AUTÒMATAS PROGRAMABLES

Programacion y Aplicacion Industrial

García Vázquez, C.A., Gil Mena, A.J., Llorens Iborra, F.,
Mañas Sánchez, C.J., Martín García, J.A.

Servicio de Publicaciones
Universidad de Cadiz

Autómatas Programables: Programación y Aplicación Industrial

GARCÍA VÁZQUEZ, C.A., GIL MENA, A. J., LLORENS IBORRA, F., MAÑAS SÁNCHEZ, C.J., MARTÍN GARCÍA, J.A.

ESCUELA POLITÉCNICA SUPERIOR DE ALGECIRAS

DEPARTAMENTO DE INGENIERÍA ELÉCTRICA

UNIVERSIDAD DE CÁDIZ

I.S.B.N.: 84-7786-566-3

Autores:
Carlos Andrés García Vázquez
Antonio José Gil Mena
Francisco Llorens Iborra
Carlos Jesús Mañas Sánchez
Juan Andrés Martín García

Título de la obra:
Autómatas Programables: Programación y aplicación industrial.

Publica:
SERVICIO DE PUBLICACIONES DE LA UNIVERSIDAD DE CADIZ

Autómatas programables : programación y aplicación industrial / García Vázquez, C.A. ... [et. al.]-- Cádiz : Servicio de Publicaciones de la Universidad, 1999. – 232 p.

ISBN 84-7786-566-3

1.Automatismos secuenciales. I. García Vázquez, Carlos Andrés. II. Universidad de Cádiz. Servicio de Publicaciones, ed. III. Título

681.5

Contenido

Preámbulo

Entre las actividades docentes desplegadas por el grupo de profesionales universitarios que componen la Sección Departamental de Ingeniería Eléctrica de la Escuela Politécnica Superior de Algeciras, se encuentra la impartición de cursos de postgrado, canalizados a través del "Secretariado de Títulos Propios y Cursos de Postgrado" de la Universidad de Cádiz.

La gran aceptación e interés que ha suscitado uno de los mismos, *"Autómatas Programables: Programación y Aplicación Industrial"*, realizado en tres ediciones, ha motivado a los profesores que lo imparten a realizar y difundir esta publicación, con el objeto de presentar una introducción a la programación de autómatas de la serie Simatic S5.

La finalidad de esta labor divulgativa es profundizar en el conocimiento a nivel de mando de accionamientos secuenciales, mediante autómatas programables, utilizando para ello variables de tipo binario, las cuales supondrán, en la generalidad de los casos, un instrumento adecuado a las exigencias que las aplicaciones industriales conllevan.

Sin embargo, la automatización de sistemas secuenciales puede resultar caótica, si carece de una estructuración apropiada y, demasiado compleja, si además no dispone de una herramienta específica de resolución de problemas. Las soluciones aportadas por los fabricantes, a través del método gráfico GRAFCET, aunque satisfactorias, pueden ser ampliadas y enriquecidas por las Redes de Petri; método más general que el primero y con aplicaciones más variadas.

Resulta obligado hacer referencia a la falta de normalización entre fabricantes de autómatas. Esta diversidad, aunque permite una comprensión teórica general, impide tener un conocimiento real del funcionamiento y peculiaridades de todos ellos. Dado que en la actualidad existen gran cantidad de obras que contemplan ampliamente el aspecto teórico, se ha planteado en esta publicación dar a conocer un autómata concreto y determinado, para, sobre él, conocer y aprender "de forma práctica" su funcionamiento real.

En cuanto al autómata elegido para el estudio, se ha optado por la familia Simatic-S5 de Siemens, y en particular, por el más potente de los miniautómatas, el 95U. Esta elección se fundamenta en datos contrastados en el entorno industrial de la zona, donde resulta ser el de mayor implantación.

Posponemos para futuras publicaciones el análisis de otros autómatas de potencia similar al que en esta obra hemos considerado.

Estructura

En primer lugar, se realiza una introducción sobre conceptos generales, donde se describe el funcionamiento del autómata, componentes internos y su entorno en cuanto a conexión con el exterior.

En el segundo capítulo, se presenta la gama de la serie Simatic, y se particulariza para el autómata S5-95U, haciendo una descripción de las especificaciones técnicas, de los elementos de servicio e indicadores y una estructuración interna a nivel de bloques.

Ya, en el capítulo tres, se introducen los conceptos teóricos que permiten modelizar automatismos, definiendo un autómata finito, y describiendo las representaciones tabulares y los grafos de estados. A continuación, se presenta la teoría de las redes de Petri, que actualmente se reconoce como el modelo mejor adaptado a la parte secuencial de los automatismos, pues combina las ventajas de la representación secuencial gráfica con la integración de los modelos preexistentes.

Conocida la teoría básica para la descripción funcional del automatismo, a continuación se procede a describir las características de la programación del autómata Simatic S5-95U. Se comienza, por tanto, en el capítulo cuatro con las definiciones de ciclo de tratamiento de un programa, los operandos de STEP 5, el juego de operaciones de las que se dispone y las diferentes formas de representación del lenguaje STEP 5. Seguidamente, se estudia la estructura y elaboración del programa y los tipos de procesamiento. Y por último las posibilidades de documentación que dispone el entorno de programación.

El capítulo cinco, describe de forma detallada y funcional los elementos de programación, es decir, las diferentes instrucciones y funciones (binarias, digitales, organización, sustitución,...), conjuntamente con ejemplos y ejercicios con objeto de contribuir a una mejor asimilación y aprendizaje encadenado de las mismas.

En este instante, se está en disposición de presentar las técnicas de realización (capítulo 6), en otras palabras, cómo a partir de la modelización del sistema y con el conocimiento del lenguaje de programación, se realiza de forma sistemática la implementación en listas de instrucciones del proceso a automatizar. Es en este capítulo donde se refleja la aplicación de todos los conocimientos aprendidos en capítulos anteriores, y a partir del cual se puede empezar a ejercitar de forma práctica el método de programación.

Por último, en el capítulo siete, se resumen las ventajas y aplicaciones de los autómatas, los diferentes tipos de automatización y la conexión del autómata mediante buses. También se esbozan las diferentes herramientas de programación en cuanto a software se refiere, existentes en el mercado, para la familia Simatic S5; y una visión sobre las tendencias de los buses de comunicaciones.

¿Qué le falta a la publicación para aportar un conocimiento total del autómata? Primero una amplia vertiente práctica, para la que se ha considerado conveniente un tratamiento más cercano al autómata, a su programación y testeo "on" y "off-line" (trabajar con el autómata "delante").

Por otra parte el empleo del autómata como controlador, lo que implica básicamente el trabajar con información digital estructurada en palabras.

Por último las comunicaciones a nivel de autómatas, que requieren un tratamiento mucho más específico y necesitan una potencia a nivel de autómata de alto nivel; trabajo más propio del fabricante que de un texto escrito de divulgación.

Respecto a las dos primeras consideraciones se intentarán soslayar con publicaciones futuras; una complementaria a ésta que trate únicamente la parte práctica y otra dedicada a trabajar con información digital.

Capítulo 1

El Autómata Programable. Conceptos Generales

Contenido

1. INTRODUCCIÓN

El autómata programable, y por extensión, el sistema de automatización del que forma parte, es utilizado cada vez más en el control y automatización de sistemas y procesos industriales. Su campo de aplicación abarca desde la sustitución de pequeños relés y contactores hasta los sistemas de control más sofisticados como pueden ser computadoras de procesos.

El autómata será por tanto, una máquina capaz de realizar distintas acciones sobre un sistema en función del estado y evolución de dicho sistema y de unas acciones sobre el mismo. Así pues debe ser capaz de comunicarse con el sistema a automatizar. La automatización la conseguirá ejecutando las instrucciones introducidas en el autómata por el programador y que constituirán el llamado programa de automatización. La ejecución de dicho programa la realiza gracias a que el autómata está constituido internamente por un sistema basado en un microprocesador.

Una característica muy importante en los autómatas es su carácter programable. El programa de automatización que ejecuta puede ser variado por el operario o programador de la máquina. Esto hace que un mismo autómata programable sea muy versátil, capaz de realizar una infinidad de tareas posibles, según el programa introducido en él.

Un sistema de automatización básico estará formado por tres elementos fundamentales:

- El autómata programable,
- el aparato de programación y
- el lenguaje de programación.

En los párrafos anteriores ya se ha hablado del autómata programable, decir únicamente que es el elemento principal de todo sistema de automatización, los encargados de llevar a cabo la automatización.

El siguiente elemento del sistema es el aparato de programación. Mediante este aparato el operario o programador elaborará el programa de automatización y lo introducirá en el autómata. También servirán para monitorizar el correcto funcionamiento tanto del programa en ejecución como del mismo autómata programable.

El lenguaje de programación es el conjunto de instrucciones y normas que hacen posible la comunicación entre el programador y el autómata. Utilizando las instrucciones de este lenguaje el programador realizará el programa que será introducido en el autómata a través del aparato de

programación. Cada fabricante de autómatas programables dispondrá de un lenguaje propio para la programación de sus propios autómatas.

1.1. LÓGICA CABLEADA FRENTE LÓGICA PROGRAMADA

Los sistemas de automatización convencionales utilizan una serie de elementos para el control de una serie de dispositivos (salidas del sistema) como por ejemplo válvulas, lámparas contactos, etc., a partir de unos dispositivos de entrada (interruptores, pulsadores...) y un estado en el sistema. Los elementos de control serán normalmente relés y contactores que se cablearán físicamente entre los dispositivos de entrada y salida. Dependiendo del modo de conexión de estos elementos (serie o paralelo) el sistema realizará distintos tipos de control.

Así pues, los sistemas de automatización convencionales, basados en relés y contactores son sistemas cerrados, o sea, difícilmente modificables ya que cualquier modificación tanto en las entradas, en las salidas o en el tipo de control, conllevará un cambio en las conexiones físicas de los distintos elementos.

En los sistemas de automatización con autómatas programables se sustituyen los relés y contactores y las conexiones entre ellos por el mismo autómata y el programa de automatización. Los contactores se sustituirán por posiciones de memoria en el interior del autómata y las conexiones entre ellos por funciones del tipo Y u O disponibles en el lenguaje de programación. El único cableado físico del sistema será el que haya que realizar entre los elementos de entrada y salida del sistema con las correspondientes tarjetas de entrada y salida del autómata.

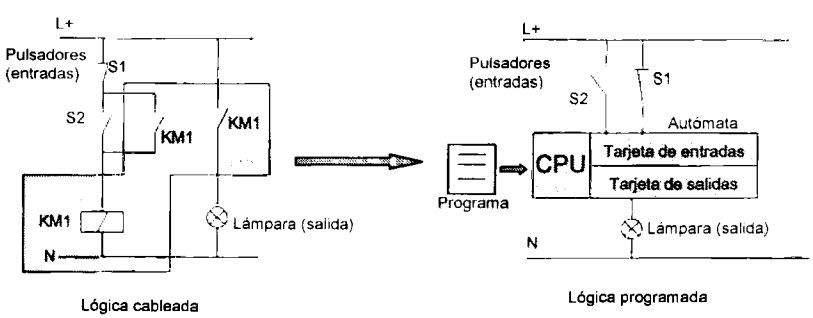


Figura 1.1: Lógica cableada y lógica programada.

Esta última realización hace que un cambio en el sistema sea relativamente sencillo ya que consistiría únicamente en una nueva conexión entre una nueva entrada o salida y el autómata o un cambio en el programa de automatización a través del aparato programador.

2. FUNCIONAMIENTO DE UN AUTÓMATA PROGRAMABLE

La primera tarea a la hora de realizar la automatización de un sistema consiste en estudiar el proceso o máquina a automatizar. Es necesario tener un profundo conocimiento del sistema y su evolución así como planificar minuciosamente la tarea de control o automatización a aplicar en dicho proceso. Debe haber una correspondencia total entre proceso y automatización para que ésta sea efectiva.

Una vez definido el proceso y la automatización a efectuar según las necesidades habrá que plasmar en papel esa automatización utilizando para ello el lenguaje de programación propio del autómata y las funciones y posibilidades que nos ofrezca. Esto se puede hacer directamente o a través de algunos de los métodos existentes de simulación de procesos secuenciales (redes de Petri o Grafcet).

La siguiente tarea es la escritura del programa de automatización ya plasmado en papel en el aparato de programación. Este aparato, según el modelo y tipo, puede tener o no funciones de depuración y comprobación del programa a medida que se escribe.

Una vez realizado el programa de automatización por el usuario, éste debe depositarse en la memoria de programa del autómata utilizando, o bien el aparato de programación, o bien, una memoria EEPROM exterior donde previamente se ha escrito el programa. Cada palabra en memoria del autómata contendrá una instrucción que habrá sido codificada por el aparato de programación en binario, o sea, en '0' y '1'.

El procesador tratará sucesiva y cíclicamente las instrucciones contenidas en el programa. Un contador de programa direcciona cada una de las posiciones de memoria que se incrementa para pasar de direccionar una instrucción a la siguiente. Este proceso lineal se puede ver interrumpido mediante saltos. En este caso el contador se incrementa según lo indicado en la instrucción de salto. Al retornar, se restablece el carácter lineal del proceso.

El procesador toma de la memoria de programa la instrucción direccionada por el contador de programa y la almacena en el registro de instrucción. En este lugar el procesador decodifica la instrucción, o sea, la interpreta y posteriormente la ejecuta, lleva a cabo la tarea indicada en dicha instrucción.

Una vez ejecutada la instrucción el contador de programa se incrementa y el procesador realiza todo el proceso anterior con la siguiente instrucción.

Este proceso continua así hasta que el contador de programa llega a la última instrucción, en cuyo caso, en vez de incrementarse, pasa a direccionar la primera instrucción del programa e iniciar de nuevo la ejecución del mismo.

Por lo tanto, el funcionamiento de un autómata se puede definir como lineal y cíclico.

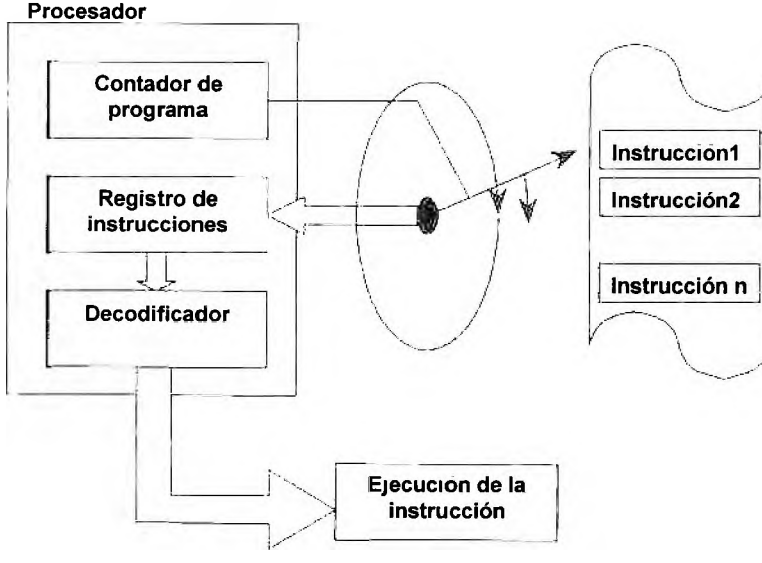


Figura 1.2: Funcionamiento cíclico.

3. EL AUTÓMATA Y SU ENTORNO

La finalidad última de un autómata programable es la automatización de una planta industrial, máquina, o un proceso en general; por lo tanto, necesita

comunicarse con dicho proceso. Esta comunicación debe realizarse en doble sentido, el autómata debe conocer en cada momento la evolución del proceso, para de esta manera evaluar su estado y actuar sobre el mismo de manera oportuna según dicte el programa en ejecución. Esta comunicación entre el proceso y el autómata no se hace de forma directa sino que se realiza a través de unos dispositivos intermedios que adecuen las señales a los requerimientos de ambos.

Los dispositivos de comunicación autómata-proceso se dividen en:

- Captadores (o sensores).- Hacen que las señales procedentes del proceso y que definen el estado del mismo lleguen hasta los terminales de entrada del autómata. (Por ejemplo: final de carrera, elementos de mando...).
- Accionadores (o actuadores).- Hacen que las salidas producidas por el autómata lleguen hasta el proceso y se vean reflejadas en él. (Por ejemplo: relés, contactores, ...).

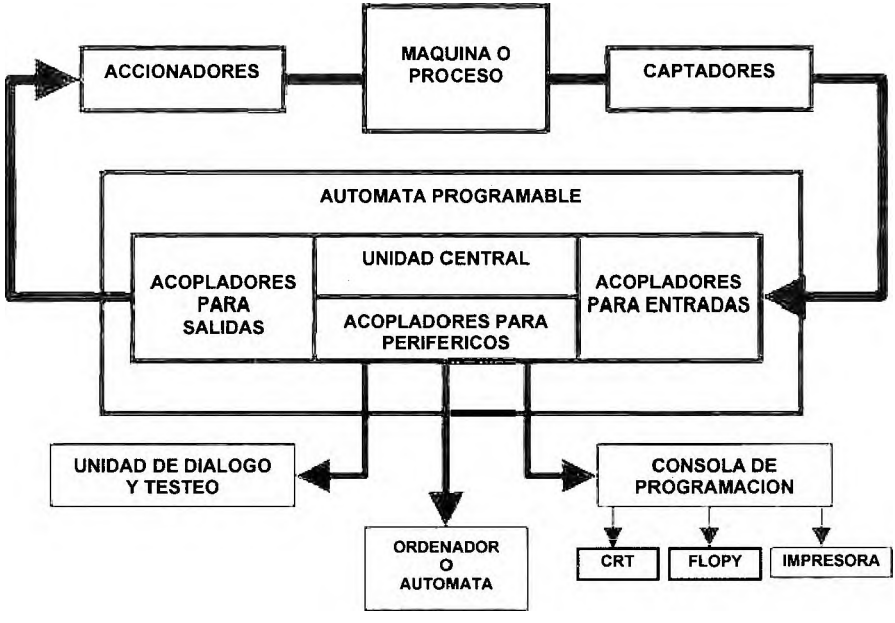


Figura 1.3: El autómata y su entorno.

Además de la comunicación autómata-proceso, el autómata debe comunicarse con una serie de periféricos que ayuden a la programación y monitorización del mismo. Estos periféricos se conectan, o bien directamente, o bien mediante la interface oportuna. Entre estos periféricos auxiliares pueden estar: unidades de programación, impresoras, unidades de almacenamiento, ordenadores, otros autómatas...

De esta manera el funcionamiento de un autómata requiere un entorno de trabajo concreto formado por una serie de dispositivos. Un esquema de este entorno será el mostrado en la figura 1.3.

4. COMPONENTES INTERNOS DEL AUTÓMATA

La estructura interna de un autómata programable es la estructura típica de un sistema basado en un microprocesador.

Dicha estructura esta formada por un elemento central, el microprocesador, encargado de gestionar el funcionamiento de todo el sistema. Este elemento central estará conectado a una serie de dispositivos auxiliares mediante unos buses de comunicación.

Estos buses son:

- Bus de datos: por donde circulan los datos desde el microprocesador a los bloques de memoria, tarjetas periféricas... o viceversa.
- Bus de direcciones: por donde circulan las direcciones de los datos que requieren el microprocesador en sus respectivos bloques de memoria.
- Bus de control: por donde circulan las señales de control (habilitación o inhabilitación de dispositivos) para la comunicación del microprocesador con los dispositivos auxiliares.

Los dispositivos auxiliares, de los que se han hecho mención anteriormente, ayudan al microprocesador en el trabajo de gestionar el sistema y ejecutar el programa. Entre ellos están:

- Memoria del procesador: Memoria que utiliza el procesador a la hora de manejar sus variables internas.
- Memoria de usuario: Memoria donde se almacena el programa de automatización.
- Memoria de entradas/salidas (E/S): Memoria donde se guardan temporalmente los estados de las entradas y salidas del autómata.
- Memoria de forzado de E/S: Permite la modificación por el usuario de entradas o salidas mientras se ejecuta el programa.

- Interface paralelo de E/S: Interface de entradas y salidas
- Interface de comunicación: Interface de comunicación del autómata con otros equipos (unidad programadora, PC, ...).

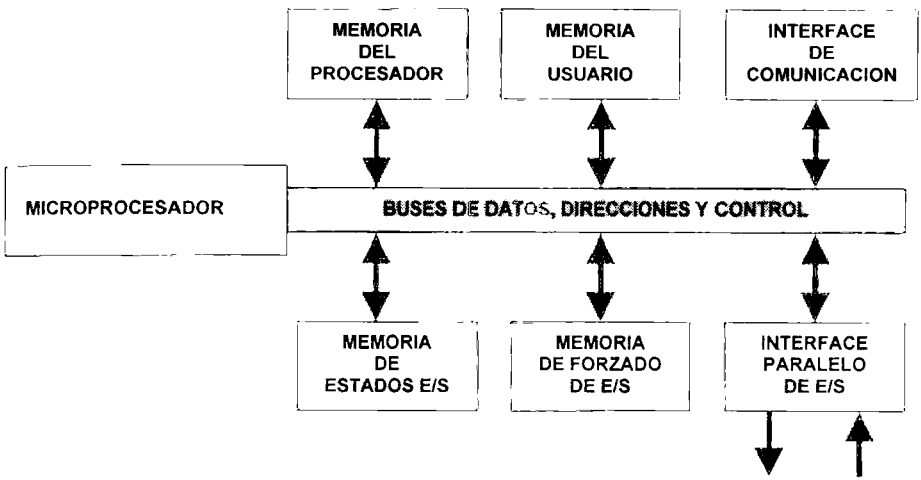


Figura 1.4: Componentes de autómata programable.

5. ESTRUCTURA TÍPICA DE UN CICLO DE EJECUCIÓN

En el apartado 2 de este capítulo se vio cómo el funcionamiento de un autómata programable era cíclico, se ejecuta el mismo programa de manera ininterrumpida. Al llegar a la última instrucción, el programa vuelve a ejecutarse desde el principio. A cada una de las veces que se ejecuta el programa se le llama ciclo de ejecución.

En un ciclo de ejecución no solo se realiza la ejecución del programa contenido en la memoria del autómata, sino que se realizan una serie de procesos antes y después de la ejecución de dicho programa. Básicamente un ciclo de ejecución de un autómata programable está formado por los siguientes elementos:

- Lectura de entradas.- Es el proceso correspondiente a la lectura por parte del autómata del estado de las entradas físicas del mismo y la realización de una copia de las mismas en la memoria de entradas. Este proceso se realiza siempre al inicio de un ciclo de ejecución y antes de

iniciarse la ejecución del programa. A partir de aquí y durante la ejecución del programa, el autómata considerará el valor de una entrada en concreto como el que dicha entrada tenga asignada en la memoria de entradas, valor que no se modificará en todo el resto del ciclo de ejecución. De esta manera, un cambio en el estado de una entrada determinada y que se produzca una vez finalizado el proceso de lectura de entradas no se considerará hasta que se realice un nuevo proceso de lectura de entradas, o sea, al inicio del siguiente ciclo de ejecución.

No obstante, existen autómatas en los que se permite la lectura de una entrada física durante la ejecución del programa.

- Lectura y ejecución del programa.- Una vez finalizado el proceso de lectura de entradas, se inicia la lectura y ejecución del programa de usuario contenido en memoria, comenzando por la primera instrucción. El programa se ejecuta instrucción a instrucción de manera lineal, tomando los valores de entradas que necesite de la memoria de entradas y escribiendo en la memoria de salida los valores que toman las salidas según el desarrollo del programa. Hay que tener en cuenta que en este periodo las salidas que el programa active o desactive no se ven reflejadas aún en la salida física, sino solo en la memoria de salida.

  Al igual que ocurre con las entradas, existen autómatas en los que la activación, o desactivación, de la salida física se puede producir en el momento en el que se active, o desactive, la salida de programa.

- Activación de salidas.- En este intervalo del ciclo de ejecución se activan físicamente las salidas según su estado en la memoria de salidas. Se realiza una copia de la memoria de salida en el dispositivo periférico de salida.

Este ciclo básico de ejecución: lectura de entradas - programa - activación de salidas, se ejecuta cíclicamente en el funcionamiento del autómata. No obstante, este ciclo puede ser interrumpido mediante la activación de una señal externa (error, alarma) o cada cierto tiempo definido por el usuario (según tipo de autómata). Al producirse la interrupción se ejecuta el código asociado a dicha interrupción y una vez finalizado éste, se vuelve al punto del ciclo de ejecución en el que se produjo la interrupción.

Además de los ciclos de ejecución e interrupciones, el autómata ejecuta antes de comenzar con los citados ciclos, un ciclo de inicialización y puesta a punto en el que se ponen a '0' las salidas, se realizan tests internos, etc. Este ciclo inicial solo se realiza una vez al inicio de la ejecución.

El ciclo básico descrito en este apartado puede verse ampliado en determinados autómatas con otras tareas. Entre estas nuevas tareas se encuentra por ejemplo, en los autómatas de nueva generación, la gestión de las comunicaciones con otros dispositivos conectados en una red.

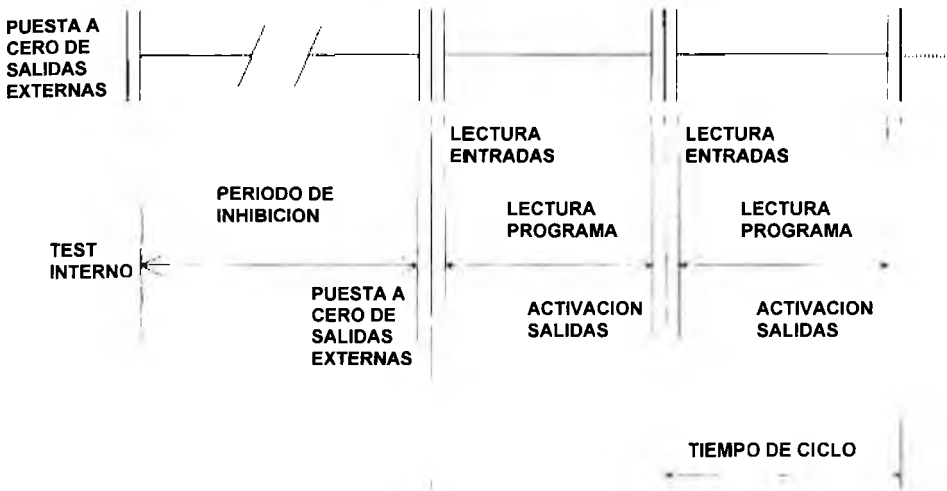


Figura 1.5: Ciclos de ejecución.

6. TERMINALES DE COMUNICACIÓN AUTÓMATA-PROCESO

El autómata se comunica con el proceso que automatiza (recibe información, entradas, y actúa sobre el mismo, salidas) a través de una serie de terminales eléctricos. Estos terminales pueden ser digitales, analógicos y de alarma y contador.

Estos terminales están situados normalmente en unos conectores dispuestos en el panel frontal del autómata, integrados en el mismo o en tarjetas periféricas añadidas al autómata.

Básicamente existen dos tipos de entradas y salidas: digitales y analógicas.

6.1. TERMINALES DIGITALES

Estas entradas/salidas pueden tomar únicamente dos valores de tensión: un valor bajo (0 V) y otro valor alto igual a la tensión de alimentación del dispositivo de entrada/salida (normalmente 24 V). Por lo tanto una entrada se considerará activada cuando la tensión de su borne correspondiente con respecto a un borne de referencia sea 24 V, y se considerará desactivada en caso contrario.

Igualmente, para las salidas, se considerará una salida activada cuando se presente 24 V entre ese terminal y uno de referencia, y desactivada cuando aparezcan 0 V.

De forma general, las entradas binarias (0 o 1) se suelen agrupar formando estructuras de palabras de 8 bits (byte), o de 16 bits (word). En algunos autómatas de la gama alta se pueden agrupar incluso en estructuras de 32 bits.

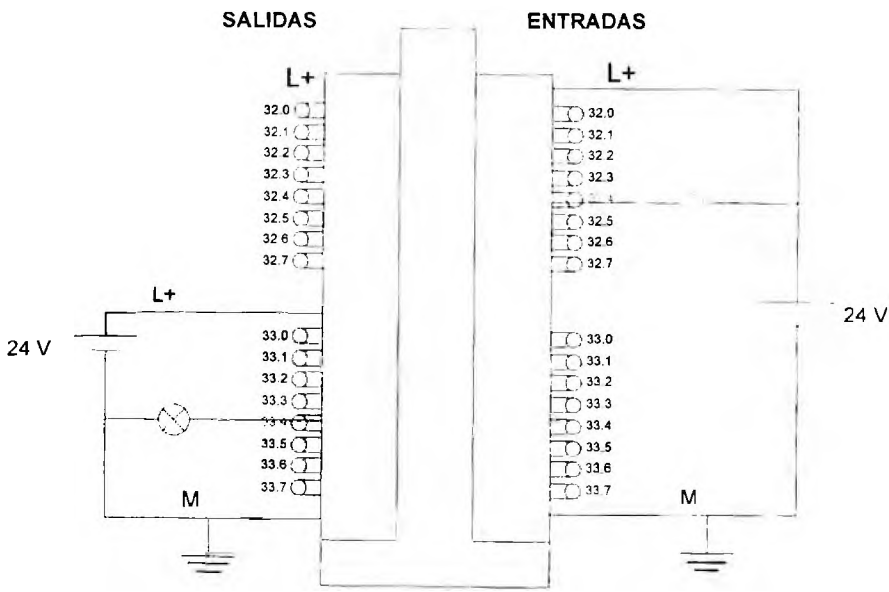


Figura 1.6: Entradas y salidas digitales.

6.2. TERMINALES ANALÓGICOS

Estas entradas/salidas pueden tomar cualquier valor comprendido en un intervalo cuyo límites dependerá del tipo de autómata. Igualmente, dependerá del tipo de autómata el número de entradas y salidas de este tipo de que se dispongan. Asimismo la salida analógica puede ser una salida de "corriente" o de "tensión".

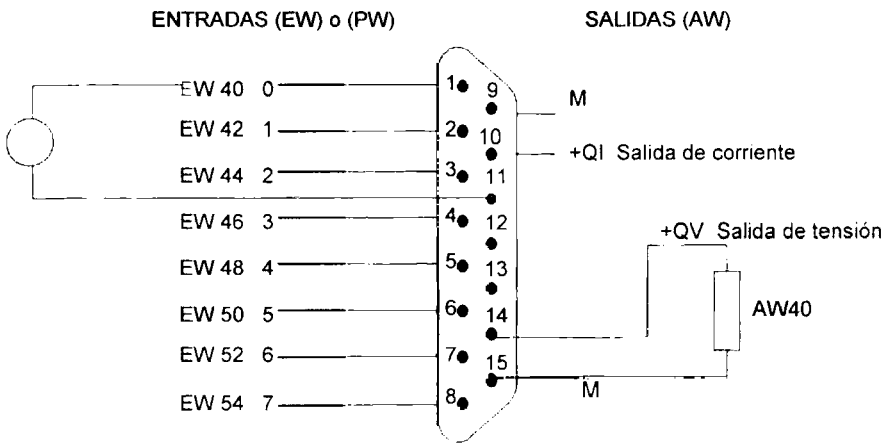


Figura 1.7: Terminales analógicos.

6.3. TERMINALES DE ALARMA Y CÓMPUTO

Este último tipo de terminales tiene una función específica que viene relacionada con operaciones debidas a alarmas (interrupción del programa principal) y contaje (computo). Por lo tanto estos terminales serán siempre entradas que en el caso de las alarmas serán digitales y en caso del contador serán también digitales pero gestionadas en el interior autómata como señales analógicas.

De esta manera las entradas de alarmas se considerarán entradas digitales normales, pero en el caso de la activación de las mismas, el programa principal se verá interrumpido para pasar a ejecutarse, solo una vez, un código asociado a dicha alarma y volver a continuación al programa principal.

La entrada de contador se utilizará en aplicaciones en las que el tiempo entre acciones es importante, o en los que se precise contar eventos que sucedan a frecuencia elevada.

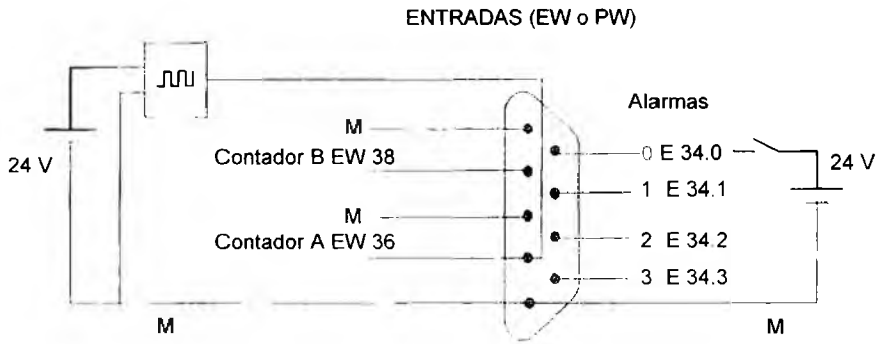


Figura 1.8: Terminales de alarma y contador.

7. UNIDADES DE PROGRAMACIÓN

El primer paso en la automatización de un proceso es el estudio del mismo y el desarrollo y escritura del programa que realice la automatización deseada. Este programa se debe depositar en la memoria que dispone el autómata para tal efecto, pero ya en forma binaria, o sea, que cada instrucción estará formada por ceros y unos.

Para facilitar la tarea de escritura, codificación y almacenamiento del programa en la memoria del autómata surgen las unidades de programación. Por lo tanto el usuario deberá escribir sus programas en estas unidades, por lo que dispondrán de teclado y pantalla para tal efecto, y una vez conectada dicha unidad al autómata mediante la interface oportuno, la unidad se encargará de codificarlo y traspasarlo al autómata

La funcionalidad y potencia de estas unidades de programación varían según el tipo de la misma: desde las unidades más simples (programadoras de mano, tipo calculadora) hasta las más complejas (tipo PC con entorno gráfico).

Además de la tarea de la escritura del programa, estas unidades de programación pueden disponer de herramientas de configuración, mantenimiento y puesta a punto del autómata y la depuración y optimización del programa de automatización.

Capítulo 2

El Autómata Programable S5-95U

Contenido

El sistema de automatización Simatic S5 de Siemens
Características del S5-95U
El lenguaje de programación Step5
Elementos de servicio e indicadores
Funcionamiento del autómata. Estructura interna

1. EL SISTEMA DE AUTOMATIZACIÓN SIMATIC S5 DE SIEMENS

La firma Siemens dispone de una amplia gama en equipos para sistemas de automatización designados con el nombre genérico de SIMATIC S5. Este sistema de automatización está constituido por:

- autómata programable (o PLC)
- aparatos de programación
- lenguaje de programación STEP5.

Los autómatas programables del sistema SIMATIC S5 se designan con la letra U y cubren, según los modelos, todas las posibilidades de automatización. Del mismo modo se presentan en dos ejecuciones mecánicas: tipo bloque para la gama baja y tipo modular para la gama alta.

Los distintos autómatas disponibles en la familia SIMATIC S5 son los siguientes:

S5-90U	gama baja.
S5-95U	gama baja-media.
S5-100U	gama baja-media.
S5-115U	gama media.
S5-135U	gama media.
S5-155U	gama alta.

De todos los autómatas anteriores nos centraremos en el S5-95U por ser este autómata el utilizado en la confección y puesta en práctica de las aplicaciones realizadas en este libro. No obstante estas aplicaciones se podrían implementar en los otros autómatas de la familia al aceptar todos ellos el lenguaje de programación STEP5 de Siemens.

2. CARACTERÍSTICAS DEL S5-95U

El SIMATIC S5-95U es un autómata de la gama baja-media tipo bloque, de aspecto compacto y reducido tamaño, pero capaz de resolver las tareas de automatización más complejas.

Se presenta dentro de una robusta carcasa de plástico que contiene el procesador, la fuente de alimentación y las entradas y salidas integradas. Esta carcasa puede fijarse sobre un perfil normalizado.

A continuación se detallan las características técnicas principales del citado autómata:

Fuente de alimentación:

Tensión de entrada 24 V, corriente continua, con una tolerancia admisible de 20... 30 V.
Tensión de salida de +9 V para la periferia integrada y +5,2 V para la unidad de programación con unas corrientes de salida respectivamente de $\leq$ 1 A y $\leq$ 0,65 A. Presenta además protección contra cortocircuito de tipo electrónica con clase de protección 1 y sin separación galvánica.

Datos técnicos internos:

Dispone de una memoria interna RAM de 8 K palabras de las cuales 4 K son para instrucciones.
Tiempo por operación binaria de aproximadamente 2 μs.
2048 marcas, de las que 512 son remanentes.
128 temporizadores con un margen de contaje de 0,01 a 9990 s.
128 contadores, de los que 8 son remanentes, con un margen de 0 a 999 y con la posibilidad de incrementar o decrementar.

Periferia integrada:

El autómata S5-95U presenta la periferia encargada de las entradas y salidas tanto digitales como analógicas, integradas dentro de su carcasa.

Entradas digitales:

Dispone de 16 entradas digitales con separación galvánica.
Tensión nominal de entrada 24 V:
Valor '0' tensión de –30 V... +5 V c.c. y corriente < 1,5 mA (a 30 V).
Valor '1' tensión de +13 V... +30 V c.c. y corriente <6,5 mA (a 30 V).

Salidas digitales:

Dispone de 16 salidas digitales con separación galvánica.
Tensión nominal de carga L+ 24 V c.c. con un margen admisible de 20... 30 V c.c. con el rizado inclusive.
Corriente de salida con señal '1' máximo de 0.5 A, a una temperatura de 60ºC.
Corriente residual con señal '0' $\leq$ 50 μA.

Tensión de salida:
Con señal '0' máximo 0,5 V con una resistencia de carga de 6 kΩ.
Con señal '1' máximo L+ -0,6 V con 0,5 A.
Protección contra cortocircuitos electrónica.
Corriente suma de 6 A ó 8 A para una temperatura ≤ 50°C.
Posibilidad de conectar las salidas en paralelo dando cada salida una corriente de 0,5 A.
Longitud máximo del cable sin apantallar de 100 m.

Entradas analógicas:

Entradas sin separación galvánica.
Valores nominales del margen de entrada 0.. +10 V
Tensión de entrada admisible –10 V... +30 V.
Resistencia de entrada 20 kΩ.
Representación digital de la señal de 11 bits.
Principio de media: aproximaciones sucesivas.
Tiempo de conversión 40 µs.
Régimen transitorio interno de 1,1 ms.
Con señalización de desbordamiento de margen.
Límite total del error en el margen de temperatura de 0... 60°C. 1,68%
Longitud máxima de cable apantallado de 100 m.

Salidas analógicas:

Salidas sin separación galvánica.
Representación digital de la señal de 11 bits.

Salida de tensión:
Valor nominal del margen de salida de 0... 10 V.
Resistencia de carga ≥ 2,5 kΩ.
Tiempo de conversión incluido, transitorio (valor típico) de 20 µs.
Con protección contra cortocircuito.
Intensidad máxima de cortocircuito de 30 mA.
Límite total del error (0.. 60°C) 1%.
Longitud máxima de cable apantallado de 100 m.

Salida de corriente:
Valor nominal del margen de salida: 0...20 mA.
Resistencia de carga < 300 Ω.
Tiempo máximo de conversión 3,0 µs.
Límite total del error (0... 60°C.) 1,1 %.
Longitud de cable apantallado de 100 m.

Entradas de alarma:

Entradas sin separación galvánica.
Las tensiones e intensidades de entrada son las mismas que para las entradas digitales.
Tiempo de retardo:
De '0' a '1' tiempo de 75 μs.
De '1' a '0' tiempo de 140 μs.
Ancho de impulso ≥ 500 μs.
Longitud máxima de cable apantallado de 100 m.

Entradas de contador:

Entradas sin separación galvánica.
Las tensiones e intensidades de entrada son las mismas que para entradas digitales.
Tiempo de retardo:
De '0' a '1' tiempo de 10 μs.
De '1' a '0' tiempo de 15 μs.
Frecuencia de contaje:
Para el contador A frecuencia de 5 kHz.
Para el contador B frecuencia de 2 kHz.
Ancho de impulso > 100 μs.
Longitud máxima de cable apantallado de 100 m.

Además de todos los dispositivos integrados en la carcasa, el S5-95U puede ampliarse con hasta 32 módulos periféricos del S5-100U

Los módulos periféricos disponibles son:

- Módulos de entrada y salida digitales.
- Módulos de entrada y salida analógicas.
- Módulos especiales para funciones de temporización externas, contaje rápido y comparación de valores analógicos.
- Módulos de diagnosis para vigilancia del bus periférico del S5-90U.
- Módulos de simulación para prueba de programas.

Estos módulos de ampliación se montan sobre el mismo carril normalizado en el que se monta el autómata y se conectan a este mediante un conector especial situado a la derecha del autómata.

3. EL LENGUAJE DE PROGRAMACIÓN STEP-5

El lenguaje de programación STEP5 es el utilizado para la programación de todos los autómatas de la familia SIMATIC S5.

Este lenguaje de programación hace posible la programación tanto de funciones sencillas (por ejemplo: Función Y) como de las más complejas (por ejemplo: Retardo a la conexión).

El STEP5 contiene todas las instrucciones necesarias para el desarrollo del programa de automatización. Los programas realizados mediante este lenguaje se pueden estructurar utilizando los "módulos". Existen distintos tipos de módulos que se utilizarán dependiendo de la misión del mismo. Así, hay módulos de datos (DB) para incluir datos a utilizar, módulos funcionales (FB) para incluir funciones que se repitan con regularidad, etc. Los módulos se pueden llamar unos a otros, así los módulos de un primer nivel podrán llamar a los de un nivel superior y así sucesivamente hasta un máximo de 8 niveles.

Mediante la estructuración del programa se consiguen las ventajas:

- Clarificar la evolución del programa.
- Modificar programas en poco tiempo.
- Comprobar el correcto funcionamiento de un programa por partes.
- Estandarizar partes del programa.

Del mismo modo el lenguaje STEP5 presenta distintos modos de representación. De esta manera el usuario podrá utilizar aquella representación que le sea más familiar o que domine mejor

Estas representaciones son:

- Lista de instrucciones (AWL).
- Plano de contactos (KOP).
- Plano de funciones (FUP).

En este apartado se ha visto, muy por encima, las generalidades del lenguaje STEP5. En el correspondiente capítulo se verán con más profundidad todas las características y potencialidad de dicho lenguaje y su aplicación en la escritura de programas.

A continuación se relaciona un listado de las operaciones básicas del lenguaje STEP 5 y una breve descripción de las mismas.

Operaciones combinacionales (lógicas):

U Combinación Y: Consulta al estado de señal '1'
UN Combinación Y negada: Consulta al estado de señal '0'
O Combinación O: Consulta al estado de señal '1'
ON Combinación O negada: Consulta al estado de señal '0'
U(Combinación Y de expresiones entre paréntesis
O(Combinación O de expresiones entre paréntesis
) Cerrar paréntesis

Operaciones de memoria (biestables):

S Poner el operando a '1' (activar el operando)
R Poner el operando a '0' (borrar el operando)
= Asignar al operando el valor del VKE (resultado de la operación anterior)

Operaciones de carga:

L Carga el dato indicado por el operando en el acumulador 1

Operaciones de transferencia:

T Transfiere el contenido del acumulador 1 a la variable que indique el operando

Operaciones de tiempo:

SI Arranca como impulso una temporización
SV Arranca como impulso prolongado una temporización
SE Arranca como retardo a la conexión una temporización
SS Arranca como retardo a la conexión memorizada una temporización
SA Arranca como retardo a la conexión una temporización
R Reponer (borrar) una temporización

Operaciones de contaje:

ZV Contaje hacia delante en 1
ZR Contaje hacia atrás en 1
S Activar (ajustar) un contador
R Borrar (reponer) un contador

Funciones aritméticas:

+F Sumar dos números en coma fija
-F Restar dos números en coma fija
!=F Comparar dos números en coma fija respecto a igualdad
><F Comparar dos números en coma fija respecto a desigualdad
>F Comparar dos números en coma fija respecto a superioridad
>=F Comparar dos números en coma fija respecto a superioridad o igualdad
<F Comparar dos números en coma fija respecto a inferioridad
<=F Comparar dos números en coma fija respecto a inferioridad o igualdad

Operaciones de llamada de módulo:

SPA Salto absoluto (incondicional) a un módulo de programa, funcional o de paso, según operando
SPB Salto condicional a un módulo de programa, funcional o de paso, según operando
A llamada a un módulo de datos
E Crear o borrar un módulo de datos

Operaciones de retorno:

BE Terminar módulo (fin de módulo)
BEB Terminar módulo de forma condicional
BEA Terminar módulo de forma absoluta

4. ELEMENTOS DE SERVICIO E INDICADORES

El autómata programable S5-95U presenta un panel frontal con una serie de elementos necesarios para la implementación del sistema de automatización.

En el presente apartado se describirán cuáles son estos elementos y cuál su finalidad. Según la figura 2.1.:

1. Compartimento de batería.- En este compartimento se puede introducir una batería de protección para salvaguardar la memoria del autómata en caso de una interrupción en la alimentación.

2. Conector frontal.-
   - Entradas digitales (E32.0...A33.7).- 16 entradas digitales designadas mediante la expresión E-.-
   - Salidas digitales (A32.0...A33.7).- 16 salidas digitales designadas mediante la expresión A-.-
3. Indicador de fallo de batería.- Este indicador se encenderá cuando se haya agotado o no exista la batería de protección.
4. Interruptor CON/DES.- Interruptor mediante el cual se conecta o se desconecta la alimentación del autómata.
5. LEDs indicadores de entradas y salidas digitales.- Led asignado a cada una de las 16 entradas y 16 salidas que se encienden en caso de estar activada su correspondiente entrada o salida.
6. Bornes de conexión de la alimentación.- Se alimentará el autómata mediante 24 V c.c. entre L+ y M conectando el tercer borne a tierra.
7. Conector para acoplar módulos S5-100U.- Conector situado en el lateral derecho del autómata y en el que se conectarán los posibles módulos de ampliación S5-100U
8. Indicador de modo.- Está formado por dos leds. El superior se iluminará de color verde cuando el autómata este en modo RUN (ejecución de un programa). El led inferior se iluminará de color rojo cuando el autómata se encuentre en STOP (detenido).
9. Conector hembra para:
   - Entradas analógicas.- 8 entradas analógicas (EW40, EW42,..., EW50).
   - Salidas analógicas.- 1 salida analógica (AW40) pudiendo ser de corriente o tensión.
10. Selector de modo.- Interruptor con tres posiciones: la posición central sitúa al autómata en modo de STOP (detenido) y no se ejecuta el programa. La posición superior coloca al autómata en modo RUN (ejecución de un programa). La tercera posición, COPY, no es estable, o sea, al colocar el interruptor en dicha posición y soltarlo, vuelve a la posición STOP. Se utiliza para copiar el programa contenido en la EPROM en la memoria de programa del autómata.

11 Receptáculo para cartucho de memoria.- Espacio destinado para la introducción de una cartucho de memoria E(E)PROM con la finalidad de contener algún programa para su posterior copia y ejecución en el autómata.

12. Conector hembra para programación o conexión a red local.- Conector para conectar un aparato de programación (PG), un aparato de operación (OP) o red local SINEC L1.
13. Conector hembra para:
    - Entradas de alarma.- 4 entradas (E34.0,..., E34.3).
    - Entradas de contador.- 2 entradas (EW36, EW38).

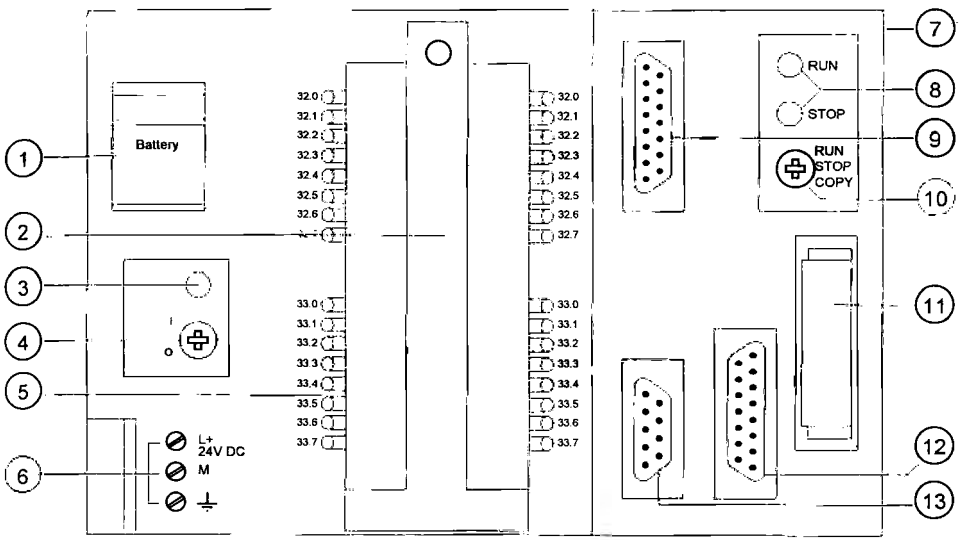


Figura 2.1: El Autómata S5-95U de Siemens

5. FUNCIONAMIENTO DEL AUTÓMATA. ESTRUCTURA INTERNA

En el interior de la carcasa del autómata programable se encuentran todos los elementos necesarios para el funcionamiento del mismo. Entre estos elementos cabe diferenciar aquellos elementos encargados de gestionar los datos y programas, o sea, la unidad central de procesos (CPU), que forma el autómata en sí, y aquellos otros elementos encargados de la comunicación del autómata (o CPU) con el exterior, o sea, la periferia integrada de entradas y salidas tanto digitales como analógicas, y como su nombre indica se considera como una periferia, es exterior al autómata, pero esta integrada, en este caso, en el mismo.

Así bien, cada uno de estos dos grandes bloques que se encuentran en el interior de la carcasa (CPU o Autómata y periferia integrada) esta constituida por los siguientes elementos:

Elementos de la CPU:

- Unidad de control (microprocesador).- Es el elemento fundamental en el funcionamiento del autómata, bajo su mando esta todo el control y gestión del sistema.
- ALU (Unidad Aritmético-Lógica).- Elemento que ayuda al microprocesador en tareas de operaciones aritméticas y operaciones lógicas.
- Memoria ROM.- Memoria de solo lectura donde se encuentra el sistema operativo necesario para el funcionamiento del sistema.
- Memoria de programa interna (RAM).- Memoria en la que se deposita el programa de automatización y que se ejecutará.
- PAE.- Imagen de proceso de las entradas. Memoria RAM en la que se realiza una copia del estado de las entradas del autómata al inicio de cada ciclo de ejecución.
- PAA.- Imagen de proceso de las salidas. Memoria en la que se escriben el estado de las salidas durante el ciclo de ejecución y que posteriormente se trasladan a las salidas físicas del autómata.
- Memoria de temporizadores, contadores y marcas.- Memoria RAM en la que memorizan variables tales como temporizadores, contadores y marcas utilizadas en el programa de usuario.

Elementos de la Periferia Integrada:

- Entradas integradas digitales.
- Entradas integradas analógicas.
- Entradas de alarmas.
- Entradas de contadores.
- Salidas integradas digitales.
- Salidas integradas analógicas.

Además de los elementos anteriores, fundamentales para el funcionamiento del autómata, existen otra serie de elementos, todos ellos periféricos, que aumentan la potencialidad del autómata. Entre estos elementos los más básicos son:

- Cartucho de memoria (EPROM/EEPROM).- Permite introducir un programa de usuario en el autómata y memorizarlo en la memoria de usuario sin necesidad de utilizar una unidad de programación.
- Unidad de programación, PC o red local.- Elementos conectables al canal serie.
- Módulos de entradas (digitales/analógicas).

- Módulos de salidas (digitales/analógicas).
- Módulos funcionales.- Módulos especiales que incorporan funciones complejas de las que en un principio no dispone el autómata en su versión básica. Entre estos módulos pueden estar: módulos de control (PID...), módulos de tratamiento de señales analógicas, módulos operacionales.

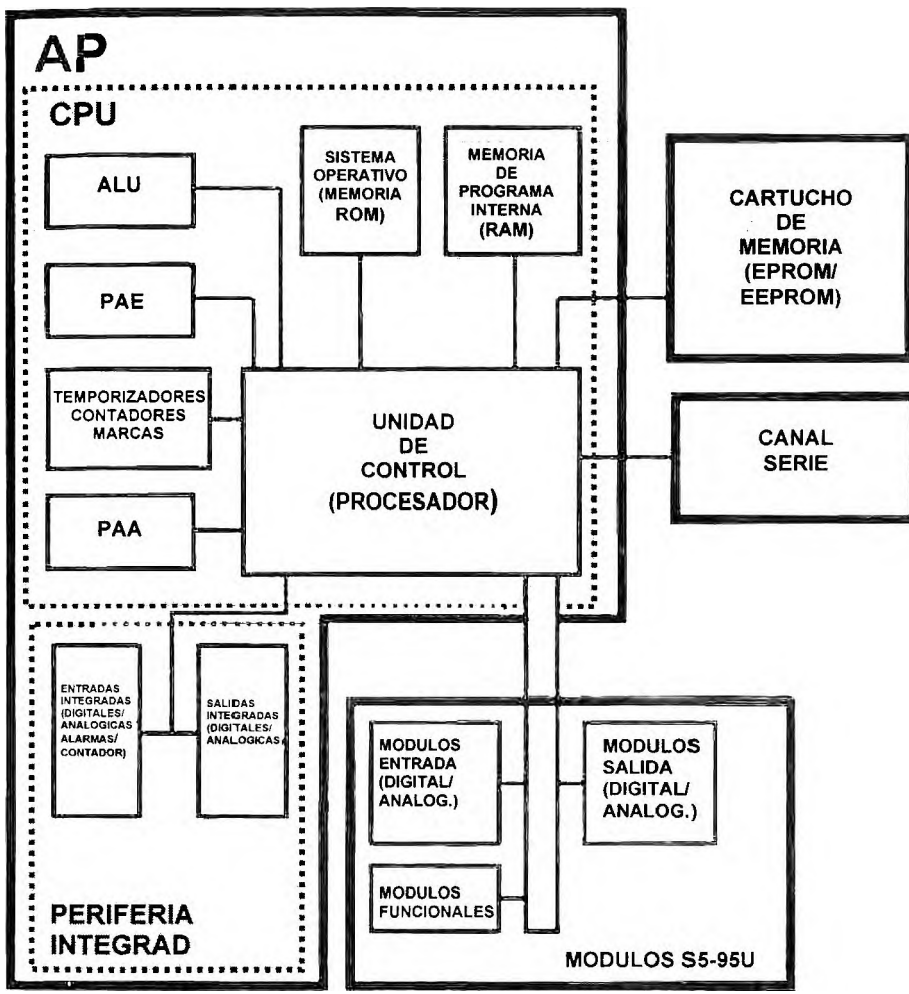


Figura 2.2: Estructura interna del S5-95U.

Capítulo 3

Redes de Petri

Contenido

1. INTRODUCCIÓN

Conocer y comprender el entorno, analizar y estudiar todas las situaciones que puedan acaecer a su alrededor, constituye una constante inquietud y necesidad para el hombre. Ese impulso investigador necesita disponer para su efectiva realización de un esquema teórico o modelo, generalmente en forma matemática o gráfica, representativo de un sistema o de una realidad compleja y destinado a facilitar tanto su comprensión como el estudio de su comportamiento.

En función del objetivo final a alcanzar, se pueden construir dos tipos de modelos:

- *Modelo estructural*: es aquel donde se enumeran las partes del sistema y sus interconexiones.

- *Modelo funcional*: en él se describe cómo opera y se desenvuelve el sistema, presentando un aspecto dinámico

En general, un sistema cualquiera se puede desglosar en dos partes bien diferenciadas, descritas a continuación y representadas en la figura 3.1:

- *Parte operativa* (PO), es la parte ejecutiva, la que acepta físicamente los mensajes del exterior (entradas) y la que realiza las acciones con el medio que le rodea (salidas).

- *Parte de control* (PC), es aquella parte inteligente, a ella están asociadas:

  - *Informes*, suministrados por la parte operativa y muestran el estado del medio externo.

  - *Consignas*, información que especifica las acciones a realizar según el estado de nuestro sistema y del medio.

  - *Ordenes*, mandato hacia la parte operativa para que ejecute las salidas pertinentes.

  - *Salidas*, permiten al observador externo comprobar el funcionamiento del sistema y no requiere un gasto apreciable de energía.

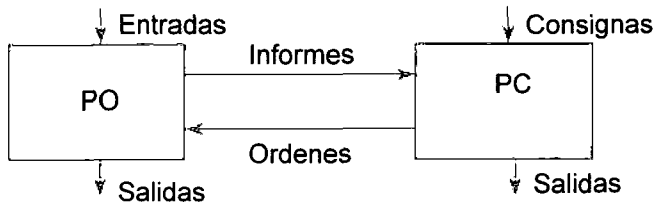


Figura 3.1: Desglose de un sistema

Las redes de Petri se presentan como una útil herramienta matemática aplicable al modelado de sistemas, que admite una valiosa representación gráfica, considerándose ésta una potente extensión de los grafos de estado.

Las finalidades que persiguen las redes de Petri son:

- Expresar de forma clara y rigurosa:
  - *Objetivos de funcionamiento*, mostrando hacia dónde se dirige nuestro sistema modelado.
  - *Seguridad funcional*, donde se especifica sin ambigüedad bajo qué circunstancias el sistema evoluciona de un estado a otro.
- Seguimiento de la evolución del sistema, identificando fácilmente su estado actual en la descripción gráfica.
- Flexibilidad en su modificación sin dañar sustancialmente el modelo original.
- Comprobación de que el modelo no es erróneo en un primer análisis.

2. DEFINICIÓN DE AUTÓMATA FINITO

Se define un autómata como aquel sistema discreto tal que:

- Recibe un número finito de símbolos (entradas -**E**-).
- Emite un número finito de símbolos (salidas -**S**-).
- Posee estados estables fácilmente identificables (estados -**Q**-).

Se dice que el autómata es finito si el número de estados es finito.

Estos tres aspectos que definen el autómata están íntimamente ligados entre sí mediante las siguientes funciones:

- La función de transición δ, según la cual el estado actual depende del estado anterior y de los símbolos de entrada recibidos:

$$\delta: QxE \rightarrow Q$$

- La función de salida λ, expone que la salida actual depende del estado anterior y de los símbolos de entrada recibidos:

$$\lambda: QxE \rightarrow S$$

En resumen, se puede afirmar que un autómata finito es la quíntupla **(E, S, Q,** λ**,** δ**)**. Los conjuntos **E** y **S** identifican el sistema; el conjunto **Q** los estados posibles y las funciones λ y δ permiten definir el comportamiento del autómata.

Se denomina estado inicial $\mathbf{Q_0}$ a aquel estado estable en el que se encuentra el autómata cuando aún no ha recibido ningún símbolo de entrada.

3. DIFERENTES REPRESENTACIONES EN LA MODELACIÓN DE SISTEMAS

Bajo este título se describe la evolución que han sufrido las diferentes formas de representar los sistemas. Se explicará la forma de realizarlos y las ventajas e inconvenientes derivados de ellos. De sus inconvenientes surgirá la necesidad de cambio y el salto hacia un nuevo peldaño en la evolución.

Las diferentes formas de representación pueden ser desglosadas en dos grandes grupos, que se corresponden cronológicamente con su evolución:

- *Representación tabular*, consistente en la creación de una tabla bidimensional. En ella aparecerán las entradas (**E**), las salidas (**S**) y los

estados (**Q**) de nuestro sistema, debidamente asociados con las funciones, ya comentadas, de salida y de transición.

- Tabla de estados de Mealy.
- Tabla de estados de Moore.
- Tabla de fases.

- *Representación gráfica*, contiene la misma información que la representación tabular; no obstante se aprecia una considerable mejora en su aspecto visual, utilizando elementos gráficos, que facilitan la identificación del estado actual del sistema y del seguimiento de su posible evolución posterior.
  - Diagrama de fases.
  - Grafo de estados.
  - Redes de Petri.

Como puede observarse, las Redes de Petri, objeto de estudio de este capítulo, ocupan el último lugar en la evolución y es hacia donde se pretende llegar de manera progresiva y razonada.

3.1. REPRESENTACIONES TABULARES

3.1.1. TABLA DE ESTADOS DE MEALY

Para la mejor comprensión de este tipo de representaciones, su estudio se llevará a cabo mediante el análisis detallado de un ejemplo.

La nomenclatura utilizada en el sistema es la siguiente:

- E=(a,b) Conjunto de entradas de nuestro sistema con dos elementos a y b.
- S=(x,y) Conjunto de salidas de nuestro sistema con dos elementos x e y.
- Q Estado estable de nuestro sistema.

- Q_a Estado que recuerda que la última entrada ha sido 'a'.
- Q_{ab} Estado que recuerda que la última entrada ha sido 'b' y la penúltima entrada ha sido 'a'.

La tabla de estados de Mealy posee la forma de la tabla 3.1:

Estado anterior	Estado actual		δ	Salida actual		λ
	a		b	a		b
Q_a	Q_a		Q_{ab}	y		y
Q_{ab}	Q_a		Q_{bb}	x		y
Q_{bb}	Q_a		Q_{bb}	y		y

Tabla 3.1: Tabla de estados de Mealy

La primera columna representa el 'estado anterior', donde se enumeran los únicos posibles estados estables.

La segunda columna representa el 'estado actual', gobernado por la función de transición δ, que precisa de los estados anteriores (primera columna) y de las entradas posibles (primera fila). En la intersección de estas filas y columnas se encuentra el nuevo estado hacia el que evoluciona el sistema, que debe ser uno de los posibles.

La tercera columna representa la 'salida actual', gobernada por la función de salida λ que precisa de los estados anteriores (primera columna) y de las entradas posibles (primera fila). En la intersección de estas filas y columnas se encuentra la nueva salida hacia la que evoluciona el sistema.

De esta forma el sistema queda perfectamente definido estructuralmente con sus conjuntos de entradas, salidas y estados; y funcionalmente con las funciones de transición y de salida.

3.1.2. TABLA DE ESTADOS DE MOORE

Se continuará con la misma estructura de presentación para la tabla de estados de Moore.

La nomenclatura utilizada es la siguiente:

- E=(a,b) Conjunto de entradas del sistema con dos elementos a y b.
- S=(x,y) Conjunto de salidas del sistema con dos elementos x e y.
- Q Estado estable de nuestro sistema.
- Q_a Estado que recuerda que la última entrada ha sido 'a'.
- Q_{aba} Estado que recuerda que la última entrada ha sido 'a', la penúltima 'b' y la antepenúltima 'a'.

La tabla de estados de Moore posee la forma de la tabla 3.2:

Estado anterior	Estado actual δ		Salida λ
	a	b	
Q_a	Q_a	Q_{ab}	y
Q_{ab}	Q_{aba}	Q_{bb}	y
Q_{aba}	Q_a	Q_{ab}	x
Q_{bb}	Q_a	Q_{bb}	y

Tabla 3.2: Tabla de estados de Moore.

En un primer acercamiento puede parecer que la tabla de Moore se comporta de igual manera que la tabla de Mealy; no obstante, esto no es real, puesto que una observación detenida y atenta nos permite apreciar determinadas características singulares que facilitan la descripción del sistema.

Ambas tablas han sido elaboradas partiendo del mismo hipotético sistema. Seguidamente serán enumeradas y estudiadas las diferencias visuales:

- La primera columna contiene cuatro estados estables posibles.

- La segunda columna, aunque nos muestra una función de transición exactamente igual que la anterior, presenta sin embargo un nuevo estado de partida. Ésta es la única distinción apreciable entre ellas.

- La última columna refleja la situación más destacada, merecedora de un mayor comentario. La función de salida no depende de las entradas del sistema. Se aprecia que a cada estado estable le corresponde una salida, independientemente de la entrada acaecida. Este hecho simplifica enormemente la función de salida y explica la aparición de nuevos estados estables respecto a la tabla de Mealy.

La ventaja de eliminar una de las funciones que definen el sistema compensa sobremanera la necesidad de aumentar el número de estados estables. Hemos mejorado, por tanto, el método anterior.

Dada la gran similitud entre estas dos representaciones, se puede pasar de un autómata de Mealy a uno de Moore sin gran dificultad

3.1.3. TABLA DE FASES

Para realizar esta nueva forma de representación se utiliza como punto de partida la tabla de Moore. Sobre esta base, con el fin de obtener una mayor adaptabilidad a los procesos reales, se llevarán a cabo leves modificaciones que serán expuestas tras la descripción del ejemplo.

La nomenclatura utilizada es la siguiente:

- $E=(x_1,x_2)$ Conjunto de entradas del sistema. Como evento se permitirá una combinación entre las distintas entradas.

- $S=(s_1,s_2)$ Conjunto de salidas del sistema, autorizándose una combinación de ellas

- $Q=(q_1,q_2,q_3)$ Conjunto de estados estables del sistema.

Para la construcción de la tabla nos decantamos por una simbología binaria positiva, donde el '1' indica dos posibilidades: la entrada correspondiente se ha producido o bien debe darse la salida asociada. El '0' por su parte refleja que la entrada no se ha dado o que la salida no debe ejecutarse.

Una representación de este tipo de tabla es la siguiente:

Estados estables	Evolución de los estados δ [entradas (x_1,x_2)]				Salidas (s_1,s_2)
	00	01	11	10	
q_1	q_1	q_2	--	q_3	10
q_2	q_1	q_2	q_3	--	01
q_3	q_1	--	q_3	q_3	00

Tabla 3.3: Tabla de fases.

En la tabla se observa:

- Una combinación de salidas asociada a cada estado estable.
- La función de transición δ toma como origen una combinación de las entradas.
- Hay situaciones que jamás se podrán dar (--): el sistema se encuentra en el estado q_1 y se produce la entrada '11'.

El sistema así descrito permite una mayor adaptación a los sistemas reales, aceptando combinaciones entre las entradas y las salidas.

Una vez propuestas las representaciones tabulares es conveniente realizar un ejemplo que permita asentar los conocimientos adquiridos. Posteriormente, un breve análisis crítico nos aclarará las causas que justifican la necesidad de obtener un nuevo y más efectivo modelo.

Incluidas en la categoría de las representaciones gráficas se encuentra el diagrama de fases, conceptuado como la transcripción directa de la tabla de fases. Dada la íntima vinculación entre ambas, seguidamente abordaremos su estudio.

3.1.4. DIAGRAMA DE FASES

La nomenclatura utilizada en la representación gráfica del sistema es la siguiente:

- Cada estado estable está representado por una circunferencia.

- Las transiciones se identifican con arcos orientados y etiquetados con las entradas que han provocado su evolución.

- Junto a cada circunferencia (estado estable) se señala la salida asociada.

En la figura 3.2 aparece el diagrama de fases correspondiente a la tabla de fases del apartado anterior. La figura precisa de poco comentario si se ha asimilado el concepto de la tabla de fases y la nueva nomenclatura gráfica. Sólo destacar unos aspectos:

- La transición que posee la combinación de entradas (**1-**) aporta la siguiente información: la transición tiene lugar cuando la primera entrada se produce, con independencia de lo que con la segunda ocurra.

- El estado $\mathbf{q_3}$ no tiene asociado ninguna salida, puesto que carece de etiquetas.

- La etiqueta $\mathbf{S_1}$ que acompaña al estado $\mathbf{q_1}$, nos informa que llegado a esta posición se ejecutará la primera salida.

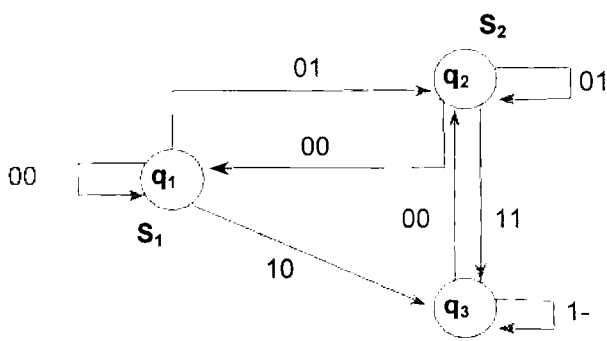


Figura 3.2: Diagrama de fases.

La información aportada por la nueva representación es exactamente igual que la vista en forma tabular, pero suministra un mayor dinamismo y es visualmente más agradable.

3.1.5. EJEMPLO

Para ilustrar el esquema teórico arriba enunciado se propone un sistema y su funcionamiento, realizándose el modelo siguiendo la descripción expuesta con la tabla de fases.

Sea un carro motorizado sobre un carril tal como se muestra en la figura 3.3, se poseen dos finales de carrera (entradas del sistema) uno en el extremo izquierdo **A** y otro en el extremo derecho **B** Las salidas del sistema actúan sobre el sentido de giro del motor, hacia la izquierda **i** y hacia la derecha **d**. El sistema posee además un pulsador de puesta en marcha **M**, que inicia la secuencia.

El funcionamiento que se desea del sistema es el siguiente:

- Si el carro se encuentra en A y pulso M, el carro se desplaza hacia la derecha.
- Si el carro se encuentra en B, el carro se desplaza hacia la izquierda.
- Si el carro se encuentra en A y M está sin pulsar, el carro permanece en reposo.

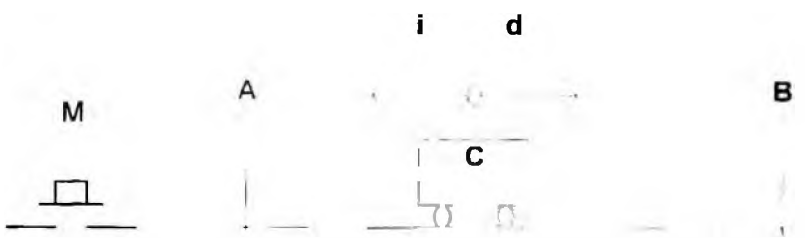


Figura 3.3: Ejemplo.

- *Modelo del sistema:*

Para una mejor comprensión se generará el modelo paso a paso.

a) *Identificaremos las entradas, las salidas y los posibles estados.*

- Entradas: E=(M A B)

- Salidas: S=(d i)
- Estados posibles: $Q=(q_1, q_2, q_3, q_4, q_5, q_6, q_7, q_8, q_9)$
  - q_1: Carro en A, M sin pulsar. Carro en reposo.
  - q_2: Carro en A, M pulsado.
  - q_3: Carro en A, M sin pulsar. Carro en movimiento.
  - q_4: Carro en trayecto hacia la derecha, M pulsado.
  - q_5: Carro en trayecto hacia la derecha, M sin pulsar
  - q_6: Carro en B, M pulsado.
  - q_7: Carro en B, M sin pulsar.
  - q_8: Carro en trayecto hacia la izquierda, M pulsado.
  - q_9: Carro en trayecto hacia la izquierda, M sin pulsar.

b) Situar sobre la tabla de fases las entradas, salidas y estados estables.

Dependiendo de las definiciones de cada estado y de las salidas asociadas se construye la primera fase de la tabla 3.4. Cada estado estable se situará en una fila diferente que coincidirá con la posición de la primera columna.

Estados estables	Evolución de estados δ Entradas (M A B)								Salidas (d,i)
	000	001	011	010	110	111	101	100	
q_1				q_1					00
q_2					q_2				10
q_3				q_3					10
q_4								q_4	10
q_5	q_5								10
q_6							q_6		01
q_7		q_7							01
q_8								q_8	01
q_9	q_9								01

Tabla 3.4: Tabla de fases con los estados estables.

Sobre la tabla podemos apreciar la existencia de dos recuadros sombreados, elegidos al azar para un detallado comentario del contenido de la tabla:

- Estado estable q_1.
  - Entradas M=0, A=1, B=0, que corresponden con el estado definido.
  - Salidas d=0 y i=0, pues según el estado el carro está en reposo.
- Estado estable q_6.
  - Entradas M=1, A=0, B=1, que corresponden con el estado definido.
  - Salidas d=0 y i=1, el carro debe iniciar su vuelta hacia la izquierda.

c) Posibles evoluciones de los estados ante las entradas en cada fila.

Sobre la tabla 3.5 se ha sombreado el estado estable de cada fila; a partir de él y según la combinación de entradas, el sistema evoluciona hacia el siguiente estado. Se deben tener presentes todas las evoluciones probables. Las entradas que no podrían darse jamás se han marcado con el signo '-'.

Estados estables	Evolución de estados δ Entradas (M A B)								Salidas (d,i)
	000	001	011	010	110	111	101	100	
q_1	-	-	-	q_1	q_2	-	-	-	00
q_2	-	-	-	q_3	q_2	-	-	q_4	10
q_3	q_5	-	-	q_3	q_2	-	-	-	10
q_4	q_5	-	-	-	-	-	q_6	q_4	10
q_5	q_5	q_7	-	-	-	-	-	q_4	10
q_6	-	q_7	-	-	-	-	q_6	q_8	01
q_7	q_9	q_7	-	-	-	-	q_6	-	01
q_8	q_9	-	-	-	q_2	-	-	q_8	01
q_9	q_9	-	-	q_1	-	-	-	q_8	01

Tabla 3.5: Posible evolución de los estados estables.

d) Reducción de la tabla utilizando la teoría de estados compatibles.

La tabla anterior se puede reducir utilizando la teoría de estados compatibles. El resultado obtenido es el siguiente:

Estados estables	Evolución de estados δ Entradas (M A B)								Salidas (d,i)
	000	001	011	010	110	111	101	100	
α	-	-	-	q_1	q_2	-	-	-	00
β	q_5	q_7	-	q_3	q_2	-	q_6	q_4	10
γ	q_9	q_7	-	q_1	q_2	-	q_6	q_8	01

Tabla 3.6: Tabla de fases reducida.

Los estados posibles definidos se han mantenido sombreados y los estados estables se han reducido sólo a tres, con sus correspondientes salidas.

Ya se ha conseguido la tabla de fases y el problema queda definitivamente descrito. A continuación pasaremos a ilustrar directamente su representación gráfica mediante el diagrama de fases.

e) Diagrama de fases de la tabla obtenida.

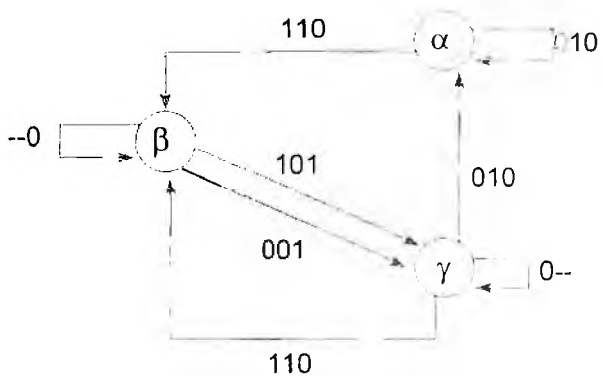


Figura 3.4: Diagrama de fases.

3.1.6. CRÍTICA A LOS MÉTODOS TABULARES DESCRITOS

Una vez ejemplificada la representación de sistemas por medio de métodos tabulares, llega el momento de la reflexión:

- La descripción es exhaustiva, reflejando todos los datos sobre posibles evoluciones del sistema. En ciertos casos podría considerarse como una ventaja.

- Descripción muy extensa si se aumenta las variables de entrada, lo que dificultaría su tratamiento para problemas de mayor complejidad.

- Se tienen en cuenta combinaciones de entradas sin significado físico.

En resumen se podría decir que para un problema tan sencillo como el analizado, el tiempo empleado ha sido apreciable y las variables utilizadas excesivas. Se hace necesario recurrir a otro método que ahorre energías y sea más concreto a la hora de analizar el problema.

3.2. GRAFO DE ESTADOS

El grafo de estados es una representación gráfica en la que la condición lógica que provoca una transición entre dos estados es cualquier función lógica de las entradas.

Esta definición no establece en un principio diferencias con el diagrama de fases ya visto, pero sí las plantea en la forma de generar el grafo.

El razonamiento del cual se parte para establecer la descripción es el siguiente:

- El origen lo situamos en el estado de reposo; una situación lógica inicial en la que no se ha recibido ninguna entrada.

- Sólo se consideran aquellos eventos que son importantes para la evolución del sistema, el resto de combinaciones de entradas posibles no se tienen en cuenta.

- No se muestran estados sin sentido físico en la evolución del sistema.

Resulta necesario recordar el ejemplo planteado anteriormente y realizar su representación gráfica utilizando las nuevas reglas. La nomenclatura utilizada es la misma que la vista en el diagrama de fases:

a) El punto de partida lo constituye el estado inicial de reposo donde el carro se encuentra en la posición A y sin movimiento.

b) El segundo paso es cuestionar ¿cuál es el evento posible que obligará al sistema a cambiar su estado? Evidentemente es imperativo el accionamiento del pulsador M; el resto de las entradas no revisten importancia y por lo tanto no son tenidas en cuenta.

c) Una vez descubierta la condición externa que obliga a un cambio de estado, se dibuja un arco orientado con la entrada asociada y el nuevo estado hacia el que se dirige. Junto a este último se sitúa la salida correspondiente. El carro se desplaza hacia la derecha, salida apreciada en nuestro ejemplo.

d) El único evento que tiene importancia en esta situación será que el carro alcance la posición B, el resto carece de consideración alguna, haciéndole corresponder la nueva transición. El nuevo estado hacia el que evoluciona es claro: el carro se desplaza hacia la izquierda, previa activación de la salida.

e) Por último el carro llega al punto de partida A y se mantiene en reposo hasta que vuelva a iniciarse el ciclo.

El grafo de estados obtenido es el correspondiente a la figura 3.5

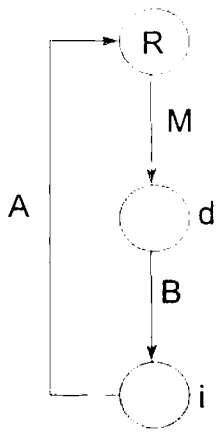


Figura 3.5: Grafo de estados del ejemplo descrito.

Como se puede observar, la simplificación de la representación es enorme. El grafo es tan reducido y sencillo que no merece la pena buscar nuevos métodos.

El paso siguiente es complicar el problema y ver cómo responde el método que se acaba de emplear.

3.2.1. EJEMPLO

Sean dos carros motorizados **C1** y **C2** sobre carriles independientes, como muestra la figura 3.6. Se poseen finales de carrera en cada carril, dos en el extremo izquierdo **A1** y **A2** y dos en el extremo derecho **B1** y **B2**, según figura. Las salidas del sistema actúan sobre el sentido de giro del motor, hacia la izquierda **i1** e **i2** y hacia la derecha **d1** y **d2**. El sistema posee además un pulsador de puesta en marcha **M**, que inicia la secuencia.

El funcionamiento que se desea del sistema es el siguiente:

- Si el carro C1 se encuentra en A1, el C2 en A2 y se pulsa M, los dos carros se desplazarán hacia la derecha.
- Si el carro C1 se encuentra en B1 y el C2 en B2, ambos carros se desplazarán hacia la izquierda.

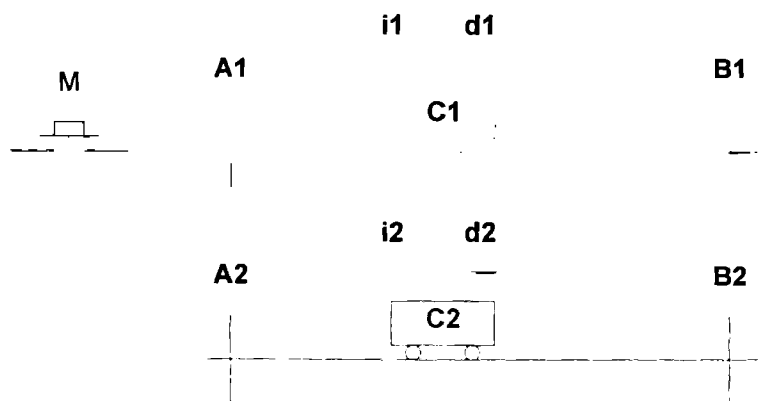


Figura 3.6: Desplazamiento simultáneo de dos carros.

La descripción del sistema según el grafo de estados se razona de la siguiente forma:

- Se parte del reposo con los dos carros en la posición *A*.
- El suceso esperado es que se pulse *M* y el sistema evoluciona hacia un nuevo estado que activa las salidas *d* de los carros.
- Los carros no circulan a la misma velocidad y la distancia a recorrer no es igual, lo que implica que un carro llegará antes que otro.
  - Si llega el *C1* (se activa el final de carrera *B1*), se debe parar el carro y mantener al segundo desplazándose hacia la derecha hasta que llegue al final (se activa el final de carrera *B2*).
  - Si llega el *C2* (se activa el final de carrera *B2*), se debe parar el carro y mantener al primero desplazándose hacia la derecha hasta que llegue al final (se activa el final de carrera *B1*).

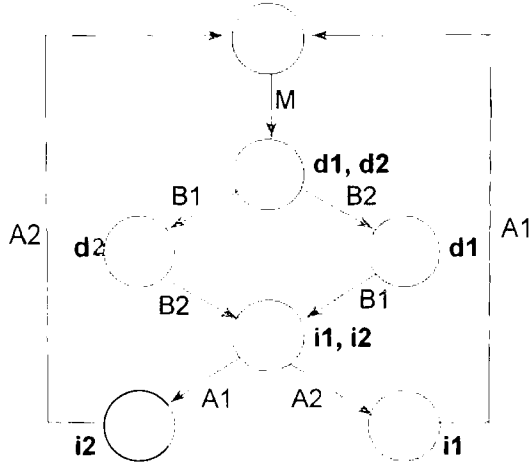


Figura 3.7: Grafo de estados del desplazamiento simultáneo de dos carros.

- Una vez que ambos carros han llegado al final, son desplazados hacia la izquierda. Pueden ocurrir entonces dos nuevas situaciones.
  - Si llega el *C1* (se activa el final de carrera *A1*), detenemos su marcha y mantenemos el segundo desplazándose hacia la izquierda hasta que llegue al punto de partida (se activa el final de carrera *A2*).

- Si llega el *C2* (se activa el final de carrera *A2*), detenemos también su macha y mantenemos el primero desplazándose hacia la izquierda hasta que llegue al origen (se activa el final de carrera *A1*).

- Una vez que ambos carros han llegado al estado inicial se los mantiene en reposo hasta que vuelva a activarse el pulsador *M*.

El grafo de estados obtenido es el de la figura 3.7.

3.2.2. CRÍTICA AL GRAFO DE ESTADOS

El último ejemplo expuesto incide en las enormes ventajas que implica la utilización del grafo de estados, ya que sigue aportando una información mínima aunque suficiente para hacer un análisis completo del comportamiento del sistema.

He aquí algunas precisiones que ya se intuyen en el desarrollo del ejemplo:

- Una ligera modificación del enunciado supone rehacer casi en su totalidad la descripción del sistema, haciendo el método poco flexible. Esta cuestión supone hoy día un gran inconveniente, puesto que los sistemas se hallan en continuo cambio y volver a diseñar todo el modelo implicaría un enorme coste.

- Cuando el sistema plantea evoluciones paralelas (simultáneas), como es el caso que nos ocupa, ya que se poseen dos carros que evolucionan físicamente separados, se observan problemas futuros que se pueden sintetizar en dos puntos:

  - Conduce a descripciones complejas, necesitándose gran número de estados para representar las interacciones lógicas entre las evoluciones paralelas. En el caso expuesto hay únicamente dos elementos que evolucionan simultáneamente y el grafo ha empezado a alejarse de la sencillez deseada. Intente el mismo problema con tres carros evolucionando simultáneamente y observe qué ocurre.

  - El modelo no permite una descripción descendente del sistema. Este problema surge en el ejemplo anterior si se desea añadir una acción secundaria a uno de los carros la cual no atañe al otro. La representación mediante el grafo de estados sí recoge dicha modificación y obliga a que todos los elementos se vean afectados.

Expuestos los problemas, se hace necesario modificar la representación para solventar estas deficiencias. Es este momento donde las redes de Petri cobran su importancia.

4. REDES DE PETRI

Este nuevo concepto del grafo aparece para solucionar los problemas anteriormente expuestos: evoluciones simultáneas, descripciones descendentes y flexibilidad. Las Redes de Petri se definen como un grafo orientado en el que intervienen dos tipos de nudos:

- *Lugares* (p): representados por circunferencias, expresan un estado estable y asociado a él se encuentra una salida del sistema.

- *Transiciones* (t): representadas por segmentos rectilíneos, expresan la espera del cumplimiento de una entrada del sistema o combinación de ellas.

Un ejemplo gráfico de representación de sistemas mediante Redes de Petri es el que se muestra en la figura 3.8:

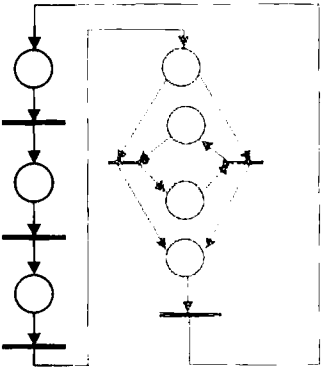


Figura 3.8: Ejemplo gráfico de una red de Petri.

A continuación se describe la nomenclatura utilizada. Es algo más amplia y compleja que las anteriores, pero se recomienda familiarizarse con ella para no tener dificultad en seguir el resto del capítulo.

- *Arco*: segmento orientado que une un lugar con una transición y viceversa.
- *Marca*: punto en el interior de un lugar.
- *Marcado*: conjunto de marcas en un instante determinado en la Red de Petri, que determina el estado total del sistema.
- *Acciones o salidas*: salida del sistema asociada a cada lugar
- *Condición externa*: función combinatoria de las variables de entrada asociada a una transición.
- *Evento o acontecimiento*: cambio de estado lógico de una condición externa.
- *Lugar (p) de entrada de una transición (t)*: lugar que tiene un arco orientado de p hacia t.
- *Lugar (p) de salida de una transición (t)*: lugar que tiene un arco orientado de t hacia p.
- *Transición sensibilizada*: transición que posee todos los lugares de entrada marcados.
- *Nudos O*: lugar que tiene varios arcos de entrada y/o de salida.
- *Nudos Y*: transición que tiene varios arcos de entrada y/o de salida. Son los que permiten la creación y extinción de evoluciones simultáneas.

Una vez nominado cada elemento del gráfico y su significado, las reglas de utilización de este tipo de grafos son las que seguidamente se detallan:

- *Disparo de una transición*: se produce cuando una transición está sensibilizada y el evento o condición externa que está asociado a la transición se verifica.
- *Evolución del marcado*: redistribución de las marcas al producirse el disparo de una transición:
  - Se elimina una marca de cada uno de los lugares de entrada.
  - Se añade una marca a cada uno de los lugares de salida.

He aquí algunos ejemplos para ilustrar las reglas de utilización:

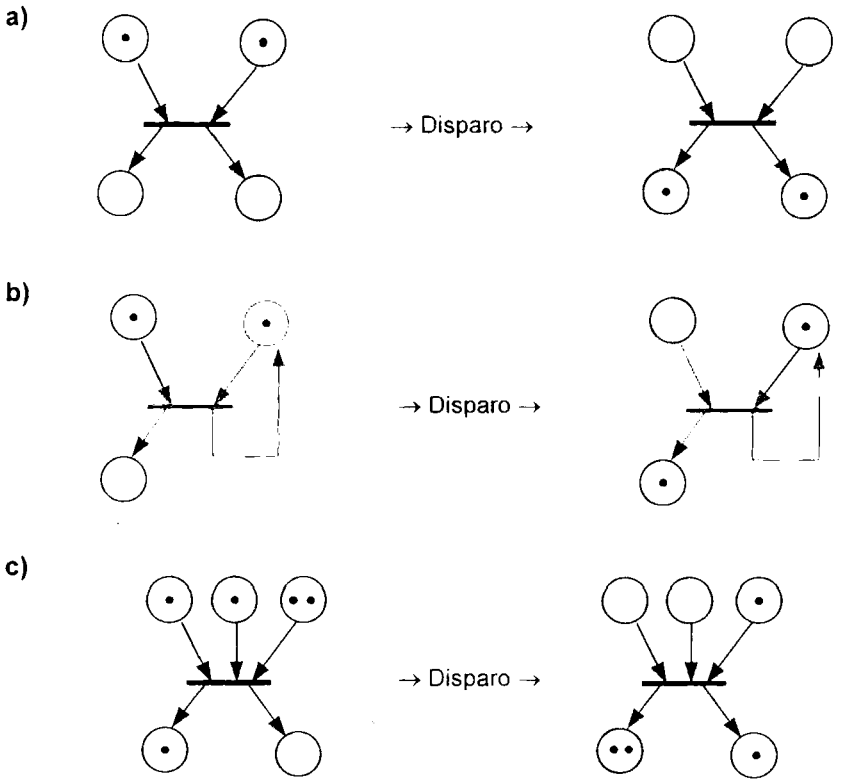


Figura 3 9: Ejemplos de evolución en las transiciones

4.1. EJEMPLO

Representaremos mediante Redes de Petri el ejemplo anterior de la evolución simultánea de los dos carros.

La filosofía de razonamiento es análoga a la empleada para el *grafo de estados*, sin embargo la introducción de las *marcas* y de los "*nudos Y*" la convierte en una herramienta mucho más útil. En la figura 3.10 aparece la Red de Petri del problema planteado:

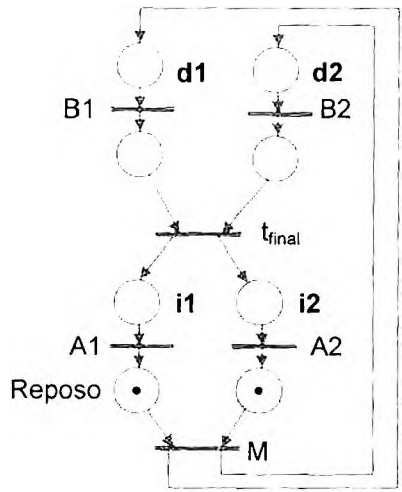


Figura 3.10: Red de Petri de la evolución simultánea de los dos carros.

En el análisis de la nueva representación cabe resaltar los siguientes aspectos:

- La construcción de la red comienza con el sistema en estado de reposo, que viene indicado en la figura 3.10 mediante dos lugares señalizados con un punto (marcado inicial). Se utilizan dos lugares porque hay dos elementos evolucionando simultáneamente y se desea su independencia en la representación. Las marcas indican dónde se encuentra actualmente el sistema.

- El siguiente suceso esperado es la activación del pulsador *M*. Este evento resulta obligado a ambos carros y para ello se utiliza un *nudo Y* al que acuden los dos lugares de reposo. A partir de este nudo Y (transición) divergen dos arcos y los carros vuelven a evolucionar independientemente, accionando las salidas *d1* y *d2*.

- Cada carro continúa hasta los lugares de entrada de la transición t_{final}, estos lugares no tienen asociada ninguna salida, por lo que al llegar la marca a esta situación el carro se detiene. Cuando las dos marcas llegan a estos lugares la transición se sensibiliza y dispara instantáneamente, puesto que dicha transición no tiene asociada ninguna condición externa.

- Una vez superada la transición t_{final} vuelven a salir dos arcos y los carros continúan su evolución independiente, accionando las salidas *i1* e *i2*.

- De nuevo cada carro continúa hasta el lugar de entrada de la transición con la condición externa *M*. Estos lugares no tienen asociada ninguna salida, deduciéndose de ello que la marca se detiene al llegar a esa situación. Cuando las dos marcas llegan a estos lugares la transición se sensibiliza y se queda a la espera del cumplimiento del evento asociado.

4.2. VENTAJAS DE LAS REDES DE PETRI

Concretando las ventajas de la nueva representación se tiene:

- El estado actual del sistema está representado por el marcado en dicho instante, donde la posición de las marcas en los estados definen qué está ocurriendo en el momento de la observación.

- Se adapta mejor a sistemas con evoluciones simultáneas, o también llamados sistemas concurrentes, incorporando descripciones menos complejas. En efecto, hay que resaltar que en el ejemplo anterior el número de elementos gráficos utilizados es mayor, en comparación con el grafo de estados, no obstante la comprensión de su funcionamiento es más sencilla. Cuanto mayor sea el número de procesos simultáneos en el sistema, tanto más resulta conveniente la utilización de las Redes de Petri.

- Facilita la descripción de secuencias alternadas. Este concepto se desarrollará más adelante en un ejemplo.

- Facilita los modelos de sistemas con exclusión mutua. Este concepto será igualmente explicado en un ejemplo posterior.

- El estado total es un conjunto de subestados, esto demuestra la facilidad para realizar modificaciones locales. En el ejemplo que nos ocupa, las acciones que realice un carro en su recorrido no afecta al otro, pudiéndose insertar nuevos estados sin necesidad de replantear el problema.

- Permite comprobar propiedades de buen funcionamiento, existiendo técnicas que detectan errores en las Redes de Petri sin necesidad de conocer el sistema que representa. Esta característica se desarrollará al final del presente capítulo.

4.3. EJEMPLO: SISTEMA CON EXCLUSIÓN MUTUA

Sea un almacén con capacidad máxima para 4 elementos, dos unidades externas, una de producción y otra de consumo. La unidad de producción elabora el producto y lo deposita en el almacén. Por su parte, la unidad de consumo extrae el elemento del almacén y le aplica el acabado superficial.

Se impone como restricción al sistema planteado, la imposibilidad de acceso simultáneo al almacén de las dos unidades externas (exclusión mutua).

El esquema que relaciona los diferentes bloques es el siguiente:

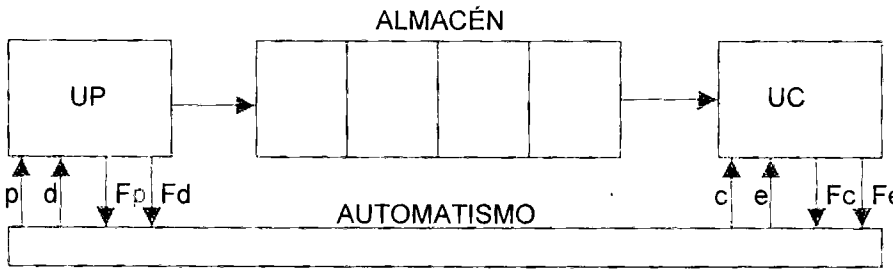


Figura 3.11: Ejemplo de sistema con exclusión mútua.

La nomenclatura utilizada en el esquema de la figura 3.11 es la siguiente:

- UP: unidad de producción.
- UC: unidad de consumo.
- p: orden de producir un elemento.
- d: orden de depositar un elemento en el almacén.
- e: orden de extraer un objeto del almacén.
- c: orden de consumir un elemento.
- Fp, Fd, Fc, Fe: sensores que indican: fin de producción, de depósito, de consumo y de extracción.

Los respectivos conjuntos de entradas y salidas del automatismo que se pretende realizar son:

- E=[Fp, Fd, Fc, Fe].

- S=[p, d, e, c].

A continuación se expondrá la representación gráfica del sistema mediante Redes de Petri, la evolución hacia cada estado así como el significado físico de cada uno de ellos.

A. Modelo del sistema con exclusión mutua descrito con el marcado inicial.

Situación actual:

- Almacén en reposo (Ar).
- Cuatro objetos en almacén (O).
- Consumidor en espera (Ec).
- Objeto en producción (p).

- Transición de A→B
  - T2 sensibilizada y sin condición externa
  - Fp se cumple.

B Nuevo estado estable.

Situación actual:

- Productor en espera (Ep).
- Objeto en extracción (e).
- Almacén con 3 elementos (O).

- Transición de B→C
  - Fe se cumple.

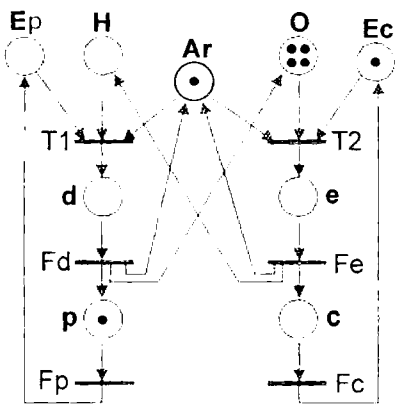


Figura 3.12: Estado A del ejemplo con exclusión mutua.

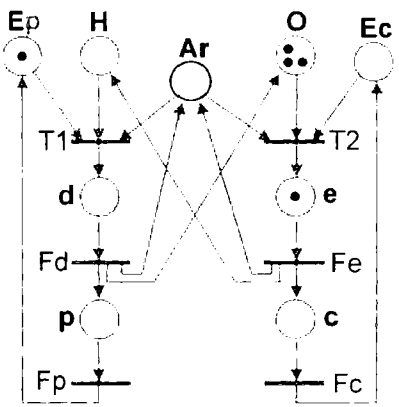


Figura 3.13: Estado B del ejemplo con exclusión mutua.

C. Nuevo estado estable.

Situación actual:

- Elemento consumiéndose (c).
- Productor en espera (Ep).
- Almacén con 3 elementos (O).
- Almacén en reposo (Ar).
- Almacén con un hueco disponible (H)

- Transición de C→D
  - Fc se cumple.
  - T1 sensibilizada y sin condición externa.

D Nuevo estado estable.

Situación actual:

- Elemento depositándose (d).
- Consumidor en espera (Ec).
- Almacén con 3 elementos (O).

- Transición de D→E
  - Fd se cumple.

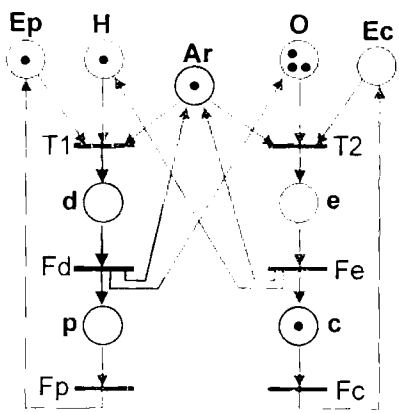


Figura 3.14: Estado C del ejemplo con exclusión mutua.

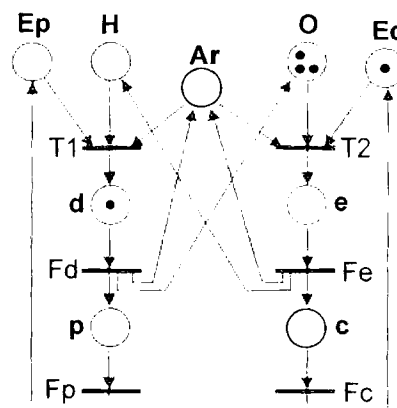


Figura 3.15: Estado D del ejemplo con exclusión mutua.

E. Vuelta al estado inicial.

Situación actual:

- Almacén en reposo (Ar).
- Cuatro objetos en almacén (O).
- Consumidor en espera (Ec).
- Objeto produciéndose (p).

- Transición de E→B
  - T2 sensibilizada y sin condición externa
  - Fp se cumple.

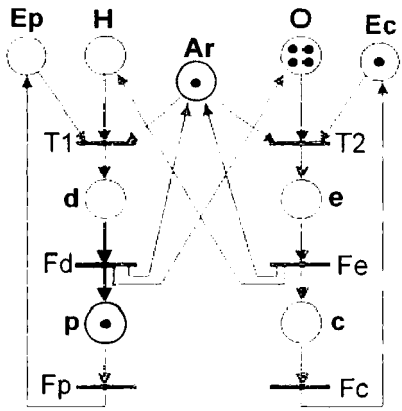


Figura 3.16: Estado E del ejemplo con exclusión mutua.

En el sistema descrito se observa la sutileza con la que se trata la exclusión mutua. Mediante la Red de Petri diseñada se impone que los lugares asociados a la producción y a la extracción nunca podrán estar marcados al mismo tiempo, pues la realización de uno excluye la ejecución del otro.

Esta posibilidad se debe a la existencia conjunta de un lugar con varios arcos de salida y de una transición con varios arcos de entrada.

4.4. EJEMPLO: SISTEMA CON SECUENCIAS ALTERNADAS

Se tiene un producto en la posición inicial (I), una pinza lo transporta hasta una cuba, donde lo sumerge durante un tiempo predeterminado. La misma pinza sin soltar el producto, lo extrae y deposita en la posición final (F).

La pinza vuelve vacía hacia la posición inicial comenzando un nuevo ciclo. La cuba recién utilizada necesita un período de reposo para que se produzca la deposición de las partículas en suspensión. Para no demorar la producción, se añade una nueva cuba que se utilizará mientras la otra se encuentra en reposo. El proceso se esquematiza en la figura 3.17.

El sistema realiza siempre las mismas acciones en cada ciclo, sin embargo debe alternar la utilización de las cubas.

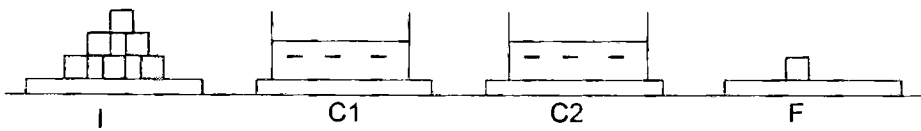


Figura 3.17: Ejemplo de sistema con secuencias alternadas.

La nomenclatura utilizada es la siguiente:

- I: inicio, lugar de carga.
- C1: cuba número 1.
- C2: cuba número 2.
- b, s: operaciones para bajar y subir la pinza.
- c, d: operaciones de carga y descarga.
- ad, ai: avance derecha e izquierda de la pinza.
- Fb, Fs, Fc, Fd: sensores que señalan el final de la bajada, subida, carga y descarga.
- F: final, lugar de descarga.

Los respectivos conjuntos de entradas y salidas del automatismo que se pretende realizar son:

- E=[Fb, Fs, Fc, Fd].
- S=[b, s, c, d, ad, ai].

A continuación se expone la representación gráfica del sistema mediante Redes de Petri así como la evolución hacia el siguiente estado estable.

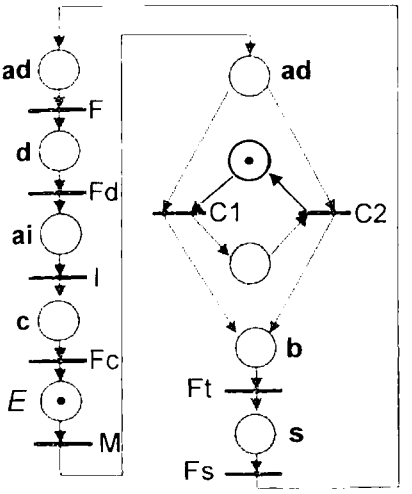


Figura 3.18: Estado A.

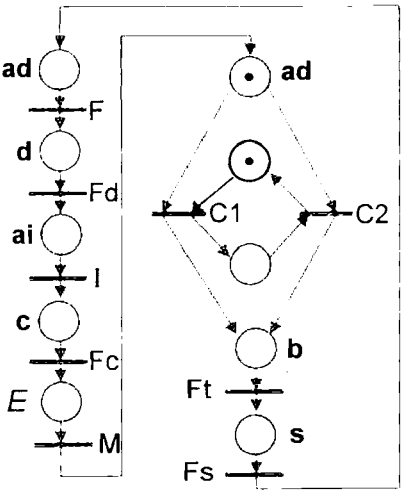


Figura 3.19: Estado B.

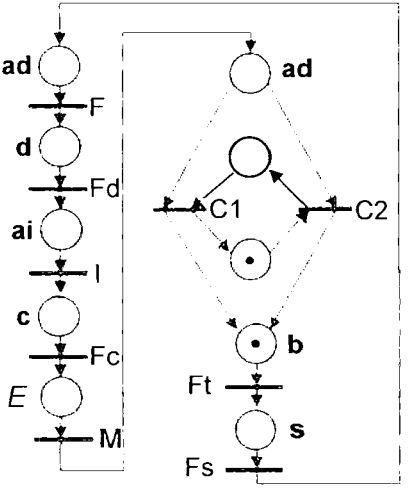


Figura 3.20: Estado C.

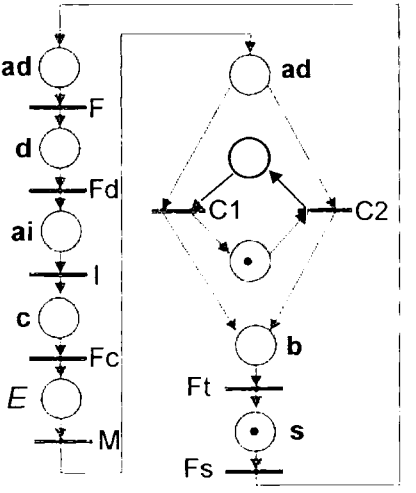


Figura 3.21: Estado D.

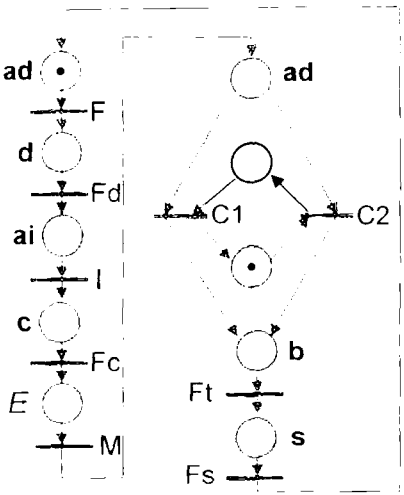


Figura 3.22: Estado E.

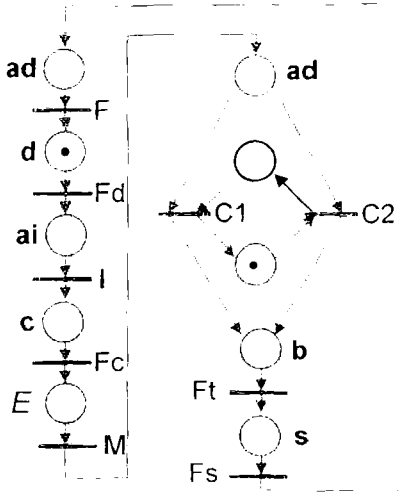


Figura 3.23: Estado F

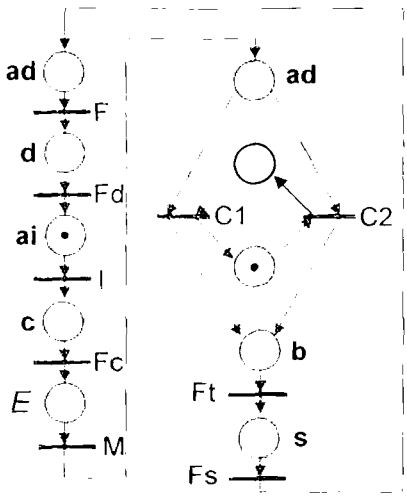


Figura 3.24: Estado G

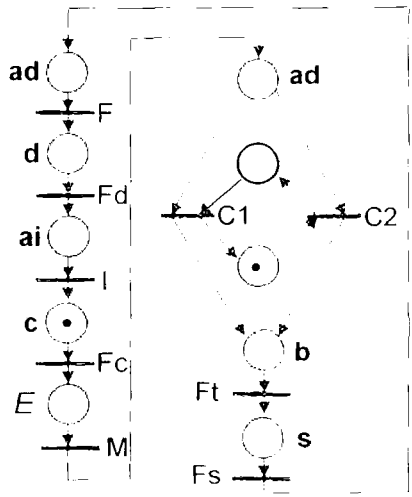


Figura 3.25: Estado H.

La evolución de la red es fácil de seguir y no se comentará, sin embargo hay dos lugares que llaman poderosamente la atención, aquellos que se encuentran entre las transiciones C1 y C2.

En el estado B, la marca más dinámica se encuentra sobre un lugar que presenta arcos hacia dos transiciones, y es la otra marca la que decide la transición sensibilizada. En el caso expuesto se circula a través de la transición C1, utilizándose la cuba número 1. El sistema continúa su evolución, ocupando la marca más sosegada, el lugar inferior en espera de que la otra marca vuelva a aparecer, mas en este caso la transición sensibilizada será la C2, utilizándose la cuba número 2.

En efecto, cabe concluir que un ciclo completo del sistema requiere dos ciclos de la marca más activa y un ciclo de la otra. Los dos movimientos de la marca menos dinámica permiten una Red de Petri más compacta.

5. AMPLIACIÓN DE LAS REDES DE PETRI

La Red de Petri aceptará nuevas propiedades que le facilitarán la representación de sistemas más complejos. A continuación se detallan algunas de esas propiedades más destacadas.

5.1. RED DE PETRI GENERALIZADA

Se define la Red de Petri generalizada como la cuádrupla $R=(P,T,\alpha,\beta)$ donde:

- P: conjunto finito de lugares.
- T: conjunto finito de transiciones.
- $P \cap T = \varnothing$: ambos conjuntos son disjuntos.
- $\alpha: P \times T \rightarrow N$ función de incidencia previa (peso del arco).
- $\beta: T \times P \rightarrow N$ función de incidencia posterior (peso del arco).

Se dice que la transición está sensibilizada, si y solo si cada uno de sus lugares de entrada posee al menos $\alpha(p,t)$ marcas.

Cuando se produce el disparo de una transición, la red evoluciona de la siguiente forma:

- Se eliminan $\alpha(p,t)$ marcas de cada lugar de entrada.

- Se añaden $\beta(t,p)$ marcas a cada lugar de salida.

5 2. RED DE PETRI ORDINARIA

Es la red vista y analizada en los apartados anteriores, constituyendo un caso particular de la Red de Petri generalizada, donde la función de incidencia previa, $\alpha(p,t)$, y posterior, $\beta(t,p)$, tienen valor 1

5.3. UTILIZACIÓN DE LAS FUNCIONES DE INCIDENCIA

En el presente apartado se representarán estas nuevas funciones sobre la red y un ejemplo pondrá de manifiesto su implicación en la evolución.

Las funciones de incidencia, tanto la previa como la posterior, tienen como misión asociar a cada arco un número natural. Si este número es la unidad, el arco se representará con una línea simple, en caso contrario, el arco se dibujará mediante una línea gruesa y junto a él el número asociado a la función de incidencia, tal como se indica en la figura 3.26.

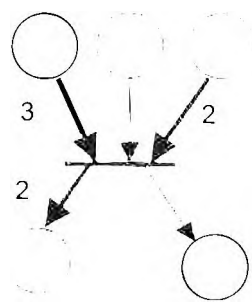


Figura 3 26: Representación gráfica de las funciones de incidencia.

Veamos un ejemplo donde aparece una transición sensibilizada, su disparo y posterior evolución:

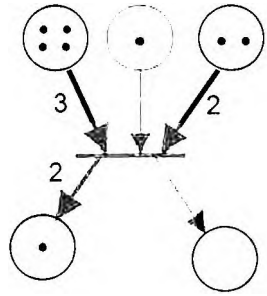


Figura 3.27: Ejemplo de transición sensibilizada con funciones de incidencia.

La transición representada en la figura 3.27 se encuentra sensibilizada porque los lugares de entrada a la transición poseen al menos el número de marcas señaladas en cada arco.

El cumplimiento de la condición externa asociada a la transición, provoca las siguientes acciones:

- Se eliminan de cada lugar de entrada el número de marcas que indica el arco que parte de él, en este caso 3, 1 y 2 respectivamente.

- Se añaden a cada lugar de salida el número de marcas que indica el arco que llega a él, en este caso 2 y 1 respectivamente

El resultado de la evolución se expresa en la figura 3.28:

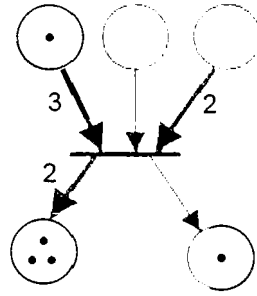


Figura 3.28: Ejemplo de evolución de una transición.

5.4. RED DE PETRI CON ARCOS INHIBIDORES

Son redes que poseen arcos especiales denominados inhibidores, con las siguientes características:

- Parten de lugares y van a transiciones.
- Permiten comprobar directamente la ausencia de marcas en un lugar.
- Representación gráfica en figura 3.29:

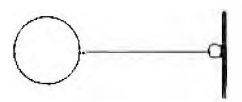


Figura 3.29: Representación gráfica de arco inhibidor.

- Una transición con arco inhibidor está sensibilizada si y solo si el lugar de donde parte el arco no posee ninguna marca.
- La evolución tras el disparo de una transición sensibilizada es el siguiente:
  - Se eliminan $\alpha(p,t)$ marcas de cada lugar de entrada. Un arco inhibidor posee una función de incidencia previa de 0.
  - Se añaden $\beta(t,p)$ marcas a cada lugar de salida.

Sea un ejemplo de sensibilización, disparo y evolución en la figura 3.30:

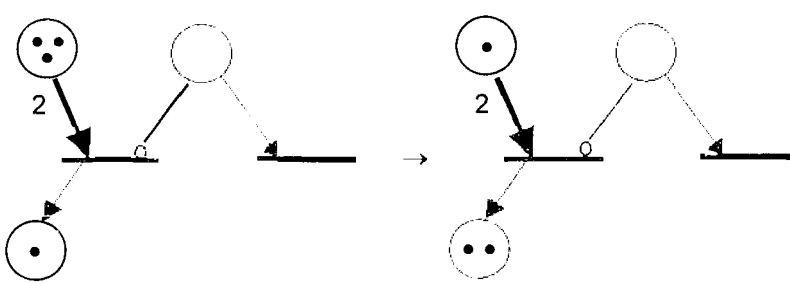


Figura 3.30: Ejemplo de sensibilización, disparo y evolución de red con arco inhibidor.

6. CONCEPTO DE SUBRED EN LAS REDES DE PETRI

Se define subred como el conjunto de secuencias representadas por una parte de la red de Petri, que puede ser utilizada desde diferentes situaciones en la evolución del sistema. En la figura 3.31 se describe un ejemplo, donde la subred se encuentra recuadrada por una línea discontinua.

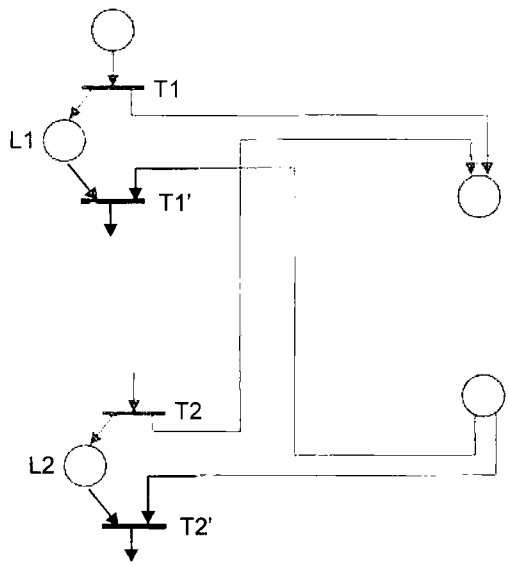


Figura 3.31: Ejemplo gráfico de subred.

Resulta obligado comentar detalladamente las llamadas a la subred, puesto que requieren un especial tratamiento e intervienen de modo decisivo en el buen funcionamiento del sistema.

En el supuesto de que la transición T1 esté sensibilizada y se dispare, surgen de ella dos marcas con cometidos íntimamente vinculados:

- Una de ellas se desplaza hacia la subred, iniciando su propia evolución y ejecutando las acciones asociadas a sus lugares. Al finalizar su recorrido se encuentra ante la disyuntiva de dilucidar hacia qué transición dirigirse.

- La segunda marca se sitúa en el lugar L1, permaneciendo inmóvil hasta que la transición T1' se sensibilice. Resulta claro la función de esta marca, puesto que recuerda al sistema desde qué lugar ha sido llamada la subred.

Ahora solo cabe precisar las ventajas de la utilización de subredes:

- Modularidad en la descripción.
- Idónea para acciones repetitivas.
- Descripciones de tamaño más reducido y por consiguiente realizaciones más económicas.

7. VALIDACIÓN FUNCIONAL DE UNA DESCRIPCIÓN

El objetivo de la validación es evitar pasar de un posible modelo erróneo a la fase de realización. Conviene pues, someter al modelo obtenido a unas pruebas específicas que verificarán la salud de la representación.

Llegado a este punto se hace necesario definir el concepto de *red autónoma*, identificando una red que no posee interpretación. es decir los lugares no poseen acciones y las transiciones no tienen asociadas condiciones externas. Por consiguiente es un simple gráfico sin sistema asociado

Las validaciones funcionales se realizarán siempre sobre una red autónoma.

7.1. PROPIEDADES BÁSICAS QUE CARACTERIZAN UN BUEN FUNCIONAMIENTO

A continuación se enumeran las propiedades básicas que debe cumplir una red de Petri para asegurar, en la medida de lo posible, un buen funcionamiento. En algunas ocasiones, la existencia de ciertas propiedades sólo pretenden advertir al diseñador, y será éste, en última instancia, quien decida si es o no correcta la representación gráfica, de acuerdo con las especificaciones del problema.

- **Vivacidad**: Una transición t es viva para un M_0 (marcado inicial), si existe un marcado posterior que la sensibiliza. Una red de Petri es viva para un M_0, si todas sus transiciones son vivas.

- **Ciclicidad**: Una red de Petri es cíclica para un M_0, si existe una secuencia de disparos que permite alcanzar M_0 a partir de cualquier marcado posterior.

- **Limitación**: Un lugar p es k-limitado para un M_0, si el número máximo de marcas que puede contener es k. Una red de Petri es k-limitada para un M_0, si todos sus lugares son k-limitados.

- **Conflictividad**: En una red de Petri existe un *conflicto* cuando un lugar posee más de una transición de salida (ver figura 3.32).

  La transición t_1 y t_2 están en *conflicto efectivo* sí:

  - Tienen un lugar de entrada en común.
  - El lugar no tiene suficientes marcas para disparar simultáneamente ambas transiciones.

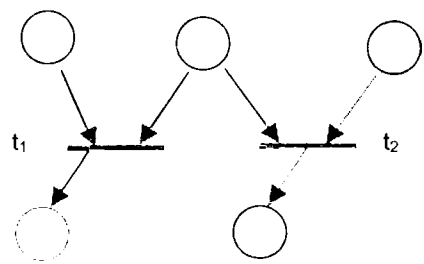


Figura 3.32: Existencia de conflicto.

- **Exclusión mutua**: Dos lugares están en exclusión mutua para un M_0, si no pueden estar marcados simultáneamente para cualquier marcado posterior. En el ejemplo de la figura 3.33, existe exclusión mutua entre los lugares que tienen asociadas las salidas d y e.

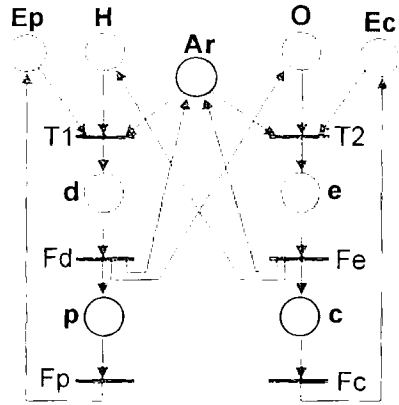


Figura 3.33: Ejemplo con exclusión mutua

7.2. MÉTODOS DE VALIDACIÓN FUNCIONAL

En general, los métodos de validación de la red de Petri se pueden clasificar en los siguientes grupos:

- Métodos estáticos: utilizan grafos y conducen a resultados exactos.
  - *Análisis por enumeración*, se basan en la construcción de un grafo que representa individualizadamente los marcados de la red de Petri y el disparo de sus transiciones. El grafo obtenido permitirá comprobar las propiedades mencionadas.
  - *Análisis por transformación*, este método transforma la red original en otra más sencilla, facilitando verificaciones directas de las propiedades de interés.
  - *Análisis estructural*, permite demostrar algunas propiedades casi independientemente del marcado inicial, considerándose fundamentalmente su estructura. Esta técnica aporta de forma eficiente el análisis de una red de Petri para diferentes marcados iniciales.
- Métodos dinámicos: acreditan cierta confianza en la descripción, pero no demuestran propiedades.
  - *Simulación*, consiste en la utilización de técnicas de simulación. No producirán nunca resultados absolutos, dado que no se basan en la demostración de una propiedad, sino que persiguen una

comprobación parcial, relativa al ámbito específico del funcionamiento simulado.

7.2.1. ANÁLISIS POR ENUMERACIÓN. GRAFO DE MARCADOS

Referencia hecha a lo anterior, este tipo de análisis pretende la construcción de un grafo de marcados. Si la red de Petri es limitada, el grafo será finito y se pueden verificar fácilmente las diferentes propiedades. En el caso de que la red sea no limitada, sucede que el grafo no es finito y, por consiguiente, es imposible construirlo.

Las particularidades que presenta un grafo de marcado son las siguientes:

- Cada nudo representa un marcado alcanzable a partir de M_0.
- Cada arco representa el disparo de una transición.

A continuación se analizará paso a paso la construcción de un grafo de marcados. Para facilitar el seguimiento, se situará a la izquierda la red de Petri y a la derecha el grafo de marcados en construcción. Sea la red de Petri de la figura 3.34 con un marcado inicial.

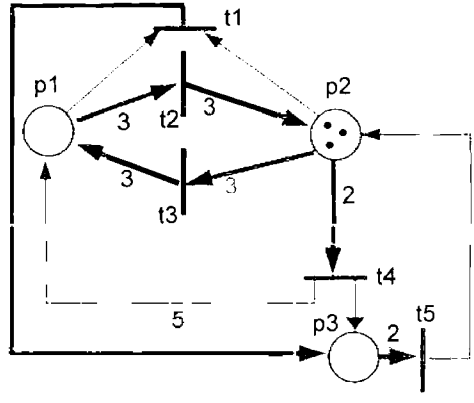


Figura 3.34: Red sobre la que realizará el grafo de marcados.

a) En la figura 3.35 aparece la obtención del primer nudo del grafo de marcados, en él se recoge numéricamente el marcado actual de la red de Petri, que en este caso corresponde al inicial. El contenido del nudo

refleja el valor '030', quiere decir que en el primer lugar hay '0' marcas, en el segundo '3' marcas y en el tercero '0' marcas.

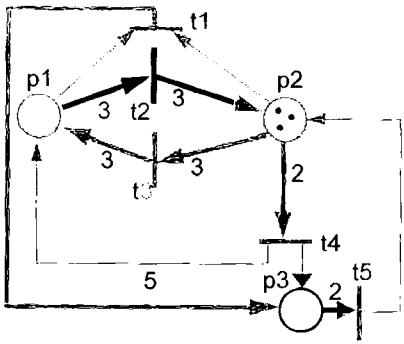
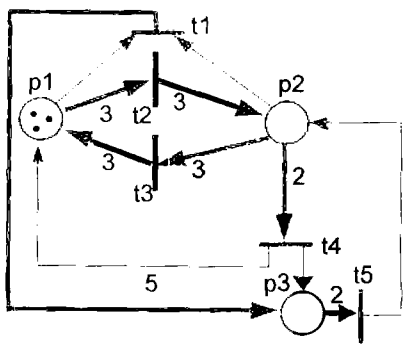


Figura 3.35: Estado intermedio para conseguir el grafo de marcados.

A partir de este estado, la red puede evolucionar disparando la transición t3 o la t4.

b) Se ha disparado la transición t3 a partir del estado inicial 'a', y el nuevo estado es el que indica la figura 3.36.



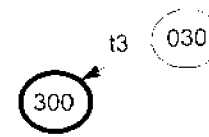


Figura 3 36: Estado intermedio para conseguir el grafo de marcados.

c) Se ha disparado la transición t4 a partir del estado inicial 'a', y el nuevo estado es el que indica la figura 3.37.

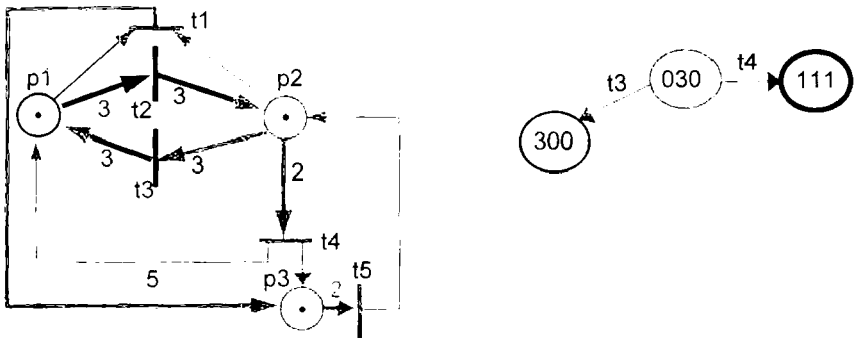


Figura 3.37: Estado intermedio para conseguir el grafo de marcados.

d) Se ha disparado la transición t2 a partir del estado 'b', y el nuevo estado es el que indica la figura 3.38.

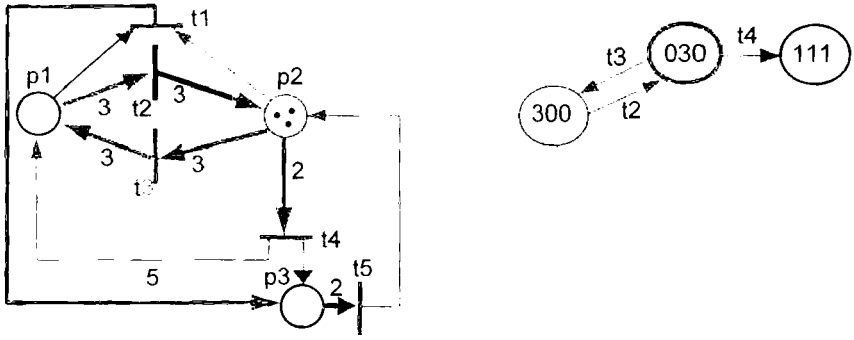


Figura 3 38: Estado intermedio para conseguir el grafo de marcados.

e) Se ha disparado la transición t1 a partir del estado 'c', y el nuevo estado es el que indica la figura 3.39.

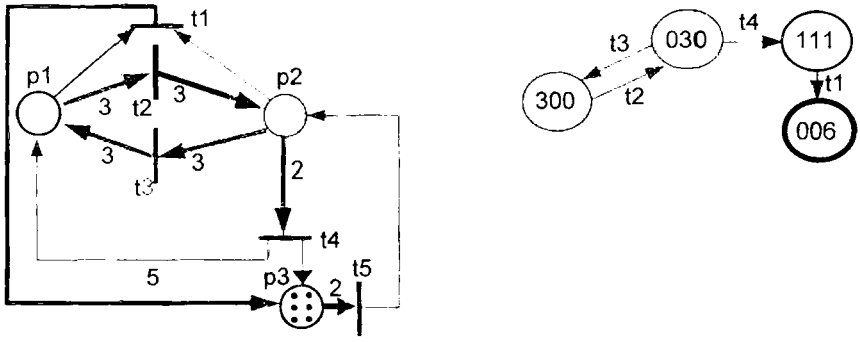


Figura 3.39: Estado intermedio para conseguir el grafo de marcados.

f) Se ha disparado la transición t5 a partir del estado 'e', y el nuevo estado es el que indica la figura 3.40.

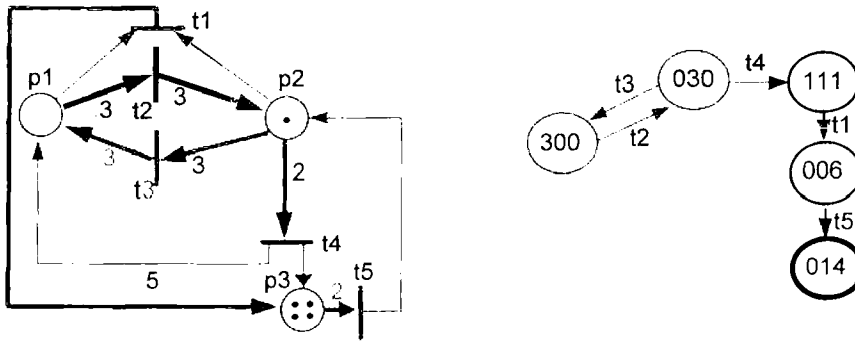


Figura 3.40: Estado intermedio para conseguir el grafo de marcados.

g) Se ha disparado la transición t5 a partir del estado 'f', y el nuevo estado es el que indica la figura 3.41

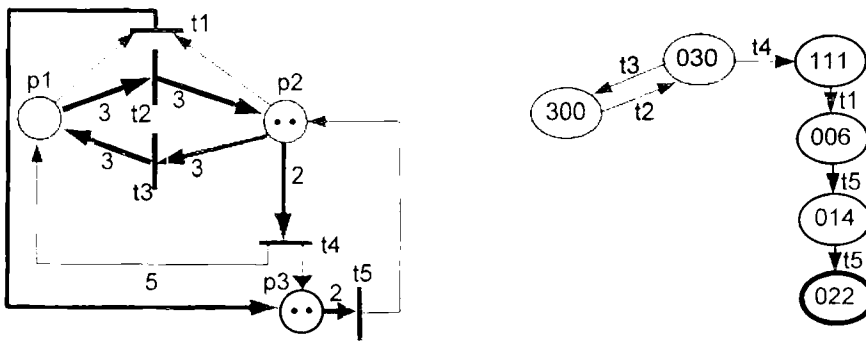


Figura 3.41: Estado intermedio para conseguir el grafo de marcados.

h) Se ha disparado la transición t4 a partir del estado 'g', y el nuevo estado es el que indica la figura 3.42.

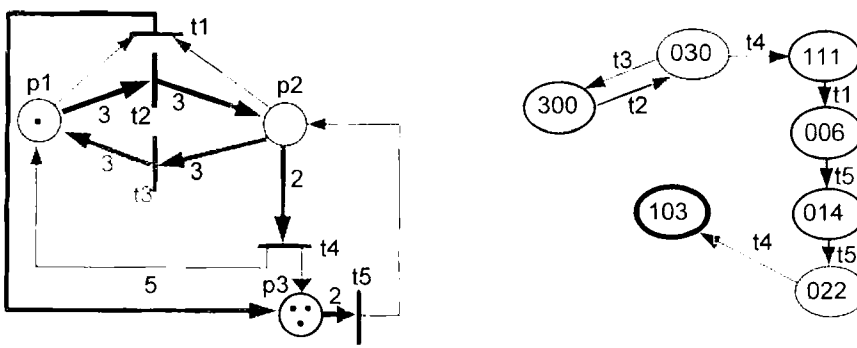


Figura 3.42: Estado intermedio para conseguir el grafo de marcados.

i) Se ha disparado la transición t5 a partir del estado 'h', y el nuevo estado es el que indica la figura 3.43.

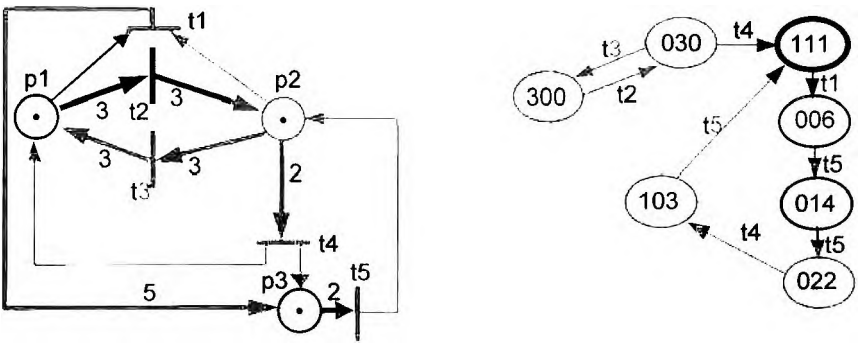


Figura 3.43: Estado intermedio para conseguir el grafo de marcados.

j) Se ha disparado la transición t5 a partir del estado 'g', y el nuevo estado es el que indica la figura 3.44, que corresponde al inicial.

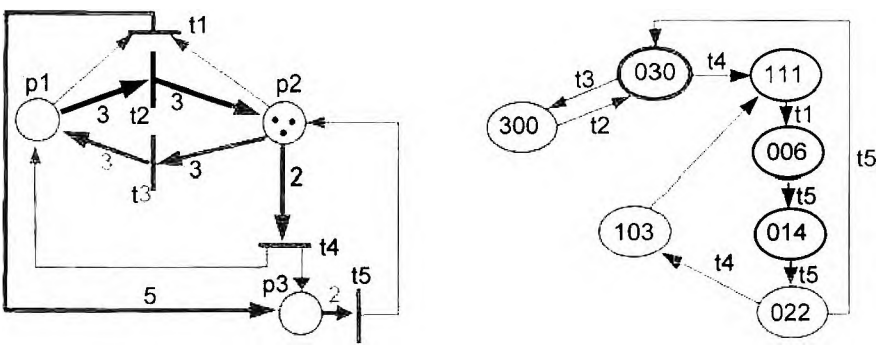


Figura 3.44: Estado intermedio para conseguir el grafo de marcados.

Una vez obtenido el grafo de marcados, se comprueban sobre él las propiedades básicas:

- *Vivacidad:* Una red de Petri para un M_0 es viva si y solo si se cumple que en su grafo de marcados:
  - No existe nudo terminal, es decir, cualquier marcado sensibiliza alguna transición.
  - Cualquier conjunto de nudos que pueda considerarse terminal, debe contener todas las transiciones.

- *Ciclicidad:* Una red de Petri para un M_0 es cíclica si y solo si se cumple que en su grafo de marcados es fuertemente conexo.

- *Conflictividad:* Una red de Petri para un M_0 presenta conflictividad si de un nudo parten dos arcos, sin embargo, no implica la existencia de un conflicto efectivo.

- *Limitación:* Se obtiene directamente del mayor número aparecido en el grafo de marcados.

- *Exclusión mutua:* Se toma el grafo de marcados y se sitúa sobre una tabla la simultaneidad entre lugares. En el ejemplo realizado es fácil de observar sin necesidad de realizarla.

  Sea un grafo con los siguientes marcados: 10103, 01102, 01013, 10012, 10101, 01100, 01011, 10010. La tabla de simultaneidad correspondiente es la representada tabla 3.7:

	p1	p2	p3	p4
p2				
p3	x	x		
p4	x	x		
p5	x	x	x	x

Tabla 3.7: Tabla de simultaneidad entre lugares

El marcado 10103 informa que los lugares p1, p3 y p5 se cumplen simultáneamente, luego se marcarán las casillas correspondientes a las intersecciones.

Han quedado sin marcar las intersecciones entre los lugares p1 y p2 y entre p3 y p4, lo cual establece que hay exclusión mutua entre dichos lugares.

Como resumen, se puede concluir que el análisis por enumeración es eficaz, pues permite descubrir directamente todas las propiedades que reflejan un buen funcionamiento de una red de Petri. El problema fundamental que plantea reside en su fuerte naturaleza combinatoria, que lo hace a veces difícilmente utilizable.

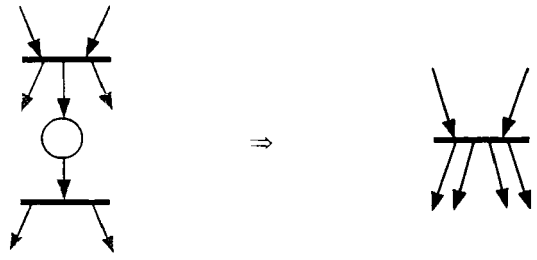
7.2.2. ANÁLISIS POR TRANSFORMACIÓN (REDUCCIÓN)

Este método se basa en la idea de reducir la complejidad de la red inicial, pero de tal forma que, al llevar a cabo la transformación, se preserven las propiedades que se desean analizar.

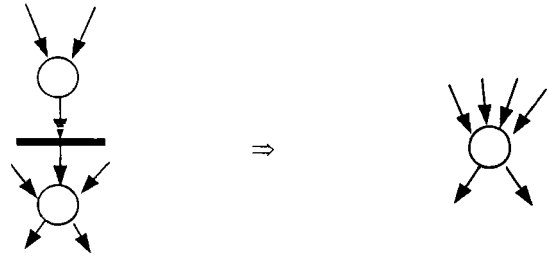
Evidentemente, las reducciones que se van a emplear no conservan la relación funcional evento-acción que describe la red interpretada inicial.

A continuación se presentan un conjunto de reglas simples para la reducción de redes de Petri ordinarias, que permiten conservar la vivacidad y la limitación.

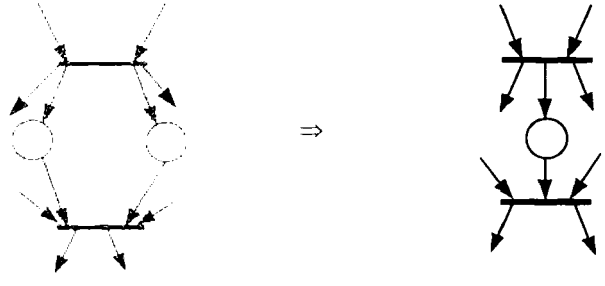
1)

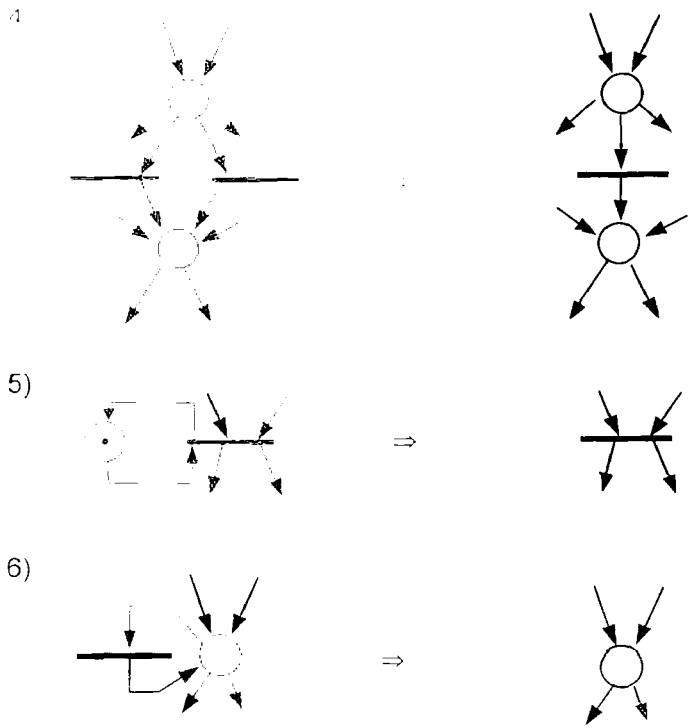


2)



3)





A continuación se muestra una red autónoma plasmando sobre ella el análisis por reducción para comprobar si cumple las propiedades de vivacidad y limitación.

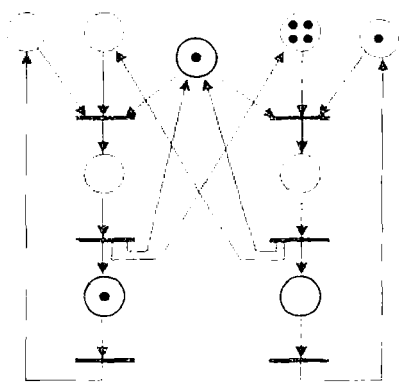


Figura 3.45: Red autónoma sobre la que se aplicará el análisis por reducción.

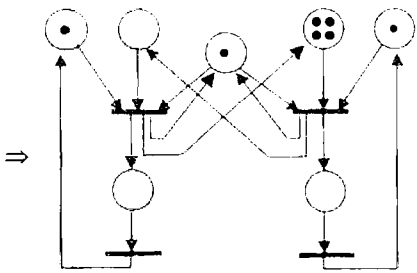


Figura 3.46: Aplicación de la regla número 1 sobre la red original.

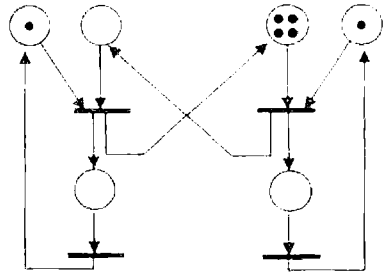


Figura 3.47: Aplicación de la regla número 5 sobre la red anterior.

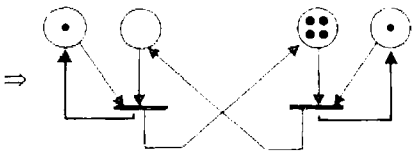


Figura 3.48: Aplicación de la regla número 1 sobre la red anterior.

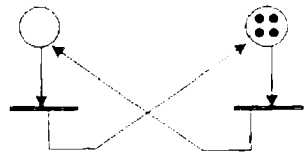


Figura 3.49: Aplicación de la regla número 5 sobre la red anterior.

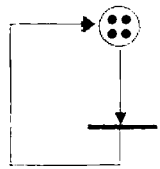


Figura 3.50: Aplicación de la regla número 1 sobre la red anterior.

La red obtenida es muy simple, verificándose que la red original es viva y 4-limitada.

Capítulo 4

Principios de Programación

Contenido

Entorno de programación del autómata programable
Desarrollo de un programa. Ciclo de tratamiento
Operandos del lenguaje de programación STEP5
Juego de operaciones
Formas de representación del lenguaje de programación en STEP5
Estructura de un programa en STEP5
Elaboración del programa. Tipos de procesamiento
Módulos funcionales
Documentación del programa

1. ENTORNO DE PROGRAMACIÓN DEL AUTÓMATA PROGRAMABLE

La estructura general en la que se basa el entorno de programación y diseño con un autómata se encuentra definido por dos grandes bloques:

En primer lugar se encuentra el usuario, quién se encargará de definir el proceso a automatizar, sus variables de entrada y de salida, y poseerá un buen conocimiento de dicho proceso.

En segundo lugar, habrá que decidir cuál será el autómata encargado de soportar el diseño, en el que se desarrollará nuestro proceso mediante la interpretación de un lenguaje de programación que se traducirá en instrucciones de programa que entiende el autómata. Este entorno estará compuesto por tres partes:

- El lenguaje que se utilizará para programar el proceso.
- Un intérprete capaz de traducir dicho lenguaje a instrucciones de programa.
- El autómata programable en sí, conectado con los elementos del proceso a automatizar.

Para el caso que nos ocupa, el lenguaje a utilizar será el STEP 5 de Siemens. Este lenguaje es traducido mediante un aparato de programación al autómata. Del aparato de programación, hay que decir, que bien puede ser específico de la marca Siemens, o bien, utilizar el software de esta misma marca que posibilita utilizar un ordenador personal como aparato de programación. Este aparato, además de usarse para la programación, tiene una gran utilidad en mantenimiento de la instalación, búsqueda y reparación de averías. De esto se deduce la ventaja de que estos elementos programadores sean portátiles.

Un esquema que resume lo anteriormente expuesto se presenta en la figura 4.1

2. DESARROLLO DE UN PROGRAMA. CICLO DE TRATAMIENTO

Por programa se entiende un conjunto de instrucciones que realizan un conjunto de acciones de forma secuencial, aunque permitiendo saltos, bifurcaciones y llamadas a subprogramas.

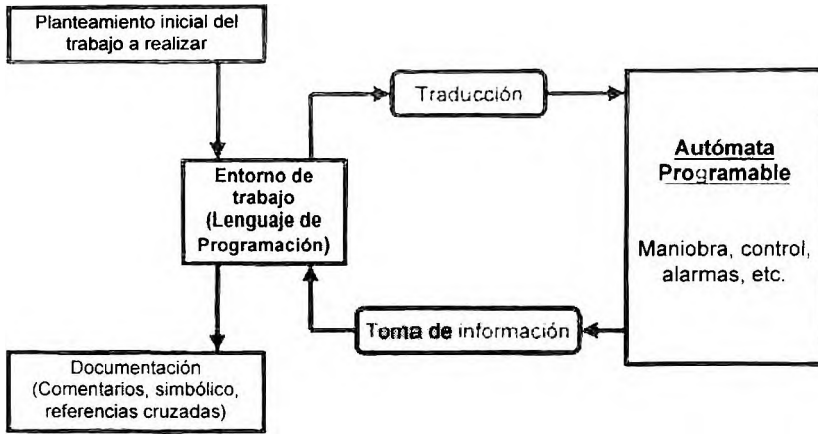


Figura 4.1:Esquema gráfico del entorno de programación

Los lenguajes de programación utilizados normalmente en los autómatas programables se caracterizan por un encadenamiento en la ejecución de sus instrucciones, con la característica esencial de presentar un funcionamiento CÍCLICO.

Estos ciclos se procesan a su vez de forma secuencial, dando la apariencia al usuario de que se procesa en tiempo real. Para decir que el programa, o el autómata actúa en tiempo real es necesario que el tiempo dedicado a cada CICLO del programa sea lo suficientemente rápido para que se dé la situación comentada de tiempo real.

A cada uno de los ciclos se le denomina CICLO DE TRATAMIENTO (CT) y al tiempo en realizar un ciclo; TIEMPO DE CICLO. El tiempo de ciclo es muy importante, dado que cada autómata (por razones de velocidad) posee unos tiempos de ciclo máximos determinados que permitan procesar la automatización en tiempo real.

El encadenamiento de los ciclos se puede realizar de varias formas, atendiendo a la estructura de diseño de cada autómata. Estos pueden ser:

DIRECTO: En este caso, justo al terminar un ciclo de tratamiento comienza el siguiente.

SÍNCRONO: Un reloj externo lanza los ciclos de tratamiento periódicamente cada cierto tiempo.

AUTOSINCRONIZADO: Los ciclos de tratamientos son lanzados solamente cuando, habiéndose terminado el ciclo anterior, se detecta por hardware externo un cambio de estado en alguna de las variables de entrada.

En la figura 4.2 aparece un esquema de cómo son realizados los ciclos de tratamiento.

En el encadenamiento directo, como se observa, no hay espacio de tiempo entre los ciclos de tratamientos, los cuales pueden tener duración de tiempo diferente debido a saltos y bifurcaciones. Los autómatas basados en esta forma de encadenamiento son fáciles en cuanto a su diseño.

En cuanto al síncrono y autosincronizado, permiten realizar otros trabajos (background) aparte de la secuenciación del programa, es decir, atender a otras tareas que pudieran tener encomendadas en su diseño de hardware.

Otro aspecto, en cuanto a diseño de los autómatas a destacar, es cómo realizan la adquisición de las entradas y emisión de las salidas. Desde este punto de vista se pueden clasificar en tres tipos:

- Adquisición y emisión directa (AD - ED).
- Adquisición en bloque y emisión directa (AB - ED).
- Adquisición en bloque y emisión en bloque (AB - EB).

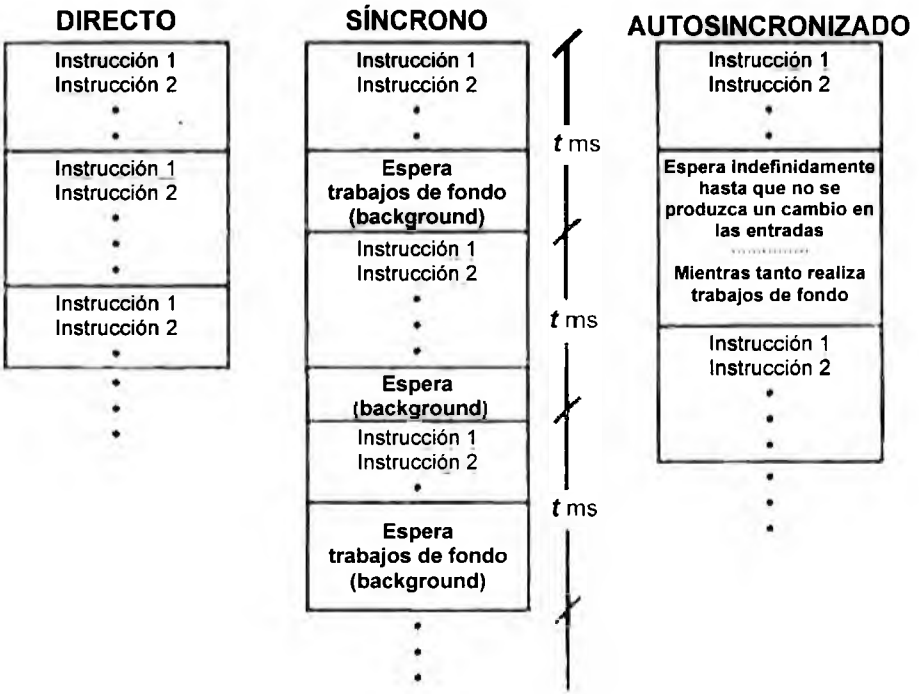


Figura 4.2: Realización de los ciclos de tratamiento

La adquisición y emisión directa significa que el autómata en un ciclo de tratamiento toma las entradas en el instante en que cambian sus estados, es decir, una entrada pudiera ser tomada con dos valores diferentes en un mismo ciclo de tratamiento. Igualmente ocurre con las salidas, se puede dar el caso que una salida dé como resultado dos estados diferentes para un mismo ciclo de tratamiento.

Estos inconvenientes que presenta la adquisición y emisión directa pueden ser evitados en cierta forma mediante una programación adecuada, pero con el añadido de sacrificar la velocidad del programa. En cambio la adquisición y emisión en bloque no presentan estos problemas, dado que, tanto las entradas como las salidas se actualizan para un instante de tiempo quedando almacenadas en una memoria intermedia. A esta forma de tratamiento se la denomina obtener una imagen del proceso de entradas y salidas.

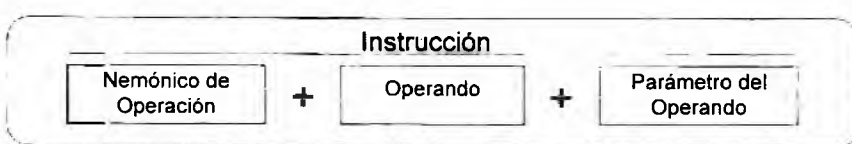
Con todo esto, se resume y concluye, que será necesario para cada aplicación específica estudiar bien el proceso, cómo se realizará la programación y cuáles son nuestras necesidades de velocidad.

3. OPERANDOS DEL LENGUAJE DE PROGRAMACIÓN STEP5

En este apartado se comienza a describir el lenguaje de programación a utilizar en este curso, dado que los ejemplos prácticos se implementarán en un autómata concreto como es el S95U de la marca Siemens. El lenguaje de programación es el denominado por dicha marca STEP 5.

Hay que decir que la filosofía de los lenguajes de programación de los autómatas en general es muy similar. Poseen un conjunto de instrucciones básicas comunes entre ellos a diferencia de los nemónicos utilizados en cada caso, siendo la diferencia más acusada entre diferentes marcas, el entorno de programación, en otras palabras, el aparato dedicado a la programación y su intérprete de usuario.

Cualquier lenguaje se encuentra definido por un conjunto de instrucciones ejecutadas de forma secuencial. La estructura de una instrucción en el lenguaje STEP 5 está formada por:



La **operación** puede ser una función que posea el lenguaje como por ejemplo la función lógica AND. Esta operación se aplica a un **operando**, el cuál designa un tipo de información, que a su vez debe ser especificada distinguiéndola de entre las del mismo tipo por una *posición* que vendrá definida por el **parámetro** del operando.

Sirva como ejemplo la siguiente instrucción, la cual realiza una función AND (operación) de una entrada (operando) situada en la posición 32.0 (parámetro del operando).

```
U  E  32.0
```

A continuación se describen los diferentes elementos básicos que estructuran el lenguaje STEP 5.

3.1. OPERANDOS DE STEP 5

Los diferentes operandos que aparecen en el lenguaje STEP5 son:

Periferia (P,Q)
El operando P (o Q para la periferia ampliada) se utiliza cuando desde el programa de aplicación se accede directamente a las tarjetas periféricas. Hay que destacar que estos operandos no son direccionables bit a bit, sino que su direccionamiento se realiza afectando a un conjunto de bits.

Entradas (E) y Salidas (A)
Son las entradas y salidas propias del autómata, con ellas se accede al exterior, tanto para leer un evento (final de carrera) como para accionar un dispositivo (contactor). Poseen la ventaja de que son direccionables bit a bit. En el autómata S95U la adquisición de las entradas y emisión de las salidas se realizan en bloque.

Marcas (M)
Son unidades de memoria utilizadas para almacenar estados de señal o valores digitales y forman parte de la unidad central. Se utilizan en la programación para guardar valores intermedios. En cuanto a su direccionamiento, al igual que las entradas y las salidas son direccionables bit a bit.
Otra particularidad importante de este tipo de operando es su *volatilidad*, es decir, al ser unidades de memoria sus estados retornan a un valor cero cuando falta la alimentación en el autómata. Sin embargo, y por norma general, los autómatas poseen una pequeña batería que en el caso que falle la alimentación mantienen la información del programa y también

mantienen la información de las marcas. Ahora bien, no todas las marcas mantienen su estado tras un fallo de la alimentación, así que el autómata poseerá (por cuestiones de diseño) un conjunto de marcas denominadas **remanentes** (aquéllas que guardan sus estado) y otras **no remanentes** (su estado se borra siendo cero). Esta particularidad deberá ser tenida en cuenta cuando se diseñe el programa.

Temporizadores (T) y Contadores (Z)
Los temporizadores son operandos que se encuentran integrados en la unidad central y se encargan de realizar diferentes tipos de retardos ante una señal de entrada. Los diferentes tipos de temporizadores que posee el autómata S95U son:

- Retardo a la conexión
- Retardo a la desconexión
- Temporización activada por flanco (impulso)
- Temporización con impulso prolongado
- Retardo a la conexión memorizada

En cuanto a los contadores, éstos también se encuentran integrados en la unidad central, siendo de dos tipos, ambos con cuenta hacia adelante y hacia atrás. Estos operandos permiten la obtención instantánea del valor del contador, así como una salida binaria que nos indica que el contador se encuentra cargado con el valor cero.

Más adelante, en el capítulo dedicado a los elementos de la programación se detallará el funcionamiento tanto de los temporizadores como de los contadores.

Datos (D)
Antes se ha comentado un operando, las marcas, los cuales permitían guardar valores intermedios para su posterior procesamiento. En el caso que el volumen de información a guardar sea elevado, se desestima su uso, utilizándose un nuevo operando denominado **datos**. En cuanto a su tratamiento, es decir guardado y recuperación de los mismos, necesitan de unas instrucciones adicionales. Concluyendo, los datos se utilizan más bien para almacenar valores digitales (numéricos), mientras que las marcas se suelen utilizar para valores binarios.

Constantes (K)
Son operandos, que en el transcurso de la ejecución del programa no van a sufrir modificaciones de su valor. Se utilizan para operar con valores numéricos pudiendo aceptar diferentes formatos numéricos.

Módulos
Son una forma especial de operandos. Tienen un significado más amplio, dado que son unidades estructuradas de organización del programa, conjunto de instrucciones, datos y funciones. En realidad son módulos software que ayudan a estructurar el programa, e incluso comentarlo. Los módulos más comunes son: OB, PB, FB y DB.

3.2. PROCESAMIENTO DE VALORES DIGITALES

Anteriormente, se ha hablado de valores binarios y valores digitales. En los autómatas de Siemens, existen diferentes formas de guardar la información, dependiendo de la magnitud de la misma. En principio, la unidad más pequeña de información es el bit, que almacena un valor "1" ó "0". A partir de esta unidad se forman otras representaciones digitales de la información, que son:

BYTE: Palabra formada por 8 bits.
WORD (PALABRA): Constituida por 16 bits o dos bytes
DOUBLE WORD (DOBLE PALABRA): Constituida por 32 bits.

3.3. PARÁMETROS ASOCIADOS A LOS OPERANDOS

Veamos a continuación la forma de asociar a cada operando un determinado tipo de información atendiendo a su capacidad. Dado que hay varias formas de direccionar la información, es necesario identificarla en los operandos. Para ello se utiliza la siguiente convención:

(nada)	Direccionamiento de un bit.
B	Direccionamiento de un byte.
W	Direccionamiento de una palabra.
D	Direccionamiento de una doble palabra.

Estos identificadores del tipo de procesamiento se escribirán seguidamente del operando, no confundiéndolos con los parámetros de los operandos. Hay que hacer notar que el identificador debe de llevar una congruencia con el parámetro del operando, pues de lo contrario se producirá un error.

Los parámetros de los operandos son direcciones de donde se encuentran los datos. Su unidad básica es el byte y así se encuentra numerado en los autómatas de la serie SIMATIC.

A continuación se presenta en un esquema (figura 4.3) cómo se direccionan los diferentes operandos atendiendo a la unidad básica (byte).

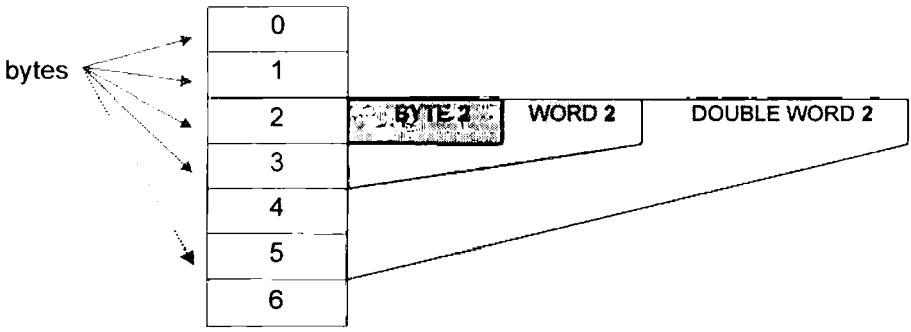


Figura 4.3: Direccionamiento de los operandos

En el caso de direccionar un sólo bit hay que indicar el byte del que se quiere dicho bit, y luego indicar la posición del bit dentro del byte, mediante un punto como separador.

En el gráfico anterior, si quisiéramos trabajar con dos palabras (dos bytes o 16 bits) hay que tener cuidado con el solapamiento. No sería viable coger la palabra 2 y la palabra 3 dado que tendrían en común el byte 3.

Como ejemplos de direccionamiento tenemos:

E	32.0	Direcciona el bit 0 del byte 32.
EB	33	Direcciona el byte 33.
EW	32	Direcciona los bytes 32 y 33.
ED	32	Direcciona los bytes 32, 33, 34 y 35

El mismo tratamiento es válido para el direccionamiento de las salidas (A) y las marcas (M).

3.4. FORMATOS DE PRESENTACIÓN NUMÉRICA

Cuando se quiere almacenar datos para un posterior uso o por circunstancias de la programación elegida, éstos se almacenarán como constantes, que como se vio en el apartado 3.1, son un tipo de operandos. Pues bien, el lenguaje STEP 5 permite guardar estas constantes con diferentes formatos numéricos.

El que se utilice un formato u otro, depende de la capacidad de representación numérica que posean nuestros datos, y por otro lado dependerá del formato específico, como pueden ser por ejemplo las constantes de tiempo necesaria en los temporizadores.

La tabla 4.1 resume los diferentes formatos:

Formato	Representación Numérica	Rango
KH	Hexadecimal	0000 a FFFF
KF	Nº en coma fija	-32768 a 32767
KC ó C	Alfanumérico	Limitado por " "
KG	Nº en coma flotante	-1469369-38 a +1701412+39
KT	Constante de tiempo y su base de tiempo, separado por un punto decimal.	0.0 a 999.3
KZ	Valor de cómputo	0 a 999
KY	Dos Bytes, en formato decimal y separados por coma. Ej: KY 34,98	0 a 255 0 a 255
KM	Binario	0000000000000000 a 1111111111111111

Tabla 4.1: Formatos de representación numérica

3.5. PARTICULARIZACIÓN PARA EL AUTÓMATA S95U

Analizado en los apartados anteriores algunas características propias del lenguaje de programación STEP 5, es necesario detallar las particularidades propias del autómata de Siemens S95U. Éstas son:

Tiempo de vigilancia de ciclo	aprox. 300 ms
Marcas	2048 (bits), de ellas 512 son remanentes Marcas remanentes: M 0.0 hasta la M 63.7 Marcas no remanentes: el resto hasta la M 255.7
Temporizadores	128, siendo su intervalo de 0,01 s a 9.990 s
Contadores	128, de ellos, los 8 primeros son remanentes. Valor de cuenta de 0 a 999
Entradas y Salidas digitales	16 entradas (bits), de la E32.0 a la E33.7 16 salidas (bits), de la A32.0 a la A33.7
Entradas de Alarma	4 (bits), de la E34.0 a la E34.3
Entradas de Contador	2 (word), EW 36 y EW38
Entradas analógicas	8 (word), EW40, EW42, EW44, EW46, EW48, EW50, EW52 y EW54
Salida analógica	1 (word), AW40
Entradas y Salidas (Periferia ext.)	Digitales: máximo total hasta 256 bytes, es decir de la E 0.0 a la E 31.7 para las entradas y de la A 0.0 a la A 31.7 para las salidas. Analógicas: máximo total 16

4. JUEGO DE OPERACIONES

El lenguaje STEP 5 posee un conjunto de operaciones, que a grosso modo se pueden dividir en cuatro conjuntos diferenciados:

- Operaciones binarias.
- Operaciones digitales.
- Operaciones o funciones de organización.
- Instrucciones de sustitución.

4.1. BINARIAS

Son operaciones que permiten operar mediante funciones lógicas del Álgebra de Boole, estados binarios. Entre ellas se pueden encontrar:

Combinaciones binarias, como la función AND, OR y NOT, que en caso de utilizar contactos corresponden con las conexiones serie, paralelo, contacto normalmente cerrado, respectivamente. Para estas operaciones los operandos más utilizados son los direccionables bit a bit, como las entradas, salidas y marcas.

Funciones de memoria, las cuales asignan los resultados de las combinaciones binarias a otros operandos, como por ejemplo a las marcas y las salidas. También se encuentran entre estas funciones las dedicadas al **set** y **reset** de los bits.

Funciones de tiempo, donde se encuentran las dedicadas a los temporizadores. Éstas funciones utilizan combinaciones binarias previas para su activación e igualmente hacen uso de su resultado (salida) para realizar nuevas combinaciones o asignaciones de salidas.

Funciones de cómputo, formada por los contadores software que dispone el lenguaje, donde aparecen funciones para arrancar los contadores, incrementarlos y decrementarlos por ejemplo.

4.2. DIGITALES

Estas se diferencian de las anteriores en que la unidad básica de operación no es el bit, sino palabras mayores, como el byte, word y double word, y por tanto, operan con valores digitales numéricos.

Estas operaciones utilizan como base los registros de la CPU del autómata, concretamente, los acumuladores. Las operaciones más usuales son la carga de datos y la transferencia de los mismos. Además, como funciones digitales se consideran:

- Las operaciones de comparación de valores numéricos.
- Las operaciones aritméticas.
- Las operaciones de combinación (AND, OR, etc.) por palabras.

4.3. ORGANIZACIÓN

Son funciones diseñadas especialmente para la estructuración del programa, destacando entre ellas:

- Funciones de módulo, en las que se encuentran las llamadas a módulos, condicionales e incondicionales, y las operaciones de final de módulo.
- Funciones de salto, que permiten realizar bifurcaciones en el programa en función de los resultados de una combinación.
- Funciones de desplazamiento, que operan en los acumuladores permitiendo desplazamientos de la información a la derecha o a la izquierda.
- Funciones de transformación, que operan sobre valores digitales realizando operaciones como el complemento a uno y a dos, transformaciones en BCD a binario y viceversa y cualquier otra que transforme el contenido del valor almacenado.
- Otras, como funciones de procesamiento, incrementar y decrementar el acumulador, bloqueo y liberación de alarmas, etc.

4.4. SUSTITUCIÓN

Son instrucciones sólo para uso en los módulos funcionales. Éstas permiten pasar parámetros a los módulos funcionales de forma que se consigue una gran estructuración del programa.

5. FORMAS DE REPRESENTACIÓN DEL LENGUAJE DE PROGRAMACIÓN EN STEP5

En general, los lenguajes de programación de los autómatas programables son ***lenguajes orientados al problema***. Esto significa que en su diseño se ha tenido en cuenta las aplicaciones que han de resolverse, en nuestro caso, aplicaciones en el entorno industrial. Existen dos concepciones diferentes en la realización de los lenguajes:

1. Descripción funcional de los automatismos.
2. Esquemas de realización cableada de los automatismos

A la primera, pertenece el lenguaje compuesto por un conjunto de instrucciones que poseen una determinada función y que es secuencial. Este lenguaje es al que se ha hecho referencia en los apartados anteriores, y se denomina **lista de instrucciones**.

Pero STEP 5 posee otras dos formas de programar, también secuencial pero con instrucciones gráficas, es decir, programación visual, en la que el programa no es otra cosa que una descripción mediante símbolos (ej: un plano), tal como se hace en el momento de diseñar un automatismo. A este caso, corresponde la segunda concepción anterior.

Dependiendo de la forma de representación de los esquemas, existen dos formas de realizar la programación Estas dos formas de programa son, ***esquema de funciones y esquema de contactos***.

En los autómatas de la serie SIMATIC S5 las tres formas de representar el lenguaje son las que muestra la figura 4.4:

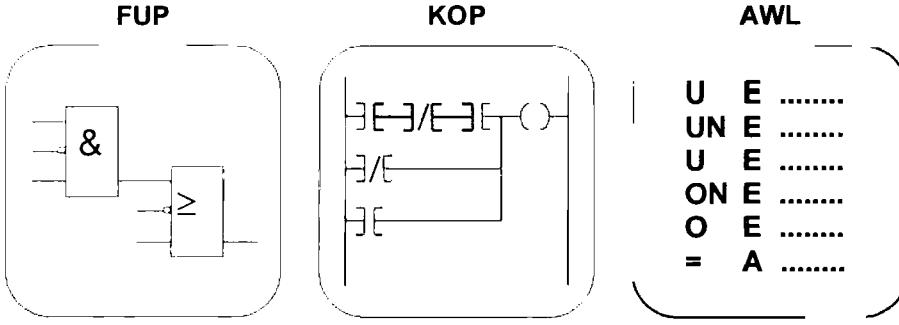


Figura 4.4: Distintas representaciones de lenguajes de programación

5.1. DIAGRAMA DE CONTACTOS -KOP-

El lenguaje en diagramas de contactos se basa en los esquemas electrotécnicos empleados para representar los automatismos formados por contactos y relés. Este lenguaje es muy utilizado en E.E.U.U. de ahí que todos los autómatas fabricados allí posean este lenguaje, e incluso algunos fabricantes sólo posean éste incluido en sus autómatas. Esta forma de programar, se argumenta en base a que es el operario quién estará a cargo del proceso a pie de máquina y por tanto debe de poseer un lenguaje lo más parecido a su forma de trabajar.

Aunque, en Europa los diagramas de contactos se encuentran normalizados, en los lenguajes de programación se sigue utilizando la simbología americana.
Un ejemplo de cómo aparece esta forma de programar en el lenguaje STEP 5 aparece en la figura 4.5. Obsérvese que existen ocho campos para insertar un símbolo. En caso de necesitar más campos es necesario añadir un segmento más de código y continuar en éste con la programación.

También es importante resaltar que en cada segmento, sólo puede aparecer una asignación.

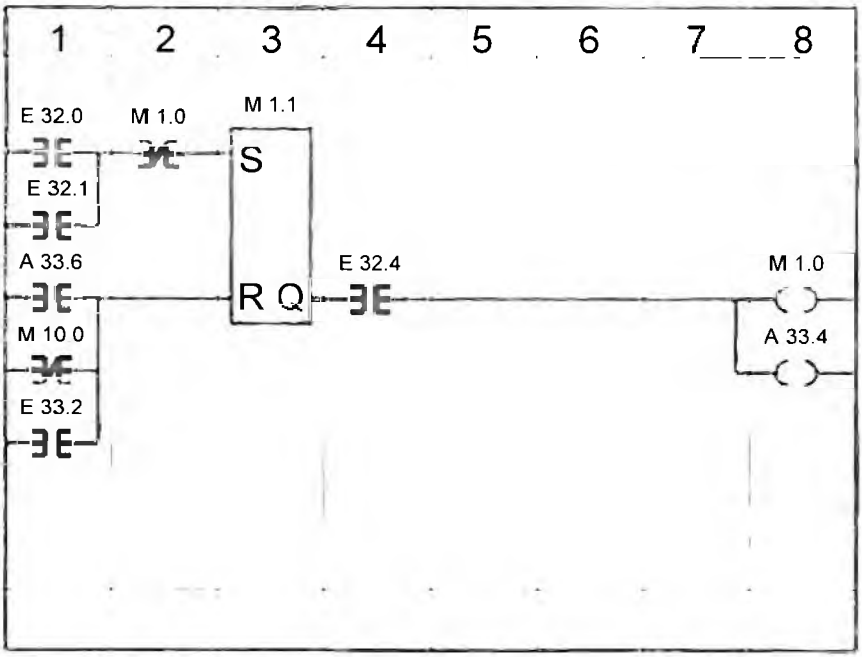


Figura 4.5: Representación en lenguaje de contactos

5.2. DIAGRAMA DE FUNCIONES -FUP-

El lenguaje de funciones utiliza los símbolos que cualquier ingeniero eléctrico usa para realizar esquemas de circuitos digitales combinacionales y secuenciales. Al igual que el diagrama de contactos su realización consiste en copiar el esquema electrónico del circuito.

A diferencia de los diagramas de contactos, la simbología que se utiliza está normalizada a nivel mundial por la CEI-117-15, que a su vez satisface las normas alemanas DIN 40700, DIN 40719 y DIN 19239.

Al igual que en los diagramas de contactos, en cada campo en los que se divide la pantalla del aparato de programación, se insertará una función hasta rellenar los ocho campos. Asimismo, sólo se permite una asignación por cada segmento.

En la figura 4.6 aparece un listado en FUP.

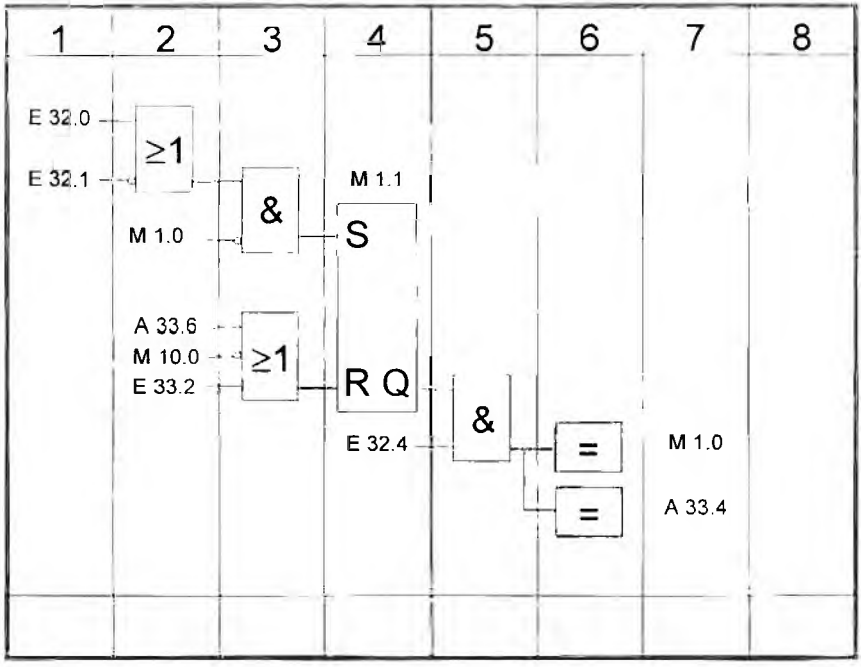


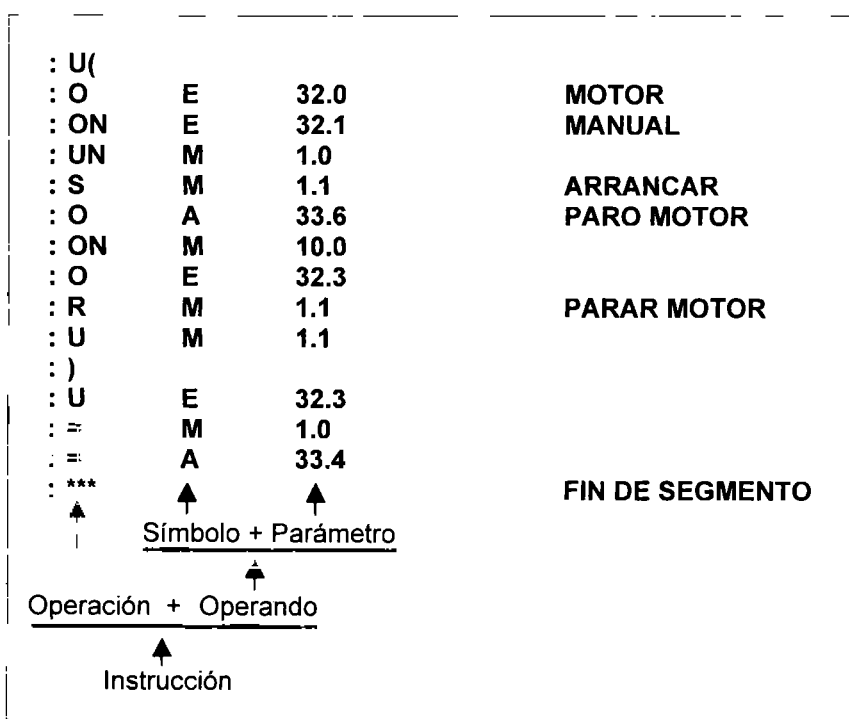
Figura 4.6: Representación en esquema de funciones

5.3. LISTA DE INSTRUCCIONES -AWL-

Programar en listas de instrucciones no corresponde a ninguna representación gráfica de los automatismos, siendo un poco más compleja la programación, pero con la ventaja que es la forma de programar que permite mayor libertad al usuario, dado que utiliza todo el conjunto de instrucciones del lenguaje. Esta forma es la más habitual para los usuarios familiarizados con el uso de ensambladores de microprocesadores y microcontroladores.

El lenguaje listas de instrucciones empleado por STEP 5 satisface la norma DIN 19239. A este respecto hay que decir que la CEI posee un lenguaje en listas de instrucciones normalizado. En general, cualquier lenguaje en listas de instrucciones posee la misma filosofía, encontrándose las diferencias en las denominaciones de los nemónicos y algunas instrucciones y funciones diferentes.

En la figura 4.7, aparece el lenguaje en lista de instrucciones correspondiente a los anteriormente expresados en KOP y FUP



6. ESTRUCTURA Y ORGANIZACIÓN DE UN PROGRAMA EN STEP5

Resumiendo lo expuesto hasta ahora, se sabe que un programa realizado para un autómata se ejecuta cíclicamente, y que dicho programa está compuesto por una secuencia de instrucciones bien en AWL, FUP o KOP.

Ahora bien, en aras de confeccionar un programa de forma estructurada y bien organizada, tanto para su desarrollo como para su mantenimiento posterior, el desarrollo del programa en los autómatas industriales se divide en MÓDULOS de programa.

Estos módulos, generalmente contienen el programa, datos, documentación, funciones repetitivas, etc. Los módulos poseen un símbolo que los identifica formado por dos letras, más un número para discernir dentro de un mismo tipo, por ejemplo: PB 35. En cuanto a su capacidad cada módulo abarca hasta 4096 palabras de longitud, es decir, la longitud máxima en cuanto a instrucciones es de 4096 instrucciones.

En el apartado 3, se definieron a los módulos como un tipo especial de operandos, y que a continuación se pasa a describir la función tecnológica asociada a cada uno de ellos.

Los diferentes módulos del lenguaje STEP 5 son:

Módulos de Organización (OB)
Los módulos de organización son las interfaces entre el sistema operativo del autómata (transparente para el usuario) y los programas que propiamente realiza el usuario. En otras palabras, cuando se da alimentación al autómata, su sistema operativo realizará una serie de llamadas a los módulos de organización que corresponda y ejecutará las órdenes que se encuentren en éstos, que previamente habrá escrito el usuario con instrucciones. Se designan por las letras OB, por ejemplo: OB21.

Si bien se ha comentado que los módulos de organización son programados por el usuario, hay que hacer salvedades. Por esto último, los módulos de organización se dividen en dos grupos:

- Los módulos de organización que llama el sistema, programados por el usuario, y que controlan la elaboración del proceso de los programas.

- Los módulos de organización que se encuentran grabados en el autómata con funciones especiales, en los que no se puede escribir

código, y a los que el usuario llamará para realizar dichas funciones especiales

Dependiendo del autómata elegido dentro de la gama SIMATIC S5, se encontrarán disponibles unos u otros módulos de organización. En particular para el autómata S95U, se tienen los siguientes: OB1, OB3, OB13, OB21, OB22, OB31, OB34 y OB251.

Módulos de Programa (PB)
Estos módulos son los que generalmente se utilizan para realizar los programas, disponiéndose de 256 módulos (0 a 255) para el autómata S95U. Para aclarar su uso, supóngase que se desea realizar dos maniobras distintas en un motor de inducción, como puede ser un arranque estrella-triángulo, o bien, un arranque por resistencias rotóricas. En este caso, el programa para cada arranque (función tecnológica) se realizará en módulos PB diferentes. Para realizar cada arranque se llamará al módulo correspondiente.

Módulos Funcionales (FB)
Realizan la misma función básica que los módulos de programa, se utilizan para realizar los programas. Sin embargo, existe una diferencia importante, y es que se utilizan cuando sea necesario tareas complejas o bien repetitivas, pues son módulos a los que se pueden pasar parámetros por referencia. Un ejemplo puede ser la necesidad de realizar un arranque estrella-triángulo a una instalación que posee 20 motores. Sería inconcebible realizar 20 PB con el mismo programa. El autómata S95U posee 256 módulos FB, del 0 al 255.

Módulos de Paso (SB)
Los módulos de paso se utilizan, por regla general, en el control de la ejecución, en unión con un **módulo de función (FB)** *"control de la cadena de ejecución"* específico.
Los módulos de paso se pueden llamar también independientemente por el usuario, comportándose entonces como simples módulos de programa (PB), siendo una opción permitida en el caso que se necesiten más de los 256 módulos de programa que dispone el autómata.

Módulos de datos (DB)
En los módulos de datos, como su nombre indica, se encuentran los datos con los que trabajan los programas de usuario. Un módulo de datos comprende 256 palabras de datos (esta restricción es debida a que el área de direccionamiento directo está limitada a las 256 primeras palabras de datos).

El autómata S95U posee un módulo de datos especial, el DB1, el cual guarda información para procesar otros módulos de organización y otras

funciones integradas. Por tanto el usuario dispondrá para almacenar datos de los módulos del 2 al 255.

6.1. ORGANIZACIÓN DEL PROGRAMA

La organización del programa consiste en determinar cuáles y en qué orden se tienen que ejecutar los módulos realizados por el usuario. Para las llamadas a los diferentes módulos se utilizan las funciones de organización, siendo estas llamadas condicionales o incondicionales según se programe.

Cuando se realiza una llamada a un módulo se ejecutan las instrucciones contenidas en dicho módulo y a su término, se continúa en la instrucción siguiente de la llamada del módulo del que se llamó. Por esta razón, se pueden anidar llamadas de módulos, o sea, un módulo que es llamado, a su vez puede llamar a otro, y así sucesivamente hasta un máximo de 16 módulos para las CPU 941 y 942, y 32 para las CPU 943 y 944. En el caso que se sobrepase este límite el autómata pasa a STOP.

Un posible ejemplo de organización de los módulos aparece en la figura 4.8.

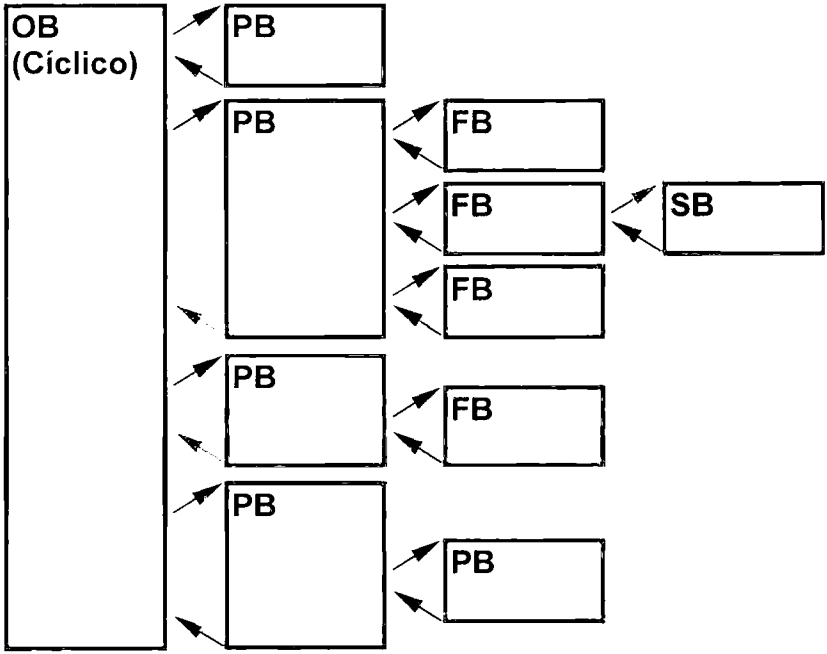


Figura 4.8: Organización de módulos

7. ELABORACIÓN DEL PROGRAMA. TIPOS DE PROCESAMIENTO

El programa que realiza el usuario generalmente se elabora de forma cíclica, tal como se anunció al principio del capítulo. El programa cíclico es el alma principal del programa, sin embargo existen diferentes procesamientos aparte del cíclico, que también son programables por el usuario, o bien, llamados por él.

Suponiendo que en el autómata se encuentra un programa de usuario cualquiera en general, las etapas o fases que sigue el autómata son las siguientes:

- En primer lugar, la primera acción que realiza el sistema operativo del autómata, una vez realizado un chequeo y las funciones que haya diseñado el fabricante, es una llamada al procesamiento de **arranque**. Este procesamiento realizará las instrucciones que encuentre en los módulos de organización correspondientes. Estos módulos sólo se ejecutan una vez.

- Y en segundo lugar, realizará una llamada al módulo de organización del procesamiento **cíclico**, el cuál contendrá la parte principal del programa de usuario. Cuando termine de realizar un ciclo volverá a ejecutar el siguiente y así indefinidamente. El procesamiento cíclico a su vez se divide en:

  - Arranque de un temporizador de supervisión del tiempo de ciclo.
  - Lectura de las señales de entrada y almacenamiento en la imagen del proceso de entrada (memoria intermedia).
  - Elaboración del programa cíclico.
  - Transferir los estados de las señales de salida de la imagen del proceso de salida a las salidas propias del autómata.

  Durante la elaboración del procesamiento cíclico, éste puede ser interrumpido por varias causas, que en principio no son previsibles y que hay que atender rápidamente:

  Alarma: El autómata posee varias entradas de alarmas, de forma que cuando se produzca alguna, se abandone el proceso cíclico y se llame a un módulo de organización específico encargado de procesar la alarma. El usuario se encargará de programar dicho módulo para que actúe de la forma desea ante una alarma del proceso que se está automatizando. Una vez

ejecutado el módulo de alarmas se devolverá el proceso a la elaboración cíclica.

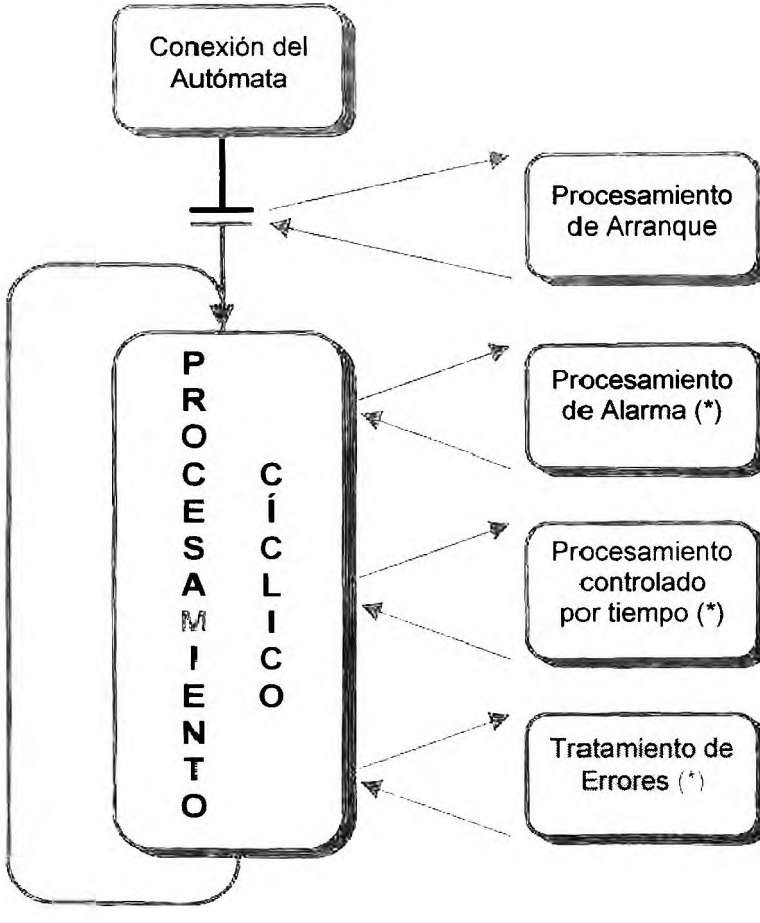
Tiempo: Son interrupciones controladas por tiempo. El autómata posee unos módulos de organización controlados por tiempo. Si estos módulos se programan por el usuario, cada determinado intervalo de tiempo (también seleccionado por el usuario) se abandonará la elaboración cíclica y se ejecutarán las instrucciones contenidas en el módulo de procesamiento controlado por tiempo. Una vez se acabe, se volverá a la elaboración cíclica hasta que nuevamente (después del intervalo de tiempo) se vuelva a interrumpir el procesamiento cíclico.

Error: Cuando el procesador encuentra un error, se interrumpirá el procesamiento cíclico y según sea el fallo, el autómata se detendrá (STOP), o bien, ejecutará un módulo de procesamiento del error que se ha producido. En este módulo el usuario programará la reacción del autómata ante un error conocido.

En la figura 4.9 aparece un esquema donde se ilustra la elaboración del programa del autómata S95U.

A continuación se describen los diferentes tipos de procesamientos que poseen los Autómatas de la serie SIMATIC, particularizando para el S95U. Los diferentes tipos de procesamiento son realizados por los módulos de organización (OB). La numeración asignada a cada módulo OB nos informará del tipo de procesamiento.

- Procesamiento de Arranque.
- Procesamiento Cíclico.
- Procesamiento controlado por Alarmas.
- Procesamiento controlado por Tiempo.
- Tratamiento de los Errores.



(*) Estas interrupciones también pueden darse cuando se esté procesando otra interrupción, dependiendo de las prioridades.

Figura 4.9: Elaboración de los distintos módulos de organización

7.1. PROCESAMIENTO DE ARRANQUE

Este procesamiento se realiza una vez después del encendido del autómata y después de haber realizado sus comprobaciones internas.
Los diferentes tipos de arranque dependerán de cómo se encuentren el interruptor de alimentación y el interruptor de arranque/parada del autómata.
La función del interruptor de alimentación (ON/OFF) es evidente, y la función del interruptor arranque/parada (RUN/STOP), es una vez el autómata se encuentra alimentado, hacer correr el programa grabado en el

autómata (RUN), o bien, llevarlo al estado de parada (STOP) donde todas las salidas se actualizan a un valor cero y el programa deja de funcionar.
Descrito el funcionamiento de estos interruptores, los posibles procedimientos de arranque de un autómata son:

Arranque en Frío: Este arranque se produce cuando el interruptor RUN/STOP se encuentra en la posición STOP y se produce un cambio de OFF a ON en el interruptor de encendido del autómata. Cuando se produce este arranque el autómata procesa el módulo de organización **OB 20**

Arranque inicial manual: El autómata se encuentra alimentado (ON) y produce un cambio en el interruptor RUN/STOP de STOP a RUN. El autómata procesa en este caso el módulo de organización **OB21**.

Arranque inicial automático: El interruptor RUN/STOP se encuentra en la posición RUN, y se procede a la alimentación del autómata, es decir, el interruptor ON/OFF pasa de OFF a ON. En este caso el autómata ejecuta el módulo de organización **OB22**.

En la tabla 4.2 aparece reflejado los tres tipos de arranque, indicando que el autómata S95U no posee el módulo de organización **OB20**.

Accionamiento	Procedimiento de Arranque	OB solicitado
OFF a ON (STOP)	Arranque en frío	~~OB20~~
STOP a RUN	Arranque inicial manual	**OB21**
OFF a ON (RUN)	Arranque inicial automático	**OB22**

Tabla 4.2: Tipos de arranque

En un arranque inicial manual y automático, el procesador realiza las siguientes operaciones:

- Ajuste previo de tarjetas de la periferia para el tratamiento previo de señales (inteligentes).
- Inicializa los datos previos o parámetros de las tarjetas de conexión.
- Inicialización de marcas.
- Liberalización de alarmas.
- No presenta limitación de tiempo.

Tras estas operaciones, se transfiere la imagen del proceso de las entradas y se realiza la llamada al OB correspondiente, para seguidamente después empezar con el procesamiento cíclico. Los módulos de arranque siempre se

ejecutarán (el que corresponda), sólo que en el caso que no posean código de programa, no realizarán nada.

7.2. PROCESAMIENTO CÍCLICO

En el procesamiento cíclico del programa de usuario, el sistema llama al módulo de organización **OB1** En el módulo OB1 empieza el programa de usuario con la primera instrucción que se encuentre. En este módulo se realizarán las llamadas a los módulos de programa (PB), módulos de funciones (FB) y módulos de datos.

¿Cómo organizamos la estructura del programa de usuario?. Ante esta pregunta, existen varias respuestas, en función del proceso que se quiera automatizar. Si bien, se pueden plantear dos concepciones diferentes a la hora de estructurar la programación.

Estructuración tecnológica
Por un lado, se puede dividir el programa tecnológicamente, es decir, diseñar módulos de programa o de funciones que realicen las operaciones tecnológicas que aparecen en el proceso a automatizar. Un ejemplo podría ser una instalación de envasado de aceite de una refinería. En este proceso aparecerá como mínimo una línea de producción donde se obtenga como producto final; latas de aceite de 20 litros. El proceso, bien podría estar dividido en:

1. Alimentación de las latas vacías.
2. Transporte hacia el llenado.
3. Llenado, pesado e inserción del tapón en las latas.
4. Transporte hacia el etiquetado.
5. Etiquetado de las latas.
6. Transporte hacia el apilado.
7. Apilamiento.

Una forma de proceder es realizar módulos de programa para cada una de las funciones tecnológicas enumeradas y secuenciales del proceso, y módulos de funciones para aquellas operaciones que sean repetitivas, como el transporte. Seguidamente cada módulo de programa se irá dividiendo en subprogramas con objeto de reducir la complejidad del diseño.

Estructuración funcional
Supóngase la automatización de un conjunto de ascensores. En este caso el proceso se puede dividir atendiendo a funciones como son subir, bajar, abrir puertas, cerrar puertas, lectura de pulsadores de planta, lectura de pulsadores de cabina, etc. Posteriormente se diseñará una función principal

que se encargue de ejecutar el algoritmo que se ha pensado para atender las llamadas del ascensor, aparcamiento y otras funciones que se añadan al funcionamiento del ascensor.

Estos dos puntos de vistas se pueden complementar y realizar una estructura híbrida que se acomode lo mejor posible al proceso en cuestión que se va a automatizar.

7.3. PROCESAMIENTO CONTROLADO POR ALARMAS

La elaboración de un proceso de alarma ocurre cuando una señal procedente del proceso arranca en el autómata programable un procedimiento, mediante el cual se interrumpe la programación cíclica del programa (se interrumpe el OB1) y se elabora un programa específico.

El programa específico se realiza en unos módulos de organización destinados a tal fin. Estos módulos serán llamados por el sistema cuando ocurra una alarma.

Los módulos de organización disponibles para el procesamiento por alarmas son los siguientes (el autómata S95U sólo dispone del módulo OB3):

Módulo	**Línea de Interrupción**
OB2	IRA
OB3	**IRB**
OB4	IRC
OB5	IRD
OB6	IRE
OB7	IRF
OB8	IRG
OB9	IRH

Las entradas de alarmas para el autómata S95U son cuatros, las E 34 0, E34.1, E34.2 y E34.3. Ante una entrada activa (+24 V) en alguna de estas patillas, el programa que se esté ejecutando se interrumpirá y se pasará a ejecutar las instrucciones que el usuario haya programado en el módulo OB3.

Para averiguar cuál o cuales de las entradas de alarmas a lanzado el OB3, es necesario leer dichas entradas, por ejemplo, mediante L PW 34, operación que carga en el acumulador la palabra (16 bits) 34 de la periferia. Las entradas de alarmas se encuentran alojadas en un conector de nueve patillas (DB9) situado en la parte frontal del autómata S95U. En la figura

4.10, se representa un esquema de conexionado para conectar dichas entradas ante señales de alarmas.

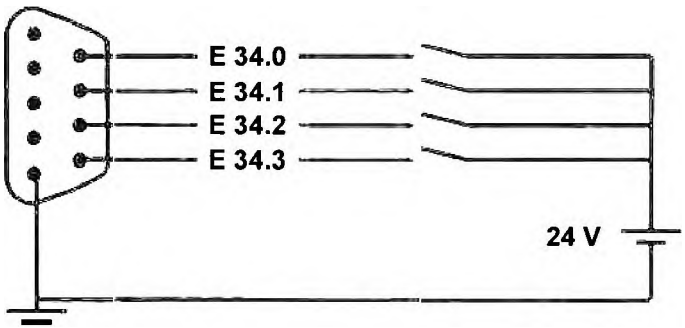


Figura 4.10: Esquema de conexionado de las alarmas

Programado el módulo de proceso de alarma (OB3), y cableada las entradas de alarmas, es necesario un ajuste o parametrización previo. Este ajuste se realizará en el módulo de datos DB1, que como ya se dijo (apartado 6) posee valores predeterminados para diversas operaciones. Un listado del contenido de dicho módulo para el autómata S95U aparece a continuación:

```
DB1                                                    A:P62RDPST.S5D
LON=165 /4

PAG.    1
     0:      KC ='DB1 OBA: AI 0 ; OBI:     ';
    12:      KC ='   ; OBC: CAP N     CBP ';
    24:      KC ='N       ;#SL1: SLN 1  SF ';
    36:      KC ='DB2  DW0   EF DB3  DW0   ';
    48:      KC ='  KBE MB100       KBS MB1';
    60:      KC ='01       PGN 1  ;# SDP: N';
    72:      KC ='T 128 PBUS N ; TFB: OB13';
    84:      KC =' 100    ; #CLP: STW MW10';
    96:      KC ='2         CLK DB5  DW0   ';
   108:      KC ='  SET 3  01.10.91 12:00:';
   120:      KC ='00    OHS 000000:00:00  ';
   132:      KC ='  TIS 3 01.10. 12:00:00 ';
   144:      KC ='   STP Y SAV Y  CF 00   ';
   156:      KC =' ; #END ';
   160:
```

En este módulo de datos existe un apartado dedicado a la parametrización de las alarmas. Este apartado empieza con el texto **OBI:**. A continuación se define cómo se desea que responda el autómata ante la señal de alarma, y

seguidamente el nº de la alarma. Las diferentes respuestas aparecen en la tabla 4.3:

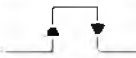
TIPOS DE FLANCOS	FLANCO	Caracteres a introducir en el DB1
Activación por flanco positivo: P		**ip**
Activación por flanco negativo: N		**in**
Activación por flanco positivo y negativo: PN		**ipn**
Activación por flanco negativo y positivo: NP		**inp**

Tabla 4.3: Parametrización del DB1 para tratamiento de alarmas

Si quisiéramos parametrizar la entrada de alarma 2 (E 34.2), que reaccione ante un flanco de subida (flanco positivo), el módulo de datos DB1 quedaría de la siguiente forma (en negrita):

```
DB1                                                    A:P62RDPST.S5D
LON=165 /4

PAG.     1
     0:      KC ='DB1 OBA: AI 0 ; OBI: ip ';
    12:      KC ='2  ; OBC: CAP N      CBP ';
    24:      KC ='N       ;#SL1: SLN 1  SF ';
    36:      KC ='DB2  DW0   EF DB3  DW0  ';
    48:      KC ='  KBE MB100       KBS MB1';
    60:      KC ='01       PGN 1  ;# SDP: N';
    72:      KC ='T 128 PBUS N ; TFB: OB13';
    84:      KC =' 100     ; #CLP: STW MW10';
    96:      KC ='2         CLK DB5  DW0   ';
   108:      KC ='  SET 3  01.10.91 12:00:';
   120:      KC ='00    OHS 000000:00:00  ';
   132:      KC ='  TIS 3 01.10. 12:00:00 ';
   144:      KC ='   STP Y SAV Y  CF 00   ';
   156:      KC =' ; #END ';
   160:
```

Varias interrupciones. Prioridades.

En el caso que hubiera varios módulos de alarmas (autómatas de gama más alta), los módulos de organización (OB) se ejecutarán con prioridad para aquellos de menor orden, es decir, el OB4 tendrá prioridad frente al OB6, en el caso que se presenten alarmas simultáneamente.

Una elaboración controlada por alarmas no puede ser interrumpida por una nueva alarma de proceso, ni tampoco por una elaboración controlada por tiempo como se verá en el siguiente apartado. Si se presentase una nueva alarma (interrupción), durante la elaboración de un proceso de alarma, éste no quedaría interrumpido y sería al final de su elaboración, si persiste la nueva alarma, cuando se procesaría la segunda alarma.

En determinadas ocasiones, cuando no se quiera que una parte del programa (elaboración cíclica o controlada por tiempo) se interrumpa por una interrupción por alarma, deberá de deshabilitarse la interrupción por alarmas (bloqueo de alarmas). Para este fin, el lenguaje STEP 5 posee dos instrucciones: Bloqueo de alarmas **(AS)** y liberalización de alarmas **(AF)**. Así que, anteponiendo la instrucción AS a la parte del programa que no se desea que se interrumpa y acabándolo con la instrucción AF, se conseguirá deshabilitar las alarmas.

7.4. PROCESAMIENTO CONTROLADO POR TIEMPO

Este tipo de procesamiento consiste en lanzar cada periodo de tiempo definido un programa que se encuentra realizado en un módulo de organización específico. Supóngase que el tiempo predefinido son 100 ms, esto quiere decir que el procesamiento cíclico se interrumpirá cada 100 ms, pasando a ejecutarse el programa controlado por tiempo, que tras su finalización devolverá el control al programa cíclico, justo donde se había interrumpido. Los módulos de organización destinados al procesamiento controlado por tiempo son los siguientes (el autómata S95U sólo dispone del módulo OB13):

OB10: Base de tiempos → 10ms
OB11: Base de tiempos → 20ms
OB12: Base de tiempos → 50ms
OB13: Base de tiempos → 100ms
OB14: Base de tiempos → 200ms
OB15: Base de tiempos → 500ms
OB16: Base de tiempos → 1s
OB17: Base de tiempos → 2s
OB18: Base de tiempos → 10s

Por tanto, en el autómata S95U, para que se produzca una elaboración controlada por tiempo se debe de programar el módulo de organización OB13.

El intervalo periódico de tiempo predefinido en el OB13 es de 100 ms, si bien, éste se puede modificar en pasos de 10 ms. Para ello, en el módulo de datos DB1, se dispone de un apartado denominado OB13:, donde justo después se indica la periodicidad de interrupción. Si por ejemplo, quisiéramos una interrupción controlada por tiempo cada 120 ms, se escribiría: **OB13: 120**.

También se puede modificar la periodicidad del proceso controlado por tiempo, por programación. La determinación de los períodos de reloj se deberán realizar en el programa de arranque (OB21 y OB22). Además, dicha programación debe ser realizada en un módulo funcional, por lo que se deberá de hacer una llamada al módulo funcional desde el programa de arranque, y en dicho módulo funcional se modificarán las palabras de datos del sistema que contienen el valor de los periodos de tiempo. Para el autómata S95U dicha palabra del sistema es la **BS97**.

El siguiente ejemplo cambiará la periodicidad a 150 ms (téngase en cuenta que la periodicidad será el valor introducido en la palabra del sistema multiplicado por 10ms).

```
L   KB  15   Carga en el acumulador la constante 15
T   BS  97   Transfiere el valor del acumulador a la palabra del
             sistema BS 97
```

Varias interrupciones. Prioridades.

Una interrupción controlada por tiempo no puede ser interrumpida por otra interrupción controlada por tiempo, pero si puede ser interrumpida por una alarma.

Si durante una interrupción por alarma se produce una de tiempo, seguirá la elaboración por alarma y cuando termine comenzará la elaboración controlada por tiempo.

Si el tiempo de ejecución de un proceso controlado por tiempo es superior a la periodicidad con el que se produce, el autómata pasa a STOP.

Para cuando se desee que una parte del programa controlado por tiempo no sea interrumpido por una alarma, se hará uso de las instrucciones bloqueo y liberalización de alarmas.

7.5. TRATAMIENTO DE LOS ERRORES

El sistema operativo puede detectar cuando se producen fallos en el procesador, errores en la elaboración de funciones de servicio y el efecto de una programación defectuosa.

Según sea el tipo de error, la CPU mandará el autómata a STOP o bien ejecutará un módulo de organización, en el que el usuario programará la reacción ante ese error o bien mandará el autómata a STOP.

En la siguiente tabla aparecen, los módulos de organización dedicados al tratamiento de errores. El autómata S95U sólo dispone de los módulos de organización de errores OB31 y OB34.

OB19: Llamada a un módulo no cargado
OB23: Retardo de acuse en acceso individual a la periferia
OB24: Retardo de acuse en la actualización de la imagen del proceso
OB25: Error de direccionamiento
OB26: Tiempo de vigilancia sobrepasado
OB27: Error de sustitución
OB28: Error general o retardo de acuse en el byte de entrada EB0
OB29: Código de operación ilegal
OB30: Parámetro ilegal
OB31: Disparo del tiempo de ciclo (NO PROGRAMABLE)
OB32: Error de transferencia en módulo de datos
OB33: Error en el procesamiento controlado por tiempo
OB34: Error en el procesamiento de un regulador (batería tampón)

8. MÓDULOS FUNCIONALES

La realización de funciones complejas y repetitivas es programada en los módulos funcionales. Éstos poseen las siguientes características:

- Programación en AWL. No permiten la programación en KOP o FUP.
- En ellos se puede utilizar el juego completo de instrucciones del lenguaje STEP 5.
- Aparte de ser numerados, poseen un nombre que los identifica.
- Son parametrizables. Esto les confiere la propiedad de ser utilizados varias veces con un mismo código, sólo hay que indicarles los parámetros de entrada y salida.

Un ejemplo que ilustra el uso de un módulo funcional, podría ser la suma de dos bytes. Si se quiere sumar dos entradas (bytes) cualesquiera y un número de veces grande, la forma de proceder sería:

Primero crear un FB donde se declaran los tres operandos formales: *sumando1*, *sumando2* y *resultado*, y justo después programar la suma.

Luego, sólo queda realizar tantas llamadas al FB como sumas queramos hacer. En cada llamada habrá que asignar el operando normal (entradas, marcas o salidas) el operando formal correspondiente creado en el FB. En la figura 4.11 aparece un esquema del ejemplo.

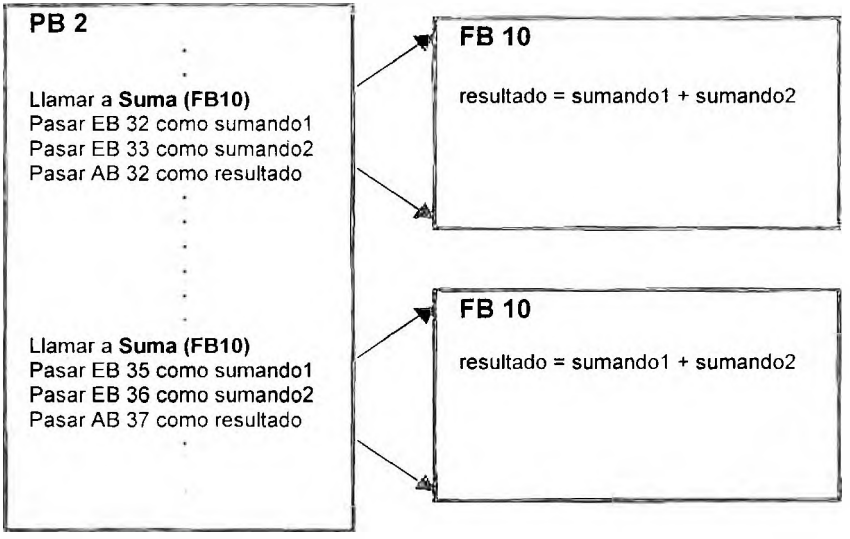


Figura 4.11: Ejemplo de uso de un FB para funciones repetitivas

En el capítulo V se detallará cómo se realizarán las declaraciones de los operandos formales en el módulo funcional y cómo se realizarán la llamada a dicho módulo.

Módulos funcionales de usuario estándar e integrados.

Los módulos funcionales en STEP 5 se suelen clasificar en tres tipos:

- Módulos Funcionales de Usuario: Son aquéllos que diseña el usuario para sus aplicaciones y en función de sus necesidades.
- Módulos Funcionales Estándar: Son módulos software que ofrece Siemens, ya programados y comprobados. Permiten realizar las funciones complejas de gran utilidad.

- Módulos Funcionales Estándar integrados: Son módulos al igual que los anteriores diseñados por Siemens, con la diferencia que se encuentran integrados en el procesador central del autómata. Estos módulos tienen asignados unos números, por lo que el usuario no podrá utilizar la misma numeración para diseñar sus propios FBs.

A continuación se recogen algunos de los módulos funcionales estándar que ofrece Siemens.

- Funciones digitales: FBs 1... 5, 10... 13 y 15... 21
- Funciones de aviso: FBs 51... 60 y 150
- Funciones de regulación: FBs 80, 104, 106, 108, 117 y 118
- Funciones secuenciales: FBs 70... 74
- Funciones de interface para tarjetas periféricas inteligentes.

Los siguientes son los módulos funcionales estándar que se encuentran integrados en el autómata S95U y que pueden ser usados por el usuario.

- **FB 240**: Conversor de código BCD a coma fija (binario).
- **FB 241**: Conversor coma fija (binario) a código BCD.
- **FB 242**: Multiplica dos números en coma fija.
- **FB 243**: Divide dos números en coma fija.
- **FB 250**: Lectura de un valor analógico.
- **FB 251**: Salida de un valor analógico.

9. DOCUMENTACIÓN DEL PROGRAMA

Como parte final de cualquier proyecto, se incluye una documentación del mismo de forma que facilite las funciones de mantenimiento del programa, y el uso por personal diferente al que lo diseñó.

El lenguaje de programación STEP 5 posee diferentes utilidades que ayudan a realizar una detallada documentación del programa. Aparte de estas utilidades, también permite documentar las instrucciones del programa en las tres formas de programación: AWL, KOP y FUP. Los comentarios que se realizan para los diferentes módulos se guardan en módulos de documentación aparte. Esto es así para que cuando se transfiera la información (programa) del aparato de programación al autómata sólo se cargue en el autómata el código ejecutable y no los comentarios.

Los módulos de comentarios se identifican con el mismo nombre que los módulos de organización, de programa, funcionales, de datos, etc., salvo

que se sustituye (operación realizada por el autómata de forma transparente al usuario) la letra B por la letra K. Para aclarar esto último, si por ejemplo los módulos PB 14 y FB21 poseen comentarios, los módulos de comentarios se denominarán respectivamente PK 14 y FK 21.

Dentro de las funciones relacionadas con la documentación del programa, se incluyen:

- Listado del programa en AWL, en FUP y en KOP.
- Listado de los módulos que comprende el programa.
- Listado de la estructura de los módulos.
- Listado de operandos simbólicos.
- Listado de referencias cruzadas.

9.1. LISTADO DEL PROGRAMA

En las siguientes páginas aparecen los listados de un programa que realiza la inversión del sentido de giro de un motor. En el listado se ha utilizado la referencia simbólica, es decir, se ha referenciado los operandos con nombres que los identifique.

En el listado del programa aparecen las referencias de los operandos, y justo al final se indican las correspondencias de los operandos con los símbolos utilizados.

En las siguientes páginas se presenta un ejemplo con los tres tipos de representaciones: AWL, FUP y KOP.

LISTADO DEL PROGRAMA INVERSIÓN DEL SENTIDO DE GIRO EN AWL

```
OB  1                                                    C:AUTODOST.S5D
LON=7

        SEGMENTO 1          0000
             :SPA PB  10
             :BE


PB  10                                                   C:AUTODOST.S5D
LON=21

        SEGMENTO 1          0000
             :UN  -P_PARO
             :U(
             :O   -P_DERCH
             :O   -C_DERCH
             :)
             :UN  -P_IZQ
             :=   -C_DERCH
             :***

E   32.0 = P_PARO                    PULSADOR DE PARO
E   32.1 = P_DERCH                   PULSADOR DE GIRO DERECHA
A   33.0 = C_DERCH                   CONTACTOR GIRO DERECHA
E   32.2 = P_IZQ                     PULSADOR DE GIRO IZQUIER


        SEGMENTO 2          0008
             :UN  -P_PARO
             :U(
             :O   -P_IZQ
             :O   -C_IZQ
             :)
             :UN  -P_DERCH
             :=   -C_IZQ
             :BE

E   32.0 = P_PARO                    PULSADOR DE PARO
E   32.2 = P_IZQ                     PULSADOR DE GIRO IZQUIER
A   33.1 = C_IZQ                     CONTACTOR GIRO IZQUIERDA
E   32.1 = P_DERCH                   PULSADOR DE GIRO DERECHA
```

```
!DEPARTAMENTO DE INGENIERÍA ELÉCTRICA  !UNIVERSIDAD DE CÁDIZ
!                                                              !
! REFERENCIA: PROYECTO 24A             !FECHA: 10 DICIEMBRE 1997 !
! INSTALACIÓN: ARRANQUE MOTORES        !MODIFICACIÓN: NUEVO      PAG.!
! NUMERO: 10                           !DESARROLLADO: AJGM         1!
```

LISTADO DEL PROGRAMA INVERSIÓN DEL SENTIDO DE GIRO EN FUP

```
OB 1                                                    C:AUTODOST.S5D
LON=7

        SEGMENTO 1          0000
                    +------+
                  +-! SPA  ! PB 10
                    +------+

                           :BE
```

```
PB 10                                                   C:AUTODOST.S5D
LON=21

        SEGMENTO 1          0000
                                +---+
                    -P_PARO   --O! & !
                      +---+     !   !
         -P_DERCH ---!>=1!      !   !
         -C_DERCH ---!   !------!   !
                      +---+     !   !     +-------+
                    -P_IZQ    --O!   !--+-! =     ! -C_DERCH
                                +---+     +-------+

E   32.0 = P_PARO                       PULSADOR DE PARO
E   32.1 = P_DERCH                      PULSADOR DE GIRO DERECHA
A   33.0 = C_DERCH                      CONTACTOR GIRO DERECHA
E   32.2 = P_IZQ                        PULSADOR DE GIRO IZQUIER


        SEGMENTO 2          0008
                                +---+
                    -P_PARO   --O! & !
                      +---+     !   !
         -P_IZQ   ---!>=1!      !   !
         -C_IZQ   ---!   !------!   !
                      +---+     !   !     +-------+
                    -P_DERCH  --O!   !--+-! =     ! -C_IZQ
                                +---+     +-------+

                                                  :BE

E   32.0 = P_PARO                       PULSADOR DE PARO
E   32.2 = P_IZQ                        PULSADOR DE GIRO IZQUIER
A   33.1 = C_IZQ                        CONTACTOR GIRO IZQUIERDA
E   32.1 = P_DERCH                      PULSADOR DE GIRO DERECHA
```

```
! DEPARTAMENTO DE INGENIERÍA ELÉCTRICA !UNIVERSIDAD DE CÁDIZ                  !
!                                                                            !
! REFERENCIA: PROYECTO 24A             !FECHA: 10 DICIEMBRE 1997             !
! INSTALACIÓN: ARRANQUE MOTORES        !MODIFICACIÓN: NUEVO            PAG.!
! NUMERO: 10                           !DESARROLLADO: AJGM                1!
```

LISTADO DEL PROGRAMA INVERSIÓN DEL SENTIDO DE GIRO EN KOP

```
OB  1                                                          C:AUTODOST.S5D
LON=/

SEGMENTO 1              0000
.
!                                                                     PB 10
!        !        . . .       . . .       . .       ;       .  ------   'A)-!
.
!
                                                                   :B.


PB  10                                                         ':AUTODOST.S5D
LON=^1

SEGMENTO 1              0000

!-P PARO   -P_DERC.H      IZQ                                       -C DERC'H
+-  -]/[---+---] [---+--- ]/[---+ --       +--------  -------  ---------
!          '    '
       !-C DERCH !
          . [---+

E    32.0 = P_PARO                    PULSADOR DE PARO
E    32.1 = P_DERCH                   PULSADOR DE GIRO DERECHA
A    33.0 = C_DERCH                   CONTACTOR GIRO DERECHA
E    32.2 = P_IZQ                     PULSADOR DE GIRO IZQUIER

SEGMENTO 2              0008

!-P_PARO   -P IZQ     -P_DERCH                                        .Q
+---]/[---+---] [---+---]/[---+--- ----  +---------+---------+--------  --    -!
:
       !-C_IZQ    !
       +- -] [---+                                                 :B.

E    3..0 = P_PARO                    PULSADOR DE PARO
E    32.2 = P_IZQ                     PULSADOR DE GIRO IZQUIER
A    33.1 = C_IZQ                     CONTACTOR GIRO IZQUIERDA
E    32.1 = P DERCH                   PULSADOR DE GIRO DERECHA
```

9.2. LISTA DE MÓDULOS Y ESTRUCTURA DEL PROGRAMA

El listado a continuación corresponde con la salida *"Estructura del programa"*, opción disponible en el entorno de programación de STEP 5. Esta salida da como información un listado de los módulos con su denominación y su longitud.

A continuación se presenta un ejemplo.

```
E S T R U C T U R A   D E   P R O G R A M A   C O N   D H

PAGIN   1

   PB   60 :-                          LONG.        15
   PB   61 :-                          LONG.        14
   PB   62 :-                          LONG.        25
   PB   63 :-                          LONG.        14
   OB    1 :-                          LONG.        11
   OB   21 :-                          LONG.         9
   OB   22 :-                          LONG.         9

  LONG.  : PB          68
  LONG.  : SB           0
  LONG.  : FB           0
  LONG.  : FX           0
  LONG.  : OB          29
  LONG.  : DB           0
  LONG.  : DX           0
  LONG.  :             97


E S T R U C T U R A   D E   P R O G R A M A   C O N   D B

PAGIN   2

+-OB  1-+=PB 60-
        I
        +-PB 61-
        I
        +-PB 62-
        I
        +-PB 63-

+-OB 21-

+-OB 22-
```

9.3. FICHERO SIMBÓLICO

Las asignaciones de los operandos con los símbolos se guardan en un fichero de texto con extensión .SEQ, el cuál se puede editar desde el entorno de programación. El siguiente es el fichero correspondiente con el ejemplo INVERSOR DEL SENTIDO DE GIRO expuesto en el apartado 9.1.

```
F CH. C:AUTOL IZ .SEQ

    OPERAND     SIMBOL    COMENTAR.
    E32.0       P_PARO    PULSADOR DE PARO
    E32.1       P_DERCH   PULSADOR DE GIRO DERECHA
    E32.2       P_IZQ     PULSADOR DE GIRO IZQUIER
    A33.0         DERCH   CONTACTOR GIRO DERECHA
    A33.1       C_IZQ     CONTACTOR GIRO IZQUIERDA
```

9.4. LISTAS DE CORRESPONDENCIAS

Una ventaja muy interesante de la que dispone el entorno de programación es la lista de correspondencias. Éstos son listados que relacionan los operandos con los módulos en los que se encuentran, indicando además el segmento en el que aparecen si el módulo tuviera varios segmentos.

En el ejemplo del inversor del sentido de giro se utilizó operandos simbólicos, sin embargo en las referencias cruzadas aparecen tanto los operandos normales como los símbolos. El siguiente es el listado correspondiente a dicho ejemplo.

```
  STA REFERENCIA: ENTRADAS
   PB    _0 : ELABORADO
   OB     1 : ELABORADO

 ISTA REFERENCIA: ENTRADAS
 E    32.0 -P PARO     PB 10      1 ,  2
 E    32.1 -P_DERCH    PB 10      1 ,  2
 E    32.2 -P IZQ      PB 10      1 ,  2
LISTA REFERENCIA: SALIDAS
 A    33.0 -C DERCH    PB 10      1*
 A    33.1 -C_IZQ      PB 10      2*
I ISTA REFERENCIA: MÓDULOS
 PB  10    -           OB  1      1
```

Capítulo 5

Elementos de programación. El Step 5

Contenido

1. INTRODUCCIÓN

Conocido el elemento con el que se va a trabajar, el autómata programable (en particular el S5-95U de Siemens); conocida la herramienta necesaria para la resolución de problemas de automatización secuenciales (redes de petri); se presenta el problema de traducir las soluciones obtenidas a algo que entienda el autómata (Lenguaje de programación Step 5), para lo que es necesario conocer los principios de programación, reglas generales a las que ceñirse, destacando fundamentalmente:

- La estructuración en módulos,
- las formas de representación y
- los ***elementos de programación, "juego de instrucciones del autómata programable"***.

A este último apartado, básico para trabajar con autómatas programables, se dedica este capitulo; basado lógicamente en el juego de instrucciones del lenguaje de programación Step 5 de Siemens.

Como ya es sabido, este conjunto de instrucciones se estructura en cuatro bloques fundamentales:

- Funciones binarias
- Funciones digitales
- Funciones de organización
- Funciones de sustitución

De estas se considerarán las funciones básicas, necesarias para realizar automatizaciones a un nivel adecuado a las pretensiones propuestas.

Además, cada una de ellas se desarrollarán atendiendo a las tres formas posibles de representación en el lenguaje de programación Step 5:

- Plano de contactos (Kop)
- Plano de funciones (Fup)
- Lista de instrucciones (Awl), mucho más potente que las anteriores, más cercano a un control numérico y básico en automatizaciones basadas en redes de petri.

Antes de describir estas funciones, es necesario considerar la simbología empleada en cada forma de representación; por ello se recomienda, antes de pasar a ver cada una de las funciones el habituarse a esa simbología:

2. SIMBOLOGIA BÁSICA EMPLEADA

2.1. PLANO DE CONTACTOS

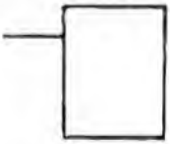
- **Contacto normalmente abierto.**

- **Contacto normalmente cerrado.**

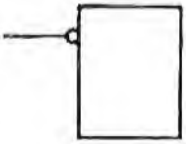
- **Bobina del relé o contactor.**

2.2. PLANO DE FUNCIONES

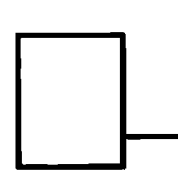
- **Entrada de un símbolo funcional.**



- **Entrada negada de un símbolo funcional.**



- **Salida de un símbolo funcional.**



2.3. LISTA DE INSTRUCCIONES

Dado que la representación simbólica no es gráfica sino mnemónica, se ha considerado conveniente describir con cada instrucción su representación asociada en lista de instrucciones. No obstante, se recomienda hojear el anexo final a este capítulo donde se representan las operaciones básicas en lista de instrucciones (Awl).

3. DESCRIPCIÓN DE LAS FUNCIONES BINARIAS BÁSICAS

Las funciones binarias permiten consultar y combinar los estados de operandos binarios (1 bit). A este tipo de funciones pertenecen, entre otras:

- Combinaciones binarias
- Funciones de memoria
- Funciones de tiempo
- Funciones de cómputo

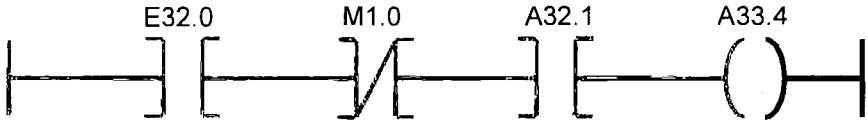
3.1. COMBINACIONES BINARIAS

Se describen la función "Y" y la función "O", así como las combinaciones de estas funciones. Como operandos se pueden utilizar:

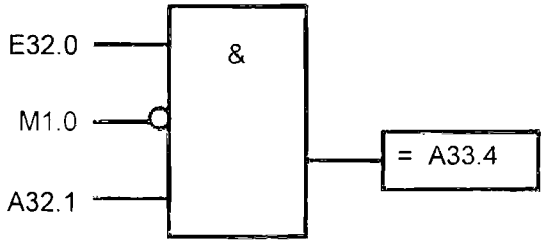
- Entradas (E32.1)
- Salidas (A33.5)
- Marcas (M2.3)
- Temporizadores (T2)
- Contadores (Z15)

3.1.1. COMBINACIÓN BINARIA "Y"

Como su nombre indica la salida es el producto lógico de todas las entradas.



En el plano de contactos la función Y se dibuja en forma de conexión serie; los operandos consultados se representan como símbolos de contacto.



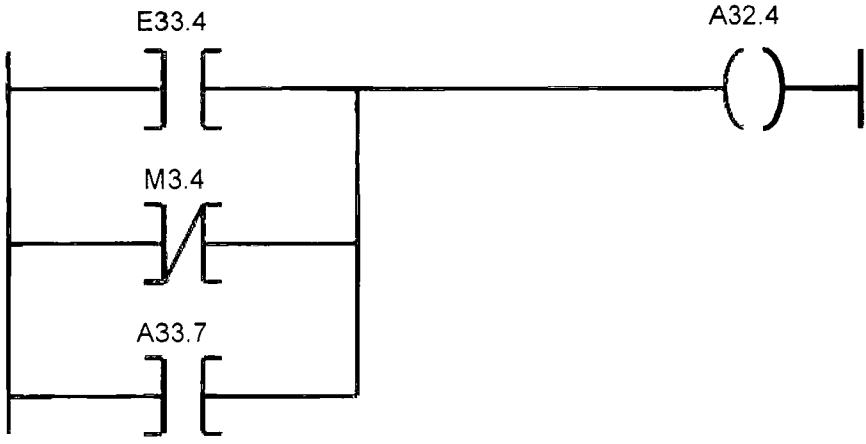
En el plano de funciones se representa gráficamente por un rectángulo con el signo "&"; las entradas a la función se realizan por la izquierda, mientras que la salida lo es por la derecha.

```
U     E 32.0
UN    M  1.0
U     A 32.1
=     A 33.4
```

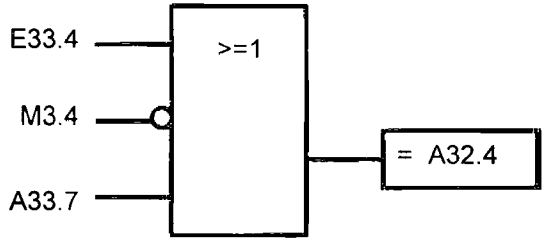
En lista de instrucciones la combinación Y se señaliza por "U" en caso de estados de consulta 1 y por "UN" en caso de estados de consulta 0.

3.1.2. COMBINACIÓN BINARIA "O"

Como su nombre indica la salida es la suma lógica de todas las entradas.



En el plano de contactos la función O se dibuja en forma de conexión paralelo; los operandos consultados se representan como símbolos de contacto.



En el plano de funciones se representa gráficamente por un rectángulo con el signo ">=1"; las entradas a la función se realizan por la izquierda, mientras que la salida lo es por la derecha.

```
O     E 33.4
ON    M  3.4
O     A 33.7
=     A 32.4
```

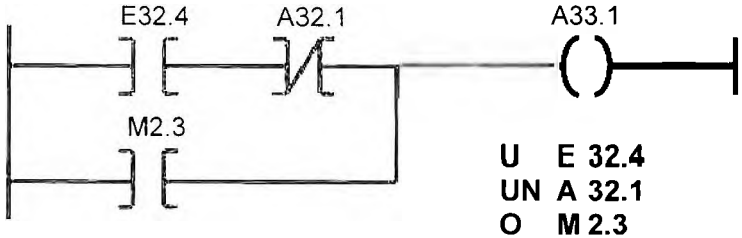
En lista de instrucciones la combinación O se señaliza por "O" en caso de estados de consulta 1 y por "ON" en caso de estados de consulta 0.

3.1.3. PRIORIDAD DE OPERACIONES BINARIAS

Al igual que en operaciones matemáticas, las dos operaciones lógicas que permite el autómata no presentan la misma prioridad; la función "Y" siempre presenta prioridad frente a la función "O". Lo que implica que si se requiere que se ejecute antes la función "O" es necesario utilizar paréntesis para conseguirlo

Se representan a continuación las dos posibilidades de prioridad, viéndose que realmente la importancia se presenta en lista de instrucciones, ya que en Kop y Fup (esta última no representada) es algo implícito al método gráfico.

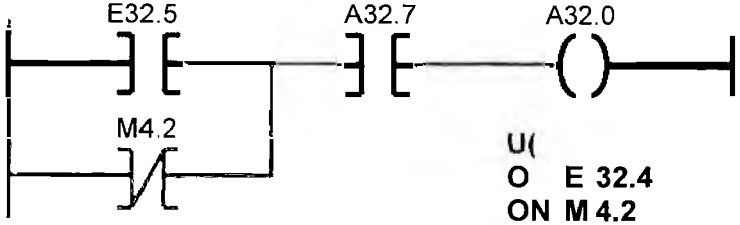
- **La función "Y" tiene prioridad frente a la "O":**




```
U   E 32.4
UN  A 32.1
O   M 2.3
=   A 33.1
```

Como se puede apreciar en este caso, la función "Y" se ha de ejecutar antes que la "O" y como tiene prioridad no es necesario el uso de paréntesis.

- **La función "Y" tiene prioridad frente a la "O":**

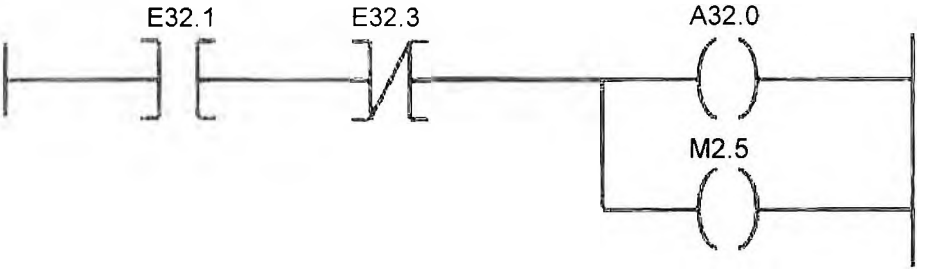



```
U(
O   E 32.4
ON  M 4.2
)
U   A 32.7
=   A 32.0
```

Como se puede apreciar en este caso, la función "O" se ha de ejecutar antes que la "Y", como la primera no tiene prioridad sobre la segunda es necesario el uso de paréntesis para conseguirla.

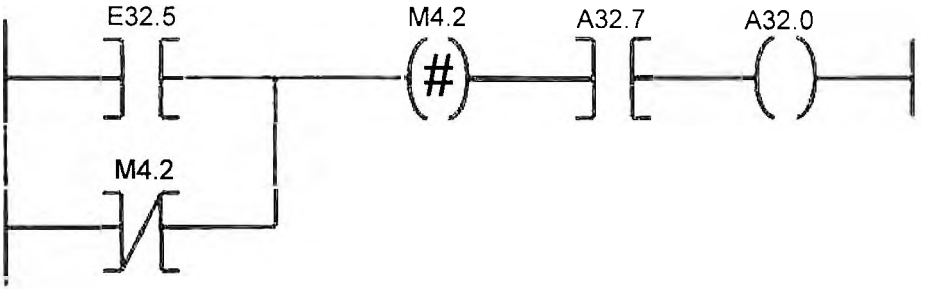
3.1.4. CONTROL DE VARIAS SALIDAS

Es posible activar varias salidas (y/o marcas) desde una misma combinación binaria, con la única condición de que las salidas (y/o marcas) han de estar en paralelo, no pudiendo haber ningún contacto intermedio que diferencie una salida de otra; es decir han de estar estrictamente en paralelo.



3.1.5. MARCAS INTERMEDIAS

En ciertos casos, una parte de la combinación binaria puede repetirse varias veces a lo largo de un programa sin necesidad de verse correspondida con una salida exterior del autómata; con objeto de reducir y aclarar la redacción del programa se puede asignar a esa parte de la combinación binaria el nombre de una marca, que de forma genérica se denomina "marca intermedia".



Lógicamente, las marcas intermedias son útiles cuando se trabaja en contactos o en funciones, ya que en lista de instrucciones carecen de sentido al tener la posibilidad de ubicar una marca donde se desee.

3.1.6. ELABORACIÓN DEL RESULTADO DE UNA CONSULTA (VKE)

Cuando se va a programar en lista de instrucciones con operaciones binarias (de un solo bit) es necesario tener en cuenta, que estas operaciones no las realiza el procesador del autómata directamente entre los operandos considerados, sino que se elaboran entre un operando y un registro denominado "resultado de una consulta" (VKE) para dar lugar a un "resultado de la combinación" (resultado parcial de la operación en juego); el cual se almacena también en el registro anterior.

Como se puede apreciar el VKE no es otra cosa que un acumulador binario, siendo siempre uno de los dos operandos al realizar una operación binaria.

¿Para qué vale? Para trabajar con operaciones condicionadas, que son todas aquellas que dependen del VKE. Son operaciones de asignación, activación y puesta a cero de entradas, salidas y marcas; arranque de temporizadores y contadores; saltos condicionados, llamadas condicionadas, etc.

Estas operaciones se realizarán siempre y cuando esté activado el resultado de la combinación (VKE→1); teniendo en cuenta que estas operaciones no modifican su valor y, por tanto, se pueden elaborar varias operaciones con un mismo resultado de combinación.

Si consideramos como ejemplo una parte de un programa y partimos de que las variables puestas en juego en esa parte tienen los valores mostrados en la tabla 5.1 (correspondientes a las imágenes de proceso de entradas y salidas, PEE y PEA).

Además del listado del programa se muestra el estado del VKE una vez realizado el resultado de la consulta, así como el nuevo estado de la salida una vez realizada la operación (se recuerda que este nuevo estado de la salida será almacenado en la imagen de proceso de salida PEA y no afectará a los elementos conectados a esa salida hasta que no se termine la elaboración del ciclo de instrucción).

VARIABLE	VALOR
VKE	0
E 32.0	1
E 32.1	0
E 33.0	1
E 33.1	0
A 32.0	0
A 32.1	0
A 32.3	0
A 33.0	0
M 1.0	0

Tabla 5.1: Estado previo de las señales puestas en juego.

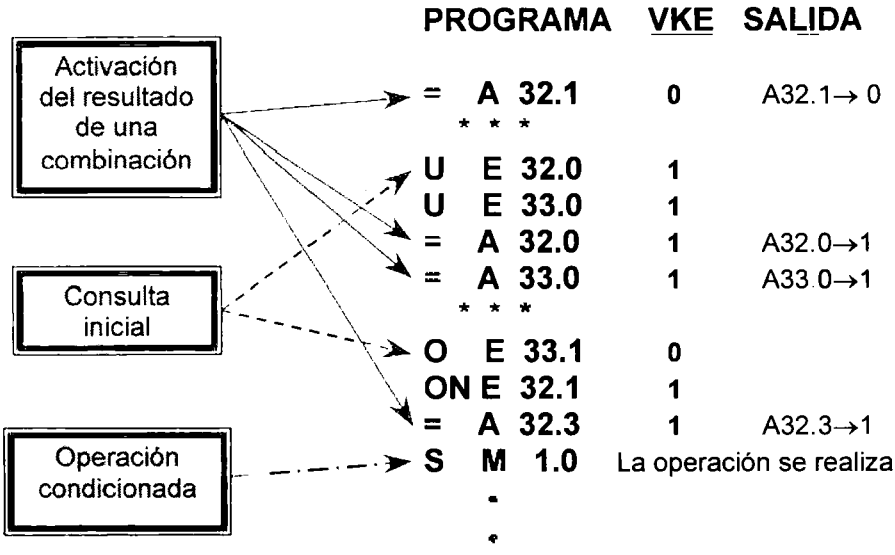


Figura 5.1: Ejemplo de utilización del resultado de una consulta (VKE).

Es necesario tener en cuenta la primera consulta o consulta inicial, que es aquella que se elabora tras la asignación del resultado de la combinación ó tras una operación condicionada. En este caso lo que se hace es cargar el VKE con el valor del operando puesto en juego, sin hacer ningún tipo de operación binaria; es decir, la primera instrucción tras la rotura de la secuencia del VKE es una operación de carga, aunque se represente en lista de instrucciones por una combinación "Y" u "O".

Por otra parte, se puede observar como las instrucciones condicionadas y las de asignación rompen la secuencia y la próxima operación realizada con el VKE es una consulta inicial.

3.1.7. CONSIDERACIONES SOBRE LAS ENTRADAS

Hasta ahora se ha considerado que a cada entrada del autómata se conecta un contacto físico normalmente abierto (correspondiente a los elementos de entrada al sistema, como aparatos de mando manual, de mando mecánico, contactos de aparatos de maniobra, sensores, detectores, etc). Dicho de otro modo, se considera que cuando no se activa desde el exterior una entrada, esta se encuentra a nivel bajo ("0", abierta); mientras que cuando se encuentre activa, la entrada del autómata estará a nivel alto ("1", cerrada).

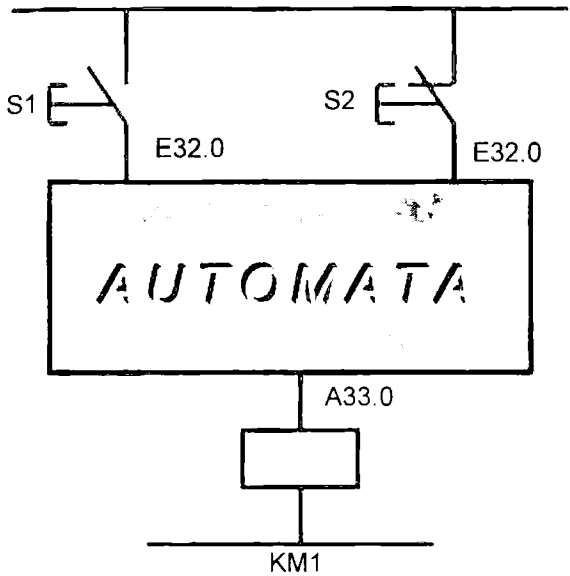
Sin embargo, no siempre se trabaja con lógica positiva, sino que en ciertos casos es necesario trabajar con entradas físicas normalmente cerradas (p.e. los contactos de seguridad de los relés de protección), o dicho de otro modo, trabajar con lógica negativa. En este caso cuando la entrada no este activada el autómata considerará un nivel alto ("1", cerrada), mientras que cuando se encuentre activa considerará un nivel bajo ("0", abierta).

La tabla 5.2 resume fácilmente las distintas opciones de trabajo:

TIPO CONTACTO EN LA ENTRADA	ESTADO DEL CONTACTO	ESTADO DE LA SEÑAL DE ENTRADA
NA	ACTUADO	1
NA	NO ACTUADO	0
NC	ACTUADO	0
NC	NO ACTUADO	1

Tabla 5.2: Opciones de funcionamiento de entradas en función del contacto utilizado.

Sea la aplicación representada en la figura 5.2, dos entradas que combinadas activan la salida correspondiente.



La combinación entre las entradas viene definida por el segmento de programa:

```
* * *
U E 32.0
U E 32.1
= A 33.0
* * *
```

Figura 5.2: Ejemplo de utilización de distintos tipos de contactos de entrada.

Las cuatro posibilidades diferentes que se presentan en la entrada del autómata (2^2) dan como resultado en la salida (KM1), los valores representados en la tabla 5.3.

S1 = E 32.0	**S2 = E 32.1**	**KM1 = A 33.0**
NO	NO	0
NO	SI	0
SI	NO	1
SI	SI	0

Tabla 5.3: Valor de la salida en función de las distintas combinaciones de las entradas.

Es necesario una consideración especial a las entradas de alarma (aquellas que activan los OB de alarma, en el S5-95U el OB3) ya que interesa que siempre sean normalmente abiertas ya que en caso contrario provocarían un salto continuo al módulo de alarma correspondiente.

3.2. FUNCIONES DE MEMORIA

Mediante las funciones de memoria es posible almacenar resultados parciales de la combinación formada por el procesador; es decir, es posible memorizar si se ha producido o no un evento.

Existen básicamente dos tipos de memorización:

- Memorización dinámica o asignación.
- Memorización estática o biestables.

Como operandos se pueden emplear:

- Entradas (E'33.6)
- Salidas (A32.5)
- Marcas (M3.6)

3.2.1. MEMORIZACIÓN DINÁMICA (ASIGNACIÓN)

Permite la puesta a "1" (SET- S) o la puesta a "0" (RESET-R) de un operando cuando el resultado de la combinación es igual a "1". Es decir, es una operación condicionada que asigna el resultado de la combinación (VKE) al operando, en caso de puesta a "1" y el valor contrario en caso de puesta a "0"

En el siguiente ejemplo se resumen las dos posibilidades de asignación, partiendo de dos operandos, E32.0 y E32.1 cuyos estados previos (en la imagen de proceso de entradas, PEE) son respectivamente "0" y "1"

		VKE	M 1.0
	U E 32.1	1	0
PUESTA A "1" →	**S M 1.0**	1	1
	U E 32.0	0	1
PUESTA A "0" →	**R M 1.0**	0	1

Figura 5.3: Ejemplo de tipos de asignación.

Como se puede apreciar, la primera instrucción de asignación se ejecutará por que el resultado de la combinación esta activado (condición cumplida); mientras que la segunda no se ejecutará por que el resultado de la combinación no ha sido activado. Recuérdese que las instrucciones

condicionadas rompen la secuencia y la siguiente instrucción de carga (U E 32.0) es una consulta inicial (carga del VKE con el contenido del operando); aunque la asignación se puede realizar a tantos operandos como se desee.

Lógicamente, tal como se ha podido ver en el ejemplo, estas funciones de asignación sólo son permitidas en lista de instrucciones (Awl).

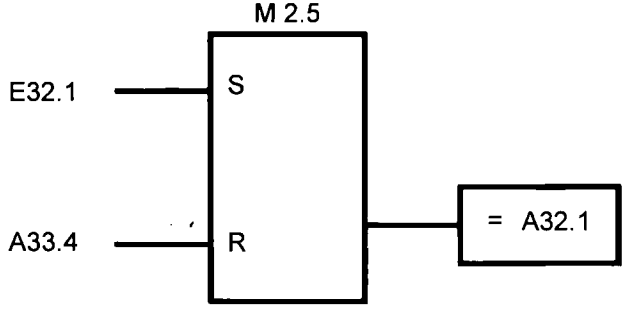
3.2.2. MEMORIZACIÓN ESTÁTICA (BIESTABLES)

Este tipo de funciones de memoria se emplea en plano de contactos y de funciones, teniendo la misma representación gráfica en ambas.

Son celdas de memorización mínimas, las cuales permiten almacenar el resultado de una combinación el tiempo que se desee, para posteriormente ser utilizado. Por esto, puede asignarse a cada celda el valor de un operando, independiente de la salida que active.

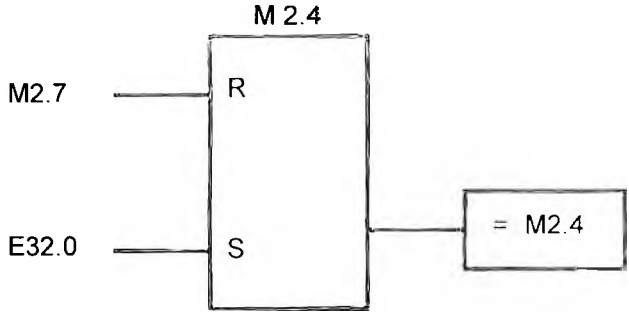
En función de la preferencia a la puesta cero ó activación se distinguen dos tipos de biestables:

- **Preferencia a la puesta a cero:**



Se activará la salida (A32.1) si el operando E32.1 esta activado y se pondrá a cero la salida (A32.1) si el operando A33.4 esta activado. En caso de que ambas condiciones se cumplan a la vez, tiene preferencia la entrada de Reset (R); es decir, la salida se pondrá a cero.

- **Preferencia a la activación:**



Se pondrá a cero la salida (M2.4) si el operando M2.7 esta activado y se activará la salida (M2.4) si el operando E32.0 esta activado. En caso de que ambas condiciones se cumplan a la vez, tiene preferencia la entrada de Set (S); es decir, la salida se activará.

3.2.3. ACTIVACIÓN DE ENTRADAS

Las funciones de memoria se pueden utilizar también para activar entradas. Esto es debido a que se trabaja con la imagen de proceso de entradas (PEE) registro interno al autómata que guarda los valores de las entradas y solo los actualiza al comienzo de cada ciclo de programa.

La activación y puesta a cero de las entradas, así como la asignación de los resultados de las combinaciones, se realiza de la misma forma que cuando los operandos son salidas o marcas.

Se presenta como ejemplo, la traducción a lista de instrucciones del biestable con preferencia a la activación mostrado anteriormente, con la única salvedad de cambiar el operando de salida por la entrada E32.5.

```
U  M  2.7
R  M  2.4
U  E  32.0
S  M  2.4
=  E  32.5
```

La activación de entradas se utiliza básicamente en la puesta en servicio, cuando se tienen que simular señales de entrada. Se realiza un módulo de programa inicial donde se activan y ponen a cero las entradas, el programa a continuación trabajará con estos valores de entradas simulados.

3.3. FUNCIONES DE TIEMPO

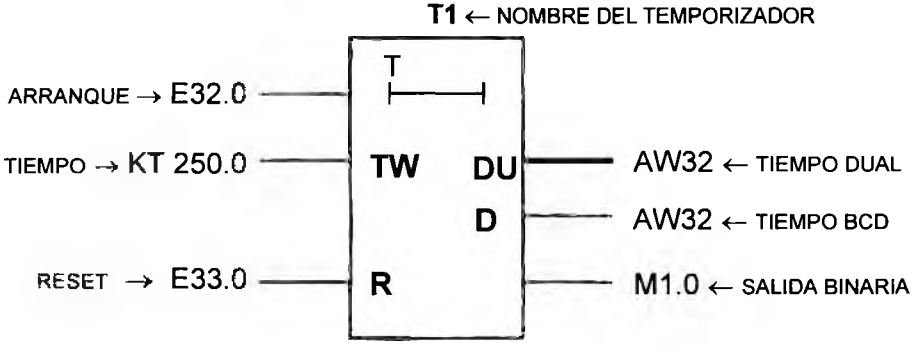
Las funciones de tiempo (temporizadores) tienen por misión aportar retardos calculados a una determinada acción.

Como operandos se pueden emplear:

- Entradas (E 32.2)
- Salidas (A33.0)
- Marcas (M5.7)

Aunque básicamente no existe diferencia en las tres formas de representación en cuanto a los temporizadores; son más fáciles de intuir de forma gráfica (Kop y Fup), por lo que se estudiarán primero así y después se particularizarán los pormenores de la representación en lista de instrucciones.

3.3.1. PLANO DE CONTACTOS Y DE FUNCIONES



- **Arranque de un temporizador.**

Un temporizador se inicializa cuando en su entrada de arranque se produce un flanco de subida. Es decir, el resultado de la combinación (VKE) ha de cambiar de "0" a "1", en el ciclo en cuestión; no activándose si se mantiene activo o si se encuentra a nivel bajo.

- **Duración de la temporización.**

La adjudicación de la duración de la temporización se realiza como asignación de constante (constante de tiempo – KT), cuyo valor puede variar entre 0 y 999. Tras este se indica la unidad de tiempo:

0 - Centésimas de segundo
1 - Décimas de segundo
2 - Segundos
3 - Decenas de segundos

La unidad adecuada dependerá de la aplicación, teniendo en cuenta que cuanto más bajo sea el valor de la unidad de tiempo, más precisión se tendrá en la temporización.

- **Puesta a cero del temporizador.**

La desactivación del temporizador se realiza cuando en la entrada de puesta a cero se active el operando correspondiente (el resultado de la combinación sea "1").

Con la puesta a cero de un temporizador concluye la elaboración del mismo; poniéndose el valor de tiempo a "cero". Mientras que esté activa esta señal el temporizador no puede realizar una nueva operación, aunque se active la señal de arranque.

- **Salida binaria del temporizador.**

Indicación binaria de finalización del periodo de temporizado, su valor depende específicamente del tipo de temporizador empleado.

- **Salidas digitales del temporizador**

Indicación digital (palabra de 16 bits) del estado de la temporización en binario (DU) o en BCD (DE).

3.3.2. LISTA DE INSTRUCCIONES

En lista de instrucciones se tiene la posibilidad de trabajar con cada parte del temporizador independientemente; por ejemplo, se puede arrancar en

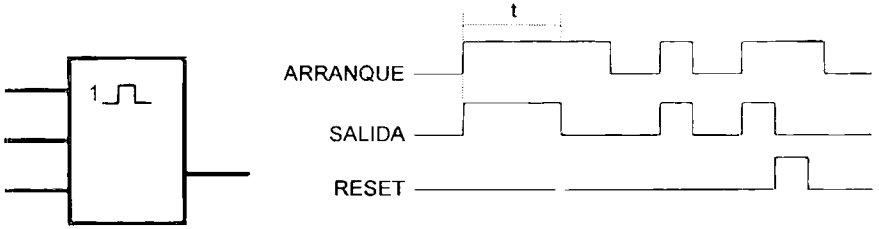
un módulo de los que consta el programa, poner a cero en otro y consultar su estado en un tercero.

```
U   E  32.0
L   KT 250.0   → ARRANQUE
SE  T 1

U   E  33.0
R   T 1        → PUESTA A CERO

L   T 1
T   AW 32      → CONSULTA DIGITAL
LC  T 1
T   AW 33

U   T 1
=   M  1.0     → CONSULTA BINARIA
```

Se presentan de forma agrupada, -para ver la equivalencia con la representación gráfica-, indicándose los bloques independientes de que consta; con la consideración especial de que el arranque se ejecuta con un flanco de subida en el resultado de la combinación (VKE), mientras que la puesta a cero se ejecuta por activación del VKE.

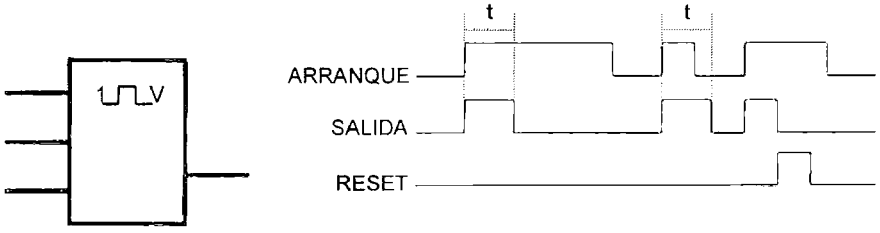
3.3.3. TIPOS DE TEMPORIZADORES

IMPULSO (SI):



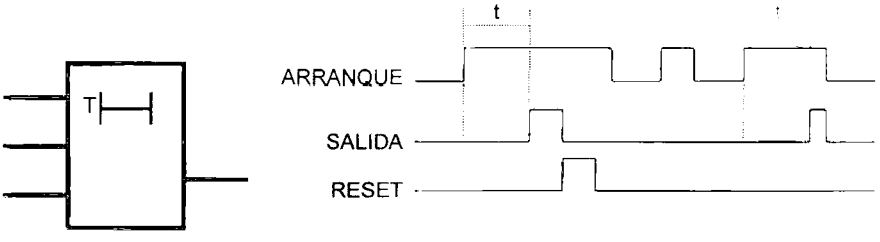
El temporizador da un impulso de duración prefijada cuando se arranca, siempre y cuando la señal de arranque se mantenga activada.

IMPULSO PROLONGADO (SV):



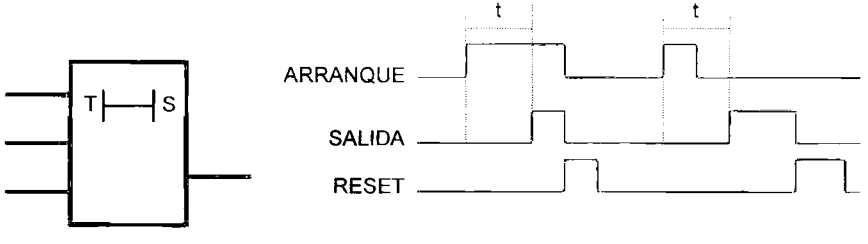
El temporizador da un impulso de duración prefijada cuando se arranca, siendo independiente del estado posterior de la señal de arranque.

RETARDO A LA CONEXIÓN (SE):



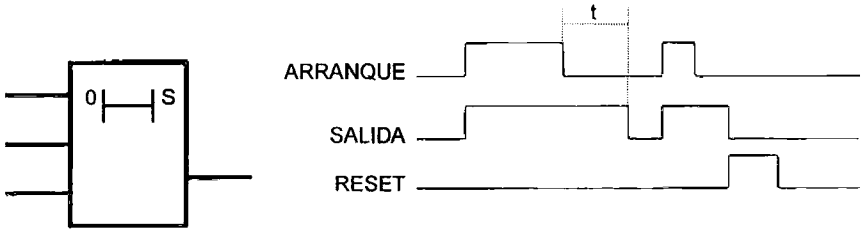
El temporizador activa su salida una vez finalizada la temporización, manteniéndola a nivel alto mientras la señal de arranque se mantenga activa.

RETARDO A LA CONEXIÓN MEMORIZADO (SS):



El temporizador activa su salida una vez finalizada la temporización, manteniéndola a nivel alto independientemente del estado de la señal de arranque, la desactivación se realiza única y exclusivamente por la activación de la señal de puesta a cero.

RETARDO A LA DESCONEXIÓN (SA):



El temporizador activa su salida una vez se arranca y la desactiva transcurrido un tiempo prefijado después de que se desactive la señal de arranque.

3.4. FUNCIONES DE CÓMPUTO

Las funciones de cómputo (contadores) tienen por misión el contaje de eventos (tanto ascendente como descendente) en una determinada acción.

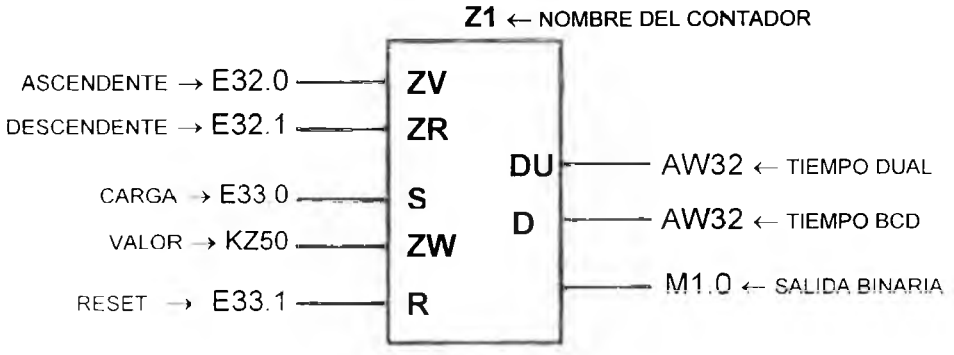
El campo del contador esta limitado a tres décadas (0 a 999).

Como operandos se pueden emplear·

- Entradas (E 32.2)
- Salidas (A33.0)
- Marcas (M5.7)

Aunque básicamente no existe diferencia en las tres formas de representación en cuanto a los contadores; son más fáciles de intuir de forma gráfica (Kop y Fup), por lo que se estudiarán primero así y después se particularizarán los pormenores de la representación en lista de instrucciones.

3.4.1. PLANO DE CONTACTOS Y DE FUNCIONES



- **Cuenta ascendente de un contador.**

El contador incrementará en uno su cuenta cuando en la entrada de cuenta ascendente (ZV) se produce un flanco de subida. Es decir, el resultado de la combinación (VKE) ha de cambiar de "0" a "1" en el ciclo en cuestión; no activándose si se mantiene activo o si se encuentra a nivel bajo.

Una vez la cuenta alcance su valor límite superior (999), ya no se incrementará más; no haciendo ningún efecto el cambio del estado de la señal en la entrada de cuenta hacia delante.

- **Cuenta descendente de un contador.**

El contador decrementará en uno su cuenta cuando en la entrada de cuenta descendente (ZR) se produce un flanco de subida. Es decir, el resultado de la combinación (VKE) ha de cambiar de "0" a "1" en el ciclo en cuestión; no activándose si se mantiene activo o si se encuentra a nivel bajo.

Una vez la cuenta alcance su valor limite inferior (0), ya no se decrementará más; no haciendo ningún efecto el cambio del estado de la señal en la entrada de cuenta hacia atrás; no se produce la cuenta con valores negativos.

- **Inicialización del contador a un valor predeterminado.**

El contador carga el valor presente en la entrada de la constante de inicio de cuenta (ZW) cuando se produce un flanco de subida en la entrada de activación del contador (S). . Es decir, el resultado de la combinación (VKE)

ha de cambiar de "0" a "1" en el ciclo en cuestión; no activándose si se mantiene activo o si se encuentra a nivel bajo.

La constante de carga puede variar entre los límites máximos de conteo, 0 y 999, no pudiendo ser nunca negativa.

- **Puesta a cero del contador.**

La inicialización de la cuenta se realiza cuando en la entrada de puesta a cero se active el operando correspondiente (el resultado de la combinación sea "1").

Con la puesta a cero de un contador, su valor de conteo pasa a ser nulo ("0"). Mientras que este activa esta señal el contador no puede realizar una nueva operación, aunque se active la señal de arranque.

- **Salida binaria.**

Indicación binaria de cuenta en el contador; es decir, esta salida estará activada cuando la cuenta sea distinta de cero, permaneciendo inactiva en caso contrario.

- **Salidas digitales del temporizador.**

Indicación digital (palabra de 16 bits) del estado del contador en binario (DU) o en BCD (DE).

3.4.2. LISTA DE INSTRUCCIONES

En lista de instrucciones se tiene la posibilidad de trabajar con cada parte del contador independientemente; por ejemplo, se puede incrementar o decrementar en un módulo de los que consta el programa, poner a cero en otro y consultar su estado en un tercero.

Se presentan de forma agrupada, -para ver la equivalencia con la representación gráfica-, indicándose los bloques independientes de que consta; con la consideración especial de que la cuenta ascendente, la cuenta descendente y la carga se ejecutan con un flanco de subida en el resultado de la combinación (VKE), mientras que la puesta a cero se ejecuta por activación del VKE.

```
U   E  32.0
ZV  Z 1        → CUENTA ASCENDENTE

U   E  32.1
ZR  Z 1        → CUENTA DESCENDENTE

U   E  33.0
L   KZ  50     → ACTIVACION DEL CONTADOR
S   Z 1

U   E  33.1
R   Z 1        → PUESTA A CERO

L   T 1
T   AW  32     → CONSULTA DIGITAL
LC  T 1
T   AW  33

U   T 1
=   M   1.0    › CONSULTA BINARIA
```

4. DESCRIPCIÓN DE LAS FUNCIONES DIGITALES BÁSICAS

Las funciones digitales permiten consultar, combinar y operar los estados de operandos digitales; estos operandos se pueden elaborar en forma de bytes (8 bits) o de palabras (16 bits). A este tipo de funciones pertenecen, entre otras:

- Funciones de carga y transferencia
- Funciones de comparación

4.1. FUNCIONES DE CARGA Y TRANSFERENCIA

Mediante las funciones de carga y transferencia, se tiene la posibilidad de intercambiar información entre los módulos de entrada y salida, la imagen de proceso de entradas y salidas (PEE y PEA), marcas y memoria de datos; así como con las elaboraciones de los temporizadores y contadores.

Este intercambio de información no se realiza de forma directa, sino que se realiza a través de un acumulador; de forma que cuando el flujo de información va hacia el acumulador la función es de carga y cuando procede de este la función es de transferencia.

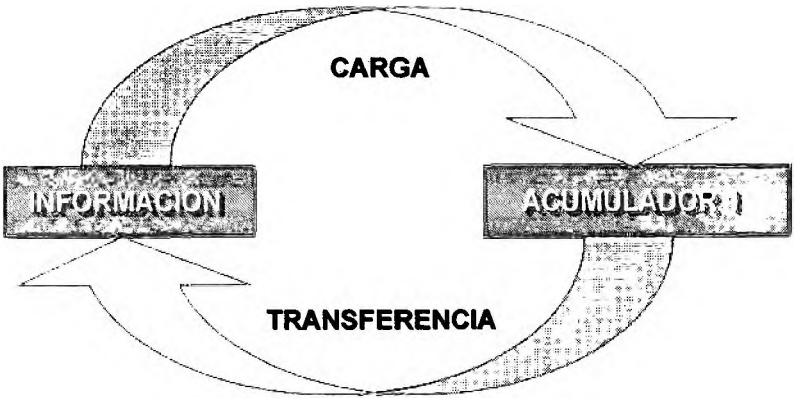


Figura 5.4: Flujo de información sobre el acumulador 1.

En muchas funciones se necesitan dos registros internos del procesador. Así, además del acumulador 1 (AKKU1), que se utiliza como acumulador principal para todas las funciones digitales; existe un acumulador 2 (AKKU2), donde se almacenan los valores digitales que han de combinarse de alguna manera con el valor del acumulador 1.

Ambos acumuladores son registros internos de 16 bits de capacidad.

4.1.1. FUNCIONES DE CARGA

Mediante la operación de carga (L), se almacena en el acumulador 1 las informaciones de las áreas de operando de:

- Entradas (E32.1)
- Salidas (A33.5)
- Marcas (M2.4)
- Temporizadores (T1)
- Contadores (Z3)
- Datos (DW25)
- Periferia de proceso (PW10)
- Constantes (en diversas representaciones)

Las operaciones de carga se ejecutan independientemente de los resultados de la combinación (VKE) y con independencia de los indicadores. Ni el resultado de la combinación ni los indicadores se modifican por la ejecución de estas operaciones.

Las funciones de carga modifican adicionalmente el contenido del acumulador 2. Ya que este registro recibe el valor anterior presente en el acumulador 1, con lo que se pierde el contenido que presentaba el acumulador 2.

- ***CARGA DE BYTES:***

L EB 32	←	ENTRADA
L AB 33	←	SALIDA
L MB 1	←	MARCA
L PY 10	←	PERIFERIA DE PROCESO
L DR 25	←	BYTE DERECHO PALABRA DE DATOS
L DL 25	←	BYTE IZQUIERDO PALABRA DE DATOS

Al cargarse un operando de un byte de amplitud, la información pasa al acumulador 1, ajustada a la derecha (se ocupan los bits nº 0 a 7); los restantes bits del acumulador (bits nº 8 a 15) se ponen a cero.

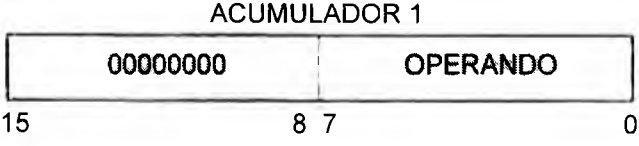


Figura 5.5: Ubicación de información en el AKKU1 en una operación de carga de un byte.

- ***CARGA DE PALABRAS:***

L EW 32	←	ENTRADA
L AW 33	←	SALIDA
L MW 1	←	MARCA
L PW 10	←	PERIFERIA DE PROCESO
L DW 25	←	PALABRA DE DATOS

Los estados de la señal de los operandos se cargan sin modificación en el acumulador 1.

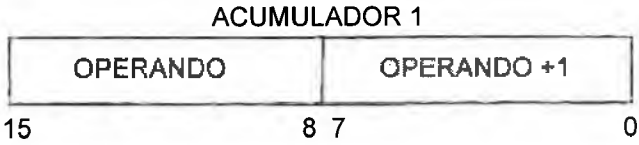


Figura 5.6: Ubicación de información en el AKKU1 en una operación de carga de una palabra.

- ***CARGA DE CONSTANTES:***

L KB 255	←	BYTE
L KF 32767	←	PALABRA
L KM 1100010	←	PALABRA BINARIA
L KH F107	←	PALABRA HEXADECIMAL
L KY 255,1	←	DOS BYTES
L KC XY	←	DOS CARACTERES ASCII
L KT 10.1	←	CONSTANTE TEMPORIZADOR
L KZ 25	←	CONSTANTE CONTADOR

Si se ha de realizar la carga del acumulador 1 con un valor fijo, se tiene la posibilidad de ubicar este valor como constante en el programa. La representación de la constante depende del tipo de aplicación, pudiéndose emplear las más comunes.

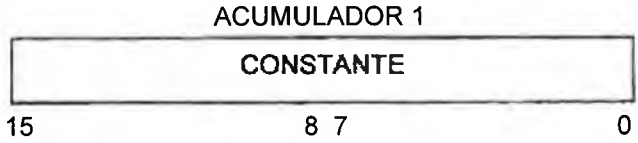


Figura 5.7: Ubicación de información en el AKKU1 en una operación de carga de constantes.

En el caso de carga del valor de temporización en una función de tiempo, se carga el valor en BCD en los 12 bits menos significativos del acumulador, reservándose los cuatro últimos para la base de tiempo.

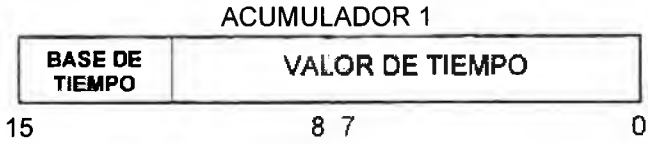


Figura 5.8: Ubicación de información en el AKKU1 en una operación de carga de la constante de un temporizador.

En el caso de carga del valor inicial del contador, se carga el valor en BCD en los 12 bits menos significativos del acumulador, poniéndose a cero los más significativos.

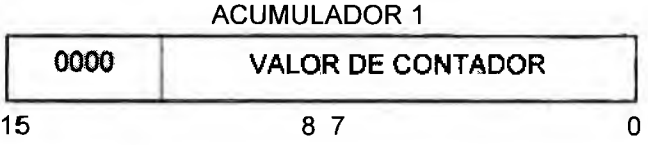


Figura 5.9: Ubicación de información en el AKKU1 en una operación de carga de la constante de un contador.

4.1.2. FUNCIONES DE TRANSFERENCIA

Mediante la operación de transferencia (T), se almacena el contenido del acumulador 1 en las áreas de operando de:

- Entradas (E32.1)
- Salidas (A33.5)
- Marcas (M2.4)
- Datos (DW25)
- Periferia de proceso (PW10)

Las operaciones de carga se ejecutan independientemente de los resultados de la combinación (VKE) y con independencia de los indicadores. Ni el resultado de la combinación ni los indicadores se modifican por la ejecución de estas operaciones.
El contenido del acumulador 1 no se modifica por la transferencia, permaneciendo inalterado aunque se realicen múltiples transferencias.

- ***TRANSFERENCIA DE BYTES:***

T EB 32	←	ENTRADA
T AB 33	←	SALIDA
T MB 1	←	MARCA
T PY 10	←	PERIFERIA DE PROCESO
T DR 25	←	BYTE DERECHO PALABRA DE DATOS
T DL 25	←	BYTE IZQUIERDO PALABRA DE DATOS

Se transfieren los ocho bits menos significativos del acumulador. Puesto que el autómata trabaja con un formato de datos de palabras (16 bits), es

necesario, en ciertos casos, la indicación en la transferencia, de "byte derecho" o "byte izquierdo". La transferencia se realizará así a este byte; el resto de la palabra de datos permanecerá inalterado

- Transferencia del "byte derecho" ("bajo" o "menos significativo"):

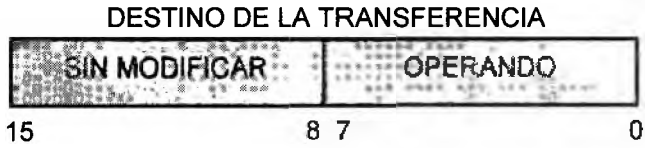


Figura 5.10: Ubicación de información en una operación de transferencia del byte derecho.

- Transferencia del "byte izquierdo" ("alto" o "más significativo"):

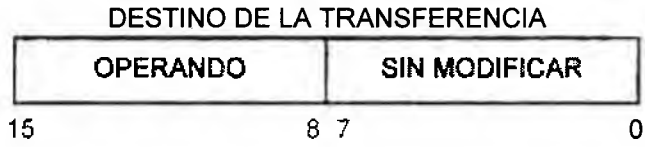


Figura 5.11: Ubicación de información en una operación de transferencia del byte izquierdo.

- ***TRANSFERENCIA DE PALABRAS:***

T EW 32 ← ENTRADA
T AW 33 ← SALIDA
T MW 1 ← MARCA
T PW 10 ← PERIFERIA DE PROCESO
T DW 25 ← PALABRA DE DATOS

Como los operandos de entrada, salida, marca y periferia de proceso poseen una longitud de un byte, cuando se transfieren palabras desde el acumulador 1 (16 bits) se carga el "byte menos significativo" del acumulador 1 en el operando indicado, y el "byte más significativo" en el operando siguiente.

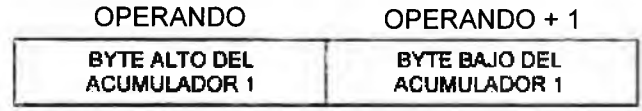


Figura 5.12: Ubicación de información en una operación de transferencia de palabras.

4.2. FUNCIONES DE COMPARACIÓN

Mediante estas funciones se comparan entre sí los valores digitales contenidos en los acumuladores 1 y 2. Como resultado se actúan, el resultado binario de la combinación(VKE) y los indicadores; la consulta del resultado de la comparación se realiza con funciones binarias o mediante funciones de salto.

Los valores que se encuentran en los acumuladores se interpretan como números de coma fija de 16 bits y se comparan entre sí, de acuerdo con ello.

4.2.1. ELABORACIÓN DE UNA FUNCIÓN DE COMPARACIÓN

Se comparan entre sí los contenidos del acumulador 1 y el acumulador 2; se deberán, por tanto, antes de realizar la función de comparación, cargar los operandos a comparar en los acumuladores.

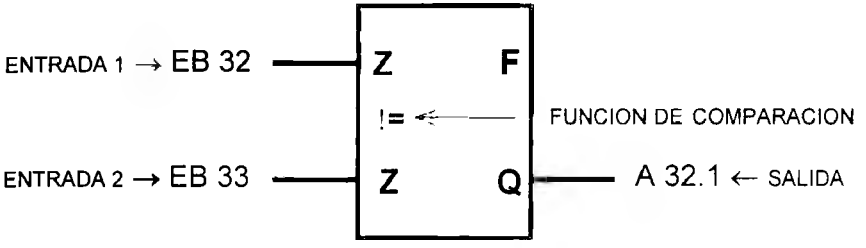
Figura 5.13: Formato de una función de comparación.

La ejecución de la función de comparación es independiente del resultado de la combinación (VKE). En la elaboración, las funciones de comparación se ejecutan siempre.

El resultado de la comparación es binario. Cuando es "1", la comparación se cumple, mientras que "0" indica que la comparación no se cumple. Este resultado de la comparación se encuentra disponible para la continuación de la elaboración de la combinación.

4.2.2. REPRESENTACIÓN DE LAS FUNCIONES DE COMPARACIÓN

PLANO DE CONTACTOS Y DE FUNCIONES:



LISTA DE INSTRUCCIONES:

```
L  EB  32     ← OPERANDO 1 (BYTE O PALABRA)
L  EB  33     ← OPERANDO 2 (BYTE O PALABRA)
!=  F         ← FUNCION
=  A  33.0    ← SALIDA BINARIA
```

4.2.3. TIPOS DE FUNCIONES DE COMPARACIÓN

COMPARACIÓN IGUAL (! = F):

La comparación se cumple cuando:

Z1=Z2 o acumulador 2 = acumulador 1

COMPARACIÓN DISTINTO (>< F):

La comparación se cumple cuando:

Z1><Z2 o acumulador 2 >< acumulador 1

COMPARACIÓN MAYOR (> F):

La comparación se cumple cuando:

Z1>Z2 o acumulador 2 > acumulador 1

COMPARACIÓN MAYOR O IGUAL (> = F):

La comparación se cumple cuando:

Z1>=Z2 o acumulador 2 >= acumulador 1

COMPARACIÓN MENOR (< F):

La comparación se cumple cuando:

Z1<Z2 o acumulador 2 < acumulador 1

COMPARACIÓN MENOR O IGUAL (<= F):

La comparación se cumple cuando:

Z1<=Z2 o acumulador 2 <= acumulador 1

5. DESCRIPCIÓN DE LAS FUNCIONES DE ORGANIZACIÓN BÁSICAS

Mediante las funciones de organización se puede controlar el orden de elaboración de los programas, ya sea mediante llamadas a módulos, o por medio de saltos dentro de los módulos. A las funciones de organización pertenecen también instrucciones que manipulan el contenido del acumulador 1, como por ejemplo: desplazamientos, incrementos o decrementos. Además, se pueden realizar también las llamadas elaboraciones indexadas de las instrucciones; es decir, se pueden modificar el software de las direcciones de los operandos en las instrucciones.

De todas estas funciones de organización, se han considerado como básicas las funciones de módulos, imprescindibles para la elaboración de un programa en Step-5.

5.1. FUNCIONES DE MÓDULOS

Mediante las funciones de módulos se estructura el programa de usuario en secciones independientes. Se adjudica a cada sección una determinada función, consiguiéndose así dividir el programa en partes comprensibles, con comunicaciones simples a otras partes del programa. Estas ventajas en la programación, tienen su efecto también en la puesta en servicio y en la simplicidad de las pruebas a realizar.

La ejecución de módulos se puede realizar:

- Siempre, de forma independiente a la elaboración del programa.
- De forma dependiente del resultado de una combinación.
- De forma dependiente de algún indicador del sistema.

Los módulos a emplear con estas operaciones pueden ser:

- Módulos de organización
- Módulos de programa
- Módulos de función
- Módulos de paso

Las dos funciones básicas de módulos son:

- Llamada a un módulo
- Finalización de un módulo

5.1.1. LLAMADA A UN MÓDULO

Para que un módulo pueda ejecutarse es necesario llamar a ese módulo, esto se realiza mediante operaciones de salto, que se encuentran en el programa en forma absoluta o dependientes del resultado de la combinación.

SALTO ABSOLUTO (SPA).

Con esta operación se abandona la elaboración lineal del programa en el módulo correspondiente; esta continua en el módulo indicado. La operación es independiente del resultado de la combinación y no influye sobre él.

Sin embargo, el resultado de la combinación se guarda del módulo anterior y se puede valorar con operaciones condicionadas.

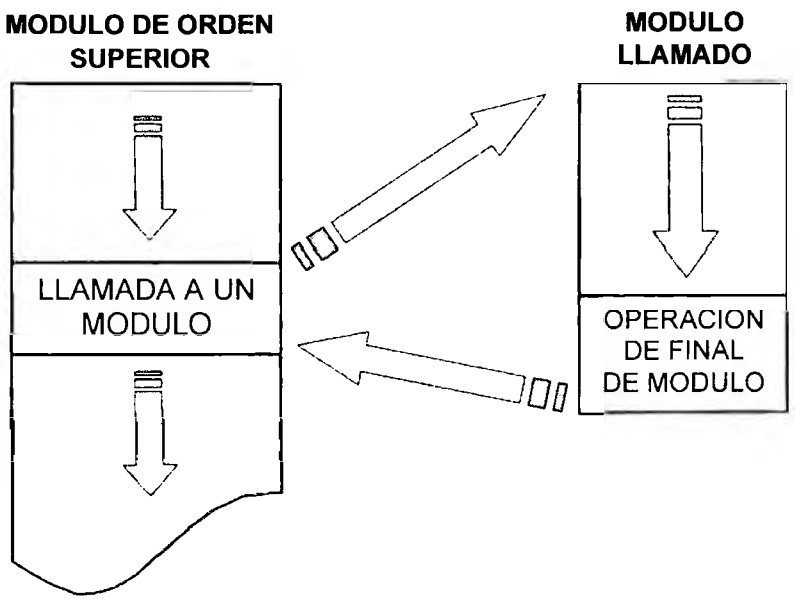


Figura 5.14: Estructuración de la programación mediante llamadas a módulos.

SALTO CONDICIONADO (SPB).

Mediante esta operación se abandona, en el módulo correspondiente, la ejecución lineal del programa, siempre y cuando se active el resultado de la combinación (VKE a " 1"), y se continúa en el módulo indicado.

En caso de no activación del resultado de la combinación se continúa la ejecución lineal del programa y se activa el resultado de la combinación.

5.1.2. FINALIZACIÓN DE UN MÓDULO

Las operaciones de finalización de módulos concluyen la elaboración de un módulo en forma absoluta o dependientes del resultado de la combinación. A continuación se proseguirá la ejecución del programa en el módulo de orden superior.

FIN DE MÓDULO (BE).

Mediante esta operación se concluye el módulo que se esté ejecutando en ese momento; se produce un retorno al módulo ejecutado previamente, en el cual se encuentra la llamada al módulo que ahora finaliza. La ejecución del programa continua con la instrucción posterior a la de llamada al módulo.

La ejecución de la operación de fin de módulo es independiente del resultado de la combinación, no modificándose este y conservándose al módulo que realiza la llamada.

Una combinación binaria comenzada en el módulo llamado no puede concluirse en el módulo de orden superior; por lo que la instrucción no condicionada inmediata a la llamada de un módulo es siempre una consulta inicial.

Esta instrucción es siempre la última instrucción de un módulo.

FIN ABSOLUTO DE MÓDULO (BEA).

Esta operación es idéntica a la anterior (BE), con las mismas características y condiciones. Pero con la diferencia de que cuando en un módulo existen saltos condicionados a distintas partes de ese mismo módulo que implican terminaciones distintas en función de esos saltos, la instrucción de fin de módulo (BE) solo es posible escribirla una vez en el módulo (la última), mientras que la instrucción de fin absoluto de módulo (BEA) puede aparecer en más de una ocasión (en medio del módulo).

```
        •
        •
      SPB := PEPE  ← SALTO A LA ETIQUETA INDICADA
        •          ← ESTA PARTE DEL PROGRAMA SOLO
        •            SE EJECUTARA SI NO SE CUMPLE
        •            EL SALTO CONDICIONADO ANTERIOR
      BEA            FIN DE MÓDULO ABSOLUTO
PEPE:   •          ← ESTA PARTE DEL PROGRAMA SOLO
        •            SE EJECUTARA SI SE CUMPLE EL
        •            SALTO CONDICIONADO ANTERIOR
      BE             ULTIMA INSTRUCCIÓN DEL MÓDULO
```

Figura 5.15: Ejemplo de utilización de la instrucción de fin absoluto de módulo (BEA).

FIN CONDICIONADO DE MÓDULO (BEB).

Esta operación es similar a la instrucción de fin de módulo (BE), con las mismas características y condiciones; pero con la diferencia de que su ejecución depende del resultado de la combinación. Si se activa el resultado de la combinación (VKE a "1"), la operación se ejecuta y el módulo actual concluye; mientras que si no se activa continua la ejecución del módulo hasta encontrar la instrucción de fin de módulo (BE).

```
•
•
BEB      ← FIN DE MÓDULO CONDICIONADO
•        ← ESTA PARTE DEL PROGRAMA SOLO
•          SE EJECUTARA SI NO SE CUMPLIO LA
•          CONDICION DE SALTO ANTERIOR
BE       ← ULTIMA INSTRUCCIÓN DEL MÓDULO
```

Figura 5.16: Ejemplo de utilización de la instrucción de fin condicionado de módulo (BEB).

6. DESCRIPCIÓN DE LAS FUNCIONES DE SUSTITUCIÓN BÁSICAS

Las instrucciones de sustitución solamente se pueden programar dentro de los módulos de función (FBs). Las instrucciones de sustitución contienen solamente la función a ejecutar y una referencia (parámetro formal), dirigida a la lista de parámetros -e indicada por un signo igual antepuesto-, de la llamada a los módulos de función.

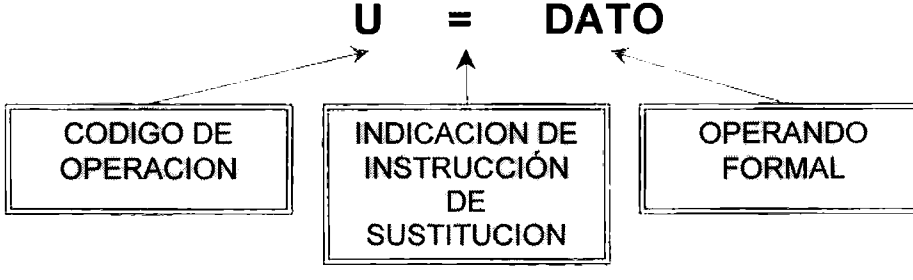


Figura 5.17: Estructura de una instrucción de sustitución.

En la ejecución de las instrucciones de sustitución el procesador sustituye el operando formal (la referencia) por el valor indicado en la lista de parámetros (operando actual) y ejecuta, a continuación la instrucción.

De esta forma se puede programar un módulo de función, que solamente se encuentre una vez en la memoria del autómata programable y se puede ejecutar las veces que se desee con distintos operandos.

El operando formal puede tener hasta cuatro caracteres de longitud; éste se define en la entrada del encabezamiento del módulo de función.

Las funciones de sustitución se dividen, como funciones básicas, en:

- Funciones binarias
- Funciones digitales
- Funciones de organización

Todas estas son similares a las descritas anteriormente, por lo que no se volverán a repetir; solamente se comentarán las particularidades concernientes a su empleo con operandos formales.

6.1. INSTRUCCIONES BINARIAS DE SUSTITUCIÓN

- **Funciones de memoria.**

Al contrario que con la puesta a cero de un operando binario (R A 32.1), en los operandos formales, se indica si se trata de una puesta a cero binaria o digital.

RB = xxxx ← PUESTA A CERO BINARIA
RD = xxxx ← PUESTA A CERO DIGITAL

- **Funciones de tiempo y cómputo.**

Además de la puesta a cero, que en este caso solo se considera instrucción con operando digital (RD); la variación se presenta en que existen instrucciones comunes a las funciones de tiempo y cómputo.

Así, de las cinco instrucciones que definían los tipos de temporizadores, las de impulso (SI) y de retardo a la conexión (SE) solo permiten como operando temporizadores.

SI = xxxx ← FUNCION IMPULSO
SE = xxxx ← FUNCION RETARDO A LA CONEXION

Las otras tres son válidas para definir el resto de tipos de temporizadores y para el control de contadores.

SVZ = xxxx ← IMPULSO PROLONGADO/CARGA CONTADOR
SSV = xxxx ← RETARDO A LA CONEXIÓN MEMORIZADO/ INCREMENTA CONTADOR
SAR = xxxx ← RETARDO A LA DESCONEXION/ DECREMENTA CONTADOR

Dependiendo del operando actual que sustituya al operando formal contenido en la instrucción de sustitución, la instrucción corresponderá a un temporizador o a un contador.

6.2. INSTRUCCIONES DIGITALES DE SUSTITUCIÓN

Las instrucciones digitales de sustitución no presentan ninguna variación destacable respecto a las instrucciones digitales comentadas anteriormente.

6.3. INSTRUCCIONES ORGANIZATIVAS DE SUSTITUCIÓN

A las instrucciones organizativas de sustitución pertenecen las operaciones:

B = mm ← ELABORACION DE PARÁMETRO DE MÓDULO
BI = mm ← ELABORACION INDIRECTA DE PARÁMETRO DE MÓDULO

Con la operación de elaboración de parámetro de módulo (B) se pueden llamar módulos que han sido dados como operandos actuales; el parámetro de módulo debe, en este caso, tener el tipo de parámetro "orden".

Con la operación de elaboración indirecta de parámetro de módulo (BI) se pueden ejecutar líneas de instrucciones (parámetros) de un módulo. Los

números de parámetros de módulos a elaborar habrán de almacenarse previamente en el acumulador 1.

Si se da un operando binario como parámetro de módulo, este se trata con la operación "Y"; en un operando digital se realiza la operación de carga (L). En caso de que el operando sea un módulo se ejecutará un salto absoluto a ese módulo; y, por último, los temporizadores y contadores se consideran como operandos binarios.

7. PROGRAMACIÓN DE MÓDULOS DE FUNCIÓN

Las instrucciones de sustitución sólo pueden ser empleadas en los módulos de función; estos son similares a los módulos de programa pero con las características siguientes:

- Aplicación del juego completo de instrucciones, cosa que el resto de módulos no permite.
- Pueden ser parametrizables, y por tanto adaptarse a cada aplicación particular.
- Solo es posible programar un módulo de función en lista de instrucciones (Awl), nunca en otro tipo de representación.
- Sin embargo, la llamada a un módulo de función si puede hacerse de forma gráfica; pudiéndose emplear, por tanto, en cualquiera de los tres tipos de representación.
- Cada módulo de función tiene un nombre identificativo, además de su número correspondiente.
- Existe gran cantidad de módulos de función estándares como producto de software prediseñado por el fabricante.

7.1. CATEGORIAS DE MÓDULOS DE FUNCIÓN

7.1.1. MÓDULOS DE FUNCIÓN SIN PARÁMETROS DE MÓDULOS

Son idénticos a los módulos de programa, tanto en su programación como en su utilización, salvo dos particularidades. La primera es que estos módulos de función vienen especificados por un nombre identificativo de ocho caracteres, el cual es necesario introducir a la hora de comenzar la programación del módulo.

La segunda es que los módulos de función permiten utilizar el juego completo de operaciones, incluyendo las denominadas operaciones específicas, solo permitidas en este tipo de módulos.

7.1.2. MÓDULOS DE FUNCIÓN CON PARÁMETROS DE MÓDULOS

Son similares a los anteriores; pero, en este caso, al disponer los módulos de parámetros formales (un máximo de 40, aunque se recomienda no superar 10), estos han de ser indicados al comienzo, antes de realizar la programación, en lo que se denomina parametrización de un módulo de función.

El proceso de parametrización de un módulo de función comienza por la asignación del nombre identificativo, de ocho caracteres como máximo:

NOMB: **EJEMPLO1**

A continuación se procede a la introducción de los parámetros (operandos formales) del módulo en cuestión:

DES: **ENTR** E/A/D/B/**T**/Z: **E** BI/BY/W/D: **BI**

El nombre de un parámetro de módulo puede tener, como máximo 4 caracteres de longitud y debe comenzar con una letra. Si no se indica ningún nombre, se concluye la entrada de la cabeza de módulo y se comienza la programación.

Como clase de parámetro de módulo se tiene:

E: ENTRADA
A: SALIDA
D: DATOS
B: LLAMADA A MÓDULO
T: TEMPORIZADOR
Z: CONTADOR

En la representación gráfica estos parámetros se colocan a la izquierda del símbolo de función, a excepción de los parámetros clase "A" que se colocan a la derecha.
El tipo de parámetro depende, lógicamente, de que clase de parámetro se considere. Así, en caso de entradas (E) o salidas (A) pueden ser:

BI - BIT (1 b)
BY - BYTE (8 b)
W - PALABRA (16 b)
D - PALABRA DOBLE (32 b)

Si la clase de parámetro es una llamada a un módulo (B), los tipos de parámetros permitidos son:

DB - MODULO DE DATOS
FB - MODULO DE FUNCION
PB - MODULO DE PROGRAMA
SB - MODULO DE PASO

Por último, si se trata de una constante (D) las posibilidades para el parámetro serán:

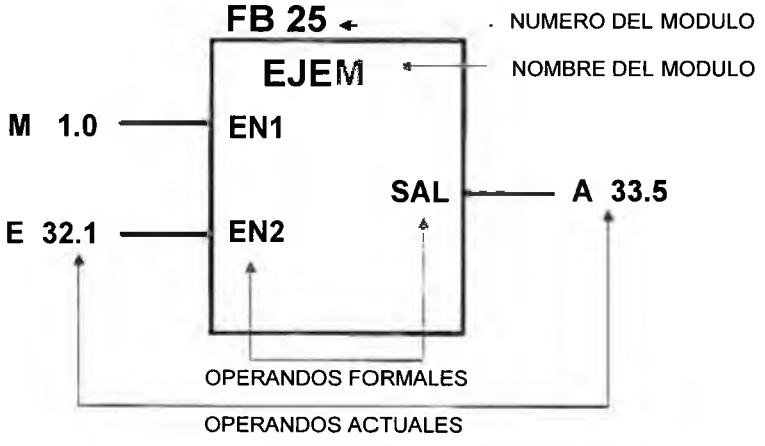
KM - BINARIA
KH - HEXADECIMAL
KY - DOS BYTES
KC - CARACTERES ASCII
KF - EN COMA FIJA
KT - DE TEMPORIZADOR
KZ - DE CONTADOR
KG - EN COMA FLOTANTE

Si la clase de parámetro es "T" o "Z" no se permite indicación de tipo adicional, solo se permite el operando.

7.2. PROCESAMIENTO DEL PROGRAMA EN MÓDULOS FUNCIONALES

El programa contenido en el módulo determina la función que se realizará durante su ejecución. Los operandos con los que tenga que ejecutarse, se indican cada vez que el módulo sea llamado, dando lugar a los denominados "operandos actuales", que sustituyen a los "operandos formales" del proceso de parametrización.

Cuando se emplea un módulo de funciones en el plano de contactos -o el de funciones- la representación es similar a una función:



En caso de representación en lista de instrucciones, la llamada a un módulo de funciones se realiza de la forma:

```
      : SPA FB 25
NOMB  : EJEMP
```

← LLAMADA AL MODULO

```
EN1   : M  1.0
EN2   : E  32.1
SAL   : A33.5
```

← DESIGNACION DE PARAMETROS ACTUALES

En la ejecución del programa, se sustituyen automáticamente los operandos formales del módulo funcional por los actuales, indicados en la llamada a ese módulo.

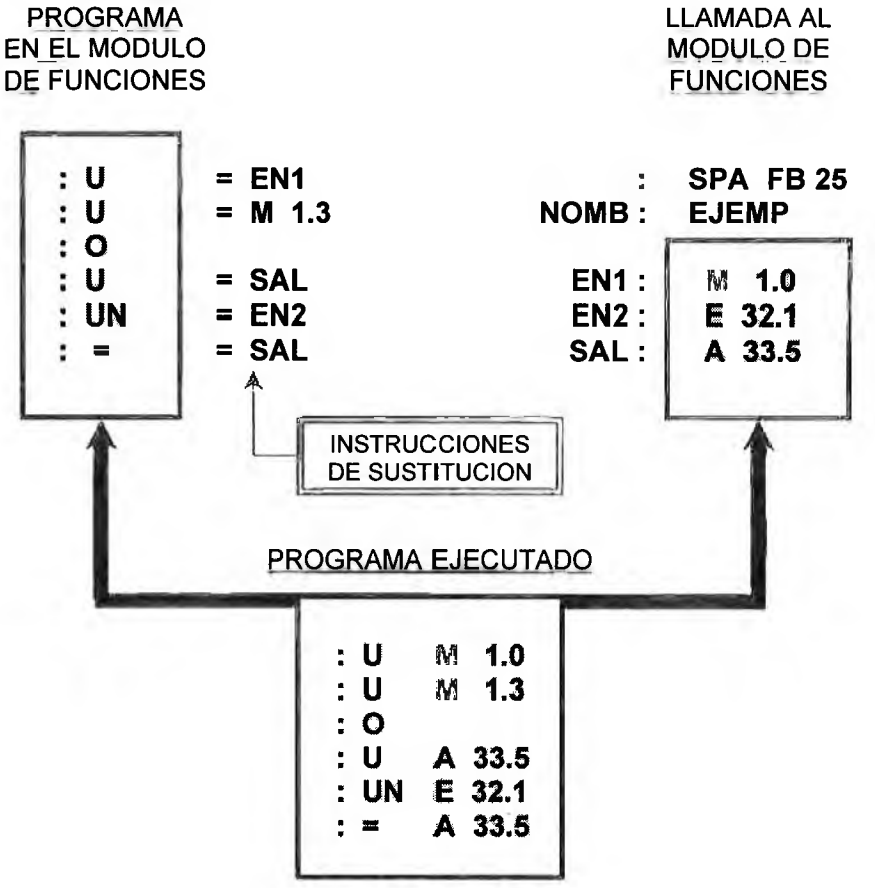


Figura 5.18: Ejemplo de ejecución del programa con módulos de funciones e instrucciones de sustitución.

8. ANEXO 1: REFERENCIA RAPIDA A ELEMENTOS DE PROGRAMACION EN LISTA DE INSTRUCCIONES (AWL)

8.1. INSTRUCCIONES BINARIAS BÁSICAS

- **U_** COMBINACION Y. CONSULTA DE SEÑAL EN ESTADO "1".
- **UN_** COMBINACION Y. CONSULTA DE SEÑAL EN ESTADO "0".
- **O_** COMBINACION O. CONSULTA DE SEÑAL EN ESTADO "1".
- **ON_** COMBINACION O. CONSULTA DE SEÑAL EN ESTADO "0".
- **O** COMBINACION O DE FUNCIONES Y.
- **U(** COMBINACION Y DE FUNCIONES O.
- **O(** COMBINACION O DE EXPRESIONES ENTRE PARENTESIS.
- **)** CERRAR PARENTESIS.
- **=_** ASIGNACION.
- **S_** ACTIVAR (FUNCION DE MEMORIA).
- **S Zn** CARGA DE UN CONTADOR.
- **R_** PUESTA A CERO (FUNCION DE MEMORIA).
- **R Tn** PUESTA A CERO DE UN TEMPORIZADOR.
- **R Zn** PUESTA A CERO DE UN CONTADOR.
- **SI Tn** ARRANQUE DE UN TEMPORIZADOR COMO IMPULSO.
- **SV Tn** ARRANQUE DE UN TEMPORIZADOR COMO IMPULSO PROLONGADO.
- **SE Tn** ARRANQUE DE UN TEMPORIZADOR COMO RETARDO A LA CONEXIÓN.
- **SS Tn** ARRANQUE DE UN TEMPORIZADOR COMO RETARDO A LA CONEXIÓN MEMORIZADO.
- **SA Tn** ARRANQUE DE UN TEMPORIZADOR COMO RETARDO A LA DESCONEXION.
- **ZV Zn** COMPUTO HACIA DELANTE DE UN CONTADOR.
- **ZR Zn** COMPUTO HACIA ATRÁS DE UN CONTADOR.

8.2. FUNCIONES DIGITALES BÁSICAS

- **L_** CARGA.
- **LC_** CARGA CODIFICADA (DE UN VALOR DE TIEMPO O CUENTA).
- **T_** TRANSFERENCIA.
- **!=F** COMPARACION IGUAL.
- **><F** COMPARACION DISTINTO.
- **>F** COMPARACION MAYOR.
- **>=F** COMPARACION MAYOR-IGUAL.
- **<F** COMPARACION MENOR.
- **<=F** COMPARACION MENOR-IGUAL.

8.3. FUNCIONES DE ORGANIZACIÓN BÁSICAS

- **SPA mm** LLAMADA INCONDICIONAL A UN MODULO.
- **SPB mm** LLAMADA CONDICIONAL A UN MODULO.
- **BE** FIN DE MODULO.
- **BEA** FIN ABSOLUTO DE MODULO.
- **BEB** FIN CONDICIONADO DE MODULO
- **STP** PARADA (STOP).
- **NOP** OPERACIÓN NULA.

8.4. FUNCIONES DE SUSTITUCION BÁSICAS

- **U = xxxx** COMBINACION Y. CONSULTA DE SEÑAL EN ESTADO "1".
- **UN = xxxx** COMBINACION Y. CONSULTA DE SEÑAL EN ESTADO "0".
- **O = xxxx** COMBINACION O. CONSULTA DE SEÑAL EN ESTADO "1".
- **ON = xxxx** COMBINACION O. CONSULTA DE SEÑAL EN ESTADO "0".
- **= = xxxx** ASIGNACION.
- **S = xxxx** ACTIVAR (FUNCION DE MEMORIA).
- **S = xxxx** CARGA DE UN CONTADOR.

- **RB = xxxx** PUESTA A CERO (FUNCION DE MEMORIA).
- **RD = xxxx** PUESTA A CERO DE UN TEMPORIZADOR O CONTADOR.
- **SI = xxxx** ARRANQUE DE UN TEMPORIZADOR COMO IMPULSO.
- **SE = xxxx** ARRANQUE DE UN TEMPORIZADOR COMO RETARDO A LA CONEXION.
- **SAR = xxxx** ARRANQUE DE UN TEMPORIZADOR COMO RETARDO A LA DESCONEXIÓN/CUENTA ATRÁS DE UN CONTADOR.
- **SSV = xxxx** ARRANQUE DE UN TEMPORIZADOR COMO RETARDO A LA CONEXIÓN MEMORIZADO/CUENTA DELANTE DE UN CONTADOR.
- **SVZ = xxxx** ARRANQUE DE UN TEMPORIZADOR COMO IMPULSO PROLONGADO/CARGA DE UN CONTADOR.
- **L = xxxx** CARGA DE UN OPERANDO ACTUAL.
- **LC = xxxx** CARGA CODIFICADA (DE UN VALOR DE TIEMPO O CUENTA).
- **LW = xxxx** CARGA DIRECTA DE UN OPERANDO ACTUAL.
- **T = xxxx** TRANSFERENCIA A UN OPERANDO ACTUAL.
- **B = mm** ELABORACION DE PARAMETROS DE MODULO.
- **BI = mm** ELABORACION INDIRECTA DE PARAMETROS DE MODULO.

Capítulo 6

Técnicas de Realización

Contenido

1. INTRODUCCIÓN

Después de haber utilizado las redes de Petri (RdP) para la descripción y validación de sistemas, se van a presentar unas técnicas que permiten *realizar* los sistemas descritos.

Recordar que el proceso de síntesis de un automatismo mediante la utilización de las redes de Petri consta de cuatro fases:

1. Descripción o Modelización.
2. Simplificación.
3. Validación.
4. Realización.

La realización, que aparece como la última fase del proceso, es la construcción del dispositivo físico capaz de realizar las funciones deseadas (automatismo).

2. DISTINTAS TÉCNICAS DE REALIZACIÓN DE LAS RdP

- Realización cableada.

Realización mediante la conexión directa de biestables y puertas lógicas.

- Realización utilizando memorias (ROM) y matrices lógicas programables (PLA).

Normalmente, para sistemas de una cierta complejidad, será más interesante una realización con macrocomponentes que otra construida exclusivamente con puertas lógicas y biestables, componentes de pequeña escala de integración. En efecto, al utilizar componentes altamente integrados se puede observar que disminuye el coste de los componentes por función que se desea realizar, se reduce el conexionado entre componentes así como el tamaño del equipo y su consumo energético, etc.

- Realización programada con autómatas programables.

- Realización programada con computadoras.

En este capítulo sólo se abordarán la técnica de realización cableada y, posteriormente, la técnica de realización programada con autómatas

programables, particularizado este último caso al lenguaje de programación STEP 5, en código de instrucciones (AWL), propio de los autómatas programables SIMATIC-S5 de la casa Siemens.

Los métodos de realización que se van a presentar se centran esencialmente en las RdP binarias (1-limitadas), o sea, todo lugar estará marcado con una marca o no estará marcado. Se recordará que esta clase de modelos es bastante adecuada para la descripción de sistemas lógicos, simplifica las realizaciones y no impone una restricción importante.

No obstante, los métodos de realización de RdP binarias que se van a estudiar son inmediatamente generalizables para realizar RdP k-limitadas.

2.1. REALIZACIÓN CABLEADA

Estas técnicas permiten la obtención directa del circuito lógico a partir de la RdP dando unas reglas de conexionado de los dispositivos lógicos básicos (puertas y biestables). De esta forma se suprimen dos de las más delicadas y fastidiosas etapas de la síntesis clásica de los sistemas secuenciales: la codificación de estados y la escritura de las ecuaciones lógicas.

La idea fundamental consiste en materializar cada lugar (binario) por un biestable que memorice si el lugar está o no marcado, y que habrá que activar o desactivar con el circuito lógico correspondiente a la transición.

La realización cableada de RdP no binarias deberá utilizar un contador por cada lugar que pueda contener hasta k marcas.

Este tipo de técnica conduce a realizaciones en las que el número de memorias (biestables) no será normalmente mínimo pero existirá una relación directa entre la descripción funcional, es decir, la RdP (qué es lo que hace) y la descripción estructural, o sea, el circuito (cómo está hecho). Dicho de otro modo, es una realización modular.

Esta realización modular tiene la ventaja de que facilita la comprensión y la modificalidad del circuito, además de reducir enormemente el tiempo en la fase de diseño.

Por otro lado, actualmente el coste de los materiales decrece continuamente y el esfuerzo por economizar debe llevarse a cabo sobre otros aspectos (puesta a punto, mantenimiento, etc.) no sobre la minimización de componentes.

2.1.1. TIPOS DE REALIZACIONES CABLEADAS

Hay dos formas de realizar las RdP binarias. En ambas, a cada lugar (binario) se le asocia un biestable especial que se denominará de forma genérica *célula* de memoria y en ambos métodos, la activación de una célula se realiza al disparar alguna de las transiciones de entrada del lugar que materializa.

Estos dos tipos de realizaciones cableadas difieren, sin embargo, en la manera en que se procede para desactivar las células de memoria correspondientes a los lugares de entrada de la transición disparada.

- **Funcionamiento por transferencia impulsional.**

Al dispararse una transición la misma señal de activación, para marcar los lugares de salida, se utiliza para la desactivación, o sea, para desmarcar los lugares de entrada.

Este esquema de conexionado es conflictivo, ya que la simulación del disparo de una transición es un pulso y éste es el que provoca directamente toda la evolución del marcado. Debido a los retrasos en la conmutación de las puertas, cuando en un circuito se interconectan diversos componentes lógicos, pueden sobrevenir diferencias entre el comportamiento esperado y el observado. Estas diferencias se conocen con el nombre de fenómenos aleatorios.

No hay más que pensar lo que sucedería si el tiempo de conmutación de las células que materializan a los lugares de entrada, es menor que el de las que representan a los lugares de salida, de una determinada transición. Obviamente, al producirse el disparo de dicha transición, se desactivarían los lugares de entrada antes de tener lugar la activación de los correspondientes lugares de salida, con lo que desaparecería la señal de activación y podría no haber dado tiempo suficiente a la activación de los lugares de salida.

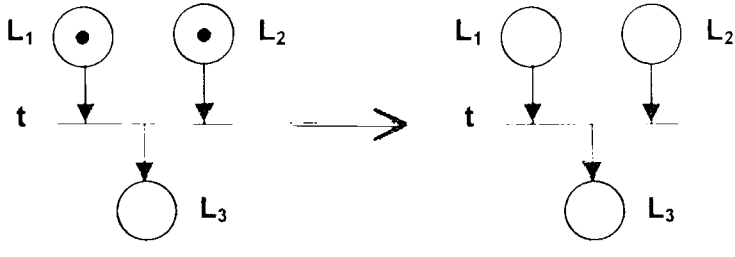


Figura 6.1: Ilustración de un fenómeno aleatorio

Se puede quitar la marca de L_1 antes de que aparezca la de L_3, en cuyo caso ya no se pondrá la marca en L_3. No se asegura por tanto que se marquen los lugares de salida.

- **Funcionamiento por llamada-respuesta.**

El disparo de una transición se realiza en dos fases:

a) La *llamada*, en la que se activan los lugares de salida.

b) La *respuesta*, en la que la activación de los lugares de salida de la transición disparada posibilita la desactivación de los lugares de entrada de la misma.

Esto quiere decir que se utilizan dos señales distintas para la activación y la desactivación, por tanto permite, en principio, garantizar la activación de las células asociadas a los lugares de salida de la transición.

Señalar que esta técnica de conexionado tiene una gran aceptación industrial.

2.1.2. LA CÉLULA DE MEMORIA

Para la realización física de un lugar se toma un biestable R-S con activación prioritaria, cuyo comportamiento lógico es el siguiente:

R	S	Q	ESTADO
0	0	Q	Memorización
0	1	1	Activación
1	1	1	Activación (prioritaria)
1	0	0	Desactivación

Tabla 6.1: Tabla de verdad de un biestable R-S con activación prioritaria

También se puede hacer un lugar con alguna de las realizaciones alternativas que se muestran en las siguientes figuras, en las que se utilizan sólo dos puertas, mucho más simples que un biestable pero suficientes para esta aplicación, y en las que se puede comprobar que el comportamiento lógico de todas es el mismo que el del mencionado biestable.

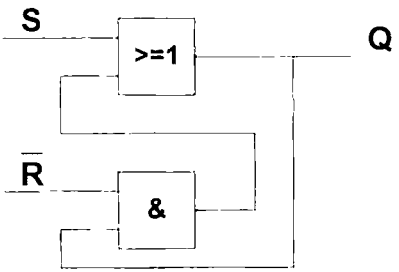


Figura 6.2: Realización alternativa que utiliza una puerta OR y una puerta AND

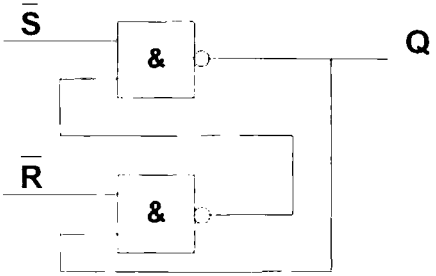


Figura 6 3: Realización alternativa que utiliza dos puertas NAND

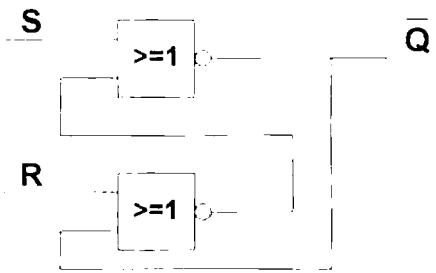


Figura 6 4: Realización alternativa que utiliza dos puertas NOR

Normalmente, para dichas realizaciones, será necesario utilizar puertas con varias señales de activación y desactivación, como se puede apreciar en las figuras mostradas a continuación.

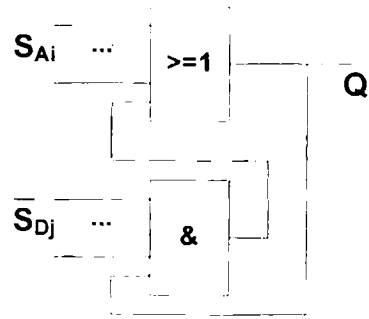


Figura 6.5: Utilizando una puerta OR y una puerta AND

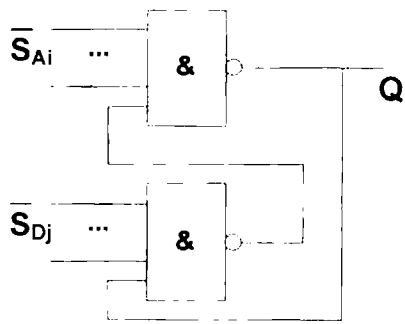


Figura 6.6: Utilizando dos puertas NAND

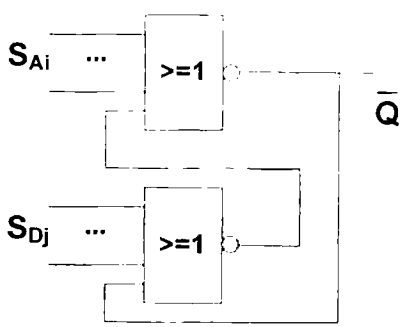


Figura 6.7: Utilizando dos puertas NOR

En efecto, para realizar un lugar binario (1-limitado) hace falta una memoria que se active al dispararse una de todas las transiciones de entrada que pueda tener el lugar que materializa. Además, la memoria debe desactivarse al dispararse alguna de todas las transiciones de salida que pueda tener el lugar que realiza.

El interés de la activación prioritaria se pone de manifiesto en el caso de que para algún lugar se disparen a la vez una transición de entrada y otra de salida; caso en el que el lugar debe permanecer marcado.

En adelante, para la realización física de los lugares, se utilizará la alternativa correspondiente al esquema NOR/NOR (dos puertas NOR) con varias señales de activación y de desactivación; que se representará de la siguiente forma:

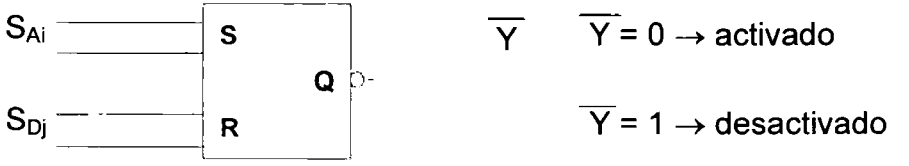


Figura 6.8: Realización física de un lugar

2.1.3. TRANSICIONES

Son las funciones lógicas para realizar la activación y la desactivación de los lugares (células de memoria) correspondientes. En definitiva, la realización física de las transiciones no será más que la implementación de los circuitos lógicos correspondientes a las condiciones lógicas de activación y de desactivación.

Para poder escribir dichas condiciones lógicas de activación y de desactivación se considerará una configuración genérica alrededor de un lugar, tal y como se muestra en la figura 6.9.

Este sería el entorno general del lugar Pj. Notar que este lugar Pj está en el conjunto {Pj1,...,Pjl,...,Pjr} de los lugares de entrada a la transición Ajk, pero Pj también es un lugar de entrada para las transiciones Aj1,..., Aj(k-1), Aj(k+1),..., Aj(m-1) y Ajm.

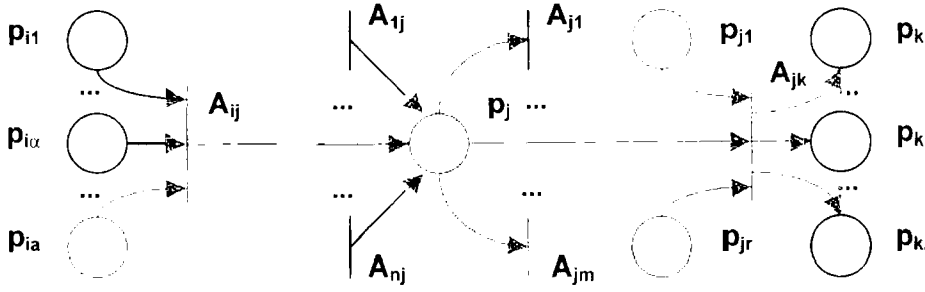


Figura 6.9: Entorno general de un lugar

A la vista de esta figura se va a definir la función lógica de activación, C_A^j, de la célula que materializaría al lugar Pj.

Función de activación del lugar Pj:

producto lógico

$$C_A^j = \sum_{i=1}^{i=n} \left(A_{ij} \cdot \prod_{\alpha=1}^{\alpha=a} P_{i\alpha} \right)$$

suma lógica

Igualmente se puede ahora definir la función lógica de desactivación, C_D^j, de la célula que materializaría al lugar Pj.

La función de desactivación del lugar Pj en la realización por transferencia impulsional sería:

$$C_D^j = \sum_{k=1}^{k=m} \left(A_{jk} \cdot \prod_{l=1}^{l=r} P_{jl} \right)$$

En cambio, en la realización por llamada-respuesta la condición de desactivación sería la siguiente:

$$C_D^j = \sum_{k=1}^{k=m} \left(\prod_{\beta=1}^{\beta=b} P_{k\beta} \right)$$

Hay que hacer notar que la señal de desactivación realizada por el método llamada-respuesta, indicada anteriormente, es una condición simplificada; caso de que no ocurran situaciones, en la RdP, del tipo mostrado en la figura 6.10.

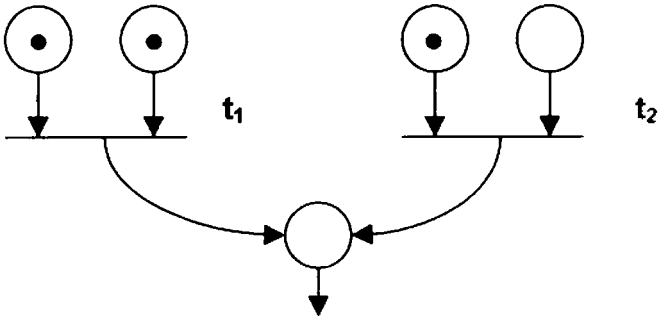


Figura 6.10: Situación que no debe ocurrir

Efectivamente, según la condición expuesta, al producirse el franqueo de t_1 se quitan también las marcas de los lugares de entrada a la transición t_2 que no deben quitarse.

Si en la RdP ocurre una situación de este tipo, cosa que no es muy frecuente, entonces hay que utilizar la condición general, no simplificada, que se puede expresar de la forma siguiente:

$$C_D^J = \sum_{k=1}^{k=m} \left(\prod_{\beta=1}^{\beta=b} P_{k\beta} \right) \cdot \left(A_{jk} \cdot \prod_{l=1}^{l=r} P_{jl} \right)$$

Con lo que ahora se vuelve a tener el mismo problema que con el funcionamiento por transferencia impulsional, es decir, no se asegura el marcado de los lugares de salida.

Para eliminar este problema se introduce en cada célula de memoria un retardo de manera que, cuando se dé la señal de desactivación, se mantenga la salida durante un pequeño intervalo de tiempo.

2.1.4. EJEMPLO DE REALIZACIÓN DE UN LUGAR (NUDO O) POR TRANSFERENCIA IMPULSIONAL

Sea el siguiente ejemplo, un nudo O, que se quiere realizar físicamente por el método de transferencia impulsional:

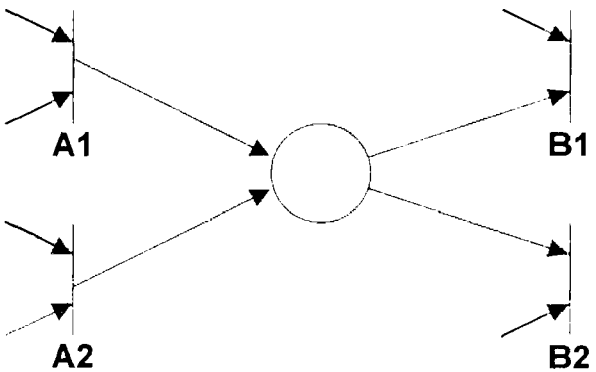


Figura 6.11: Nudo O

La realización física sería la siguiente, donde se incluyen los circuitos de activación y de desactivación:

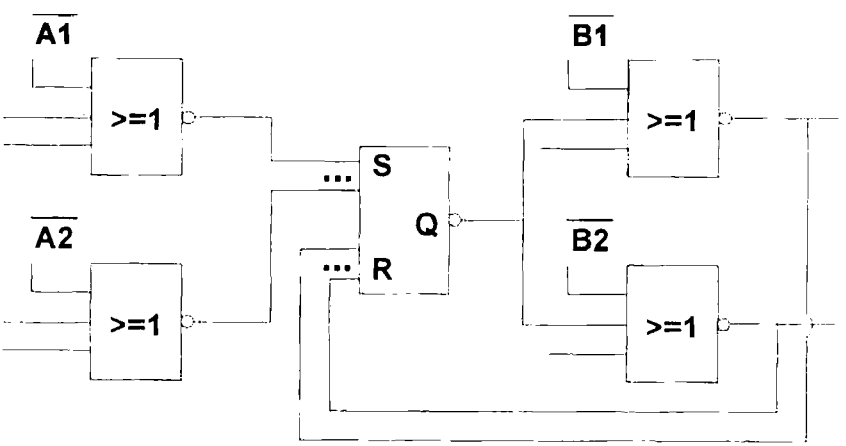


Figura 6.12: Realización física, cableada, de un nudo O

Como se puede apreciar, el lugar equivale a una célula de memoria y cada transición a una puerta lógica NOR, según el criterio que se ha tomado.

2.1.5. EJEMPLO DE REALIZACIÓN DE UNA TRANSICIÓN (NUDO Y) POR TRANSFERENCIA IMPULSIONAL

Sea el siguiente ejemplo, un nudo Y, que se quiere realizar físicamente por el método de transferencia impulsional:

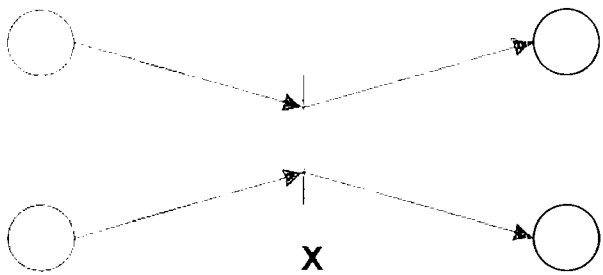


Figura 6.13: Nudo Y

La realización física sería la siguiente, donde se incluyen los circuitos de activación y de desactivación:

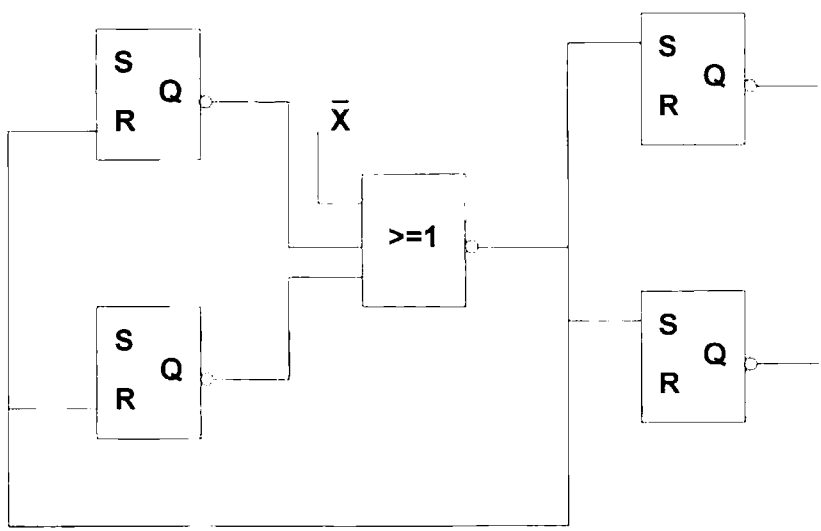


Figura 6.14: Realización física, cableada, de un nudo Y

Se ve, atendiendo a la figura anterior, que a cada lugar equivale una célula de memoria y a la transición una puerta lógica NOR.

Recordar que para la realización física de los lugares se ha tomado el criterio de utilizar células de memoria compuestas por dos puertas NOR.

Hacer notar que si bien este circuito, al igual que el del ejemplo anterior, simula perfectamente la evolución del marcado de la RdP que es el modelo de nuestro sistema (automatismo), no genera ninguna salida.

2.1.6. SALIDAS

Las salidas se realizan de forma inmediata, para ello se harán las siguientes consideraciones:

- Las entradas están asociadas a las transiciones.
- Las salidas están asociadas a los lugares.
- Se pueden asociar salidas a las transiciones cuando éstas se tratan de un pulso (activar de un temporizador, incrementar un contador, etc.).

Se considerarán RdP donde todas las salidas estarán asociadas solamente a lugares, con lo que, tomando en cuenta esta consideración, una salida se debe activar cuando esté marcado el lugar al cual está asociada, o sea, cuando esté activada la célula de memoria correspondiente a ese lugar.

De esta forma, si la salida S_i está asociada al lugar L_i, entonces:

$$S_i = L_i$$

Así, si el lugar está marcado, o sea, si la célula de memoria que materializa al lugar está activada entonces valdrá $L_i = 1$, con lo que la salida asociada a ese lugar valdrá $S_i = 1$, tomando el valor 0 en caso contrario.

De la misma manera, si la salida S_i está asociada a más de un lugar; sean $L_1,\ldots, L_n$, entonces:

$$S_i = L_1 + L_2 + \ldots + L_n$$

En este caso, si alguno de los lugares está marcado entonces la salida valdrá 1 y solamente tomará el valor 0 en el caso de que ninguno de los lugares a los que esté asociada esté marcado.

Hay que tener en cuenta que, con el criterio que se ha tomado para la realización física de los lugares, cuando un determinado lugar esté marcado, o sea; cuando la célula de memoria que corresponde a ese determinado lugar esté activada, su salida será un "0" lógico. Por tanto, para dicha realización física, la condición lógica que debe cumplir cada salida será la negada de las expresadas más arriba, según el caso.

2.1.7. MARCAJE INICIAL

Otra cosa necesaria para hacer la realización física es la señal de inicialización general, que debe activar las células asociadas a los lugares marcados inicialmente y debe desactivar las memorias asociadas a los lugares no marcados inicialmente.

Este marcaje o estado inicial se puede conseguir con un circuito similar al siguiente:

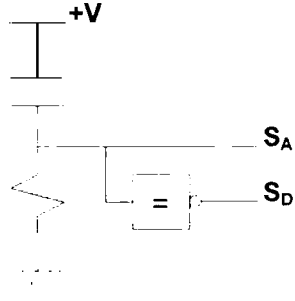


Figura 6.15: Circuito para realizar el marcaje inicial

Donde:

S_A es la señal de activación para los lugares que haya que marcar inicialmente.

S_D es la señal de desactivación para el resto de los lugares que no deben estar marcados en el instante inicial (al encender el circuito nunca se sabe si se parte de un "0" o un "1" y hay que asegurarse).

La función que realiza este circuito es simplemente un "Reset" o inicialización del equipo.

2.1.8. EJEMPLO DE REALIZACIÓN FÍSICA

Considérese el carro *C* de la figura sobre el contacto de fin de carrera *A*. Al pulsar el botón *M*, el carro se desplaza hacia la derecha gracias a la acción de un motor mandado por un relé *d*. Al llegar al contacto *B*, el carro vuelve a *A* por la acción de un motor mandado por un relé *i*. Cuando *C* vuelve a *A*, se para si *M* no está pulsado; de lo contrario comienza un nuevo ciclo.

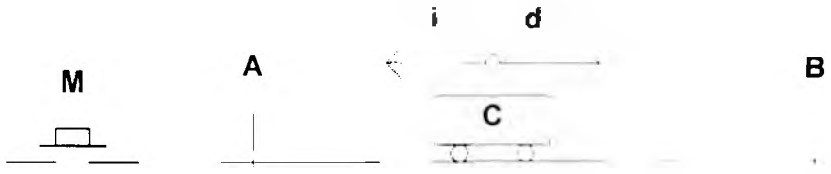


Figura 6.16: Ejemplo de carro que va y viene

La descripción funcional mediante una RdP es la que sigue:

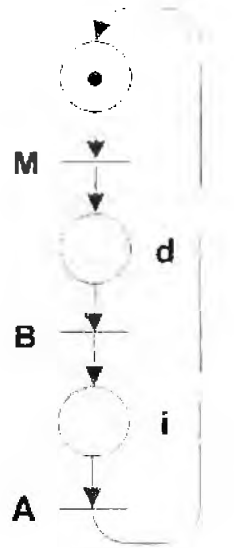


Figura 6.17: Red de Petri del ejemplo de la figura 6.16

Una vez hecha la RdP (descripción del sistema) se sabe que el siguiente paso es la validación de dicha red.

Se deja como ejercicio al lector interesado la comprobación de este punto.

El último paso es pues, la realización física que se pasa a concretar a continuación. Se dibujará en primer lugar el circuito de activación, segundo el circuito de desactivación y por último la realización física completa.

El circuito de activación es el que aparece en la siguiente figura. Darse cuenta de que se ha materializado cada lugar como una célula de memoria y cada transición como una puerta NOR:

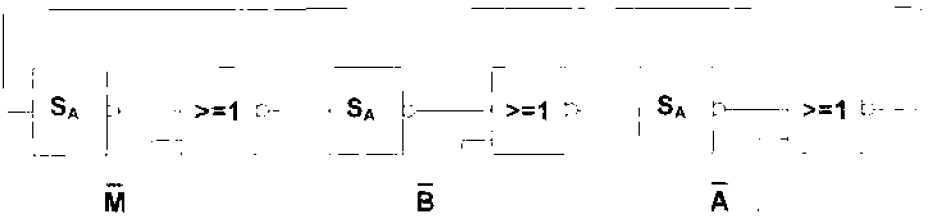


Figura 6.18: Circuito de activación del ejemplo de la figura 6.16

El circuito de desactivación se ha realizado por el método de transferencia impulsional y es el que se muestra seguidamente. Notar que se tratan de los mismos lugares y transiciones, es decir, de las mismas células de memoria y puertas NOR que en la figura anterior:

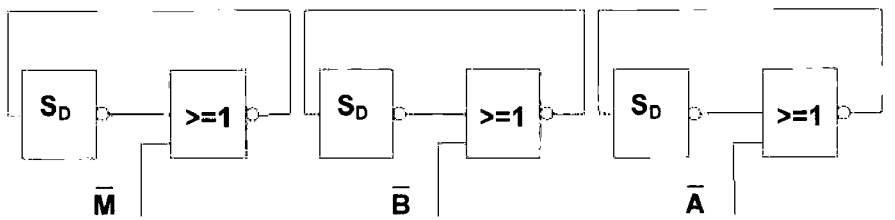


Figura 6.19: Circuito de desactivación del ejemplo de la figura 6.16

La realización física completa, donde también se han incluido las salidas, se muestra en la figura 6.20.

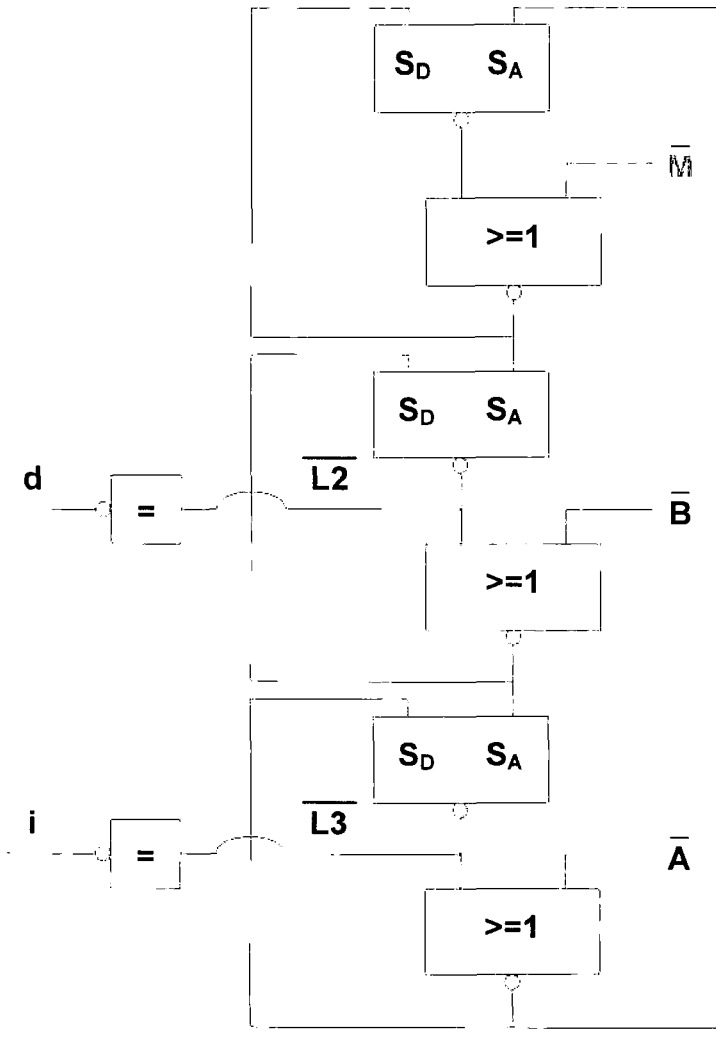


Figura 6.20: Realización física completa

No se ha dibujado el circuito que ya se ha visto para el marcaje inicial, en el primer momento.

2.2. REALIZACIÓN PROGRAMADA CON AUTÓMATAS PROGRAMABLES (AP)

Los lenguajes de programación de los AP, o las formas de representación, se pueden concebir en dos grupos según sean:

- Una transposición tecnológica inmediata de los esquemas de realización cableada de los automatismos → KOP, FUP

- Una transcripción, en forma programada, de una descripción funcional de los automatismos (RdP) → AWL

Se considerará en adelante, un modo de programar un AP de la familia SIMATIC-S5, de Siemens, con el lenguaje de programación STEP 5 en AWL, a partir de una descripción funcional del automatismo basada en RdP.

Se verá como se van a utilizar tan sólo instrucciones fundamentales, es decir, básicas y como con éstas se posibilita la programación de las funciones lógicas.

2.2.1. PROCESO INDIRECTO

Al cómo utilizar el lenguaje STEP 5 en la forma de secuencia o lista de instrucciones, AWL, para codificar una RdP, se puede proponer un primer esquema basado en la simple transcripción del diagrama lógico obtenido mediante la realización cableada, como se puede ver en el ejemplo de la figura 6.21

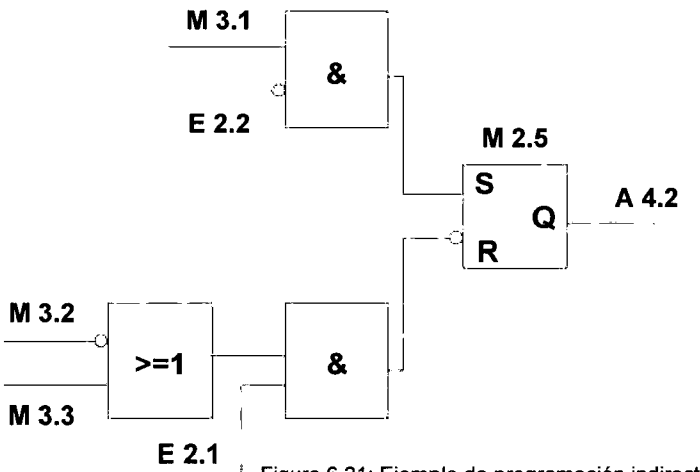


Figura 6.21: Ejemplo de programación indirecta

Hacer notar que M 7.4 es una variable intermedia pues, no existe la instrucción RN (Reset Negado).

2.2.2. PROCESO SISTEMÁTICO (DIRECTO)

Las RdP, así como los circuitos lógicos, evolucionan en paralelo. Dado que los AP son máquinas secuenciales, éstos procederán simulando en serie las evoluciones de la RdP.

Antes de proceder a la programación, parece lógico asignar a cada lugar de la RdP una variable o marca que tomará el papel de célula de memoria, de esta forma se podrá saber en cada momento que lugar está marcado y cual no:

$$P_i = M\_ \cdot \_$$

Por otro lado, las entradas y salidas estarán cableadas:

$$E_i = E\_ \cdot \_$$
$$S_i = A\_ \cdot \_$$

Para realizar este proceso de programación de una forma sistemática, lo más natural parece ser la codificación en cuatro fases:

- Marcaje inicial
- Evolución de la Red
- Salidas
- Mímico

Cada una de estas fases puede constituir un módulo distinto de programación (PB); de esta forma, el programa principal, realizado en el módulo de organización OB1, constará únicamente de las llamadas a los distintos módulos de programación de forma secuencial.

PROGRAMA PRINCIPAL

Teniendo en cuenta lo expresado en el apartado anterior, el programa principal sería:

Módulo OB1:

```
U  M 1.0
SPB PB1
SPA PB2
SPA PB3
SPA PB4
BE
```

Donde:

La primera fase de la programación, el marcaje inicial de la RdP, se ha realizado en el módulo PB1. Este marcaje inicial hay que realizarlo una sola vez, al principio; para ello se utiliza la variable M 1.0, que debe valer "1" sólo en el instante inicial, y se llama al módulo de programación correspondiente, PB1, con una instrucción de salto condicional al valor de dicha marca.

Para poder asegurar que en un principio la variable M 1.0 tiene el valor "1", se pueden utilizar los módulos de arranque:

Módulo OB21 ≡ OB22:

```
O  M 1.0
ON M 1.0
S  M 1.0
BE
```

Estos módulos se ejecutan una sola vez, antes del OB1, cuando se arranca el autómata; ejecutándose uno u otro módulo (el OB21 o el OB22) dependiendo de la forma en que se dé alimentación al autómata.

La segunda fase de la programación, la evolución de la RdP, se ha realizado en el módulo PB2. A este módulo se llama con una instrucción de salto incondicional pues se debe estar ejecutando continuamente.

La tercera fase de la programación, la asignación de las salidas, se ha realizado en el módulo PB3. A este módulo de programación también se llama con una instrucción de salto incondicional pues también se debe ejecutar de una forma continua.

La cuarta y última fase de la programación, correspondiente al mímico, se ha realizado en el módulo de programación PB4, módulo que se llama con la instrucción de salto incondicional.

En los siguientes apartados se pasa a describir la función y codificación de cada uno de los cuatro módulos PB, correspondientes a las cuatro fases en que se ha dividido la programación.

MARCAJE INICIAL

Consiste en hacer un SET a cada variable o marca asociada a un lugar que debe estar marcado inicialmente y un RESET a las restantes.

Módulo PB1:

```
S  M _._
S  M _._
...

R  M _._
R  M _._
...

R  M 1.0
BE
```

Se hace SET a los lugares marcados inicialmente, o sea, de las marcas asignadas a los lugares que deben estar marcados al principio y RESET de los lugares no marcados inicialmente, es decir, de sus variables asociadas. Es además evidente que no importa el orden en que se programen estas instrucciones.

Finalmente y para que no se ejecute más este módulo, recordar que el marcaje inicial sólo debe realizarse una vez y al principio, se ha hecho un RESET de la marca M 1.0. Está claro que para tal menester podría haberse utilizado otra marca, cualquiera permitida por el autómata y que dicha marca (en este caso la M 1.0) no debe ser utilizada en adelante en el programa.

Al ejecutarse el programa principal de forma cíclica (módulo OB1), en el próximo ciclo de tratamiento no será llamado este módulo PB1 correspondiente al marcaje inicial, pues la llamada se realiza mediante la instrucción de salto condicional al valor almacenado en el VKE y previamente a dicha instrucción, se ha cargado el valor de la variable M 1.0 que ya vale "0" y que no va a ser modificada más por el programa.

Darse cuenta de que la instrucción SPB, situada en el módulo OB1, sólo funciona si VKE = " 1" y de que esta condición se cumple una única vez, en el primer ciclo de tratamiento pues previamente, ha dicha instrucción de salto, se ha cargado la variable M 1.0, que por otro lado se ha asegurado que tenga el valor " 1" al principio, antes de ejecutarse el primer ciclo, utilizando para ello los módulos OB21 y OB22.

EVOLUCIÓN DE LA RED

A cada transición se le asocia el código que permita evaluar la condición de disparo (intersección de la condición de sensibilización y del evento asociado).

Además, si hay que disparar la transición, se dispondrá el código que permita desactivar sus lugares de entrada y activar sus lugares de salida.

Para cada transición habrá que introducir un bloque de código de instrucciones de la forma:

Módulo PB2:

```
U   M
U   M
U   E
S   M
S   M
R   M
R   M
BE
```

Para ver si se cumple la condición de sensibilización de la transición (intersección de sus lugares de entrada) simplemente hay que hacer un AND de todas las marcas correspondientes a los lugares de entrada de la transición de la que se trate. El resultado será almacenado en el VKE.

Para ver si se cumple la condición de disparo de la transición (intersección de la condición de sensibilización con el evento asociado a la transición) únicamente habrá que hacer una instrucción AND entre el resultado anterior y el valor del evento, que normalmente será una entrada al sistema (automatismo). El resultado será almacenado en el VKE.

Recordar que la instrucción U realiza la función AND entre el valor de la variable y el valor almacenado en el VKE, excepto cuando se rompa la secuencia, en cuyo caso la primera instrucción U realiza una carga del valor de la variable en el VKE.

Darse cuenta de que las instrucciones S y R, sólo funcionan si VKE = " 1" y de que esta condición se cumple si el resultado de las operaciones que se han realizado hasta ahora es " 1", es decir, si se cumple la condición de disparo de la transición. Además S y R aunque rompen la secuencia no modifican el VKE.

A partir de ahora, siguiendo con el código de programación asociado a la transición de que se trate, sólo se utilizarán la instrucción S, para la activación de los lugares de salida de la transición, ya que deben ser marcados, y la instrucción R, para la desactivación de los lugares de entrada, ya que deben ser desmarcados. Estudiar como este último bloque de instrucciones SET y RESET no se ejecutan siempre, sino sólo cuando se da la condición de disparo de la transición.

SALIDAS

A cada lugar se le asocia el código que posibilite la elaboración de las acciones asociadas.

Módulo PB3:

```
...
- __
O  M  _._
O  M  _._
...

=  A  _._
____
...

BE
```

Si una misma salida está asociada a más de un único lugar, no se puede tratar cada lugar por separado. En realidad se asocia el código a cada acción, o sea, hay que introducir un bloque de código de instrucciones como el anterior para cada acción diferente, no para cada lugar.

Primero se evalúa si alguno de los lugares a los que está asociada la salida de que se trate está marcado. Para comprobar si se cumple dicha condición se realiza un OR entre todos los lugares que tienen esa misma salida. El resultado será almacenado en el VKE.

No se puede utilizar el SET, S, para activar las salidas pues esta instrucción funciona sólo si VKE = "1", ocurre entonces que cuando sea VKE = "0" no pondrá a "0" la salida sino que la dejaría al valor que tuviese; se utiliza la asignación, =, cuyo funcionamiento no depende del VKE, dando a la salida el valor almacenado en éste, "1" o "0" dependiendo de si se cumple o no la condición anterior

MÍMICO

Este módulo no sirve para el automatismo sino para chequeo y saber como va funcionando. Por lo tanto no es necesaria su programación.

Primeramente, se asocia a cada lugar una salida (no cableada).

Y ahora sí, para cada lugar habrá que introducir un bloque de código de instrucciones de la forma:

Módulo PB4:

```
...
U  M _._
=  A _._
...

BE
```

Además, en este caso, da igual resultado utilizar la instrucción U que usar la instrucción O. En este caso, ambas realizarán la función de carga en el VKE.

PROBLEMAS QUE APARECEN

La técnica de programación expuesta es muy sistemática, pero puede conducir a ciertos funcionamientos anómalos conocidos de forma general con el nombre de aleatoriedades de programación. Éstas se deben al funcionamiento secuencial, en serie, del autómata.

Caso 1:

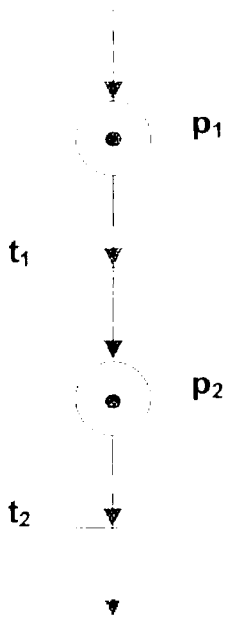


Figura 6.22: Ilustración del problema correspondiente al caso 1

Si la RdP es binaria, siempre que se programe t_1 antes que t_2 se puede alcanzar transitoriamente un marcado no binario (2 marcas) lo que llevaría a un incorrecto funcionamiento (caso en que t_1 y t_2 disparasen a la vez, es decir, en el mismo ciclo de tratamiento).

Otro de los problemas que aparecen se puede observar en la figura 6.23.

Caso 2:

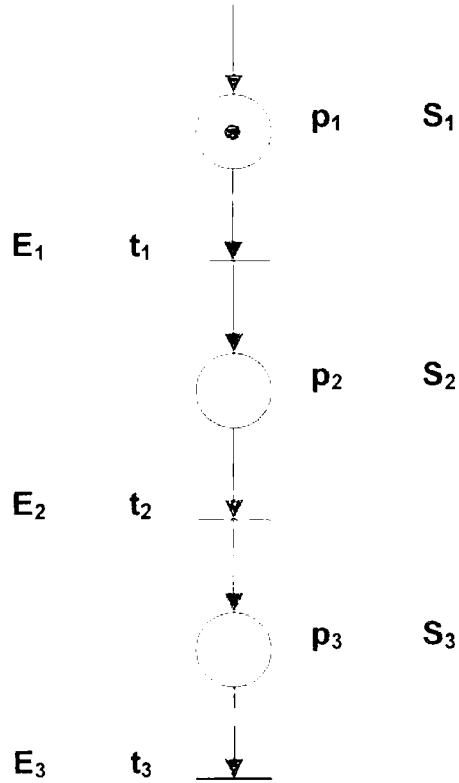


Figura 6.23: Ilustración del problema correspondiente al caso 2

Supóngase que las transiciones se programan en el orden t_1, t_2 y t_3. Si se dan simultáneamente los eventos E_1 y E_2 (en el mismo ciclo de tratamiento) ocurre que la marca evolucionará directamente de P_1 a P_3 y no se daría la salida S_2 asociada a P_2.

Para resolver los dos problemas planteados basta simplemente con programar t_2 antes que t_1, lo que revela que importa el orden en que se programen las transiciones.

SOLUCIÓN DE LOS PROBLEMAS ANTERIORES

La adopción del principio de jugar con el orden en que se programen las transiciones es factible, pero puede complicar sensiblemente la programación en un caso general.

Un método simple y también sistemático para resolver las aleatoriedades de programación es hacer que la RdP *evolucione de forma síncrona*. Esto quiere decir que la RdP se codificará de forma que su marcado evolucione en cada ciclo de tratamiento de manera que cada marca sólo pueda "cruzar" una transición en cada uno de dichos ciclos.

PROGRAMACIÓN PARA LA EVOLUCIÓN SÍNCRONA DE UNA RDP

La evolución síncrona de una RdP se puede conseguir subdividiendo la fase de *evolución de la red* en dos; una primera en la que se realice el *cálculo de las condiciones de evolución del marcado* y una segunda en la que se codifique la *actualización del marcado*. Con lo que ahora la programación constará de cinco fases. Teniendo esto en cuenta, el programa principal sería:

Módulo OB1

```
U  M 1.0
SPB PB1
SPA PB2
SPA PB3
SPA PB4
SPA PB5
BE
```

Los módulos OB21 y OB22 quedarían exactamente igual que antes.

En el módulo PB1 se realiza la programación correspondiente al marcaje inicial de la red (primera fase de la programación), de forma idéntica a como se ha explicado hasta ahora.

La evolución de la RdP se codificará en este caso, en dos módulos (fases) distintos, el PB2 y el PB3.

En el módulo de programación PB2 se incluirán las instrucciones necesarias para el realizar el cálculo de las condiciones de evolución del marcado, en el que se pueden elaborar las condiciones de disparo de las transiciones o, directamente, las condiciones de marcado y desmarcado de los lugares.

Por otro lado, en el módulo de programación PB3 se realizará la actualización del marcado, o sea, el desarrollo del nuevo marcado.

Las dos restantes fases de la programación se incluirán en los módulos PB4 y PB5 respectivamente. En el módulo PB4 se elaborarán las acciones (salidas) asociadas a los lugares (anteriormente programado en el PB3) y en el módulo PB5 se incluirá la codificación correspondiente al mímico (anteriormente programado en el PB4), de igual manera a como se ha expuesto en los apartados anteriores.

Módulo PB2:

```
...
U   M
U   M
...

U   E
=   M 2.0
...
```

Se comprueba si se cumple la condición de disparo de la transición (intersección de la condición de sensibilización con el evento asociado a la transición), igual que se hacía anteriormente, pero en esta ocasión no se procede de inmediato a actualizar el marcado (tarea que se realizará en la siguiente fase), sino que se guarda en una variable (en este caso la marca M 2.0) la condición de evolución.

Destacar que, de esta forma, se ha asignado una marca diferente a cada una de las transiciones.

Ni que decir tiene que habrá que introducir un bloque de código de instrucciones de la forma anteriormente mostrada para cada transición.

Señalar por último, que con este bloque de instrucciones se han elaborado de forma directa las condiciones de marcado y desmarcado de lugares.

Módulo PB3:

```
...
---
U  M 2.0
S  M _._
S  M _._
...

R  M _._
R  M _._
...

R  M 2.0
---
...
```

En este módulo PB3 también habrá que introducir un bloque de código de instrucciones de la forma anteriormente mostrada para cada transición.

Solamente hay que evaluar la condición de disparo de la transición (en el caso que se está tratando la marca M 2.0) y en función de dicha condición actualizar el marcado, es decir; activar los lugares de salida y desactivar lugares de entrada.

El módulo donde se realiza el cálculo de las condiciones de evolución del marcado (PB2), se puede programar de otra manera, de forma que se elaboren las condiciones de disparo de las transiciones, pero para ello será necesaria otra marca o variable intermedia.

Por ejemplo, sea una transición cualquiera:

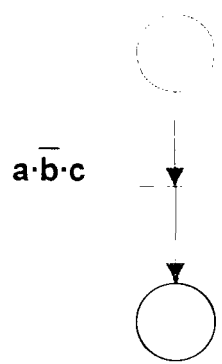


Figura 6.24: Transición cualquiera

Así, para cada transición se incluiría el siguiente código de programación (tomando como referencia la figura anterior):

Módulo PB2:

```
...
U  a
UN b
U  c
=  M 3.0

...
U  M _._
U  M _._
...

U  M 3.0
=  M 2.0
...
```

Se puede apreciar como se necesita otra marca (en este caso la M 3.0) para almacenar la condición de disparo de la transición. La marca M 2.0 almacena la condición de evolución.

Se resuelve de esta forma el problema del caso 2 pero no el del caso 1.

La forma más inmediata y sistemática de resolver la dificultad apuntada consiste en considerar dos subfases en la fase de *actualización del marcado*, con lo que la programación constará de seis fases.

El programa principal, por lo tanto, sería:

Módulo OB1:

```
U  M 1.0
SPB PB1
SPA PB2
SPA PB3
SPA PB4
SPA PB5
SPA PB6
BE
```

El marcaje inicial se realiza en el módulo PB1.

El cálculo de las condiciones de evolución del marcado se hace en el módulo PB2.

La actualización del marcado se codifica ahora en dos módulos (fases) de programación diferentes, el PB3 y el PB4.

El PB3 corresponde a la fase de *desactivación de los lugares de entrada*, en la que a partir de las condiciones de disparo de las transiciones se desactivarán sus lugares de entrada.

El módulo PB4 corresponde a la fase de *activación de lugares de salida*, en la que a partir de las condiciones de disparo de las transiciones se activarán sus lugares de salida.

Las salidas se elaboran en el módulo PB5.

El mímico se programa en el módulo PB6.

Módulo PB3:

```
...
____
U   M 2.0
R   M _._
R   M _._
...

_
...
```

Módulo PB4:

```
...
 _
U   M 2.0
S   M _._
S   M _._
..

R   M 2.0
____
...
```

La programación síncrona resuelve de forma sistemática el problema de la simulación de las RdP, pero necesita mayor cantidad de código, así como un mayor tiempo para la ejecución de los programas que la programación no-síncrona.

TRATAMIENTO DE ALARMAS

Sea PE un pulsador de emergencia, que dispara una alarma. Como a la alarma habría que responder en cualquier momento, vaya por donde vaya el proceso, entonces para hacer la descripción funcional mediante una RdP se tendría que operar de la forma que se indica en la figura 6.25.

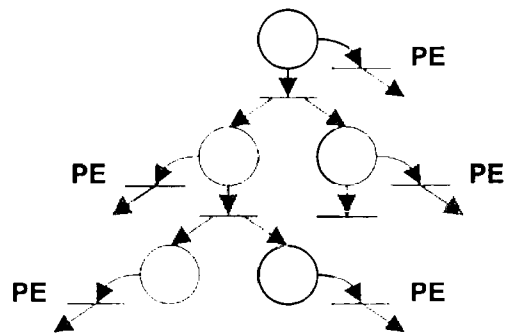


Figura 6.25: De cada lugar de la red sale una transición con el mismo evento asociado

Si no hay un número elevado de lugares, esta forma de proceder es correcta; sin embargo, puede resultar muy laboriosa, sobre todo a medida que aumenta el número de lugares.

Para evitar esto, se puede hacer un tratamiento de la alarma a nivel de autómata:

Módulo OB1:

```
U  PE
SPB PB1
UN PE
SPB PB2
BE
```

En el módulo PB1 habría que incluir el código de programación correspondiente al tratamiento de emergencia y en el módulo PB2 se incluirán las instrucciones correspondientes al tratamiento normal:

Módulo PB2:

```
U  M 1.0
SPB PB3  →  marcaje inicial
SPA PB4  →  evolución red
SPA PB5  →  salidas
SPA PB6  →  mímico
BE
```

El módulo PB4 se puede subdividir a su vez en distintas fases si se quiere conseguir una evolución síncrona de la red.

Este tipo de tratamiento ya no es correcto pues la alarma hay que dispararla lo antes posible, pero teniendo en cuenta que el programa se ejecuta continuamente y que un ciclo de tratamiento es muy rápido puede ser válido, siempre dependiendo de la importancia que tenga la alarma.

Téngase siempre presente que, el autómata tiene un módulo de organización especial, el módulo de alarma, OB3, en el que habría que codificar las instrucciones correspondientes al tratamiento de alarma. Este módulo se ejecuta una sola vez cuando ocurre la alarma y lo hace de manera inmediata (interrupción) rompiendo el ciclo de tratamiento.

EJEMPLO DE PROGRAMACIÓN DE UNA RDP

Considérese el carro *C* de la figura sobre el contacto de fin de carrera *A*. Al pulsar el botón *M*, el carro se desplaza hacia la derecha gracias a la acción de un motor mandado por un relé *d*. Al llegar al contacto *B*, el carro vuelve a *A* por la acción de un motor mandado por un relé *i*. Cuando *C* vuelve a *A*, se para si *M* no está pulsado; de lo contrario comienza un nuevo ciclo.

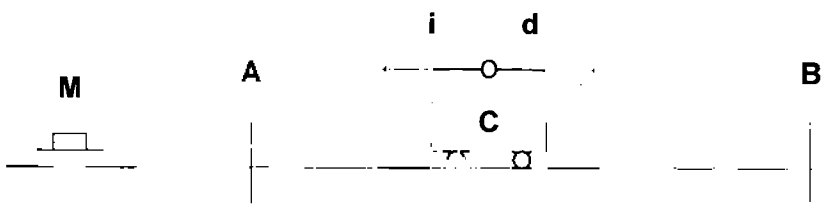


Figura 6.26: Ejemplo de carro que va y viene

Como ya se ha visto anteriormente en este capítulo (punto 2.1.8.) la descripción funcional mediante una RdP es la que sigue:

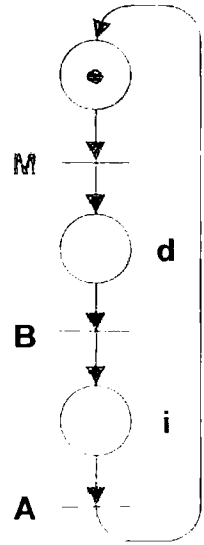


Figura 6.27: Red de Petri del ejemplo de la figura 6.26

Se asigna a cada lugar una variable, por ejemplo:

1er lugar (arriba) ≡ M 1.1
2º lugar (centro) ≡ M 1.2
3er lugar (abajo) ≡ M 1.3

Las entradas y salidas estarán cableadas. Las entradas por ejemplo en:

A ≡ E 1.2
B ≡ E 1.1
M ≡ E 1.0

Y las salidas por ejemplo en:

i ≡ A 5.6
d ≡ A 5 0

El programa sería:

Módulo OB1:

```
U  M 7.0
SPB PB1
SPA PB2
SPA PB3
BE
```

Módulo OB21 ≡ OB22:

```
O  M 7.0
ON M 7.0
S  M 7.0
BE
```

Módulo PB1:

```
S  M 1.1
R  M 1.2
R  M 1.3
R  M 7.0
BE
```

Módulo PB2:

```
U  M 1.1
U  E 1.0
S  M 1.2
R  M 1.1
U  M 1.2
U  E 1.1
S  M 1.3
R  M 1.2
U  M 1.3
U  E 1.2
S  M 1.1
R  M 1.3
BE
```

Módulo PB3:

```
O  M 1.2
=  A 5.0
O  M 1.3
=  A 5.6
BE
```

Capítulo 7

Posibilidades de Aplicación del Autómata Programable

Contenido

1. VENTAJAS DEL AUTÓMATA PROGRAMABLE

El autómata programable es actualmente el elemento de control más utilizado en funciones de automatización industrial. Su utilización no es cuestionada bajo ningún concepto y su fiabilidad y flexibilidad ya han quedado demostradas.

En un principio, a la hora de decidirse por la utilización del autómata programable se comparaba el coste del número de relés necesarios para realizar un determinado automatismo, con el coste del autómata. Actualmente la comparación se realiza entre distintos autómatas.

No obstante el empleo de autómatas programables no está totalmente asumido en todos los sectores industriales. Las industrias del automóvil y química son las áreas donde el autómata programable está más extendido debido al alto número de automatismos de estas industrias. En cambio, en las demás áreas el nivel de uso del autómata programable no sigue siendo del todo bueno, en comparación con otros países de la CEE.

En industrias ya establecidas, surge un problema a la hora de la incorporación de los autómatas programables. Normalmente el cambio desde los automatismos convencionales hacia los autómatas se hace de manera progresiva. Este hecho dificulta la integración de los aparatos de automatización nuevos y antiguos a la hora de la coexistencia en la misma planta.

La evolución tecnológica en el campo de los automatismos está dirigida a subsanar este problema. Se tiende a crear unos sistemas de control en los que distintos autómatas sean compatibles entre sí. No obstante, esta tarea es dificultosa porque conlleva la puesta de acuerdo y estandarización de distintos fabricantes de autómatas con intereses contrapuestos.

El autómata programable se ha impuesto sobre el automatismo convencional en que los primeros aportan una serie de soluciones que lo hacen más ventajoso

Entre estas ventajas están:

- Los autómatas programables están diseñados y construidos para su aplicación en ambiente industrial.
- Flexibilidad por su carácter programable.
- Facilidad de instalación y reutilización.
- Facilidad de mantenimiento y localización de averías.
- Posibilidad de empleo en múltiples tipos de tareas de control.

- Facilidad de control en la tarea global de control o sistema de producción integrado.

2. ELECCIÓN DEL AUTÓMATA

La adopción de una determinada tecnología para la automatización de una máquina o proceso requiere considerar no solo los aspectos funcionales sino también la interrelación presente y futura con otros sistemas de control, aspecto que como se comentó en el apartado anterior puede presentar problemas de integración.

Entre los factores a considerar se tiene:

- Factores cuantitativos.- Se refieren a la capacidad del equipo para soportar todas aquellas características específicas para el sistema de control, y se pueden agrupar en:
    - Entradas/salidas.- Cantidad, tipo prestaciones...
    - Tipo de control.- Centralizado, distribuido...
    - Memoria.- Cantidad, tecnología expandibilidad...
    - Software - Potencia del lenguaje de programación, facilidad de uso...
    - Periféricos.- Ampliaciones, equipos de programación, diálogo hombre-máquina...
    - Físicos y ambientales.- Características constructivas, temperatura...
- Factores cualitativos.- Factores menos tangibles, fundamentalmente económicos, que se ocultan en las mismas características del equipo y en las del fabricante o suministrador del autómata.
    - Ayudas al desarrollo del programa - Herramientas de programación, documentación...
    - Fiabilidad del producto.- Tiempo medio entre fallos, experiencia de otros usuarios e instalaciones
    - Servicios del suministrador.- Soporte Técnico, cursos de formación, información técnica...
    - Normalización en planta.- Familias de productos...

3. TIPOS DE AUTOMATIZACIÓN

Históricamente han sido dos los tipos de sistemas de automatización empleados. En un principio se tendió a implementar sistemas de automatización centralizados. En la actualidad se tiende a realizar los mismos de manera distribuida. A continuación se describen brevemente las características de uno y otro sistema:

3.1. SISTEMA DE AUTOMATIZACIÓN CENTRALIZADO

En este modo de realización se utiliza un solo autómata para automatizar todo el proceso. Esto conlleva, que para sistemas de gran tamaño, el autómata programable empleado deba ser de gran potencia e incorporar una gran cantidad de entradas y salidas.

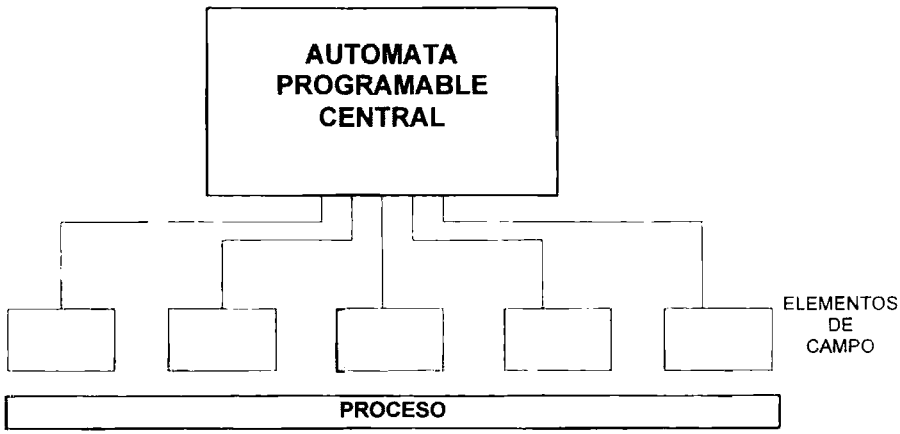


Figura 7.1: Sistema de automatización centralizado.

Del mismo modo el programa de automatización será muy extenso y por lo tanto su tiempo de ejecución grande. Esto mismo conlleva que una vez implementado el sistema y realizado el programa sea complicado la modificación o ampliación del mismo porque llevaría consigo no solo la modificación de la parte afectada sino también de todo su entorno.

3.2. SISTEMA DE AUTOMATIZACIÓN DISTRIBUIDO

En este tipo de sistemas se utilizan varios autómatas programables, cada uno de ellos encargado de la automatización de una parte específica del proceso. Son autómatas de menor potencia que los utilizados en el tipo centralizado, con una menor cantidad de entradas y salidas. Como cada uno de los autómatas se encargan de una parte del proceso, los programas de automatización serán de menor tamaño, ejecutándose con mayor rapidez.

No obstante, para que todas las partes específicas del proceso creen una sola unidad superior, cada uno de los autómatas encargados de cada parte debe estar comunicado con otros autómatas de su mismo nivel, así como con los de nivel superior, que pueden ser autómatas o PC que se encargarían de supervisar su tarea.

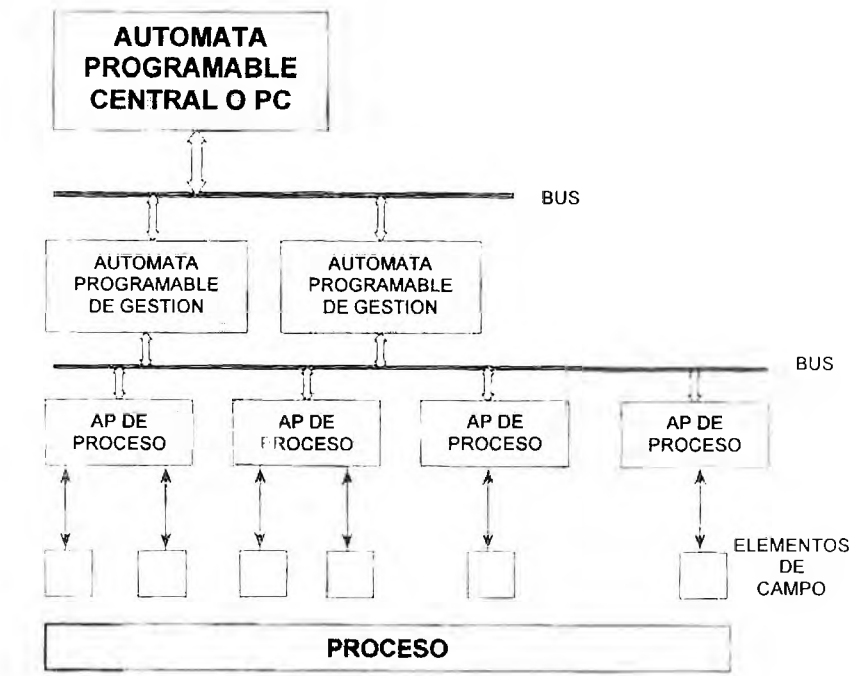


Figura 7.2: Sistema de automatización distribuido.

Todas estas comunicaciones se realizan mediante distintos tipos de bus dependiendo del nivel de automatización. Se consigue así una estructura jerarquizada desde el nivel superior, dirección, y el nivel inferior, nivel de campo.

Lo anterior hace que este tipo de organización sea fácilmente modificable o ampliable, ya que cualquier forma de modificación sólo afectaría a una parte en concreto de todo el proceso.

4. AUTOMATIZACIÓN CUBRIENDO TODOS LOS NIVELES

Se ha visto en el apartado anterior que la forma de automatización más recomendable será la automatización distribuida, sobre todo en sistemas extensos. Este tipo de sistemas se organiza en forma piramidal con una jerarquía clara que distingue entre distintos niveles.

Cada equipo tiene una asignación puntual y claramente delimitado de tareas dentro de su nivel, así como unas interfaces definidas para el intercambio de datos con los otros equipos.

Esta organización piramidal esta constituida fundamentalmente por los siguientes niveles:

- Nivel de gestión de la empresa.-

Este es el nivel superior. Aquí se encuentran computadoras que se encargan de funciones administrativas y comerciales para todo el proceso. Estos entregan datos primarios (por ejemplo: cantidad y tipo de producto a fabricar) a los equipos en los niveles inferiores y, partiendo de los datos adquiridos en dichos niveles, confeccionan estadísticas para los encargados de dirigir el proceso.

- Nivel de control de la producción.-

Es el nivel inmediatamente inferior. Es el nivel encargado de gobernar la totalidad del proceso. Aquí se recogen todos los datos referentes al proceso y proporcionados por los niveles inferiores. Estos datos se filtran y acondicionan para enviarlos al nivel superior. Aquí también se visualiza la evolución del proceso pudiéndose actuar manualmente sobre él. Se evalúan los mensajes de alarmas, perturbaciones, etc.

Este nivel puede estar constituido por un PC, que puede coincidir con el situado en el nivel de gestión, o también por un autómata de la gama alta.

- Nivel de control de grupos de automatización.-

En este nivel se encuentran autómatas que gestionan áreas del proceso interrelacionadas tecnológicamente (por ejemplo: líneas de montaje, máquinas completas...). Este nivel recibe información del nivel superior referentes al tipo de tarea a realizar, eventos ocurridos... Asimismo estos autómatas recogen toda la información del proceso en su área y los entrega al nivel superior.

- Nivel de mando y regulación.-

Este nivel es el inferior e integra a los autómatas a pie de proceso. Estos autómatas son los que están realmente en contacto con las máquinas, consultan los sensores del proceso, y en función del programa establecido, controlan los actuadores y señalizadores.

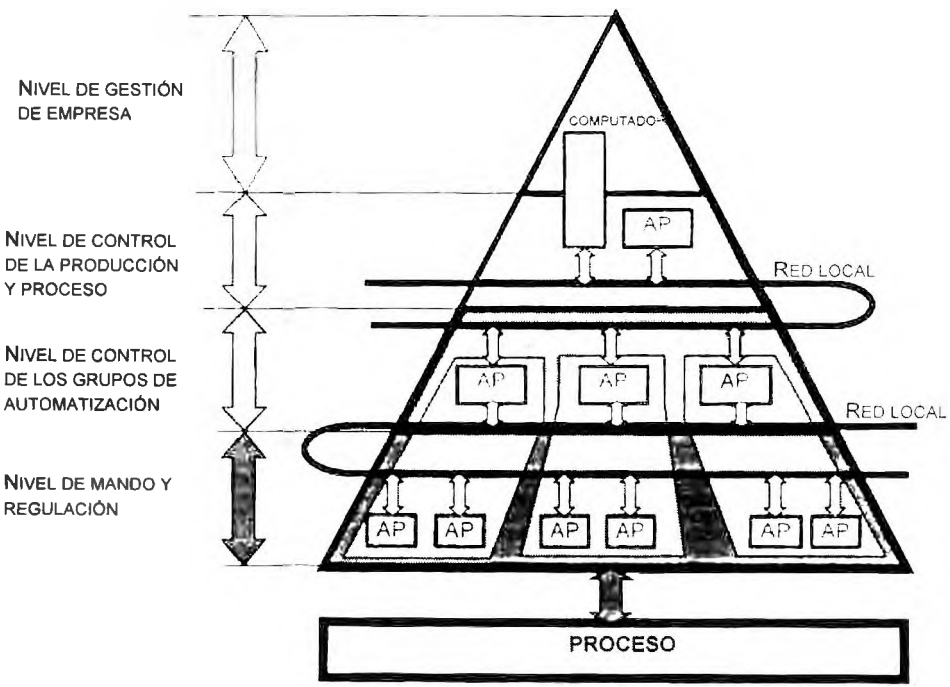


Figura 7.3: Automatización cubriendo todos los niveles.

El intercambio de datos entre los distintos autómatas pertenecientes a un mismo nivel o a niveles diferentes se realiza mediante redes locales, buses que serán distintos según el nivel.

5. CONEXIÓN DEL AUTÓMATA A BUS

Como ya se ha comentado anteriormente, la conexión entre distintos equipos en un sistema de automatización distribuido se realiza por medio de bus.

No obstante, existe gran diversidad en cuanto al tipo de bus a utilizar, dependiendo fundamentalmente del nivel del equipo en la pirámide de automatización. Así en los niveles inferiores se utilizarán los llamados bus de campo (buses con gran resistencia a perturbaciones), utilizándose en los niveles superiores los llamados bus de oficina o administración (buses diseñados para intercambiar gran cantidad de datos).

A la diferenciación anterior hay que añadir también otra característica que dificulta la integración de equipos de distintas marcas. Cada fabricante de autómatas programables diseña sus autómatas para que se comuniquen mediante sus propios buses. Así cada fabricante dispondrá de sus propios buses para los distintos niveles con sus propios protocolos de comunicación. Aunque en los distintos fabricantes la filosofía de los distintos buses es la misma no se rigen mediante un estándar común, cada uno tendrá sus propias características. De este modo para conectar equipos de distintas marcas en un mismo bus habrá que disponer de tarjetas interfaces que adapten los protocolos de comunicación, si estas tarjetas existen.

Independientemente del fabricante del equipo y bus, las características de los mismos en los distintos niveles suelen ser parecidas: Así en niveles superiores, gestión y administración, se utilizan los buses estándar para la transmisión de información en oficinas, como por ejemplo el protocolo Ethernet Estándar. La comunicación entre PCs de un nivel superior y autómatas de un nivel inferior se realiza mediante protocolo Ethernet industrial, Profibus, Sinec H1... La comunicación entre los autómatas del nivel de grupos de automatización y los autómatas del nivel de mando se realiza utilizando Profibus DP, Sinec L1... Por último el nivel de inferior comunicación de autómatas con el proceso, el llamado nivel de campo, se realiza mediante protocolos Profibus PA, AS-i..

La estructura anterior es la propuesta para sistemas de automatización distribuida basada en autómatas Siemens.

Para otro fabricante, los buses utilizados en cada nivel tendrán las mismas características aunque no serán del todo compatible. Por ejemplo, los buses empleados por la firma Telemecanique serán, en orden de nivel superior a inferior: Ethernet, Ethway, Mapway, Fipway, Telway, Uni-telway...

6. OTROS MIEMBROS DE LA FAMILIA S5

El autómata S5-95U forma junto con el S5-90U y el S5-100U la gama baja-media de la familia S5. Este tipo de autómata es compacto, o sea, forman un bloque único. Los autómatas de la gama medio-alta, S5-115U, S5-135U y S5-155U, cambian su filosofía de montaje. El cuerpo base de estos autómatas son un bastidor donde se conectan tarjetas según las necesidades. Hay que tener en cuenta que todos los elementos, incluidos alimentación y CPU tienen esta filosofía de montaje.

El aparato central (bastidor) contiene puestos de enchufe para las siguientes tarjetas:

- Tarjetas centrales (CPU).- hasta un máximo de 4 de ellas en paralelo (para los S5-135U y S5-155U). Por lo tanto este autómata puede funcionar en modo "multiprocesador".
- Tarjetas periféricas - Interface física a la máquina o proceso (entradas y salidas) o para realizar distintas funciones.
- Aparatos de ampliación.- Son nuevos bastidores iguales al aparato central que alojan las tarjetas periféricas que no entran en el aparato central. Los aparatos de ampliación se unen al aparato central mediante un bus periférico.
- Procesadores de comunicaciones - Permiten al autómata intercambiar información con otros equipos por medio de un acoplamiento en serie.

6.1. OPERACIÓN EN MODO MULTIPROCESADOR

Los autómatas S5-135U y S5-155U pueden trabajar hasta con 4 tarjetas centrales (CPU) en paralelo. Este modo de funcionamiento se denomina "multiprocesador" a diferencia del modo "monoprocesador", una sola tarjeta central.

Cuando el autómata trabaje en multiprocesador necesitará un procesador de coordinación que controle el acceso de las tarjetas centrales al bus periférico común. Cada tarjeta puede trabajar independientemente de las restantes, siempre que lo haga en su propio espacio de direcciones. Sin

embargo, cada tarjeta puede acceder a la periferia siempre que lo habilite el procesador de coordinación, hecho que ocurre cíclicamente cada dos microsegundos.

Cada tarjeta puede direccionar cualquier entrada o salida. Pero también se puede restringir el acceso de una tarjeta central a una entrada o salida determinada. Para ello se creará un módulo de datos en el que se asocie cada entrada o salida a una tarjeta central.

El intercambio de información entre tarjetas centrales se realiza a través de las marcas de acoplamiento. Estas son marcas normales que han sido asignadas mediante un módulo de datos a una tarjeta central. Estas marcas serán consideradas como entradas o salidas físicas por el procesador de coordinación. Las leerá al principio de cada ciclo y las pondrá al alcance de las tarjetas centrales.

6.2. TARJETAS PERIFÉRICAS

Mediante estas tarjetas se amplía la potencialidad del autómata. Pueden ser de muchos tipos, según la función de las mismas. Básicamente se dividen en:

- Entradas-salidas.- Nexo de unión entre la tarjeta central y el proceso. Pueden ser digitales, analógicas, con o sin separación galvánica...
- Tarjetas Preprocesadoras de señal.- También llamadas tarjetas inteligentes (IP). Se utilizan para realizar funciones que el procesador central del autómata no puede realizar a la velocidad deseada, ya que tiene que ocuparse de las operaciones propias del control. De esta manera se amplía considerablemente el campo de aplicación de los autómatas programables. El tratamiento de las señales procedentes del proceso se realiza fundamentalmente en la tarjeta periférica correspondiente, para lo cual dispone normalmente de procesador propio. La tarjeta periférica puede parametrizarse y controlarse desde el programa del procesador central a través de módulos funcionales (FB).

  Las tarjetas inteligentes disponibles en la familia S5 son:

  - Lectura digital de recorrido IP241.
  - Contadores IP242 e IP242A
  - Tarjeta analógica IP243.
  - Tarjeta reguladora de temperatura IP244.
  - Mando de válvulas IP245.
  - Tarjeta de posicionamiento IP246

- Tarjeta reguladora IP252 e IP260
- Procesador de E/S IP257 y
- Tarjeta dosificadora IP261.

6.3. PROCESADORES DE COMUNICACIONES

Estas tarjetas (CP) se utilizan para conectar al autómata con un equipo exterior, desde impresoras, pantallas de visualización, aparatos de almacenamiento externo, diagnosis del proceso conexión a red local (SINEC L1, SINEC H1).

- CP524/525.- Listado por impresora y acoplamiento a PC o autómata.
- CP530/535.- Acoplamiento a bus (SINEC L1, SINEC H1).
- CP527.- Visualización en monitor, operación por teclado y listado por impresora.
- CP513.- Almacenamiento externo.
- CP551.- Almacenamiento y proceso de datos.
- CP552.- Diagnosis del proceso.

7. HERRAMIENTAS Y SOFTWARE DE PROGRAMACIÓN

Los autómatas de la familia SIMATIC S5 se programan, como se ha visto a lo largo del presente libro, en el lenguaje de programación STEP5, en sus tres representaciones, plano de contactos, lista de instrucciones o plano de funciones. Se ha visto también cómo se utiliza este lenguaje y sus tres representaciones para realizar una programación secuencial sistemática de cualquier sistema utilizando la teoría de las Redes de Petri. Esta combinación de elementos nos proporciona una potente herramienta en la programación secuencial y nos proporciona un modelo para cualquier sistema.

El método visto aquí sería un método general. Existe otro método para la programación de sistemas secuenciales derivado del método de las Redes de Petri y simplificando el mismo. Este método es el denominado GRAFCET (Gráfico Funcional de Control Etapa Transición).

Este método no dispone de todo el potencial visto en las Redes de Petri, pero es utilizado a menudo en la programación de autómatas incluyéndolo

el fabricante de algunos tipos de autómatas en el software de programación de los mismos.

Como se ha dicho anteriormente el GRAFCET es una simplificación de las Redes de Petri, por lo tanto utiliza parte de su nomenclatura. No obstante hay algunas leyes que se modifican.

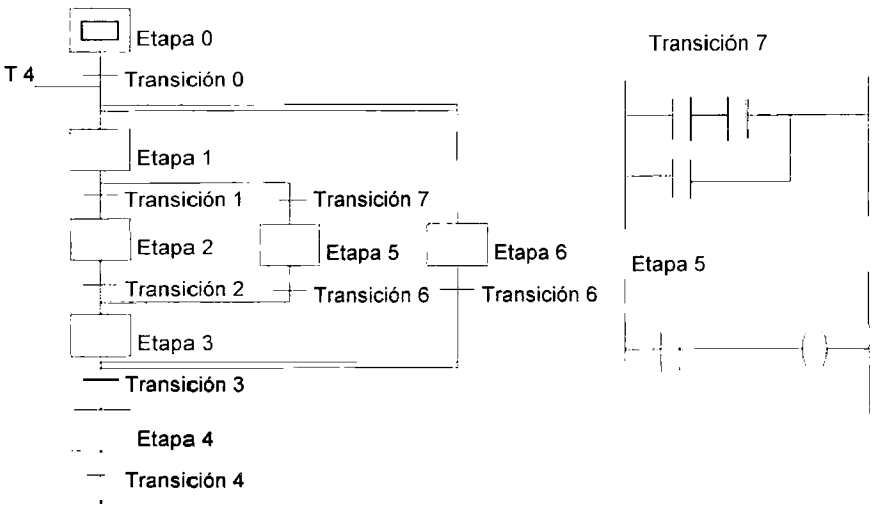


Figura 7.4: GRAFCET.

Las leyes que rigen el GRAFCET son:

- El lugar en RdP pasa a representarse como un cuadrado y se llamará etapa.
- Las etapas iniciales tienen que ser activadas por defecto, y se representan mediante el símbolo que aparece en la figura 7.4.
- Una transición puede ser validada o no. Será validada en el momento en que todas las etapas inmediatamente anteriores sean activas, y cuando la receptividad asociada a la transición sea verdadera, ésta es entonces obligatoriamente franqueada.
- El paso por una transición implica la activación de todas las etapas inmediatamente posteriores y la desactivación de todas las etapas inmediatamente anteriores.
- Todas las transiciones simultáneamente franqueables deben ser simultáneamente franqueadas.
- Si una misma etapa debe ser activada y desactivada simultáneamente, ésta debe permanecer activa.

- Existen dos tipo de bifurcaciones:
    - Tipo Y.- Se activan dos o más etapas a la vez.
    - Tipo O.- Se activa solo una etapa.

Existe software de programación de autómatas en el mercado que utiliza este método. Dicho software utiliza la representación de contactos o funciones para simular la evolución en las transiciones o activar salidas en las etapas.

8. EL FUTURO DE LOS SISTEMAS AUTOMATIZADOS

El mercado de los sistemas automatizados mediante autómatas programables está siguiendo una evolución clara hacia la utilización de sistemas distribuidos. Todos los equipos de las nuevas generaciones están enfocados en esta dirección, por lo que son fácilmente conectables a bus.

Así toda la nueva familia de Siemens para autómatas SIMATIC-S7 y para sistemas de control de procesos SIMATIC PCS 7 está orientada hacia los sistemas distribuidos de automatización.

La familia de autómatas SIMATIC-S7 está formada por los autómatas S7-200, S7-300 y S7-400 que presentan un aspecto mas compacto y robusto que sus antecesores los S5. Son también más potentes e igualmente ampliables mediante bloques periféricos para toda la gama.

Su lenguaje de programación es el STEP7, evolución del STEP5.

Quizás la mayor ventaja de la nueva familia respecto a la antigua es su facilidad de integración en un sistema distribuido ya montado, conectando autómatas directamente a la red.

Aumenta la potencialidad de las redes los nuevos buses de campo. Entre éstos están el bus AS-i (Actuador-sensor interfase) y el Profibus PA (Bus de campo a dos hilos). A continuación se describen brevemente estos dos tipos de bus.

8.1. Bus AS-I

El bus AS-i, Actuador Sensor interfase es un sistema de red para sensores y actuadores binarios (o sea, todo o nada) para el nivel inferior de campo. Está destinado a la conexión de los elementos de control de nivel más bajo (autómatas de campo) con los actuadores y sensores del proceso.

La conexión utilizada es un cable simple de dos hilos normalizado, común para todos los elementos, en él se conectan todos los sensores o actuadores en paralelo. De esta manera se consigue eliminar todos los cables que en un principio unían las entradas y salidas del autómata con los distintos elementos.

Para el programa en el autómata no hay diferencia si los cableados se realizan de manera convencional o se utiliza el bus AS-i, únicamente necesitaría un módulo de ampliación especial denominado maestro AS-i CP2433, que se conectaría al autómata como si se tratase de una tarjeta periférica. De este módulo saldría el cable común de conexión para todos los elementos.

Por lo tanto se puede implementar este sistema en autómatas ya en funcionamiento porque esto no conllevaría cambios en los programas de automatización del mismo.

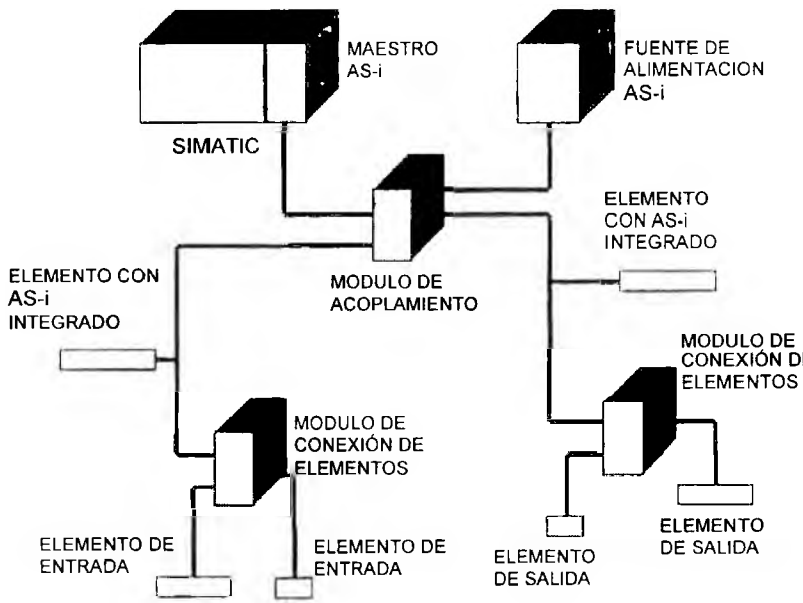


Figura 7.5: Bus AS-i.

8.2. PROFIBUS PA

El PROFIBUS-PA es un bus de campo para procesos industriales derivado del ya existente PROFIBUS (PROcess Field BUS) pero con modificaciones encaminadas a la automatización de procesos (PA).

Está especialmente diseñado, al igual que el AS-i, para la conexión de los autómatas de nivel inferior con los elementos actuadores y sensores del proceso, pero con la ventaja, respecto al anterior, de que se puede utilizar con elementos de campo digitales o analógicos.

El PROFIBUS PA es un bus a dos hilos y en el mismo cable se pueden conectar todos los elementos. Utiliza cable apantallado para la protección de los datos en ambientes industriales adversos y la técnica de transmisión de datos se ha optimizado para evitar posibles perturbaciones. Además tiene la ventaja de que la alimentación de los distintos elementos se realiza por los mismos cables de datos.

La realización de este bus necesita un elemento de enlace entre el bus PROFIBUS normal que conectaría los autómatas entre sí. El PROFIBUS PA conectaría el bus PROFIBUS con los elementos de campo.

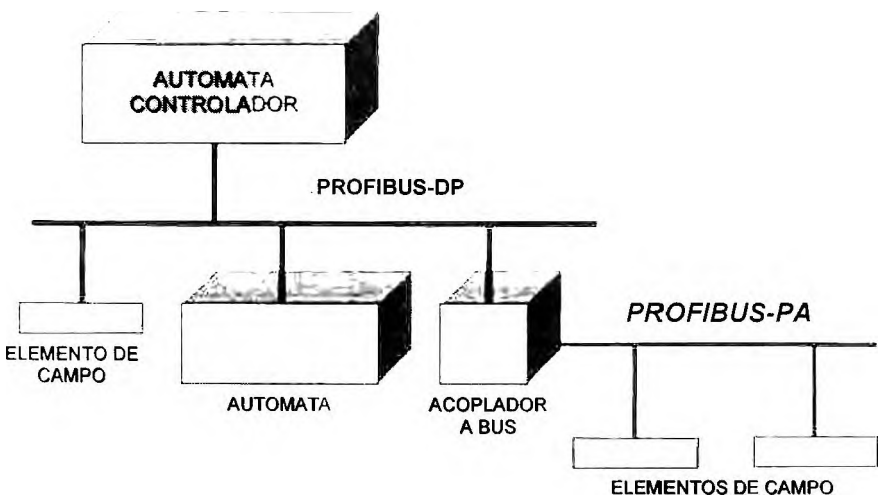


Figura 7.6: Bus PROFIBUS-PA.

SERVICIO DE PUBLICACIONES
UNIVERSIDAD DE CÁDIZ

ISBN 84-7786-566-3