STEP 7

PROGRAMACION AVANZADA

TODO LO QUE
SIEMPRE QUISISTE SABER
Y NADIE TE HABIA DICHO

Exclusión de responsabilidad.

Se ha probado el contenido de este libro de programación, y la concordancia entre el hardware descrito y el software. Sin embargo, es posible que se den algunas desviaciones con respecto a las diferentes posibilidades de los temas descritos, lo cual impide tomar garantía completa de los temas expuestos. Es necesario comprobar la concordancia del contenido de los temas con las aplicaciones propias a realizar, no aceptándose ninguna responsabilidad de ningún tipo por no adoptar dicha medida de precaución..

Autor: José Martínez Torres.
Fecha: 1999.
Edición: 1.0
Impreso en Valencia, España.

“Nuestros sueños son nuestra única vida real”

A mi familia, mi novia, y mi perro.

Colaboradores:

El presente libro se ha realizado gracias a la colaboración de:

Enrique Martí Navarro, por su testeo de los diferentes ejemplos del presente libro.
Raúl Salinas Inarejos, por la pegatina de la portada.
Joaquín Guerola, por sus aportaciones a las comunicaciones.

Agradecimientos:

Deseamos agradecer la ayuda prestada y el asesoramiento en aquellos temas que escapaban a nuestro conocimiento a las siguientes personas:

Miguel Madrid, por sus consejos sobre S7.
Enrique Suarez, por sus aportaciones sobre visualización.

Indice de contenidos

1
Introducción

La programación de los equipos Simatic S7 actualmente ha superado ya la fase de maduración, lo cual permite asegurar, sin aventurarse en demasía, que ya nos encontramos con el sistema mas avanzado en el campo de la automatización de procesos mediante PLC.

Sin embargo, la rápida evolución del mismo tanto a nivel de software como de hardware en los últimos dos años, con sucesivas versiones que acercan cada vez más el entorno de programación de PLC's a los actuales de programación de ordenadores (pese a que aún quedan grandes distancias que salvar, en especial en los procesos de depuración y auto rellenado), han obligado a que no existiera aún un manual de referencia que permitiese adquirir en un periodo razonable de tiempo los conceptos al respecto de este nuevo sistema. Dichos conocimientos se encontraban repartidos en multitud de manuales de referencia, o simplemente no documentados.

El objeto del presente manual no es introducir al lector en el manejo de PLC's de manera genérica. Tampoco está enfocado a aquellos que aún disponiendo de nociones de programación, no se hayan adentrado mínimamente en la programación de los equipos S7. Para todos ellos, les recomiendo el estudio de dichos conceptos mediante los manuales de cursos de aprendizaje Simatic S7 nivel I y nivel II.

Este libro esta enfocado a aquellos programadores que ya conocen perfectamente S5, que se han introducido ligeramente en S7 (incluso medianamente en el mismo), y desean descubrir aquellas particularidades del mismo que, ya sea por no haber tenido ocasión, o por no disponer aún de tiempo, no habían asimilado.

En las explicaciones se omiten aquellos aspectos que se consideran triviales, centrándose exclusivamente en los puntos conflictivos de los diferentes apartados tratados.

Espero que esta obra sea de utilidad a los diferentes programadores que como yo, disfrutan viendo un sistema de automatización, pero siempre piensan que puede existir una forma mejor y más sencilla de hacerlo.

José Martínez.
Valencia. 1999.

2
Filosofía de programación

2.1 "Mi idioma es el mejor…"

Cualquier programador de Simatic habrá tenido a lo largo de su vida numerosas discusiones con compañeros de trabajo al respecto del mejor lenguaje para programar los Simatic.

Los defensores del AWL, a menudo acérrimos y mayoría en este "mundillo", buscan siempre aquella subrutina magistral de la cual sentirse orgullosos, esperando no tener que volver a "meterle mano" a la misma unos meses después. Dicha defensa de un lenguaje único les lleva a pasarse hasta minutos enteros delante de una simple combinación de paralelos y series de contactos, sabiendo que no se cumple el RLO, pero sin acertar directamente con la causa del problema en el proceso técnico.

Los defensores del KOP, no bien considerados por los anteriores, normalmente estructuran bien el programa, pero suelen terminar necesitando una "caja" para cada nueva función que se les plantea, incapaces de recurrir a combinaciones aceptables de funciones en KOP.

Por último, los amantes del FUP (entre los que me encuentro), suelen cometer el error anterior, con el agravante de que Step 7 aún no está orientado a este tipo de programación, notándose como un "pegote" introducido a última hora en el Software.

La programación no obliga a utilizar un lenguaje común para todo el proyecto, siendo nuestra propia limitación a la hora de cambiar de idioma lo que nos obliga a obcecarnos en uno en particular, enmascarando sus defectos cuando utilizamos el lenguaje que no corresponde en una tarea de automatización.

La decisión no es pues **qué** lenguaje de programación utilizar, sino **cuando** utilizar cada uno. Para conocer dicha respuesta, deberemos aprender primero las ventajas de cada uno de ellos (y también los inconvenientes).

2.2 Ventajas y desventajas.

Repasemos las **ventajas del AWL**, el más utilizado en Simatic:

- Es el lenguaje que necesita menos instrucciones de programación de todos, con lo cual el programa ocupa igualmente menos código compilado. Esto permite optimizar la cantidad de memoria disponible en el PLC, pero sobre todo el tiempo de ciclo es menor, por lo que se pueden utilizar CPU's "lentas" para procesos relativamente rápidos, aún con programas considerables.
- Es el más indicado para operaciones de saltos condicionales en función de valores analógicos. Cualquier tratamiento analógico que sobrepase la simple comparación es el terreno del AWL.
- Permite introducir una gran cantidad de sentencias en la misma pantalla, con lo cual los test status no requieren de desplazamientos en la misma.

Pero llegan **los inconvenientes**…:

- La programación de secuencias de pasos en procesos (set y reset de etapas) carece de sentido en este lenguaje, ya que no se gana memoria y el programa se convierte en ininteligible hasta para el propio programador.
- El programar una línea debajo de otra lleva a los programadores a realizar "chorizos" de varias páginas sin darse cuenta (por lo menos a mí).

Veamos ahora las **ventajas del KOP**:

- Es muy sencillo ver en él los pasos de programa que no cumplen, y seguir las condiciones del proceso.
- Totalmente indicado para programadores más cercanos al mundo eléctrico que al informático en tareas de tratamiento digital (bobinas, set, reset…)

Y los **inconvenientes del KOP**:

- Las series y paralelos requieren tanto espacio en pantalla que enseguida se nos salen de la misma, por lo que en status nos debemos de desplazar a menudo. La solución a dicho problema es sencilla: utilizar marcas.

- El realizar comparaciones de salto analógicas es misión imposible, a poco que se complique el tema.
- Y el principal problema: las cajas de KOP necesitan una sistemática de proceso por parte del Step 7 que hace que no se optimice el código de las mismas, por lo que el programa haciendo lo mismo va más lento.

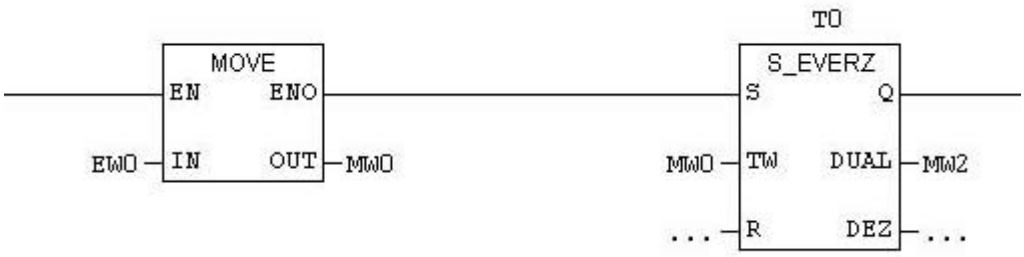


Figura 1. Aspecto de un segmento de programa en lenguaje KOP.

```
U(
L     EW     0
T     MW     0
SET
SAVE
CLR
U     BIE
)
L     MW     0
SE    T      0
NOP   0
L     T      0
T     MW     2
NOP   0
NOP   0
```

Figura 2 . Aspecto de ese mismo segmento de programa convertido en AWL. Obsérvese la cantidad de instrucciones innecesarias desde el punto de vista funcional en AWL. El código no está nada optimizado.

Por último las **ventajas del FUP** son:

- Permite realizar gran cantidad de series y paralelos en la misma pantalla, con lo cual se acerca a la ventaja del AWL, pero con mayor claridad en el diagnóstico.
- Es el indicado para los programadores electrónicos.

Y los inconvenientes del FUP son:

- No es útil, al igual que le pasaba al KOP, para tratar valores analógicos ni condiciones de salto.
- Sufre el mismo problema de optimización de código en el tratamiento que realiza del mismo el Step 7.
- Los programadores de KOP suelen no identificar de una manera rápida las combinaciones *and* y *or* en un status de programa.

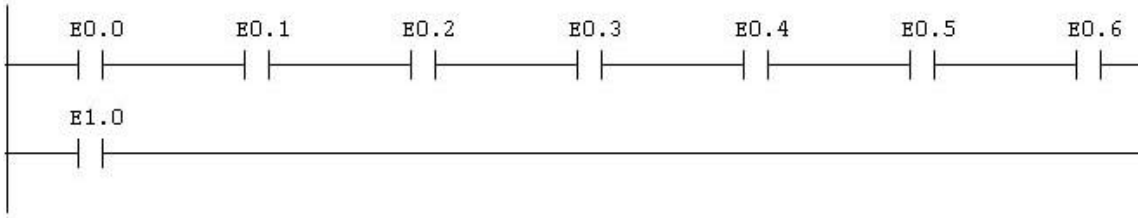


Figura 3 . El problema del KOP a la hora de tratar estados digitales. Se puede apreciar que no se puede visualizar todo el segmento en pantalla, y por consiguiente conocer el estado de la bobina.

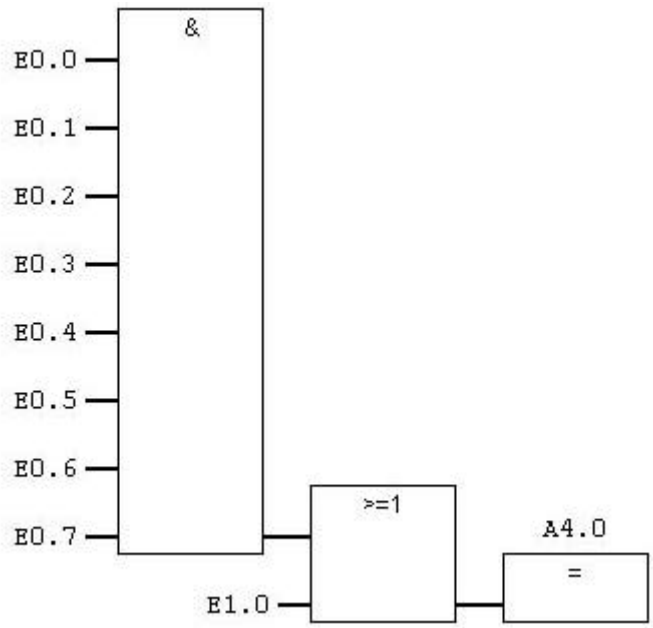


Figura 4 . Este problema no existe en FUP, en el cual las condiciones se agrupan de manera vertical, por lo que se pueden visualizar más en status.

2.3 ¿Qué idioma debo hablar?

La respuesta es evidente, cada vez uno. Para las tareas que no sean setear, resetear o activar bits, el AWL es sin dudas el lenguaje a utilizar. Las ventajas del mismo sobrepasan ampliamente los inconvenientes.

Sin embargo para todas las activaciones (series y contactos que van a parar a bobinas, ya sean enclavadas o no) la decisión debe de ser KOP o FUP, dependiendo del gusto de cada uno, debiendo, en igualdad de capacidades para comprender la programación (espero que en sentido positivo del término), decantarse por el FUP.

2.4 Tipos de instalaciones

Resumir todas las posibilidades de programación en un par de tipos de instalaciones, evidentemente es una simplificación excesiva Sin embargo, como una primera aproximación a la clasificación de las instalaciones, podríamos hablar sin temor a equivocarnos en demasía de:

- Instalaciones globales
- Procesos y
- Máquinas

Veamos las particularidades de estas automatizaciones y cual sería la forma ideal de programación en ellas.

2.4.1 Instalaciones globales.

Un autómata controla un gran número de pequeños procesos, cuya conexión entre ellos es prácticamente nula a nivel de automatización, aunque a nivel humano se requiere un funcionamiento común. Es el caso típico de grandes superficies con baja automatización en sus procesos, p. Ej. una piscina, un parque, un supermercado, etc…

Al ser los procesos independientes entre ellos (el alumbrado de la piscina no está coordinado con el control de temperatura del agua de las duchas), es conveniente estructurar la programación en subrutinas que engloban cada una un proceso, y que son llamadas desde la OB1.

2.4.2 Procesos.

Los procesos son instalaciones en las cuales las tareas de automatización sí suelen estar interconexionadas unas con otras. Ejemplos de este tipo de instalaciones los constituyen un transporte de líneas de carrocerías de coches, o la fabricación de productos para la laca de azulejos mediante compuestos químicos.

En estas automatizaciones no existe un principio de funcionamiento (procesos continuos), y suelen estar acompañadas de sistemas scada para el control estadístico de producción.

La subdivisión del proceso en subrutinas debe de intentarse en la medida de lo posible, y estructurar el programa para un fácil tratamiento mediante recetas desde scada u OP.

2.4.3 Máquinas.

Las máquinas se caracterizan siempre por ser un proceso no continuo en el tiempo (siempre posee un botón de marcha y de parada). Aquí adquiere gran importancia las recetas de producción, por lo que todos los parámetros deben de estructurarse en DB's para que sea fácil su tratamiento desde una OP. No se suele gastar scada, sino más bien con una OP de líneas a menudo es suficiente.

En este tipo de automatizaciones es importantísimo estructurar el programa en etapas o pasos, ya que las diferentes secuencias de la máquina están predefinidas, y al determinar etapas para estas transiciones el tratamiento de errores a posteriori es muy sencillo.

No siempre la etapa será única, ya que existen procesos en los que dos o más etapas se pueden superponer en el tiempo en un determinado instante. En dicho caso, es necesario utilizar marcas para determinar a través de un visualizador el estado actual de la máquina de manera unívoca.

2.5 Lo que nunca se debe hacer.

No es que se pueda decir de manera concluyente cual es la mejor forma de programar (por desgracia). Sin embargo, existen algunas acciones que conducen a cualquier programador, más tarde o más temprano, a encontrarse en callejones sin salida de los que uno suele salir más mal que bien, y sólo después de haber perdido mucho tiempo y muchas neuronas.

La primera de ellas es definir simbólicamente todas las variables **ANTES** de utilizarlas en el programa. Mucha gente piensa que es una pérdida de tiempo el realizar esta acción, ya que desconocen su verdadero valor, que no es como se piensan el recordarnos de una manera más próxima al proceso el origen de la señal que estamos tratando, sino el evitar el gran problema de la programación: el **solapamiento de variables**.

Este es el principal problema de una programador, utilizar una variable en dos sitios, o utilizar una variable como bit, y a su vez como byte o palabra de manera inadvertida. Mediante la simbolización de las variables este problema desaparece, siempre y cuando además el programador siga **la regla de las zonas**. Esta regla se basa en agrupar el los mapas de memoria las variables por su tamaño, de tal manera que p. Ej. los bits de marcas comenzaremos a simbolizarlos en la M0.0 y terminarán en la M49.7, las palabras empezarán en la MW50 y terminarán en la MW98, y los números reales o dobles enteros en la MD100 hasta la MD200.

La segunda regla para no tener problemas es setear o activar las variables **SOLO EN UN SITIO**. Es la segunda pesadilla con la que uno se enfrenta programando, las condiciones para la activación de una salida no se cumplen, y sin embargo se activa. Siempre es por lo mismo: en otra parte del programa está activándose dicha salida, y la llamada a dicho segmento va después de la que en estos momentos estamos observando, por lo que como tiene prioridad se activa.

La bobina, el set y el reset deben ser únicos en un programa, por lo que suele ser necesario utilizar marcas auxiliares en conjunción con *or* y *and* lógicas para activarlas. El ejemplo típico es el funcionamiento en manual o automático de un motor. Existe un bit que indica si dicho motor está en manual o automático, y unas condiciones de funcionamiento en automático y otras de manual.

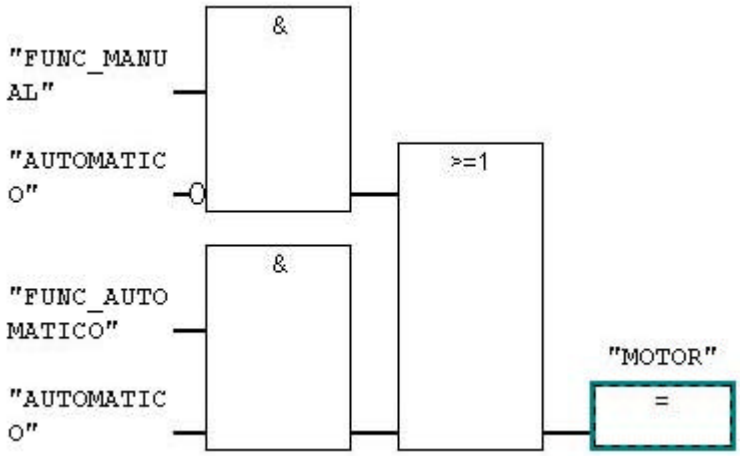


Figura 5 . Ejemplo de arranque de motores. Aunque puede activarse por varias condiciones, solamente lo hará en este segmento dentro del programa. Nos ayudaremos de marcas para determinar las diferentes condiciones de activación. Esto nos permite acudir rápidamente para saber cual es el motivo de que un motor no arranque.

3

Hardware y memoria en S7

Lo primero que uno debe conocer antes de empezar a programar es de qué hardware dispone, y donde (en qué zonas de memoria del PLC) se encuentra ubicado.

El presente capítulo se introduce en la memoria de la CPU, y su forma de trabajar.

3.1 La periferia del S7.

3.1.1 Direccionamiento fijo por puesto o variable.

El direccionamiento de las entradas y salidas de nuestra periferia en S7 puede ser de dos tipos:

- **Fijo por puesto:** Siempre que no dispongamos de una CPU con tarjeta maestra DP integrada. En este caso, dependiendo de donde coloquemos el módulo, adoptará una periferia concreta, determinada por la tabla adjunta inferior. Para conseguir esto, cada tarjeta ocupa realmente 4 bytes del área digital, y 16 bytes del área analógica, independientemente de si el módulo colocado es de 16 o 32 puntos.

Rack	Module Start Addresses	Slot Number										
		1	2	3	4	5	6	7	8	9	10	11
0	Digital	PS	CPU	IM	0	4	8	12	16	20	24	28
	Analog				256	272	288	304	320	336	352	368
1[1]	Digital	–		IM	32	36	40	44	48	52	56	60
	Analog	–			384	400	416	432	448	464	480	496
2[1]	Digital	–		IM	64	68	72	76	80	84	88	92
	Analog	–			512	528	544	560	576	592	608	624
3[1]	Digital	–		IM	96	100	104	108	112	116	120	124[2]
	Analog	–			640	656	672	688	704	720	736	752[2]

1 Not with CPU 312 IFM/313
2 Not with CPU 314 IFM

Figura 6 . Direccionamiento fijo por puesto y la asignación de periferia según slot de la tarjeta.

- **Variable:** Cuando el equipo posee la funcionalidad DP integrada en la CPU. En este caso, el Hardware de Step 7 nos asigna una zona de periferia que se encuentre libre, pero podemos redireccionar nuestro módulo a otra zona, siempre que la misma aún no esté ocupada. La optimización de la periferia es mucho mejor, ya que podemos redistribuirla a nuestro antojo.

3.1.2 Tipos de acceso a la periferia.

Existen dos tipos de periferia: con imagen de proceso y sin la misma. La imagen de proceso es una zona de memoria dentro del PLC donde se copian los estados de cierta zona de la periferia el PLC al principio del ciclo de programa. A partir de ese momento no se vuelve a acceder a la periferia, con lo cual se asegura una consistencia de los valores durante todo el ciclo de programa. No se puede dar el caso de que en una subrutina una entrada tenga un valor, y un par de líneas más abajo cambie el mismo (lo cual sería catastrófico).

La periferia con imagen de proceso depende del tipo de CPU, y se encuentra descrita más adelante. Al resto de la misma se debe de acceder mediante:

- Si la cantidad de bytes a leer es de 1, 2, o 4 bytes, mediante instrucciones de carga y transferencia directa de la periferia. Ejemplos de ello sería *L PEW128* o *T PAW128*.
- Si la cantidad de bytes es impar, o es superior a 4 bytes, se debe de utilizar la SFC15 y SFC14 del sistema para la lectura y escritura en periferia manteniendo la consistencia de la información (que todos los bytes hayan sido actualizados en el mismo ciclo de scan de CPU).

3.1.3 Periferia integrada

Las CPU´s IFM disponen de entradas y salidas integradas, cuyo direccionamiento se encuentra alojado en el final del área de memoria del PLC. El direccionamiento de estas señales integradas es el siguiente:

CPU 312 IFM

ENTRADAS / SALIDAS	ÁREA
10 ED	E 124.0 a E 125.1
6 SD	A 124.0 a E 125.1

CPU 314 IFM

ENTRADAS / SALIDAS	ÁREA
20 ED	E 124.0 a E 126.3
16 SD	E 124.0 a E 125.1
4 EA	EW 128 a EW 134
1 SA	AW 128

3.1.4 PAE y PAA.

Veamos las zonas de PAE y PAA así como la periferia máxima que pueden soportar:

CPU 312 IFM

TIPO	AREAS	CON PAE/PAA	CANTIDAD MAXIMA
ENTRADAS DIGITALES	E 0.0	E 0.0	E/S DIG. 128
	E 125.7	E 31.7	
SALIDAS DIGITALES	A0.0	A 0.0	
	A 125.7	A 31.7	
ENTRADAS ANALOGICAS	EW 256		E/S ANAL. 32
	EW 368		
SALIDAS ANALOGICAS	AW 256		
	AW 368		

CPU 313

TIPO	AREAS	CON PAE/PAA	CANTIDAD MAXIMA
ENTRADAS DIGITALES	E 0.0	E 0.0	E/S DIG. 128
	E 125.7	E 31.7	
SALIDAS DIGITALES	A0.0	A 0.0	
	A 125.7	A 31.7	
ENTRADAS ANALOGICAS	EW 256		E/S ANAL.
	EW 368		

SALIDAS ANALOGICAS	AW 256 AW 368		32

CPU 314 IFM

TIPO	AREAS	CON PAE/PAA	CANTIDAD MAXIMA
ENTRADAS DIGITALES	E 0.0 E 127.7	E 0.0 E 123.7	E/S DIG. 512
SALIDAS DIGITALES	A0.0 A 127.7	A 0.0 A 123.7	
ENTRADAS ANALOGICAS	EW 256 EW 752		E/S ANAL. 64
SALIDAS ANALOGICAS	AW 256 AW 752		

Nota: En la CPU 314 IFM no se puede utilizar el último slot del tercer bastidor de ampliación.

CPU 314

TIPO	AREAS	CON PAE/PAA	CANTIDAD MAXIMA
ENTRADAS DIGITALES	E 0.0 E 127.7	E 0.0 E 127.7	E/S DIG. 512
SALIDAS DIGITALES	A0.0 A 127.7	A 0.0 A 127.7	
ENTRADAS ANALOGICAS	EW 256 EW 766		E/S ANAL. 64
SALIDAS ANALOGICAS	AW 256 AW 766		

CPU 315

TIPO	AREAS	CON PAE/PAA	CANTIDAD MAXIMA
ENTRADAS DIGITALES	E 0.0	E 0.0	E/S DIG. 1024
	E 255.7	E 127.7	
SALIDAS DIGITALES	A0.0	A 0.0	
	A 255.7	A 127.7	
ENTRADAS ANALOGICAS	EW 256		E/S ANAL. 128
	EW 766		
SALIDAS ANALOGICAS	AW 256		
	AW 766		

CPU 315-2 DP

TIPO	AREAS	CON PAE/PAA	CANTIDAD MAXIMA
ENTRADAS DIGITALES	E 0.0	E 0.0	E/S DIG. 1024
	E 255.7	E 127.7	
SALIDAS DIGITALES	A0.0	A 0.0	
	A 255.7	A 127.7	
ENTRADAS ANALOGICAS	EW 256		E/S ANAL. 128
	EW 766		
SALIDAS ANALOGICAS	AW 256		
	AW 766		

CPU 316

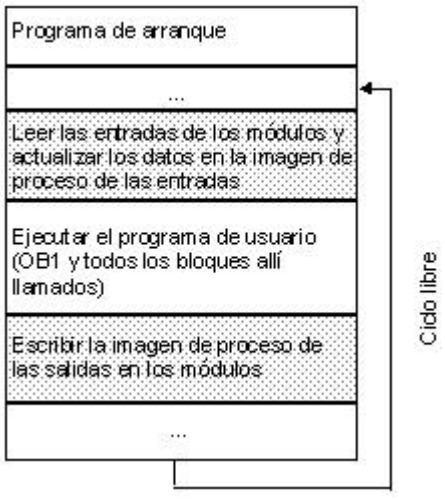
TIPO	AREAS	CON PAE/PAA	CANTIDAD MAXIMA
ENTRADAS DIGITALES	E 0.0	E 0.0	E/S DIG. 1024
	E 255.7	E 127.7	
SALIDAS DIGITALES	A0.0	A 0.0	
	A 255.7	A 127.7	
ENTRADAS ANALOGICAS	EW 256		E/S ANAL.
	EW 766		

SALIDAS ANALOGICAS	AW 256 AW 766		128

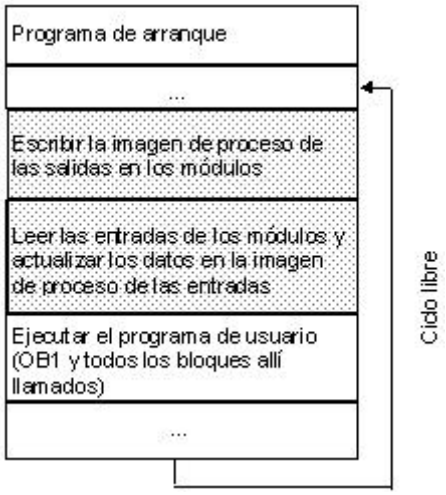
CPU 318-2 DP

TIPO	AREAS	CON PAE/PAA	CANTIDAD MAXIMA
ENTRADAS DIGITALES	E 0.0 E 511.7	E 0.0 E 255.7	E/S DIG. 1024
SALIDAS DIGITALES	A0.0 A 511.7	A 0.0 A 255.7	
ENTRADAS ANALOGICAS	EW 256 EW 766		E/S ANAL. 128
SALIDAS ANALOGICAS	AW 256 AW 766		

Anteriormente, desde la época de S5, la CPU trabajaba de la siguiente manera con la PAE y la PAA:



Sin embargo, con las nuevas CPU´s de S7 (a partir del 10/98), la filosofía es la siguiente:



En las CPU's S7-300, las entradas y salidas de imágenes de proceso no asignadas se pueden utilizar como áreas de marcas adicionales. Los programas que utilizan dicha posibilidad no pueden procesarse en las CPU's S7-400.

3.2 La memoria en S7.

3.2.1 Zonas de memoria

La memoria de las CPU's S7 se subdivide en tres áreas diferenciadas:

- **La memoria de carga**: permite almacenar el programa de usuario sin asignación simbólica de operandos o comentarios (éstos se almacenan en el disco duro del ordenador). La memoria de carga puede ser RAM o Flash-EPROM. En la memoria de carga se almacena no solo el programa sino también los datos del sistema.
- **La memoria de trabajo (RAM integrada)**: contiene la partes del programa S7 relevantes para la ejecución del programa. La ejecución del programa tiene lugar exclusivamente en el área correspondiente a las memorias de trabajo y del sistema.
- **La memoria del sistema (RAM)**: contiene los elementos de memoria que cada CPU pone a disposición del programa de usuario, tales como: la imagen de proceso de las entradas y salidas, marcas, temporizadores y contadores. Contiene además las pilas de bloques y de interrupción. La memoria del sistema de la CPU ofrece además una memoria temporal (pila de datos locales) asignada al programa para los datos locales del bloque llamado. Estos datos sólo tienen vigencia mientras esté activo el bloque correspondiente (la zona de la tabla de declaración de una OB, o una FC).

Por lo tanto, nuestro programa tendrá un consumo de memoria de carga y otro de memoria de trabajo. En ninguno deberemos de superar la memoria de trabajo, ya que no es posible su ampliación, por lo que nos veremos obligados a cambiar de CPU. La memoria de carga sí que puede ser ampliada mediante Flash o RAM externas.

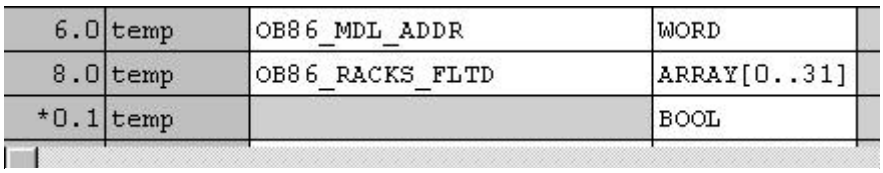
El tamaño de la memoria de los diferentes autómatas que componen la serie S7 es el siguiente:

CPU	MEMORIA CARGA	MEMORIA TRABAJO
312 IFM	20 KB	6 KB
313	20 KB	12 KB
314	40 KB	24 KB

314 IFM	48 KB	32 KB
315	80 KB	48 KB
315-2 DP	96 KB	64 KB
316	192 KB	128 KB
318-2 DP	64 KB	512 KB

3.2.2 Acceso a la pila de datos locales

A veces es necesario acceder a una variable de la zona de declaración, pero se encuentra agrupada en una estructura o un array (ver imagen). Podemos en cualquier momento acceder a la misma a través de el direccionamiento L.



6.0	temp	OB86_MDL_ADDR	WORD
8.0	temp	OB86_RACKS_FLTD	ARRAY[0..31]
*0.1	temp		BOOL

Figura 7. Array de la tabla de declaración de la OB86. Obsérvese como a la izquierda se nos indica que el array comienza en la LB8.

Para acceder a este array, en formato de byte en lugar de bool, que es como está generado, únicamente debemos de realizar la llamada:

L LB8

Con lo que cargaremos los 8 primeros bits del array. Estamos realizando una carga en el acumulador del PLC del byte 8 de la zona de L's.

3.2.3 Mapas de memoria de sistema.

Un mapa de memoria de sistema define el tamaño que asigna la CPU para cada uno de los tipos de variables de que va a disponer el programa, entendiendo por variables desde entradas digitales hasta temporizadores, pasando por marcas o DB´s.

CPU 312 IFM

	AREAS	REMANENTES
MARCAS	M 0.0	M 0.0
	M 127.7	M 71.7
CONTADORES	Z 0	Z 0

	Z 31	Z31
TEMPORIZADORES	T 0 T63	Ninguno

CPU 313

	AREAS	REMANENTES
MARCAS	M 0.0 M 255.7	M 0.0 M 71.7
CONTADORES	Z 0 Z 63	Z 0 Z35
TEMPORIZADORES	T 0 T127	T 0 T35

CPU 314 IFM

	AREAS	REMANENTES
MARCAS	M 0.0 M 255.7	M 0.0 M 255.7
CONTADORES	Z 0 Z 63	Z 0 Z63
TEMPORIZADORES	T 0 T127	T 0 T71

CPU 314

	AREAS	REMANENTES
MARCAS	M 0.0 M 255.7	M 0.0 M 255.7
CONTADORES	Z 0	Z 0

	Z 63	Z63
TEMPORIZADORES	T 0 T127	T 0 T71

CPU 315

	AREAS	REMANENTES
MARCAS	M 0.0 M 255.7	M 0.0 M 255.7
CONTADORES	Z 0 Z 63	Z 0 Z63
TEMPORIZADORES	T 0 T127	T 0 T127

CPU 315-2 DP

	AREAS	REMANENTES
MARCAS	M 0.0 M 255.7	M 0.0 M 255.7
CONTADORES	Z 0 Z 63	Z 0 Z63
TEMPORIZADORES	T 0 T127	T 0 T127

CPU 316

	AREAS	REMANENTES
MARCAS	M 0.0 M 255.7	M 0.0 M 255.7
CONTADORES	Z 0 Z 63	Z 0 Z63
TEMPORIZADORES	T 0 T127	T 0 T127

CPU 318-2 DP

	AREAS	REMANENTES
MARCAS	M 0.0 M 1023.7	M 0.0 M 1023.7
CONTADORES	Z 0 Z 511	Z 0 Z 511
TEMPORIZADORES	T 0 T 511	T 0 T 511

3.2.4 ¿Cómo saber lo qué nos ocupa un programa?

Para conocer la memoria que ocupa un programa, en el administrador de Step 7 seleccionamos un bloque de S7, y presionando el botón derecho accedemos a sus propiedades, que nos indicarán tanto el tamaño de la memoria de carga requerida por el mismo, como el de la memoria de trabajo.

Para conocer cuanto nos ocupa todo el programa, incluyendo los datos de sistema, deberemos de seleccionar el subdirectorio bloques, y visualizar sus propiedades.

Los datos de sistema, la carpeta SDB de nuestro proyecto, contienen la configuración Hardware del equipo, que hemos proyectado en Hardware de S7.

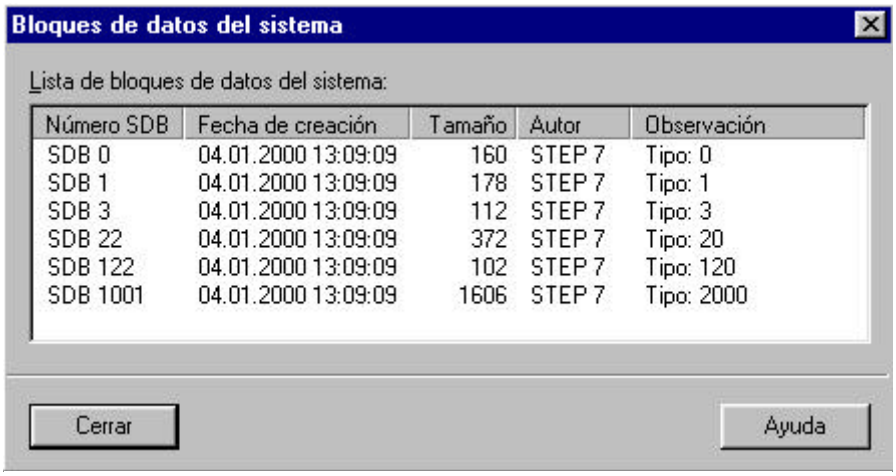


Figura 8. Datos del sistema de una configuración S7.

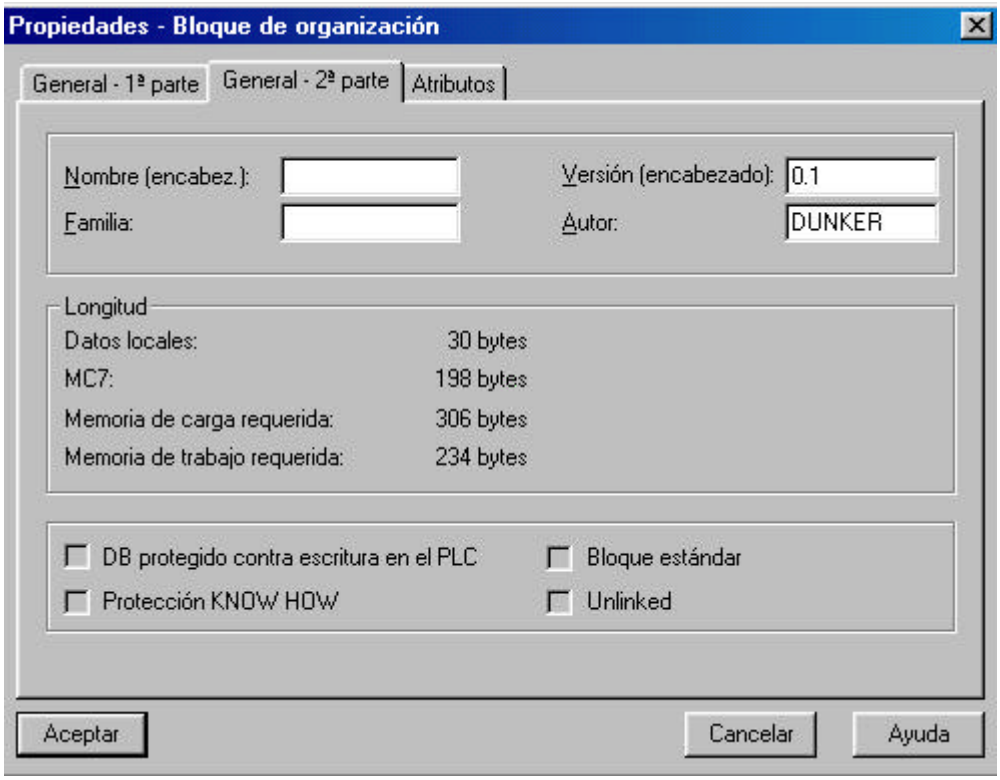


Figura 9. Propiedades de un bloque de S7. En él se nos indica la cantidad de datos locales (variables L) consumidas por la tabla de declaración, así como la memoria de carga y de trabajo requerida por la CPU para ejecutarse.

3.2.5 ¿Que se borra ante un borrado total?.

Para realizar un borrado en un equipo S7 desde hardware es necesario seguir los siguientes pasos:

- Mantener el selector frontal de la CPU en la posición *MRES* durante al menos 3 segundos. Durante este tiempo el led de stop de la CPU luce alternativamente 2 veces.
- Soltar el selector, que vuelve a la posición de STOP, y antes de 3 segundos, volver a mantenerlo en la posición *MRES*, al menos durante 1 segundo. El led de STOP comienza a parpadear, y durante 3 segundos se borra la memoria del PLC.

Pero, ¿qué es lo que hemos borrado?. Durante estos 3 segundos en la CPU ocurren las siguientes cosas:

- Se borra completamente el programa de PLC. Este borrado se realiza en la RAM del equipo, pero no en la Flash-Eprom, si la hubiera en el mismo (los equipos IFM disponen de esta memoria incorporada).
- En caso de que el equipo en cuestión sea IFM, se vuelca el programa almacenado en la Flash-Eprom a la RAM.
- La zona de marcas es inicializada a 0.
- Se mantiene intacto:
    - El buffer de diagnóstico del equipo.
    - La dirección MPI así como su parametrización.
    - El contenido del contador de horas de operación.

Por lo tanto, para borrar una CPU IFM la mejor solución es borrar los módulos desde Step 7, y a continuación realizar una copia de RAM a ROM, en el administrador de Simatic (*Sistema destino->Copiar RAM en ROM*).

3.2.6 Tipos de variables en S7.

Las variables en Simatic S7 vienen determinadas por dos características: el tamaño de las mismas y su ubicación dentro del mapa de memoria del PLC. Veamos los tipos de datos de que podemos disponer, así como su rango de valores:

- **Variables simples:** aquellas que solo están formadas por una única variable.

Tipo	Bits	Formatos	Rango	Ejemplo
BOOL	1	texto	TRUE/FALSE	TRUE
BYTE	8	hexadecimal	B#16#0 a B#16#FF	L B#16#23
WORD	16	binario	2#0 a 2#1111111111111111	L 2#00101
		hexadecimal	W#16#0 a W#16#FFFF	L W#16#234F
		Bcd	C#0 a C#999	L C#997
		Decimal sin signo	B#(0,0) a B#(255,255)	L B#(14,245)
INT	16	Decimal con signo	-32768 hasta 32767	L 345
DWORD	32	binario	2#0 a 2#11111111111111111111111111111111	L 2#11011
		hexadecimal	DW#16#0 a DW#16#FFFF_FFFF	L DW#16#3FT
		Decimal sin signo	B#(0,0,0,0) a B#(255,255,255,255)	L B#(0,1,2,3)
DINT	32	Decimal con signo	L# -2147483648 hasta L# 2147483647	L L#400000
REAL	32	Coma flotante	1.175 495e-38 a 3.402823e+38	L 23.5678
S5TIME	16	Tiempo S7	S5T#0H_0M_0S_10MS hasta S5T#2H_46M_30S_0MS	L s5t#2s
TIME	32	Tiempo IEC	-T#24D_20H_31M_23S_648MS hasta T#24D_20H_31M_23S_647MS	L T#2H
DATE	16	Fecha IEC	D#1990-1-1 hasta D#2168-12-31	L D#1994-3-15
TIME_OF_DAY	32	Hora del día	TOD#0:0:0.0 hasta TOD#23:59:59.999	L TOD#1:10:3.3
CHAR	8	carácter	'A','B' etc.	L 'E'

- **Variables compuestas:** aquellas que se basan en la agrupación de varias variables simples.

Tipo	Bits	Significado
DATE_AND_TIME	64	Unión de una variable DATE y otra TIME_OF_DAY. Rango de valores: DT#1990-1-1-0:0:0.0 a DT#2089-12-31-23:59:59.999
STRING	-	Cadena de caracteres. Rango de valores: STRING[1] a STRING[254]

ARRAY	-	Array de un determinado tipo de variable simple. Ejemplo: ARRAY [1..20,1..10] of INT El número máximo de dimensiones de un array es de 6. E número máximo de índice por dimensión es de 32767.
STRUCT	-	Define un agrupamiento de tipos de datos compuestos o simples.

3.3 El hardware en S7.

3.3.1 Los leds de la CPU.

Veamos cual es el significado de los leds de las CPU´s S7.

LED	INDICA	SIGNIFICADO
SF (ROJO)	Fallo o error de sistema.	Luce en caso de: Fallo hardware. Fallo del firmware del equipo. Instrucción de programa incorrecta. Asignación de parámetros de sistema erróneos. Errores aritméticos internos. Errores de tiempo. Flash-Eprom externa errónea. Fallo de la batería. Fallo de acceso a la periferia (no para la periferia integrada en la CPU).
BATF (ROJO)	Fallo de batería.	Luce si la batería no es capaz de salvaguardar el programa en caso de desconexión de la alimentación del equipo. También puede lucir si en lugar de una pila hemos conectado un acumulador de reloj hardware, ya que el programa no se encuentra respaldado por la misma.
5 VDC (green)	Alimentación BUS	La CPU está alimentando al bus trasero a 5 V correctamente.
FRCE (AMARILLO)	Forzado activado.	Luce si se está forzando una señal.
RUN (VERDE)	Estado RUN.	El programa se está ejecutando.
STOP (AMARILLO)	Estado Stop.	El programa no se ejecuta. Parpadea si la CPU solicita un borrado total de memoria, debido generalmente a memoria corrompida.

En el caso de que la CPU disponga de tarjeta maestra de Profibús DP, el significado de los leds de la tarjeta son:

SF	DP	Significado	Solución
LED off	**LED off**	Configuración O.K.	
LED on	**LED on**	Fallo de bus de comunicaciones a nivel de hardware. Diferente configuración de velocidades de bus en modo multimaestro.	Chequee el cable y las resistencias terminadoras.

LED on	**Parpadea**	Fallo de estación. Cada vez que parpadea es que intenta acceder a un esclavo y este no responde.	Acudir a información del módulo->buffer de diagnóstico para saber que esclavo no responde.
LED on	**LED off**	Incorrecta configuración de los datos DP	Evaluar el buffer de diagnóstico.

4

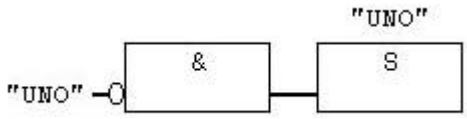
Ladrillos de un programa

Existen acciones o rutinas de programa que a menudo son necesarias para realizar determinadas acciones en programación. Inexplicablemente, continúan sin haber sido incluidas en las bibliotecas de funciones del software, pese a que son deficiencias heredadas de la época del Step 5.

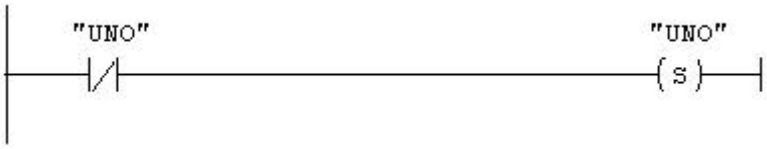
4.1 Rutinas frecuentes

4.1.1 Marca uno

Marca utilizada para habilitar acciones que siempre se deben de cumplir, y en las cuales no se puede colocar TRUE.

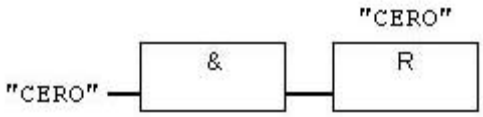



```
UN    "UNO"
S     "UNO"
```

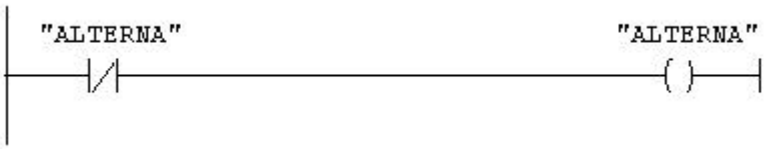


4.1.2 Marca cero

Marca que anula condiciones e impide saltos.



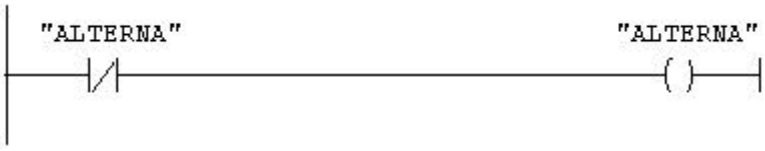

```
U     "CERO"
R     "CERO"
```



4.1.3 Marca alterna

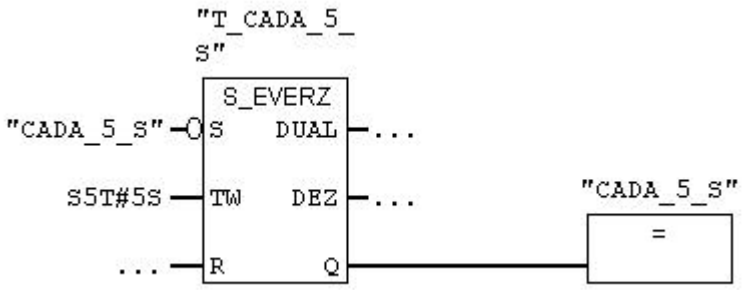
La marca alterna se activa cada 2 ciclos de programa de PLC. Es ideal para activaciones de funciones del tipo “cuando esté una rutina no estará la otra”, siendo estas rutinas incompatibles entre sí (p. Ej. funciones que utilicen llamadas a instrucciones del sistema que no deben de ser ejecutadas más de una vez por ciclo de PLC).


```
UN    "ALTERNA"
=     "ALTERNA"
```

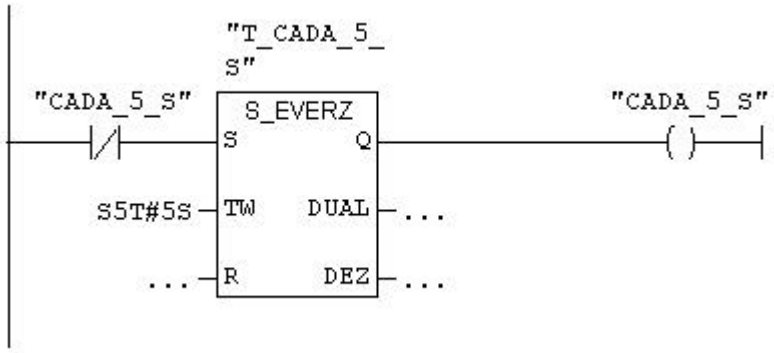


4.1.4 Activar tareas cada cierto tiempo.

A veces puede ser necesario realizar tareas cada cierto tiempo. Esto se puede realizar desde el propio autómata programando una OB de tiempo. Sin embargo, si no se desea tanta sofisticación desde programa se puede realizar con el siguiente código, que además posibilita la parametrización del tiempo de muestreo si dejamos el TW del temporizador en una variable.




```
UN    "CADA_5_S"
L     S5T#5S
SE    "T_CADA_5_S"
U     "T_CADA_5_S"
=     "CADA_5_S"
```

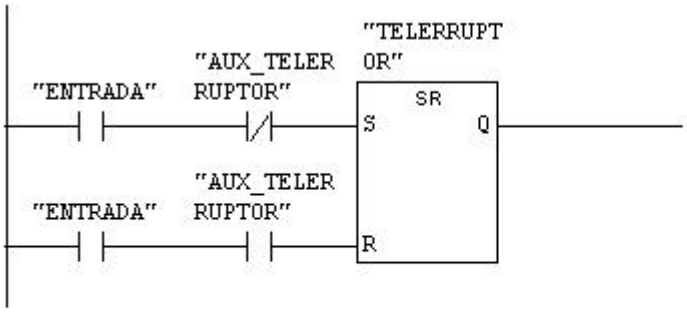


Es necesario que la condición para saltar a la FC en cuestión tenga como condición que esté la variable “CADA_5S”.

4.1.5 Telerruptor.

Un telerruptor es una señal del PLC que debe de activarse con una entrada, y con esa misma entrada debe de desactivarse la siguiente vez. Es el típico arranque con un botón que ejerce él mismo de paro cuando vuelve a ser presionado.

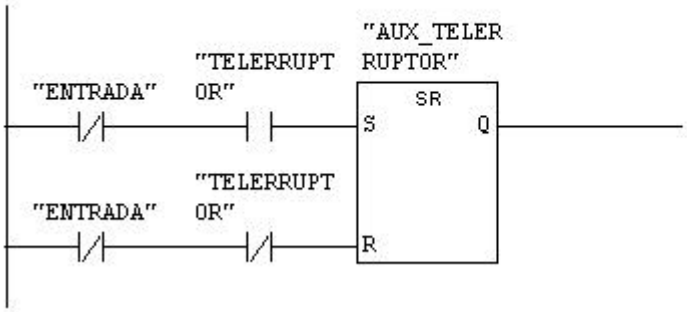
Para realizar esta función necesitaremos un auxiliar que memoriza el estado actual de nuestra salida para saber si debemos en la siguiente activación parar o arrancar la salida.



Información del símbolo:

E0.0	ENTRADA	ENTRADA DEL TELERRUPTOR
M0.4	AUX_TELERRUPTOR	MARCA AUXILIAR TELERRUPTOR
M0.5	TELERRUPTOR	MARCA TELERRUPTOR

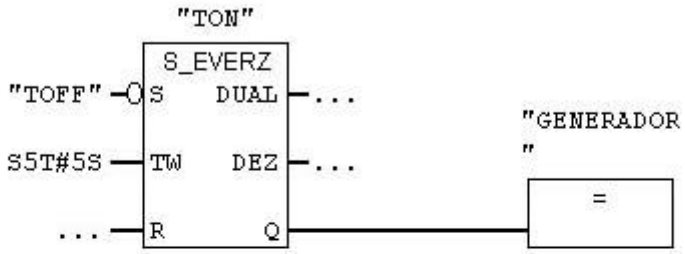
Segm. 2: Título:



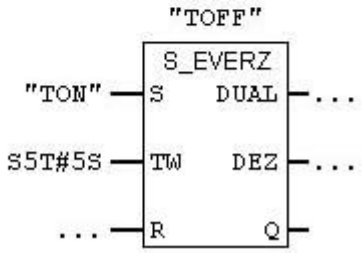
4.1.6 Generador de pulsos asíncronos.

El generador asíncrono se suele gastar en tareas de bombeo, señales de intermitencia no regulares o engrase de motores (engrasar *x* segundos y esperar *y segundos).* El código solo requiere de dos temporizadores y la salida a activar. La base de los temporizadores es la que nos va a marcar el Ton y Toff de nuestra señal asíncrona (en el ejemplo es 5 segundos inactiva y 5 segundos activa). Estos valores podrían ser dos parámetros que se podrían ajustar desde una OP.

Segm. 7 : GENERADOR DE PULSOS ASINCRONOS TIEMPO INACTIVO



Segm. 8 : GENERADOR DE PULSOS ASINCRONOS TIEMPO ACTIVO



5

Instrucciones básicas

En el presenta capítulo vamos a repasar las instrucciones básicas de que disponemos en un equipo S7, buscando una aplicación práctica a las mismas para una mejor comprensión de su utilidad.

5.1 Flancos

En ocasiones necesitamos que una determinada acción sólo se realice una vez mientras se cumplan las condiciones para la activación de la misma. Una gran cantidad de sets de variables mejorarían si se les aplicase una señal de flanco positivo a sus condiciones de activación. La señal de flanco, tanto positivo como negativo en el Step 7 requiere de una marcha que no puede ser utilizada en otra parte del programa, por lo que es importante simbolizarla como exclusiva de ese flanco en cuestión

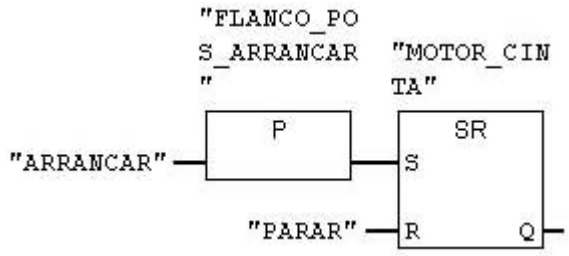


Figura 10 . Utilización de flancos en activación de sets.

¿Qué ventaja nos aporta el flanco positivo en esta acción?. Hay que reconocer que no son evidentes, pero pertenecen a esos pequeños detalles que convierten a un programa en robusto y fiable (a prueba de bombas).

Si el usuario presiona el pulsador de arranque y de paro a la vez, el motor no arranca, ya que el reset es posterior. Pero si deja de presionar el paro, y no colocamos una señal de flaco positivo, el motor arranca. Algunos pueden pensar que esto es un fallo del operario, y que por lo tanto la responsabilidad recae sobre él. Esta excusa carece de fundamento si pensamos que puede quedarse enclavado el botón de arranque inadvertidamente para el operario, y que no va a poder parar el sistema si no es

manteniendo presionado el botón de paro, con lo que difícilmente puede encontrar la avería del botón por sí solo. Es responsabilidad del programador prever en la medida de lo posible las acciones que pueden producirse en la instalación, y ante la duda buscar aquellas situaciones más seguras (en este caso que no arranque el motor).

5.2 Temporizadores en S7

En S7 los temporizadores continúan manteniendo el mismo funcionamiento heredado de S5, pese a disponer de un nuevo formato para contaje de tiempos (TIME). Por lo tanto, se mantiene el problema de contaje de tiempo limitado a 9990 segundos como periodo máximo de contaje.

Para periodos de tiempo mayores, es obligatorio recurrir a las OB´s de alarma horaria, o realizar un concatenamiento entre un generador de pulsos y un contador que vaya incrementando su valor.

Los tipos de temporizadores de que se dispone en S7 son:

– SI: Impulso
– SV: Impulso prolongado
– SE: Retardo a la conexión
– SS: Retardo a la conexión con memoria
– SA: Retardo a la desconexión

Un temporizador en S7 se compone de:

- Una palabra de 16 bits que identifica su valor actual de contaje.
- Un bit, que identifica su estado (activado o desactivado).

En la palabra del temporizador es donde cargaremos el valor de contaje, junto con su base de tiempos, y podremos consultarla para conocer su estado durante el descontaje, mientras que el bit nos activará acciones cuando finalice o mientras se desarrolle el proceso de contaje.

5.2.1 Formato de tiempo en los temporizadores.

La introducción de la base de tiempo en los temporizadores ha pasado del formato KT x.y que se utilizaba en los S5 a uno más intuitivo, S5T#x, ya que el valor de la base de tiempo lo calcula el sistema automáticamente. Únicamente es necesario especificar un periodo de tiempo como valor x.

Existen dos formas de cargar el valor de temporización en un temporizador S7:

- Utilizar la filosofía de S5, calculando nosotros la base de tiempo, e introduciendo el valor según el formato:

L W#16#wxyz

siendo:

w = la base de tiempo (intervalo o resolución)

xyz = valor de temporización en formato BCD

- Utilizar la filosofía de S7, en la que la base de tiempos es calculada por el software Step 7. El formato de tiempos es:

  L S5T#aH_bbM_ccS_dddMS

  siendo:

  a = horas, bb = minutos, cc = segundos y dd = milisegundos.

  La base de tiempo se selecciona automáticamente, redondeándose el valor al próximo valor inferior con la base de tiempo correspondiente.

En cualquier caso, el valor de temporización máximo que puede introducirse es de 9990 segundos o de 2H_46M_30S.

El formato de la base de tiempo de la palabra del temporizador va a determinar tanto la resolución del mismo, como el valor máximo de contaje que podemos alcanzar. Las diferentes posibilidades de que disponemos para los bits 12, 13, 14, y 15 de la palabra son:

FORMATO DE BITS				BASE TIEMPO	RANGO
15	14	13	12		
X	X	0	0	0,01 S	10MS a 9S_990MS
X	X	0	1	0,1 S	100MS a 1M_39S_900MS
X	X	1	0	1 S	1S a 16M_39S
X	X	1	1	10 S	10S a 2HR_46M_30S

Nota: los bits marcados como X son irrelevantes, pudiendo adoptar 0 o 1 indistintamente.

5.2.2 Los temporizadores en AWL.

Veamos cual es la estructura de un temporizador en AWL mediante un ejemplo:

```
U    E     0.0          // SI SE ACTIVA LA ENTRADA
L    S5T#5S             // CARGA EN EL ACUMULADOR 5 SEGUNDOS
SI   T     0            // ACTIVA EL TEMPORIZADOR 0 EN FORMATO SI CON 5 SEGUNDOS
```

```
U    T     0              // MIENTRAS ESTÉ ACTIVO EL TEMPORIZADOR
=    A     4.0            // ACTIVA LA SALIDA
```

Las tres primeras líneas realizan la carga del valor de tiempos en el temporizador, y además activan su arranque. A partir de ese instante comienza a descontar el valor actual del temporizador cada x tiempo especificado en la base de tiempos del temporizador, hasta llegar a 0, donde finaliza su contaje.

Dependiendo del tipo de temporizador que hayamos seleccionado en la instrucción Sx T0 (siendo x el tipo de temporizador) se comportará su bit de estado de una manera u otra.

```
U    E     0.1            // SI ESTA LA ENTRADA
R    T     0              // EL TEMPORIZADOR SE RESETEA
```

También es posible resetear el temporizador mediante una entrada, con lo cual el valor del temporizador pasa a 0 y el bit del mismo se deshabilita automáticamente.

```
U    E     0.3            // SI ESTA LA ENTRADA
FR   T     0              // COMIENZA DE NUEVO EL CONTAJE
```

Otra posibilidad es relanzar el contaje del temporizador, mediante la función FR de liberación de temporización. Cuando se active la entrada, el contador comienza de nuevo su proceso de contaje desde el último valor que se le había asignado como valor preseleccionado.

```
L    T     0              // CARGA EL VALOR ACTUAL DEL TEMPORIZADOR
T    MW    0              // TRANSFIERELO EN DECIMAL

LC   T     0              // CARGA CODIFICADO EL VALOR EL TEMPORIZADOR
T    MW    2              // TRANSFIERELO EN FORMATO BCD
```

Por último nos puede ser interesante conocer el estado actual del temporizador (cuanto tiempo le resta por contar). Para ello, únicamente debemos de cargar el valor de la palabra del temporizador. Esta carga se puede realizar de dos modos: normal en formato decimal (para comparaciones), o codificada en formato BCD (utilizada en displays).

5.2.3 Tipos de temporizadores

Existen 5 tipos diferentes de temporizadores. La siguiente imagen explica claramente el funcionamiento de cada uno de ellos y el estado del bit asociado al temporizador en función de la entrada de activación.

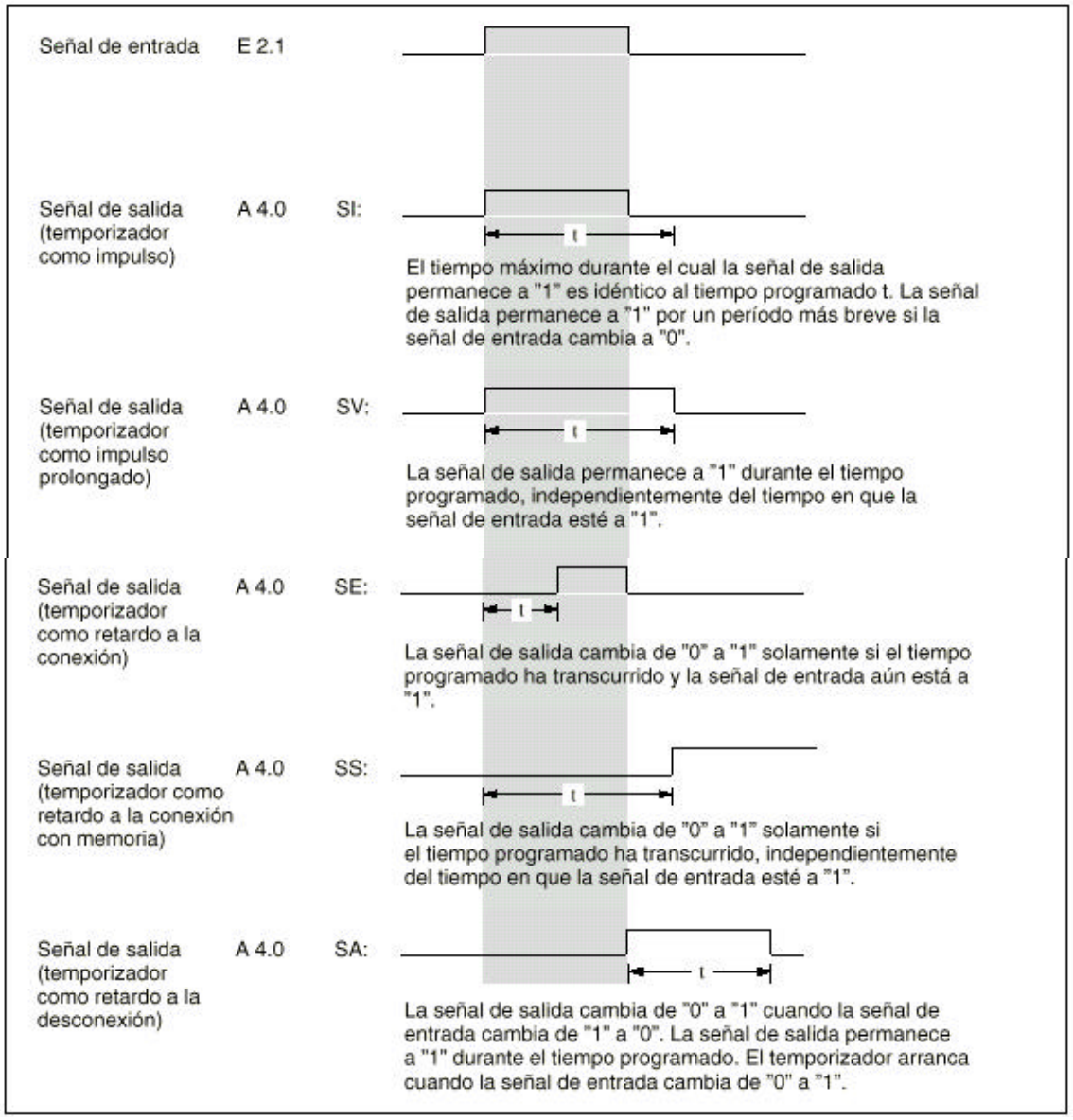


Figura 11. Tipos de temporizadores en S7.

5.2.4 Temporizadores desde equipos de visualización.

En algunas aplicaciones de visualización (OP´s o scadas), no nos es posible disponer del formato de datos S5T#, por lo que la introducción de consignas de temporizadores es bastante complicada.

La única manera de resolver este problema, es crear una función dentro del PLC que construya el formato de tiempo de S7. En el siguiente apartado se describe una subrutina que ya realiza estos menesteres.

0.0	in	TIEMPO	REAL
4.0	out	SALIDA_T	S5TIME
	in_out		
0.0	temp	TEMPORAL_W	WORD

La FC posee un parámetro de entrada real, nuestra consigna de tiempo, y devuelve la misma consigna pero ya en formato de tiempos.

```
L    #TIEMPO                 // CARGA CONSIGNA EN REAL
L    1.000000e+001           // CARGA 10 PORQUE TENEMOS 1 DECIMAL
*R
RND                          // CONVIERTE A DOBLE ENTERO
DTB                          // CONVIERTE A BCD
T    #TEMPORAL_W             // GUARDA EN UN TEMPORAL DE TIPO WORD

SET
S    L     0.4               // ACTIVA LA BASE DE TIEMPOS DE DECIMAS DE SEGUNDO

L    #TEMPORAL_W
T    #SALIDA_T               // TRANSFIERE LA SALIDA YA EN FORMATO S5T
```

Veamos qué es lo que realiza la función y sus posibilidades. En primer lugar, es necesario conocer de antemano cual es el valor máximo posible de consigna de tiempos. Supongamos que se encuentra siempre por debajo de 99,9 segundos. El siguiente paso es conocer cual es el incremento mínimo que se debe de poder introducir como consigna de tiempos. Supongamos que con 0,1 segundos es suficiente. En este caso, podremos tomar un temporizador de base 0,1 segundos.

En el equipo de visualización seleccionaremos una variable real, que puede tener los decimales que queramos, pero como veremos posteriormente, solo será elaborado el primer decimal del valor. Debemos de asegurarnos que el valor no debe ser superior a 99,9, impidiéndolo desde el scada u OP.

La función anterior, tomará el valor real, lo multiplica por 10 y lo convierte a BCD. A continuación activa la retícula de base de tiempos de décimas de segundos, con lo que ya tenemos nuestra consigna de temporización.

Para otros rangos de tiempos, será necesario modificar ligeramente la función en el número a multiplicar y en el bit a setear como base de tiempos.

5.2.5 Temporizadores de más de 2 horas 46 minutos.

Desgraciadamente utilizando un temporizador únicamente podemos contar hasta 2 horas y 46 minutos. El formato T#, que permite valores superiores de tiempo, no se puede gastar como base de tiempos en un temporizador (debido a compatibilidad con S5).

Existen varias maneras de solucionar este problema, pero una de ellas podría ser la expuesta a continuación. Su pongamos que necesitamos un temporizador que cuente 80000 horas de funcionamiento de un motor. Primeramente deberemos de programas la rutina vista en el apartado *"Ladrillos de un programa",* denominada "*activar tareas cada cierto tiempo".*

Gracias a ella entraremos a una FC que vamos a crear cada hora. La FC (que puede estar parametrizada para su reutilización tiene el siguiente aspecto.

```
      U    E   124.0          // si pulsamos reset (puede ser un parámetro de la FC)
      SPB  m002               // borraremos el contador

      U    E   124.1          // si pulsamos inicializar (puede ser un parámetro de la FC)
      SPB  m003               // inicializaremos el tiempo

      L    MD    0            // carga el contador
      L    L#80000            // carga valor preselección de contaje (puede ser un parámetro de la FC)
      ==D                     // si ha llegado a preselección
      SPB  m001               // salta para activar salida

      L    1
      L    MD    0
      +D
      T    MD    0            // incrementa valor contaje en 1.
      BEA

m001: NOP  0
      S    A   124.0          // activa salida por valor preselección alcanzado
      BEA

m002: NOP  0
      L    0                  // resetea contador (puede ser un parámetro de la FC)
```

```
    T    MD    0
    BEA

m003: NOP  0
    L    2000                  // inicializar contador (puede ser un parámetro de la FC)
    T    MD    0
```

5.3 Contadores en S7

Cuando el intervalo de contaje no depende del tiempo, sino de una señal asíncrona deberemos de recurrir a los contadores.

5.3.1 Los contadores en FUP o KOP.

Es conveniente realizar el segmento del contador en FUP o KOP, ya que se sigue con más claridad el contaje o descontaje del contador.

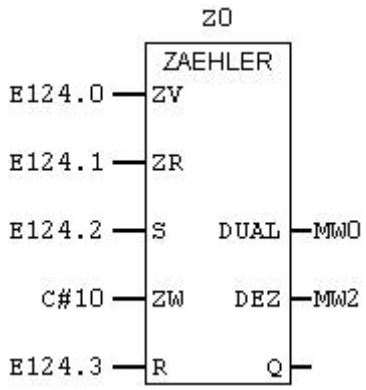


Figura 12 . Ejemplo contador en FUP.

Veamos cuales son los parámetros del contador:

PARAMETRO	TIPO	FORMATO	SIGNIFICADO
ZV	ENTRADA	BOOL	Incrementar el contador en una unidad.
ZR	ENTRADA	BOOL	Decrementar el contador en una unidad.
S	ENTRADA	BOOL	Setear el contador al valor ZW.
ZW	ENTRADA	COUNTER	Valor de seteo del contador.
R	ENTRADA	BOOL	Resetear el contador a cero.
DUAL	SALIDA	WORD	Valor actual del contador en entero.
DEZ	SALIDA	WORD	Valor actual del contador en BCD.
Q	SALIDA	BOOL	Contador activado.

5.3.2 ¿Cómo gastar un contador?

Existe una particularidad de los contadores que produce bastante perplejidad la primera vez que un programador utiliza uno en S7. Los contadores se activan siempre que su valor sea distinto de 0, y su salida no presenta ninguna variación por el hecho de alcanzar el valor de preselección. Esto obliga a :

- realizar comparaciones con la palabra de salida DEZ del contador, lo cual, aunque no es un defecto, no debe ser nuestra única alternativa, o
- utilizar el contador siempre como decrementador.

Para utilizarlo como decrementador, primeramente seteamos el valor a ZW con la entrada S, y posteriormente comenzamos a descontar. En el segmento siguiente al contador realizamos la consulta: si está el bit de S, y no está el bit del contador Zx, activamos la salida correspondiente. Con esto obtendremos una salida cuando termine el contaje, sin necesidad de utilizar comparaciones del contador.

5.3.3 Contadores en AWL.

Aunque no es recomendable, existen casos en los que se debe de programar en AWL el contador. Las diferentes posibilidades de programación son:

```
U    E   124.0
ZV   Z    0            // CONTAR HACIA ADELANTE

U    E   124.1
ZR   Z    0            // CONTAR HACIA ATRAS

U    E   124.2
L    C#10
S    Z    0           // SETEAR EL CONTADOR A UN VALOR PRESELECCIONADO

U    E   124.3
R    Z    0           // RESETAR EL CONTADOR

L    Z    0
T    MW   0            // CARGAR EL VALOR ACTUAL EN DECIMAL

LC   Z    0
T    MW   2            // CARGAR EL VALOR ACTUAL EN BCD
```

5.3.4 Contadores de valor superior a 999.

Si necesitamos contar más de 999, podemos utilizar la rutina vista anteriormente en los temporizadores. La única modificación necesaria es que la entrada a la FC del ejemplo no debe de hacerse a través de una función "*activar tareas cada cierto tiempo",* sino cada vez que se active la condición de contaje.

5.4 Direccionamiento indirecto y punteros.

Existen dos tipos de direccionamiento en la programación de S7:

- **Directo.** Es aquel en el que la instrucción va precedida del operando con el que se desea operar. Ejemplos: S M4.0, L DB10.DBW2.
- **Indirecto.** El operando indica la dirección del valor que va a procesar la operación.

5.4.1 ¿Cómo trabajar con direccionamiento indirecto?

Existen dos tipos de punteros para el direccionamiento indirecto:

- **puntero formato de palabra:** utilizado para las declaraciones de DB´s, temporizadores y contadores.
- **Puntero formato doble palabra:** utilizado para el resto de zonas de memoria (marcas, variables de DB´s, entradas, salidas).

En cualquier caso, las variables que pueden contener el formato de puntero para el direccionamiento indirecto únicamente pueden ser de la zona de marcas o de DB´s. Veamos como funciona el direccionamiento indirecto mediante ejemplos simples. Comenzaremos con ejemplos de punteros de palabra:

- **U T[MW0]** -> El número de temporizador es el valor que se encuentre dentro de la MW0.
- **AUF DB[MW0]** -> La DB a abrir es el valor que se encuentre dentro de la MW0.
- **L Z[mw0]** -> El número de temporizador del que se desea cargar el valor actual se encuentra en el valor de la MW0.

Y a continuación veremos el caso de punteros de doble palabra:

- **U E[MD0]** -> Consulta el estado de la entrada de número almacenado en la MD0. Si en la MD0 tuviésemos un 2.0, el resultado transferido al procesador del PLC sería U E 2.0.
- **= A[MD4]** -> Activa la salida indicada en la MD4.
- **L DB1.DBW[MD8]** -> Carga en la DB1, en la palabra cuya dirección se encuentra dentro del valor de la MD8. Si la MD8 vale 10, la palabra a cargar será la DBW10.

El único inconveniente con el que se encuentra el programador a la hora de trabajar con punteros es desconocer como funciona el formato de puntero de doble palabra.

Para poder trabajar con direcciones del tipo 3.4, o 7.0, en Step 7 se ha creado el formato denominado de puntero, que se basa en la siguiente estructura:

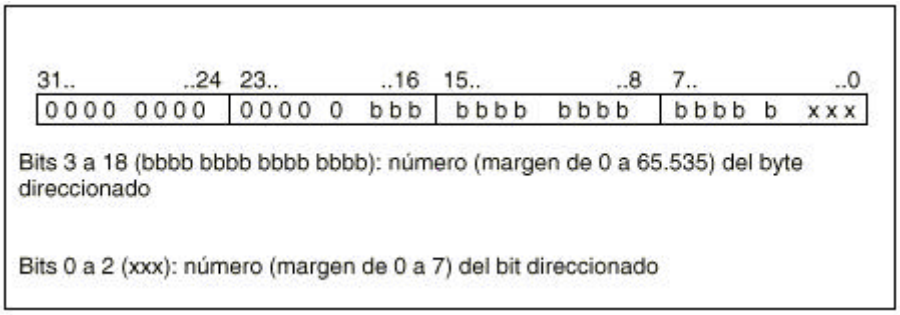


Figura 13. Formato de puntero de doble palabra.

Esto nos obliga a desplazar tres bits hacia la izquierda un valor que deseemos cargar cuando la dirección a la que nos queremos dirigir es distinta de un bit. Por ejemplo, supongamos que queremos acceder a la dirección DB1.DBD, siendo la dirección del valor real un valor que introducimos desde una OP. Supongamos también que el valor de la OP se encuentra en la MD0, y actualmente vale 6. Necesitaremos desplazar dicho valor 3 bits hacia la izquierda para que adopte el formato 6.0, requerido para poder acceder a la dirección indicada. Posteriormente el valor cargado lo almacenamos siempre en una palabra que está asociada al OP.

```
L    MD    0            // CARGA EL VALOR DE OP
SLD  3                  // DESPLAZA 3 A LA IZQUIERDA
T    MD    4            // GUARDAR YA CON FORMATO PUNTERO

AUF  DB    1            // ABRE DB1
L    DBD [MD 4]         // CARGA UN VALOR REAL CON FORMATO INDIRECTO
T    MD    20           // TRANSFERIR A LA PALABRA DE DISPLAY DE OP
```

5.4.2 Ejemplo de direccionamiento indirecto.

El direccionamiento indirecto generalmente se usa para introducir los datos de un trozo de programa parametrizado desde la tabla de declaración de una función. Supongamos que deseamos crear una función que nos permita copiar una zona determinada de una DB en una zona de marcas.

Creamos una función con una cabecera como la siguiente:

Dirección	Declaración	Nombre	Tipo
0.0	in	DATO_DB	INT
2.0	in	DATO_DBB_INI	INT
4.0	in	DATO_DBB_FIN	INT
6.0	in	DATO_MB_INI	INT
	out		
	in_out		
0.0	temp	TEMP_DB	WORD
2.0	temp	TEMP_DBB_INI	DWORD
6.0	temp	TEMP_DBB_FIN	DWORD
10.0	temp	TEMP_MB	DWORD

La función presenta el siguiente código:

```
L    #DATO_DB                  // CARGA DB A ABRIR
T    #TEMP_DB                  // TRANSIERE A TEMPORAL

AUF  DB [#TEMP_DB]             // ABRE DB PARAMETRIZADA

L    #DATO_DBB_INI             // CARGA DBB INICIAL
SLD  3                         // DESPLAZA
T    #TEMP_DBB_INI             // GUARDA EN FORMATO DE PUNTERO

L    #DATO_DBB_FIN             // CARGA DBB FINAL
SLD  3                         // DESPLAZA
T    #TEMP_DBB_FIN             // GUARDA EN FORMATO DE PUNTERO


L    #DATO_MB_INI              // CARGA MB INICIAL
SLD  3                         // DESPLAZA
T    #TEMP_MB                  // GUARDA EN FORMATO DE PUNTERO


M002: L    DBB [#TEMP_DBB_INI] // CARGA VALOR DBB
T    MB [#TEMP_MB]             // TRANSFIERE A MB

L    #TEMP_DBB_INI             // CARGA DBB ACTUAL
L    #TEMP_DBB_FIN             // CARGA DBB FINAL
==D                            // SI SE HA LLEGADO AL FINAL
SPB  M001                      // TERMINA

L    #TEMP_DBB_INI             // CARGA DBB ACTUAL
L    8                         // DESPLAZA 8 DEBIDO AL FORMATO DE UN PUNTERO
+D
T    #TEMP_DBB_INI             // GUARDA DBB ACTUAL
```

```
    L    #TEMP_MB                  // CARGA MB ACTUAL
    L    8                         // DESPLAZA 8 DEBIDO AL FORMATO DE UN PUNTERO
    +D
    T    #TEMP_MB                  // GUARDA MB ACTUAL
    SPA  M002                      // VUELVE A REALIZAR EL BUCLE

M001: NOP  0                       // FINALIZA
```

La llamada a la función puede tener el siguiente aspecto:

```
CALL  FC    3              // FUNCION DE COPIA DE DBS A MBS
 DATO_DB    :=10           // DB FUENTE
 DATO_DBB_INI:=0           // DBB INICIAL FUENTE
 DATO_DBB_FIN:=10          // DBB FINAL FUENTE
 DATO_MB_INI :=18          // MB INCIAL DESTINO
```

5.5 Operaciones matemáticas.

Las CPU´s del S7 asisten una gran variedad de operaciones matemáticas tanto en formato de coma fija, como en coma flotante. En el caso de las operaciones matemáticas, nos encontramos en el campo preferido por el AWL, por lo que es aconsejable realizar estas líneas de código en dicho lenguaje, evitando el KOP o FUP.

Vamos a realizar una serie de ejemplos con cada una de las posibilidades que nos ofrece el equipo S7.

5.5.1 Operaciones con enteros.

Veamos una suma de enteros:

```
L    MW    0          // carga un entero
L    MW    2          // carga otro
+I                    // sumar
T    MW    4          // resultado
```

Debemos de tener cuidado de que la suma no sea superior a 32767. En caso contrario, el resultado de la operación lo almacenaremos en una MD, en lugar de una MW. Lo importante es que el acumulador donde se realiza la operación +I posee un cálculo de 32 bits, por lo que podemos transferir este resultado a una doble palabra.

Resta de enteros:

```
L    MW    0          // carga un entero
L    MW    2          // carga otro
-I                    // resta
T    MW    4          // resultado
```

Multiplicación de enteros:

```
L    MW    0          // carga un entero
L    MW    2          // carga otro
*I                    // multiplicación
T    MD    4          // resultado
```

Y por último la división de enteros:

```
L    MW    0          // carga un entero
```

```
L    MW    2          // carga otro
/I                    // división
T    MD    4          // cociente y resto
```

El resultado (MD4) se compone de la MW4 en la que se almacena el resto, y la MW6, en la que se almacena el cociente.

5.5.2 Operaciones con enteros dobles.

Veamos una suma de enteros dobles:

```
L    MD    0          // carga un entero doble
L    MD    4          // carga otro
+D                    // sumar
T    MD    8          // resultado
```

De igual manera disponemos de la resta (-D), multiplicación (*D), y división (/D). Como la división de enteros dobles únicamente devuelve el cociente, existe una nueva instrucción en operaciones con enteros dobles, para el calculo del resto:

```
L    MD    0          // carga un entero doble
L    MD    4          // carga otro
MOD                   // devuelve el resto
T    MD    8          // el valor del resto de la división de MD0 entre MD4
```

Esta instrucción únicamente es válida en enteros dobles, no en divisiones de enteros.

5.5.3 Operaciones con reales.

Veamos la suma de reales:

```
L    MD    0          // carga un entero doble
L    MD    4          // carga otro
+R                    // sumar
T    MD    8          // resultado
```

De igual manera disponemos de la resta (-R), multiplicación (*R), y división (/R). Como la división de enteros dobles únicamente devuelve el cociente, existe una nueva instrucción en operaciones con enteros dobles, para el calculo del resto:

En números reales puede ser interesante devolver el valor absoluto de un número calculado previamente. Esta operación se realiza mediante la instrucción ABS, como muestra el siguiente código:

```
L    MD   0          // carga un entero doble
L    MD   4          // carga otro
*R                   // sumar
ABS                  // calcula el valor absoluto
T    MD   8          // resultado
```

5.6 Operaciones de comparación.

Las operaciones de comparación se dividen en tres grupos, según sea el tamaño de las variables a comparar:

- Comparación de números enteros.
- Comparación de dobles enteros.
- Comparación de reales.

Vamos a estudiar la comparación en todos los lenguajes, pese a que recomendamos realizar comparaciones en AWL.

5.6.1 Comparación de números enteros.

OPERACIÓN	Descripción
==I	El entero del ACCU2 es igual al entero del ACCU1.
<>I	El entero del ACCU2 es distinto al entero del ACCU1.
>I	El entero del ACCU2 es mayor que el entero del ACCU1.
<I	El entero del ACCU2 es menor que el entero del ACCU1.
>=I	El entero del ACCU2 es mayor o igual que el entero del ACCU1.
<=I	El entero del ACCU2 es menor o igual que el entero del ACCU1.

Ejemplo:

```
L MW0       // carga la MW0 en el ACCU1
L MW2       // carga la MW2 en el ACCU1 y desplaza la MW0 al ACCU2
>I          // si la MW0 es mayor que la MW2
SPB M001    // salta
```

5.6.2 Comparación de números dobles enteros.

OPERACIÓN	Descripción
==D	El doble entero del ACCU2 es igual al doble entero del ACCU1.
<>D	El doble entero del ACCU2 es distinto al doble entero del ACCU1.
>D	El doble entero del ACCU2 es mayor que el doble entero del ACCU1.
<D	El doble entero del ACCU2 es menor que el doble entero del ACCU1.
>=D	El doble entero del ACCU2 es mayor o igual que el doble entero del ACCU1.
<=D	El doble entero del ACCU2 es menor o igual que el doble entero del ACCU1.

Ejemplo:

```
L MD0        // carga la MD0 en el ACCU1
L MD4        // carga la MD4 en el ACCU1 y desplaza la MD0 al ACCU2
>D           // si la MD0 es mayor que la MD4
SPB M001     // salta
```

5.6.3 Comparación de números reales.

OPERACIÓN	Descripción
==R	El real del ACCU2 es igual al real del ACCU1.
<>R	El real del ACCU2 es distinto al real del ACCU1.
>R	El real del ACCU2 es mayor que el real del ACCU1.
<R	El real del ACCU2 es menor que el real del ACCU1.
>=R	El real del ACCU2 es mayor o igual que el real del ACCU1.
<=R	El real del ACCU2 es menor o igual que el real del ACCU1.

Ejemplo:

```
L MD0        // carga la MD0 en el ACCU1
L MD4        // carga la MD4 en el ACCU1 y desplaza la MD0 al ACCU2
>R           // si la MD0 es mayor que la MD4
SPB M001     // salta
```

5.7 Operaciones de conversión.

5.7.1 Formatos de valores.

Como sabemos, un entero es un número de 16 bits, mientras que un entero doble posee 32 bits. Pero, ¿cuál es realmente el formato de un real, o de un número en BCD?.

Para un real, la estructura es la siguiente:

Un número en BCD agrupa sus bits en tétradas, ya que realmente es un valor en hexadecimal, pero con la salvedad de que el valor máximo que contiene en una tétrada no supera nunca el 9. Por lo tanto, en los cuatro bits de la tétrada aparecerá como máximo un 1001.

5.7.2 Conversión de enteros.

OPERACIÓN	SIGNIFICADO
BTI	Convertir de BCD a entero
BTD	Convertir de BCD a entero doble
ITB	Convertir de entero a BCD
ITD	Convertir de entero a doble entero
DTB	Convertir de doble entero a BCD

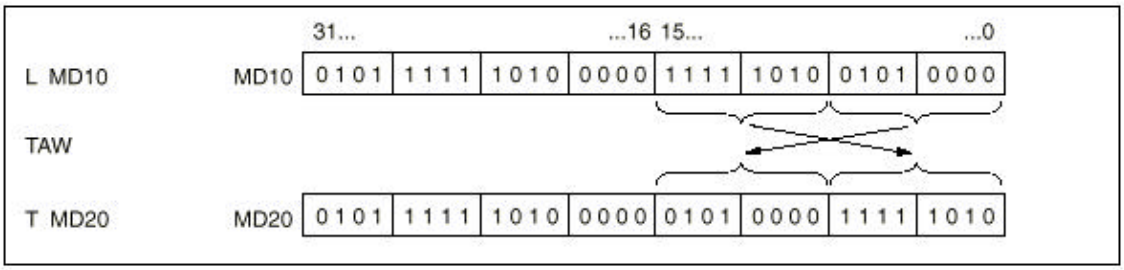
5.7.3 Conversión de reales

OPERACIÓN	SIGNIFICADO	Descripción
DTR	Convertir de doble entero a real	Convertir de doble entero a real
RND	Redondea a entero	Esta operación redondea el número convertido al próximo entero. Si la fracción del número convertido se encuentra exactamente entre un resultado par y un resultado impar, la operación elige el resultado par.
RND+	Redondea a entero doble superior	Esta operación redondea el valor real al entero doble superior con respecto a los decimales del real.
RND-	Redondea a entero doble inferior	Esta operación redondea el valor real al entero doble inferior con respecto a los decimales del real.
TRUNC	Truncar un real en entero doble.	Esta operación convierte de real a entero doble despreciando el resto.

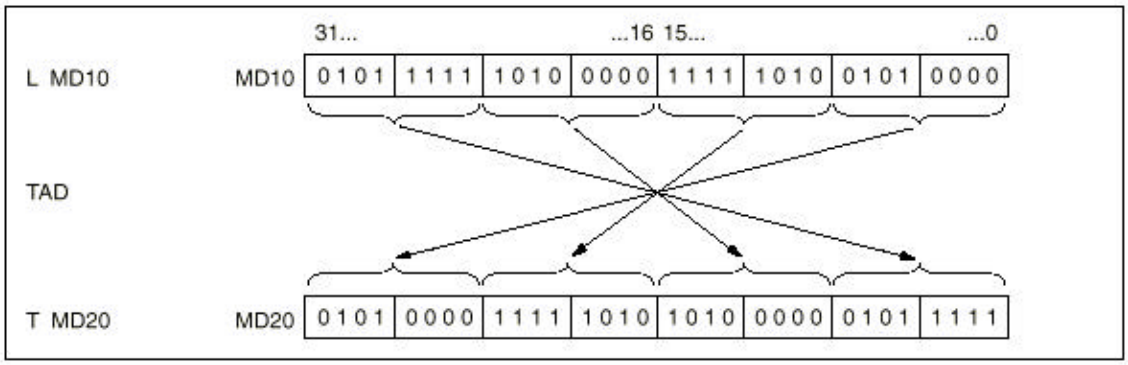
5.7.4 Operaciones de inversión de bytes.

Las operaciones de inversión invierten el orden de los bytes dentro de una palabra o doble palabra.

La operación TAW invierte los bytes de una palabra contenida en el acumulador.



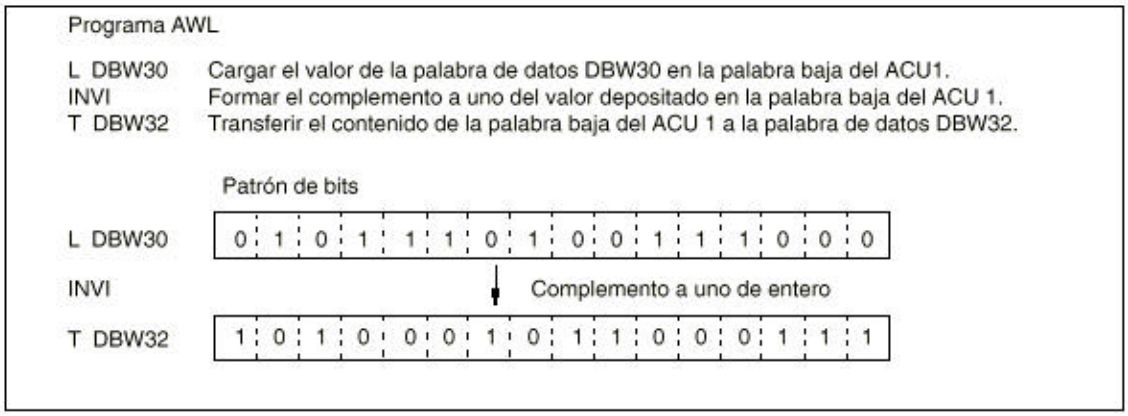
La operación TAD invierte los bytes de una doble palabra uno a uno.



5.7.5 Operaciones de inversión de bits.

Puede ser interesante en ciertas ocasiones invertir el estado de cada uno de los bits de una variable. Disponemos de las siguientes posibilidades:

- INVI: invertir un entero bit a bit.
- INVD: invertir un doble entero bit a bit.

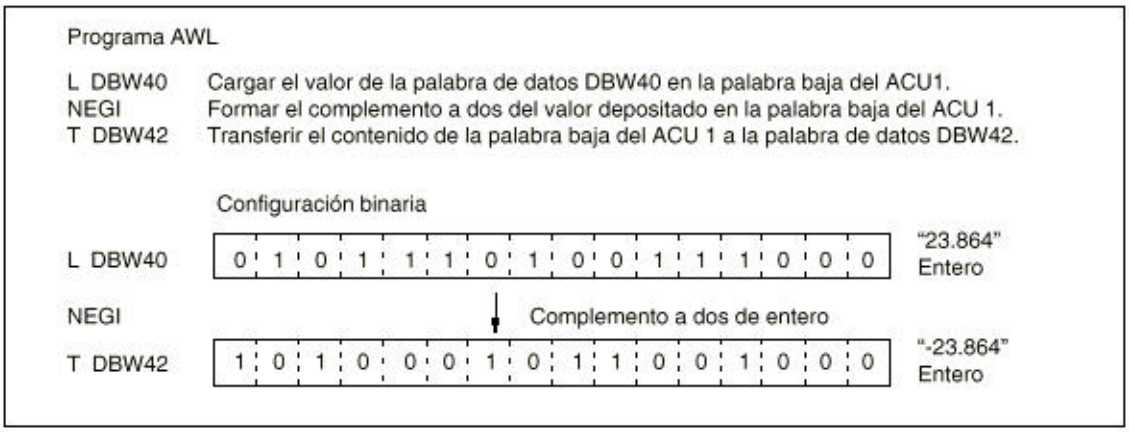


5.7.6 Operaciones de negación.

Las operaciones de negación sirven para convertir un número positivo en su homónimo negativo. Se realiza el complemento a dos de la variable a tratar.

Disponemos de tres instrucciones para realizar dicha tarea:

- NEGI: niega un número entero.
- NEGD: niega un número doble entero.
- NEGR: niega un número real.



5.8 Operaciones lógicas.

Las operaciones lógicas son instrucciones muy útiles a la hora de tratar estados de la instalación. A menudo su desconocimiento obliga al programador a realizar complicadas secuencias de bits, cuando se hubiera podido tratar todos los datos agrupados en una doble palabra, y resolver la tarea en pocas líneas de código.

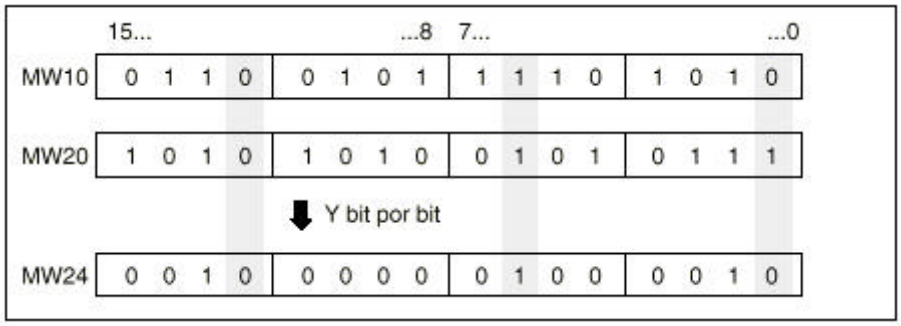
5.8.1 Operación lógica Y.

La Y lógica en Step 7 puede realizarse en dos formatos distintos:

- UW: realiza una Y lógica entre los dos acumuladores a nivel de palabra (utilizando los 16 primeros bits).
- UD: realiza una Y lógica entre los dos acumuladores a nivel de doble palabra (utilizando los 32 bits).

Las operaciones Y se utilizan para “quitar” bits (o aplicar una máscara a una variable) con pocas instrucciones. Evidentemente, esto se podría hacer repitiendo 32 o 16 veces la sentencia “si no tengo este bit, reseteamos este otro”. Pero la forma elegante de programar esta acción se basa en la Y lógica.

Supongamos que deseamos resetear una serie de alarmas de una palabra determinada (MW10), eliminando los indicados en una máscara de alarmas no útiles (MW20). Las alarmas que queden (MW24) son para nosotros las válidas.



5.8.2 Operación lógica O.

La O lógica en Step 7 puede realizarse en dos formatos distintos:

- OW: realiza una O lógica entre los dos acumuladores a nivel de palabra (utilizando los 16 primeros bits).
- OD: realiza una O lógica entre los dos acumuladores a nivel de doble palabra (utilizando los 32 bits).

Las operaciones O se utilizan para “poner” bits (o aplicar una máscara a una variable) con pocas instrucciones. Al igual que anteriormente, esto se podría hacer repitiendo 32 o 16 veces la sentencia “si tengo este bit, activamos este otro”. Pero la forma elegante de programar esta acción se basa en la O lógica.

Supongamos que tenemos que activar un motor en una palabra de estado de motores. Las instrucciones correspondientes serían:

```
L MW10     // 2#0100_0001_1000_0000
L MW20     // 2#0000_0000_0000_0001
OW
T MW24     // 2#0100_0001_1000_0001
```

5.8.3 Operaciones lógicas XOR.

La XOR lógica en Step 7 puede realizarse en dos formatos distintos:

- XOW: realiza una XOR lógica entre los dos acumuladores a nivel de palabra (utilizando los 16 primeros bits).
- XOD: realiza una XOR lógica entre los dos acumuladores a nivel de doble palabra (utilizando los 32 bits).

Las operaciones XOR se utilizan para detectar cambios de estado en bits de variables con pocas instrucciones. Evidentemente, esto se podría hacer repitiendo 32 o 16 veces la sentencia “si antes valía x y ahora vale y, activamos la variable”. Pero la forma elegante de programar esta acción se basa en la XOR lógica.

Supongamos que deseamos conocer la activación de alguna alarma de las 32 que componen una doble palabra de marcas (MD0). Cuando se produzca una transición positiva de alguno de sus bits deseamos setear una salida, que está cableada a una sirena

que hace sonar la alarma en la instalación. Con la siguiente subrutina, utilizando la XOD, podremos realizar esta acción, muy útil en todas las instalaciones.

```
    L    MD    0          // carga valor_alarmas_actual
    L    MD    4          // carga valor_alarmas_old
    XOD                   // or lógica
    L    MD    0          // carga valor_alarmas_actual
    UD                    // y lógica
    L    0                // carga 0
    ==D                   // comprueba que el ACU esta a cero
    SPB  m001
    SET                   // si hay algún bit con flanco positivo
    S    A     4.0        // activa la sirena

m001: NOP  0
    U    E     0.0        // si presionan el pulsador de parar sirena
    R    A     4.0        // para la sirena

    L    MD    0          // iguala valor_alarmas_old
    T    MD    4          // a valor_alarmas_actual
```

El único punto que puede no ser obvio es la parte en la que se realiza la Y lógica con el valor del ACU. Después de realizar la XOD, necesitamos conocer si el cambio en algún bit de la palabra se debe a que aparece la alarma o a que desaparece. Es por esto por lo que se requiere realizar esta operación, ya que en caso contrario la sirena sonaría cuando aparece la alarma y cuando desaparece.

La función XOR es la gran desconocida por parte de los programadores para detectar qué está cambiando en su proceso.

5.9 Desplazamiento y rotación.

El desplazamiento y la rotación son las operaciones necesarias para poder realizar un control secuencial (por pasos) de un programa. Normalmente se utilizará el desplazamiento en la mayoría de los casos, ya que la rotación únicamente tiene sentido en un caso concreto frente al desplazamiento, como vamos a ver a continuación.

5.9.1 Desplazamiento de palabras.

El desplazamiento desplaza el contenido del ACU1 en una dirección, ya sea a la derecha o a la izquierda. Las diferentes posibilidades son:

OPERACIÓN	SIGNIFICADO
SLW x	Desplazamiento de la palabra del ACU1 a la izquierda x bits.
SRW x	Desplazamiento de la palabra del ACU1 a la derecha x bits.
SLD x	Desplazamiento de la palabra doble del ACU1 a la izquierda x bits.
SRD x	Desplazamiento de la palabra doble del ACU1 a la derecha x bits.

Podemos ver un ejemplo de funcionamiento a nivel de bits de un desplazamiento de tres bits a la derecha (SRD 3).

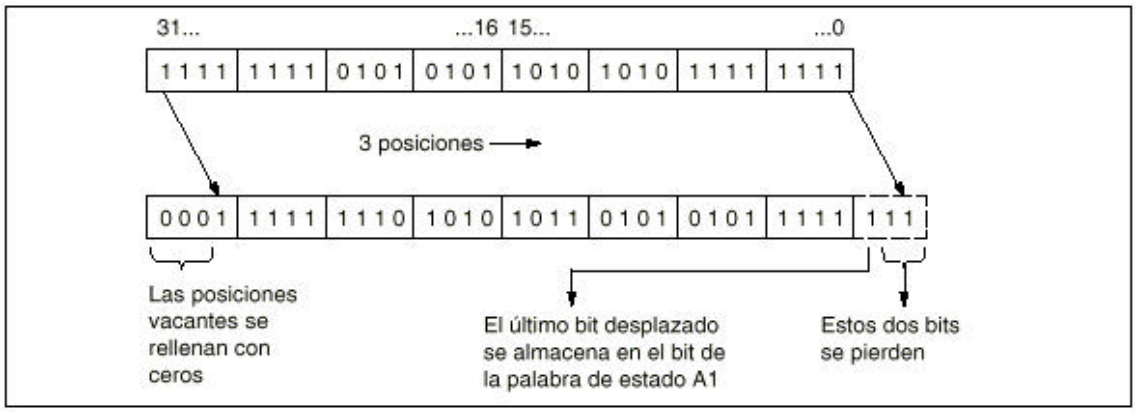


Figura 15. Desplazamiento de una doble palabra 3 posiciones a la derecha.

La utilidad de estas instrucciones va a ser la activación de una serie de motores o etapas de programa de una manera secuencial. Suponemos que queremos activar una serie

de 10 bombas, conectadas a las marcas M1.0 hasta la M0.1. Con un bit, que se va a desplazar en este rango, indicaremos que bomba debe de arrancar cada vez.

```
    L    0
    L    MW    0          // si no hay ningún bit activo en la variable
    ==I                   // vamos a activar el primero
    SPB  m001

    L    MW    0          // carga la variable
    SLW  1                // y desplázala un bit a la izquierda
    T    MW    0          // guárdala ya desplazada

    U    M     1.2        // si hemos llegado a la ultima bomba
    SPB  m001             // volvemos a activar la primera
    BEA

m001: NOP  0
    L    1                // activa la primera bomba
    T    MW    0
```

Es importante tener cuidado con dos cosas en estas acciones:

- No activar bits de esta palabra por otras partes del programa.
- Entrar a esta subrutina exclusívamente cuando se deba realizar el cambio de bomba (normalmente es cada cierto tiempo). Para conseguirlo se puede realizar un salto condicional desde la OB1 utilizando la función "*activar tareas cada cierto tiempo"* vista anteriormente en "*Ladrillos de un programa".*

5.9.2 Rotación de palabras.

La rotación de palabras rota el contenido del ACU1, como se puede apreciar en la imagen inferior.

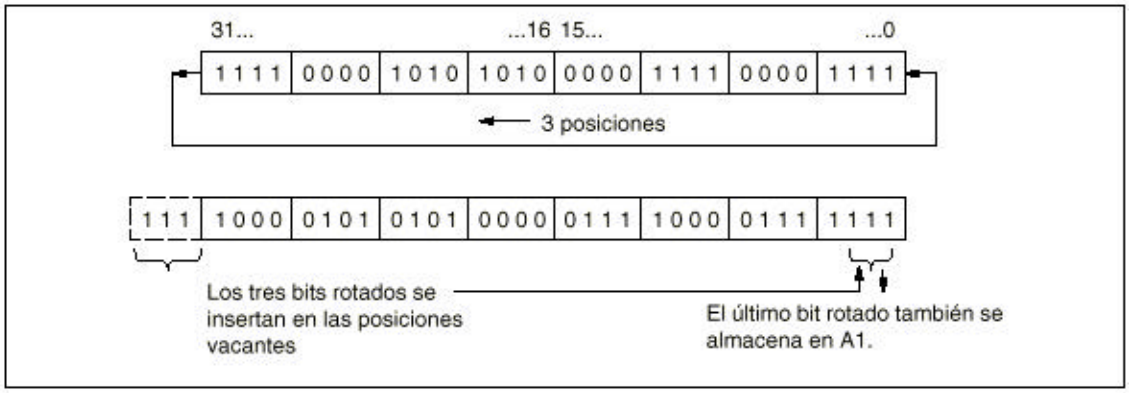


Figura 16. Rotación de dobles palabras en tres posiciones a la izquierda.

Esto la verdad es que sirve para bien poco, salvo en el caso que vamos a describir. Supongamos una instalación en la que deseamos arrancar tres motores secuencialmente, y esta agrupación de motores se repite varias veces. De esta manera, arrancará el primero, cuarto, séptimo, etc... Posteriormente, el segundo, quinto, octavo, etc...Y por último, el tercero, sexto, noveno...volviéndose a repetir la secuencia.

```
    L    0
    L    MD    0              // si no hay ningún bit activo en la variable
    ==I                      // vamos a activar la primera secuencia
    SPB  m001

    L    MD    0              // carga la variable
    RLD  1                   // y desplázala un bit a la izquierda
    T    MD    0              // guárdala ya desplazada

    BEA

m001: NOP  0
    L    2#10010010010010010010010010010001   // activa la primera secuencia
    T    MD    0
```

Se podría haber realizado con un desplazamiento, pero con una rotación queda mucho más elegante, y ocupa menos líneas de código, ya que no debemos de preocuparnos por comprobar cuando finaliza la secuencia para volver a la primera.

Las operaciones de rotación de que disponemos son:

OPERACIÓN	SIGNIFICADO
RLD x	Rotación de la palabra del ACU1 a la izquierda x bits.
RRD x	Rotación de la palabra del ACU1 a la derecha x bits.

5.10 Operaciones de control de programa.

Las operaciones de control de programa nos van a permitir decidir por donde debe de discurrir, al igual que el agua por las acequias, nuestro programa de PLC. Aunque son de sobra conocidas por el lector, intentaremos darle una utilidad a las mismas que pueda ser interesante para el mismo.

5.10.1 SPL: el "*select case*" repartidor de tareas.

La instrucción SPL, tiene exactamente la misma funcionalidad que el famoso "*select case...*" de programación. La intención es disponer de una operación, que dependiendo de el valor de una variable salte a una parte del programa en concreto.

Una posible utilidad de esto sería realizar una tarea de arranque de la instalación, cuando pulsamos un botón. Al arrancar, calentaremos motores, después liberaremos frenos, avisaremos del arranque mediante una sirena, y por fin arrancaremos. Mediante la MB0 iremos controlando el estado de la secuencia de arranque en la cual nos encontramos en cada momento.

```
    UN    M     11.0          // mientras no se indique arrancar instalación
    BEB                       // no se ejecuta la secuencia

    L     MB    0             // carga marca de estado de secuencia
    SPL   fin                 // salta a fin si MB 0 > 4
    SPA   eta0                // salta a eta1 si MB 0 = 0
    SPA   eta1                // salta a eta1 si MB 0 = 1
    SPA   eta2                // salta a eta1 si MB 0 = 2

fin:  NOP   0                 // arrancar la instalación
    R     M     10.0
    R     M     10.1
    R     M     10.2
    R     M     11.0          // finaliza la secuencia de arranque
    BEA

eta0: NOP   0                 // instrucciones para calentar motores
    NOP   0                   // cuando estén calientes activamos M10.0
    UN    M     10.0          // si los motores aun no están calientes
    BEB
    L     1
```

```
    T    MB    0                    // pasa a la siguiente etapa de arranque
    BEA

eta1: NOP  0                        // liberamos frenos
    NOP  0                          // cuando estén liberados activamos M10.1
    UN   M    10.1                  // si los motores aun no están liberados
    BEB
    L    2
    T    MB    0                    // pasa a la siguiente etapa de arranque
    BEA

eta2: NOP  0                        // avisamos del arranque mediante una sirena
    NOP  0                          // cuando finalice el tiempo de aviso activamos la M10.2
    UN   M    10.2                  // si los motores aun no están liberados
    BEB
    L    3
    T    MB    0                    // pasa a la siguiente etapa de arranque
    BEA
```

5.10.2 SPBN: el "*if...then*" de los PLC's.

La operación SPBN realiza la misma función que la sentencia de programación de cualquier lenguaje de alto nivel "*if...then*". Además, con un poco de gracia, se construye uno un "*if...then...else*" con muy pocas líneas de código, como muestra el programa inferior.

```
    U    E    0.0                   // si se activa la entrada
    SPBN m001
    NOP  0                          // realizamos este código (la parte de if.. then).
    NOP  0
    BEA

m001: NOP  0                        // en caso contrario, estas otras (la parte de else).
    NOP  0
```

5.10.3 LOOP: el "for..to..next" que envía a STOP.

La operación LOOP nos permite realizar bucles de programa al estilo "*For..to..next*" en nuestro programa de Step 7. Puede parecer interesante esta operación, pero la verdad es que existen formas mejores de repetir una secuencia de código dentro de nuestro programa, sin correr el riesgo de caer en un bucle recurrente que nos tire abajo la CPU.

En cualquier caso, veamos la utilidad de LOOP. Supongamos que deseamos realizar un retardo en el arranque de nuestra instalación de unas décimas de segundo. En Step 7 esto

no es necesario realizarlo por programa, ya que en Hardware de la CPU, dentro de la solapa "*Arranque*", podemos definir este tiempo de retardo para disponer de alimentación en los módulos de la periferia (por defecto, tarda 650 ms en pasar de STOP a RUN). Este tiempo, como mínimo será de 100 ms, aumentándose en la cantidad que indiquemos en el parámetro "*señal ready de los módulos*".

Pero supongamos que deseamos realizar un bucle de repetición de código. La función LOOP tendría el siguiente aspecto

```
    L    10

cont: T    MW    0              // inicializa contador

    NOP  0                  // estas líneas de código se van a realizar
    NOP  0                  // 10 veces por ciclo de programa

    L    MW    0
    LOOP cont               // decrementa el ACU1 y comprueba si ha llegado a 0 saltando en caso contrario a cont
```

6

OB´s, FB´s, FC´s y DB´s en S7

6.1 Los OB´s.

Los OB´s en S7 aunque pueden generarse como nuevos bloques, es recomendable que sean importados de la librería *Standard Library* de Step 7. En caso contrario, el nombre simbólico de la OB no será importado a nuestro proyecto (en versiones anteriores a la versión 5 de Step 7, tampoco los símbolos de las variables temporales de la tabla de declaración de variables de la OB).

6.1.1 El OB1.

Como es conocido, el OB1 es la subrutina principal en la que empieza el ciclo de programa de PLC y por la que finaliza. Toda subrutina para poder ser ejecutada debe de ser llamada desde la OB1. De no ser así no se ejecutará, pese a encontrarse en la memoria del PLC.

En la tabla de declaración de la OB1 existen una serie de parámetros que pueden ser de utilidad para el programador. Estos son:

- **OB1_PREV_CYCLE INT**: Tiempo de ejecución del ciclo anterior (en ms).
- **OB1_MIN_CYCLE INT:** Tiempo de ciclo mínimo (ms) desde el último arranque.
- **OB1_MAX_CYCLE INT**: Tiempo de ciclo máximo (ms) desde el último arranque.

Mediante el uso de estos valores temporales podremos controlar que la CPU se nos va a ir por tiempo de ciclo. Este es un problema que puede pasar al programador en una puesta en marcha, al ir añadiendo rutinas al programa, con lo que aumenta el tiempo de ciclo. Puede llegar el momento en que sobrepasemos el valor preseleccionado en Hardware de la CPU. Si esto es así, la instalación se para, lo que en ciertas aplicaciones críticas puede ser desastroso.

Para evitar este inconveniente, podemos comparar **OB1_MAX_CYCLE INT** con nuestro tiempo de ciclo, y si nos acercamos al mismo activar un mensaje en la OP o hacer

sonar una sirena. En dicho caso, podrías utilizar la SFC43 que nos relanzaría el perro guardián, aunque es preferible ir a hardware y aumentar el tiempo de vigilancia de ciclo.

```
L     #OB1_MAX_CYCLE            // CARGA EL TIEMPO DE CICLO
L     100                       // CARGA 100 (MS)
>I                              // SI EL TIEMPO DE CICLO ES MAYOR DE 100
=     A    124.0                // ACTIVA SIRENA (TENEMOS 50 MS DE SEGURIDAD AUN)
```

6.1.2 OB's de alarma horaria (OB 10 hasta OB 17).

Un OB de alarma horaria es una subrutina que automáticamente es llamada por el sistema, una vez o cada cierto periodo de tiempo, a partir de una fecha y hora determinada. Disponemos en S7 de ocho OB´s distintas para estos menesteres, pero los S7 300 únicamente soportan una de ellas, la OB10.

La mayor utilidad de este tipo de funciones es según las utilicemos:

- Arranque único: útil para avisar desde un display de un determinado evento que ocurrirá en una fecha determinada a una hora determinada.
- Arranque cíclico: útil para labores de mantenimiento (engrase de cojinetes, aviso de revisión de motores, etc...).

Los intervalos de llamada que aceptan las OB´s de alarma horaria son:

- una vez
- cada minuto
- cada hora
- cada día
- cada semana
- cada mes
- cada año.

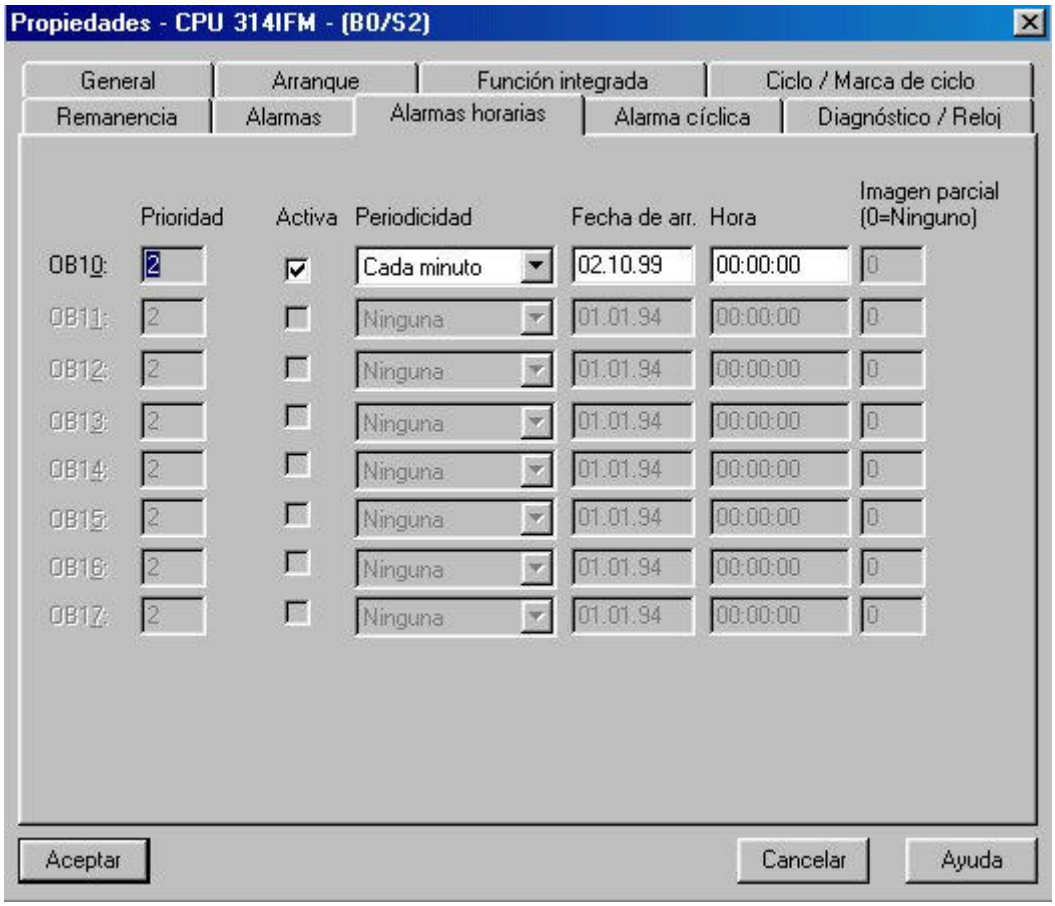
Existen dos maneras o formas de arrancar las alarmas horarias:

- desde el Hardware de Step 7, de manera automática.
- A través la SFC 30 del sistema, que nos permite además ajustar la frecuencia de llamada.

A continuación se muestran los pasos a seguir para el primero de los casos: arrancar la alarma horaria desde Step 7. Dentro de Hardware, parametrizamos la alarma como muestra la figura inferior.

Necesitaremos disponer de la OB10 en nuestro PLC, por lo que la importaremos de la librería *Standard Library->Obs*. En caso contrario, cuando se cumpla el tiempo seleccionado, al no encontrar la OB en cuestión, el sistema busca la OB85 (error de programación de módulo), y si tampoco la encuentra, el equipo pasará a stop.

Dentro de la OB podemos programar una señal alterna a la M0.0 (ver apartado ladrillos de un programa). Si asociamos esta señal en la OB1 a una salida de nuestro equipo, observaremos una cadencia de activación de 1 minuto en la misma.



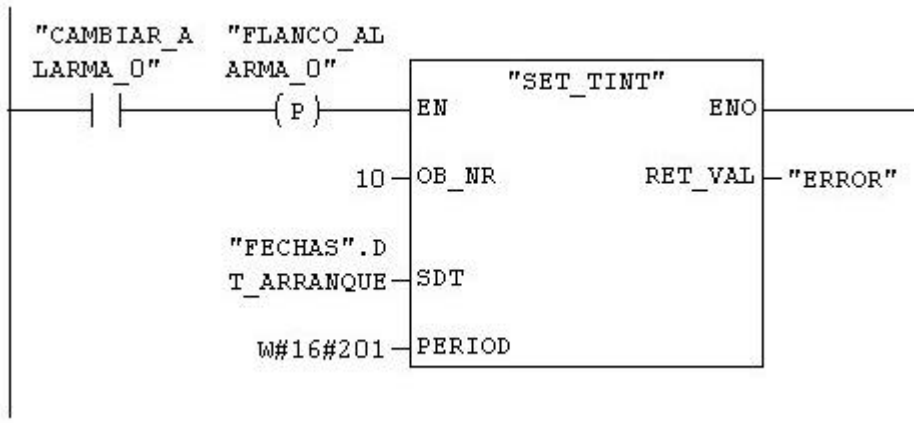
La segunda posibilidad es controlar la alarma horaria a través de funciones de sistema. Podemos configura, como ya hemos visto, la fecha y hora de arranque de la alarma horaria desde Step 7, pero también a través de la SFC28.

Los parámetros de la SFC28 son:

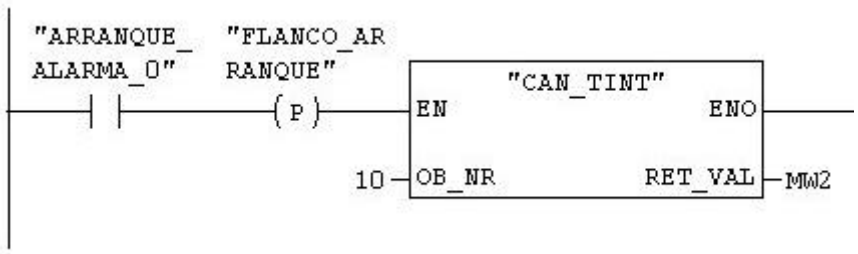
PARAMETRO	TIPO	SIGNIFICADO
OB_NR	INT	Número de OB de alarmas horarias a la que se desea cambiar la fecha y hora de arranque. Valores: 10 a 17. En los S7 300 sólo es válido el valor 10.
SDT	DATE_AND_TIME	Fecha y hora nueva a asignar a la OB de alarmas.
PERIOD	WORD	Periodo asignado a la repetición de la llamada a la OB. Los valores que puede adoptar son: W#16#0000 = una vez W#16#0201 = cada minuto W#16#0401 = cada hora W#16#1001 = diaria W#16#1201 = semanal W#16#1401 = mensual

		W#16#1801 = anual W#16#2001 = al final del mes
RET_VAL	WORD	Valor de retorno de error. Puede adoptar los siguientes valores en hexadecimal: 0000 No ha ocurrido ningún error 8090 Parámetro OB_NR erróneo 8091 Parámetro SDT erróneo 8092 Parámetro PERIOD erróneo 80A1 El instante de arranque ajustado ya ha pasado.

Un ejemplo del seteo de una nueva parametrización de fecha y hora a una alarma horaria es el siguiente:



Una vez se dispone de la parametrización correcta de fecha, hora y frecuencia para el tratamiento de la alarma horaria, es necesario activarla con la SFC30.

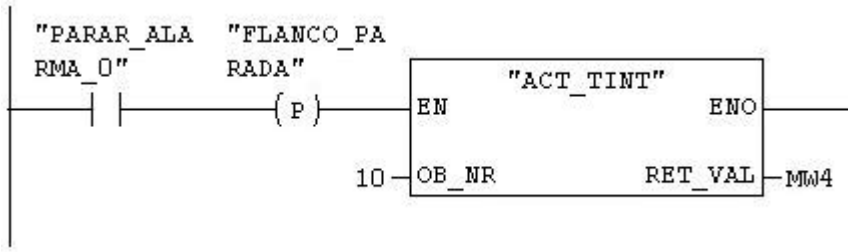


Los parámetros de la misma son:

PARAMETRO	TIPO	SIGNIFICADO
OB_NR	INT	Número de OB de alarmas horarias a la que se desea cambiar la fecha y hora de arranque. Valores: 10 a 17. En los S7 300 sólo es válido el valor 10.
RET_VAL	WORD	Valor de retorno de error. Puede adoptar los siguientes valores en hexadecimal: 0000 No ha ocurrido ningún error. 8090 Parámetro OB_NR erróneo

		80A0 No se han ajustado la fecha y hora de arranque para el OB de alarma horaria indicado 80A1 El tiempo activado está en el pasado; el error ocurre solamente en caso de ejecución única.

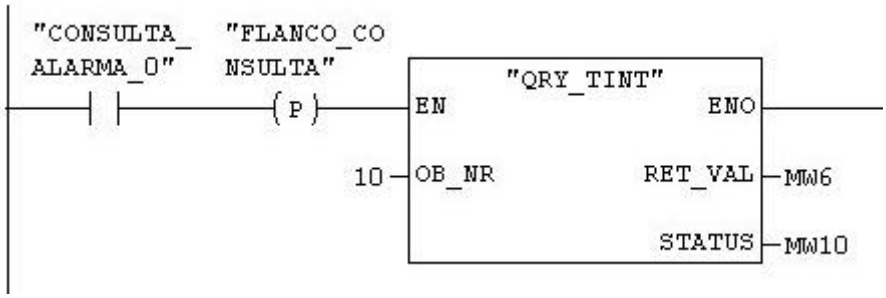
Una vez arrancada la alarma horaria se puede detener llamando a la función del sistema SFC29, como muestra la figura:



PARAMETRO	TIPO	SIGNIFICADO
OB_NR	INT	Número de OB de alarmas horarias a la que se desea cambiar la fecha y hora de arranque. Valores: 10 a 17. En los S7 300 sólo es válido el valor 10.
RET_VAL	WORD	Valor de retorno de error. Puede adoptar los siguientes valores en hexadecimal: 0000 No ha ocurrido ningún error 8090 Parámetro OB_NR erróneo 80A0 No se han definido la fecha y hora de arranque para el OB de alarma horaria indicado

Es importante tener en cuenta que una vez se haya detenido la alarma, se requiere volver a asociar con la SFC28 una fecha y hora a la alarma antes de arrancarla de nuevo con la llamada a la SFC30.

En cualquier momento podremos consultar el estado de la alarma horaria mediante la SFC31.



PARAMETRO	TIPO	SIGNIFICADO
OB_NR	INT	Número de OB de alarmas horarias a la que se desea cambiar la fecha y hora de arranque. Valores: 10 a 17. En los S7 300 sólo es válido el valor 10.
RET_VAL	WORD	Valor de retorno de error. Puede adoptar los siguientes valores en hexadecimal: 0000 No ha ocurrido ningún error. 8090 Parámetro OB_NR erróneo.
STATUS	WORD	Estado actual de la alarma horaria. El significado de cada uno de los bits de la palabra es: Bit 0 Alarma horaria habilitada por el sistema operativo. Bit 1 No se rechazan nuevas alarmas horarias. Bit 2 Alarma horaria sin activar o transcurrida. Bit 3 – – Bit 4 OB de alarma horaria sin cargar. Bit 5 La ejecución del OB de alarma horaria está bloqueada por una función de prueba en curso.

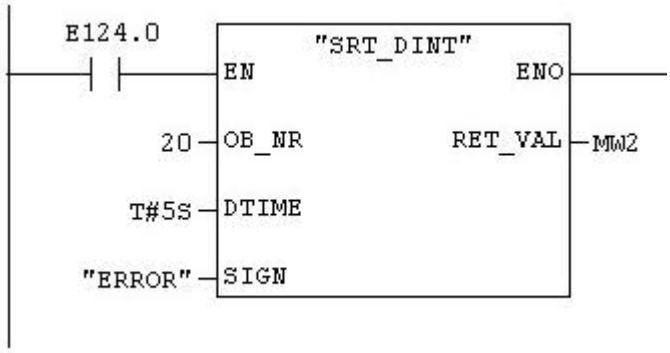
6.1.3 OB's de alarma de retardo (OB 20 hasta OB 23).

Las OB´s de retardo sirven para ejecutar una acción un tiempo después de que produzca un evento. No se repiten automáticamente, por lo que necesariamente deberemos de volver a lanzar la llamada desde el código de nuestro programa.

Los equipos S7 en principio disponen de 4 alarmas de retardo, que activan respectivamente de la OB20 a la OB23 (en los S7 300 sólo está habilitada la OB20).

Supongamos que deseamos engrasar un motor diez horas después de que arranque el mismo. En principio se podría pensar en utilizar un temporizador con retardo a la conexión memorizado. El inconveniente estriba en el hecho de que la base de tiempo de un temporizador es de 9990 segundos como máximo, por lo que tendríamos que recurrir a combinaciones de temporizadores. Con una llamada a la OB20 solventaremos el problema, ya que la base de tiempos de llamada es de formato TIME, muy superior a S5TIME, la base de los temporizadores S5.

Para arrancar la llamada a la OB20 utilizaremos la SFC32 según la figura siguiente.

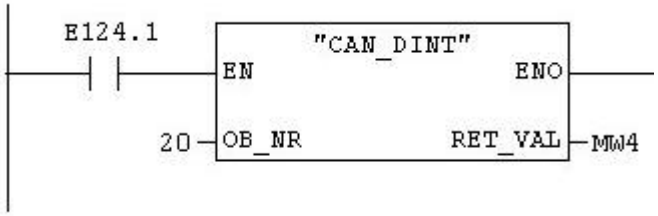


Los parámetros de la función del sistema son:

PARAMETRO	TIPO	SIGNIFICADO
OB_NR	INT	Número de OB de alarmas de retardo. Valores posibles: 20 a 23. En los S7 300 sólo es válido el valor 20.
DTIME	WORD	Valor de retarda para entrar en la OB. Valores posibles: 1 a 60000 ms.
SIGN	WORD	Valor que se pasa como parámetro a la OB en la tabla de declaración cuando es llamada. Se puede utilizar como información a utilizar posteriormente dentro de la OB.
RET_VAL	INT	Código de error de la función. Los valores posibles son:

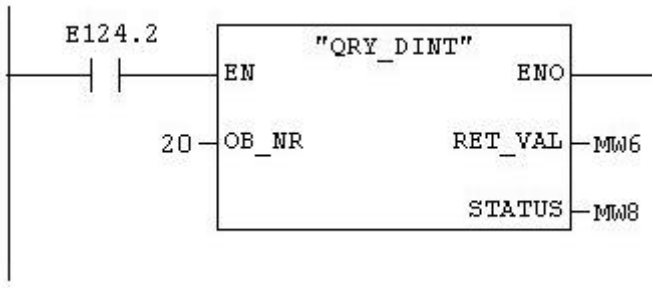
		0000 No ha ocurrido ningún error. 8090 Parámetro OB_NR erróneo. 8091 Parámetro DTIME erróneo.

Una vez arrancada la alarma de retardo, y antes de que se cumpla el tiempo y salte automáticamente a la OB asociada, podemos bloquear la llamada a la misma mediante la función SFC33.



PARAMETRO	TIPO	SIGNIFICADO
OB_NR	INT	Número de OB de alarmas de retardo. Valores posibles: 20 a 23. En los S7 300 sólo es válido el valor 20.
RET_VAL	INT	Código de error de la función. Los valores posibles son: 0000 No ha ocurrido ningún error. 8090 Parámetro OB_NR erróneo. 80A0 Alarma de retardo sin arrancar.

También podremos consultar el estado de una petición de alarma de retardo mediante la función SFC34.



PARAMETRO	TIPO	SIGNIFICADO
OB_NR	INT	Número de OB de alarmas de retardo. Valores posibles: 20 a 23. En los S7 300 sólo es válido el valor 20.
RET_VAL	INT	Código de error de la función. Los valores posibles son: 0000 No ha ocurrido ningún error. 8090 Parámetro OB_NR erróneo.
STATUS	INT	Estado de la alarma de retardo. El significado de los bits de la palabra es el siguiente:

		Bit 0 Alarma de retardo habilitada por el sistema operativo. Bit 1 No se rechazan nuevas alarmas de retardo. Bit 2 Alarma de retardo no activada o transcurrida. Bit 3 – – Bit 4 OB de alarma de retardo no cargado. Bit 5 La ejecución del OB de alarma de retardo ha sido bloqueada por una función de prueba en curso.

6.1.4 OB's de alarma cíclica (OB 30 hasta OB 38).

Las OB´s de alarma cíclica son bloques llamados a intervalos regulares de tiempo por el sistema operativo del PLC. Para que se produzca la llamada a los mismos el único requisito es su existencia dentro de la memoria del PLC. El periodo de tiempo para las llamadas a las OB´s de alarma cíclica es parametrizable desde Hardware de Step 7.

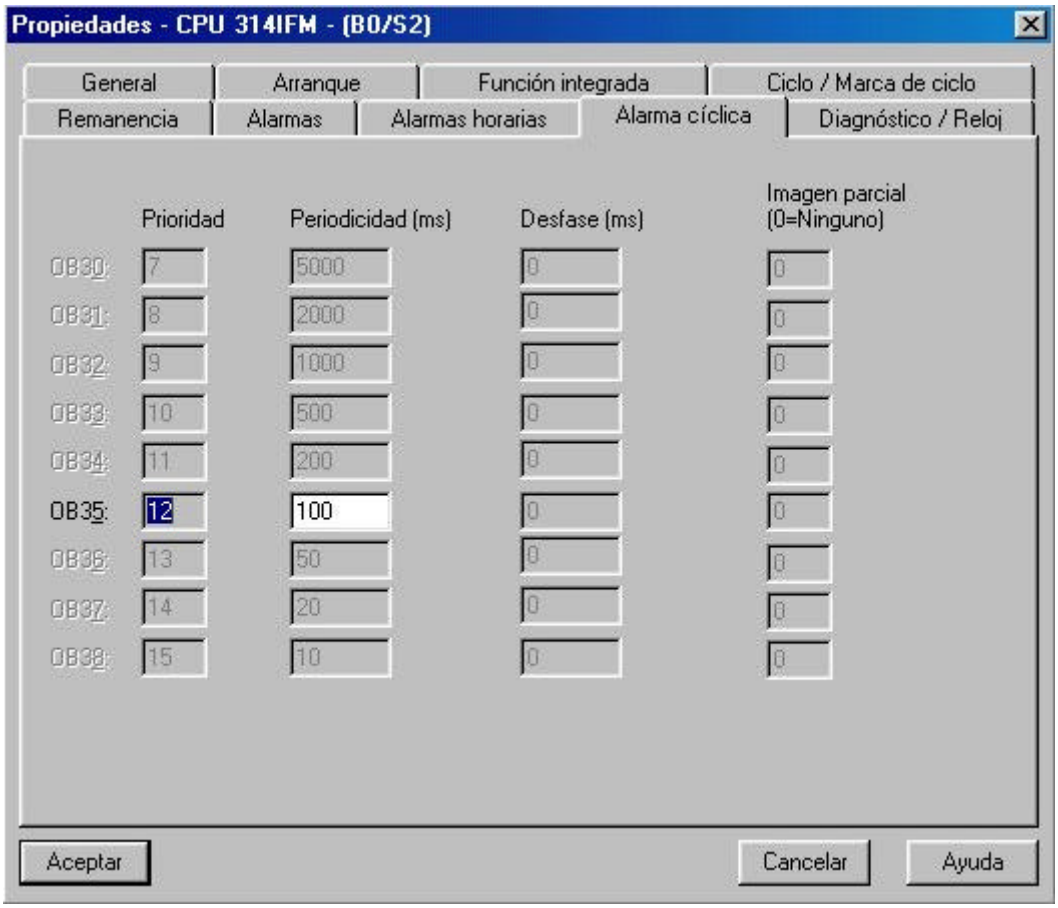


Figura 18. Parametrización del tiempo e llamada a la OB35 desde Hardware de Step 7.

En los Simatic S7 300 únicamente se encuentra habilitada la OB35 para estos menesteres. Se suele utilizar la OB35 para las llamadas a lecturas analógicas, regulación PID, y otros procesos que requieran un tratamiento uniforme en el tiempo, e independiente del tiempo de ciclo del programa.

6.1.5 OB's de alarma de proceso (OB 40 hasta OB 47).

Determinados módulos de función, o módulos de entradas analógicas son capaces de generar alarmas de proceso que detienen la secuencia de programa del PLC, y realizan un salto a la OB asociada a los mismos. Dentro de dicha OB se debe de programar el código asociado al evento que generó la alarma de proceso.

En los S7 se dispone de 8 alarmas de proceso (en los S7 300 únicamente la OB40) que se asociarán a los eventos que generen los módulos en cuestión. Un ejemplo sería el tratamiento de las alarmas generadas por el desbordamiento de las entradas analógicas de un módulo de S7.

Más adelante en labores de contaje rápido utilizaremos la OB40.

6.1.6 OB de error de tiempo (OB 80).

Como sabemos, los PLC´s disponen de una función de perro guardián que se dispara si el programa del autómata no es capaz de finalizar antes de que pase un determinado tiempo, parametrizable desde Hardware de Step 7.

Cuando esto ocurre, el sistema automáticamente busca la OB 80 en memoria, y si existe, salta a la misma. Lógicamente, esta posibilidad se nos da para poder solventar el problema de superar el tiempo de ciclo de manera inadvertida mientras se está realizando la puesta en marcha. Dentro de la OB80 deberemos de activar una alarma que nos indique que se está produciendo esta circunstancia, además de intentar relanzar el tiempo de ciclo para que no se pare la instalación por tiempo de programa.

Existen tres posibilidades por las cuales una instalación se nos vaya por tiempo de ciclo:

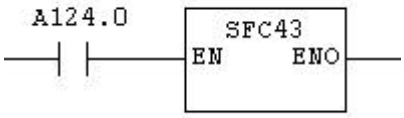
- Que estuviésemos muy cerca del tiempo de ciclo máximo, y hemos introducido una nueva subrutina que aumenta el mismo por encima del límite. Esta es la situación más frecuente.
- Que alguna condición que no se había producido aún entre en conjunción con otras que haga que se deban procesar varias tareas que normalmente nunca se realizan simultáneamente. Esta es la situación más interesante de tratar, ya que es imprevisible y suele no ocurrir durante la puesta en marcha, sino posteriormente, cuando el técnico ya no se encuentra en la instalación.
- Que entremos en un bucle de programa del cual no se puede salir. Esto solo ocurre durante la puesta en marcha al equivocarnos en un salto a una meta. Suele convenir que la CPU pase a Stop, ya que el PLC deja de controlar la instalación (se queda colgado como si fuese un ordenador).

Por lo tanto, solo desearemos que se nos prolongue el tiempo de ciclo cuando se rebase el tiempo máximo establecido, que se nos avise con una alarma y además, que si se queda en un bucle el programa de PLC pare la instalación. Esto se consigue programando la OB80, y colocando el siguiente código:

```
SET
S   A   124.0
```

La salida A124.0 será nuestra alarma de tiempo de ciclo sobrepasado.

En nuestro programa principal (OB1) programaremos el siguiente segmento:



La SFC43 “redispara” el tiempo de ciclo, por lo que dispondremos del doble de tiempo de ciclo. Con esto el tiempo de ciclo solo se relanza una vez, por lo que si volvemos en un mismo ciclo a sobrepasar el tiempo de ciclo, el PLC pasa a stop, asegurándonos de no dejar “colgada” la instalación.

6.1.7 OB de fallo de alimentación (OB 81).

La OB81 es llamada por el sistema operativo del PLC cada vez que se detecta un cambio en el respaldo de la alimentación por parte de la pila de la CPU. Debe de ser utilizada para que el usuario de la instalación sepa que la vida útil de su batería ha llegado a su fin y es necesario sustituirla.

Sin embargo, al contrario de lo que suele pensar la gente, la OB81 no es únicamente llamada cuando falla la batería. Se entra una sola vez en la OB81 cuando:

- Si se detecta estando en RUN, que la batería está fallando.
- Si frente a una transición de stop a RUN se detecta fallo de batería.
- Si se detecta que se ha repuesto la batería agotada por otra recargada.

Este último supuesto nos obliga a consultar si el evento por el que entramos a la OB81 es de fallo de batería o de reposición de batería. Para ello utilizaremos la variable *OB81_EV_CLASS* de la tabla de declaración del a OB81, que nos indica el tipo de evento por el que entramos a la OB. El programa siguiente introducido en la OB81 nos indica si es una alarma de fallo (cambiar batería), o un aviso (batería correctamente sustituida).

```
    L    #OB81_EV_CLASS
    L    B#16#39
    ==I
    SPB   m001                  // FALLO EN BATERIA TAMPON

    L    #OB81_EV_CLASS
    L    B#16#38
    ==I
    SPB   m002                  // REPOSICION DE BATERIA TAMPON

    BEA

m001: NOP   0
    SET
    S    A    124.0             // ALARMA FALLO BATERIA TAMPON
    BEA
m002: NOP   0
    SET
    S    A    124.1             // AVISO REPOSICION BATERIA TAMPON
    BEA
```

6.2 Las DB´s.

6.2.1 Como trabajar con DB´s en S7.

En el lenguaje S7, el tratamiento de las variables contenidas dentro de una DB debe de realizarse preferentemente en simbólico. Esta es la mejor forma de acceder a una estructura irregular de variables como puede ser una DB. Para ello, en cuanto se haya creado la DB, se debe de simbolizar, para que uno pueda acceder a la misma en simbólico.

Un apunte interesante, es el hecho de que no se puede realizar una llamada a una variable de una DB que no existiera en el momento de abrir el bloque que la llama, ya sea una FB, FC o una OB. En otras palabras, supongamos que tenemos abierta la OB1 y queremos cargar la DB1.DBD4. Si acudimos a la DB1 y la creamos dicha variable, al volver a la OB1 no podremos llamarla en formato simbólico hasta que no cerremos el bloque y lo volvamos a abrir. Esto obliga al programador a estructurar correctamente sus bases de datos antes de poder acceder a las mismas.

6.2.2 Tipos de DB´s

Existen tres tipos distintos de bases de datos o DB´s:

- Normales: una agrupación de variables que pueden ser compuestas o simples en cualquier orden.
- Asociadas a una FB: La DB está compuesta por los parámetros de entrada y salida de la FB.
- Asociada a una UDT: La DB está compuesta por los valores procedentes de una base de datos que actúa como “plantilla”.

La generación de un bloque de datos normal únicamente debe guardar la regla de que para generar variable de tipo BOOL o BYTE, es conveniente que se encuentren consecutivas dentro de la DB, ya que la asignación de memoria dentro de la DB se realiza en formato de palabra. En la figura inferior se puede apreciar como perdemos memoria si no colocamos las variables booleanas seguidas.

Dirección	Nombre	Tipo	Valor inicial
0.0		STRUCT	
+0.0	bit_1	BOOL	FALSE
+2.0	temperatura	REAL	0.000000e+000
+6.0	bit_2	BOOL	FALSE
+8.0	consigna	INT	0
+10.0	humedad	BYTE	B#16#0
+11.0	bit_3	BOOL	FALSE
+12.0	presion	BYTE	B#16#0
=14.0		END_STRUCT	

Dirección	Nombre	Tipo	Valor inicial
0.0		STRUCT	
+0.0	bit_1	BOOL	FALSE
+0.1	bit_2	BOOL	FALSE
+0.2	bit_3	BOOL	FALSE
+2.0	temperatura	REAL	0.000000e+000
+6.0	consigna	INT	0
+8.0	presion	BYTE	B#16#0
+9.0	humedad	BYTE	B#16#0
=10.0		END_STRUCT	

Figura 19 . Según ordenemos los mismos datos dentro de la DB, pueden ocuparnos 14 bytes, o sólo 10. Es importante tener en cuenta la agrupación de datos por tamaño siempre que sea posible para optimizar memoria.

Con respecto a las DB´s asociadas a una FB, únicamente resaltar que no es necesarios crearlas desde el Administrador Simatic como un bloque de datos y asociarlas a la DB. Es mucho más sencillo y cómodo preocuparse únicamente de generar la FB correcta. Cuando esté finalizada, en lugar de realizar la llamada a la misma en online, acceder al bloque u OB que ha de llamarla por offline. En el instante que realicemos la llamada a la FB, y le asociemos la DB, el software detecta que no existe, y nos la genera automáticamente. Esto no es posible si la llamada la estamos intentando realizar desde online.

Las DB´s asociadas a una UDT tienen sus ventajas y sus inconvenientes. Por un lado, al crear una DB asociada a una UDT rellenamos automáticamente la base de datos utilizando la UDT como plantilla. Sin embargo, posee este método un inconveniente grave: si se realiza una modificación en la UDT posteriormente, ya sea añadiendo datos o sustrayéndolos, el cambio no solo no se refleja automáticamente en las DB´s asociadas, sino que se pierde el valor simbólico de cada una de las variables dentro de la misma. Es

necesario pues borrar todas las DB´s asociadas a la UDT y volver a generarlas. Por todo esto, el asociar DB´s a UDT´s se debe de realizar exclusivamente si se conoce de antemano que no se variará la UDT. Si existe esta posibilidad, es recomendable utilizar las UDT´s incrustadas dentro de la DB, método que sí permite actualizar las modificaciones posteriormente sin necesidad de borrar la base de datos.

6.2.3 Tipos de variables de una DB.

Como hemos visto, el manejo de DB´s se debe de realizar preferentemente en simbólico. Es por esto que antes de generar una nueva variable, el Step 7 nos reclama un nombre simbólico para la misma antes de poder continuar.

El siguiente parámetro importante en una variable de una DB es el tipo de variable. En Step 7 se agrupan por datos simples, aquellos que no están compuestos de otros, y datos compuestos, que son agrupaciones de datos simples en un determinado orden.

Datos simples.

TIPO DE DATOS	FORMATO	EJEMPLO
BOOL	TRUE/FALSE	TRUE
BYTE	B#16#	B#16#10
WORD	W#16#	W#16#324
DWORD	DW#16#	DW#16#35400
INT		2300
DINT	L#	L#35000
REAL	0e+0	1.000000e+000
S5TIME	S5T#	S5T#1S
TIME	T#	T#2D12H
DATE	D#	D#1999-10-20
TIME_OF_DAY	TOD#	TOD#10:35:1.0
CHAR	' '	'A'

Datos compuestos.

TIPO DE DATOS	FORMATO	EJEMPLO
DATE_AND_TIME	DT#90-1-1-0:0:0.0	DT#99-10-28-23:15:1.0
STRING	STRING[254]	'SIMATIC'
ARRAY	ARRAY[0..X,0..Y]	ARRAY[0..10,0..20]
STRUCT	-	-
UDT	UDTX	UDT1

+62.0	PRESION_TEMP	ARRAY[0..10,0..10]
*2.0		INT

Figura 20 . Ejemplo de un array de valor de presión con respecto a la temperatura.

+304.0	MOTOR_1	STRUCT	
+0.0	MARCHA	BOOL	FALSE
+0.1	FALLO1	BOOL	FALSE
+2.0	CONSIGNA	INT	0
+4.0	VELOCIDAD	REAL	0.000000e+000

Figura 21 . Ejemplo de una estructura de un motor, agrupando las variables relacionadas con el mismos.

+312.0	MOTOR_2	UDT1	

Figura 22 . Ejemplo de variable de tipo UDT. Este sistema de generación de datos presenta dos inconvenientes: por un lado no podemos saber las variables que componen MOTOR_1 estando en visualización de declaración de variables si no abrimos la UDT (también podemos cambiar en el menú a ver->datos. Esto se consigue presionando sobre la variable con el botón derecho y seleccionando *abrir bloque*. El segundo inconveniente es la necesidad de actualizar la variable si se modifica la UDT original. Para ello, una vez modificada la UDT, acudiremos a la DB, y presionando sobre la variable con el botón derecho, en el menú contextual de la variable seleccionaremos *actualizar declaraciones*.

6.2.4 Como abrir DB´s.

Existen dos formas de abrir DB´s en S7:

- Abrirla al principio de la subrutina mediante la instrucción AUF, o
- Abrirla en la misma línea de llamada de la variable (p. Ej. L DB1.DBW0).

La segunda manera es un poco más lenta de ejecutarse en el microprocesador del PLC, pero posee la ventaja de permitirnos trabajar con transferencias de variables entre varias DB´s de una manera rápida y sencilla.

El primer método, se basa en la función AUF, que nos permite abrir hasta dos tipos de DB´s a la vez:

- AUF DBx : abrir bloque de datos normal.
- AUF Dix : abrir bloque de datos de instancia.

Por este método no podremos tener nunca más de dos DB´s abiertas a la vez.

El segundo método sigue la siguiente estructura:

Nombre de la db.nombre de la variable

Ejemplos del mismos son:

- **Consulta de bit:** U DB4.DBX 0.0
- **Activar bit:** = DB4.DBX 0.1
- **Cargar un valor:** L DB4.DBW2

Se recomienda utilizar el segundo método, ya que es mucho más claro conocer en todo momento la procedencia de la variable que se está tratando.

6.2.5 Datos de una DB.

A veces puede ser interesante conocer información al respecto de la DB con la que estamos o vamos a tratar, ya sea para consultar datos o para almacenarlos. Uno de los problemas más comunes en las paradas de las máquinas es la pérdida de una o todas las DB´s. Si durante el programa el PLC no encuentra la base de datos con la que debe de trabajar, el sistema pasa a STOP. Para evitar este problema, se puede utilizar una comprobación en el arranque del equipo de la existencia de las DB´s con las que se va a trabajar, y su longitud adecuada. En caso contrario, el sistema puede generar una alarma, y : o generar las DB´s que falten, inicializándolas con unos valores prefijados, o no arrancar el sistema.

Para realizar todo esto lo primero que necesitamos es conocer la existencia de una determinada base de datos, y su longitud, para poder compararla con unos valores conocidos por nosotros.

De una manera sencilla, podemos obtener en todo momento la longitud de la DB abierta actualmente mediante la instrucción L DBLG, así como del número de DB abierto actualmente mediante L DBNO.

Pero para poder consultar la DB con estas instrucciones tenemos que haberla abierto anteriormente mediante AUF DB, y si no existe la base de datos el sistema interpretará esta instrucción como errónea.

Para evitar este problema de recurrencia, existe dentro del firmware de las CPU´s una SFC, concretamente la SFC 24 "TEST_DB", que nos facilitará las cosas. Con la SFC 24 ”TEST_DB” (test data block) se obtienen informaciones sobre un bloque de datos existente en la memoria interna de la CPU. La SFC determina para el DB seleccionado la cantidad de los bytes de datos y comprueba si el DB está protegido contra escritura.

Parámetros de la SFC24.

Parámetro	Declaración	Tipo de datos	Descripción
DB_NUMBER	INPUT	WORD	Número del DB a comprobar
RET_VAL	OUTPUT	INT	Código de error.
DB_LENGTH	OUTPUT	WORD	Cantidad de bytes que componen el DB.
WRITE_PROT	OUTPUT	BOOL	Si está protegido contra escritura. FALSE = no protegido. TRUE= protegido.

El código que podemos obtener como error en RET_VAL es el siguiente:

RET_VAL	Descripción
0000	No ha aparecido ningún error.
80A1	Error en el número de DB en DB_NUMBER: El parámetro proporcionado es: • 0, o • es mayor que el número DB máximo posible para la CPU aplicada.
80B1	El DB con el número indicado no existe en la CPU.
80B2	El DB fue creado con la palabra clave UNLINKED.

El error que nos interesa controlar será lógicamente si aparece un 80B1 en RET_VAL. Si llamamos a la SFC24 en la OB100 y consultamos ret_val ya estaremos en disposición de controlar la ausencia de la DB, o si le faltan variables, mediante la consulta del parámetro DB_LENGHT.

6.2.6 Crear DB´s por programa.

Supongamos que hemos detectado en el arranque de nuestro programa de PLC que no existe una base de datos con la que más adelante vamos a trabajar. ¿y ahora qué?. Lógicamente lo que deberemos de hacer es generarla de nuevo, inicializándola a unos valores prefijados. Para cumplir estos menesteres disponemos de la SFC 22 "CREAT_DB".

La SFC crea un bloque de datos asignando un número del margen indicado y con el tamaño que le indicamos. Para crear un DB con un número determinado, que es lo normal, es necesario asignar el mismo número a los límites superior e inferior del margen que le indicamos a la SFC. Los números de los DB's contenidos ya en el programa de usuario no se pueden volver a asignar. La longitud de los DB's debe ser un número par.

Parámetros de la SFC22.

Parámetro	Declaración	Tipo de datos	Descripción
LOW_LIMIT	INPUT	WORD	El valor límite inferior es el menor número del margen que se puede asignar al bloque de datos.
UP_LIMIT	INPUT	WORD	El valor límite inferior es el mayor número del margen que se puede asignar al bloque de datos.
COUNT	INPUT	WORD	Cantidad de bytes que debe tener la DB. El valor debe ser par.
RET_VAL	OUTPUT	WORD	Código de error.
DB_NUMBER	OUTPUT	WORD	Número de DB generada. Si existe un error en la generación el valor devuelto en el parámetro es 0.

El código que podemos obtener como error en RET_VAL es el siguiente:

RET_VAL	Descripción
0000	No ha aparecido ningún error.
8091	Se ha llamado a la SFC 22 varias veces seguidas.
8092	Está actuando la función de compresión de memoria.
80A1	Número de DB erróneo. Puede ser porque: • El número es 0

	• El número sobrepasa al número DB específico de la CPU • Límite inferior > límite superior
80A2	Error en la longitud del DB. Puede ser porque: • La longitud es 0 • La longitud fue definida como número impar • La longitud es superior a la admisible por la CPU
80B1	No hay número de DB disponible. Están todos ocupados en el rango indicado.
80B2	Espacio insuficiente en memoria. Se puede intentar comprimir la memoria.
80B3	Espacio de memoria contiguo insuficiente. Se puede intentar comprimir la memoria.

6.2.7 Borrar DB´s desde programa.

Lo visto anteriormente funciona, siempre y cuando no exista la DB, pero: ¿y si el problema reside en que hemos perdido una serie de variable, y no toda la DB?. En ese caso, el tamaño de la misma no será correcto, pero no podremos crearla, ya que existe. En este caso nos vemos obligados a borrarla, para posteriormente volver a generarla.

Para ayudarnos en estos menesteres de borrar bases de datos, disponemos de la SFC23.

Con la SFC 23 ”DEL_DB” (delete data block) se borra un bloque de datos existente en la memoria interna y, dado el caso, en la memoria de carga de la CPU. El DB a borrar no debe estar abierto en el nivel de ejecución actual ni tampoco en un nivel de ejecución de menor prioridad. Es decir, no debe estar consignado en uno de los dos registros DB (ni bloque de datos actual ni de instancia). De lo contrario, en la llamada a la SFC 23, la CPU cambia al estado operativo STOP.

Si el equipo posee FLASH-EPROM, ya sea porque es IFM, ya sea porque se la hemos insertado, y se encuentra grabada la DB en la misma, lógicamente no vamos a poder borrarla mediante esta instrucción del sistema.

Parámetros de la SFC23.

Parámetro	Declaración	Tipo de datos	Descripción
DB_NUMBER	INPUT	WORD	Número de DB a borrar.

RET_VAL	OUTPUT	INT	Código de error.

El código que podemos obtener como error en RET_VAL es el siguiente:

RET_VAL	Descripción
0000	No ha aparecido ningún error.
8091	En llamadas SFC 23 anidadas se superó la profundidad de anidado máxima de la CPU empleada.
8092	La función ”Borrar un DB” no puede realizarse de momento, porque: • la función ”Comprimir la memoria de usuario” está activa en el momento • la función ”Guardar programa de usuario” está activa en el momento.
80A1	Error en el parámetro de entrada DB_NUMBER: El parámetro actual elegido: • tiene el valor 0 • es mayor que el número DB máximo posible para la CPU aplicada.
80B1	El DB con el número indicado no existe en la CPU.
80B2	El DB con el número indicado fue creado con la palabra clave UNLINKED.
80B3	El DB se encuentra en la Flash card.

7

Miscelánea de S7

7.1 Proteger el programa de miradas indiscretas.

Existen dos circunstancias por las que podemos desear proteger nuestro código:

- instalaciones de las cuales somos únicos responsables, p. ej. propietarios de instalaciones o empresas que poseen la contrata exclusiva para el mantenimiento de la máquina: en estos casos se puede utilizar una protección global del programa de PLC, ya que nadie debe de entrar al mismo para realizar modificaciones salvo nosotros.
- instalaciones o máquinas en las cuales únicamente se realiza el programa, pero no el mantenimiento posterior de la misma en exclusividad. Suele ser el caso de fabricantes de maquinaria que venden sus máquinas a un cliente final, o ingenierías que realizan una instalación para un cliente que posee su propio equipo de mantenimiento. En este caso, la protección global genera un gran conflicto para el cliente final, ya que no puede realizar el mantenimiento de su instalación. Para estas situaciones es aconsejable realizar protecciones individuales de bloques, en los cuales se asienta la parte de conocimiento tecnológico que se desea preservar de la competencia.

Vamos a estudiar como realizar los dos tipos de protección, tanto de programa completo como de módulos individuales.

7.1.1 Protección global del programa

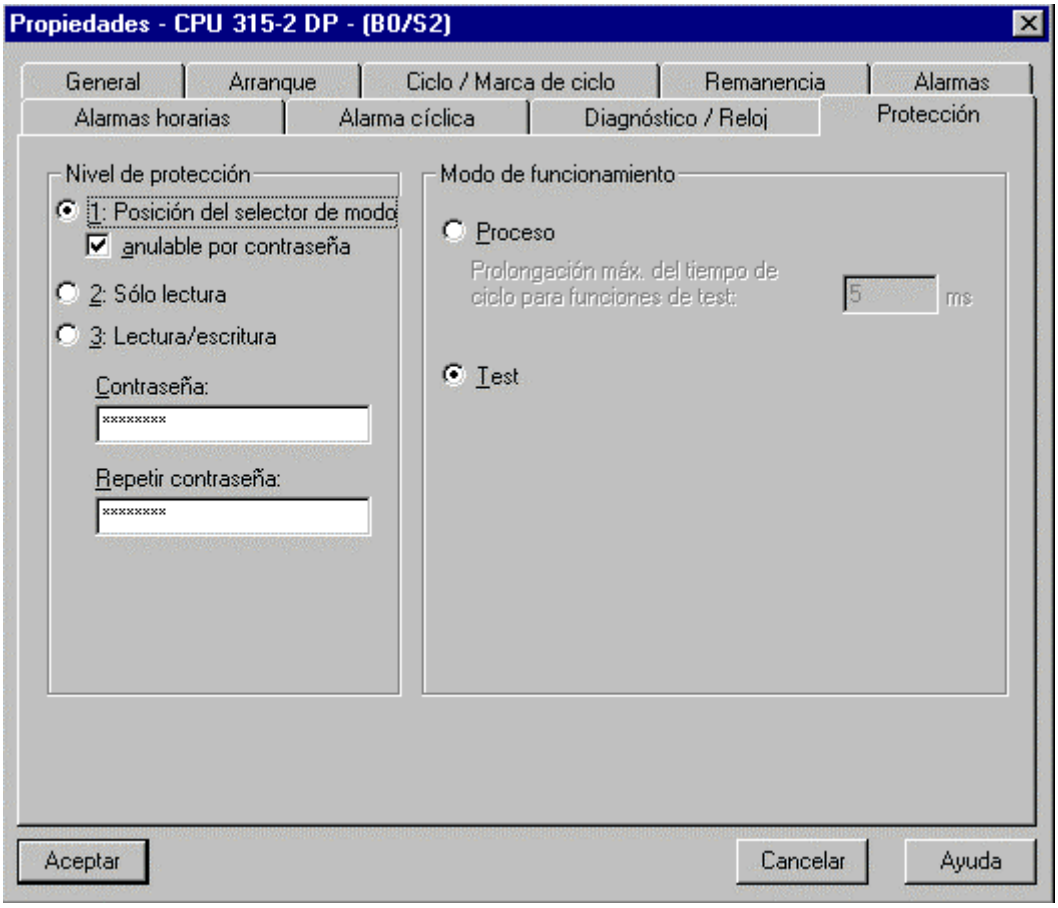
Supongamos que deseamos proteger todo nuestro programa. Dentro de este caso disponemos de dos posibilidades:

- **protección a nivel de lectura:** es el caso en el cual el temor de la empresa se basa en que operarios no cualificados puedan modificar el programa de PLC, cometiendo errores que perjudiquen al funcionamiento de la instalación. No existe temor en este caso a copias de programa por parte de la competencia para su posterior estudio de conocimientos.

- **protección a nivel de lectura / escritura:** en dicho caso la preocupación del programador se basa en que en su instalación se pueda realizar una copia del programa por parte de la competencia para posteriormente utilizar dichos conocimientos en otras instalaciones ahorrando el tiempo de desarrollo necesario.

En el hardware de Step 7, si seleccionamos las propiedades de la CPU, podemos acceder a la solapa *protección* en la que disponemos de las siguientes opciones:

- **Opción 1:** selector de modo anulable por contraseña. Esta primera opción es equivalente a no tener ninguna protección en el programa. En el caso de que seleccionemos la opción “anulable por contraseña”, cuando dispongamos el selector e la CPU en modo RUN, e intentemos transferir una modificación a la misma, se nos solicitará la palabra de password que hemos asignado en el campo contraseña. Esta posibilidad es útil para aquellos casos en los cuales se extrae la llave frontal del equipo dejándolo en modo RUN. El técnico de mantenimiento puede ir a la instalación, y sin necesidad de colocar la llave en el equipo realizarle las modificaciones oportunas.
- **Opción 2:** Solo lectura. En este modo de trabajo se nos va a permitir de manera ilimitada la visualización de todos los módulos, restringiéndosenos la modificación de los mismos, para la cual deberá de estar el selector en modo RUN-P. En principio nada impide que alguien con paciencia pueda copiar el programa a base de copiar / pegar creando un nuevo proyecto en el disco duro, por lo que la seguridad es relativa (solo para gente con escasos conocimientos informáticos).
- **Opción 3:** Lectura / escritura. En este modo de trabajo tanto la lectura como la escritura están protegidas por un password. Este modo únicamente puede ser observados o modificados los bloques mediante la introducción de una palabra de password. Este modo es bastante seguro, presentando únicamente el inconveniente de que imposibilita a un ajeno solventar paradas de instalaciones, ya que no se permite realizar *test-status* de módulo sin dicho password. No es recomendable para ingenierías, ya que genera conflictos con el cliente final a la larga.



7.1.2 Protección de bloques de programa.

La opción de proteger bloques individuales de programa es mucho más interesante, ya que permite realizar test-status, copias de seguridad del mismo por parte del cliente, e incluso imprimirlo en papel. Sin embargo, al proteger a nivel de visualización ciertos bloques, que lógicamente es donde se concentra la parte compleja de nuestro desarrollo, se impide que la competencia pueda alcanzar a conocer cómo hemos realizado dicha programación, pudiendo exclusivamente observar los resultados de la misma (lo que desde luego no es de ninguna ayuda, puedo asegurarlo...). Es decir, posee esta opción todas las ventajas y ningún inconveniente, por lo que se recomienda su utilización frente a las vistas anteriormente.

Proteger bloques es una tarea relativamente sencilla, pero que debe de hacerse siguiendo los pasos indicados, ya que una vez protegido un bloque, no existe ninguna manera de desprotegerlo (no existe la famosa palabra de password). ¿El truco donde está?. Es la pregunta que rápidamente asalta al programador. Todo se basa en crearse el código fuente del o los módulos a proteger. Pero si se pierde el código fuente, ya no existe manera de desprotegerlo, de ahí recalcar la importancia de seguir los pasos indicados de manera correcta.

Lo primero que debemos de tener, lógicamente, serán subrutinas con código a proteger. Realizar este método en la OB1 es absurdo, ya que estaríamos en el caso de una protección total de programa, visto anteriormente. Por lo tanto, suponemos que tenemos dos FC's que son llamadas desde la OB1 y deseamos protegerlas. Editamos la OB1 (o cualquier bloque de programa), y el menú *Archivo->Generar fuente* podremos seleccionar aquellos bloques que deseamos convertir a código fuente.

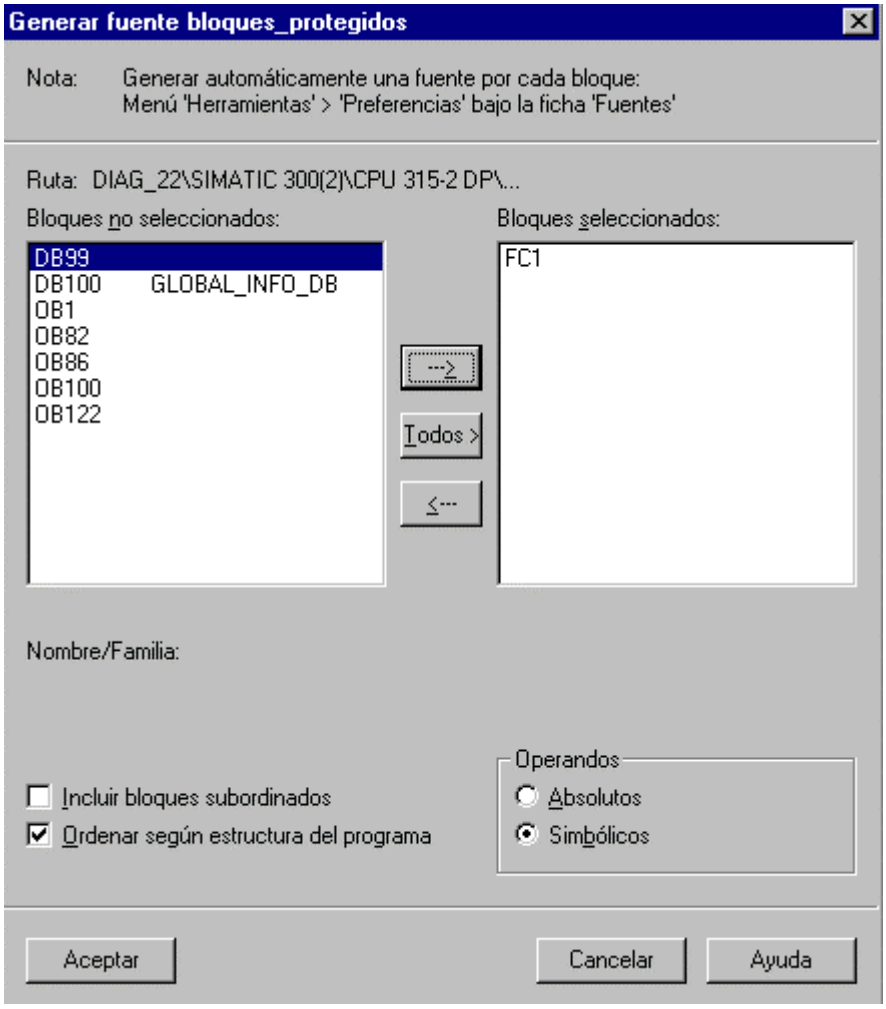


Figura 24. Selección de bloques desde el editor KOP-FUP-AWL para su protección posterior.

Las opciones de que disponemos son:

- **Incluir bloques subordinados:** genera la fuente de todas las subrutinas que sean llamadas desde los bloques que seleccionemos, aunque no se hayan seleccionado por nuestra parte.
- **Ordenar según estructura del programa:** al generar la fuente AWL, realmente estamos juntando todos los bloques seleccionados en un solo archivo. Aquí indicamos

que se ordenen por el orden con el que serán llamados desde la OB1 de programa. Si no seleccionamos esta opción se ordenarán dentro del archivo fuente según se encuentren en la ventana de la derecha seleccionados.

- **Convertir por operando absoluto / relativo:** indica si la fuente se debe de generar teniendo en cuenta el operando absoluto de la variable, o el simbólico. Normalmente nos interesa el absoluto, salvo en el caso de que deseemos realizar una sustitución global de una variable a través del simbólico, en lugar del absoluto, que es lo común.

Una vez convertidas las funciones a fuente AWL, vamos al administrador de Step 7, y en nuestro programa de S7, además del subdirectorio *bloques* y *simbólico*, disponemos de la opción *fuente*, en la que editamos nuestro archivo fuente AWL, teniendo el siguiente aspecto:

```
FUNCTION FC 1 : VOID
VERSION : 0.1

BEGIN
NETWORK
TITLE =

    R    M     3.0;
    S    M     3.1;
END_FUNCTION
```

Introducimos la sentencia *know_how_protect* debajo de la cabecera de declaración del código fuente AWL:

```
FUNCTION FC 1 : VOID
VERSION : 0.1
know_how_protect

BEGIN
NETWORK
TITLE =

    R    M     3.0;
    S    M     3.1;
END_FUNCTION
```

Volvemos a compilar el código en archivo->compilar. Con esto ya tenemos protegido el/los bloques seleccionados. Para desprotegerlos, deberemos volver al código fuente, y colocar dos barras delante de la palabra *know_how_protect*, volviendo a compilar de nuevo.

```
FUNCTION FC 1 : VOID
VERSION : 0.1
```

```
//know_how_protect

BEGIN
NETWORK
TITLE =

    R    M    3.0;
    S    M    3.1;
END_FUNCTION
```

7.2 Conversión de programas de S5 a S7.

El proceso de conversión de programas de S5 a S7 es gracias a la herramienta que se nos proporciona bastante sencillo, pero es conveniente conocer sus limitaciones para poder "reajustar" convenientemente el programa de S7 una vez se ha convertido el de S5.

Comenzaremos estudiando el proceso de conversión de programas para pasar posteriormente a ver las limitaciones de dicha conversión.

7.2.1 ¿Cómo se pasa un programa de S5 a S7?

Partimos de la base de que ya tenemos un programa de S5 completamente terminado y operativo. El primer paso a realizar es realizar referencias cruzadas dentro de S5, para tener actualizados todos los vínculos de orden de programa.

Si no realizamos esta acción, la utilidad conversora desconocerá el orden de las llamadas de funciones dentro del programa, realizando una conversión lineal del mismo (la primera subrutina que encuentra), lo que conduce a errores en la conversión.

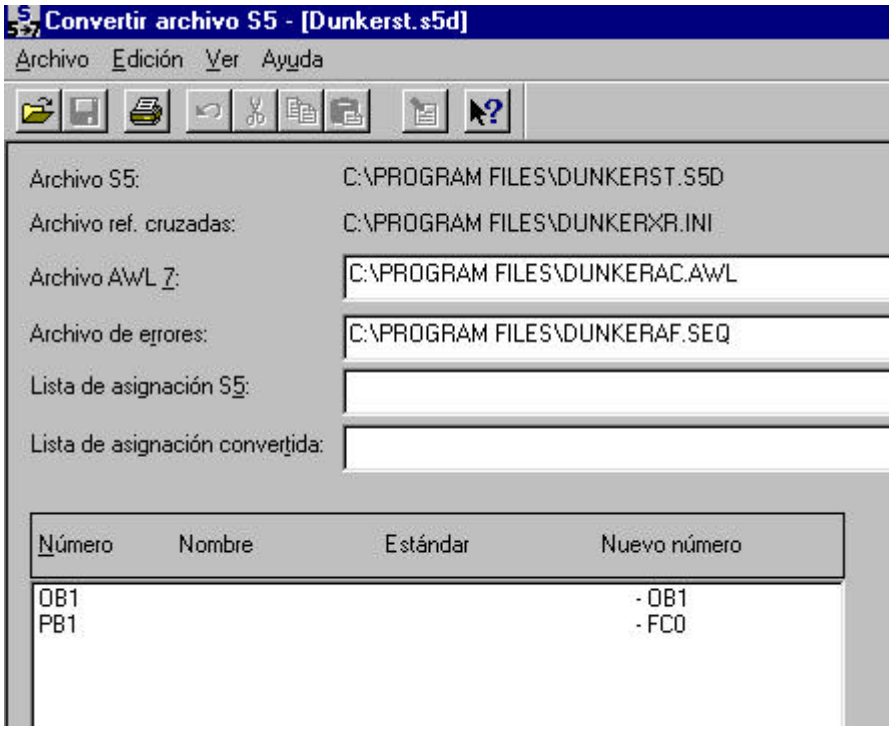


Figura 25. Selección del fichero de S5, y visualización de la conversión de los bloques de S5 en los de S7.Obsérvese como no se convierte el PB1 por el FC1, sino por el FC0. La organización de los bloques cambia completamente una vez se ha convertido el programa.

Una vez realizadas las referencias cruzadas, utilizamos la utilidad de conversión, programa independiente que se encuentra en *Inicio->Simatic->Step 7*.

La compilación, si se ha finalizado sin errores, debe de generarnos un archivo en el mismo subdirectorio donde se encontraba el programa de S5, y de nombre *nombre_proyectoac.awl*

Cerramos el conversor, entramos en Step 7, y en *Programa S7->Fuentes*, de su menú contextual seleccionamos la opción *Insertar Nuevo objeto->fuente externa*. Una vez hemos seleccionado el archivo *fuente AWL*, lo compilamos, generándosenos dentro de bloques de nuestro proyecto los correspondientes bloques, pero ya convertidos a S7, o casi...

7.2.2 Limitaciones en la conversión de S5 a S7.

Evidentemente todo no se puede convertir de manera automática. Veamos algunos de los errores más comunes que pueden aparecer en una conversión:

- **OB's:** en S7, los OB's no se pueden llamar desde el programa de usuario. Los OB's de funciones especiales, al igual que las instrucciones especiales como p. ej. *E DB*, han sido sustituidas por nuevas instrucciones o por SFC's. Si se llama p. ej. a la OB31 desde la OB1, deberá de corregirse a mano esta acción (que tal vez con la CPU nueva ni sea necesaria), ya que el programa no conoce la forma de realizar estas sustituciones.
- **DIRECCIONAMIENTO:** en S7 se direccionan todas las áreas de memoria (E, A, M, D, etc.) byte a byte. Es decir, *L DW 2* en S5 viene a ser *L DBW 4* en S7.
- **NUMERO DE ACUs:** Las CPU S7 disponen igual que las CPU S5 de 2 ó 4 acumuladores. En caso de convertir un programa de una CPU S5 con 2 acumuladores (115U) para una CPU S7 con 4 acumuladores, es preciso insertar la instrucción ENT antes de cada operación aritmética (+,-,*,/), siempre que tenga que seguir utilizando el contenido del ACU 2.//
- **ANCHO DE ACUs:** Los acumuladores de todas las CPU S7 tienen 32 bits de ancho. Al usar en S7 instrucciones de coma fija, en caso de desbordamiento, la palabra alta del ACU1 permanece inalterada (135U : signo); no se convierte automáticamente a formato de 32 bits (155U).
- **LIR, TIR:** En S7 no existen instrucciones de direccionamiento absoluto. Algunas se sustituyen por nuevas instrucciones (p. ej. acceso a palabras de datos > 255), algunas se sustituyen por funciones de sistema (SFC). Lo mismo rige para los registros de inicio y longitud de DB's así como para el registro base.
- **AREAS DE DATOS BS,BT,BA,BB,Q:** Para usar estas áreas de datos es preciso intervenir manualmente. Soluciones posibles :

  - usar áreas de marcas ampliadas
  - definir bloques de datos más extensos
  - usar funciones de sistema S7
- **INSTRUCCIONES DE STOP:** En S7 sólo existe un instrucción de STOP que se ejecuta llamando a una SFC. Corresponde a la instrucción STS de S5.

Aquellas conversiones manuales que requieran una gran cantidad de cambios repetitivos se pueden dejar a la utilidad de macros del conversor de S5 a S7, que permite sustituir una línea de S5 por una cantidad de líneas de S7.

En el ejemplo hemos sustituido la instrucción de S5 L EW5 (que no funcionaba) por la instrucción L EW4 y un comentario al respecto.

```
$MAKRO:L EW5
L EW4
// comentario del a modificación
$ENDMAKRO
```

7.3 Marcas de ciclo.

Las CPU´s de S7 disponen de la posibilidad de configurar un byte de marcas de tal manera que cada uno de los bits del mismo varíe con una frecuencia determinada. A estos bits que componen este byte que se asigna para esta función se les denomina marcas de ciclo.

La modificación del estado binario de los bits se realiza periódicamente con un ciclo de trabajo de 1:1. Las marcas de ciclo se pueden utilizar en el programa de usuario, por ejemplo, para controlar avisadores luminosos con luz intermitente o para iniciar procesos que se repitan periódicamente (como la captación de un valor real).

Las marcas de ciclo corren de forma asíncrona al ciclo de la CPU, es decir, que en los ciclos largos puede cambiar varias veces el estado de la marca de ciclo.

Veamos la duración del periodo encendido / apagado de cada uno de los bits del byte de marcas de ciclo, así como su frecuencia resultante.

Bit	7	6	5	4	3	2	1	0
Duración del período (s):	2	1,6	1	0,8	0,5	0,4	0,2	0,1
Frecuencia (Hz):	0,5	0,625	1	1,25	2	2,5	5	10

Esta frecuencia es fija, no pudiéndose modificar ni siquiera por programa. Si se desea una frecuencia que no es múltiplo de las del byte de marcas es necesario utilizar el generador asíncrono visto anteriormente. Si por el contrario sí es múltiplo de alguna de las frecuencias, basta con realizar un divisor de frecuencia por programa.

7.4 Remanencia de variables.

7.4.1 Remanente o no remanente, esa es la cuestión.

Una variable remanente es aquella que pese a que pase el PLC a stop, cuando vuelve a arrancar en su paso a RUN, no pierde su valor anterior (no se inicializa a 0). Lógicamente, variables no remanentes son aquellas que al arrancar el PLC automáticamente se inicializan a 0.

La pregunta es, ¿qué variables conviene hacer o dejar remanentes, y cuales no?. La función que van a desempeñar esas variables, ya sean temporizadores, contadores, marcas o DB's nos responderá a la cuestión. En la mayoría de las ocasiones las variables remanentes son peligrosas, sobre todo en producciones en las cuales una interrupción genera un perjuicio considerable. El porqué a esta afirmación es el siguiente: supongamos una instalación de transporte de cajas. En un punto concreto, si no se detecta pieza (determinado por una marca), la pieza siguiente puede salir. Si por un error de programa o por rotura de un elemento que lo falsea se queda enclavada dicha marca, las piezas no pararán de salir, golpeándose unas con otras. Si paramos el autómata y lo volvemos a arrancar, continuarán saliendo piezas, ya que continua remanente dicha variable.

Bien, pues no gastemos marcas remanentes, ¿no?. Supongamos que las cajas, después de salir del alimentador, son transportadas para ser recogidas por un robot. Desde que salen se activa una variable de transición que se resetea en el momento que el robot coge la pieza. Si en una de esas transiciones nos quedamos sin corriente en el cuadro eléctrico, al retornar, se habrá borrado la variable que indicaba la transición de la pieza, arrancarán los rodillos de transporte y el brazo "ignorará" todas las piezas que se encontraban sobre la cinta en el momento del corte eléctrico.

Entonces, ¿gastamos o no gastamos variables remanentes?. Como le gusta decir a un compañero mío, si y no. Lo que de verdad es muy importante, es darse cuenta de que la selección de variables remanentes o no remanentes no se realiza de manera individual. Uno desgraciadamente no puede decir esta variable sí, esta no. Por lo tanto, la planificación de las acciones que van a realizar las variables en nuestra instalación es un detalle importantísimo, y que en la mayoría de las ocasiones el programador únicamente se plantea una vez ha repetido la variable por todo el programa, teniendo que redireccionarla a otra que pueda definir como remanente.

7.4.2 Como determinar la remanencia en S7.

Para poder configurar las zonas remanentes o áreas de memoria dentro del PLC con dicha funcionalidad deberemos seleccionar nuestra CPU S/ en Hardware de Step 7.

Las zonas remanentes en un equipo por defecto son:

- Los 16 primeros bytes de marcas, del MB0 al MB15.
- Ningún temporizador remanente. Nota: Para la CPU 312 IFM no se puede parametrizar el temporizador S7 como área remanente.
- Los 8 primeros contadores, de Z0 al Z7.
- Ninguna DB.

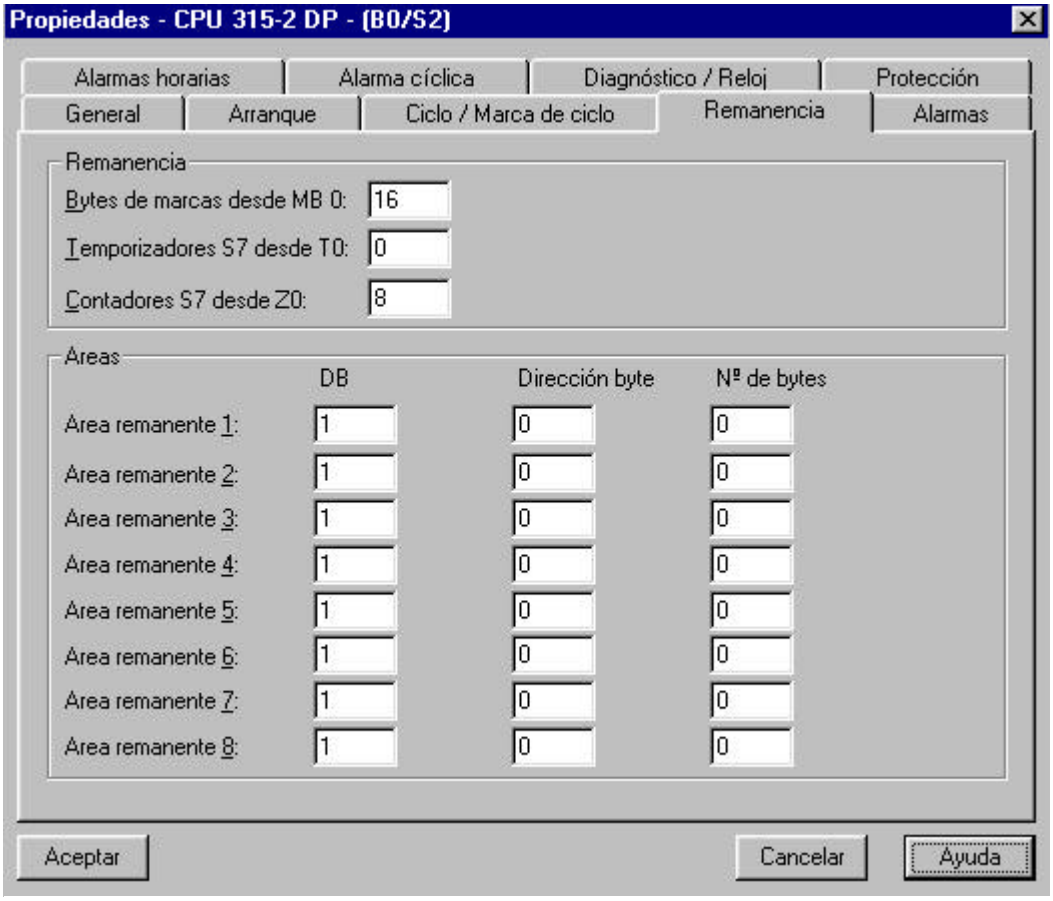


Figura 26. Selección de las áreas remanentes de una CPU desde Hardware de Step 7.

Es importante destacar el concepto de área remanente que se indica en esta solapa. Evidentemente, las DB´s son remanentes, ya que están respaldadas por la pila tampón. En caso contrario, cada vez que se parase una instalación habría que volver a introducir todos los datos de las DB´s. El concepto de esta ventana es indicar qué datos de qué DB´s, si no disponemos de pila tampón que respalde nuestro programa, no deben de inicializarse, sino mantenerse en la memoria interna del equipo. De esto se desprende que disponemos en la

CPU de una Flash-Eprom para almacenar el programa, ya que en caso contrario no solo perdería sin pila el valor actual de la DB, sino la DB y el resto de programa.

8

Trabajar con analógicas

8.1 Cómo funcionan las analógicas.

Las tarjetas analógicas de S7 convierten:

- Para la lectura de entradas analógicas, un valor analógico procedente de un sensor o transductor en un valor digital de 16 bits que se almacena en la periferia del S7 (no confundir estos 16 bits con la resolución de la entrada analógica). Esta resolución oscilará según los casos entre 11 bits mas signo y 8 bits.
- Para las salidas analógicas, un valor digital de 16 bits de la periferia de salidas en una señal analógica mediante un conversor digital-analógico.

Existen dos parámetros que determinan una entrada o una salida analógica:

- El tipo de sonda (en el caso de entradas), o el tipo de actuador (en el caso de salidas), a conectar (4-20 mA, 0-10 V, etc...)
- La resolución a alcanzar en la lectura o escritura. Cuanta mayor sea la resolución, mayor será la exactitud de la lectura y menor el error entre el valor real y el almacenado en el autómata. Por contra, también será mayor el tiempo de conversión analógico / digital y por lo tanto las variaciones en el proceso tardarán más en reflejarse en el PLC.

8.2 Tipos de señales analógicas.

Las tarjetas analógicas del S7 soportan una gran cantidad de tipos de medidas, configurándose desde *Hardware* de Step 7 a través de los adaptadores de margen laterales de la tarjeta (para más información, ver manual “Datos de los módulos S7”).

Sin embargo, es muy importante tener en cuenta que las entradas analógicas de los S7 300 se agrupan en grupos de 2 canales, por lo que ambos deben de estar configurados de la misma manera, ya sea para 0-10 V, para 4-20 mA, etc... Por lo tanto, no se puede mezclar en el canal 0 una señal de 0-10 V, p . ej., y en el canal 1 una de 4-20 mA. Deberá de dejarse vacío dicho canal si no se dispone de ninguna sonda de 0-10 V, y cablear la de 4-20 mA en el siguiente grupo libre.

Vamos a realizar un repaso a las conexiones más usuales con S7 de sensores de campo.

8.2.1 Medidas de tensión.

La más utilizada en la industria es 0-10 V. El Simatic S7 300 posee el rango +/-10 V, que es el adecuado para esta lectura. Como la sonda no va a dar tensión negativa, se va a comportar de manera unipolar, por lo siempre vamos a tener un valor de 0 al valor máximo que hayamos seleccionado de resolución.

El inconveniente de este tipo de lectura es que al ser una tensión, las distancias sin atenuación de la señal debido a caídas de tensión en el cable son relativamente cortas, por lo que la sonda debe estar cerca del cuadro eléctrico donde se encuentre el módulo analógico.

Los límites de lectura son:

Lectura +/-10 V:

Estado de lectura	Tensión en la entrada anal.	Valor de la entrada anal.
Desbordamiento positivo	>= 11,759	32.767
Rebase positivo	De 11,7589 a 10,0004	De 32511 a 27649
Valor nominal	De 10,0 a –10,0	De 27648 a –27648
Rebase negativo	De -10,0004 a -11,759	De –27649 a –32512
Desbordamiento negativo	<= -11,76	-32.768

8.2.2 Medidas de intensidad.

Dentro de las medidas de intensidad se suelen gastar principalmente dos tipos: 0-20 mA y 4-20 mA. Este último tipo es el más utilizado en la lectura analógica, ya que permite grandes distancias al ser la lectura por corriente, y a la vez es fácil reconocer la rotura del hilo, ya que por debajo de 4 mA es que no llega lectura del elemento.

Lectura 0 a 20 mA:

Estado de lectura	Corriente (mA)	Valor de la entrada anal.
Desbordamiento positivo	>= 23,516	32.767
Rebase positivo	De 23,515 a 20,0007	De 32511 a 27649
Valor nominal	De 20 a 0	De 27648 a 0
Rebase negativo	De –0,0007 a –3,5185	De -1 a –4864
Desbordamiento negativo	<= -3,5193	-32.768

Lectura 4 a 20 mA:

Estado de lectura	Corriente (mA)	Valor de la entrada anal.
Desbordamiento positivo	>= 22,815	32.767
Rebase positivo	De 22,810 a 20,0005	De 32511 a 27649
Valor nominal	De 20 a 4	De 27648 a 0
Rebase negativo	De 3,9995 a 1,1852	De -1 a –4864
Desbordamiento negativo	<= 1,1845	-32.768

8.2.3 Medidas PT100.

Las sondas PT100 se utilizan para la medida de la temperatura en procesos que oscilen entre 850ºC y –200ºC. Una PT100 es una termoresistencia que varía su resistividad en función de la temperatura que exista en contacto con la misma. A través de dos hilos (1 canal del módulo de entradas analógicas), se hace circular por la PT100 una corriente constante. Otros dos hilos toman la medida de la resistencia en los extremos de la sonda, con lo que se obtiene la variación de resistencia, y al ser conocida su linealidad con respecto a la temperatura en los márgenes anteriormente citados, se obtiene ésta.

Lectura PT100:

Estado de lectura	Temperatura	Valor de la entrada anal.
Desbordamiento positivo	>= 1000,1	32.767
Rebase positivo	De 1000 a 850,1	De 1000 a 850,1
Valor nominal	De 850 a –200	De 8500 a –2000
Rebase negativo	De –200,1 a -243	De –2000 a –2430

Desbordamiento negativo	<= -243,1	-32.768

Observar que lógicamente una sonda PT100 consumirá dos canales de entradas analógicas: 1 para la lectura en extremos de la sonda, y otro para la corriente constante que se suministra a la misma. Si la lectura se realizara por el mismo cable con el que se alimenta la PT100, las fluctuaciones de la resistencia del cable de cobre que une el PLC a la sonda con respecto a la temperatura variarían la magnitud de lectura. Por lo tanto, en una tarjeta de 8 entradas analógicas sólo se podrán conectar 4 sondas PT100.

8.3 Conexión de entradas analógicas.

Existen dos partes en una lectura analógica: la parte correspondiente al módulo de entradas analógicas del PLC, y la correspondiente al propio transductor o aparato de campo que genera la magnitud.

Desde el punto de vista del módulo del PLC puede ser:

- **Entrada analógica activa:** es la que se encarga de dar la energía a la conexión entre la tarjeta y la sonda, ya sea para que exista tensión (0-10 V) o corriente (4-20 mA). Se denomina a estas configuraciones “dos hilos”.
- **Entrada analógica pasiva:** es la que no genera ni tensión ni corriente en sus extremos, limitándose a recoger la medida esperada y convertirla a un valor digital. Se denomina a estas configuraciones “a cuatro hilos”.

Desde el punto de vista de la sonda, existen también dos tipos:

- **Sensores activos:** estos sensores poseen además de los dos hilos para la conexión con el PLC, dos adicionales para la alimentación externa del sensor. Transmiten ellos la energía al cable que une el PLC con la sonda.
- **Sensores pasivos:** sensores que sólo poseen dos hilos y que se limitan a generar la señal.

Las tarjetas de entradas analógicas del S7 300 pueden comportarse como activas o pasivas (dar corriente o no), por lo que sólo es necesario tener en cuenta que no se pueden mezclar configuraciones activas y pasivas en un mismo grupo de entradas. Ejemplo: si el canal 0 se configura pasivo, el canal 1 también es pasivo, por lo que no se puede configurar en él una sonda que no sea activa (que no disponga de alimentación externa).

Veamos los cableados posibles de sondas.

8.3.1 Cableado de sondas 0-10V pasiva con tarjetas de EA +/-10V activas.

Para leer una sonda de 0-10 V pasiva se cablea como muestra la figura:

En *Hardware*, se selecciona en tipo de medición U, y en margen +/-10V. El transductor lateral de la tarjeta para este grupo de canales debe de colocarse en la posición B.

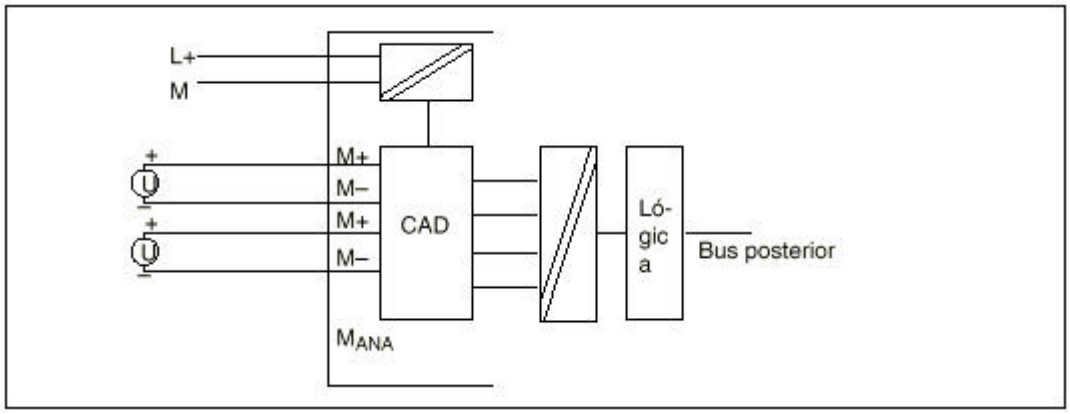


Figura 27 . Transmisor de tensión alimentado desde la tarjeta.

8.3.2 Cableado de sondas 4-20 mA pasivas con tarjetas 4-20 mA activas.

Las sondas 4-20 mA pasivas se cablean como muestra la figura:

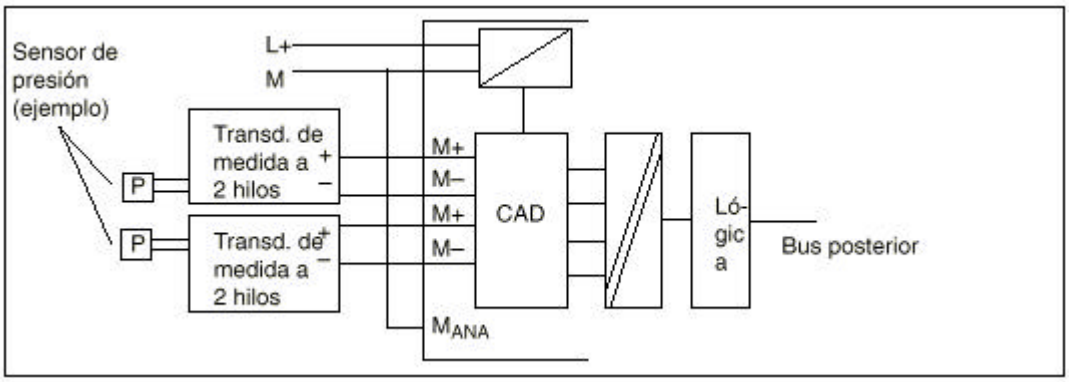


Figura 28 . Transductor de corriente a 2 hilos (pasivo) alimentado desde la tarjeta.

En *Hardware*, se selecciona en tipo de medición Intensidad (transductor de medida a 2 hilos), y el margen es fijo ya a 4-20 mA. El transductor lateral de la tarjeta para este grupo de canales debe de colocarse en la posición D. La corriente en este caso la suministrará el módulo de entradas analógicas.

8.3.3 Cableado de sondas 0-20 o 4-20 mA activas:

Las sondas 0-20 o 4-20 mA activas se cablean como muestra la figura:

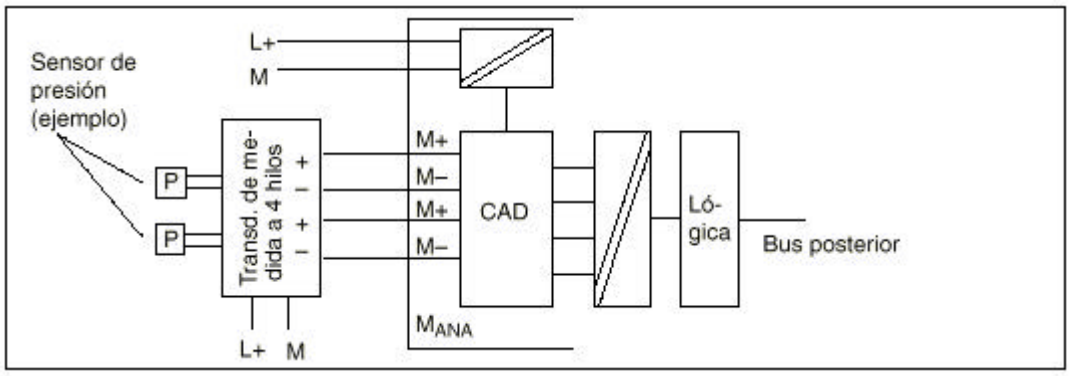


Figura 29 . Transductor de corriente activo (4 hilos). La tarjeta se comporta de manera pasiva.

En *Hardware*, se selecciona en tipo de medición Intensidad (transductor de medida a 4 hilos), y en margen seleccionamos si deseamos 0-20 o 4-20 mA. El transductor lateral de la tarjeta para este grupo de canales debe de colocarse en la posición C.

8.3.4 Cableado de PT100 o Ni100:

Las sondas PT100 o Ni100 se cablean como muestra la figura:

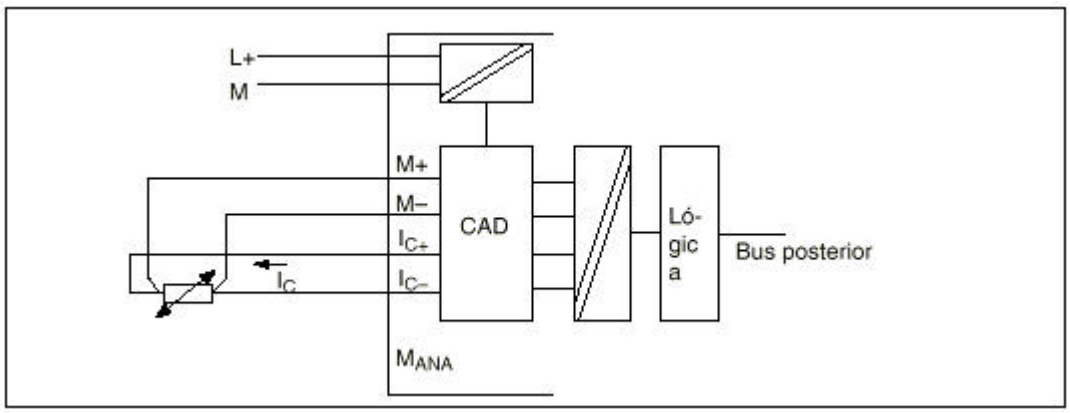


Figura 30 . Cableado de una termoresistencia. Observar como se pasa una corriente a través de Ic+ e Ic-, mientras que obtiene la caída de tensión entre M+ y M-.

En *Hardware*, se selecciona en tipo de medición Termoresistencia (lineal 4 hilos), y en margen seleccionamos si deseamos PT100 o Ni100. El transductor lateral de la tarjeta para este grupo de canales debe de colocarse en la posición A.

8.3.5 Cableado de sondas 0-10 V pasivas con tarjetas pasivas.

A veces puede ocurrir que solo dispongamos de un canal de entradas en nuestra instalación, y el mismo sea pasivo, al igual que la sonda que deseamos conectar. En estos casos, no nos queda más remedio que realizar una alimentación externa del canal, como muestra la figura:

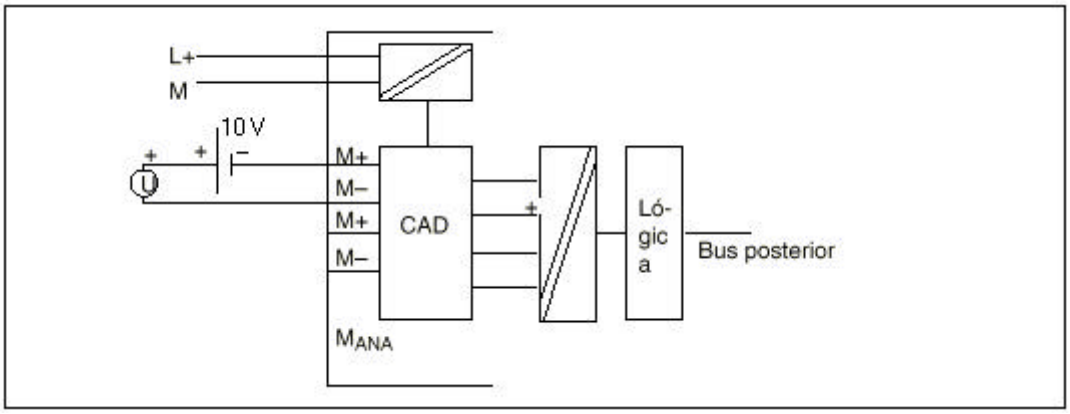


Figura 31 . Para conectar una sonda pasiva a una tarjeta pasiva seriamos una fuente de 10 V (generalmente realizamos un divisor de tensión con respecto a una fuente de tensión de 24V DC).

8.3.6 Cableado de sondas 4-20 mA pasivas con tarjetas 4-20 mA pasivas.

Si nos encontramos en la circunstancia de tener que cablear una sonda 4-20 mA a una tarjeta pasiva, deberemos de utilizar una fuente de tensión externa de 24 V que nos va a generar los mA necesarios para la lectura del transductor.

El cableado necesario es el siguiente:

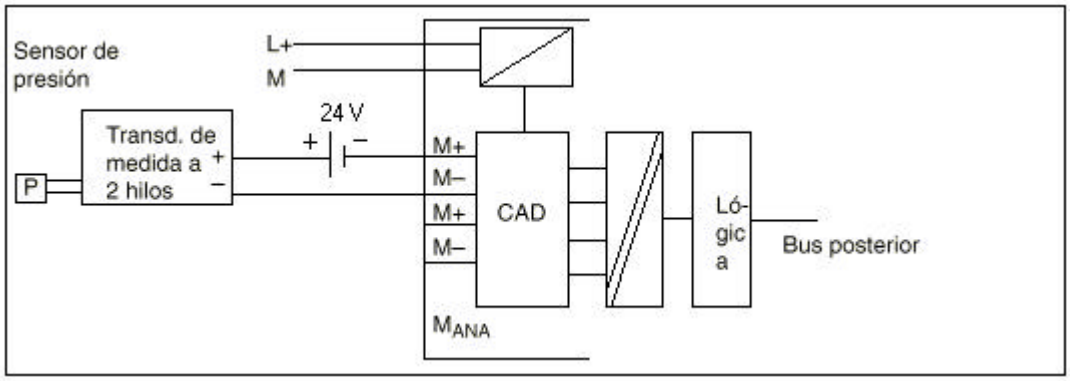


Figura 32 . Necesitamos una fuente que nos aporte la energía al transductor para que obtengamos la señal de 4 a 20 mA.

8.4 Tipos de tarjetas analógicas de S7 300.

8.4.1 Tarjeta de 8 EA SM331.

La tarjeta dispone de 8 entradas analógicas, agrupadas en 4 canales. Los canales no utilizados en la tarjeta deben de cortocircuitarse y unirse con la patilla Mana. Si un grupo de 2 entradas analógicas no se está utilizando, es interesante seleccionar en tipo de canal: desactivado, con el objeto de que la tarjeta, que realiza una lectura consecutiva de los canales para realizar la conversión analógica-digital, no pierda tiempo en este grupo y se acelere el tiempo de conversión.

Sin embargo, esta desactivación afecta a los dos canales del grupo. Si uno de ellos sí se está gastando y el otro no, se deberá proceder de la siguiente forma, según el caso:

- Sondas de 0-10 V: cortocircuitar el canal que no se utiliza.
- Sondas de 4-20 mA a 2 hilos: se puede o
- Dejar la entrada no utilizada abierta y no habilitar la función de diagnóstico para el grupo de canales.
- Si se desea habilitar la alarma de diagnóstico para el grupo de canales es necesario conectar una resistencia de 3,3 KΩ en la entrada libre.
- Sondas de 4-20 mA a 4 hilos: se deberá poner en serie con la entrada utilizada de su grupo de canales.

Se debe unir Mana con la patilla M de la tarjeta, que estará referenciada a la tierra del cuadro, para evitar diferencias de tensión en el circuito de conversión que falseen la señal analógica. Si no se va a utilizar la compensación externa, es recomendable cortocircuitarla (bornes COMP+ y COMP-). Si se utiliza sensores 0-10 V es necesario unir el borne Mana con el borne M- de la entrada analógica. Sin embargo, cuando se conecten transductores 4-20 mA a 2 hilos o termoresistencias PT100 no debe de unirse M- con Mana.

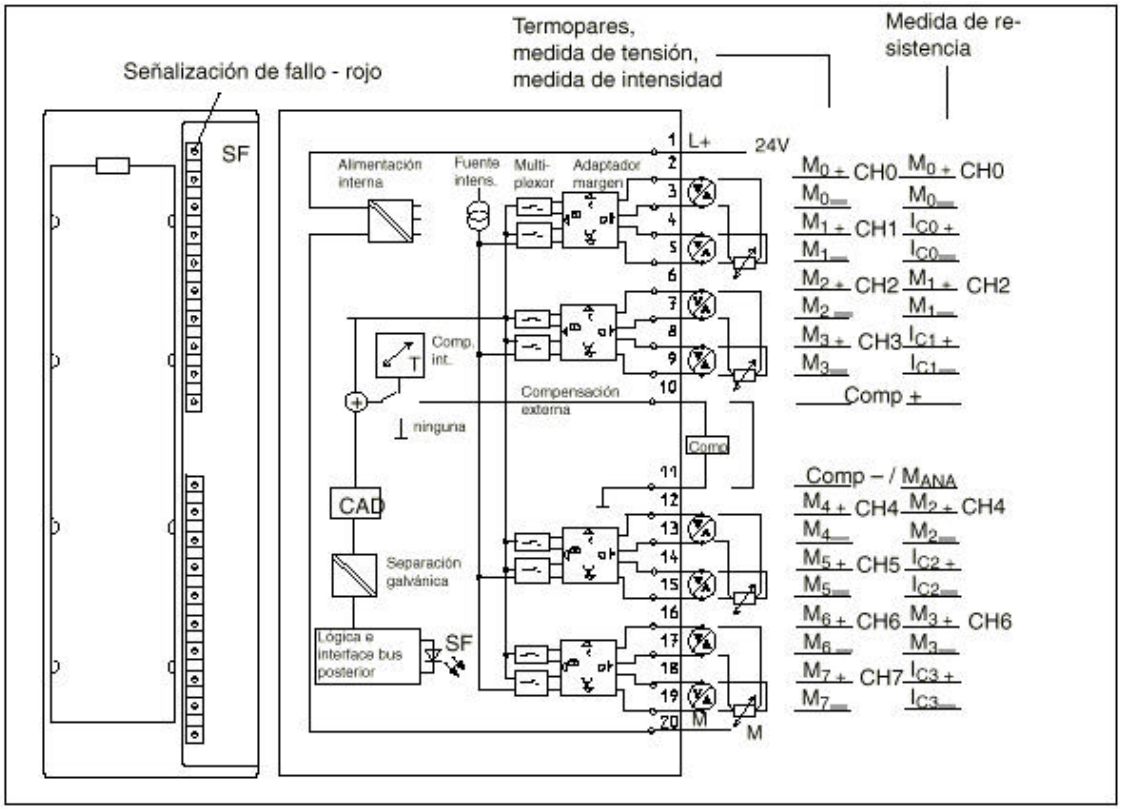


Figura 33 . Esquema del módulo de 8 entradas analógicas SM331.

El “periodo de integración” es el tiempo que tarda el valor en ser procesado y convertido a un valor digital interno. Este periodo es parametrizable en la opción *periodo de integración*, que aparece presionando sobre *supresión de frecuencias perturbadoras*, dentro de las propiedades del módulo desde hardware. Los rangos posibles son:

Resolución (bits)	Período de integración (ms)
9+signo	2,5
12+signo	16
12+signo	20
14+signo	100

8.4.2 Tarjeta de 4 SA SM332.

Los canales de salida no utilizados se deben de desactivar desde Step 7, y además dejar abiertos.

La resolución máxima en dicha tarjeta de salidas analógicas es de 12 bits. Las salidas pueden ser de tensión o de intensidad. Las salidas poseen vigilancia contra cortocircuito en salidas a tensión. La vigilancia por rotura de hilo solo se realiza en las salidas configuradas por corriente.

El cableado de las salidas puede realizarse a 2 o 4 hilos, dependiendo de la exactitud que deseemos en la misma.

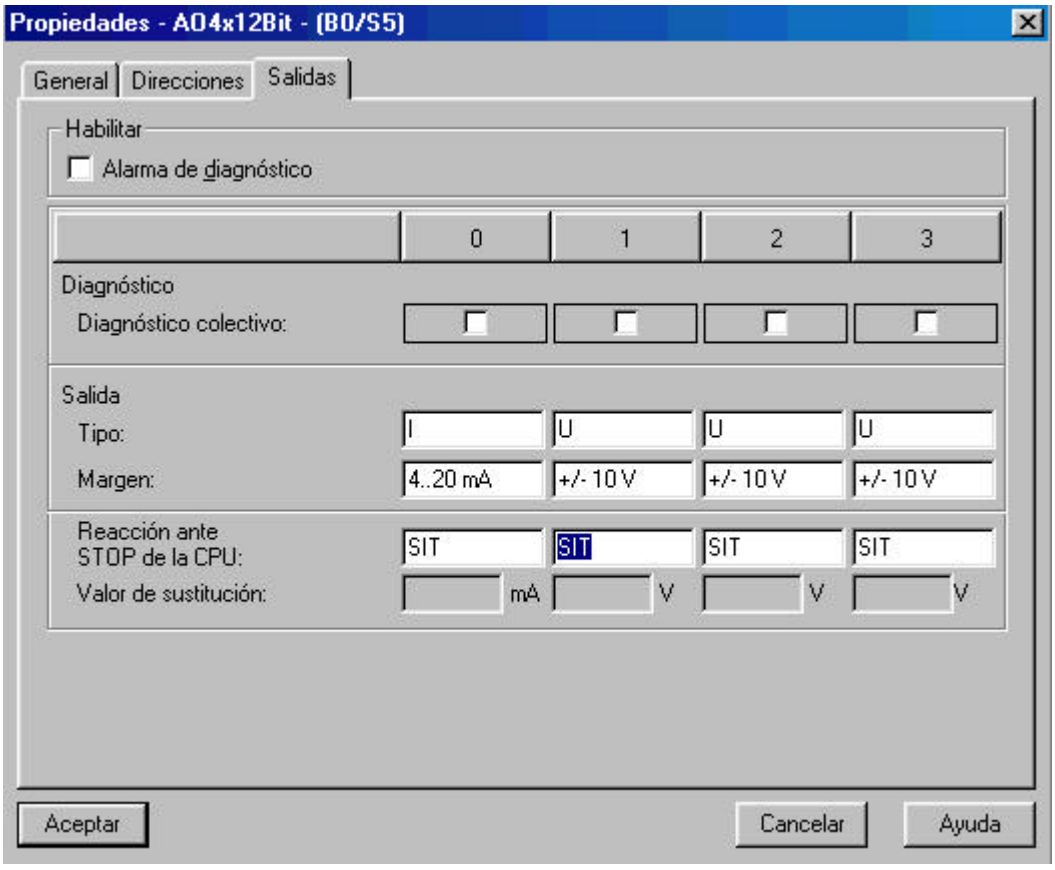


Figura 34 . Configuración del módulo de 4 SA desde Step 7.

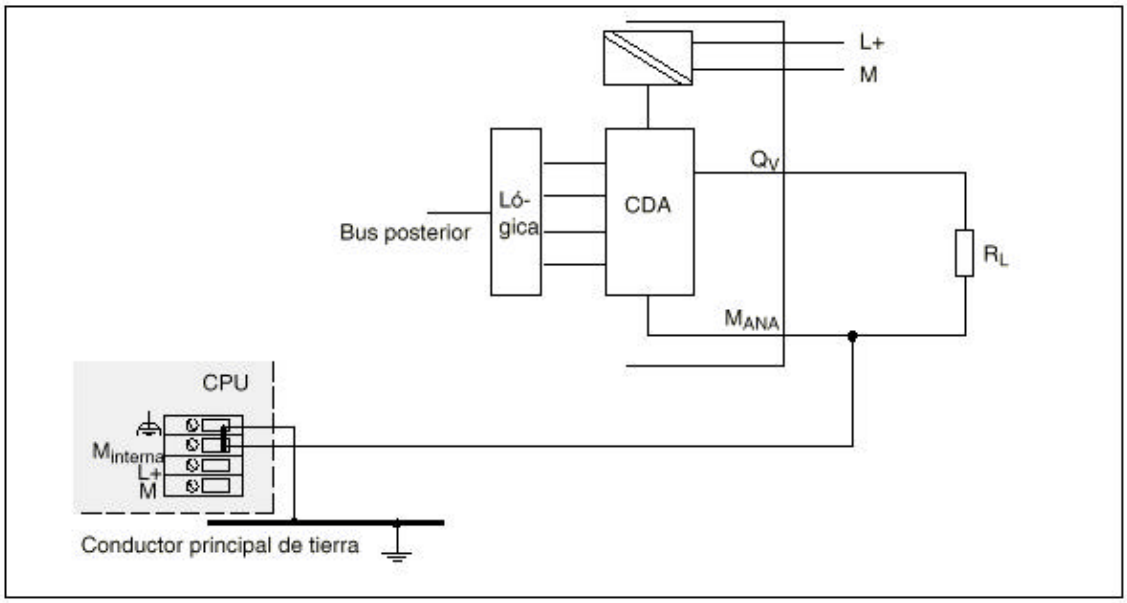


Figura 35 . Cableado de una salida de tensión a 2 hilos.

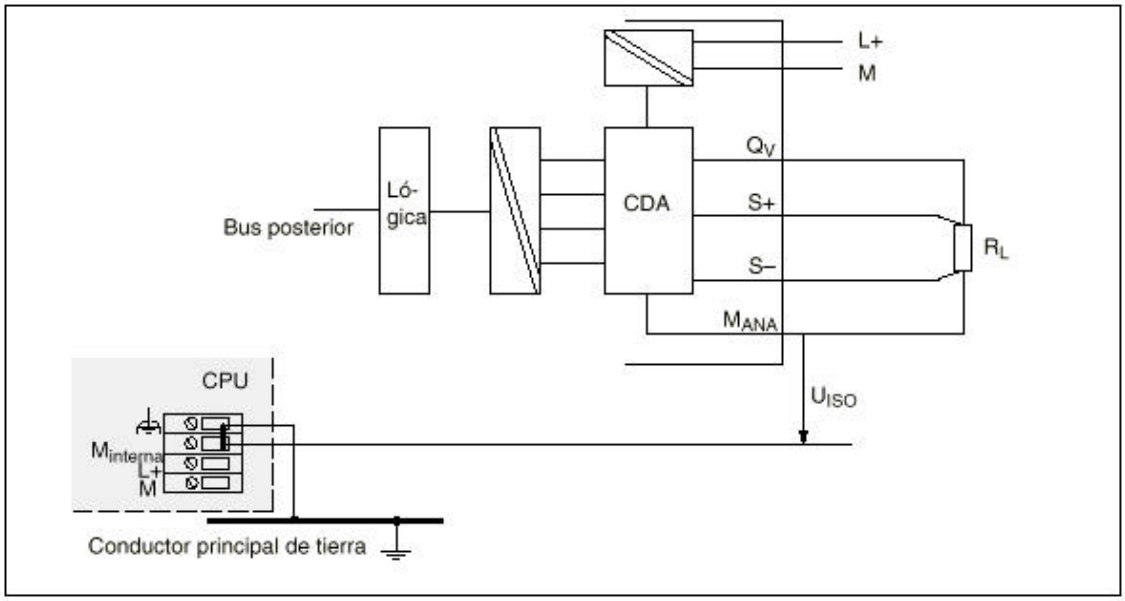


Figura 36 . Cableado de sondas de tensión a 4 hilos (mayor exactitud en el valor analógico).

El conexionado de sensores a 4 hilos como muestra la figura permite ajustar mucho más el valor real de salida, pero requiere de mayor cableado, además de ser obligatorio que no exista una diferencia de tensión entre S- y Mana. Esta configuración sólo es posible en salidas de tensión, no de intensidad.

8.4.3 Tarjeta 4 EA/2 SA SM334.

Esta tarjeta posee 4 entradas analógicas de 8 bits de resolución y 2 salidas analógicas también de 8 bits.

A diferencia de las demás tarjetas de Simatic S7, la configuración del tipo de entradas o salidas no se realiza por software desde Step 7, sino según el cableado que se realice en la misma tarjeta. No poseen por tanto transductores en el lateral para seleccionar el tipo de medida.

Un borne de masa Mana debe de conectarse a la masa de la CPU. Los canales de entradas deberán de cortocircuitarse y unirse con Mana. Los canales de salida no utilizados deberán de dejarse abiertos.

Los márgenes de entradas o analógicas pueden ser de 0-10 V o de 0-20 mA, no pudiéndose tomar registros negativos, al no disponer de +/-10 V ni de +/- 20 mA.

El módulo no posee separación galvánica entre las entradas/salidas y el bus de comunicaciones.

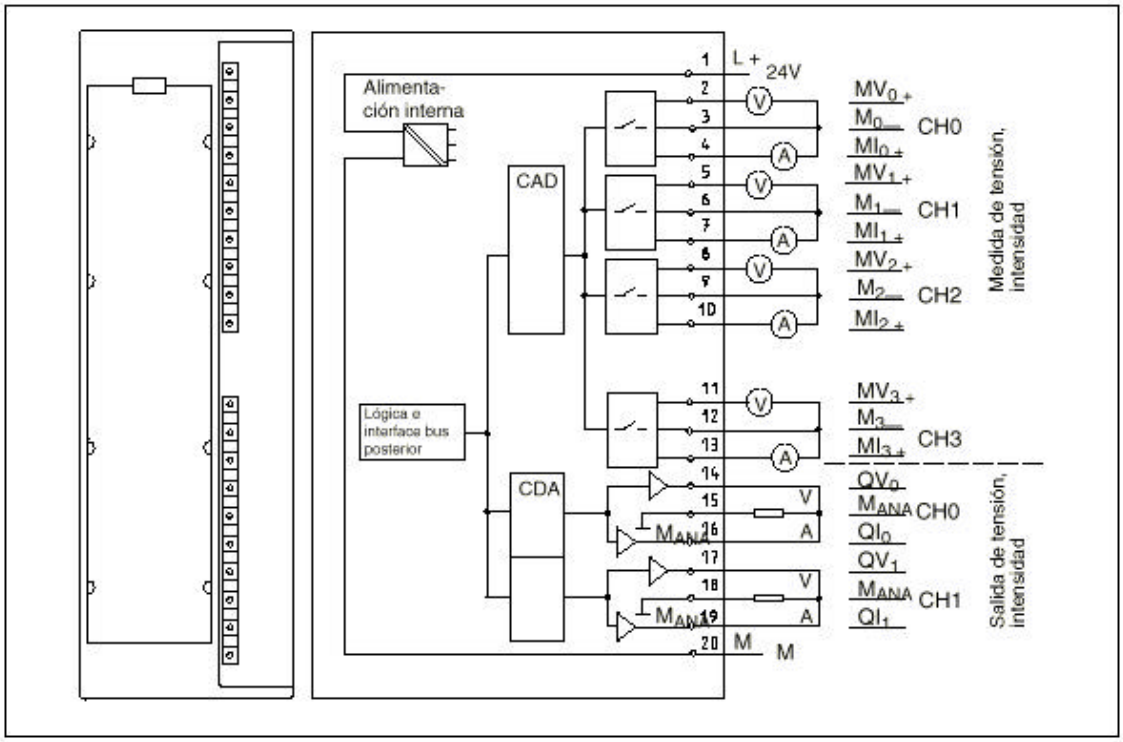


Figura 37 . Esquema de funcionamiento módulo 4 EA/ 2 SA SM334.

Si se desconecta la unión entre Mana y M de la tarjeta el módulo saca el valor 0 por las salidas y da una lectura de 32767 en las entradas.

Las entradas analógicas son pasivas tanto en tensión como en intensidad, por lo que es necesario utilizar transductores activos.

Las salidas analógicas son activas, por lo que se pueden realizar conexiones a 2 hilos tanto en tensión como en intensidad. No es posible la conexión a 4 hilos en tensión.

Atención: la inversión de la polaridad en la alimentación de las entradas analógicas produce la destrucción de los 4 canales de entradas, ya que no están separados galvánicamente. Compruebe por tanto que conecta correctamente los transductores activos con la polaridad adecuada o inutilizará la tarjeta sin solución.

El tiempo de conversión para los cuatro canales de entradas es de 5 ms.

Los canales e entradas no utilizados se deberán de cortocircuitar en tensión, mientras que los de salidas dejarse abiertos.

8.4.4 Entradas / salidas integradas CPU 314 IFM.

La CPU 314 IFM es la única CPU que posee 4 entradas analógicas y 1 salida analógica integradas. Las entradas analógicas van de la EW128 a la EW132, mientras que la salida es la AW128.

Los márgenes admisibles tanto para las entradas analógicas como para las salidas analógicas son +/-10 V o +/-20 mA. La parametrización de las mismas se realiza por cableado, no a través del software de Step 7.

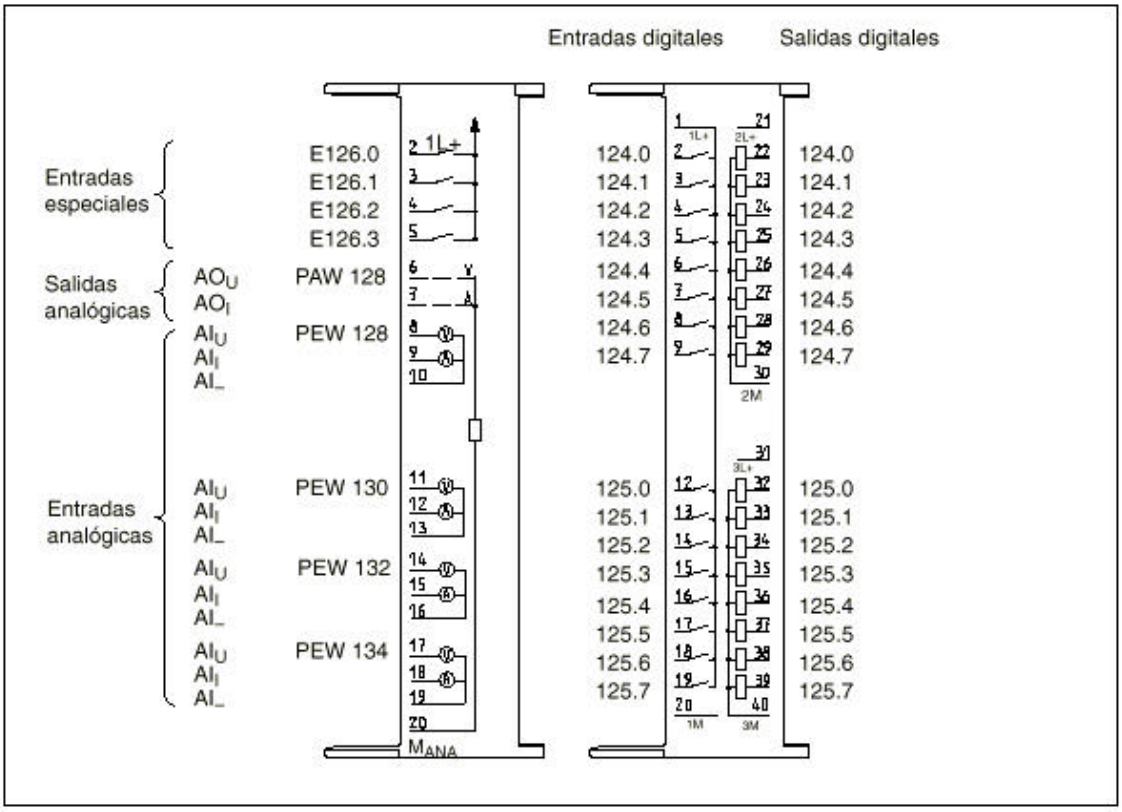
Las entradas analógicas generan tensión (son activas por el lado de +/-10 V), pero no corriente, por lo que las configuraciones a 2 hilos de sondas pasivas de corriente no son posibles. Es necesario que si la sonda es de corriente, tenga alimentación propia.

La salida analógica en corriente y en tensión es activa, siendo imposible el conexionado de cargas a 4 hilos en tensión (exclusivamente a 2 hilos).

La resolución es fija tanto en entradas como en salidas a 11 bits mas signo. El tiempo de conversión de las entradas analógicas es de 100 microsegundos, y de 40 microsegundos para la salida analógica.

El cableado de las sondas debe tener un máximo de 100 m.

Existe separación galvánica entre los canales y el bus de comunicaciones.



Se debe enlazar AI- con Mana, y esta ponerla a tierra. Los canales no utilizados de entradas se deben de cortocircuitar en tensión.

Ni las entradas ni las salidas poseen funciones de diagnóstico ni de alarmas frente a rotura de cable ni desbordamiento.

La salida analógica está protegida contra cortocircuitos.

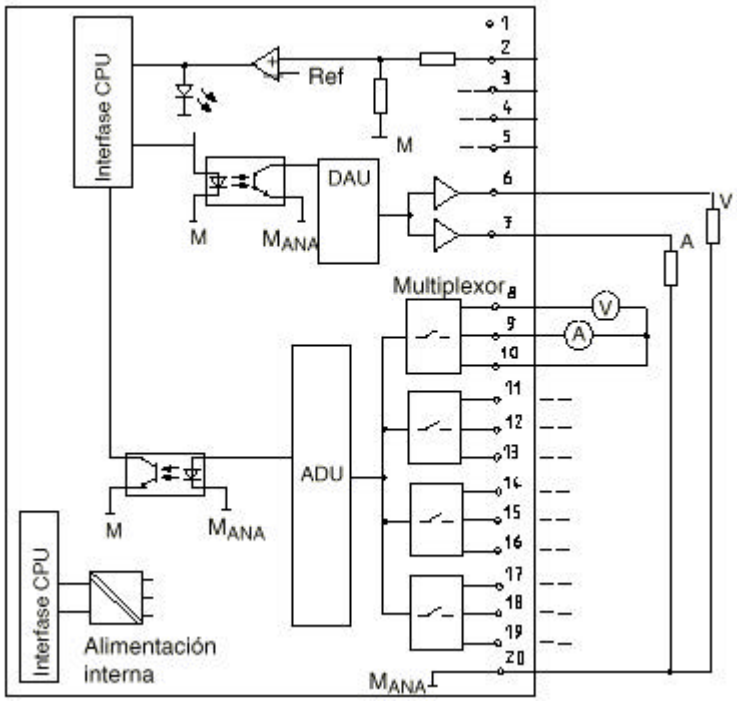


Figura 40 . Esquema interno de E/S analógicas CPU 314 IFM.

8.5 Programación de valores analógicos.

Existen dos posibilidades a la hora de tratar estos valores analógicos obtenidos con los módulos anteriores: adquirir directamente de la periferia el valor o tratarlo a través de funciones de la librería de Step 7.

8.5.1 Lectura directa de periferia.

Una vez convertido por la tarjeta el valor de una entrada analógica a un valor digital, este se almacena en una palabra de 16 bits en la dirección que se indique para dicho canal dentro del administrador hardware de Step 7.

Si disponemos de una CPU maestra de Profibús DP, el direccionamiento del módulo analógico podremos modificarlo y direccionarlo de la zona de entradas analógicas (EW256 y superior) a la zona de entradas digitales (EW0 a EW254). Si esto no es posible, ya que no disponemos de una CPU con esta característica, deberemos de acceder al valor analógico con un acceso directo a la periferia, mediante la instrucción PEW (la P indica el acceso directo a la periferia).

Dicho valor deberá de ser transferido a una palabra de marcas, para no tener que acceder a la periferia cada vez que operemos con él (esta acción aumenta el tiempo de ciclo con relación al acceso a valores a través de la PAE). Posteriormente será necesario un escalado de dicho valor a la magnitud con la que trabajemos en programa.

8.5.2 Lectura mediante FC's.

Evidentemente el método anterior de lectura de analógicas es muy rudimentario, a la vez que engorroso. Para solventar este punto, existen dentro de la librería *Standard Library (STDLIB30)*, en el apartado *TI-S7 Converting Blocks,* dos FC's que se encargan de estos menesteres:

- **FC105 Scale:** lectura normalizada de entradas analógicas.
- **FC106 Unscale:** escritura normalizada de salidas analógicas.

Para utilizarlas deberemos de copiarlas en nuestro proyecto, y posteriormente llamarlas desde nuestro programa.

Veamos primeramente la lectura de valores mediante la FC105. En el ejemplo inferior se aprecia un tratamiento analógico.

```
Segm. 1: LECTURA DE UN VALOR ANALOGICO
      CALL  "SCALE"                    FC105          -- Scaling Values
       IN     :=PEW288
       HI_LIM :="ANAL".MAX             DB1.DBD4       -- LIMITE MAXIMO ANALOGICO
       LO_LIM :="ANAL".MIN             DB1.DBD8       -- LIMITE MINIMO ANALOGICO
       BIPOLAR:="CERO"                 M0.1           -- VALOR SIEMPRE CERO
       RET_VAL:="TEMPW1"               MW100          -- TEMPORAL WORD 1
       OUT    :="ANAL".VALOR           DB1.DBD0       -- VALOR ANALOGICO ACTUAL


Segm. 2: CONTROL DE ERRORES LECTURA ANALOGICA
      L     "TEMPW1"                   MW100          -- TEMPORAL WORD 1
      L     0
      ==I
      SPB   BOT1

      L     0
      T     "ANAL".VALOR               DB1.DBD0       -- VALOR ANALOGICO ACTUAL
      S     "ALARMA_1"                 M20.0          -- ALARMA 1

BOT1: NOP   0
```

En el ejemplo se puede apreciar como accedemos al valor de periferia PEW288 mediante la función FC105. Los parámetros de la función son:

- **IN - word:** valor de periferia indicado en la parte de hardware de Step 7 para dicho canal a leer.
- **HI_LIMIT – real - entrada:** límite superior a escalar el valor de periferia. Es aconsejable dejarlo como parámetro de una DB para poder modificar el escalado de la analógica a posteriori desde un panel o scada. Esto no era posible en Step 5 y supone una gran ventaja en programación.
- **LO_LIMIT – real - entrada:** límite inferior a escalar el valor de periferia. Es aconsejable dejarlo como parámetro de una DB para poder modificar el escalado de la analógica a posteriori desde un panel o scada.
- **BIPOLAR – bool - entrada:** Parámetro que indica si el valor a leer en la periferia va a posee valores negativos o no. En configuraciones +/-10V y +/-20 mA y PT100 deberá de configurarse como bipolar = TRUE, mientras que en el resto a FALSE.
- **RET_VAL – word - entrada:** Código de error de la lectura. Si el valor analógico de la periferia rebasa los márgenes establecidos por hardware o software, desde esta función el valor de esta palabra es distinto de cero.
- **OUT –real – salida:** Valor escalado y convertido a real.

En el segmento 2 del ejemplo se puede apreciar un pequeño tratamiento de errores utilizando RET_VAL para si existe error poner a cero el valor leído y activar una marca de alarmas.

A continuación veremos la escritura de valores analógicos mediante la FC106:

```
Segm. 3: ESCRITURA DE VALORES ANALOGICOS
      CALL  "UNSCALE"                  FC106          -- Unscaling Values
       IN     :="ANAL".VALOR           DB1.DBD0       -- VALOR ANALOGICO ACTUAL
       HI_LIM :="ANAL".MAX             DB1.DBD4       -- LIMITE MAXIMO ANALOGICO
       LO_LIM :="ANAL".MIN             DB1.DBD8       -- LIMITE MINIMO ANALOGICO
       BIPOLAR:=FALSE
       RET_VAL:="TEMPW1"               MW100          -- TEMPORAL WORD 1
       OUT    :=PAW272


Segm. 4 : CONTROL DE ERRORES ESCRITURA ANALOGICA

      L     "TEMPW1"                   MW100          -- TEMPORAL WORD 1
      L     0
      ==I
      SPB   BOT2

      S     "ALARMA_1"                 M20.0          -- ALARMA 1

BOT2: NOP   0
```

Los parámetros poseen el mismo valor que anteriormente. Tomamos un valor real IN y lo sacamos escalado a través de la palabra PAW272, que es la que tenemos asignada al canal de salidas analógicas. Se puede apreciar en el siguiente segmento un tratamiento del error posible en la analógica en cuestión.

9

SISTEMAS DE REGULACION PID.

9.1 Teoría sobre regulación.

El sistema de regulación PID es la manera más extendida de controlar una magnitud de un proceso actuando sobre una de sus características físicas.

En nuestra vida diaria regulamos muchos procesos de manera inconsciente:

- Todas las mañanas al ducharnos, regulamos la temperatura del agua a nuestro gusto.
- En el coche, al ir al trabajo, regulamos la temperatura del coche, ya sea manualmente, o los más afortunados, con climatizador.

En cualquier caso, es evidente que conseguir una buena regulación es posiblemente uno de los apartados más importantes en un sistema de control, y de los más gratificantes a nivel personal (también de los menos cuando no va, sobre todo en la ducha...).

Existe una primera división en los sistemas de regulación PID:

- Regulación de lazo abierto, y
- Regulación de lazo cerrado.

9.1.1 Lazo abierto y lazo cerrado.

En la regulación de lazo abierto el control desconoce el efecto que su acción está produciendo en el sistema, por lo que es incapaz de adecuarse a variaciones en el mismo. Ejemplos de estos sistemas serían el aire acondicionado de un coche, el cual, al no controlar la temperatura del mismo, debe de ser regulado cada cierto tiempo por el usuario para no congelarse. Por lo tanto, este tipo de control carece de interés para nuestro propósito de control de instalaciones.

En la regulación de lazo cerrado, que es la que nos va a ocupar en el presente capítulo, el control está realimentado por un valor procedente de la magnitud a controlar, de tal

manera que es capaz el sistema de control de anticiparse a las fluctuaciones que su acción genera sobre el sistema, y alcanzar la consigna y mantenerla.

9.1.2 Tipos de regulación de lazo cerrado.

Dentro de la regulación de lazo cerrado, la más difundida es la regulación PID. Este algoritmo, que no es más que una función matemática, se compone de tres partes:

- P: acción proporcional. Esta acción responde al error que existe entre el valor que deseamos alcanzar y el que existe actualmente de manera proporcional.
- I: acción integral. Esta acción responde al error de manera incrementativa en el tiempo.
- D: acción derivada. Responde esta acción a aumentos del error en el tiempo. Cuanto mayor sea la derivada del error (pendiente), mayor será la acción derivada.

Vamos a estudiar la regulación PID poniendo como ejemplo el control de la temperatura de una sala con un Simatic S7 300.

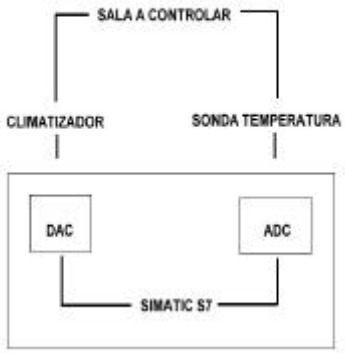


Figura 41. Esquema de un control de temperatura desde PLC

Dentro de la regulación de lazo cerrado, podemos realizar regulaciones de:

- Lazo de primer orden.
- Lazo de segundo orden.

El orden de un lazo viene determinado por la cantidad de variables que dependen entre sí. Ejemplo de un sistema de primer orden es el control de la temperatura de la sala, mientras que un sistema de segundo orden sería controlar la temperatura y la

humedad de la sala, ya que ambas magnitudes físicas están relacionadas a través de relaciones (curvas higrométricas).

La regulación en los autómatas S7 está pensada para sistemas de primer orden, por lo que cualquier sistema de segundo orden se deberá de tratar como dos de primero.

9.2 Regulación PID en S7.

La regulación PID en los S7 se realiza de manera software a través de funciones, por lo que el procesamiento del PID está en función de la velocidad de la CPU y del número de PID's así como de la cantidad de programa adicional que deba de procesar.

Existen dos posibilidades en regulación S7 300:

- **Que se realice mediante funciones integradas en el firmware de la CPU**; esto sólo es posible en las CPU's IFM, que poseen las funciones de sistema SFC41, SFC42 y SFC43.
- **Que se realice mediante funciones normales importadas de la librería de Step 7**. Esta opción es la que se debe de realizar en todas las CPU's que no sean IFM, y que por lo tanto no posean dichas SFC's. Esta opción requiere, por un lado, ocupar espacio de programa para dichas FC's, cosa no necesaria en las IFM's que ya las llevan incorporadas (la memoria empleada es de 1 K para la FB41, 1,8 K para la FB42 y algo menos de 1 K para la FB43). Por otro lado, la ejecución de la función en los equipos IFM es ligeramente más rápida que utilizando las funciones de la librería, aunque el resultado formal de ambas es el mismo, por lo que se pueden utilizar ambos métodos sin problemas.

9.2.1 Funciones PID de la biblioteca S7.

Dentro de la librería *Standard Library* disponemos del apartado *PID Control Blocks* en el cual se encuentran las funciones:

- **FB 41 – CONT_C:** Función que se encarga de la regulación continua.
- **FB 42 – CONT_S:** Función que se encarga de la regulación por pulsos PI.
- **FB 43 – PULSEGEN:** Función que se encarga de la regulación por pulsos PID.

Para poderlos utilizar deberemos e copiarlos en nuestro proyecto, y posteriormente ser llamados desde nuestro programa de PLC. Al ser FB's deberemos crear una DB asociada a los mismos para cada una de las llamadas que realicemos a estas funciones, lo cual constituirá un regulador independiente por llamada.

9.2.2 FB41. Regulación continua.

Para la regulación continua se utiliza la FB41

El esquema de bloques del calculo de regulación es el que muestra la figura:

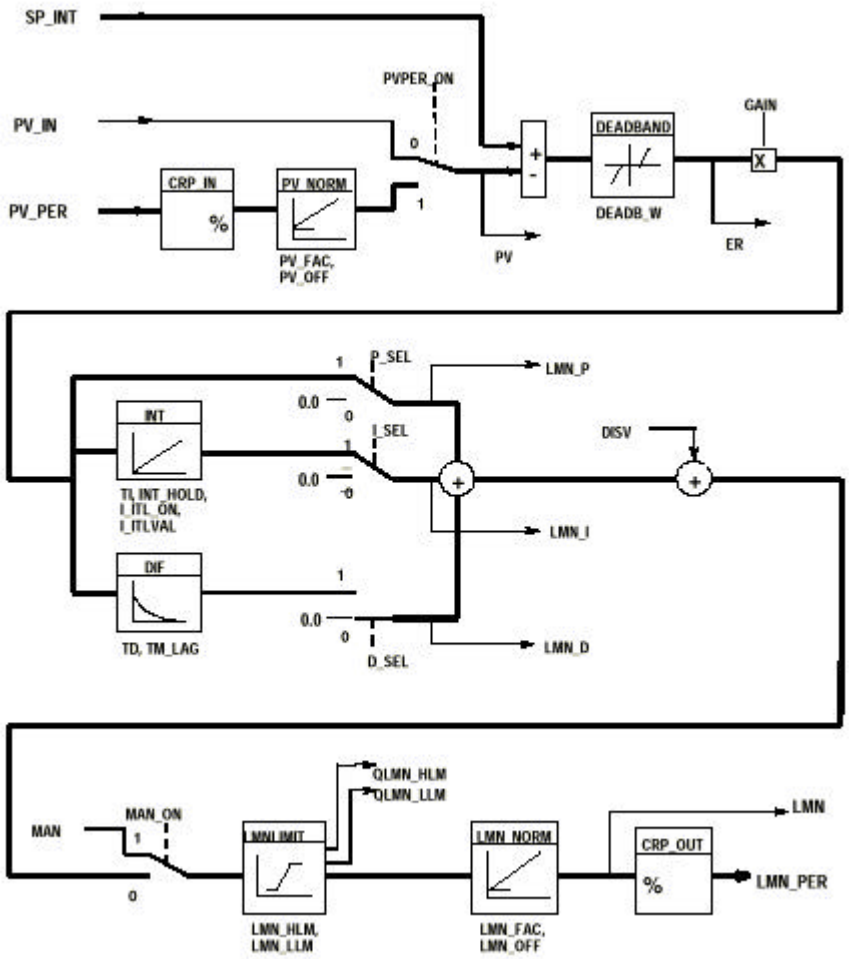


Figura 42 . Esquema de bloques regulación continua FB41.

Veamos que parámetros componen este tipo de regulación:

- **COM_RST – bool:** Rearranque del PID. La activación de esta entrada produce el rearranque y la inicialización de los valores de salida del PID. Esta acción debe de realizarse en el paso de STOP-RUN del PLC ya que si se desconecta por cualquier motivo la sonda que indica el valor de retorno del PID (señal *PV_IN*) el PID va acumulando cálculos de error, y en su caso abriendo o cerrando la salida, según sea una regulación positiva o negativa. Si se vuelve a conectar la sonda, hasta que desaparezca el error acumulado en el PID no volverá la regulación a entrar en los márgenes de regulación, por lo que continuará abierta o cerrada según el caso. Para evitar esto, se puede colocar una marca que se activará siempre en la OB100, y que

se reseteará después del último PID. Esta marca activará *COM_RST*, de tal manera que cada vez que se pare la instalación se reinicializan los reguladores. Si no es posible para la instalación por ser un proceso continuo, asociar esta marca a un panel o variable de scada para poder modificarla externamente de manera manual.

- **MAN_ON –bool:** Indica si el regulador se debe de comportar en manual o en automático. Por defecto el valor viene a TRUE, por lo que si no se modifica, el regulador continuará en manual dando en la salida el valor que indiquemos en el parámetro MAN. Es recomendable dejar este parámetro como una opción seleccionable desde un equipo de visualización, para poder pasar el regulador a manual.
- **PVPER_ON – bool:** Existen dos formas de introducir el valor de la instalación a regular (la temperatura, la presión, la humedad...):
    - Leer una entrada analógica mediante la función FC105, que convierte el valor de periferia a un valor real, que se suele almacenar en un dato real de una DB, o
    - Leer directamente el valor de la periferia si ser tratado antes, directamente en formato WORD.

Mediante el parámetro *PVPER_ON* podemos seleccionar con FALSE, que tome el valor de periferia del parámetro *PV_IN*, y con TRUE, que tome el parámetro de *PV_PER*. Es generalmente interesante dejarlo a FALSE, que es como viene por defecto, e introducir el valor de proceso para el regulador escalándolo anteriormente mediante una llamada a la FC105, ya que así podemos por un lado parametrizar el escalado de la magnitud del proceso mediante los límites superior e inferior, y por otro, disponemos del valor ya escalado, para poder visualizarlo en un equipo OP o scada. Pro defecto viene a FALSE.

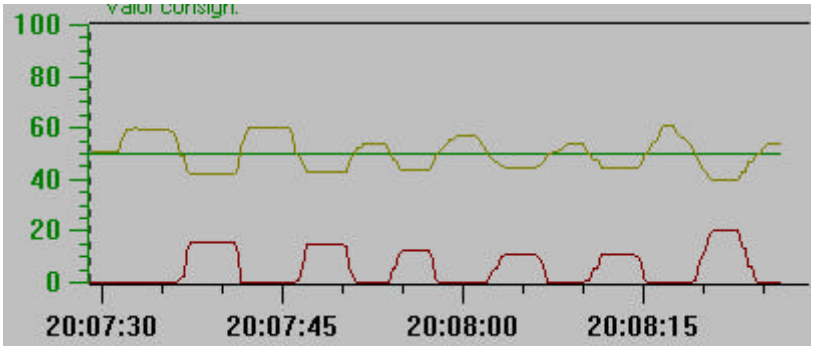


Figura 43 . Ejemplo de regulación exclusívamente proporcional, sin acción integral ni derivada. Obsérvese como cuando no existe una variación en el valor real, el error se va a mantener estacionario, ya que la salida tiende a mantenerse constante, y no responde frente al mismo. Debido a esto, un regulador P generará un error estacionario grande, debiéndose de combinar con la parte integral, y/o con la derivativa.

- **P_SEL – bool:** Activar la parte proporcional del PID. Se permite desconectar la parte P de la regulación, pero los tipos usuales de regulación son PID y PI, por lo que no es recomendable esta desconexión. Por defecto viene a TRUE (activada).
- **I_SEL – bool:** Activar la parte integral del PID. Se permite desconectar la parte I de la regulación, pero los tipos usuales de regulación son PID y PI, por lo que no es recomendable esta desconexión. Por defecto viene a TRUE (activada).

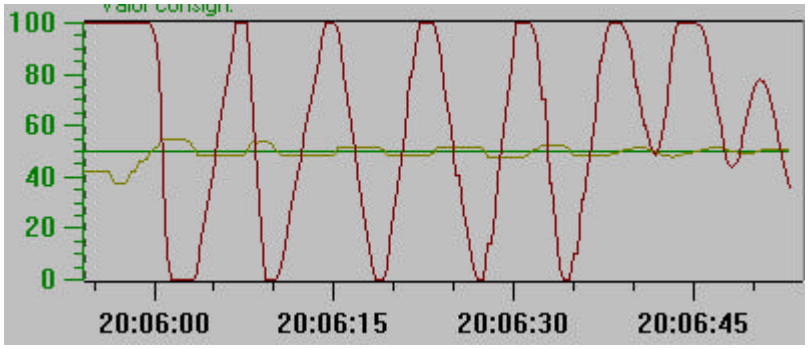


Figura 44 . Ejemplo de regulación con únicamente acción integral. Destacar como frente al error estacionario esta vez sí se actúa, pero si deseamos responder con celeridad a las variaciones del valor real solo con parte I, el valor de la parte integral deberá de ser tan grande que se producirán sobreamortigüamientos en la instalación. Es por esto que se suele gastar combinado con la parte proporcional, para que el efecto integral sólo se encargue de corregir errores estacionarios.

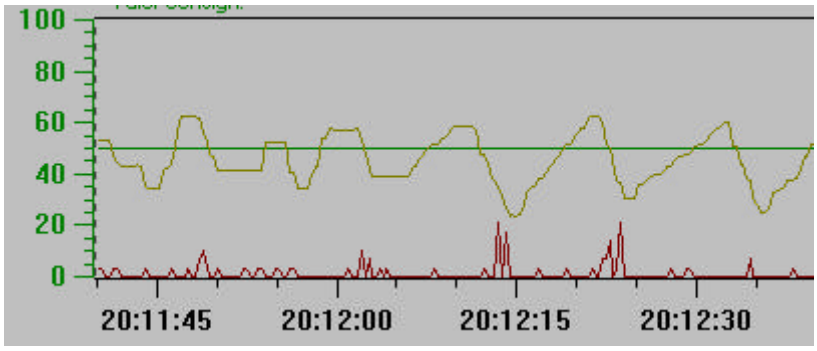


Figura 45 . Ejemplo de una instalación sólo con parte derivativa. Como se puede observar, tan solo responde la salida frente a variaciones bruscas del valor real. Esto es insuficiente para controlar por sí solo el proceso, pero puede ser un efecto beneficioso, sumado a los anteriores, si nuestra instalación pierde en momentos concretos y de manera brusca la consigna. Es en esos casos cuando interesa un efecto que contrarreste esta pérdida.

- **INT_HOLD – bool:** Mediante este parámetro se puede congelar la acción integral, de tal manera que el regulador no actúa frente a errores estacionarios. Por defecto viene a FALSE (sin mantener la acción integral).
- **I_ITL_ON - bool:** Inicializar la acción integral. Podemos inicializar activando esta entrada la parte integral a un valor deseado. Se puede utilizar para resetear la parte integral. Por defecto viene a FALSE (no inicializar la parte integral).

- **D_SEL – bool:** Activar la parte derivativa del PID. Por defecto viene a FALSE, por lo que mientras no se active, el regulador es de tipo PI.
- **CYCLE – time:** Tiempo de muestreo el PID. El tiempo de muestreo indica cada cuanto se ejecuta el algoritmo PID, generando un resultado efectivo. La función PID se debe de ejecutar en ciclos regulares de tiempo para que la salida sea uniforme en el mismo. Para ello debe de ser llamada la función desde una subrutina que no dependa del tiempo de ciclo del PLC, que es variable dependiendo de si se ejecutan unas instrucciones u otras. Esto se consigue con la ejecución del PID dentro de una OB de tiempo (OB 35 p. Ej). El tiempo de ciclo indicado en este parámetro deberá de ser mayor o igual al tiempo de llamada de la OB (en el caso de la OB 35 viene prefijado a 100 ms, aunque se puede cambiar desde hardware de Step 7). Valores de CYCLE inferiores no son efectivos, ya que cada vez que entre a la OB le tocará ejecutarse al PID, por lo que regirá el tiempo de la OB y no este parámetro.
- **SP_INT – real:** consigna interna del PID.
- **PV_IN – real:** valor real interno. La función del PID es controlar una magnitud física. En este parámetro se refleja el valor actual de dicha magnitud (la temperatura de la sala, la presión de la tubería, la humedad del ambiente...). Normalmente aquí se indica un valor que es obtenido anteriormente de la FC105, escalando una entrada analógica.
- **PV_PER – real:** valor real de la periferia. No es necesario tratar anteriormente la entrada analógica, ya que se puede indicar en este parámetro dicho valor sin tener que escalarlo. Para seleccionar entre este valor y el anterior, ver el parámetro anterior PVPER_ON.
- **MAN – real:** valor manual. Este parámetro se traslada a la salida del PID cuando indicamos el funcionamiento manual del PID. Observar que lo que se indica en este parámetro no es la temperatura, sino la posición de la compuerta de calor que la regula, no es la presión, sino las revoluciones del variador que la genera, no es la humedad, sino la cantidad de aire frío que la condensa, etc...
- **GAIN – real:** ganancia del PID. La ganancia indica la constante de la parte proporcional del PID. Cuanto mayor sea el parámetro GAIN, mayores serán los incrementos en la evolución de la salida del PID y viceversa. Esto tiene como resultado efectivo que un valor reducido de ganancia genera unas evoluciones lentas en la salida del PID, pero por el contrario el error estacionario que se produce en la regulación es menor. La regulación es lenta pero exacta, tendiendo a un sistema de control subamortigüado. Por el contrario, valores altos de ganancia producen evoluciones rápidas en la salida, pero aumenta el error estacionario y se tiende a sistemas sobreamortigüados. Valores de 1 o inferiores son interesantes para la mayoría de los procesos “tranquilos” en su evolución.

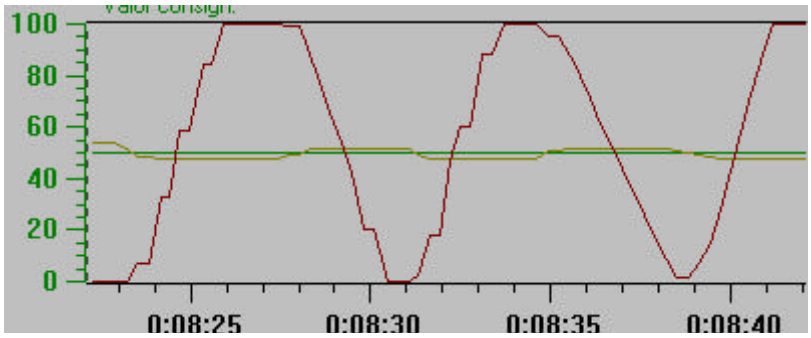


Figura 46 . Regulador PI con una ganancia de 2.0. Observar como se produce una salida de regulación con escalones bruscos y evolución rápida. En la mayoría de los procesos esto no es deseable, prefiriéndose una regulación suave.

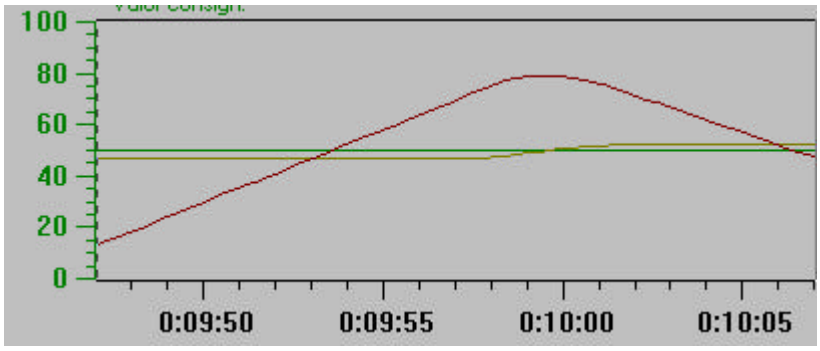


Figura 47 . El mismo regulador PI, pero esta vez con GAIN a 0,2. La evolución es más suave, pero también mucho más lenta. Esto es deseable en la mayoría de las regulaciones.

- **TI – time:** tiempo de acción integral. Este parámetro determina el tiempo que se tardará en realizar la integración de la parte integral del algoritmo PID. Esto no quiere decir que hasta que no se cumpla este tiempo no se adiciona la parte integral a la salida, sino que dicha parte se reparte en este periodo de tiempo. Según esto, con periodos de tiempo cortos en este parámetro la evolución de la salida es brusca, conduciendo al sistema a procesos sobreamortigüados, mientras que valores grandes de tiempo en él generan evoluciones suaves, tendiendo a sistemas subamortigüados. Un tiempo de 120 segundos en este parámetro es usualmente interesante para la mayoría de los procesos.
- **TD – time:** tiempo de acción derivativa. El equivalente al anterior pero para la parte derivativa. Igualmente, valores reducidos generan inestabilidad y rápida evolución, mientras que valores grandes tienden a reducir la propia acción derivativa.
- **TM_LAG – time:** retardo de la acción derivativa. Como se sabe, la acción derivativa se encuentra en función de la derivada de la magnitud del proceso, es decir, de la pendiente de variación de dicha magnitud. Se permite con este parámetro retardar dicha acción, para que la propia entrada de la acción derivativa no influencie la

variación de la salida, y se retroalimente a sí misma disparándose la parte derivativa del PID.

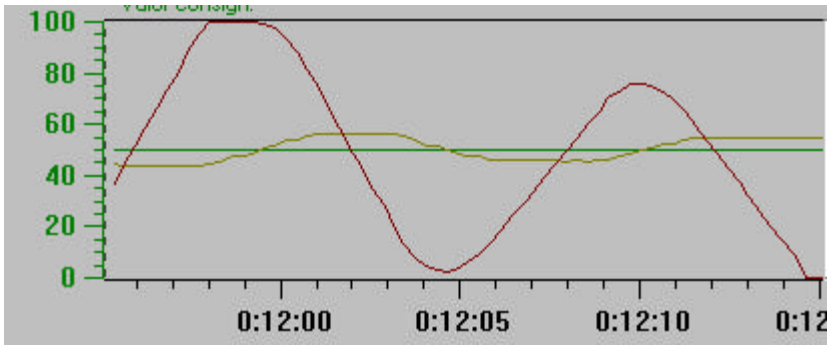


Figura 48 . Regulación PI con un tiempo de integración de 10 segundos. Cuanto menor sea el tiempo de integración, en menos tiempo se sumará la acción integral a la salida del regulador, y por lo tanto su evolución será más rápida.

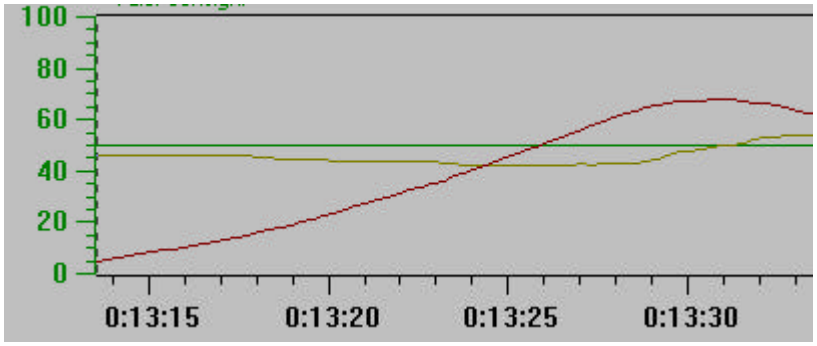


Figura 49 . El mismo regulador anterior, pero ahora con un tiempo de integración de 60 segundos. Se aprecia una evolución muy suave, gracias a que la acción integral tiene un tiempo mayor para actuar. No se debe tampoco de olvidar que una evolución demasiado lenta conducirá a un sistema subamortiguado, lo cual nunca es deseable.

- **DEADB_W – real:** ancho de la zona muerta. Cuando el valor de proceso alcanza la consigna seleccionada, la salida del PID disminuye hasta alcanzar si fuese el caso el valor nulo. Un disminución de la salida suele producir un alejamiento del valor real de su consigna, con lo cual vuelve a actuar la acción del PID, recuperando la consigna, y repitiéndose el ciclo. Estas pequeñas fluctuaciones (o grandes, según sea el parámetro GAIN) en algunos procesos generan más complicaciones que beneficios, no siendo recomendables. El ejemplo más claro es el control de llenado de un recipiente. Si deseamos mantener su altura constante, independientemente del agua que extraigamos a través de un grifo que vacía el recipiente, aplicaremos un PID a la altura del mismo. Pero al alcanzar la consigna de altura, el variador que impulsa el agua de llenado tenderá a parar. Ya sea por pequeñas olas del

recipiente o por el propio vaciado, inmediatamente se generará un error que volverá a arrancar el variador, pero debido a que este error es muy pequeño, la acción del PID será igualmente baja, por lo que funcionará a bajas revoluciones, con el consiguiente perjuicio a la refrigeración del motor conectado. La solución pasa por una banda muerta, que se adiciona a la consigna, y sólo después de haber disminuido la altura del recipiente por debajo de dicho valor actuará el algoritmo PID. Como en es e momento el error a corregir será suficientemente elevado, el variador arrancará a una frecuencia prudente para la vida útil del motor.

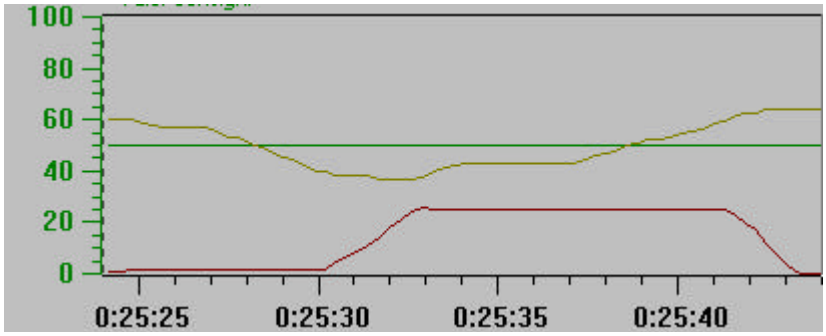


Figura 50 . Regulador con una banda muerta de 10. Cuando bajamos por debajo de la consigna, tan solo actúa la parte proporcional, hasta que superamos la zona muerta. Entonces entra la parte integral, se vuelve a entrar en zona muerta y se queda actuando la parte proporcional de nuevo. Si subimos por encima de la consigna, mientras estemos en zona muerta continua la parte P, hasta que una vez superada por arriba, el regulador cierra totalmente.

- **LMN_HLM – real:** límite superior de la salida del PID. El valor máximo que puede alcanzar la salida del PID, medido en unidades del actuador del proceso: velocidad máxima del variador que controla el ventilador, posición máxima de la compuerta que controla el aire, posición máxima de la electroválvula que deja pasar el agua caliente, etc...
- **LMN_LLM – real:** límite inferior de la salida del PID. El valor mínimo de dicha salida. Normalmente es cero en la mayoría de los procesos.
- **PV_FAC – real:** valor por el que multiplicar el valor de PV_PER. Es importante destacar que esta constante sólo afecta al valor real de periferia, no al valor interno PV_IN, que suele ser el utilizado en la mayoría de las regulaciones.
- **PV_OFF – real:** este valor se suma al valor PV_PER * PV_FAC, resultando:

  Valor real de periferia = PV_OFF + PVPER * PV_FAC
- **LMN_FAC – real:** límite del valor manipulado. Constante que multiplica a la salida del PID.
- **LMN_OFF – real:** offset del valor manipulado. Este valor se suma al valor de salida del PID después de haberlo multiplicado por LMN_FAC. Por lo tanto, el valor de salida del PID será:

Valor de salida del PID = LMN_OFF + salida * LMN_FAC

- **I_ITLVAL – real:** inicialización de la acción integral. Se puede seleccionar un valor de comienzo para la acción integral. Esto puede ser interesante en procesos en los cuales al arrancar el PID existe un gran error que debe de ser corregido inmediatamente, sin embargo se desea que una vez alcanzada la consigna evolucione muy lentamente la salida del PID. Se selecciona un valor de tiempo de integración alto, pero se inicializa la acción integral a un valor que permita evolucionar la principio con presteza.
- **DISV – real:** variable perturbadora. Si se conoce una magnitud que puede influir en el proceso, además lógicamente de la magnitud a controlar, se puede indicar dicho valor en este parámetro, aunque su influencia debe de ser reducida, ya que en caso contrario nos conduciría a un sistema de segundo orden que se convertiría en inestable.
- **LMN – real:** valor de salida del PID. Este valor es el que deberemos de enviar a la salida analógica en cuestión que regule el actuador de la instalación. Lógicamente, será necesario después del PID realizar una llamada a la FC106 para convertir el valor real a una palabra de la periferia del autómata.
- **LMN_PER – real:** valor de salida de periferia. Si no se desea realizar un escalado de la variable mediante la FC106, en este parámetro disponemos de la salida ya en formato de 16 bits para poder ser enviada a la periferia del autómata.
- **QLMN_HLM – bool:** límite de salida máximo alcanzado. Este bit nos indica que el valor de salida del PID ha alcanzado el valor indicado como límite de salida máximo.
- **QLMN_LLM – bool:** límite de salida mínimo alcanzado. Este bit indica que el valor de salida del PID ha alcanzado el límite mínimo indicado.
- **LMN_P – real:** parte de la salida del PID correspondiente a la acción proporcional.
- **LMN_I – real:** parte de la salida del PID correspondiente a la acción integral.
- **LMN_D – real:** parte de la salida del PID correspondiente a la acción derivativa.
- **PV – real:** valor real de la instalación. Si se ha seleccionado como valor real PV_IN, corresponde a este, mientras que si se ha seleccionado como valor real PV_PER, corresponde a el mismo después de haber sido manipulado.
- **ER – real:** error de la instalación. Es la diferencia que existe en cada momento entre el valor de consigna y el valor real de la instalación.

9.2.3 Regulación por pulsos PI. FB 42.

El regulador FB42 sirve para regulaciones PI con salida por pulsos. El es quema de bloques del funcionamiento del regulador es el que muestra la figura siguiente.

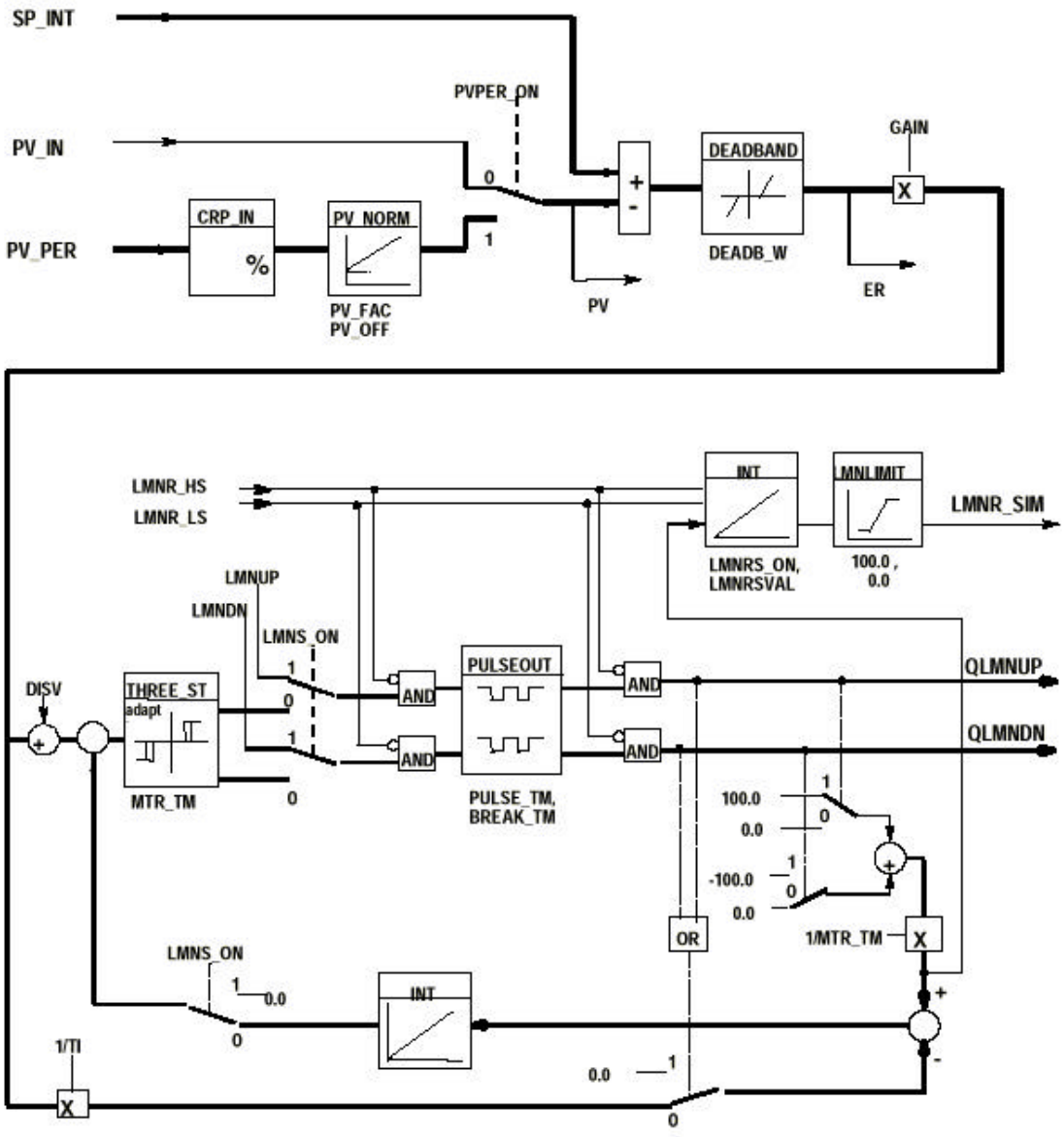


Figura 51 . Esquema de bloques regulador FB42.

Los parámetros de que constan el regulador son:

- **COM_RST – bool:** Rearranque del PID. La activación de esta entrada produce el rearranque y la inicialización de los valores de salida del PID. Igual funcionalidad que su homólogo anteriormente descrito. **MAN_ON –bool:** Indica si el regulador se debe de comportar en manual o en automático. Por defecto el valor viene a TRUE, por lo que si no se modifica, el regulador continuará en manual dando en la salida el valor que indiquemos en el parámetro MAN. Es recomendable dejar este parámetro como una opción seleccionable desde un equipo de visualización, para poder pasar el regulador a manual.
- **LMNR_HS – bool:** señal de límite superior de la respuesta de posición. Indica que la apertura de la válvula ha llegado a su tope físico. En este parámetro se debe de conectar la entrada del final de carrera de posición superior de la válvula, para que no continúe abriendo el PID una vez alcanzado el tope físico.
- **LMNR_LS – bool:** señal de límite inferior de la respuesta de posición. Indica que el cierre de la válvula ha llegado a su tope físico. En este parámetro se debe de conectar la entrada del final de carrera de posición inferior de la válvula, para que no continúe cerrando el PID una vez alcanzado el tope físico.
- **LMNS_ON – bool:** Conectar modo manual. Por defecto está a TRUE, con lo que el regulador no actuará en automático hasta que no cambiemos dicho parámetro a FALSE.
- **LMNUP – bool:** Subir salida del PID en manual. Si ponemos el regulador en manual, con esta entrada abriremos la válvula.
- **LMNDN – bool:** Bajar salida del PID en manual. Si ponemos el regulador en manual, con esta entrada cerraremos la válvula.
- **PVPER_ON – bool:** Existen dos formas de introducir el valor de la instalación a regular (la temperatura, la presión, la humedad...):
    - Leer una entrada analógica mediante la función FC105, que convierte el valor de periferia a un valor real, que se suele almacenar en un dato real de una DB, o
    - Leer directamente el valor de la periferia si ser tratado antes, directamente en formato WORD.

  Mediante el parámetro *PVPER_ON* podemos seleccionar con FALSE, que tome el valor e periferia del parámetro *PV_IN*, y con TRUE, que tome el parámetro de *PV_PER*. Es generalmente interesante dejarlo a FALSE, que es como viene por defecto, e introducir el valor de proceso para el regulador escalándolo anteriormente mediante una llamada a la FC105, ya que así podemos por un lado parametrizar el escalado de la magnitud del proceso mediante los límites superior e inferior, y por otro, disponemos

del valor ya escalado, para poder visualizarlo en un equipo OP o scada. Pro defecto viene a FALSE.

- **CYCLE – time:** Tiempo de muestreo del regulador. Por efecto viene a 1 segundo.
- **SP_INT – real:** consigna interna del PID.
- **PV_IN – real:** valor real interno. La función del PID es controlar una magnitud física. En este parámetro se refleja el valor actual de dicha magnitud (la temperatura de la sala, la presión de la tubería, la humedad del ambiente...). Normalmente aquí se indica un valor que es obtenido anteriormente de la FC105, escalando una entrada analógica.
- **PV_PER – real:** valor real de la periferia. No es necesario tratar anteriormente la entrada analógica, ya que se puede indicar en este parámetro dicho valor sin tener que escalarlo. Para seleccionar entre este valor y el anterior, ver el parámetro anterior PVPER_on.
- **GAIN – real:** ganancia del PID. La ganancia indica la constante de la parte proporcional del PID. Cuanto mayor sea el parámetro GAIN, mayores serán los incrementos en la evolución de la salida del PID y viceversa. Esto tiene como resultado efectivo que un valor reducido de ganancia genera unas evoluciones lentas en la salida del PID, pero por el contrario el error estacionario que se produce en la regulación es menor. La regulación es lenta pero exacta, tendiendo a un sistema de control subamortigüado. Por el contrario, valores altos de ganancia producen evoluciones rápidas en la salida, pero aumenta el error estacionario y se tiende a sistemas sobreamortigüados. Valores de 1 o inferiores son interesantes para la mayoría de los procesos “tranquilos” en su evolución.
- **TI – time:** tiempo de acción integral. Este parámetro determina el tiempo que se tardará en realizar la integración de la parte integral del algoritmo PID. Esto no quiere decir que hasta que no se cumpla este tiempo no se adiciona la parte integral a la salida, sino que dicha parte se reparte en este periodo de tiempo. Según esto, con periodos de tiempo cortos en este parámetro la evolución de la salida es brusca, conduciendo al sistema a procesos sobreamortigüados, mientras que valores grandes de tiempo en él generan evoluciones suaves, tendiendo a sistemas subamortigüados. Un tiempo de 120 segundos en este parámetro es usualmente interesante para la mayoría de los procesos.
- **DEADB_W – real:** ancho de la zona muerta. Igual significado que en el regulador anterior.
- **PV_FAC – real:** valor por el que multiplicar el valor de PV_PER. Es importante destacar que esta constante sólo afecta al valor real de periferia, no al valor interno PV_IN, que suele ser el utilizado en la mayoría de las regulaciones.
- **PV_OFF – real:** este valor se suma al valor PV_PER * PV_FAC, resultando:

  Valor real de periferia = PV_OFF + PVPER * PV_FAC

- **PULSE_TM – time:** duración mínima del pulso. El pulso de salida debe de actuar sobre la bobina de una electroválvula, generalmente. Si el pulso de es muy corta duración, puede ser que la bobina no llegue a excitarse lo suficiente para que se mueva el mismo. Con este parámetro nos aseguramos que el Ton (tiempo de señal activo) del pulso no será demasiado corto. Por defecto viene configurado a 3 segundos.
- **MTR_TM – time:** En este parámetro se indica el tiempo que tarda la válvula partiendo desde cero en llegar a la posición de apertura máxima.
- **BREAK_TM – time:** duración mínima de pausa. Igual que el parámetro anterior, pero esta vez referido al Toff (tiempo de señal inactiva) del pulso.
- **DISV – real:** variable perturbadora. Si se conoce una magnitud que puede influir en el proceso, además lógicamente de la magnitud a controlar, se puede indicar dicho valor en este parámetro, aunque su influencia debe de ser reducida, ya que en caso contrario nos conduciría a un sistema de segundo orden que se convertiría en inestable.
- **QLMNUP – bool:** salida de abrir válvula.
- **QLMNDN – bool:** salida de cerrar válvula.
- **PV – real:** Salida real. En esta variable se indica la apertura real que existe actualmente.
- **ER – real:** error de regulación. En este parámetro se indica el error que existe actualmente entre el valor e consigna y el valor real de la instalación, y sobre el cual actuará el cálculo del PI para corregirlo.

9.2.4 Regulación por pulsos PID. FB 43.

El FB43 (PULSEGEN) está indicado en la regulación por pulsos mediante PID en los Simatic S7 300. Se compone de dos reguladores concatenados: un regulador continuo cuya salida ataca a un regulador de salida por pulsos.

El regulador por pulsos recibe la salida del regulador continuo, y convierte los escalones del mismo en un tren de impulsos cuyo ancho de pulso es proporcional a la misma.

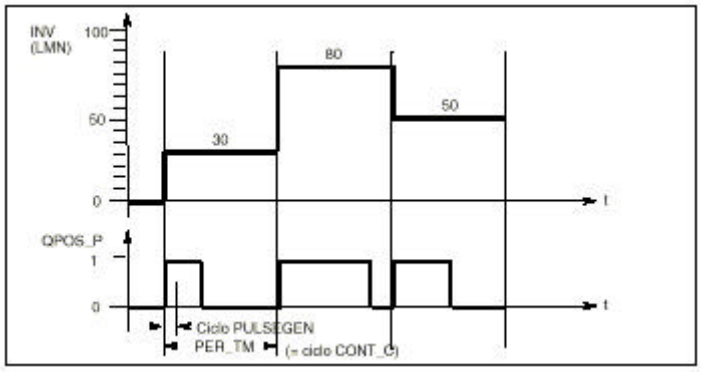


Figura 52 . Regulación por ancho de pulso FB43.

Este tipo de regulador permite tres modos diferentes de operación:

- Regulación de tres puntos.
- Regulación de dos puntos bipolar, y
- Regulación de dos puntos unipolar.

En la regulación de tres puntos (parámetro STEP3_ON = TRUE), las salidas QPOS_P y QNEG_P se encuentran conectadas a dos actuadores distintos, activándose solo una de ellas, pero el tren de pulsos de cada una de ellos puede ser diferente, en función de cómo se comporte la regulación. Supongamos que disponemos de una resistencia que se caliente en función del tren de pulsos aplicado a través de la entradas QPOS_P, y otro elemento que enfría en función del tren con modulación de pulsos que se le aplica por la salida QNEG_P. Los pulsos se aplicarán por una u otra (calentar o enfriar), nunca por las dos, pero no tiene porqué aplicarse la misma cantidad en el proceso de calentamiento que en el de enfriamiento.

En la regulación de dos puntos, la salida del regulador se aplica únicamente sobre QPOS_P. Existen dentro de este tipo de regulación dos posibilidades:

- Regulación de dos puntos bipolar: el valor de la entrada del regulador (valor de proceso) puede adoptar valores negativos (de –100 a 100). En este caso se debe parametrizar STEP3_ON = FALSE y ST2BI_ON = TRUE.
- Regulación de dos puntos unipolar: el valor de la entrada del regulador (valor de proceso) no puede adoptar valores negativos (de 0 a 100). En este caso se debe parametrizar STEP3_ON = FALSE y ST2BI_ON = FALSE.

En ambos casos la salida QNEG_P toma el valor invertido de la que se le asigne a QPOS_P, para aquellos actuadores que requieran señal invertida para dejar de actuar (electroválvula de dos bobinas, p. Ej.).

Los parámetros del regulador son:

- **INV – real:** variable de entrada al regulador. Valor de proceso de la instalación.
- **PER – time:** periodo de la salida de pulsos. Este periodo es constante para el tren de pulsos, determinando el mismo el tiempo máximo del ancho de pulso en estado *on*, y por consiguiente la resolución del tren de impulsos máxima.
- **P_B_TM – time:** Duración mínima del tren de pulsos en estado *on.* Este tiempo determina el mínimo tiempo que la salida deberá estar en estado alto en el tren de pulsos de salida del regulador.
- **RATIOFAC – real:** Factor de relación de la salida positiva y negativa. Mediante este parámetro se puede establecer una relación entre el tren de pulsos positivos y negativos.
- **STEP3_ON – bool:** Conectar regulación en tres puntos.
- **ST2BI_ON – bool:** Trabajar con regulación de dos puntos bipolar. Cuando se trabaja en regulación de 2 puntos, mediante este parámetro se determina si se va a trabajar en forma bipolar (TRUE) o unipolar (FALSE). Para ello necesariamente STEP3_ON debe estar en FALSE.
- **MAN_ON – bool:** Conectar el PID en modo manual.
- **POS_P_ON – bool:** En modo manual, al conectar esta entrada, si estamos en:
  - Regulación de tres puntos: se activa la salida QPOS_P.
  - Regulación de dos puntos: se activa y se invierte el valor de QPOS_N respecto del de QPOS_P.
- **NEG_P_ON – bool:** Pulso negativo *on*. En modo manual, al conectar esta entrada si estamos en:
  - Regulación de tres puntos: se activa la salida QPOS_N.
  - Regulación de dos puntos: se activa y se invierte el valor de QPOS_N respecto del de QPOS_P.
- **SYN_ON – bool:** Conectar sincronización. Mediante esta variable se activa el modo de sincronización, por el cual la respuesta en pulsos del regulador viene en función de la variación en la magnitud de la entrada del regulador continuo en el tiempo.
- **COM_RST – bool:** Inicialización del PID.

- **CYCLE – time:** Tiempo de muestreo. Este tiempo determina cada cuanto se ejecuta el PID, y por consiguiente debe ser menor o igual al periodo del tren de pulsos.
- **QPOS_P – bool:** Señal de salida de pulsos positiva.
- **QNEG_P – bool:** Señal de salida de pulsos negativa.

9.3 Autotunning en S7.

Como hemos podido observar, existen una serie de parámetros que van a determinar la respuesta de nuestra salida en función de las características de la instalación. Existirán unos valores que nos darán un tipo de respuesta más lenta, y otros una más rápida, decidiendo el usuario que evolución es la más adecuada a su proceso.

Sin embargo, podemos realizar esta adecuación de los parámetros de manera automática. Es lo que se denomina en regulación autotunning de un PID. En un autotunning, durante un periodo de tiempo que puede ser fijo o variable, el regulador va a variar por sí mismo los parámetros de ganancia, tiempo de integración y tiempo derivativo (siempre controlando que la magnitud del proceso no exceda de una determinada banda, para evitar sobreoscilaciones peligrosas en la instalación).

Esta funcionalidad no se encuentra implementada en el software de Step 7, sino en una librería adicional, en forma de dos funciones: autotunning para regulación continua y para regulación por pulsos.

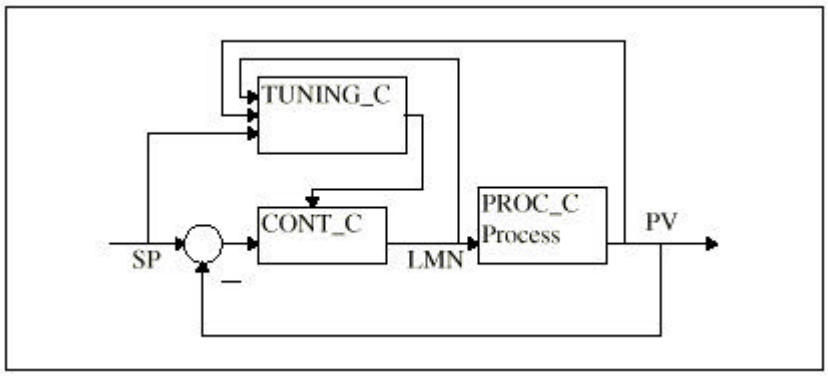
Las FB's son:

- FB39, para regulación continua.
- FB40, para regulación por pulsos.

Deberemos de copiarlas a nuestro proyecto, para poder hacer uso de ellas. Veamos a continuación como utilizarlas para ajustar nuestros PID's en S7.

9.3.1 FB 39. Autotunning para PID continuo.

El esquema de bloques de conexionado de la función con respecto a la FB41 (PID continuo) es el siguiente:



El FB39 Autotunning_C realiza ajustes automáticos de PID's continuos. Los parámetros de entrada de la función son:

Parámetro	Tipo	Significado	Valor	Defecto
SP	REAL	Consigna del PID.	-	0.0
PV	REAL	Valor real del PID.	-	0.0
LMN	REAL	Salida actual del PID.	0-100 (%)	0.0
MIN_STEP	REAL	Mínima valor en la salida del PID que se desea debido a la acción de autotunning.	-	10.0
LHLM_TUN	REAL	Valor máximo de salida del PID debido a la acción de autotunning.	0-100 (%)	80.0
MAN	REAL	Valor de salida del PID en manual.	0-100 (%)	0
MAN_ON	BOOL	PID en manual	-	FALSE
STRUC_ON	BOOL	Control por pasos.	-	TRUE
PID_ON	BOOL	Modo PID activado.	-	TRUE
COM_RST	BOOL	Reset del PID.	-	FALSE
CYCLE	TIME	Tiempo de ciclo del PID.	> 1 ms.	100 ms.

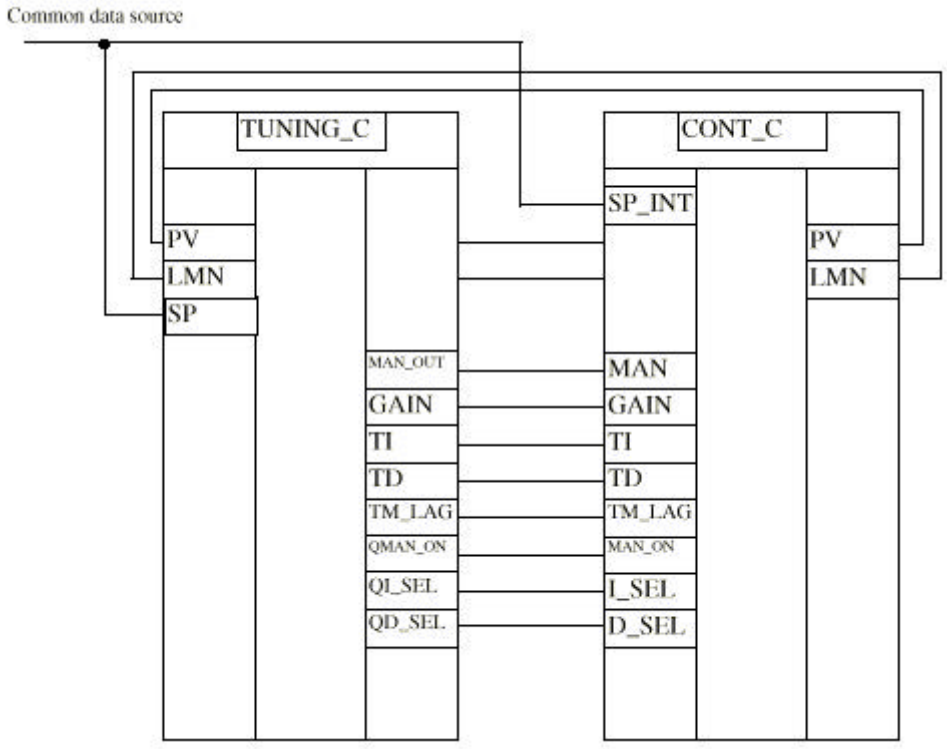
Los parámetros de salida de la función son:

Parámetro	Tipo	Significado	Valor	Defecto
MAN_OUT	REAL	Valor de salida en manual.	-	0.0
GAIN	REAL	Cálculo de ganancia.	-	1.0
TI	TIME	Cálculo de Tiempo de integración.	-	10 s.
TD	TIME	Cálculo de Tiempo derivativo.	-	0
TM_LAG	TIME	Cálculo de Tiempo de retardo	-	1 s.
PHASE	INT	Fase actual del autotunning.	-	0
QP_INFL	BOOL	Punto de inflexión encontrado.	-	FALSE
QMAN_ON	BOOL	Valor manual activado.	-	FALSE
QI_SEL	BOOL	Acción integral activada.	-	TRUE
QD_SEL	BOOL	Acción derivativa seleccionada.	-	FALSE
QWRITE	BOOL	FB39 escribe los parámetros en la salida.	-	FALSE

Por último los parámetros que son tanto de entrada como de salida son:

Parámetro	Tipo	Significado	Defecto
TUN_ON	BOOL		FALSE
ADAPT_ON	BOOL		FALSE
STEADY	BOOL		FALSE

La forma de conectar las dos funciones para que TUNING_C (FB39) regule a CONT_C (FB41) es la siguiente:



La función de autotunning dispone de varios modos de funcionamiento:

- **Modo de inicialización:** en este modo (TUN_ON = TRUE) la función inicializa el ajuste ante un proceso desconocido. Sería el modo para comenzar la regulación de un horno que estuviera apagado, ya que partimos de un gran error, y es necesario alcanzar la consigna de temperatura sin excederse en demasía .
- **Modo de adaptación:** en este modo (ADAPT_ON = TRUE) la función modifica la salida del PID regulado siempre dentro de los límites indicados.
- **STRUCT_ON:**
- **Modo manual:** en él el PID actúa en manual.

Modo de Inicialización:

El modo de inicialización sirve para inicializar el autoajuste del PID. Para entrar en este modo debe de activarse TUN_ON = TRUE. A partir de ese instante, cuando se produzca una diferencia entre el valor de consigna y el real mayor que el valor del parámetro MIN_STEP comenzará la adaptación.

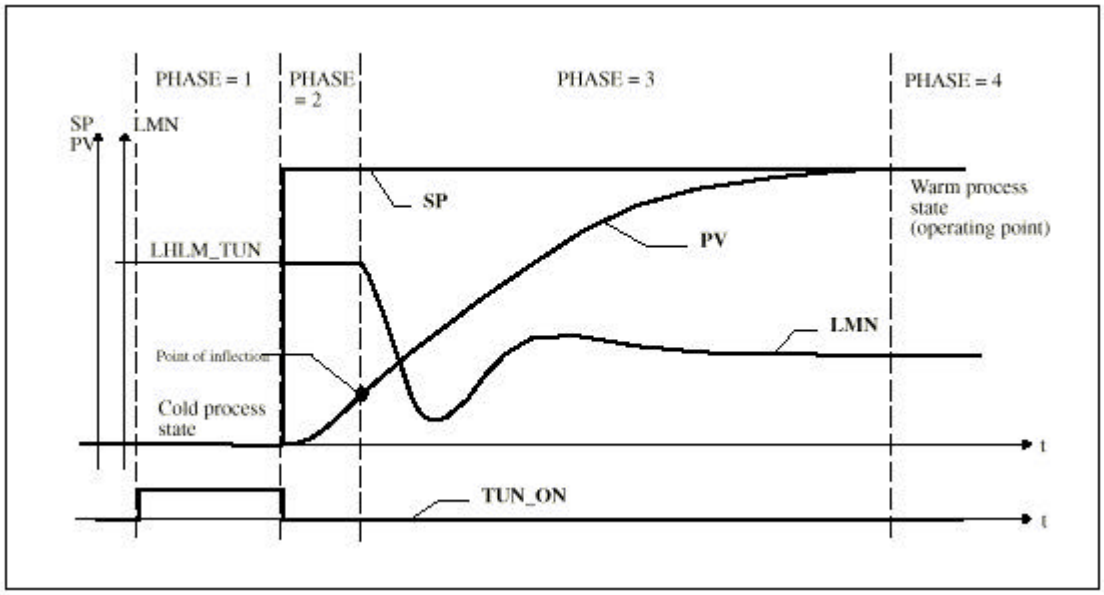


Figura 54 . Modo de inicialización autotunning FB39.

El funcionamiento del proceso de adaptación se basa en encontrar el punto de inflexión que determina que la curva de acercamiento al valor de consigna cambia su pendiente (punto de inflexión en la gráfica). Para ello, el proceso de adaptación se divide en 4 fases:

FASE	ESTADO
FASE 0	Cuando se crea una DB asociada a la FB39, la fase actual es 0.
FASE 1	Al activar TUN_ON, el PID toma un valor de salida de 0, ya que se le aplica una ganancia nula. Es necesario esperar hasta que la función obtenga el valor STEADY, que indica que se puede comenzar el proceso. Se desconecta entonces la señal TUN_ON y el regulador pasa automáticamente a fase 2.
FASE 2	Tan pronto como se aplica en el proceso una consigna cuyo error con respecto al valor actual del proceso es mayor que el parámetro MIN_STEP, se asigna a MAN_OUT el valor LHLM_TUN, y se activa QMAN_ON. MIN_STEP debería ser mayor que el 10% del valor máximo del fondo de escala de la consigna. Cuando se encuentra el punto de inflexión el regulador pasa automáticamente a fase 3.
FASE 3	Cuando el punto de inflexión es detectado por la función, QP_INFL=TRUE, o la variable de proceso ha alcanzado el 60% de la consigna, se aplica unos parámetros de control PI cautelosos y se comienza a observar la evolución del proceso con los mismos. El ajuste intenta llevar al proceso a un estado de consigna (STEADY = TRUE). Si en un momento de la adaptación se desean dar los parámetros calculados en ese momento como definitivos, se puede activar STEADY para pasar de fase. Si durante el proceso de aprendizaje se produce una sobreoscilación, o no se alcanza el punto de inflexión, probablemente es debido a que se ha seleccionado un valor demasiado alto en LHLM_TUN. Rearranque el proceso de aprendizaje (p. Ej. Pasando a manual, y posteriormente activando de nuevo TUN_ON), y esta vez introduzca un nuevo valor en LHLM_TUN (un 20% aprox. menor). Si la función detecta un estado estable (STEADY), pasa automáticamente a fase 4. Si se ha seleccionado PID_ON = TRUE, se aplica un regulador PID. En caso contrario, se aplica un regulador PI. El valor de ganancia que la función FB39 puede aplicar a la FB41 va desde 0.4 a 15.
FASE 4	El PID actuará a partir de este instante con los parámetros calculados, y sin modificarlos.

Modo de adaptación:

Si ya se ha realizado alguna vez una inicialización en la instalación, y la misma se encuentra funcionando con los parámetros que se calcularon en aquel momento, puede ser interesante en determinadas situaciones realizar ajustes sobre los mismos, para que el regulador se ajuste mucho mejor al funcionamiento actual del sistema.

Para estos menesteres es más apropiado el modo de manipulación, que se muestra en la figura:

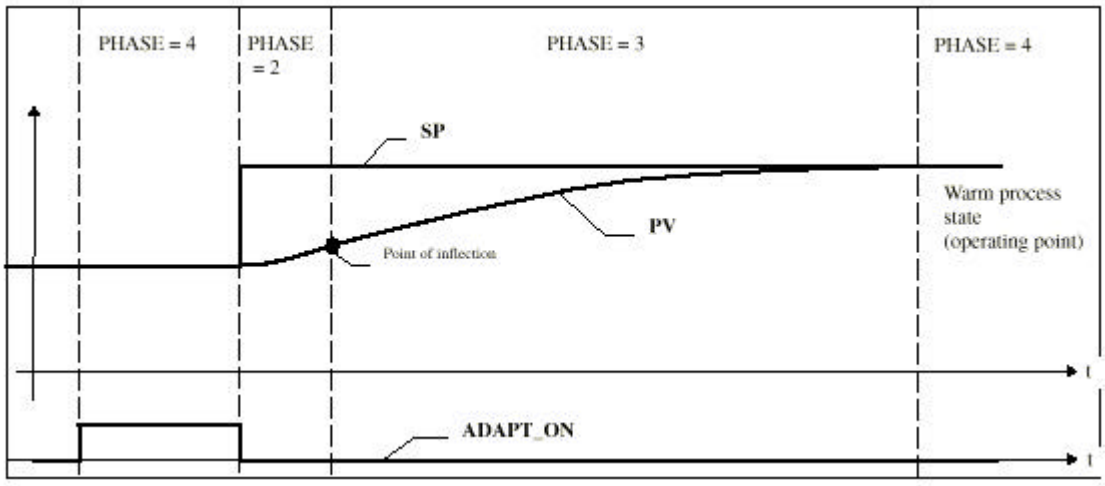


Figura 55 . Modo de adaptación autotunning FB39.

10

Comunicaciones

10.1 Generalidades

10.1.1 Protocolos y soporte físico.

Con frecuencia se suele confundir en las explicaciones técnicas de los especialistas los soportes físicos de las comunicaciones con los protocolos que soportan las mismas. Vamos a intentar en este apartado explicar aquellos que conciernen a las comunicaciones relacionadas con el mundo de Simatic S7.

Se entiende por protocolo según los manuales: “el convenio exacto al bit entre interlocutores para poder ejecutar un determinado servicio de comunicación. El protocolo define el contenido estructural del tráfico de datos en la línea física, definiendo p. ej. el modo de operación, la forma de realizar el establecimiento del enlace, la protección de los datos o la velocidad de transferencia.”

En otras palabras, el **protocolo** es el lenguaje que se utiliza en la comunicación para que se entiendan dos equipos a la hora de comunicar.

Por otro lado, se entiende por soporte de transmisión o **soporte físico** “el medio físico sobre el que se transmite el protocolo de comunicaciones”. Es decir, es por donde circula dicho lenguaje o telegrama de comunicaciones.

Si realizáramos una analogía entre estos términos y una autopista, diríamos que el protocolo serían el tipo de coches (2 blancos y 1 negro seguidos) que circulan por ella, mientras que el soporte físico el tipo de autopista (de 4 carriles, con barreras laterales, etc...).

10.1.2 Tipos de soportes físicos.

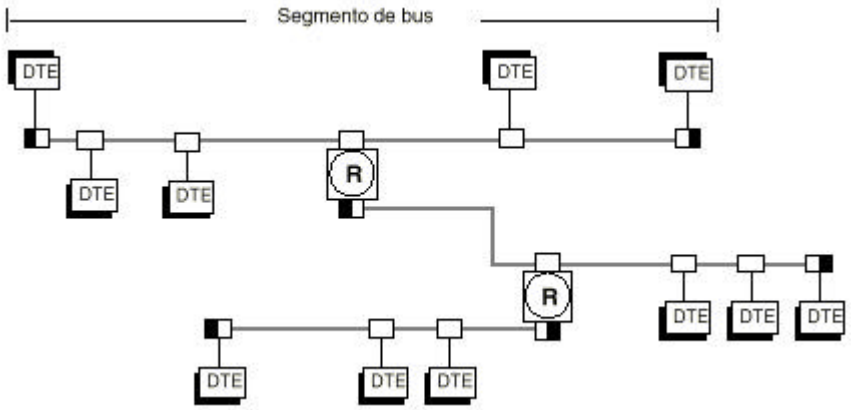
En el mundo de S7 se trabaja con tres tipos de soportes físicos para la comunicación entre equipos:

- **Cable bifilar[1] no trenzado, sin apantallar:** es el utilizado en el bus de comunicaciones AS-i. El cable de AS-i es de mayor grosor que los demás soportes, debido a que por sus características especiales soporta no solo el protocolo de comunicaciones sino que alimenta a través de sí a la electrónica de los equipos esclavos con los que comunica el maestro. El hecho de que no se trence y que no esté apantallado se debe a que las interferencias electromagnéticas no afectan a la red AS-i. Esto es así, debido a que aunque un campo electromagnético genere una fluctuación en la señal que transporta el cable AS-i, al afectar a ambos conductores de la misma forma, el diferencial entre la tensión de los dos conductores seguirá invariable. Como el protocolo As-i se transmite como un rizado sobre una señal continua (la alimentación a los módulos esclavos), no se ve afectado por estas variaciones comunes a los dos conductores del cable.
- **cable bifilar trenzado y apantallado:** es el utilizado en la comunicación PROFIBUS en sus diferentes variantes y en MPI.
- **fibra óptica:** puede ser utilizada para PROFIBUS o para ETHERNET, aunque es más usual en este último protocolo. La fibra a su vez puede ser de vidrio o de plástico, dependiendo de la distancia que separa a los equipos a enlazar. La fibra de plástico alcanza una distancia máxima de 80 m. entre interlocutores, por lo que se suele utilizar la fibra de vidrio, con la cual utilizando repetidores se pueden llegar a salvar grandes distancias de decenas de kilómetros. Tiene el inconveniente la fibra de vidrio frente a la de plástico e que debe de ser manipulada mediante herramientas especiales, lo cual dificulta su manipulación y hace recomendable adquirir cables preconfeccionados que ya posean los terminales de conexión. Esta solución tiene el inconveniente de que es necesario conocer exactamente la distancia entre elementos para solicitar dicho cable.

10.1.3 La norma RS485.

La mayoría de los protocolos que más adelante describiremos, utilizan como soporte físico el cable trenzado y apantallado siguiendo la norma RS485. Esta norma se basa en una diferencia de tensión entre ambos conductores, por lo que es bastante inmune al ruido debido a que una elevación en la tensión de ambos conductores no afecta a la señal transmitida.

[1] Bifilar: dos hilos.



Los cables de comunicaciones en esta norma se denominan señal A y señal B, realizándose la conexión de los elementos mediante conexiones en paralelo. Las derivaciones en estrella son posibles mediante repetidores de comunicaciones.

En los extremos de la red es necesario que se conecten unas resistencias de finalización del bus de datos.

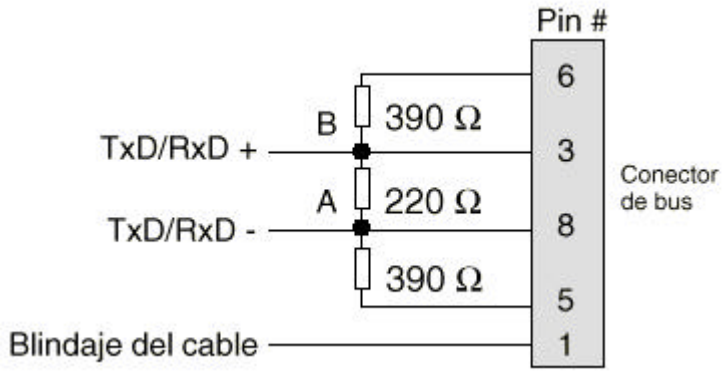


Figura 56 . Resistencias de finalización RS485.

Se suele llamar a esta norma el soporte físico de los protocolos Profibús DP, FMS, FDL, Datos globales y MPI en sus conexiones eléctricas, aunque es más la norma del soporte que el propio soporte en sí.

A continuación estudiaremos el cableado de los conectores del Simatic S7 200 y S7 300.

Pin	Denominación PROFIBUS	Interfaces 0 y 1	Interface DP
1	Blindaje	Hilo lógico	Hilo lógico
2	Hilo de retorno 24 V	Hilo lógico	Hilo lógico
3	Señal B RS-485	Señal B RS-485	Señal B RS-485
4	Request-to-Send	Sin conexión	Request-to-send[1]
5	Hilo de retorno 5 V	Hilo lógico	Isolated +5 V Return[2]
6	+5 V	+5 V, 100 Ω series limit	+5 V, con separación galvánica, 90 mA
7	+24 V	+24 V	+24 V
8	Señal A RS-485	Señal A RS-485	Señal A RS-485
9	No aplicable	Sin conexión	Sin conexión
Carcasa del enchufe	Blindaje	Hilo lógico (CPU 212/214) Tierra (CPU 215/216)	Tierra

Figura 57 . Conector S7 200.

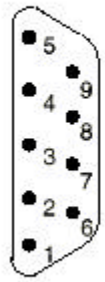
Vista	Pin No.	Nombre de señal	Denominación
	1	–	–
	2	M24V	Masa 24 V
	3	RxD/TxD-P	Circuito de datos B
	4	RTS	Request To Send
	5	M5V2	Potencial de referencia de datos (de la estación)
	6	P5V2	Polo positivo de alimentación (de la estación)
	7	P24V	24 V
	8	RxD/TxD-N	Circuito de datos A
	9	–	–

Figura 58 . Conector S7 300.

Las velocidades y las distancias de transmisión en esta norma están relacionadas. Así, para la comunicación MPI sobre RS485, las distancias/velocidades son:

- Sin amplificar la señal MPI: 50 m. (salvo CPU 318-2 DP que son 100 m.)
- Con cada repetidor RS485: 1000 m. a 187 Kbaud.

Para la comunicación en Profibús en sus distintas modalidades, la relación velocidad/distancia es:

- 100 m. a 12 Mbaud.

- 200 m. a 1.5 Mbaud.
- 400 m. a 500 Kbaud.
- 1000 m. a 187 Kbaud.

Estas distancias se duplican por repetidor. Con respecto a los repetidores tener en cuenta que la amplificación de la señal se realiza en ambos extremos, de tal manera que si colocamos un repetidor en una red, p. Ej. MPI, pasaremos a tener una distancia de 500 m. por el lado derecho del repetidor y otros 500 m. por el lado izquierdo. Es muy importante tener este dato en cuenta, ya que no siempre es posible colocar un repetidor en el centro del segmento de comunicaciones, además de que es necesario alimentarlo a 24 Vdc, por lo que se requiere una fuente de continua en las inmediaciones del repetidor.

10.1.4 Tipos de transmisión de datos.

El criterio a la hora de transmitir la información a través de la red de comunicaciones también establece una diferenciación clara entre las existentes, y es sin duda la que más caracteriza la forma de comportarse la red en la instalación.

Existen una primera división que se compone de:

- Redes maestro-esclavo y
- Redes multi-maestro.

Redes maestro-esclavo:

Las redes maestro-esclavo se caracterizan por poseer un único maestro y todos los demás equipos son esclavos. Se entiende por equipo maestro aquel que toma el control del bus de comunicaciones (con exclusividad) y emite al mismo un telegrama por iniciativa propia, y sin que se le haya solicitado previamente dicha acción. Se entiende por esclavo aquel equipo que es capaz de emitir un telegrama a la red si recibe previamente una petición por parte del maestro dirigida al mismo, pero que no puede solicitar datos a ningún equipo de la red por iniciativa propia. Esto actualmente empieza a no ser del todo cierto, ya que existe la posibilidad de intercambiar información entre esclavos sin necesidad de pasar dicha información a través del maestro en profibús-DP, pero es una particularidad que actualmente está en desarrollo en Simatic S7.

En el método maestro-esclavo, el maestro siempre es el mismo equipo. Debido a esta circunstancia, en una red maestro esclavo sólo hay un maestro, y los demás equipos son incapaces de comunicar si por cualquier circunstancia se interrumpe la comunicación entre los mismos y el maestro. Esta es sin duda la peor característica de este sistema, ya que condiciona las comunicaciones al funcionamiento de un solo equipo. En contrapartida, al siempre disponer el mismo equipo de las comunicaciones, la transferencia de información mediante este sistema es mucho mas rápida, por lo que está indicado en

aquellos procesos que requieren intercambiar información en tiempos críticos (periferia descentralizada).

Redes multimaestro:

Para evitar el inconveniente anterior aparecen las redes en las cuales todos los equipos pueden tomar la iniciativa de la comunicación en la red. Pero dentro de esta posibilidad existen también dos subdivisiones:

- Redes Token-Passing, o de paso por testigo y
- Redes CSMA/CD, o de acceso aleatorio.

Esta subdivisión es necesaria debido a que en un determinado instante, solo uno de los equipos puede emitir un telegrama a la red, y hasta que dicho telegrama sea acusado por el equipo receptor, ningún otro equipo debe de emitir. Esto obliga a establecer un criterio en el tráfico de la información por el soporte físico. Y aquí aparecen las dos posibilidades anteriormente mencionadas.

En las redes de paso por testigo (PROFIBUS FMS o FDL), el token o testigo es la autorización para poder disponer de la red de comunicaciones y emitir las peticiones o los envíos que el equipo en cuestión desee. Mientras una CPU dispone del token, las demás se comportan como esclavas, por lo que exclusivamente podrán responder ante una petición del maestro actual. Cada cierto tiempo determinado por el sistema, el testigo pasa al siguiente equipo que esté configurado como maestro. El orden de paso no viene determinado por la posición física de los equipos en la red, sino por la ordenación numérica de sus número de estación dentro de la misma. Generalmente una red de paso por testigo es en anillo (token-ring), por lo que cuando se alcanza el último maestro, vuelve el sistema a pasárselo al primero de los mismos de nuevo y así sucesivamente. Si uno de los equipos que deben de tomar el testigo no responde, el sistema automáticamente lo pasa al siguiente en la lista de autorizaciones.

En las redes de acceso aleatorio, este paso por testigo no existe en absoluto. En este método, todas las estaciones pueden enviar en cualquier instante siempre que no emita ninguna en dicho momento. Lógicamente, esto genera a la larga conflictos condicionados por tiempos de propagación cuando dos estaciones intentan emitir al mismo tiempo por haber detectado que está libre el bus. En este método, ambas estaciones detectan la colisión, por lo que dejan de emitir, y sólo reemprenden ésta después de esperar un tiempo definido aleatoriamente, con lo que la probabilidad de colisionar de nuevo es muy baja. En cualquier caso, si esto ocurriera, se repetiría el proceso anterior. Este método es el utilizado por ETHERNET para comunicar entre sus equipos.

10.2 La norma ISO

Visto lo anterior, y todas las posibilidades que existen a la hora de definir una red de comunicaciones, con divisiones y subdivisiones por diferentes materias, no es de extrañar que se hiciera imprescindible una norma internacional que permitiera determinar mediante una escala que equipos son capaces de comunicar entre sí, independientemente del fabricante que los fabrica.

A esta norma se le denominó ISO[2], que es la organización que la generó. En la misma se establecen un total de siete niveles o capas, entendiendo por nivel una característica que deben cumplir ambos equipos que desean comunicar. Si un equipo cumple un determinado nivel, no necesariamente debe de poder cumplir los niveles inferiores.

Así, para lograr un entendimiento suficiente y seguro se precisan imprescindiblemente
los niveles 1, 2 y 4.

Veamos qué significa cada nivel:

- **Nivel 1: (físico).** Este nivel o capa procura la transmisión transparente de bits a través del soporte físico en el orden definido por el nivel de enlace (2). Aquí se definen las características eléctricas y mecánicas así como los tipos de transmisión.
- **Nivel 2: (enlace).** Este nivel tiene como misión asegurar la transmisión de la cadena de bits entre dos sistemas. Entre sus misiones figura detectar y eliminar o comunicar errores de transmisión y el control del flujo. En redes locales, el nivel de enlace procura también el acceso exclusivo al soporte de transmisión. Para ello, dicho nivel se divide en dos subniveles, medium Access Control (MAC) y Logic Link Control (LLC), que se designan también como niveles 2a y 2b. Las normas más conocidas para los métodos de acceso aplicados en el subnivel MAC son: IEEE 802.3 (ETHERNET, CSMA/CD), IEEE 802.4 (Token Bus), IEEE 802.5 (Token Ring). Para el subnivel LLC se aplica generalmente la norma IEEE 802.2. En base a las características de tiempo real exigidas normalmente a sistemas de bus de campo, éstos utilizan en parte métodos de acceso considerablemente modificados.
- **Nivel 3: (red):** Este nivel se encarga de la intercomunicación de datos entre sistemas finales. Como sistemas finales se considera el emisor y receptor de una información cuyo recorrido puede llevar bajo circunstancias a través de diversos sistemas de tránsito. Por ello, el nivel de red debe seleccionar la ruta ha seguir lo que normalmente

[2] ISO: International Standard Organization. Organización internacional con sede en Ginebra encargada de crear normas de validez en el sector de las comunicaciones internacionales.

se denomina encaminamiento (Routing).

Nivel	Designación	Función	Características
7	**Application layer** (aplicación)	Funciones de usuario Oferta de servicios de comunicación específicos de usuario	Servicios de comunicación p. ej. Read/Write Start/Stop
6	**Presentation layer** (presentación)	Represent. de datos Conversión del tipo de representación normalizado del sistema de comunicación en un formato adecuado al equipo	Lenguaje común
5	**Session layer** (sesión)	Sincronización Establecimiento, disolución y vigilancia de una sesión	Coordinación de la sesión
4	**Transport layer** (transporte)	Establecimiento/disolución de enlace Repet.de paquetes, clasificación de paquetes, formación de paquetes	Transmisión asegurada de paquetes
3	**Network layer** (red)	Direccionamiento de otras redes (encaminamiento) (Routing), control de flujo	Comunicación entre dos subredes
2	**Data link layer** (enlace)	Método de acceso Limitación de los bloques de datos, transmisión asegurada, detección y eliminación de errores	CRC-Check CSMA/CD Token
1	**Physical layer** (físico)	Características físicas, soporte de transmisión, velocidad, definición de los parámetros eléctricos, mecánicos y funcionales de la línea/bus	Cable coax/triax cable óptico cable bifilar

Figura 59 . Niveles ISO.

- **Nivel 4: (transporte):** Este nivel tiene como misión ofrecer al usuario un enlace terminal-terminal fiable. Los servicios ofrecidos incluyen el establecimiento del enlace de transporte, la transmisión de datos así como la disolución del enlace. Para ello el usuario puede exigir en general una determinada calidad en el servicio (QoS, Quality of Service). Parámetros de calidad son por ejemplo la velocidad de transferencia y la tasa de errores residuales.
- **Nivel 5: (sesión).** La tarea principal del nivel de sesión es sincronizar las relaciones de comunicación. Además, los servicios del nivel de sesión permiten definir puntos de sincronización en transmisiones prolongadas para que, en caso de una interrupción

intempestiva del enlace, no sea necesario repetir de nuevo toda la transmisión sino que pueda restablecerse desde un determinado punto de sincronización.

- **Nivel 6: (presentación).** Generalmente, al intercambiar datos diferentes sistemas utilizan lenguajes diferentes. El nivel de presentación traduce los diferentes lenguajes de las estaciones de comunicación a un lenguaje unificado con una sintaxis abstracta. Para ello se utiliza en la mayor parte de los casos el Abstract Syntax Notation One (ASN.1) definido en ISO 8824 y las Basic Encoding Rules (BER) asociadas.
- **Nivel 7: (aplicación).** El nivel de aplicación comprende los servicios específicos de la apli-cación de las diferentes aplicaciones de comunicación. Como existen multitud de aplicaciones es particularmente difícil establecer estándares unificados. El estándar más importante para aplicaciones de automatización es el Manufacturing Message Specification (MMS), que describe los servicios y protocolos del nivel de aplicación (MAP, Manu-facturing Automation Protocol). Los sistemas de bus de campo modernos se orientan fuertemente en MMS a la hora de diseñar el nivel de aplicación.

10.3 Protocolos de comunicaciones en S7.

10.3.1 Recursos en S7

Las configuraciones de las posibles comunicaciones en S7 vienen determinadas generalmente por los “recursos” de la CPU. Un recurso es la posibilidad de realizar una comunicación mediante funciones S7 por parte de la CPU. Como veremos posteriormente, el tipo de comunicaciones más sencillo con otros equipos (PC’s, OP’s, PLC’s) va a ser mediante funciones S7. Es por esto importante conocer de cuantos recursos disponemos en nuestra CPU, para saber si podemos realizar una configuración determinada de comunicaciones.

Recursos en S7 300, disponemos de 4, mientras que en S7 400 disponemos de 16. Pero, ¿qué tipo de comunicaciones nos va a consumir recurso, y cuales no?. La tabla siguiente ilustra las diferentes posibilidades referidas a un S7 300.

EQUIPO	TIPO COMUNICACIÓN		¿CONSUME RECURSO?
PG/PC	MPI		SI
	DP		SI
OP	MPI		SI
	DP		SI
PLC	DATOS GLOBALES		SI
	PROFIBUS	FDL	NO
		FMS	SI
		FUNCIONES S7	SI
	ETHERNET	SEND/REC	NO
		TCP/IP	NO
FM (350, 351,...)	TODAS		SI
WINCC	MPI		SI
	DP MAESTRO		NO
	Profibús FDL		NO
	Profibús funciones S7		SI
	Profibús FMS		SI
CP´S	CP 342-2 (AS-i)		NOI
	CP 342-5	DP	NO
		FUNCIONES S7	SI
		FDL	NO

Lógicamente, para las comunicaciones entre PLC´s interesará utilizar Profibús FDL, que no consume recursos, mientras que para la comunicación entre PLC y ordenador o OP será interesante utilizar el protocolo Funciones S7, ya sea por MPI o por Profibús, ya que gestionará todas las comunicaciones de manera transparente para nosotros desde el equipo de visualización, pese a que consumimos un recurso o enlace. Para una gran cantidad de ordenadores, se recomienda TCP/IP.

10.3.2 Protocolos soportados por redes.

Un protocolo de comunicaciones determina el telegrama a transmitir en la red de comunicaciones. Por lo tanto, es el "lenguaje" con el que dialogan los equipos. Lógicamente, como pasa en la lengua hablada, no tiene porqué existir en un determinado lenguaje un servicio que sí que existe en otro, al igual que un refrán en un idioma puede no tener traducción en otro distinto. A estas posibilidades se les denominan prestaciones de los servicios de comunicaciones. Las diferentes redes que existen en S7 son:

- AS-i
- MPI
- Profibús y
- ETHERNET.

Pues bien, veamos que protocolos o servicios soportan cada una de estas redes de comunicaciones.

- La red AS-i únicamente soporta el protocolo AS-i.
- La red MPI la comunicación por datos globales (GD) y la comunicación mediante funciones S7.
- La red Profibús soporta la comunicación mediante Profibús DP, Profibús FDL, Profibús FMS y funciones S7.
- La red ETHERNET soporta la comunicación mediante protocolo TCP/IP, ISO-Transport y funciones S7.

Veamos en profundidad que realizan cada uno de estos protocolos, y cual es su finalidad, sus virtudes y sus limitaciones.

10.3.3 Servicios AS-i

El protocolo de comunicaciones AS-i está diseñado para la comunicación entre un equipo maestro y actuadores o sensores de campo (detectores, electroválvulas, etc...).

Este planteamiento inicial ha dado paso mediante una evolución lógica a una red maestro-esclavo con todas las posibilidades de comunicaciones y diagnosis mínimas que requiere un sistema de comunicaciones. Actualmente es posible incluso la lectura de valores analógicos a través de la misma, pese a que el protocolo fue concebido para la optimización de la lectura de valores digitales. La cantidad de participantes en una red AS-i es de 31 equipos. El tiempo de ciclo máximo de la red AS-i (tiempo que tarda en volver a leerse el estado de un esclavo o escribirse en el mismo) es inferior o igual a 5 ms, lo cual la convierte en el sistema más rápido de periferia descentralizada e ideal para estas tareas si la distancia como veremos lo permite.

Su programación es muy sencilla: se trabaja sobre una zona de bytes asignados a la tarjeta maestra AS-i. Cada esclavo dispone de un byte, del cual 4 bits son para entradas y los cuatro siguientes para salidas.

La velocidad de transferencia es de 167 Kbaud.

10.3.4 Comunicación por datos globales (GD).

- Soporte físico: RS485.
- Protocolo: Funciones S7.

Es el tipo de comunicación más sencilla que existe entre equipos S7, y la más económica, ya que no requiere a diferencia de las otras comunicaciones (salvo MPI) requiere de una tarjeta adicional (CP).

En la comunicación con datos globales podemos transferir o acceder a cualquier zona del mapa de memoria del PLC con el que deseemos comunicarnos, y tan sólo es necesario indicar donde queremos que se nos dejen los datos leídos o cual es la fuente de datos a enviar.

Los inconvenientes de este sistema son:

- Cada comunicación mediante datos globales exige un enlace configurado, por lo que si queremos comunicar un equipo S7 300, como un enlace es para la programadora, solo disponemos de 3 equipos a comunicar mediante este sistema.
- El tamaño máximo de los datos globales entre las CPU's que están comunicando sólo puede ser de 88 bytes. Aunque se indica que sólo se pueden utilizar 22 bytes entre dos CPU's, si solo vamos a comunicar 2 equipos en total se pueden utilizar los 88 bytes para este enlace, pero siempre agrupándolos en paquetes de 22 bytes máximo (más adelante veremos que esta barrera se puede superar).

	Identificador GD	314\ CPU314IFM	314 2\ CPU314IFM
1	GD 1.1.1	DB10.DBB0:22	>DB10.DBB0:22
2	GD 2.1.1	DB11.DBB0:22	>DB11.DBB0:22
3	GD 3.1.1	DB12.DBB0:22	>DB12.DBB0:22
4	GD 4.1.1	DB13.DBB0:22	>DB13.DBB0:22
5	GD 5.1.1		

Figura 60 . Dos CPU's comunicando 88 bytes entre ellas.

La velocidad de comunicación de una red MPI es fija a 187 Kbaud. La comunicación MPI soporta una longitud de segmento total de 50 m. sin repetidor. Recordar que a una velocidad de 187 Kbaud, cada repetidor amplificará la red MPI en 1000 m., siempre y cuando el repetidor esté colocado en el centro del segmento a amplificar.

El número máximo de equipos en una red MPI es de 32 participantes, aunque recordemos que cada uno podrá dialogar como máximo con 3 de ellos.

10.3.5 Servicio de Funciones S7.

El protocolo de funciones S7 es el ideal para comunicar equipos Simatic S7 entre ellos, ya que al estar implementado en el sistema de las tarjetas, únicamente es necesario establecer el enlace entre el equipo que se desea comunicar con su respectivo y que información es necesario compartir.

Uno de los inconvenientes de este sistema de comunicaciones es que lógicamente todos los participantes en la comunicación deben de ser equipos S7. Además de esto, debe de crearse un enlace entre los dos equipos a comunicar.

Un enlace de funciones S7 determina una relación unívoca entre dos equipos para comunicar la información entre ellos. Estos enlaces de S7 pueden ser de dos tipos:

- **Enlaces configurados:** se establecen al transferir los datos de sistema, y son fijos.
- **Enlaces no configurados:** se establecen al llamar la CPU a una SFC del sistema que realiza el enlace entre equipos. Dicho enlace no es fijo, por lo que otra llamada a la SFC con diferente equipo destino puede deshacerlo y generar uno nuevo con otro equipo. El tamaño máximo de los datos a transmitir mediante este método es de 76 bytes. Es importante destacar que aunque no esté configurado, este método gasta un enlace libre, pese a que solo se utilice de manera temporal.

Evidentemente lo más cómodo sería establecer enlaces configurados entre equipos S7 a la hora de proyectar la instalación, ya que no requieren programación adicional en lo que se considera propiamente el programa de PLC (ausencia de llamadas a subrutinas).

Si embargo, las CPU's de S7 tienen limitados el número de enlaces configurados para no sobrecargar con las comunicaciones el microprocesador del PLC y permitirle realizar el código contenido en las diferentes OB's, FC's y FB's del programa en un tiempo de ciclo suficientemente rápido.

La cantidad de enlaces configurados que soportan las CPU's de S7 son:

- Los S7 300 soportan 4 enlaces configurados, de los cuales 1 está reservado para la programación del PLC a través del Step 7, ya sea desde la maleta de programación o desde un portátil con el cable PC/MPI.
- Los S7 400 soportan 16 enlaces configurados, de los cuales igual que anteriormente 1 debe de ser para el equipo de programación.

10.3.6 Protocolo Profibús DP

El protocolo Profibús DP es el más utilizado de todos los tipos de comunicaciones en S7 ya que permite de una manera sencilla a nivel técnico y ajustada en precio realizar una descentralización de las señales de campo, con el consiguiente ahorro en cableado y horas de montaje en campo. Se corresponde con la norma europea EN 50170 Vol. 2, PROFIBUS.

El número máximo de participantes es de 127.

10.3.7 Protocolo Profibús FDL.

En SIMATIC S7, los servicios FDL se utilizan para comunicarse usando los bloques AG_SEND y AG_RECV a través de la subred PROFIBUS. FDL se corresponde con la norma europea EN 50 170 Vol.2 PROFIBUS. Con el servicio FDL, el SIMATIC S7 ofrece funciones de comunicación para emitir y recibir datos a través de enlaces estáticos. Pero a diferencia de la comunicación con funciones S7, es posible realizar configuraciones mixtas entre equipos S5 y S7, siempre que todos estén equipados con tarjetas que soporten este protocolo.

El número máximo de participantes es de 127.

10.3.8 Protocolo Profibús FMS.

El protocolo PROFIBUS-FMS (Fieldbus Message Specification) ofrece servicios para transmitir datos estructurados (variables FMS). El servicio FMS puede clasificarse en

el nivel 7 del modelo de referencia ISO. Se corresponde con la norma europea EN 50170 Vol.2 PROFIBUS y permite así una comunicación abierta con equipos no Siemens (terceros).

El número máximo de participantes es de 127.

10.3.9 Protocolo ISO-Transport.

ISO-Transport ofrece servicios para transmitir datos a través de enlaces. Dicho servicio vigila automáticamente el enlace. El servicio ISO-Transport (ISO 8073 clase 4) se corresponde con el nivel 4 del modelo de referencia ISO. Como es posible segmentar los datos en diferentes telegramas, el servicio ISO-Transport permite transmitir grandes cantidades de datos.

El número máximo de participantes es superior a 1000.

10.3.10 Protocolo TCP/IP

El servicio ISO-on-TCP se corresponde con el estándar TCP/IP (Transmission Control Protocol/Internet Protocol) incluyendo la ampliación RFC 1006 de acuerdo al nivel 4 del modelo de referencia ISO.

Los servicios ISO-on-TCP se utilizan en SIMATIC S7 para comunicarse con los bloques AG_SEND y AG_RECV vía la subred Industrial ETHERNET (v. cap. 4).

El número máximo de participantes es superior a 1000.

10.4 Comunicación con Simatic S7 200.

Existen dos formas de comunicarse con una CPU 200 desde un 300: mediante Profibús-DP, o a través de funciones X_GET y X_PUT (puerto de comunicaciones MPI). Vamos a estudiar las dos opciones, aunque en equipos anteriores a la serie 22x de los 200 no es interesante la segunda opción, ya que obliga a reducir la velocidad de transferencia MPI a 19200 baud.

10.4.1 Comunicación con S7 215-DP.

La gama de productos Simatic S7 200 posee una CPU, la 215-DP, que posee un interface para comunicación según norma Profibús DP. Esta interface es la 1, con una velocidad de transferencia parametrizable entre 9600 baudios y 12 Mbaud. El equipo 215 se comporta en calidad de esclavo en cualquier caso, por lo que será siempre el equipo S7 300 el que realizará las peticiones de envío o recepción de comunicaciones. Es debido a esto por lo que, pese a que existan varias CPU's 215 en una red DP, no será posible intercambiar información entre ellas directamente, siendo necesario pasar esta información a la CPU maestra (en este caso S7 300) para reenviarla al segundo esclavo S7 200, con el consiguiente peligro de caída del maestro y pérdida de comunicación entre equipos.

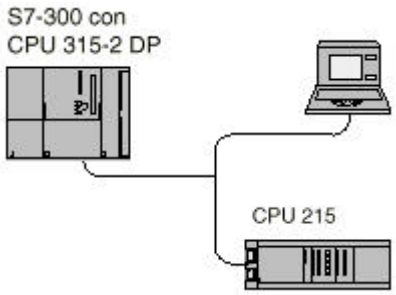


Figura 61 . Esquema de configuración 300-200 por DP.

Desde el punto de vista del maestro, CPU 300, la información a recibir se almacena en la periferia de entradas (generalmente en la zona de las analógicas), mientras que la información a enviar se guarda en la zona de las salidas (también suele ser en las direcciones analógicas).

La cantidad de información máxima que se puede transmitir mediante este sistema es de 64 bytes de entradas y 64 bytes de salidas. La velocidad de transferencia y distancias máximas rigen de manera idéntica a cualquier red Profibús.

Es importante destacar el concepto de coherencia en los búfers de envío y recepción. No siempre todas la información puede ser enviada en un solo telegrama, por lo que aparece el concepto de coherencia en comunicación. Se denomina información coherente a aquella que llega del emisor, o es enviada al emisor al mismo tiempo. Este dato puede llegar a se importante, ya que si no tenemos en cuenta la coherencia en la información recibida del 215, p. Ej., podemos estar leyendo un valor analógico compuesto por 4 bytes, de los cuales 2 llegan en un determinado instante y los dos siguientes más tarde, falseándose la señal recibida durante un determinado instante de tiempo, en el cual la CPU del 300 está procesando dicho valor mezclado con el antiguo dato. En este tipo de comunicación que ahora nos ocupa, se cumple la coherencia en los siguientes términos:

- Coherencia del búfer completo hasta 16 bytes de entrada y 16 de salida.
- Coherencia a nivel de palabras hasta 32 palabras de entradas y 32 de salidas.

Será necesario seleccionar coherencia de búfers si se van a transferir números reales o dobles enteros. En cualquier otro caso, con coherencia de palabras será suficiente.

Para realizar este tipo de comunicación, vamos a realizar primeramente la configuración de la comunicación desde el punto de vista del 315-2 DP. Para ello, basta con seleccionar en el catálogo *hardware ->profibus-dp->simatic->s7 200 cpu 215-2 dp*

Posteriormente, es necesario indicar el tamaño de los búfers de entradas y salidas, para los cuales el Step 7 asignará automáticamente la periferia de la que disponga libre según la configuración existente en ese instante.

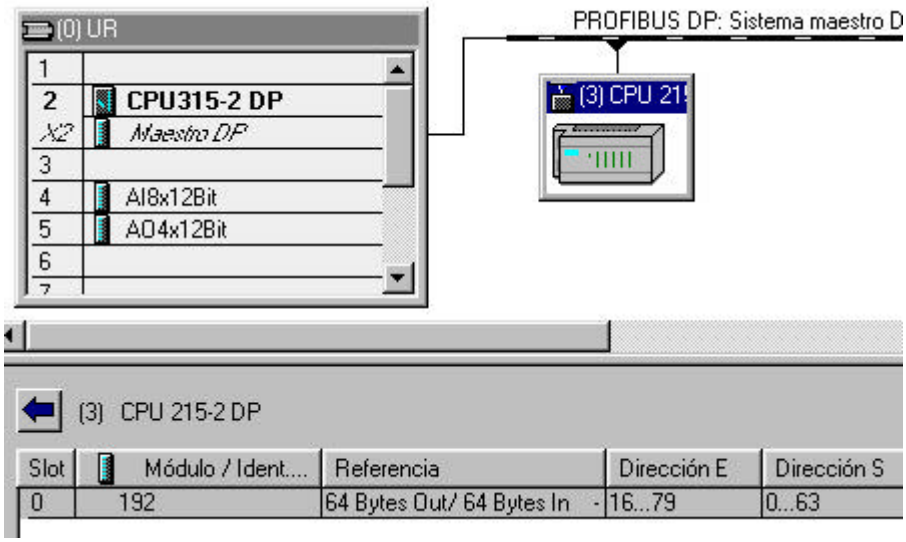


Figura 62 . Asignación de zonas de periferia del 300 para comunicar con el 200 desde Hardware de Step 7.

Esta configuración se transfiere a la CPU del 300 y con esto se finaliza la programación de la comunicación por la parte del S7 300.

Con respecto al 200, existen una serie de marcas especiales, desde la SMB110 a la SMB115, que se encargan de realizar la configuración de la comunicación DP.

La función de cada una de ellas es:

- **SMB 110. Palabra de estado de la comunicación DP.** Sólo los primeros 2 bits de la palabra se utilizan, permaneciendo los restantes siempre a cero. El significado de los mismos es:
  00 : comunicación DP no inicializada desde el arranque.
  01 : error de configuración o de parametrización.
  10 : intercambio de datos activado.
  11 : intercambio de datos desactivado.

- **SMB111 . Dirección del esclavo DP.** El valor de este byte le indica a la tarjeta profibús cual es su número de estación dentro de la red DP.

- **SMW112 . Dirección de la memoria del búfer de salidas.** Esta palabra indica a la tarjeta donde comienza el buzón de recepción. El buzón de envío lo calcula automáticamente, ya que viene a continuación del de recepción, es decir, SMW112 + cantidad de bytes a enviar = byte de la zona de variables donde empieza el buzón de envío.

- **SMB114 . Tamaño del búfer de salida (en bytes).**
- **SMB115 . Tamaño del búfer de entradas (en bytes).**

El funcionamiento de la comunicación entre el 200 y el 300 es el siguiente. Cuando se conecta el 200 a la red, una vez parametrizados estos bytes de sistema, se establece el modo de intercambio de datos entre los equipos de manera automática. Si tiene éxito esta inicialización de las comunicaciones entre los equipos, el piloto DP del 215 luce en verde y el byte SMB110 se encuentra a 2.

Si por cualquier circunstancia al inicializar los datos anteriores se observa un error de configuración, el led DP luce en rojo parpadeante.

Si durante el proceso de comunicación se interrumpe esta entre el 200 y el 300, el led DP luce en rojo. Una vez se restablezca el enlace entre los equipos, automáticamente vuelve a conectarse el modo de transferencia y el piloto verde de dp vuelve a lucir.

10.4.2 Comunicación entre S7 300 y S7 200 mediante enlaces no configurados.

Como vimos anteriormente las CPU´s de los equipos S7 disponen de un tipo de comunicación via MPI denominado enlaces S7. Dichos enlaces, que se realizan entre dos interlocutores con el fin de intercambar información, pueden ser configurados (o estáticos) o no configurados (dinámicos).

Veamos cual es la cantidad de enlaces de cada tipo de que disponemos por CPU S7.

CPU	Enlaces estáticos	Enlaces dinámicos
312 IFM	4	4
313	4	4
314	4	8
314 IFM	4	8
315	4	8
315-2 DP	4	8
316	4	8
318-2 DP	32 de ambos	

La coherencia en la información transmitida mediante este método en las CPU´s 300 en cualquier caso será siempre de 8 bytes como máximo.

Una de las características más interesantes de la utilización de enlaces no configurados para la comunicación se encuentra en la transferencia de información vía MPI entre S7 22x y equipos S7 300/400.

Supongamos que deseamos comunicar un equipo S7 300 con otro S7 224.para enviar y recibir información vía MPI. En primera instancia conectaremos ambos equipos físicamente. A continuación realizaremos una simple programación en el equipo 300, como muestra el programa adjunto. En primer lugar veremos como leer una zona de memoria de un 200 desde el 300. Para ello utilizaremos la SFC67 (X_GET), encargada de realizar la comunicación de lectura en un enlace no configurado. El programa en la OB1 es el siguiente:

```
CALL  SFC   67              // XGET: FUNCIÓN DE RECEPCION PARA ENLACES NO CONFIGURADOS
REQ    :=E124.7             // PETICION DE RECEPCION
CONT   :=M100.0              // CONTROL DE ENLACE
DEST_ID :=W#16#3             // DIRECCION MPI DEL INTERLOCUTOR
VAR_ADDR:=P#M 0.0 BYTE 2     // AREA DEL INTERLOCUTOR A LEER Y TAMAÑO
```

```
RET_VAL :=MW2                // RETORNO DE ERROR
BUSY    :=M4.0               // INDICADOR DE RECEPCION
RD      :=P#M 50.0 BYTE 2    // AREA DONDE DEJAR LOS DATOS RECIBIDOS
```

Veamos el significado de cada uno de los parámetros de la SFC64:

- REQ: El parámetro de entrada REQ (request to activate) es un parámetro de control disparado por nivel. Sirve para lanzar la petición de lectura. Pueden darse dos circunstancias:
    - Si se realiza una petición de lectura en dicho enlace poniendo REQ a 1 la SFC activa la salida BUSY y lanza la petición a la red de comunicaciones.
    - Si se lanza una petición y ésta aún no ha finalizada y se llama la SFC para la misma petición, entonces la SFC no evalúa REQ.
- CONT: El parámetro CONT (continue) permite determinar si debe permanecer o no el enlace abierto entre los dos interlocutores tras la finalización de la petición de lectura.
    - Si se selecciona CONT=0 durante la primera llamada, entonces el enlace vuelve a interrumpirse tras finalizar la transferencia de datos. Con ello queda disponible para intercambiar datos con otro interlocutor. Esta forma de proceder asegura que sólo se ocupan recursos del enlace que son actualmente necesarios.
    - Si se selecciona CONT=1 durante la primera llamada, entonces el enlace permanece tras finalizar la transferencia de datos. Esta forma de proceder es adecuada p. ej. para el intercambio de datos cíclicos entre dos estaciones que van a realizarlo de manera regular. En este caso, podríamos decir que convertimos un enlace dinámico en estático hasta que cambiemos CONT=0.ç
- VAR_ADDR: Dirección MPI del interlocutor. En nuestro ejemplo, en este parámetro seleccionaremos la dirección de la estación S7 224, que previamente hemos configurado en el MicroWin, como muestra la figura.
- RET_VAL: Valor de retorno de estado de comunicaciones. Mientras no exista error de comunicaciones la palabra presenta el valor 0. Frente a un evento de comunicaciones, ya sea un error o un estado, puede adoptar los siguientes valores:

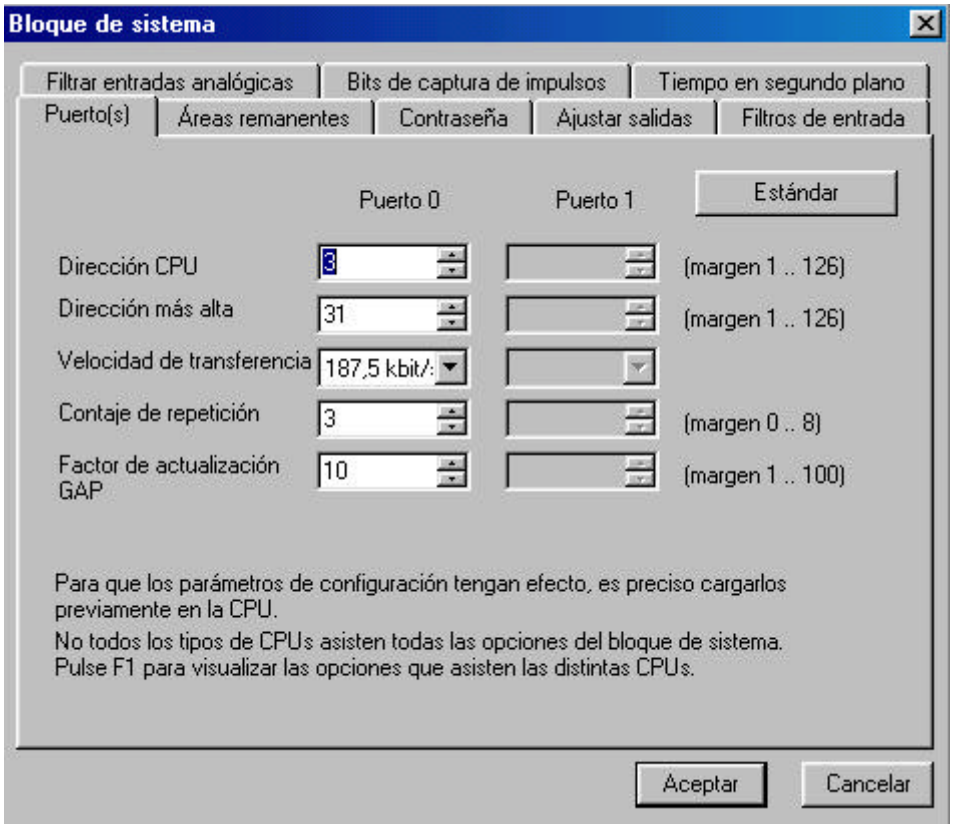


Figura 63. Parametrización de las comunicaciones DP en un 215-2 DP desde MicroWin.

Para la escritura de parámetros en el equipo S7 22x desde el S7 300 utilizaremos la función SFC68 (X_PUT). La llamada a la función tiene el siguiente aspecto:

```
CALL  "X_PUT"
REQ     :=E4.6                  // señal de activación de la escritura
CONT    :=M100.1                // parámetro de control
DEST_ID :=W#16#3                // dirección del interlocutor MPI
VAR_ADDR:=P#A 0.0 BYTE 2        // zona de memoria a escribir en el interlocutor
SD      :=P#M 90.0 BYTE 2       // fuente de la información a escribir
RET_VAL :=MW6                   // palabra de estado
BUSY    :=M4.1                  // bus ocupado
```

El significado de los mismos es:

- REQ: el mismo significado que en la SFC67.
- CONT: el mismo significado que en la SFC67.
- DEST_ID: el mismo significado que en la SFC67.
- VAR_ADDR: Referencia sobre el área en la CPU asociada en la que se desea escribir. Es necesario elegir un tipo de datos soportado por el interlocutor.
- SD: Referencia sobre el área de la CPU propia que contiene los datos a emitir. Se permiten los tipos de datos siguientes: BOOL, BYTE, CHAR, WORD, INT, DWORD, DINT, REAL, DATE, TOD, TIME, S5_TIME,

DATE_AND_TIME así como arrays de los tipos de datos mencionados con excepción de BOOL.SD debe tener la misma longitud que el parámetro VAR_ADDR del interlocutor. Además deben coincidir los tipos de datos en SD y VAR_ADDR.

- RET_VAL: código de control.
- BUSY: Si vale TRUE la emisión aún no ha finalizado.Si tiene valor igual a FALSE la emisión ha finalizado o no hay ninguna emisión activa.

Los valores que puede adoptar RET_VAL tanto para la SFC67 como la SFC68 son los siguientes:

VALOR (Hex)	Significado
0000	Sin error.
00xy	En XY RET_VAL indica la longitud del paquete de datos recibido.
7000	Llamada con REQ = 0 (llamada sin procesamiento), BUSY tiene valor 0 no hay activada ninguna transferencia de datos.
7001	Primera llamada con REQ=1: transferencia lanzada; BUSY tiene el valor 1.
7002	Llamada intermedia (REQ sin importancia): transferencia ya activa; BUSY tiene el valor 1
8090	La dirección de destino del interlocutor indicada no es válida, p. ej.· -IOID erróneo· -dirección base errónea· -dirección MPI errónea (> 126)
8092	Error en SD o RD. Puede ser debido a: -no se permite direccionar el área de datos locales. · -longitud ilegal en RD· Para SFC 67: -La longitud o el tipo de datos de RD no coincide con los datos recibidos o -RD=NIL no admisible. Para SFC 68: -longitud ilegal en SD· o -SD=NIL no admisible
8095	El bloque se procesa ya en una prioridad inferior.
80ª0	Error en el acuse recibido

	Para SFC 68: - El tipo de datos indicado en el SD de la CPU emisora no es soportado por el interlocutor.
80ª1	Problemas de comunicación: llamada de SFC tras interrumpir el enlace establecido
80B0	Objeto no accesible, p. ej. DB no cargado
80B1	Error en puntero ANY. La longitud del área de datos a transmitir es errónea. -
80B2	Avería hardware: módulo no presente. Puede ser por: -El slot configurado no está ocupado.· -Tipo de módulo real diferente del teórico· -Periferia descentralizada no está disponible -En el SDB asociado no hay ningún registro para el módulo
80B3	Los datos sólo pueden o leerse o escribirse, p. ej. DB protegido en escritura
80B4	Error de tipo de datos en puntero ANY o no se permite ARRAY del tipo de datos indicado. El tipo de datos indicado en VAR_ADDR no es soportado por el interlocutor.
80B5	Procesamiento rechazado por estado operativo no permitido
80B6	En el acuse recibido hay un código de error desconocido. -
80BA	La respuesta del interlocutor no cabe en el telegrama de comunicaciones.
80C0	El enlace indicado está ya ocupado por otra petición.
80C1	Cuello de botella de recursos en la CPU en la que corre la SFC, p. ej.:· -Ya se procesa el número máximo de peticiones de emisión diferentes en el módulo.· -El recurso del enlace está ya ocupado p. ej. por una recepción.-
80C2	Falta temporal de recursos en el interlocutor, p. ej.:· -El interlocutor procesa momentáneamente el máximo de peticiones.· -Los recursos necesarios (memoria, etc.) están ocupados.· -Insuficiente espacio en memoria de trabajo(activar la compresión de la memoria). -
80C3	Error al establecer enlace, p. ej.:· -El equipo S7 propio no está conectado a la subred MPI.· -Se ha direccionado el equipo propio en la subred MPI.· -El interlocutor ya no está accesible.· -Falta temporal de recursos del interlocutor

En cualquier caso, se debe tener en cuenta que aunque se mantengan los enlaces S7 permanentemente, el intercambio de información de estos pequeños paquetes está en

torno a los 50 ms entre dos estaciones S7 300, pero aumenta a más de medio segundo para configuraciones mixtas S7 300/200, por lo que no se debe utilizar para intercambiar rápidamente información entre las mismas.

10.5 Comunicaciones mediante datos globales.

La comunicación de datos globales (comunicación GD) es una variante de comunicación sencilla integrada en el sistema operativo de las CPU's S7-300/S7-400.
Mediante ella se consigue el intercambio de datos de manera cíclica entre CPU's a través del interface MPI. El intercambio cíclico de datos se lleva a cabo con la imagen normal del proceso.

La comunicación de datos globales se configura con STEP 7; la transferencia de los datos globales es cosa del sistema, por lo que no se tiene que programar.

Vamos a estudiar cómo calcular, con ayuda de los datos técnicos indicados para cada CPU (número de círculos GD, tamaño y número de paquetes GD, etc.), la cantidad de datos que pueden intercambiar las CPU's mediante el procedimiento "Comunicación GD".

Los datos globales que se utilizan en la comunicación mediante datos globales (comunicación GD) son las siguientes áreas de operandos de la CPU:

- entradas, salidas (de la imagen del proceso)
- marcas
- áreas de bloques de datos
- temporizadores, contadores (no recomendables, porque los valores del emisor ya no son actuales; configurables sólo como áreas de operandos)

Las áreas de la periferia (PE y PA) y los datos locales no se pueden utilizar para la comunicación mediante datos globales.

La comunicación mediante datos globales funciona según el procedimiento broadcast, es decir, no se acusa recibo de los datos globales. El emisor no recibe información alguna acerca de si hay un receptor que ha recibido los datos globales enviados y, en caso de haberlo, cuál es. Si el proceso requiere una transferencia de datos segura, utilice otro servicio, como por ejemplo, las funciones S7

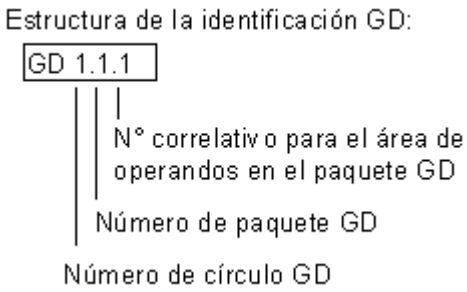
Vamos a definir una serie de conceptos técnicos que utilizaremos posteriormente en las comunicaciones mediante datos globales.

10.5.1 Definición de conceptos en datos globales.

	GD Identifier	Station_1/CPU1	Station_2/CPU1	Station_3/CPU1
Encabez. para col. CPU ➔				
Línea de estado global ➔	GST	MD100		
Línea de estado ➔	GDS 1.1	MD90	MD80	
Línea de factor de ciclo ➔	SR 1.1	8	8	0
Línea de datos globales ➔	GD 1.1.1	>DB5.DBW0:5	DB8.DBW10:5	
	GDS 1.2	MD104		
	SR 1.2	4	8	0
	GD 1.2.1	MW20	>MW30	
	GD 1.2.2	DB6.DBW0:4	>DB8.DBW0:4	
	GDS 2.1			MD80
	SR 2.1	8	0	1
	GD 2.1.1	>DB5.DBW0:10		DB8.DBW20:10

Círculos GD.

Todas las CPU's que intervienen en el intercambio de un paquete de datos común en calidad de emisoras o receptoras, "consumen" un círculo GD.



Paquetes GD

Un paquete GD es un telegrama que envía una sola CPU "de una pasada" a una o varias CPU's.

Un paquete GD contiene como máximo la siguiente cantidad de datos

Máx 22 bytes en el caso del S7-300

máx. 54 bytes en el caso del S7-400

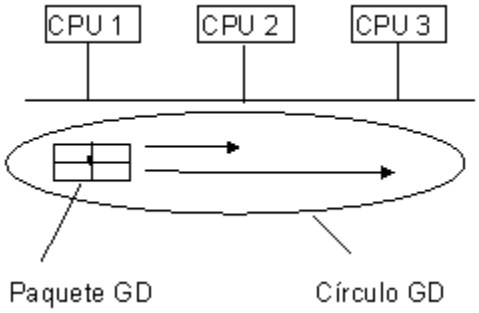
Supongamos que deseamos utilizar el área de emisión más grande de una CPU S7-300 para poder enviar desde un bloque de datos.

Introducimos el área de emisión de la CPU S7-300 en la tabla de datos globales:

DB8.DBB0:22 (es decir, 22 bytes de datos del DB8 a partir del byte de datos 0)

Introducimos el área de recepción de otra CPU (tiene que ser igual de grande que el área de emisión) en la tabla de datos globales:

MW100:11 (es decir, 11 palabras de marcas a partir de MW 100)



El cálculo del tamaño de los paquetes DG tiene una serie de particularidades:

- Si no sólo desea enviar datos del área de operandos, tiene que restar dos bytes por cada área de datos adicional del número máximo de datos netos. Es decir, si seleccionamos dentro de una misma CPU en un mismo paquete DG dos áreas de memoria distintas perdemos 2 bytes por área de memoria adicional a la primera seleccionada. Está demás comentar que es preferible agrupar las variables en buzones de emisión y de recepción.
- Un operando constituido por un bit (p. ej. M 4.1) ”consume" un byte de datos netos del paquete GD.

Supongamos que deseamos enviar desde el bloque de datos y desde la imagen del proceso de las salidas. En este caso, el tamaño del paquete GD no debe superar los 20 bytes. Introduzca las áreas de emisión de la CPU S7-300 en la tabla de datos globales:

- DB8.DBB0:10 (es decir, 10 bytes de datos del DB8 a partir del byte de datos 0)
- AW0:10 (es decir, 10 palabras de salida a partir de AW0)

Introducimos las áreas de recepción de las otras CPU’s igual que en el primer ejemplo; el ”ancho de los datos” tiene que corresponderse con el área de emisión.

Factor de ciclo GD.

El factor de ciclo permite definir para cada CPU que interviene en el intercambio del paquete GD lo siguiente:

- cada cuántos ciclos se envía el paquete GD (sólo para la CPU que está marcada como emisor).
- cada cuántos ciclos se recibe el paquete GD

Existe una excepción a esta regla: el factor de ciclo ”0” significa que el paquete GD se transfiere por control de eventos(y no cíclicamente), (sólo en el S7-400 con la SFC 60/SFC 61).

Supongamos que seleccionamos un factor de ciclo de 20 para un paquete GD en la CPU emisora significa que la CPU enviará el paquete GD desde el punto de control del ciclo cada 20 ciclos.

Un factor de ciclo de 8 para un paquete GD en la CPU receptora significa que la CPU enviará el paquete GD desde el punto de control del ciclo cada 8 ciclos, es decir, que recibirá el paquete GD en el área de operandos).

En cualquier caso, debe ceñirse a las condiciones siguientes para no sobrecargar la comunicación de la CPU:

- CPU’s S7-300: Factor de ciclo Tiempo de ciclo >= 60 ms
- CPU’s S7-400: Factor de ciclo Tiempo de ciclo >= 10 ms

En el receptor, para evitar la pérdida de paquetes GD hay que recibir más paquetes GD de los que se envían.

Para ello se ha de cumplir lo siguiente:

factor de ciclo (receptor) tiempo de ciclo (receptor) < factor de ciclo (emisor) tiempo de ciclo (emisor).

Los factores de ciclo permitidos tanto para el emisor como para el receptor son 0 y valores entre 1 y 255. No obstante, tenga en cuenta que los factores de ciclo muy bajos sobrecargan excesivamente a la CPU.

Recomendación: mantenga el factor de ciclo predeterminado y vigile que el producto del ciclo de tiempo x factor de ciclo sea superior a 0,5s.

El factor de ciclo 0 es característico de una transferencia de datos controlada por eventos vía las SFC’s en el programa de usuario (no todas las CPU’s lo permiten).

Si no introduce ningún factor de ciclo, se utilizará el ajuste predeterminado.

Por ejemplo, supongamos que el programa de usuario de una CPU 412 tiene un tiempo de ciclo de aproximadamente 50ms. El factor de ciclo predeterminado es de22.

50 ms x 22 = 1100ms

Ello significa que aprox. Cada 1,1s se envían o reciben datos globales en esta CPU.
Si el programa de usuario hace aumentar el tiempo de ciclo a p. ej. 80ms, cada 80ms x 22 = 1760s se enviarán o recibirán datos globales.

Para volver a alcanzar el valor de 1100s, hay que volver a calcular el factor de ciclo:

Factor de ciclo (nuevo) = 1100ms/80ms = 13,75

Es decir, prolongando el ciclo hay que ajustar el factor de ciclo a 14 para mantener el mismo intervalo.

Tiempo de respuesta

El tiempo de respuesta de dos equipos que intercambian paquetes GD a través de una subred MPI, se calcula aproximadamente con la fórmula siguiente:
tiempo de respuesta factor de ciclo (emisor) tiempo de ciclo (emisor)+ factor de ciclo (receptor) tiempo de ciclo (receptor) + número (estación MPI) 10 ms

Líneas de estado GDS.

Para cada paquete GD es posible determinar una palabra doble de estado por cada CPU que intervenga en la comunicación. Las palabras dobles de estado se indican en la tabla con el identificador "GDS". Si asigna la palabra doble de estado (GDS) a un operando de CPU del mismo formato, puede evaluar el estado en el programa de usuario o en la línea de estado (GDS).

Línea de estado global GST.

STEP 7 genera un estado global (GST) para todos los paquetes GD. El estado global, que es también una palabra doble con estructura idéntica a la de la palabra doble de estado (GDS), se crea en base a una combinación O (OR) de todas las palabras dobles de estado.

La figura muestra la estructura de la palabra doble de estado y el significado de los bits creados.
Un bit permanece activado hasta que sea desactivado por el programa de usuario o por la PG.
Los bits no indicados están reservados y carecen actualmente de importancia. El estado GD ocupa una palabra doble (MD). Para facilitar la comprensión, la figura representa la MD 120.

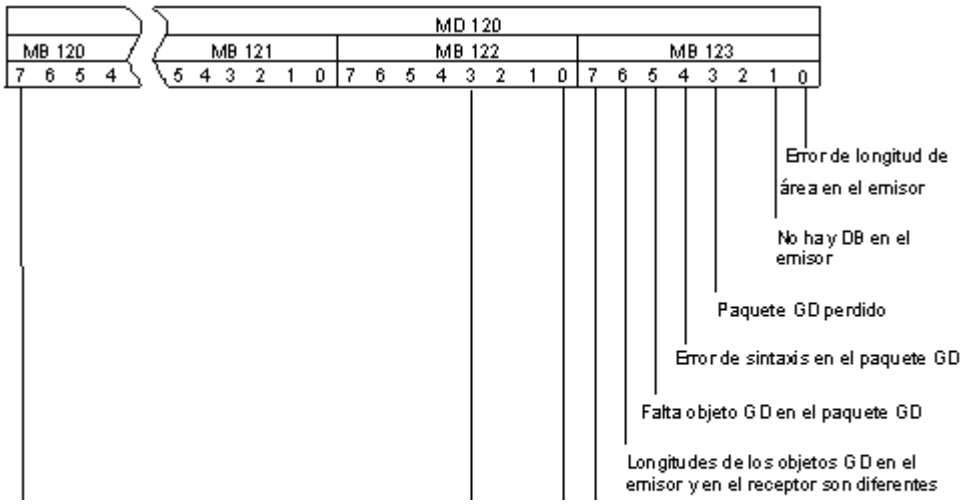


Figura 64. Palabra de estado de las comunicaciones DG. Es importante tener en cuenta que los bits que activa el sistema no son reseteados en ningún momento, por lo que deberemos de acusarlos desde nuestro programa de PLC.

10.5.2 Funcionamiento de la comunicaciones DG.

El intercambio de datos globales se efectúa de la siguiente forma:

- La CPU emisora envía los datos globales al final de un ciclo.
- La CPU receptora lee los datos al principio de un ciclo.

Con ayuda del factor de ciclo puede determinar el número de ciclos al cabo de los cuales debe realizarse la transferencia de datos o la recepción de datos.

- Compile la tabla de datos globales, salvo que se encuentre aún en la fase 1 (puede consultarse en la entrada de la línea de estado en el borde inferior de la pantalla).
- Si aún no se visualiza ningún factor de ciclo en la tabla GD, seleccione el comando de menú **Ver > Factores de ciclo**.
- Introduzca los factores de ciclo deseados. Sólo es posible introducir datos en las columnas en las que el paquete GD asociado disponga de entradas. **Nota:** Si visualiza las líneas de estado y/o las líneas de factores de ciclo, sólo podrá editar dichas líneas y ninguna otra.
- Compile nuevamente la tabla de datos globales (2ª fase).

10.5.3 Configuración de datos globales.

Para poder definir una comunicación DG primeramente deberemos de tener más de una CPU en nuestro proyecto. El número máximo de participantes en un círculo DG es de 4, pero eso no implica que únicamente se puedan enlazar dicho número entre sí. Si utilizamos una como puente entre dos círculos DG podemos superar dicha barrera.

Otro dato importante a la hora de configurar los datos globales es que pese a que los datos a transmitir se encuentren en dos círculos DG que no tengan ninguna relación, deben de programarse en la misma tabla DG o se producirán fallos de comunicaciones. Dicho de otro modo, si tenemos dos CPU´s dialogando entre ellas por DG, y unidas por el cable MPI otras dos realizando su respectivo diálogo, y programamos las tablas DG en dos proyectos distintos, la comunicación fallará. El concepto de un proyecto único por red MPI es muy importante en instalaciones que se vayan ampliando a posteriori, ya que si no se tiene en cuenta obliga a reconstruir la tabla DG (no existe manera de importarla de otro proyecto).

Supongamos que hemos creado un proyecto con dos CPU´s. Si en el administrador Simatic entramos a la red MPI de nuestro proyecto (offline) podremos enlazar los dos equipos a la red.

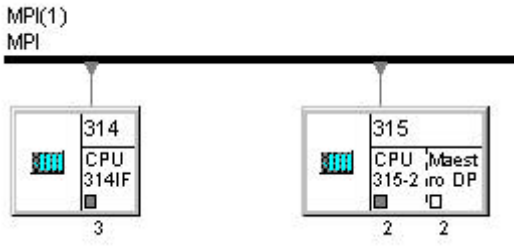


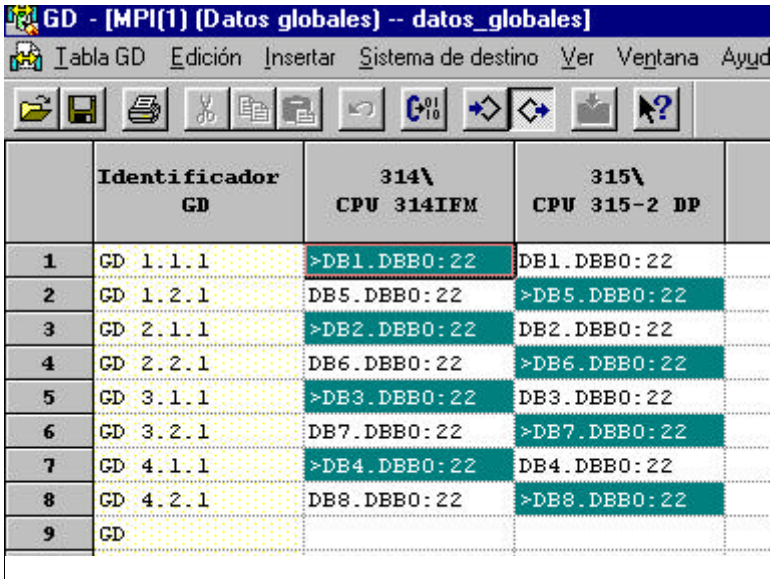
Figura 65. Enlace de CPU's en MPI desde NETPRO.

Una vez enlazadas las CPU´s a la red MPI, podemos acceder a la tabla de datos globales a través del menú de Netpro que se encuentra en Herramientas->Definir datos globales. Para que se active es necesario previamente seleccionar la red MPI dentro de netpro. Si se desea se puede acceder a dicha tabla de DG estando en el administrador Simatic, seleccionando la red MPI, en el menú contextual (botón derecho del ratón).

Para la configuración de la red deberemos de clickear en la cabecera de cada una de las columnas de la tabla, seleccionando la CPU correspondiente. Una vez estableceidas las dos CPU´s, indicaremos las zonas de memoria emisoras y receptoras. Obsérvese que mediante una disposición acertada de las variables a transferir se están transfiriendo un total de 88 bytes de emisión y 88 bytes de recepción, máximo de un cículo DG. La estructura de definición de zonas de memoria a transferir es:

Zona de memoria:cantidad de dichas variables

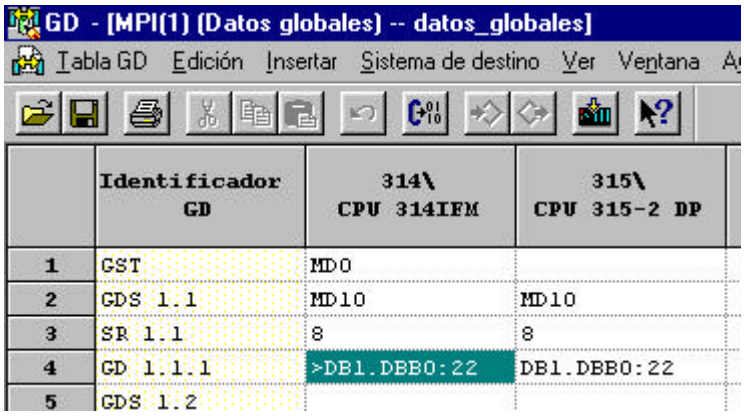
Por lo tanto el segundo número (en nuestro caso 22) no indica la cantidad de bytes, sino de variables. En el caso de palabras, serían un máximo de 11 por paquete DG (que no puede ser superior a 22 bytes).



	Identificador GD	314\ CPU 314IFM	315\ CPU 315-2 DP
1	GD 1.1.1	>DB1.DBB0:22	DB1.DBB0:22
2	GD 1.2.1	DB5.DBB0:22	>DB5.DBB0:22
3	GD 2.1.1	>DB2.DBB0:22	DB2.DBB0:22
4	GD 2.2.1	DB6.DBB0:22	>DB6.DBB0:22
5	GD 3.1.1	>DB3.DBB0:22	DB3.DBB0:22
6	GD 3.2.1	DB7.DBB0:22	>DB7.DBB0:22
7	GD 4.1.1	>DB4.DBB0:22	DB4.DBB0:22
8	GD 4.2.1	DB8.DBB0:22	>DB8.DBB0:22
9	GD		

Figura 66. Definición de datos globales, comunicando 88 bytes de envío y 88 de recepción.

Si accedemos al menú *Ver->Factores de Ciclo* obtendremos el intervalo entre envío de paquetes DG (por defecto cada 8 ciclos), pudiéndose modificar.



	Identificador GD	314\ CPU 314IFM	315\ CPU 315-2 DP
1	GST	MD0	
2	GDS 1.1	MD10	MD10
3	SR 1.1	8	8
4	GD 1.1.1	>DB1.DBB0:22	DB1.DBB0:22
5	GDS 1.2		

Figura 67. Asignación de marcas para estado de comunicaciones en el paquete de datos globales 1

Si accedemos al menú *Ver->Estado DG* podremos definir las variables en las cuales deseemos que el sistema indique el estado de las comunicaciones de dicho círculo DG (para la variable GDS) y del total de las comunicaciones (para la variable GST). En realidad,

actualmente siempre indica un valor 8, no mostrando ninguna variación en su valor. Suponemos que en próximas versiones de firmware las variables sí se comporten como se indica en el apartado anterior sobre variables GST.

Terminada la configuración de la tabla, únicamente queda compilarla y transferírsela a las CPU´s. Dicho proceso puede realizarse de manera simultánea (si las poseemos conectadas mediante una red MPI), o en dos pasos de manera individual.

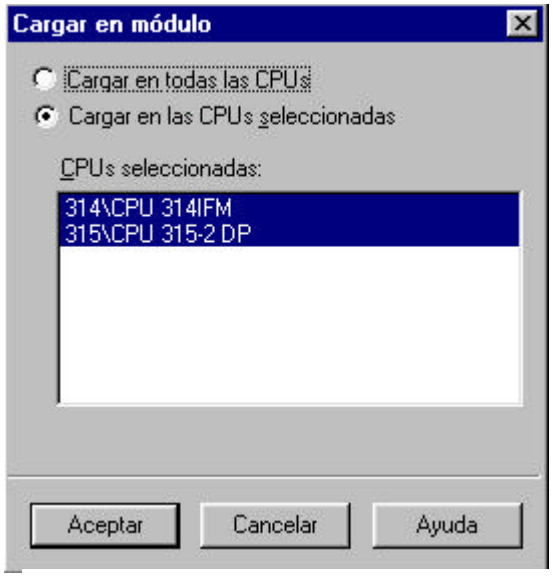


Figura 68. Transferencia de datos globales a las CPU's.

10.5.4 Multiplexación de paquetes DG.

Bien, esto para 88 bytes. Pero, ¿y si necesitamos más?. Lógicamente para tamaños superiores de información lo recomendable es instalar una red Profibús o ETHERNET. Pero en el caso de que únicamente necesitemos sobrepasar esta cantidad ligeramente, y nuestros tiempos de respuesta en comunicaciones puedan prolongarse hasta el segundo, podemos multiplexar los paquetes de comunicaciones DG.

Realizar esta tarea a través de variables de estado de comunicaciones no s posible, debido a que la red MPI no indica la finalización del envío de un determinado paquete. Por lo tanto, recurriremos a un método simple pero efectivo, basado en realizar una copia alternativa de un paquete u otro siempre a una zona de datos común, que es la que hemos definido en la tabla DG. En la primera variable de dicho paquete indicamos la DB en la que deseamos almacenar dichos datos en el receptor.

Supongamos que la DB5 tiene los primeros 22 bytes definidos como zona de emisión DG. Vamos a realizar unas líneas de código que nos van a copiar alternativamente la DB10 o la DB11 en la DB5, introduciendo en el primer byte el destino de estos datos en el receptor.

```
    U    T     0                  // ALTERNANCIA CADA X TIEMPO DE DICHA VARIABLE
    SPB  m001

    CALL  "BLKMOV"                          // COPIAR LA DB10 A LA DB5
     SRCBLK :=P#DB10.DBX 0.0 BYTE 22        // ZONA A COPIAR
     RET_VAL:=MW0
     DSTBLK :=P#DB5.DBX 0.0 BYTE 22         // ZONA DONDE COPIAR

    L    10                                 // INDICAMOS LA DB DONDE SE DEBEN
    T    DB5.DBB    0                       // DE ALMACENAR ESTOS DATOS EN EL RECEPTOR

    BEA
m001: NOP  0
    CALL  "BLKMOV"                          // COPIAR LA DB11 A LA DB5
     SRCBLK :=P#DB11.DBX 0.0 BYTE 22        // ZONA A COPIAR
     RET_VAL:=MW0
     DSTBLK :=P#DB5.DBX 0.0 BYTE 22         // ZONA DONDE COPIAR

    L    10                                 // INDICAMOS LA DB DONDE SE DEBEN
    T    DB5.DBB    0                       // DE ALMACENAR ESTOS DATOS EN EL RECEPTOR
```

La alternancia de la variable T0 se realiza mediante el código explicado en el *generador de pulsos asíncrono* (ver capítulo *Instrucciones básicas*).

En el lado del receptor únicamente es necesario realizar la acción contraria, en función del primer byte de la DB5. El tiempo de alternancia, que viene determinado por la base de los temporizadores del generador de pulsos depende de la cantidad de paquetes que estén circulando por la red y del ciclo del programa de PLC. Si se elige un tiempo corto, se corre el riesgo de que se realice la alternancia antes de que se envíe el paquete correspondiente (recordar que se enviaba por defecto cada 8 ciclos de CPU).

Para optimizar este tiempo de alternancia, y por lo tanto agilizar las comunicaciones en multiplexación se pueden hacer dos cosas:

- Reducir el factor de ciclo (no muy recomendable).
- Aumentar el tiempo dedicado a las comunicaciones por la CPU.

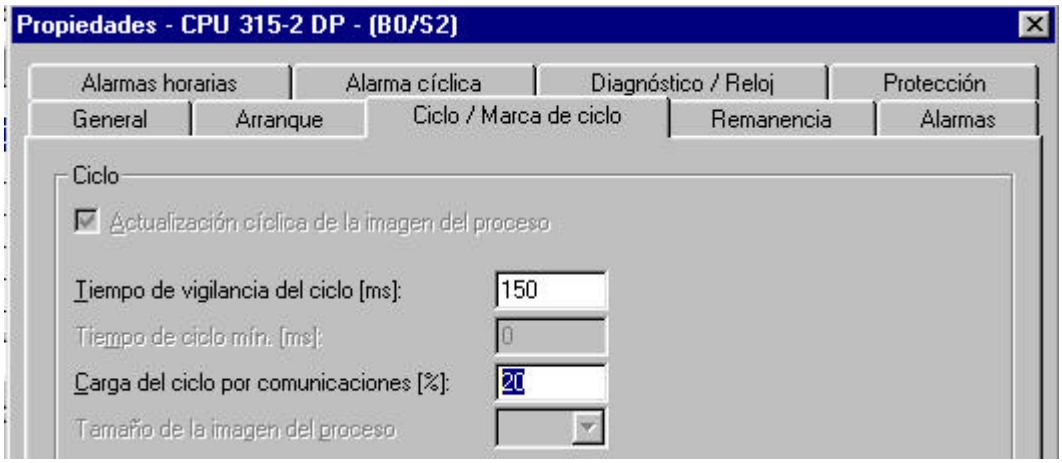


Figura 69. Parametrización del tiempo dedicado a comunicaciones desde Hardware de Step 7.

Esta segunda solución, pese a prolongar el tiempo de ciclo, utilizada moderadamente permite optimizar un poco las comunicaciones MPI. Esto se consigue en el Hardware de la CPU, en la solapa *Ciclo/Marcas de ciclo*. En cualquier caso, el parámetro "Carga del ciclo por comunicaciones“ permite controlar la duración de procesos de comunicación que prolongan siempre el tiempo de ciclo. Este parámetro no afecta las funciones de test con la PG, por lo que puede prolongarse considerablemente el tiempo de ciclo. El sistema operativo de la CPU ofrece continuamente el porcentaje de rendimiento del ciclo ajustado para la comunicación (técnica de segmentación de tiempos). Si este rendimiento no se requiere para la comunicación, estará disponible para otros procesos.

Si aumenta la carga de la comunicación se prolonga el tiempo de ciclo, incluso desproporcionadamente, p. ej. debido a tiempos de ejecución adicionales del sistema operativo. Además, el tiempo de ciclo puede variar mucho si la carga de la comunicación es inestable. Compruebe por lo tanto cómo repercute un cambio del valor del parámetro "Carga del ciclo por comunicaciones“ estando la instalación en marcha.

10.6 Comunicaciones en Profibús FDL.

10.6.1 Características generales de la comunicación FDL.

La comunicación FDL se va a realizar a través de enlaces. Un enlace es la interconexión “virtual” (ya que físicamente están todos los equipos de la red conectados) entre dos equipos que desean comunicarse. El enlace no posee un sentido de comunicación emisor-receptor, sino que será nuestra llamada a las funciones de comunicaciones utilizando el enlace el que dará una dirección de emisión y recepción al mismo.

La comunicación FDL en S7 300 se realiza a través de la tarjeta 342-5. Teniendo en cuenta este dato, veamos las características de la comunicación FDL en S7:

- Cantidad de equipos conectables: 16 máximo.
- Tamaño máximo del área de emisión (por enlace): 240 bytes.
- Tamaño máximo del área de recepción (por enlace): 240 bytes.

Lógicamente, cuantos más enlaces tengamos en nuestra red FDL, y cuanto mayores sean los paquetes de emisión, mayor será el tiempo de rotación de testigo. La siguiente tabla muestra una estimación de tiempos de transferencia de paquetes, con respecto a enlaces configurados y tamaño de los paquetes:

Número de enlaces / Long. telegrama	1	4	8	16
1 byte	77 / s	92 / s	92 / s	92 / s
100 bytes	73 / s	88 / s	88 / s	88 / s
240 bytes	67 / s	82 / s	83 / s	84 / s

Figura 70. Frecuencia con la que se envía un paquete con respecto a la cantidad de enlaces y el tamaño que pueden poseer los paquetes. Obsérvese que en el caso más desfavorable, se envía 5,25 veces cada uno de los paquetes al receptor. La gráfica se ha realizado con una velocidad de transferencia de 1,5 Mbaud.

10.6.2 Un vistazo a la CP342-5.

La CP 342-5 soporta los siguientes servicios de comunicaciones:

- Maestro de DP. La cantidad máxima de esclavos que soporta es de 64. En estos 64 esclavos, la suma de los bytes que poseen los módulos de entradas / salidas en total no puede superar los 240 bytes de entradas y los 240 bytes de salidas.
- Esclavo de DP. El área de entradas máxima con el que se comunicará con el maestro DP es de 86 bytes, y el de salidas de 86 bytes.
- Enlaces FDL. Es el tipo de comunicación que vamos a estudiar en este apartado.
- Comunicación S7. Si todos los equipos de la red de comunicaciones son S7, se puede realizar comunicación mediante funciones S7.
- Funciones PG. Es posible programar los equipos de la red a través de la conexión a una CP.
- Funciones HMI. Se pueden conectar equipos de visualización a través de la CP342-5 a la CPU.

Estos servicios de comunicaciones se pueden utilizar a la vez (la tarjeta soporta multiprotocolo). Sin embargo esto no es muy recomendable en l caso más usual, que es la utilización de la CP 342-5 como maestro de DP y a la vez conectada a una red mediante funciones S7o FDL, ya que se ralentiza en demasía la adquisición de datos de la periferia con respecto a la parte de maestro DP. Es recomendable utilizar una CPU con maestro DP integrado, y una CP 342-5 exclusivamente para comunicaciones FDL, aunque el primer caso es técnicamente posible.

Se puede realizar programación de los equipos de una red a través de la CP 342-5, pero con una limitación: no se pueden realizar descargas a un proyecto de configuraciones Hardware de los equipos desde Step 7.

Nota: Para parametrizar la CP, es necesario disponer del paquete NCM S7, que actualmente ya se encuentra incorporado en Step 7 V5.x. En versiones anteriores de Step 7 es necesario disponer del mismo.

LED STOP (amarillo)	LED RUN (verde)	LED SF (rojo)	Estado operativo de CP
●	☀	○	Arranque o se carga FW
○	●	○	RUN
☀	●	○	STOPPING
☀	○	○	Espera actualización de FW (duración: 10 s)
☀	○	●	Espera actualización de FW (CP contiene actualmente un FW incompleto)
●	○	○	STOP
○	○	●	STOP con error
○	●	☀	RUN; pero perturbaciones en PROFIBUS; o falta(n) esclavo(s) DP en el PROFIBUS
☀	☀	☀	Error de componente /error del sistema

Leyenda: ● encendido ○ apagado ☀ destellando

			ocupado en
Pin	Nombre señal	Designación PROFIBUS	RS 485
1	PE	Tierra de protección	sí
2	SIL	-	-
3	RxD/TxD-P	Línea de datos - B	sí
4	RTS (AG)	Control - A	-
5	M5V2	Potencial referencia datos	sí
6	P5V2	Polo + alimentación	sí
7	BATT	-	-
8	RxD/TxD-N	Línea de datos - A	sí
9	RTS (PG)	Control - B	-

Figura 72. Asignación de pines en el conector de la CP 342-5.

10.6.3 Ejemplo de comunicación FDL.

Para poder realizar una comunicación FDL en S7, suponemos que poseemos en un proyecto de Step 7dos equipos S7 con su tarjeta CP342-5.

El primer paso consistirá en parametrizar las tarjetas CP 342-5. Para ello, estando en Hardware de Step 7, seleccionaremos las propiedades de la CP, y en la solapa General, volvemos a seleccionar Propiedades.

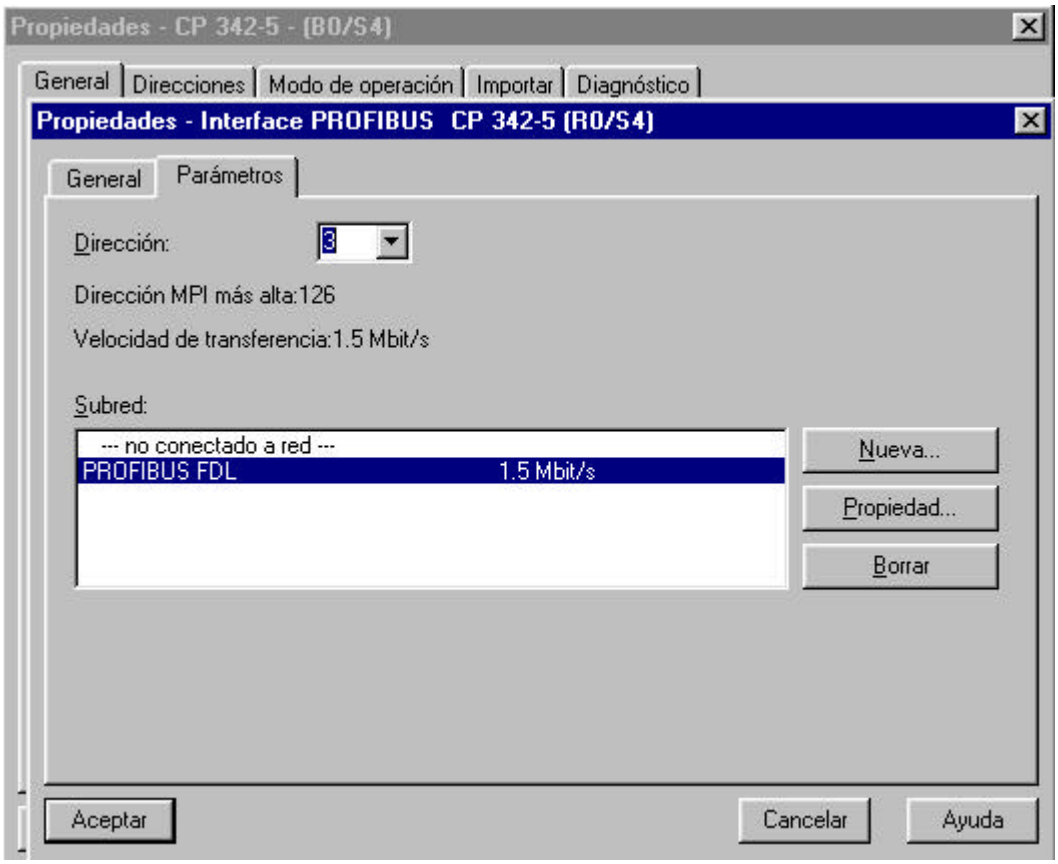


Figura 73. Configuración de la red FDL desde Hardware de Step 7.

En esta ventana deberemos de crear una nueva red Profibús (si no existiera anteriormente), y podremos asignar una dirección Profibús a la tarjeta. Las direcciones de los equipos CP que se configuren en la red deben lógicamente ser distintas, siendo conveniente apuntarnos en un papel qué equipos poseen qué direcciones (para no perderse...).

Si en esta solapa seleccionamos el botón Propiedades, obtenemos en la solapa la posibilidad de modificar la velocidad de nuestra red de comunicaciones, así como el tipo de protocolo Profibús que vamos a utilizar (perfil). Las diferentes posibilidades son:

- DP: lógicamente para comunicación DP maestra o esclava.
- Estándar: Para comunicación FDL o Funciones S7.
- Universal: Para poder realizar comunicación multiprotocolo, es decir, comunicar por la CP como maestro o esclavo de DP, y a la vez enviar datos por el mismo cable a un participante de una red.

- Personalizado:

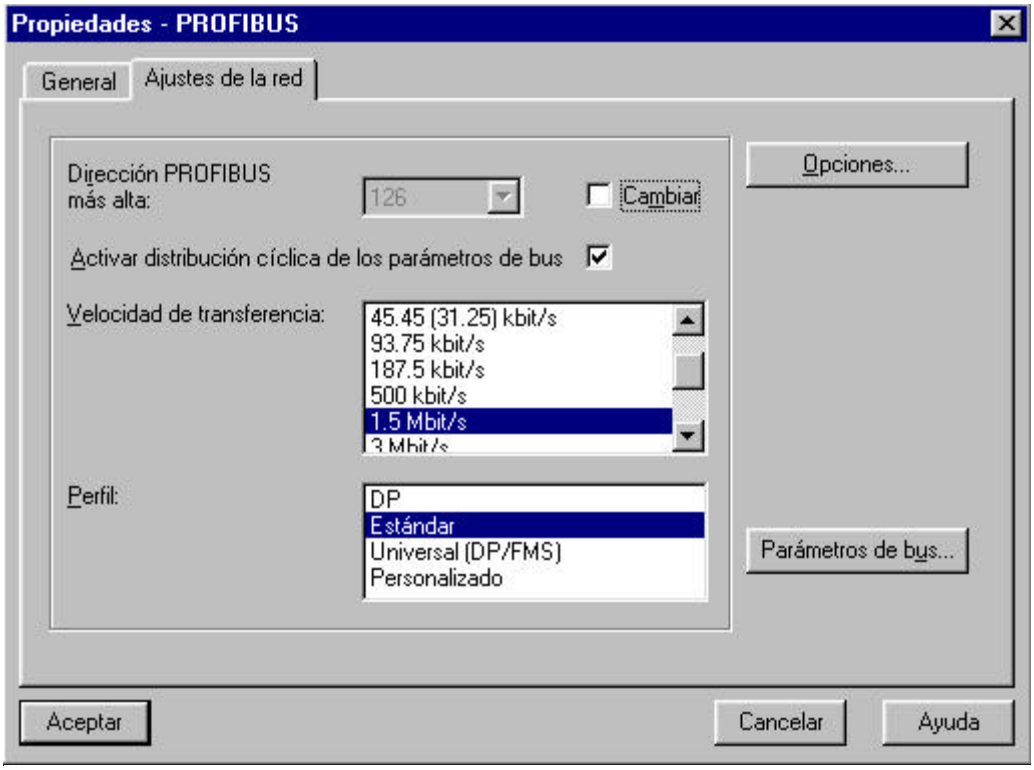


Figura 74. Selección del tipo de protocolo Profibús y su velocidad.

También podemos modificar la dirección de la estación más alta que puede existir en la red, pero esto únicamente tendrá efecto en las reconexiones de los equipos a la red una vez salgan de ella, ya que el testigo no deberá de “buscar” estaciones más allá del parámetro que indiquemos en este apartado. No es en verdad apreciable el efecto de la disminución de este parámetro.

Repetimos esta operación de configuración en todas las CP’s que van a componer la red. Una vez finalizado este proceso, accedemos al paquete NETPRO, a través de nuestra red Profibús que se encuentra en el raíz de nuestro proyecto estando en Administrador de Step 7.

En esta pantalla deberemos de configurar los enlaces entre nuestros equipos de comunicaciones. En este ejemplo vamos a realizar dos entre nuestra CPU´s, para poder utilizar uno como emisión desde la CPU 314 IFM y otro desde el 315-2 DP.

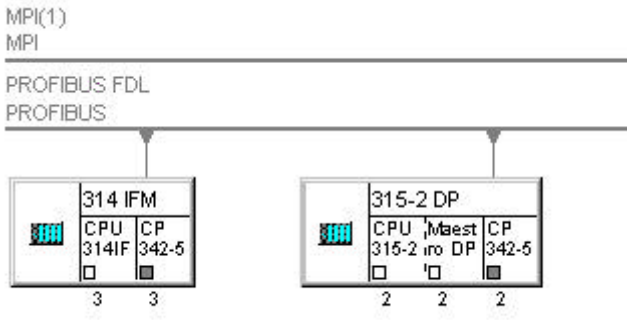


Figura 75. Configuración de una red FDL en NETPRO.

Seleccionaremos primero la CPU 314 IFM (pero exactamente en el cuadrado de la CPU, como nos indica en el texto de la pantalla inferior en la ventana de NETPRO). Si con el botón derecho seleccionamos *Enlace...* comenzaremos a configurar nuestro primer enlace.

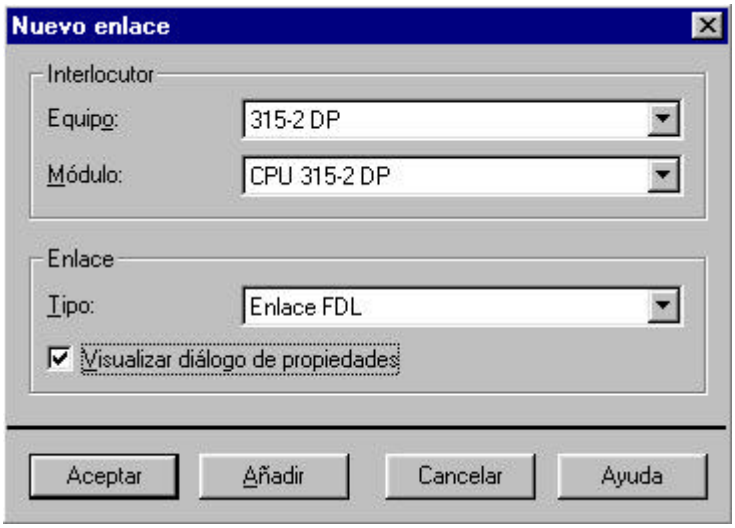


Figura 76. Configuración de un enlace FDL.

Veamos los parámetros a configurar:

- **Interlocutor:** El equipo con el que nos queremos conectar desde la estación en la que estamos configurando el enlace. En este caso, como estamos en la CPU 314 IFM, deberemos de conectarnos al 315-2 DP.
- **Tipo de enlace:** Indica el tipo de protocolo Profibús que vamos a utilizar.
- **Ver diálogo de propiedades:** nos permitirá modificar las propiedades del enlace en el siguiente paso, una vez aceptemos el mismo.

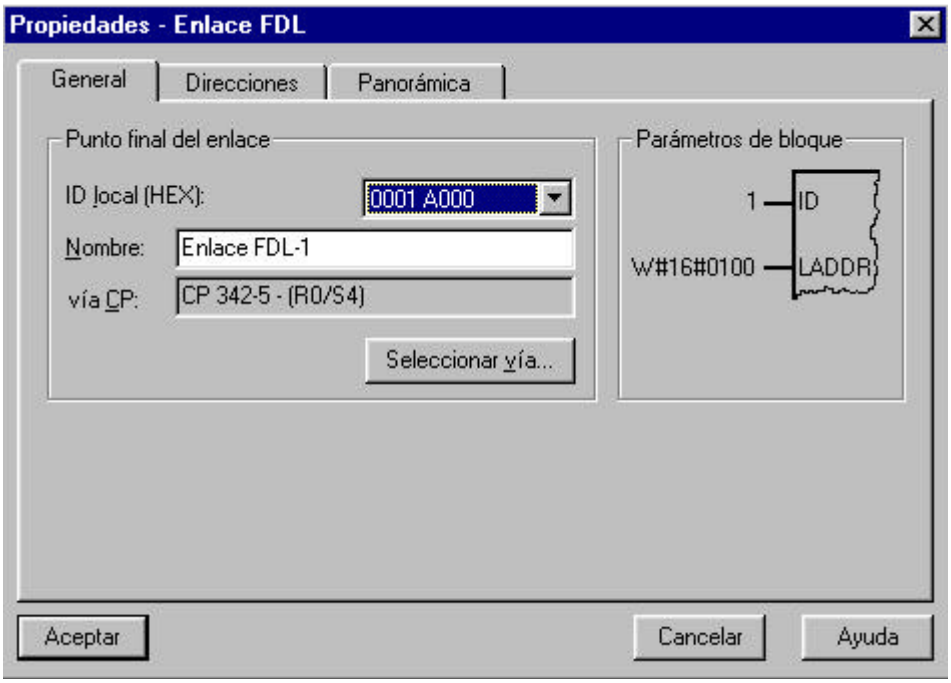
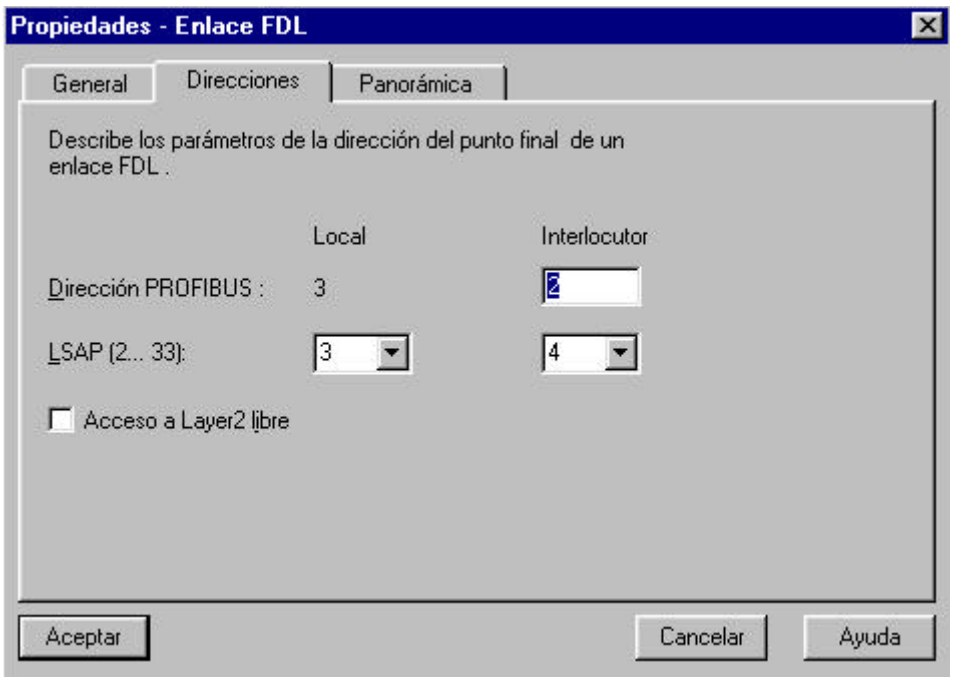


Figura 77. Propiedades del enlace FDL.

- **ID local:** Es el parámetro al que nos referiremos posteriormente al realizar en la CPU las llamadas de envío / recepción de paquetes de datos para saber por qué enlace deseamos enviarlos / recibirlos.

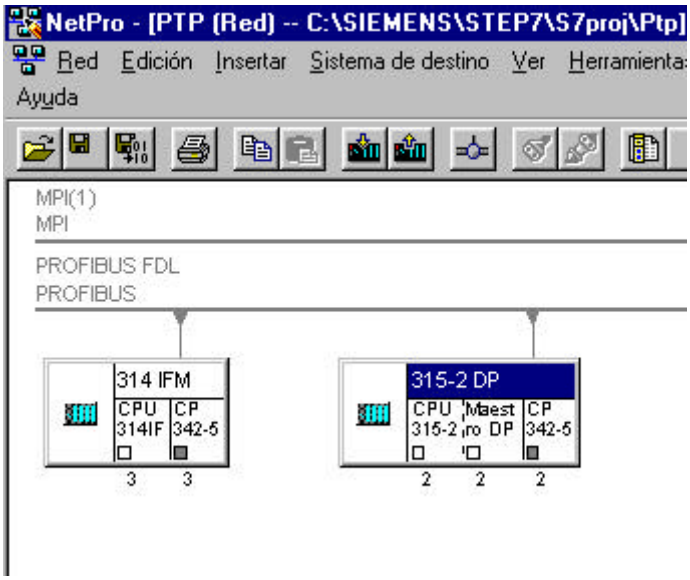


- **Nombre:** Etiqueta a nivel indicativo del enlace.

- **Vía CP:**
- **Dirección Profibús:** La estación de profibús a la que deseamos conectarnos desde la que estamos actualmente configurando el enlace.
- **LSAP:** Los LSAP´s local y remoto. Disponemos de uno local y otro remoto (interlocutor). No es necesario modificar dichos valores, ya que el sistema se encarga de asignarnos aquellos que se encuentran disponibles actualmente en el proyecto.
    - LSAP local (Link ServiceAccessPoint): El LSAP local controla la receptibilidad del CP PROFIBUS. En el CP PROFIBUS se ponen a disposición los recursos de recepción para el LSAP a fin de que pueda recibir los datos en el enlace FDL.
    - LSAP remoto (Link ServiceAccessPoint): El LSAP remoto controla la emisión en el CP PROFIBUS. El CP PROFIBUS envía datos a través del LSAP a la estación en el enlace FDL. La estación de destino tiene que estar disponible para recibir este SAP.

Con esto tendremos definido un enlace entre las CPU´s, que en principio no tiene un sentido de emisión o recepción de datos, pero que una vez lo utilicemos en nuestro programa de PLC únicamente podrá realizar una de las funciones. Para que ambas CPU´s puedan emitir y recibir deberemos de configurar otro enlace, esta vez seleccionando la CPU 315-2 DP, y siguiendo los pasos anteriores.

Por último, deberemos de transferir esta configuración a los equipos. Para ello, será necesario conectarnos a cada uno de ellos, y seleccionar la CPU correspondiente, ya que en caso contrario no se nos activa el icono de envío de la configuración al PLC (ver imagen inferior).



Esta operación la repetiremos en ambas CPU´s. Realizado esto, podemos salir de Netpro, para pasar a configurar nuestras llamadas desde el programa de PLC. Para ello,

utilizaremos las funciones FC5 y FC6 de la biblioteca SIMATIC_NET_CP, que importaremos a nuestro proyecto (recordar que debe de realizarse primero al disco duro, no al PLC, para posteriormente pasarlas al PLC).

El bloque AG_SEND / AG_LSEND (FC5), transfiere datos al CP PROFIBUS para transferirlos a través de un enlace FDL configurado (envio). El área de datos indicada puede ser un área PA, un área de marcas o bien un área de bloques de datos. Para que se indique que la función ha sido ejecutada correctamente se tiene que haber enviado todo el área de datos de usuario a través del PROFIBUS . Veamos los parámetros que posee:

PARAMETRO	TIPO	FORMATO	SIGNIFICADO
ACT	ENTRADA	BOOL	ACT = TRUE se envían bytes LEN del área de datos FDL indicada con el parámetro SEND. ACT = 0 se actualizan los códigos de condición de estado DONE, ERROR y STATUS, pero no se envían los datos.
ID	ENTRADA	INT	En el parámetro ID se indica el número del enlace FDL. .
LADDR	ENTRADA	WORD	Dirección inicial del módulo. Al configurar el CP con la herramienta de configuración de STEP 7 se visualiza la dirección inicial del módulo en la tabla de configuración. Indique aquí esta dirección.
SEND	ENTRADA	ANY	Indicar la dirección y la longitud. La dirección del área de datos FDL puede señalar a una de las áreas siguientes: > Area PA > Area de marcas > Area de bloques de datos Si la llamada se hace con encabezamiento de petición, el área de datos FDL incluye dicho encabezamiento y los datos útiles.
LEN	ENTRADA	INT	Número de bytes a enviar del área de datos FDL con esta petición. El rango de valores posibibles va de 1 a la ”Longitud especificada en el parámetro SEND”. En caso de llamada con encabezamiento, hay que considerar: datos útiles + encabezamiento, LEN >= 4
DONE	SALIDA	BOOL	Este parámetro indica si la petición se ha terminado con o sin errores. Siempre que se envíen los datos sin error su valor es TRUE. Cuando los datos se están enviando, o cuando existe un error su valor es FALSE.
ERROR	SALIDA	BOOL	Código de error. Indica que el envío ha terminado con error. Siempre que se no se puedan enviar los datos su valor es TRUE. Si los datos se están enviando, o no ha habido error en el envío de datos, su valor es FALSE.
STATUS	SALIDA	WORD	Código de estado Para saber su significado ver tabla inferior.

STATUS (hex)	SIGNIFICADO
7000H	Código sólo posible en S7-400: La FC se llamó con ACT=0, pero no se trata la petición.
8181H	Petición en curso.
8183h	Falta la configuración o no se han iniciado aún el servicio FDL en el CP PROFIBUS-CP.
8184H	Enlace FDL sin búfer de peticiones: error del sistema. Enlace FDL con búfer de peticiones: parámetro LEN<4 o parámetro ilegal en encabezamiento (en caso de acceso a layer 2 libre).
8185H	Parámetro LEN mayor que el área fuente SEND.
8186H	Parámetro ID no valido. ID != 1,2....15,16.
8301H	SAP no activado en equipo de destino.
8302H	No se dispone de recursos de recepción en el equipo de destino. La estación receptora (equipo) no procesa los datos recibidos con velocidad suficiente o bien no ha suministrado recursos de recepción.
8303H	El servicio PROFIBUS (SDA-SendDatawithAcknowledge) no es asistido por el equipo de destino en este SAP.
8304H	El enlace FDL no ha sido establecido.
8311H	El equipo de destino no es accesible con la dirección PROFIBUS indicada.
8312H	Error PROFIBUS en el CP: P. ej., cortocircuito en bus; estación fuera del anillo.
8315H	Error de parámetro interno en enlace FDL con encabezamiento: Parámetro LEN<4 o parámetro ilegal en encabezamiento (en caso de acceso a layer 2 libre).
8F22H	Area fuente no valida, p. ej.:Area no existente en el DB.Parámetro LEN < 0
8F24H	Error de área al leer un parámetro.
8F28H	Error de alineación al leer un parámetro.
8F32H	El parámetro contiene un número DB demasiado alto.
8F33H	Número DB erróneo.
8F3AH	Area no cargada (DB).
8F42H	Retardo en acuse al leer un parámetro del área de periferia.
8F44H	Dirección del parámetro al leer inhibida en la pista de acceso.
8F7FH	Error interno. P. ej., referencia ANY no permitida p. ej., parámetro LEN = 0 .
8090H	No existe ningún módulo con esta dirección inicial.
80ª4H	No está establecida la conexión vía bus K entre CPU y CP (a partir de la actualización del sistema operativo de la CPU S7-400 del 3/978).
8091H	La dirección inicial del módulo no esta en formato de palabra doble.
80ª4H	El enlace vía bus K entre CPU y CP no está establecido. (en CPUs con versiones más actuales)
80B0H	El módulo no conoce este registro.
80B1H	Area de destino no valida.P. ej., área de destino > 240 Bytes.
80B2H	El enlace de bus K entre la CPU y el CP no está establecido (en versiones de CPU más antiguas; si no, 80A4H.).
80C0H	No se puede leer el registro.
80C1H	El registro indicado está siendo procesado.
80C2H	Hay demasiadas peticiones pendientes.
STATUS (hex)	**SIGNIFICADO**
80C3H	Recursos ocupados (memoria).
80C4H	Error de comunicación (se presenta temporalmente, por lo que es conveniente la

	repetición en el programa de usuario).
80D2H	Dirección inicial del módulo erróneo.

El bloque AG_RECV (FC6) recibe del CP PROFIBUS los datos transferidos a través de un enlace FDL configurado (recepción). El área de datos indicada para recibir los datos puede ser un área PA, un área de marcas o bien un área de bloques de datos. Se señala que la función ha sido ejecutada sin errores cuando se hayan podido recibir los datos del CP PROFIBUS.

PARAMETRO	TIPO	FORMATO	SIGNIFICADO
ID	ENTRADA	INT	En el parámetro ID se indica el número de enlace FDL.
LADDR	ENTRADA	WORD	Dirección inicial del módulo Al configurar el módulo el CP con HWConfig de STEP 7 se visualiza la dirección inicial de configuación. Indique aquí esta dirección.
RECV	ENTRADA	ANY	Indicar la dirección y la longitud.La dirección del área de datos FDL puede señalar a una de las siguientes áreas: > Area PA > Area de marcas >Area de bloques de datos Si la llamada se hace con encabezamiento de petición, el área de datos FDL incluye dicho encabezamiento y los datos útiles.
LEN	SALIDA	INT	Indica el número de bytes que han sido aceptados por el CP PROFIBUS en el área de datos FDL.En caso de llamada con encabezamiento, hay que considerar datos útiles + encabezamiento, LEN >= 4
NDR	SALIDA	BOOL	Este parámetro indica si se aceptaron nuevos datos. Para saber el significado de este parámetro en relación con los parámetros ERROR y STATUS, véase la tabla siguiente.
ERROR	SALIDA	BOOL	Código de error Para saber el significado de este parámetro en relación con los parámetros NDR y STATUS, véase la tabla siguiente.
STATUS	SALIDA	WORD	Código de estado.

STATUS (hex)	SIGNIFICADO
0000H	Se han aceptado nuevos datos.
8180H	Aún no se dispone de datos.
8181H	Petición en curso.
8183H	Falta la configuración o aún no se ha iniciado el servicio FDL en el CP PROFIBUS.
8184H	Error de sistema.
8185H	Búfer de destino (RECV) demasiado pequeño.
8186H	Parámetro ID no válido. ID != 1,2....15,16.

8304H	El enlace FDL no ha sido establecido.
8f23H	Area fuente no válida. P. ej.: Area no existente en el DB.
8F25H	Error de área al escribir un parámetro.
8F29H	Error de alineación al escribir parámetro.
8F30H	El parámetro no se encuentra en el 1er bloque de datos act. protegido contra escritura.
8F31H	El parám. no se encuentra en el 2o bloque de datos act. prot. contra escr.
8F32H	El parámetro contiene un número DB demasiado alto.
8F33H	Número DB erróneo.
8F3AH	Area de destino no cargada (DB).
8F43H	Retardo en acuse al escribir un parámetro en el área de periferia.
8F45H	Dirección del parámetro a escribir inhibida en la pista de acceso.
8F7FH	Error interno. P. ej., referencia ANY no permitida.
8090H	No existe ningún módulo con esta dirección inicial.
8091H	La dir. inicial del módulo no está en formato de palabra doble.
80A0H	Acuse negativo al leer el módulo.
80B0H	El módulo no conoce el registro.
80B1H	Area de destino no válida.
80B2H	El enlace del bus K entre la CPU y el CP no está establecido (en versiones de CPU más antiguas; si no, 80A4H.).
80C0H	El registro no puede ser leído.
80C1H	El registro indicado está siendo procesado.
80C2H	Hay demasiadas peticiones pendientes.
80C3H	Recursos ocupados (memoria).
80C4H	Error de comunicación (se presenta temporalmente, por lo que es conveniente la repetición en el programa de usuario).
80D2H	Dirección inicial del módulo errónea.

Las llamadas a las funciones FC5 y FC6 las deberemos de realizar en la OB1 de cada uno de nuestros autómatas. Para el caso del 314 IFM la configuración será la siguiente:

```
CALL  "AG_SEND"                  // FC5: ENVIAR 240 BYTES
 ACT   :=TRUE                    // ENVIAR SIEMPRE
 ID    :=2                       // NUMERO DE ENLACE
 LADDR :=W#16#100                // DIRECCIÓN FÍSICA DE LA TARJETA CP 342-5
 SEND  :=P#DB10.DBX 0.0 BYTE 240 // ZONA DE MEMORIA A ENVIAR
 LEN   :=240                     // LONGITUD DE DATOS A ENVIAR (EN BYTES)
 DONE  :=M10.0                   // TRANSFERENCIA FINALIZADA
 ERROR :=M10.1                   // ERROR EN TRANSFERENCIA
 STATUS:=MW12                    // ESTADO DE LA COMUNICACION

CALL  "AG_RECV"                  // FC6: RECIBIR 240 BYTES
 ID    :=1                       // NUMERO DE ENLACE
 LADDR :=W#16#100                // DIRECCION FISICA DE LA TARJETA CP 342-5
 RECV  :=P#DB11.DBX 0.0 BYTE 240 // ZONA DE MEMORIA DONDE DEJAR LOS DATOS
 NDR   :=M0.0                    // RECEPCION FINALIZADA
 ERROR :=M0.1                    // ERROR EN RECEPCION
 STATUS:=MW2                     // ESTADO DE LA COMUNICACION
 LEN   :=MW4                     // CANTIDAD DE DATOS RECIBIDOS (EN BYTES)
```

10.7 Comunicaciones punto a punto

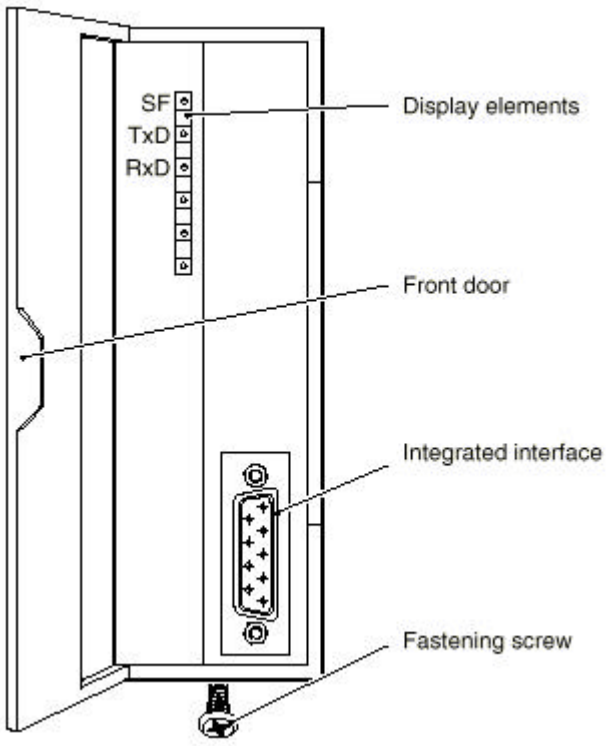
10.7.1 CP 340

La CP 340 es la tarjeta indicada para realizar acoplamientos punto a punto mediante protocolo ASCII libre (comunicar con una pesadora, con un lector de código de barras, etc..).

Existen tres tipos de tarjetas, según sea el soporte físico de las comunicaciones que va a realizar la misma:

- RS 232C
- TTY
- RS 422/485

Comenzaremos estudiando el significado de los leds de la tarjeta



- **SF:** Fallo de la tarjeta.
- **TxD:** Transmisión de un telegrama de comunicaciones.
- **RxD:** Recepción de un telegrama de comunicaciones.

11

Variadores y Profibús DP

11.1 El protocolo USS.

El protocolo USS es el utilizado para las comunicaciones entre los variadores de frecuencia micro / midimaster y los autómatas programables. A través de Profibús DP este protocolo se simplifica, ya que las funciones de cabecera de telegrama y control de errores (BCC) desaparecen, siendo realizadas por el adaptador de comunicaciones que se debe de colocar al variador de frecuencia.

Este adaptador (OPMP2) no solo se encarga de realizar la implementación de la cabecera y final del protocolo USS, sino además realiza el interface entre la comunicación del puerto del variador, que va a 9600 baudios, y la red Profibús, que suele funcionar a 1,5 Mbaud. Además, no es necesario indicar el número de estación que emite el telegrama, ya que al habérselo indicado en un parámetro del variador, la tarjeta lo adiciona al mismo antes de emitirlo a la red, y al contrario, capta de la misma todos aquellos que provengan del PLC por DP y sean para su número de estación.

El presente capítulo se centrará exclusivamente en las particularidades de la comunicación en DP. En este ámbito de actuación, el protocolo USS se puede implementar de cinco maneras distintas, de las cuales la comunicación con los micromaster / midimaster solo soporta dos:

- PPO1: Subdivisión del protocolo USS que permite en un solo telegrama controlar el variador (enviar consigna, arrancar o parar e indicar el sentido de giro), y además leer o escribir un solo parámetro cada vez. Necesita para su funcionamiento 12 bytes de entradas y 12 bytes de salidas en periferia del autómata.
- PPO3: Subdivisión de USS que solo permite controlar el variador, sin posibilidad de modificar ni leer parámetros del mismo por comunicaciones. Necesita para su funcionamiento solo 4 bytes de entradas y 4 de salidas en periferia.

	PKW				PZD	
	PKE	IND	PWE		STW	HSW
					ZSW1	HIW
PPO1	WORD	WORD	DWORD		WORD	WORD
PPO3					WORD	WORD

Figura 78 . Estructura de los telegramas de un micromaster/midimaster tanto en envío como en recepción.

11.1.1 Telegrama de emisión del PLC al variador.

Vamos a explicar el significado de cada una de estas palabras de los telegramas, así como su composición y su utilidad.

- **PKE:** Identificador de parámetro. Esta palabra se usa para indicar al convertidor un número de parámetro, y la acción que debe de realizar con el mismo, ya sea leerlo o modificarlo. La estructura de este parámetro es la siguiente:
  - Del bit 0 al 10 se almacena el número de parámetro a modificar en formato binario.
  - El bit 11 siempre debe estar a 0.
  - De los bits 12 a 15 se indica la acción a realizar con dicho parámetro según la regla:

Acción	Bit 15	Bit 14	Bit 13	Bit 12
Sin acción	0	0	0	0
Leer parámetro	0	0	0	1
Escribir parámetro en RAM y EEPROM	0	0	1	0
Sin acción	0	0	0	0

Por ejemplo, si deseamos leer el parámetro 3 de un variador, deberemos de indicar en la palabra PKE lo siguiente:

Bits:	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Binario:	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	1
Valor:		2			0						3					

- **IND:** Subíndice de parámetro. Esta palabra indica, si el parámetro del variador indicado en PKE lo tuviera, el número del subíndice del mismo. En micro/midimaster no existen actualmente índices, por lo que su valor siempre será 0. En caso de que existiese algún parámetro con subíndices, el número de los mismos se almacena en el byte alto de la palabra, es decir, en los bits de 8 al 15, permaneciendo el byte bajo a 0.

- **PWE:** Valor del parámetro, en el caso de que deseemos escribirlo. Esta doble palabra de 32 bits almacena un valor que, en el caso de que estemos escribiendo un parámetro indica el dato al que deberemos escribirlo. En caso de lecturas de parámetros, aunque coloquemos cualquier valor en esta doble palabra no tendrá efecto. Es importante tener en cuenta la cantidad de decimales que acepta dicho parámetro. Existen parámetros con dos decimales y parámetros con uno, por lo que es importante consultar el manual antes de introducir este parámetro.

- **STW:** Palabra de control. Esta palabra determina como debe de comportarse el variador. Cada uno de los bits que la componen indicará un estado del mismo. Repasemos cada uno de ellos:

BIT	Valor	Significado
0	1	El variador se pone en funcionamiento.
	0	El variador se para siguiendo la rampa de deceleración.
1	1	Frenado 2 no está activado.
	0	El variador para por frenado 2, es decir, se desconecta la alimentación al motor inmediatamente, sin seguir la rampa de deceleración, y el variador parará por inercia.
2	1	Frenado 3 no está activado.
	0	Se produce un frenado 3, esto es, una parada breve. Si el tiempo de la rampa de deceleración (parámetro 3) es menor de 10 segundos, se reduce a la mitad de dicho valor. Si el tiempo especificado en este parámetro de deceleración es superior a 10 segundos, se produce una parada en 5 segundos.
3	1	Convertidor habilitado.
	0	Convertidor deshabilitado. El variador no generará pulsos de salida.
4	1	Parada rápida por inyección de continua desactivada.

	0	Fast Stop. Se produce una parada tan rápida como sea posible por inyección de continua.
5	1	Generador de rampa habilitado. El variador tiene liberada la consigna, que puede ser modificada..
	0	Generador de rampa deshabilitado. El variador no varía su consigna, manteniéndola en el último valor que tenía antes de poner este bit a 0.
6	1	Consigna habilitada.
	0	Consigna deshabilitada. La consigna del variador se toma como 0.
7	1	Un flanco positivo en este bit realiza un reconocimiento de un error que pudiera haberse producido anteriormente en el variador. Inmediatamente después, el variador pasa al estado arranque deshabilitado, por lo que hay que volver a habilitarlo.
	0	
8	1	Jogging a derechas.
	0	
9	1	Jogging a izquierdas.
	0	
10	1	Palabra de control válida. Indica que esta palabra debe de ser realizada por el variador.
	0	Palabra de control no válida. No se debe de acatar esta palabra por parte del variador.
11	-	Siempre a cero. Sin significado.
12	-	Siempre a cero. Sin significado.
13	-	Siempre a cero. Sin significado.
14	1	Giro a derechas.
	0	Giro a izquierdas.
15	-	Siempre a cero. Sin significado.

- **HSW:** Consigna de frecuencia. En esta palabra se indica en hexadecimal el valor de consigna de frecuencia al que deseamos que marcha el variador. Es importante tener en cuenta que este valor no se almacena en ningún parámetro del mismo, por lo que se puede hablar de una consigna en local (ya sea a través de analógicas, o por frecuencias fijas), y una consigna de comunicaciones. El variador

solo obedecerá a una de ellas. El valor que se transmite en este parámetro está escalado de la siguiente manera:

- De 0 a 16384 (4000 en hexadecimal), que corresponde al 100% de la frecuencia que esté almacenada en el parámetro P094 del variador. Por defecto este valor es 50.0 Hz, aunque podemos modificarlo en cualquier momento, variando nuestra escala de consignas. Suponiendo un valro de 50.0 Hz en el parámetro, para valores superiores en el HSW de 16384 obtendríamos frecuencias superiores a 50 Hz. Así, el valor 32767, que es el valor positivo mayor que se puede introducir en una palabra, equivale a 100 Hz. Si sobrepasamos este número (valores mayores de 32767), obtendremos porcentajes negativos, y por consiguiente frecuencias negativas. La fórmula que resulta del cálculo de la frecuencia en el HSW es la siguiente:

$$F = (\text{consigna} * P094)/16384$$

- El valor resultante debe de expresarse en hexadecimal.

11.1.2 Telegrama de recepción del PLC procedente del variador.

Si hemos enviado el telegrama correctamente según lo explicado en el apartado anterior, unos milisegundos después el variador deberá contestar un telegrama. Veamos el significado de cada una de las partes del mismo:

- **PKE:** Respuesta de número de identificación de parámetro. Al igual que en el mensaje de emisión, esta palabra se estructura en tres partes:
  - Del bit 0 al 10, se almacena en binario el número de parámetro del cual se está devolviendo en este telegrama su valor.
  - El bit 11 siempre está a cero.
  - De los bits 12 a 15 se almacena la siguiente codificación:

15	14	13	12	Significado
0	0	0	0	No hay respuesta
0	0	0	1	Respuesta del valor de un parámetro
0	0	1	1	Respuesta del elemento de descripción
0	1	0	0	Sin uso en PPO1 y PPO3
0	1	1	0	Sin uso en PPO1 y PPO3
0	1	1	1	Tarea no ejecutable

1	0	0	0	La tarjeta de profibús no tiene mando exclusivo

- **IND:** Indice del parámetro. En esta palabra se indica el índice del parámetro cuyo valor se está contestando en el telegrama actual. Como los variadores micro/midimaster no poseen índices de parámetros, su valor en la respuesta del variador siempre será 0.
- **PWE:** Valor del parámetro devuelto. En esta doble palabra se recoge el valor actual del parámetro que se solicitó al variador. Si lo que se solicitó al convertidor fue la modificación de un parámetro, en el telegrama de respuesta en esta palabra se recoge dicho valor, para indicarnos que la acción se ha realizado.
- **ZSW1:** Palabra de estado del variador. Cada uno de los bits de dicha palabra posee un significado.

BIT	Valor	Significado	Nota
0	1	Listo para funcionar.	El convertidor ya está inicializado.
	0	Convertidor no “ready”.	El convertidor no ha sido inicializado.
1	1	Listo para arrancar.	Variador sin fallo y listo para arrancar.
	0	No preparado.	Causa: Hay un fallo o OFF2 o OFF3 están activos.
2	1	En servicio	La salida del convertidor tiene tensión.
	0	Sin servicio	La salida del convertidor se encuentra off.
3	1	Fallo	Existe actualmente un fallo en el variador. El número de error se indica en el parámetro de fallo del variador.
	0	Variador sin fallo.	
4	1	No activo OFF2	No se encuentra activada parada libre OFF2.
	0	OFF2 activa	El variador se para por parada libre OFF2.
5	1	No activo OFF3	No se encuentra activa parada rápida OFF3.
	0	OFF3 activa.	El variador hace una parada rápida OFF3.
6	1	Convertidor bloqueado	Arranque a través de OFF1 y luego ON.
	0	Convertidor desbloqueado	

7	1	Aviso	Aviso, con convertidor en funcionamiento. No es necesario realizar un acuse de un aviso. El variador no para y continua en servicio.
	0	Sin aviso.	
8	1	Siempre a uno. Sin significado.	
9	1	Control remoto	El variador permite el control remoto.
	0	Control local.	El variador indica que no acepta control remoto, ya que se encuentra parametrizado para control local.
10	1	Consigna alcanzada	El variador ha alcanzado la consigna de frecuencia.
	0	Consigna no alcanzada.	El variador aún no ha alcanzado la consigna indicada.
11	-	No usado.	
12	-	No usado.	
13	-	No usado.	
14	1	Giro a derechas.	El variador gira a derechas.
	0	Giro a izquierdas.	El variador gira a izquierdas.
15	-	No usado.	

- **ZSW:** frecuencia actual del variador. En este parámetro se refleja escalado siguiendo la misma relación que el parámetro HSW visto anteriormente la frecuencia a la que actualmente está marchando en cada momento el variador. Con esta lectura podemos saber no solo la velocidad actual del variador, sino el error en cada instante en las transiciones o cambios de consigna hasta alcanzar la misma. Es importante recordar que el valor que se obtiene se encuentra no solo escalado, sino también en BCD.

Con esto se finaliza la explicación de los protocolos tanto de transmisión (PLC->variador) como de recepción (variador ->PLC).

11.2 El OPMP2.

En el siguiente apartado nos centraremos en los parámetros que posee el variador para poder comunicar a través de Profibús DP, así como de sus capacidade de diagnóstico frente a errores.

11.2.1 Parámetros del OPM2.

Como vimos anteriormente, para comunicar con un variador en DP necesitaremos el módulo OPMP2. Una vez colocado en el mismo con el variador desconectado, conectaremos este a la red de alimentación, y activaremos el parámetro P099, que se encarga de habilitar todos aquellos parámetros del variador que tiene que ver con Profibús. Veamos una descripción de los mismos:

Par.	Función	Margen [ajuste fab.]	Descripción
P700	Versión del software del módulo Profibús.	00,00 – 99,99 [-]	Contiene el número de versión del software del módulo profibús y no puede ser modificado.
P701	Número de equipo.	0 – 255 [0]	Este parámetro asigna un número de equipo a cada uno de los convertidores. Atención: no es el número de esclavo DP.
P880	Datos de diagnóstico indexados	-	El OPMP2 almacena en cada uno de los índices de este parámetro los datos de diagnóstico (ver diagnóstico de comunicaciones).
P918	Dirección del esclavo DP	1 – 126 [126]	Este parámetro contiene el numero que identifica al variador dentro de la red DP.
P927	Ajuste de parámetros local/remoto.	0 – 1 [0]	0: Los parámetros pueden ser cambiados únicamente en modo local. 1: Los parámetros pueden ser

			cambiados únicamente vía Profibús DP.
P928	Mando local/remoto vía Profibús.	0 – 3 [0]	Mando local o remoto: 0: Control total local. 1: Control remoto total. 2: Control local para cambiar parámetros, pero remoto para controlar el variador y cambiar la consigna. 3: Control local para cambiar la consigna y controlar el variador, pero remoto para cambiar parámetros del mismo.
P947	Memoria de fallos	-	P947.0 : Contiene el último código de fallo no acusado. P947.1 a P947.7 : fijos a 0. P947.8 : Contiene el último código de error acusado. P947.9 a P947.15 : fijos a 0.
P958	Código de aviso	0 – 9999 [-]	Este parámetro indica el último aviso que se ha producido. Este valor no se almacena si se produce una desconexión de la alimentación del variador. Valores posibles: 2 : Límite de corriente alcanzado. 3 : Límite de tensión alcanzado. 4: Límite de desplazamiento alcanzado. 5: Sobretemperatura en motor.
P963	Velocidad de transmisión de Profibús DP	0 – 1500 [-]	En este parámetro se muestra la velocidad a la que está configurada la red de comunicaciones DP. Este valor es de sólo lectura, ya que es el maestro el que ajusta la velocidad de la red, y las tarjetas OPMP2 se adaptan a la misma automáticamente. Valores posibles de este parámetro son: 0 : velocidad desconocida. 1 : 9600 baud.

			2 : 19.2 Kbaud. 3 : 45,45 Kbaud 4 : 93,75 Kbaud. 5 : 187,5 Kbaud. 6 : 500 Kbaud. 7 : 1,5 Mbaud. 8 : 3,0 Mbaud. 9 : 6,0 Mbaud. 10 : 12,0 Mbaud.
P967	Palabra de mando	-	Muestra la última palabra de mando recibida.
P968	Palabra de estado	-	Muestra la última palabra de estado del variador.
P970	Reset a ajustes de fábrica	0 – 1 [1]	Resetear todos los parámetros del variador (excepto el P101) a sus ajustes de fábrica. Valores posibles: 0 : Reset de parámetros. 1: Sin efecto.
P971	Actualización de la EEPROM	0 – 1 [1]	Indica donde deben de almacenarse los cambios de parámetros que se realicen en el variador. Valores posibles: 0 : Cambios de parámetros almacenados solo en RAM. Si se desconecta la alimentación del equipo, al volver a conectar se cargarán en la RAM los últimos valores almacenados en la EEPROM. 1 : Cambio de parámetros almacenados en RAM y EEPROM. Una transición de 0 a 1 carga todos los parámetros de la RAM a la EEPROM.

11.2.2 Diagnóstico del OPMP2

Como se ha comentado anteriormente, en cada uno de los índices del parámetro P880 se almacena un buffer de diagnóstico al respecto del funcionamiento de la comunicación DP.

Veamos el significado de cada uno de los índices que posee este parámetro P880:

P88.i	Significado
P880.0	Contador de telegramas recibidos sin defecto. Cada vez que se recibe un telegrama sin error se incrementa este contador interno.
P880.1	Muestra el número de esclavo indicado en el parámetro P918.
P880.2	Número de bytes de identificación recibidos del maestro. Indica el tamaño de la cabecera que recibe el variador del maestro. Si este número no es 1 o 2, el variador generará un error F033, debido a que no consigue interpretar un inicio de telegrama válido.
P880.3	Número de bytes del PKW. Este valor debe de ser de 0 (para el protocolo PPO3) o de 8 (para el protocolo PPO1). En caso contrario, se genera un error F033.
P880.4	Número de bytes del PZD. Este valor deberá ser siempre 4, ya que para ambos protocolos (PPO1 y PPO3) el tamaño de esta zona es 4 bytes. En caso contrario, se genera un error F033.
P880.5	Tipo de protocolo PPO. Deberá valor o 1 o 3. En caso contrario, se genera un error F033.
P880.6	Contador de error hardware. Se incrementa en 1 cada vez que aparece un problema hardware a la hora de poder leer el protocolo de comunicaciones. Esto puede ser debido a interferencias electromagnéticas.
P880.7	Contador de CLEAR_DATA. Se incrementa en 1 cuando se recibe un telegrama de CLEAR_DATA (borrado de datos).
P880.8	Contador SYNC. Se incrementa en 1 cuando se recibe un telegrama SYNC.
P880.9	Identificador de grupo. Se almacena el identificador de grupo del telegrama recibido.
P880.10	Tiempo de vigilancia. Valor del tiempo de vigilancia de recepción de telegrama. Expirado este tiempo sin haber recibido ningún telegrama coherente, la tarjeta emite un error de fallo de comunicaciones.
P880.11	Contador de tiempo de vigilancia sobrepasado. Se incrementa en 1 este valor cada vez que se sobrepasa el tiempo de vigilancia, o lo que es lo mismo, cada vez que fallan las comunicaciones.
P880.12	Dirección del maestro Profibús dentro de la red DP al que está conectado el

	equipo.	
P880.13	Estado del esclavo. Puede adoptar los siguientes valores: 0 : Software aún no inicializado. 1 : CB15 esperando la parametrización de los parámetros concernientes a Profibús. 2 : CB15 esperando la configuración del número de estación Profibús. 3 : CB15 en funcionamiento cíclico. 4 : Tiempo de vigilancia de comunicaciones sobrepasado.	
P880.14	Velocidad de transmisión. Indica el valor de la velocidad de comunicaciones de la red.	
P880.15	Bits de avisos. Esta palabra indica si se está activando algún aviso dentro del variador. El significado de cada uno de los bits es el siguiente:	
	Bit 0	Recibido por parte del maestro un número de identificación incorrecto, es decir, el número de DP indicado en el variador y el que se le asigna por parte del
	Bit 1	Software Profibús aún no inicializado.
	Bit 2	Software de Profibús inicializado pero todavía no habilitado.
	Bit 4	El maestro ha recibido un número incorrecto de bytes de identificación (cabecera de protocolo). Se activa el fallo F033.
	Bit 5	El maestro ha recibido un número incorrecto de bytes de PKW o de PZD. Se activa el error F033.
	Bit 8	Velocidad de transmisión no detectada.
	Bit 9	Recepción de un telegrama CLEAR_DATA.
	Bit 10	CB15 en modo SYNC.
	Bit 11	Tiempo de vigilancia de comunicaciones transcurrido. Se activa el error F030.
	Bit 12	No se consigue conectar con el maestro. Aparece el error F030.

11.2.3 Códigos de alarmas.

Además de almacenar los errores en la anteriormente vista estructura de diagnóstico, el módulo OPMP2 fuerza en la pantalla del variador un código de averia según las siguientes reglas:

Código	Causa	Solución
F030	Fallo de las comunicaciones con el maestro de Profibús DP.	• Comprobar que está bien cableada la unión entre el módulo CB15 y el maestro. • Comprobar que la dirección del variador concuerda con la asignada en el equipo maestro para ese variador. • Comprobar que el maestro está en funcionamiento.
F031	Avería en el enlace entre módulo CB15 y variador.	• Comprobar que está bien montado el módulo al variador.
F033	Error de telegrama de Profibús.	• Comprobar que el maestro envía telegramas de tipo PPO1 o PPO3. • Comprobar las interferencias electromagnéticas.
F036	Fallos de programa.	• Desconectar la alimentación al variador y volver a conectarlo.

11.3 Función de control de variadores.

11.3.1 Parámetros de la función.

El bloque de función FB que vamos a presentar a continuación implementa de una manera sencilla todas las posibilidades de comunicación de los variadores micro/midimaster en una sola subrutina parametrizable. Al estar asociada a un DB, podremos controlar varios variadores simplemente realizando llamadas sucesivas a la misma, teniendo en cuenta que en cada una de ellas irá asociada a una DB diferente.

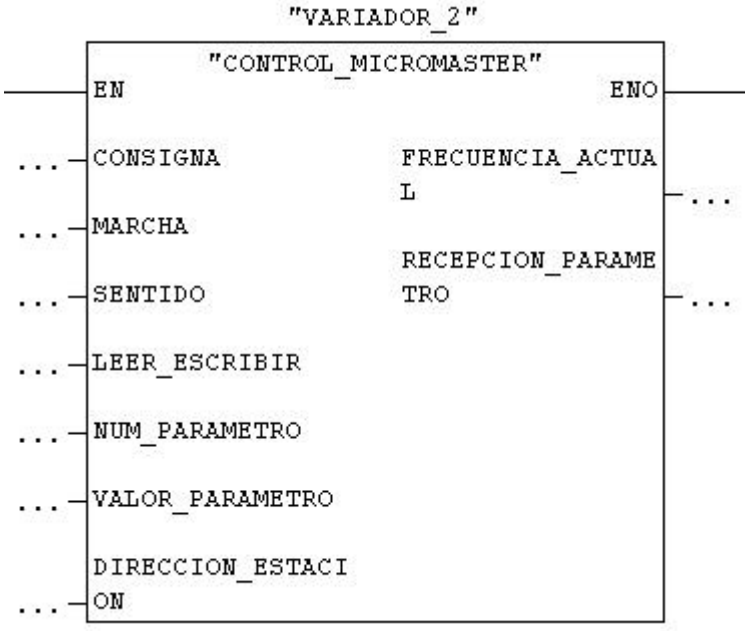


Figura 79 . FB de control variadores micromaster por DP.

Veamos cuales son los parámetros de la función y su significado:

Parámetros de entrada:

PARAMETRO	TAM.	DESCRIPCION
CONSIGNA	Real	Consigna de frecuencia del variador. Es la frecuencia a la que

		deseamos que funcione el variador micromaster. Este valor es distinto de la consigna de frecuencia en local, ya que la frecuencia del parámetro CONSIGNA se le va a introducir por Profibús, no se va a almacenar en ningún parámetro del micromaster.
MARCHA	Bit	Arranque o paro del variador a través de comunicaciones. Un valor TRUE en esta entrada arrancará el variador y uno FALSE lo parará con una rampa de deceleración dependiente del parámetro 3 del mismo (rampa de deceleración).
SENTIDO	bool	Sentido de giro del variador. Un valor TRUE hace girar el variador a derechas, mientras que un valor FALSE lo hace girar a izquierdas.
LEER_ESCRIBIR		Lectura o escritura de parámetros. Como sabemos, el protocolo USS posee la peculiaridad de permitirnos no sólo controlar el variador, sino también leer o escribir parámetros en el mismo, y lo que es más importante, todo ello en un solo telegrama. Por ello, pese a que en ocasiones no deseemos leer ni escribir en el variador ningún parámetro, es interesante utilizar esta propiedad del protocolo para obtener, p. Ej, la intensidad de consumo del mismo. Si en este parámetro colocamos TRUE escribiremos un parámetro, mientras que si colocamos FALSE leeremos un parámetro.
NUM_PARAMETRO	Int	Número de parámetro a leer o escribir, dependiendo de la elección anterior. Para conocer los parámetros del variador, vea el manual micro/midimaster, en el cual se realiza una descripción de los mismos.
VALOR_PARAMETRO	int	Valor del parámetro a modificar. Este parámetro sólo tiene sentido si seleccionamos LEER_ESCRIBIR = TRUE.
DIRECCION_ESTACION	int	Dirección de la estación DP. Para poder enviar o recibir el telegrama proveniente del variador, se requiere conocer el área de memoria asignada en periferia tanto de entradas como de salidas para dicho equipo. El número de estación DP asignado al variador no es válido para este menester, ya que en un equipo DP la asignación de periferia es variable, dando el sistema unas direcciones posibles (generalmente comenzando en el área analógica si está libre). Pero esta asignación, el sistema la adapta según existan ya zonas de periferia ocupadas. Además, se permite al usuario la posibilidad de variar dicha asignación de periferia para el variador desde *Hardware* de Step 7. Por todo esto, para poder comunicar con el variador necesitaremos dos cosas: - conocer la zona de comienzo de la periferia asignada al

		variador. - Que la zona de comienzo de entradas asignada al mismo sea el mismo byte que la zona de comienzo de salidas. En otras palabras, si el variador comienza su periferia de entradas en el byte PEB256, su periferia de salidas también deberá ser PAB256.

Parámetros de salida:

PARAMETRO	TAM.	DESCRIPCION
FRECUENCIA_ACTUAL	Real	Frecuencia actual a la que está marchando el variador en cada momento.
RECEPCION_PARAMETRO	Int	Si se ha solicitado en el telegrama de envío la lectura de un parámetro, en RECEPCION_PARAMETRO se devuelve dicho valor por parte de la función.

Al ser una FB, deberá de asociársele una DB. Si esa DB no existe en el proyecto en el momento de asignarla, Step 7 la genera mediante la estructura de parámetros de entrada y salida descritos.

11.3.2 Código de la función.

Además de los parámetros anteriormente descritos, en la tabla de declaración se utilizan las siguientes variables temporales:

0.0	temp	CONSIGNA_ESCALADA	INT	CONSIGNA ESCALADA DE 0 A 4000H
2.0	temp	PALABRA_DE_CONTROL	INT	PALABRA DE CONTROL DEL TELEGRAMA DE COMUNICACIONES
4.0	temp	IDP_ENVIO	INT	IDENTIFICADOR DE PARAMETRO
6.0	temp	IND_ENVIO	INT	INDICE DE PARAMETRO
8.0	temp	VAP_ENVIO	DINT	VALOR DEL PARAMETRO
12.0	temp	IDP_RECEPCION	INT	ACCION + NUMERO DE PARAMETRO ENVIADO POR EL VARIADOR
14.0	temp	IND_RECEPCION	INT	SIN EFECTO
16.0	temp	VAP_RECEPCION_1	INT	SIN EFECTO
18.0	temp	VAP_RECEPCION_2	INT	VALOR DEL PARAMETRO SOLICITADO AL VARIADOR
20.0	temp	TEMPORAL	DINT	VARIABLE UTILIZADA PARA CALCULAR LA DIRECCION DEL VARIADOR A PARTIR DE
24.0	temp	ERROR_ESCRITURA_COM	INT	ERROR ESCRITURA COMUNICACIONES PROFIBUS DP
26.0	temp	ERROR_LECTURA_COM	INT	ERROR LECTURA COMUNICACIONES PROFIBUS DP
28.0	temp	CONVERSION_TEMPORAL	WORD	ES NECESARIA UNA CONVERSION DE ENTERO A WORD, QUE SE REALIZA EN ESTA VA

FB1 : CONTROL VARIADOR MICROMASTER

ESTA FUNCION REALIZA EL CONTROL DE UN VARIADOR MICROMASTER MEDIANTE EL PROTOCOLO PPO1.

Segm. 1: ESCALADO DE CONSIGNA

ESCALADO DE LA CONSIGNA DE 0 A 100% DEL VALOR DEL PARAMETRO P94

```
L    #CONSIGNA                  // CARGAR CONSIGNA EN REAL
L    3.280000e+002              // 16384 DIVIDIDO ENTRE 50
*R                              // NOS DA EL VALOR EN HEXADECIMAL ESCALADO
TRUNC                           // ENTRE 0 Y 4000 EN HEXADECIMAL.
T    #CONSIGNA_ESCALADA

L    #DIRECCION_ESTACION        // CALCULO DE LA PALABRA DE PERIFERIA EN LA QUE SE
L    10                         // ENCUENTRA LA PALABRA DE CONSIGNA
+I                              // BASE + 10 = PALABRA DE CONSIGNA.
SLD  3                          // DEBEMOS DE DIRECCIONAR EN FORMATO DE PUNTERO,
T    #TEMPORAL                  // DESPLAZAMOS TRES POSICIONES A LA IZQUIERDA LA
                                // PALABRA PARA QUE TENGA EL FORMATO ADECUADO.

L    #CONSIGNA_ESCALADA         // TRANSFERIMOS LA CONSIGNA ESCALADA A LA PALABRA DE
T    PAW [#TEMPORAL]            // CONSIGNA DEL PROTOCOLO PPO1
```

Segm. 2 : GENERACION DE LA PALABRA DE CONTROL

ESTE SEGMENTO CREA LA PALABRA DE CONTROL SEGUN EL FORMATO DE PALABRA DE MANDO PMD DEL PROTOCOLO USS.

```
L    W#16#447E                  // ASIGNAMOS UNA CONFIGURACION BASE A LA PALABRA
T    #PALABRA_DE_CONTROL        // DE CONTROL TEMPORAL

U    #MARCHA                    // LAS VARIABLES TEMPORALES SON ACCESIBLES EN
                                // FORMATO DE BIT CON LA INSTRUCCION L
=    L    3.0                   // CARGA LA ORDEN DE MARCHA
U    #SENTIDO
=    L    2.6                   // CARGA EL SENTIDO DE GIRO

L    #DIRECCION_ESTACION        // LA PALABRA DE CONTROL DEL PPO1 ESTA A 8 BYTES DE LA
                                // DIRECCION BASE
L    8                          // POR LO QUE CALCULAMOS DE NUEVO EL PUNTERO A
                                // PARTIR DE LA MISMA.
+I
SLD  3
T    #TEMPORAL

L    #PALABRA_DE_CONTROL        // TRANSFERIMOS LA PALABRA DE CONTROL (MARCHA-PARO
```

```
                                    // Y SENTIDO DE GIRO)
T    PAW [#TEMPORAL]                // A LA PALABRA DE CONTROL DEL TELEGRAMA DE
                                    // COMUNICACIONES.
```

Segm. 3 : LEER O ESCRIBIR PARAMETROS

```
EL TELEGRAMA USS PERMITE NO SOLO CAMBIAR CONSIGNA Y ESTADO DEL VARIADOR, SINO
TAMBIEN LEER PARAMETROS O ESCRIBIR LOS MISMOS.
EL SIGUIENTE SEGMENTO SE ENCARGA DE ESTAS ACCIONES. TENER EN TAL COMO SE HA
HECHO ESTE EJEMPLO (VALOR BOOLEANO DEL PARAMETRO DE ENTRADAS LEER_ESCRIBIR,
SIEMPRE QUE CAMBIEMOS CONSIGNA O ESTADO ESTAMOS LEYENDO O CAMBIANDO UN
PARAMETRO.
ESTA ACCION NO RALENTIZA LAS COMUNICACIONES, AUNQUE SI SE DESEA NO REALIZARLA,
SERA NECESARIO MODIFICAR LA CABECERA DE DICHO PARAMETRO PARA QUE NO SEA
BOOLEANO, Y ENVIAR EN LA IDP_ENVIO UN 0 EN LUGAR DE UN 1 O UN 2 QUE SON LEER O
ESCRIBIR RESPECTIVAMENTE.
```

```
L    #NUM_PARAMETRO             // EN LA PALABRA IDP_ENVIO SE MEZCLA EL NUMERO DE
                                // PARAMETRO (PRIMEROS 12 BITS)
T    #IDP_ENVIO                 // POR LO QUE PRIMERO CARGAMOS EL NUMERO DE
                                // PARAMETRO Y LUEGO LO QUE DESAMOS HACER CON EL.

U    #LEER_ESCRIBIR             // SI DESEAMOS LEER, LOS ULTIMOS 4 BITS SERAN 001 (1),
=    L     4.5                  // MIENTRAS QUE SI DESAMOS ESCRIBIR...

UN   #LEER_ESCRIBIR             // LOS ULTIMOS 4 BITS SERAN 0010 (2). EN EL EJEMPLO NO SE
                                // COMPLETA, PERO
=    L     4.4                  // SI NO DESAMOS LEER NI ESCRIBIR HABRIA QUE TRANFERIR
                                // UN 0000 (0).
L    #VALOR_PARAMETRO           // SI DESEARAMOS GRABAR, EN VALOR PARAMETRO
                                // TENEMOS EL VALOR AL CUAL QUEREMOS
T    #VAP_ENVIO                 // QUE CAMBIE NUMERO DE PARAMETRO. SI QUEREMOS LEER,
                                // ESTE PARAMETRO NO HACE NADA.

L    #DIRECCION_ESTACION
T    #CONVERSION_TEMPORAL

CALL  "DPWR_DAT"                // MEDIANTE LA SFC15 ENVIAMOS 8 BYTES CONJUNTAMENTE A
                                // PROFIBUS DP
LADDR  :=#CONVERSION_TEMPORAL   // INDICAMOS LA DIRECCION DEL VARIADOR
RECORD :=P#L 4.0 BYTE 8         // INDICAMOS QUE ENVIAMOS IDP_ENVIO + IND_ENVIO +
                                // VAP_ENVIO
RET_VAL:=#ERROR_ESCRITURA_COM   // PALABRA DE ERROR DE COMUNICACIONES
```

Segm. 4: RECEPCION DE ESTADO DEL VARIADOR Y PARAMETROS A LEER

RECIBE EL ESTADO DEL VARIADOR, LO DESCOMPONE Y OBTIENE ADEMAS EL VALOR DEL PARAMETRO, EN EL CASO DE QUE SE HUBIESE SOLICITADO LA LECTURA DE UN VALOR.

```
CALL  "DPRD_DAT"                  // MEDIANTE LA SFC14 LEEMOS 8 BYTES DE PROFIBUS DP
LADDR  :=#CONVERSION_TEMPORAL     // DIRECCION DEL VARIADOR DEL QUE LEEMOS.
RET_VAL:=#ERROR_LECTURA_COM       // VARIABLE ERROR LECTURA COMUNICACIONES
RECORD :=P#L 12.0 BYTE 8          // RECEPCION DE IDP_RECEPCION + IND_RECEPCION +
                                  // VAP_RECEPCION_1 + VAP_RECEPCION_2

L     #VAP_RECEPCION_2            // VALOR DEL PARAMETRO ENVIADO POR EL VARIADOR. LA
                                  // VARIABLE
T     #RECEPCION_PARAMETRO        // DEL PROTOCOLO ES DOBLE ENTERO, PERO TODOS LOS
                                  // PARAMETROS A LEER SON ENTEROS POR LO QUE
                                  // COGIENDO LA PARTE BAJA (VAP_RECEPCION_2) ES
                                  // SUFICIENTE, Y MAS FACIL DE DEFINIR EN LA LLAMADA A LA
                                  // FUNCION.

L     #DIRECCION_ESTACION         // LECTURA DE LA FRECUENCIA ACTUAL DEL VARIADOR.
L     10                          // CALCULAMOS BASE + 10 = DIRECCION DE PALABRA DE
                                  // ESTADO DE LA FRECUENCIA ACTUAL DEL VARIADOR.
+I
SLD   3
T     #TEMPORAL
L     PEW [#TEMPORAL]             // LEEMOS LA PALABRA DEVUELTA POR EL VARIADOR, EN LA
                                  // QUE SE ENCUENTRA SU FRECUENCA ACTUAL, ESCALADA
                                  // DE 0 A 4000H.

L     500                         // CALCULAMOS EL VALOR PROPORCIONAL PARA ESCALARLO
                                  // A 50 HERCIOS.
*D
L     L#16384
/D
DTR                               // CONVERTIMOS EL VALOR A REAL
L     1.000000e+001               // APLICAMOS UNA DIVISION POR 10 PARA QUE APAREZCA EL
                                  // PUNTO DECIMAL.
/R
T     #FRECUENCIA_ACTUAL          // FRECUENCIA ACTUAL DEL VARIADOR ESCALADA DE 0 A 50
                                  // HERCIOS.
```

Observar que las llamadas a las SFC14 y SFC15 son necesarias para poder acceder al telegrama de la periferia DP manteniendo la consistencia en los datos, es decir, asegurándonos de que todos los bytes leídos pertenecen al mismo ciclo de lectura. Si accedemos copiando directamente los bytes, p. Ej., a marcas, o escribiendo el telegrama en las

palabras de salida de periferia, en el instante de la transición puede ser que solo se haya actualizado una parte de la periferia, por lo que el telegrama se mezcla con el que existiera anteriormente. Es para evitar esto, por lo que se gastan estas llamadas a las funciones de sistema.

También será necesario disponer de la OB122 y la OB86 cargadas en nuestro PLC, si no deseamos que ante un fallo de comunicaciones el autómata pase a Stop. Dichos bloques de organización se encuentran en la librería Standard de Step 7.

11.4 Como configurar la comunicación.

Una vez visto el protocolo y la función de comunicación para variadores, veamos como configurar el variador y el PLC para que podamos intercambiar datos entre ellos a través de DP.

11.4.1 Parametrización del maestro S7.

En el maestro tan solo es necesario asignar el esclavo a la red DP, dentro de Hardware de nuestro proyecto. Es importante recordar el byte de inicio de periferia que se ha asignado al telegrama del variador, para poder después asignárselo al parámetro DIRECCION_ESTACION de la función FB1.

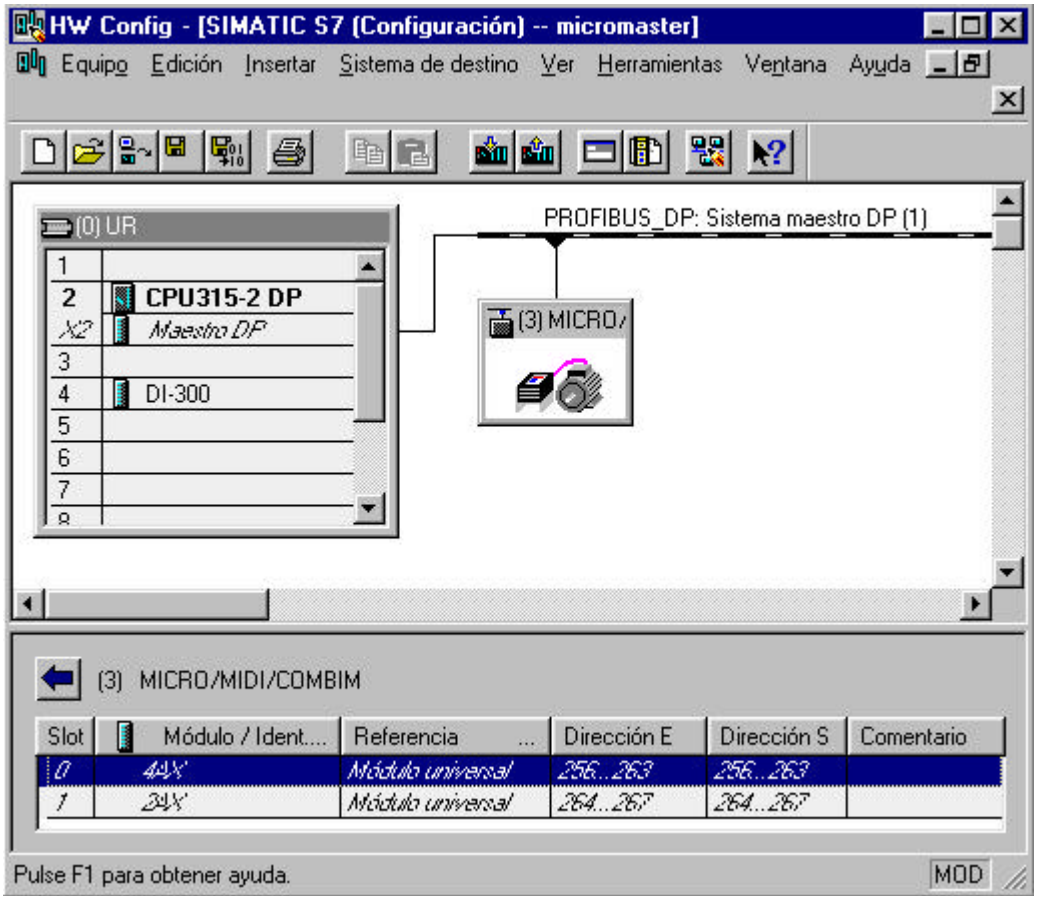


Figura 80 . Parametrización Hardware de un variador desde Step 7. Obsérvense el byte de inicio del telegrama de recepción (PEB256) y de envío (PAB256).

Dentro del catálogo, se puede encontrar el módulo CB15 en:

Catálogo ->Profibús-DP-> Simovert -> Micro-Midi-Combimaster CB15/CB155

11.4.2 Parametrización del variador.

Para que podamos modificar parámetros a través de DP, el P927 debe de estar a 1. Pero deberemos de introducirlo el último, ya que si no es así, a partir de ese momento no podremos modificar en local los siguientes parámetros del variador.

El valor del parámetro P918 debe de coincidir con el número de estación que se haya asignado en Hardware de Step 7.

Si deseamos controlar y cambiar parámetros desde comunicaciones DP, el P928 deberemos de ponerlo a 1. Ahora ya podemos colocar el P927 a 1 para tener comunicación con el maestro DP.

11.4.3 Pasos en la comunicación.

El variador micro/midimaster para poder gobernarse correctamente desde comunicaciones debe de seguir un orden en la recepción de su palabra de control.

Primeramente, para poder estar preparado para arrancar a través de comunicaciones, debe de recibir la palabra de control:

- Giro a derechas: 447E.
- Giro a izquierdas: 047E.

El significado es el siguiente:

Hex.	Bit	Valor	Significado
E	0	0	No arrancar variador
	1	1	Parada OFF2 no habilitada.
	2	1	Parada OFF3 no habilitada.
	3	1	El variador puede marchar
7	4	1	Parada por rampa normal.
	5	1	Consigna liberada.
	6	1	Consigna liberada.
	7	0	Sin acusar alarmas.
4	8	0	Sin jogging a derechas.
	9	0	Sin jogging a izquierdas.
	10	1	La palabra de control es válida.
	11	0	Sin significado.
4	12	0	Sin significado.

	13	0	Sin significado.
	14	1	Giro a derechas.
	15	0	Sin significado.

Partiendo de este estado, cambiando el último bit arrancaremos el variador, es decir:

- Para girar a derechas: 447F.
- Para girar a izquierdas: 047F.

Si una vez en marcha deseamos parar el variador, podemos realizar tres tipos de parada: la normal (447E), la parada libre (447D), y la parada rápida (447B). En cualquier caso, siempre deberemos de partir de un estado de parada normal (OFF1). Por lo tanto, si realizamos parada OFF2 o OFF3, a continuación, y durante algunas décimas de segundo, es necesario enviar la parada OFF1.

Indicar como apunte final que si deseamos realizar movimiento mediante jogging, la palabra de control debe de adoptar lógicamente el valor:

- Para girar a derechas: 057F
- Para girar a izquierdas: 067F.

11.4.4 ¿Cómo realizar las llamadas a la FB1.

Las llamadas a la FB1 deberán de realizarse preferiblemente desde la OB35 para no sobrecargar las comunicaciones de DP. Además es interesante en la OB100 realizar una llamada a la FB1 parando todos los variadores. Así, si la CPU pasa a stop, o cae la alimentación al PLC, al volver a RUN parará todos los variadores e inicializará las comunicaciones para poder volver a gobernarlos a través de DP.

```
CALL  FB     1 , DB1                    // LLAMADA A LA FB1 GENERANDO LA DB1 ASOCIADA.
CONSIGNA          :=3.550000e+001       // CONSIGNA DEL VARIADOR EN HERCIOS
MARCHA            :=E0.0                // PARO-MARCHA DEL VARIADOR  CON LA E0.0
SENTIDO           :=TRUE                // SENTIDO DE GIRO DEL MOTOR A DERECHAS
LEER_ESCRIBIR     :=FALSE               // LEER EL PARAMETRO 2
NUM_PARAMETRO     :=2                   // VALOR DEL PARAMETRO A LEER
VALOR_PARAMETRO   :=0                   // COMO LEEMOS UN PARAMETRO, NO GASTAMOS VALOR A
                                        // ESCRIBIR
DIRECCION_ESTACION :=256                // ESTE VALOR CORRESPONDE AL ASIGNADO A LA ESTACION
                                        // DP EN HARDWARE
FRECUENCIA_ACTUAL  :=MD0                // NOS DEJA LA FRECUENCIA ACTUAL EN LA MD0 EN VALOR
                                        // REAL
RECEPCION_PARAMETRO:=MW4                // NOS DEJA EL VALOR DEL PARAMETRO 2 (RAMPA DE
                                        // DECELERACION) EN LA MW4
```

12

El tiempo en S7

12.1 Fecha y hora

12.1.1 Ajustar la fecha y hora desde el software Step 7.

El primer paso para poder realizar un tratamiento del reloj de los Simatic S7 se basa en tener ajustada la fecha y hora actuales e la CPU. Esta acción, como veremos más adelante, se puede realizar desde programación, pero en un primer instante es mucho más cómodo el realizar un primer ajuste desde el software de Step 7. En el apartado:

Administrador Simatic -> Sistema Destino -> Ajustar la Hora...

En el mismo podemos incluso igualar la hora de nuestro PLC con la actual del ordenador.

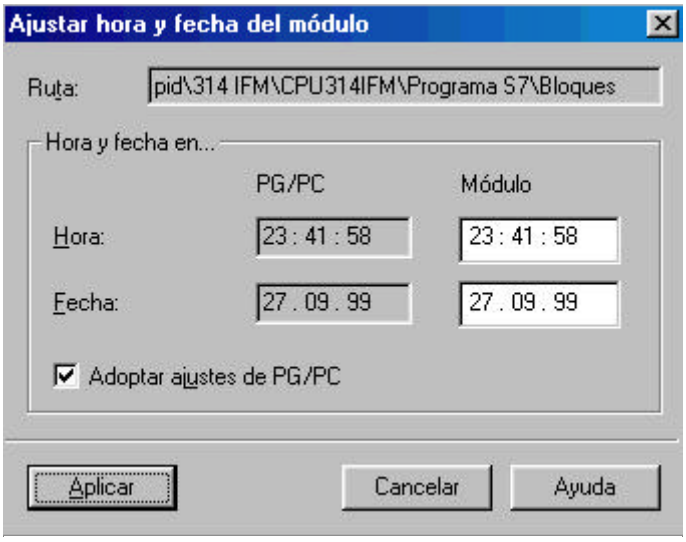


Figura 81. Ajusta la fecha y hora del PLC desde Step 7.

12.1.2 Leer la fecha y hora actual.

Los Simatic S7 300/400 disponen de la SFC1 para la lectura de la fecha y hora actual de aquellas CPU's que posean reloj hardware. Antes de realizar la llamada a dicha función de sistema, es necesario generar en una DB una variable de tipo DATE_AND_TIME. Dicha variable se compone de 8 bytes, en los cuales se va a almacenar tanto la fecha como la hora actual del PLC.

Después de haber generado dicha variable, es necesario simbolizar la DB en cuestión, ya que para poder direccionar una variable superior a una doble palabra (4 bytes), es necesario realizarlo de manera simbólica.

Para que la lectura del reloj sea más precisa, es conveniente no realizarla desde la OB1, cuyo período de ejecución depende del tiempo de ciclo, y acceder a la misma desde la OB35, que se ejecutará a tiempos predefinidos (por defecto 100 ms).

Suponiendo que se haya creado la DB10 bajo el simbólico FECHAS, la llamada a misma será la siguiente:

```
CALL  "READ_CLK"
     RET_VAL:=MW0                              // marca de error de lectura de reloj
     CDT    :="FECHAS".FECHA_HORA_ACTUAL       // variable DATE_AND_TIME
```

En realidad una variable DATE_AND_TIME posee la siguiente estructura:

BYTE	SIGNIFICADO
0	Año.
1	Mes.
2	Día
3	Hora
4	Minuto
5	Segundo
6	Parte alta de las milésimas de segundo.
7	Parte baja de las milésimas de segundo + día de la semana.

Los bytes 6 y 7 se estructuran de la siguiente manera:

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Milésimas de segundo actuales												Día de la semana			

Todos estos valores se expresan en hexadecimal, lo cual hay que tener en cuenta a la hora de visualizar dichos valores desde un panel de operador o un sistema scada.
Destacar que el día de la semana guarda en principio la norma americana semanal: el domingo es 1, el lunes 2, y así hasta el sábado que es 7. Este valor de día de la semana no se puede modificar, por lo que cualquier operación que se desee realizar con los días de la semana deberá de tener esto muy en cuenta.

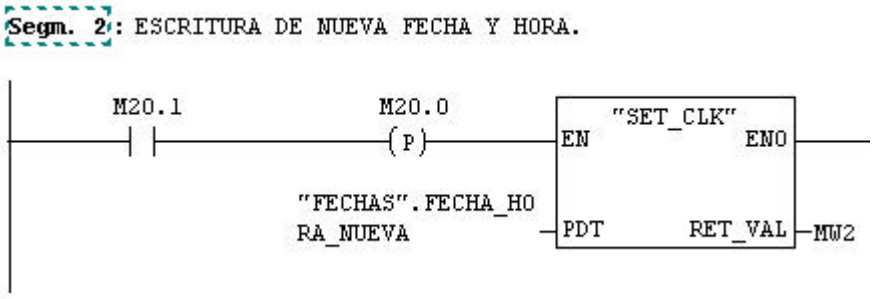
El parámetro RET_VAL contiene una palabra de error. Si el valor de la misma es distinto de 0 la fecha y hora actuales no son válidas. Para el tratamiento de errores ver apartado de errores generales.

12.1.3 Modificar la fecha y hora actual.

La SFC0 se encarga de modificar la fecha y hora actuales de nuestra CPU. Para ello son necesarias dos cosas:

- Indicarle una variable de tipo DATE_AND_TIME donde se encuentre la fecha y hora nueva deseada.
- Que la llamada a la SFC0 se efectue con un flanco, ya que en caso contrario siempre estaremos escribiendo la nueva fecha, con lo que impediremos que avance el reloj del PLC.

Una posible llamada a la función sería la siguiente:



Esta llamada no es necesaria hacerlo lógicamente a intervalos regulares, por lo que se puede colocar en cualquier FC que se esté llamando desde la OB1, o en la misma OB1.

El parámetro RET_VAL contiene una palabra de error. Si el valor de la misma es distinto de 0 la fecha y hora de seteo no son válidas. El autómata S7 posee un calendario interno, el cual le indica si la fecha u hora que estamos intentando introducir es válida. En caso contrario, genera uno de los dos posibles tipos de error:

- RET_VAL = 8080 (hex.) o –32640 (dec.)-> fecha no válida.

- RET_VAL = 8081 (hex) o –32641 (dec)-> hora no válida.

Por supuesto, si el valor no es válido, el PLC no lo acepta. A continuación se propone un tratamiento de error, suponiendo que alguien introduce una fecha y hora errónea desde una OP o scada. El código lo detecta, y activa una alarma según sea el error.

```
Segm. 3 : TRATAMIENTO ERROR DE INTRODUCCION DE FECHA
      L     MW     2                    // carga RET_VAL
      L     -32640                      // CARGA CONSTANTE FECHA ERRONEA
      ==I                               // SI SON IGUALES
      S     "alarma_fecha_erronea"      // ACTIVA ALARMA FECHA ERRONEA

Segm. 4 : TRATAMIENTO ERROR DE INTRODUCCION DE HORA
      L     MW     2                    // CARGA RET_VAL
      L     -32641                      // CARGA CONSTANTE HORA ERRONEA
      ==I                               // SI SON IGUALES
      S     "alarma_hora_erronea"       // ACTIVA ALARMA HORA ERRONEA
```

Figura 82 . Tratamiento de errores modificación de fecha en PLC.

12.1.4 Sincronización de relojes en S7.

En aquellos sistemas de automatización en los cuales se distribuya el control en varios PLC's S7 conectados a un sistemas de visualización, suele ser interesante la sincronización de los relojes de los diferentes PLC's. Esto nos permitirá modificar la fecha y hora actual de uno de ellos desde la OP o el scada, y que esta modificación quede reflejada en los demás relojes de las restantes CPU's de la instalación.

Para poder realizar la sincronización de los diferentes relojes de los equipos, primeramente deberemos de definir uno de ellos como reloj maestro del bus de comunicaciones. Esto se realiza desde Hardware de Step 7.

Con esto hemos definido un reloj maestro y los demás relojes de la red como esclavos. El intervalo de tiempo entre una sincronización y otra puede tomar los siguientes valores:

- 1 segundo.
- 10 segundos.
- 1 minuto.
- 10 minutos.

- 1 hora.
- 12 horas.
- 24 horas.

Sin embargo, existe una manera de forzar la sincronización aunque no se cumpla actualmente el momento de la sincronización. Además de este establecimiento a nivel de DB's de configuración de hardware de los diferentes PLC's participantes de la red MPI, será necesario llamar a la SFC48 del sistema desde el PLC que hemos definido como maestro.

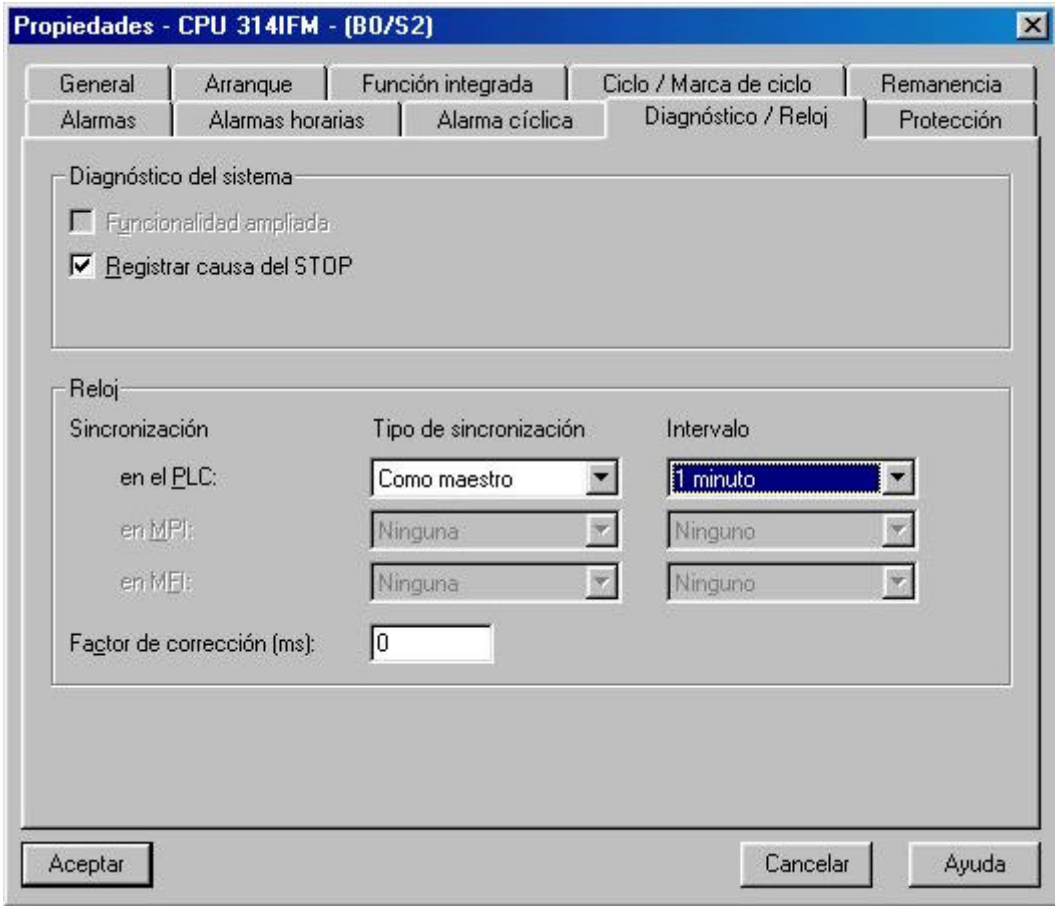


Figura 83 . Selección de la CPU maestra desde hardware de Step 7. Las demás cpu's deben de parametrizarse como esclavas.

La llamada a la función SFC48 es la siguiente:

```
CALL  "SNC_RTCB"
 RET_VAL:=MW0
```

Siendo el valor de RET_VAL:

- RET_VAL=0 -> no existió error en la sincronización de los relojes.

- RET_VAL=1 -> no se parametrizó el presente PLC como maestro del bus. Debe de ajustarlo dentro de hardware de Step 7.

12.2 Horas de funcionamiento.

12.2.1 Un contador de horas de funcionamiento.

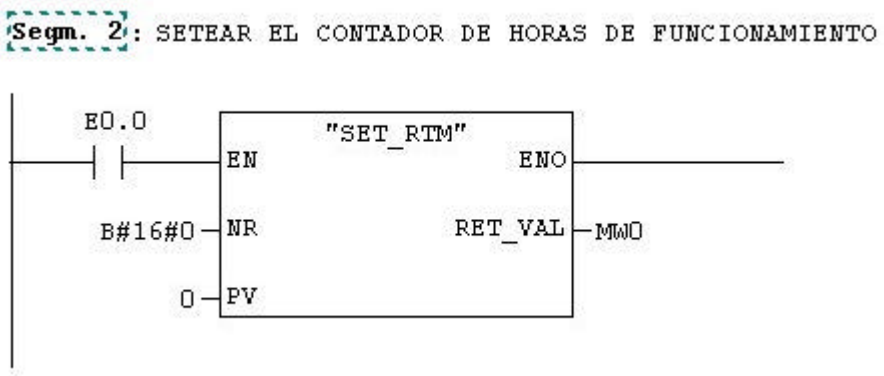
Un contador de horas de funcionamiento nos genera un valor que contiene la cantidad de horas que la CPU se encuentra en RUN, es decir, ejecutándose el proceso. Si durante un determinado tiempo se pasa a stop el equipo, el contador no volverá a incrementarse hasta que rearranquemos el PLC.

El contador de horas puede almacenar un máximo de 32767 horas.

Lógicamente, los contadores de horas de funcionamiento no se gastan para conocer cuanto tiempo estamos en RUN en una instalación, sino para asociarlos al funcionamiento de un motor y poder saber cuanto tiempo ha estado funcionando.

12.2.2 Setear el contador de horas de funcionamiento.

Lo primero que es necesario conocer en un contador de horas es como inicializarlo a un valor deseado. Este ajuste se realiza con la SFC2 del firmware de los PLC's S7.



Los parámetros de la función SFC2 son:

PARÁMETRO	TIPO	SIGNIFICADO
NR	BYTE	Número de contador de horas de funcionamiento. El equipo S7 dispone de 8 contadores, desde el 0 al 7.
PV	INT	Valor de preselección del contador. Puede valer de 0 a 32767.
RET_VAL	INT	Código de error de la función. Los valores posibles son: 0 : Sin error. 8080 (hex): El número del contador es erróneo.

		8081 (hex): Se ha parametrizado un valor de preselección negativo.

12.2.3 Controlar el contador de horas.

Podemos parar o arrancar el contador de horas de funcionamiento desde la SFC3 del sistema. El ejemplo siguiente muestra como ir contando el funcionamiento de un motor gobernado por la marca MOTOR_35 mediante la SFC3.

```
CALL  "CTRL_RTM"          // SFC3
 NR    :=B#16#0           // NUMERO DE CONTADOR
 S     :="MOTOR_35"       // ARRANCAR-PARAR EL CONTADOR
 RET_VAL:=MW0             // ERROR DEL CONTADOR
```

PARÁMETRO	TIPO	SIGNIFICADO
NR	BYTE	Número de contador de horas de funcionamiento. El equipo S7 dispone de 8 contadores, desde el 0 al 7.
S	BOOL	Activar el contador de horas.
RET_VAL	INT	Código de error de la función. Los valores posibles son: 0 : Sin error. 8080 (hex): El número del contador es erróneo.

12.2.4 Leer el contador de horas de funcionamiento.

Mediante la SFC4 podemos leer el valor actual del contador de horas de funcionamiento de nuestro motor.

```
CALL  "READ_RTM"                          // SFC4
 NR    :=B#16#0                           // NUMERO DE CONTADOR
 RET_VAL:=MW0                             // ERROR DEL CONTADOR
 CQ    :=M10.1                            // ESTADO DEL CONTADOR
 CV    :="CONTADORES".VALOR_CONTADOR_0
```

PARÁMETRO	TIPO	SIGNIFICADO

NR	BYTE	Número de contador de horas de funcionamiento. El equipo S7 dispone de 8 contadores, desde el 0 al 7.
RET_VAL	INT	Código de error de la función. Los valores posibles son: 0 : Sin error. 8080 (hex): El número del contador es erróneo. 8081 (hex): Desbordamiento del contador de horas de funcionamiento.
CQ	BOOL	Indica si el contador de horas de funcionamiento está en marcha actualmente..
CV	INT	Valor actual del contador de horas de funcionamiento. Suele almacenarse en una palabra de una DB.

12.2.5 Contar tiempo del sistema

En ciertas ocasiones puede ser interesante conocer el tiempo que transcurre entre dos eventos dentro del sistema operativo del S7. Un caso podría ser conocer el tiempo que tarda un telegrama de comunicaciones en ser contestado por el receptor desde que sale de nuestra CPU. También podría ser interesante el tiempo consumido por una subrutina en procesarse.

Para estos menesteres podemos gastar la SFC64, que nos indica el valor actual del reloj del sistema. Dicho reloj cuenta desde 0 hasta 2147483647 ms. (600 horas aprox). Transcurrido este tiempo comienza de nuevo a contar desde 0. La precisión del cronómetro es de 10 ms en los S7 300 y de 1 ms en los S7 400.

Cada vez que se produce un rearranque, el cronómetro del sistema pasa a 0 y comienza a contar, hasta que el autómata pasa a STOP.

```
CALL  "TIME_TCK"                          // SFC64
 RET_VAL:="CONTADORES".TIEMPO_INICIO      // LEER EL TIEMPO DEL SISTEMA

CALL  FC    1                             // EJECUTA UNA FUNCION

CALL  "TIME_TCK"                          // SFC64
 RET_VAL:="CONTADORES".TIEMPO_FIN         // LEE DE NUEVO EL TIEMPO DEL SISTEMA

L     "CONTADORES".TIEMPO_FIN             // RESTAMOS LOS DOS TIEMPOS
L     "CONTADORES".TIEMPO_INICIO
-D
T     "CONTADORES".TIEMPO_FUNCION         // EL TIEMPO QUE TARDÓ LA FUNCIÓN FC1
```

13

Redes AS-i

13.1 Tarjeta CP342-2.

13.1.1 Direccionamiento CP 342-2.

La tarjeta CP 342-2 es la tarjeta maestra para una red AS-i en los equipos S7 300. Para su direccionamiento utiliza 16 bytes de entradas de la periferia del PLC y 16 bytes de salida.

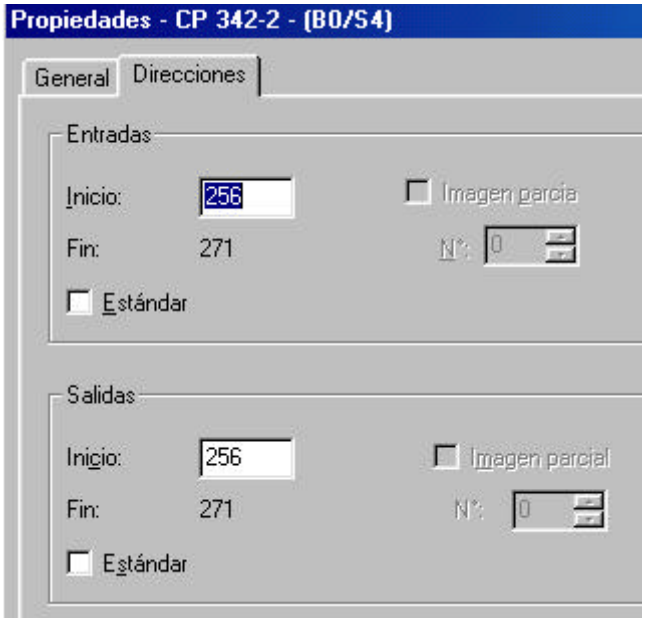
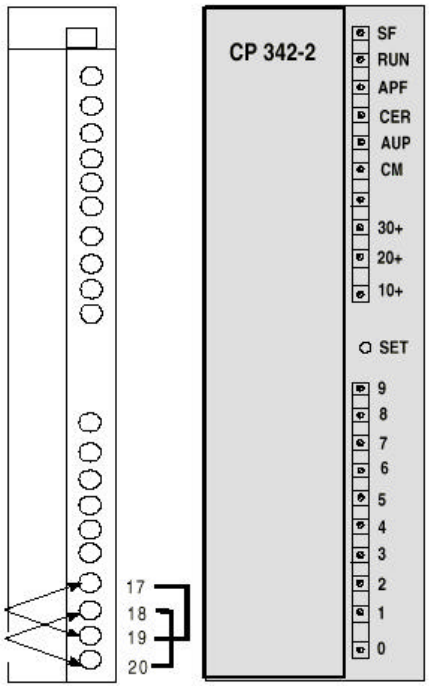


Figura 84 . Direccionamiento de la CP 342-2 en hardware de Step 7.

13.1.2 Cableado y leds de la CP 342-2.

El cableado frontal de la CP 342-2 es el que muestra la figura inferior. Recordar que es necesario un conector de 20 polos, no suministrado con la tarjeta.



El significado de los leds indicadores frontales es el siguiente:

- **SF:** Fallo de sistema. El led luce en las siguientes situaciones:
  1 . Cuando la CP está en modo protegido y existe un fallo de configuración, generalmente un fallo de esclavo configurado y no presente en la red.
  2 . Cuando la CP detecta un fallo interno en la EEPROM.
  3 . Cuando la CP no puede realizar el cambio de modo seleccionado en la configuración de switch de la tarjeta, p .ej, debido a la existencia de un esclavo con dirección de estación 0.
- **RUN:** Luce indicando que la CP ha arrancado correctamente.
- **APF:** Fallo de alimentación AS-i. Indica que el voltaje suministrado por la fuente AS-i es demasiado bajo o inexistente.
- **CER:** Error de configuración. Indica que la configuración de esclavos en la red AS-i difiere con respecto a la almacenada en la configurada en la CP. Esto puede ser

debido a:

1 . Un esclavo AS-i configurado no existe en el cable AS-i. Puede ser que el esclavo no exista, que esté estropeado o que no esté alimentado.

2 . Cuando existe un esclavo en la red AS-i pero no fue configurado en la tarjeta CP.

3 . Mientras la CP está en la fase de OFF-LINE.

- **AUP:** Autoprogramación disponible. Indica que la dirección de un esclavo puede ser programada automáticamente. En este modo la CP programa el esclavo 0 por el esclavo que no se encuentre presente en dicho momento en la red, pero que haya sido programado dentro de la CP. Es el caso en el que sustituimos un esclavo estropeado y conectamos uno nuevo que de fábrica viene con la dirección 0.
- **CM:** Modo de configuración. Si luce este led nos encontramos en modo de configuración, mientras que si no lo hace nos encontramos en modo protegido. En el modo de configuración la CP lee las direcciones de los esclavos conectados a la red y los almacena en la EEPROM de la tarjeta.
- **SET:** El botón de SET sirve para configurar la red AS-i en modo standard. Antes de presionar el botón es necesario colocar el PLC en stop. Además la CP debe de encontrarse en modo de configuración (luce el led CM). Si no lo está, al presionar el botón la CP pasa automáticamente a modo de configuración. Al presionar el botón SET durante al menos 0.5 segundos, se realizan en la tarjeta los siguientes pasos:

  1 . Las direcciones de los esclavos que se encuentran conectados en la red se almacenan permanentemente dentro de la tarjeta.

  2 . A continuación la CP cambia automáticamente a modo protegido (deja de lucir el led CM).
- **Leds de esclavos activos:** Existen en el frontal de la CP unos leds (del 30+ al 0) mediante los cuales se indica los esclavos activos actualmente en la red AS-i. Además gracias a ellos se puede conocer el fallo de uno de ellos, ya que comienza a parpadear el/los leds correspondientes. Los leds 10+, 20+ y 30+ indican el grupo de esclavos seleccionados, agrupados de 10 en 10. De esta manera, los esclavos del 0 al 9 se muestran cuando ninguno de los anteriores leds referidos lucen, del 10 al 19 cuando luce el led 10+, del 20 al 29 cuando luce el led 20+, y del 30 al 31 cuando luce el led 30+.

13.1.3 Modos de operación de la CP 342-2.

Existen dos modos de operación en la tarjeta:

- **Standard:** En este modo de operación, la CP 342-2 trabaja como un maestro AS-i convencional con tratamiento de entradas / salidas digitales / analógicas. En el modo standard de operación no es posible transferir a los esclavos parámetros especiales o comandos desde el maestro. Este modo corresponde al perfil M0 de las especificaciones AS-i.
- **Extendido:** En el modo de operación extendido, el maestro puede utilizar toda las funcionalidades extendidas del protocolo AS-i. Entre otras capacidades, es posible realizar un envío de comandos que modifique el número de esclavo de un determinado módulo desde el maestro. Este modo de operación corresponde al perfil M1 de las especificaciones AS-i. Para trabajar en modo extendido, es necesario disponer dentro de la CPU de la FC ASI_3422 que realiza estos envío especiales a la red AS-i.

13.1.4 Modo de configuración standard de la CP 342-2.

A diferencia del modo de configuración extendido, en el modo de configuración standard no es necesario disponer dentro de la CPU de ninguna FC para realizar las acciones posibles. Por ello, es el más utilizado para la tarjeta.

Existen dos modos de trabajo dentro del modo de operación standard:

- **Modo de configuración:** Utilizado para almacenar la configuración e los esclavos presentes en la instalación AS-i. Teniendo la CPU en stop, y presionando el botón de SET, se pasa de este modo al modo protegido.
- **Modo protegido:** En el modo protegido, la CP intercambia información únicamente con la lista de esclavos interna de su EEPROM. Indicará error si uno de ellos no responde a una petición de comunicación, pero no detecta la aparición de un nuevo esclavo en la red.

La forma de trabajar con los diferentes modos de operación es la siguiente:

- Colocar la CPU en stop.
- Presionar el botón de SET una vez, si no se encuentra la CP en modo de configuración (lucir el led CM).
- Es posible instalar o retirar esclavos e la red durante este periodo de tiempo.

- Una vez se haya completado la red AS-i, presionar el botón de SET nuevamente, con lo que se memoriza la configuración de esclavos actual, y la CP pasa a modo protegido (deja de lucir el led CM).

13.1.5 Direccionamiento de la CP 342-2.

Según el slot en el que se coloque la CP, adoptará una periferia del PLC u otra. En el siguiente gráfico se observa las zonas de memoria ocupadas para cada uno de los slots posibles:

Rack 0

Module	PS	CPU	IM	CP	CP	CP	CP	CP	CP	CP	CP
Slot number	1	2	3	4	5	6	7	8	9	10	11
Start address	1	2	3	256	272	288	304	320	336	352	368

Rack 1

Module			IM	CP	CP	CP	CP	CP	CP	CP	CP
Slot number			3	4	5	6	7	8	9	10	11
Start address				384	400	416	432	448	464	480	496

Rack 2

Module			IM	CP	CP	CP	CP	CP	CP	CP	CP
Slot number			3	4	5	6	7	8	9	10	11
Start address				512	528	544	560	576	592	608	624

Rack 3

Module			IM	CP	CP	CP	CP	CP	CP	CP	CP
Slot number			3	4	5	6	7	8	9	10	11
Start address				640	656	672	688	704	720	736	752

13.1.6 Direccionamiento de los esclavos en la CPU.

Cada uno de los esclavos configurados dentro de la CP ocuparán lógicamente 4 bits de la periferia asignada a la tarjeta. Los cuatro bits de entradas corresponderán a las entradas del esclavo, mientras que los cuatro de salidas a las salidas del mismo. Si un esclavo AS-i solo es de entradas, los bits correspondientes a las salidas no poseen ninguna relevancia, y al contrario. Igualmente ocurre con un módulo de 2 entradas/ 2 salidas, en los cuales los dos bits más significativos tanto de entradas como de salidas no tienen ninguna utilidad.

El mapeado de los esclavos en la memoria del PLC es el siguiente:

BYTE	7	6	5	4	3	2	1	0

n+0	RESERVADO	ESCLAVO 1
n+1	ESCLAVO 2	ESCLAVO 3
n+2	ESCLAVO 4	ESCLAVO 5
n+3	ESCLAVO 6	ESCLAVO 7
n+4	ESCLAVO 8	ESCLAVO 9
n+5	ESCLAVO 10	ESCLAVO 11
n+6	ESCLAVO 12	ESCLAVO 13
n+7	ESCLAVO 14	ESCLAVO 15
n+8	ESCLAVO 16	ESCLAVO 17
n+9	ESCLAVO 18	ESCLAVO 19
n+10	ESCLAVO 20	ESCLAVO 21
n+11	ESCLAVO 22	ESCLAVO 23
n+12	ESCLAVO 24	ESCLAVO 25
n+13	ESCLAVO 26	ESCLAVO 27
n+14	ESCLAVO 28	ESCLAVO 29
n+15	ESCLAVO 30	ESCLAVO 31

Los bits del 4 al 7 del primer byte de entradas únicamente se utilizan en modo extendido. En modo estándar, el sistema los coloca como 1000 o 1110 con una periodicidad de 2.5 segundos. Los bits de salida del 4 a 7 del primer byte no poseen ningún significado en ninguno de los modos de operación.

13.1.7 Lectura/escritura de esclavos en AS-i.

Una vez que conocemos donde se encuentran los bits correspondientes a nuestros esclavos AS-i, deberemos de leer las entradas y escribir las salidas de los mismos.

Para realizar esta tarea es necesario tener en cuenta dos requisitos:

1. al encontrarse en la zona de las analógicas, no poseen dichas entradas o salidas PAE ni PAA, por lo que se deberán de utilizar accesos directos a periferia. Lo normal es mapear las entradas y salidas en marcas del PLC.
2. No es posible realizar lecturas o escrituras de bytes sueltos, por lo que siempre nos dirigiremos a palabras o dobles palabras de la periferia de la CP 342-2.

Ejemplos de lecturas de entradas de esclavos serían:

```
L PEW 260
T MW20
```

```
L PED 260
T MD20

L MW40
T PAW260


L MD40
T PAD60
```

13.1.8 Diagnosis de errores en la CP 342-2.

Si se produce un error en la comunicación con alguno de los módulos AS-i o la CP reconoce un error interno propio, genera hacia la CPU un evento de error. Dicho evento, es tratado en la OB82. En el caso de que no exista dicha OB en el PLC, la cpu pasa automáticamente a stop.

Existen por lo tanto dos tipos de error, unos procedentes de un error en la propia tarjeta CP, y otros procedentes de un fallo en alguno de los módulos AS-i. El tratamiento para estas dos fuentes de errores, se realizará en la OB82, pero de diferente manera en cada caso.

Para el caso de un error en la tarjeta CP, deberemos de consultar el estado de las variables de la tabla de declaración de la OB82. A continuación veremos el significado de cada una de las variables, así como su dirección local, para su tratamiento dentro de la OB.

Dirección	Variable	Significado
L 8.0	OB82_MDL_DEFECT	TRUE: Fallo en el módulo.
L 8.1	OB82_INT_FAULT	TRUE: Error interno de la propia CP.
L 8.2	OB82_EXT_FAULT	TRUE: Error externo a la CP (generalmente fallo de algún esclavo o de la alimentación de la tarjeta).
L 8.3	OB82_PNT_INFO	Al menos un esclavo es difiere entre la configuración teórica de esclavos y la real.
L 8.4	OB82_EXT_VOLTAGE	El voltaje de alimentación de la fuente AS-i es demasiado bajo.
LB 9	OB82_MDL_TYPE	Si se ha generado la interrupción debido a la CP 342-2 el valor de esta variable es siempre 1C.
L 10.0	OB82_SUB_NDL_ERR	Al menos un esclavo es difiere entre la configuración teórica de esclavos y la real.
L 10.2	OB82_MDL_STOP	FALSE: la CP está en modo normal. TRUE: la CP se encuentra en off-line.
L 10.3	OB82_WTCH_DOG_FLT	Fallo de hardware en la CP.
L 11.2	OB82_EPROM_FLT	Fallo en la EEPROM de la CP.

El otro tratamiento de error que puede interesarnos es reconocer el esclavo que está fallando en cada momento para p. Ej. Mostrar un mensaje en una OP o scada. Para ello, es necesario realizar una llamada a la SCF 59 dentro de la OB82. Se le debe de transferir una zona de marcas de 11 bytes, de las cuales se utilizan:

- Del byte 0 al 3: se copian los bytes locales del 8 al 11 de la OB82, que acabamos de describir.
- Del byte 4 al 6: poseen siempre los valores 60 20 20
- Del byte 7 al 11: Estos bytes son los interesantes, ya que se encuentra en ellos el estado de los esclavos AS-i, indicándonos con un valor alto que se ha producido un error en dicho módulo. El orden de los esclavos va desde el 7.0 que es el esclavo 1 de la red, al 11.7, que es el esclavo 31.

Veamos que tratamiento se puede realizar dentro de la OB82 para conocer el estado de los esclavos AS-i.

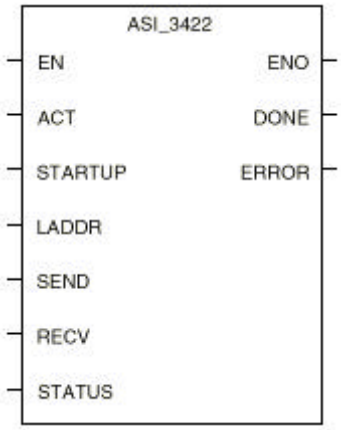
```
    L    #OB82_MDL_ADDR          // CARGA DIRECCION TARJETA CON ERROR
    L    B#16#1C                 // SI ES LA cp 342-2
    ==I
    SPB  err1
    BEA

err1: NOP  0

    S    #DISPARO                // ACTIVA LA CARGA DE DIAGNOSTICO

LOOP: CALL  "RD_REC"
      REQ   :=#DISPARO           // DISPARO DE DIAGNOSTICO
      IOID  :=B#16#54            // VALOR FIJO PARA LA CP 342-2
      LADDR :=W#16#1C            // VALOR FIJO PARA LA CP 342-2
      RECNUM :=B#16#1            // VALOR FIJO
      RET_VAL:=#RETORNO          // RETORNO DE FUNCION
      BUSY  :=#ACCION            // ACCION FINALIZADA
      RECORD :=P#M 10.0 BYTE 11  // ZONA DONDE SE ALAMACENARÁ EL DIAGNOSTICO

    CALL  "RE_TRIGR"             // REDISPARAR EL TIEMPO DE CICLO MIENTRAS REALIZA EL DIAGNOSTICO
    U    #ACCION                 // SI AUN ESTA DIAGNOSTICANDO
    R    #DISPARO                // QUITA EL DISPARO
    SPB  LOOP                    // CONTINUA EN EL BUCLE
```

A partir de ese instante desde cualquier parte del programa de PLC se pueden chequear los bytes del MB17 al MB20, y según el bit que esté activado indicar que esclavo está fallando.

13.1.9 Operaciones extendidas en la CP 342-2.

Como hemos visto, existen dos tipos de modo de funcionamiento con la CP 342-2: operaciones básicas, para trabajar únicamente a nivel de lectura de entradas y escritura de salidas, y realizar diagnosis de los esclavos, y operaciones extendidas, mediante las cuales disponemos de un control total sobre los telegramas de envío del maestro a la red.

El inconveniente de trabajar con las operaciones extendidas es la necesidad de utilizar la FC ASI_3422, contenida en los discos del manual de la CP. La FC tiene como finalidad mandar un determinado servicio o telegrama a la red cuando activemos dicha tarea en la subrutina. Solo un trabajo, o telegrama, como deseemos denominarlo, es enviado a la red en un determinado ciclo de CPU, por lo que no se deben de realizar varias llamadas a la FC en el programa principal del PLC.



Veamos los parámetros de la FC para operaciones extendidas:

Parámetro	Tipo	Significado
ACT	BOOL	Un flanco positivo en dicha entrada ejecuta la tarea si no se está ejecutando otra anteriormente.
STARTUP	BOOL	
LADDR	WORD	Dirección del módulo de arranque.
SEND	ANY	Buffer de envío. Zona de memoria donde se contiene una cantidad de informaión a enviar.
RECV	ANY	Buffer de recepción. Zona de momeia donde se reciben una serie de datos.

DONE	BOOL	Trabajo completado sin error.
ERROR	BOOL	Trabajo finalizado con error.
STATUS	DWORD	La primera palabra contiene el código de error en el caso de que lo hubiese en el trabajo solicitado. La segunda palaba es utilizada por la FC, y no debe ser modificado su valor.

Cuando realizamos un flaco positivo en ACT, y mientras se está realizando la acción, las salidas DONE y ERROR permanecen a FALSE, y en la palabra de STATUS obtenemos un valor 8181, indicándonos “trabajo en curso”. Una vez ha finalizado el trabajo, pueden darse dos circunstancias:

- Que concluya sin error, con lo cual se activa la salida DONE, y en la palabra de status obtenemos un valor 0.
- Que concluya con error, con lo cual se activa la salida ERROR, y obtenemos un código de error en la palabra de status, que puede ser:

Valor Status	Significado
8090	Dirección inválida en LADDR.
8ª0	Reconocimiento negativo leyendo del módulo.
80ª1	Reconocimiento negativo escribiendo en el módulo.
80B0	El módulo no reconoce la grabación de datos.
80B1	La longitud de los datos a escribir en el módulo es errónea.
80C0	Los datos no pueden ser leídos del módulo.
80C1	Los datos especificados están siendo grabados.
80C2	Demasiados trabajos pendientes.
80C3	Recursos de memoria ocupados.
80C4	Error de comunicaciones.
8182	Identificación para rearranque completo.
8184	Formato de tipo de datos recibidos no permitidos.
8381	Dirección de esclavo errónea.
8382	Esclavo no está activo (no en la lista de esclavos existentes).
8383	Error con el interface del PLC.
8384	Comando no permitido en status de CP.
8385	Ya existe un esclavo 0
83ª1	No se ha encontrado un esclavo con la dirección indicada para ser modificado.
83ª2	Esclavo 0 ya existe.

83ª3	Nueva dirección a la que se desea cambiar el esclavo ya existe n el interface del PLC.
83ª4	La dirección del esclavo no puede ser borrada.
83ª5	La dirección del esclavo no puede ser activada.
83ª6	La dirección del esclavo no puede ser guardada en la EEPROM:
83F8	Número de trabajo desconocido.
83F9	Error en EEPROM.
8F22	Error en la longitud del área de recepción (no es múltiplo de 8).
8F23	Error en la longitud del área de envío (no es múltiplo de 8).
8F24	Error en la longitud del área de recepción (no es una zona de memoria válida).
8F25	Error en la longitud del área de envío (no es una zona de memoria válida).
8F28	Error de alineamiento leyendo parametros.
8F29	Error de alineamiento escribiendo parametros.
8F30	El parámetro está colocado en un DB protegido.
8F31	El parámetro está colocado en un DB protegido.
8F32	El parámetro contiene un número de DB demasiado alto.
8F3A	El parámetro contiene un número de unaDB que no existe.
8F42	Un error de acceso ha ocurrido mientras el sistema esperaba a leer un parámetro del área de la periferia.
8F43	Un error de acceso ha ocurrido mientras el sistema esperaba escribir un parámetro del área de la periferia.
8F44	El acceso a lectura de un parámetro ha sido denegado.
8F45	El acceso a escritura de un parámetro ha sido denegado.
8F7F	Error interno.

13.1.10 Comandos para operaciones extendidas.

Dependiendo del código que contenga la zona de envío indicada en el parámetro SEND de la FC en su primer byte, realizaremos un trabajo u otro. A continuación veremos los posibles códigos que podemos indicar en este primer byte del telegrama de envío, así como la longitud para cada caso y estructura del mismo. También veremos la estructura de datos que recibiremos en el parámetro RECV según sea el trabajo que estemos ejecutando. Únicamente explicaremos aquí aquellos que por su importancia poseen una mayor relevancia, emplazando al lector al manual de la tarjeta para conocer el resto de posibilidades a nivel extendido.

Escribir parámetros permanentemente (JOB 00)

Mediante este comando un parámetro de un esclavo determinado es grabado permanentemente en la EEPROM de la tarjeta. El parámetro no es transmitido al esclavo inmediatamente por la tarjeta, sino únicamente frente a una transición STOP-RUN de la misma.

La estructura del telegrama de envío es:

Byte	Valor	
0	0	0
1	Dirección del esclavo	
2	0	Parámetro

Leer parámetros de la EERPOM(JOB 01)

Con esta instrucción un parámetro determinado de un esclavo es leído desde la EEPROM de la tarjeta.

El telegrama de envío será el siguiente:

Byte	Valor	
0	0	1
1	Dirección del esclavo	

Y el de recepción:

Byte	Valor	
0	0	Parámetro

Escribir un parámetro (JOB 02)

Mediante este trabajo se envía un determinado parámetro a un esclavo a través del bus AS-i. El telegrama de envío es:

Byte	Valor	
0	0	2
1	Dirección del esclavo	
2	0	Parámetro

El módulo, una vez ha aceptado el nuevo parámetro devuelve dicho parámetro a modo de eco para que sepamos de su aceptación. La estructura de recepción será:

Byte	Valor	
0	0	Parámetro

Leer un parámetro (JOB 03)

Mediante este trabajo se lee un determinado parámetro de un esclavo a través del bus AS-i. El telegrama de envío es:

Byte	Valor	
0	0	3
1	Dirección del esclavo	

La estructura de recepción será:

Byte	Valor	
0	0	Parámetro

Almacenar la configuración actual (JOB 04).

Mediante este trabajo se almacena en la configuración de la tarjeta la parametrización actual de los esclavos. El telegrama de envío es:

Byte	Valor	
0	0	4

Cambiar dirección de un esclavo (JOB DH).

Esta función nos permite por programa direccionar los módulos sin necesidad de direccionadora AS-i de mano. El único requisito es que cada vez que se llame a este trabajo exista únicamente un esclavo con la dirección indicada, y que no exista previamente el esclavo al que deseamos direccionarlo.

Los requisitos para que se pueda redireccionar el módulo son:

- que exista un módulo con la dirección antigua.
- Si la dirección antigua del módulo que se desea redireccionar no es la 0, no puede existir a la vez un módulo con la dirección 0.
- No debe existir un esclavo con la nueva dirección previamente al cambio.

Byte	Valor

0	0	DH
1	Dirección del esclavo vieja	
2	Dirección del esclavo nueva	

Lectura de status de módulo (JOB FH).

Anteriormente hemos visto como diagnosticar un error en el módulo esclavo. Sin embargo, esta manera no nos aporta ninguna información acerca de cual es la causa del error. Mediante este trabajo podemos conocer dicha causa, por lo que es indicado realizar una llamada a la misma una vez conozcamos que se ha producido dicho error.

El telegrama de envío será el siguiente:

Byte	Valor	
0	0	FH
1	Dirección del esclavo	

El telegrama de recepción:

Byte	Valor			
0	S3	S2	S1	S0

El significado de cada uno de los bits es el siguiente:

- S0: Este bit se activa cuando el módulo reconoce que su dirección difiere respecto a la que tiene asignada en el maestro AS-i.
- S1: Error de paridad detectado.
- S2: Error en final de bit. Se ha producido un error en la lectura del bit de final de telegrama e recepción.
- S3: Error en la lectura de la memoria no volátil.

Lectura de configuración (JOB 1H0)

Mediante este telegrama se puede conocer el estado de los esclavos configurados, los existentes y el estado de funcionamiento actual de la tarjeta.

El telegrama de envío es el siguiente:

Byte	Valor	
0	1H	00

El telegrama de recepción es el siguiente:

Byte	Valor							
0	LAS 0..3				LAS 4..7			
1	LAS 8..11				LAS 12..15			
2	LAS 16..19				LAS 20..23			
3	LAS 24..27				LAS 28..31			
4	LDS 0..3				LDS 4..7			
5	LDS 8..11				LDS 12..15			
6	LDS 16..19				LDS 20..23			
7	LDS 24..27				LDS 28..31			
8	LPS 0..3				LPS 4..7			
9	LPS 8..11				LPS 12..15			
10	LPS 16..19				LPS 20..23			
11	LPS 24..27				LPS 28..31			
12	S7	S6	S5	S4	S3	S2	S1	S0
11	C7	C6	C5	C4	C3	C2	C1	C0

Se debe de tener en cuenta que LAS significa esclavos activos (que se encuentran presentes en la red), LDS son esclavos configurados, mientras que LPS son esclavos presentes en la red. El significado de los bits de status es el siguiente:

- S0:Offline ready. La tarjeta se encuentra en modo offline.
- S1:Error APF. El bus AS-I tiene la tensión de alimentación demasiado baja.
- S2: Modo normal. La tarjeta se encuentra en modo normal.
- S3: Modo protegido. La tarjeta se encuentra en modo protegido.
- S4: Autoaddress disponible. Algún esclavo se encuentra fuera de la configuración y es posible realizar un autodireccionamiento de módulos esclavos.
- S5: Autoaddress asignable. Es posible realizar un autodireccionamiento, ya que un módulo está fuera de la red y además no existe ningún esclavo que lo impida.
- S6: Esclavo 0. Esta marca indica que existe un esclavo con la dirección 0 en la red.
- S7: Esta variable está activada cuando la configuración de esclavos diseñada y la real coinciden (todos los esclavos presentes).
- C0: CP en modo offline.
- C1: EEPROM de CP correcta sin errores.
- C2: Autoaddress activo.
- C3 a C7: reservados.

13.1.11 Indicación de errores en la CP 342-2.

La indicación de los errores de la CP se realiza mediante los leds frontales. El significado de las posibles combinaciones es el siguiente:

Error	Causa	Remedio
APF luce	La fuente AS-i no está conectada	Comprobar cableado
	Demasiado consumo en la red AS-i	Consultar consumos de esclavos.
SF luce sin presionar ningún botón	La CP está en modo protegido y ha fallado un esclavo.	Eliminar el error de configuración
	Error interno de eeprom	Cambiar la CP
SF luce mientras se presiona el botón de selección de modo de operación.	Se quiere cambiar a modo de protección, pero un esclavo con la dirección está actualmente conectado a la red.	Quitar el esclavo con dirección 0 de la red.
CER luce permanentemente	La CP no está todavía configurada.	Configurar la CP con el pulsador CM
	Un esclavo ha fallado	Reemplazar el esclavo que falla.
	Se ha conectado un esclavo que no estaba configurado a la red.	Quitar el esclavo no configurado
	Se ha conectado un esclavo en el que no concuerdan sus datos ID con los preconfigurados.	Comprobar que tipo de esclavo se ha conectado a la red.
CER parpadea	Fallo en algún contacto.	Comprobar las conexiones eléctricas del esclavo en cuestión.
	Interferencias en el cable AS-i	Comprobar que el PLC está puesto a tierra, y que se ha conectado la patilla Shield de la fuente de alimentación AS-i a tierra.
La CP no cambia de modo configuración a protegido o al revés.	El PLC está en RUN	Poner el PLC en STOP
	No se ha presionado el selector el tiempo suficiente.	Presionar durante al menos medio segundo.
	Existe un esclavo 0	Quitar el esclavo 0

No se puede realizar el direccionamiento automático aunque se encuentra activo AUP	Los datos de configuración del esclavo que se ha sustituido no concuerdan con el anterior.	Comprobar el tipo de módulo sustituído.
	Módulo defectuoso.	Cambiar por otro.

14

Contaje y posicionamiento

El contaje y posicionamiento en S7 se puede realizar de tres formas distintas:

- por programa de PLC, con lo cual tenemos que tener en cuenta no solo la frecuencia de conmutación de la entrada digital de contaje, sino también el retardo del tratamiento en el PLC de las líneas de código que nos realizan el contaje o posicionamiento.
- Mediante funciones integradas IFM, que nos realizan esta funcionalidad por hardware, o
- Mediante bloques especiales de hardware, que van a realizar esta acción al igual que las funciones IFM, pero con mayores posibilidades en el control del propio contaje y posicionamiento, a la vez que con mayor celeridad en la ejecución.

Veremos exclusívamente las dos últimas posibilidades, al ser las más seguras a la hora de tratar el contaje y la posición.

14.1 Funciones integradas IFM.

14.1.1 ¿Qué son las funciones IFM?.

Las CPU´s 312 IFM y 314 IFM además de ser las únicas que poseen periferia integrada en la propia CPU poseen una funcionalidad adicional, llamada IFM.

En la CPU 312 IFM esta funcionalidad le permite:

- realizar función de contador adelante/atrás.
- Realizar función de frecuenciómetro.
- Entrada de alarma.

En la CPU 314 IFM esta función le permite:

- Realizar función de frecuenciómetro
- Realizar función de contador, 1 adelante/atrás
- Realizar función decontadores A/B, 2 adelante/atrás.

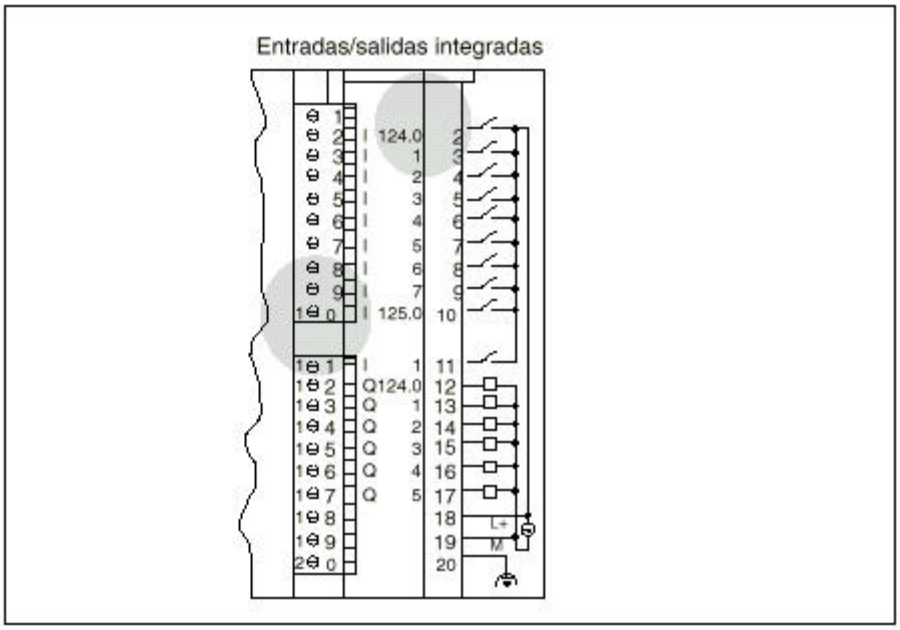
- Realizar función de posicionamiento en lazo abierto.
- Entrada de alarma.

14.1.2 ¿Dónde están las funciones integradas?.

La CPU 312 IFM posee 4 entradas especiales, localizadas en el conector de las entradas/salidas analógicas (primer conector), en las cuales se puede implementar la funcionalidad IFM, y 1salidas digitales que si se aplica esta funcionalidad serán gastadas por la tarjeta IFM para realizar las acciones pertinentes.

En la figura inferior se aprecia la posición de las mismas en el frontal de la CPU, siendo su direccionamiento en memoria de PLC:

- Para las entradas digitales de la E124.6 a E125.1.
- Para las salidas digitales la A124.0.

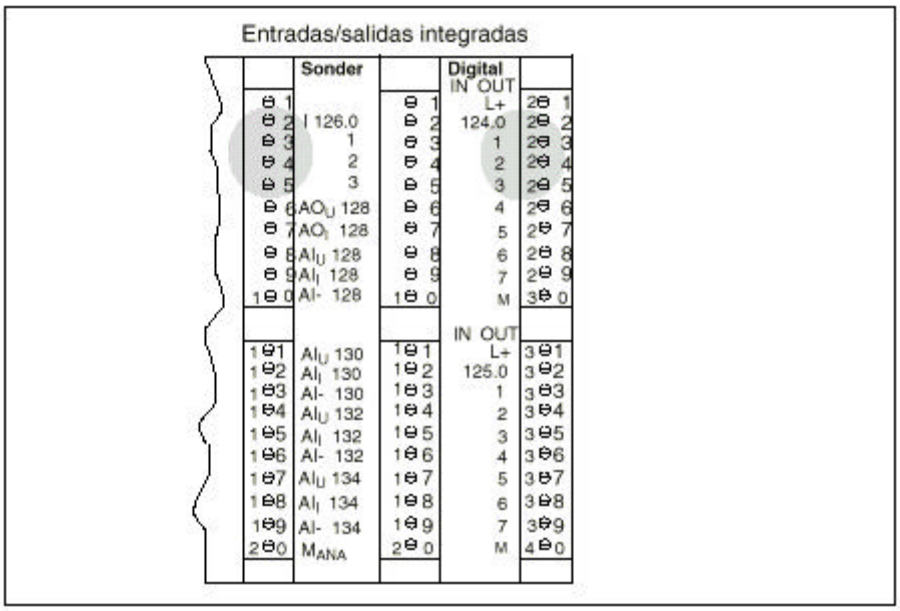


La CPU 314 IFM posee 4 entradas especiales, localizadas en el conector de las entradas/salidas analógicas (primer conector), en las cuales se puede implementar la funcionalidad IFM, y 4 salidas digitales que si se aplica esta funcionalidad serán gastadas por la tarjeta IFM para realizar las acciones pertinentes.

En la figura inferior se aprecia la posición de las mismas en el frontal de la CPU, siendo su direccionamiento en memoria de PLC:

- Para las entradas digitales de la E126.0 a la E126.3.

- Para las salidas digitales de la A124.0 a la A124.3



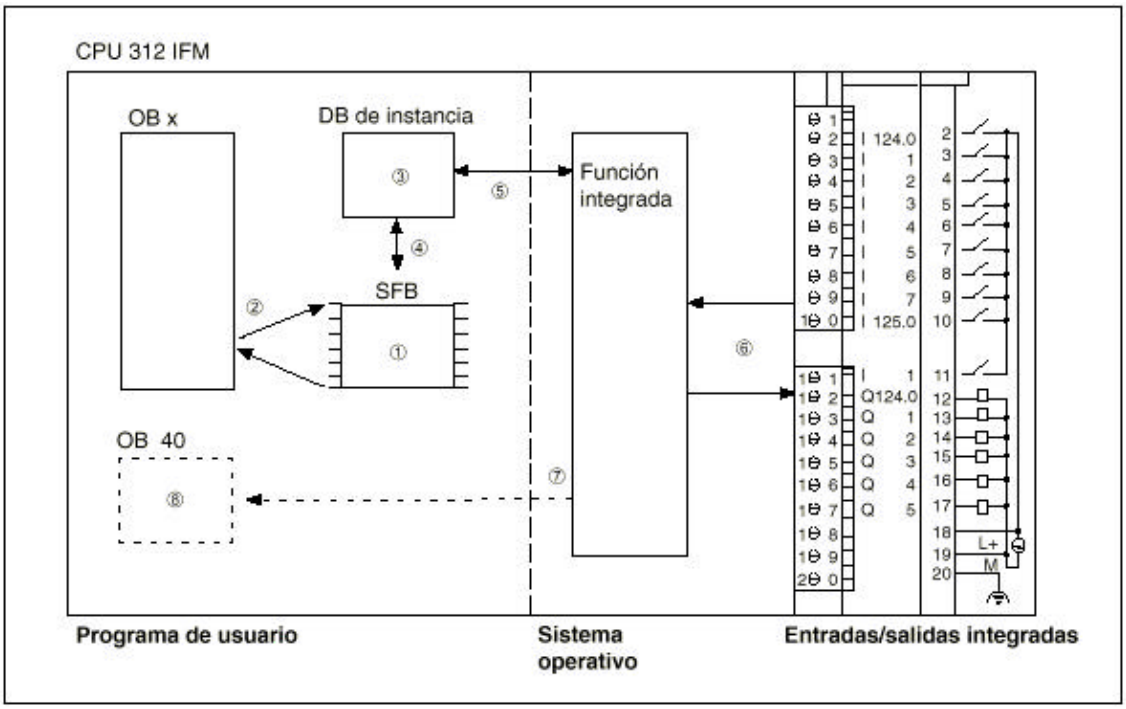
14.1.3 Funcionamiento interno de la IFM.

Desde cualquier bloque que se ejecute llamamos a una SFB del sistema contenida dentro de la EPROM de las CPU´s IFM, y que lleva asociada una DB de instancia, en la cual se encuentra la estructura de datos que se utilizará para el intercambio de información entre la SFB y la función integrada.

La función integrada leer los datos de la DB de instancia de la SFB, oera con ellos, accede directamente a la periferia del PLC (debido a esto los tiempos de reacción son bastante reducidos), y vuelve a dejar los resultados en la DB de instancia.

A continuación, la SFB toma estos valores de la función integrada y los deposita en el DB de instancia para que nuestro programa de usuario pueda consultarlos.

Si se produce un evento que dispare una alarma de proceso, la función integrada, sin que sea llamada por la SFB del programa de PLC, ejecutará la llamada a la OB40, donde se deberá de haber tratado dicho evento por el usuario.


CPU 312 IFM
OB x
DB de instancia
SFB
Función integrada
OB 40
Programa de usuario
Sistema operativo
Entradas/salidas integradas
I 124.0
I 125.0
Q124.0
L+
M

14.1.4 Función integrada alarma de proceso.

La primera de las funciones integradas que vamos a estudiar es función de alarma. Cualquiera de las entradas especiales IFM de la CPU, ya sea por flanco positivo o negativo, según lo parametricemos, nos generará una interrupción en el programa de PLC y un salto automático a la OB40, en la cual deberemos de realizar el tratamiento adecuado.

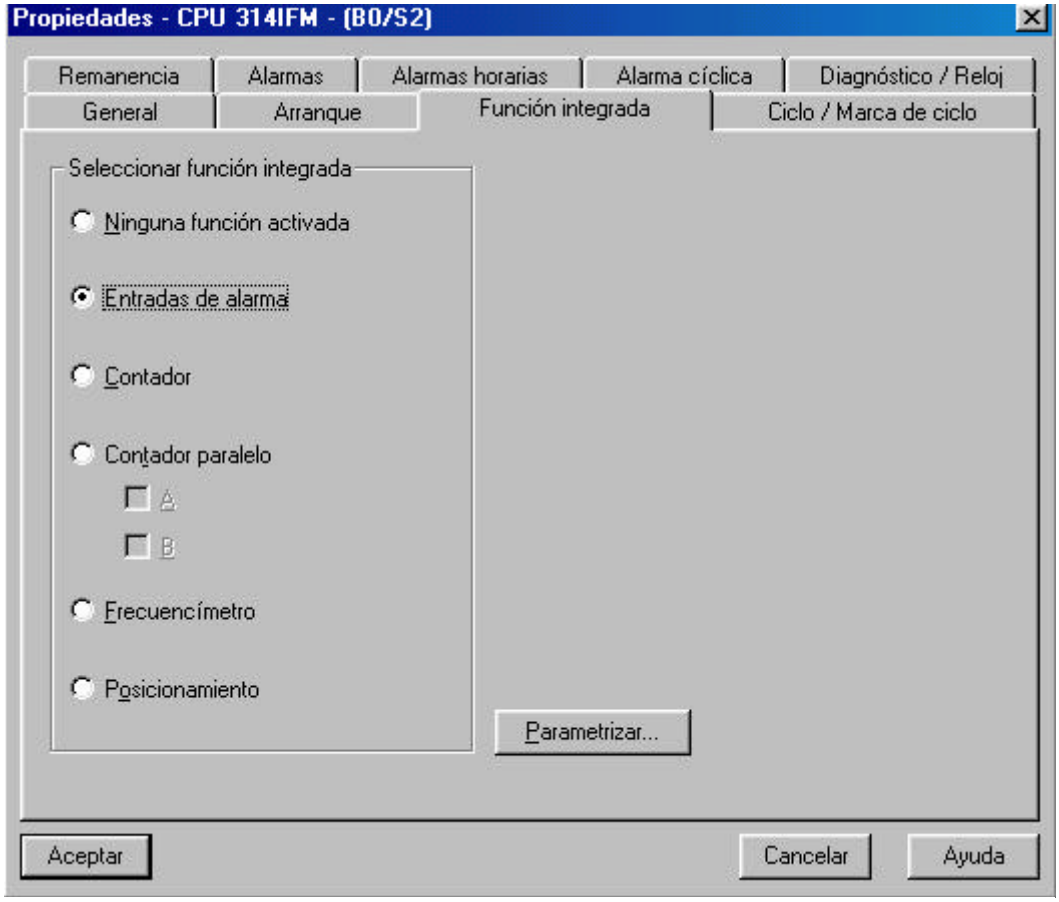
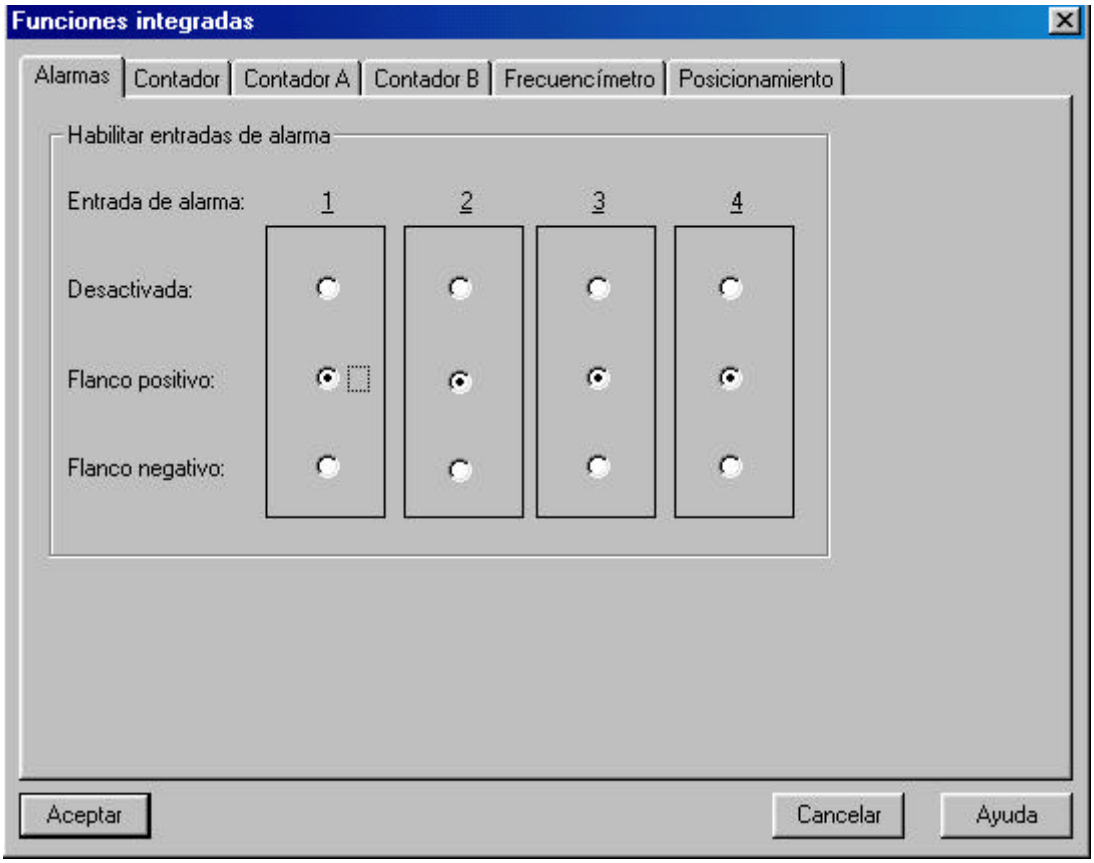


Figura 86. Activación de las funciones integradas desde Hardware de Step 7 en un equipo con funcionalidad IFM.

La parametrización de todas las funciones integradas en los equipos IFM, tanto la que estamos viendo ahora como las siguientes, se realiza desde Hardware de Step 7. Las propiedades de Hardware para esta función integrada de alarma nos permiten activarla por flanco positivo de la entrada de alarma o por flanco negativo.



Supongamos que deseamos reaccionar inmediatamente ante una entrada, actuando sobre la A 124.7. En la OB1 resetearemos la salida con otra entrada, como muestra el código:

```
U   E   124.0
R   A   124.7
```

Creamos la OB40 (en el disco duro, no en PLC, para que nos genere la tabla de declaración con simbólico), y programamos el siguiente código, transfiriéndoselo al PLC:

```
SET
S   A   124.7
```

Con esto, cuando activemos cualquier entrada de alarma desde la E126.0 a la E126.3 se activará la salida A124.7, siendo necesario resetearla con la E124.0.

Bien, pero ¿cómo podemos discriminar entre la entrada a la OB40 generada por una determinada entrada digital u otra?. Para ello utilizaremos la doble palabra #OB40_POINT_ADDR de la tabla de declaración de la OB40. En esta doble palabra se

contendrá un valor que nos indica cual es la entrada que genera la alarma de proceso. Los valores que tomará la variable irán según sea la entrada digital generadora desde 10001 a 10008.

14.1.5 Función integrada contador.

Para realizar la función integrada contador se utilizan las siguientes entradas y salidas:

CPU 312 IFM	CPU 314 IFM	Significado
E 124.6	E 126.0	Entrada contar adelante
E 124.7	E 126.1	Entrada contar atrás
E 125.0	E 126.2	Entrada sentido
E 125.1	E 126.3	Entrada start/stop
A 124.0	A 124.0	Salida digital A
E 124.1	A 124.1	Salida digital B

El esquema de bloques de la función contador es el siguiente:

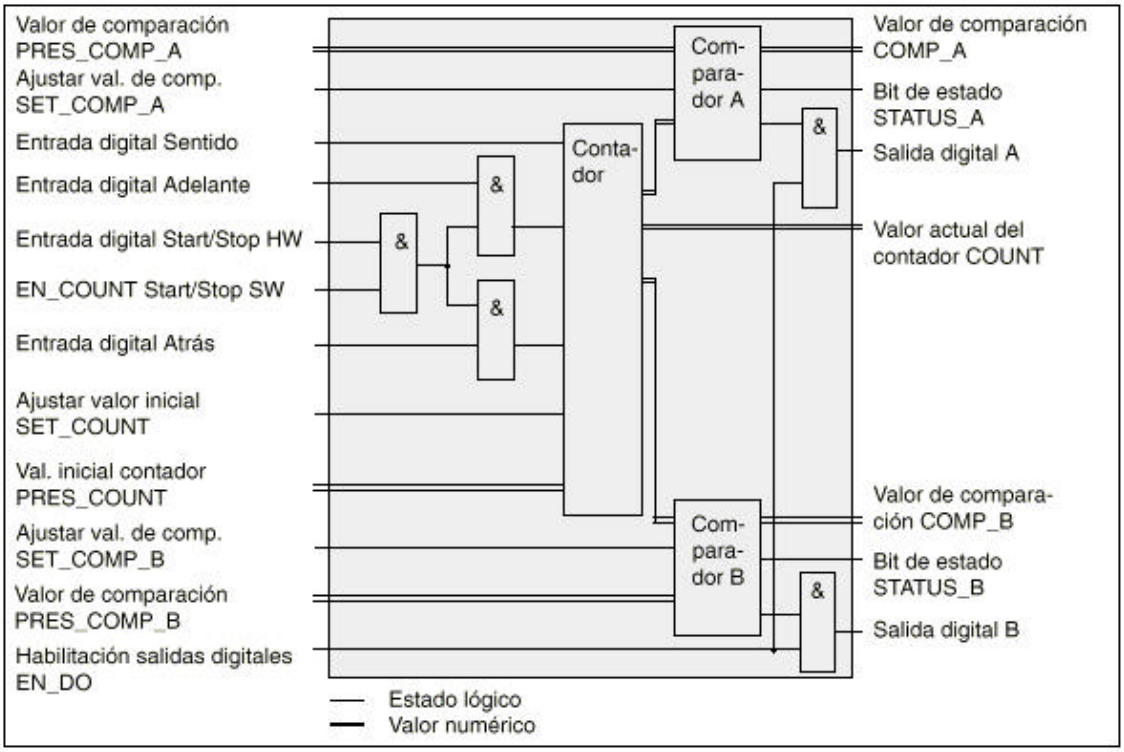


Figura 88. Esquema de bloques de los contadores rápidos de un equipo 314 IFM.

Mediante la función integrada contador se puede tener un contador adelante/atrás con una frecuencia máxima de contaje de 10 KHz y dos valores de preselección posibles.

Para utilizar este modo de contaje primeramente deberemos de definirlo en hardware de la CPU dentro de Step 7. Seleccionaremos modo contador y parametrizaremos el mismo.

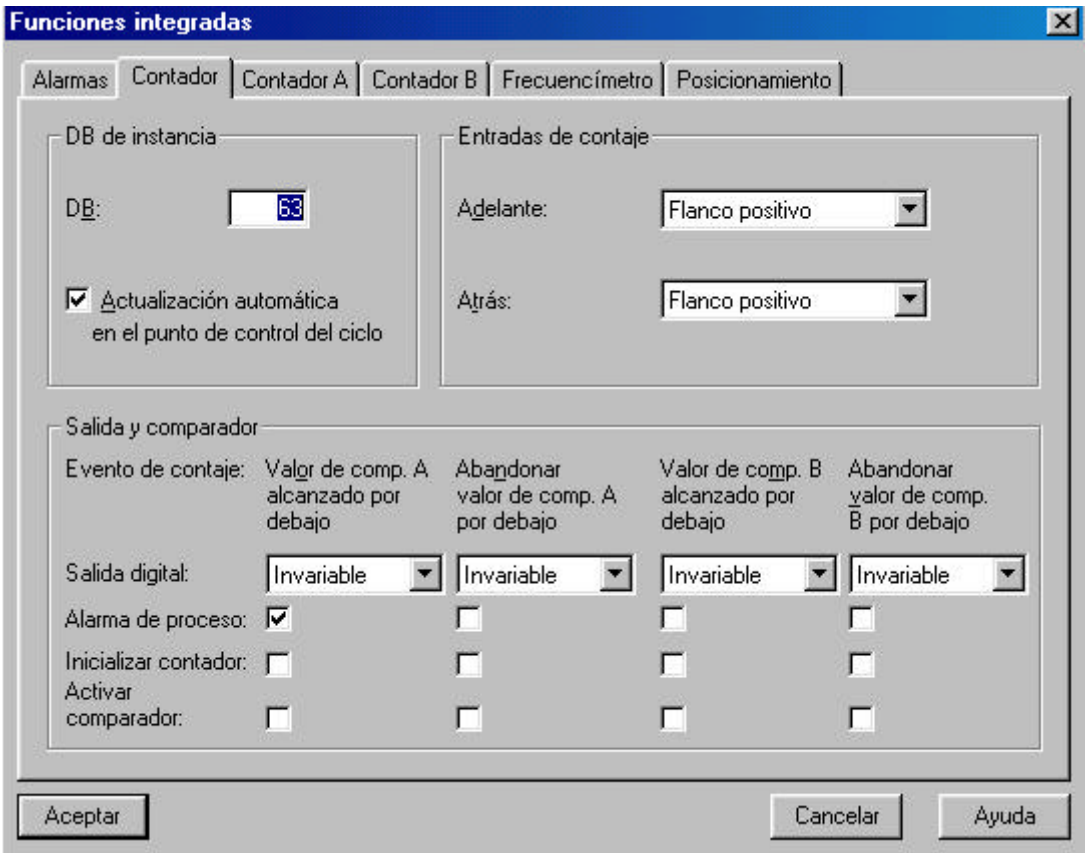


Figura 89. Parametrización de un contador IFM desde Hardware de Step 7.

Veamos las posibilidades que nos ofrece esta máscara a la hora de parametrizar:

- **DB de instancia:** este número de Db de instancia será la fuente de la que la tarjeta IFM tome y deje los datos de contaje. Por lo tanto, posteriormente a la SFB39 deberemos de asignarle el mismo número de DB que pongamos aquí.
- **Actualizar automáticamente en el punto de ciclo:** indica que se debe transferir el valor actual del contador cada vez que finalice el ciclo de programa desde la tarjeta a la DB de instancia.
- **Entradas de contaje:** nos permite definir que tipo de flanco se considera válido para contar tanto hacia delante como hacia atrás (flanco positivo o negativo).
- **Salida y comparador:** en este apartado podemos definir como se debe de comportar la tarjeta IFM tanto cada vez que alcanza el valor de contaje, como cuando lo

abandona, y además esta parametrización puede ser independiente para cada uno de los dos canales de preselección. Las posibilidades que se nos ofrecen son:

- Salida digital: cada comparador tiene asignada una salida digital. Aquí podemos definir si cuando llega a un determinado valor tanto por arriba o por debajo debe activar, desactivar o no influir en la salida digital.
- Generar una alarma de proceso: cuando se da la condición (según sea alcanzar o abandonar el valor de preselección) la tarjeta genera un evento que interrumpe el programa y entra a la OB40.
- Inicializar contador: cada vez que se produce la condición el contador se inicializa a cero.
- Activar comparador:

Una vez ha sido parametrizado nuestro funcionamiento de contaje, deberemos de realizar una llamada a la SFB 29 (HS_COUNT), gracias a la cual podremos tanto generar la DB de intercambio de datos entre programa y tarjeta IFM, como inicializar el contador, setear valores de preselección, y visualizar su estado actual.

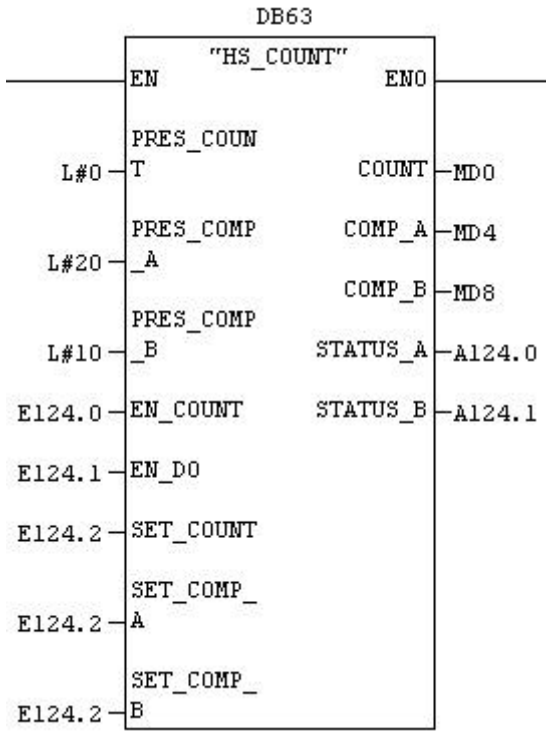


Figura 90. Llamada a la SFB 29 para el contaje rápido.

Recordemos que la Db de instancia que asociemos debe de coincidir con la seleccionada en el apartado Hardware de Step 7. Comenzaremos estudiando cada uno de los parámetros de esta SFB del sistema:

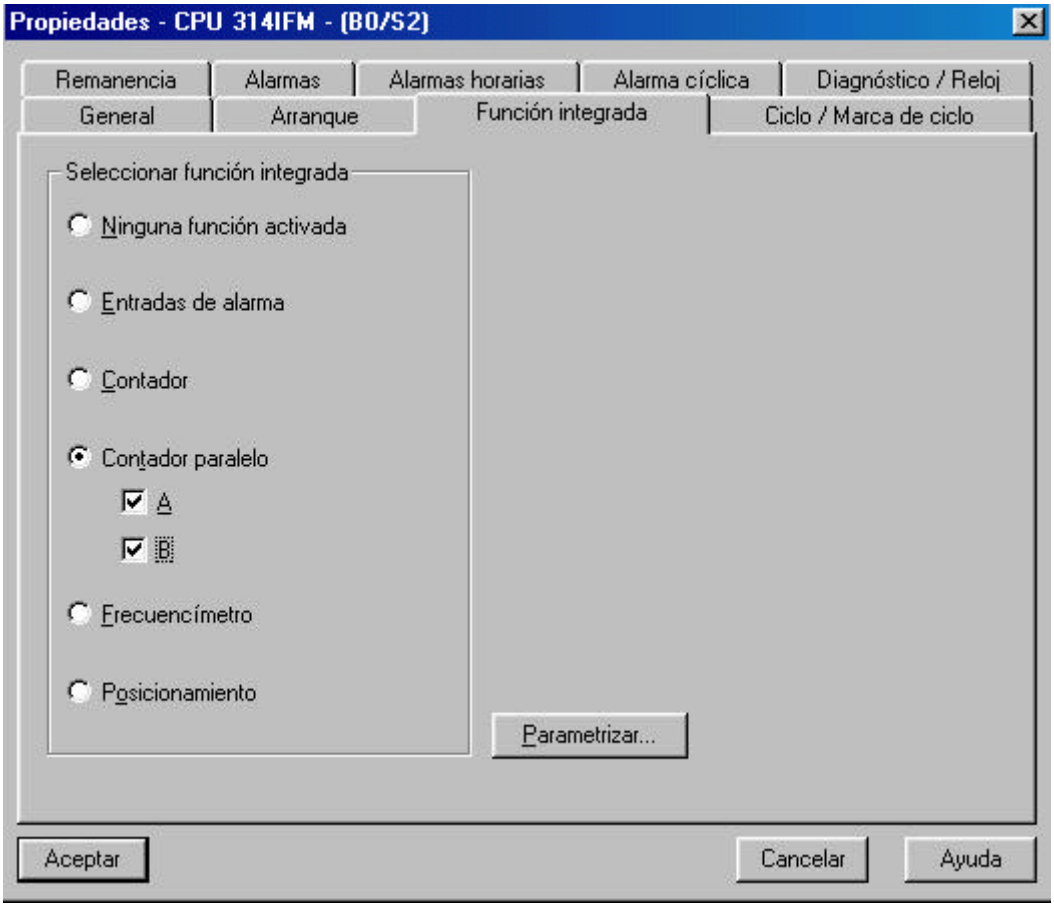
PARAMETRO	TIPO	FORMATO	SIGNIFICADO

PRES_COUNT	ENTRADA	DINT	Valor de preselección del contador. Valor al que se inicializa el contador.
PRES_COMP_A	ENTRADA	DINT	Valor de preselección comparador A.
PRES_COMP_B	ENTRADA	DINT	Valor de preselección comparador B.
EN_COUNT	ENTRADA	BOOL	Habilitación del contador. Mientras no se encuentre activada esta entrada el contador no cuenta en ningún sentido.
EN_DO	ENTRADA	BOOL	Habilitación de las salida digitales.
SET_COUNT	ENTRADA	BOOL	Setear el contador. Cuando se produzca un flanco positivo en esta entrada el contador se setea al valor que tenga en el parámetro PRES_COUNT.
SET_COMP_A	ENTRADA	BOOL	Setear el contador. Cuando se produzca un flanco positivo en esta entrada el comparador A se setea al valor que tenga en el parámetro PRES_COMP_A.
SET_COMP_B	ENTRADA	BOOL	Setear el contador. Cuando se produzca un flanco positivo en esta entrada el comparador A se setea al valor que tenga en el parámetro PRES_COMP_B.
STATUS_A	SALIDA	BOOL	Estado del comparador A. Se activa cuando el valor del contador es igual o superior al valor del comparador A.
STATUS_B	SALIDA	BOOL	Estado del comparador B. Se activa cuando el valor del contador es igual o superior al valor del comparador B.

14.1.6 Función integrada contador A/B.

La función integrada contador A/B nos permite disponer de dos contadores adelante/atrás.

La parametrización como siempre se realiza desde el Hardware de Step 7.

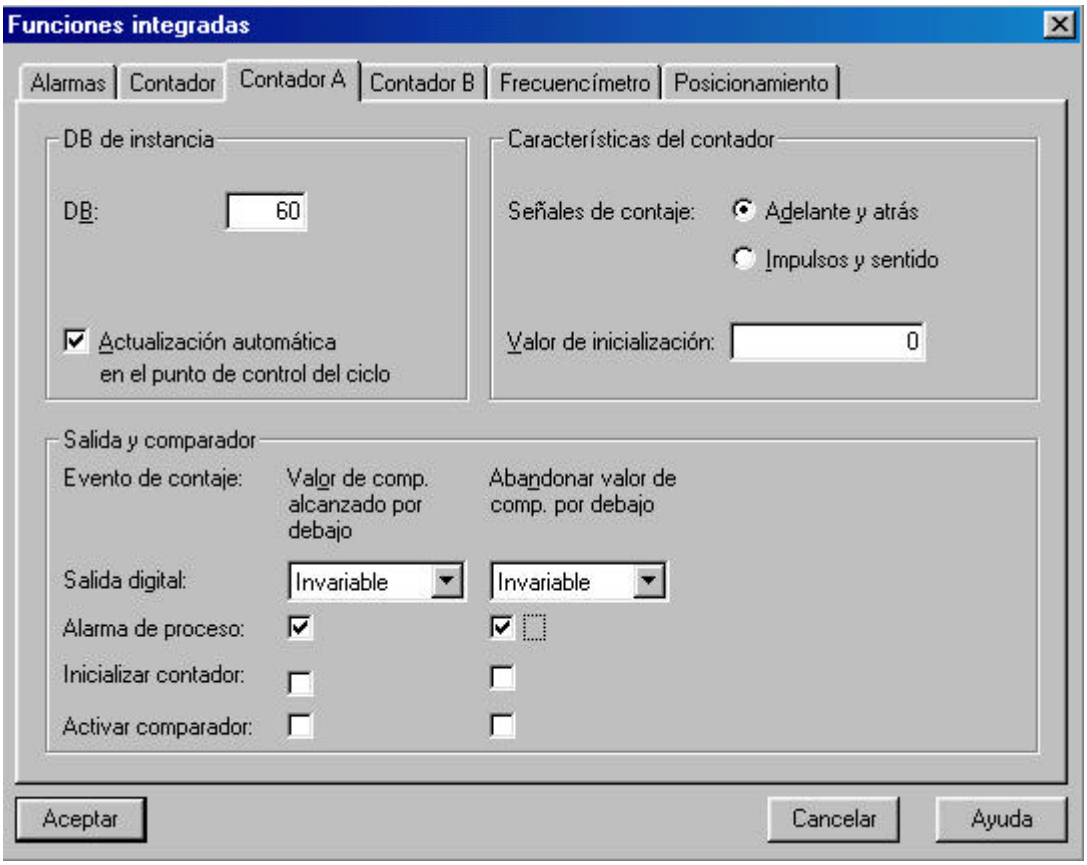


Si seleccionamos exclusívamente el contador A o el B nos encontramos en el caso anterior de función contador rápido, salvo que con la limitación de disponer exclusívamente de un valor de preselección. Deberemos de parametrizar cada uno de los canales A y B de manera independiente como muestra la figura siguiente.

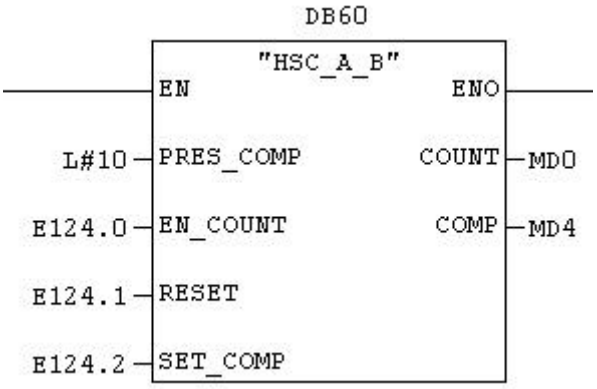
Cada uno de los contadores (A y B) dispondrá sus datos en una DB independiente, que por defecto son la 60 para el canal A y la 61 para el B. Los parámetros de los cuales podemos hacer uso son:

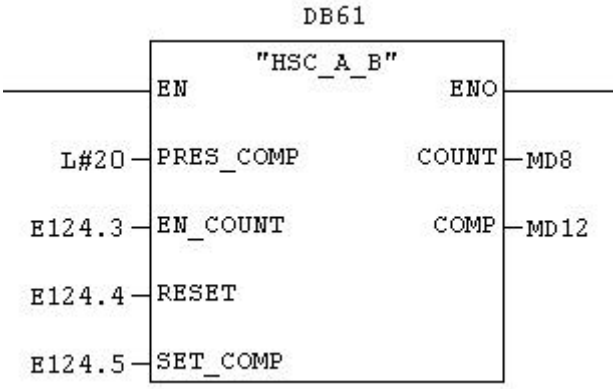
- DB de instancia: db de datos para dicho canal. Deberá coincidir con el db asociado a la SFB38.

- Actualización automática en el punto de control del ciclo: actualizar el dato actual de contaje cada ciclo de programa de PLC.
- Características del contador: podemos decidir si deseamos que las dos entradas asignadas al contador A cuenten adelante y atrás, o una cuente y la otra indique el sentido de contaje (contar o descontar).
- Valor de inicialización: Valor al que se desea inicializar el contador si lo vamos a resetear automáticamente en la OB40.
- Salida digital: podemos activar o desactivar automáticamente una salida digital al alcanzar o abandonar el valor de preselección por el canal actual, ya sea el A o el B.
- Alarma de proceso: Cuando alcance o abandone el valor de preselección el contador, que nos genere un evento de alarma de proceso que entre automáticamente a la OB40.
- Inicializar el contador: Al entrar a la OB40 que se inicialice el contador del canal actual al valor que indiquemos en valor de inicialización.
- Activar comparador:



Una vez tengamos parametrizado el hardware deberemos llamar desde nuestro programa a la SFB38 (HSC_A_B), que se va a encargar del manejo de los contadores A/B. Si vamos a trabajar con dos contadores, deberemos de llamar dos veces a la SFB38, una por cada canal, asignándole cada vez una DB de instancia.





Los parámetros de la SFB38 son:

PARAMETRO	TIPO	FORMATO	SIGNIFICADO
PRES_COMP	ENTRADA	DINT	Valor de preselección canal A o B
EN_COUNT	ENTRADA	BOOL	Habilitar el contador A o B. Si no activamos esta entrada el contador no cuenta en ningun sentido.
RESET	ENTRADA	BOOL	Resetear el contador A o B.
SET_COMP	ENTRADA	BOOL	Setear el valor de preselección del canal A o B al valor PRES_COMP
COUNT	SALIDA	DINT	Valor actual del contador A o B

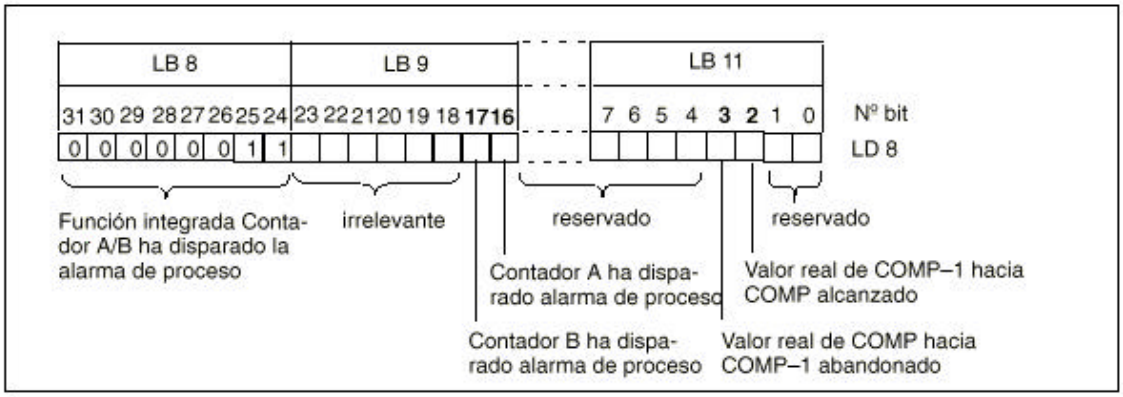
COMP	SALIDA	DINT	Valor del comparador del canal A o B.

Como siempre tenemos la necesidad de tratar cual de los eventos es el que nos está generando la OB40, siempre que trabajemos con varios canales. Para ello, podemos implementar las siguientes líneas de código en la OB40 cuando trabajemos con contadores A/B:

```
    L    #OB40_POINT_ADDR
    L    L#50397188
    ==D
    SPB  m001              // valor comparación A alcanzado

    L    #OB40_POINT_ADDR
    L    L#50462724
    ==D
    SPB  m002              // valor comparación B alcanzado

    L    #OB40_POINT_ADDR
    L    L#50397192
    ==D
    SPB  m003              // valor comparación A abandonado

    L    #OB40_POINT_ADDR
    L    L#50462728
    ==D
    SPB  m004              // valor comparación B abandonado

    BEA

m001: NOP  0
    SET
    S    A   125.0
    BEA

m002: NOP  0
    SET
    S    A   125.1
    BEA

m003: NOP  0
    SET
    S    A   125.2
    BEA

m004: NOP  0
```

```
SET
S   A   125.3
BEA
```

El tratamiento de la alarma de proceso viene determinado por el significado de cada uno de los bits de la palabra OB40_POINT_ADDR. De nuevo en la palabra OB40_POINT_ADDR tendremos un 7C indicándonos que el evento ha sido generado por la propia CPU. El significado de los bits de la OB40_POINT_ADDR es el siguiente:



El cableado de las entradas digitales según sea el tratamiento que hayamos parametrizado para el canal será el siguiente:

PARAMETRO	ADELANTE/ATRÁS	CONTAR/SENTIDO
E 126.0	Contar adelante canal A	Contar canal A
E 126.1	Contar atrás canal A	Sentido contaje canal A
E 126.2	Contar adelante canal B	Contar canal B
E 126.3	Contar atrás canal B	Sentido contaje canal B

14.1.7 Función integrada posicionamiento.

Esta función nos permite posicionar en un eje mediante un encoder de 24 V asimétrico (captador A/B). Únicamente está disponible en la CPU 314 IFM, no existiendo en la 312 IFM.

15

Diagnosis de averías

La diagnosis frente a errores en la gama S7 es muy sencilla en comparación a los sistemas actuales en el sector de la automatización, permitiendo al programador una gran libertad de acción al poder interceptar los eventos de error que se producen en el sistema y darles un tratamiento individualizado.

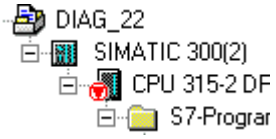
Vamos a describir todas las opciones de tratamiento de errores en S7, pasando posteriormente a realizar una breve, debido a lo extenso de la materia, exposición de las posibles causas.

15.1 Diagnóstico mediante Step 7.

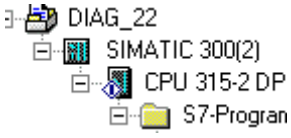
15.1.1 ¿Cómo sabemos que la instalación se ha parado?.

Evidentemente, viendo el piloto de stop en la CPU. Pero si nos es imposible observar la misma desde el software de S7 se nos indica el estado actual de la CPU, ya sea en RUN, STOP, o con algún fallo determinado.

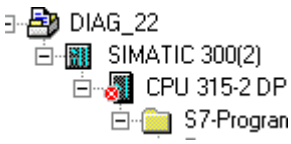
- **CPU en STOP**



- **CPU en RUN**



- **CPU con algún error**



Es importante destacar que las modificaciones en el estado de la CPU no se reflejan de manera automática en el software, siendo necesario refrescarlas presionando la tecla F5. Por

lo tanto, si se intuye que se puede haber parado la instalación, o que se está produciendo un error hardware, presionar el botón F5 en el administrador de Step 7 es el primer paso a realizar.

15.1.2 Existe un error, y ahora ¿qué?.

El siguiente paso lógicamente es descubrir la causa del error que se está generando en nuestro equipo. Para ello, en el administrador de Simatic seleccionaremos nuestra CPU, y en el menú contextual que emerge presionando el botón derecho del ratón, seleccionaremos la opción sistema destino->información del módulo.

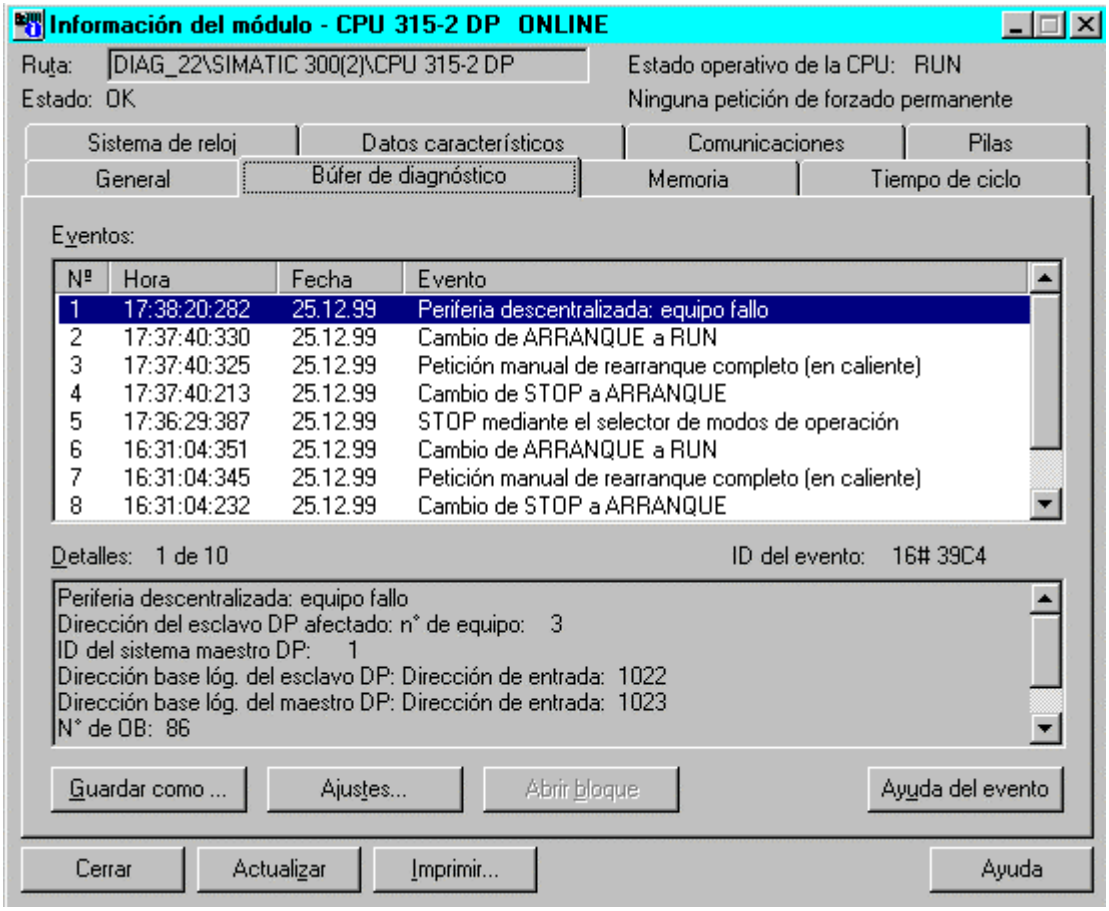
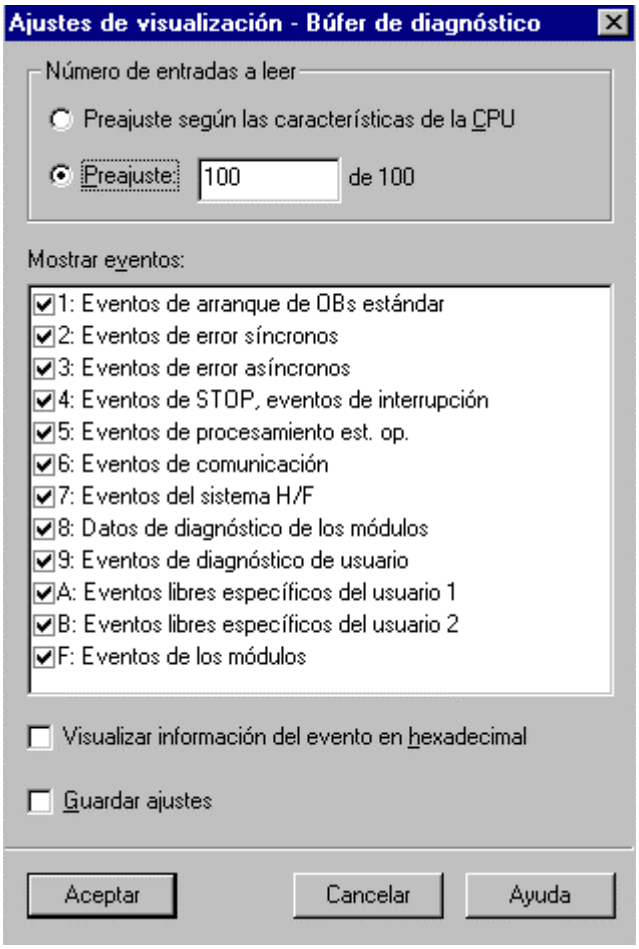


Figura 91. Buffer de diagnóstico de una CPU S7.

En la solapa buffer de diagnóstico aparecen los últimos evento que se han producido en dicha CPU, registrados de manera cronológica comenzando por el más reciente. El tambor del buffer de diagnóstico suele venir predeterminado para los 10 últimos eventos. Como un

evento no necesariamente es un error (de hecho, todo error genera varios eventos: el propio error, si pasa la CPU a STOP, la solución del error y el rearranque de la CPU), dicho buffer puede ser algo corto si deseamos observar un error acontecido hace algún tiempo. Para poder modificar el tamaño del buffer de errores y ampliarlo, seleccionamos el botón ajustes de esta ventana.



El tamaño máximo de eventos a almacenar en la CPU es de 100. Es importante destacar que este buffer no se borra pese a realizar un borrado total en la CPU.

En la ventana *buffer de diagnóstico* disponemos también del botón Guardar como..., que nos permite realizar una exportación a un fichero de texto del buffer de diagnóstico, p. Ej. Para realizar un informe de los errores o paradas que se han producido en la CPU durante un tiempo determinado.

Una exportación a este fichero de los eventos tiene el siguiente aspecto:

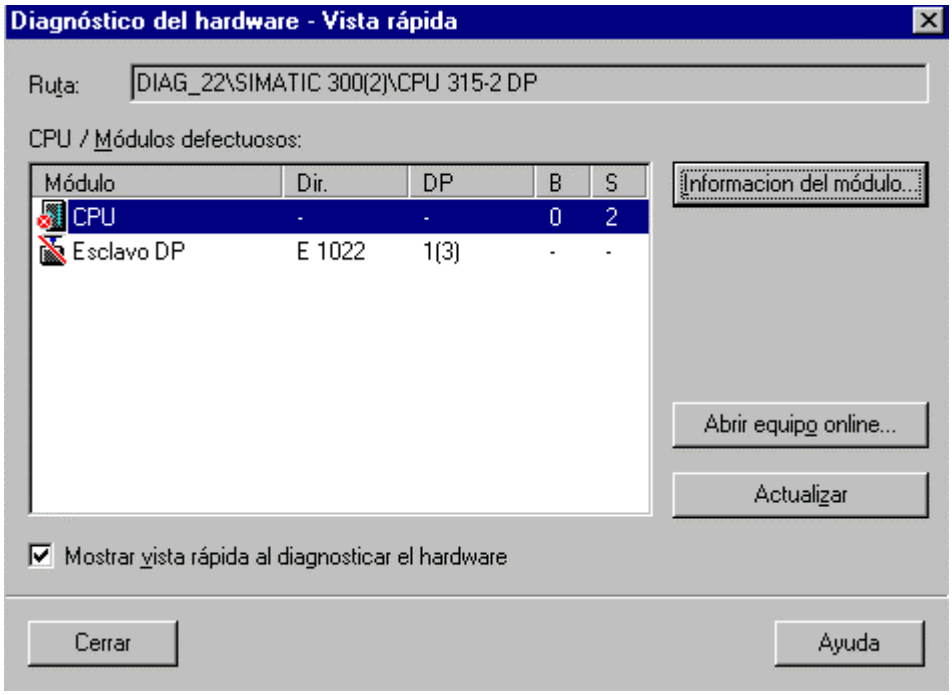
```
Evento 1 de 100:  ID de evento 16# 39C4
Periferia descentralizada: equipo fallo
Dirección del esclavo DP afectado: n° de equipo:    3
```

```
ID del sistema maestro DP:      1
Dirección base lóg. del esclavo DP: Dirección de entrada:  1022
Dirección base lóg. del maestro DP: Dirección de entrada:  1023
N° de OB:  86
Prioridad: 26
Error externo, Evento entrante
17:55:03:136   25.12.99
```

A su vez, desde la ventana de diagnóstico podemos solicitar ayuda al respecto del evento que se seleccione. Esta ayuda en la mayoría de las ocasiones es muy limitada, pero nos puede servir como orientación para la búsqueda de la causa del error que genera el evento.

15.1.3 Ya se cual es el error ¿dónde se genera?

En la información anterior se nos indicaba cual era el error que se estaba produciendo y el lugar. Pero existe una forma más gráfica de detectar el módulo que está fallando o generando el error.



Si nos posicionamos sobre nuestra CPU, y en el menú contextual que emerge presionamos el botón derecho del ratón, seleccionaremos la opción *sistema destino->Diagnosticar Hardware*, obteniendo una imagen gráfica del estado de los módulos que

contienen nuestra instalación. En la figura se puede apreciar como el esclavo 3 de DP es el causante del fallo detectado en la CPU.

Desde esta ventana se puede abrir la configuración hardware en online, al igual que solicitar la información de cada uno de los módulos que contienen error actualmente.

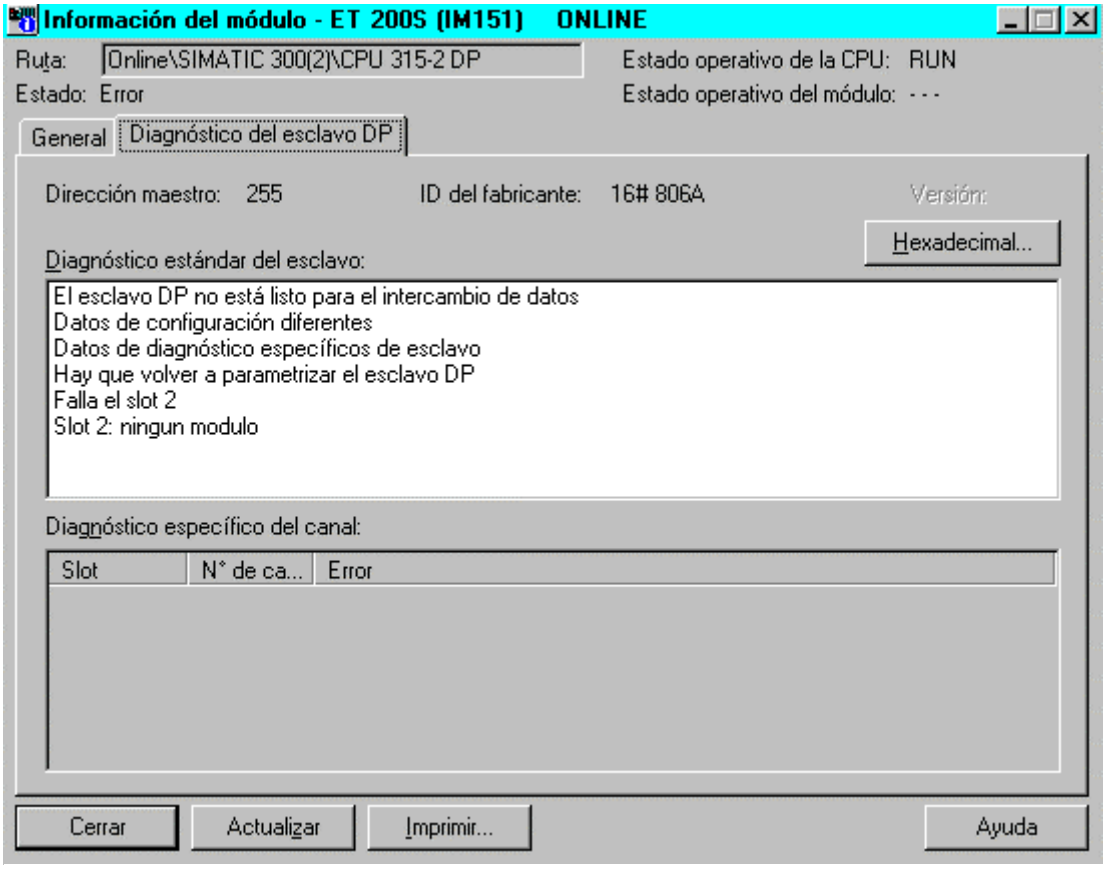


Figura 93. Información de un módulo de Step 7.

Se puede apreciar en la imagen como se nos indica hasta el módulo que está fallando dentro del esclavo DP.

15.2 Diagnosis para Profibús DP.

Existen dos maneras de realizar la diagnosis en Profibús DP:

- Mediante la programación de la SFC13 dentro de la OB82...
- Mediante la utilización de la FB99, FB que no es estándar (no se encuentra integrada dentro de las CPU's de S7), que ya nos realiza la gestión de los fallos de DP.

15.2.1 Diagnosis mediante la FB99.

El bloque de función FB99 nos facilita el tratamiento de errores en la adquisición de datos desde el maestro de Profibús DP hasta las estaciones que cuelguen del mismo. Mediante ella es posible:

- Conocer que (o cuales) estación(es) de DP está(n) fallando actualmente.
- Descubrir cual es el módulo, en el caso de que únicamente estuviese fallando una tarjeta, que se encuentra actualmente generando el error.
- Discernir entre las posibles causas de error que se estén produciendo en la estación o el módulo involucrados.

Esta funcionalidad únicamente se puede aplicar para las estaciones de Profibús DP, no reconociendo errores de módulos centralizados.

La llamada a la FB99 tiene el siguiente aspecto:

```
CALL  FB    99 , DB99
 DP_MASTERSYSTEM      :=1                    // SISTEMA MAESTRO DP
 EXTERNAL_DP_INTERFACE :=FALSE               // INTERFACE EXTERNA DP
 SINGLE_STEP          :=E8.0                 // SIGUIENTE FALLO DP
 RESET                :=E8.1                 // RESET FALLO DP
 ALL_DP_SLAVES_OK     :=M100.0               // TODOS LOS ESCLAVOS OK
 SUM_SLAVES_NOT_PRESENT:=MB102               // SUMA DE ESCLAVOS NO PRESENTES
 SUM_SLAVES_ERROR     :=MB103                // SUMA DE ESCLAVOS CON ERROR
 SLAVE_ADR            :=MB104                // DIRECCION DEL ESCLAVO
 SLAVE_NOT_PRESENT    :=M100.1               // ESCLAVO NO PRESENTE
 SLAVE_ERROR          :=M100.2               // ESCLAVO CON ERROR
 SLOT_DIAG_PRESENT    :=M100.3               // DIAGNOSIS PRESENTE
 SLOT_NO              :=MB105                // NUMERO DE SLOT
 FURTHER_SLOT_DIAG    :=M100.4               // DIAGNOSIS DE SLOT ACTUAL
 CHANNEL_DIAG_PRESENT  :=M100.5              // DIAGNOSIS DE CANAL PRESENTE
 CHANNEL_SLOT_NO      :=MB106                // NUMERO DE SLOT DE CANAL
 CHANNEL_NO           :=MB107                // NUMERO DE CANAL
```

```
CHANNEL_ERROR_TYPE    :=MB108                  // TIPO DE ERROR DE CANAL
FURTHER_CHANNEL_DIAG  :=M100.6                 // DIAGNOSIS DE CANAL ACTUAL
MODULE_STATE_PRESENT  :=M100.7                 // STATUS DMODULO PRESENTE
MODULE_NO             :=MB109                  // NUMERO DE MODULO
MODULE_STATE          :=MB110                  // ESTADO DEL MODULO
FURTHER_MODULE_STATE  :=M101.0   //
DEVICE_DIAG_PRESENT   :=M101.1
DATE_DAY              :=MB111    // DIA DEL ERROR
DATE_MONTH            :=MB112
CLOCK_HOUR            :=MB113
CLOCK_MINUTE          :=MB114
CLOCK_SECOND          :=MB115
DIAG_OVERFLOW         :=M101.2
BUSY                  :=M101.3
```

Recordemos que esta FB no se encuentra en ninguna librería de Step 7 (pese a que debería de pertenecer a la misma), por lo que es necesario descargársela de internet.

Veamos el significado de cada uno de los parámetros:

PARAMETRO	**TIPO**	**FORMATO**	**SIGNIFICADO**
DP_MASTERSYSTEM	ENTRADA	INTEGER	En este entero indicaremos el número de maestros DP de que disponemos en nuestro sistema (generalmente 1).
EXTERNAL_DP_INTERFACE	ENTRADA	BOOL	Este parámetro especifica si el maestro DP está integrado en la CPU o no. Los valores posibles son: FALSE: no integrado en la CPU. TRUE: integrado en la CPU. Para las CPU´s de tipo 2-DP será necesario seleccionar TRUE, mientras que para las que utilicen como maestro de DP una CP será necesario seleccionar FALSE.
SINGLE_STEP	ENTRADA	BOOL	Mediante la activación de esta entrada se continúa a la visualización del siguiente fallo de DP. Si actualmente se están produciendo dos fallos en DP, alternaremos con esta entrada entre ambos.
RESET	ENTRADA	BOOL	Es necesario realizar un reset de la Fb después de que se haya producido un desbordamiento en el buffer de diagnóstico interno de la FB, indicado por la salida DIAG_OVERFLOW.
ALL_DP_SLAVES_OK	SALIDA	BOOL	Esta salida indica que todos los equipos de la red DP están funcionando correctamente. La negación de esta salida es una primera indicación de alarma, para tratar en un sistema de visualización.
SUM_SLAVES_NOT_PRESENT	SALIDA	BYTE	Total de esclavos que fallan actualmente.
SUM_SLAVES_ERROR	SALIDA	BYTE	Número total de esclavos que poseen un error. La diferencia entre error de esclavo y fallo de esclavo reside en que: en fallo, el esclavo no comunica en absoluto, debido a que o el equipo no tiene tensión, o se ha roto el cable

			de comunicaciones, o que simplemente se ha estropeado el módulo DP. Sin embargo en error, el equipo DP sí que se encuentra comunicando, por lo que el cable es correcto, y la alimentación llega a la estación, sin embargo el equipo envía un mensaje de fallo, debido a que uno de sus módulos posee un problema.
SLAVE_NO	SALIDA	BYTE	Estación DP que actualmente está teniendo el fllo o error. Si existe más de una estación con fallo actualmente, activando un flanco positivo en la entrada SINGLE_STEP de la FB podremos ir cambiando de manera ascendente entre las estaciones con error.
SLAVE_NOT_PRESENT	SALIDA	BOOL	Cuando se encuentra en TRUE, indica que el fallo del esclavo actual es que no se encuentra presente (no responde al maestro).
SLAVE_ERROR	SALIDA	BOOL	Cuando se encuentra en TRUE, indica que el fallo del esclavo actual es que está fallando algun ode sus módulos.
SLOT_DIAG_PRESENT	SALIDA	BOOL	Cuando se encuentra a TRUE indica que existe información al respecto del módulo que ha fallado de la estación DP.
SLOT_NO	SALIDA	BYTE	Este parámetro indica el número de slot que está fallando dentro de la estación DP. El valor se muestra en hexadecimal.
FURTHER_SLOT_DIAG	SALIDA	BOOL	El módulo que está fallando posee una posibilidad de diagnóstico adicional a la descrita anteriormente.
CHANEL_DIAG_PRESENT	SALIDA	BOOL	Si un esclavo se encuentra fallando, se indica que el módulo posee diagnosis de canal.
CHANEL_SLOT_NO	SALIDA	BYTE	Número de slot donde se produce el error dentro de la estación DP.
CHANEL_NO	SALIDA	BYTE	Número de canal donde se produce el error dentro de la estación DP.
CHANEL_ERROR_TYPE	SALIDA	BYTE	Tipo de error dentro del canal del módulo que está generando el fallo en el esclavo.
FURTHER_CHANNEL_DIAG	SALIDA	BOOL	Un nuevo informe de diagnostico de canal esta disponible.
MODULE_STATE_PRESENT	SALIDA	BOOL	Si un esclavo esta fallando, se indica que el diagnosis de modulo esta disponible.
MODULE_NO	SALIDA	BYTE	Número de módulo donde se produce el error dentro de la estación DP.
MODULE_STATE	SALIDA	BYTE	Estado del módulo que esta fallando. Los diferentes estados son: 00 Módulo OK (disponible) 01 Módulo defectuoso 10 Módulo equivocado 11 Módulo no presente

FURTHER_MODULE_STATE	SALIDA	BOOL	Nos informa que un diagnosis adicional de módulo esta disponible.
DEVICE_DIAG_PRESENT	SALIDA	BOOL	Un informe de diagnostico de esclavo esta disponible (diagnostico tanto para esclavos específicos como para DP standards).
DATE_DAY	SALIDA	BYTE	Muestra el día en que ocurrió el fallo o defecto en el esclavo.
DATE_MONTH	SALIDA	BYTE	Muestra el mes en que ocurrió el fallo o defecto del esclavo (sacado de la fecha de la CPU).
CLOCK_HOUR	SALIDA	BYTE	Muestra la hora del día en que se produjo el fallo o error en el esclavo (sacado del reloj de la CPU).
CLOCK_MINUTE	SALIDA	BYTE	Muestra el minuto de la hora en que se produjo el fallo o defecto en el esclavo (sacado del reloj de la CPU).
CLOCK_SECOND	SALIDA	BYTE	Muestra el segundo del minuto en que ocurrió el fallo o defecto en el esclavo (sacado del reloj de la CPU).
DIAG_OVERFLOW	SALIDA	BOOL	Nos indica que el número de mensajes simultáneos de diagnostico recibidos es mayor que 16. Esto es significativo para llevar a cabo un RESET.
BUSY	SALIDA	BOOL	Este parámetro nos indica que la evaluación del sistema DP por la FB 99 esta en marcha. Espera a que la evaluación este completa para mostrar una nueva información. (BUSY 1 a 0)

Las llamadas a la FB se deben de realizar en:

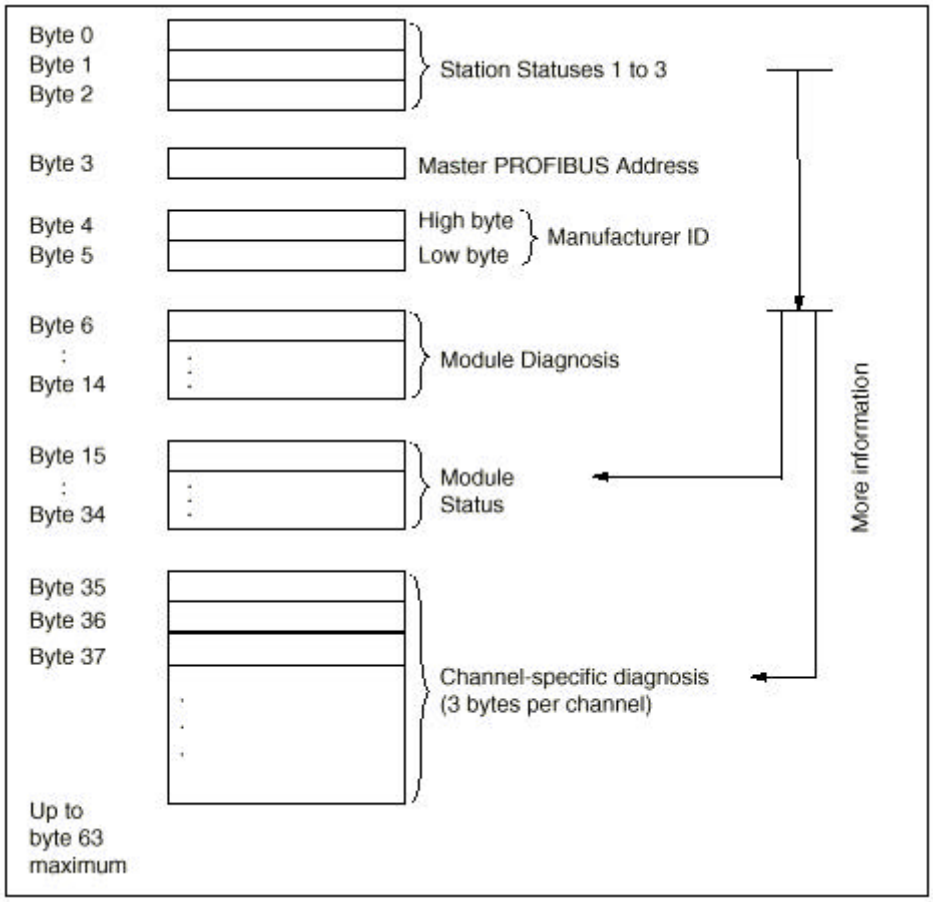
- La OB 1, para reconocimiento cíclico de la desaparición del error.
- La OB 82, para el reconocimiento de la alarma de diagnóstico que genera el módulo en cuestión.
- La OB 86, para el reconocimiento del fallo de una estación DP.

El número máximo de mensajes simultáneos de error que pueden ser tratados por la FB99 es de 16. Si fallan un número superior de estaciones durante un cierto periodo de tiempo (no es necesario que el fallo se produzca de manera simultánea en el tiempo) se produce un desbordamiento en el buffer de la FB, indicándosenos mediante un bit.

Además de la FB99, el sistema utiliza la SFC 51 y la SFC 13, para obtener los datos que posteriormente nos indicará la FB99 en sus salidas. Para que funcione correctamente, no es posible realizar llamadas a estas SFC's desde la OB1, ya que las está llamando la FB99, que está siendo llamada en la propia OB1.

15.2.2 Diagnosis mediante la SFC13

Estructura del diagnostico de esclavo:



Estado de la estación (Station Statuse) 1 a 3:

Estado 1:

BIT	SIGNIFICADO	CAUSA / REMEDIO
0	El esclavo DP no puede ser direccionado por el maestro DP.	• La dirección PROFIBUS elegida en el esclavo DP no es la correcta. • El conector de BUS no esta conectado. • El esclavo DP no tiene voltaje. • El repetidor RS 485 no esta seleccionado correctamente. • Se ha producido un reset en el esclavo DP.

1	El esclavo DP ya no esta preparado para intercambiar datos.	Espera un tiempo hasta que el esclavo DP empiece.
2	La configuración enviada por el maestro DP al esclavo DP no corresponde con la que tiene el esclavo.	El tipo de estación o la configuración del esclavo DP entrada en la configuración software no es correcta.
3	Diagnostico externo disponible. (Indicación del grupo de diagnostico)	Evaluación del diagnostico de módulo, el estado del módulo y/o el diagnostico especifico de canal. El bit 3 es reseteado tan pronto como los fallos son corregidos. El bit es reseteado cuando hay un nuevo mensaje de diagnostico en los bytes de diagnostico mencionado arriba.
4	La función que fue requerida es no soportada por el esclavo (por ejemplo, cambiar la dirección de PROFIBUS por medio del software).	Comprueba la configuración.
5	El maestro no puede interpretar la respuesta del esclavo DP.	Comprobar la colocación del BUS.
6	La configuración del esclavo DP no corresponde con la configuración software.	El tipo de estación entrado en la configuración software no es correcto.
7	Los parametros han sido asignados al esclavo DP por un maestro diferente (no el que actualmente tiene acceso al esclavo DP).	Este bit esta siempre a 1, por ejemplo, cuando tu accedes al esclavo DP por medio de la maleta de programación u otro maestro DP. La dirección PROFIBUS del maestro DP que asigna parametros al esclavo DP esta localizada en el byte de diagnostico “master PROFIBUS address”.

16

APENDICE A: Etiquetas

Las presentes hojas se pueden utilizar como plantillas para realizar las etiquetas frontales de los módulos de nuestras instalaciones.