2 Introducción a los Micro-PLCs S7-200

Como ya hemos indicado, vamos a diseñar una plataforma que nos permita gobernar las comunicaciones entre la red GSM y un PLC con capacidad de comunicación bajo el estándar RS-232 de forma que interfiramos lo menos posible con el citado PLC. Sin pérdida de generalidad, se ha montado el sistema completo empleando un autómata programable Micro-PLC S7-200 de Siemens. Las causas de esta elección son la disponibilidad de dicho equipo en las instalaciones del laboratorio, evitando así la compra de uno de estos equipos. Se dispone también de autómatas de la gama S7-300 del mismo fabricante, sin embargo, se ha optado por el S7-200 porque a pesar de tener un menor potencial, dispone de un modo de comunicación llamado *Freeport* no presente en la gama S7-300 que facilita enormemente la comunicación entre el PLC y nuestro equipo. Indiquemos, que el autómata elegido no dispone de interfaz de comunicación RS-232, utiliza el estándar RS-485. Sin embargo, el conector que utiliza para conectarse con un PC y poder ser programado, realiza la adaptación entre ambos estándares, nos referimos al "cable PC/PPI" que aparece en la Figura 2.1, de este modo, cualquier equipo que conectemos en sustitución del PC de la figura, podrá comunicarse con el PLC utilizando la norma RS-232.

La gama S7-200 comprende diversos sistemas de automatización pequeños (Micro-PLCs) que se pueden utilizar para numerosas tareas. Gracias a su diseño compacto, su capacidad de ampliación, su bajo costo y su amplio juego de operaciones, los Micro-PLCs S7-200 se adecuan para numerosas aplicaciones pequeñas de control. Además, los diversos tamaños y fuentes de alimentación de las CPUs ofrecen la flexibilidad necesaria para solucionar las tareas de automatización.

2.1 Funciones de los diversos Micro-PLCs S7-200

2.1.1 Equipos necesarios

La Figura 2.1 muestra la estructura básica de un Micro-PLC S7-200 que incluye una CPU S7-200, un PC, el software de programación STEP 7-Micro/WIN y un cable de comunicación. Si desea utilizar un PC, debe disponer de uno de los siguientes equipos adicionales:

- Un cable PC/PPI.
- Un procesador de comunicaciones (CP) y un cable de interfaz multipunto (MPI).
- Una tarjeta de interfaz multipunto (MPI). El cable de comunicación se suministra junto con la tarjeta MPI.

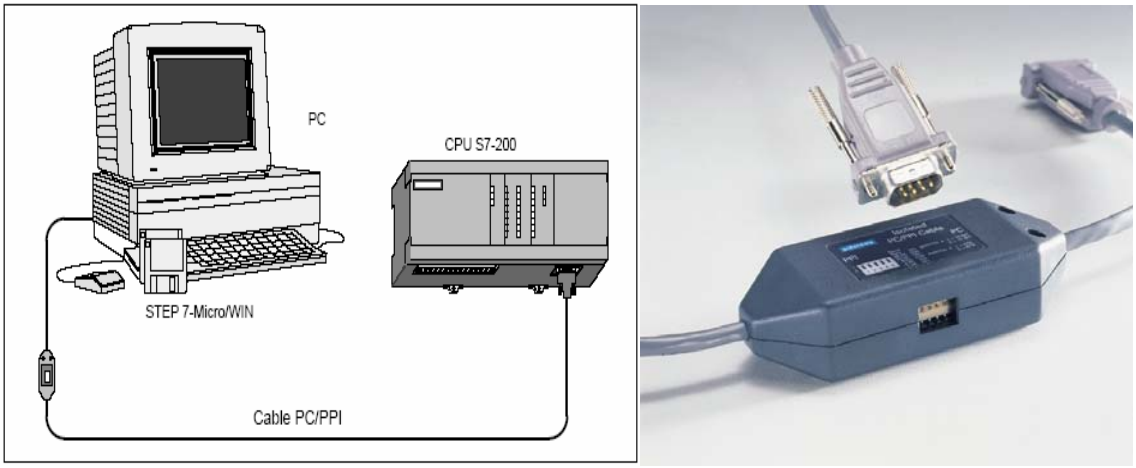


Figura 2.1 *Componentes de un Micro-PLC S7-200 y detalle del cable PC/PPI*

2.1.2 Capacidad de las CPUs S7-200

La serie S7-200 comprende diversas CPUs. Por lo tanto, se dispone de una amplia gama de funciones que permiten diseñar soluciones de automatización a un precio razonable. La Tabla 2.1 resume las principales funciones de cada CPU.

	CPU 221	CPU 222	CPU 224	CPU 226	226 XM
E/S integradas	6 DI / 4 DO	8 DI / 6 DO	14 DI /10 DO	24 DI /16 DO	24 DI /16 DO
Máx. nº E/S con EMs	-	40 / 38	94 / 74	128 / 120	128 / 120
Máx. nº de canales	10	78	168	248	248
Canales Analógicos	-	8 / 4 / 10	28 / 14 / 35	28 / 14 / 35	28 / 14 / 35
Mem. de prog/datos	4 KB / 2 KB	4 KB / 2 KB	8 KB / 5 KB	8 KB / 5 KB	16 KB/10 KB
Tiempo ejec/instruc.	0,37 µs	0,37 µs	0,37 µs	0,37 µs	0,37 µs
Marc./Contad./Temp.	256/256/256	256/256/256	256/256/256	256/256/256	256/256/256
Contadores rápidos	4 x 30 kHz	4 x 30 kHz	6 x 30 kHz	6 x 30 kHz	6 x 30 kHz
Reloj en tiempo real	opcional	opcional	Integrado	Integrado	Integrado
Salidas de impulsos	2 x 20 kHz	2 x 20 kHz	2 x 20 kHz	2 x 20 kHz	2 x 20 kHz
Puertos de comun.	1 x RS 485	1 x RS 485	1 x RS 485	2 x RS 485	2 x RS 485
Potenciómetros anal.	1	1	2	2	2

Tabla 2.1 *Resumen de las CPUs S7-200*

2.2 Principales componentes de un Micro-PLC S7-200

Un Micro-PLC S7-200 puede comprender una CPU S7-200 sola o conectada a diversos módulos de ampliación opcionales.

2.2.1 CPU S7-200

La CPU S7-200 es un aparato autónomo compacto que incorpora una unidad central de procesamiento, una fuente de alimentación, así como entradas y salidas digitales. Hemos de indicar que, aún siendo similares, existen dos subfamilias dentro de la gama S7-200, la S7-21X y la S7-22X que se corresponde con versiones más modernas y potentes de la anterior. Aquí nos referiremos a la S7-21X que es de la que se dispone en el laboratorio y con la que se ha trabajado consecuentemente.

- La CPU (Central Procesing Unit) es la encargada de ejecutar el programa de usuario y de ordenar la transferencia de información en el sistema de entradas y salidas. Esta parte del autómata, toma de memoria las instrucciones una a una y realiza las operaciones asignadas con el fin de ejecutar el programa de usuario. El funcionamiento se realiza, salvo raras ocasiones, decodificando las instrucciones cada vez que son ejecutadas.

- La memoria de trabajo es el almacén donde el autómata guarda todo cuanto necesita para ejecutar la tarea de control:
    - Datos de programa: Señales de E/S, variables internas y datos alfanuméricos y constantes.
    - Datos de control: Instrucciones de usuario (programas) y configuración del autómata.

- La memoria de datos almacena el estado de las variables que maneja el autómata: Entradas, salidas, contadores, etc. La memoria interna fija sus características en función a la capacidad de direccionamiento de E/S, y número y tipo de variables internas manipuladas. Las variables contenidas en la memoria interna pueden ser modificadas todas las veces que se desee, por lo que esta actualización continua obliga a construir esta área con memorias de tipo RAM. El área de memoria almacena las últimas señales leídas en la entrada y enviadas a la salida, actualizándose después de cada ejecución completa del programa.

- La memoria de programa almacena el programa de usuario aunque también puede contener datos alfanuméricos y textos variables. Las memorias de usuario suelen ser RAM + batería o EPROM/EEPROM.

  El conjunto de direcciones correspondientes a todas las posiciones de memoria que puede direccionar la CPU se denomina mapa de memoria y su longitud depende de tres factores:
    - De la capacidad de direccionamiento de la CPU.
    - Número de E/S conectadas, que determina la longitud de la memoria imagen E/S.
    - Longitud de la memoria de usuario utilizada.

- La fuente de alimentación adapta las tensiones necesarias para el buen funcionamiento de los distintos circuitos del sistema. Debido a que el autómata está formado por bloques que requieren tensiones y potencias de diferentes niveles no es de extrañar que la alimentación se obtenga de varias fuentes separadas, procurando independizar las siguientes partes del circuito:
    - Unidad central e interfaces E/S.
    - Alimentación de las entradas.
    - Alimentación de las salidas de tipo electromagnético.

En casi todos los autómatas se requieren dos fuentes, una para la alimentación del autómata y otra para los emisores de señal y para los actuadores de salida. La primera, la del autómata, incorpora una batería de tampón que se utiliza para el mantenimiento de algunas posiciones internas y del programa de usuario cuando falla la alimentación o se desconecta el autómata. En Siemens este tipo de módulo lleva el nombre de PS. No obstante, nuestro sistema contará únicamente con la fuente capaz de proporcionar 1.3 A a 24 Vdc.

- Las entradas y salidas controlan el sistema de automatización. Las entradas vigilan las señales de los aparatos de campo (por ejemplo sensores e interruptores) y las salidas vigilan las bombas, motores u otros dispositivos del proceso.

  Los módulos de E/S establecen la comunicación entre la unidad central y el proceso, filtrando, adaptando y codificando de forma comprensible para dicha unidad las señales procedentes de los elementos de entrada, y decodificando y amplificando las señales generadas durante la ejecución del programa antes de enviarlas a los elementos de salida. Por el tipo de señal, se pueden clasificar en:
  - Digital de 1 bit.
  - Digitales de varios bits.
  - Analógicas.

- El interfaz de comunicación permite conectar la CPU a una unidad de programación o a otros dispositivos. Algunas CPUs S7-200 disponen de dos interfaces de comunicación. Para la comunicación M2M o hombre-máquina y viceversa, es posible equipar el PLC con procesadores de comunicación. A ellos se les puede conectar diferentes periféricos como por ejemplo impresoras, terminales monitores, así como otros autómatas y computadoras. A este tipo de módulos, Siemens les da el nombre de CP y ejemplos de comunicaciones que se pueden establecer son:
  - AS-interfaz
  - Industrial Ethernet
  - PROFIBUS
  - Point to Point

- Los diodos luminosos indican el modo de operación de la CPU (RUN o STOP), el estado de las entradas y salidas integradas, así como los posibles fallos del sistema que se hayan detectado.

La Figura 2.2, se muestra el esquema de una CPU S7-22X.

1. Salidas digitales integradas
2. LEDs de estado de las salidas digitales
3. Terminales de alimentación
4. Conmutador Stop/Run
5. Conector para el cable de ampliació
6. LEDs de estado de la CPU
7. Ranura para el cartucho de memoria
8. Puerto de comunicaciones (p. e. PPI)
9. Entradas digitales integradas
10. LEDs de estado de las entradas digitales

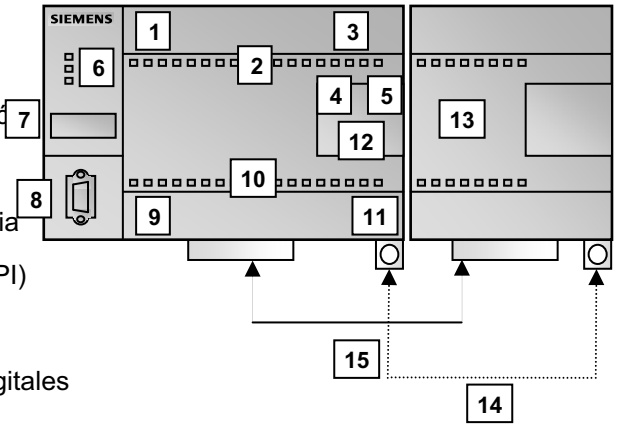


Figura 2.2 ***CPU 22X***

2.2.2 Módulos de ampliación

Los módulos de ampliación para las CPU S7-200 ofrecen un número determinado de entradas y salidas integradas. Si se conecta un módulo de ampliación se dispondrá de más entradas y salidas. Como muestra la Figura 2.3, los módulos de ampliación disponen de un conector de bus para su unión al aparato central.

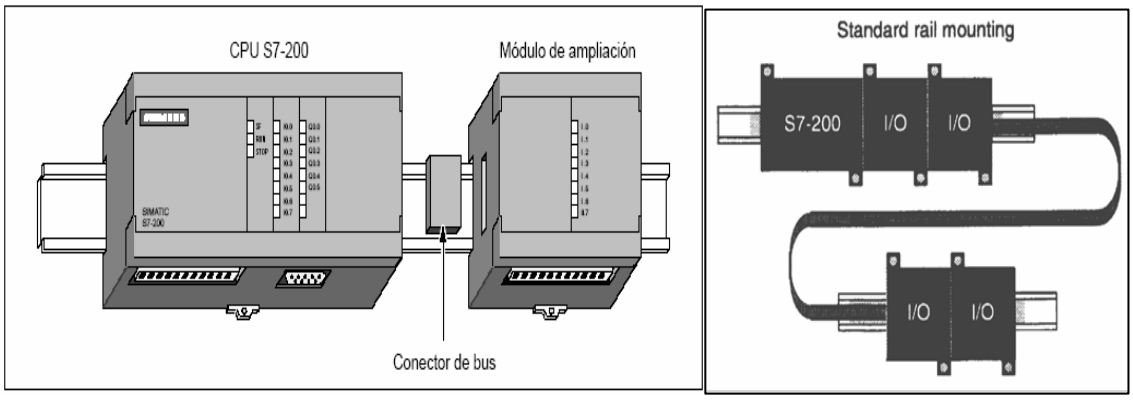


Figura 2.3 ***CPU con un módulo de ampliación y esquema de instalación en dos filas***

Se denomina aparato central o bastidor 0 a la configuración compuesta por fuente de alimentación, CPU y tarjetas periféricas. A la derecha de la CPU puede montarse como máximo 8 módulos (SM, FM CP). Es posible en Siemens disponer al menos para la gama 300 de una ampliación del sistema sin CPU en otros bastidores. Las interfaces se encargarán de acoplar el aparato de ampliación con el aparato central, procurando la continuidad del BUS posterior de un bastidor al siguiente.

Para tareas especiales se disponen de tarjetas procesadoras de señales, que pueden facilitar por ejemplo:

- Cómputo de trenes de impulsos de alta frecuencia.
- Lectura de procesamientos de desplazamientos.
- Medida de velocidad y tiempo.
- Regulación de velocidad y de accionamientos.

Estas tarjetas disponen normalmente de procesador propio y descargan con ello a la CPU de realizar tareas que incluso les puede llevar mucho tiempo y agotar o ampliar demasiado el tiempo del ciclo de la propia CPU. De esta forma se permite ejecutar en paralelo tareas de medida regulación y mando. En Siemens, a este tipo de módulo se le suele dar el nombre de FM o módulo de funciones especiales.

2.3 Ciclo de funcionamiento

El funcionamiento del autómata es, salvo el proceso inicial que sigue a un reset, de tipo secuencial y cíclico, es decir, las operaciones tienen lugar una tras otra y se van repitiendo continuamente mientras el autómata está bajo tensión. (Ver Figura 2.4).

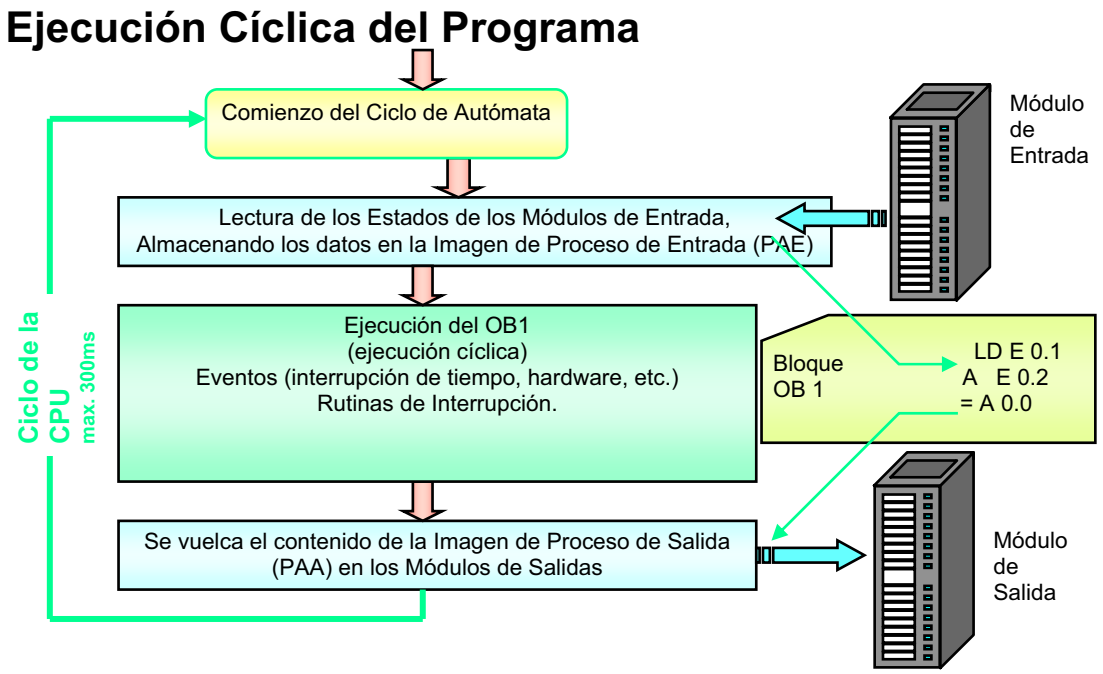


Figura 2.4 *Ciclo de funcionamiento del autómata*

2.3.1 Proceso inicial

En el proceso inicial el autómata se dedica a chequear el hardware mediante unas rutinas ubicadas en el monitor ROM y sus cometidos son comprobar:

- El bus de conexión de las unidades de E/S.
- El nivel de la batería.
- La conexión de las memorias internas del sistema.
- El módulo de memoria exterior conectado si existe.

Si se encuentra algún error en este proceso se encenderá el LED de ERROR y se podrá parar el chequeo según la magnitud del fallo. Comprobadas las conexiones, se inicializan las variables internas, es decir, se ponen a cero las posiciones de la memoria interna, se borran todas las posiciones de memoria imagen de E/S y se borran todos los contadores y temporizadores.

2.3.2 Proceso común

En el proceso común se comprueba el reloj de guarda y se realizan los chequeos de conexiones y de memoria de programa protegiendo al sistema contra errores de hardware y de sintaxis en el programa de usuario. Este proceso no suele superar uno ó dos milisegundos.

2.3.3 Ejecución del programa de usuario

En el bloque de ejecución del programa de usuario se consultan y actualizan los estados de las entradas y las salidas y se elaboran las órdenes de mando a partir de ellos. El tiempo de ejecución de este bloque depende de los siguientes factores:

- Del tiempo de acceso a interfaces de E/S: Este factor depende de sí las interfaces están cableadas como locales (a través del bus interno) o como remotas (conectadas a la CPU mediante el procesador de comunicaciones).

- Número de entradas y salidas instaladas.

2.3.4 Servicio a periféricos externos

El último bloque es el de servicio a periféricos externos. Este bloque sólo se atiende si hay algún intercambio con el exterior. Estos periféricos se comunican con el autómata, bien por un conector situado en la CPU, o bien a través de procesadores de comunicación específicos. El conector de la CPU se suele reservar para la unidad de programación. Una vez establecida la comunicación con los periféricos, la CPU dedica solamente 1 ó 2 milisegundos en atender los intercambios de datos, si no se ha terminado en este tiempo, se interrumpe la comunicación hasta el siguiente ciclo.

2.4 Lenguajes de programación para las CPUs S7-200

Las CPUs S7-200 (y STEP 7-Micro/WIN) ofrecen tres lenguajes de programación: AWL, KOP y FUP. La lista de instrucciones (AWL) comprende un juego de operaciones nemotécnicas que representan las funciones de la CPU. El esquema de contactos (KOP) es un lenguaje de programación gráfico con componentes similares a los elementos de un esquema de circuitos. El Esquema de Funciones Lógicas (FUP) utiliza “cajas” para cada función. El símbolo que se encuentra dentro de la caja indica su función.

STEP 7-Micro/WIN ofrece además dos representaciones nemotécnicas para visualizar las direcciones y las operaciones del programa: internacional y SIMATIC. Tanto la nemotécnica internacional como la de SIMATIC se refieren al mismo juego de operaciones del S7-200. Hay una correspondencia directa entre las dos representaciones, siendo idénticas las funciones de ambas.

2.4.1 Elementos básicos de KOP

Al programar con KOP, se crean y se disponen componentes gráficos que conforman un segmento de operaciones lógicas. Como muestra la figura 6-3, se ofrecen los siguientes elementos básicos para crear programas:

- Contactos: un contacto representa un interruptor por el que circula la corriente cuando está cerrado.

- Bobinas: una bobina representa un relé que se excita cuando se le aplica tensión.

- Cuadros: un cuadro representa una función que se ejecuta cuando la corriente circula por él.
- Segmentos: cada uno de estos elementos constituye un circuito completo. La corriente circula desde la barra de alimentación izquierda pasando por los contactos cerrados para excitar las bobinas o cuadros.

  Para más detalle consultar la Figura 2.5.

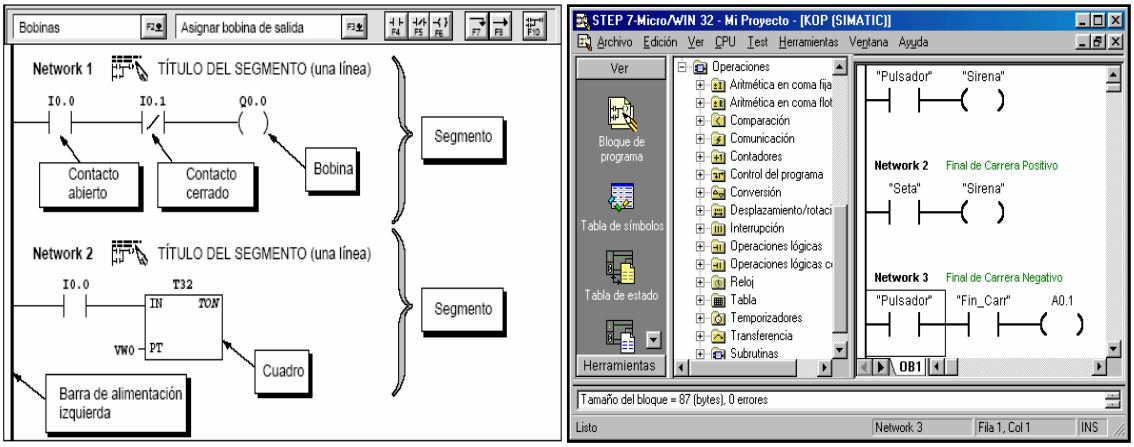


Figura 2.5 **Elementos *básicos de KOP y muestra de una ventana de STEP 7***

2.4.2 Operaciones de AWL

La lista de instrucciones (AWL) es un lenguaje de programación en el que cada línea del programa contiene una operación que utiliza una abreviatura nemotécnica para representar una función de la CPU. Las operaciones se combinan en un programa, creando así la lógica de control de la aplicación.

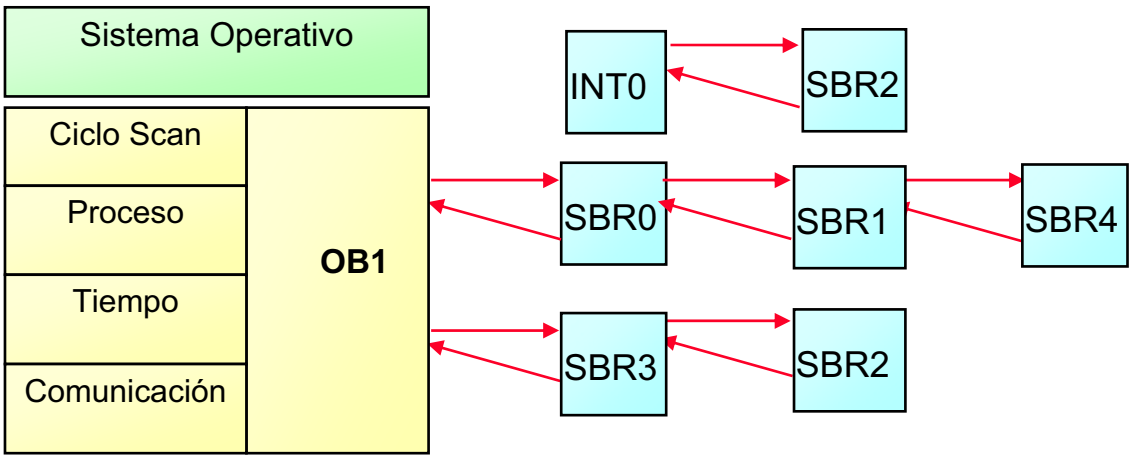
Las instrucciones básicas de manejo de datos, se centran en el manejo de las entradas y salidas del equipo tanto analógicas como digitales y las marcas y marcas de sistema. Dichas marcas se corresponden con variables de memoria interna multipropósito direccionable a nivel de bit, Byte, Word o Dword, las marcas de sistema tienen la misma estructura pero son utilizadas por el PLC para configurar su comportamiento en distintas situaciones. La forma más simple de modificar estas variables es el uso de las instrucciones set y reset para cargar unos y ceros lógicos respectivamente a nivel de bit (`S M0.1` activa el bit 1 de la marca 0 por ejemplo) y la instrucción de carga inmediata `movx`, donde la x debe ser sustituida por b, w ó d según se trate de almacenamiento a nivel de Byte, Word o Dword (`MOVB 16#A4,QB0` carga en el primer byte de salidas digitales la cifra hexadecimal A4. Indiquemos igualmente que el almacenamiento en marcas se indica con la letra `M`, marcas de sistema como `SM`, entradas y salidas digitales con `I` y `Q` respectivamente y entradas y salidas analógicas con `AI` y `AQ` respectivamente.

2.5 Estructura de programación STEP 7

Para llevar a cabo la programación de los autómatas se tienen que utilizar el software apropiado que hace de interfaz entre el usuario y el autómata, bien para cargar, depurar u observar las variables. En el caso del Siemens y más concretamente de los autómatas de la gama S7 200, se utiliza como programa STEP 7-Micro/WIN. El software de programación STEP 7-Micro/WIN permite estructurar el programa de usuario, es decir, subdividirlo en secciones individuales. Esto aporta las siguientes ventajas propias de la programación modular:

- Los programas de gran tamaño se pueden "escribir" de forma clara.
- Se pueden estandarizar secciones individuales del programa.
- Se simplifica la organización del programa.
- Las modificaciones del programa pueden ejecutarse más fácilmente.
- Se simplifica el testeo del programa, ya que puede ejecutarse por partes.
- Se simplifica la puesta en servicio.

Para su programación conviene saber que en STEP 7 se disponen de distintos tipos de bloques estructurables como se aprecia en la Figura 2.6



OB	= Bloque de Organización		Máximo 8 niveles de anidamiento
SBR	= Subrutina	0 - 63	
INT	= Rutina de Interrupción	0 - 127	

Figura 2.6 ***Estructura de programación***

2.5.1 OB (Bloque lógico)

Los bloques de organización constituyen el interfaz entre el sistema operativo y el programa de usuario. Son llamados por el sistema operativo y controlan el procesamiento cíclico y controlado por alarmas del programa, el comportamiento de arranque del sistema de automatización y el tratamiento de los errores. Programando los bloques de organización se define el comportamiento de la CPU. Los bloques de organización determinan la secuencia (eventos de arranque) en la que habrán que ejecutarse las diferentes partes del programa. La ejecución de un OB puede ser interrumpida por la llamada de otro OB, dependiendo de la prioridad, un OB podrá o no interrumpir a otro OB, pudiendo, lógicamente, los OBs de mayor prioridad interrumpir a los de menor. La menor prioridad la tiene el OB de tarea no prioritaria. Existen dos formas de ejecutar el programa que son:

- **Ejecución cíclica de programas:** Es la ejecución normal en autómatas programables, es decir, el sistema operativo se ejecuta en un bucle llamado ciclo. Cada vez que se recorre un ciclo, el sistema operativo llama al bloque de organización OB 1 en el programa principal.

- **Ejecución controlada por alarmas:** En esta forma de controlar la ejecución la ejecución cíclica del programa es interrumpida por determinados eventos de arranque (alarmas). Si se presenta tal evento de arranque, se interrumpe el bloque actualmente en tratamiento en el límite de una instrucción y se trata el bloque de organización asignado al evento de arranque. Luego se continúa ejecutando el programa cíclico a partir del punto de interrupción.

De este modo existe la posibilidad de ejecutar sólo en caso necesario aquellas partes del programa de usuario que no deben procesarse cíclicamente. El programa de usuario se puede dividir en programas parciales y distribuir en diferentes bloques de organización.

Si el programa de usuario debe reaccionar a una señal importante que se presente con poca frecuencia (por ejemplo, si el indicador de nivel de un tanque indica que se ha alcanzado el nivel de llenado), el programa parcial que deba correr cuando se emita la señal se puede depositar en un OB que se ejecute de forma controlada por eventos.

2.5.2 Bloques preprogramados

No es necesario programar cada función. Las CPUs S7 ofrecen bloques preprogramados que se pueden llamar desde el programa de usuario.

- **Bloques de funciones del sistema:** Un SFB es un bloque de funciones integrado en la CPU S7. Como los SFBs forman parte del sistema operativo, no se cargan como parte integrante del programa. Al igual que los FBs, los SFBs son bloques "con memoria". Para los SFBs se han de crear también bloques de datos de instancia y cargar en la CPU como parte integrante del programa. Las CPUs S7 ofrecen SFBs para la comunicación vía enlaces configurados y para las funciones especiales integradas como medidas de frecuencia o PIDs de lazo cerrado.

- **Funciones del sistema:** Una función del sistema es una función preprogramada y probada integrada en la CPU S7. La SFC se puede llamar desde el programa. Como las SFCs forman parte del sistema operativo, no se cargan como parte integrante del programa. Al igual que las FCs, las SFCs son bloques "sin memoria". Las CPUs S7 ofrecen SFCs para funciones de copia y de bloque, control del programa, manipulación del reloj y del contador de horas de funcionamiento, transferencia de registros de datos, transferencia de eventos en el modo multiprocesamiento desde una CPU a todas las CPUs insertadas, manipulación de alarmas horarias y de retardo, manipulación de eventos de errores síncronos, eventos de errores de alarma y asíncronos, diagnóstico del sistema, actualización de imágenes del proceso y tratamiento de campos de bits, direccionamiento de módulos, periferia descentralizada, comunicación por datos globales, la comunicación vía enlaces no configurados y generar mensajes de bloque.

2.5.3 FB (Bloques de función)

Los bloques de función son bloques programables "con memoria". Dispone de un bloque de datos asignado como memoria (bloque de datos de instancia, DB). Los parámetros que se transfieren al FB, así como las variables estáticas, se memorizan en dicho DB de instancia, mientras que las variables temporales se memorizan en la pila de datos locales. Los datos memorizados en el DB de instancia no se pierden al concluir el tratamiento del FB. Los datos memorizados en la pila de datos locales se pierden al concluir el tratamiento del FB. Un FB contiene un programa que se ejecuta siempre cuando el FB es llamado por otro bloque lógico. Los bloques de función simplifican la programación de funciones complejas de uso frecuente. A cada llamada de un bloque de función que transfiere parámetros está asignado un bloque de datos de instancia. Mediante la llamada de varias instancias de un FB es posible controlar varios equipos

con un FB. Un FB para un tipo de motor puede controlar, por ejemplo, diferentes motores, utilizando datos de instancia diferentes para los diferentes motores. Los datos para cada motor (tales como número de revoluciones, rampas, tiempo de funcionamiento acumulado, etc.) se pueden memorizar en uno o varios DBs de instancia.

2.5.4 Funciones

Son bloques programables "sin memoria". Las variables temporales de las FCs se memorizan en la pila de datos locales. Estos datos se pierden tras el tratamiento de las FCs. Para fines de memorización de datos, las funciones pueden utilizar bloques de datos globales. Como una FC no tiene memoria asignada, se han de indicar siempre parámetros actuales. A los datos locales de una FC no se pueden asignar valores iniciales. La FC contiene un programa que se ejecuta siempre cuando la FC es llamada por otro bloque lógico. Las funciones se pueden utilizar para devolver un valor de función al bloque invocante.

2.5.5 DB de instancia

A cada llamada de un bloque de función que transfiere parámetros está asignado un bloque de datos de instancia. En el DB de instancia están depositados los parámetros actuales y los datos estáticos del FB. Las variables declaradas en el FB definen la estructura del bloque de datos de instancia. La instancia define la llamada de un bloque de función. Si, por ejemplo, un bloque de función se llama cinco veces en el programa de usuario S7, existen cinco instancias de dicho bloque. Para crear un DB de instancia antes se debe existir el FB asociado. El número de dicho FB se debe indicar al crear el bloque de datos de instancia.

- **Un DB de instancia para cada instancia:** Si se asignan varios bloques de datos de instancia a un bloque de función (FB) que controla un motor, se puede utilizar este FB para controlar varios motores. Los diversos datos de cada uno de los motores (por ejemplo, número de revoluciones, tiempo de aceleración, tiempo total de servicio) se memorizan en los diversos bloques de datos. Dependiendo de qué DB se asigne al FB, al efectuar la llamada, se puede controlar un motor diferente

- **Un DB de instancia para varias instancias de un FB (multiinstancias):** A un FB se pueden transferir conjuntamente en un DB de instancia los datos de instancia para diferentes motores. A tal efecto, la llamada de los controles de motores se ha de efectuar en otro FB y en el área de declaración del FB invocante se deben declarar las variables estáticas con el tipo de datos de un FB para las diferentes instancias. Utilizando un DB de instancia para varias instancias de un FB se ahorra capacidad de memoria y optimiza el uso de los bloques de datos.

- **Un DB de instancia para varias instancias de FB diferentes (multiinstancias):** En un bloque de función se pueden llamar a instancias de otros FBs ya existentes. Los datos de instancia necesarios al respecto se pueden

asignar al bloque de datos de instancia del FB invocante, es decir que en este caso no se necesitan bloques de datos adicionales para los FBs que se han llamado. Para dichas multiinstancias, un DB de instancia deberá declarar, en la tabla del FB invocante, variables estáticas del mismo tipo del FB llamado, haciéndolo para cada una de las instancias. La llamada en el FB se efectúa entonces sólo con el nombre de la variable, es decir, sin indicar DB de instancia.

2.5.6 DBs de datos globales

Al contrario de los bloques lógicos, los bloques de datos no contienen instrucciones STEP 7. Sin embargo, contienen datos variables con los que trabaja el programa de usuario. Los bloques de datos globales contienen datos de usuario utilizables desde otros bloques. El tamaño de los DBs puede variar. La estructura de bloques de datos globales se puede definir discrecionalmente.

2.5.7 Rutinas de interrupción

Las rutinas de interrupción son funciones especiales opcionales que sólo son llamadas cuando se produce un evento que previamente hemos asociado a dicha rutina de interrupción. Se deben añadir siempre al final del programa principal, utilizando siempre una operación retorno absoluto desde rutina de interrupción (RETI) para terminarla. Toda rutina de interrupción se puede identificar con una marca de interrupción que indica el comienzo de la rutina. Ésta comprende las operaciones que se colocan entre dicha marca y la operación retorno absoluto desde rutina de interrupción.

2.6 Comunicación en redes con CPUs S7-200

Las CPUs S7-200 poseen diversos métodos de comunicación. Dependiendo de la CPU utilizada, la red ofrece uno o varios de los siguientes protocolos de comunicación:

- Interfaz punto a punto (PPI).
- Interfaz multipunto (MPI).
- PROFIBUS-DP.

CPU	Inter-faz	Esclavo PPI	Maestro PPI	Esclavo PROFIBUS	Esclavo MPI	Freeport	Velocidad de transferencia
222	0	Sí	No	No	No	Sí	9,6 kbit/s,19,2 kbit/s
224	0	Sí	Sí	No	No	Sí	9,6 kbit/s,19,2 kbit/s
215	0	Sí	Sí	No	Sí	Sí	9,6 kbit/s,19,2 kbit/s
	DP, DPV2	No	No	Sí	Sí	No	9,6 kbits/s, 19,2 kbits/s, 500 kbits/s, 1 Mbit/s,1,5 Mbit/s, 3 Mbit/s, 6 Mbit/s, 12 Mbit/s
226	0	Sí	Sí	No	Sí	Sí	9,6 kbit/s,19,2 kbit/s
	1	Sí	Sí	No	Sí	Sí	9,6 kbit/s,19,2 kbit/s

Tabla 2.2 ***Posibilidades de comunicación de las CPUs S7-200***

Estos protocolos se basan en la intercomunicación de sistemas abiertos (OSI) de la arquitectura de siete capas. Los protocolos PPI, MPI y PROFIBUS-DP se implementan en una red "token ring" (red de anillo con testigo) conforme al estándar Process Field Bus (PROFIBUS) que se describe en la norma europea EN 50170.

Se trata de protocolos asíncronos de caracteres que utilizan un bit de inicio, ocho bits de datos, un bit de paridad par y un bit de parada. Los bloques de comunicación dependen de los caracteres especiales de inicio y de parada, de las direcciones de estación de fuente y de destino, de la longitud de dichos bloques y de la suma de verificación para garantizar la integridad de los datos. Los tres protocolos se pueden utilizar simultáneamente en una red sin que interfieran entre sí, con la condición de que usen una misma velocidad de transferencia.

La red PROFIBUS utiliza el estándar RS-485 con cables de par trenzado. Ello permite interconectar hasta 32 dispositivos en un segmento de la red. Los segmentos pueden tener una longitud máxima de 1.200 m, dependiendo de la velocidad de transferencia. Es posible conectar repetidores para poder incorporar más dispositivos en la red o con objeto de utilizar cables más largos. Si se usan repetidores, las redes pueden tener una longitud de hasta 9.600 m, dependiendo de la velocidad de transferencia.

Los protocolos prevén dos tipos de dispositivos de red: los maestros y los esclavos. Los maestros pueden enviar una petición a otros dispositivos. En cambio, los esclavos sólo pueden responder a las peticiones de los maestros, sin poder lanzar nunca una petición por su propia cuenta.

Estos protocolos proveen 127 direcciones (0 a 126) en una red. Una red puede comprender 32 maestros como máximo. Todos los dispositivos que formen parte de una red deberán tener direcciones unívocas para poder comunicarse entre sí. El ajuste estándar para las unidades de programación SIMATIC y para los PCs con STEP 7-Micro/WIN es la dirección "0". El visualizador de textos TD 200 y los paneles de operador OP3 y OP7 tienen la dirección predeterminada "1". La dirección estándar de los sistemas de automatización es "2".

2.6.1 Protocolos definidos por el usuario (Freeport)

La comunicación Freeport es un modo de operación con el que el programa de usuario puede controlar el interfaz de comunicación de la CPU S7-200. Con el modo Freeport se pueden implementar protocolos de comunicación definidos por el usuario para crear enlaces con numerosos dispositivos inteligentes como se aprecia en el esquema de la Figura 2.7.

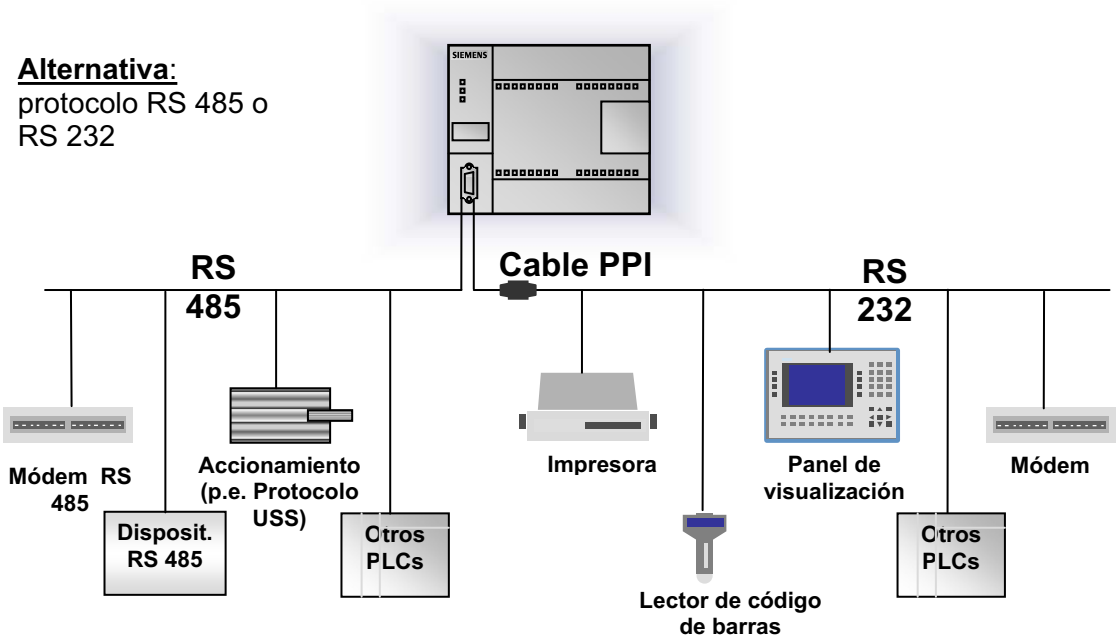


Figura 2.7 *Posibilidades de comunicación del modo Freeport*

El programa de usuario controla el funcionamiento del interfaz de comunicación utilizando interrupciones de recepción y de transmisión, así como las operaciones Transmitir mensaje (XMT) y Recibir mensaje (RCV). En modo Freeport, el programa de usuario controla por completo el protocolo de comunicación. El modo Freeport se habilita con las marcas especiales SMB30 (interfaz 0) y SMB130 (interfaz 1), estando activo únicamente cuando la CPU se encuentre en modo RUN. Cuando la CPU retorna a modo STOP, la comunicación Freeport se detiene y el interfaz de comunicación vuelve a utilizar el protocolo PPI normal.

En el caso más simple se puede enviar un mensaje a la impresora o a la pantalla con sólo utilizar la operación Transmitir mensaje (XMT). Otros ejemplos incluyen la conexión a un lector de código de barras, una báscula o una soldadora. En todo caso, el programa deberá asistir el protocolo con el que la CPU se comunica en modo Freeport.

Para poder utilizar el modo Freeport, es preciso que la CPU esté en modo RUN. El modo Freeport se habilita ajustando el valor 01 en el campo de selección del protocolo de SMB30 (interfaz 0) o de SMB130 (interfaz 1). Estando en modo Freeport, la CPU no se puede comunicar con la unidad de programación.

El paso a modo Freeport se puede controlar con la marca especial SM0.7 que indica la posición actual del selector de modos de operación. Si SM0.7 = 0, el selector está en posición TERM. Si SM0.7 = 1, el selector está en posición RUN. Si el modo Freeport se habilita sólo cuando el selector esté en RUN, la unidad de programación se podrá utilizar para vigilar o controlar el funcionamiento de la CPU, cambiando el selector a una posición diferente.

2.6.1.1 Inicializar el modo Freeport

SMB30 y SMB130 se utilizan para inicializar el modo Freeport en los interfaces de comunicación 0 y 1, respectivamente, permitiendo elegir la velocidad de transferencia, la paridad y el número de bits por carácter. La Tabla 2.3 muestra los bytes de control del modo Freeport.

Interface 0	Interface 1	Descripción
Formato de SMB30	Formato de SMB130	MSB 7 … LSB 0 p p d b b b m m Byte de control del modo Freeport
SM30.6 y SM30.7	SM130.6 y SM130.7	pp Selección de paridad 00 = sin paridad 01 = paridad par 10 = sin paridad 11 = paridad impar
SM30.5	SM130.5	d Bits por carácter 0 = 8 bits por carácter 1 = 7 bits por carácter
SM30.2 a SM30.4	SM130.2 a SM130.4	bbb Velocidad de transferencia 000 = 38.400 bits/s (para la CPU 212: = 19.200 bits/s) 001 = 19.200 bits/s 010 = 9.600 bits/s 011 = 4.800 bits/s 100 = 2.400 bits/s 101 = 1.200 bits/s 110 = 600 bits/s 111 = 300 bits/s
SM30.0 y SM30.1	SM130.0 y SM130.1	mm Selección de protocolo 00 = Protocolo de interface punto a punto (PPI/modo esclavo) 01 = Protocolo Freeport 10 = PPI/modo maestro 11 = Reservado (estándar: PPI/modo esclavo)
Nota: En el caso del interface 0 se genera un bit de parada para todas las configuraciones, con excepción de los 7 bits por carácter (sin paridad), donde se generan dos bits de parada. En el caso del interface 1 se genera un bit de parada para todas las configuraciones.		

Tabla 2.3 ***Bytes de marcas especiales SMB30 y SMB130***

2.6.1.2 Utilizar la operación XMT para transmitir datos

La operación XMT facilita la transferencia de datos. Con dicha operación se puede enviar un búfer de uno o más caracteres (hasta un máximo de 255). Una vez transmitido el ultimo carácter del búfer, se genera una interrupción (evento de interrupción 9 para el interfaz 0 y evento de interrupción 26 para el interfaz 1), si una rutina de interrupción se ha asociado al evento Transmisión finalizada. También es posible transmitir datos sin utilizar interrupciones (por ejemplo enviar un mensaje a una impresora), vigilando SM4.5 ó SM4.6 hasta que finalice la transmisión.

2.6.1.3 Utilizar la operación RCV para recibir datos

La operación RCV facilita la recepción de mensajes. Con dicha operación se puede recibir un búfer de uno o más caracteres (hasta un máximo de 255). Una vez recibido el ultimo carácter del búfer, se genera una interrupción (evento de interrupción 23 para el interfaz 0 y evento de interrupción 24 para el interfaz 1), si una rutina de interrupción se ha asociado al evento Recepción de mensajes finalizada. También es posible recibir mensajes sin utilizar interrupciones, vigilando a tal efecto la marca especial SMB86.

SMB86 (o SMB186) no será igual a cero al estar desactivado el cuadro RCV. En cambio, será igual a cero cuando se estén recibiendo datos.

La operación RCV permite seleccionar las condiciones para el comienzo y el final de un mensaje. La Tabla 2.4 (SMB86 a SMB94 para el interfaz 0 y SMB186 a SMB194 para el interfaz 1) describe dichas condiciones.

Es importante reseñar que la recepción de mensajes se finalizará de modo automático si se produce un desbordamiento o un error de paridad. Para la operación Recibir mensaje es preciso definir una condición inicial (x ó z) y una condición final (y, t ó el número máximo de caracteres). En la Tabla 2.4 se puede apreciar con mayor nitidez todo lo expuesto.

Interface 0	Interface 1	Descripción
SMB86	SMB186	MSB 7 … LSB 0 n r e 0 0 t c p — Byte de estado de recepción de mensajes n: 1 = El usuario ha inhibido la recepción de mensajes r: 1 = Se finalizó la recepción de mensajes: error en parámetros de entrada o falta condición inicial o final e: 1 = Carácter final recibido t: 1 = Se finalizó la recepción de mensajes: ha transcurrido la temporización c: 1 = Se finalizó la recepción de mensajes: se ha excedido el número máximo de caracteres p: 1 = Se finalizó la recepción de mensajes debido a un error de paridad
SMB87	SMB187	MSB 7 … LSB 0 n x y z m t 0 0 — Byte de control de recepción de mensajes n: 0 = Inhibida la función de recibir mensajes. 1 = Habilitada la función de recibir mensajes. El bit para habilitar/inhibir la recepción de mensajes se comprueba cada vez que se ejecuta la operación RCV. x: 0 = Ignorar SMB88 o SMB188. 1 = Utilizar el valor de SMB88 o de SMB188 para detectar el comienzo del mensaje. y; 0 = Ignorar SMB89 o SMB189. 1 = Utilizar el valor de SMB89 o de SMB189 para detectar el fin del mensaje. z: 0 = Ignorar SMW90 o SMB190. 1 = Utilizar el valor de SMW90 para detectar una condición de inactividad. m: 0 = Utilizar el temporizador como temporizador entre caracteres. 1 = Utilizar el temporizador como temporizador de mensajes. t: 0 = Ignorar SMW92 o SMW192. 1 = Finalizar la recepción si se excede el período de tiempo indicado en SMW92 o SMW192. Estos bits definen los criterios para identificar el mensaje (incluyendo los criterios para el comienzo y el fin del mensaje). Para determinar el comienzo de un mensaje, los criterios habilitados a tal efecto se combinan mediante Y, debiendo presentarse en forma de secuencia (línea de inactividad seguida de un carácter inicial). Para determinar el fin de un mensaje, los criterios habilitados a tal efecto se combinan mediante O. Ecuaciones de los criterios para el comienzo y el fin de un mensaje: Comienzo del mensaje = z * x Fin del mensaje = y + t + número máximo de caracteres alcanzados Nota: La recepción de mensajes se finalizará automáticamente si se produce un desbordamiento o un error de paridad. Para la operación Recibir mensaje es preciso definir una condición inicial (x ó z) y una condición final (y, t ó el número máximo de caracteres).
SMB88	SMB188	Carácter de comienzo del mensaje.
SMB89	SMB189	Carácter de fin del mensaje.
SMB90 SMB91	SMB190 SMB191	Tiempo de línea de inactividad en milisegundos. El primer carácter recibido una vez transcurrido el tiempo de línea de inactividad es el comienzo del nuevo mensaje. SM90 (o SM190) es el byte más significativo y SM91 (o SM191) es el byte menos significativo.
SMB92 SMB93	SMB192 SMB193	Vigilancia de tiempo del temporizador entre caracteres/de mensajes en milisegundos. Si se excede el tiempo, se finaliza la recepción de mensajes. SM92 (o SM192) es el byte más significativo y SM93 (o SM193) es el byte menos significativo.
SMB94	SMB194	Número máximo de caracteres a recibir (1 a 255 bytes). Nota: Este margen debe ajustarse al tamaño máximo esperado para el búfer, incluso si no se utiliza la terminación de mensajes por el contaje de caracteres.

Tabla 2.4 ***Marcas especiales SMB86 a SMB94 y SMB186 a SMB194***

2.6.1.4 Recibir datos mediante interrupciones de caracteres

Para disponer de una mayor flexibilidad en los protocolos asistidos, los datos se pueden recibir también de forma controlada por interrupciones de caracteres. Cada carácter recibido se deposita en SMB2 y el estado de la paridad (si se ha habilitado) se deposita en SM3.0 y, finalmente, se genera una interrupción. Ello sucede justo antes de ejecutarse la rutina de interrupción asociada al evento Recibir carácter.

SMB2 es el búfer de recepción de caracteres en modo Freeport. Cada carácter recibido en modo Freeport se deposita en esta dirección para que el programa de usuario pueda acceder rápidamente a los valores. SMB3 contiene un bit de error de paridad. Todos los demás bits del byte se reservan.

SMB2 y SMB3 son compartidos por los interfaces 0 y 1. Si debido a la recepción de un carácter por el interfaz 0 se ejecuta la rutina de interrupción asociada a ese evento (evento de interrupción 8), SMB2 contendrá el carácter recibido por el interfaz 0, en tanto que SMB3 contendrá la paridad de dicho carácter. Ídem para el interfaz 1 (evento de interrupción 25).

2.6.1.5 Utilización del cable PC/PPI en modo Freeport

El cable PC/PPI y el modo Freeport se pueden utilizar para conectar las CPUs S7-200 a numerosos dispositivos compatibles con el estándar RS-232, pudiendo asistir velocidades de transferencia comprendidas entre 600 y 38.400 bit/s. Los interruptores DIP dispuestos en la carcasa del cable PC/PPI se emplean para configurar la velocidad de transferencia correcta, detallándose en dicha carcasa como configurar el cable.

El interfaz RS-232 del cable PC/PPI se considera un DCE. Las únicas señales presentes en dicho interfaz son: transmitir datos, recibir datos y tierra. El cable PC/PPI no utiliza ni envía ninguna de las señales de control del RS-232, tales como RTS, CTS, etc. Por tanto, es inútil utilizarlas en nuestra tarjeta, lo cual nos lleva a utilizar un protocolo por hardware emulado sin handshaking que se detalla en 5.3, página 102.

El cable PC/PPI se encuentra en el modo de transmisión cuando los datos se envían del interfaz RS-232 al RS-485. En cambio, se encuentra en modo de recepción al estar en vacío, o bien cuando los datos se transmiten del interfaz RS-485 al RS-232. El cable cambia inmediatamente de modo de recepción a transmisión cuando detecta caracteres en el canal de transmisión del RS-232, cambiando nuevamente a recepción cuando el canal de transmisión del RS-232 está en vacío durante el tiempo de inversión del cable. Dicho tiempo depende de la velocidad de transferencia seleccionada con los interruptores DIP, siendo de 2ms a 9600 bps y 1ms para velocidades mayores, a bajas velocidades, este periodo aumenta progresivamente, alcanzando los 28ms a 600 bps.

La comprensión de este concepto es vital para lograr la interactividad entre un dispositivo RS-232 y la CPU S7-200 pues tras recibir una petición del dispositivo RS-232, la transmisión de una respuesta de la CPU S7-200 se deberá retardar por un período mayor o igual al tiempo de inversión del cable. Del mismo modo, tras recibir una respuesta del dispositivo RS-232, la transmisión de la siguiente petición de la CPU S7-200 se deberá retardar por un período mayor o igual al tiempo de inversión.