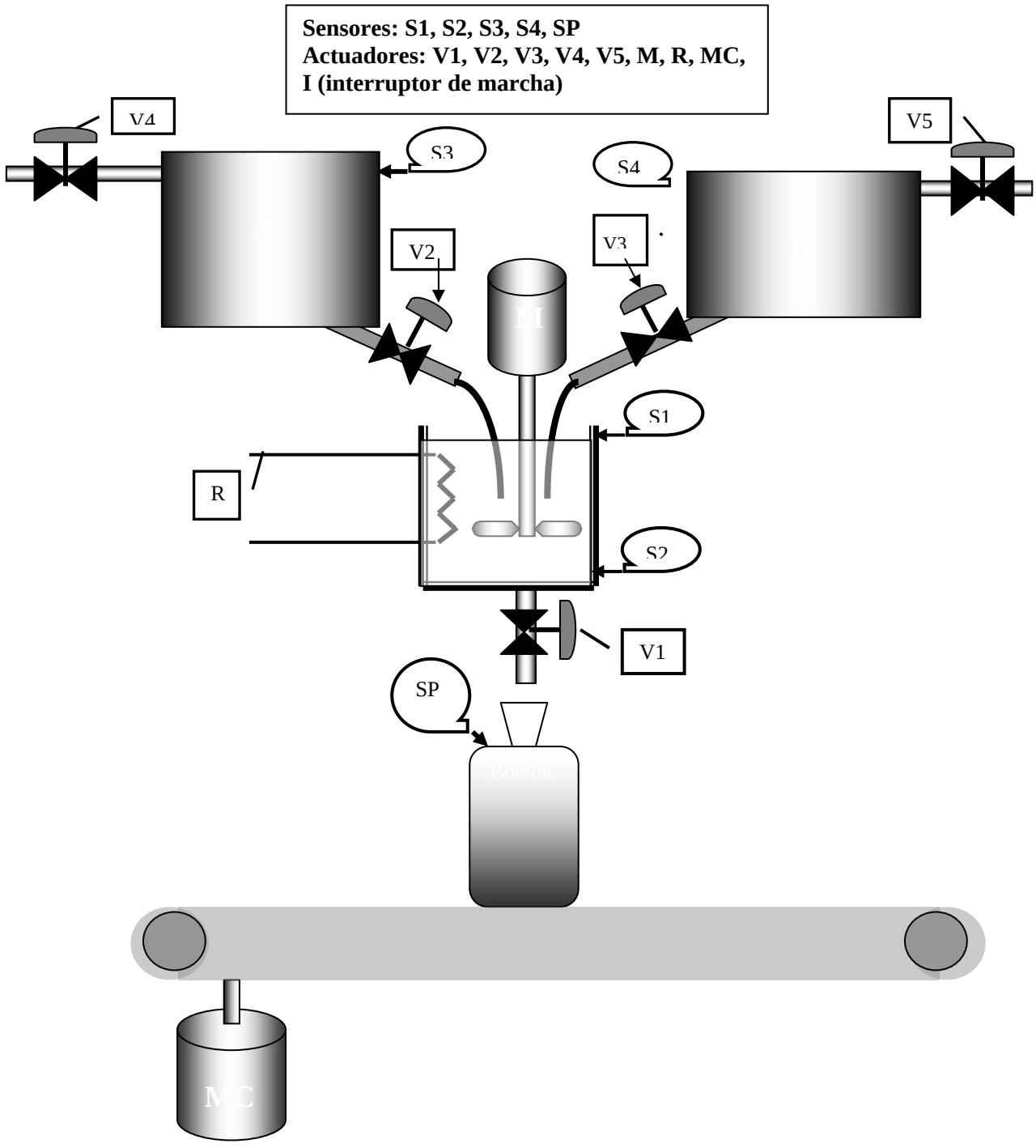
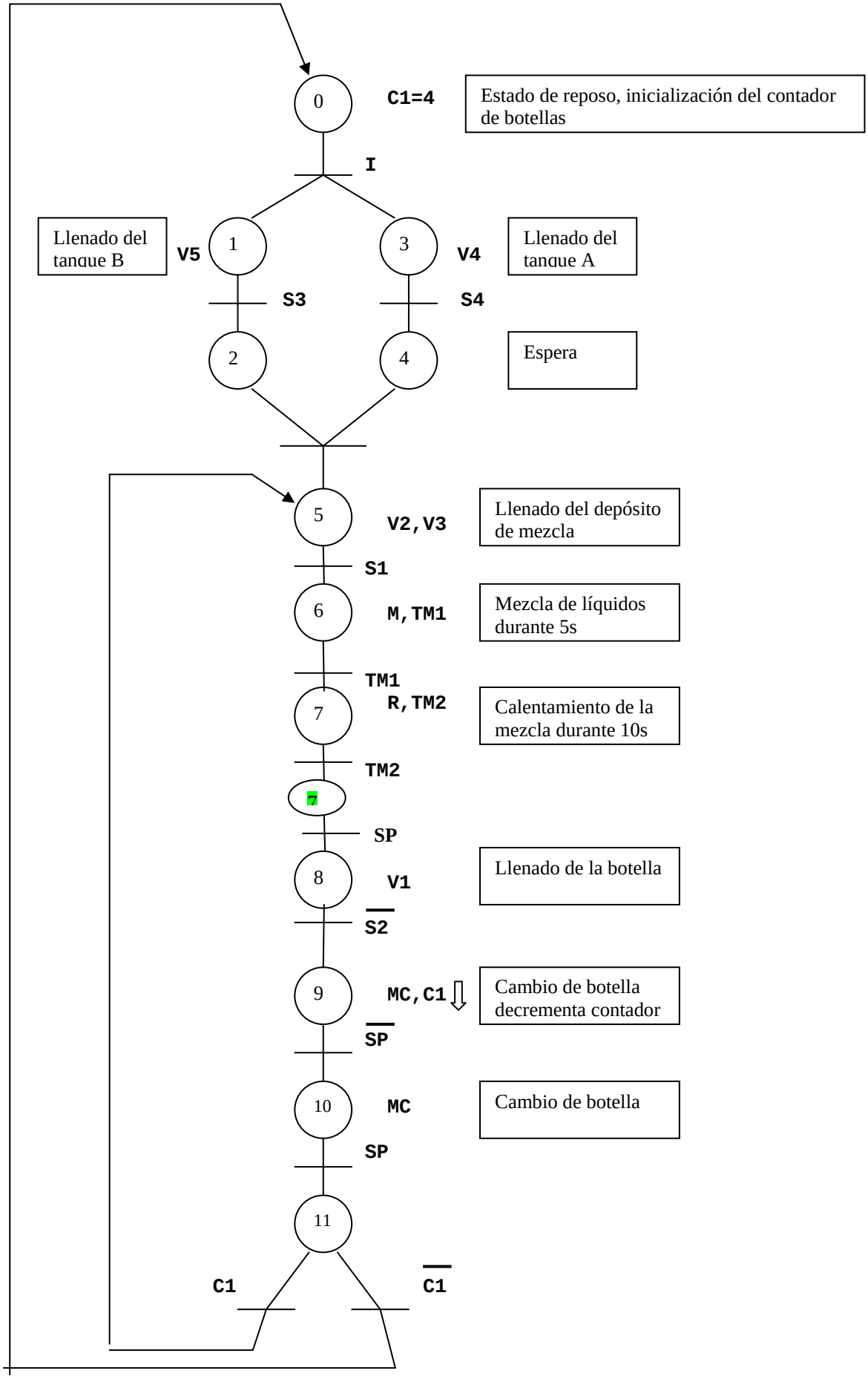
EJEMPLO DE DESARROLLO DE UN SISTEMA DE CONTROL Y PROGRAMACIÓN DE UN AUTÓMATA PROGRAMABLE DE LA FAMILIA SCHNEIDER TSX3710

El ejemplo ha sido extraído de los apuntes del profesor Fernando Castaño de la Universidad de Sevilla (asignatura Informática Industrial).

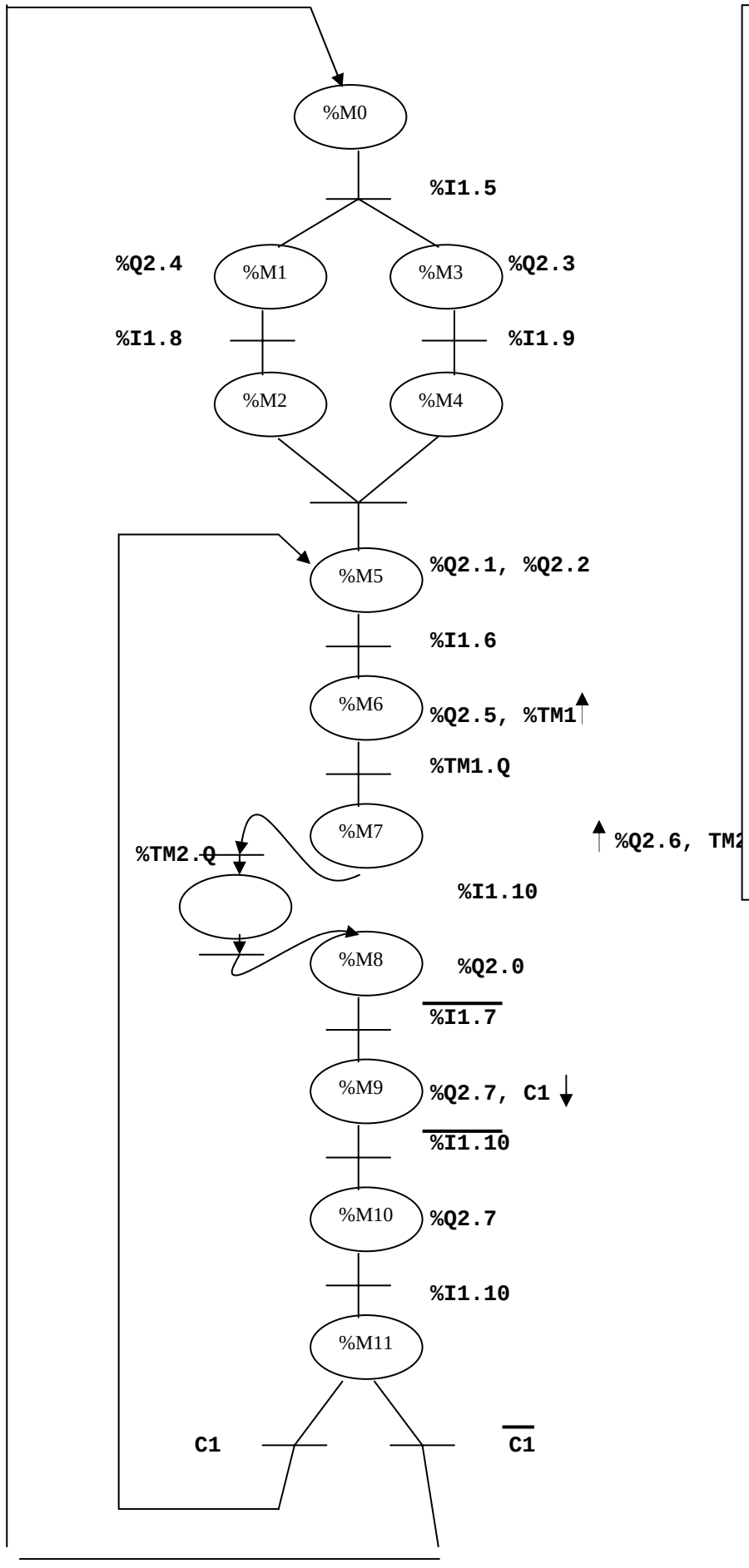
Se tiene una planta de embotellado que consta de dos depósitos grandes de acumulación de dos líquidos distintos. Los dos depósitos vierten sobre otro más pequeño con capacidad para una botella. En este depósito pequeño los líquidos se mezclan durante un tiempo de 5 segundos, luego se calienta la mezcla durante 10 segundos, y por último se vierte dentro de una botella. Las botellas son transportadas por una cinta transportadora hasta el punto de llenado. Una vez se hayan procesado 4 botellas, se procede al llenado de los depósitos grandes.



Planteamiento del problema con un enfoque de Redes de Petri (RdP)



Asignación de entradas, salidas y marcas internas



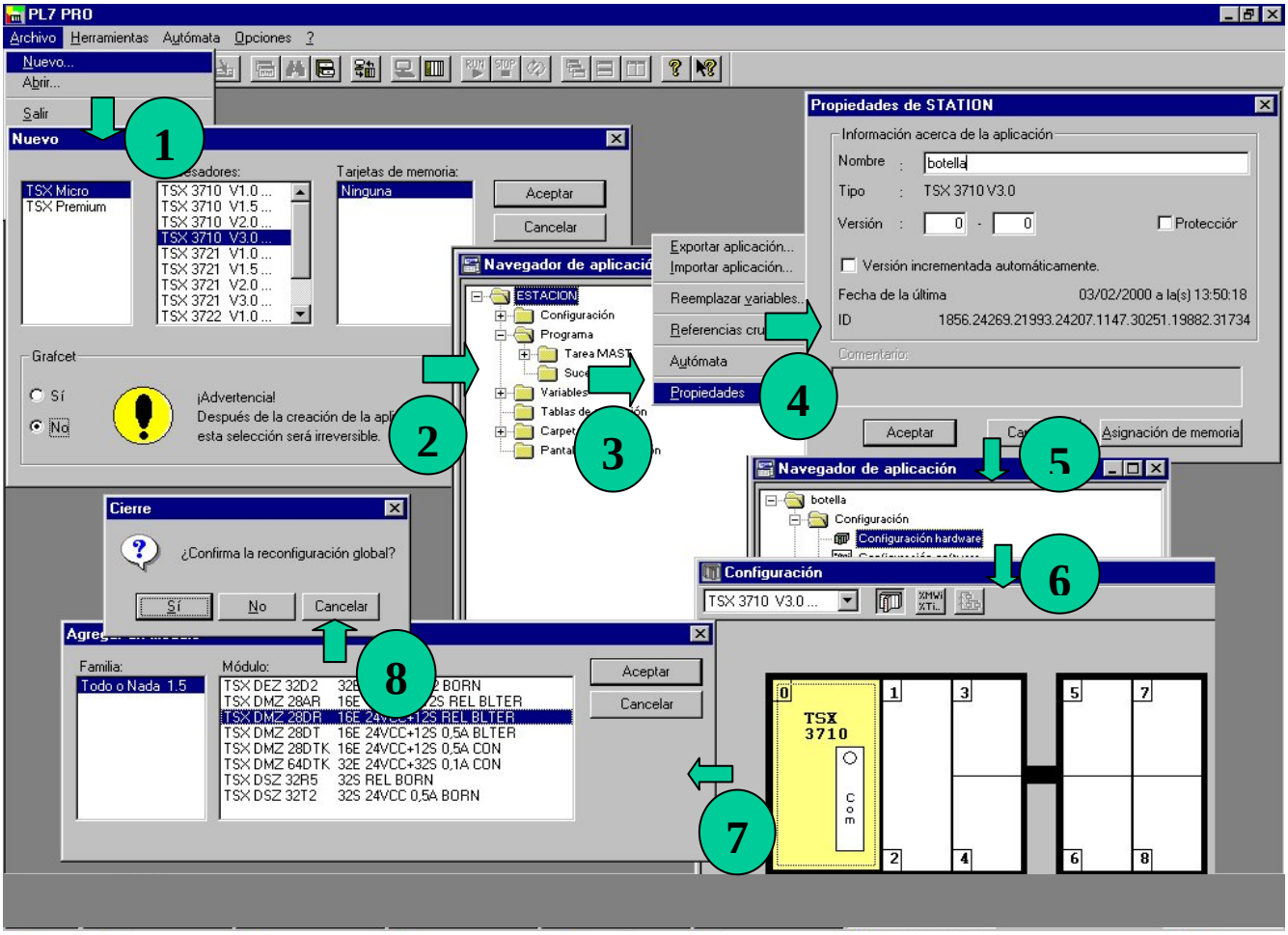
ENTRADAS	
I	%I1.5
S1	%I1.6
S2	%I1.7
S3	%I1.8
S4	%I1.9
SP	%I1.10
SALIDAS	
V1	%Q2.0
V2	%Q2.1
V3	%Q2.2
V4	%Q2.3
V5	%Q2.4
M	%Q2.5
R	%Q2.6
MC	%Q2.7

Módulos para el autómata Schneider-Telemecanique TSX37-10

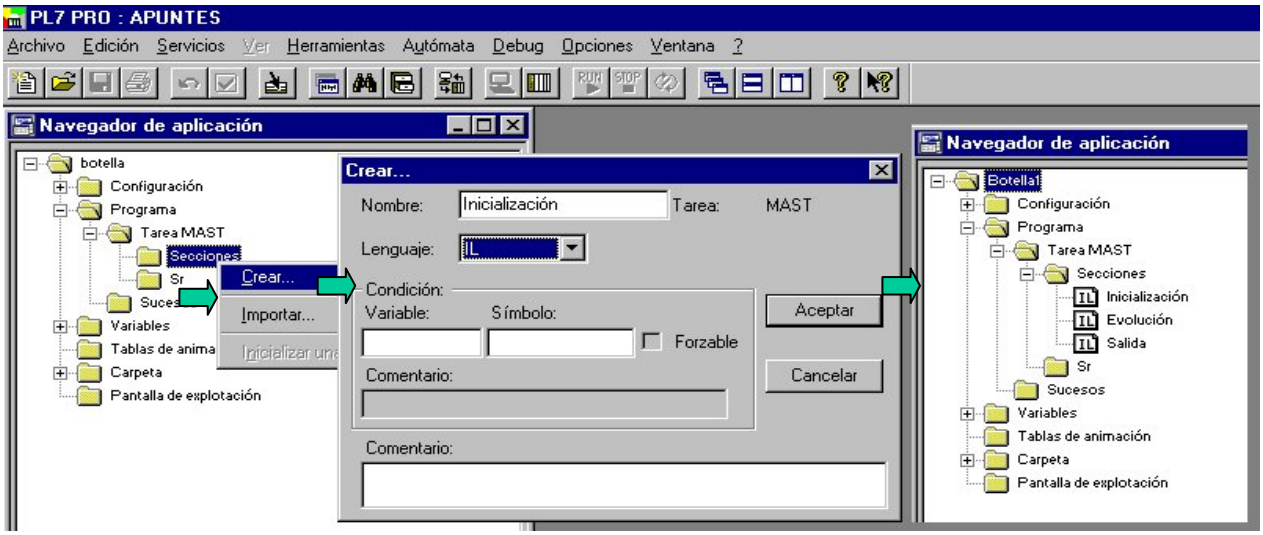
Programación mediante lista de instrucciones

En lo que sigue, se explica secuencialmente el proceso a seguir para realizar la programación mediante lista de instrucciones del ejemplo expuesto.

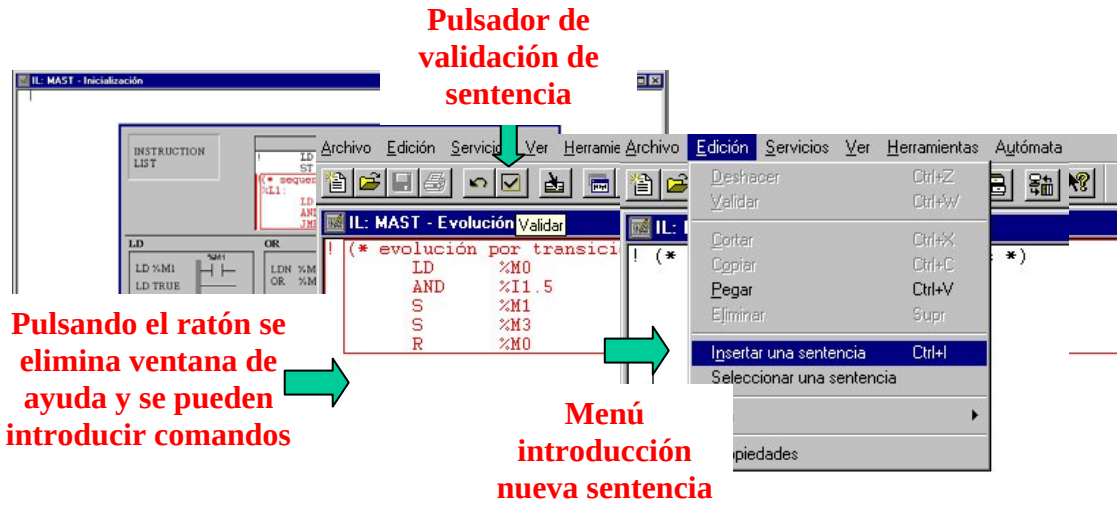
- *Paso 1. Arrancar el programa*:
  `Inicio ⇒ Programas ⇒ Modicon ⇒ Telemecanique PL7 Pro V3.1`
- *Paso 2. En el menú principal comenzar una nueva aplicación*:
  `Archivo ⇒ Nuevo ⇒ TSX Micro TSX 3710 V3.0 ⇒ Poner nombre a la aplicación en Navegador de aplicación (Propiedades) ⇒ Configurar el hardware para añadir el módulo de E/S digitales TSX DMZ 28DR 16E 24VCC+12S REL BLTER ⇒ Cerrar la ventana y confirmar reconfiguración global`



- *Paso 3. Programación de las secciones en la tarea MAST*: en el `Navegador de aplicación`, se abre la carpeta de la tarea MAST y sobre la subcarpeta `Secciones`, utilizando el pulsador derecho del ratón se selecciona crear una nueva. En el ejemplo se van a crear tres secciones: Inicialización, Evolución y Salida (repitiendo el proceso tres veces). Nótese que se escoge como lenguaje de programación IL (*Instruction List* o lista de instrucciones).



- *Paso 4: Programación de las distintas secciones*: haciendo doble-click con el ratón sobre las secciones recién creadas, se puede acceder a la pantalla de edición de las mismas. Es importante programar segmentos de código en distintas sentencias (empiezan con !), pues resulta más fácil ir validando sentencias cortas.



En el caso en que haya errores sintácticos, al pulsar el botón de validación de sentencia aparecerá un mensaje de error indicando que estamos programando una sentencia no válida en ese contexto. A continuación se indica el resultado final de la programación en lenguaje de lista de instrucciones del ejemplo propuesto (existen muchas alternativas distintas al diseño que se ha llevado a cabo).

En el módulo de inicialización, se lleva a cabo la puesta a cero o a uno de las distintas marcas de memoria utilizadas en la aplicación. Para ello, se comprueba en el primer ciclo el bit de sistema %S13, que precisamente nos indica si nos encontramos o no en el primer ciclo de programa.

En el módulo de evolución aparecen reflejadas los distintos estados por los que atraviesa la RdP mediante activación de las transiciones. En este ejemplo, se han utilizado dos temporizadores en modo TON (retardo con conexión) y un contador que va decrementando su valor preseleccionado a medida que recibe un pulso a la entrada. La preselección del tipo de temporizador y contador, valores de tiempo y contaje etc. se realiza mediante el editor de variables que se encuentra en el Navegador de aplicación.

En el módulo de salida se programan todas las etapas de salida (interacción con el exterior), resultado de la evolución del autómata.

Inicialización

IL: MAST - Inicialización

```
! (* Módulo de inicialización *)
%L1:
        LD      %S13
        S       %M0
        R       %M1
        R       %M2
        R       %M3
        R       %M4
        R       %M5
        R       %M6
        R       %M7
        R       %M8
        R       %M9
        R       %M10
! (* Fin de módulo *)
```

Evolución

IL: MAST - Evolución

```
! (* evolución por transiciones *)
        LD      %M0
        AND     %I1.5
        S       %M1
        S       %M3
        R       %M0
!       LD      %M1
        AND     %I1.8
        S       %M2
        R       %M1
!       LD      %M3
        AND     %I1.9
        S       %M4
        R       %M3
!       LD      %M2
        AND     %M4
        S       %M5
        R       %M2
        R       %M4
!       LD      %M5
        AND     %I1.6
        S       %M6
        R       %M5
```

Evolución (cont.)

IL: MAST - Evolución

```
! (* temporizador *)
        LD      %M6
        AND     %TM1.Q
        S       %M7
        R       %M6
!       LD      %M7
        AND     %I1.10
        AND     %TM2.Q
        S       %M8
        R       %M7
!       LD      %M8
        ANDN    %I1.7
        S       %M9
        R       %M8
!       LD      %M9
        ANDN    %I1.10
        S       %M10
        R       %M9
!       LD      %M10
        AND     %I1.10
        S       %M11
        R       %M10
!       LD      %M11
        AND     %C1.D
        S       %M5
        R       %M11
```

Evolución (cont.)

IL: MAST - Evolución

```
! (* Activación de contador - preselección en editor variables *)
        BLK     %C1
        LD      %M0
        S
        LD      %M9
        CD
        OUT_BLK
        LD      D
        ST      %Q2.11
        END_BLK
! (* Activación de temporizador 1 - valores en editor variables *)
        BLK     %TM1
        LD      %M6
        IN
        END_BLK
! (* Activación de temporizador 2 - valores en editor variables *)
        BLK     %TM2
        LD      %M7
        IN
        END_BLK
```

Salida

IL: MAST - Salida

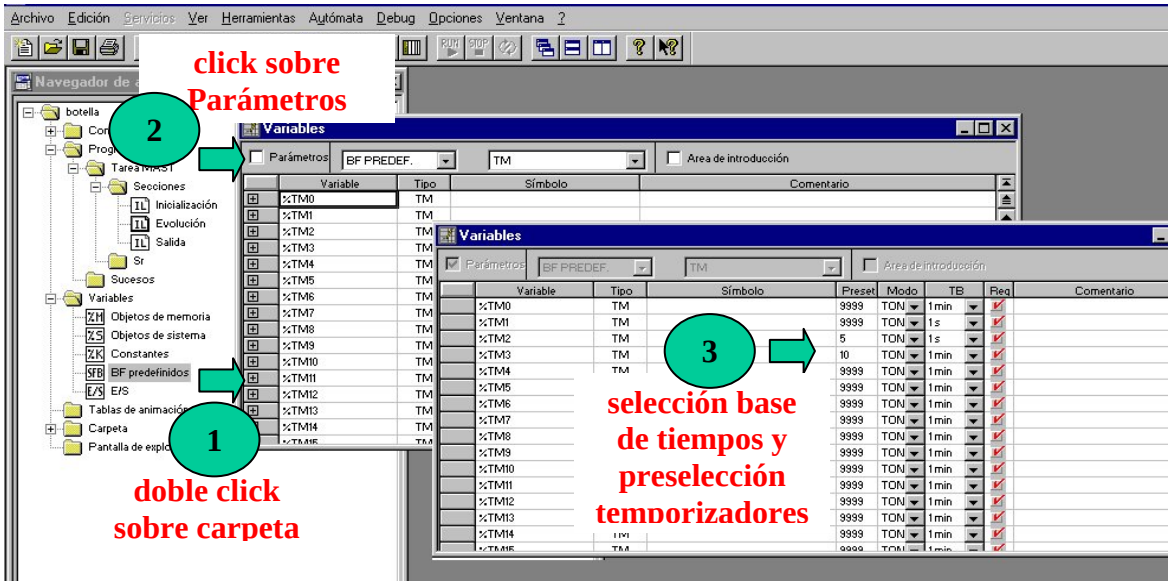
```
! (* etapa de salida *)
        LD      %M1
        ST      %Q2.4
        LD      %M3
        ST      %Q2.3
        LD      %M5
        ST      %Q2.1
        ST      %Q2.2
        LD      %M6
        ST      %Q2.5
        LD      %M7
        ST      %Q2.6
        LD      %M8
        ST      %Q2.0
        LD      %M9
        OR      %M10
        ST      %Q2.7
        LD      TRUE
        [%MW0:=%C1.V]
        S       %S69
        [%SW67:=16#00D0]
        [%SW68:=16#0300]
        [%SW69:=16#0000]
```

Las cuatro últimas líneas de program de la sección Salida son opcionales y permiten ver el valor del contador en formato decimal en los leds del autómata. ⇒

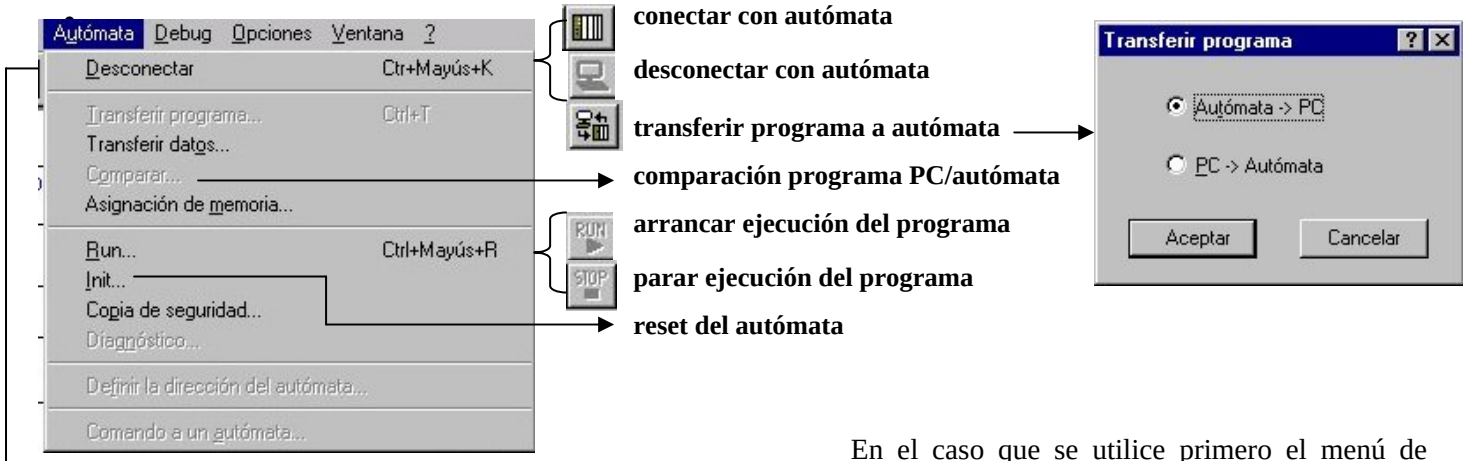
Como se ha indicado, antes de pasar a la ejecución del programa en el autómata, hay que configurar los valores predeterminados de temporizadores, contadores, etc., desplegando la carpeta `Variables` del Navegador de aplicación y seleccionando `BF predefinidos` (bloques de función predefinidos). En la siguiente figura se muestra el caso de los temporizadores (la configuración del contador se realiza de forma similar).

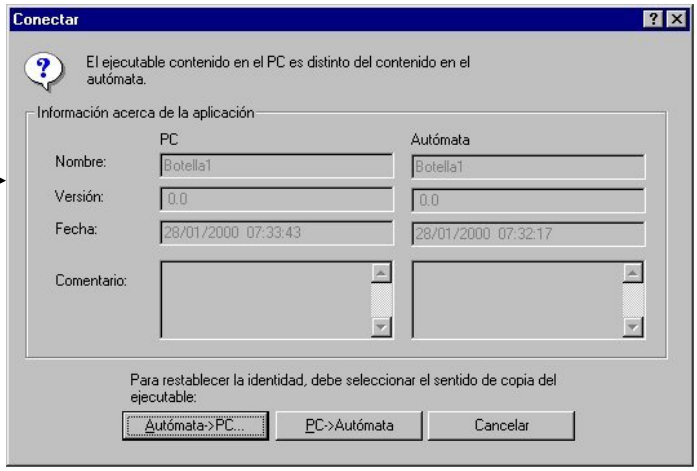
Nota: hay que inicializar el contador a 3. En el código anterior falta código del contador para pasar a %M0, que puede ser algo parecido a lo que sigue:

```
LD %M11
ANDN  %C1.D
S     %M5
R     %M11
LD    %M11
AND   %C1.D
S     %M0
R     % M11
```



- *Paso 5: Transferencia del programa al autómata*: para realizar esta función se pueden utilizar los botones de la barra de herramientas o las funciones del menú `Autómata`.

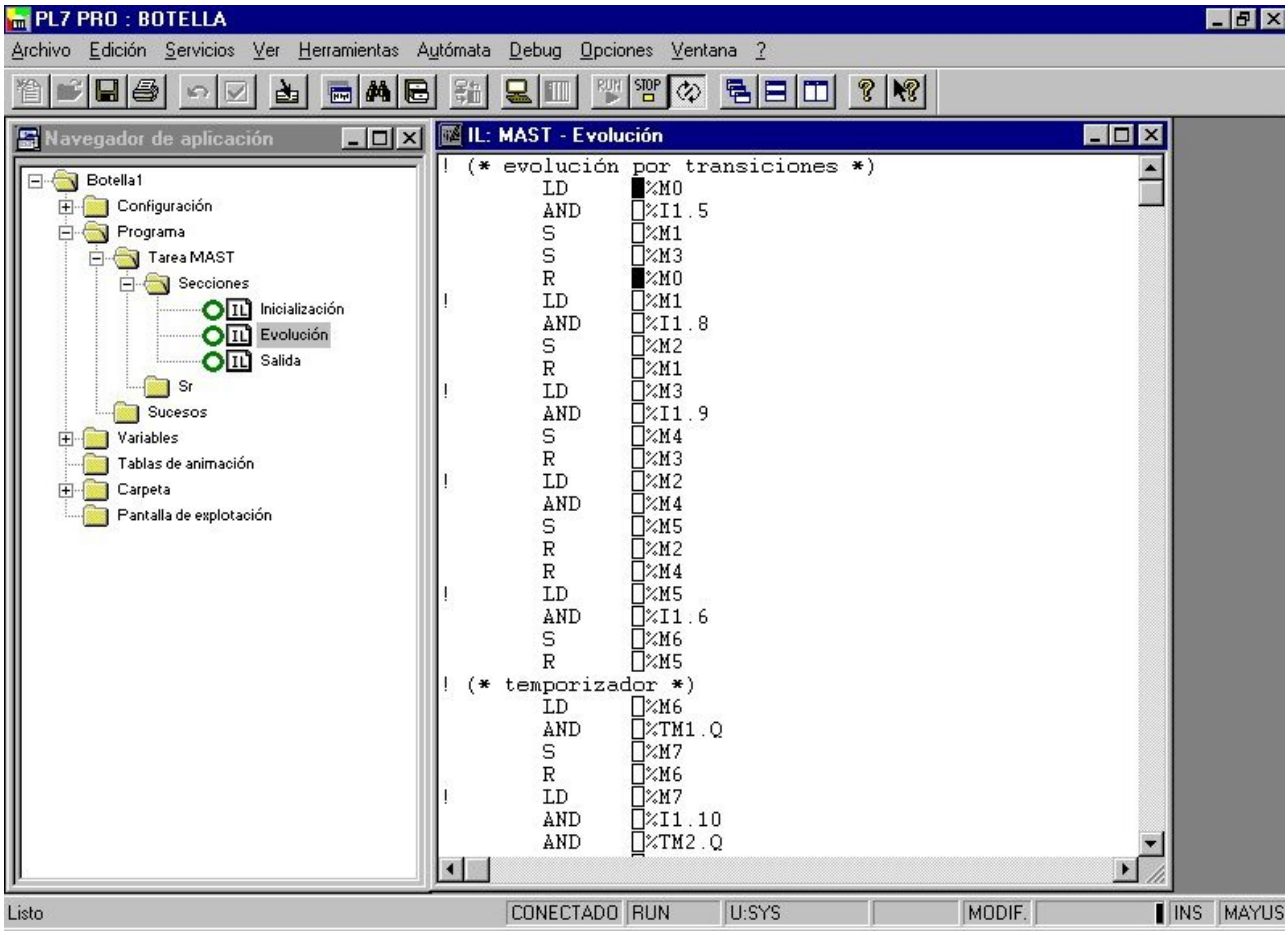




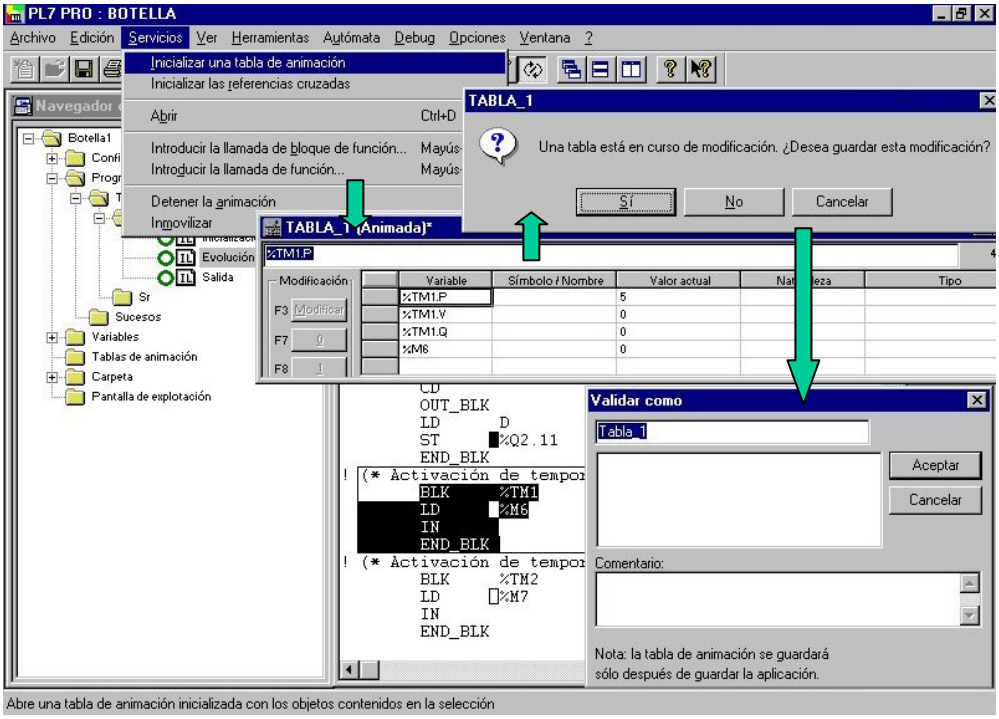
En el caso que se utilice primero el menú de transferencia de programa, posteriormente habrá que conectar el autómata con la finalidad de que se pueda pasar el mismo a modo RUN. Si se utiliza el menú de conexión con el autómata, directamente realiza las dos funciones (transferencia del programa y conexión).

En la barra inferior de mensajes del programa principal aparecerá un mensaje indicativo de la aceptación o no de la transferencia del programa al autómata. Una vez aceptada la transferencia al autómata y estando en modo conectado se puede pasar a modo RUN, de forma que se empezará a ejecutar el programa en el autómata. El estado de los leds luminosos en la carcasa del autómata indica el estado del mismo en todo momento.

Paso 6: Ejecución del programa: consiste en pasar el autómata a modo RUN. Las marcas de memoria que se van activando aparecen señaladas con un rectángulo relleno de negro, que da una indicación de la evolución de los estados del autómata a medida que se van franqueando las transiciones.

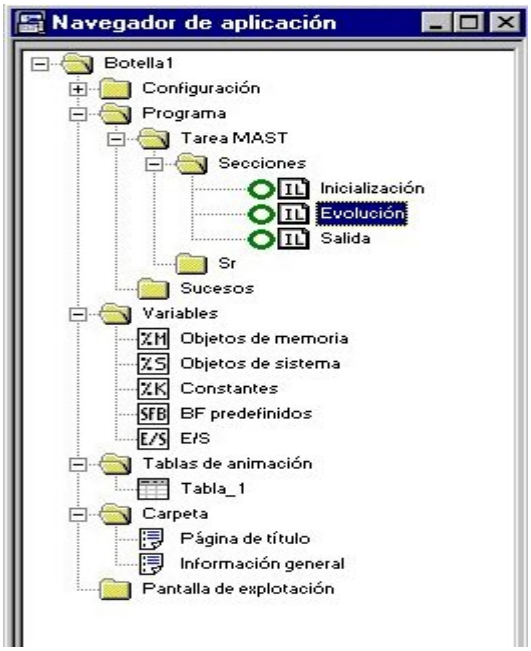


Existe una herramienta muy útil para comprobar la correcta evolución del autómata, que son las tablas de animación. Mediante la selección de un segmento de código (por ejemplo, el correspondiente a un temporizador), se puede analizar el valor de variables de memoria significativas a medida que evoluciona el autómata.



Se pueden crear distintas tablas correspondientes a distintos bloques funcionales, apareciendo todas en la carpeta correspondiente del Navegador de aplicación.

De este modo, el Navegador de aplicación contiene todos los elementos que componen la aplicación desarrollada. Se puede completar con la inclusión de información dentro de la subcarpeta `Carpeta`, donde se puede introducir una página de título e información general sobre la aplicación que se imprimirá, permitiendo documentar adecuadamente el programa desarrollado.

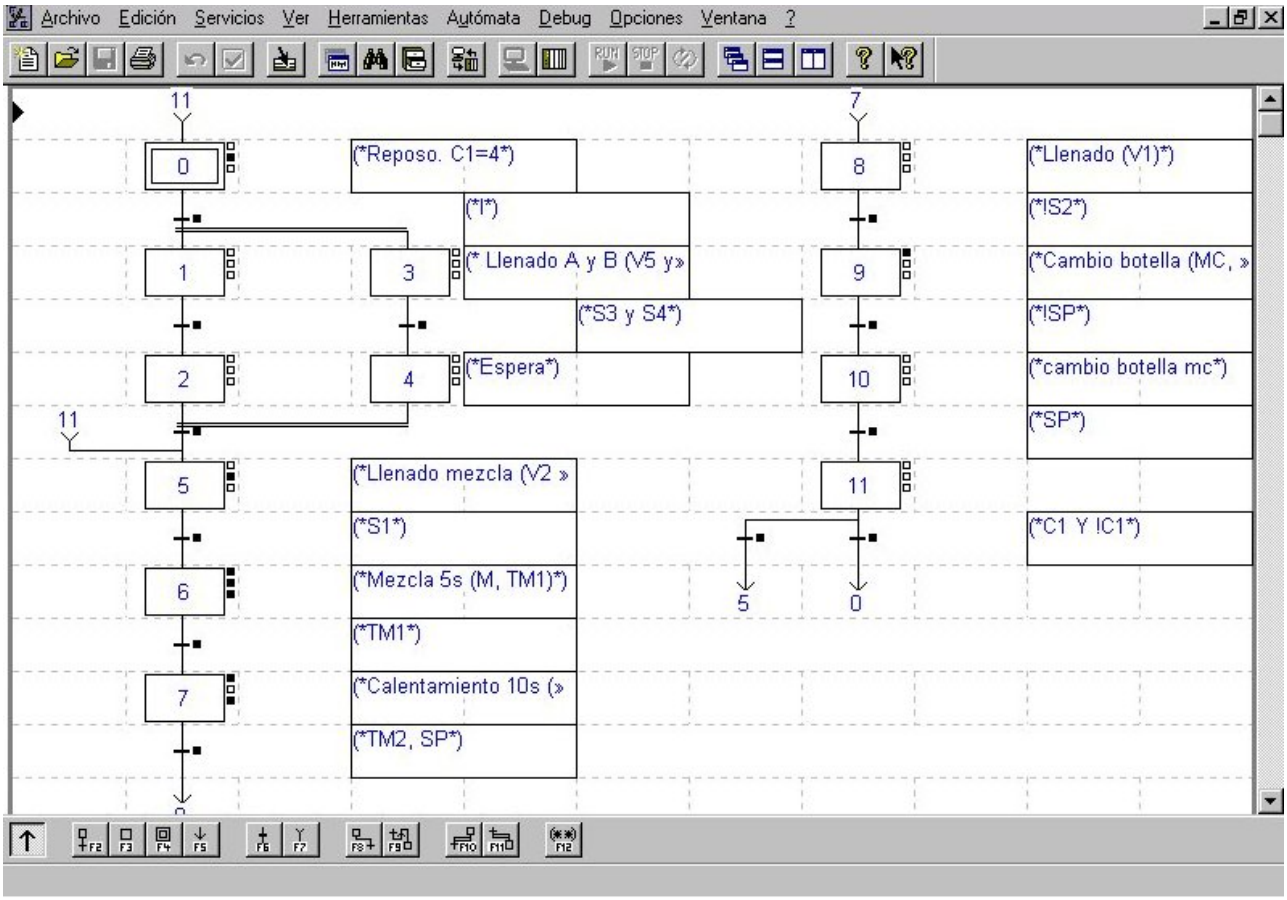


Programación mediante Gracet

Se van a describir en este apartado los pasos necesarios para realizar la programación del ejemplo propuesto en lenguaje Grafcet a partir de las RdP propuestas. Los pasos que son similares al ejemplo con lista de instrucciones no se van a describir en su totalidad por ser redundantes.

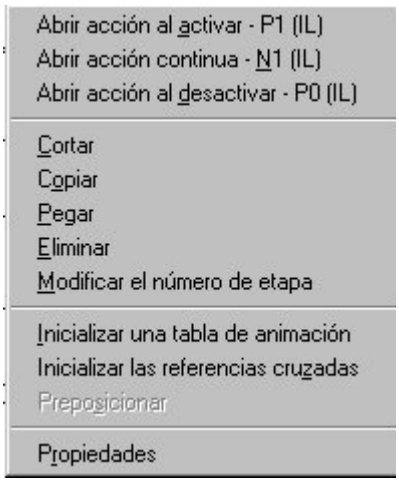
- *Paso 1. Arrancar el programa*: idem que en el caso anterior.
- *Paso 2. En el menú principal comenzar una nueva aplicación*: idem que en el caso anterior, con la salvedad de que hay que seleccionar la opción de Grafcet en el menú **`Nuevo`**.
- *Paso 3. Programación de las secciones en la tarea MAST*: en el `Navegador de aplicación`, se abre la carpeta de la tarea MAST y en la subcarpeta `Secciones`, aparecerán por defecto las secciones `Prl`, `Chart` y `Post`. En las secciones `Prl` y `Post` se pueden programar acciones preliminares y de actuación en lenguajes `LD`, `IL` y `ST`. El cuerpo del programa (evolución de estados y franqueo de transiciones) se programa en la sección `Chart` en Grafcet.

- *Paso 4: Programación de las distintas secciones*: la programación de las secciones `Prl` y `Post` se lleva a cabo de la misma forma que en el caso tratado en la programación mediante lista de instrucciones. La programación de la sección `Chart` se realiza de forma gráfica, utilizando el ratón y las funciones predefinidas. En lo que sigue se va a mostrar la programación utilizando marcas de memoria y posteriormente se utilizarán los bits propios de activación de las etapas de Grafcet, que permiten llevar a cabo una programación más simplificada.

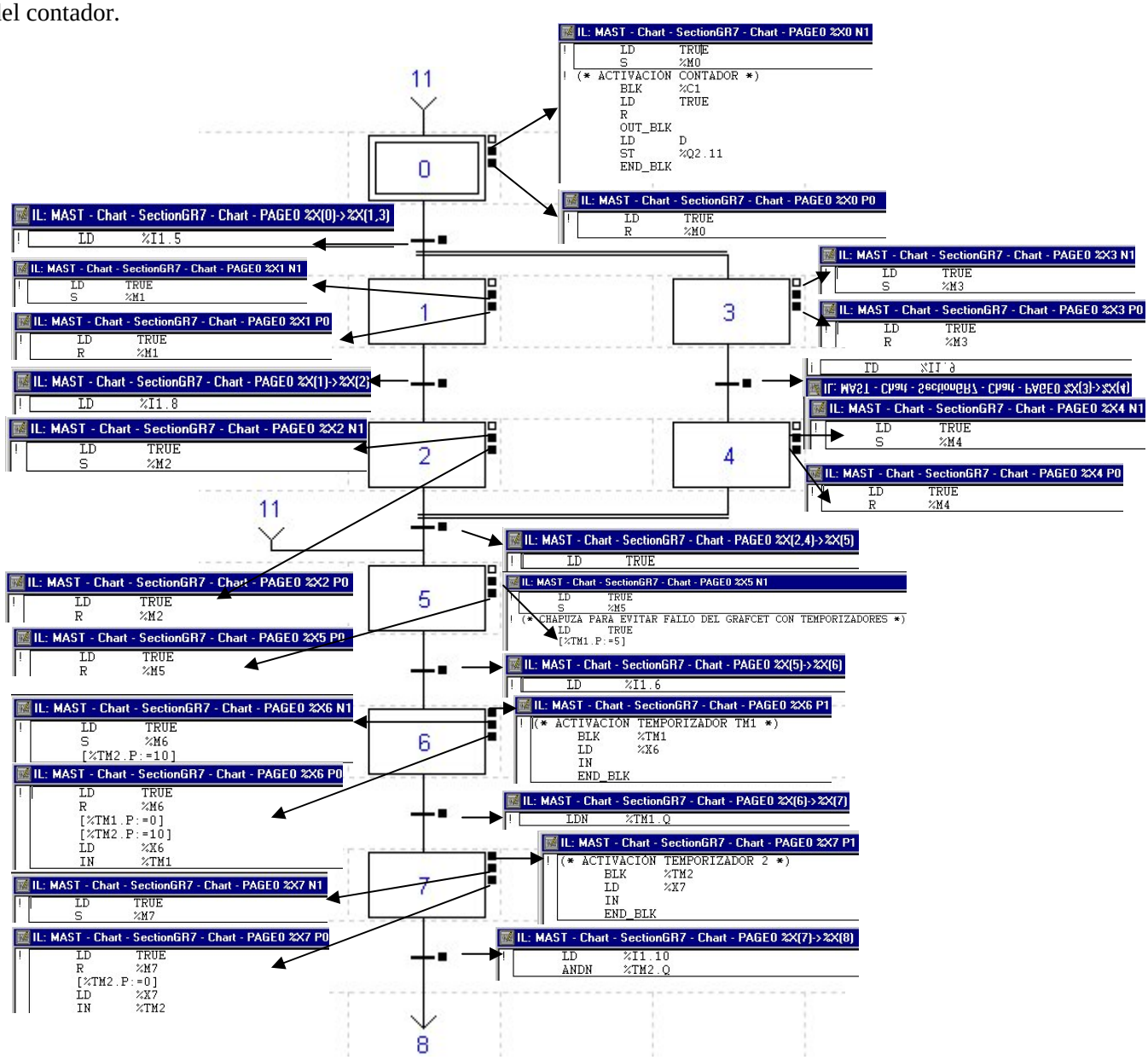


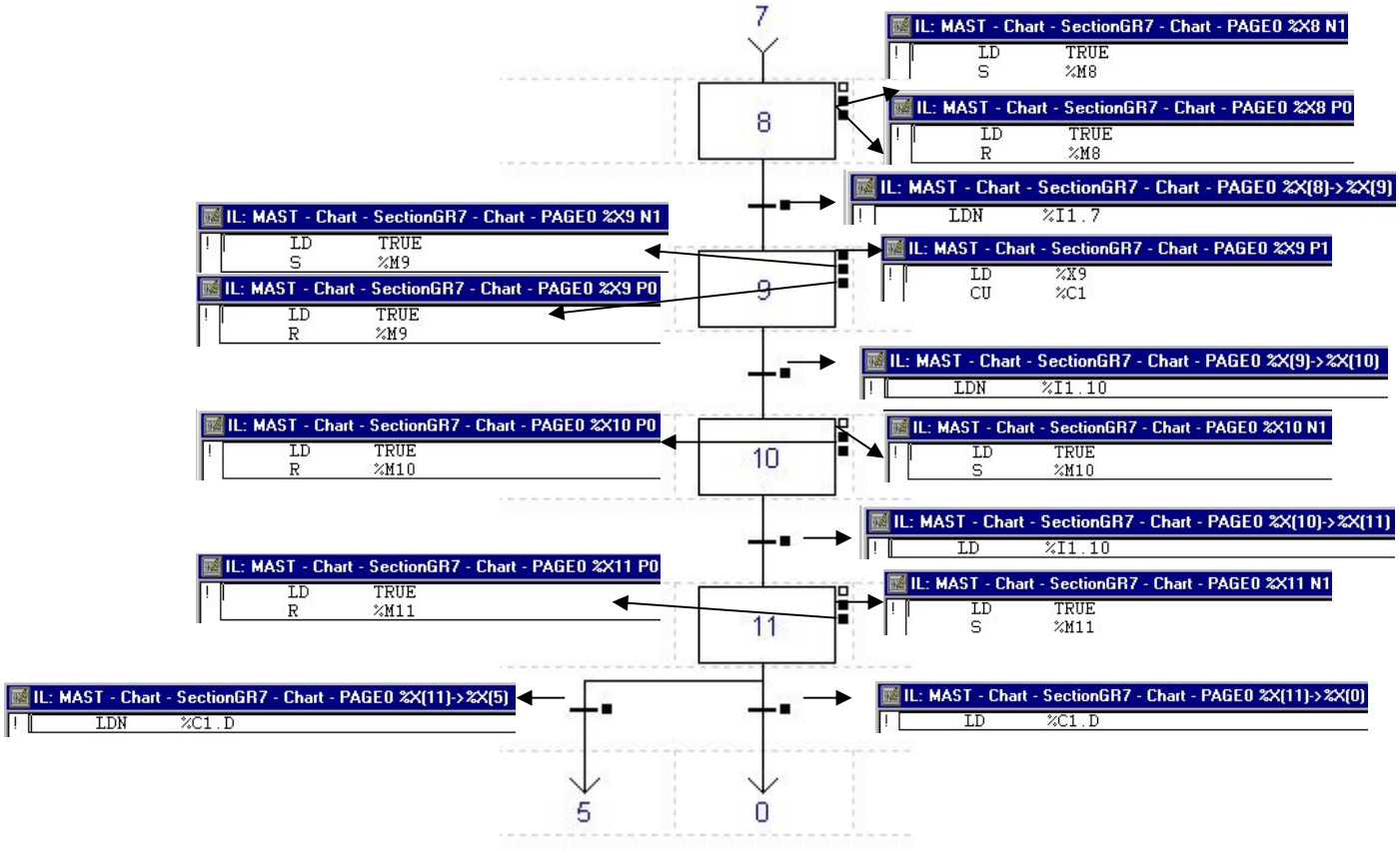
Como se puede observar en la figura, se parte de una etapa inicial y se van conectando etapas con transiciones utilizando el editor gráfico. Una vez terminadas las conexiones del Grafcet hay que validar la aplicación, tal y como se hacía en la programación con lista de instrucciones. Se puede acceder a la programación de las etapas y transiciones pulsando el botón derecho del ratón sobre ellas. Como se observa, en las transiciones aparece sólo un cuadro junto a su símbolo. Si el cuadro está hueco, indica que no se han programado las condiciones que permitirán franquear la transición, y en ese caso la evolución del autómata será incapaz de franquear dicha transición. Es por tanto necesario que todas las transiciones lleven aparejado un código que permita franquearlas. El código asociado al flanqueo de transiciones y etapas se puede escribir en lenguajes `LD`, `IL` y `ST`. Típicamente, el código asociado a las transiciones consta de operaciones de carga y comparación lógica (`LD`, `AND`, etc.).

El código asociado a transiciones se puede programar en tres segmentos de programa, según el tipo de acción que se quiera llevar a cabo: acción al activar la etapa, acción continua durante la duración de la etapa y acción al desactivar. Las acciones al activar la etapa están típicamente asociadas al incremento de contadores o acciones relacionadas con flancos ascendentes. Las acciones continuas son aquellas mediante las cuales se consigue un efecto que perdura mientras está la etapa activada (por ejemplo, activación de marcas asociadas a estados). Las acciones relacionadas con la desactivación de la etapa permiten por ejemplo poner a cero la marca asociada a la misma (como veremos esto se puede realizar de una forma más sencilla). Los tres cuadros que aparecen junto a las etapas en el Grafcet se corresponden con cada uno de estos módulos de programa. Si están huecos indican que no se han programado condiciones asociadas a ese módulo y si están rellenos indica la programación del mismo.

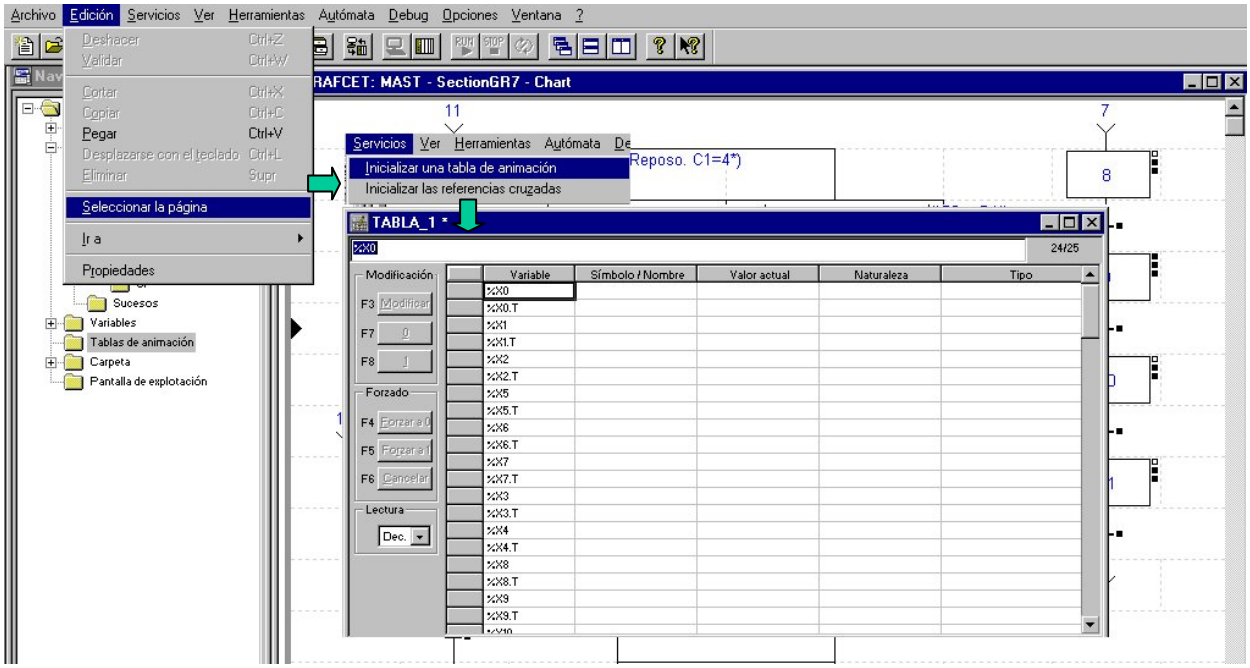


Las secciones `Prl` y `Post` en el ejemplo son iguales que las secciones Inicialización y Salida en el ejemplo programado con lista de instrucciones, por lo que se va a centrar la explicación en la etapa Grafcet. En el ejemplo se han programado las condiciones referentes a etapas y transiciones en lenguaje de lista de instrucciones. Al igual que en el ejemplo anterior, cada vez que se complete la introducción de una sentencia hay que validarla antes de cerrar la ventana de edición para poder activar otra. Con la finalidad de realizar una aplicación lo más variada posible, se van a utilizar en este caso temporizadores tipo TP y se van a comentar algunos fallos asociados a este tipo de temporizadores que trae la aplicación y una forma sencilla de soslayarlos. Los temporizadores tipo `TP` mantienen una señal activa un cierto tiempo predeterminado cuando detectan un flanco de entrada en su señal de activación (IN). Se supone, que una vez que el valor interno del temporizador (`%TMi.V`) alcanza el valor preseleccionado (`%TMi.P`), acaba la cuenta, la salida del temporizador (`%TMi.Q`) vuelve a cero y el valor interno (`%TMi.V`) también vuelve a cero. La experiencia ha demostrado que cuando se programa este temporizador dentro de una sección Grafcet, el valor interno `%TMi.V` no vuelve automáticamente a cero, por lo que una posible solución para forzar el reset del temporizador es pasar temporalmente el valor de preselección `%TMi.P` a cero y volver posteriormente a ponerlo a su valor por defecto. En el ejemplo que se muestra a continuación se ha programado esta opción. Hay que hacer notar que para que el contador actúe de forma adecuada, las señales externas que recibe el autómata deben tener valores “lógicos” acordes con la aplicación (fundamentalmente `%I1.10` y `%I1.7`). Como las pruebas se llevan a cabo de forma emulada utilizando los interruptores que se han incorporado al autómata, es posible que algunas de estas señales estén desactivadas en la evolución del autómata (cuando en el caso real estarían activadas). Esto puede provocar que se atraviesen los estados instantáneamente al franquearse las transiciones asociadas a estas entradas, pudiendo provocar un funcionamiento no esperado del contador.



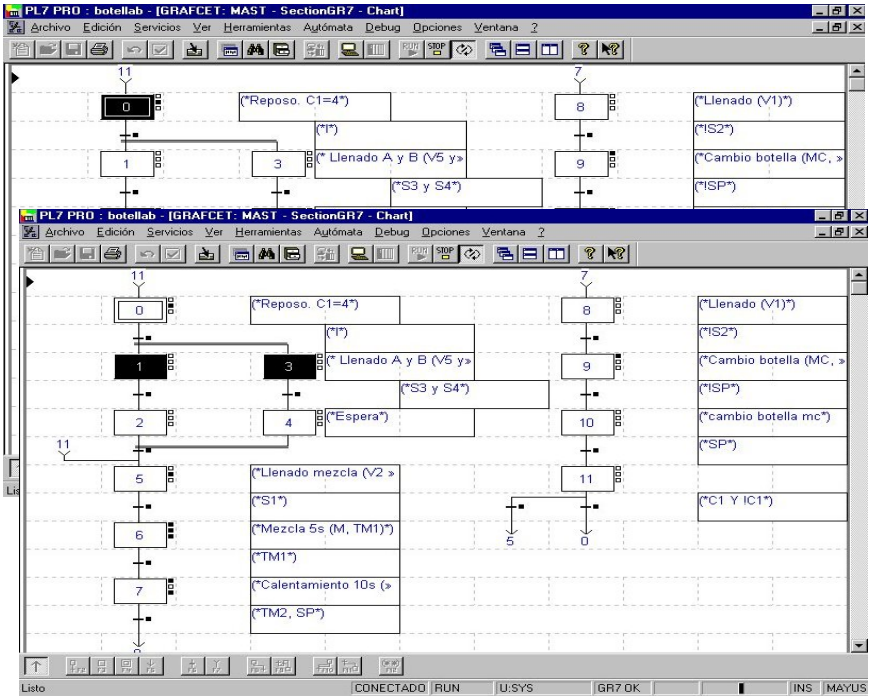


- *Paso 5: Creación de la tabla de animación del Grafcet y transferencia del programa al autómata*: antes de transferir el programa y pasar el autómata a modo RUN conviene crear una tabla de animación del Grafcet. Cada etapa del Grafcet tiene asociado un bit `%Xi` que se pone a 1 cuando se activa la etapa y a 0 cuando se desactiva (aparece también el tiempo asociado a la activación de esa etapa `%Xi.T`). Al crear una tabla de animación global, se podrá ver cómo se oscurecen los cuadros representativos de las etapas en el Grafcet a medida que se van franqueando las distintas transiciones. La forma común de crear esta tabla de animación del Grafcet se explica a continuación: se sitúa el puntero del ratón en la página del Grafcet. En el menú **`Edición`**, se elige la opción **`Seleccionar la página`**. Una vez hecho esto, en el menú **`Servicios`**, se selecciona **`Inicializar una tabla de animación`**, apareciendo la ventana que se muestra a continuación.

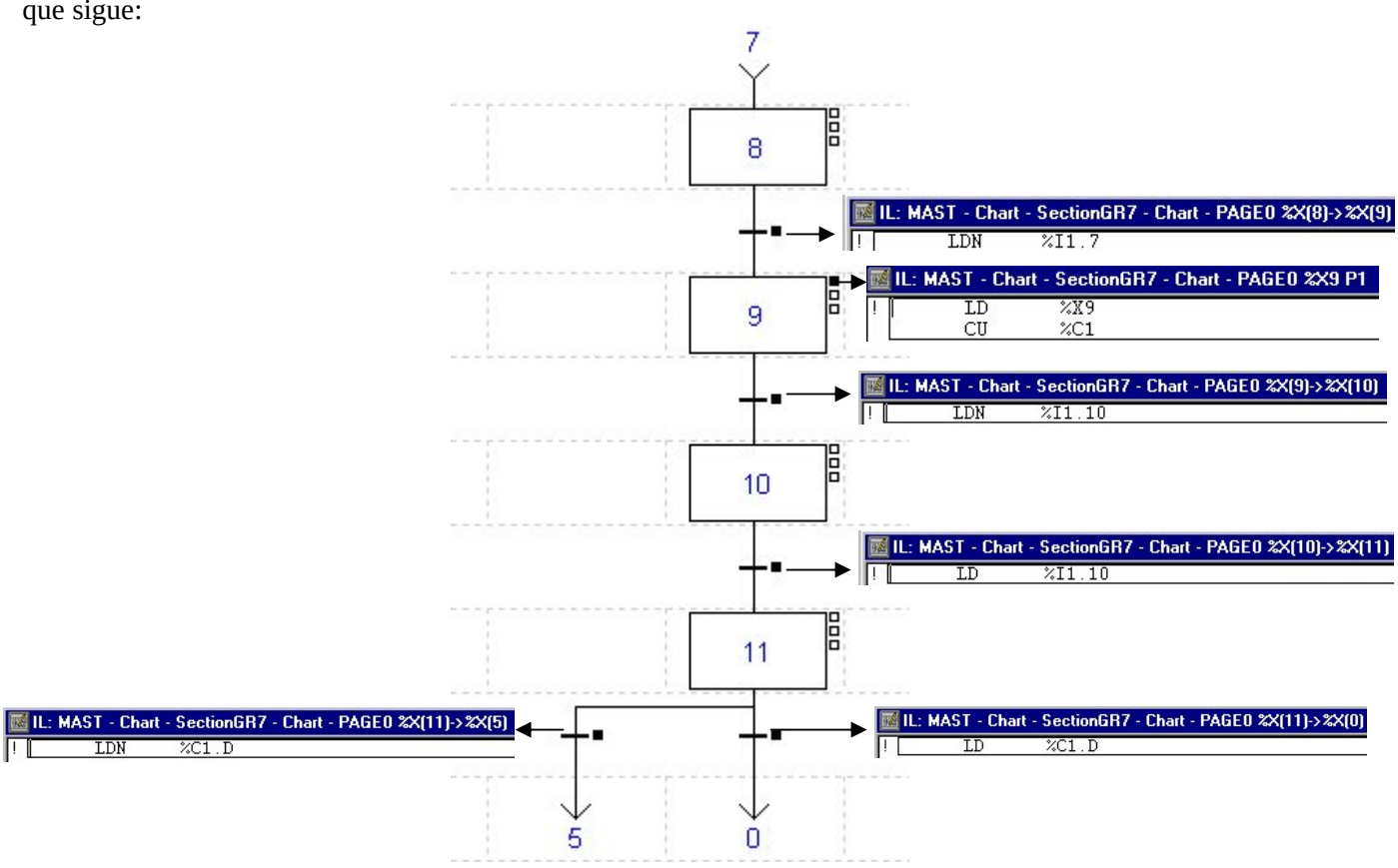


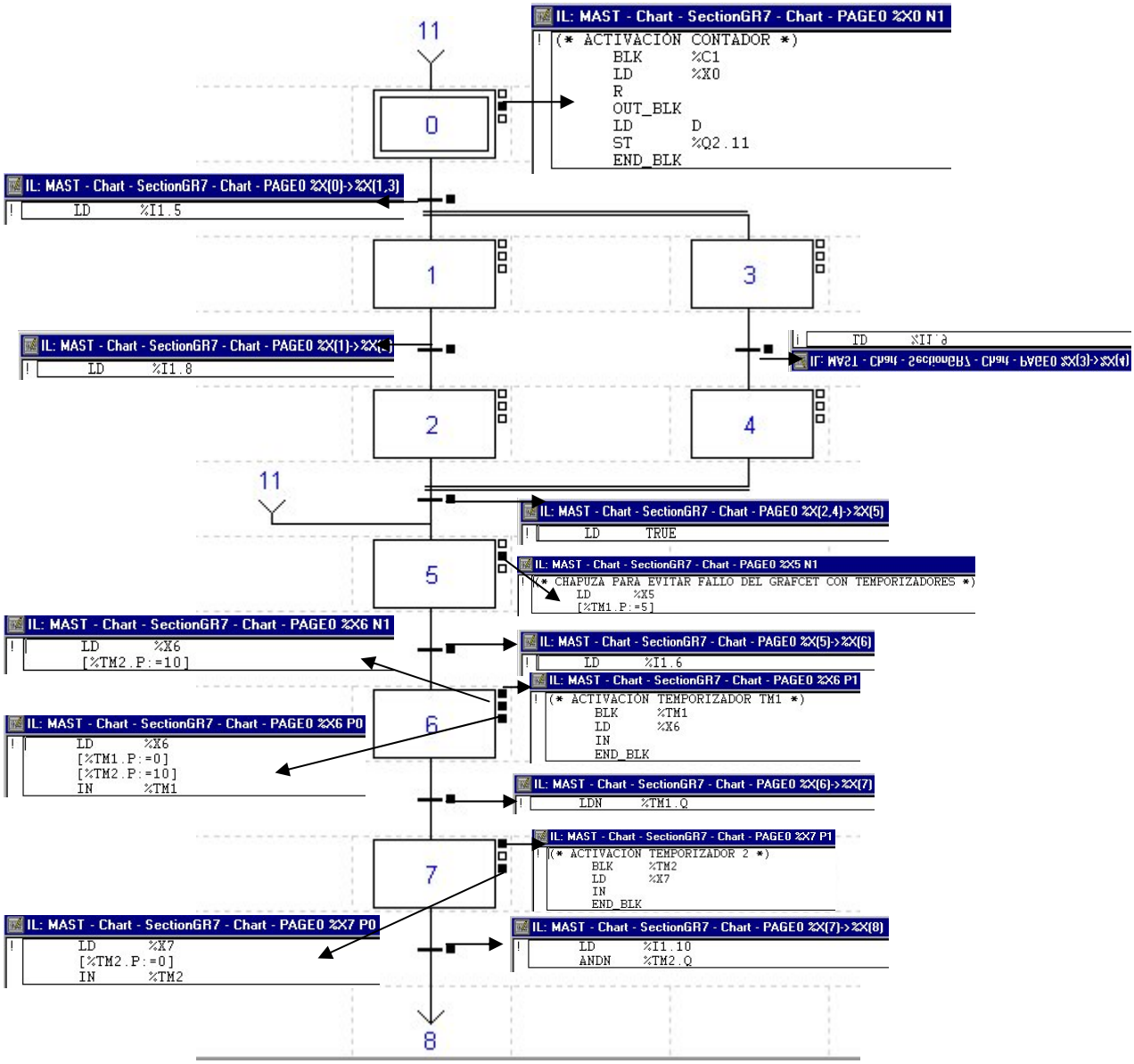
Además, se pueden crear una vez se esté ejecutando el programa distintas tablas de animación para analizar la evolución de valores de temporizadores, contadores, variables de memoria, etc. Una vez creada la tabla de animación global se puede transferir el programa al autómata, conectarlo y pasarlo a modo `RUN` como se explicó en el ejemplo de programación con lista de instrucciones.

- *Paso 6: ejecución del programa*: una vez se pasa el autómata a modo RUN, se puede ver la evolución del mismo sobre el propio Grafcet (animación de las etapas asociadas y franqueo de transiciones.



El ejemplo analizado se puede desarrollar de una forma mucho más sencilla aprovechando los bits de activación de las etapas del Grafcet (`%Xi`), de modo que actúan como las marcas utilizadas en el ejemplo anterior. Se puede observar como la programación del Grafcet en el caso anterior puede quedar reducida a lo que sigue:




IL: MAST - Chart - SectionGR7 - Chart - PAGE0 %X0 N1
(* ACTIVACIÓN CONTADOR *)
BLK %C1
LD %X0
R
OUT_BLK
LD D
ST %Q2.11
END_BLK
11
0
IL: MAST - Chart - SectionGR7 - Chart - PAGE0 %X(0)->%X(1,3)
LD %I1.5
1
3
IL: MAST - Chart - SectionGR7 - Chart - PAGE0 %X(1)->%X(2)
LD %I1.8
2
4
11
IL: MAST - Chart - SectionGR7 - Chart - PAGE0 %X(2,4)->%X(5)
LD TRUE
5
IL: MAST - Chart - SectionGR7 - Chart - PAGE0 %X5 N1
(* CHAPUZA PARA EVITAR FALLO DEL GRAFCET CON TEMPORIZADORES *)
LD %X5
[%TM1.P:=5]
IL: MAST - Chart - SectionGR7 - Chart - PAGE0 %X6 N1
LD %X6
[%TM2.P:=10]
IL: MAST - Chart - SectionGR7 - Chart - PAGE0 %X(5)->%X(6)
LD %I1.6
IL: MAST - Chart - SectionGR7 - Chart - PAGE0 %X6 P1
(* ACTIVACIÓN TEMPORIZADOR TM1 *)
BLK %TM1
LD %X6
IN
END_BLK
6
IL: MAST - Chart - SectionGR7 - Chart - PAGE0 %X6 P0
LD %X6
[%TM1.P:=0]
[%TM2.P:=10]
IN %TM1
IL: MAST - Chart - SectionGR7 - Chart - PAGE0 %X(6)->%X(7)
LDN %TM1.Q
IL: MAST - Chart - SectionGR7 - Chart - PAGE0 %X7 P1
(* ACTIVACIÓN TEMPORIZADOR 2 *)
BLK %TM2
LD %X7
IN
END_BLK
7
IL: MAST - Chart - SectionGR7 - Chart - PAGE0 %X7 P0
LD %X7
[%TM2.P:=0]
IN %TM2
IL: MAST - Chart - SectionGR7 - Chart - PAGE0 %X(7)->%X(8)
LD %I1.10
ANDN %TM2.Q
8