indicaciones sobre la programación:
- Los procedimientos sólo se ejecutarán si la entrada **EN** es igual a 1 o si la entrada **EN** no se utiliza (consulte también "*EN y ENO, página 589*").
 - Si se utilizan variables **VAR_IN_OUT** (véase *página 590*), se imponen condiciones especiales.
 - Hay dos formas de llamar los procedimientos:
 - Llamada formal (véase *página 588*) (llamada con nombres de parámetros formales)
En este caso es posible asignar variables a las salidas mediante el operador **=>**.
 - Llamada informal (véase *página 588*) (llamada sin nombres de parámetros formales)

Llamada formal

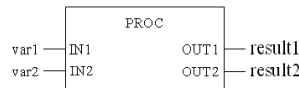
Con este tipo de llamada (llamada con nombres de parámetros formales), los procedimientos se llaman por medio de una instrucción compuesta por el nombre del procedimiento seguido de una lista entre paréntesis con asignaciones de parámetros actuales a los parámetros formales. La asignación de los parámetros formales de entrada se realiza mediante la asignación `:=`, y la asignación de los parámetros formales de salida, mediante la asignación `=>`. La secuencia en la que se enumeran los parámetros formales de entrada y de salida **no es significativa**.

Con este tipo de llamada se puede utilizar `EN` y `ENO`.

Llamada de un procedimiento con nombres de parámetros formales



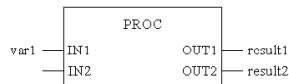
Llamada del mismo procedimiento en FBD



Con la llamada informal no es necesario asignar un valor a todos los parámetros formales (consulte también "*Parámetro, página 586*").

```
PROC (IN1:=var1, OUT1=>result1, OUT2=>result2);
```

Llamada del mismo procedimiento en FBD



Llamada informal

Con este tipo de llamada (llamada sin nombres de parámetros formales), los procedimientos se llaman por medio de una instrucción compuesta por el nombre del procedimiento, seguido de una lista entre paréntesis de los parámetros actuales de las entradas y las salidas. La secuencia en la que se enumeran los parámetros actuales en una llamada de procedimiento **es significativa**.

Con este tipo de llamada **no** se puede utilizar `EN` y `ENO`.

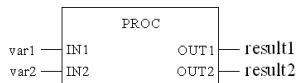
Llamada de un procedimiento sin nombres de parámetros formales

Nombre del procedimiento Parámetros actuales (entradas)

```
PROC (var1, var2, result1, result2) ;
```

Parámetro actual (salida)

Llamada del mismo procedimiento en FBD



Con la llamada informal tampoco es necesario asignar un valor a todos los parámetros formales (consulte también "*Parámetro, página 586*").

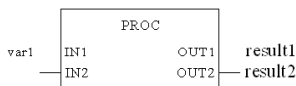
Se trata de una ampliación de la norma IEC 61131-3 y se debe habilitar de forma explícita.

Para excluir un parámetro se utiliza un campo de parámetros vacío.

Llamada con campo de parámetros vacío

```
PROC (var1, , result1, result2) ;
```

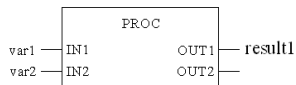
Llamada del mismo procedimiento en FBD



Si se suprimen los parámetros formales al final, no se tiene que utilizar un campo de parámetros vacío.

```
PROC (var1, var2, result1) ;
```

Llamada del mismo procedimiento en FBD



EN y ENO

En todos los procedimientos se puede configurar una entrada **EN** y una salida **ENO**.

Si el valor de **EN** es "0", al llamar la función no se ejecutarán los algoritmos definidos por dicho procedimiento y **ENO** se ajustará a "0".

Si el valor de **EN** es "1", al llamar el procedimiento se ejecutarán los algoritmos con los que se haya definido dicho procedimiento. Si no hay problemas en la ejecución de estos algoritmos, el valor de **ENO** se ajustará a "1". Si se produce un error, **ENO** se ajustará a "0".

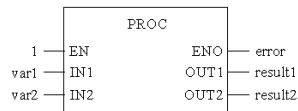
Si **ENO** está ajustado a "0" (porque **EN**=0 o debido a un error en la ejecución), las salidas del procedimiento se ajustarán a "0".

El comportamiento de salida del procedimiento no depende de si el procedimiento se ha ejecutado sin **EN** o con **EN**=1.

Si va a utilizar **EN** o **ENO**, la llamada del procedimiento deberá ser formal. La asignación de una variable a **ENO** debe realizarse con el operador **=>**.

```
PROC (EN:=1, IN1:=var1, IN2:=var2,
      ENO=>error, OUT1=>result1, OUT2=>result2) ;
```

Llamada del mismo procedimiento en FBD



Variable VAR_IN_OUT

A menudo, los procedimientos se utilizan para leer una variable en la entrada (variables de entrada), procesarla y volver a emitir los valores modificados de la misma variable (variables de salida). Este caso específico de variable de entrada/salida también se denomina variable **VAR_IN_OUT**.

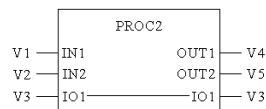
Tenga en cuenta las siguientes particularidades cuando utilice procedimientos con variables **VAR_IN_OUT**:

- Hay que asignar obligatoriamente una variable a todas las entradas **VAR_IN_OUT**.
- No se puede asignar ningún literal ni ninguna constante a las entradas **VAR_IN_OUT**.
- No es posible asignar **ningún** valor a las salidas **VAR_IN_OUT**.
- Las variables **VAR_IN_OUT** **no** se pueden utilizar fuera de la llamada de procedimientos.

Llamada de un procedimiento con variable **VAR_IN_OUT** en ST:

```
PROC2 (IN1:=V1, IN2:=V2, IO1:=V3,
      OUT1=>V4, OUT2=>V5) ;
```

Llamada del mismo procedimiento en FBD



Las variables `VAR_IN_OUT` **no** se pueden utilizar fuera de la llamada de procedimientos.

Por este motivo las siguientes llamadas de procedimientos son **inválidas**:

Llamada **inválida**, ejemplo 1.

<pre>InOutProc.inout := V1;</pre>	Asignación de las variables <code>V1</code> a un parámetro <code>VAR_IN_OUT</code>. Errores: No se puede realizar la operación porque no es posible acceder al parámetro <code>VAR_IN_OUT</code> fuera de la llamada de procedimientos.

Llamada **inválida**, ejemplo 2.

<pre>V1 := InOutProc.inout;</pre>	Asignación de un parámetro <code>VAR_IN_OUT</code> a la variable <code>V1</code>. Errores: No se puede realizar la operación porque no es posible acceder al parámetro <code>VAR_IN_OUT</code> fuera de la llamada de procedimientos.

Por el contrario, las siguientes llamadas de procedimientos son **válidas**:

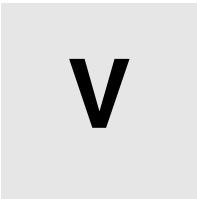
Llamada **válida**, ejemplo 1.

<pre>InOutProc (inout:=V1);</pre>	Llamada de un procedimiento con parámetro <code>VAR_IN_OUT</code> y asignación formal de los parámetros actuales dentro de la llamada de procedimientos.

Llamada **válida**, ejemplo 2.

<pre>InOutProc (V1);</pre>	Llamada de un procedimiento con parámetro <code>VAR_IN_OUT</code> y asignación formal de los parámetros actuales dentro de la llamada de procedimientos.

Bloques de funciones del usuario (DFB)



Objeto

En esta parte se muestran:

- los bloques de funciones del usuario (DFB),
- la estructura interna de los DFB,
- los DFB de diagnóstico,
- los tipos e instancias de DFB y
- las llamadas de las instancias mediante diferentes lenguajes.

Contenido de esta parte

Esta parte contiene los siguientes capítulos:

Capítulo	Nombre del capítulo	Página
16	Presentación de los bloques de funciones del usuario (DFB)	595
17	Descripción de los bloques de funciones del usuario (DFB)	601
18	Instancia de los bloques de funciones del usuario (DFB)	613
19	Utilización de los DFB a partir de los diferentes lenguajes de programación	621
20	usuario, DFB de diagnóstico	641

Presentación de los bloques de funciones del usuario (DFB)

16

Objeto

En este capítulo se muestran los bloques de funciones del usuario (DFB) y las diferentes etapas de instalación.

Contenido de este capítulo

Este capítulo contiene los siguiente apartados:

Apartado	Página
Introducción a los bloques de funciones del usuario	596
Implementación de un bloque de función DFB	598

Introducción a los bloques de funciones del usuario

Introducción

El software Unity Pro permite crear bloques de función de usuario DFB utilizando los lenguajes de automatismos. Un DFB es un bloque de programa que se ha escrito con el fin de responder a las características específicas de su aplicación. Incluye:

- una o varias secciones escritas en lenguaje de contactos (LD), en lista de instrucciones (IL), en literal estructurado (ST) o en lenguaje de bloques funcionales (FBD),
- parámetros de entradas/salidas y
- variables internas públicas o privadas.

Los bloques de función permiten estructurar y mejorar la aplicación. Se pueden utilizar desde el momento en el que una secuencia de programa se repite varias veces en la aplicación o bien para configurar una programación estándar (por ejemplo, el algoritmo de comando de un motor que incluya el reconocimiento de los mecanismos locales de seguridad).

La exportación y posterior importación de estos bloques de función permite que los utilice un grupo de programadores que trabaje en una misma aplicación o en aplicaciones diferentes.

Ventajas de la utilización de un DFB

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 la depuración de la aplicación (todas las variables introducidas por el bloque de función se identifican en la interfaz) y
- 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; sólo se generan los datos correspondientes a las instancias).

Comparación con una subrutina

En relación con un subprograma, la utilización de un DFB permite:

- parametrizar más fácilmente el procesamiento,
- utilizar variables internas propias del DFB, es decir, independientes de la aplicación, y
- comprobar su funcionamiento con independencia de la aplicación.

Además, los lenguajes LD y FBD permiten visualizar de forma gráfica los DFB, lo que facilita el diseño y la depuración del programa.

DFB creados con los programas anteriores

Los DFB creados con PL7 y Concept deben previamente convertirse con los convertidores incluidos en el producto antes de utilizarse en la aplicación.

Campo de uso

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

Función	Ámbito
Autómatas para los que se pueden utilizar los DFB.	Premium\Atrium y Quantum
Programa de creación de los DFB	Unity Pro
Programas con los que se pueden utilizar los DFB.	Unity Pro o Unity Pro Medium
Lenguaje de programación para la creación del código de los DFB.	IL, ST, LD o FBD (1)
Lenguajes de programación con los que se pueden utilizar los DFB.	IL, ST, LD o FBD (1)

(1) IL: Lista de instrucciones, ST: literal estructurado, LD: Lenguaje de contactos (LaDder), FBD: lenguaje de Bloques Funcionales.

Implementación de un bloque de función DFB

Procedimiento de implementación

El procedimiento de implementación de un bloque de funciones DFB consta de 3 pasos:

Paso	Acción
1	Crear el propio modelo de DFB (denominado: tipo de DFB).
2	Crear una copia de dicho bloque de función, denominada instancia, para cada utilización del DFB en la aplicación.
3	Usar las instancias de DFB en el programa de aplicación.

Creación del tipo de DFB

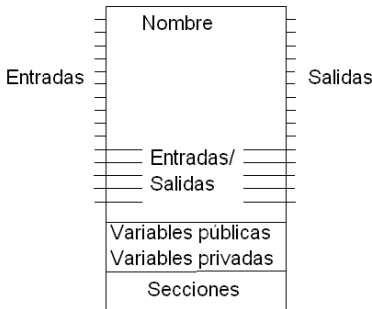
Esta operación consiste en diseñar un modelo del DFB que desea utilizar en la aplicación. Para ello, el editor de DFB permite definir y codificar todos los elementos que componen el DFB:

- La descripción del bloque de función: nombre, tipo (DFB), activación del diagnóstico, comentario.
- La estructura del bloque de función: parámetros, variables, secciones de código.

NOTA: Si utiliza un DFB que ya se encuentra en la biblioteca definida por el usuario y lo modifica, el nuevo tipo modificado se utilizará para cualquier otra instancia del proyecto abierto. La biblioteca definida por el usuario, sin embargo, permanecerá invariable.

Descripción de un tipo de DFB

La ilustración siguiente representa de forma gráfica un modelo de DFB.



El bloque de función incluye los elementos siguientes:

- Nombre: nombre del tipo de DFB (32 caracteres como máximo). Este nombre debe ser el único en las librerías, los caracteres que se pueden utilizar dependen de la elección realizada en el área **Identificadores** de la ficha **Extensiones de lenguaje en ajustes del proyecto** (véase *Unity Pro, Modalidades de funcionamiento*,):
- Entradas: parámetros de entradas (excluidos los parámetros de entradas/salidas).
- Salidas: parámetros de salidas (excluidos los parámetros de entradas/salidas).
- Entradas/Salidas: parámetros de entradas/salidas.
- Variables públicas: variables internas a las que se puede acceder a través del programa de aplicación.
- Variables privadas: variables internas o DFB enlazados, a los que no se puede acceder a través del programa de aplicación.
- Secciones: secciones de código del DFB en lenguaje LD, IL, ST o FBD.
- Comentario de 1.024 caracteres como máximo. Los caracteres de formato (retorno, tabulación, etc.) no están permitidos.

Para cada tipo de DFB también se puede acceder a una ficha descriptiva a través de un cuadro de diálogo: tamaño del DFB, número de parámetros y variables, número de versión, fecha de la última modificación, nivel de protección, etc.

Ayuda en línea para los tipos de DFB

Es posible vincular un fichero de ayuda HTML a cada DFB en la biblioteca definida por el usuario. Dicho fichero debe:

- Tener un nombre idéntico al del DFB vinculado,
- Ubicarse en el directorio `\Schneider Electric\FFBLibset\CustomLib\MyCustomFam\ Idioma` (dónde **Idioma** se denomina **Eng**, **Fre**, **Ger**, **Ita**, **Spa** o **Chs** según el idioma deseado).

Creación de una instancia de DFB

Una vez creado el tipo DFB, se puede definir 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.

Utilización de las instancias de DFB

Una instancia del DFB se utiliza a continuación

- como un bloque de función estándar en lenguaje de contactos LD o diagrama de bloques de función FBD,
- como una función elemental en lenguaje literal estructurado ST o lista de instrucciones IL.

Se puede utilizar una instancia de DFB en todas las tareas del programa de aplicación, excepto en las tareas de sucesos y las transiciones del diagrama funcional en secuencia SFC.

Fichero

Los tipos de DFB que cree el usuario se pueden almacenar (*véase Unity Pro, Modalidades de funcionamiento,*) en la librería de función y de bloque de funciones.

Descripción de los bloques de funciones del usuario (DFB)

17

Objeto

En este capítulo se muestran los diferentes elementos que componen los bloques de funciones del usuario.

Contenido de este capítulo

Este capítulo contiene los siguiente apartados:

Apartado	Página
Definición de datos internos de bloques de funciones DFB	602
Parámetros de DFB	605
Variables de DFB	609
Sección de código de DFB	611

Definición de datos internos de bloques de funciones DFB

Introducción

Los datos internos de los DFB son de dos tipos:

- Parámetros: entrada, salida o entrada/salida.
- Variables públicas o privadas.

Es necesario definir los datos internos del DFB de forma simbólica (estos datos no pueden enviarse como dirección).

Elementos que se deben definir para cada parámetro

Cuando se crea un bloque de función, se debe definir para cada uno de los parámetros:

- Nombre: nombre del tipo de DFB (32 caracteres como máximo). Dicho nombre debe ser único en las bibliotecas, los caracteres que se pueden utilizar dependen de la elección realizada en el área **Identificadores** de la ficha **Extensiones de lenguaje** en **Ajustes del proyecto** (véase *Unity Pro, Modalidades de funcionamiento*,).
- Un tipo de objeto (BOOL, INT, REAL, etc.)
- Un comentario opcional con un máximo de 1.024 caracteres. Los caracteres de formato (retorno, tabulación, etc.) no están permitidos.
- Un valor inicial
- El atributo de lectura/escritura que define si la variable se puede escribir en la pantalla de ejecución: L (sólo lectura) o L/E (lectura/escritura). Este atributo sólo debe definirse para las variables públicas.
- El atributo de almacenamiento que define si la variable se puede guardar.

Tipos de objetos

Los tipos de objetos que se pueden definir para los parámetros de DFB pertenecen a las siguientes familias:

- Familia de datos elementales: EDT. Esta familia incluye los siguientes tipos de objetos: Booleano (BOOL, EBOOL), Entero (INT, DINT, etc.), Real (REAL), Cadena de caracteres (STRING), Cadena de bits (BYTE, WORD, etc.), entre otros.
- Familia de datos derivados: DDT. Esta familia incluye los tipos de objeto de matriz (ARRAY) y estructura (usuario o IODDT).
- Familias de datos genéricos: ANY_ARRAY_xxx.
- Familia de los bloques de funciones: FB. Esta familia agrupa los tipos de objeto EFB y DFB.

Objetos permitidos para los diferentes parámetros

Por motivos de rendimiento, el modo de direccionamiento de los parámetros de DFB se debe transferir mediante dirección para las familias de objetos siguientes:

- Entradas
- Entradas/Salidas
- Salidas

El modo de direccionamiento de un elemento de bloque de función está vinculado al tipo de elemento. Los modos de direccionamiento se procesan:

- Por valor (VAL)
- Por entrada de la tabla de reubicación (RTE)
- Por dirección lógica: RTE + Offset (L-ADR)
- Por dirección lógica y número de elementos (L-ADR-LG)
- Por estructura de canales de E/S (IOCHS)

Para cada uno de los parámetros de DFB, se pueden utilizar las familias de objetos siguientes con sus modos de direccionamiento asociados:

Familias de objetos	EDT	STRING	Anónimo o matriz DDT	DDT ⁽¹⁾	IODDT	GDT: ANY_ARRAY_x	FB	ANY...
Entradas	VAL	L-ADR-LG	L-ADR-LG	L-ADR	No	L-ADR-LG	No	L-ADR-LG
Entradas/salidas	L-ADR ⁽²⁾	L-ADR-LG	L-ADR-LG	L-ADR	IOCHS (véase página 626)	L-ADR-LG	No	L-ADR-LG
Salidas	VAL	VAL	L-ADR-LG	VAL	No	L-ADR-LG	No	L-ADR-LG
Variables públicas	VAL	VAL	VAL	VAL	No	No	No	No
Variables privadas	VAL	VAL	VAL	VAL	No	No	RTE	No
Leyenda:								
(1)	Familia de datos derivados, excepto los datos de entrada/salida derivados (IODDT).							
(2)	Excepto para las variables estáticas de tipo EBOOL, con los PLC Quantum.							

ATENCIÓN

Desplazamiento del índice para los parámetros ANY_ARRAY_x

Se produce un desplazamiento del índice cuando utilizamos la variable ARRAY con índice de inicio no nulo en una entrada ANY_ARRAY_x. El valor del desplazamiento corresponde al valor del índice de inicio.

Si no se siguen estas instrucciones pueden producirse lesiones personales o daños en el equipo.

Parámetros de DFB

Ilustración

Esta ilustración presenta ejemplos de parámetros de DFB.



Descripción de parámetros

En la tabla siguiente se describe la función de cada tipo de parámetro:

Parámetros	Número máximo	Función
Entradas	32 (1)	Estos parámetros permiten pasar valores del programa de aplicación al programa interno del DFB. Se puede acceder a ellos en modalidad de lectura a través del DFB, pero no a través del programa de aplicación.
Salidas	32 (2)	Estos parámetros permiten pasar valores del DFB al programa de aplicación. Se puede acceder a ellos en modalidad de lectura a través del programa de aplicación, excepto en el caso de los parámetros de tipo de ARRAY.
Entradas/Salidas	32	Estos parámetros permiten pasar los datos del programa de aplicación al DFB, que los puede modificar y pasarlos de nuevo al programa de aplicación. Sólo se puede acceder a estos parámetros a través del programa de aplicación.

Leyenda:

- (1) Número de entradas + número de entradas/salidas inferior o igual a 32.
- (2) Número de salidas + número de entradas/salidas inferior o igual a 32.

NOTA: El IODDT relacionado con los dispositivos CANopen para Modicon M340 no puede usarse como un dispositivo de E/S DFB. Durante la fase de análisis y generación de un proyecto, el mensaje: "Este IODDT no puede usarse como parámetro DFB" advierte de las limitaciones al usuario.

Parámetros a los que se puede acceder a través del programa de aplicación

Los únicos parámetros a los que se puede acceder a través del programa de aplicación fuera de la llamada son los parámetros de salidas. Para ello, se debe utilizar la sintaxis siguiente en el programa: **Nombre_DFB.Nombre_parámetro**.

Nombre_DFB representa el nombre de la instancia del DFB utilizado (32 caracteres como máximo).

Nombre_parámetro representa el nombre del parámetro de salida (32 caracteres como máximo).

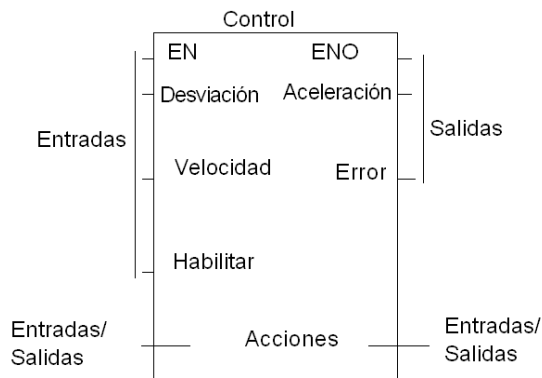
Ejemplo: `Control.Acel` indica la salida `Acel` de la instancia DFB denominada `Control`.

Caso de los parámetros EN y ENO

EN es un parámetro de entrada y **ENO** es un parámetro de salida. Ambos son de tipo BOOL y se pueden utilizar o no (opcional) al definir un tipo de DFB.

En caso de que el usuario desee utilizarlos, el editor los coloca automáticamente; EN es el primer parámetro de entrada y ENO el primer parámetro de salida.

Ejemplo de ejecución de los parámetros EN/ENO.



Si el parámetro de entrada EN de una instancia recibe el valor 0 (FALSE):

- las secciones que constituyen el código del DFB no se ejecutan (se gestiona a través del sistema),
- el sistema pone el parámetro de salida ENO en el estado 0 (FALSE).

Si el parámetro de entrada EN de una instancia recibe el valor 1 (TRUE):

- las secciones que constituyen el código del DFB se ejecutan (se gestiona a través del sistema),
- el sistema pone el parámetro de salida ENO en el estado 1 (TRUE).

Si se detecta un error (error de proceso, por ejemplo) por la instancia del DFB, el usuario puede, si así lo desea, poner el parámetro de salida ENO en el estado 0 (FALSE). En este caso:

- los parámetros de salidas se fijan en el estado que tenían en el tratamiento anterior hasta la desaparición del fallo,
- o bien, el usuario prevé en el código del DFB un forzado de las salidas en el estado que desee hasta la desaparición del fallo.

Variable VAR_IN_OUT

A menudo, los bloques de funciones se utilizan para leer una variable en la entrada (variables de entrada), procesarla y volver a emitir los valores modificados de la misma variable (variables de salida). Este caso específico de variable de entrada/salida también se denomina variable VAR_IN_OUT.

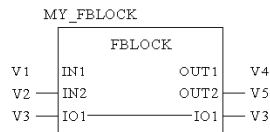
Tenga en cuenta las siguientes particularidades al utilizar bloques de funciones/DFB con variables VAR_IN_OUT:

- Hay que asignar obligatoriamente una variable a todas las entradas VAR_IN_OUT.
- No se puede asignar ningún literal ni ninguna constante a las entradas VAR_IN_OUT.
- No es posible asignar **ningún** valor a las salidas VAR_IN_OUT.
- Las variables VAR_IN_OUT **no** se pueden utilizar fuera de la llamada de bloque.

Llamada de un bloque de funciones con variable VAR_IN_OUT en IL:

```
CAL MY_FBLOCK(IN1:=V1, IN2:=V2, IO1:=V3,
              OUT1=>V4, OUT2=>V5)
```

Llamada del mismo bloque de funciones en FBD:



Las variables VAR_IN_OUT **no** se pueden utilizar fuera de la llamada de un bloque de funciones.

Por este motivo las siguientes llamadas de bloque de funciones son **inválidas**:

Llamada **inválida**, ejemplo 1.

LD V1	Carga de la variable V1 en el acumulador.
CAL InOutFB	Llamada de un bloque de funciones con parámetros VAR_IN_OUT. El acumulador está cargado ahora con la referencia a un parámetro VAR_IN_OUT.
AND V2	Operación AND del contenido del acumulador con la variable V2. Error: No se puede realizar la operación porque no es posible acceder al parámetro VAR_IN_OUT (contenido del acumulador) fuera de la llamada del bloque de funciones.

Llamada **inválida**, ejemplo 2.

LD V1	Carga de la variable V1 en el acumulador.
AND InOutFB.inout	Operación AND del contenido del acumulador con la referencia a un parámetro VAR_IN_OUT. Error: No se puede realizar la operación porque no es posible acceder al parámetro VAR_IN_OUT fuera de la llamada del bloque de funciones.

Por el contrario, las siguientes llamadas del bloque de funciones son **válidas**:

Llamada **válida**, ejemplo 1.

CAL InOutFB (IN1:=V1,inout:=V2)	Llamada de un bloque de funciones con parámetro VAR_IN_OUT y asignación de los parámetros actuales dentro de la llamada del bloque de funciones.

Llamada **válida**, ejemplo 2.

LD V1	Carga de la variable V1 en el acumulador.
ST InOutFB.IN1	Asignación del contenido del acumulador al parámetro IN1 del bloque de funciones IN1.
CAL InOutFB(inout:=V2)	Llamada del bloque de funciones con asignación del parámetro actual (V2) al parámetro VAR_IN_OUT.

Variables de DFB

Descripción de las variables

La tabla describe la función de cada uno de los tipos de variables.

Variable	Número máximo	Función
Pública	ilimitada	Estas variables internas del DFB se pueden utilizar por el DFB, por el programa de aplicación y por el usuario en modo de ajuste.
Privada	ilimitada	Estas variables internas del DFB sólo las puede emplear este bloque de función y, por lo tanto, no se puede acceder a ellas a través del programa de aplicación. Sin embargo, se puede acceder a este tipo de variables mediante la tabla de animación. Estas variables suelen ser por lo general necesarias para programar el bloque, pero no tienen interés para el usuario (por ejemplo, el resultado de un cálculo medio, etc.).

NOTA: Los DFB intercalados se declaran como variables privadas del DFB principal. Por tanto, tampoco se puede acceder a sus variables a través de la programación, sino a través de la tabla de animación.

Variables a las que se puede acceder a través del programa de aplicación

Las únicas variables a las que se puede acceder a través del programa de aplicación son las variables públicas. Para ello, se debe utilizar en el programa la sintaxis siguiente: **Nombre_DFB.Nombre_variable**

Nombre_DFB representa el nombre de la instancia del DFB utilizado (32 caracteres como máximo),

Nombre_variable representa el nombre de la variable pública (8 caracteres como máximo).

Ejemplo: `Control.Gan` indica la variable pública `Gan` de la instancia de DFB denominada `Control`.

Fichero de las variables públicas

Al poner a 1 el bit de sistema `%S94`, se guardan las variables públicas que se han modificado por programa o por ajuste, en lugar de los valores iniciales de estas variables (definidas en las instancias de DFB).

Sólo se pueden sustituir si el atributo de guardado está correctamente colocado para la variable.

ATENCIÓN

FALLO DE CARGA

Durante una carga, el bit %S94 no debe ponerse a 1.

Si el bit %S94 se pone a 1, puede producirse un fallo en la carga.

Si no se siguen estas instrucciones pueden producirse lesiones personales o daños en el equipo.

Sección de código de DFB

Generalidades

Las secciones de código definen el procesamiento que efectuará el DFB en función de los parámetros definidos.

Si la opción IEC está colocada, sólo se puede asignar una sección al DFB. En caso contrario, un DFB puede contener un número ilimitado de secciones de código.

Lenguajes de programación

Para programar las secciones de DFB se pueden utilizar los siguientes lenguajes:

- Lista de instrucciones (IL)
- Literal estructurado (ST)
- Lenguaje de contactos (LD)
- Lenguaje de bloques funcionales (FBD)

Definición de una sección

Una sección se define mediante:

- Un nombre simbólico que identifica la sección (32 caracteres como máximo),
- una condición de validación que define la ejecución de la sección,
- un comentario (256 caracteres como máximo), y
- un atributo de protección (sin protección, sección protegida contra escritura, sección protegida contra lectura/escritura).

Reglas de programación

Cuando se ejecuta, una sección de DFB sólo puede utilizar los parámetros que se han definido para el bloque de funciones (parámetros de entradas, de salidas, de entradas/salidas y variables internas).

Esto tiene como consecuencia que un bloque de funciones DFB no pueda utilizar las variables globales de la aplicación ni los objetos de entradas/salidas, con la excepción de los bits y las palabras de sistema (%Si, %SWi y %SDi).

Una sección de DFB tiene los derechos de acceso máximos (lectura y escritura) sobre estos parámetros.

Ejemplo de código

El siguiente programa ofrece un ejemplo de código literal estructurado ST

```

CHR_200:=CHR_100;
CHR_114:=CHR_104;
CHR_116:=CHR_106;
RESET INICIO;
(*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;
END_FOR;

```

Instancia de los bloques de funciones del usuario (DFB)

18

Objeto

En este capítulo, se muestra la creación de una instancia de DFB y su ejecución.

Contenido de este capítulo

Este capítulo contiene los siguiente apartados:

Apartado	Página
Creación de una instancia de DFB	614
Ejecución de una instancia de DFB	616
Ejemplo de programación de un bloque de función derivado (DFB)	617

Creación de una instancia de DFB

Instancia de DFB

Una instancia de DFB es una copia del modelo de DFB (tipo de DFB):

- Utiliza el código del tipo de DFB (el código no se duplica) y
- crea una zona de datos específica para esta instancia, que es una copia de los parámetros y de las variables del tipo de DFB. Esta zona se sitúa en el espacio de datos de la aplicación.

Deberá definir la dirección de cada instancia de DFB que cree mediante un nombre de 32 caracteres como máximo, los caracteres que se pueden utilizar dependen de la elección realizada en la zona **Identificadores** de la ficha **Extensiones de lenguaje** en los **ajustes del proyecto** (véase *Unity Pro, Modalidades de funcionamiento*,).

El primer carácter debe ser una letra. No se permite utilizar palabras clave o símbolos.

Creación de una instancia

A partir de un tipo de DFB, se pueden crear tantas instancias como sean necesarias. El único límite lo establece el tamaño de la memoria del autómata.

Valores iniciales

Los valores iniciales de los parámetros y variables públicas que se han definido durante la creación del tipo de DFB se pueden modificar para cada instancia del DFB.

No todos los parámetros de los DFB poseen un valor inicial.

Modificación de los valores iniciales de los elementos en las instancias de DFB

	EDT (excepto el tipo String)	Tipo String	Matric es	Estructur a DDT	FB	ANY_ARRA Y	IODDT	ANY_...
Entradas	Sí	No	No	No	-	No	-	No
Entradas\Salidas	No	No	No	No	-	No	No	No
Salidas	Sí	Sí	No	Sí	-	-	-	No
Variables públicas	Sí	Sí	Sí	Sí	-	-	-	-
Variables privadas	No	No	No	No	No	-	-	-

Modificación de los valores iniciales de los elementos en el tipo de DFB

	EDT (excepto el tipo String)	Tipo String	Matric es	Estructur a DDT	FB	ANY_ARRAY	IODDT	ANY_...
Entradas	Sí	No	No	No	-	No	-	No
Entradas\Salidas	No	No	No	No	-	No	No	No
Salidas	Sí	Sí	No	Sí	-	-	-	No
Variables públicas	Sí	Sí	Sí	Sí	-	-	-	-
Variables privadas	Sí	Sí	Sí	Sí	No	-	-	-

Ejecución de una instancia de DFB

Funcionamiento

Una instancia de DFB se ejecuta de la manera siguiente.

Paso	Acción
1	Cargar los valores en los parámetros de entradas y de entradas/salidas. Todas las entradas sin asignación toman en la inicialización (o en el re arranque en frío) el valor inicial definido en el tipo de DFB. A continuación, guardan el último valor que se les ha asignado.
2	Ejecutar el programa interno del DFB.
3	Escribir los parámetros de salidas.

NOTA: Las variables internas de los DFB no se reinician al utilizar el comando **Generar proyecto online** tras modificar una salida. Para reinicializar todas las variables internas, utilice el comando **Regenerar todos los proyectos**.

Depuración de los DFB

El programa Unity Pro ofrece varias herramientas de depuración de los DFB:

- tablas de animación: todos los parámetros, variables públicas y variables privadas están animadas y aparecen en tiempo real. Se puede modificar y forzar los objetos
- Punto de parada, paso a paso y diagnóstico del programa
- Pantallas de explotación: para la depuración unitaria

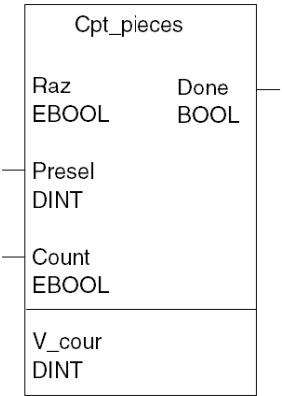
Ejemplo de programación de un bloque de función derivado (DFB)

Generalidades

Este ejemplo de programación de un contador, a partir de un DFB, tiene una finalidad didáctica.

Características del tipo de DFB

El tipo de DFB empleado para realizar el contador es el siguiente.



Los elementos del tipo de DFB `Cpt_piezas` son los siguientes.

Elementos	Descripción
Nombre del tipo de DFB	Cpt_piezas
Parámetros de entradas	● Puesta a cero: puesta a cero del contador (tipo EBOOL) ● Presel.: valor de preselección del contador (tipo DINT) ● Count: entrada de contaje (tipo EBOOL)
Parámetros de salidas	Done : salida de valor de preselección alcanzada (tipo BOOL)
Variable interna pública	V_cour : valor actual del contador (tipo DINT)

Funcionamiento del contador

El funcionamiento del contador debe ser el siguiente.

Fase	Descripción
1	El DFB cuenta los flancos ascendentes en la entrada Count .

Fase	Descripción
2	La variable <code>V_cour</code> memoriza el número de flancos contados. Esta variable se vuelve a poner a cero por un flanco ascendente en la entrada <code>Restablecer</code> .
3	Cuando el número de flancos contados es igual al valor de preselección, la salida <code>Done</code> se coloca en 1. Esta variable se vuelve a poner a cero por un flanco ascendente en la entrada <code>Restablecer</code> .

Programa interno del DFB

El programa interno del tipo de DFB `Cpt_piezas` se define en lenguaje ST de la manera siguiente.

```

!(*Programación del DFB Cpt_piezas*)
IF RE (Raz) THEN
    V_cour=0;
END_IF;
IF RE (Count) THEN
    V_cour=V_cour+1;
END_IF;
IF (V_cour>=Presel) THEN
    SET (Done);
ELSE
    RESET (Done);
END_IF;

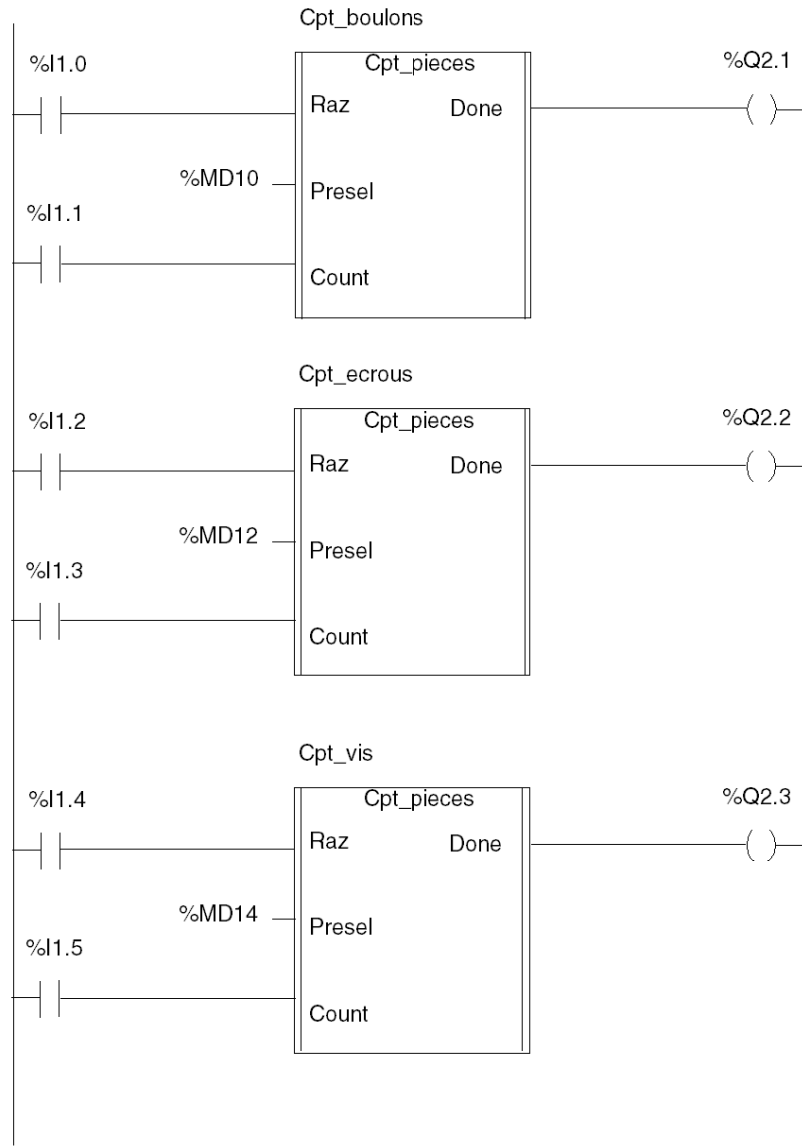
```

Ejemplo de uso

Supongamos que la aplicación necesita contar 3 tipos de piezas (por ejemplo, arandelas, tuercas y tornillos). Se puede utilizar 3 veces el tipo de DFB `Cpt_piezas` (3 instancias) para realizar los distintos contajes.

El número de piezas que se deben suministrar para cada tipo se define respectivamente en las palabras `%MD10`, `%MD12` y `%MD14`. Cuando se alcanza el número de piezas, el contador envía un comando a una salida (`%Q1.2.1`, `%Q1.2.2` o `%Q1.2.3`) que controla la parada del sistema de suministro de piezas correspondiente.

Los datos se introducen en el programa de aplicación en lenguaje de contactos de la manera siguiente. Utiliza los 3 DFB (instancias) Cpt_arandelas, Cpt_tuercas y Cpt_tornillos para contar las diferentes piezas.



Utilización de los DFB a partir de los diferentes lenguajes de programación

19

Objeto

En este capítulo se muestra la llamada de las instancias de DFB a partir de los diferentes lenguajes de programación.

Contenido de este capítulo

Este capítulo contiene los siguiente apartados:

Apartado	Página
Reglas de uso de los DFB en un programa	622
Utilización de los IODDT en un DFB	626
Utilización de un DFB en un programa en lenguaje de contactos	629
Utilización de un DFB en un programa en literal estructurado	631
Utilización de un DFB en un programa en lista de instrucciones	634
Utilización de un DFB en un programa en lenguaje de bloques funcionales	638

Reglas de uso de los DFB en un programa

General

Las instancias de DFB se pueden utilizar en todos los lenguajes (lista de instrucciones IL, literal estructurado ST, lenguaje de contactos LD y diagrama de bloques de funciones FBD) y en todas las tareas del programa de aplicación (secciones, subprograma, etc.), excepto las tareas de eventos y las transiciones del programa SFC.

Reglas generales de uso

Cuando se utiliza un DFB, se deben respetar las reglas siguientes, independientemente del lenguaje empleado:

- No es necesario conectar todos los parámetros de entradas/salidas o de salidas, excepto los parámetros siguientes, que se deben asignar obligatoriamente:
 - Los parámetros de entradas de tipos de datos genéricos (ANY_INT, ANY_ARRAY, etc.).
 - Los parámetros de entradas/salidas
 - Los parámetros de salidas de tipo de datos (fuera de matrices) genéricos (ANY_INT, ANY_REAL, etc.).
 - Los parámetros de entrada de tipo STRING.
- Los parámetros de entradas sin conectar guardan el valor de la llamada anterior o el valor de inicialización definidos para estos parámetros, si nunca se ha llamado el bloque.
- 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 de DFB (por ejemplo: Si el tipo INT está definido para el parámetro de entrada "velocidad", no se le puede asignar el tipo DINT ni REAL).

Únicamente pueden mezclarse los tipos BOOL y EBOOL para los parámetros de entradas o de salidas (nunca para los parámetros de entradas/salidas).

Ejemplo: El parámetro de entrada "Validación" se puede definir como BOOL y se puede asociar a un bit interno %Mi, que es de tipo EBOOL. Por el contrario, en el código interno del tipo DFB, el parámetro de entrada tiene efectivamente la propiedad de un tipo BOOL (no puede gestionar los flancos).

Asignación de los parámetros

En la tabla siguiente se resumen las diferentes posibilidades de asignación de los parámetros en los diferentes lenguajes de programación.

Parámetros	Tipo	Asignación del parámetro (1)	Asignación
Entradas	EDT (2)	Conectado, valor, objeto o expresión	Opcional (3)
	BOOL	Conectado, valor, objeto o expresión	Opcional
	DDT	Conectado, valor u objeto	Obligatoria
	ANY_...	Conectado u objeto	Obligatoria
	ANY_ARRAY	Conectado u objeto	Obligatoria
Entradas/salidas	EDT	Conectado u objeto	Obligatoria
	DDT	Conectado u objeto	Obligatoria
	IODDT	Conectado u objeto	Obligatoria
	ANY_...	Conectado u objeto	Obligatoria
	ANY_ARRAY	Conectado u objeto	Obligatoria
Salidas	EDT	Conectado u objeto	Opcional
	DDT	Conectado u objeto	Opcional
	ANY_...	Conectado u objeto	Obligatoria
	ANY_ARRAY	Conectado u objeto	Opcional

(1) Conectado en lenguaje de contactos (LD) o en lenguaje de bloques funcionales (FBD). Valor u objeto en los lenguajes lista de instrucciones (IL, Instruction List) o texto estructurado (ST, Structured Text).

(2) Excepto parámetros de tipo BOOL.

(3) Excepto parámetros de tipo STRING que son obligatorios.

Asignación de los parámetros

En la tabla siguiente se resumen las diferentes posibilidades de asignación de los parámetros en los diferentes lenguajes de programación.

Parámetros	Tipo	Asignación del parámetro (1)	Asignación
Entradas	EDT (2)	Conectado, valor, objeto o expresión	Opcional (3)
	BOOL	Conectado, valor, objeto o expresión	Opcional
	DDT	Conectado, valor u objeto	Obligatoria
	ANY_...	Conectado u objeto	Obligatoria
	ANY_ARRAY	Conectado u objeto	Obligatoria
Entradas/salidas	EDT	Conectado u objeto	Obligatoria
	DDT	Conectado u objeto	Obligatoria
	IODDT	Conectado u objeto	Obligatoria
	ANY_...	Conectado u objeto	Obligatoria
	ANY_ARRAY	Conectado u objeto	Obligatoria
Salidas	EDT	Conectado u objeto	Opcional
	DDT	Conectado u objeto	Opcional
	ANY_...	Conectado u objeto	Obligatoria
	ANY_ARRAY	Conectado u objeto	Opcional

(1) Conectado en lenguaje de contactos (LD) o en lenguaje de bloques funcionales (FBD). Valor u objeto en los lenguajes lista de instrucciones (IL, Instruction List) o texto estructurado (ST, Structured Text).

(2) Excepto parámetros de tipo BOOL.

(3) Excepto parámetros de tipo STRING que son obligatorios.

Reglas al usar DFB con matrices

**ADVERTENCIA**

FUNCIONAMIENTO INESPERADO DEL EQUIPO
Compruebe el tamaño de las matrices al copiar del origen a las matrices de destino con DFB.
Si no se siguen estas instrucciones pueden producirse lesiones personales graves o mortales o daños en el equipo.

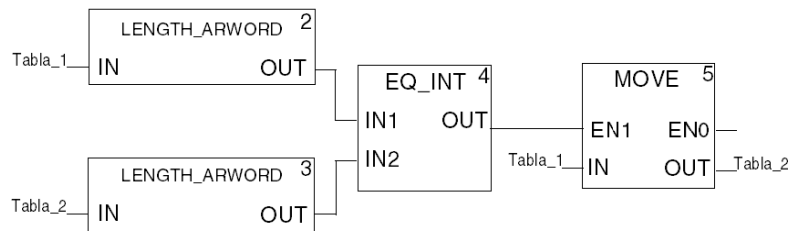
Al usar matrices dinámicas, es obligatorio comprobar el tamaño de las matrices idénticas. En casos específicos, al usar matrices dinámicas como una salida o como una entrada/salida, un desborde podría llevar a una ejecución correcta del programa y detener el PLC.

Este comportamiento se produce si se cumplen las siguientes condiciones simultáneamente:

- El uso de un DFB con al menos una salida o un parámetro de E/S del tipo de matriz dinámica (`ANY_ARRAY_XXX`).
- Al codificar un DFB, use una función o un bloque de funciones (FFB de tipo FIFO, LIFO, MOVE, MVX, T2T, SAH o SEL). Tenga en cuenta que la función o el FFB necesita dos parámetros de tipo `ANY` con al menos uno definido en la salida.
- El parámetro DFB de la matriz dinámica se usa para escribir durante la llamada FFB (en el parámetro de tipo `ANY`) . Para los demás parámetros `ANY`, se usa una matriz con un tamaño fijo.
- El tamaño de la matriz de tamaño fijo es mayor que el tamaño de la matriz dinámica calculada para almacenar el resultado.

Ejemplo de la comprobación del tamaño de las matrices

En el siguiente ejemplo se muestra cómo comprobar el tamaño de las matrices con la función `LENGTH_ARWORD` en un DFB.



En este ejemplo, **Tabla_1** es una matriz con un tamaño fijo, **Tabla_2** es una matriz dinámica de tipo `ANY_ARRAY_WORD`. Este programa comprueba el tamaño de cada matriz. Las funciones `LENGTH_ARWORD` calculan el tamaño de cada matriz para condicionar la ejecución de la función `MOVE`.

Utilización de los IODDT en un DFB

Presentación

Las siguientes tablas muestran los diferentes IODDT de los PLC Modicon M340, Premium y Quantum que se pueden utilizar en un DFB (exclusivamente como parámetros de entradas/salidas (*véase página 603*)).

IODDT que pueden utilizarse en un DFB

En la tabla siguiente se resumen los IODDT de las distintas aplicaciones para PLC Modicon M340, Premium y Quantum que pueden utilizarse en un DFB.

Familias de IODDT	Modicon M340	Premium	Quantum
Aplicación binaria			
T_DIS_IN_GEN	No	No	No
T_DIS_IN_STD	No	No	No
T_DIS_EVT	No	No	No
T_DIS_OUT_GEN	No	No	No
T_DIS_OUT_STD	No	No	No
T_DIS_OUT_REFLEX	No	No	No
Aplicación analógica			
T_ANA_IN_GEN	No	No	No
T_ANA_IN_STD	No	No	No
T_ANA_IN_CTRL	No	Sí	No
T_ANA_IN_EVT	No	Sí	No
T_ANA_OUT_GEN	No	No	No
T_ANA_OUT_STD	No	No	No
T_ANA_IN_BMX	Sí	No	No
T_ANA_IN_T_BMX	Sí	No	No
T_ANA_OUT_BMX	Sí	No	No
T_ANA_IN_VE	No	No	No
T_ANA_IN_VWE	No	No	No
T_ANA_BI_VWE	No	No	No
T_ANA_BI_IN_VWE	No	No	No
Aplicación de conteo			
T_COUNT_ACQ	No	Sí	No
T_COUNT_HIGH_SPEED	No	Sí	No
T_COUNT_STD	No	Sí	No

Familias de IODDT	Modicon M340	Premium	Quantum
T_SIGN_CPT_BMX	Sí	No	No
T_UNSIGN_CPT_BMX	Sí	No	No
T_CNT_105	No	No	No
Aplicación de leva electrónica			
T_CCY_GROUP0	No	No	No
T_CCY_GROUP1_2_3	No	No	No
Aplicación de control de ejes			
T_AXIS_AUTO	No	Sí	No
T_AXIS_STD	No	Sí	No
T_INTERPO_STD	No	Sí	No
T_STEPPER_STD	No	Sí	No
Aplicación Sercos			
T_CSY_CMD	No	Sí	No
T_CSY_TRF	No	Sí	No
T_CSY_RING	No	Sí	No
T_CSY_IND	No	Sí	No
T_CSY_FOLLOW	No	Sí	No
T_CSY_COORD	No	Sí	No
T_CSY_CAM	No	Sí	No
Aplicación de comunicación			
T_COM_STS_GEN	Sí	Sí	No
T_COM_UTW_M	No	Sí	No
T_COM_UTW_S	No	Sí	No
T_COM_MB	No	Sí	No
T_COM_CHAR	No	Sí	No
T_COM_FPW	No	Sí	No
T_COM_MBP	No	Sí	No
T_COM_JNET	No	Sí	No
T_COM_ASI	No	Sí	No
T_COM_ETY_1X0	No	Sí	No
T_COM_ETY_210	No	Sí	No
T_COM_IBS_128	No	Sí	No
T_COM_IBS_242	No	Sí	No
T_COM_PBY	No	Sí	No

Familias de IODDT	Modicon M340	Premium	Quantum
T_COM_CPP100	No	Sí	No
T_COM_ETYX103	No	Sí	No
T_COM_ETHCOPRO	No	Sí	No
T_COM_MB_BMX	Sí	No	No
T_COM_CHAR_BMX	Sí	No	No
T_COM_CO_BMX	Sí	No	No
T_COM_ETH_BMX	Sí	No	No
Aplicación de ajuste			
T_PROC_PLOOP	No	Sí	No
T_PROC_3SING_LOOP	No	Sí	No
T_PROC_CASC_LOOP	No	Sí	No
T_PROC_SPP	No	Sí	No
T_PROC_CONST_LOOP	No	Sí	No
Aplicación de pesaje			
T_WEIGHING_ISPY101	No	Sí	No
Comunes a todas las aplicaciones			
T_GEN_MOD	No	No	No

Utilización de un DFB en un programa en lenguaje de contactos

Principio

En el lenguaje de contactos LD, existen dos posibilidades para llamar a un bloque de función DFB:

- A través de una llamada textual en un bloque de operación en el que la sintaxis y los límites de los parámetros son idénticos a los del lenguaje literal estructurado, o
- mediante una llamada gráfica

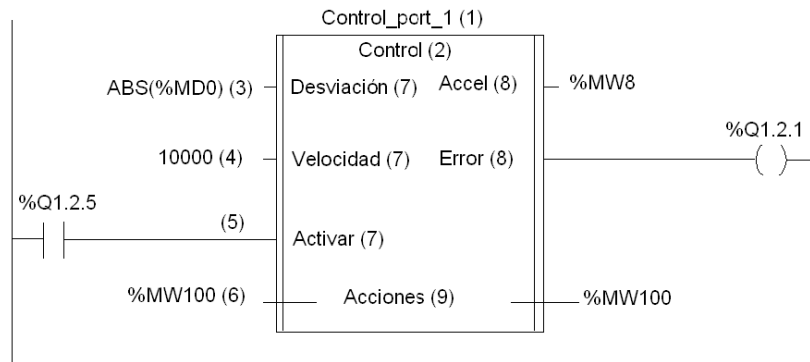
Las entradas de los bloques de función pueden estar conectadas o se les puede asignar un valor, un objeto o una expresión. En todos los casos, el tipo de elemento exterior (valor, evaluación de la expresión, etc.) debe ser idéntico al del parámetro de entrada.

Un bloque DFB debe tener al menos una entrada booleana conectada y una salida (si es necesario). Para ello, se puede utilizar los parámetros de entrada EN y el parámetro de salida ENO (véase a continuación la descripción de los parámetros).

Es obligatorio conectar o asignar las entradas de tipo ANY_ARRAY, las salidas de tipo datos genéricos (ANY_...) y las entradas/salidas de un bloque DFB.

Representación gráfica de un bloque DFB

La siguiente ilustración presenta un ejemplo simple de programación de un DFB.



Elementos del bloque DFB

En la siguiente tabla, se enumeran los diferentes elementos del bloque DFB, indicados en la figura anterior.

Variable	Elemento
1	Nombre del DFB (instancia)
2	Nombre del tipo de DFB
3	Entrada asignada por una expresión
4	Entrada asignada por un valor
5	Entrada conectada
6	Entrada asignada por un objeto (dirección o símbolo)
7	Parámetros de entradas
8	Parámetros de salidas
9	Parámetros de entradas/salidas

Utilización de los parámetros EN\ENO

Véase *Caso de los parámetros EN y ENO*, página 606

Utilización de un DFB en un programa en literal estructurado

Principio

En literal estructurado ST, la llamada de un bloque de función de usuario se efectúa mediante una llamada del DFB: nombre de la instancia de DFB seguido de una lista de argumentos. En la lista, materializados por paréntesis, los argumentos están separados con comas.

La llamada del DFB puede ser de 2 tipos:

- Llamada formal, cuando los argumentos son asignaciones (parámetro = valor). En este caso, el orden de introducción de los argumentos en la lista no tiene importancia.
Se puede utilizar el parámetro de entrada EN y el parámetro de salida ENO para controlar la ejecución del bloque de función,
- Llamada informal, cuando los argumentos son valores (expresión, objeto o un valor inmediato). En este caso, el orden de introducción de los argumentos en la lista debe respetar el orden de los parámetros de entradas del DFB, incluidas las entradas sin asignar (el argumento es un campo vacío).
No se pueden utilizar los parámetros EN ni ENO.

`Nombre_DFB (argumento 1, argumento 2, ..., argumento n)`

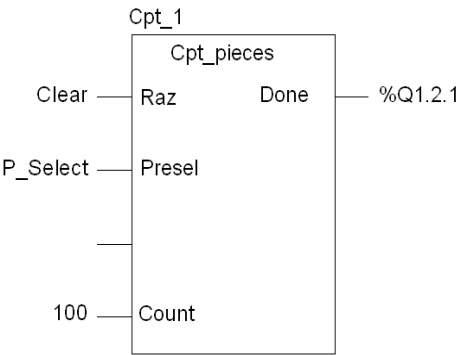
NOTA: Los parámetros de entradas de tipo ANY_ARRAY, las salidas de tipo de datos genéricos (ANY_...) y hay que asignar las entradas/salidas de un DFB.

Utilización de los parámetros EN\ENO

Véase *Caso de los parámetros EN y ENO*, página 606

Ejemplo de DFB

El ejemplo simple siguiente va a permitir comprender las diferentes llamadas de un DFB en lenguaje literal estructurado. Se toma la instancia `Cpt_1` del DFB de tipo `Cpt_piezas`:



Llamada formal del DFB

La llamada formal del DFB `Cpt_1` se efectúa según la sintaxis:

```
Cpt_1 (Restablecer:=Borrar, Presel:=P_Select, Conteo:=100,
Ejecutado=>%Q1.2.1);
```

Caso en el que únicamente los parámetros de entradas asignadas por un valor (expresión, objeto o valor inmediato) se introducen en la lista de los argumentos.

```
Cpt_1 (Restablecer:=Borrar, Presel:=P_Select, Conteo:=100);
...
%Q1.2.1:=Cpt_1.Ejecutado;
```

Elementos de la secuencia

En la siguiente tabla se enumeran los distintos elementos de la secuencia de programa, cuando se realiza una llamada formal del DFB.

Elemento	Significado
Cpt_1	Nombre de la instancia de DFB
Restablecer, Presel, Conteo	Parámetros de entradas
:=	Símbolo de asignación de una entrada
Clear	Objeto de asignación de una entrada (símbolo)
100	Valor de asignación de una entrada

Elemento	Significado
Done	Parámetro de salida
=>	Símbolo de asignación de una salida
%Q1.2.1	Objeto de asignación de una salida (dirección)
;	Símbolo de final de frase
,	Símbolo de separación de los argumentos

Llamada informal del DFB

La llamada informal del DFB Cpt_1 se efectúa según la sintaxis:

```
Cpt_1 (Borrar, %MD10, , 100);
...
%Q1.2.1:=Cpt_1.Ejecutado;
```

Elementos de la frase

En la siguiente tabla se enumeran los distintos elementos de la secuencia de programa, cuando se realiza una llamada formal del DFB.

Elemento	Significado
Cpt_1	Nombre de la instancia de DFB
Borrar, %MD10, ,100	Objeto o valor de asignación de las entradas. Las entradas no asignadas se representan con un campo vacío
;	Símbolo de final de frase
,	Símbolo de separación de los argumentos

Utilización de un DFB en un programa en lista de instrucciones

Principio

En lista de instrucciones (IL), la llamada de un bloque de función de usuario se realiza mediante una instrucción `CAL`, seguida del nombre de la instancia de DFB como operando y de una lista de argumentos (opcional). En la lista, materializados por paréntesis, los argumentos están separados con comas.

Existen 3 posibilidades para llamar a un DFB en lenguaje IL:

- La instrucción `CAL nombre_DFB` va seguida de una lista de argumentos que son asignaciones (parámetro = valor). En este caso, el orden de introducción de los argumentos en la lista no tiene importancia.
Se puede utilizar la entrada `EN` para controlar al ejecución del bloque de función.
- La instrucción `CAL nombre_DFB` va seguida de una lista de argumentos que son valores (expresión, objeto o valor inmediato). En este caso, el orden de introducción de los argumentos en la lista debe respetar el orden de los parámetros de entradas del DFB, incluidas las entradas sin asignar (el argumento es un campo vacío).
No se pueden utilizar los parámetros `EN` ni `ENO`.
- la instrucción `CAL nombre_DFB` no va seguida de una lista de argumentos. En tal caso, esta instrucción debe ir precedida de la asignación de los parámetros de entradas a través de un registro: carga del valor (`Load`) y a continuación asignación al parámetro de entrada (`Store`). El orden de asignación de los parámetros (`LD/ST`) no es importante; no obstante, se deben asignar todos los parámetros de entradas que lo necesiten antes de ejecutar el comando `CAL`. No se pueden utilizar los parámetros `EN` ni `ENO`.

```
CAL Nombre_DFB (argumento 1, argumento 2, ..., argumento n)
O
LD Valor 1
ST Parámetro 1
...
LD Valor n
ST Parámetro n
CAL Nombre_DFB
```

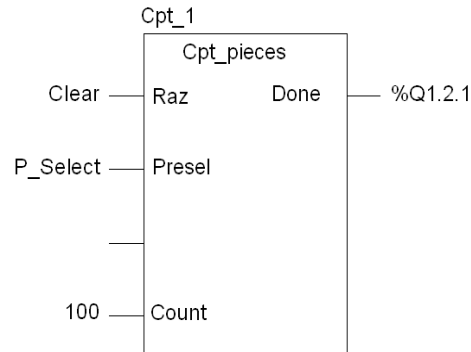
NOTA: Los parámetros de entradas de tipo `ANY_ARRAY`, las salidas de tipo de datos genéricos (`ANY_...`). y hay que asignar las entradas/salidas de un DFB.

Utilización de los parámetros `EN/ENO`

Véase *Caso de los parámetros `EN` y `ENO`*, página 606

Ejemplo de DFB

El ejemplo siguiente va a permitir comprender las diferentes llamadas de un DFB en lista de instrucciones. Se toma la instancia `Cpt_1` del DFB de tipo `Cpt_piezas`:



Llamada del DFB cuando los argumentos son asignaciones

Cuando los argumentos son asignaciones, la llamada del DFB `Cpt_1` se efectúa según la sintaxis:

```
CAL Cpt_1 (Restablecer:=Borrar, Presel:=%MD10, Conteo:=100,
Ejecutado=>%Q1.2.1)
```

Caso en el que únicamente los parámetros de entradas asignados por un valor (expresión, objeto o valor inmediato) se introducen en la lista de los argumentos:

```
CAL Cpt_1 (Restablecer:=Borrar, Presel:=%MD10, Conteo:=100)
```

...

```
LD Cpt_1.Ejecutado
```

```
ST %Q1.2.1
```

Con el fin de hacer más legible el programa de aplicación, se puede introducir un retorno a la siguiente línea después de las comas de separación de los argumentos. La frase tiene entonces la siguiente sintaxis:

```
CAL Cpt_1 (
Restablecer:=Borrar,
Presel:=%MD10,
Conteo:=100,
Ejecutado=>%Q1.2.1)
```

Elementos del programa de llamada del DFB

En la siguiente tabla se enumeran los diferentes elementos del programa de llamada del DFB.

Elemento	Significado
CAL	Instrucción de llamada del DFB
Cpt_1	Nombre de la instancia de DFB
Restablecer, Presel, Conteo	Parámetros de entradas
:=	Símbolo de asignación de una entrada
Borrar, %MD10, 100	Objeto o valor de asignación de las entradas
Ejecutado	Parámetro de salida
=>	Símbolo de asignación de una salida
%Q1.2.1	Objeto de asignación de una salida
,	Símbolo de separación de los argumentos

Llamada del DFB cuando los argumentos son valores

Cuando los argumentos son valores, la llamada del DFB Cpt_1 se efectúa según la sintaxis:

```
CAL Cpt_1 (Borrar, %MD10,, 100)
...
LD Cpt_1.Ejecutado
ST %Q1.2.1
```

Elementos del programa de llamada del DFB

En la siguiente tabla se enumeran los diferentes elementos del programa de llamada del DFB.

Elemento	Significado
CAL	Instrucción de llamada del DFB
Cpt_1	Nombre de la instancia de DFB
Borrar, %MD10, 100	Objeto o valor de asignación de las entradas
,	Símbolo de separación de los argumentos

Llamada de un DFB sin argumento

Cuando no hay argumento, la llamada del DFB Cpt_1 se efectúa según la sintaxis:

```
LD Borrar
ST Cpt_1.Restablecer
```

```
LD %MD10
ST Cpt_1.Presel
LD 100
ST Cpt_1.Conteo
CAL Cpt_1(
...
LD Cpt_1.Ejecutado
ST %Q1.2.1
```

Elementos del programa de llamada del DFB

En la siguiente tabla se enumeran los diferentes elementos del programa de llamada del DFB.

Elemento	Significado
LD Borrar	Instrucción de carga del valor <code>Borrar</code> en un registro
ST Cpt_1.Restablecer	Instrucción de asignación del contenido del registro al parámetro de entrada <code>Cpt_1.Restablecer</code>
CAL Cpt_1(	Instrucción de llamada del DFB <code>Cpt_1</code>

Utilización de un DFB en un programa en lenguaje de bloques funcionales

Principio

En el lenguaje FBD (Diagrama de Bloques de Función), los bloques de función del usuario se representan de la misma forma que en el lenguaje de contactos y se llaman de forma gráfica.

Las entradas de los bloques de función del usuario pueden estar conectadas o se les puede asignar un valor inmediato, un objeto o una expresión. En todos los casos, el tipo de elemento exterior debe ser idéntico al del parámetro de entrada.

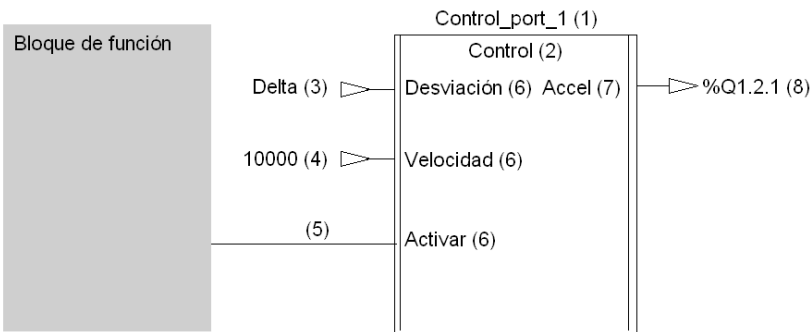
Sólo se puede asignar un único objeto (enlace hacia otro bloque o variable) en una entrada del DFB. Por el contrario, se pueden conectar varios objetos a una misma salida.

Un bloque DFB debe tener al menos una entrada booleana conectada y una salida (si es necesario). Para ello, se puede utilizar un parámetro de entrada EN y un parámetro de salida ENO.

Es obligatorio conectar o asignar las entradas de tipo ANY_ARRAY, las salidas de tipo datos genéricos (ANY_...) y las entradas/salidas de un bloque de DFB.

Representación gráfica de un bloque DFB

La siguiente ilustración presenta un ejemplo simple de programación de un DFB.



Elementos del bloque DFB

En la siguiente tabla, se enumeran los diferentes elementos del bloque DFB, indicadas en la figura anterior.

Variable	Elemento
1	Nombre del DFB (instancia)
2	Nombre del tipo de DFB

Variable	Elemento
3	Entrada asignada por un objeto (símbolo)
4	Entrada asignada por un valor
5	Entrada conectada
6	Parámetros de entradas
7	Parámetro de salida
8	Salida asignada por un objeto (dirección)

Utilización de los parámetros EN\ENO

Véase Caso de los parámetros EN y ENO, página 606

Presentación de los DFB de diagnóstico de usuario

Generalidades

El software Unity Pro le permite crear sus propios DFB de diagnóstico (*véase Unity Pro, Modalidades de funcionamiento, .*).

Dichos DFB de diagnóstico son DFB estándar configurados previamente mediante la **propiedad Diagnóstico** y en la que se ha utilizado las dos funciones siguientes:

- REGDFB (*véase Unity Pro, Diagnósticos, Biblioteca de bloques*) para el registro y el establecimiento de la fecha de la alarma.
- DEREGB (*véase Unity Pro, Diagnósticos, Biblioteca de bloques*) para la anulación del registro de la alarma.

NOTA: Se recomienda encarecidamente que programe sólo una instancia DFB de diagnóstico una vez dentro de la aplicación.

Estos DFB le permiten controlar el proceso. Además, transmiten de forma automática la información seleccionada en el Visualizador. Así, se pueden vigilar los cambios de estado o las variaciones en el proceso.

Ventajas

Estas son las principales ventajas de este servicio:

- El diagnóstico queda integrado en el proyecto; de este modo, se puede considerar en el momento del desarrollo para poder satisfacer de la mejor manera las necesidades del usuario.
- El sistema de registro y de establecimiento de la fecha de los errores se lleva a cabo en el origen (en el autómatas) para que la información sea el fiel reflejo del estado del proceso.
- Es posible conectar varios visualizadores (Unity Pro, Magelis, Factory Cast) que transcribirán al operador el estado exacto del proceso. Cada visualizador es independiente y toda acción que se realice en uno de ellos (por ejemplo, una confirmación) se visualiza automáticamente en los demás.

Apéndices



Objetivo de este capítulo

Este apéndice contiene información adicional.

Contenido de este anexo

Este anexo contiene los siguientes capítulos:

Capítulo	Nombre del capítulo	Página
A	Códigos y valores de error de EFB	645
B	Conformidad con IEC	693

Códigos y valores de error de EFB



Introducción

En las tablas siguientes se muestran los códigos y valores de error creados para los EFB ordenados por biblioteca y familia.

Contenido de este capítulo

Este capítulo contiene los siguiente apartados:

Apartado	Página
Tabla de códigos de error de la librería base	646
Tabla de códigos de error de la librería de diagnóstico	648
Tablas de códigos de error de la librería de comunicación	649
Tablas de códigos de error para la librería de gestión de E/S	654
Tabla de códigos de error de la librería CONT_CTL	665
Tabla de códigos de error de la biblioteca de movimiento	675
Tablas de códigos de error de la biblioteca obsoleta	679
Errores comunes de coma flotante	691

Tabla de códigos de error de la librería base

Introducción

En las siguientes tablas se muestran los códigos y valores de error creados para los EFB de la librería base.

Fecha y hora

Tabla de códigos y valores de error creados para los EFB de la familia Fecha y hora.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
DIVTIME	E_DIVIDE_BY_ZERO	F	-30.176	16#8A20	División por cero.
DIVTIME	E_NEGATIVE_INPUT_FOR_TIME_OPERATION	F	-30.177	16#8A1F	Un valor negativo no se puede convertir en un tipo de dato <code>TIME</code> .
DIVTIME	E_ARITHMETIC_ERROR	F	-30.170	16#8A26	Error aritmético.
DIVTIME	E_ERR_ARITHMETIC	F	-30.003	16#8ACD	Desborde aritmético (%S18 establecido).
DIVTIME	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
MULTIME	E_ERR_ARITHMETIC	F	-30.003	16#8ACD	Desborde aritmético (%S18 establecido).
MULTIME	E_ARITHMETIC_ERROR_MUL_OV	F	-30.172	16#8A24	Error aritmético / Desborde de multiplicación.
MULTIME	E_ARITHMETIC_ERROR_ADD_OV	F	-30.173	16#8A23	Error aritmético / Desborde de adición.
MULTIME	E_ARITHMETIC_ERROR_BIG_PAR	F	-30171	16#8A25	Error aritmético / Los parámetros exceden el rango.
MULTIME	E_NEGATIVE_INPUT_FOR_TIME_OPERATION	F	-30.177	16#8A1F	Un valor negativo no se puede convertir en un tipo de dato <code>TIME</code> .
MULTIME	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.

Estadística

Tabla de códigos y valores de error creados para los EFB de la familia Estadística.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
AVE	E_INPUT_VALUE_OUT_OF_RANGE	F	-30.183	16#8A19	Valor de entrada fuera de rango.
AVE	E_DIVIDE_BY_ZERO	F	-30.176	16#8A20	División por cero.
AVE	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante, página 691</i> .
AVE	E_ARITHMETIC_ERROR	F	-30.170	16#8A26	Error aritmético
AVE	E_FP_STATUS_FAILED	F	-30.150	16#8A3A	Operación de coma flotante no permitida
AVE	E_ARITHMETIC_ERROR_MUL_OV	F	-30.172	16#8A24	Error aritmético / Desborde de multiplicación
AVE	E_ARITHMETIC_ERROR_ADD_OV	F	-30.173	16#8A23	Error aritmético / Desborde de adición
AVE	E_ARITHMETIC_ERROR_BIG_PAR	F	-30.171	16#8A25	Error aritmético / Los parámetros exceden el rango
AVE	E_ARITHMETIC_ERROR_UNSIGN_OV	F	-30.174	16#8A22	Error aritmético / Desborde sin signo
MAX	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante, página 691</i> .
MIN	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante, página 691</i> .
MUX	E_SELECTOR_OUT_OF_RANGE	F	-30.175	16#8A21	El selector está fuera de rango.

Tabla de códigos de error de la librería de diagnóstico

Introducción

Las siguientes tablas muestran los códigos y valores de error creados para los EFB de la librería de diagnóstico.

Diagnósticos

Tabla de códigos y valores de error creados para los EFB de la familia Diagnósticos.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
ONLEVT	E_EFB_ONLEVT	V/F	-30.196	16#8A0C	Error de EFB ONLEVT Estados ENO ● Verdadero = Registro de error correcto ● Falso = Registro de error fallido

Tablas de códigos de error de la librería de comunicación

Introducción

En las tablas siguientes se muestran los códigos y valores de error creados para los EFB de la librería de comunicación.

Extendido

Tabla de códigos y valores de error creados para los EFB de la familia `Extendido`.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en dec.	Valor de error en hex.	Descripción del error
CREAD_REG	E_EFB_MSTR_ERROR	F	-30.191	16#8A11	Error de comunicación MSTR.
CREAD_REG	E_EFB_NOT_STATE_RAM_4X	F	-30.531	16#88BD	Variable no asignada al campo % MW (4x).
CREAD_REG	-	F	8.195	16#2003	Valor mostrado en la palabra de estado. (Aparece con E_EFB_MSTR_ERROR.)
CREAD_REG	-	F	8.206	16#200E	Valor mostrado en la palabra de estado. Aparece con E_EFB_NOT_STATE_RAM_4X.
CREAD_REG	-	F	-	-	Consulte las tablas de: ● Códigos de error de Modbus Plus y SY/MAX EtherNet (véase <i>Unity Pro, Comunicación, Biblioteca de bloques</i>) ● Códigos de error específicos de SY/MAX (véase <i>Unity Pro, Comunicación, Biblioteca de bloques</i>) ● Códigos de error TCP/IP EtherNet (véase <i>Unity Pro, Comunicación, Biblioteca de bloques</i>)

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en dec.	Valor de error en hex.	Descripción del error
CWRITE_REG	E_EFB_MSTR_ERROR	F	-30.191	16#8A11	Error de comunicación MSTR.
CWRITE_REG	-	F	8.195	16#2003	Valor mostrado en la palabra de estado. Aparece con E_EFB_MSTR_ERROR.
CWRITE_REG	-	F	8.206	16#200E	Valor mostrado en la palabra de estado. Aparece con E_EFB_NOT_STATE_RAM_4X.
CWRITE_REG	-	F	-	-	Consulte las tablas de: ● Códigos de error de Modbus Plus y SY/MAX EtherNet (véase <i>Unity Pro, Comunicación, Biblioteca de bloques</i>) ● Códigos de error específicos de SY/MAX (véase <i>Unity Pro, Comunicación, Biblioteca de bloques</i>) ● Códigos de error TCP/IP EtherNet (véase <i>Unity Pro, Comunicación, Biblioteca de bloques</i>)
MBP_MSTR	E_EFB_OUT_OF_RANGE	F	-30.192	16#8A10	Error interno: EFB ha detectado una violación (por ejemplo, que la escritura sobrepasa los límites de %MW [4x])
MBP_MSTR	E_EFB_NOT_STATE_RAM_4X	F	-30.531	16#88BD	Variable no asignada al campo % MW (4x).
MBP_MSTR	-	F	8.195	16#2003	Valor mostrado en la palabra de estado. Aparece con E_EFB_MSTR_ERROR en el estado del bloque de control.
MBP_MSTR	-	F	8.206	16#200E	Valor mostrado en la palabra de estado. Aparece con E_EFB_NOT_STATE_RAM_4X en el estado del bloque de control.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en dec.	Valor de error en hex.	Descripción del error
MBP_MSTR	-	F	-	-	Consulte las tablas de: ● Códigos de error de Modbus Plus y SY/MAX EtherNet (véase <i>Unity Pro, Comunicación, Biblioteca de bloques</i>) ● Códigos de error específicos de SY/MAX (véase <i>Unity Pro, Comunicación, Biblioteca de bloques</i>) ● Códigos de error TCP/IP EtherNet (véase <i>Unity Pro, Comunicación, Biblioteca de bloques</i>)
READ_REG	W_WARN_OUT_OF_RANGE	F	30.110	16#759E	Parámetro fuera de rango.
READ_REG	E_EFB_NOT_STATE_RAM_4X	F	-30.531	16#88BD	Variable no asignada al campo % MW (4x).
READ_REG	E_EFB_MSTR_ERROR	F	-30.191	16#8A11	Error de comunicación MSTR.
READ_REG	-	F	8.195	16#2003	Valor mostrado en la palabra de estado. Aparece con W_WARN_OUT_OF_RANGE.
READ_REG	MBPUNLOC	F	8.206	16#200E	Valor mostrado en la palabra de estado. Aparece con E_EFB_NOT_STATE_RAM_4X.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en dec.	Valor de error en hex.	Descripción del error
READ_REG	-	F	-	-	Consulte las tablas de: ● Códigos de error de Modbus Plus y SY/MAX EtherNet (véase <i>Unity Pro, Comunicación, Biblioteca de bloques</i>) ● Códigos de error específicos de SY/MAX (véase <i>Unity Pro, Comunicación, Biblioteca de bloques</i>) ● Códigos de error TCP/IP EtherNet (véase <i>Unity Pro, Comunicación, Biblioteca de bloques</i>)
WRITE_REG	W_WARN_OUT_OF_RANGE	F	30.110	16#759E	Parámetro fuera de rango.
WRITE_REG	E_EFB_NOT_STATE_RAM_4X	F	-30.531	16#88BD	Variable no asignada al campo % MW (4x).
WRITE_REG	E_EFB_MSTR_ERROR	F	-30.191	16#8A11	Error de comunicación MSTR.
WRITE_REG	-	F	8.195	16#2003	Valor mostrado en la palabra de estado. Aparece con W_WARN_OUT_OF_RANGE.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en dec.	Valor de error en hex.	Descripción del error
WRITE_REG	-	F	8.206	16#200E	Valor mostrado en la palabra de estado. Aparece con E_EFB_NOT_STATE_RAM_4X.
WRITE_REG	-	F	-	-	Consulte las tablas de: ● Códigos de error de Modbus Plus y SY/MAX EtherNet (véase <i>Unity Pro, Comunicación, Biblioteca de bloques</i>) ● Códigos de error específicos de SY/MAX (véase <i>Unity Pro, Comunicación, Biblioteca de bloques</i>) ● Códigos de error TCP/IP EtherNet (véase <i>Unity Pro, Comunicación, Biblioteca de bloques</i>)

Tablas de códigos de error para la librería de gestión de E/S

Introducción

En las tablas siguientes se muestran los códigos y valores de error creados para los EFB de la librería de gestión de E/S.

Configuración de E/S analógicas

Tabla de códigos y valores de error creados para los EFB de la familia Configuración de E/S analógicas.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
I_FILTER	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
I_SET	E_EFB_USER_ERROR_1	F	-30200	16#8A08	La entrada IN_REG no está conectada con el número de una palabra de entrada (%IW).
I_SET	E_EFB_USER_ERROR_2	F	-30201	16#8A07	La entrada IN_REG está conectada con un número no válido de una palabra de entrada (%IW).
I_SET	E_EFB_USER_ERROR_3	F	-30202	16#8A06	MN_RAW MX_RAW
I_SET	E_EFB_USER_ERROR_4	F	-30203	16#8A05	Valor desconocido para MN_PHYS
I_SET	E_EFB_USER_ERROR_5	F	-30204	16#8A04	Valor desconocido para MX_PHYS
I_SET	E_EFB_USER_ERROR_11	F	-30210	16#89FE	ST_REG no introducido
I_SET	E_EFB_USER_ERROR_12	F	-30211	16#89FD	ST_REG demasiado largo

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
I_SET	E_EFB_USER_ERROR_13	F	-30212	16#89FC	ST_CH no introducido
O_FILTER	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
O_SET	E_EFB_USER_ERROR_1	F	-30200	16#8A08	La entrada OUT_REG no está conectada con el número de una palabra de salida (%MW).
O_SET	E_EFB_USER_ERROR_2	F	-30201	16#8A07	La entrada OUT_REG está conectada a un número no válido de una palabra de salida (%MW).
O_SET	E_EFB_USER_ERROR_3	F	-30202	16#8A06	MN_RAW MX_RAW
O_SET	E_EFB_USER_ERROR_4	F	-30203	16#8A05	Valor desconocido para MN_PHYS
O_SET	E_EFB_USER_ERROR_5	F	-30204	16#8A04	Valor desconocido para MX_PHYS
O_SET	E_EFB_USER_ERROR_11	F	-30210	16#89FE	ST_REG no introducido
O_SET	E_EFB_USER_ERROR_12	F	-30211	16#89FD	ST_REG demasiado largo
O_SET	E_EFB_USER_ERROR_13	F	-30212	16#89FC	ST_CH no introducido

Escalado de E/S analógicas

Tabla de códigos y valores de error creados para los EFB de la familia Escalado de E/S analógicas.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
I_NORM	E_EFB_NEG_OVER_RANGE	F	-30187	16#8A15	Desborde negativo
I_NORM	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
I_NORM_WARN	E_EFB_NO_WARNING_STATUS_AVAILABLE	F	-30189	16#8A13	El módulo no entrega estado de advertencia.
I_NORM_WARN	E_EFB_POS_OVER_RANGE	F	-30186	16#8A16	Desborde positivo
I_NORM_WARN	E_EFB_NEG_OVER_RANGE	F	-30187	16#8A15	Desborde negativo
I_NORM_WARN	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
I_PHYS	E_EFB_NO_WARNING_STATUS_AVAILABLE	F	-30189	16#8A13	El módulo no entrega estado de advertencia.
I_PHYS	E_INPUT_VALUE_OUT_OF_RANGE	F	-30183	16#8A19	Valor de entrada fuera de rango.
I_PHYS	E_EFB_NO_MEASURING_RANGE	F	-30185	16#8A17	Error interno
I_PHYS	E_EFB_POS_OVER_RANGE	F	-30186	16#8A16	Desborde positivo
I_PHYS	E_EFB_NEG_OVER_RANGE	F	-30187	16#8A15	Desborde negativo
I_PHYS	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
I_PHYS_WARN	E_EFB_NO_WARNING_STATUS_AVAILABLE	F	-30189	16#8A13	El módulo no entrega estado de advertencia.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
I_PHYS_WARN	E_EFB_FILTER_SQRT_NOT_AVAIL	F	-30195	16#8A0D	El filtro SQRT no está disponible.
I_PHYS_WARN	E_INPUT_VALUE_OUT_OF_RANGE	F	-30183	16#8A19	Valor de entrada fuera de rango.
I_PHYS_WARN	E_EFB_NO_MEASURING_RANGE	F	-30185	16#8A17	Error interno
I_PHYS_WARN	E_EFB_POS_OVER_RANGE	F	-30186	16#8A16	Desborde positivo
I_PHYS_WARN	E_EFB_NEG_OVER_RANGE	F	-30187	16#8A15	Desborde negativo
I_PHYS_WARN	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
I_RAW	E_EFB_OUT_OF_RANGE	F	-30192	16#8A10	Error interno: EFB ha detectado una violación (por ejemplo, que la escritura sobrepasa los límites de %MW [4x]).
I_RAW	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
I_RAWSIM	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
I_SCALE	E_EFB_POS_OVER_RANGE	F	-30186	16#8A16	Desborde positivo
I_SCALE	E_EFB_NEG_OVER_RANGE	F	-30187	16#8A15	Desborde negativo
I_SCALE	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
I_SCALE_WARN	E_EFB_NO_WARNING_STATUS_AVAILABLE	F	-30189	16#8A13	El módulo no entrega estado de advertencia.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
I_SCALE_WARN	E_EFB_POS_OVER_RANGE	F	-30186	16#8A16	Desborde positivo
I_SCALE_WARN	E_EFB_NEG_OVER_RANGE	F	-30187	16#8A15	Desborde negativo
I_SCALE_WARN	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
O_NORM	E_EFB_POS_OVER_RANGE	F	-30186	16#8A16	Desborde positivo
O_NORM	E_EFB_NEG_OVER_RANGE	F	-30187	16#8A15	Desborde negativo
O_NORM	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
O_NORM_WARN	E_EFB_POS_OVER_RANGE	F	-30186	16#8A16	Desborde positivo
O_NORM_WARN	E_EFB_NEG_OVER_RANGE	F	-30187	16#8A15	Desborde negativo
O_NORM_WARN	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
O_PHYS	E_EFB_NO_MEASURING_RANGE	F	-30185	16#8A17	Error interno
O_PHYS	E_EFB_POS_OVER_RANGE	F	-30186	16#8A16	Desborde positivo
O_PHYS	E_EFB_NEG_OVER_RANGE	F	-30187	16#8A15	Desborde negativo
O_PHYS	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
O_PHYS_WARN	E_EFB_NO_MEASURING_RANGE	F	-30185	16#8A17	Error interno
O_PHYS_WARN	E_EFB_POS_OVER_RANGE	F	-30186	16#8A16	Desborde positivo
O_PHYS_WARN	E_EFB_NEG_OVER_RANGE	F	-30187	16#8A15	Desborde negativo
O_PHYS_WARN	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
O_RAW	E_EFB_NEG_OVER_RANGE	F	-30187	16#8A15	Desborde negativo
O_RAW	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
O_SCALE	E_INPUT_VALUE_OUT_OF_RANGE	F	-30183	16#8A19	Valor de entrada fuera de rango.
O_SCALE	E_EFB_POS_OVER_RANGE	F	-30186	16#8A16	Desborde positivo
O_SCALE	E_EFB_NEG_OVER_RANGE	F	-30187	16#8A15	Desborde negativo
O_SCALE	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
O_SCALE_WARN	E_INPUT_VALUE_OUT_OF_RANGE	F	-30183	16#8A19	Valor de entrada fuera de rango.
O_SCALE_WARN	E_EFB_POS_OVER_RANGE	F	-30186	16#8A16	Desborde positivo

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
O_SCALE_WARN	E_EFB_NEG_OVER_RANGE	F	-30187	16#8A15	Desborde negativo
O_SCALE_WARN	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.

E/S inmediatas

Tabla de códigos y valores de error creados para los EFB de la familia E/S inmediatas.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción de errores
IMIO_IN	-	F	0000	0000	Operación correcta
IMIO_IN	-	F	8193	2001	Tipo de operación no válido (por ejemplo, el módulo de E/S direccionado no es un módulo de entrada).
IMIO_IN	-	F	8194	2002	Bastidor o número de slot no válido (la asignación de E/S en el configurador no contiene ninguna entrada de módulo para este slot).
IMIO_IN	-	F	8195	2003	Número de slot no válido
IMIO_IN	-	F	-4095	F001	El módulo no es correcto.
IMIO_OUT	-	F	0000	0000	Operación correcta
IMIO_OUT	-	F	8193	2001	Tipo de operación no válido (por ejemplo, el módulo de E/S direccionado no es un módulo de entrada).

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción de errores
IMIO_OUT	-	F	8194	2002	Bastidor o número de slot no válido (la asignación de E/S en el configurador no contiene ninguna entrada de módulo para este slot).
IMIO_OUT	-	F	8195	2003	Número de slot no válido
IMIO_OUT	-	F	-4095	F001	El módulo no es correcto.

Configuración de Quantum E/S

Tabla de códigos y valores de error creados para los EFB de la familia Configuración de Quantum E/S.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción de errores
ACI030	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
ACI040	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
ACI040	E_EFB_CURRENT_MODE_NOT_ALLOWED	F	-30197	16#8A0B	Error EFB: la modalidad actual no está permitida.
ACO020	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
ACO130	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
ACO130	E_EFB_CURRENT_MODE_NOT_ALLOWED	F	-30197	16#8A0B	Error EFB: la modalidad actual no está permitida.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción de errores
AI330	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
AI330	E_EFB_ILLEGAL_CONFIG_DATA	F	-30198	16#8A0A	Error EFB: datos de configuración no válidos.
AI33010	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
AI33010	E_EFB_CURRENT_MODE_NOT_ALLOWED	F	-30197	16#8A0B	Error EFB: la modalidad actual no está permitida.
AIO330	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
AIO330	E_EFB_CURRENT_MODE_NOT_ALLOWED	F	-30197	16#8A0B	Error EFB: la modalidad actual no está permitida.
AMM090	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
ARI030	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
ARI030	E_EFB_ILLEGAL_CONFIG_DATA	F	-30198	16#8A0A	Error EFB: datos de configuración no válidos.
ATI030	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
AVI030	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
AVO020	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción de errores
DROP	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
ERT_854_10	ES_WRONG_SLOT	F	20480	16#5000	-
ERT_854_10	E_WRONG_SLOT	F	-30215	16#89F9	Definido como E_EFB_USER_ERROR_16.
ERT_854_10	ES_HEALTHBIT	F	24576	16#6000	-
ERT_854_10	E_HEALTHBIT	F	-30216	16#89F8	Definido como E_EFB_USER_ERROR_17.
ERT_854_10	ES_TIMEOUT	F	32768	16#8000	-
ERT_854_10	E_TIMEOUT	F	-30210	16#89FE	Definido como E_EFB_USER_ERROR_11.
ERT_854_10	Valores E_ERT_BASIC	F	-30199	16#8A09	Definido como E_EFB_USER_ERROR_1 + 1.
ERT_854_10	E_WRONG_ANSW	F	-30211	16#89FD	Definido como E_EFB_USER_ERROR_12.
ERT_854_10	ES_CBUF_OFLOW	F	28672	16#7000	-
ERT_854_10	E_CBUF_OFLOW	F	-30217	16#89F7	Definido como E_EFB_USER_ERROR_18.
ERT_854_10	ES_WRONG_PAKET	F	8192	16#2000	-
ERT_854_10	E_WRONG_PAKET	F	-30212	16#89FC	Definido como E_EFB_USER_ERROR_13.
ERT_854_10	ES_WRONG_FELD	F	12288	16#3000	-
ERT_854_10	E_WRONG_FELD	F	-30213	16#89FB	Definido como E_EFB_USER_ERROR_14.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción de errores
QUANTUM	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
QUANTUM	E_EFB_UNKNOWN_DRIVER	F	-30190	16#8A12	Estación desconocida / Sin Traffic Cop Quantum
XBE	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.
XBE	E_EFB_UNKNOWN_DRIVER	F	-30190	16#8A12	Estación desconocida / Sin Traffic Cop Quantum
XDROP	E_EFB_NOT_CONFIGURED	F	-30188	16#8A14	La configuración EFB no coincide con la configuración del hardware.

NOTA: para obtener información sobre ERT_854_10, consulte la descripción de ERT_854_10 (véase *Unity Pro, Gestión de E/S, Biblioteca de bloques*) en la librería de gestión de E/S.

Tabla de códigos de error de la librería CONT_CTL

Introducción

Las siguientes tablas muestran los códigos y valores de error creados para los EFB de la librería CONT_CTL.

Condicionamiento

Tabla de códigos y valores de error creados para los EFB de la familia Condicionamiento.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
DTIME	W_WARN_OUT_OF_RANGE	V	30.110	16#759E	Parámetro fuera de rango.
DTIME	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
DTIME	Valores de palabra de estado	V/F	-	-	Para obtener información detallada sobre la palabra de estado DTIME, consulte la descripción de DTIME (véase <i>Unity Pro, Control, Librería de bloques</i>).
INTEGRATOR	E_ERR_DEN	F	-30.152	16#8A38	El número de coma flotante no es válido.
INTEGRATOR	E_ERR_IB_MAX_MIN	F	-30.102	16#8A6A	YMAX < YMIN
INTEGRATOR	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
LAG_FILTER	E_ERR_DEN	F	-30.152	16#8A38	El número de coma flotante no es válido.
LAG_FILTER	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
LDLG	E_ERR_DEN	F	-30.152	16#8A38	El número de coma flotante no es válido.
LDLG	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
LEAD	E_ERR_DEN	F	-30.152	16#8A38	El número de coma flotante no es válido.
LEAD	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
MFLOW	W_WARN_OUT_OF_RANGE	V	30.110	16#759E	Parámetro fuera de rango.
MFLOW	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
MFLOW	Valores de palabra de estado	V/F	-	-	Para obtener información detallada sobre la palabra de estado MFLOW , consulte la descripción de MFLOW (véase <i>Unity Pro, Control, Librería de bloques</i>).
QDTIME	E_ERR_DEN	F	-30.152	16#8A38	El número de coma flotante no es válido.
SCALING	E_ERR_NULL_INPUT_SCALE	F	-30.121	16#8A57	Sin escala de entrada: los límites Máx. y Mín. tienen que ser diferentes.
SCALING	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
SCALING	Valores de palabra de estado	V/F	-	-	Para obtener información detallada sobre la palabra de estado SCALING , consulte la descripción de SCALING (véase <i>Unity Pro, Control, Librería de bloques</i>).

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
TOTALIZER	W_WARN_OUT_OF_RANGE	V	30.110	16#759E	Parámetro fuera de rango.
TOTALIZER	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
TOTALIZER	W_WARN_TOTALIZER_CTER_MAX	V	30.113	16#75A1	Se alcanzó valor máximo de ctr.
TOTALIZER	Valores de palabra de estado	V/F	-	-	Para obtener información detallada sobre la palabra de estado TOTALIZER , consulte la descripción de TOTALIZER (véase <i>Unity Pro, Control, Librería de bloques</i>).
VEL_LIM	E_ERR_DEN	F	-30.152	16#8A38	El número de coma flotante no es válido.
VEL_LIM	E_ERR_AB1_MAX_MIN	F	-30.101	16#8A6B	YMAX < YMIN
VEL_LIM	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.

Controlador

Tabla de códigos y valores de error creados para los EFB de la familia
Controlador.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
AUTOTUNE	W_WARN_OUT_OF_RANGE	V	30.110	16#759E	Parámetro fuera de rango.
AUTOTUNE	E_ERR_NULL_INPUT_SCALE	F	-30.121	16#8A57	Sin escala de entrada: los límites Máx. y Mín. tienen que ser diferentes.
AUTOTUNE	W_WARN_AUTOTUNE_FAILED	V	30.111	16#759F	Fallo en AUTOTUNE
AUTOTUNE	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
AUTOTUNE	E_ERR_AUTOTUNE_ID_UNKNOWN	F	-30.120	16#8A58	No se acepta aquí el EFB ajustado o aún no fue llamado.
AUTOTUNE	Valores de palabra de estado	V/F	-	-	Para obtener información detallada sobre la palabra de estado AUTOTUNE, consulte la descripción de AUTOTUNE (véase <i>Unity Pro, Control, Librería de bloques</i>).
PI_B	W_WARN_OUT_OF_RANGE	V	30.110	16#759E	Parámetro fuera de rango.
PI_B	E_ERR_NULL_INPUT_SCALE	F	-30.121	16#8A57	Sin escala de entrada: los límites Máx. y Mín. tienen que ser diferentes.
PI_B	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
PI_B	Valores de palabra de estado	V/F	-	-	Para obtener información detallada sobre la palabra de estado PI_B consulte la descripción de PI_B (véase <i>Unity Pro, Control, Librería de bloques</i>).

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
PIDFF	W_WARN_OUT_OF_RANGE	V	30.110	16#759E	Parámetro fuera de rango.
PIDFF	E_ERR_NULL_INPUT_SCALE	F	-30.121	16#8A57	Sin escala de entrada: los límites Máx. y Mín. tienen que ser diferentes.
PIDFF	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
PIDFF	Valores de palabra de estado	V/F	-	-	Para obtener información detallada sobre la palabra de estado <code>PIDFF</code> , consulte la descripción de <code>PIDFF</code> (véase <i>Unity Pro, Control, Librería de bloques</i>).
SAMPLETM	E_EFB_SAMPLE_TIME_OVERFLOW	F	-30.184	16#8A18	Error interno
STEP2	W_WARN_OUT_OF_RANGE	V	30.110	16#759E	Parámetro fuera de rango.
STEP2	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
STEP2	Valores de palabra de estado	V/F	-	-	Para obtener información detallada sobre la palabra de estado <code>STEP2</code> , consulte la descripción de <code>STEP2</code> (véase <i>Unity Pro, Control, Librería de bloques</i>).
STEP3	W_WARN_OUT_OF_RANGE	V	30.110	16#759E	Parámetro fuera de rango.
STEP3	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
STEP3	Valores de palabra de estado	V/F	-	-	Para obtener información detallada sobre la palabra de estado <code>STEP3</code> , consulte la descripción de <code>STEP3</code> (véase <i>Unity Pro, Control, Librería de bloques</i>).

Matemáticas

Tabla de códigos y valores de error creados para los EFB de la familia Matemáticas.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
COMP_DB	W_WARN_OUT_OF_RANGE	V	30.110	16#759E	Parámetro fuera de rango.
COMP_DB	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
K_SQRT	W_WARN_OUT_OF_RANGE	V	30.110	16#759E	Parámetro fuera de rango.
K_SQRT	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
MULDIV_W	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
SUM_W	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.

Medición

Tabla de códigos y valores de error creados para los EFB de la familia Medición.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
AVGMV	E_ERR_DEN	F	-30.152	16#8A38	El número de coma flotante no es válido.
AVGMV	W_WARN_AVGMV	V	30.108	16#759C	AVGMV: N tiene que ser <= 50

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
AVGMV	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
AVGMV_K	E_ERR_DEN	F	-30.152	16#8A38	El número de coma flotante no es válido.
AVGMV_K	W_WARN_AVGMV_K	V	30.109	16#759D	AVGMV_K: N tiene que ser ≤ 10.000
AVGMV_K	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
DEAD_ZONE	E_ERR_DEN	F	-30.152	16#8A38	El número de coma flotante no es válido.
DEAD_ZONE	E_ERR_DZONE	F	-30.119	16#8A59	DZONE: DZ tiene que ser ≥ 0
DEAD_ZONE	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
LOOKUP_TABL E1	E_ERR_DEN	F	-30.152	16#8A38	El número de coma flotante no es válido.
LOOKUP_TABL E1	E_ERR_POLY_ANZAHL	F	-30.107	16#8A65	La cantidad de entradas no es par.
LOOKUP_TABL E1	E_ERR_POLY_FOLGE	F	-30.108	16#8A64	Coordenadas de referencia $x(i) \leq x(i-1)$
LOOKUP_TABL E1	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.

Procesamiento de salida

Tabla de códigos y valores de error creados para los EFB de la familia
Procesamiento de salida.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
MS	W_WARN_OUT_OF_RANGE	V	30.110	16#759E	Parámetro fuera de rango.
MS	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
MS	Valores de palabra de estado	V/F	-	-	Para obtener información detallada sobre la palabra de estado MS , consulte la descripción de MS (véase <i>Unity Pro, Control, Librería de bloques</i>).
PWM1	WAF_PBM_TMINMAX	F	-30.113	16#8A5F	$t_{min} < t_{max}$
PWM1	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
SERVO	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
SERVO	Valores de palabra de estado	V/F	-	-	Para obtener información detallada sobre la palabra de estado SERVO , consulte la descripción de SERVO (véase <i>Unity Pro, Control, Librería de bloques</i>).
SPLRG	W_WARN_OUT_OF_RANGE	V	30.110	16#759E	Parámetro fuera de rango.
SPLRG	E_ERR_NULL_INPUT_SCALE	F	-30.121	16#8A57	Sin escala de entrada: los límites Máx. y Mín. tienen que ser diferentes.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
SPLRG	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
SPLRG	Valores de palabra de estado	V/F	-	-	Para obtener información detallada sobre la palabra de estado SPLRG , consulte la descripción de SPLRG (véase <i>Unity Pro, Control, Librería de bloques</i>).

Gestión de consigna

Tabla de códigos y valores de error creados para los EFB de la familia *Gestión de consigna*.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
RAMP	W_WARN_OUT_OF_RANGE	V	30.110	16#759E	Parámetro fuera de rango.
RAMP	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
RAMP	Valores de palabra de estado	V/F	-	-	Para obtener información detallada sobre la palabra de estado RAMP , consulte la descripción de RAMP (véase <i>Unity Pro, Control, Librería de bloques</i>).
RATIO	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
RATIO	Valores de palabra de estado	V/F	-	-	Para obtener información detallada sobre la palabra de estado RATIO , consulte la descripción de RATIO (véase <i>Unity Pro, Control, Librería de bloques</i>).

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
SP_SEL	W_WARN_OUT_OF_RANGE	V	30.110	16#759E	Parámetro fuera de rango.
SP_SEL	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
SP_SEL	Valores de palabra de estado	V/F	-	-	Para obtener información detallada sobre la palabra de estado SP_SEL, consulte la descripción de SP_SEL (véase <i>Unity Pro, Control, Librería de bloques</i>).

Tabla de códigos de error de la biblioteca de movimiento

Introducción

Las siguientes tablas muestran los códigos y valores de error creados para los EFB de la biblioteca de movimiento.

Inicio MMF

Tabla de códigos y valores de error creados para los EFB de la familia Inicio MMF.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
CFG_CP_F	BAD_REVISION	F	-30.200	16#8A08	Definido como E_EFB_USER_ERROR_1
CFG_CP_F	MMF_BAD_4X	V	9.010	16#2332	-
CFG_CP_F	MMF_ABORT_SUB	V	7.004	16#1B5C	Error de protocolo SubNum/SubNumEcho
CFG_CP_V	BAD_REVISION	F	-30.200	16#8A08	Definido como E_EFB_USER_ERROR_1
CFG_CP_V	MMF_BAD_4X	V	9.010	16#2332	-
CFG_CP_V	MMF_ABORT_SUB	V	7.004	16#1B5C	Error de protocolo SubNum/SubNumEcho
CFG_CS	BAD_REVISION	F	-30.200	16#8A08	Definido como E_EFB_USER_ERROR_1
CFG_CS	MMF_ABORT_SUB	V	7.004	16#1B5C	Error de protocolo SubNum/SubNumEcho
CFG_FS	BAD_REVISION	F	-30.200	16#8A08	Definido como E_EFB_USER_ERROR_1
CFG_FS	MMF_ABORT_SUB	V	7.004	16#1B5C	Error de protocolo SubNum/SubNumEcho

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
CFG_IA	BAD_REVISION	F	-30.200	16#8A08	Definido como E_EFB_USER_ERROR_1
CFG_IA	MMF_ABORT_SUB	V	7.004	16#1B5C	Error de protocolo SubNum/SubNumEcho
CFG_RA	BAD_REVISION	F	-30.200	16#8A08	Definido como E_EFB_USER_ERROR_1
CFG_RA	MMF_ABORT_SUB	V	7.004	16#1B5C	Error de protocolo SubNum/SubNumEcho
CFG_SA	BAD_REVISION	F	-30.200	16#8A08	Definido como E_EFB_USER_ERROR_1
CFG_SA	MMF_ABORT_SUB	V	7.004	16#1B5C	Error de protocolo SubNum/SubNumEcho
DRV_DNLD	BAD_REVISION	F	-30.200	16#8A08	Definido como E_EFB_USER_ERROR_1
DRV_DNLD	MMF_ABORT_SUB	V	7.004	16#1B5C	Error de protocolo SubNum/SubNumEcho
DRV_UPLD	BAD_REVISION	F	-30.200	16#8A08	Definido como E_EFB_USER_ERROR_1
DRV_UPLD	MMF_ABORT_SUB	V	7.004	16#1B5C	Error de protocolo SubNum/SubNumEcho
IDN_CHK	BAD_REVISION	F	-30.200	16#8A08	Definido como E_EFB_USER_ERROR_1
IDN_CHK	MMF_ABORT_SUB	V	7.004	16#1B5C	Error de protocolo SubNum/SubNumEcho
IDN_XFER	BAD_REVISION	F	-30.200	16#8A08	Definido como E_EFB_USER_ERROR_1

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
IDN_XFER	MMF_ABORT_SUB	V	7.004	16#1B5C	Error de protocolo SubNum/SubNumEcho
MMF_BITS	BAD_REVISION	F	-30.200	16#8A08	Definido como E_EFB_USER_ERROR_1
MMF_ESUB	BAD_REVISION	F	-30.200	16#8A08	Definido como E_EFB_USER_ERROR_1
MMF_ESUB	MMF_ABORT_SUB	V	7.004	16#1B5C	Error de protocolo SubNum/SubNumEcho
MMF_IDNX	BAD_REVISION	F	-30.200	16#8A08	Definido como E_EFB_USER_ERROR_1
MMF_IDNX	MMF_ABORT_SUB	V	7.004	16#1B5C	Error de protocolo SubNum/SubNumEcho
MMF_JOG	BAD_REVISION	F	-30.200	16#8A08	Definido como E_EFB_USER_ERROR_1
MMF_JOG	MMF_ABORT_SUB	V	7.004	16#1B5C	Error de protocolo SubNum/SubNumEcho
MMF_JOG	MMF_SUB_TIMEOUT	V	7.005	16#1B5D	La subrutina no se completa a tiempo
MMF_MOVE	BAD_REVISION	F	-30.200	16#8A08	Definido como E_EFB_USER_ERROR_1
MMF_MOVE	MMF_ABORT_SUB	V	7.004	16#1B5C	Error de protocolo SubNum/SubNumEcho
MMF_RST	BAD_REVISION	F	-30.200	16#8A08	Definido como E_EFB_USER_ERROR_1
MMF_SUB	BAD_REVISION	F	-30.200	16#8A08	Definido como E_EFB_USER_ERROR_1

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
MMF_SUB	MMF_ABORT_SUB	V	7.004	16#1B5C	Error de protocolo SubNum/SubNumEcho
MMF_USUB	BAD_REVISION	F	-30.200	16#8A08	Definido como E_EFB_USER_ERROR_1
MMF_USUB	MMF_ABORT_SUB	V	7.004	16#1B5C	Error de protocolo SubNum/SubNumEcho

NOTA: Para obtener detalles acerca de los códigos de error y los valores de error MMF, consulte la descripción de Notificación de errores y fallos (véase *Unity Pro, Control de unidad, Librería de bloques*) en la librería de movimiento.

Tablas de códigos de error de la biblioteca obsoleta

Introducción

En las tablas siguientes se muestran los códigos y valores de error creados para los EFB de la biblioteca obsoleta.

CLC

Tabla de códigos y valores de error creados para los EFB de la familia CLC.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
DELAY	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
INTEGRATOR1	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
INTEGRATOR1	E_ERR_IB_MAX_MIN	F	-30102	16#8A6A	YMAX < YMIN
INTEGRATOR1	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
LAG1	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
LAG1	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
LEAD_LAG1	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
LEAD_LAG1	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
LIMV	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
LIMV	E_ERR_AB1_MAX_MIN	F	-30101	16#8A6B	YMAX < YMIN
LIMV	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante, página 691</i> .
PI1	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
PI1	E_ERR_PI_MAX_MIN	F	-30103	16#8A69	YMAX < YMIN
PI1	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante, página 691</i> .
PID1	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
PID1	E_ERR_PID_MAX_MIN	F	-30104	16#8A68	YMAX < YMIN
PID1	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante, página 691</i> .
PIDP1	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
PIDP1	E_ERR_PID_MAX_MIN	F	-30104	16#8A68	YMAX < YMIN
PIDP1	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante, página 691</i> .
SMOOTH_RATE	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
SMOOTH_RATE	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante, página 691</i> .

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
THREE_STEP_CON1	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
THREE_STEP_CON1	W_WARN_DSR_TN	V	30101	16#7595	TN = 0
THREE_STEP_CON1	W_WARN_DSR_TSN	V	30102	16#7596	TSN = 0
THREE_STEP_CON1	W_WARN_DSR_KP	V	30103	16#7597	KP <= 0
THREE_STEP_CON1	E_ERR_DSR_HYS	F	-30105	16#8A67	$2 * UZ < HYS $
THREE_STEP_CON1	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
THREEPOINT_CON1	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
THREEPOINT_CON1	W_WARN_ZDR_XRR	F	30105	16#7599	DR: $XRR < -100$ o $XRR > 100$
THREEPOINT_CON1	W_WARN_ZDR_T1T2	F	30104	16#7598	$T2 > T1$
THREEPOINT_CON1	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
THREEPOINT_CON1	E_ERR_ZDR_HYS	F	-30106	16#8A66	$2 * UZ < HYS $
TWOPOINT_CON1	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
TWOPOINT_CON1	W_WARN_ZDR_XRR	F	30105	16#7599	DR: XRR < -100 o XRR > 100
TWOPOINT_CON1	W_WARN_ZDR_T1T2	F	30104	16#7598	T2 > T1
TWOPOINT_CON1	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
TWOPOINT_CON1	E_ERR_ZDR_HYS	F	-30106	16#8A66	$2 * UZ < HYS $

CLC_PRO

Tabla de códigos y valores de error creados para los EFB de la familia CLC_PRO.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
ALIM	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
ALIM	WAF_AB2_VMAX	F	-30111	16#8A61	$vmáx \leq 0$
ALIM	WAF_AB2_BMAX	F	-30112	16#8A60	$bmáx \leq 0$
ALIM	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
COMP_PID	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
COMP_PID	WAF_KPID_KUZ	F	-30110	16#8A62	gain_red < 0 o gain_red > 1
COMP_PID	WAF_KPID_OGUG	F	-30104	16#8A68	YMAX < YMIN
COMP_PID	WAF_KPID_UZ	F	-30109	16#8A63	db < 0
COMP_PID	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante, página 691</i> .
DEADTIME	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
DERIV	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
DERIV	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante, página 691</i> .
FGEN	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
FGEN	WAF_SIG_TV_MAX	F	-30116	16#8A5C	t_acc > t_rise / 2
FGEN	WAF_SIG_TH_MAX	F	-30117	16#8A5B	t_rise demasiado grande
FGEN	WAF_SIG_TA_MAX	V	30106	16#759A	t_off >= Medio período
FGEN	WAF_SIG_T1_MIN	V	30107	16#759B	t_max <= t_min
FGEN	WAF_SIG_FKT	F	-30118	16#8A5A	func_no <= 0 o func_no > 8

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
FGEN	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
INTEG	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
INTEG	E_ERR_IB_MAX_MIN	F	-30102	16#8A6A	YMAX < YMIN
INTEG	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
LAG	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
LAG	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
LAG2	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
LAG2	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
LEAD_LAG	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
LEAD_LAG	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
PCON2	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
PCON2	W_WARN_ZDR_XRR	V	30105	16#7599	DR: XRR < -100 o XRR > 100
PCON2	W_WARN_ZDR_T1T2	V	30104	16#7598	T2 > T1

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
PCON2	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante, página 691</i> .
PCON2	E_ERR_ZDR_HYS	F	-30106	16#8A66	$2 * UZ < HYS $
PCON3	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
PCON3	W_WARN_ZDR_XRR	V	30105	16#7599	DR: $XRR < -100$ o $XRR > 100$
PCON3	W_WARN_ZDR_T1T2	V	30104	16#7598	$T2 > T1$
PCON3	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante, página 691</i> .
PCON3	E_ERR_ZDR_HYS	F	-30106	16#8A66	$2 * UZ < HYS $
PD_OR_PI	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
PD_OR_PI	WAF_PDPI_OG_UG	F	-30103	16#8A69	$YMAX < YMIN$
PD_OR_PI	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante, página 691</i> .
PDM	PDM_TMAX_TMIN	F	-30115	16#8A5D	$t_{max} \leq t_{min}$
PDM	PDM_OG_UG	F	-30114	16#8A69	$ pos_{up_x} > pos_{lo_x} $ o $ neg_{up_x} > neg_{lo_x} $
PDM	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante, página 691</i> .

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
PI	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
PI	E_ERR_PI_MAX_MIN	F	-30103	16#8A69	YMAX < YMIN
PI	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
PID	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
PID	E_ERR_PID_MAX_MIN	F	-30104	16#8A68	YMAX < YMIN
PID	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
PID_P	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
PID_P	E_ERR_PID_MAX_MIN	F	-30104	16#8A68	YMAX < YMIN
PID_P	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
PIP	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
PIP	E_ERR_PI_MAX_MIN	F	-30103	16#8A69	YMAX < YMIN
PIP	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
PPI	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
PPI	E_ERR_PI_MAX_MIN	F	-30103	16#8A69	YMAX < YMIN
PPI	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
PWM	WAF_PBM_TMINMAX	F	-30113	16#8A5F	$t_{\min} < t_{\max}$
PWM	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
QPWM	WAF_PBM_TMINMAX	F	-30113	16#8A5F	$t_{\min} < t_{\max}$
QPWM	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
SCON3	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
SCON3	W_WARN_DSR_TN	V	30101	16#7595	TN = 0
SCON3	W_WARN_DSR_TSN	V	30102	16#7596	TSN = 0
SCON3	W_WARN_DSR_KP	V	30103	16#7597	KP ≤ 0
SCON3	E_ERR_DSR_HYS	F	-30105	16#8A67	$2 * UZ < HYS $

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
SCON3	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.
VLIM	E_ERR_DEN	F	-30152	16#8A38	El número de coma flotante no es válido.
VLIM	E_ERR_AB1_MAX_MIN	F	-30101	16#8A6B	YMAX < YMIN
VLIM	FP_ERROR	F	-	-	Véase la tabla <i>Errores comunes de coma flotante</i> , página 691.

Extensión/Compatibilidad

Tabla de códigos y valores de error creados para los EFB de la familia Extensión/Compatibilidad.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
AKF_TA	E_AKFEFB_TIMEBASE_IS_ZERO	F	-30482	16#88EE	Base de tiempo es cero.
AKF_TE	E_AKFEFB_TIMEBASE_IS_ZERO	F	-30482	16#88EE	Base de tiempo es cero.
AKF_TI	E_AKFEFB_TIMEBASE_IS_ZERO	F	-30482	16#88EE	Base de tiempo es cero.
AKF_TS	E_AKFEFB_TIMEBASE_IS_ZERO	F	-30482	16#88EE	Base de tiempo es cero.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
AKF_TV	E_AKFEFB_TIMEBASE_IS_ZERO	F	-30482	16#88EE	Base de tiempo es cero.
FIFO	E_INPUT_VALUE_OUT_OF_RANGE	F	-30183	16#8A19	Valor de entrada fuera de rango.
GET_3X	E_INPUT_VALUE_OUT_OF_RANGE	F	-30183	16#8A19	Valor de entrada fuera de rango.
GET_4X	E_INPUT_VALUE_OUT_OF_RANGE	F	-30183	16#8A19	Valor de entrada fuera de rango.
GET_BIT	E_INPUT_VALUE_OUT_OF_RANGE	F	-30183	16#8A19	Valor de entrada fuera de rango.
IEC_BMDI	E_EFB_USER_ERROR_1	F	-30200	16#8A08	El valor de entrada es un tipo de registro inválido (SourceTable).
IEC_BMDI	E_EFB_USER_ERROR_2	F	-30201	16#8A07	El offset de entrada (OffsetInSourceTable) selecciona una dirección que excede los límites aceptables.
IEC_BMDI	E_EFB_USER_ERROR_3	F	-30202	16#8A06	El offset de entrada (OFF_IN) no es 1 o un múltiplo de 16+1.
IEC_BMDI	E_EFB_USER_ERROR_4	F	-30203	16#8A05	El valor de salida es un tipo de registro inválido (DestinationTable).
IEC_BMDI	E_EFB_USER_ERROR_5	F	-30204	16#8A04	El offset de salida (OffsetInDestinationTable) selecciona una dirección que excede los límites aceptables.
IEC_BMDI	E_EFB_USER_ERROR_6	F	-30205	16#8A03	El offset de salida (OffsetInDestinationTable) no es 1 o un múltiplo de 16+1.

Nombre de EFB	Código de error	Estado de ENO en caso de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
IEC_BMDI	E_EFB_USER_ERROR_7	F	-30206	16#8A02	El valor de (NumberOfElements) es 0.
IEC_BMDI	E_EFB_USER_ERROR_8	F	-30207	16#8A01	El valor de (NumberOfElements) hace referencia a más de 1.600 bits.
IEC_BMDI	E_EFB_USER_ERROR_9	F	-30208	16#8A00	El valor de (NumberOfElements) hace referencia a más de 100 palabras.
IEC_BMDI	E_EFB_USER_ERROR_10	F	-30209	16#89FF	El valor de (NumberOfElements) selecciona una dirección de fuente que excede los límites aceptables.
IEC_BMDI	E_EFB_USER_ERROR_11	F	-30210	16#89FE	El valor de (NumberOfElements) selecciona una dirección de destino que excede los límites aceptables.
IEC_BMDI	E_EFB_USER_ERROR_12	F	-30211	16#89FD	El valor de (NumberOfElements) no es un múltiplo de 16.
IEC_BMDI	E_EFB_USER_ERROR_13	F	-30212	16#89FC	Advertencia: Solapamiento de direcciones de entrada y salida.
LIFO	E_INPUT_VALUE_OUT_OF_RANGE	F	-30183	16#8A19	Valor de entrada fuera de rango.
PUT_4X	E_INPUT_VALUE_OUT_OF_RANGE	F	-30183	16#8A19	Valor de entrada fuera de rango.
MUX_DINTARR_125	E_SELECTOR_OUT_OF_RANGE	F	-30175	16#8A21	El selector está fuera de rango.
SET_BIT	E_INPUT_VALUE_OUT_OF_RANGE	F	-30183	16#8A19	Valor de entrada fuera de rango.

Errores comunes de coma flotante

Introducción

En la tabla siguiente se muestran los códigos y valores de error comunes creados para los errores de coma flotante.

Errores comunes de coma flotante

Tabla de errores comunes de coma flotante

Códigos de error	Valor de error en Dec	Valor de error en Hex	Descripción del error
FP_ERROR	-30150	16#8A3A	Valor de base (no aparece como valor de error)
E_FP_STATUS_FAILED_IE	-30151	16#8A39	Operación de coma flotante ilegal
E_FP_STATUS_FAILED_IE	-30152	16#8A38	El operando no está normalizado, no es un número REAL válido
E_FP_STATUS_FAILED_ZE	-30154	16#8A36	División por cero ilegal
E_FP_STATUS_FAILED_ZE_IE	-30155	16#8A35	Operación de coma flotante ilegal / División por cero
E_FP_STATUS_FAILED_OE	-30158	16#8A32	Rebasamiento de coma flotante
E_FP_STATUS_FAILED_OE_IE	-30159	16#8A31	Operación de coma flotante ilegal / Rebasamiento
E_FP_STATUS_FAILED_OE_ZE	-30162	16#8A2E	Rebasamiento de coma flotante / División por cero
E_FP_STATUS_FAILED_OE_ZE_I E	-30163	16#8A2D	Operación de coma flotante ilegal / Rebasamiento / División por cero
E_FP_NOT_COMPARABLE	-30166	16#8A2A	Error interno

Conformidad con IEC

B

Descripción general

Este capítulo contiene las tablas de conformidad requerida por la EIC 61131-3.

Contenido de este capítulo

Este capítulo contiene las siguientes secciones:

Sección	Apartado	Página
B.1	Información general relativa a la norma IEC 61131-3	694
B.2	Tablas de conformidad con las normas IEC	696
B.3	Extensiones de la norma IEC 61131-3	722
B.4	Sintaxis de lenguajes textuales	724

B.1 Información general relativa a la norma IEC 61131-3

Información general relativa a la conformidad con la norma IEC 61131-3

Presentación

La norma IEC 61131-3 (cf. la subcláusula 1.4) especifica la sintaxis y semántica de un conjunto unificado de lenguajes de programación para controladores programables. Éstos controladores están compuestos por dos lenguajes textuales, IL (lista de instrucciones) y ST (Texto estructurado), y dos lenguajes gráficos, LD (diagrama de contactos) y FBD (diagrama de bloques de función).

Además, los elementos del lenguaje de gráfica de función secuencial (SFC) se definen para estructurar la organización interna de los programas de controladores programables y los bloques de función. También se definen los elementos de configuración, que admiten la instalación de programas de controladores programables en los sistemas de estos últimos.

NOTA: Unity Pro utiliza las siglas inglesas para los lenguajes de programación.

Asimismo, se definen las funciones que facilitan la comunicación entre controladores programables y otros componentes de los sistemas automatizados.

Conformidad de Unity Pro con la norma IEC 61131-3

La versión actual del sistema de programación Unity Pro admite un subconjunto compatible de los elementos de lenguaje definidos en la norma.

En este caso, compatible quiere decir lo siguiente:

- La norma permite al encargado de implementar un sistema de programación IEC elegir o cerrar las funciones de lenguaje específicas o incluso completar lenguajes fuera de las tablas de función que forman parte inherente de las especificaciones; un sistema que solicite conformidad con la norma debe ejecutar las funciones elegidas de acuerdo con las especificaciones de la norma.
- Además, la norma permite al encargado mencionado con anterioridad utilizar los elementos del lenguaje de programación definido en un entorno de programación interactivo. Debido a que la norma afirma explícitamente que la especificación de dichos entornos no está dentro de su competencia, dicho encargado posee cierta libertad para proporcionar una presentación optimizada y procedimientos de manipulación para elementos de lenguaje específicos en beneficio del usuario.
- Unity Pro utiliza esta libertad mediante, por ejemplo, la introducción de la noción "Proyecto" para la manipulación combinada de los elementos de lenguaje IEC "Configuración" y "Recurso". Igualmente, hace uso de su libertad, por ejemplo, en los mecanismos proporcionados para la manipulación de declaraciones de variable o las instanciaciones de bloques de funciones.

Tablas de las normas IEC

Las funciones admitidas y demás información específica de implementación se proporcionan en la declaración de conformidad y tablas siguientes tal y como requiere la norma.

B.2 Tablas de conformidad con las normas IEC

Descripción general

Este sistema cumple los requisitos de la norma IEC 61131-3 para el lenguaje y las funciones que se muestran en las siguientes tablas.

Contenido de esta sección

Esta sección contiene los siguientes apartados:

Apartado	Página
Elementos comunes	697
Elementos del lenguaje IL	710
Elementos de lenguaje ST	712
Elementos gráficos comunes	714
Elementos del lenguaje LD	715
Parámetros dependientes del tipo de implementación	716
Condiciones de error	720

Elementos comunes

Elementos comunes

Tabla de conformidad con IEC para elementos comunes:

N.º de tabla	N.º de función	Descripción de la función
1	2	Caracteres en minúscula
	3a	Signo de número (#)
	4a	Símbolo del dólar (\$)
	5a	Barra vertical ()
2	1	Mayúsculas y números
	2	Mayúsculas y minúsculas, números, guiones bajos
	3	Mayúsculas y minúsculas, números, guiones altos o bajos
3	1	Comentarios
3a	1	Construcciones pragmáticas
4	1	Literales enteros
	2	Literales reales
	3	Literales reales con exponentes
	4	Literales de base 2
	5	Literales de base 8
	6	Literales de base 16
	7	Cero y uno booleanos
	8	FALSE y TRUE
	9	Literales tipados
5	1	Cadena de caracteres de un solo byte
	3	Literales de cadena tipada de un solo byte
6	2	Símbolo del dólar
	3	Comilla simple
	4	Avance de línea
	5	Línea nueva
	6	Alimentación de formulario (página)
	7	Retorno de carro
	8	Tabulación
	9	Comillas dobles

N.º de tabla	N.º de función	Descripción de la función
7	1a	Literales de duración sin subrayado: prefijo corto
	1b	Prefijo largo
	2a	Literales de duración con subrayado: prefijo corto
	2b	Prefijo largo
8	1	Literales de fecha (prefijo largo)
	2	Literales de fecha (prefijo corto)
	3	Literales de hora del día (prefijo largo)
	4	Literales de hora del día (prefijo corto)
	5	Literales de hora y fecha (prefijo largo)
	5	Literales de hora y fecha (prefijo corto)
10	1	Tipo de datos <code>BOOL</code>
	3	Tipo de datos <code>INT</code>
	4	Tipo de datos <code>DINT</code>
	7	Tipo de datos <code>UINT</code>
	8	Tipo de datos <code>UDINT</code>
	10	Tipo de datos <code>REAL</code>
	12	Tipo de datos <code>TIME</code>
	13	Tipo de datos <code>DATE</code>
	14	Tipo de datos <code>TIME_OF_DAY</code> o <code>TOD</code>
	15	Tipo de datos <code>DATE_AND_TIME</code> o <code>DT</code>
	16	Tipo de datos <code>STRING</code>
	17	Tipo de datos <code>BYTE</code>
	18	Tipo de datos <code>WORD</code>
	19	Tipo de datos <code>DWORD</code>
12	4	Tipos de datos de matriz
	5	Tipos de datos estructurados
14	4	Inicialización de tipos de datos de matriz
	6	Inicialización de tipos de datos estructurados derivados

N.º de tabla	N.º de función	Descripción de la función
15	1	Ubicación de entrada
	2	Ubicación de salida
	3	Ubicación de memoria
	4	Tamaño de bit único (prefijo X)
	5	Tamaño de bit único (n.º de prefijo)
	7	Tamaño de palabra (16 bits)
	8	Tamaño de palabra doble (32 bits)
	9	Tamaño (64 bits) de palabra (cuádruple)
17	3	Declaración de ubicaciones de variables simbólicas (<i>Nota 5., página 706</i>)
	4	Asignación de ubicación de matrices (<i>Nota 5., página 706</i>)
	5	Asignación de memoria automática de variables simbólicas
	6	Declaración de matriz (<i>Nota 11, página 709</i>)
	7	Declaración de matriz retentiva (<i>Nota 11, página 709</i>)
	8	Declaración de variables estructuradas
18	1	Inicialización de variables representadas directamente (<i>Nota 11, página 709</i>)
	3	Ubicación y asignación de valores iniciales a variables simbólicas
	4	Asignación de ubicación de matrices e inicialización
	5	Inicialización de variables simbólicas
	6	Inicialización de matriz (<i>Nota 11, página 709</i>)
	7	Declaración e inicialización de matriz retentiva (<i>Nota 11, página 709</i>)
	8	Inicialización de variables estructuradas
	9	Inicialización de constantes
	10	Inicialización de instancias de bloques de función
19	1	Entrada negada
	2	Salida negada
19a	1	Llamada de bloque de función/función formal
	2	Llamada de bloque de función/función no formal
20	1	Uso de EN y ENO mostrado en LD
	2	Uso sin EN y ENO mostrado en FBD

N.º de tabla	N.º de función	Descripción de la función
20a	1	Declaración de variables de E/S (textual)
	2	Declaración de variables de E/S (gráfica)
	3	Conexión gráfica de variables de E/S para diferentes variables (gráficas)
21	1	Funciones sobrecargadas
	2	Funciones tipadas
22	1	*_TO_* (Nota 1., página 705)
	2	TRUNC (Nota 2, página 706)
	3	*_BCD_TO_* (Nota 3., página 706)
	4	**_TO_BCD_* (Nota 3., página 706)
23	1	Función ABS
	2	Función SQRT
	3	Función LN
	4	Función LOG
	5	Función EXP
	6	Función SIN
	7	Función COS
	8	Función TAN
	9	Función ASIN
	10	Función ACOS
	11	Función ATAN
24	12	Función ADD
	13	Función MUL
	14	Función SUB
	15	Función DIV
	16	Función MOD
	17	Función EXPT
	18	Función MOVE
25	1	Función SHL
	2	Función SHR
	3	Función ROR
	4	Función ROL

N.º de tabla	N.º de función	Descripción de la función
26	5	Función AND
	6	Función OR
	7	Función XOR
	8	Función NOT
27	1	Función SEL
	2a	Función MAX
	2b	Función MIN
	3	Función LIMIT
	4	Función MUX
28	5	Función GT
	6	Función GE
	7	Función EQ
	8	Función LE
	9	Función LT
	10	Función NE
29	1	Función LEN(<i>Nota 4., página 706</i>)
	2	Función LEFT(<i>Nota 4., página 706</i>)
	3	Función RIGHT(<i>Nota 4., página 706</i>)
	4	Función MID(<i>Nota 4., página 706</i>)
	6	Función INSERT(<i>Nota 4., página 706</i>)
	7	Función DELETE(<i>Nota 4., página 706</i>)
	8	Función REPLACE(<i>Nota 4., página 706</i>)
	9	Función FIND(<i>Nota 4., página 706</i>)

N.º de tabla	N.º de función	Descripción de la función
30	1a	Función ADD (Nota 6, página 708)
	1b	Función ADD_TIME
	2b	Función ADD_TOD_TIME
	3b	Función ADD_DT_TIME
	4a	Función SUB (Nota 6, página 708)
	4b	Función SUB_TIME
	5b	Función SUB_DATE_DATE
	6b	Función SUB_TOD_TIME
	7b	Función SUB_TOD_TOD
	8b	Función SUB_DT_TIME
	9b	Función SUB_DT_DT
	10a	Función MUL (Nota 6, página 708)
	10b	Función MULTIME
	11a	Función DIV (Nota 6, página 708)
	11b	Función DIVTIME
33	1a	Descriptor RETAIN para variables internas (Nota 11, página 709)
	2a	Descriptor RETAIN para variables de salida (Nota 11, página 709)
	2b	Descriptor RETAIN para variables de entrada (Nota 11, página 709)
	3a	Descriptor RETAIN para bloques de funciones internas (Nota 11, página 709)
	4a	Declaración de VAR_IN_OUT (textual)
	4b	Declaración y uso de VAR_IN_OUT (gráficos)
	4c	Declaración de VAR_IN_OUT con asignación de variables diferentes (gráficas)
34	1	Bloque de funciones biestables (definir dominante)
	2	Bloque de funciones biestables (restablecer dominante)
35	1	Detecto en flanco ascendente
	2	Detecto en flanco descendente

N.º de tabla	N.º de función	Descripción de la función
36	1a	Bloque de funciones CTU (contador progresivo)
	1b	Bloque de funciones CTU_DINT
	1d	Bloque de funciones CTU_UDINT
	2a	Bloque de funciones CTD (contador regresivo)
	2b	Bloque de funciones CTD_DINT
	2d	Bloque de funciones CTD_UDINT
	3a	Bloque de funciones CTUD (contador progresivo/regresivo)
	3b	Bloque de funciones CTUD_DINT
	3d	Bloque de funciones CTUD_UDINT
37	1	Bloque de funciones TP (pulso)
	2a	Bloque de funciones TON (retardo Act)
	3a	Bloque de funciones TOF (retardo Des)
39	19	Uso de variables representadas directamente
40	1	Paso y paso inicial: formato gráfico con enlaces directos
	3a	Flag de paso: formato general
	4	Tiempo de paso transcurrido: formato general
41	7	Uso de nombre de la transición
	7a	Condición de transición relacionada a través de un nombre de transición que utiliza lenguaje LD
	7b	Condición de transición relacionada a través de un nombre de transición que utiliza lenguaje FBD
	7c	Condición de transición relacionada a través de un nombre de transición que utiliza lenguaje IL
	7d	Condición de transición relacionada a través de un nombre de transición que utiliza lenguaje ST
42	1	Cualquier variable booleana declarada en un bloque de VAR o VAR_OUTPUT o sus equivalentes gráficos pueden considerarse una acción.
	2l	Declaración gráfica de acción en lenguaje LD
	2f	Declaración gráfica de acción en lenguaje FBD
	3s	Declaración textual de acción en lenguaje ST
	3i	Declaración textual de acción en lenguaje IL

N.º de tabla	N.º de función	Descripción de la función
43	1	Bloque de acciones adyacente física o lógicamente al paso (<i>Nota 7., página 708</i>)
	2	Bloque de acciones concatenadas adyacente física o lógicamente al paso (<i>Nota 8., página 708</i>)
44	1	Descriptor de acción en bloque de acciones compatible
	2	Nombre de acción en bloque de acciones compatible
45	1	Ninguno: sin descriptor
	2	Descriptor N
	3	Descriptor R
	4	Descriptor S
	5	Descriptor L
	6	Descriptor D
	7	Descriptor P
	9	Descriptor DS
	11	Descriptor P1
	12	Descriptor P0
45a	2	Control de acciones sin "ciclo final"
46	1	Secuencia única
	2a	Divergencia de selección de secuencia: prioridad de izquierda a derecha de las evaluaciones de transiciones
	3	Convergencia de selección de secuencia
	4	Secuencias simultáneas: divergencia y convergencia
	5a	Salto de secuencia: prioridad de izquierda a derecha de las evaluaciones de transiciones
	6a	Bucle de secuencia: prioridad de izquierda a derecha de las evaluaciones de transiciones
49	1	Construcción CONFIGURATION...END_CONFIGURATION (<i>Nota 12, página 709</i>)
	5a	Construcción TASK periódica
	5b	Construcción TASK no periódica
	6a	Construcción WITH para la asociación de PROGRAM a TASK (<i>Nota 9, página 708</i>)
	6c	Declaración PROGRAM sin asociación de TASK (<i>Nota 10, página 708</i>)

N.º de tabla	N.º de función	Descripción de la función
50	5a	Planificación sin orden de prioridad (<i>Nota 13, página 709</i>)
	5b	Planificación por orden de prioridad (<i>Nota 14, página 709</i>)

Nota 1.

Lista de funciones de conversiones de tipo:

- `BOOL_TO_BYTE`, `BOOL_TO_DINT`, `BOOL_TO_INT`, `BOOL_TO_REAL`, `BOOL_TO_TIME`, `BOOL_TO_UDINT`, `BOOL_TO_UINT`, `BOOL_TO_WORD`, `BOOL_TO_DWORD`
- `BYTE_TO_BOOL`, `BYTE_TO_DINT`, `BYTE_TO_INT`, `BYTE_TO_REAL`, `BYTE_TO_TIME`, `BYTE_TO_UDINT`, `BYTE_TO_UINT`, `BYTE_TO_WORD`, `BYTE_TO_DWORD`, `BYTE_TO_BIT`
- `DINT_TO_BOOL`, `DINT_TO_BYTE`, `DINT_TO_INT`, `DINT_TO_REAL`, `DINT_TO_TIME`, `DINT_TO_UDINT`, `DINT_TO_UINT`, `DINT_TO_WORD`, `DINT_TO_DWORD`, `DINT_TO_DBCD`, `DINT_TO_STRING`
- `INT_TO_BOOL`, `INT_TO_BYTE`, `INT_TO_DINT`, `INT_TO_REAL`, `INT_TO_TIME`, `INT_TO_UDINT`, `INT_TO_UINT`, `INT_TO_WORD`, `INT_TO_BCD`, `INT_TO_DBCD`, `INT_TO_DWORD`, `INT_TO_STRING`
- `REAL_TO_BOOL`, `REAL_TO_BYTE`, `REAL_TO_DINT`, `REAL_TO_INT`, `REAL_TO_TIME`, `REAL_TO_UDINT`, `REAL_TO_UINT`, `REAL_TO_WORD`, `REAL_TO_DWORD`, `REAL_TO_STRING`
- `TIME_TO_BOOL`, `TIME_TO_BYTE`, `TIME_TO_DINT`, `TIME_TO_INT`, `TIME_TO_REAL`, `TIME_TO_UDINT`, `TIME_TO_UINT`, `TIME_TO_WORD`, `TIME_TO_DWORD`, `TIME_TO_STRING`
- `UDINT_TO_BOOL`, `UDINT_TO_BYTE`, `UDINT_TO_DINT`, `UDINT_TO_INT`, **`UDINT_TO_REAL`**, **`UDINT_TO_TIME`**, **`UDINT_TO_UINT`**, `UDINT_TO_WORD`, `UDINT_TO_DWORD`
- `UINT_TO_BOOL`, `UINT_TO_BYTE`, `UINT_TO_DINT`, `UINT_TO_INT`, `UINT_TO_REAL`, `UINT_TO_TIME`, `UINT_TO_UDINT`, `UINT_TO_WORD`, `UINT_TO_DWORD`,
- `WORD_TO_BOOL`, `WORD_TO_BYTE`, `WORD_TO_DINT`, `WORD_TO_INT`, `WORD_TO_REAL`, `WORD_TO_TIME`, `WORD_TO_UDINT`, `WORD_TO_UINT`, `WORD_TO_BIT`, `WORD_TO_DWORD`
- `DWORD_TO_BOOL`, `DWORD_TO_BYTE`, `DWORD_TO_DINT`, `DWORD_TO_INT`, `DWORD_TO_REAL`, `DWORD_TO_TIME`, `DWORD_TO_UDINT`, `DWORD_TO_UINT`, `DWORD_TO_BIT`,

Los efectos de cada conversión se describen en el texto de ayuda suministrado con la librería base.

Nota 2

Lista de tipos para función truncada:

- REAL_TRUNC_DINT, REAL_TRUNC_INT, REAL_TRUNC_UDINT, REAL_TRUNC_UINT

Los efectos de cada conversión se describen en el texto de ayuda suministrado con la librería base.

Nota 3.

Lista de tipos para función de conversión BCD:

- BCD_TO_INT, DBCD_TO_INT, DBCD_TO_DINT

Lista de tipos para función de conversión BCD:

- INT_TO_BCD, INT_TO_DBCD, DINT_TO_DBCD

Los efectos de cada conversión se describen en el texto de ayuda suministrado con la librería base.

Nota 4.

Lista de tipos para funciones de cadena:

- LEN_INT, LEFT_INT, RIGHT_INT, MID_INT, INSERT_INT, DELETE_INT, REPLACE_INT, FIND_INT

Nota 5.

Se puede asignar una variable a otra representada directamente si éstas son del mismo tipo.

Es decir, una variable de tipo INT sólo se puede asignarse a una variable representada directamente de tipo INT.

Sin embargo, existe una excepción a esta regla: para las variables de palabra interna (%MW<i>), plana (%IW<i>) y de memoria (%KW<i>) se admite cualquier tipo de variable declarada.

Asignaciones permitidas:

	Sintaxis	Tipo de datos	Tipos de variables admitidas
Bit interno	%M<i> o %MX<i>	EBOOL	EBOOL ARRAY [...] OF EBOOL
Palabra interna	%MW<i>	INIC	Se admiten todos los tipos salvo: ● EBOOL ● ARRAY [...] OF EBOOL

	Sintaxis	Tipo de datos	Tipos de variables admitidas
Palabra doble interna	%MD<i>	DINT	No hay asignación debido al solapamiento entre %MW<i> y %MD<i> y %MF<i>.
Real interno	%MF<i>	REAL	No hay asignación debido al solapamiento entre %MW<i> y %MD<i> y %MF<i>.
Palabra constante	%KW<i>	INIC	Se admiten todos los tipos salvo: ● EBOOL ● ARRAY [...] OF EBOOL
Palabra doble constante	%KD<i>	DINT	No hay asignación debido al solapamiento entre %KW<i> y %KD<i> y %KF<i>. Este tipo de variables sólo existe en los PLC Premium.
Real constante	%KF<i>	REAL	No hay asignación debido al solapamiento entre %KW<i> y %KD<i> y %KF<i>. Este tipo de variables sólo existe en los PLC Premium.
Bit de sistema	%S<i> o %SX<i>	EBOOL	EBOOL
Palabra de sistema	%SW<i>	INIC	INIC
Palabra doble de sistema	%SD<i>	DINT	DINT
Bit plano	%I<i>	EBOOL	EBOOL ARRAY [...] OF EBOOL Este tipo de variables sólo existe en los PLC Quantum

	Sintaxis	Tipo de datos	Tipos de variables admitidas
Palabra plana	%IW<i>	INIC	Se admiten todos los tipos salvo: ● EBOOL ● ARRAY [...] OF EBOOL Este tipo de variables sólo existe en los PLC Quantum
Palabra común	%NWi.j.k	INIC	INIC
Variables topológicas	%I..., %Q..., ...	...	Mismo tipo (En algunos módulos de E/S digitales se admite la asignación de matrices de EBOOL en objetos %IX<topo> y %QX<topo>.)
Bits de extracción	%MWi.j, ...	BOOL	BOOL

Nota 6

Operador único "+" (para ADD), "-" (para SUB), "*" (para MUL) o "/" (para DIV) en lenguaje ST.

Nota 7.

Esta función sólo se presenta en la "vista ampliada" del gráfico.

Nota 8.

Esta función se incluye en la "vista ampliada" del gráfico; sin embargo, no se presenta como bloques concatenados, sino como un lista que se puede desplazar de los nombres de acciones con descriptores asociados dentro de un símbolo de bloque único.

Nota 9

Sólo hay una asignación de uno a uno (de la instancia de programa a la tarea). El formato textual se sustituye por un cuadro de diálogo de propiedades.

Nota 10

El formato textual se sustituye por un cuadro de diálogo de propiedades.

Nota 11

Todas las variables son descriptores retentivos (el descriptor `RETAIN` se asume implícitamente en las declaraciones de variable).

Nota 12

El formato textual se sustituye por la representación del explorador de proyectos.

Nota 13

Al utilizar una instrucción `Mask-IT`, el usuario puede adoptar un comportamiento no prioritario. Encontrará `MASKEVT` (enmascaramiento de EVT global) y `UNMASKEVT` (desenmascaramiento de EVT global) en las funciones de sistema del conjunto de librerías.

Nota 14

De manera predeterminada, el sistema multitarea es prioritario.

Elementos del lenguaje IL

Elementos del lenguaje IL

Tabla de conformidad con la norma IEC para elementos de lenguaje IL:

Nº de tabla	Nº de función	Descripción de la función
51b	1	Expresión entre paréntesis que comienza por un operador explícito
51b	2	Expresión entre paréntesis (forma corta)
52	1	Operador LD (con modificador "N")
	2	Operador ST (con modificador "N")
	3	S, operador R
	4	Operador AND (con modificadores "(", "N")
	6	Operador OR (con modificadores "(", "N")
	7	Operador XOR (con modificadores "(", "N")
	7a	Operador NOT
	8	Operador ADD (con modificador "(")
	9	Operador SUB (con modificador "(")
	10	Operador MUL (con modificador "(")
	11	Operador DIV (con modificador "(")
	11a	Operador MOD (con modificador "(")
	12	Operador GT (con modificador "(")
	13	Operador GE (con modificador "(")
	14	Operador EQ (con modificador "(")
	15	Operador NE (con modificador "(")
	16	Operador LE (con modificador "(")
	17	Operador LT (con modificador "(")
	18	Operador JMP (con modificadores "C", "N")
	19	Operador CAL (con modificadores "C", "N")
	20	Operador RET (con modificadores "C", "N") (Nota, página 711)
	21	) (Evaluación de la operación retardada)

Nº de tabla	Nº de función	Descripción de la función
53	1a	CAL de bloque de función con lista de argumentos no formal
	1b	CAL de bloque de función con lista de argumentos formal
	2	CAL de bloque de función con carga/almacenamiento de argumentos
	4	Ejecución de función con lista de argumentos formal
	5	Ejecución de función con lista de argumentos no formal

Nota

Sólo en DFB

Elementos de lenguaje ST

Elementos de lenguaje ST

Tabla de conformidad con la norma IEC para elementos de lenguaje ST:

Nº de tabla	Nº de función	Descripción de la función
55	1	Inserción entre paréntesis (expresión)
	2	Evaluación de las funciones: functionName(listOfArguments)
	3	Potenciación: **
	4	Negación: -
	5	Complemento: NOT
	6	Multiplicación: *
	7	División: /
	8	Módulo: MOD
	9	Suma: +
	10	Resta: -
	11	Comparación: <, >, <=, >=
	12	Igualdad: =
	13	Desigualdad: <>
	14	AND booleano: &
	15	AND booleano: AND
	16	OR exclusivo booleano: XOR
	17	OR booleano: OR
56	1	Asignación
	2	Ejecución del bloque de función y utilización de salida del bloque de función
	3	Instrucción RETURN (Nota, página 713)
	4	Instrucción IF
	5	Instrucción CASE
	6	Instrucción FOR
	7	Instrucción WHILE
	8	Instrucción REPEAT
	9	Instrucción EXIT
	10	Instrucción vacía

Nota

Sólo en DFB.

Elementos gráficos comunes

Elementos gráficos comunes

Tabla de conformidad con la norma IEC para elementos gráficos comunes:

N° de tabla	N° de función	Descripción de la función
57	2	Líneas horizontales: gráficas o semigráficas
	4	Líneas verticales: gráficas o semigráficas
	6	Conexión vertical/horizontal: gráfica o semigráfica
	8	Cruces de líneas sin conexión: gráficos o semigráficos
	10	Esquinas conectadas y no conectadas: gráficas o semigráficas
	12	Bloques con líneas conectadas: gráficos o semigráficos
58	1	Salto incondicional: lenguaje FBD:
	2	Salto incondicional: lenguaje LD:
	3	Salto condicional: lenguaje FBD:
	4	Salto condicional: lenguaje LD:
	5	Retorno condicional: lenguaje LD (<i>Nota, página 714</i>)
	6	Retorno condicional: lenguaje FBD (<i>Nota, página 714</i>)
	7	Retorno incondicional desde la función o bloque de funciones (<i>Nota, página 714</i>)
	8	Retorno incondicional: lenguaje LD (<i>Nota, página 714</i>)

Nota

Sólo en DFB

Elementos del lenguaje LD

Elementos del lenguaje LD

Tabla de conformidad con la norma IEC para elementos de lenguaje LD:

N° de tabla	N° de función	Descripción de la función
59	1	Rail de alimentación izquierdo
	2	Rail de alimentación derecho
60	1	Conexión horizontal
	2	Conexión vertical
61	1	Contacto normal abierto (barra vertical) <i>(Nota, página 715)</i>
	3	Contacto normal cerrado (barra vertical) <i>(Nota, página 715)</i>
	5	Transición positiva-contacto de detección (barra vertical) <i>(Nota, página 715)</i>
	7	Contacto de detección de transición negativa (barra vertical) <i>(Nota, página 715)</i>
62	1	Bobina
	2	Bobina negada
	3	Bobina SET (retención)
	4	Bobina RESET (sin retención)
	8	Bobina de detección de transición positiva
	9	Bobina de detección de transición negativa

Nota

Únicamente representación gráfica

Parámetros dependientes del tipo de implementación

Parámetros dependientes del tipo de implementación

Tabla de conformidad con la norma IEC para parámetros dependientes del tipo de implementación:

Parámetros	Limitaciones/comportamiento
Longitud máxima de los identificadores	32 caracteres
Longitud máxima de los comentarios	En Unity Pro: un máximo de 1.024 caracteres para cada objeto editor Importación: limitada por restricciones XML o por la utilización de cadenas UDB en la capa persistente.
Sintaxis y semántica pragmáticas	Unity V1.0 sólo implementa 1 construcción pragmática, que se usa para el convertidor de herencia: <pre>{ ConvError (' error text'); }</pre> se hace caso omiso de cualquier otra construcción pragmática (se muestra un mensaje de advertencia)
Sintaxis y semántica para el uso del carácter de comillas dobles cuando una implementación concreta admite la función n.º 4 pero no la n.º 2 de la tabla 5.	(Se admite la función n.º 2 de la tabla 5.)
Rango de valores y precisión de representación de variables de tipo TIME, DATE, TIME_OF_DAY y DATE_AND_TIME	Para TIME: t#0ms - t#4294967295ms (=t#49D_17H_2M_47S_295MS) Para DATE: D#1990-01-01 - D#2099-12-31 Para TOD: TOD#00:00:00 - TOD#23:59:59
Precisión de representación de segundos en variables del tipo TIME, TIME_OF_DAY y DATE_AND_TIME	TIME: 1 ms de precisión TIME_OF_DAY: 1 s de precisión
Cantidad máxima de valores enumerados:	No es aplicable
Cantidad máxima de índices de matriz	6
Tamaño máximo de matriz	64 Kbytes
Cantidad máxima de elementos de estructura	Sin límite
Tamaño máximo de estructura	64 Kbytes
Rango máximo de valores de índice	Rango DINT

Parámetros	Limitaciones/comportamiento
Cantidad máxima de niveles de estructuras intercaladas	10
Longitud máxima predeterminada de las variables <code>STRING</code> y <code>WSTRING</code>	16 caracteres
Longitud máxima permitida de las variables <code>STRING</code> y <code>WSTRING</code>	64 Kbytes
Cantidad máxima de niveles jerárquicos Asignación lógica o física	Premium: asignación física (5 niveles) Quantum: asignación lógica (1 nivel)
Número máximo de entradas de funciones ampliables	El número de todos los parámetros de entrada (incluidos los parámetros E/S) está limitado a 32. De este modo, el límite para parámetros de entrada ampliable es (32 - número de parámetros de entrada - número de parámetros de E/S) El límite para parámetros de salida ampliable es (32 - número de parámetros de salida - número de parámetros de E/S)
Efectos de conversiones de tipo con precisión	Consulte la ayuda en línea.
Condiciones de error durante las conversiones de tipo	Las condiciones de error se describen en la ayuda en línea. De manera global, <code>%S18</code> está definido para errores de desbordamiento. También se define <code>ENO</code> . El resultado depende de la función específica.
Precisión de funciones numéricas	Procesamiento o emulación de coma flotante INTEL.
Efectos de las conversiones de tipo entre los tipos de datos de tiempo y otros no definidos en la tabla 30.	Consulte la ayuda en línea.
Cantidad máxima de especificaciones e instanciaciones de los bloques de funciones.	Únicamente limitado por el tamaño máximo de una sección.
Asignación de variable de entrada del bloque de funciones cuando <code>EN</code> es <code>FALSE</code>	Sin asignación

Parámetros	Limitaciones/comportamiento
Pvmin, Pymax de contadores	Contadores de base INT: ● Pvmin=-32.768 (0x8.000) ● Pymax=32.767 (0x7FFF) Contadores de base UINT: ● Pvmin=0 (0x0) ● Pymax=65.535 (0xFFFF) Contadores de base DINT: ● Pvmin=-2.147.483.648 (0x80.000.000) ● Pymax=2.147.483.647 (0x7FFFFFFF) Contadores de base UDINT: ● Pvmin=0 (0x0) ● Pymax=4.294.967.295 (0xFFFFFFFF)
Efectos de un cambio en el valor de una entrada PT durante una operación de temporización.	Los nuevos valores PT se tienen inmediatamente en cuenta a la vez. Incluso durante una ejecución, la operación de temporización funciona con los nuevos valores.
Limitaciones del tamaño del programa	Depende del tipo de controlador y memoria.
Precisión de tiempo de paso transcurrido	10 ms
Cantidad máxima de pasos por SFC	1.024 pasos por sección SFC
Cantidad máxima de transiciones por SFC y por paso	Limitado por el área disponible para introducir pasos/transiciones y por la cantidad máxima de pasos por sección SFC (1.024 pasos). 32 transiciones por paso Limitado por el área disponible para introducir bifurcaciones alternativas o simultáneas. El máximo son 32 filas.
Cantidad máxima de bloques de acción por paso	20
Acceso al equivalente funcional de las salidas Q o A.	No es aplicable
Tiempo de borrado de transición	Dependiente de destino; siempre < 100 microsegundos
Ancho máximo de construcciones de divergencia/convergencia	32
Contenido de las librerías RESOURCE	No es aplicable
Efecto de utilizar el acceso READ_WRITE para las salidas de bloques de funciones.	No es aplicable

Parámetros	Limitaciones/comportamiento
Cantidad máxima de tareas:	Depende del tipo de controlador. Cantidad máxima en el controlador más potente: 9 tareas
Resolución en el intervalo de tareas	10 ms
Longitud máxima de las expresiones	Prácticamente no tiene límite
Longitud máxima de las instrucciones	Prácticamente no tiene límite
Cantidad máxima de selecciones CASE	Prácticamente no tiene límite
Valor de variable de control a la terminación del bucle FOR	No definido
Restricciones en la topología de red	Sin restricciones
Orden de evaluación de los bucles de realimentación	Se ejecuta primero el bloque conectado a la variable de realimentación.

Condiciones de error

Condiciones de error

Tablas de las normas IEC para condiciones de error:

Condiciones de error	Tratamiento (véase <i>Nota, página 721</i>)
Comentarios intercalados	2) se informa del error durante la programación
El valor de una variable es superior al del rango especificado	4) se informa del error durante la ejecución
Falta la configuración de una especificación de dirección incompleta (notación "**")	No es aplicable
Intento de una unidad de organización de programa para modificar una variable que se ha declarado <code>CONSTANT</code>	2) se informa del error durante la programación
Uso inadecuado de variables representadas directamente o externas en funciones.	No es aplicable
Una variable <code>VAR_IN_OUT</code> no está "asignada correctamente".	2) se informa del error durante la programación
Error de conversión de tipo	4) se informa del error durante la ejecución
Los resultados numéricos superan el rango para tipo de datos.	4) se informa del error durante la ejecución
División entre cero	4) se informa del error durante la ejecución
Tipos de datos de entrada combinados para una función de selección	2) se informa del error durante la programación
El resultado supera el rango para tipo de datos.	4) se informa del error durante la ejecución
No se especifica ningún valor para una variable de E/S	2) se informa del error durante la programación
Cero o más de un paso inicial en la red SFC	3) se informa del error durante el proceso de análisis/carga/enlace
El programa de usuario intenta modificar un estado u hora de paso	2) se informa del error durante la programación
Efectos secundarios en la evaluación de la condición de transición	3) se informa del error durante el proceso de análisis/carga/enlace
Error de contención de control de acción	3) se informa del error durante el proceso de análisis/carga/enlace
Transiciones verdaderas y no prioritarias a la vez en una divergencia de selección	No es aplicable

Condiciones de error	Tratamiento (véase <i>Nota, página 721</i>)
SFC no seguro o no alcanzable	3) se informa del error durante el proceso de análisis/carga/enlace
Conflicto del tipo de datos en VAR_ACCESS	No es aplicable
Una tarea no consigue planificarse o cumplir el plazo límite de ejecución	4) se informa del error durante la ejecución
Los resultados numéricos superan el rango para tipo de datos	4) se informa del error durante la ejecución
Los resultados actuales y los operandos no son del mismo tipo de datos.	2) se informa del error durante la programación
División entre cero	4) se informa del error durante la ejecución
Los resultados numéricos superan el rango para tipo de datos	4) se informa del error durante la ejecución
Tipo de datos no válidos para la operación	4) se informa del error durante la ejecución
Retorno desde la función sin valor asignado	No es aplicable
La repetición no consigue terminar	4) se informa del error durante la ejecución
Se ha utilizado el mismo identificador como etiqueta del conector y nombre de elemento	No es aplicable
Variable de realimentación no inicializado	1) no se informa del error

Nota

Identificaciones para el tratamiento de condiciones de error de acuerdo con la norma IEC 61131-3, subcláusula 1.5.1, d):

- 1) no se informa del error
- 2) se informa del error durante la programación
- 3) se informa del error durante el proceso de análisis/carga/enlace
- 4) se informa del error durante la ejecución

B.3 Extensiones de la norma IEC 61131-3

Extensiones de la norma IEC 61131-3, 2ª edición

Presentación

Además de las funciones IEC estándar enumeradas en las (véase página 696), el entorno de programación de Unity Pro heredó un número de funciones de PL7. Estas extensiones se proporcionan de forma opcional; pueden ser verificadas o no en el correspondiente cuadro de diálogo de opciones. El cuadro de diálogo y las funciones se describen con detalle en un capítulo de la ayuda en línea titulado *Datos y lenguajes* (véase *Unity Pro, Modalidades de funcionamiento*,).

En el cuadro de diálogo de opciones no está incluida otra extensión, que se heredó de los entornos de programación de PL7 y Concept: Unity Pro proporciona las construcciones de la llamada Sección en todas las lenguajes de programación, lo que permite la subdivisión de unidad de organización de programa (POU). Esta construcción presenta la posibilidad de combinar varios lenguajes (por ejemplo, secciones FBD, SFC) en un cuerpo POU, función que, en caso de utilizarse con este fin, constituye una extensión de la sintaxis de IEC. Un cuerpo POU compatible debería contener una única sección. Las secciones no crean un campo de aplicación de nombre distinto. El campo de aplicación de nombre para todos los elementos de lenguaje es el POU.

Propósito de las secciones

Las secciones tienen diferentes fines:

- Las secciones permiten subdividir cuerpos POU grandes de acuerdo con aspectos funcionales: el usuario tiene la posibilidad de subdividir el cuerpo POU en partes con funcionalidad significativa. La lista de secciones representa un tipo de tabla de contenidos funcional de un cuerpo POU amplio, que, de otro modo, estaría desestructurado.
- Las secciones permiten subdividir cuerpos POU grandes de acuerdo con aspectos gráficos: el usuario tiene la posibilidad de diseñar estructuras de cuerpos POU grandes de acuerdo con una presentación gráfica. Puede crear secciones gráficas, grandes o pequeñas, según prefiera.
- La subdivisión de cuerpos POU grandes permite cambios en línea más rápidos: en Unity Pro, la sección se utiliza como unidad para cambio en línea. Si un cuerpo POU se modifica durante la ejecución en ubicaciones diferentes, todas las secciones afectadas por los cambios se descargan automáticamente si se solicita explícitamente.

- Las secciones permiten volver a organizar el orden de ejecución de partes específicas y etiquetadas de un cuerpo POU: el nombre de la sección sirve como etiqueta de aquella parte del cuerpo que está contenida en la sección, y al ordenar estas etiquetas, la ejecución de esas partes es posible.
- Las secciones permiten utilizar distintos lenguajes de forma paralela en el mismo POU: esta función es una ampliación importante de la sintaxis de la norma IEC, que sólo permite la utilización de un único lenguaje IEC para un cuerpo POU. En un cuerpo compatible, SFC debe usarse para gestionar diferentes lenguajes dentro de un cuerpo (cada transición y acción deben formularse en su propio lenguaje).

B.4 Sintaxis de lenguajes textuales

Sintaxis de lenguaje textual

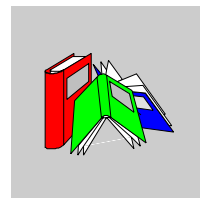
Descripción

El entorno de programación Unity Pro V1.0 no proporciona apoyo para una importación o exportación de archivos de texto compatibles con la sintaxis de lenguajes textuales completos, tal y como se especifica en el Anexo B de la norma IEC 61131-3, 2ª edición.

Sin embargo, en secciones de lenguaje textual se admite la sintaxis textual de IL y ST, como se precisa en los anexos B.2 Y B.3 de IEC 61131-3, 2ª edición, incluidas todas las producciones referenciadas de forma directa o indirecta fuera del anexo B.1.

No se implementan aquellas producciones de sintaxis en el Anexo B de IEC 61131-3, 2ª edición que pertenecen a aquellas funciones que no están admitidas por Unity Pro, de acuerdo con las tablas de conformidad (*véase página 696*).

Glosario



0-9

%I

Según la normativa IEC, %I indica un objeto de lenguaje de tipo de entrada binaria.

%ID

Según la normativa IEC, %MW indica un objeto de lenguaje de tipo de palabra doble de entrada.

Sólo los objetos de E/S posibilitan la localización de instancias de tipo (%MD<i>, %KD<i>, %QD, %ID, %MF<i>, %KF<i>, %QF, %IF) mediante su dirección topológica (por ejemplo, %MD0.6.0.11, %MF0.6.0.31).

%IF

Según la normativa IEC, %MW indica un objeto de lenguaje de tipo real de entrada.

Sólo los objetos de E/S posibilitan la localización de instancias de tipo (%MD<i>, %KD<i>, %QD, %ID, %MF<i>, %KF<i>, %QF, %IF) mediante su dirección topológica (por ejemplo, %MD0.6.0.11, %MF0.6.0.31).

%IW

Según la normativa IEC, %IW indica un objeto de lenguaje de tipo de entrada analógica.

%KD

Según la normativa IEC, %MW indica un objeto de lenguaje de tipo de palabra doble constante.

Para los PLC Premium/Atrium, las instancias de tipo doble de los datos ubicados (%MD<i>, %KD<i>) o flotantes (%MF<i>, %KF<i>) deben localizarse mediante un tipo de entero (%MW<i>, %KW<i>). Sólo los objetos de E/S posibilitan la localización de instancias de tipo (%MD<i>, %KD<i>, %QD, %ID, %MF<i>, %KF<i>, %QF, %IF) mediante su dirección topológica (por ejemplo, %MD0.6.0.11, %MF0.6.0.31).

Para PLC Modicon M340, las instancias de tipo doble de datos ubicados (%MD<i>, %KD<i>) o flotantes (%MF<i>, %KF<i>) no están disponibles.

%KF

Según la normativa IEC, %MW indica un objeto de lenguaje de tipo real constante.

Para los PLC Premium/Atrium, las instancias de tipo doble de los datos ubicados (%MD<i>, %KD<i>) o flotantes (%MF<i>, %KF<i>) deben localizarse mediante un tipo de entero (%MW<i>, %KW<i>). Sólo los objetos de E/S posibilitan la localización de instancias de tipo (%MD<i>, %KD<i>, %QD, %ID, %MF<i>, %KF<i>, %QF, %IF) mediante su dirección topológica (por ejemplo, %MD0.6.0.11, %MF0.6.0.31).

Para PLC Modicon M340, las instancias de tipo doble de datos ubicados (%MD<i>, %KD<i>) o flotantes (%MF<i>, %KF<i>) no están disponibles.

%KW

Según la normativa IEC, %KW indica un objeto de lenguaje de tipo de palabra constante.

Para los PLC Premium/Atrium, las instancias de tipo doble de los datos ubicados (%MD<i>, %KD<i>) o flotantes (%MF<i>, %KF<i>) deben localizarse mediante un tipo de entero (%MW<i>, %KW<i>). Sólo los objetos de E/S posibilitan la localización de instancias de tipo (%MD<i>, %KD<i>, %QD, %ID, %MF<i>, %KF<i>, %QF, %IF) mediante su dirección topológica (por ejemplo, %MD0.6.0.11, %MF0.6.0.31).

Para PLC Modicon M340, las instancias de tipo doble de datos ubicados (%MD<i>, %KD<i>) o flotantes (%MF<i>, %KF<i>) no están disponibles.

%M

Según la normativa IEC, %M indica un objeto de lenguaje de tipo de bit de memoria.

%MD

Según la normativa IEC, %MW indica un objeto de lenguaje de tipo de palabra doble de memoria.

Para los PLC Premium/Atrium, las instancias de tipo doble de los datos ubicados (%MD<i>, %KD<i>) o flotantes (%MF<i>, %KF<i>) deben localizarse mediante un tipo de entero (%MW<i>, %KW<i>). Sólo los objetos de E/S posibilitan la localización de instancias de tipo (%MD<i>, %KD<i>, %QD, %ID, %MF<i>, %KF<i>, %QF, %IF) mediante su dirección topológica (por ejemplo, %MD0.6.0.11, %MF0.6.0.31).

Para PLC Modicon M340, las instancias de tipo doble de datos ubicados (%MD<i>, %KD<i>) o flotantes (%MF<i>, %KF<i>) no están disponibles.

%MF

Según la normativa IEC, %MW indica un objeto de lenguaje de tipo real de memoria.

Para los PLC Premium/Atrium, las instancias de tipo doble de los datos ubicados (%MD<i>, %KD<i>) o flotantes (%MF<i>, %KF<i>) deben localizarse mediante un tipo de entero (%MW<i>, %KW<i>). Sólo los objetos de E/S posibilitan la localización de instancias de tipo (%MD<i>, %KD<i>, %QD, %ID, %MF<i>, %KF<i>, %QF, %IF) mediante su dirección topológica (por ejemplo, %MD0.6.0.11, %MF0.6.0.31).

Para PLC Modicon M340, las instancias de tipo doble de datos ubicados (%MD<i>, %KD<i>) o flotantes (%MF<i>, %KF<i>) no están disponibles.

%MW

Según la normativa IEC, %MW indica un objeto de lenguaje de tipo de palabra de memoria.

Para los PLC Premium/Atrium, las instancias de tipo doble de los datos ubicados (%MD<i>, %KD<i>) o flotantes (%MF<i>, %KF<i>) deben localizarse mediante un tipo de entero (%MW<i>, %KW<i>). Sólo los objetos de E/S posibilitan la localización de instancias de tipo (%MD<i>, %KD<i>, %QD, %ID, %MF<i>, %KF<i>, %QF, %IF) mediante su dirección topológica (por ejemplo, %MD0.6.0.11, %MF0.6.0.31).

Para PLC Modicon M340, las instancias de tipo doble de datos ubicados (%MD<i>, %KD<i>) o flotantes (%MF<i>, %KF<i>) no están disponibles.

%Q

Según la normativa IEC, %Q indica un objeto de lenguaje de tipo de salida binaria.

%QD

Según la normativa IEC, %MW indica un objeto de lenguaje de tipo de palabra doble de salida.

Sólo los objetos de E/S posibilitan la localización de instancias de tipo (%MD<i>, %KD<i>, %QD, %ID, %MF<i>, %KF<i>, %QF, %IF) mediante su dirección topológica (por ejemplo, %MD0.6.0.11, %MF0.6.0.31).

%QF

Según la normativa IEC, %MW indica un objeto de lenguaje de tipo real de salida.

Sólo los objetos de E/S posibilitan la localización de instancias de tipo (%MD<i>, %KD<i>, %QD, %ID, %MF<i>, %KF<i>, %QF, %IF) mediante su dirección topológica (por ejemplo, %MD0.6.0.11, %MF0.6.0.31).

%QW

Según la normativa IEC, %QW indica un objeto de lenguaje de tipo de salida analógica.

A

Animación de conexiones

También se denomina **flujo de potencia**, y hace referencia a un tipo de animación que se utiliza con el lenguaje Ladder y los bloques de función. Las conexiones aparecen en rojo, verde o negro, en función de las variables conectadas.

ANY

Existe una jerarquía entre los distintos tipos de datos. En el DFB, a veces es posible declarar las variables que pueden contener distintos tipos de valores. En este caso, utilizaremos tipos ANY_XXX.

El siguiente diagrama muestra una estructura jerárquica.

```

ANY
  ANY_ELEMENTARY
    ANY_MAGNITUDE_OR_BIT
      ANY_MAGNITUDE
        ANY_NUM
        ANY_REAL
        REAL
        ANY_INT
        DINT, INT, UDINT, UINT
        TIME
      ANY_BIT
        DWORD, WORD, BYTE, BOOL
    ANY_STRING
      STRING
    ANY_DATE
      DATE_AND_TIME, DATE, TIME_OF_DAY
      EBOOL
  ANY_DERIVED
    ANY_ARRAY
      ANY_ARRAY_ANY_EDT
      ANY_ARRAY_ANY_MAGNITUDE
        ANY_ARRAY_ANY_NUM
        ANY_ARRAY_ANY_REAL
        ANY_ARRAY_REAL
        ANY_ARRAY_ANY_INT
        ANY_ARRAY_DINT
        ANY_ARRAY_INT
        ANY_ARRAY_UDINT
        ANY_ARRAY_UINT
      ANY_ARRAY_TIME
      ANY_ARRAY_ANY_BIT
      ANY_ARRAY_DWORD
      ANY_ARRAY_WORD
      ANY_ARRAY_BYTE
      ANY_ARRAY_BOOL
      ANY_ARRAY_ANY_STRING
      ANY_ARRAY_STRING
      ANY_ARRAY_ANY_DATE
      ANY_ARRAY_DATE_AND_TIME
      ANY_ARRAY_DATE
      ANY_ARRAY_TIME_OF_DAY
      ANY_ARRAY_EBOOL
    ANY_ARRAY_ANY_DDT
  ANY_STRUCTURE
    ANY_DDT
    ANY_IODDT
    ANY_FFB
    ANY_EFB
    ANY_DFB

```

ASCII

ASCII son las siglas de American Standard Code for Information Interchange (código normalizado americano para el intercambio de información).

Se trata de un código americano (pero que se ha convertido en una norma internacional) que utiliza 7 bits para definir cada carácter alfanumérico que se utiliza en inglés, los símbolos de puntuación, algunos caracteres gráficos y otros comandos de distinta naturaleza.

B

BCD

El formato decimal codificado en binario (BCD) se utiliza para representar números decimales comprendidos entre 0 y 9 mediante un grupo de cuatro bits (medio byte).

En este formato, los cuatro bits utilizados para codificar los números decimales tienen un rango de combinaciones que no se utilizan.

Ejemplo de codificación BCD:

- el número 2.450
- se codifica: 0010 0100 0101 0000

BIT

Se trata de una unidad binaria para una cantidad de información que pueden representar dos valores (o estados) diferentes: 0 ó 1.

Bloque de función

Consulte EFB.

BOOL

BOOL es la abreviatura del tipo «booleano». Éste es el elemento de datos básico en computación. Una variable de tipo **BOOL** tiene un valor: 0 (FALSO) o 1 (VERDADERO).

Un bit de extracción de palabras de tipo **BOOL**, por ejemplo: %MW10.4.

BYTE

Un **BYTE** es un conjunto de ocho bits. Un **BYTE** se puede introducir como número binario o en base 8.

El tipo **BYTE** se codifica en formato de 8 bits, que, en formato hexadecimal, ocupa el rango de 16#00 a 16#FF.

C

Constantes

Una variable de tipo `INT`, `DINT` o `REAL` situada en el campo de constantes (`%K`), o variables utilizadas en el direccionamiento directo (`%KW`, `%KD` o `%KF`). El programa no puede modificar el contenido de estas variables durante la ejecución.

Controlador

Programa que indica al sistema operativo del ordenador la presencia y las características de un dispositivo periférico. También se utiliza el término «controlador de dispositivos periféricos». Los controladores más populares son los de impresión. Para que un PLC se comuniquen con un PC, es necesario disponer de controladores de comunicación instalados (Uni-Telway, XIP, Fipway, etc.).

Convención de nombres (identificador)

Un identificador es una secuencia de letras, números y guiones bajos que comienza con una letra o un guión bajo (por ejemplo, el nombre de un tipo de bloque de función, una instancia, una variable o una sección). Las letras de conjuntos de caracteres nacionales (por ejemplo: ö, ü, é, ò) no podrán utilizarse en los nombres de proyectos y de DFB. Los guiones bajos son significativos como identificadores; por ejemplo, `A_BCD` y `AB_CD` se interpretan como dos identificadores distintos. No se permiten varios guiones bajos iniciales ni seguidos.

Los identificadores no pueden contener espacios en blanco. No realiza distinción entre mayúsculas y minúsculas; por ejemplo, `ABCD` y `abcd` se interpretan como el mismo identificador.

Según la normativa IEC 61131-3, no pueden utilizarse cifras antepuestas para los identificadores. No obstante, puede utilizarlos si, en el cuadro de diálogo

Herramientas → **Ajustes del proyecto**, en la ficha **Extensiones de lenguaje**, selecciona la casilla de verificación **Permitir cifras antepuestas**.

Los identificadores no pueden ser palabras clave.

CPU

Son las siglas de unidad central de proceso (Control Processing Unit).

Se trata del microprocesador. Está formada por la unidad de control combinada con la unidad aritmética. El objetivo de la unidad de control es extraer la instrucción para ejecutarla y los datos necesarios para ejecutar dicha instrucción de la memoria central, establecer conexiones eléctricas en la unidad aritmética y lógica y ejecutar el proceso de estos datos en esta unidad. En ocasiones, podemos encontrar memorias ROM o RAM en un mismo chip o incluso interfaces de E/S o búferes.

D

DATE

El tipo `DATE` se codifica en BCD con formato de 32 bits y contiene la información siguiente:

- el año, codificado en un campo de 16 bits;
- el mes, codificado en un campo de ocho bits;
- el día, codificado en un campo de ocho bits.

El tipo `DATE` se introduce de la forma siguiente: `D# <Año> - <Mes> - <Día>`

En la tabla siguiente se muestran los límites superior e inferior de cada campo:

Campo	Límites	Comentario
Año	[1990,2099]	Año
Mes	[01,12]	El 0 a la izquierda siempre aparece en pantalla, pero puede omitirse en el momento de la introducción.
Día	[01,31]	Para los meses 01\03\05\07\08\10\12
	[01,30]	Para los meses 04\06\09\11
	[01,29]	Para el mes 02 (años bisiestos).
	[01,28]	Para el mes 02 (años no bisiestos).

DATE_AND_TIME

Consulte `DT`

DBCD

Representación de un entero doble con formato BCD doble.

El formato de decimal codificado en binario (BCD) se utiliza para representar números decimales comprendidos entre 0 y 9 mediante un grupo de cuatro bits.

En este formato, los cuatro bits utilizados para codificar los números decimales tienen un rango de combinaciones que no se utilizan.

Ejemplo de codificación DBCD:

- el número 78.993.016
- se codifica: 0111 1000 1001 1001 0011 0000 0001 0110

DDT

DDT son las siglas de «tipo de datos derivados» (del inglés Derived Data Type).

Un tipo de datos derivados es un conjunto de elementos del mismo tipo (`MATRIZ`) o de distintos tipos (estructura).

DFB

DFB son las siglas de «bloque de función derivado» (del inglés Derived Function Block).

Los tipos DFB son bloques de función que el usuario puede programar en ST, IL, LD o FBD.

Si se utilizan tipos DFB en una aplicación, es posible:

- Simplificar el diseño y la introducción de datos del programa.
- Aumentar la legibilidad del programa.
- Facilitar la depuración del programa.
- Reducir el volumen del código generado.

DINT

DINT son las siglas del «entero doble» (del inglés Double Integer), que se codifica en 32 bits.

Los límites superior e inferior son: de $-(2 \text{ elevado a } 31)$ a $(2 \text{ elevado a } 31) - 1$.

Ejemplo:

`-2.147.483.648, 2.147.483.647, 16#FFFFFFFF.`

Documentación

Contiene toda la información del proyecto. La documentación se imprime una vez compilada y se utiliza para fines de mantenimiento.

La información de la documentación incluye:

- La configuración de hardware y de software
- El programa
- Los tipos de DFB
- Las variables y las tablas de animación
- Las referencias cruzadas
- ...

Cuando se crea un archivo de documentación, es posible incluir todos o alguno de estos elementos.

DT

DT son las siglas de «fecha y hora» (del inglés Date and Time).

El tipo DT se codifica en BCD con formato de 64 bits y contiene la información siguiente:

- el año, codificado en un campo de 16 bits;
- el mes, codificado en un campo de ocho bits;

- el día, codificado en un campo de ocho bits;
- la hora, codificada en un campo de ocho bits;
- los minutos, codificados en un campo de ocho bits;
- los segundos, codificados en un campo de ocho bits.

NOTA: Los ocho bits de menor valor no se utilizan.

El tipo **DT** se introduce de este modo:

DT# <Año> - <Mes> - <Día> - <Hora> : <Minutos> : <Segundos>

En la tabla siguiente se muestran los límites superior e inferior de cada campo:

Campo	Límites	Comentario
Año	[1990,2099]	Año
Mes	[01,12]	El 0 a la izquierda siempre aparece en pantalla, pero puede omitirse en el momento de la introducción.
Día	[01,31]	Para los meses 01\03\05\07\08\10\12
	[01,30]	Para los meses 04\06\09\11
	[01,29]	Para el mes 02 (años bisiestos).
	[01,28]	Para el mes 02 (años no bisiestos).
Hora	[00,23]	El 0 a la izquierda siempre aparece en pantalla, pero puede omitirse en el momento de la introducción.
Minuto	[00,59]	El 0 a la izquierda siempre aparece en pantalla, pero puede omitirse en el momento de la introducción.
Segundo	[00,59]	El 0 a la izquierda siempre aparece en pantalla, pero puede omitirse en el momento de la introducción.

DWORD

DWORD es la abreviatura de «palabra doble» (del inglés Double Word).

El tipo **DWORD** se codifica en formato de 32 bits.

La tabla siguiente muestra los límites superior e inferior de las bases que pueden utilizarse:

Base	Límite inferior	Límite superior
Hexadecimal	16#0	16#FFFFFFFF
Octal	8#0	8#3777777777
Binaria	2#0	2#11111111111111111111111111111111

Ejemplos de representación:

Contenido de los datos	Representación en una de las bases
00000000000010101101110011011110	16#ADCDE
00000000000000010000000000000000	8#200000
00000000000010101011110011011110	2#10101011110011011110

E

EBOOL

EBOOL es la abreviatura del tipo «booleano extendido» (del inglés Extended Boolean). Una variable de tipo **EBOOL** proporciona un valor (0 [**FALSO**] ó 1 [**VERDADERO**]), así como flancos ascendentes o descendentes y capacidades de forzado.

Una variable de tipo **EBOOL** ocupa un byte de memoria.

El byte se divide en:

- un bit para el valor
- un bit para el bit de historial (cada vez que el objeto del estado varía, el valor se copia en el bit de historial)
- un bit para el bit de forzado (igual a cero si el objeto no está forzado e igual a uno si el bit está forzado)

El tipo de valor predeterminado de cada bit es 0 (**FALSO**).

EDT

EDT son las siglas de «tipo de datos elementales» (del inglés Elementary Data Type).

Estos tipos son:

- **BOOL**
- **EBOOL**
- **WORD**
- **DWORD**
- **INT**
- **DINT**
- **UINT**
- **UDINT**
- **REAL**
- **DATE**
- **TOD**
- **DT**

EF

Son las siglas de «función elemental» (del inglés Elementary Function).

Se trata de un bloque que se utiliza en un programa y que realiza una función de software predefinida.

Las funciones no tienen información de estado interna. Si se realizan varias invocaciones de una misma función con los mismos parámetros de entrada, los valores de salida serán idénticos. Encontrará información detallada sobre la ejecución de esta función en «[Bloque de funciones (instancia)]». Al contrario de lo que sucede al ejecutar bloques de funciones, las ejecuciones de funciones sólo tendrán una salida sin nombre, ya que el nombre será el mismo que el de la función. En FBD, cada ejecución se expresa mediante un [número] unívoco a través del bloque gráfico; este número se genera automáticamente y no puede modificarse.

El usuario puede ubicar y configurar estas funciones en el programa para ejecutar la aplicación.

También pueden desarrollarse otras funciones mediante el kit de desarrollo del software SDKC.

EFB

Son las siglas de «bloque de funciones elemental» (del inglés Elementary Function Block).

Se trata de un bloque que se utiliza en un programa y que realiza una función de software predefinida.

Los EFB tienen estados y parámetros internos. Aún cuando las entradas sean idénticas, los valores de salida pueden ser diferentes. Por ejemplo, un contador tiene una salida que indica que se ha alcanzado el valor de preselección. Esta salida se establece en 1 cuando el valor actual es igual al valor de preselección.

Ejecución cíclica

La tarea maestra se ejecuta de forma cíclica o periódica. La ejecución cíclica consiste en ciclos concatenados sin tiempo de espera entre ellos.

Ejecución periódica

La tarea maestra se ejecuta de forma cíclica o periódica. En la modalidad periódica, el usuario determina un tiempo específico (período) en el que se ejecutará la tarea maestro. Si se ejecuta dentro de este tiempo, se generará un tiempo de espera antes del ciclo siguiente. Si se ejecuta cuando ya ha terminado este tiempo, un sistema de control indicará que se ha sobrepasado el límite de tiempo. Si este desborde ha sido demasiado grande, el PLC se detendrá.

EN/ENO (Habilitar/Indicación de error)

EN significa **EN**able (Activar), y se trata de una entrada de bloque opcional.

Si EN = 0, el bloque no se activa, su programa interno no se ejecuta y ENO se establece en 0.

Si EN = 1, el programa interno del bloque se ejecuta y el sistema establece ENO en 1. Si surge algún error, ENO se pondrá a 0.

ENO corresponde a **Error NOT**ification (Notificación de error) y es la salida asociada a la entrada opcional EN.

Si ENO se define como 0 (debido a que EN = 0 o a que se ha producido un error de ejecución)

- las salidas de los bloques de función permanecen en el estado en el que se encontraban en el último ciclo de exploración correcto ejecutado, y
- las salidas de las funciones y los procedimientos se ponen a «0».

NOTA: Si EN no se ha conectado, se establece automáticamente en 1.

Escalón

Un escalón es el equivalente de una secuencia en LD; otros términos relacionados son «Red Ladder» o, de forma más general, «Red». Un escalón se inscribe entre dos barras potenciales de un editor LD y se compone de un grupo de elementos gráficos interconectados mediante conexiones verticales u horizontales. Las dimensiones de un escalón son de 17 a 256 líneas y de 11 a 64 columnas como máximo.

Estructura

Vista del explorador de proyectos que representa la estructura del proyecto.

F**FBD**

FBD son las siglas de «diagrama de bloques de función» (del inglés Function Block Diagram).

FBD es un lenguaje de programación gráfico que funciona como si se tratara de un esquema lógico. Además de los bloques lógicos simples (Y, O, etc.), cada función o bloque de funciones del programa se representa mediante esta forma gráfica. En cada bloque, las entradas se sitúan a la izquierda y las salidas, a la derecha. Las salidas de los bloques se pueden conectar a las entradas de otros bloques para formar expresiones complejas.

FFB

Siglas que incluyen EF (función elemental), EFB (bloque de función elemental) y DFB (bloque de función derivado).

Flash Eprom

Tarjeta de memoria PCMCIA que contiene el programa y las constantes de la aplicación.

FNES

FNES son las siglas de «ficheros neutros de E/S» (del francés Fichiers Neutres d'Entrées Sorties).

El formato FNES realiza las descripciones mediante una estructura de árbol de los PLC en lo referente a los bastidores, las tarjetas y los canales.

Se basa en la normativa CNOMO (comité de normalisation des outillages de machines outils, comité de normalización de los medios de producción).

Función

Consulte EF.

Función elemental

Consulte EF.

G

GRAY

El código Gray o código «binario reflejado» se utiliza para codificar valores numéricos desarrollados en una cadena de configuraciones binarias que pueden diferenciarse por el cambio de estado de tan sólo un bit.

Este código puede utilizarse, por ejemplo, para evitar que se produzca el siguiente evento aleatorio: en un código binario puro, el cambio del valor 0111 a 1000 puede producir números aleatorios comprendidos entre 0 y 1.000, ya que los bits no cambian el valor en su conjunto de forma simultánea.

Equivalencia entre decimal, BCD y Gray:

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

H

Hipervínculo

La función de hipervínculo permite que se creen vínculos entre el proyecto y los documentos externos. Es posible crear hipervínculos en todos los elementos del directorio del proyecto, en las variables, en los objetos de la pantalla de procesamiento, etc.

Los documentos externos pueden ser páginas web o archivos (xls, pdf, wav, mp3, jpg, gif, etc.).

I

IEC 61131-3

Normativa internacional: relativa a los controles de software programables.

Apartado 3: relativo a los lenguajes de programación.

IL

IL son las siglas de «lista de instrucciones» (del inglés Instruction List).

Este lenguaje consiste en una serie de instrucciones básicas.

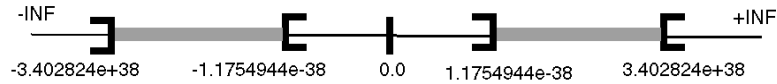
Este lenguaje es muy similar al lenguaje ensamblador utilizado en los procesadores de programa.

Cada instrucción está compuesta por un código de instrucción y por un operando.

INF

Se utiliza para indicar que un número sobrepasa los límites permitidos.

Si se trata de números enteros, los rangos de valores (que se muestran en color gris en el gráfico) son los siguientes:



Si el resultado de un cálculo es:

- Menor que $-3,402824\text{e}+38$, aparece el símbolo $-\text{INF}$ (para -infinito).
- Mayor que $+3,402824\text{e}+38$, aparece el símbolo $+\text{INF}$ (para +infinito).

Instancia DFB

Un tipo de instancia DFB se produce cuando se llama a una instancia desde un editor de lenguaje.

La instancia procesa un nombre, interfaces de entrada/salida, las variables públicas y privadas se duplican (una duplicación por instancia, el código no se duplica).

Un tipo DFB puede disponer de varias instancias.

Instanciar

Instanciar un objeto significa asignarle un espacio de memoria cuyo tamaño dependerá del tipo de objeto que se va a instanciar. Cuando se instancia un objeto, éste está disponible y el programa puede manipularlo.

INT

`INT` es la abreviatura del formato de entero simple, que está codificado en 16 bits.

Los límites superior e inferior son: $-(2 \text{ elevado a } 31)$ a $(2 \text{ elevado a } 31) - 1$.

Ejemplo:

-32.768 , 32.767 , $2\#1111110001001001$, $16\#9\text{FA4}$.

IODDT

IODDT son las siglas de «tipo de datos derivados de entrada/salida» (del inglés Input/Output Derived Data Type).

El término IODDT designa un tipo de datos estructurados que representa un módulo o un canal de un módulo de PLC. Cada módulo experto de una aplicación posee sus propios IODDT.

L

LD

LD son las siglas de «diagrama de contactos» (del inglés Ladder Diagram).

LD es un lenguaje de programación que representa las instrucciones que se deben ejecutar en forma de diagramas gráficos muy similares a los esquemas eléctricos (contactos, bobinas, etc.).

Literales de base 10

Los valores literales de base 10 se utilizan para representar valores enteros decimales. Este valor puede ir precedido de los signos «+» y «-». El carácter «_» se puede utilizar en estos valores literales, pero no tendrá ningún significado.

Ejemplo:

-12, 0, 123_456, +986

Literales de base 16

Los valores literales de base 16 se utilizan para representar valores enteros hexadecimales. La base está determinada por el número «16» y el signo «#». No se permite el uso de los signos «+» y «-». Para facilitar su lectura, puede utilizar el signo «_» entre bits.

Ejemplo:

16#F_F o 16#FF (255 en formato decimal)

16#F_F o 16#FF (224 en formato decimal)

Literales de base 2

Los valores literales de base 2 se utilizan para representar enteros binarios. La base está determinada por el número «2» y el signo «#». No se permite el uso de los signos «+» y «-». Para facilitar su lectura, puede utilizar el signo «_» entre bits.

Ejemplo:

2#1111_1111 ó 2#11111111 (255 en formato decimal)

2#1110_0000 ó 2#11100000 (224 en formato decimal)

Literales de base 8

Los valores literales de base 8 se utilizan para representar enteros octales. La base está determinada por el número «8» y el signo «#». No se permite el uso de los signos «+» y «-». Para facilitar su lectura, puede utilizar el signo «_» entre bits.

Ejemplo:

8#3_77 ó 8#377 (255 en formato decimal)

8#34_0 ó 8#340 (224 en formato decimal)

Literales de tiempo

Las unidades de tipo `TIME` son las siguientes: días (`d`), horas (`h`), minutos (`m`), segundos (`s`) y milisegundos (`ms`). Un valor literal del tipo `TIME` se representa mediante una combinación de las unidades anteriormente mencionadas precedida por `T#`, `t#`, `TIME#` o `time#`.

Ejemplos: `T#25h15m`, `t#14`, `7S`, `TIME#5d10h23m45s3ms`

Literales enteros

Los literales enteros se utilizan para introducir valores enteros en el sistema decimal. Los valores pueden ir precedidos de un signo (+/-). Los guiones bajos simples (`_`) entre números no aportan ningún significado.

Ejemplo:

`-12`, `0`, `123_456`, `+986`

Literales reales

Un valor literal real es un número expresado en uno o más decimales.

Ejemplo:

`-12.0`, `0.0`, `+0.456`, `3.14159_26`

Literales reales con exponente

Un valor literal decimal se puede expresar utilizando la anotación científica estándar. La representación es la siguiente: mantisa + exponente.

Ejemplo:

`-1,34E-12` ó `-1,34e-12`

`1,0E+6` ó `1,0e+6`

`1,234E6` ó `1,234e6`

M

MATRIZ

Una `MATRIZ` es una tabla de elementos del mismo tipo.

La sintaxis es la siguiente: `MATRIZ [<terminales>] DE <Tipo>`

Ejemplo:

MATRIZ [1..2] DE BOOL es una tabla monodimensional formada por dos elementos de tipo **BOOL**.

MATRIZ [1..10, 1..20] DE INT es una tabla bidimensional formada por 10 x 20 elementos de tipo **INT**.

Módulo funcional

Un módulo funcional es un grupo de elementos de programa (secciones, subprogramas, pasos de macro, tablas de animación, pantallas de ejecución, etc.), cuyo fin es realizar la función de un sistema de automatización.

Un módulo funcional, a su vez, puede dividirse en módulos funcionales de menor nivel, que realizan una o más subfunciones de la función principal del sistema de automatización.

Monotarea

Aplicación que incluye una única tarea, por lo que será, necesariamente, la tarea maestra.

N

NAN

Se utiliza para indicar que el resultado de una operación no es un número (**NAN** = Not A Number).

Ejemplo: cálculo de la raíz cuadrada de un número negativo.

NOTA: La norma IEC 559 define dos clases de **NAN**: **NAN** de reposo (**QNaN**) y **NaN** de señalización (**SNaN**) **QNaN** es un **NAN** con el bit de fracción de mayor valor definido, y un **SNaN** es un **NAN** con el bit de fracción de mayor valor eliminado (número de bit 22). Se permite la propagación de los **QNaN** mediante operaciones aritméticas sin señalar una excepción. **SNaN** señala generalmente una excepción de operación no válida, siempre que aparecen como operandos en operaciones aritméticas (consulte %SW17 y %S18).

O

Objeto de E/S

Un objeto de E/S es un objeto de lenguaje implícito o explícito para un módulo de función experto o un dispositivo E/S de un bus de campo. Pueden ser de los siguientes tipos: %Ch, %I, %IW, %ID, %IF, %Q, %QW, %QD, %QF, %KW, %KD, %KF, %MW, %MD y %MF.

La dirección topológica de un objeto depende de la posición del módulo en el bastidor o de la posición del dispositivo en el bus.

Para los PLC Premium/Atrium, las instancias de tipo doble de los datos ubicados (%MD<i>, %KD<i>) o flotantes (%MF<i>, %KF<i>) deben localizarse mediante un tipo de entero (%MW<i>, %KW<i>). Sólo los objetos de E/S posibilitan la localización de instancias de tipo (%MD<i>, %KD<i>, %QD, %ID, %MF<i>, %KF<i>, %QF, %IF) mediante su dirección topológica (por ejemplo, %MD0.6.0.11, %MF0.6.0.31).

Para PLC Modicon M340, las instancias de tipo doble de datos ubicados (%MD<i>, %KD<i>) o flotantes (%MF<i>, %KF<i>) no están disponibles.

Objetos SFC

Un objeto SFC es una estructura de datos que representa las propiedades de estado de una acción o transición de una gráfica secuencial.

P

Palabra clave

Una palabra clave es una combinación única de caracteres que se utiliza como elemento sintáctico de lenguaje de programación (consulte la definición del anexo B de la normativa IEC 61131-3. Todas las palabras clave utilizadas en Unity Pro y en esta normativa se enumeran en el anexo C de la normativa IEC 61131-3. No pueden utilizarse como identificadores en el programa [nombres de variables, secciones, tipos de DFB, etc.]).

Pantalla de operador

Se trata de un editor integrado en Unity Pro que se utiliza para facilitar el funcionamiento de un proceso automatizado. El usuario regula y controla el funcionamiento de la instalación y, en caso de que surjan problemas, puede actuar de forma rápida y sencilla.

Paso de macro

Un paso de macro es la representación simbólica de un conjunto único de pasos y transiciones, comenzando con un paso de entrada y terminando con un paso de salida.

Un paso de macro puede llamar a otro paso de macro.

Procedimiento

Los procedimientos se consideran funciones desde el punto de vista técnico. La única diferencia con respecto a las funciones elementales es que los procedimientos pueden ocupar más de una salida y que admiten el tipo de datos `VAR_IN_OUT`. Los procedimientos no se diferencian visualmente de las funciones elementales.

Los procedimientos constituyen un suplemento de la normativa IEC 61131-3.

Procesamiento de eventos

Procesamiento de eventos 1 es una sección del programa que se abre con un evento. Las instrucciones programadas en esta sección se ejecutan cuando el procesador recibe un evento de una aplicación de software (temporizador) o un evento de hardware (módulo específico de la aplicación).

Los procesos de eventos tienen prioridad frente a otras tareas, y se ejecutan en el momento en que se detecta el evento en cuestión.

El proceso de evento EVT0 tiene la mayor prioridad. El resto tiene el mismo nivel de prioridad.

NOTA: Para M340, los eventos de E/S con el mismo nivel de prioridad se almacenan en un FIFO y se tratan en el orden en el que se reciben.

Todos los temporizadores tienen el mismo nivel de prioridad. Cuando varios temporizadores finalizan al mismo tiempo, el número de temporizador más bajo se procesa en primer lugar.

La palabra de sistema `%SW48` cuenta el número de eventos de E/S procesados.

Protección

Opción que evita que se lea el contenido del elemento de un programa (protegido contra lectura) o que se escriban o modifiquen los contenidos del elemento de un programa (protegido contra lectura y escritura).

La protección se confirma mediante una contraseña.

Punto de observación

Se utiliza en la modalidad de «depuración» de la aplicación.

Permite que la visualización de variables animadas se sincronice con la ejecución de un elemento de programa (que contiene el punto de observación) para averiguar sus valores en ese preciso punto del programa.

Punto de parada

Se utiliza en la modalidad de «depuración» de la aplicación.

Es exclusivo (no puede haber dos o más al mismo tiempo) y, una vez alcanzado, envía una señal al procesador para que detenga la ejecución del programa.

Si se utiliza en la modalidad conectada, se puede posicionar en uno de los elementos de programa siguientes:

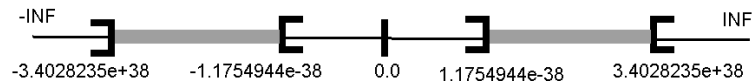
- Red LD
- Secuencia de texto estructurado o lista de instrucciones
- Línea de texto estructurado (modalidad de líneas)

R

REAL

El tipo REAL es un tipo codificado en 32 bits.

En el siguiente diagrama, los rangos de valores posibles aparecen coloreados en gris.



Si el resultado de un cálculo es:

- Un número entre $-1,175494\text{e-}38$ y $1,175494\text{e-}38$, se considera un **DEN**.
- Menor que $-3,4028234\text{e+}38$, aparece el símbolo **-INF** (para -infinito).
- Mayor que $+3,4028234\text{e+}38$, aparece el símbolo **INF** (para +infinito).
- Indefinido (raíz cuadrada de un número negativo), aparecerá el símbolo **NAN** o **NAN**.

NOTA: La norma IEC 559 define dos clases de **NAN**: **NAN** de reposo (**QNAN**) y **NAN** de señalización (**SNAN**). **QNAN** es un **NAN** con el bit de fracción de mayor valor definido, mientras que **SNAN** es un **NAN** con el bit de fracción de mayor valor eliminado (número de bit 22). Se permite la propagación de los **QNAN** mediante operaciones aritméticas sin señalar una excepción. **SNAN** señala generalmente una excepción de operación no válida siempre que aparece como operando en operaciones aritméticas (consulte %SW17 y %S18).

NOTA: Cuando un operando es **DEN** (número no normalizado), el resultado no es significativo.

Red

Se utiliza principalmente en las comunicaciones y se trata de un grupo de estaciones que se comunican entre sí. El término «red» también se utiliza para definir un grupo de elementos gráficos conectados entre sí. Este grupo constituye, por lo tanto, una parte de un programa que puede estar compuesto por un grupo de redes.

RS 232C

Comunicación serie estándar que define el voltaje del servicio siguiente:

- Una señal de +12 V indica un 0 lógico.
- Una señal de -12 V indica un 1 lógico.

Sin embargo, en caso de que se atenúe la señal, existe una detección que se proporciona hasta los límites de -3 V y +3 V.

Entre estos dos límites, la señal se considerará no válida.

Las conexiones RS 232 son bastante sensibles a las interferencias. La normativa especifica que no se debe superar la distancia de 15 m ni los 9.600 baudios (bits/s).

RS 485

Normativa de conexión serie que funciona en un diferencial de 10 V/+5 V. Utiliza dos conductores para enviar/recibir. Las salidas de «3 estados» les permiten conectarse a la modalidad de escucha cuando finaliza la transmisión.

RUN

Función que permite que se inicie el programa de la aplicación del PLC.

RUN Auto

Función que permite que el programa de la aplicación del PLC se ejecute para que se inicie automáticamente en caso de que se produzca un arranque en frío.

S**Sección**

Módulo de programa que pertenece a una tarea que se puede escribir en el lenguaje elegido por el programador (FBD, LD, ST, IL o SFC).

Una tarea puede estar compuesta por distintas secciones; el orden de ejecución de estas secciones corresponde a su orden de creación, y se puede modificar.

SFC

Son las siglas de «gráfica de función secuencial» (del inglés Sequential Function Chart).

SFC permite representar gráficamente y de forma estructurada el funcionamiento de un sistema de automatización secuencial. Esta descripción gráfica del comportamiento secuencial de un sistema de automatización y de las distintas situaciones resultantes se realiza utilizando símbolos gráficos simples.

ST

ST son las siglas del lenguaje de «texto estructurado» (del inglés Structured Text).

Este lenguaje es un lenguaje de alto nivel similar a los lenguajes de programación de ordenadores. Permite estructurar series de instrucciones.

STRING

Las variables de tipo `STRING` son cadenas de caracteres de código ASCII. Cada cadena puede tener una longitud máxima de 65.534 caracteres.

Subrutina

Módulo de programa perteneciente a una tarea (Mast, Fast, Aux) que se puede escribir en el lenguaje elegido por el programador (FBD, LD, ST o IL).

Una subrutina sólo se puede llamar desde una sección o desde otra subrutina que pertenezca a la tarea en la que se declare.

T

Tarea

Grupo de secciones y subrutinas ejecutadas cíclica o periódicamente si se trata de la tarea MAST, o periódicamente si se trata de la tarea FAST.

Una tarea siempre tiene un nivel de prioridad y tiene asociadas entradas y salidas del PLC. Estas entradas y salidas se actualizarán en consecuencia.

Tarea FAST

Se trata de una tarea que se inicia periódicamente (la frecuencia se ajusta en la configuración del PC) y se utiliza para ejecutar una parte de la aplicación con un nivel superior de prioridad a la tarea MAST (maestro).

Tarea maestra

Tarea principal del programa.

Es obligatoria y se utiliza para ejecutar un procesamiento secuencial del PLC.

Tarea múltiple

Aplicación que comprende distintas tareas (tareas MAST, MAST, auxiliares o de procesamiento de eventos).

El orden de prioridad de ejecución de las tareas viene definido por el sistema operativo del PLC.

Tareas auxiliares

Tareas periódicas opcionales utilizadas para procesar procedimientos que no requieren un procesamiento rápido, por ejemplo: medición, ajuste, ayuda de diagnóstico, etc.

Tiempo de espera

En proyectos de comunicación, el tiempo de espera es un retraso tras el que se detiene la comunicación si no se recibe ninguna respuesta del dispositivo de destino.

TIME

El tipo `TIME` expresa una duración en milisegundos. Este tipo se codifica en formato de 32 bits y permite obtener períodos de 0 a $(2 \text{ elevado a } 32)-1$ milisegundos.

TIME_OF_DAY

Consulte `TOD`

TOD

`TOD` son las siglas de «hora del día» (del inglés Time of Day).

El tipo `TOD` se codifica en BCD con formato de 32 bits y contiene la información siguiente:

- la hora, codificada en un campo de ocho bits;
- los minutos, codificados en un campo de ocho bits;
- los segundos, codificados en un campo de ocho bits.

NOTA: Los ocho bits de menor valor no se utilizan.

El tipo `TOD` se introduce así: `TOD# <Hora> : <Minutos> : <Segundos>`

En la tabla siguiente se muestran los límites superior e inferior de cada campo:

Campo	Límites	Comentario
Hora	[00,23]	El 0 a la izquierda siempre aparece en pantalla, pero puede omitirse en el momento de la introducción.
Minuto	[00,59]	El 0 a la izquierda siempre aparece en pantalla, pero puede omitirse en el momento de la introducción.
Segundo	[00,59]	El 0 a la izquierda siempre aparece en pantalla, pero puede omitirse en el momento de la introducción.

Ejemplo: TOD#23:59:45.

Token

Cada paso activo de una SFC se conoce como token.

Token individual

Modalidad de servicio de una gráfica SFC en la que no puede haber varios pasos activos al mismo tiempo.

Token múltiple

Modalidad de servicio de un SFC. En la modalidad de token múltiple, el SFC puede disponer de varios pasos activos al mismo tiempo.

U

UDINT

UDINT es la abreviatura del formato «entero doble sin signo» (del inglés Unsigned Double Integer), que se codifica en formato de 32 bits. Los límites superior e inferior son: de 0 a (2 elevado a 32) - 1.

Ejemplo:

0,4294967295,2#11111111111111111111111111111111,8#377777777777,16#FFFFFFFF.

UINT

UINT es la abreviatura del formato «entero sin signo» (del inglés Unsigned Integer), que se codifica en formato de 16 bits. Los límites superior e inferior son: de 0 a (2 elevado a 16) - 1.

Ejemplo:

0,65535,2#1111111111111111,8#177777,16#FFFF.

V

Variable

Entidad de memoria del tipo `BOOL`, `WORD`, `DWORD`, etc., cuyos contenidos se pueden modificar desde el programa durante su ejecución.

Variable no ubicada

Una variable no ubicada es una variable cuya posición en la memoria del autómatas no puede conocerse. Las variables que no tienen asignadas direcciones se consideran no ubicadas.

Variable ubicada

Una variable ubicada es una variable cuya posición en la memoria del PLC se puede conocer. Por ejemplo, la variable `Presión_del_agua` se asocia a `%MW102`. Se dice entonces que `Presión_del_agua` está localizada.

Ventana de visualización

Esta ventana, a la que también se denomina ventana de observación, muestra las variables que no pueden animarse en los editores de lenguaje. Sólo aparecen las variables que son visibles en un momento determinado en el editor.

Vista de la función

Esta vista hace posible que se pueda ver parte del programa de la aplicación a través de los módulos funcionales que crea el usuario (consulte la definición de módulo funcional).

W

WORD

El tipo `WORD` se codifica en formato de 16 bits y se utiliza para procesar cadenas de bits.

En la tabla siguiente se muestran los límites superior e inferior de las bases que se pueden utilizar:

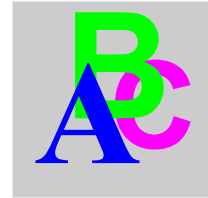
Base	Límite inferior	Límite superior
Hexadecimal	16#0	16#FFFF

Base	Límite inferior	Límite superior
Octal	8#0	8#177777
Binaria	2#0	2#1111111111111111

Ejemplos de representación

Contenido de los datos	Representación en una de las bases
0000000011010011	16#D3
1010101010101010	8#125252
0000000011010011	2#11010011

Índice



Symbols

%S, 162

%SW

genérico, 187

Modicon M340, 248

Premium, 217

Quantum, 231

A

ADD

IL, 501

AND

IL, 499

ST, 548

ANY_ARRAY, 309

ARRAY, 289

automático, arranque en RUN, 131

B

bloque de función derivado (DFB)

representación, 598

BOOL, 263

BYTE, 286

C

CAL, 505

caliente, arranque, 131

canal, estructura de datos, 299

CASE...OF...END_CASE

ST, 559

comparación

IL, 496

LD, 393

ST, 544

compatibilidad

tipos de datos, 313

Conformidad con IEC, 693

D

D

SFC, 442

DATE, 274

datos, instancias, 317

datos, tipos, 259

DDT, 288

derivado, bloque de función (DFB), 593

representación, 302

derivados, tipos de datos (DDT), 288, 292

DFB

representación, 598

diagnóstico, DFB, 641

DINT, 268

direccionamiento

entrada/salida, 324

instancias de datos, 324

DIV

IL, 502

DS

SFC, 442

DT, 276
DWORD, 286

E

EBOOL, 263
EDT, 259
EFB, 301
elemental, bloque de función (EFB), 301, 302
ELSE, 556
ELSIF...THEN, 557
EN/ENO
 FBD, 355
 IL, 516, 526, 533
 LD, 388
 ST, 576, 584, 589
entrada/salida
 direccionar, 324
EQ
 IL, 503
error, códigos, 645
estructura, 288
estructurado, texto (ST)
 estructura, 539
 instrucciones, 549
 lenguaje, 537, 572, 577, 586
 operadores, 544
etiquetas
 FBD, 359
 IL, 508
 LD, 392
 ST, 569
eventos, procesamiento, 96
EXIT, 565

F

FBD
 estructura, 346
 lenguaje, 345, 348
flotante, coma, 278
FOR...TO...BY...DO...END_FOR
 ST, 560
forzados, bits, 263

frío, arranque, 131
funcionamiento, 393

G

GE
 IL, 503
GT
 IL, 503

H

HALT, 159

I

IF...THEN...END_IF
 ST, 554
IN_OUT
 FBD, 357
 IL, 526, 534
 LD, 390
 ST, 584, 590
inicio en frío, 145
instrucciones, lista (IL)
 estructura, 489
 lenguaje, 487, 512, 517, 529
 operadores, 496
INT, 268

J

JMP
 FBD, 359
 IL, 506, 508
 LD, 392
 SFC, 450
 ST, 569

L

L
 SFC, 442

LD
 estructura, 374
 lenguaje, 373, 381
LD, operadores
 IL, 373
LE
 IL, 504
Límite de alineación, 296
LT, 504

M

memoria, estructuras, 115, 117
MOD
 IL, 502
 ST, 545
MUL
 IL, 501

N

NE
 IL, 504
NOT
 IL, 500

O

OR
 IL, 499
 ST, 548

P

P
 SFC, 442
P0
 SFC, 442
P1
 SFC, 442
palabras de sistema
 Modicon M340, 248
 Quantum, 231

privadas, variables
 DFB, 609
 FBD, 519, 579
públicas, variables
 DFB, 609
 IL, 518
 ST, 578

R

R
 IL, 498
 LD, 379
 SFC, 442
REAL, 278
REPEAT...UNTIL...END_REPEAT, 564
RETURN
 FBD, 359
 IL, 506
 LD, 392
 ST, 567

S

S
 IL, 498
 LD, 379
 SFC, 442
secciones, 82, 83
SFC
 estructura, 423
 lenguaje, 421, 439
SFCCHART_STATE, 425
SFCSTEP_STATE, 431
SFCSTEP_TIMES, 431
sistema, bits, 162
sistema, palabras, 187
 Premium, 217, 223
 Quantum, 237
STRING, 281
SUB
 IL, 501
subrutinas, 82, 86

T

tareas, 75, 79

 cíclicas, 90

 periódicas, 91

TIME, 270

tipos de datos elementales (EDT), 259

TOD, 275

U

UDINT, 268

UINT, 268

V

variables privadas

 FBD, 354, 387

variables públicas

 FBD, 353

 LD, 387

W

watchdogs

 monotarea, 93

 multitarea, 101

WHILE...DO...END_WHILE

 ST, 563

WORD, 286

X

XOR

 IL, 500

 ST, 548