INSTITUTO TECNOLOGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS MONTERREY

PROGRAMA DE GRADUADOS EN INGENIERIA

"METODOLOGIAS PARA LA OBTENCION DEL MODELO Y EL PROGRAMA DE CONTROL PARA UN PROCESO DE EVENTOS DISCRETOS BASADO EN GRAFCET UTILIZANDO UN PLC"

T E S I S

PRESENTADA COMO REQUISITO PARCIAL

PARA OBTENER EL GRADO ACADEMICO DE:

MAESTRO EN CIENCIAS

ESPECIALIDAD EN AUTOMATIZACION

(INGENIERIA DE CONTROL)

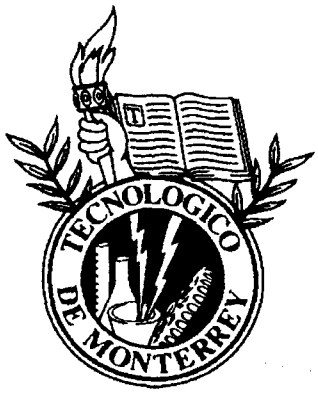
AGUSTIN PANDO DELGADO

MAYO 2002

INSTITUTO TECNOLÓGICO Y DE ESTUDIOS SUPERIORES DE MONTERREY

CAMPUS MONTERREY

PROGRAMA DE GRADUADOS EN INGENIERÍA



"METODOLOGÍAS PARA LA OBTENCIÓN DEL MODELO Y EL PROGRAMA DE CONTROL PARA UN PROCESO DE EVENTOS DISCRETOS BASADO EN GRAFCET UTILIZANDO UN PLC"

TESIS

PRESENTADA COMO REQUISITO PARCIAL
PARA OBTENER EL GRADO ACADÉMICO DE

MAESTRO EN CIENCIAS
ESPECIALIDAD EN AUTOMATIZACIÓN
(INGENIERÍA DE CONTROL)

AGUSTÍN PANDO DELGADO
MAYO 2002

Agradecimientos

A DIOS por darme la oportunidad de vivir y llegar hasta aquí y por iluminarme en los momentos difíciles de mi vida.

A mi esposa Margarita por su gran amor, cariño y comprensión. Porque en todo momento me apoyó para cumplir mis metas, y porque una sola de sus palabras me alentaba para seguir adelante y lograr lo que me propusiera. Gracias mi amor.

A mis padres Agustín y Rosa María, por el inmenso amor que me tienen y todo el apoyo incondicional que me han dado, porque gracias a ustedes he podido lograr una meta más en mi vida. Gracias mamá por tus oraciones y gracias papá por tus consejos.

A mi hermana Erika y mi sobrina Andrea, porque con su cariño y afecto han sabido plantar en mi una luz de esperanza y amor. Gracias Erikiux por tu apoyo incondicional y tus palabras y gracias Andy por tus preciosos detalles y cartitas que guardó en mi corazón.

A mi abuelito Lalo porque me ha enseñado a luchar para seguir adelante y a ser un buen hombre, gracias abue por sus invaluables consejos.

A mi familia: Héctor, Emilia, Larissa, Hectorcito, Luis Fernando, Carmelita, Karla, Paty, Javier, Javiercito, Vania, Arturo, Chuy, Eduardo, a mis padrinos Baltasar y Luz Elena, Gaspar, Baltita, a mi tío Mario y a mi tío Chacho por todo su amor y apoyo.

A mis suegros Lindoro y Norma, porque representan para mí una parte muy importante de mi familia, y porque como padres me han sabido dar amor, apoyo y comprensión. Gracias doctor por sus palabras y gracias señora por su cariño.

Al Dr. Luis Lauro Cantú Salinas por sus consejos y apoyo incondicional.

A mis amigos de la maestría: Luis, Francisco (Calleja), Teodoro, Francisco (Montes), Fernando (Fer) y Abraham, porque con ustedes aprendí tantas cosas y por lo buenos momentos que compartimos. Gracias por su apoyo.

A mis amigos: Horacio (Lacho), Jorge (Montes), Rolando (Rolas), Ramiro (Ramirín), Enrique (mi compadre Quique) y Eric (mi compadre Eric), porque han sabido apoyarme y ofrecerme su cariño en todo momento.

Agradecimientos

A mi asesor el Dr. Antonio Favela Contreras por su apoyo y dedicación para la realización de este trabajo.

Al Ing. M.C. Francisco Palomera Palacios por su ayuda incondicional y su apoyo desde un inicio para lograr la meta de escribir este trabajo de tesis. Gracias Inge.

Al Dr. Jorge Limón Robles, porque con sus consejos, palabras de aliento y su ayuda incondicional, pude aprender y lograr muchas cosas en mi vida.

Al claustro de Mecatrónica: Dr. Carlos Narváez, Dr. José de Jesús Rodríguez, Dr. Ricardo Ramírez, Ing. M.C. Elvira del Rosario Niño, Ing. M.C. Irma Yolanda Sánchez, por su apoyo.

Al Dr. Hassane Alla, Tibi, Rosa y Monika por su apoyo durante mi estancia en la Ciudad de Grenoble, Francia.

A Amparo Herrera, Pedro Sánchez y Juan Pineda por su ayuda incondicional.

ÍNDICE

Lista de Figuras.

Lista de Tablas.

Introducción.

Algunos de los lenguajes de programación de Controladores Lógicos Programables (PLCs) en aplicaciones industriales para la elaboración de programas de control de eventos discretos son: la lista de instrucciones, los bloques funcionales, los diagramas escalera (LLD), y GRAFCET/SFC. El diagrama de escalera ha sido el lenguaje tradicional en la elaboración de los diagramas de control, debido a su origen histórico como reemplazo al diagrama eléctrico de escalera; además, porque cualquier PLC incluye, por lo menos, al diagrama de escalera como su lenguaje básico de programación.

Pero, GRAFCET/SFC es una de las herramientas poderosa incluidas en algunos PLCs para la programación de secuencias de mediana y alta complejidad, y que además, día a día va ganando más adeptos.

El método más común para la elaboración del diagrama de escalera (programa de control) se deja más al sentido común (*feeling*) del programador. El programa obtenido no necesariamente tendrá las cualidades de ser óptimo, claro, simple, y confiable. Además, los diagramas de escalera elaborados no facilitan al usuario final el análisis y detección de fallas dada su complejidad, y resultan de poca utilidad para la capacitación del personal para entender el funcionamiento del proceso.

En el espacio de búsqueda bibliográfica de esta tesis, se encontró sólo la metodología de [14] para estructurar un diagrama escalera sobre la base de un estudio del proceso que se desea controlar, pero ésta no considera condiciones de seguridad, interacción con el proceso a ser controlado como lo son paros de emergencia, interrupciones (pausas) entre otras. Esto presenta algunas limitaciones para ser utilizada por un usuario, pero al menos constituye un

esfuerzo formal por contribuir a la estructuración de un programa de control lógico en sistemas de eventos discretos.

En este trabajo de tesis se presentan dos metodologías:

- La primera **IPTG en GRAFCET**, permite al usuario modelar su proceso y obtener la documentación del modelo y el programa de control en cinco etapas.
- La segunda **RIMAnI en GRAFCET** permite al usuario modelar el proceso por medio de la recopilación de la información del experto y obtener el programa de control en cuatro etapas.

Ambas metodologías se apoyan fuertemente en el uso de Grafcet, considerada como herramienta gráfica para la modelación de eventos discretos, y de GRAFCET/SFC como lenguaje de programación de eventos lógicos discretos.

Las metodologías propuestas pueden ser utilizadas por ingenieros de proceso con diferente grado de dominio en la elaboración de programas de control utilizando PLCs. Esto es, para quienes es la primera vez que elaborarán un programa de control para sus procesos; también para quienes ya tienen un diagrama escalera pero sólo quieren obtener la documentación del modelo del comportamiento de su proceso y conservar su diagrama de control en otro lenguaje de programación diferente a GRAFCET/SFC. Y finalmente, para quienes ya tienen su programa de control en diagrama escalera u otro lenguaje de programación de PLCs pero desean elaborar y estructurar su documentación del modelo del comportamiento del proceso y el programa de control en GRAFCET/SFC. Por supuesto, no se deja a un lado a los usuarios potenciales que se están formando en las aulas universitarias o en centros de capacitación, y serán los mejores promotores del uso de las metodologías propuestas..

Existen dos tipos de procesos industriales: los continuos y los de eventos discretos. Además, pueden presentarse una combinación de éstos, como lo son los procesos denominados "*Batch*" o por lotes. El presente trabajo enfoca su aplicación sólo a los procesos de eventos discretos. Ejemplos de ellos son los sistemas de ensamble de manufactura, en donde la evolución del proceso ocurre de manera discreta y asíncrona.

En el contexto de esta tesis se denominará como *proceso instrumentado* a un proceso industrial de eventos discretos automatizado. Esto es, uno que incorpora en su operación automática a sensores y actuadores binarios (on-off) que se encuentran conectados a un PLC para ejecutar el control de un comportamiento deseado. Además, se podrá contar o no con la documentación de las características de su instrumentación, de los diagramas correspondientes de escalera, de tuberías e instrumentación (DTI o P&ID).

Motivación.

Ante la poca información y metodologías disponibles para la modelación y control de sistemas de eventos discretos [22], se proponen dos metodologías para contribuir a subsanar la ausencia de dicha información disponible para ingenieros de procesos industriales, y principalmente los relacionados con sistemas de eventos discretos. Dichas metodologías facilitarán no sólo la modelación del comportamiento de sus procesos de eventos discretos, sino también la estructuración de la elaboración de su diagrama de control, sin ser restrictivo al uso de algún lenguaje de programación de PLCs. Además, las metodologías mantienen una formalidad para su aplicación.

Las metodologías incluyen una componente académica y una parte de experiencia industrial del autor.

Objetivo.

Proponer y desarrollar dos metodologías con las que se obtenga la estructura de implementación GRAFCET para cualquier proceso de eventos discretos que haya sido previamente instrumentado para trabajar automáticamente, basándose en la experiencia del experto del proceso o de un modelo autómata discreto previamente establecido.

Alcance.

El presente trabajo de tesis sólo incluirá aquellos procesos previamente instrumentados para trabajar automáticamente, con entradas y salidas relacionadas a la ocurrencia de eventos discretos, donde además, pueden existir condiciones en las que se genere un evento discreto a partir de una variable continua.

Objetivo.

Proponer y desarrollar dos metodologías con las que se obtenga la estructura de implementación GRAFCET para cualquier proceso de eventos discretos que haya sido previamente instrumentado para trabajar automáticamente, basándose en la experiencia del experto del proceso o de un modelo autómata discreto previamente establecido.

Alcance.

El presente trabajo de tesis sólo incluirá aquellos procesos previamente instrumentados para trabajar automáticamente, con entradas y salidas relacionadas a la ocurrencia de eventos discretos, donde además, pueden existir condiciones en las que se genere un evento discreto a partir de una variable continua.

Resumen.

En el capítulo 1 se presentan las herramientas formales de modelación e implementación de sistemas de eventos discretos, algunas de éstas son utilizadas en capítulos posteriores en el desarrollo de las metodologías. Estas herramientas son:

- Redes de Petri.
- Autómatas.
- GRAFCET.

La primera metodología propuesta es una modificación al trabajo de [14]. Ésta se presenta en el capítulo 2 y se denomina IPTG en GRAFCET, la cual parte de un diagrama IDEF0 para formar un modelo bajo la forma de Red de Petri simplificada y finalmente obtener el lenguaje de programación GRAFCET.

La segunda metodología es la denominada RIMAnI en GRAFCET descrita en el capítulo 3. Ésta consiste en recolectar la información del proceso de una manera ordenada, para después obtener su correspondiente modelo autómata discreto de éste. Para evitar que este modelo tenga secuencias prohibidas, se realiza un análisis del modelo basado en un autómata, y si éste es favorable se obtiene la estructura de implementación GRAFCET.

Con la finalidad de mostrar las bondades de las dos metodologías expuestas(capítulos 2 y 3), se utiliza un proceso de estampado de latas industrial ya instrumentado. Los resultados teóricos obtenidos se muestran en el Anexo 2 (metodología IPTG en GRAFCET) y en el Anexo 4 (metodología RIMAnI en GRAFCET).

En el Anexo 1 y 2 se muestran las tablas en blanco para las metodologías IPTG y RIMAnI en GRAFCET respectivamente.

Para validar los resultados obtenidos para el proceso de estampado de latas con las metodologías IPTG y RIMAnI en GRAFCET se realizaron pruebas experimentales en el PLC TSX-3705 de Modicon Telemecanique en el Laboratorio de Control Lógico del Departamento de Mecatrónica y Automatización. El desarrollo así como los resultados obtenidos se muestran en el capítulo 4.

Finalmente en el capítulo 5 se establecen las conclusiones y perspectivas de este trabajo de tesis.

Capítulo 1

Herramientas formales para la modelación e implementación de sistemas de eventos discretos.

En este capítulo se realizará un estudio de las herramientas más importantes para el análisis e implementación de sistemas de eventos discretos, como lo son las Redes de Petri, la teoría de Autómatas y el lenguaje de programación GRAFCET.

1.1 Conceptos Básicos de Sistemas de Eventos Discretos.

Antes de iniciar con el análisis de sistemas de eventos discretos, es necesario conocer algunos conceptos que serán utilizados en los siguientes capítulos y que representan parte importante de este trabajo de investigación.

1.1.1 Sistema.

Existen muchas definiciones de lo que es un sistema, pero todas conllevan a una sola, así es que es posible definir sistema como:

" *Un sistema consiste de componentes que interactúan entre si con una función en común a ser realizada [3]*"

Los sistemas están constituidos de bloques funcionales, entradas y salidas. Dependiendo de las entradas y su condición inicial, el sistema tendrá un *estado*, el cual se definirá a continuación.

1.1.2 El estado.

En la teoría de sistemas el término *estado* tiene un significado muy importante en el proceso de modelación, y en muchas técnicas de análisis. Así, se define "*estado*" de la siguiente manera:

"*El estado de un sistema a un tiempo* t_0 *representa la información necesaria en* t_0 *de tal manera que la salida del sistema a cualquier tiempo* $t \geq t_0$ *es determinada de manera única por esta información y por las entradas a todo tiempo* $t \geq t_0$ "

1.1.3 Sistemas lineales.

Se dice que una función g es lineal si y solo si:

$$g(a_1u_1 + a_2u_2) = a_1g(u_1) + a_2g(u_2) \qquad (1)$$

para cualquier entrada u_1, $u_2 \in U$, y cualquier número real a_1, a_2. En un sistema multivariable la Ecuación (1) es una ecuación vectorial-matricial donde u_1 y u_2 son vectores de entradas, a_1 y a_2 matrices reales constantes y g la relación dinámica del sistema.

Basándose en la propiedad representada por la Ecuación (1) podemos concluir, que un sistema lineal puede ser analizado por separado, observando el efecto producido a la salida debido a cada entrada, para después realizar la suma de todos los efectos. Esto es posible debido al principio de superposición de los sistemas lineales.

1.1.4 Sistemas de estados discretos.

En los modelos de estados discretos, el espacio de estados es un conjunto discreto. En este caso, las variables de estado sólo pueden brincar en puntos discretos en el tiempo desde un valor de estado discreto a otro. Por ejemplo, una válvula solenoide sólo tiene dos estados discretos, abierta o cerrada. Así, en general, los sistemas de eventos discretos son aquellos donde sus elementos funcionales, entradas y salidas son discretas, es decir sólo pueden tomar valores como 0, 1, 2, 3,... [3].

1.1.5 Sistemas de eventos discretos (SED)

Cuando el espacio de estados de un sistema puede ser descrito como un conjunto discreto, y las transiciones entre los estados son sólo observadas como puntos discretos en el tiempo, es posible asociar estas transiciones con *eventos* y entonces el sistema se define con un *sistema de eventos discretos* [3].

1.1.5.1 Evento.

Un evento es aquel que tiene una ocurrencia instantánea y que provoca las transiciones discretas de un valor de estado a otro en un sistema determinado. Un evento puede ser identificado con una acción en específico, por ejemplo: la ocurrencia de una alarma de alto nivel en un tanque.

En lo posterior se utilizará la letra *e* para simbolizar un evento, y un conjunto de eventos será denotado por la letra E [3].

1.1.5.2 Características de un sistema de eventos discretos.

Los sistemas de eventos discretos satisfacen las siguientes dos propiedades:

- El espacio de estado es un conjunto discreto.
- El mecanismo de transición de estado está basado en la ocurrencia espontánea de eventos (*event-driven*)

1.2 Redes de Petri [7]

1.2.1 Introducción a las Redes de Petri (RdP)

Las RdP son un modelo matemático de propósito general que es usado para describir las relaciones existentes entre condiciones y eventos. Dos de las características más interesantes de las RdP son:

- Permiten modelar y visualizar tipos de comportamiento que tienen paralelismo, concurrencia, sincronización y recursos compartidos.
- Los resultados teóricos son abundantes.

1.2.2 Conceptos Básicos de las Redes de Petri.

En esta sección se analizarán algunos conceptos importantes al tratamiento de las RdP con la finalidad de entender su funcionamiento y aplicación en la modelación de sistemas de eventos discretos.

1.2.2.1 Plazas, transiciones, arcos y tokens.

Una RdP tiene dos tipos de nodos llamados plazas y transiciones. Una plaza es representada por un círculo y una transición por una barra. Las plazas y transiciones son conectadas por arcos. Cabe señalar que el número de plazas y transiciones es finito y diferente de cero.

Un arco es directo y conecta tanto a una plaza con una transición como a una transición con una plaza. El conjunto de plazas en una RdP es llamado P y es igual a $P = \{P_1, P_2, P_3,..., P_n\}$ donde n es un número entero. El conjunto de transiciones en una RdP es llamado T y es igual a $T = \{T_1, T_2, T_3,..., T_n\}$

Una plaza que está antes de una transición T_x se define como una entrada a T_x. Si la plaza está a la salida de T_x se define a ésta como una salida de T_x. También, si una transición no tiene entradas entonces será una transición

fuente (source), por otro lado si ésta no tiene salidas entonces se define como una transición terminal (sink)

Un token se define como un recurso en una plaza. Por ejemplo, si una plaza representa una cola de botellas, y si físicamente existen dos lugares vacíos en la cola, entonces la plaza tendrá dos tokens, representando cada token el lugar disponible en la cola. Así, cuando un lugar de la cola se ocupe, la plaza disminuirá su cantidad en un token, y de igual manera si se desocupa un lugar, el número de tokens en la plaza se incrementará en uno.

En la Figura 1.1 se muestran los elementos definidos en los párrafos anteriores.

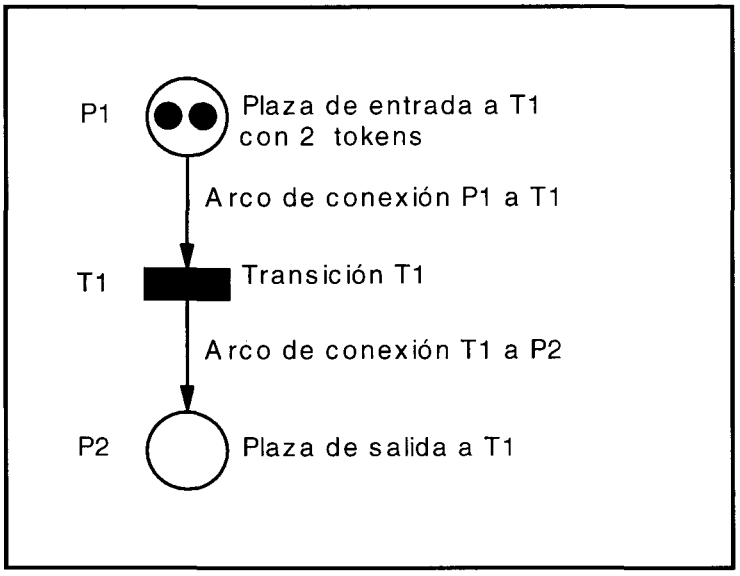


Figura 1.1. Ejemplo de las definiciones de plazas, transiciones, arcos y tokens.

1.2.2.2 Marcaje.

El vector formado por los tokens por plaza, por ejemplo M_k = [1 0 0 1,..., 2], representa en cierto momento k el estado de la RdP conocido como marcaje. Así en la RdP de la Figura 1.1 se tiene el marcaje inicial M_i = [2 0]

1.2.2.3 Disparo de una transición.

A la ejecución de una transición se le conoce como *disparo de una transición. Una transición sólo puede ser disparada si todas sus plazas de entrada contienen al menos un token.* Se dice que una transición es habilitada en el momento que todas sus plazas de entrada tienen al menos un token. Una transición es disparada si y solo si se encuentra habilitada y su condición se cumple lo que tendrá como consecuencia que las plazas de salida sumen un token. Es importante observar que el tiempo de duración de una transición es igual a cero.

En la Figura 1.2 se muestra una RdP con un token en P1 y un instante después cuando se dispara la transición.

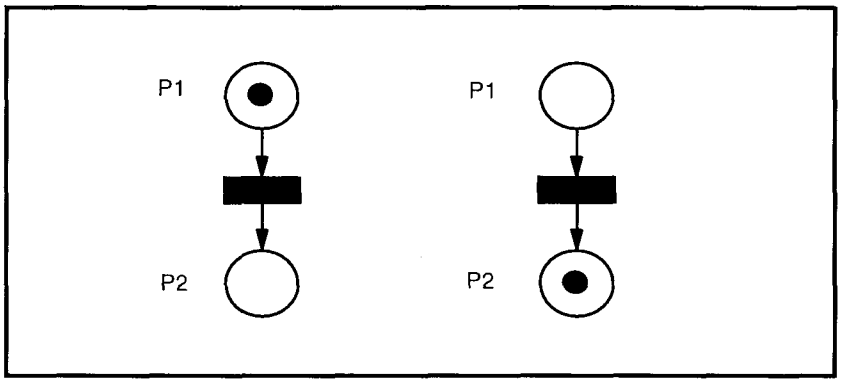


Figura 1.2. Ejemplo del disparo de una transición en una RdP.

1.2.2.4 Red de Petri Autónoma.

Una RdP autónoma describe el comportamiento de un sistema de una manera autónoma, es decir los instantes de disparo son desconocidos o bien no están indicados. La Figura 1.3 muestra una RdP autónoma donde se observa que los instantes de disparo no están definidos por ningún agente externo, sino por la misma evolución de la RdP.

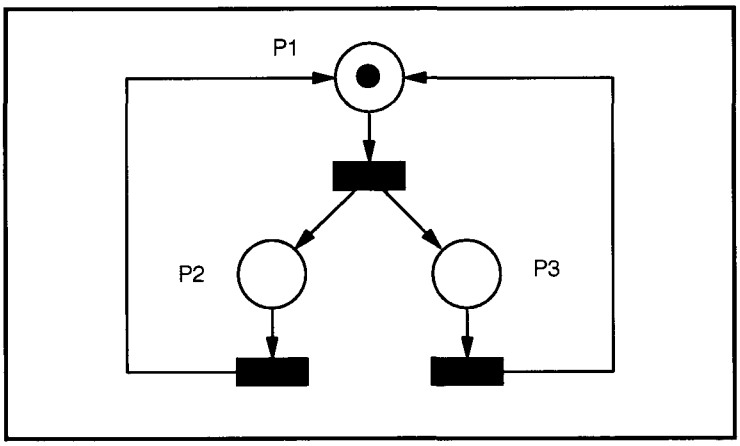


Figura 1.3. RdP autónoma

1.2.2.5 Red de Petri No-Autónoma.

Se dice que una RdP es no-autónoma cuando el disparo de sus transiciones es definido por factores externos. La Figura 1.4 muestra una RdP no autónoma, donde se observa que las transiciones dependen del valor del reloj externo X, el cual provocará que las transiciones se disparen. Cada una de las transiciones tiene asignado un valor entero, el cual indica que éstas se dispararán cuando el valor del reloj X sea igual al valor asignado a cada transición.

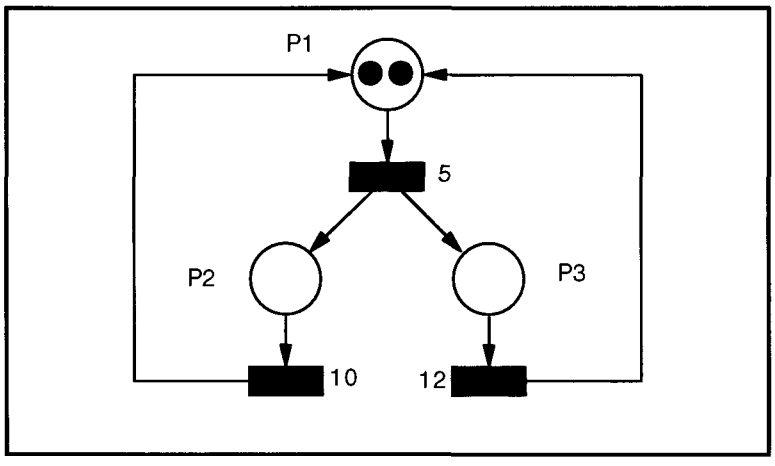


Figura 1.4. RdP no autónoma.

1.2.3 Álgebra Lineal usada en las Redes de Petri.

En esta sección se definirán los elementos fundamentales en el álgebra lineal utilizada en las RdP. Las herramientas desarrolladas serán de utilidad para el análisis de las RdP.

1.2.3.1 Notaciones y definiciones.

Una **RdP ordinaria sin marcaje** se define como un cuádruple:

$$Q = (P, T, Pre, Post)$$

donde:

$P = \{P_1, P_2, P_3, ..., P_n\}$ es un conjunto de plazas finito y no vacío.

$T = \{T_1, T_2, T_3, ..., T_m\}$ es un conjunto de transiciones finito y no vacío.

$P \cap T = \emptyset$, es decir, los conjuntos de P y T son disjuntos.

Pre: $P \times T \rightarrow \{0,1\}$ es la aplicación de la entrada de incidencia.

Post: $P \times T \rightarrow \{0,1\}$ es la aplicación de la salida de incidencia.

Pre (P_i,T_j) significa el peso del arco de $P_i \rightarrow T_j$. Este peso es **1** si el arco existe y **0** sí no. *Post* (P_i,T_j) significa el peso del arco de $T_j \rightarrow P_i$. Por ejemplo para la RdP de la Figura 1.5 , los valores son: Pre $(P_1, T_1) = 1$, Post $(P_2, T_1) = 1$ y Pre $(P_2, T_1) = 0$.

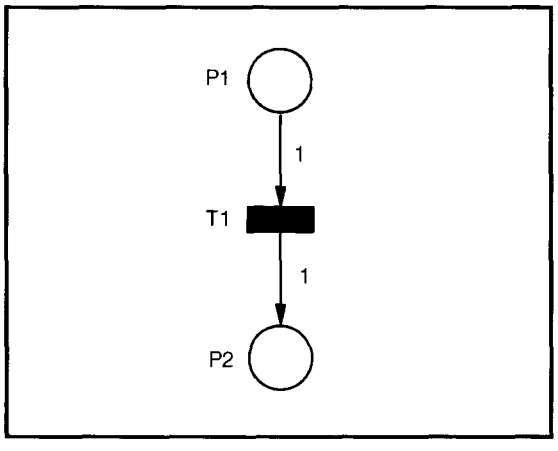


Figura 1.5. Ejemplo de una RdP ordinaria sin marcaje.

Una RdP generalizada sin marcaje es definida igual que una RdP ordinaria sin marcaje excepto en que,

$$Pre: P \times T \rightarrow N$$
$$Post: P \times T \rightarrow N$$

donde N representa el conjunto de números enteros.

En este tipo de redes se utiliza la siguiente notación:

$^\circ T_j = \{P_i \chi P \xi Pre(P_i,T_j) \nu 0\}$ = conjunto de plazas de entrada de T_j.

$T_j^\circ = \{P_i \chi P \xi Post(P_i,T_j) \nu 0\}$ = conjunto de plazas de salida de T_j.

$^\circ P_i = \{T_j \chi T \xi Pre(P_i,T_j) \nu 0\}$ = conjunto de transiciones de entrada de P_i.

$P_i^\circ = \{T_j \chi T \xi Post(P_i,T_j) \nu 0\}$ = conjunto de transiciones de salida de P_i.

Una RdP con marcaje es un par $R = \{Q, M_o\}$, donde $Q = \{P, T, Pre, Post\}$ es una RdP sin marcaje y M_o un marcaje inicial. Las condiciones de validación para este tipo de RdP son definidas de la siguiente manera:

"La transición T_j es habilitada por un marcaje M sí y sólo sí $M(P_i) \mu Pre(P_i,T_j)$ para cada $P_i \chi {}^\circ T_j$".Así por ejemplo, para la RdP de la Figura 1.6, si se define que el $Pre(P_1, T_1) = 2$, $Pre(P_2, T_2) = 3$ y que $Pre(P_3, T_3) = 1$, entonces las transiciones T_1 y T_3 serán habilitadas por el marcaje actual, mientras que la transición T_2 no.

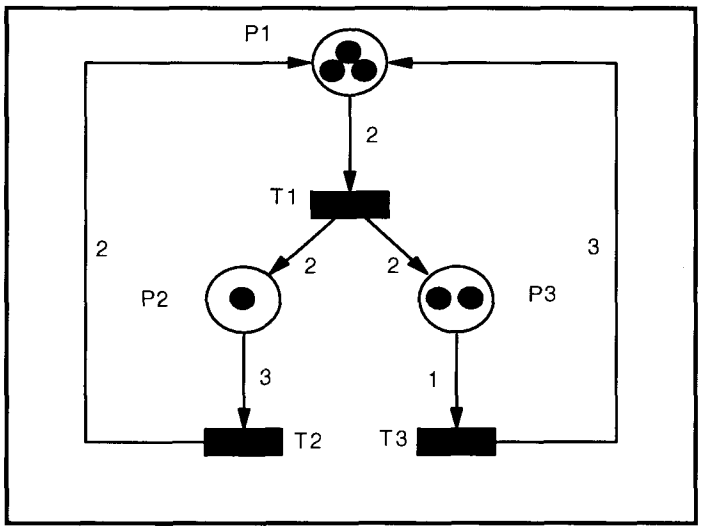


Figura 1.6. Ejemplo de transiciones habilitadas en un RdP.

Ahora se define **la matriz de incidencias de entrada** como se muestra en la Ecuación (2).

$$W^- = [w^-ij], \text{donde } w^-ij = Pre(Pi, Tj) \qquad (2)$$

Se define **la matriz de incidencia de salida** como se muestra en la Ecuación (3).

$$W^+ = [w^+ij], \text{donde } w^+ij = Post(Pi, Tj) \qquad (3)$$

De las Ecuaciones (2) y (3) se define la matriz de incidencia como se muestra en la Ecuación (4).

$$W = W^+ - W^- = [wij] \qquad (4)$$

Así una columna de la matriz de incidencia corresponde a la modificación en el marcaje, lo cual es causado a su vez, por el disparo de la transición correspondiente. Si una RdP es pura, entonces ésta puede ser reconstruida

de su matriz de incidencia, en caso contrario no podrá ser reconstruida. Por ejemplo, en la RdP de la Figura 1.6, si se definen las matrices W^- y W^+ como:

$$W^- = \begin{bmatrix} T_1 & T_2 & T_3 & \\ 2 & 0 & 0 & P_1 \\ 0 & 3 & 0 & P_2 \\ 0 & 0 & 1 & P_3 \end{bmatrix} \qquad W^+ = \begin{bmatrix} T_1 & T_2 & T_3 & \\ 0 & 2 & 3 & P_1 \\ 2 & 0 & 0 & P_2 \\ 2 & 0 & 0 & P_3 \end{bmatrix}$$

$$W = W^+ - W^- = \begin{bmatrix} T_1 & T_2 & T_3 & \\ -2 & 2 & 3 & P_1 \\ 2 & -3 & 0 & P_2 \\ 2 & 0 & -1 & P_3 \end{bmatrix}$$

En la Figura 1.7 se muestra la RdP antes y después del disparo de las transiciones habilitadas. Los signos positivos en la matriz W indica que se sumarán tokens a la plaza correspondiente, mientras que el signo negativo indica que se removerán tokens. Así por ejemplo, la columna 1, fila 1, indica que se removerán 2 tokens de la plaza P_1 cuando se dispare la transición T_1. Y la columna 2, fila 1, indica que se agregarán 2 tokens en la plaza P_1 cuando se dispare transición T_2.

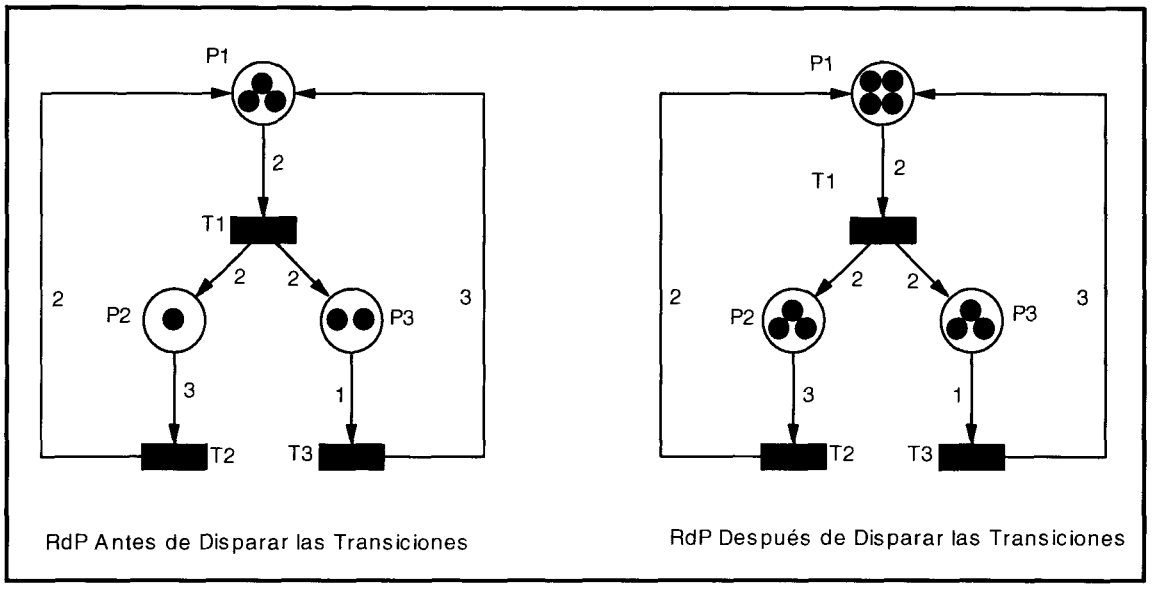


Figura 1.7. Ejemplo de una RdP antes y después de que se disparen las transiciones.

1.2.3.2 Ecuación fundamental.

Sea S una secuencia de disparos la cual será llevada a cabo a partir de un marcaje M_i. Así la representación, $M_i\ [S\rangle$ (significa que con la secuencia S, la RdP pasará de M_i a otra M_k). El vector característico de la secuencia S, representado por $\underline{S}$, representa el vector de m-componentes (donde m es el número de transiciones en la RdP) en el cual el componente j corresponde al número de disparos de la transición T_j en la secuencia S.

Si la secuencia de disparos S es tal que $M_i\ [S\rangle\ M_k$, entonces la ***ecuación fundamental*** se obtiene de la Ecuación (5).

$$M_k = M_i + W \bullet \underline{S} \qquad (5)$$

De la Ecuación (5) se puede obtener el marcaje M_k que resultará después de una secuencia de disparos predeterminada S. El vector característico $\underline{S}$ es

realizable si al menos una secuencia de disparos S es relacionada con el marcaje M_i.

De la Ecuación (5) se puede concluir que: *dos secuencias de disparos que tienen el mismo vector característico resultan en el mismo marcaje* M_k. En la Figura 1.8, se muestra una RdP, con un marcaje inicial M_i y una matriz de incidencia W, donde:

$$M_i = \begin{bmatrix} 3 & P_1 \\ 1 & P_2 \\ 2 & P_3 \end{bmatrix} \qquad W = \begin{bmatrix} T_1 & T_2 & T_3 & \\ -2 & 2 & 3 & P_1 \\ 2 & -3 & 0 & P_2 \\ 2 & 0 & -1 & P_3 \end{bmatrix}$$

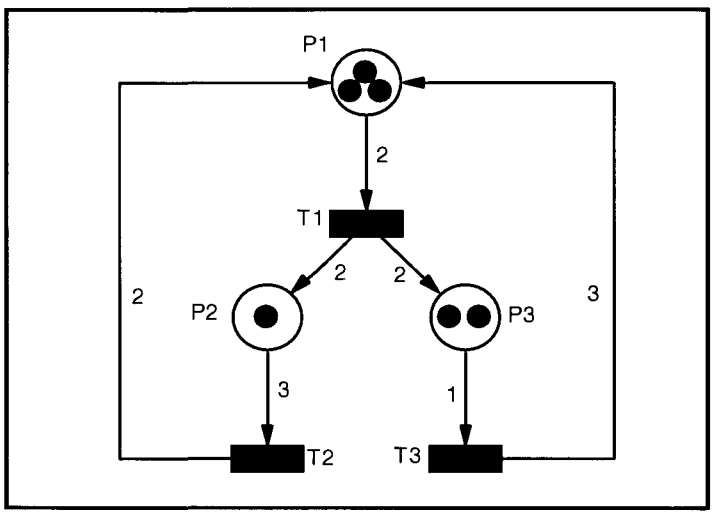


Figura 1.8. RdP para el ejemplo de la ecuación fundamental.

Si se disparan las transiciones T_1 y T_3, el vector característico $\underline{S}$ y la ecuación fundamental quedarían de la siguiente manera:

$$\begin{bmatrix} 3 & P_1 \\ 1 & P_2 \\ 2 & P_3 \end{bmatrix} + \begin{bmatrix} T_1 & T_2 & T_3 & \\ -2 & 2 & 3 & P_1 \\ 2 & -3 & 0 & P_2 \\ 2 & 0 & -1 & P_3 \end{bmatrix} \cdot \begin{bmatrix} 1 & T_1 \\ 0 & T_2 \\ 1 & T_3 \end{bmatrix} = \begin{bmatrix} 4 & P_1 \\ 3 & P_2 \\ 3 & P_3 \end{bmatrix}$$

M_i W $\underline{S}$ M_k

Así, si las transiciones T_1 y T_3 son disparadas la RdP tendrá 4 tokens en P_1, 3 en P_2 y 3 en P_3.

1.2.4 Redes de Petri Temporizadas.

Una RdP temporizada modela a un sistema en el cual su dinámica de funcionamiento está en función del tiempo. Este tipo de redes son muy usadas para evaluar el desempeño de un proceso. Existen dos métodos para modelar la dependencia del tiempo: si la dependencia del tiempo está asociada con las plazas, se dice que la RdP es *P-Timed*, o bien si está asociada con las transiciones, se dice que la RdP es *T-Timed*.

1.2.4.1 Red de Petri P-Timed.

Una RdP P-Timed es un par {*R, Tempo*} tal que *R* es una RdP marcada y *Tempo* es una función del conjunto *P* de plazas y el conjunto de números racionales positivos incluyendo al cero. Por ejemplo, $Tempo(P_i) = d_i$ es el tiempo del evento asociado con la plaza P_i, definido por d_i. En la Figura 1.9 se muestra una plaza con un tiempo $d_1 = 5$ segundos, lo que significa que el evento asociado con la plaza P_1 tendrá una duración de 5 segundos.

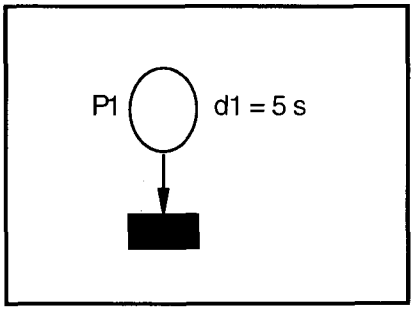


Figura 1.9. RdP P-Timed.

1.2.4.1.1 Principio de funcionamiento.

Cuando un token es depositado en la plaza P_i, éste debe permanecer en esta plaza al menos por un tiempo d_i. Durante este tiempo el token se encontrará no disponible. Una vez que el tiempo d_i ha transcurrido el token estará disponible.

La operación de una RdP P-Timed se define de la siguiente manera:

- En el primer instante el marcaje M_0 (conjunto inicial de tokens) está formado sólo por los tokens disponibles.
- En un tiempo t, el marcaje presente M_t es la suma de dos marcajes M^a y M^u, tal que M^a es el conjunto de tokens disponibles y M^u el de no disponibles. Entonces, una transición será habilitada por el marcaje $M_t = M^a + M^u$, si existen tokens en el marcaje M^a que la habiliten.
- El disparo de una transición es equivalente al de una RdP sin temporizar, es decir, sólo se remueven las marcas disponibles de las plazas de entrada, y se depositan en las plazas de salida, según los pesos de los arcos correspondientes. Los disparos de las transiciones tienen una duración igual a cero.
- Si una marca es depositada en una plaza P_i en un disparo que sucedió en un instante t, entonces ésta estará no disponible en un intervalo $(t,t+d_i)$.

En la Figura 1.10 se muestra una RdP antes de ser disparada (en tiempo t = 0) y 5 segundos después. Se aprecia como sólo la transición T_1 es disparada ya que en 5 segundos la token en P_1 estará disponible, mientras que las transiciones T_2 y T_3 serán habilitadas 1 y 2 segundos después del instante t = 5, respectivamente.

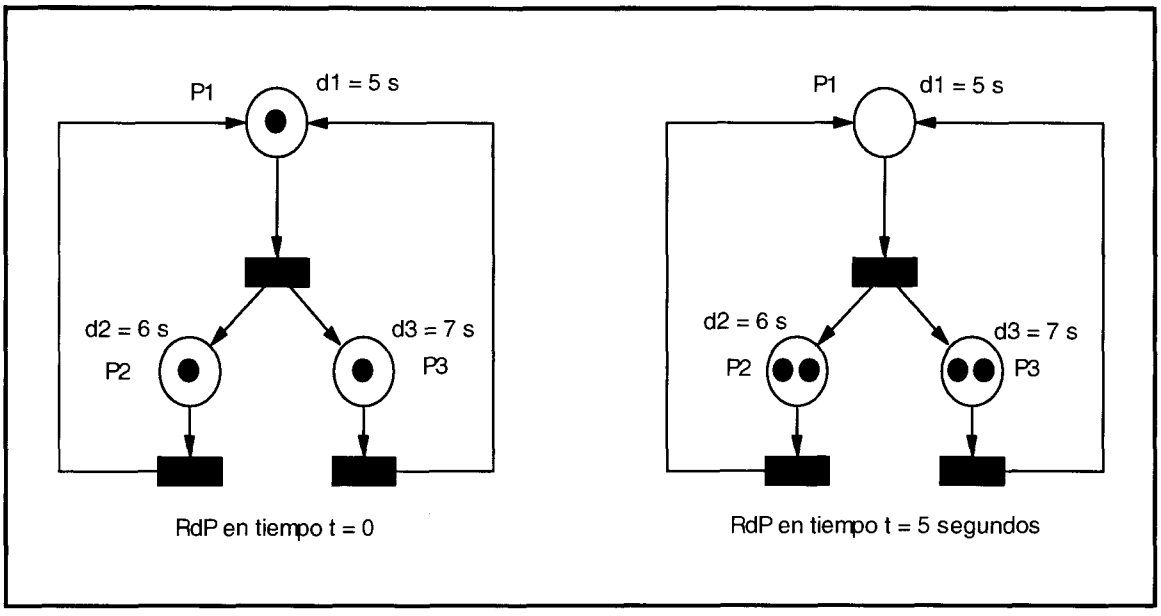


Figura 1.10. Ejemplo del principio de operación de una RdP P-Timed

El concepto de **funcionamiento a máxima velocidad** se refiere a que una transición es disparada tan pronto ésta es habilitada.

1.2.4.1.2 Conflictos efectivos.

Se define un conflicto efectivo como la existencia de un conflicto estructural, K, y un marcaje, M, tal que el número de tokens en la plaza P_i es menor que el número de transiciones de salida habilitadas de la plaza P_i. En la Figura 1.11, se muestra un conflicto efectivo K, en la plaza P_1, debido a que sólo existe un token en ésta y dos transiciones de salida habilitadas por el marcaje de la RdP.

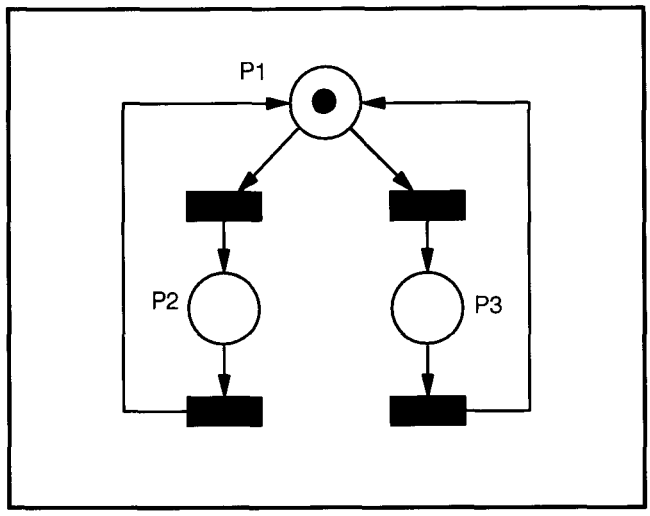


Figura 1.11. RdP con un conflicto efectivo en P_1.

1.2.4.1.3 Comportamiento Estacionario.

Sea R_T una RdP P-Timed, en la cual los tiempos asociados con las plazas son números racionales. Si no existen conflictos efectivos y partiendo de un marcaje inicial M_0, el funcionamiento a máxima velocidad se convierte en un *funcionamiento periódico* dentro de un tiempo finito, entonces R_T es limitada. Así, el comportamiento estacionario es alcanzado cuando el número de tokens que entra en una plaza es en promedio igual, al número de tokens que salen de ésta.

La *frecuencia de disparo* F_j de una transición T_j es la media de disparos de T_j por unidad de tiempo, una vez que la RdP se encuentra en el estado estacionario.

Así, se establece la siguiente propiedad:

Sea R_T un grafo de eventos fuertemente conectados P-Timed, en donde el conjunto de transiciones es $T = \{T_1, T_2, ..., T_m\}$, y en el cual el conjunto de circuitos elementales es $C = \{C_1, C_2, ..., C_k, ...\}$.

Sea $C_k = P_1T_1P_2T_2...P_rT_r$ un circuito elemental, $d(C_k) = d_1 + d_2 + ... + d_r$, y $M(C_k) = M_0(P_1) + M_0(P_2) + ... + M_0(P_r)$. Entonces,

- La *frecuencia de disparo* correspondiente a un *funcionamiento a máxima velocidad* del circuito elemental C_k está dada por la Ecuación (6).

$$F(Ck) = \frac{M(C_k)}{d(C_k)} \qquad (6)$$

- Todas las transiciones tienen la misma frecuencia de disparo.
- La *frecuencia de disparo máxima* para R_T está dada por la Ecuación (7).

$$F(R_T) = \min(F(C_1), F(C_2), ..., F(C_k), ...) \qquad (7)$$

Para la RdP de la Figura 1.12, por ejemplo, se tienen dos circuitos elementales dados por:

$$C_1 = P_1T_1P_2T_2$$
$$C_2 = P_3T_1P_2T_2$$

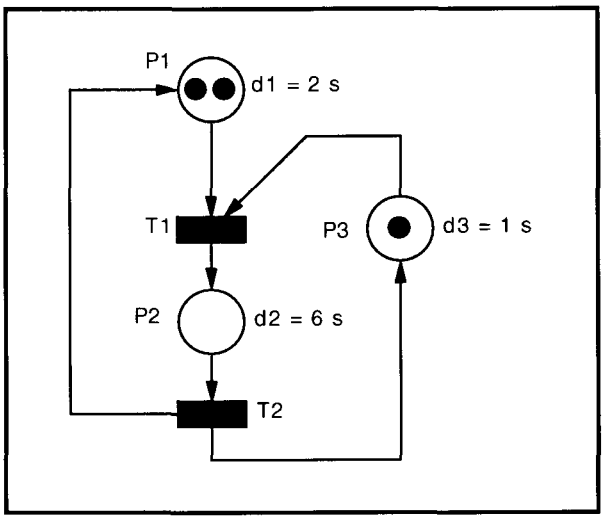


Figura 1.12. RdP para el ejemplo de la frecuencia máxima de disparo.

Se definen los valores de $M(C_1)$ y $M(C_2)$ para cada uno de los circuitos elementales, entonces,

$$M(C_1) = M_0(P_1) + M_0(P_2) = 2 + 0 = 2$$

$$M(C_1) = M_0(P_3) + M_0(P_2) = 1 + 0 = 1$$

Se obtienen los tiempos para cada uno de los circuitos elementales,

$$d(C_1) = d_1 + d_2 = 2 + 3 = 5$$

$$d(C_2) = d_3 + d_2 = 1 + 3 = 4$$

Se calculan las frecuencias de disparo para cada circuito y se obtiene, usando la Ecuación (7), la frecuencia de disparo de máxima velocidad para la RdP de la Figura 1.12, entonces,

$$F_1 = \frac{M(C_1)}{d(C_1)} = \frac{2}{5} = 0.4 \qquad F_2 = \frac{M(C_2)}{d(C_2)} = \frac{1}{4} = 0.25$$

$$\therefore \; F_{RdP} = \min(F_1, F_2) = \min(0.4, 0.25) = 0.25$$

Por lo tanto la RdP de la Figura 1.12, funcionará a una frecuencia máxima de 0.25 disparos por segundo.

1.2.4.2 Red de Petri T-Timed.

Una RdP T-Timed es un par {*R, Tempo*} tal que, *R* es una RdP marcada, y *Tempo* es una función del conjunto de transiciones *T* y el conjunto de números racionales positivos incluyendo al cero. Así, $Tempo(T_j) = d_j$ corresponde al tiempo del evento asociado con la transición T_j.

En la Figura 1.13, se muestra una RdP, con una transición de duración 3 segundos.

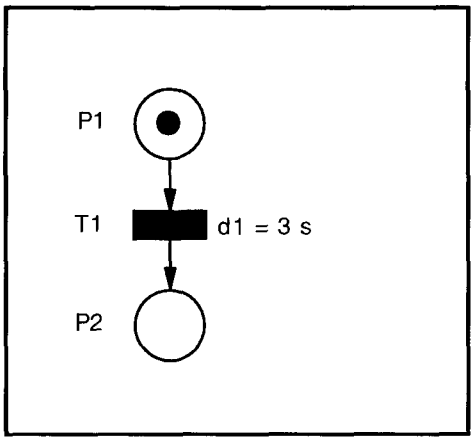


Figura 1.13. RdP T-Timed.

1.2.4.2.1 Principio de funcionamiento.

Un token puede tener dos estados: éste puede estar *reservado* para el disparo de una transición T_j, o puede estar *no-reservado*.

Si en un instante t se dispara una transición T_j con un tiempo d_j asociado, entonces en un tiempo igual a $(t + d_j)$ este disparo habrá terminado. Así, el tiempo t es el inicio y $(t + d_j)$ el fin del disparo. La transición T_j es habilitada si y solo sí los tokens no reservados en la plaza correspondiente son suficientes. Entonces, en el instante t los tokens requeridos para la transición T_j son reservadas y permanecerán así durante el intervalo de tiempo $(t,t + d_j)$. En la Figura 1.14, por ejemplo, se muestran dos RdP T-Timed, en el instante t = 0, se encuentra en la plaza P_1 un token reservado para la transición T_1, al cumplirse el tiempo d_1 = 3 s, este token pasará a la plaza P_2, cabe mencionar que la función de salida está dada por $Post(P_1, T_1)$ = 1. En la plaza P_2, se encuentra tres tokens, de los cuales dos están reservados por la transición T_2, y el otro se encuentra no-reservado, ya que la función de salida está definida por: $Post(P_1, T_2)$ = 2. Al cumplirse el tiempo d_2 = 3 s, la transición T_2 se disparará, en el lado izquierdo de la Figura 1.14 se muestra la RdP en el instante t = 3 segundos, es decir, después de que se han disparado las transiciones T_1 y T_2.

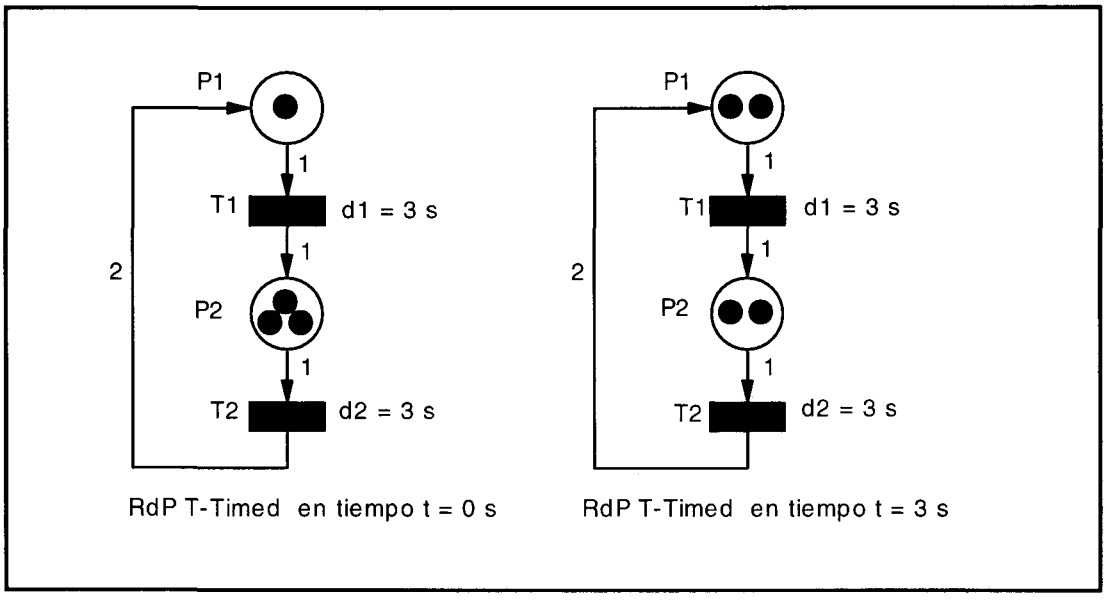


Figura 1.14. RdP T-Timed, mostrada paro los instantes t = 0 y t = 3 segundos.

El concepto de funcionamiento a máxima velocidad se aplica de igual manera que para las RdP P-Timed. Así, la media de las marcas en la plaza P_i es al menos igual, al producto de la frecuencia de su transición de salida T_j multiplicada por su duración d_j, en caso de que esta transición fuera única, de lo contrario será al menos igual a la suma de los productos $F_j d_j$ correspondientes a las transiciones T_j que pertenecen a P°_i.

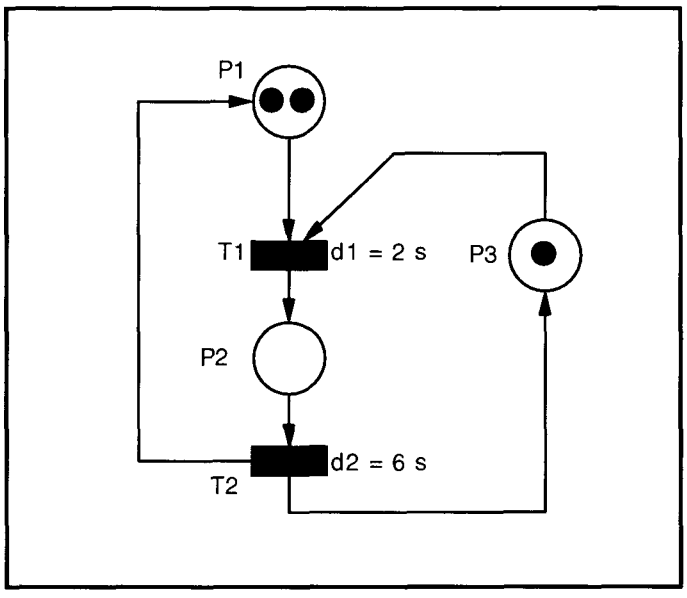


Figura 1.15. RdP T-Timed para el ejemplo de máxima frecuencia de disparo.

Así, para la RdP de la Figura 1.15, las transiciones tienen un tiempo d asociado. Así, es necesario definir los valores de $M(C_1)$ y $M(C_2)$, para cada uno de los circuitos elementales, entonces:

$$M(C_1) = M_0(P_1) + M_0(P_2) = 2 + 0 = 2$$

$$M(C_1) = M_0(P_3) + M_0(P_2) = 1 + 0 = 1$$

Posteriormente es necesario calcular los tiempos de ejecución para cada uno de los circuitos elementales, así:

$$d(C_1) = d_1 + d_2 = 2 + 6 = 8$$

$$d(C_2) = d_1 + d_2 = 2 + 6 = 8$$

Con estos datos se calculan las frecuencias de disparo para cada circuito y se obtiene, usando la Ecuación (7), la frecuencia de disparo de máxima velocidad para la para la RdP de la Figura 1.15, entonces:

$$F_1 = \frac{M(C_1)}{d(C_1)} = \frac{2}{8} = 0.25 \qquad F_2 = \frac{M(C_2)}{d(C_2)} = \frac{1}{8} = 0.125$$

$$\therefore \; F_{RdP} = \min(F_1, F_2) = \min(0.25, 0.125) = 0.125$$

Así, la RdP de la Figura 1.15 funcionará a una frecuencia máxima de 0.125 disparos por segundo.

1.2.4.3 Equivalencia de las Redes de Petri P-Timed y T-Timed.

La transformación de una RdP T-Timed a una P-Timed se logra cambiando la transición T_j, que tiene asociado un tiempo d_j, por una plaza P'_j la cual tendrá un tiempo $d'_j = d_j$, y dos transiciones T'_j y T''_j, que serán de entrada y salida a la plaza P'_j respectivamente. En la Figura 1.16, se muestra un ejemplo de esta conversión.

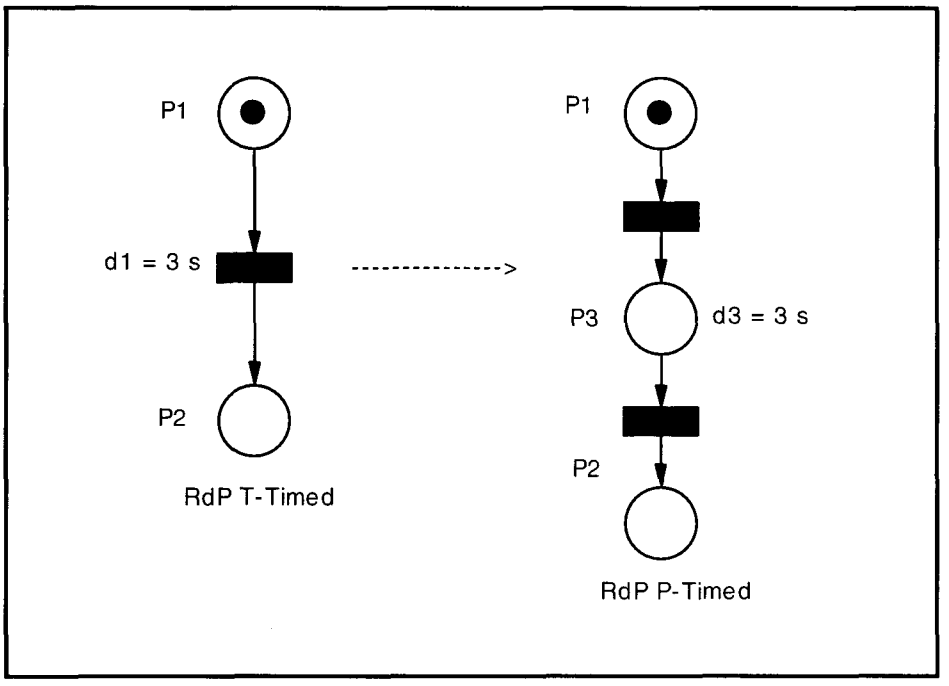


Figura 1.16. Conversión de una RdP T-Timed a una RdP P-Timed.

La transformación necesaria para obtener una RdP T-Timed a partir de una RdP P-Timed se basa en cambiar la plaza P_j por dos plazas, P'_j y P''_j, y una transición T''_j, la cual tendrá un tiempo asociado igual que el de la plaza P_j. En la Figura 1.17 se muestra un ejemplo de esta transformación.

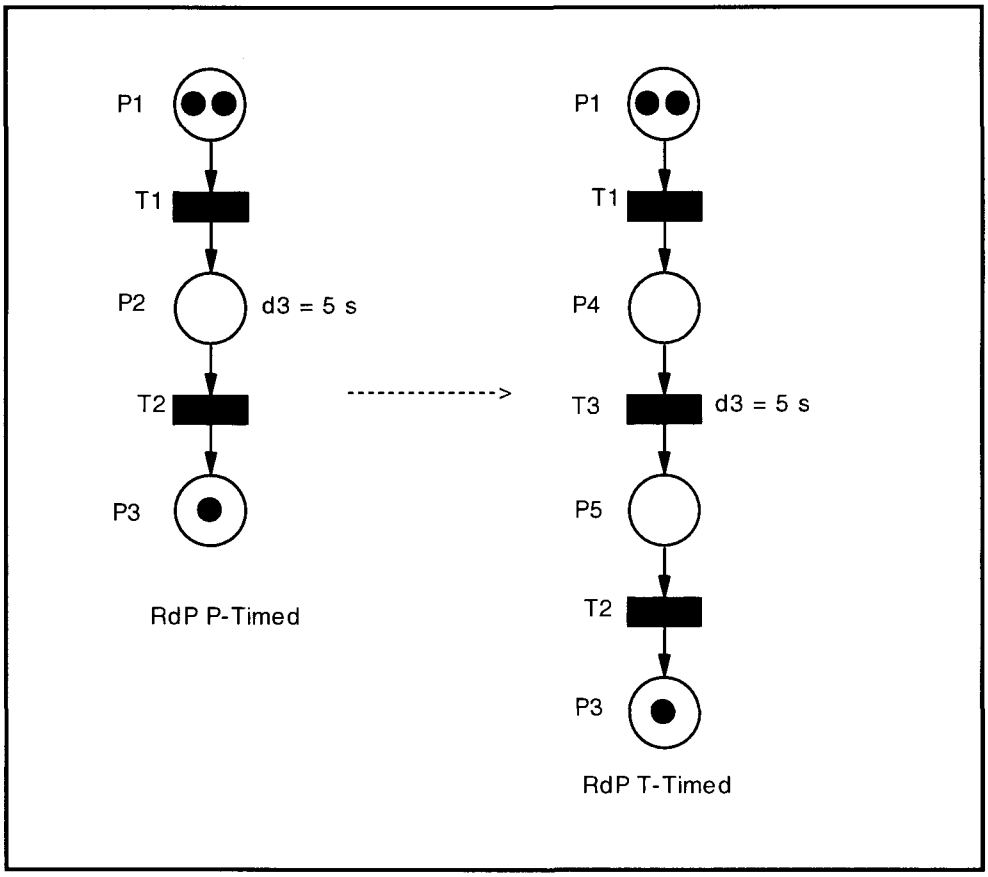


Figura 1.17. Conversión de una RdP P-Timed a una RdP T-Timed.

1.3 Autómatas Discretos [5]

1.3.1 Lenguajes Formales.

En el manejo de autómatas es necesario manejar diferentes tipos de lenguajes, uno de ellos son los *lenguajes formales*.

1.3.1.1 Notaciones y definiciones elementales.

Un *alfabeto* (o vocabulario) es un conjunto finito de eventos que representan un SED, es decir, es el conjunto de posibles eventos en el sistema. Una *cadena* (una palabra o secuencia) definida en un alfabeto Σ es una serie finita de eventos. La *longitud* de una cierta cadena x representada como $|x|$, es el número de eventos en ésta. Por ejemplo si el alfabeto de un SED está definido por Σ = {a, b, c}, y se forma una cadena x = ab, entonces la longitud de la cadena x estará dada por, $|x| = 2$.

El símbolo Σ^* representa el conjunto de cadenas de una longitud n cualquiera que pueden ser construidas a partir del alfabeto Σ, tal como se muestra en la Ecuación (8).

$$\Sigma^* = \bigcup_{i \geq 0} \Sigma^i \qquad (8)$$

Así, por ejemplo, para el alfabeto Σ = {a, b, c}, y ε el elemento vacío, tendremos, Σ^* = {ε, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, ...}.

Un *lenguaje* (L) definido sobre el alfabeto Σ, será un subconjunto de Σ^*, ya que Σ^* contiene todas las posibles cadenas que se pueden formar con los eventos del alfabeto Σ. Un *lenguaje infinito* es aquel que tiene un número infinito de cadenas. Por ejemplo si $L_1 = \Sigma^*$, y $L_2 = \Sigma^2$ entonces L_1 representa un *lenguaje infinito* y L_2 un *lenguaje finito* ya que contiene un número finito de cadenas.

Así, por ejemplo, para el alfabeto $\Sigma = \{a, b, c\}$, se forma $L_1 = \Sigma^* = \{\varepsilon, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc, aaa, ...\}$ entonces, L_1 representa un lenguaje infinito, ya que contiene un número infinito de cadenas, mientras que $L_2 = \Sigma^2 = \{\varepsilon, a, b, c, aa, ab, ac, ba, bb, bc, ca, cb, cc\}$ contiene un número finito y es un lenguaje finito.

Si w es una cadena definida en el alfabeto Σ, se define a la cadena v como el *prefijo* de w si existe una cadena $x \in \Sigma^*$ tal que $w = vx$. Así, por ejemplo, el conjunto de prefijos de la cadena "abba" es: $\{\varepsilon, a, ab, abb, abba\}$. Un *Prefix-Closed* ($L^-$) es el conjunto que contiene todos los prefijos de las cadenas de un lenguaje L. Entonces, por ejemplo, si $L = \{abba\}$ su prefix-closed está definido por $L^- = \{\varepsilon, a\ ab, abb, abba\}$. Por definición L^- siempre contiene a L.

Un *Prefix-Clos* es un lenguaje que es exactamente igual a su Prefix-Closed, es decir $L = L^-$. Entonces del ejemplo anterior, el lenguaje $L = \{abba\}$ no es Prefix-Clos, mientras que el lenguaje $L^- = \{\varepsilon, a\ ab, abb, abba\}$ si lo es, ya que contiene a todos sus prefijos [5].

1.3.1.2 Operaciones con lenguajes.

Si L_1 y L_2 son dos lenguajes construidos de un mismo alfabeto Σ. Entonces, se definen las siguientes operaciones con lenguajes:

- La *unión* de L_1 y L_2, será denotada como $L_1 \cup L_2$, y es un lenguaje que contiene todas las cadenas contenidas en L_1 o L_2. Así, por ejemplo para $L_1 = \{a\}$ y $L_2 = \{a, b, c\}$, entonces $L_1 \cup L_2 = \{a, b, c\}$.
- La *intersección* de L_1 y L_2 será denotada como $L_1 \cap L_2$, y es un lenguaje que contiene todas las cadenas contenidas en L_1 y L_2. Por ejemplo para $L_1 = \{a\}$ y $L_2 = \{a, b, c\}$, entonces $L_1 \cap L_2 = \{a\}$.
- La *concatenación* de L_1 y L_2 será denotada por $L_1 \bullet L_2$, y contiene todas las cadenas formadas de una cadena de L_1 seguida de una cadena de

L_2. Así, por ejemplo, para $L_1 = \{a\}$ y $L_2 = \{a, b, c\}$, entonces la concatenación está dada por $L_1 \bullet L_2 = \{ \varepsilon, aa, ab, ac\}$.

- La *Kleene-closure* de un lenguaje L, será denotada por L^*, y es el conjunto de cadenas formadas por una concatenación finita de cadenas de L. Es decir, $L^* = \{ \varepsilon \} \cup L \cup LL \cup LLL \cup \ldots$. Por ejemplo, para el lenguaje $L = \{aa, bb\}$, se obtiene la Kleene-closure $L^* = \{ \varepsilon, aa, bb, aaaa, aabb, bbaa, bbbb, ...\}$.
- La *proyección* del lenguaje L, del alfabeto Σ, en el alfabeto Σ' (donde $\Sigma' \subseteq \Sigma$), es denotada por PROJ $[\Sigma \rightarrow \Sigma']$(L), y es el lenguaje que se obtiene eliminando de todas las cadenas de L, los símbolos que no pertenecen al alfabeto Σ'. Así. por ejemplo, dado el lenguaje $L = \{ \varepsilon, ac, bc, cab, abc, cbca\}$ construido a partir del alfabeto $\Sigma = \{a, b, c\}$, y dado que el alfabeto $\Sigma' = \{a, b\}$ está contenido en Σ, el lenguaje L' = PROJ $[\Sigma \rightarrow \Sigma']$(L) representa el conjunto de cadenas obtenidas a partir de las cadenas de L, suprimiendo el símbolo c. Entonces, $L' = \{\varepsilon, a, b, ab, ba\}$.
- La *proyección inversa* del lenguaje L', del alfabeto Σ', en el alfabeto Σ (donde $\Sigma' \subseteq \Sigma$), es denotada por PROJ^{-1} $[\Sigma' \rightarrow \Sigma]$(L'), es el conjunto de cadenas obtenidas al insertar en cualquier plaza de las cadenas de L', un número cualquiera de símbolos Σ / Σ' (donde Σ / Σ' significa que pertenece a Σ y que no pertenece a Σ'). Así, por ejemplo, dado el lenguaje $L' = \{ab\}$, del alfabeto $\Sigma' = \{a, b\}$, y el alfabeto $\Sigma = \{a, b, c\}$ que contiene a Σ', se obtiene el lenguaje L = PROJ^{-1} $[\Sigma' \rightarrow \Sigma]$(L'), el cual está definido por $L = \{ab, cab, acb, abc, ccab, cacb, accb, acbc, abcc, ...\}$.

1.3.1.3 Expresiones regulares y lenguajes regulares.

Las *expresiones regulares* constituyen una herramienta que permite representar de una manera concisa un tipo de lenguajes, llamados *lenguajes regulares.* Una expresión regular de un alfabeto Σ, es una expresión donde

los operandos son los símbolos de Σ y los operadores son el conjunto formado por la unión {+}, la concatenación {•} y la iteración cerrada {*}. Entonces, por ejemplo, la expresión regular b+a•b*, representa el lenguaje: {b} ∪ {a}•{b}*.

Un *lenguaje regular*, es un lenguaje que está definido por una expresión regular tal como se muestra en la Figura 1.18. Un lenguaje finito proviene de una expresión regular, así, los lenguajes regulares son gran utilidad para la modelación de los SED. Por ejemplo el lenguaje {b} ∪ {a}•{b}* es un lenguaje regular, ya que es representado por la expresión regular b+a•b*.

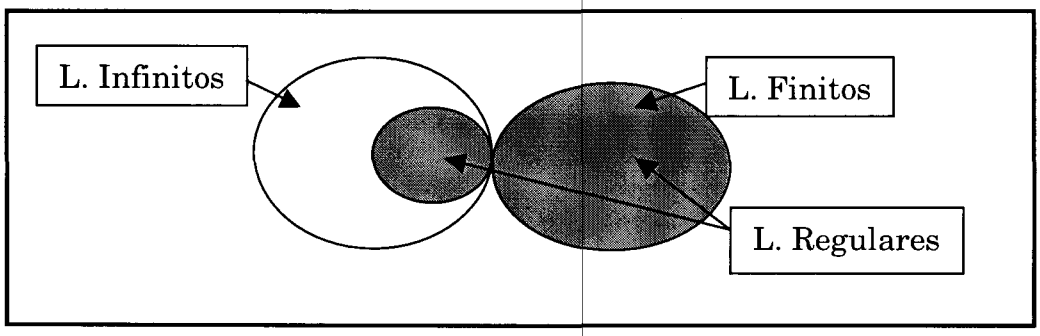


Figura 1.18. Ubicación de un lenguaje regular.

1.3.2 Modelos de Autómatas.

Un *autómata* es una máquina de estados que permite describir el funcionamiento de un sistema, es decir, describe la dinámica de las entradas y las salidas de éste.

1.3.2.1 Autómatas determinísticos.

Un autómata tiene entradas y salidas discretas y a partir de una modificación de sus entradas existe un cambio en sus salidas.

Un autómata determinístico M es un séxtuple definido por:

$$M = (Q, \Sigma, \delta, q_o, F, \Gamma),$$

donde:

- Q es el conjunto de estados.
- Σ es un conjunto finito de los eventos de entrada, o denominado también como el alfabeto de entrada.
- δ es la función de transición de estados Q x Σ.
- q_o es el estado inicial.
- F es el conjunto de estado finales tal que $F \subseteq Q$.
- Γ es el conjunto de eventos prohibidos o no permisibles en cada estado del autómata.

Un *autómata finito* es aquel en donde Q es un conjunto finito de estados, mientras que un *autómata infinito* es aquel donde Q es un conjunto infinito de estados.

Un autómata M es determinístico desde el punto de vista que para todo estado, no existen dos transiciones de salida que estén asociadas a un mismo símbolo y que conduzcan a dos estados diferentes. En un autómata M, una cadena de eventos es *aceptable (o reconocida)*, si partiendo de un estado inicial y ejecutando los eventos de la cadena de entrada, el autómata M es conducido a un estado final. Un *lenguaje aceptado* para un autómata es el conjunto de cadenas aceptables para el autómata, como se muestra en la Ecuación (9).

$$L(M) = \{x \in \Sigma^* \mid \delta(q_o, x) \in F\} \qquad (9)$$

En la Figura 1.19 se muestra un modelo de autómata determinístico M, y se observa como un autómata puede ser representado por un gráfico de transición de estados. Los estados iniciales son marcados con una flecha

entrante, mientras que los estados finales son representados con un doble círculo.

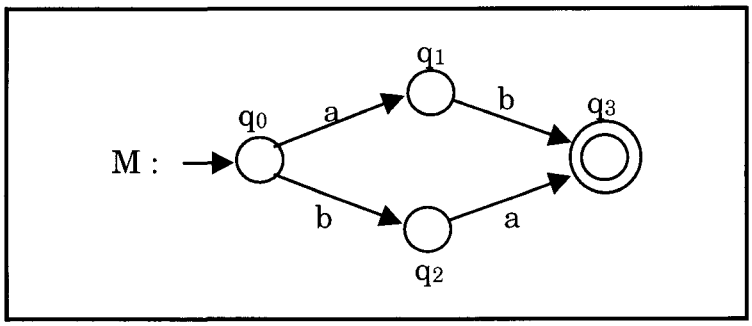


Figura 1.19. Modelo gráfico para el autómata M.

1.3.3 Modelos aceptadores.

Un *aceptador* A es definido como un cuádruple, A = (Q, Σ, δ, q_0, Γ). La diferencia con la definición del autómata determinístico es que el aceptador no contempla un conjunto de estados finales F. En un aceptador, la función δ es llamada *parcial* ya que no está definida para todos los elementos del producto Q x Σ. El *lenguaje reconocido* por un aceptador A es representado por la Ecuación (10).

$$L(A) = \{x \in \Sigma^* \mid \delta(q_o, x) \in Q\} \qquad (10)$$

En la Figura 1.20, por ejemplo, se muestra el modelo gráfico del aceptador A, en este modelo no se indica ningún estado final, pero si un estado inicial. Una propiedad de los aceptadores es que su lenguaje reconocido es Prefix-Clos.

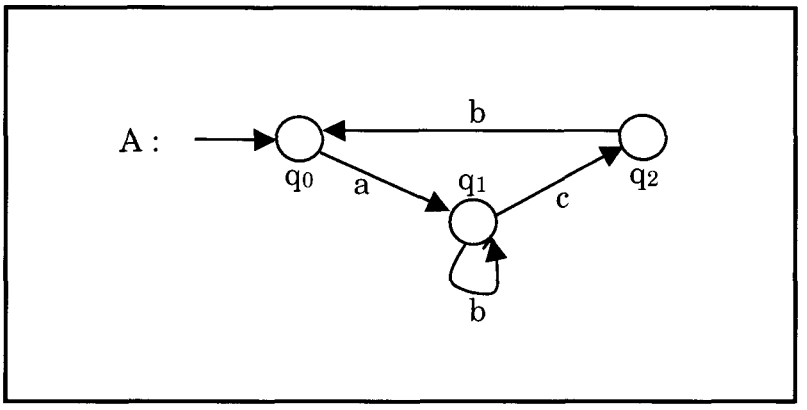


Figura 1.20. Modelo gráfico del aceptador A.

1.3.3.1 Composición sincrónica de dos modelos aceptadores.

Dados dos aceptadores $A_1 = \{Q, \Sigma_1, \delta, q_o\}$ y $A_2 = \{X, \Sigma_2, \xi, x_o\}$ la composición sincrónica M de A_1 y A_2 es representada por la Ecuación (11).

$$M = A_1 \,||_s\, A_2 \qquad (11)$$

y está definida por el cuádruple $(Q \times X,\ \Sigma_1 \cup \Sigma_2,\ \delta \times \xi,\ q_o \times x_o\}$ donde:

- $Q \times X$, es el conjunto de estados.
- $\Sigma_1 \cup \Sigma_2$, es el alfabeto de M.
- $q_o \times x_o$ es el estado inicial de M.
- $\times\ \xi$, es la función de transición de estados. Y está definida de la siguiente manera:

  - Para $a \notin \Sigma_1 \cap \Sigma_2$:
    - $(\delta \times \xi)((q, x),a) = (q', x)$ si $\delta(q, a) = q'$ si $a \in \Sigma_1$.
    - $(\delta \times \xi)((q, x),a) = (q, x')$ si $\xi(x, a) = x'$ si $a \in \Sigma_2$.
  - Para $a \in \Sigma_1 \cap \Sigma_2$:
    - $(\delta \times \xi)((q, x),a) = (q', x')$ si $\delta(q, a) = q'$ y $\xi(x, a) = x'$.

El lenguaje reconocido para un aceptador se define en la Ecuación (12).

$$L(M) = PROJ^{-1}[\Sigma_1 \rightarrow \Sigma]L(A_1) \cap PROJ^{-1}[\Sigma_2 \rightarrow \Sigma]L(A_2) \qquad (12)$$

Si los dos alfabetos son *idénticos*, entonces un lenguaje reconocido por la composición sincrónica M está definido por la Ecuación (13).

$$L(M) = L(A_1) \cap L(A_2) \qquad (13)$$

En la Figura 1.21, por ejemplo, se muestran dos aceptadores, A_1 y A_2. El aceptador A_1 está construido basándose en el alfabeto de dos símbolos {p, j} El aceptador A_2 está construido basándose en el alfabeto de dos símbolos {b, j}

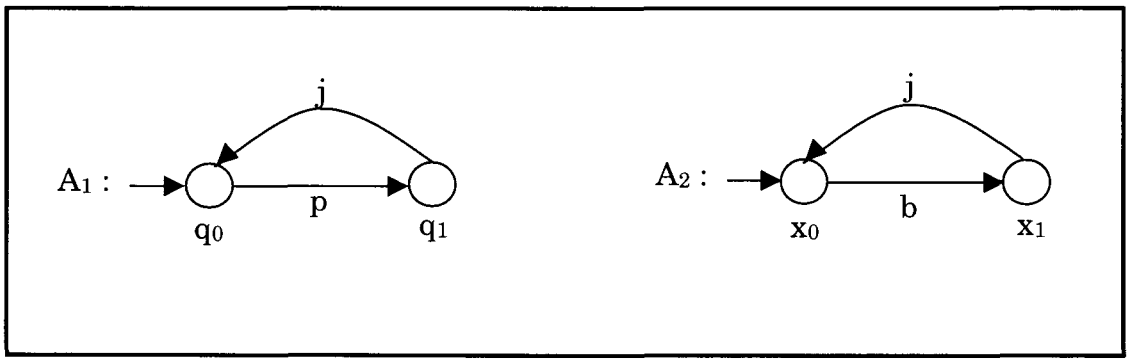


Figura 1.21. Modelos aceptadores A_1 y A_2.

Aplicando la composición sincrónica para los aceptadores A_1 y A_2, se obtiene el modelo global de la máquina, mostrado en la Figura 1.22.

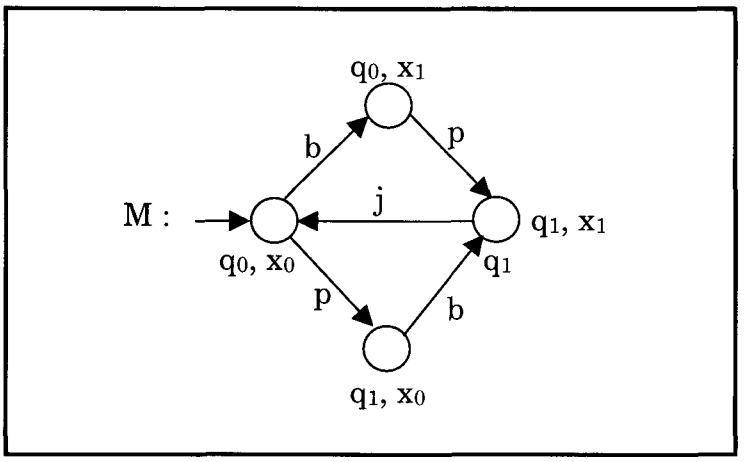


Figura 1.22. Modelo gráfico del aceptador global M.

1.4 GRAFCET [7]

1.4.1 Introducción a GRAFCET.

El GRAFCET o ***Graphe de Comande Etape Transition*** [16], es un lenguaje de programación de PLCs que proviene de un esfuerzo entre personal académico y de la industria por ayudar al programador y/o usuario a lograr una mejor representación y entendimiento del control de eventos discretos, y por consiguiente, lograr una mejor elaboración de su programa de control. También es conocido como el lenguaje de programación GRAFCET/SFC [13].

Un controlador lógico es un SED que tiene como función mantener al sistema dentro de un perfil de comportamiento, utilizando como herramienta la información de las entradas y la modificación de las salidas del SED.

Mientras que los SED fueron pequeños, los modelos clásicos eran suficientes para especificar las reglas de control lógico. Uno de los primeros modelos fue el de los circuitos secuenciales asincrónicos. Los modelos posteriores fueron basados en la representación de las funciones de control, tal y cual se implementaban en los tableros eléctricos, la más famosa era la lógica de relevador en escalera (o relay ladder logic RLL por sus siglas en inglés). Cuando hubo la necesidad de manejar más salidas y entradas, los modelos anteriores comenzaron a presentar ciertas limitantes, fue entonces cuando nacieron las siguientes necesidades para el diseño de controladores lógicos:

- Describir la secuencia de estados de un SED el cual puede contener un número muy grande de estados.
- Tomar en cuenta la concurrencia, ya que algunos subsistemas pueden ser parcialmente independientes.

- En general, dado un estado del sistema, sólo unas cuantas entradas pueden afectarlo, y sólo unas cuantas salidas pueden cambiarlo. Entonces, sólo se debe describir el comportamiento correspondiente al cambio de esas entradas.
- Obtener un claro entendimiento del comportamiento de las entradas y de las salidas para un controlador lógico.

El lenguaje de programación GRAFCET [7] vino a cubrir estas necesidades, éste está basado en las RdP, las cuales representan un modelo formal para la especificación de controladores lógicos. De aquí en delante se manejará GRAFCET (en mayúsculas) como el lenguaje de programación usado en algunos controladores comerciales, y Grafcet (en minúsculas) como la herramienta de modelación de eventos discretos (grafo)*.

En GRAFCET, una etapa está definida por su estado interno y el estado de sus salidas.

Algunas de las limitaciones de los primeros modelos son:

- Las tablas de transición de estados son muy detalladas, por lo que resultan poco prácticas cuando el número de entradas en el sistema es muy grande, así, por ejemplo, un sistema con 20 entradas tendría 1,048,576 posibles combinaciones posibles que tienen que ser definidas.
- La mayor desventaja de los diagramas de estado es que no es posible tomar en cuenta la concurrencia.
- La limitación de los RLL es que para llevar a cabo un análisis del sistema se requiere de mucho tiempo, aún cuando los sistemas son medianamente grandes.

---

* Gráfico que relaciona estados, transiciones y eventos del SED.

1.4.2 Condiciones y eventos en GRAFCET.

Las condiciones y eventos representan la información que define el comportamiento de un controlador lógico. El estado de un SED siempre puede ser definido por valores Booleanos, es decir los eventos continuos pueden ser discretizados usando variables booleanas.

La ocurrencia de un *evento,* que no tiene una duración fija, provoca un cambio de estado en el sistema. La notación utilizada para indicar que un evento ha ocurrido con una transición positiva (de 0 a 1) es: $\uparrow x_1$, y para una transición negativa (de 1 a 0) es: $\downarrow x_1$)

1.4.2.1 Definiciones importantes en el Álgebra de Eventos.

En el álgebra de eventos de Grafcet existen algunas definiciones importantes como lo son:

- Dos eventos e_1 y e_2 son independientes si no existe ningún evento e_i que los relacione.
- Se denomina $e_1 = \phi$ si e_1 es un evento que no puede ocurrir.
- Si $e_1 = e_2$ entonces los dos eventos suceden SIEMPRE al mismo tiempo.
- El producto de un evento y una condición es un evento.
- La probabilidad de que dos eventos independientes sucedan al mismo tiempo es cero.
- Los teoremas de álgebra booleana se aplican en el análisis de eventos discretos.

1.4.3 Definiciones en GRAFCET.

En esta sección se introducirán los elementos que conforman el lenguaje GRAFCET, así como la herramienta de modelación de eventos discretos Grafcet. Cabe mencionar que GRAFCET contiene un modelo Grafcet como parte de su estructura.

1.4.3.1 Etapas.

Una etapa es representada por un cuadrado. Si una etapa está activa el cuadrado es "pintado" de negro. El estado inicial de un sistema se representa con un cuadro doble. En la Figura 1.23, por ejemplo, se muestra una etapa inactiva, una activa y una inicial.

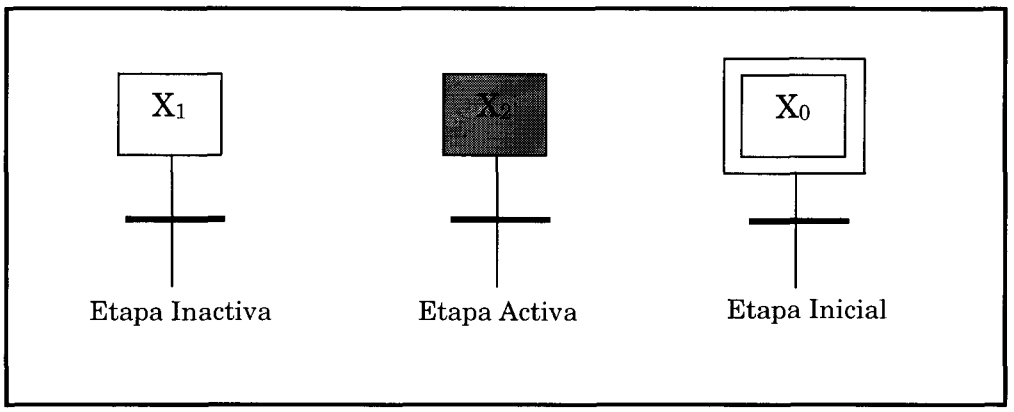


Figura 1.23. Etapas de GRAFCET.

1.4.3.2 Transiciones.

Las transiciones están relacionadas con las etapas. Una transición vertical sin flecha, significa que su dirección es de arriba hacia abajo, mientras que una transición vertical de abajo hacia arriba *siempre* debe incluir flechas de dirección en la unión directa, como se explicará en el punto 1.4.3.3. Los diferentes tipos de transiciones en Grafcet se muestran en la Figura 1.24.

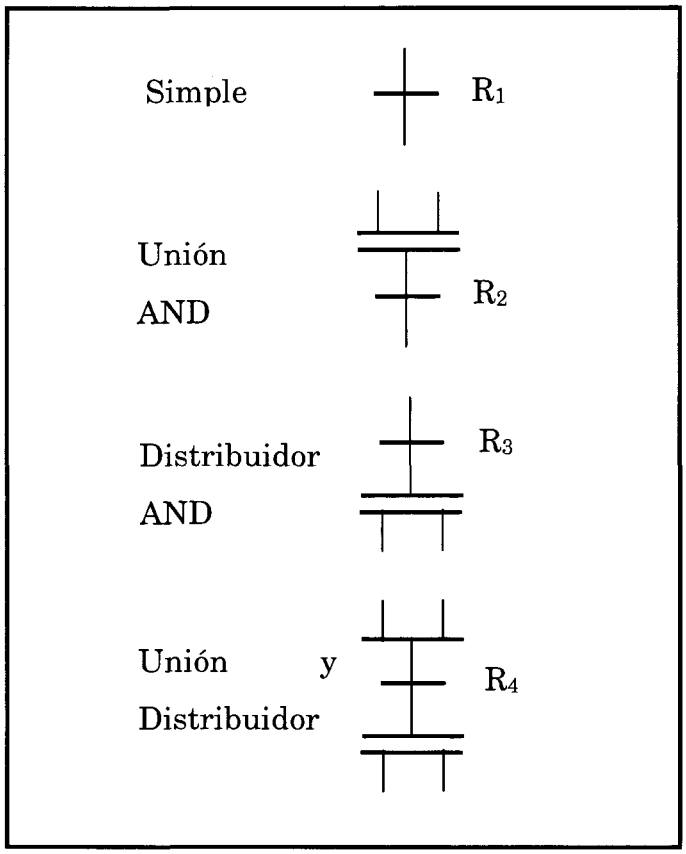


Figura 1.24. Diferentes tipos de transiciones en Grafcet.

1.4.3.3 Uniones directas (o Arcos).

Una unión directa siempre debe ir de una etapa a una transición o bien de una transición a una etapa, además una unión directa siempre debe de tener un nodo de salida y un nodo de llegada Cuando dos uniones directas tienen como llegada la misma etapa entonces estas deben ser agrupadas en un punto en común, al igual cuando dos uniones salen de una misma etapa.

Para representar una unión directa con dirección de abajo hacia arriba es necesario dibujar una flecha de dirección sobre la línea. En la Figura 1.25, se muestran los diferentes tipos de uniones directas en Grafcet.

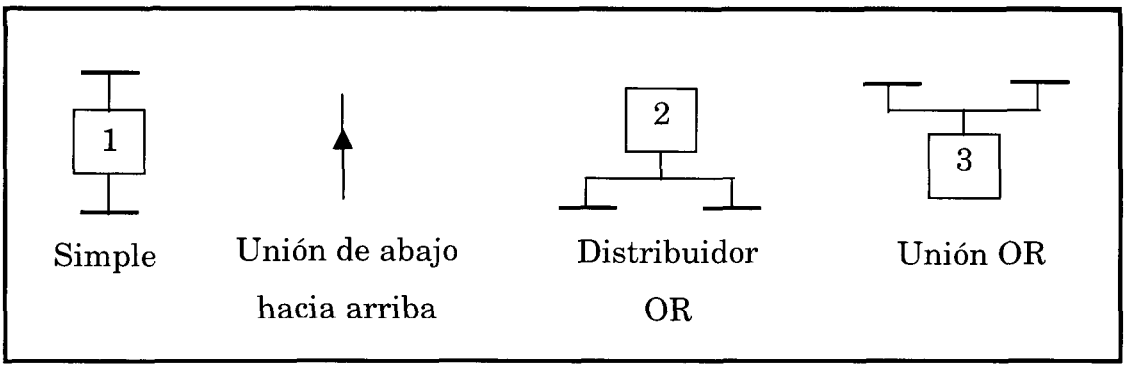


Figura 1.25. Diferentes casos de uniones directas en Grafcet.

En una etapa es posible que no existan transiciones de entrada y/o salida, al igual en una transición es posible que no existan etapas de entrada y/o salida. Una transición sin etapas de entrada es conocida como una *transición fuente (source),* y una sin etapas de salida como una *transición terminal (sink).*

1.4.3.4 Disparo de Transiciones.

La evolución del estado actual del SED se logra por el disparo de las transiciones en éste. Las entradas del controlador lógico están asociadas con las transiciones y las salidas con las etapas. Así, una transición puede ser disparada si y sólo si las dos siguientes condiciones se cumplen:

- Todas las etapas que preceden a la transición están activas, es decir, la transición se encuentra habilitada.
- La receptividad de la transición es verdadera.

La *receptividad* (R) de una transición se puede presentar en tres casos:

- Si la receptividad es una condición, por ejemplo R = a, donde "a" es una variable booleana.
- Si la receptividad es un evento, por ejemplo R = ↑a.

- Si la receptividad es el producto de un evento y una condición, por ejemplo R = ↑a + b, donde "b" es una variable booleana.

Una receptividad es un evento, ya que en general ésta puede ser representada como el producto de un evento y una condición. Así, se definen las reglas que rigen el disparo de una transición, las cuales son:

Regla 1. Todas las transiciones disparables son disparadas inmediatamente.
Regla 2. Todas las transiciones que llegan a ser disparables simultáneamente son simultáneamente disparadas.
Regla 3. Cuando una etapa es simultáneamente activada y desactivada, ésta permanece activa.

1.4.3.5 Acciones y Salidas.

Existen dos categorías de *acciones* en Grafcet:

- *Acción de Nivel.* La acción de nivel es modelada por una variable booleana y puede ser condicional o incondicional. Este tipo de acciones sólo permanecen encendidas cuando la etapa en la que se encuentran está activa. Cuando son condicionales, las acciones de nivel dependen de la etapa activa y del estado de otras variables booleanas, además sólo se aplican para estados estables.
- *Acción de Impulso.* Una acción de impulso cambia el valor de una variable discreta. Este tipo de acciones se activa cuando la etapa asociada pasa de un estado inactivo a uno activo sin importar el tiempo en el que la etapa se encuentre activa. Una acción impulso tiene como duración un tiempo muy pequeño, aún cuando el estado es inestable.

Las *salidas* son las señales que actúan sobre el SED. Así, es posible tener *acciones* que sirvan como señales de control dentro del programa de ejecución de GRAFCET, y *salidas* que afecten al SED para propósitos de control, por ejemplo.

1.4.3.6 Concurrencia y Sincronización.

La *concurrencia* define que "n" (donde n es un número entero) etapas pueden ser activadas con el disparo de una sola transición y a la vez ejecutadas independientemente.

La *sincronización* define que "n" etapas independientes pueden ser unidas en una transición común y sigan una misma secuencia.

La *concurrencia* y la *sincronización* son representadas por una línea horizontal doble en el diagrama Grafcet. En la Figura 1.26, por ejemplo, se muestra un modelo Grafcet con un caso de concurrencia y sincronización.

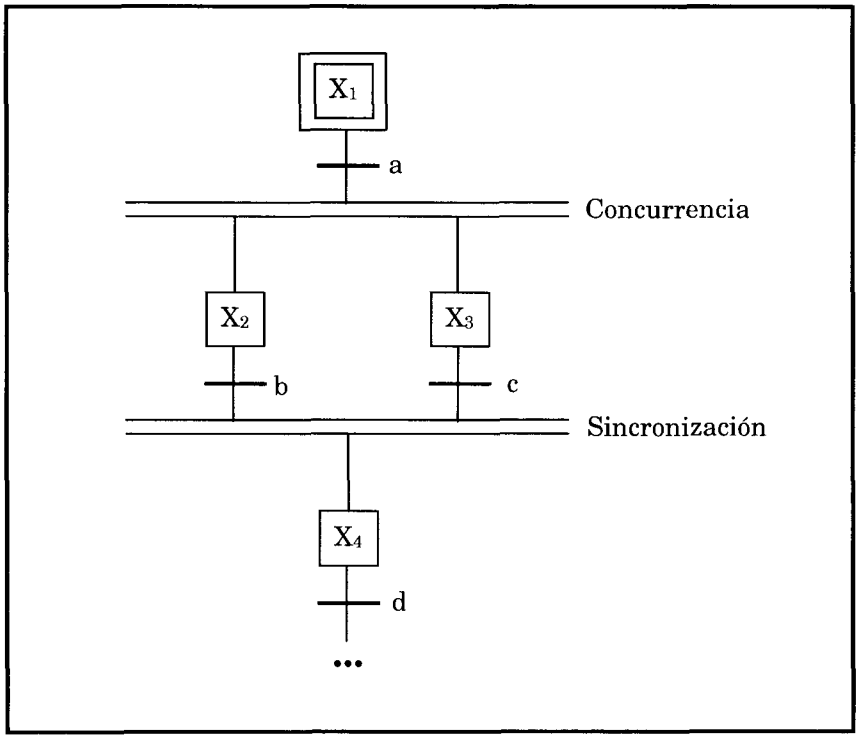


Figura 1.26. Concurrencia y sincronización en el modelo Grafcet.

1.4.3.7 Definiciones del Estado Interno y el Tiempo.

Se define:

- La variable X_i es booleana y es igual a 1 cuando la etapa i se encuentra activa, esta variable define el estado interno.
- La variable $t\ /\ i\ /\ \Delta$ es booleana y es igual a 1 si ha transcurrido un tiempo mayor o igual a Δ desde la última vez que el estado i fue activado.

1.4.3.8 Macroetapas y Macroacciones en Grafcet.

Las macroetapas y las macroacciones corresponden a abreviaciones del Grafcet que facilitan su programación y representación esquemática.

1.4.3.8.1 Macroetapas.

El objetivo de la macroetapa es facilitar la descripción de los sistemas complejos y clarificar la representación gráfica de un Grafcet detallando sólo ciertas partes por separado. Una macroetapa es representada por un cuadrado dividido en tres. En estas divisiones se escribe el número de la macroetapa, y el número de la etapa de salida y entrada acompañado de una "M". Así, un Grafcet puede contener varias macroetapas.

Una macroetapa está regida por las siguientes reglas:

Regla 1. Una macroetapa tiene una sola etapa de entrada, denominada **I**, y una sola etapa de salida, denominada **O**.

Regla 2. Todos los disparos de la transición localizada antes de la macroetapa deben activar la etapa de entrada y su expansión posterior.

Regla 3. La etapa de salida debe habilitar la transición localizada debajo de la macroetapa, en concordancia con la estructura del diagrama Grafcet.

Regla 4. No existe ninguna unión directa hacia fuera de la macroetapa.

En la Figura 1.27, por ejemplo, se muestra un diagrama Grafcet con una macroetapa así como su diagrama correspondiente. El código 5/M30, significa que es la macroetapa 5, y que la etapa de entrada y salida es la número 30.

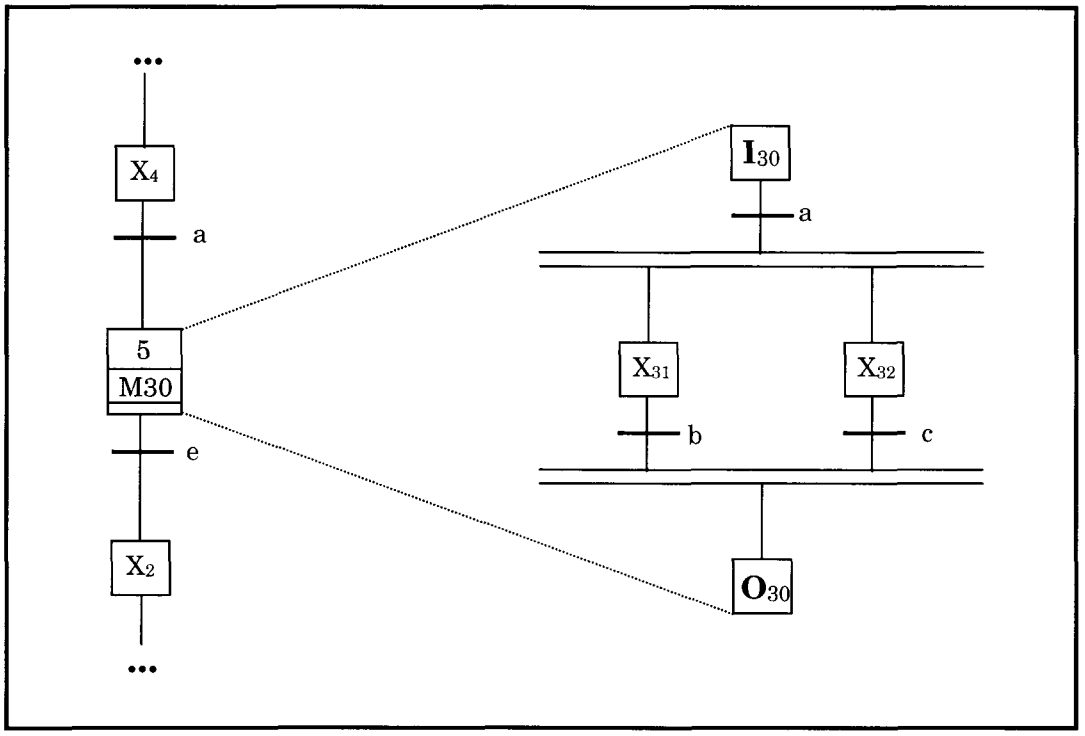


Figura 1.27. Ejemplo de una macroetapa en Grafcet.

1.4.3.8.2 Macroacciones.

Cuando se modelan sistemas complejos, el tamaño de los diagramas Grafcet pueden llegar a ser muy grandes tal que resulten difíciles de entender, corregir, actualizar o analizar. Una macroacción es aquella que se realiza en una estructura jerárquica, es decir cuando un controlador lógico ($\mathbf{G_1}$) tiene una influencia global sobre otro ($\mathbf{G_2}$). Una macroacción puede ser de nivel o de impulso y es producida por $\mathbf{G_1}$ para controlar el comportamiento de $\mathbf{G_2}$. En la Figura 1.28, por ejemplo, el diagrama Grafcet G_1 tiene control para "congelar" la ejecución del diagrama Grafcet G_2, cuando G_1 se encuentre en la etapa X_6, para lograr esto todas las transiciones de G_2 dependen del estado de la etapa X_6 de G_1.

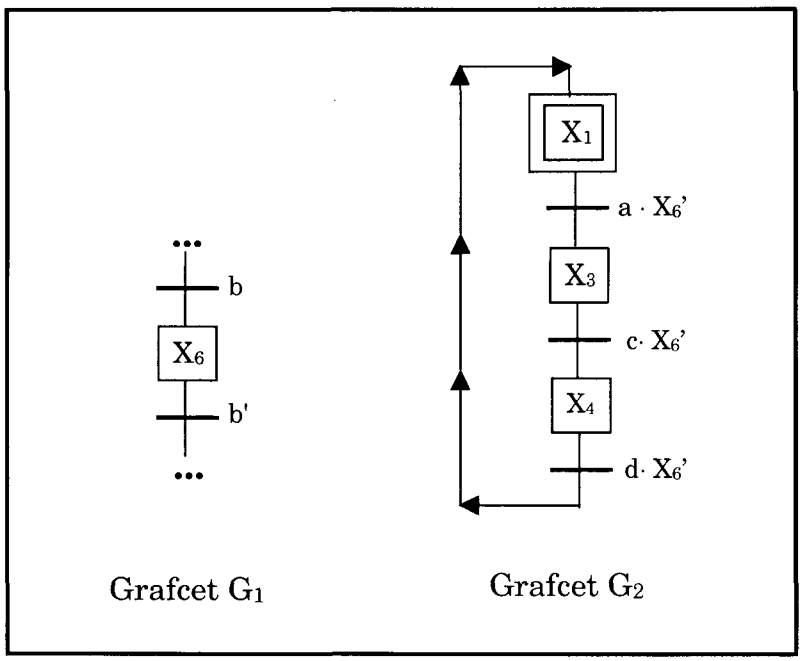


Figura 1.28. Ejemplo de una macroacción del Grafcet G_1 hacia el Grafcet G_2.

1.4.3.9 Secciones de GRAFCET.

En el lenguaje de programación GRAFCET existen tres etapas en las cuales se definen los bits de control, el diagrama Grafcet y las salidas para el SED. En esta sección se explicará la función de cada una de estas etapas.

1.4.3.9.1 Sección preliminar.

En esta sección se definen las condiciones para los bits de control del programa, los cuales son:

- Bit de arranque.
- Bit de paro.
- Bit de pausa.

Además es posible agregar en las condiciones de los bits anteriores, dos bits propios de GRAFCET:

- *Bit de arranque en frío.* Este bit se activa después de un arranque en el que el controlador ha perdido los datos de memoria.
- *Bit de arranque en caliente.* Este bit se activa después de un arranque en que el controlador **no** ha perdido los datos de memoria.

1.4.3.9.2 Sección Grafcet.

En esta sección se definen las etapas iniciales, las etapas intermedias y las transiciones entre cada una de éstas. Las etapas iniciales son aquellas que el controlador activará cuando se inicialice el Grafcet (una transición positiva en el bit de arranque de la sección preliminar) y las etapas intermedias son aquellas que preceden a la etapa inicial. Es posible que en la sección Grafcet existan varias etapas iniciales, entonces el controlador iniciará cada una de ellas por separado cuando se inicialice el Grafcet.

1.4.3.9.3 Sección posterior.

En esta sección se relacionan las salidas de los actuadores con cada una de las etapas de la sección Grafcet y además se pueden manejar todo tipo de herramientas lógicas, como lo son temporizadores, contadores, memorias, entre otras.

La relación de las salidas con las etapas, se logra construyendo un diagrama lógico en el que se describan las ecuaciones booleanas correspondientes.

Capítulo 2

Metodología IPTG en GRAFCET

En este capítulo se presenta y se propone una modificación a la metodología IPTL originalmente propuesta por Jin-Shyan Lee y Pau-Lo Hsu [14]. La modificación consiste básicamente en utilizar el lenguaje de programación de controladores lógicos programables denominado GRAFCET, en lugar de los diagramas lógicos de escalera.

2.1 Introducción.

La metodología IPTL (**I**DEF0, **S**PNC, **T**PL, **L**LD)[†] fue originalmente propuesta por Jin-Shyan Lee y Pau-Lo Hsu como una herramienta para representar un proceso secuencial usando la técnica de modelación de "Integration Definition Language 0" (IDEF0). Esta técnica facilita su representación bajo la forma de un "Simplified Petri Net Controller" (SPNC) mediante el cual las señales de control son las encargadas de disparar cada una de las transiciones. Una vez que se tiene una representación bajo la forma de SPNC el siguiente paso es transformar la SPNC a un modelo "Token Passing Logic" (TPL) en el que se cambian las plazas de la SPNC por bits de memoria. Como parte del trabajo de investigación realizado en esta tesis, en la fase final de implementación se propone la transformación del modelo TPL al lenguaje de programación GRAFCET[‡] para los controladores lógicos programables (PLC).

Debido a que existen cambios en la estructura de lo que se propone en [14], se decidió cambiar el nombre original de la metodología IPTL, al de IPTG en GRAFCET(**I**DEF0, **S**PNC, **T**PL, **G**RAFCET), tal que ésta última se refiere a la metodología con la inclusión de la estructura de implementación **G**RAFCET en la última etapa.

En la Figura 2.1 se muestra un diagrama de flujo de las etapas para la metodología IPTG en GRAFCET.

[†] IDEF0 (Integration Definition Language 0), SPNC (Simplified Petri Net Controller), TPL (Token Passing Logic) y LLD (Logic Ladder Diagram)

[‡] Se manejará GRAFCET (con mayúsculas) como el lenguaje de programación usado en algunos controladores comerciales, y Grafcet (en minúsculas) como la herramienta de modelación de eventos discretos.

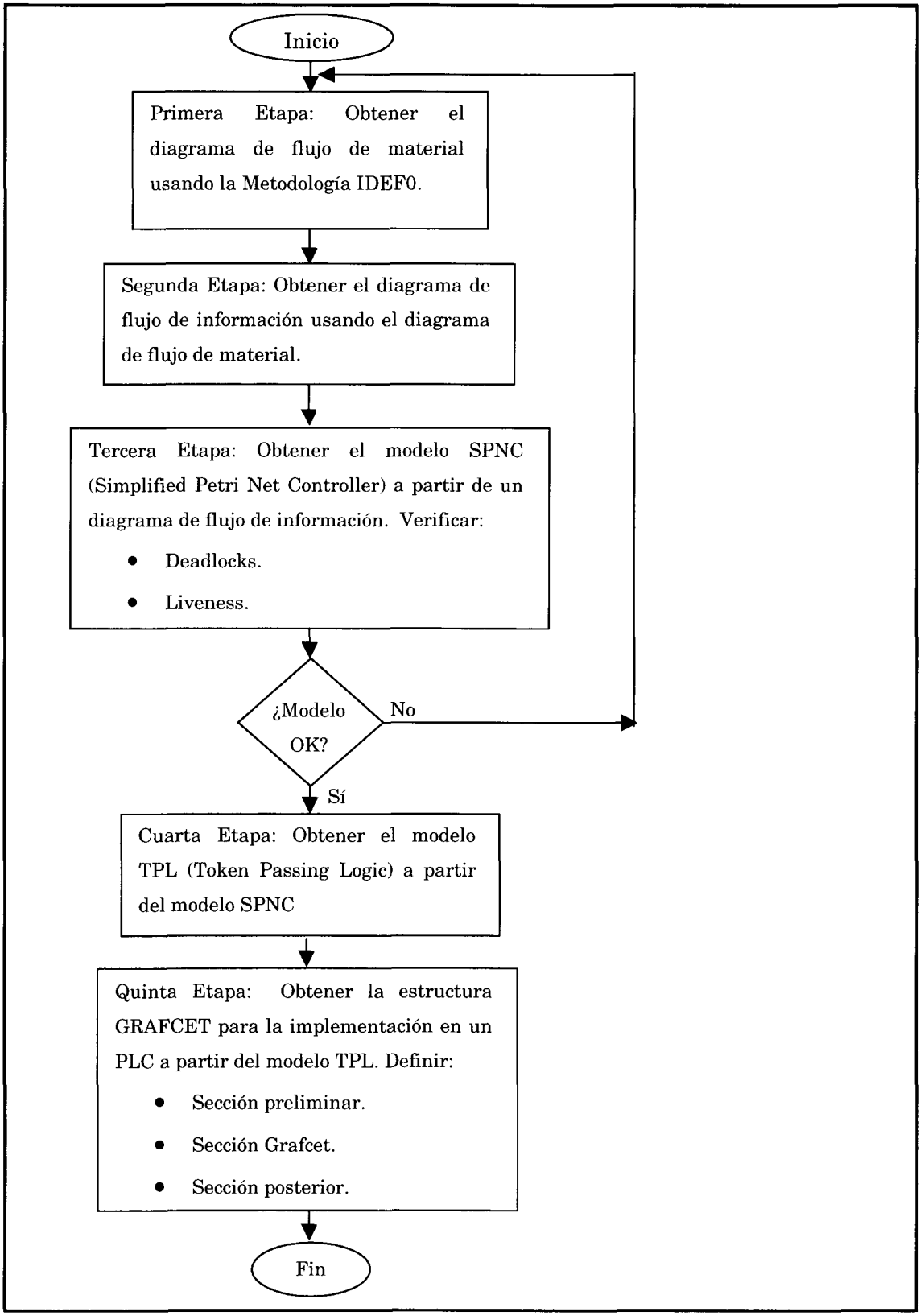


Figura 2.1. Diagrama de las etapas para la metodología IPTG en GRAFCET.

2.2 Desarrollo de la metodología IPTG en GRAFCET.

2.2.1 Descripción del proceso de estampado de latas que se utilizará en el desarrollo de la metodología IPTG en GRAFCET.

En la Figura 2.2 se muestra un proceso de estampado de latas el cual es utilizado en esta sección como ejemplo para describir la metodología. La composición y dinámica del proceso es la siguiente:

- Un motor mueve una banda, la cual transporta las latas a estampar hasta las dos estaciones de impresión.
- Existen cuatro botones: START, STOP, PAUSA y R (Reestablecimiento después de una pausa)
- La primera estación de impresión tiene un tope parador que detiene las latas de entrada, y además un magneto que sostiene la lata mientras se imprime. En esta estación existe un sensor de presencia de lata (S1) Al terminar la impresión, la máquina impresora manda una señal de "fin de impresión".
- Después del fin de la impresión se desactiva el magneto y la lata sale de la estación. Es hasta ese momento cuando el tope parador deja pasar la siguiente lata a ser etiquetada.
- Al terminar en la estación 1, la lata pasa a una segunda estación de impresión, donde al igual que la estación 1, se tiene un parador, un magneto y un sensor (S2). La impresora también manda una señal de "fin de impresión".
- Al salir de la estación 2 la lata es empacada manualmente.
- Las señales de control para los topes paradores son las siguientes:

    - R1 y R2, representan, respectivamente, la retracción del tope parador 1 y 2.

- A1 y A2, representan, respectivamente, la extensión del tope parador 1 y 2.

- Las señales de control para los magnetos 1 y 2 están representadas por M1 y M2, respectivamente.
- Las señales de control para las impresoras 1 y 2 están representadas por IMP1 y IMP2, respectivamente.

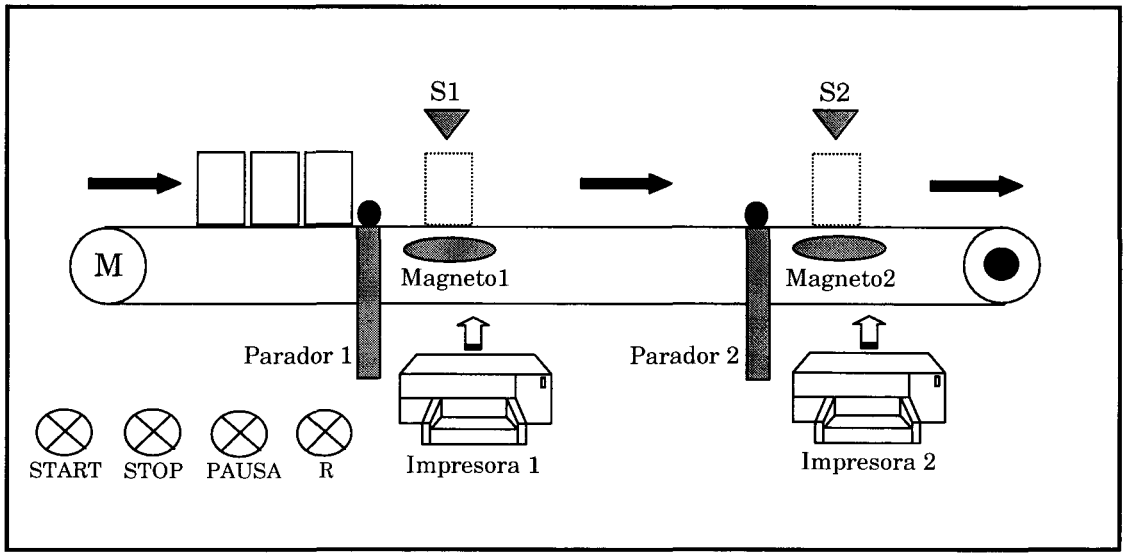


Figura 2.2. Proceso de estampado de latas.

Las condiciones iniciales establecidas son las siguientes:

- Las dos estaciones están vacías.
- El motor está apagado.
- Los paradores están extendidos (R1 = R2 = 0 y A1 = A2 = 1)
- Los magnetos están desactivados (M1 = M2 = 0)
- Las impresoras se encuentran en el estado de "no imprimir" (IMP1 = IMP2 = 0)
- La secuencia inicia con el botón START.

2.2.2 Primera Etapa: Obtener el diagrama de flujo de material usando la Metodología IDEF0 (Integration Definition Language 0) [11] [14]

La metodología IDEF0 puede ser usada para realizar un análisis funcional del sistema, así como para mostrar los diferentes mecanismos que lo componen. El método de modelación tiene una tendencia de lo general a lo particular, es decir, primero se empieza con el sistema en general, y después se descompone en cada una de sus partes aplicando el mismo método. Así, se logra obtener un diagrama más específico de lo que se desea modelar, cuyo resultado es un modelo a detalle del sistema actual.

En la Figura 2.3 se muestran los componentes de un diagrama IDEF0.

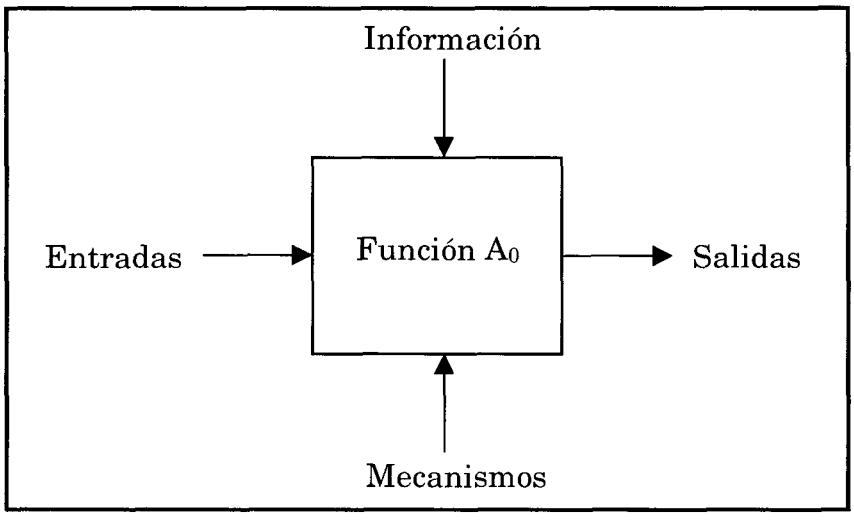


Figura 2.3. Diagrama general IDEF0.

Los componentes del modelo principal IDEF0 son diagramas, donde todas las funciones y conexiones del sistema son representadas como bloques y arcos respectivamente.

Como se puede observar en la Figura 2.3 los *datos de manejo* de la producción entran por arriba del bloque (manejo). Los *materiales y la información* necesaria para llevar a cabo la función, entran por el lado izquierdo (entradas). Los *resultados* de la ejecución de la función son representados por la flecha de salida en la derecha del bloque (salidas). Los *mecanismos* necesarios para realizar la función, como máquinas, computadoras, entre otras, son representados por la flecha que entra por abajo del bloque (mecanismos).

La primera etapa de la metodología IPTG se utiliza para comprender las funciones y operaciones del sistema [14], para lo cual se describe y especifica cada función dentro de éste usando la metodología IDEF0. Es necesario llevar a cabo un proceso de descomposición tantas veces como sea necesario, hasta lograr que cada función no represente varias funciones del sistema y conseguir que las secuencias del proceso queden totalmente definidas en el diagrama. Así, en esta etapa se obtiene un diagrama del flujo de material del sistema.

Para el proceso de estampado de latas, el diagrama de flujo de material se muestra en la Figura 2.4. Este diagrama indica la manera en que fluye el material en la línea de producción, por ejemplo, después de que el parador 1 y el magneto 1 se desactivan, las latas pasan a la estación 2, como se muestra en la Figura 2.4. En cada una de las funciones se pueden apreciar las entradas, el manejo de producción, los mecanismos y las salidas. En el caso del "Estampado 1", la entrada es una lata, el manejo es el número de etiqueta de la lata, el mecanismo es la impresora y la salida es la misma lata.

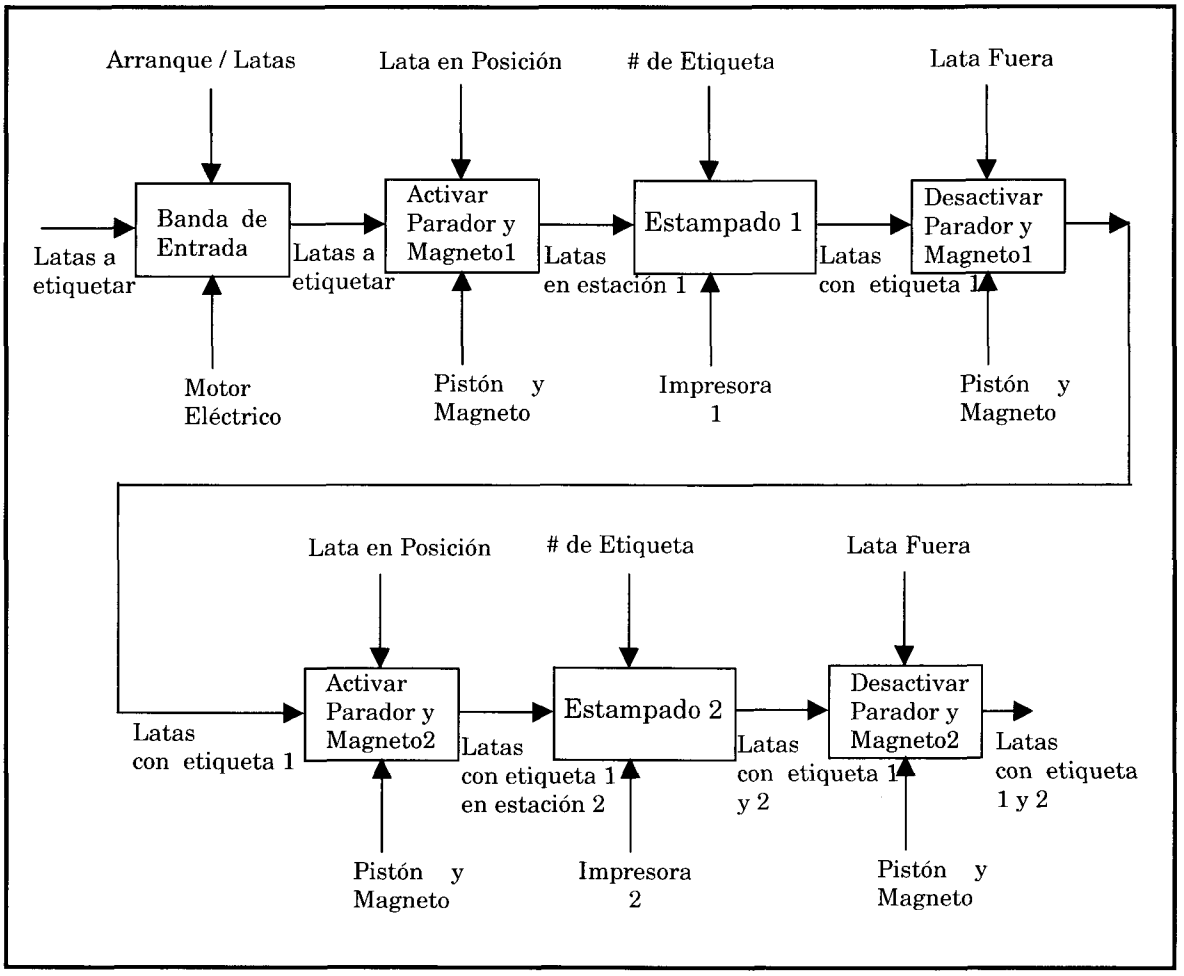


Figura 2.4. Diagrama de flujo de material para el proceso de estampado de latas.

2.2.3 Segunda Etapa: Obtener el diagrama de flujo de información usando el diagrama de flujo de material [14]

En la segunda etapa de la metodología, se utiliza el flujo de información para controlar el flujo de material del sistema de manufactura, es decir, se construirá un diagrama de flujo de información basado en el diagrama de flujo de material de la primer etapa. A partir de este momento a la función se le denominará actividad.

En el diagrama de flujo de información el comando de entrada habilitará la actividad y los mecanismos (arcos entrando por debajo de la actividad) son eliminados ya que no son relevantes para el control y simplifican la representación.

En el diagrama de flujo de información se deben incluir las lecturas de los sensores que son dibujados como flechas que entran por arriba de la actividad. Al final de esta etapa se debe obtener un modelo del sistema con las señales para el control de cada actividad, o bien, un diagrama de flujo de información.

Para el proceso de estampado de latas, el diagrama de flujo de información se muestra en la Figura 2.5. En este diagrama se incluyen las señales de los sensores para cada etapa, las cuales provocarán que el sistema cambie de estado. En esta etapa los mecanismos para realizar cada una de las actividades han sido eliminados del diagrama.

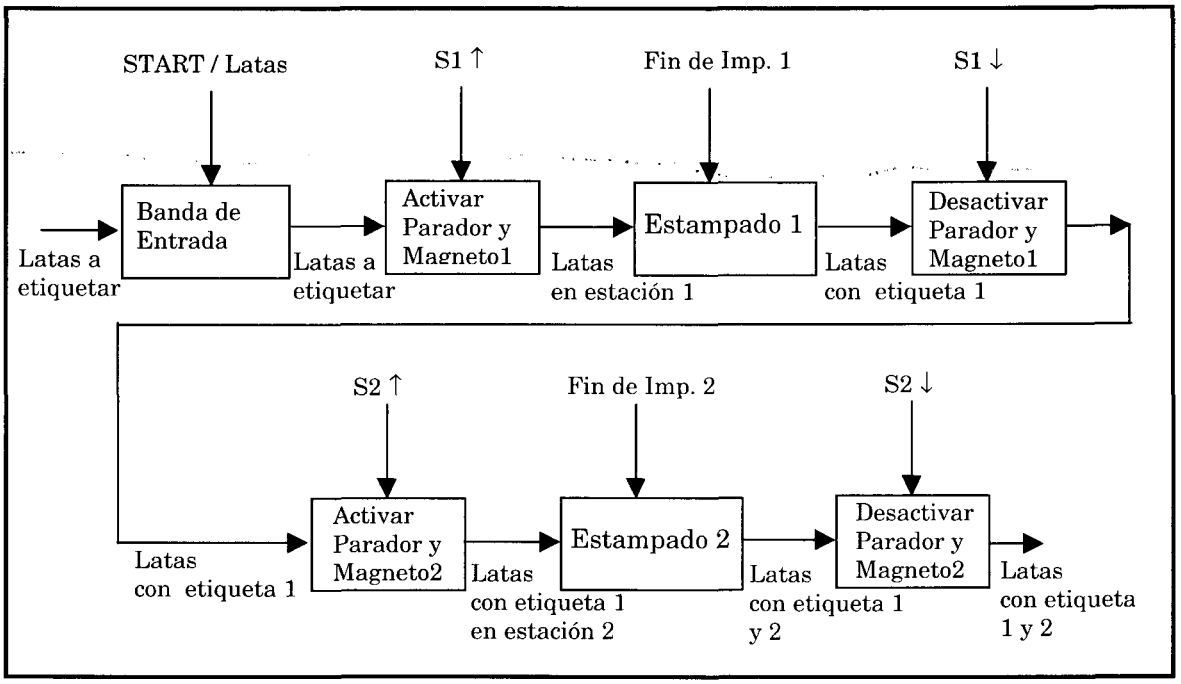


Figura 2.5. Diagrama de flujo de información para el proceso de estampado de latas.

2.2.4 Tercera Etapa: Obtener el modelo SPNC (Simplified Petri Net Controller) a partir de un diagrama de flujo de información [14]

El concepto del modelo SPNC, nace de la necesidad de simplificar la representación de las lecturas de los sensores en las Redes de Petri, es decir, en vez de utilizar varias plazas para la representación de un sensor, esta señal se modela directamente en la transición. Esto reduce considerablemente el número de plazas y transiciones, además de que permite visualizar de mejor manera las condiciones para cada disparo.

El diagrama de flujo de información sólo representa las actividades en el sistema y las interrelaciones que existen entre éstas, y no muestra una lógica directa ni una dependencia dinámica entre las diferentes actividades. Estas carencias serán cubiertas por el modelo SPNC. En esta tercera etapa de la metodología IPTG es necesario convertir el diagrama de flujo de información en un modelo SPNC.

Una SPNC se define como un quíntuple (P, T, A, S, M_0), donde:

- $P = \{P_1, P_2, ..., P_m\}$ es un conjunto finito de plazas.
- $T = \{T_1, T_2, ..., T_n\}$ es un conjunto finito de transiciones donde se debe cumplir que $P \cup T \neq \varnothing$ y $P \cap T = \varnothing$.
- $A \subseteq (P \times T) \cup (T \times P)$ es el conjunto de arcos entre las plazas y las transiciones.
- $S = \{S_1, S_2, ..., S_n\}$ es el conjunto de estados de los sensores.
- $M_0 : P \rightarrow 1$ es el marcado inicial.

En la Figura 2.6 se muestran los cinco componentes de un modelo SPNC, los cuales son:

- La *plaza,* la cual se dibuja como un círculo.

- La *transición,* que se dibuja como una barra.
- El *estado del sensor*, el cual es dibujado como un círculo pequeño con una flecha punteada.
- Los *arcos,* que son representados por flechas dirigidas desde una plaza a una transición o viceversa.
- Una *token* el cual es representado como un círculo relleno color negro.

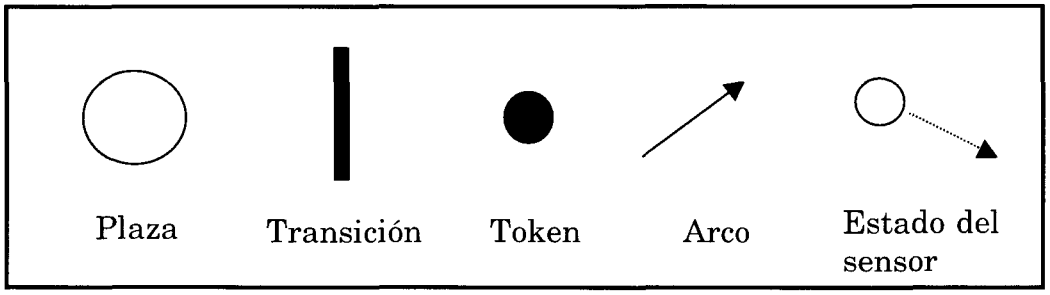


Figura 2.6. Elementos de una SPNC.

En el proceso de modelación, las *plazas* representan el estado del sistema y las transiciones representan los eventos. Una *transición* tiene un conjunto de plazas de entrada y un conjunto de plazas de salida, las cuales representan, respectivamente, condiciones antes y después del evento. El *estado del sensor* representa la condición de disparo en una transición. Cabe mencionar que el estado del sensor es una variable Booleana el cual será inactivo cuando su valor sea "0" o realice una transición negativa y activo cuando sea "1" o realice una transición positiva.

El *marcaje* de la SPNC se refiere al número de tokens en cada plaza. La presencia de un *token* en una plaza, significa que la condición asociada es verdadera y que las acciones relacionadas en esa plaza deberán ser ejecutadas.

Cabe mencionar que en el modelo SPNC todos los arcos tendrán un peso igual a 1, es decir, que cuando una transición se dispare, siempre se pasará un token a las plazas de salida de ésta.

Las reglas para habilitar y disparar una transición son:

- *Regla para habilitar una transición.* Una transición T es habilitada si cada plaza de entrada P de T contiene al menos un token.
- *Regla para disparar una transición.* En el modelo SPNC, una transición habilitada T es disparada dependiendo del estado de los sensores relacionados con ésta. Si todos los sensores de la transición T tienen un valor igual a 1, entonces ésta será disparada. Cuando la transición T es disparada es necesario remover un token de cada plaza de entrada y depositar uno en cada plaza de salida.

La transformación de un diagrama de flujo de información a un modelo SPNC se realiza aplicando las siguientes cuatro reglas:

Regla 1. Cada bloque de las actividades en el diagrama de flujo de información es transformado en una transición del modelo SPNC.

Regla 2. Los comandos de entrada y salida son transformados, respectivamente, en plazas de entrada y salida a una transición.

Regla 3. Las señales de control o lecturas de los sensores, son transformadas en estados de los sensores.

Regla 4. El marcaje inicial del modelo SPNC es definido por las condiciones iniciales del sistema.

En la Figura 2.7 se muestra un ejemplo gráfico de la transformación de un diagrama de flujo de información a un modelo SPNC.

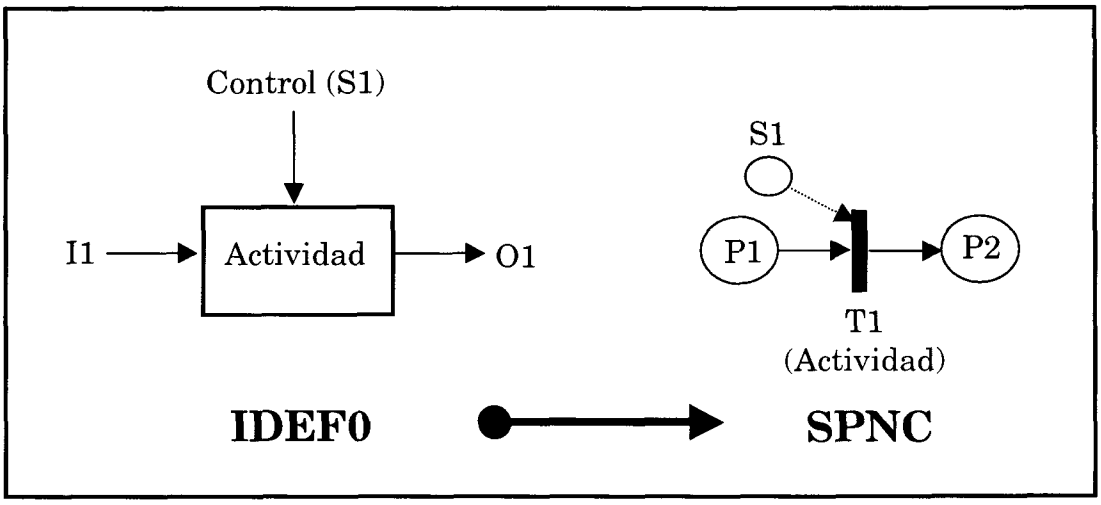


Figura 2.7. Transformación de un diagrama de flujo de información a una SPNC.

En esta etapa de la metodología se deben analizar las propiedades del modelo SPNC tales como seguridad y conservabilidad entre otras, utilizando simulación por computadora, por ejemplo, en el software Visual Object Net++ [10]. Así, el modelo SPNC obtenido podrá ser validado desde un punto de vista de comportamiento dinámico.

Basándose en el diagrama de flujo de información del proceso de estampado de latas, se obtiene el modelo SPNC presentado en la Figura 2.8, en éste las señales de los sensores se representan como entradas en cada una de las transiciones. A diferencia del diagrama de flujo de información el modelo SPNC de la Figura 2.8, muestra más claramente la dinámica del proceso de estampado de latas.

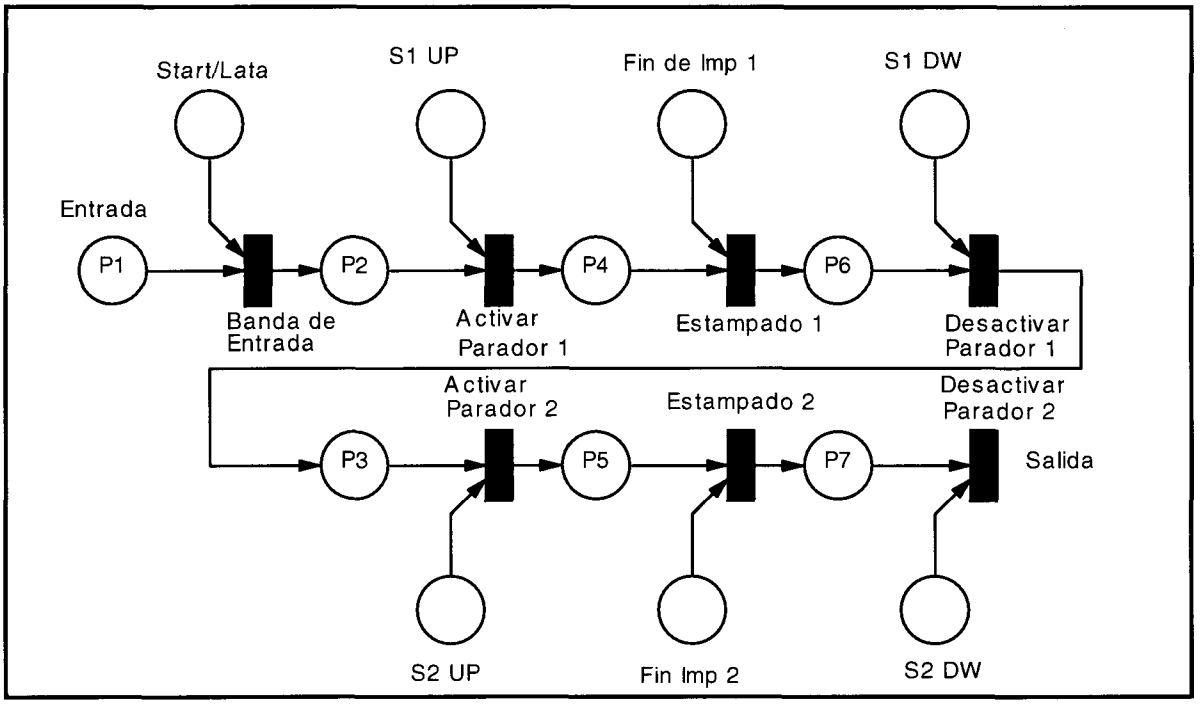


Figura 2.8. Modelo SPNC para el proceso de estampado de latas.

Usando el modelo de la Figura 2.8 se debe obtener una simulación de la red para verificar sus propiedades y su comportamiento dinámico. Para llevar a cabo la simulación del modelo SPNC es necesario realizar algunas modificaciones e insertar los tokens iniciales en éste, así en la Figura 2.9 se muestra la SPNC en tiempo de simulación igual a cero con las condiciones iniciales, las cuales están dadas por:

- 500 tokens (latas) en la estación de entrada.
- 500 tokens (señales de control) en Fin de impresión 1 y 2 y S1 DW y S2 DW (ya que por ahí pasarán 500 latas)
- Un token (señal de control) en S1 UP y S2 UP.

En la Figura 2.10 se muestra la SPNC después de terminada la simulación, y se puede concluir que el comportamiento dinámico del modelo SPNC obtenido

es aceptable ya que las 500 latas son transportadas a la estación de salida en un tiempo finito.

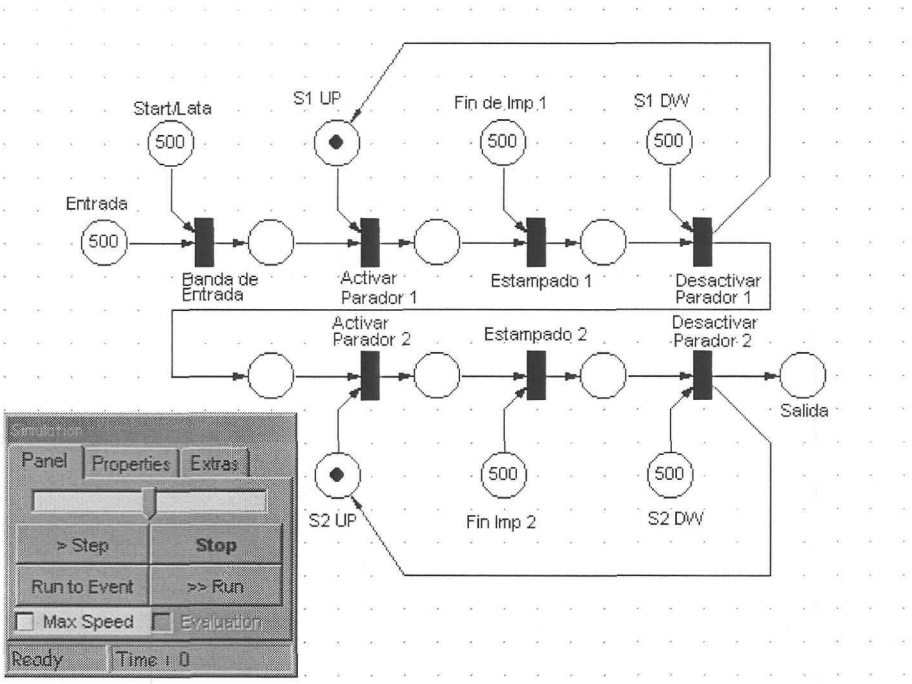


Figura 2.9. Simulación de la SPNC en tiempo de simulación igual a cero.

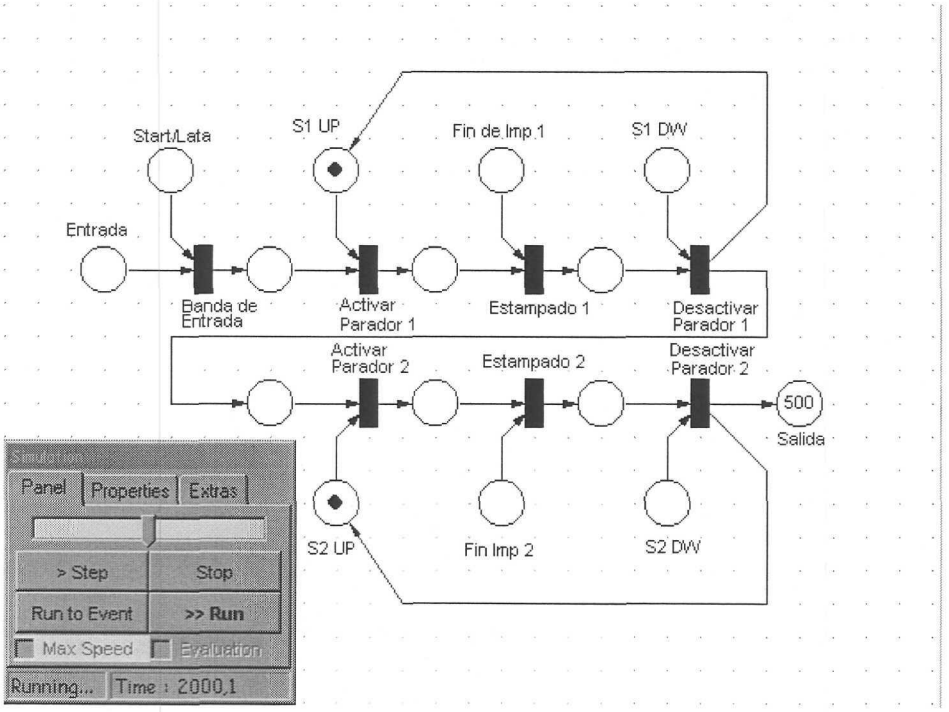


Figura 2.10. Simulación de la SPNC al final del tiempo de simulación.

2.2.5 Cuarta Etapa: Obtener el modelo TPL (Token Passing Logic) a partir del modelo SPNC [14]

Para simplificar la conversión del modelo SPNC al lenguaje GRAFCET, el modelo TPL resulta de mucha utilidad. La ventaja principal de este modelo es que facilita la conversión directa de un modelo SPNC a una forma genérica de control lógico, la cual puede ser implementada en Lógica de Diagrama Escalera (LLD por sus siglas en inglés) o bien en GRAFCET.

El modelo SPNC se puede transformar en un modelo TPL, donde se reasignarán las lecturas de los sensores por sensores de entrada y se relacionarán las salidas con cada etapa del proceso.

El modelo TPL es un séxtuple (M, T, A, in, out, $time$) donde:

- $M = \{M_1, M_2, ..., M_m\}$ es un conjunto finito de bits de memoria.
- $T = \{T_1, T_2, ..., T_n\}$ es un conjunto finito de transiciones.
- $A \subseteq (M \times T) \cup (T \times M)$ es el conjunto de arcos entre las memorias y las transiciones.
- $in = \{in_1, in_2, ..., in_n\}$ es el conjunto de sensores de entrada.
- $out = \{out_1, out_2, ..., out_m\}$ es el conjunto de salidas para los actuadores.
- $time = \{time_1, time_2, ..., time_m\}$ es el conjunto de temporizadores de retraso.

La transformación del modelo SPNC al TPL, está basada en cinco reglas según la referencia [14], pero debido al uso del lenguaje de programación GRAFCET, en este trabajo de investigación se agregó otra regla (Regla 6) la cual simplifica la visión de la dinámica del proceso. Así, las seis reglas para llevar a cabo la transformación del modelo SPNC al TPL son las siguientes:

Regla 1. La transición del modelo SPNC es transformada en una transición del modelo TPL.

Regla 2. Una plaza P_x es transformada en un bit de memoria M_x.

Regla 3. El estado del sensor es transformado en un sensor de entrada.

Regla 4. Para una acción relacionada con una plaza P_x, se debe relacionar una salida con el bit de memoria M_x.

Regla 5. Para los retrasos relacionados con una plaza P_x, se debe relacionar un temporizador de retraso con el bit de memoria M_x.

Regla 6. Separar el modelo TPL en celdas, es decir si existen partes del proceso independientes entonces éstas deberán ser separadas ya que esto simplificará su representación en la quinta etapa. En el modelo SPNC todavía se respeta el flujo de información de la segunda etapa, pero para la implementación esto no resulta muy favorable, por lo que es necesario separar el modelo TPL por celdas independientes. La separación debe realizarse de tal manera que se encuentre un punto de inicio, fin y retroalimentación (en caso de que existiera) para las celdas independientes[§].

En la Figura 2.11 se muestra un ejemplo de cómo se aplican estas reglas en la transformación de un modelo SPNC a un modelo TPL. Se asume que el temporizador 1 (*time 1*) se relaciona con la plaza P1, es decir, se maneja como una plaza P-Timed. Así, se relaciona el temporizador 1 con la plaza y el bit de memoria correspondientes. En esta figura la plaza P2 y P3 son independientes y tienen en común la transición T1, entonces aplicando la Regla 6 pueden ser separadas en el modelo TPL.

[§] Regla agregada por Agustín Pando Delgado, como una mejora al algoritmo original [14] .

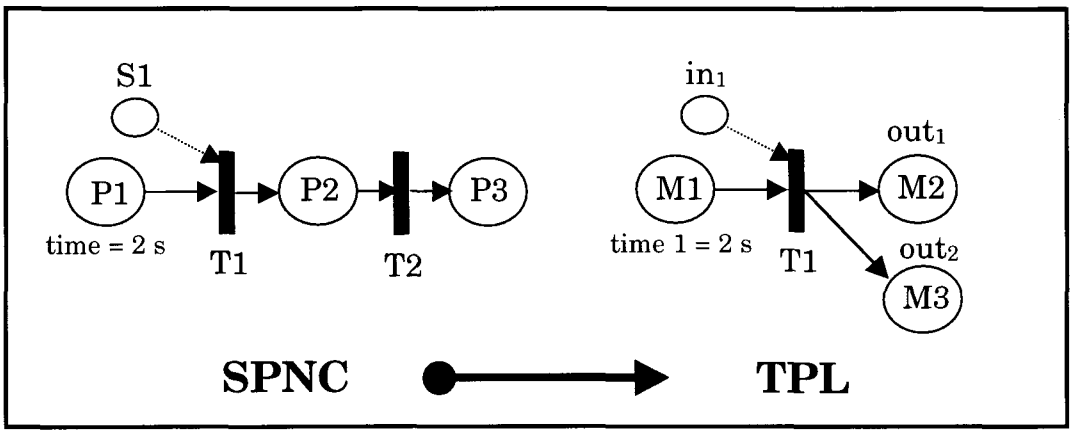


Figura 2.11. Transformación de un modelo en SPNC a un modelo TPL.

La inclusión de temporizadores es sencilla ya que sólo se asocian con la salida, con la entrada o con el bit de memoria que se desee retrasar por un tiempo determinado. En el caso del ejemplo de la Figura 2.11 el temporizador está relacionado con la plaza de entrada P1.

Para el modelo SPNC del proceso de estampado de latas, se obtiene el modelo TPL de la Figura 2.12. En este modelo la aplicación de la *Regla 6* resulta de mucha utilidad, ya que este proceso puede ser dividido en dos celdas independientes, es decir, la estación 1 y la 2. El punto de inicio para las dos celdas se relaciona con el botón de entrada "START".

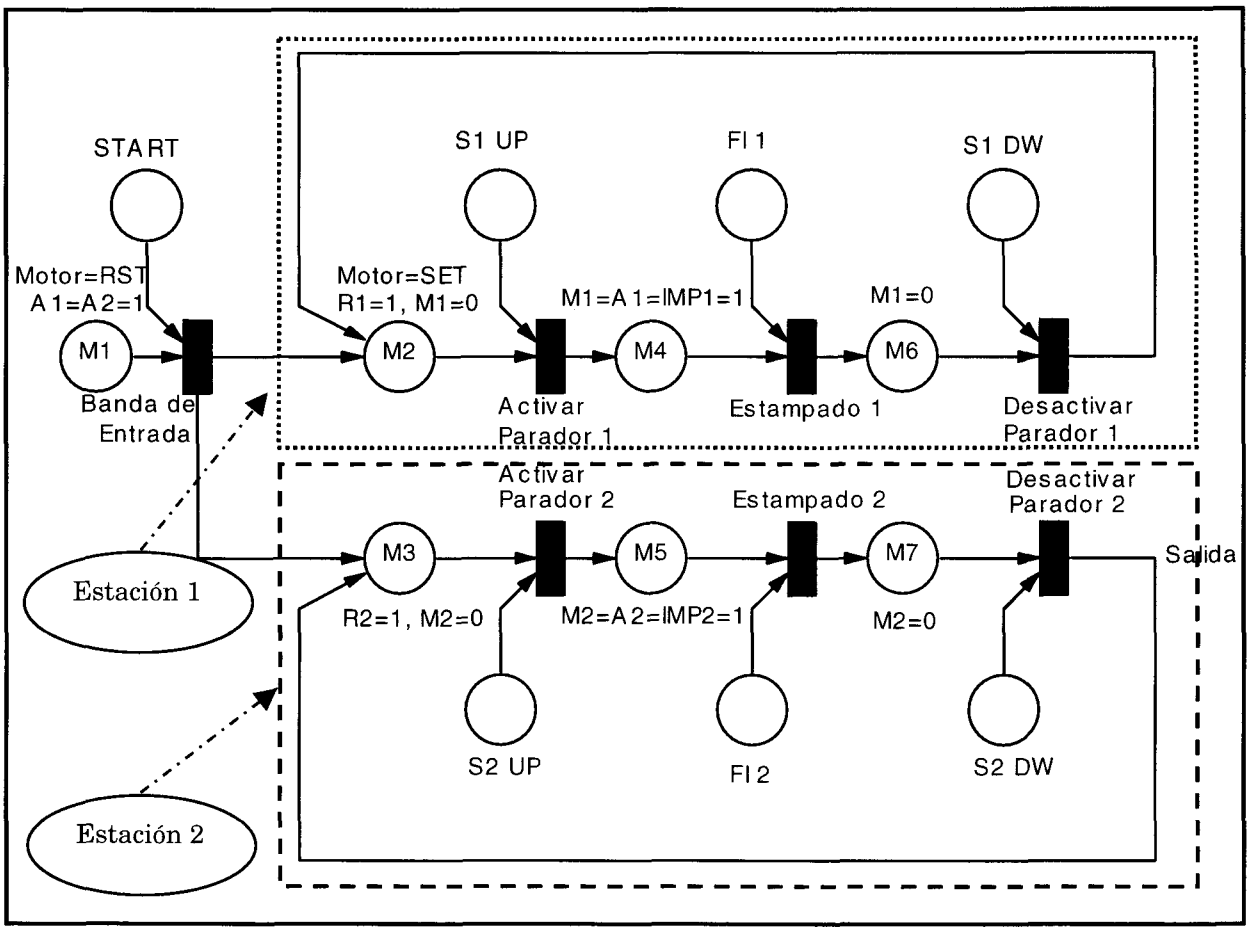


Figura 2.12. Modelo TPL para el proceso de estampado de latas.

En la Figura 2.13 se muestra el diagrama lógico de escalera que se obtendría si se aplicara la metodología IPTL de la referencia [14] al modelo TPL de la Figura 2.12. En este diagrama LLD se puede apreciar como difícilmente se puede conocer el estado actual del proceso, además que resulta muy complicado, mezclar los bits de estado con el manejo de las salidas. El diagrama está compuesto de once escalones cada uno con al menos dos subdivisiones.

En el diagrama de la Figura 2.13, no se contemplan las condiciones de pausa, así como de arranque en frío y en caliente. Aunque para introducir la función

de "congelamiento" del proceso, sería necesario agregar un bit de control en cada uno de los escalones del diagrama LLD, lo cual resultaría en una acción complicada de realizar en diagramas medianamente grandes, incluso en éste.

Además al aplicar la metodología propuesta en [14] no se tiene una referencia cruzada de los bits de memoria y las salidas del PLC, lo cual conlleva a que el estado actual y futuro del proceso no se obtenga fácilmente, y en consecuencia no se pueda hacer un análisis sencillo de la activación y desactivación de las salidas del proceso.

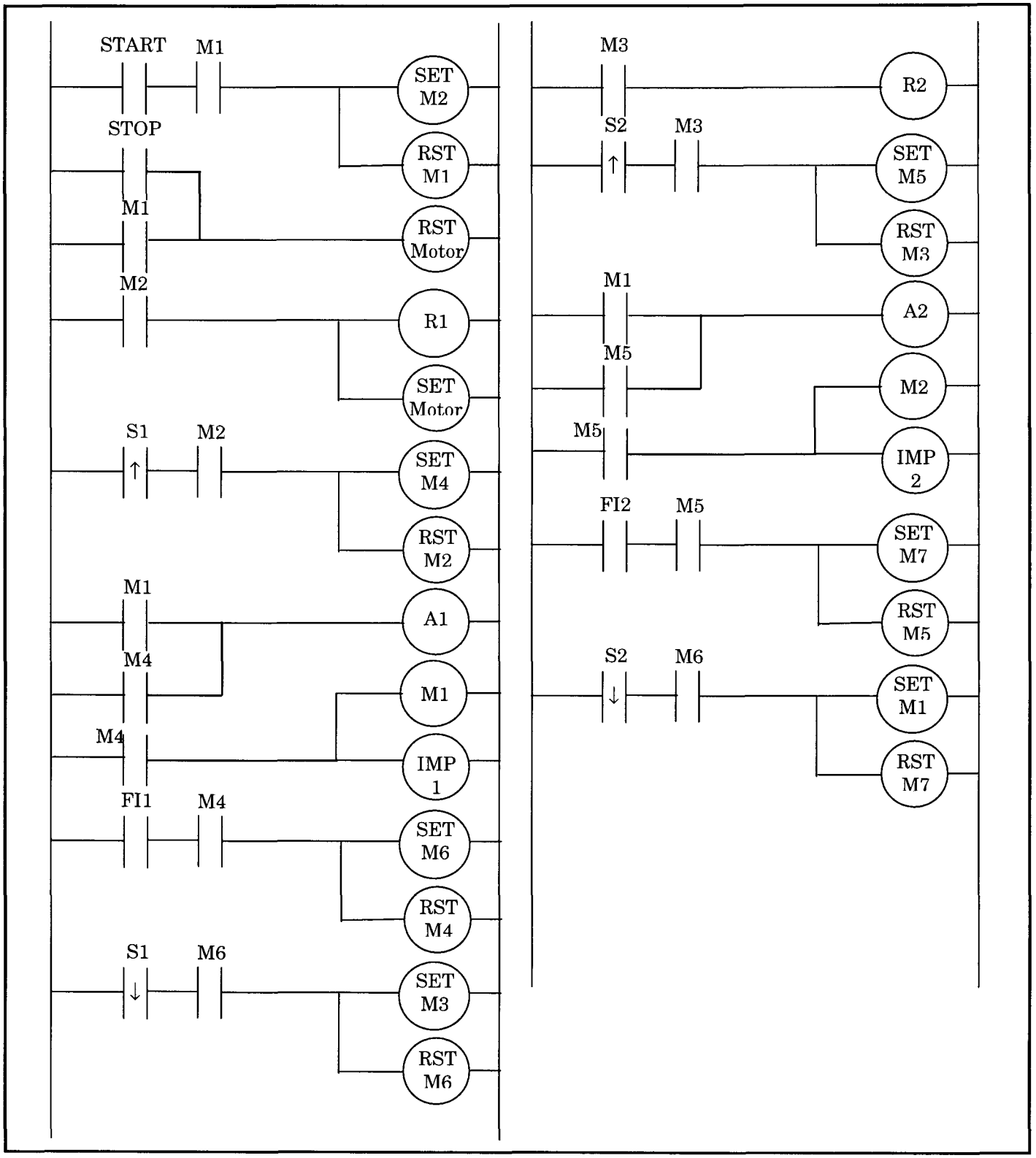


Figura 2.13. Diagrama escalera de la metodología IPTL propuesta en [14].

2.2.6 Quinta Etapa: Obtener la estructura GRAFCET para la implementación en un PLC a partir del modelo TPL.

Obtener un modelo que sea directamente implementable en un PLC es de gran importancia en las aplicaciones industriales. Actualmente existen diferentes lenguajes de programación para los PLCs, como son:

- Los diagramas lógicos de escalera (LLD)
- Los diagramas por bloques funcionales (and, or, not, timers, etc.)
- Los diagramas GRAFCET.
- Los diagramas SFC (Sequential Function Chart).

Los LLD son los más clásicos ya que nacieron de la primera representación para automatizar secuencias con relevadores. Aunque su uso es muy común, estos modelos resultan ser complicados cuando se desea automatizar procesos complejos, ya que es muy difícil dar seguimiento a la secuencia del controlador, es decir, no es fácil conocer el estado actual del proceso.

Por esta razón se decidió pasar del modelo TPL a un diagrama Grafcet, el cual tiene mucho parecido con los diagramas SFC [13] (aunque cabe mencionar que GRAFCET fue creado primero que SFC).

Algunas ventajas que presenta el uso del lenguaje de programación GRAFCET son las siguientes:

- Es posible congelar la ejecución del programa fácilmente, mediante la sección de control de GRAFCET llamada sección preliminar.
- Es posible tener control sobre lo que se desea realizar después de una falla en el suministro eléctrico, o bien cualquier falla en la que se pierda control sobre el proceso (arranque en frío o caliente).

- El monitoreo del estado del proceso es sencillo, gracias a los nuevos programas computacionales basados en la plataforma Windows ™.
- Las salidas están claramente ligadas a cada una de las etapas del modelo Grafcet, lo cual resulta de mucha utilidad en el análisis de fallas. Gracias a esto es posible reducir los tiempos de falla, debidos al análisis del proceso para encontrar las causas del problema.

En la quinta etapa se pasa del modelo TPL, a la estructura de implementación GRAFCET en donde se sugiere definir cada una de las siguientes tres secciones para obtener una mejor interacción con el programa de control:

- ***Sección Preliminar***[**]. Sección donde se programa la interacción del usuario con el programa (arranque, paro y pausa del programa).
- ***Sección Grafcet***. Sección donde se define la dinámica (secuencia) del proceso.
- ***Sección Posterior***. Sección donde se relacionan los estados de la dinámica del proceso con las salidas del PLC.

Cabe mencionar que los bits de memoria iniciales del proceso, definidos en el modelo TPL, ahora se convertirán en etapas iniciales de la sección Grafcet.

A diferencia de las etapas anteriores, en las que sólo se manejaron reglas de transformación, en esta quinta etapa se expondrán pasos y reglas para lograr la transformación al lenguaje de programación GRAFCET.

[**] Cabe mencionar que el funcionamiento del diagrama Grafcet no se ve afectado si no se incluye la sección preliminar, sólo se pierde la interacción con el programa GRAFCET.

2.2.6.1 Construcción de la sección preliminar, a partir de las condiciones del sistema y del diseñador.

Se sugiere definir los siguientes tres puntos, aunque como ya se mencionó se puede prescindir de éstos:

- Condición de inicialización (se refiere a que el programa se posicionará en la(s) etapa(s) inicial(es))y paro (se refiere a que la ejecución del programa se detendrá) de GRAFCET.
- La condición de pausa de GRAFCET en caso de existir. En este paso es necesario definir la condición de SET (encendido y sostenimiento de la pausa) y de RST (apagar la acción de pausa).
- Acción (inicialización, paro o pausa) que se deberá tomar en el re-arranque del programa con datos en memoria (arranque en caliente) o sin datos en memoria (arranque en frío)

2.2.6.1.1 ***Condición de Inicialización.*** Se construye un diagrama lógico de escalera con las condiciones de inicialización como contactos, las cuales habilitarán al bit de inicialización de GRAFCET. Cabe mencionar que este bit puede cambiar en los diferentes PLC's, por lo que sólo se describirá como el bit IG. En la Figura 2.14 S_1 ó S_2 representa la condición de inicialización de GRAFCET.

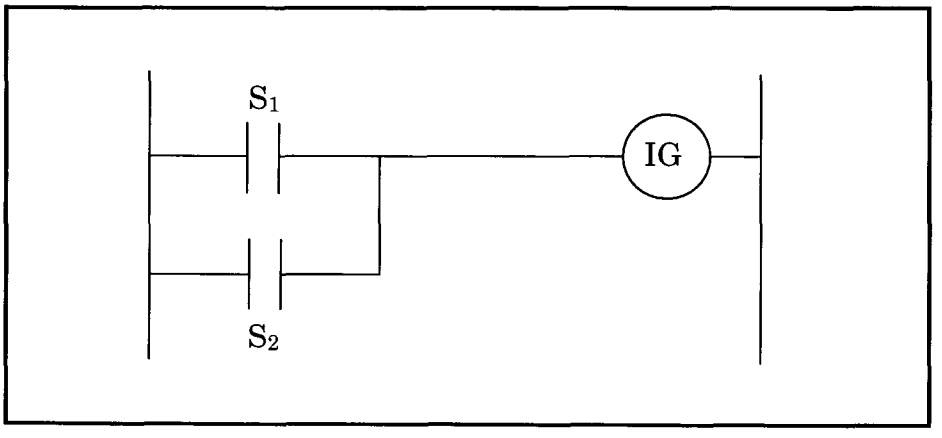


Figura 2.14. Ejemplo de construcción de la sección preliminar para el bit IG.

2.2.6.1.2 *Condición de Paro.* Añadir al diagrama anterior la condición de paro de GRAFCET, es decir agregar la condición con contactos, para habilitar el bit de paro. Al igual que en el punto anterior, este bit se representará como PG. En la Figura 2.15 S_3 y S_4, representa la condición de paro de GRAFCET.

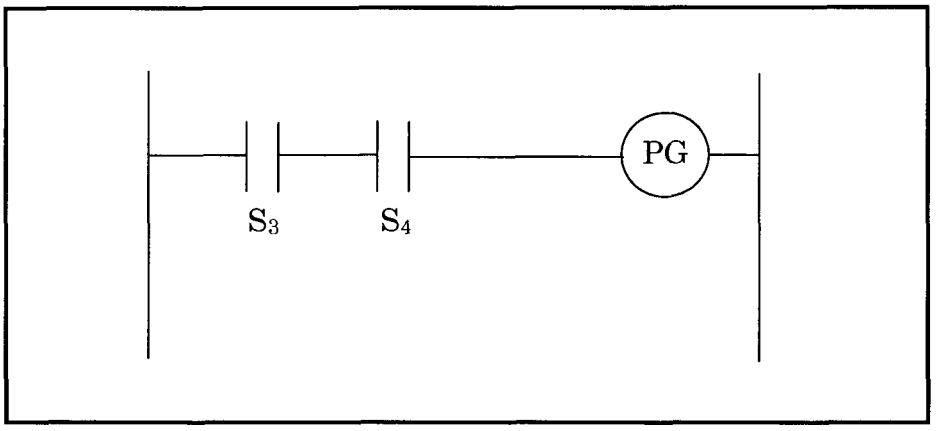


Figura 2.15. Ejemplo de construcción de la sección preliminar para el bit PG.

2.2.6.1.3 *Condición de Pausa.* Añadir al diagrama de la sección anterior, la condición que debe ocasionar que el GRAFCET entre en modo de pausa. Al igual que en las secciones anteriores este bit será representado como CG. Se debe agregar también la condición para apagar el bit CG. En la Figura 2.16 S_4 y S_5 son la condición de SET, mientras que S_7 es la condición de RST.

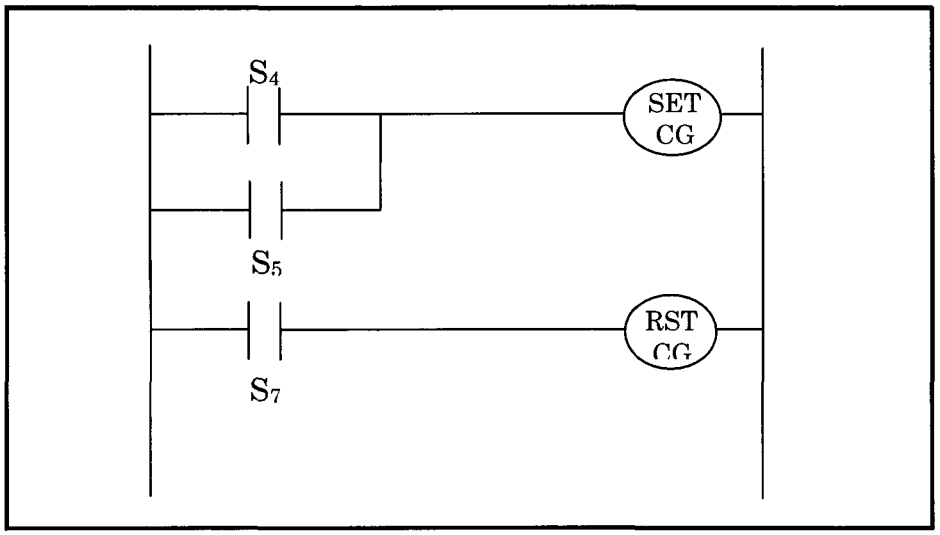


Figura 2.16. Ejemplo de construcción de la sección preliminar para el bit CG.

2.2.6.1.4 ***Condición de Rearranque.*** Añadir al diagrama anterior, la condición de arranque en frío y/o arranque en caliente, dependiendo de la acción que se desea para cada condición. Por ejemplo, si en el arranque en caliente se desea que el programa entre en el estado de pausa, entonces la condición de pausa para la Figura 2.16 será:

Condición de pausa = S_4 ó S_5 ó Arranque en caliente

Para el proceso de estampado de latas, es necesario establecer las condiciones de inicialización, paro, pausa, reestablecimiento de pausa, arranque en frío y arranque en caliente, las cuales son:

- La inicialización será llevada a cabo con los botones PAUSA y R presionados simultáneamente.
- El paro se activará con el botón STOP.
- El SET para la pausa será llevado a cabo con el botón PAUSA.
- El RST (o reestablecimiento) de la pausa será llevado a cabo con el botón R.

- El arranque en frío detendrá la ejecución del programa.
- El arranque en caliente provocará que el programa entre en el modo pausa.

Con las condiciones anteriores es posible realizar el diseño del diagrama escalera para la sección preliminar, el cual se muestra en la Figura 2.17. El arranque en frío se expresa como AF, y el arranque en caliente como AC.

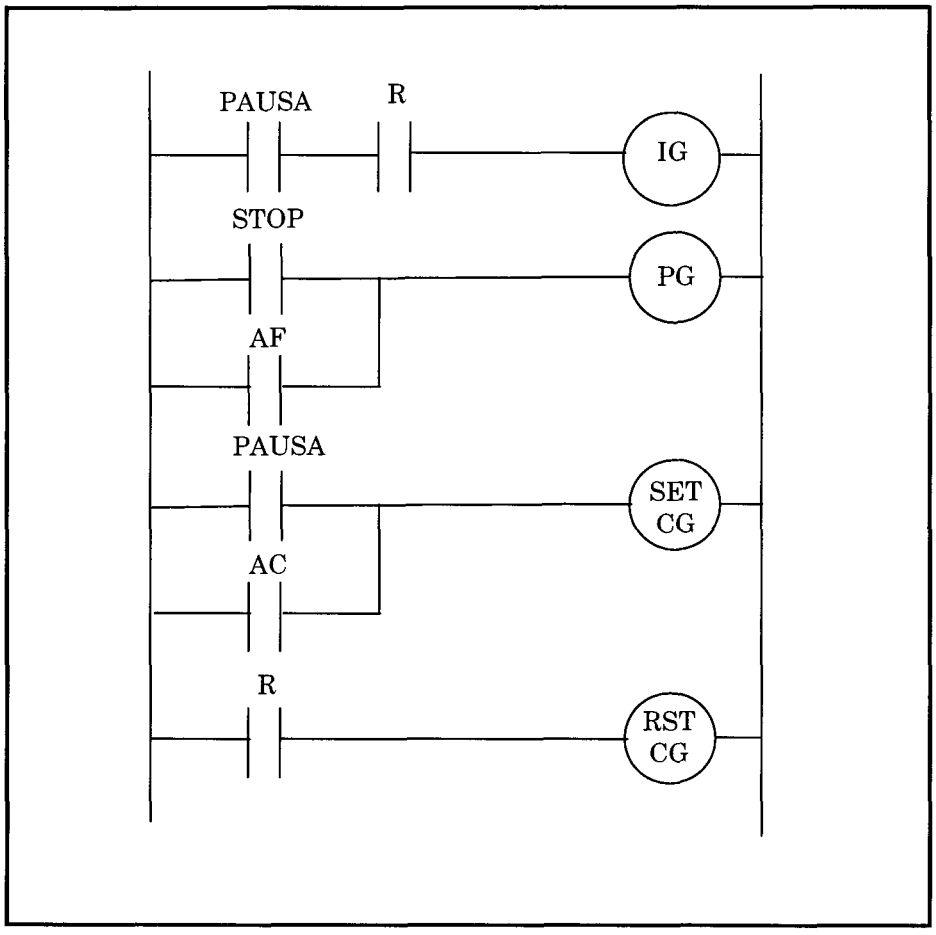


Figura 2.17. Diagrama escalera para la sección preliminar del proceso de estampado de latas.

2.2.6.2 Construcción de la sección Grafcet, a partir del modelo TPL.

La sección Grafcet permite modelar el comportamiento dinámico del proceso, es decir, en esta sección se le podrá dar seguimiento al estado actual y a la evolución provocada por los eventos discretos del proceso.

1) Se define cada bit de memoria (M) del modelo TPL como una etapa *X* del diagrama Grafcet.

2) Basándose en las condiciones iniciales definidas desde la primera etapa de la metodología, se definen los bits de memoria iniciales como etapas iniciales *X* del diagrama Grafcet. Como ya se había mencionado una de las ventajas de los diagramas Grafcet, es la versatilidad de poder ejecutar varios diagramas Grafcet a la vez, por lo que es posible declarar varias etapas iniciales. Por ejemplo, si los bits de memoria M_1, M_2 y M_3 son condiciones iniciales, entonces se deben iniciar tres diagramas Grafcet, como se muestra en el ejemplo de la Figura 2.18.

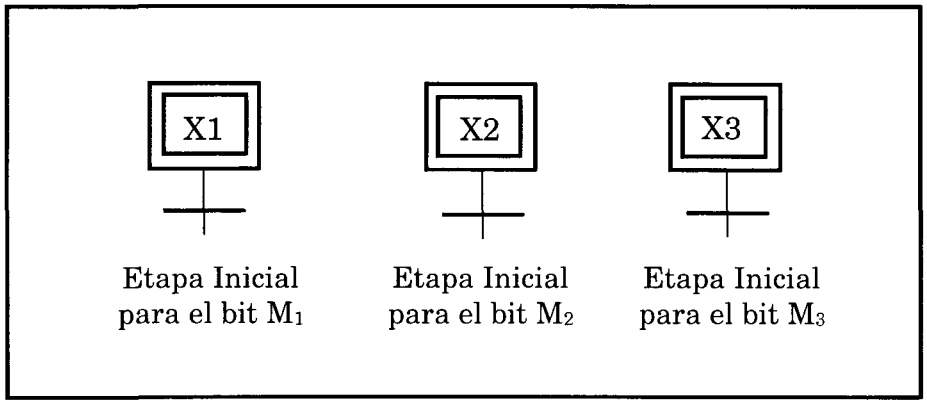


Figura 2.18. Ejemplo de construcción del diagrama Grafcet para tres estados iniciales.

3) Una vez que se han dibujado las condiciones iniciales dentro del diagrama Grafcet es necesario añadir las etapas posteriores, basándose en el modelo TPL. Dado que las etapas iniciales coinciden con los bits de

memoria iniciales del modelo TPL, se propones las siguientes reglas para realizar esta transformación:

Regla 1. Cada transición del modelo TPL es una transición en el diagrama Grafcet.

Regla 2. Cada bit de memoria *M* del modelo TPL es una etapa *X* del diagrama Grafcet.

Regla 3. El sensor de entrada a la transición en el modelo TPL se convertirá en la condición necesaria para cambiar de estado en el diagrama Grafcet, es decir, será la condición que gobernará la transición específica entre dos estados.

Regla 4. En el caso de que existan retrasos en los sensores de entrada, es decir, que se desee actuar hasta un cierto tiempo después de que se presentó el sensor, este tiempo será representado en la transición como un retraso.

Regla 5. Las salidas relacionadas con cada bit de memoria serán tratadas en la sección posterior.

En la Figura 2.19, se muestra la transformación de un modelo TPL a un diagrama Grafcet. En esta figura el bit de memoria M_1 es una condición inicial.

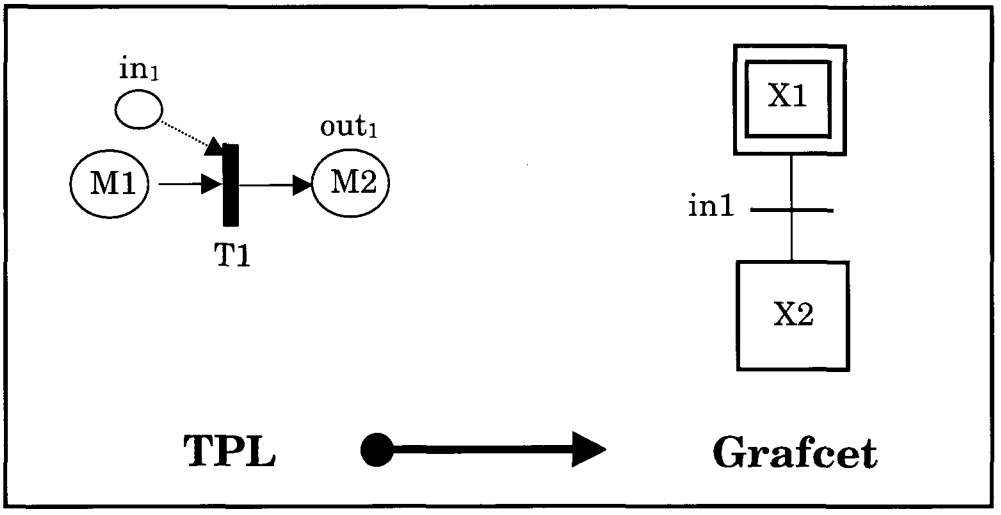


Figura 2.19. Ejemplo de construcción del diagrama Grafcet, a partir del modelo TPL.

Para el modelo TPL del proceso de estampado de latas mostrado en la Figura 2.12, se pueden aplicar estas reglas y obtener el diagrama Grafcet correspondiente. Este diagrama se muestra en la Figura 2.20. Dado que el modelo TPL se divide en dos etapas independientes, en el diagrama Grafcet existen dos diagramas que se ejecutarán de manera aislada, pero que parten de una transición común para los dos.

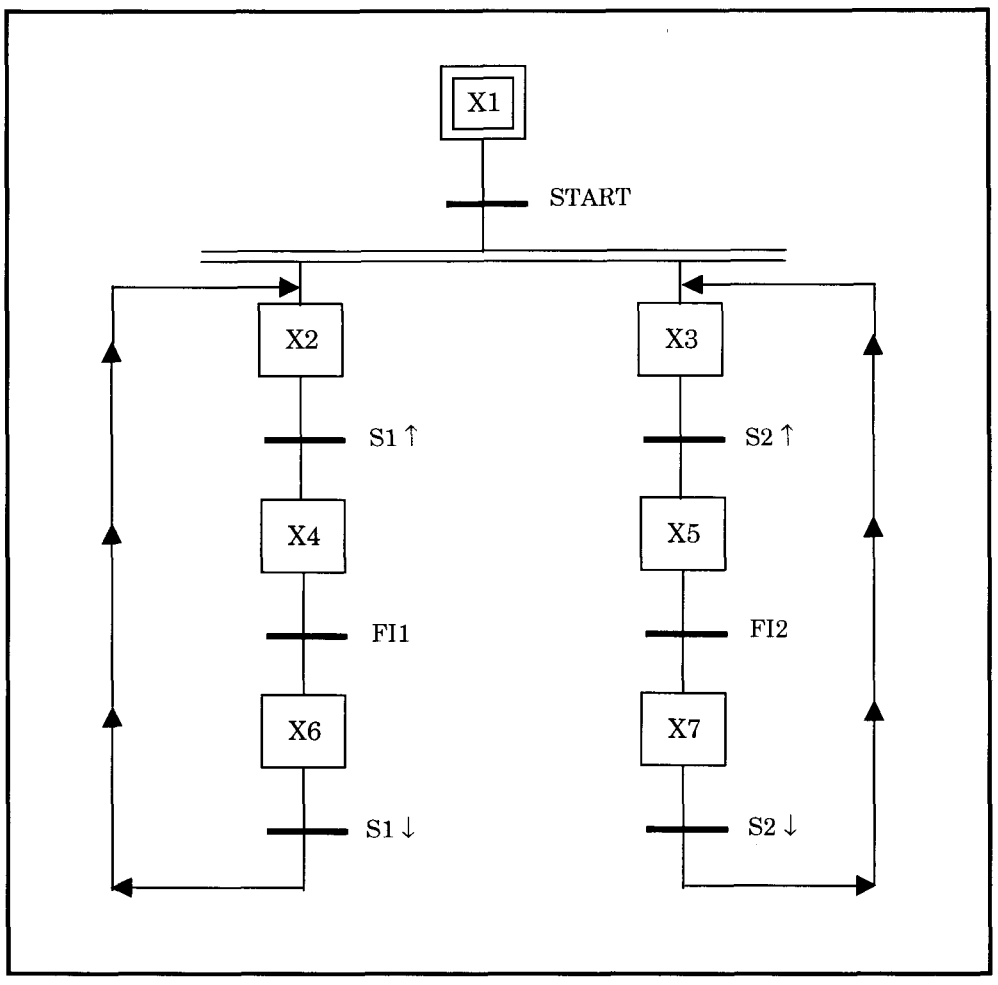


Figura 2.20. Sección Grafcet para el proceso de estampado de latas.

2.2.6.3 Construcción de la sección posterior a partir del modelo TPL.

En esta última parte, se construirá el diagrama escalera para que las salidas del proceso sean encendidas conforme su relación con los bits de memoria del modelo TPL.

1) Con base a la información proporcionada en el modelo TPL, las salidas relacionadas con cada bit de memoria, son ahora asignadas a cada etapa del diagrama Grafcet. Para que esta operación resulte un poco más sencilla al momento de programar, primero se llenará una tabla, como la que se muestra en la Tabla 2.1. En el Anexo 1 se muestra esta tabla sin llenar. En esta tabla, la columna de la izquierda muestra el número de salidas del PLC, la columna de en medio una pequeña descripción de la salida, mientras que la columna de la derecha la función lógica booleana[††] para cada salida con la indicación del tiempo de retraso en caso de que exista. Esta ecuación lógica booleana debe estar representada en función de las etapas de Grafcet y/o alguna otra entrada,.

Salidas del PLC	*Descripción de la Salida del PLC*	*Función lógica booleana*
Salida 1	Arranque del Motor 1	$X_1 \vee (X_4$ Retraso = 1 s)
Salida 2	Arranque de la Válvula 7	$X_3 \vee X_5$

Tabla 2.1. Tabla para relacionar las salidas del PLC con las etapas de la sección Grafcet y sus respectivos retrasos.

En la Tabla 2.1, se muestra también un ejemplo de cómo se llenaría esta tabla. Por ejemplo, la salida 1 (Arranque del motor 1) del PLC, se enciende cuando las etapas X_1 o X_4 de Grafcet están activas. Sin

[††] Se define el símbolo $\wedge$ como la operación booleana "Y", $\vee$ como la operación booleana "O" y & como la operación booleana "NO"

embargo, la salida 1 se encenderá después de 1 segundo que se activó la etapa X_4, ya que ésta está relacionada con un temporizador de retraso de ese valor.

Este tipo de representación resulta de mucha utilidad para programar la sección posterior cuando el número de salidas en el PLC es muy grande.

2) Basándose en la Tabla 2.1 se construirá el diagrama escalera de la sección posterior del lenguaje de programación GRAFCET. Se maneja a las etapas del diagrama Grafcet como contactos que encienden una salida, con sus respectivos retrasos. Primero deberán programarse las condiciones para la salida 1, después la salida 2 y así sucesivamente. La Figura 2.21 muestra la sección posterior correspondiente a la Tabla 2.1.

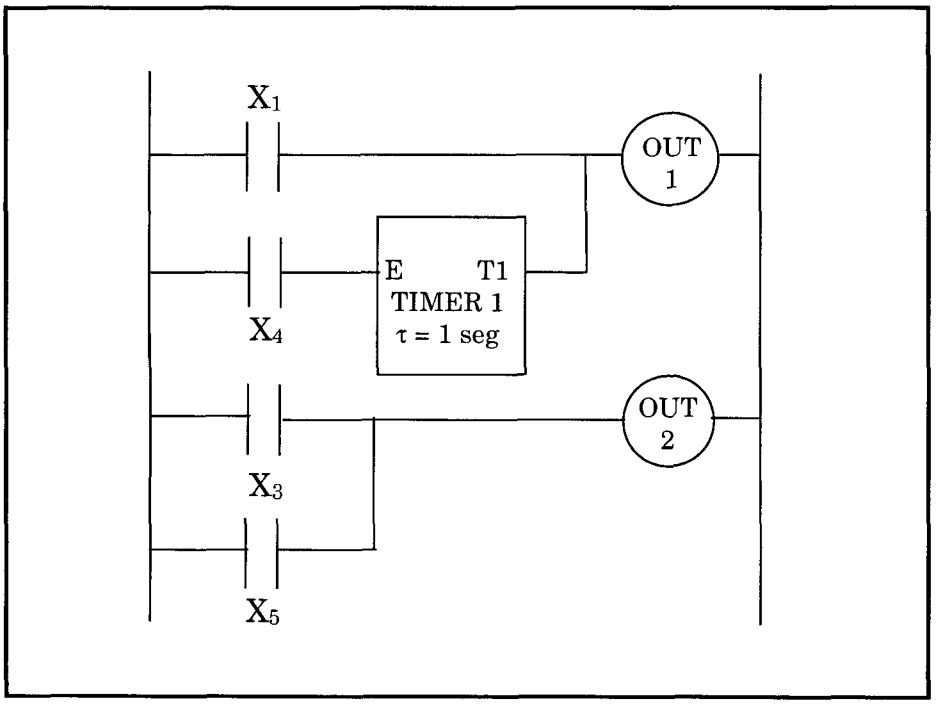


Figura 2.21. Ejemplo de construcción de la sección posterior basándose en la Tabla 2.1

Para el ejemplo del proceso de estampado de latas, es necesario formar la tabla que relacione las salidas del PLC con cada una de las etapas del diagrama Grafcet de la Figura 2.20. La Tabla 2.2 muestra esta relación.

Salidas del PLC	*Descripción de la Salida del PLC*	*Función lógica booleana*
Salida 1	SET del Motor	X2
Salida 1	RST del Motor	X1
Salida 2	Activar parador 1 (A1)	X1∨ X4
Salida 3	Retroceder parador 1 (R1)	X2
Salida 4	Activar parador 2 (A2)	X1∨ X5
Salida 5	Retroceder parador 2 (R2)	X3
Salida 6	Activar magneto 1 (M1)	X4
Salida 7	Activar magneto 2 (M2)	X5
Salida 8	Habilitar impresora 1 (IMP1)	X4
Salida 9	Habilitar impresora 2 (IMP2)	X5

Tabla 2.2. Relación de las salidas del PLC y las etapas del diagrama Grafcet para el proceso de estampado de latas.

Basándose en la Tabla 2.2 se forma el diagrama lógico de escalera de la sección posterior, el cual se muestra en la Figura 2.22. En este diagrama se utilizaron los nombres relacionados en la descripción de la salida para representarlas en cada escalón del diagrama escalera.

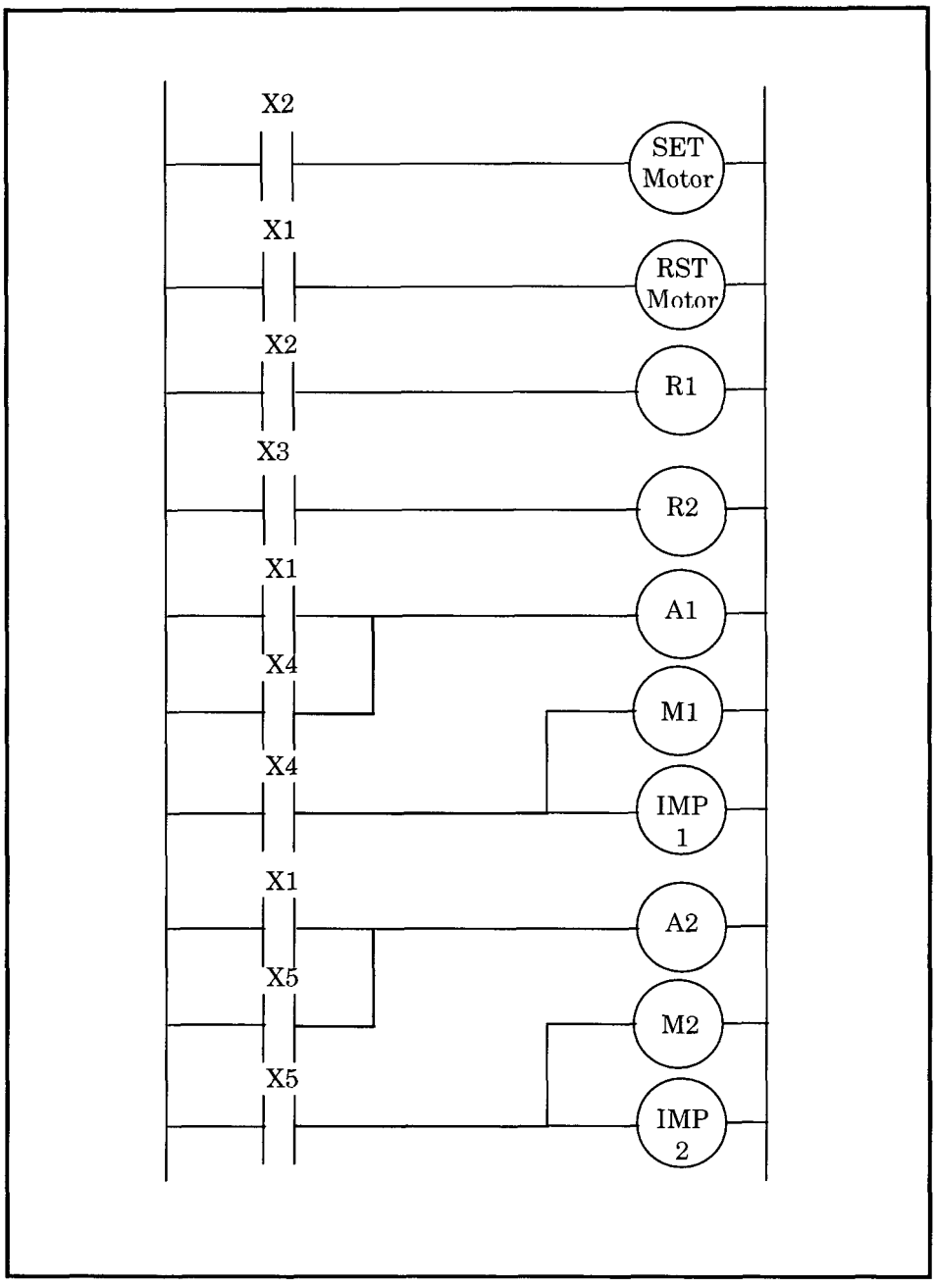


Figura 2.22. Diagrama lógico de escalera de la sección posterior para el proceso de estampado de latas.

3) Si se desea que cuando la condición de paro GRAFCET sea verdadera las salidas sostenidas se apaguen, es necesario añadir al diagrama lógico de escalera de la sección posterior esta condición activando la función

RST de la(s) salida(s) sostenida(s) que se desea(n) desactivar. Así, por ejemplo, si Z1 y Z2 son salidas sostenidas y P es la condición de paro de GRAFCET, el diagrama de la Figura 2.23 muestra como al presionar P se activará un RST a las salidas Z1 y Z2. Cabe mencionar que esta acción no es deseable en algunos casos, esto dependerá de la decisión del experto del proceso.

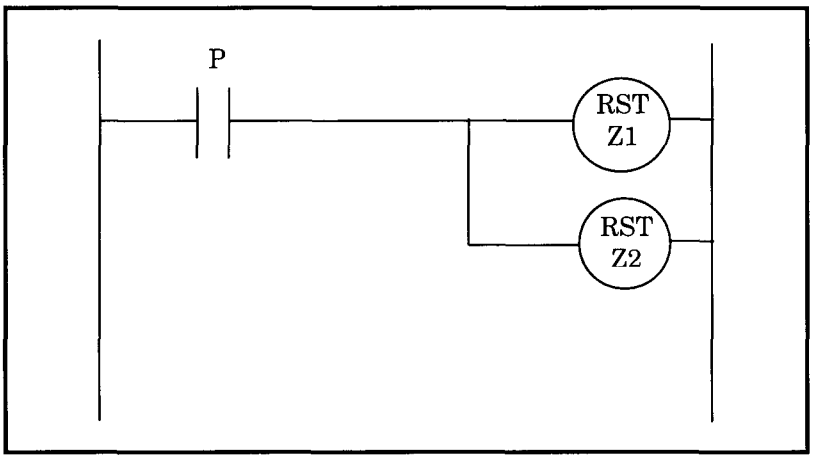


Figura 2.23. Diagrama escalera de la condición P activando el RST de Z1 y Z2.

Para el proceso de estampado de latas esta acción si es deseable, así el diagrama de la Figura 2.24 se adhiere a la sección posterior de GRAFCET. La condición de paro definida en este diagrama es la misma que se definió para la sección preliminar y la única salida sostenida es el motor de la banda (M).

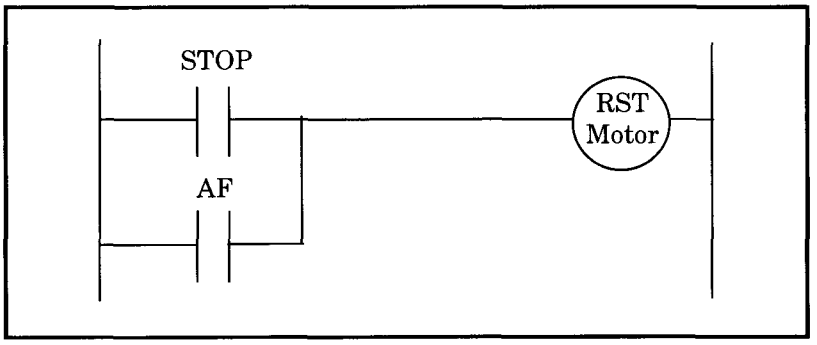


Figura 2.24. Diagrama de escalera para apagar el motor de la banda con la condición de paro de GRAFCET para el proceso de estampado de latas.

2.2.7 Comentarios finales.

En el Anexo 2 se muestra un resumen de los resultados obtenidos por etapa para el proceso de estampado de latas utilizando la metodología IPTG en GRAFCET.

Así en las cinco etapas anteriores se ha desarrollado la metodología IPTG en GRAFCET. Al final de la quinta etapa es posible obtener los diagramas necesarios para ser implementados en cualquier PLC que maneje el lenguaje de programación GRAFCET. Aunque sólo se deben realizar ciertos ajustes correspondientes a la nomenclatura de las salidas, etapas del diagrama Grafcet, bits de la sección preliminar, entre otros ajustes propios de la nomenclatura para cada marca de PLC.

El seguimiento de las salidas por etapa (sección posterior) y el diagrama Grafcet, le permiten al usuario un monitoreo del sistema más eficiente. La programación de las acciones de arranque, paro, pausa, arranque en frío y en caliente es muy sencilla, en comparación con la lógica del diagrama escalera donde resultaría más complicado.

Al aplicar la metodología propuesta en [14] no se tiene una referencia cruzada de los bits de memoria y las salidas del PLC, como la que se propuso en la quinta etapa del algoritmo de este trabajo de investigación.

En esta metodología se empieza desde la recopilación de los datos importantes en el proceso de producción hasta la obtención del programa implementable en un PLC. En el siguiente capítulo se mostrará una metodología con el mismo principio y el mismo resultado, pero en la que se cubrirán muchos aspectos que ésta no cubre, como lo son: fallas y eventos prohibidos, entre otros.

Capítulo 3

Metodología

RIMAnI en GRAFCET

En este capítulo se desarrollará la metodología RIMAnI en GRAFCET la cuál permite recolectar la información, realizar la modelación, el análisis, y finalmente la implementación en GRAFCET de un sistema de eventos discretos tomando como referencia la experiencia de sus ingenieros y utilizando las herramientas de modelación y análisis para este tipo de sistemas.

3.1 Introducción.

La metodología RIMAnI** en GRAFCET†† tiene como objetivo recolectar la información de un sistema automático a través de la experiencia de sus ingenieros. Con esta información se podrá modelar el sistema de eventos discretos o proceso como un autómata discreto, para después analizarlo y formar el modelo de autómata final que se utilizará en la implementación en GRAFCET.

Esta metodología viene a solucionar los problemas de implementación que se tienen cuando se desea programar un controlador lógico programable (PLC) para controlar un sistema de eventos discretos (SED). Algunas de las complicaciones que se tienen en la programación de PLCs cuando no se sigue una metodología son:

- Pobre estructuración al programar.
- Carencia de un modelo que describa al sistema por completo.
- Inclusión de etapas redundantes del sistema en el programa, lo cual lo hace difícil de analizar y modificar por el que lo realizó después de un cierto tiempo de haberlo realizado y pero aún por terceras personas.
- Se corre el riesgo de omitir algunos eventos prohibidos por estado y las acciones a tomar cuando éstos se presentan.

Básicamente la metodología RIMAnI en GRAFCET, sugerida en este trabajo de investigación, se puede representar como se muestra en la Figura 3.1. Por medio de ésta se suman las aportaciones de la experiencia del personal que

---

** **R**ecolección de la **i**nformación, **M**odelación, **An**álisis e **I**mplementación

†† Se manejará GRAFCET (con mayúsculas) como el lenguaje de programación usado en algunos controladores comerciales, y Grafcet (en minúsculas) como la herramienta de modelación de eventos discretos

conoce el SED, así como de la teoría de los autómatas discretos logrando un modelo implementable en GRAFCET.

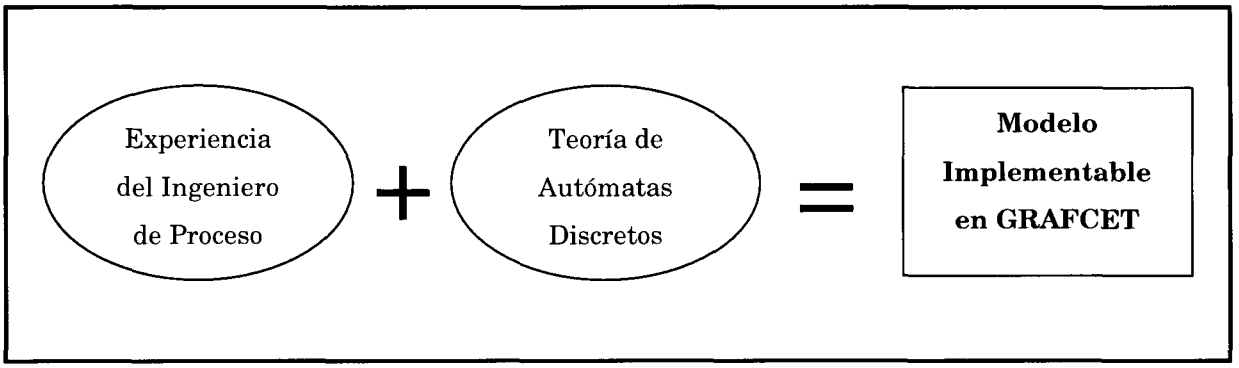


Figura 3.1. Representación de la metodología RIMAnI en GRAFCET.

La metodología RIMAnI en GRAFCET parte de un proceso previamente instrumentado para trabajar automáticamente, pero no se descarta la idea de que como resultado de aplicar dicha metodología, se observen cambios en la instrumentación para un mejor desempeño del mismo. Así, el objetivo principal de esta metodología es obtener el modelo autómata discreto del proceso, para después construir el diagrama GRAFCET que controlará el SED por medio de un PLC.

En la Figura 3.2 se muestra un diagrama de flujo de las etapas para la metodología RIMAnI en GRAFCET.

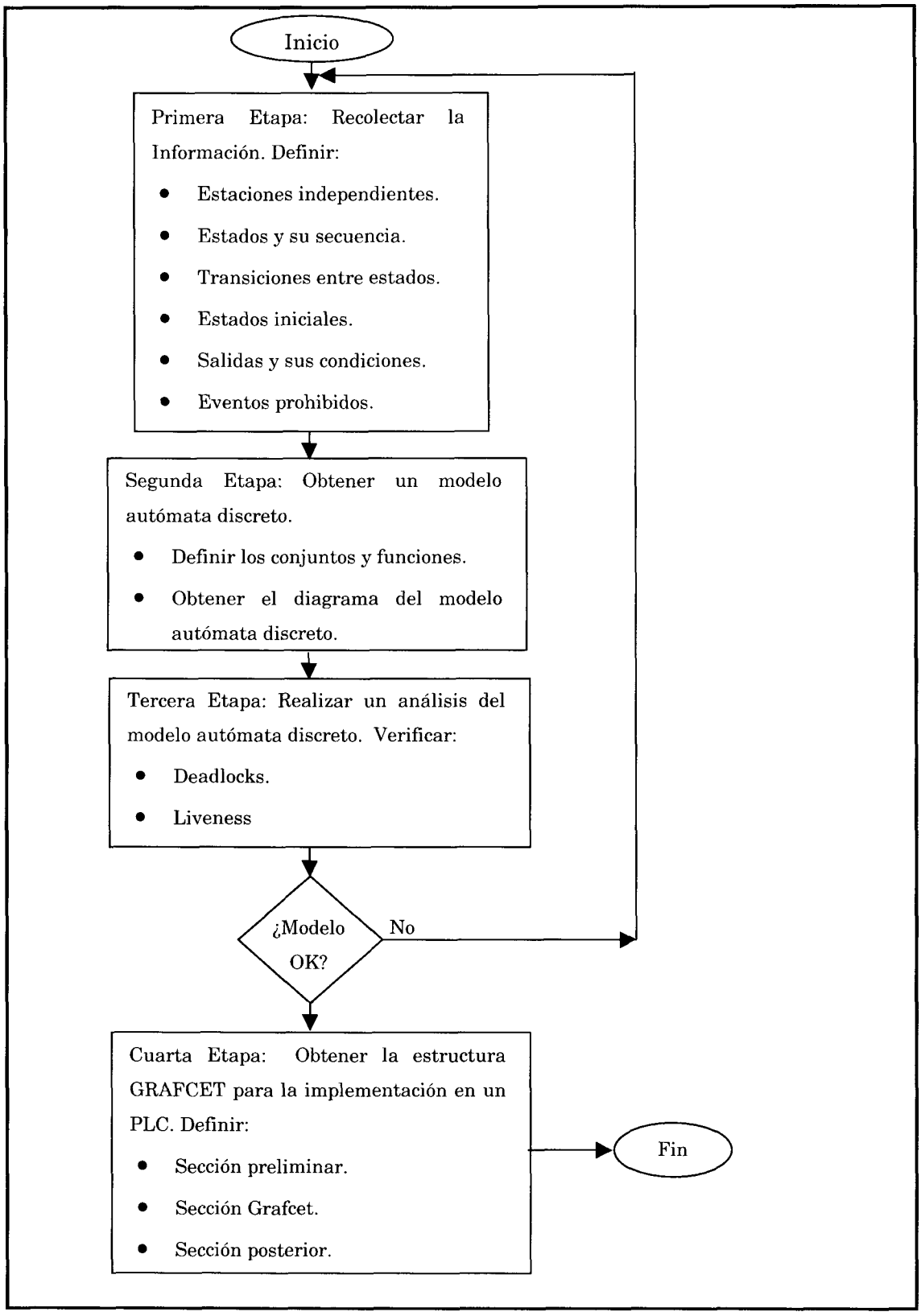


Figura 3.2. Diagrama de las etapas para la metodología RIMANI en GRAFCET.

3.2 Desarrollo de la metodología RIMAnI en GRAFCET.

3.2.1 Descripción del proceso de estampado de latas que se utilizará en el desarrollo de la metodología RIMAnI en GRAFCET.

En el capítulo 2 de este trabajo de investigación se utilizó el proceso de estampado de latas como ejemplo para desarrollar la metodología IPTG en GRAFCET. Con la finalidad de llevar a cabo una comparación al final de este capítulo, se utilizará el mismo proceso como ejemplo para mostrar el desarrollo de la metodología RIMAnI en GRAFCET. En la Figura 3.3 se muestra el proceso de estampado de latas.

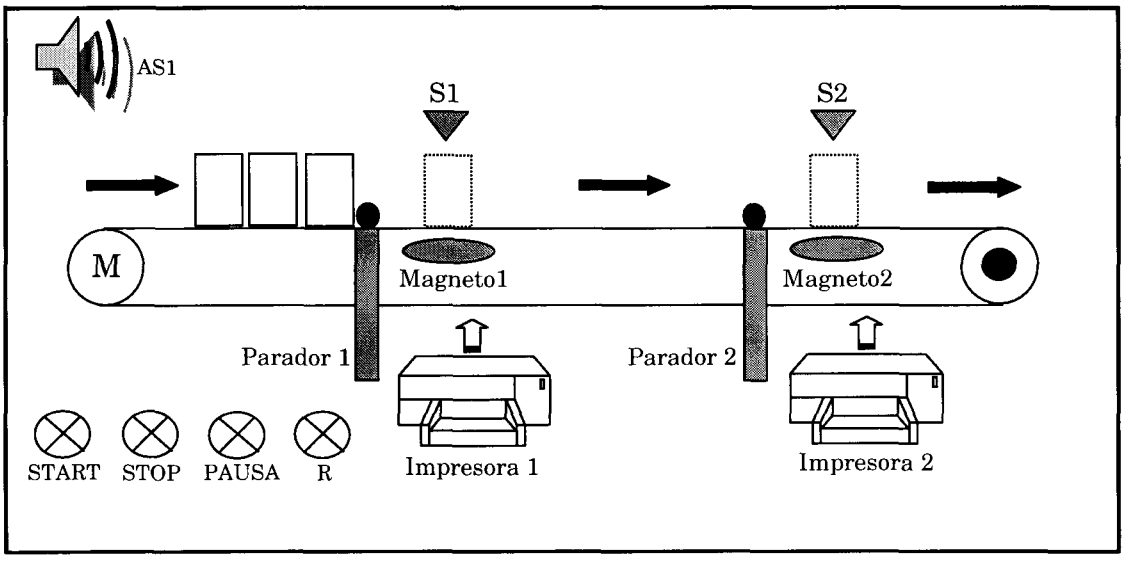


Figura 3.3. Proceso de estampado de latas.

La composición y dinámica de éste es la siguiente:

- Un motor (M) mueve una banda, la cual transporta las latas a estampar hasta las dos estaciones de impresión.
- Existen cuatro botones: START, STOP, PAUSA y R (Reestablecimiento después de una pausa).

- La primera estación de impresión tiene un tope parador que detiene las latas de entrada, y además un magneto que sostiene la lata mientras se imprime. En esta estación existe un sensor de presencia de lata (S1). Al terminar la impresión, la máquina impresora genera una señal de "fin de impresión" (FI1).
- Después del fin de la impresión se desactiva el magneto y la lata sale de la estación. Es hasta ese momento cuando el tope parador deja pasar la siguiente lata a ser etiquetada.
- Al terminar en la estación 1, la lata pasa a una segunda estación de impresión, donde al igual que la estación 1, se tiene un parador, un magneto y un sensor (S2). La impresora también manda una señal de "fin de impresión" (FI2).
- Al salir de la estación 2 la lata es empacada manualmente.
- Las señales de control y de sensado para los topes paradores son las siguientes:
  - R1 y R2, representan, respectivamente, las señales manipuladoras para la retracción de los topes paradores 1 y 2[##].
  - SR1 y SR2 son, respectivamente, los sensores para detectar que los topes paradores 1 y 2 están retraídos.
  - A1 y A2, representan, respectivamente, las señales manipuladoras para la extensión de los topes paradores 1 y 2.
  - SA1 y SA2 son, respectivamente, los sensores para detectar que los topes paradores están extendidos.

- Las señales de control para los magnetos 1 y 2 están respectivamente representadas por M1 y M2.

[##] Para el proceso los topes paradores utilizados son pistones de doble efecto, es decir, utilizan válvulas distribuidoras 4/2.

- Las señales de control para las impresoras 1 y 2 están respectivamente representadas por IMP1 e IMP2.
- La alarma sonora AS1 se activará cuando ocurra un mal funcionamiento del proceso (ocurrencia de eventos prohibidos).
- La secuencia inicia con el botón START.

Las condiciones iniciales establecidas son las siguientes:

- Las dos estaciones están vacías (S1 = 0 y S2 = 0).
- El motor está apagado (M = 0).
- Los paradores están extendidos (R1 = R2 = 0 y A1 = A2 = 1).
- Los magnetos están desactivados (M1 = M2 = 0).
- Las impresoras se encuentran en el estado de "no imprimir" (IMP1 = IMP2 = 0).
- La alarma sonora está desactivada (AS1 = 0).

3.2.2 Primera Etapa: Recolectar la información del proceso.

En esta etapa se obtendrá la información necesaria del proceso a partir de preguntas dirigidas al "experto" encargado de éste. En esta etapa se asume que el proceso ya está instrumentado para trabajar automáticamente, es decir, ya cuenta con actuadores, sensores, alarmas, indicadores entre otros. Al finalizar esta etapa deben obtenerse los datos necesarios para en la siguiente etapa poder modelar el SED utilizando el modelo autómata discreto.

3.2.2.1 Estaciones independientes.

Una estación A es independiente de otra B si y solo si el funcionamiento de la estación A no depende de ninguna acción realizada en la estación B. Así, la pregunta a contestar es ¿cuántas y cuáles estaciones independientes tiene el proceso?. Las estaciones pueden ser diversas como por ejemplo: estación de alarmas, estación de llenado del tanque, estación de pulido, entre otras.

Una vez que se han definen N estaciones independientes es necesario nombrarlas como: E1, E2, E3, ..., EN, donde $N \geq 1$.

Para el ejemplo del proceso de estampado de latas se definen dos funciones independientes, las cuales son:

- E1: Estación para la impresión de la primera etiqueta.
- E2: Estación para la impresión de la segunda etiqueta.

En la Figura 3.4 se muestra el proceso de estampado de latas dividido en las dos estaciones E1 y E2. Así, la acción de impresión en cada estación es totalmente independiente y puede ser vista como una estación en la que entran latas, se imprimen y salen de ésta. Físicamente, el motor M es el mismo para las dos estaciones.

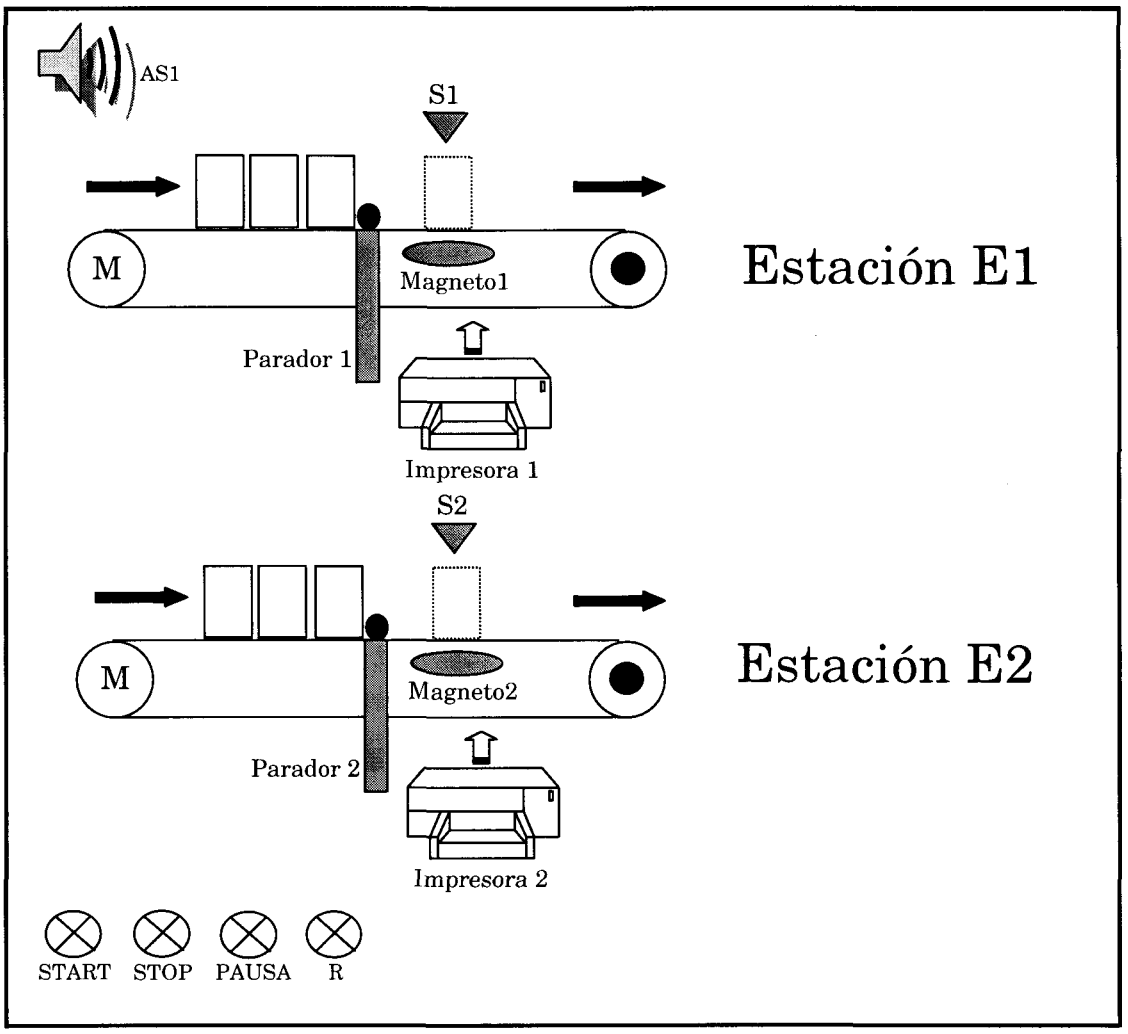


Figura 3.4. Estaciones E1 y E2 para el proceso de estampado de latas.

3.2.2.2 Definir los estados y su secuencia dentro de cada estación.

Para cada estación del punto anterior se define un objetivo final y como consecuencia los estados necesarios para llevarlo a cabo. Por ejemplo, para pintar una barda (objetivo final) es necesario pasar por lo siguientes estados:

- Cepillar la barda.
- Rezanar la barda.
- Aplicar sellador.
- Aplicar la pintura.

Entonces, para cumplir el objetivo de pintar la barda es necesario pasar a través de cuatro estados. Así, la definición de estados dentro de una estación debe ser tal que se describa a detalle cualquier paso necesario para llevar a cabo una acción. Para facilitar el proceso de definición de estados se proponen las siguientes reglas:

Regla 1. Un estado se representará como un cuadrado con la acción escrita dentro de él.

Regla 2. Un estado como parte de una estación se distingue cuando se realiza una o varias acciones como parte de ésta, por ejemplo, pintar, taladrar, abrir una válvula, cerrar una puerta, prender un motor, entre otras.

Regla 3. No se deben juntar dos acciones secuenciales o mutuamente exclusivas en un solo estado, es decir, si para realizar la acción A es necesario llevar a cabo la B, entonces es necesario definir un estado para la acción A y otro para la B. Por ejemplo, la acción de lijar y aplicar el sellador no se debe de tomar como un sólo estado ya que primero se debe lijar y después aplicar el sellador.

Regla 4. Los estados definidos deben ser ordenados en forma secuencial.

Regla 5. Los estados de mal funcionamiento del sistema (estados prohibidos) merecen un tratamiento especial y serán considerados en 3.2.2.6.

Para el proceso de estampado de latas se definen los siguientes estados para la estación E1:

- Ausencia de latas en la estación E1.
- Activar el parador 1 y el magneto 1.
- Imprimir la etiqueta 1.
- Desactivar el magneto 1 (lata dejando la estación E1).
- Desactivar el parador 1.

Para la estación E2, los estados son:

- Ausencia de latas en la estación E2.
- Activar el parador 2 y el magneto 2.
- Imprimir la etiqueta 2.
- Desactivar el magneto 2 (lata dejando la estación E2).
- Desactivar el parador 2.

En la Figura 3.5 se muestra la secuencia de estos estados para las estaciones E1 y E2.

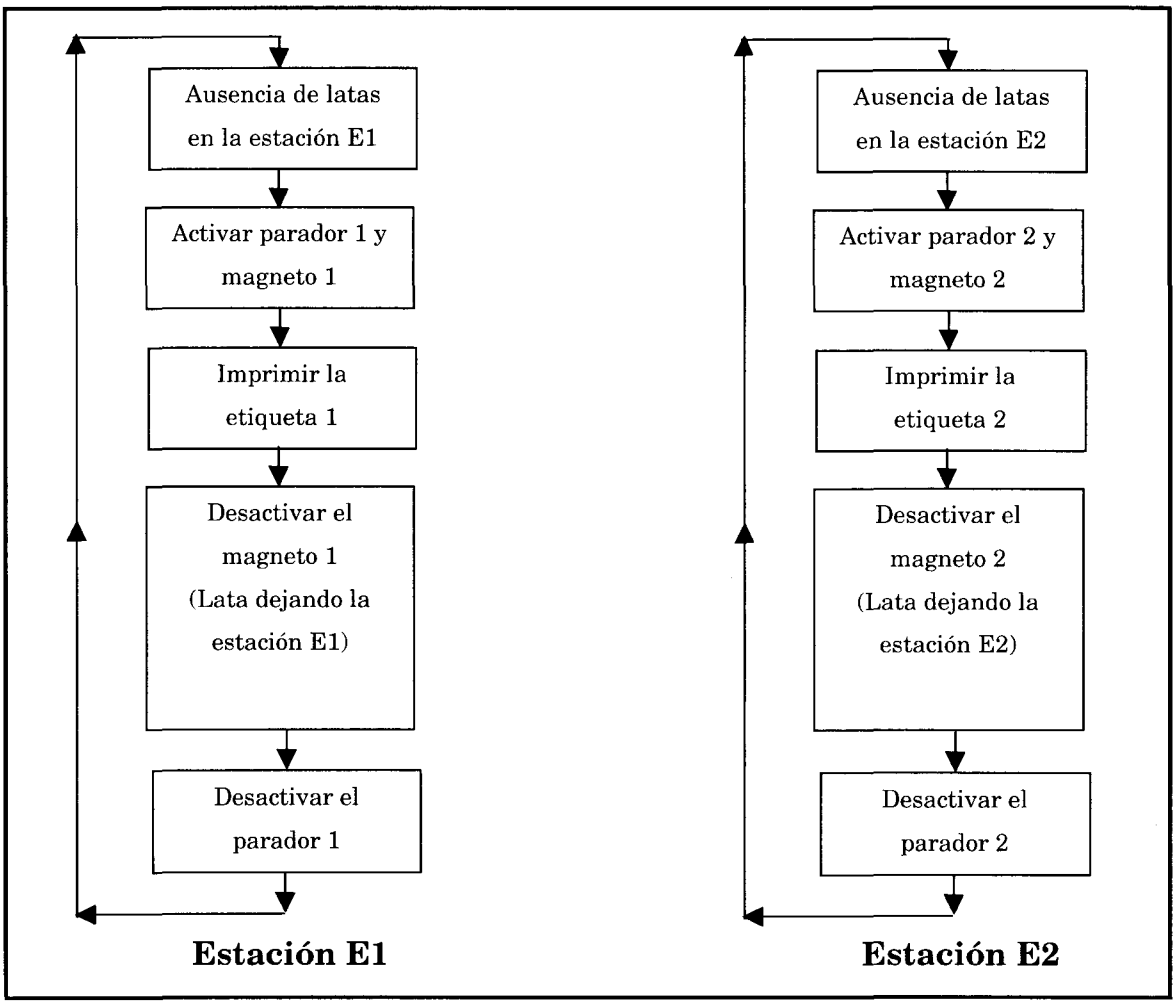


Figura 3.5. Secuencia de estados para las estaciones E1 y E2.

3.2.2.3 Definir las condiciones para llevar a cabo las transiciones entre los estados.

En este paso se deben definir las condiciones necesarias en las transiciones para pasar de un estado a otro. Así, estas condiciones son ecuaciones booleanas función de las señales de los sensores y botones del proceso.

En esta etapa se debe obtener un diagrama con los estados y las transiciones definidas por los elementos del proceso. Para clarificar qué es una condición entre estados se expone el siguiente ejemplo: un alumno se encuentra en séptimo semestre (estado 7) por lo que su siguiente estado es el octavo semestre (estado 8) y la condición para pasar del estado 7 al 8, es decir la condición entre estados, es que todas las materias del estado 7 estén aprobadas.

Para la definición de transiciones se proponen las siguientes reglas:

Regla 1. La transición se representa como una flecha con una línea perpendicular indicando el sentido unidireccional de ésta.

Regla 2. La condición de ejecución de la transición es una ecuación booleana[§§] escrita a un lado de la línea perpendicular.

Regla 3. Todos los estados deben de tener al menos una transición de entrada y al menos una de salida, excepto que sea un estado inicial (no tiene transición de entrada) o un estado final (no tiene transición de salida).

Regla 4. Si se desea incluir el efecto de retraso al pasar de un estado a otro(durante la transición), éste se debe definir como un estado intermedio con un timer asociado. Así, por ejemplo si la condición para la ejecución

[§§] Se define el símbolo $\wedge$ como la operación booleana "Y", $\vee$ como la operación booleana "O" y & como la operación booleana "NO"

de una transición entre el estado X1 y X2 es: (A = 2 seg.)∧(S1), se define el estado intermedio X1' para el retraso, como se muestra en la Figura 3.6.

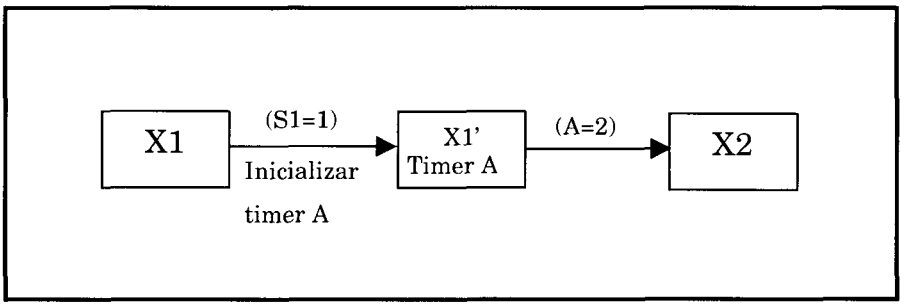


Figura 3.6. Diagrama para el ejemplo de la regla 5.

Para el proceso de estampado de latas se obtiene el diagrama de estados de la Figura 3.7. En esta figura, la condición (START ∧ SA1 ∧ SA2) es común para las dos estaciones, así se dibujó como una línea y dos flechas entrando a los estados, como lo marca la Regla 1.

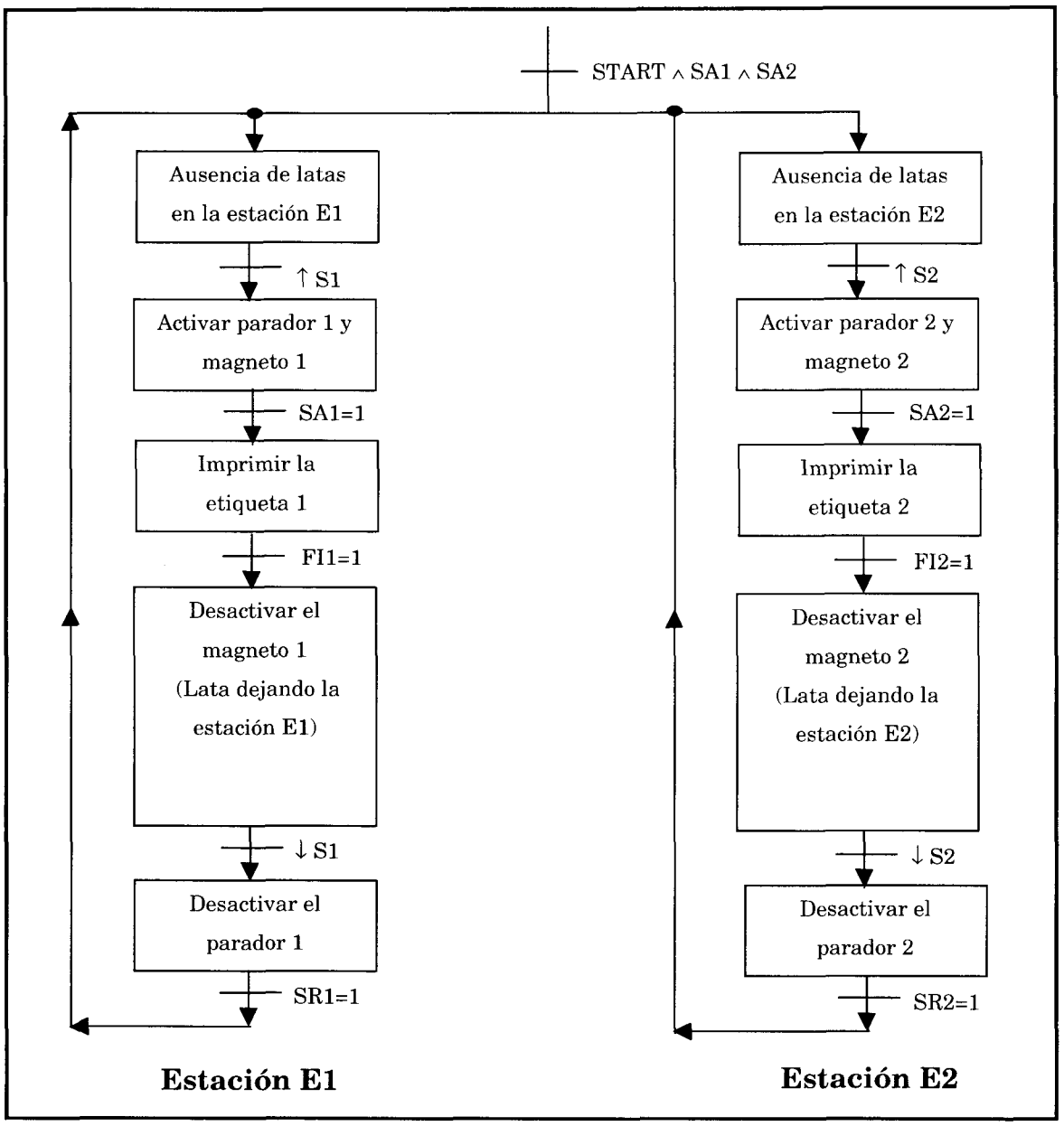


Figura 3.7. Diagrama de estados para el proceso de estampado de latas.

3.2.2.4 Definir el(los) estado(s) inicial(es) del proceso y relacionarlos con cada estación.

Basándose en las condiciones iniciales del proceso es necesario definir los estados iniciales del proceso, es decir, aquellos donde estas condiciones

existan. Cada estación definida debe tener un estado inicial asociado con ésta.

Para obtener los estados iniciales del proceso se proponen las siguientes reglas:

Regla 1. El estado inicial se representará como un cuadrado doble con la acción descrita dentro de él.

Regla 2. Un estado inicial es aquel donde las condiciones iniciales del proceso existen, así, este estado es aquel donde el SED inicia por primera vez.

Regla 3. Un estado inicial es donde una estación debe empezar su ciclo después de una inicialización.

Regla 4. Un estado inicial puede servir como un estado de seguridad, es decir, es posible iniciar el programa pero no la secuencia sino hasta que se cumplan las condiciones iniciales del sistema.

Regla 5. Para cada estado inicial es necesario definir las transiciones de salida y entrada a éste en caso de existir.

La proposición de estados iniciales se basa en las reglas anteriores, así es posible completar el diagrama de estados que se construyó en el punto anterior.

Para el proceso de estampado de latas se define sólo un estado inicial, ya que es necesario asegurarse de que las condiciones iniciales existan antes de empezar la impresión en las dos estaciones. La transición de salida del estado inicial estará definida por la condición (START $\wedge$ SA1 $\wedge$ SA2) y la

transición de entrada será igual a vacío debido a que este estado será del tipo *source*[***]. El diagrama de estados resultante se muestra en la Figura 3.8.

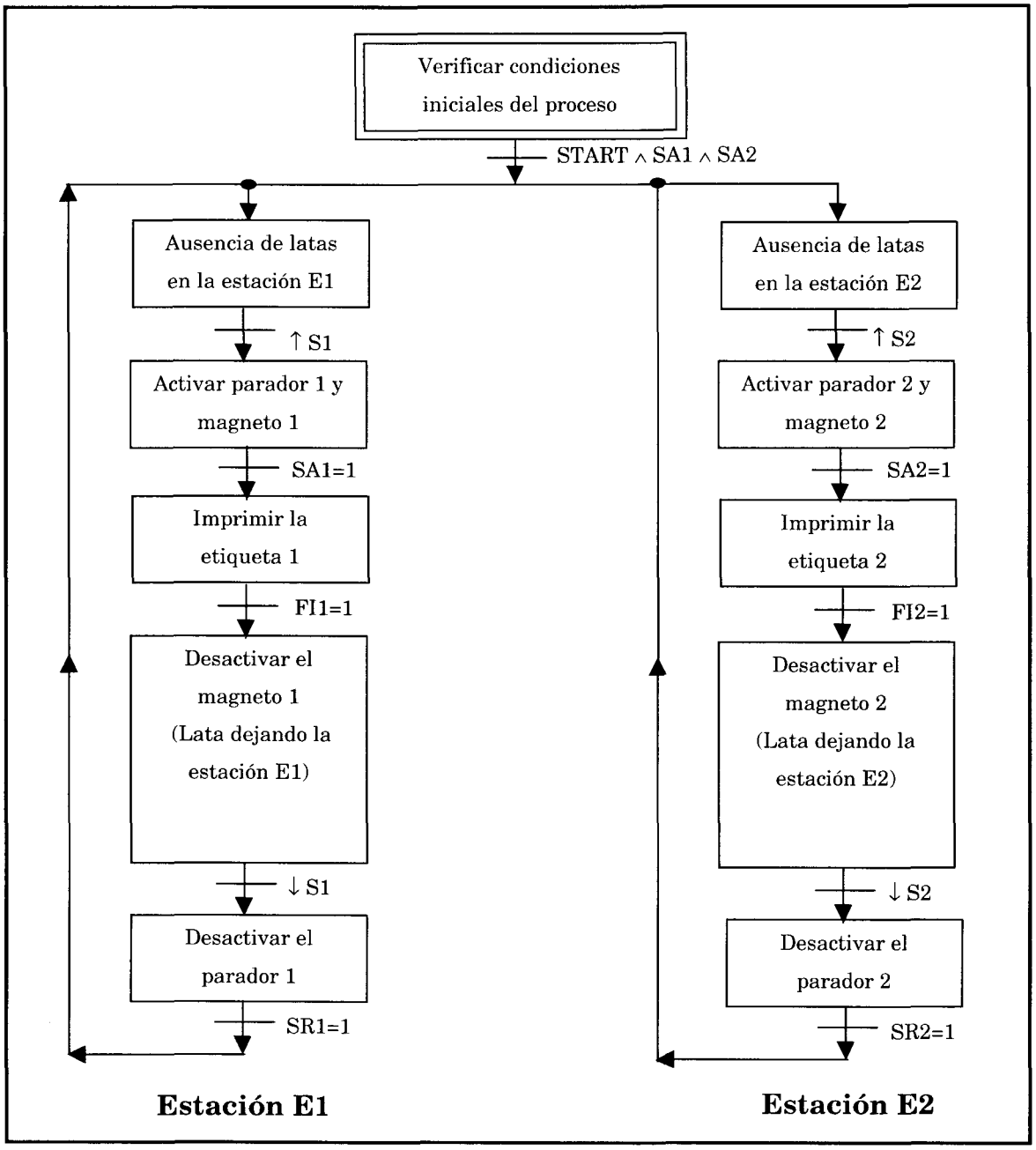


Figura 3.8. Diagrama de estados para el proceso de estampado de latas.

[***] Se define un estado *source* como aquel que no tiene transiciones de entrada y el *sink* como aquel que no tiene transiciones de salida.

3.2.2.5 Definir las salidas de control hacia el proceso y sus condiciones para cada estado.

Una vez que se ha definido el diagrama de estados del proceso es necesario contemplar las salidas relacionadas con cada estado del proceso, es decir, que acciones se llevarán a cabo en cada etapa del SED. Para esto se propone la Tabla 3.1 que relaciona el estado del proceso con las salidas correspondientes definidas por el usuario. En el Anexo 3 se muestran las tablas en blanco que se utilizan en la metodología RIMAnI en GRAFCET.

Para llenar la Tabla 3.1 se proponen las siguientes reglas:

Regla 1. Para los *n* estados definidos en el diagrama de estados se deberá asignar un nombre a cada uno de ellos, éste estará formado por la letra X y el número de estado iniciando la numeración con el 0, es decir, X0, X1, X2, ..., X*n*.

Regla 2. La primer columna de la tabla se llena con los nombres de los estados (X0, X1, ..., X*n*).

Regla 3. La segunda columna de la tabla se llena con la descripción o acción que se realiza en ese estado (lijar la barda, pintar la barda, aplicar el sellador, entre otros)

Regla 4. La tercer columna de la tabla se llena con las salidas que se activan en cada estado, y se debe hacer mención del tipo de salida. Para esto, se definen dos tipos de salidas:

a. *Salida SET-RST.* Este tipo de salida se activa (SET) en el estado y permanece así hasta que es desactivada (RST)
b. *Salida no sostenida.* Este tipo de salida permanece activa sólo cuando el proceso se encuentra en el(los) estado(s) asociado(s) con ésta.

Si no se indica nada, se asume que la salida es no sostenida. Pero si la salida es tipo SET-RST es necesario indicar si la salida se activará (SET) o desactivará (RST)

Regla 5. Al definir la salida en la tercera columna, se debe escribir la condición booleana que la activará. Si sólo se escribe el nombre de la salida se asume que sólo la activación del estado es condición suficiente, en caso contrario es necesario escribir la ecuación booleana asociada a la salida.

Nombre del Estado	Descripción del Estado	Salidas a activar con condición booleana y tipo de salida

Tabla 3.1. Tabla que relaciona los estados y las salidas del proceso.

Para el proceso de estampado de latas, se obtuvo la Tabla 3.2 siguiente. En el estado inicial se define la salida del motor de la banda transportadora como del tipo SET-RST, y se indica en X0 que la salida se desactivará (RST), mientras que el estado X1 la activará (SET).

Nombre del Estado	Descripción del Estado	Salidas a activar con condición booleana y tipo de salida
X0	Verificar condiciones iniciales del proceso	• Motor (RST) • A1 • A2
X1	Ausencia de latas en la estación E1	• Motor (SET) • R1
X2	Ausencia de latas en la estación E2	• R2
X3	Activar parador 1 y magneto 1	• A1 • M1(SET)
X4	Activar parador 2 y magneto 2	• A2 • M2 (SET)
X5	Imprimir la etiqueta 1	• IMP1
X6	Imprimir la etiqueta 2	• IMP2
X7	Desactivar el magneto 1 y Lata dejando la estación E1	• M1(RST)
X8	Desactivar el magneto 2 y Lata dejando la estación E2	• M2(RST)
X9	Desactivar el parador 1	• R1
X10	Desactivar el parador 2	• R2

Tabla 3.2. Tabla que relaciona los estados y las salidas para el proceso de estampado de latas.

3.2.2.6 Definir las señales, sensores o botones del proceso que en caso de ocurrir en un estado se considerarán eventos prohibidos.

En algunos estados la ocurrencia de ciertos eventos, llamados eventos prohibidos, significa que existe un problema en el proceso, por lo que es necesario activar una alarma o suspender el proceso, es decir, tomar una

acción que le avise al usuario que existe un malfuncionamiento y que puede interrumpir o no la secuencia de funcionamiento normal (sin fallas) del SED.

Por esto el experto del proceso debe indicar para cada estado cuáles son eventos prohibidos. Así, se propone la Tabla 3.3 y las siguientes reglas para definir adecuadamente los eventos prohibidos:

Regla 1. Un evento (discreto) puede ser la señal de un sensor, un botón, o cualquier señal discreta proveniente del proceso.

Regla 2. Un evento prohibido en el estado X*n* es aquél que en caso de ocurrir en este estado significará un mal funcionamiento del proceso.

Regla 3. Un evento prohibido puede ser una ecuación booleana, es decir, puede estar formado por una combinación lógica indeseada de posibles estados de variables, por ejemplo, $(A \wedge B) \wedge (C \geq 2.51) = F$, donde si A=1, B=1 y C es mayor o igual a 2.51 entonces F será igual a 1.

Regla 4. Un evento prohibido puede estar temporizado y realizar diferentes acciones dependiendo de su duración, por ejemplo, si el evento *A* sucede en el estado X*n* por 5 segundos se activará una alarma, pero si su duración es de 10 segundos el sistema se detendrá.

Regla 5. La primer columna de la tabla se llena con los nombres de los estados (X0, X1, ..., X*n*)

Regla 6. La segunda columna de la tabla se llena con la descripción o acción que se realiza en ese estado (lijar la barda, pintar la barda, aplicar el sellador, entre otros)

Regla 7. La tercer columna de la tabla se llena con los eventos prohibidos para cada estado. Si el evento tiene una duración en específico que lo hace evento prohibido se debe incluir el dato de tiempo en segundos, por ejemplo, si el evento A es prohibido sólo si su duración es de 0.5 segundos éste deberá especificarse como A(0.5)

Regla 8. La cuarta columna de la tabla se llena con la acción que se debe llevar a cabo después de la ocurrencia de un evento prohibido para cada estado; es decir, si el estado X*n* tiene *k* eventos prohibidos es necesario definir la acción que se llevará a cabo para los *k* eventos, que podemos interpretar como los estados siguientes a un evento prohibido.

Regla 9. Las salidas de la cuarta columna, deben especificarse de acuerdo a su tipo y además con su condición booleana correspondiente. Para esto, se definen dos tipos de salidas:

a. *Salida SET-RST.* Este tipo de salida se activa (SET) en el estado y permanece así hasta que es desactivada (RST)
b. *Salida no sostenida.* Este tipo de salida permanece activa sólo cuando el proceso se encuentra en el(los) estado(s) asociado(s) con ésta.

si no se indica nada, se asume que la salida es no sostenida. Pero si la salida es tipo SET-RST es necesario indicar si la salida se activará (SET) o se desactivará (RST).

Regla 10. Al definir la salida y su tipo en la cuarta columna se debe escribir la condición booleana que la activará. Si sólo se escribe el nombre de la salida se asume que sólo la activación del estado es condición suficiente, en caso contrario es necesario escribir la ecuación booleana asociada a la salida.

Regla 11. La ocurrencia de un evento prohibido en un estado X*n* ocasionará que el proceso cambie a un estado prohibido $X^i n$ donde se llevarán a cabo las acciones definidas en la cuarta columna en función del evento prohibido y el superíndice "i" representa el i-ésimo estado prohibido subsecuente al estado Xn debido a la ocurrencia de un evento prohibido.

Regla 12. Si un estado no tiene eventos prohibidos asociados, entonces éste no tendrá ningún estado prohibido y en consecuencia las celdas correspondientes de la tabla quedarán vacías.

Nombre del Estado	Descripción del Estado	Evento(s) prohibido(s)	Acción a llevar a cabo para cada evento prohibido (tipo y condición booleana)

Tabla 3.3. Tabla que relaciona los estados del proceso y los eventos prohibidos.

Para el ejemplo de estampado de latas siguiendo las reglas antes mencionadas se obtiene la Tabla 3.4. Así, por ejemplo en el estado X0, por ejemplo, la ocurrencia del evento S1 o S2 ocasionará que el proceso entre en un estado prohibido de alarma ya que la estación E1 o E2 tiene una lata y el sistema no puede iniciar. En consecuencia, la acción a tomar es que se encienda la alarma sonora AS1. Otros dos eventos prohibidos relacionados con este estado se producen cuando (SA1=1 $\wedge$ SR1=1) ó (SA2=1 $\wedge$ SR2=1), lo que significa que alguno o los dos sensores están fallando.

Nombre del Estado	Descripción del Estado	Evento(s) prohibido(s)	Acción a llevar a cabo para cada evento prohibido (tipo y condición booleana)
X0	Verificar condiciones iniciales del proceso	➢ S1 ➢ S2 ➢ SA1∧SR1 ➢ SA2∧SR2 ➢ EI1 ➢ EI2	➢ (S1,S2, FI1, FI2)→ Prender AS1 ➢ (SA1∧SR1, SA2∧SR2,) → Prender AS1
X1	Ausencia de latas en la estación E1	➢ SA1∧SR1	➢ (SA1∧SR1) → Prender AS1
X2	Ausencia de latas en la estación E2	➢ SA2∧SR2	➢ (SA2∧SR2) → Prender AS1
X3	Activar parador 1 y magneto 1	➢ SA1∧SR1 ➢ &S1	➢ (SA1∧SR1) → Prender AS1 ➢ (&S1) → Prender AS1
X4	Activar parador 2 y magneto 2	➢ SA2∧SR2 ➢ &S2	➢ (SA2∧SR2) → Prender AS1 ➢ (&S2) → Prender AS1
X5	Imprimir la etiqueta 1	➢ SA1∧SR1 ➢ &S1	➢ (SA1∧SR1) → Prender AS1 ➢ (&S1) → Prender AS1
X6	Imprimir la etiqueta 2	➢ SA2∧SR2 ➢ &S2	➢ (SA2∧SR2) → Prender AS1 ➢ (&S2) → Prender AS1
X7	Desactivar el magneto 1 y lata dejando la estación E1	➢ SA1∧SR1	➢ (SA1∧SR1) → Prender AS1
X8	Desactivar el magneto 2 y Lata dejando la estación E2	➢ SA2∧SR2	➢ (SA2∧SR2) → Prender AS1
X9	Desactivar el parador 1	➢ SA1∧SR1	➢ (SA1∧SR1) → Prender AS1
X10	Desactivar el parador 2	➢ SA2∧SR2	➢ (SA2∧SR2) → Prender AS1

Tabla 3.4. Tabla que relaciona los estados y los eventos prohibidos para el proceso de estampado de latas.

3.2.3 Segunda Etapa: Obtener un modelo autómata discreto para el SED.

En esta etapa se obtendrá un modelo autómata del SED basándose en la información obtenida del experto en la etapa anterior.

3.2.3.1 Definir los conjuntos y funciones para el modelo autómata discreto del proceso.

Por definición[†††], un autómata determinístico finito está compuesto por:

- El conjunto de eventos del SED, formado por:
  - El conjunto de eventos permitidos denominado E.
  - El conjunto de eventos prohibidos denominado T.
- El conjunto de estados del SED, denominado X.
- El conjunto de estados iniciales del SED, denominado X_0.
- El conjunto de funciones de transición de estados del SED, formado por:
  - La función de transición de estados permitidos denominada f.
  - La función de transición de estados permitidos denominada t.

Así, se proponen las siguientes reglas para la construcción del modelo autómata discreto a partir de la información obtenida en la primera etapa de esta metodología:

Regla 1. El conjunto de eventos E del modelo autómata se forma con todas las señales y botones del proceso.

[†††] Definición expuesta en la sección 1.3 del capítulo 1 de este trabajo de investigación.

Regla 2. El conjunto de estados X del modelo autómata se forma con todos los estados definidos en el punto 3.2.2.2, se deben incluir los estados de todas las estaciones.

Regla 3. El conjunto de estados iniciales X_0 del modelo autómata se forma con el(los) estado(s) inicial(es) definido(s) en el punto 3.2.2.3

Regla 4. El conjunto de eventos prohibidos T del modelo autómata se forma con los eventos definidos en la tercer columna de la tabla del punto 3.2.2.6, los eventos repetidos sólo se mencionan una vez.

Regla 5. Cada función de transición de estados f se forma con las transiciones definidas en el punto 3.2.2.3, si ésta está representada por una ecuación booleana, dicha ecuación debe ser utilizada en la función. Por definición las transiciones deberán ser definidas de la siguiente manera:

$$f(\text{estado actual, evento}) \rightarrow \text{estado siguiente}$$

Por ejemplo, si la transición del estado X1 al estado X2 está condicionada al evento $(a \wedge b) \vee c$, entonces la función de transición de estados se define como:

$$f(X1, (a \wedge b) \vee c) \rightarrow X2$$

Regla 6. La función de transición de estados t se forma con los eventos prohibidos, los estados y los estados prohibidos[###]. La función t se define de la misma manera que la función f, pero ésta es bidireccional por lo que es necesario definirla en dos sentidos, aunque cabe mencionar que el evento relacionado en la función de regreso es que no esté presente ningún evento prohibido para el estado prohibido correspondiente. Así,

[###] Definido en la Regla 11 del punto 3.2.2.6

por ejemplo, si el estado de a=1 ó b=1 es prohibido en el estado X3, entonces la función t asociada es:

$$t(\mathrm{X3}, a \vee b) \rightarrow \mathrm{X}^{1}3$$
$$t(\mathrm{X}^{1}3, \&(a \vee b)) \rightarrow \mathrm{X3}$$

Regla 7. Las salidas del proceso no se tomarán en cuenta en el modelo autómata discreto, ya que sólo se analizará la ocurrencia de eventos y esto simplifica el diagrama.

Para el proceso de estampado de latas, se obtienen los siguientes conjuntos y funciones de transición:

- E = {START, S1, S2, SA1, SR1, SA2, SR2, FI1, FI2, T}
  T = {S1, S2, FI1, FI2, &S2, &S1, (SA1∧SR1), (SA2∧SR2)}
- X = {X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X^{1}0, X^{1}1, X^{1}2, X^{1}3, X104, X^{1}5, X^{1}6, X^{1}7, X^{1}8, X^{1}9, X^{1}10}
- X_0 = {X0}
- Las funciones de transición de estados f definidas por:
  - f(X0, START ∧ SA1 ∧ SA2) → X1
  - f(X0, START ∧ SA1 ∧ SA2) → X2
  - f(X1, ↑S1) → X3
  - f(X2, ↑S2) → X4
  - f(X3, SA1) → X5
  - f(X4, SA2) → X6
  - f(X5, FI1) → X7
  - f(X6, SA2) → X8
  - f(X7, ↓S1) → X9

- $f(X8, \downarrow S2) \rightarrow X10$
- $f(X9, SR1) \rightarrow X1$
- $f(X10, SR2) \rightarrow X2$

Así como las funciones de transición de eventos prohibidos t definidas por:

- $t(X0, S1 \vee S2 \vee (SA1 \wedge SR1) \vee (SA2 \wedge SR2) \vee FI1 \vee FI2) \rightarrow X^10$
- $t(X^10, \&(S1 \vee S2 \vee (SA1 \wedge SR1) \vee (SA2 \wedge SR2) \vee FI1 \vee FI2)) \rightarrow X0$
- $t(X1, (SA1 \wedge SR1)) \rightarrow X^11$
- $t(X^11, \&(SA1 \wedge SR1)) \rightarrow X1$
- $t(X2, (SA2 \wedge SR2)) \rightarrow X^12$
- $t(X^12, \&(SA2 \wedge SR2)) \rightarrow X2$
- $t(X3, (SA1 \wedge SR1) \vee \&S1) \rightarrow X^13$
- $t(X^13, \&((SA1 \wedge SR1) \vee \&S1)) \rightarrow X3$
- $t(X4, (SA2 \wedge SR2) \vee \&S2) \rightarrow X^14$
- $t(X^14, \&((SA2 \wedge SR2) \vee \&S2)) \rightarrow X4$
- $t(X5, (SA1 \wedge SR1)(1) \vee \&S1) \rightarrow X^15$
- $t(X^15, \&((SA1 \wedge SR1)(1) \vee \&S1)) \rightarrow X5$
- $t(X6, (SA2 \wedge SR2)(1) \vee \&S2) \rightarrow X^16$
- $t(X^16, \&((SA2 \wedge SR2)(1) \vee \&S2)) \rightarrow X6$
- $t(X7, (SA1 \wedge SR1)) \rightarrow X^17$
- $t(X^17, \&(SA1 \wedge SR1)) \rightarrow X7$
- $t(X8, (SA2 \wedge SR2)) \rightarrow X^18$
- $t(X^18, \&(SA2 \wedge SR2)) \rightarrow X8$
- $t(X9, (SA1 \wedge SR1)) \rightarrow X^19$
- $t(X^19, \&(SA1 \wedge SR1)) \rightarrow X9$
- $t(X10, (SA2 \wedge SR2)) \rightarrow X^110$
- $t(X110, \&(SA2 \wedge SR2)) \rightarrow X10$

3.2.3.2 Obtener el diagrama del modelo autómata discreto del proceso a partir de los conjuntos y funciones definidas.

Para poder llevar a cabo el análisis computacional en la siguiente etapa de esta metodología es necesario obtener el diagrama del modelo autómata discreto definido en el punto anterior, así, se proponen las siguientes reglas:

Regla 1. Los estados son representados por círculos.

Regla 2. El(los) estado(s) inicial(es) se representa(n) por un círculo doble con una flecha entrando.

Regla 3. Las transiciones entre estados son representadas por flechas unidireccionales que indican el flujo de ésta con una etiqueta la cual representa el evento(condición) asociado a la transición. Esta regla aplica para las funciones *f* y *t*. Así, por ejemplo, en la Figura 3.9 se muestra la transición definida por $f(X1, (a \wedge b) \vee c) \rightarrow X2$, donde X1 es un estado inicial.

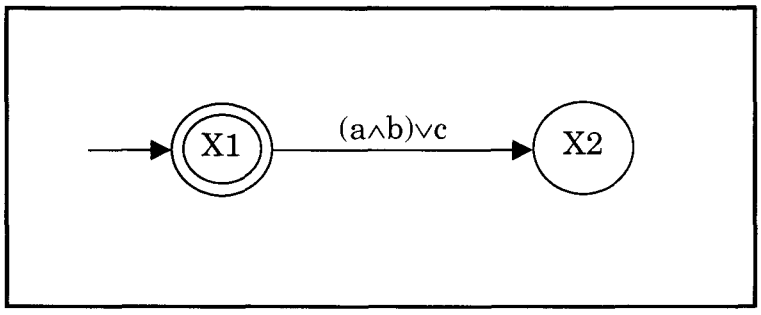


Figura 3.9. Dibujo para la transición $f(X1, (a \wedge b) \vee c) \rightarrow X2$.

Así, siguiendo lo establecido en la etapa 2, la Figura 3.10 muestra el diagrama del modelo autómata discreto diseñado para el proceso de estampado de latas.

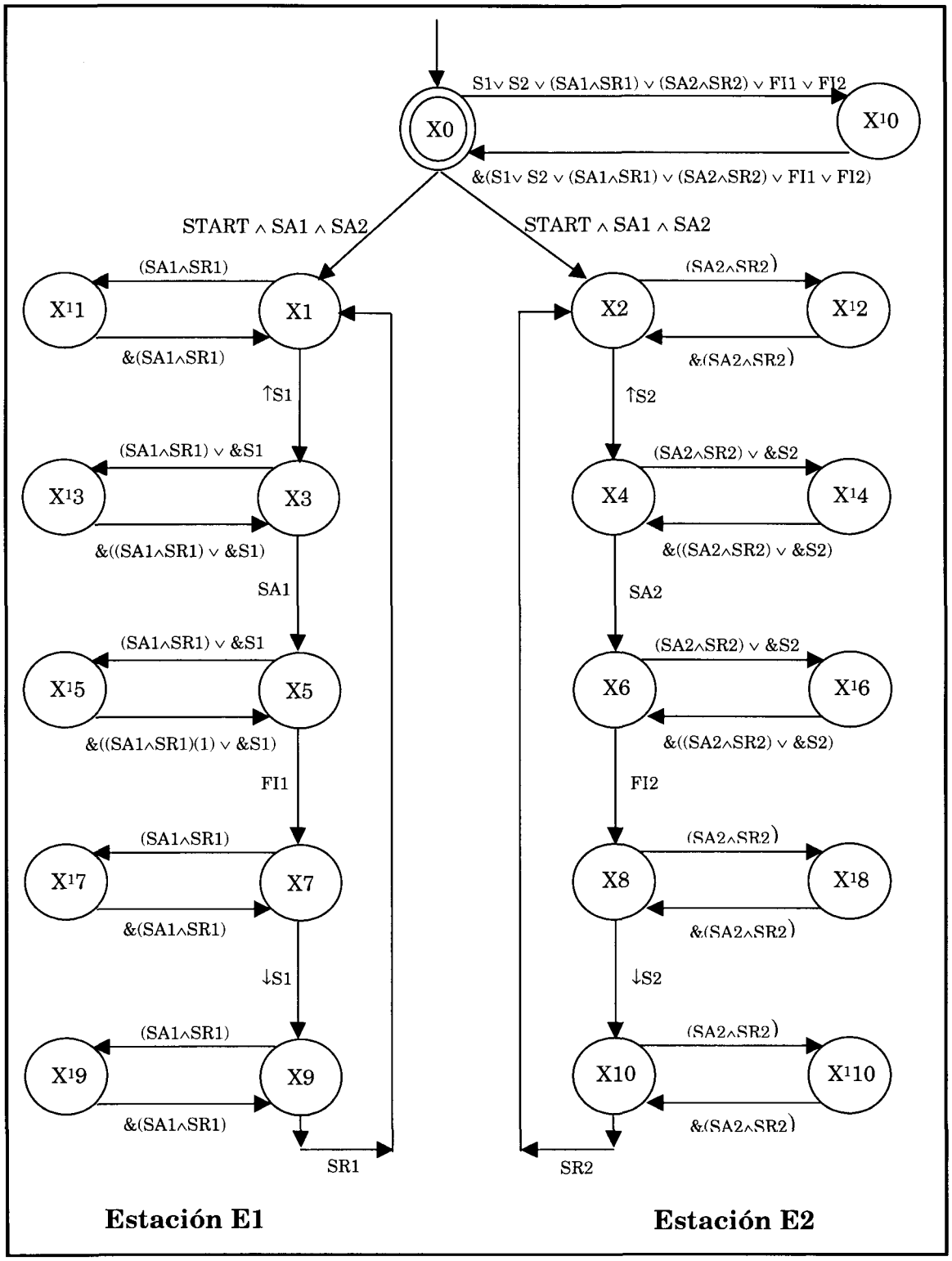


Figura 3.10. Diagrama del modelo autómata discreto para el proceso de estampado de latas.

3.2.4 Tercera Etapa: Realizar el análisis del modelo autómata discreto para el proceso.

En esta etapa se analiza el modelo autómata discreto obtenido en la etapa anterior validando, como parte de su buen funcionamiento:

- Los candados de fin (deadlocks), es decir que no exista una secuencia de eventos que lleve al SED a un estado del que no pueda salir.
- La viveza (liveness), es decir que no exista una secuencia de eventos que lleve al SED a oscilar entre dos estados únicamente.
- El funcionamiento del sistema acuerdo con las especificaciones.

Para llevar a cabo este análisis ya existen herramientas computacionales tal como Model Vision Studium (MVS)® [12], por lo que en esta etapa se utilizará este software como herramienta de análisis.

Para llevar a cabo el análisis del modelo autómata discreto a través de una simulación computacional se proponen las siguientes reglas:

Regla 1. Si existen dos funciones similares e independientes sólo se realizará el análisis correspondiente a una de ellas, de los resultados obtenidos para ésta se podrá evaluar el desempeño de las otras.

Regla 2. En la simulación del modelo autómata discreto no se considera el comportamiento ante la ocurrencia de eventos prohibidos, ya que un estado prohibido es, por definición§§§, un estado paralelo al estado no prohibido.

Regla 3. Las propiedades a valorar en el modelo autómata discreto son:

- Que el autómata no presente deadlocks.
- Que el autómata esté vivo (liveness).

§§§ Se definió en la segunda etapa de la metodología RIMAnI, punto 3.2.3, Regla 6.

Regla 4. En caso que los resultados del análisis del modelo autómata discreto fueran desfavorables, es decir que presente deadlocks o falta de viveza (liveness), será necesario empezar desde la primer etapa para redefinir el modelo.

Para el autómata creado en el proceso de estampado de latas la estación E1 y E2 son similares e independientes, así aplicando la Regla 1, sólo se realizará el análisis correspondiente para la estación E2, y de los resultados obtenidos para ésta se podrán evaluar las propiedades de E1 y E2. En la Figura 3.11 se muestran los autómatas que se diseñaron en el software MVS para simular el comportamiento del autómata de la estación E2..

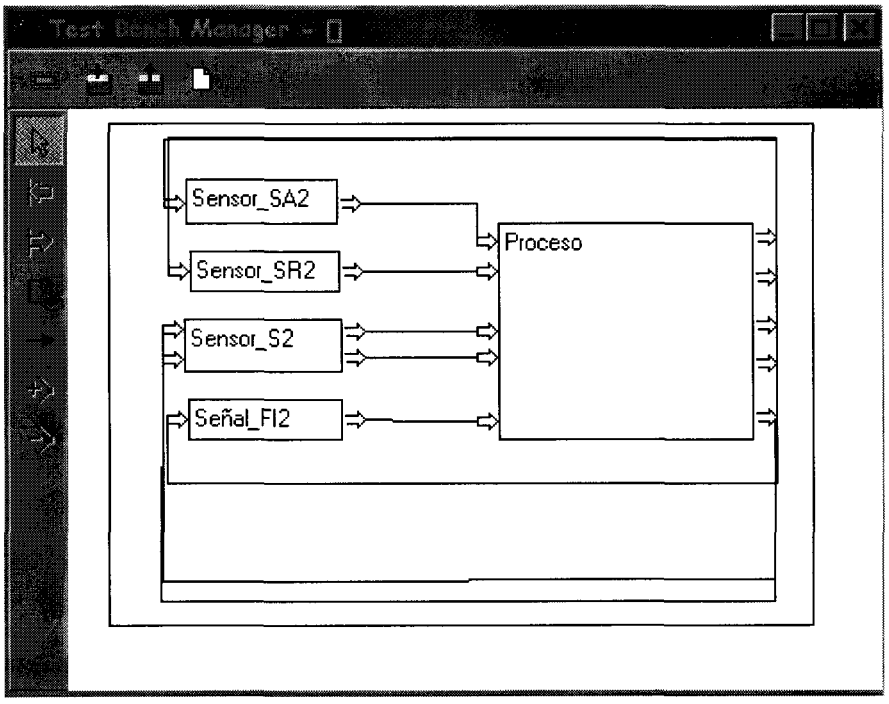


Figura 3.11. Diagrama de autómatas para la estación E2 en MVS.

Cada uno de los cinco autómatas contiene un diagrama de estados que define su comportamiento dinámico. En la Figura 3.12 se muestra el correspondiente al bloque: "Proceso", el cual simula el comportamiento del

proceso, es decir, los estados: X2, X4, X6, X8 y X10. Dado que que el software no permite simular la ocurrencia de eventos, como lo son los sensores y señales del proceso, fue necesario crear un modelo autómata para cada uno de éstos (SA2, SR2, S2 y FI2), así en la Figura 3.13 se muestra el modelo autómata correspondiente a SA2, en la Figura 3.14 el de SR2, en la Figura 3.15 el de S2**** y en la Figura 3.16 el de FI2.

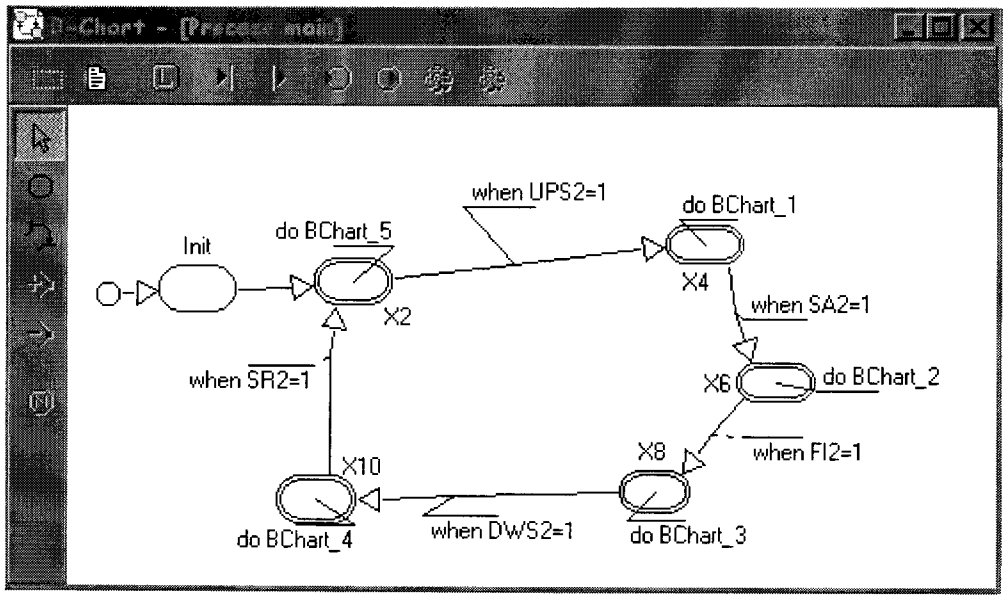


Figura 3.12. Diagrama de estados para el autómata "Proceso".

**** La señal ↑S2 se denomina UPS2 y la señal ↓S2 se denomina DWS2.

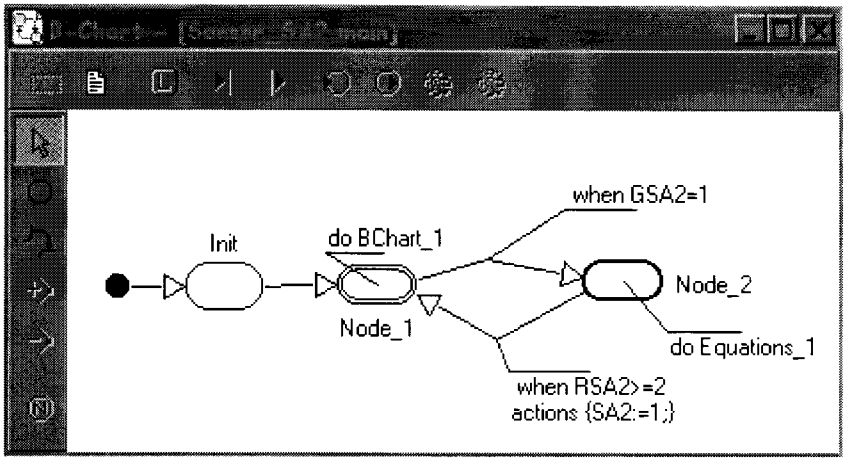


Figura 3.13. Diagrama de estados para el autómata del sensor SA2.

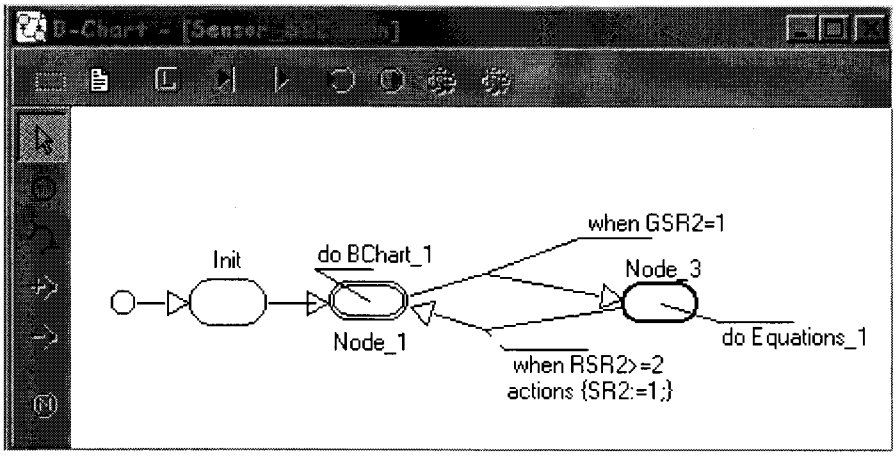


Figura 3.14. Diagrama de estados para el autómata del sensor SR2.

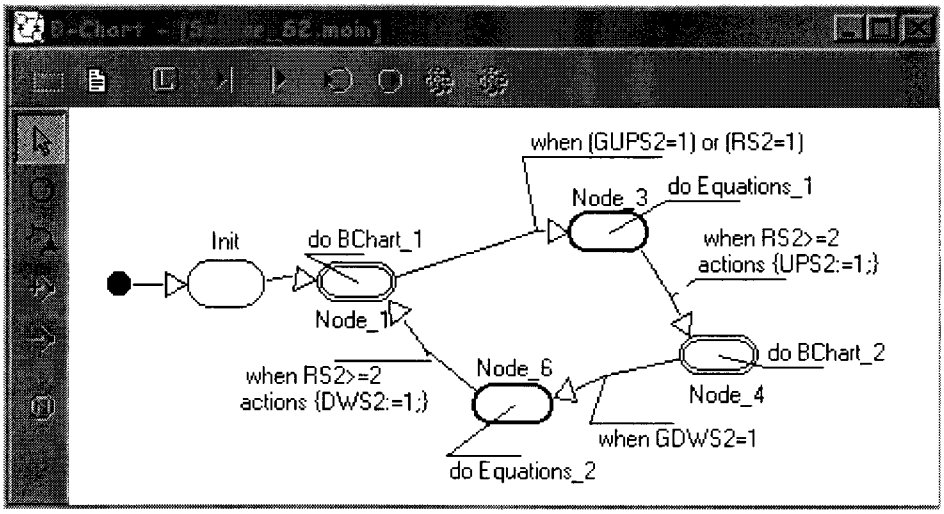


Figura 3.15. Diagrama de estados para el autómata del sensor S2.

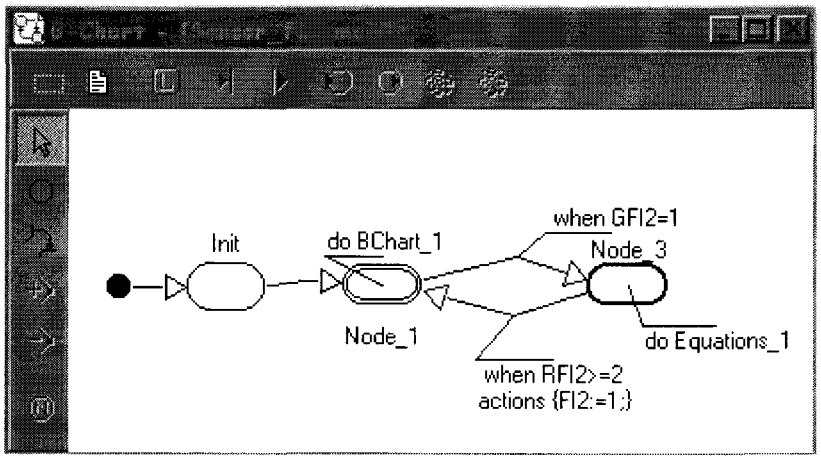


Figura 3.16. Diagrama de estados para el autómata del sensor FI2.

Finalmente en la Figura 3.17 se muestra la pantalla de simulación con los diagramas de tiempo correspondientes al modelo autómata discreto para la estación E2. El proceso "genera" una señal (salidas, diagrama inferior) para que el autómata de cada sensor o señal actúe y mande la entrada correspondiente al proceso. Así por ejemplo, en el tiempo t=1 seg. el proceso manda generar la señal SA2, la cual ocurre como entrada a éste dos segundos después es decir, en t=3s.

En los diagramas de tiempo mostrados en la Figura 3.17 se puede apreciar como los estados tienen un comportamiento dinámico periódico debido a la ocurrencia similar de sus eventos(salidas de los autómatas de los sensores), por lo que el modelo autómata discreto no presenta deadlocks ya que la ocurrencia de los eventos permitidos no bloquea al sistema. La viveza se evalúa desde el punto de vista que la ocurrencia de los eventos permitidos no hace que el sistema oscile entre dos estados, sino que todos son alcanzados y activados según las condiciones entre estados, por lo que la viveza es aceptable. Así, el modelo obtenido en la segunda etapa de esta metodología puede ser utilizado como base para diseñar el GRAFCET que se implementará en el PLC.

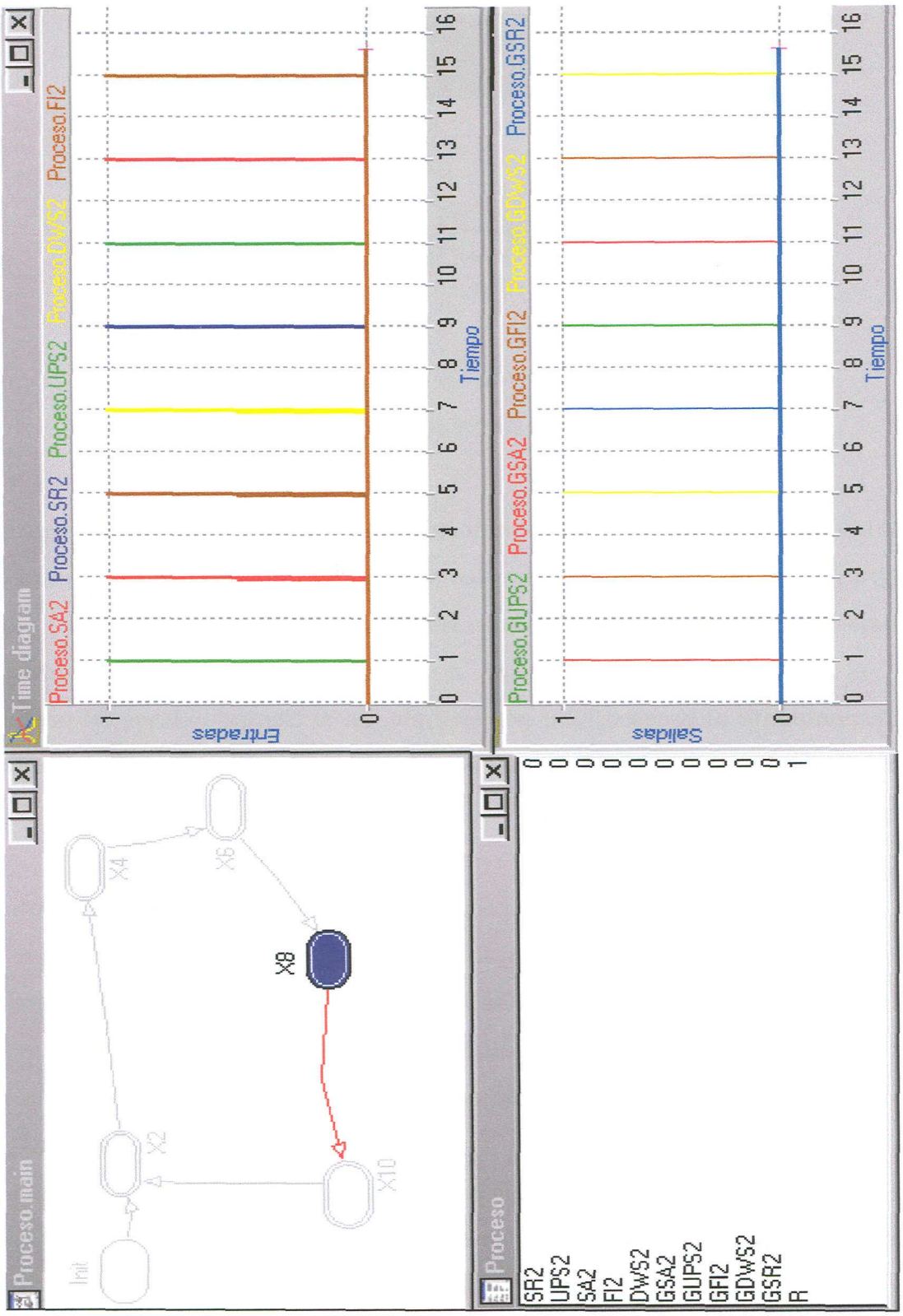


Figura 3.17. Pantalla de simulación para el modelo autómata de la estación E2.

3.2.5 Cuarta Etapa: Obtener la estructura GRAFCET para la implementación en un PLC.

El uso de la estructura GRAFCET como lenguaje de programación en algunos PLC representa una gran facilidad en el análisis de los SED, ya que es posible conocer inmediatamente el estado del sistema y relacionar las acciones y eventos con éste.

Algunas ventajas que presenta el uso del lenguaje de programación GRAFCET son las siguientes:

- Es posible congelar la ejecución del programa fácilmente.
- Es posible tener control sobre lo que se desea realizar después de una falla en el suministro eléctrico, o bien cualquier falla en la que se pierda control sobre el proceso (arranque en frío o caliente).
- El monitoreo del estado del proceso y las salidas es simple.

Básicamente esta cuarta etapa consiste en pasar del modelo autómata discreto al lenguaje de programación GRAFCET, en la cual se sugiere definir cada una de las siguientes tres secciones, para obtener una mejor interacción con el programa de control:

- ***Sección Preliminar***[††††]. Sección donde se programa la interacción del usuario con el programa (arranque, paro y pausa del programa).
- ***Sección Grafcet***. Sección donde se define la dinámica (secuencia) del proceso.
- ***Sección Posterior***. Sección donde se relacionan los estados de la dinámica del proceso con las salidas del PLC.

[††††] Cabe mencionar que el funcionamiento del diagrama Grafcet no se ve afectado si no se incluye la sección preliminar, sólo se pierde la interacción con el programa GRAFCET.

3.2.5.1 Construcción de la sección preliminar, a partir de las condiciones del sistema y del diseñador.

Para la construcción de la sección preliminar, a partir de las condiciones del sistema y del diseñador se hace necesario definir los siguientes tres puntos:

- La condición de inicialización (se refiere a la que hace que el programa se posicione en la(s) etapa(s) inicial(es))y de paro (se refiere a la que hará que la ejecución del programa se detenga) de GRAFCET.
- La condición de pausa de GRAFCET en caso de existir. En este paso es necesario definir la condición de SET (encendido y sostenimiento) y de RST (apagado) de la acción de pausa.
- La acción (inicialización, paro o pausa) que seguirá el re-arranque del programa, con datos en memoria (arranque en caliente) o sin datos en memoria (arranque en frío)

3.2.5.1.1 ***Condición de Inicialización.*** Se construye un diagrama lógico de escalera con las condiciones de inicialización como contactos, las cuales habilitarán al bit de inicialización de GRAFCET (arranque del programa). Cabe mencionar que este bit puede cambiar en los diferentes PLC's, por lo que sólo se describirá como el bit IG. En la Figura 3.18, S_1 ó S_2 representa la condición de inicialización de GRAFCET.

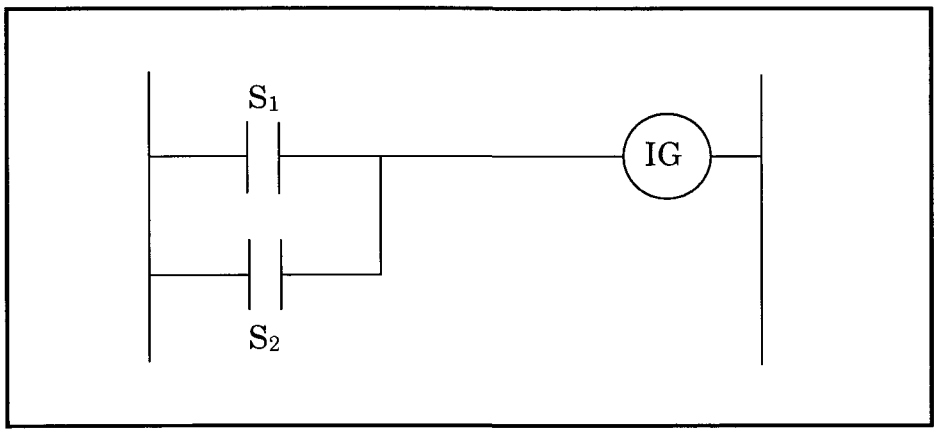


Figura 3.18. Ejemplo de construcción de la sección preliminar para el bit IG.

3.2.5.1.2 ***Condición de Paro.*** Añadir al diagrama anterior la condición de paro de GRAFCET (paro de programa), es decir agregar la condición con contactos, para habilitar el bit de paro. Al igual que en el punto anterior, este bit se representará como PG. En la Figura 3.19, S_3 y S_4, representa la condición de paro de GRAFCET.

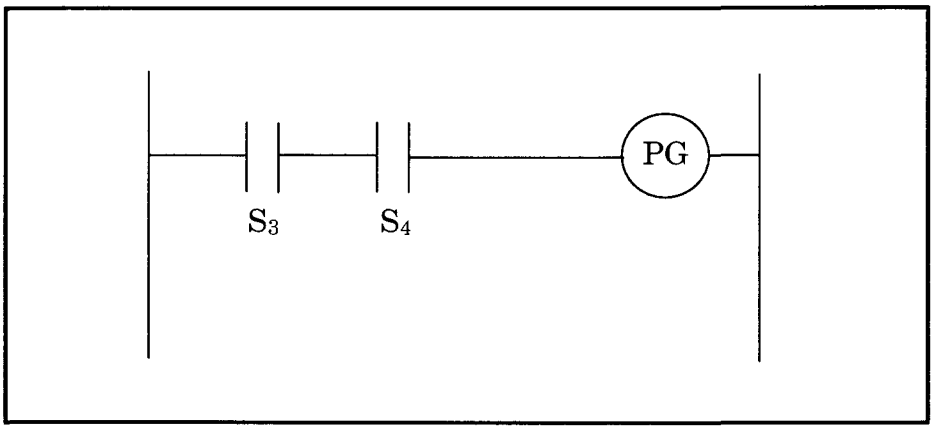


Figura 3.19. Ejemplo de construcción de la sección preliminar para el bit PG.

3.2.5.1.3 ***Condición de Pausa.*** Añadir al diagrama de la sección anterior, la condición que debe ocasionar que el GRAFCET entre en modo de pausa. Al igual que en las secciones anteriores este bit será representado como CG. Se debe agregar también la condición para apagar el bit CG (salir del modo pausa). En la Figura 3.20, S_4 y S_5 son la condición de SET, mientras que S_7 es la condición de RST del bit CG de pausa.

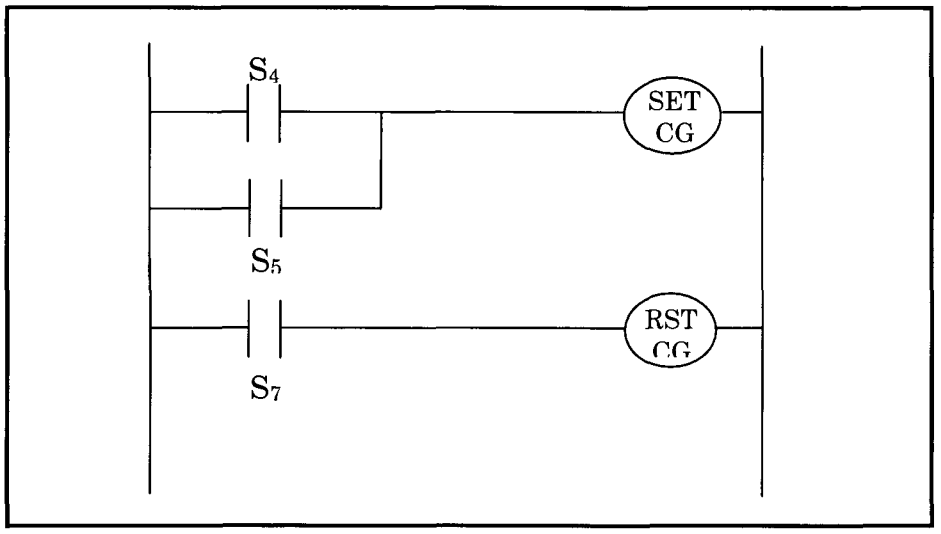


Figura 3.20. Ejemplo de construcción de la sección preliminar para el bit CG.

3.2.5.1.4 ***Condición de Rearranque.*** Añadir al diagrama anterior, la condición de arranque en frío y/o arranque en caliente, dependiendo de la acción que se desea para cada condición. Por ejemplo, si en el arranque en caliente se desea que el programa entre en el estado de pausa, entonces la condición de pausa para la Figura 3.20 será:

Condición de pausa = S_4 ó S_5 ó Arranque en caliente

Para el ejemplo del proceso de estampado de latas, es necesario establecer las condiciones de inicialización, paro, pausa, reestablecimiento de pausa, arranque en frío y arranque en caliente, las cuales son:

- La inicialización será llevada a cabo con los botones PAUSA y R presionados simultáneamente.
- El paro se activará con el botón STOP.
- El SET para la pausa será llevado a cabo con el botón PAUSA.

- El RST (o reestablecimiento) de la pausa será llevado a cabo con el botón R.
- El arranque en frío detendrá la ejecución del programa.
- El arranque en caliente provocará que el programa entre en el modo pausa.

Con las condiciones anteriores es posible realizar el diseño del diagrama escalera para la sección preliminar, el cual se muestra en la Figura 3.21. El arranque en frío se expresa como AF, y el arranque en caliente como AC.

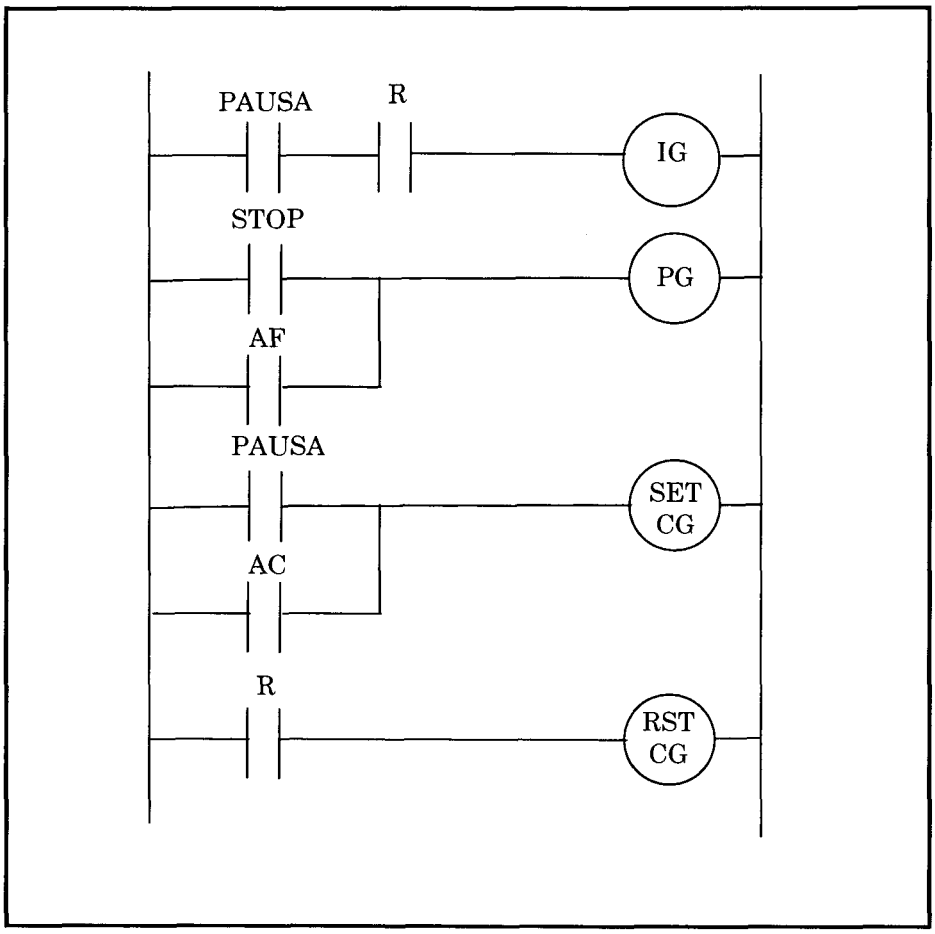


Figura 3.21. Diagrama escalera para la sección preliminar del proceso de estampado de latas.

3.2.5.2 Construcción de la sección Grafcet, a partir del modelo autómata discreto.

La sección Grafcet permite modelar el comportamiento dinámico del proceso, es decir, en esta sección se le podrá dar seguimiento al estado actual y a la evolución (secuencia) provocada por los eventos discretos del proceso.

Para construir la sección Grafcet es necesario:

1) Se define cada estado del conjunto X del autómata discreto como una etapa X del diagrama Grafcet. Así, un estado no prohibido en el modelo autómata será una etapa no prohibida en el diagrama Grafcet y lo mismo aplica para los estados prohibidos.

2) Basándonos en las condiciones iniciales definidas desde la primera etapa de la metodología, se definen estados iniciales del conjunto X_0 como etapas iniciales X del diagrama Grafcet. Como ya se había mencionado, una de las ventajas de los diagramas Grafcet es la versatilidad de poder ejecutar varios diagramas Grafcet a la vez, por lo que es posible declarar varias etapas iniciales. Por ejemplo, si X0, X1 y X2 son estados iniciales en el autómata discreto, entonces se deben iniciar tres diagramas Grafcet, como se muestra en la Figura 3.22.

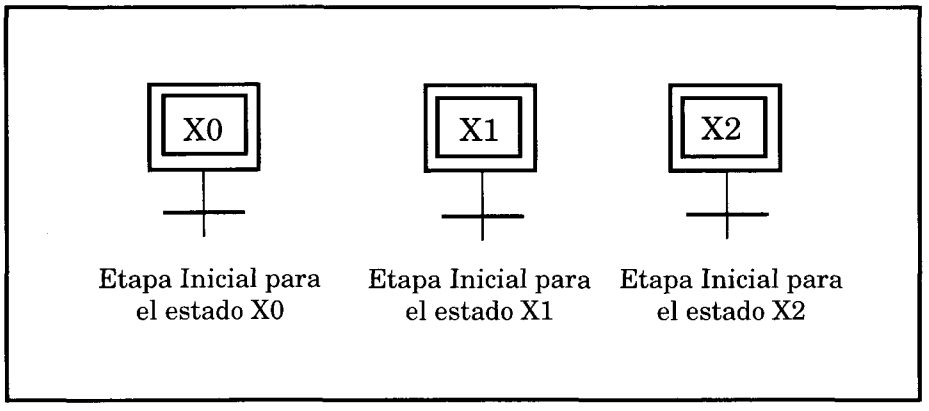


Figura 3.22. Ejemplo de construcción del diagrama Grafcet para tres estados iniciales.

3) Una vez que se han dibujado las etapas iniciales dentro del diagrama Grafcet es necesario añadir las etapas posteriores, basándose en el modelo autómata discreto.

Dado que las etapas iniciales coinciden con estados iniciales del modelo autómata, se proponen las siguientes reglas para realizar esta transformación:

Regla 1. Cada transición del modelo autómata discreto es una transición en el diagrama Grafcet.

Regla 2. Cada estado del conjunto *X* del modelo autómata discreto es una etapa X del diagrama Grafcet.

Regla 3. El número correspondiente en el diagrama Grafcet para cada estado no prohibido es el mismo que el asignado en el modelo autómata discreto.

Regla 4. El número correspondiente a cada estado prohibido X^ik en el diagrama Grafcet se define utilizando el algoritmo de la Figura 3.23, considerando, como anteriormente se definió, que se tienen *n* estados no prohibidos.

Regla 5. La transición entre dos etapas no prohibidas del diagrama Grafcet es definida por la función de transición *f* del modelo autómata discreto. Así por ejemplo, la función $f(X_n, a) \rightarrow X_m$, define que la transición entre la etapa X_n y X_m está dada por la ocurrencia del evento "a".

Regla 6. La transición entre una etapa prohibida y una no prohibida del diagrama Grafcet es definida por la función de transición *t* del modelo autómata discreto. Así por ejemplo, la función $t(X_n, (a \vee b)) \rightarrow X_m$, define que la transición entre la etapa X_n y X_m está dada por la ocurrencia del evento "a" o el "b".

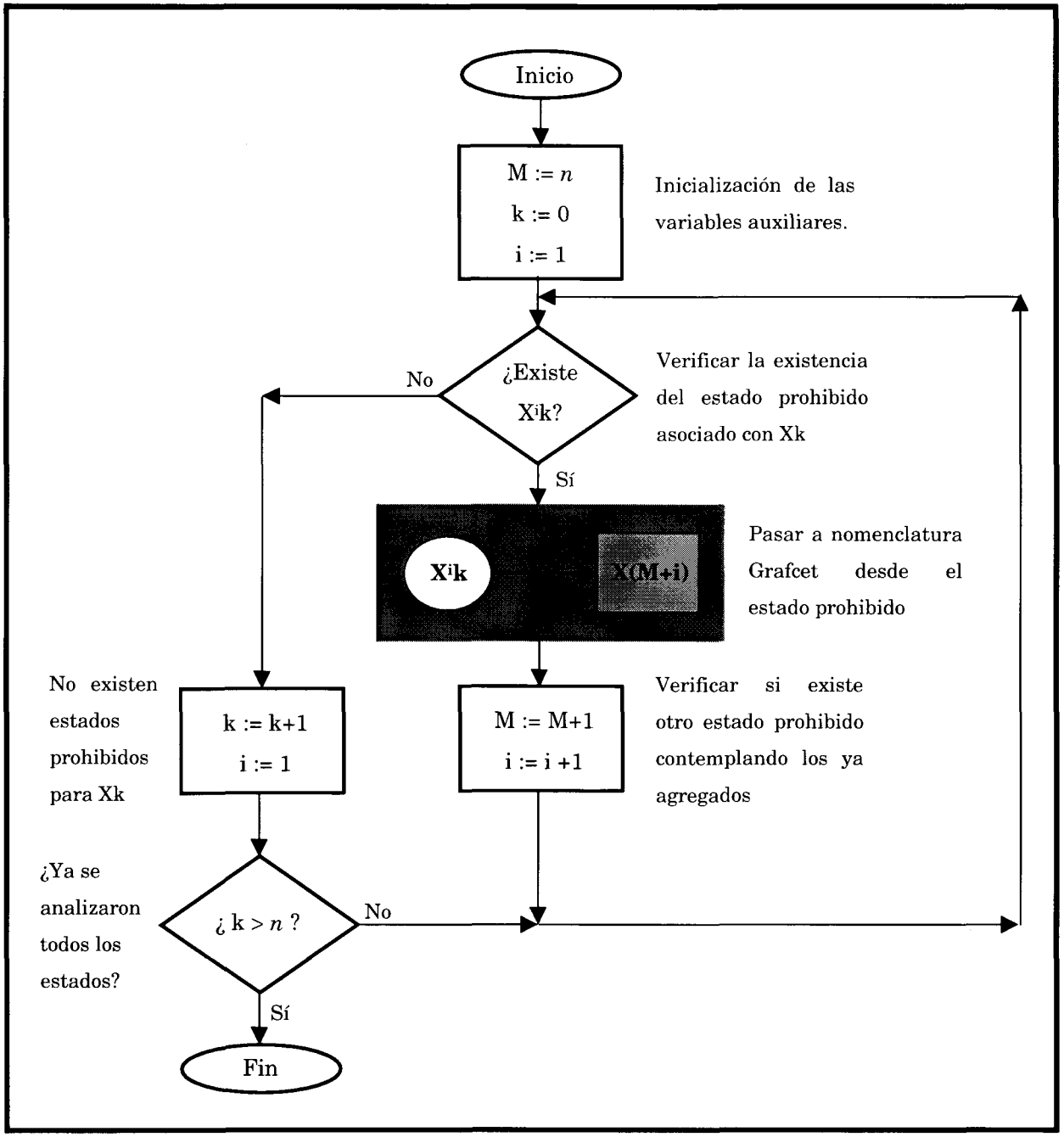


Figura 3.23. Algoritmo para encontrar el número correspondiente a cada estado prohibido en el diagrama Grafcet.

Regla 7. En el caso de que existan retrasos en los sensores de entrada, es decir, que se desee actuar hasta un cierto tiempo después de que se

presentó la señal, este tiempo será representado en la transición como un retraso.

Regla 8. Las salidas relacionadas con cada estado del modelo autómata discreto serán tratadas en la sección posterior de GRAFCET.

En la Figura 3.24, se muestra la transformación de un modelo autómata discreto con estados prohibidos y no prohibidos a un diagrama Grafcet.

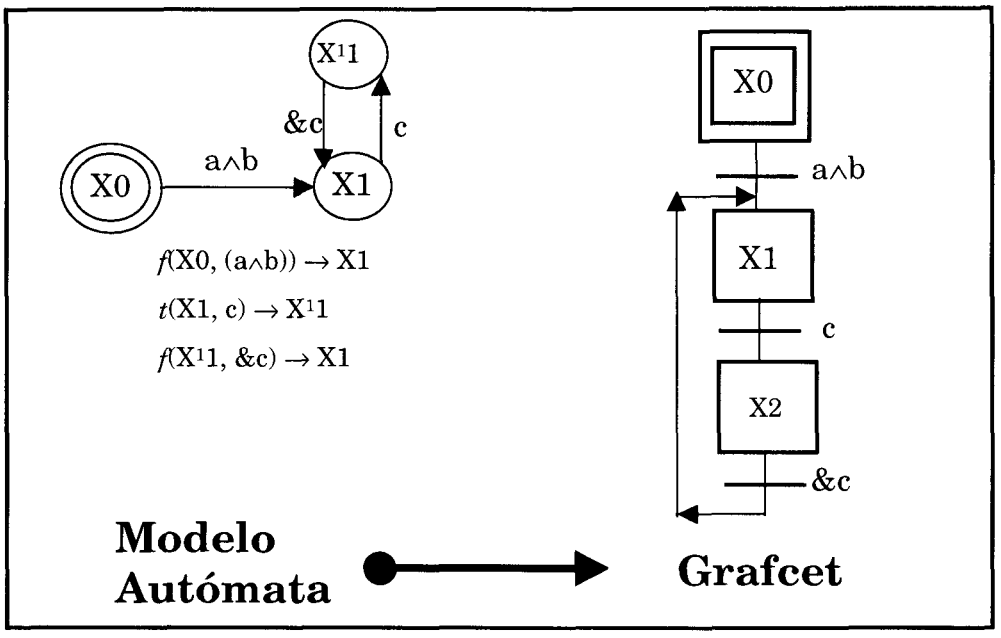


Figura 3.24. Ejemplo de construcción del diagrama Grafcet.

En esta figura el estado X0 es un estado inicial, X1 es un estado no prohibido y X¹1 es un estado prohibido. La aplicación del algoritmo de la Regla 3 arroja como resultado que el estado prohibido X¹1 se identifique como X2 en el diagrama Grafcet.

Para el modelo autómata discreto del proceso de estampado de latas mostrado en la Figura 3.10, se pueden aplicar estas reglas y obtener el diagrama Grafcet correspondiente. Este diagrama se muestra en la Figura 3.25.

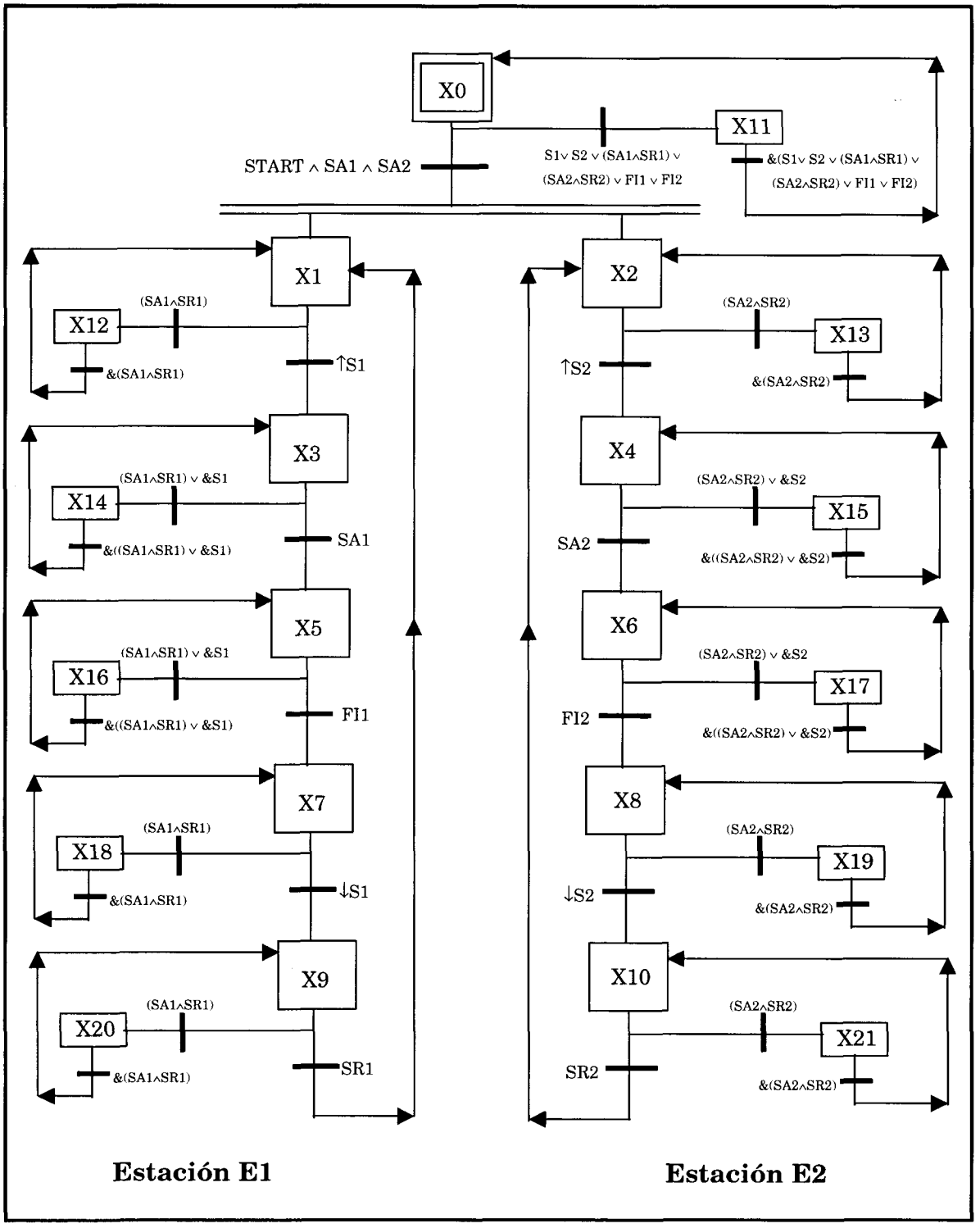


Figura 3.25. Sección Grafcet para el proceso de estampado de latas.

3.2.5.3 Construcción de la sección posterior a partir de la información del proceso.

En esta última parte se construirá el diagrama escalera para que las salidas del proceso sean encendidas conforme su relación con los estados del modelo autómata discreto.

Para una correcta construcción de la sección posterior de la estructura GRAFCET se deben seguir los siguientes pasos:

1) Con base a la información proporcionada en el punto 3.2.2 de esta metodología (Tabla 3.1 y Tabla 3.3), se obtiene una relación entre las salidas del proceso y los estados del modelo autómata discreto, los cuales ya fueron asignados con las etapas del diagrama Grafcet. Para que esta operación resulte un poco más sencilla al momento de programar, primero se llenará una tabla, como la que se muestra en la Tabla 3.5. En ésta, la columna de la izquierda muestra el número de salida del PLC, la columna de en medio una pequeña descripción de la salida, , mientras que la columna de la derecha la función lógica booleana para cada salida con la indicación del tiempo de retraso en caso de que exista. Esta ecuación lógica booleana debe estar representada en función de las etapas de Grafcet y/o alguna otra entrada,.

Salidas del PLC	*Descripción de la Salida del PLC*	*Función lógica booleana*
Salida 1	Arranque del Motor 1	$X_1 \vee X_4$
Salida 2	Arranque de la Válvula 7	$X_3 \vee X_5$

Tabla 3.5. Tabla para relacionar las salidas del PLC con las etapas de la sección Grafcet y sus respectivos retrasos.

En la Tabla 3.5, se muestra también un ejemplo de cómo se llenaría esta tabla. Por ejemplo, la salida 1 (Arranque del motor 1) del PLC, se enciende cuando las etapas X_1 o X_4 de Grafcet están activas.

Este tipo de representación resulta de mucha utilidad para programar la sección posterior cuando el número de salidas en el PLC es muy grande.

2) Basándose en la información proporcionada por la Tabla 3.5 se construye el diagrama escalera de la sección posterior. Se interpretan las etapas del diagrama Grafcet como contactos que encienden una salida, con sus respectivos retrasos. La Figura 3.26 muestra la sección posterior correspondiente a la Tabla 3.5.

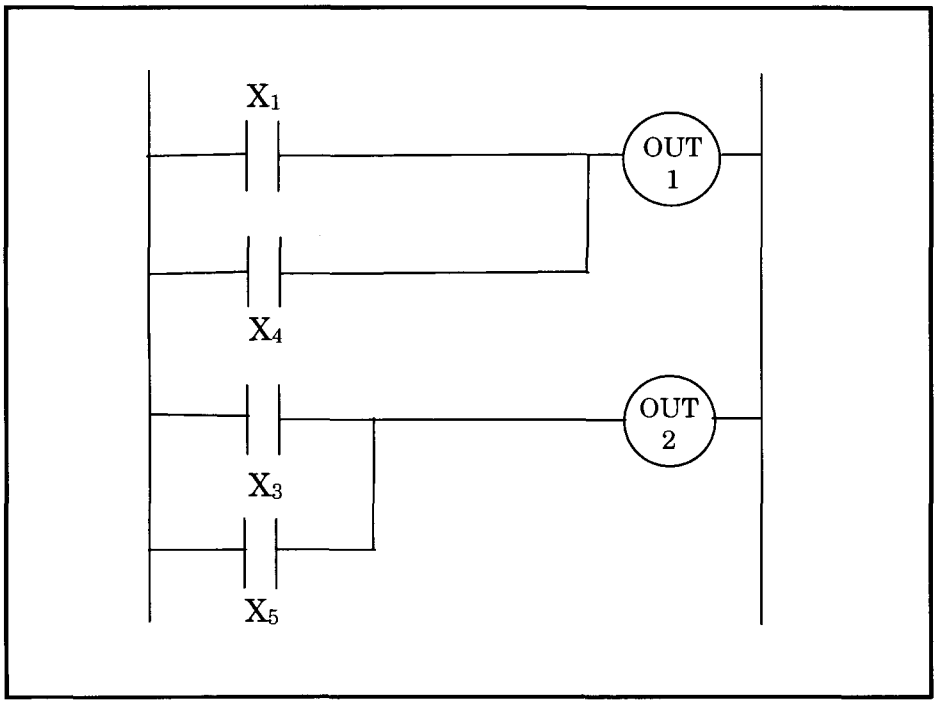


Figura 3.26. Ejemplo de construcción de la sección posterior basándose en la Tabla 3.5

Para el ejemplo del proceso de estampado de latas, es necesario formar la tabla que relacione las salidas del PLC con cada una de las etapas del diagrama Grafcet de la Figura 3.25. Así, tomando la información de la Tabla 3.2 y la Tabla 3.4 se obtiene la Tabla 3.6.

Salidas del PLC	***Descripción de la Salida del PLC***	***Función lógica booleana***
Salida 1	SET del Motor	X1
Salida 1	RST del Motor	X0
Salida 2	Activar parador 1 (A1)	X0 ∨ X3
Salida 3	Retroceder parador 1 (R1)	X1 ∨ X9
Salida 4	Activar parador 2 (A2)	X0 ∨ X4
Salida 5	Retroceder parador 2 (R2)	X2 ∨ X10
Salida 6	SET magneto 1 (M1)	X3
Salida 6	RST magneto 1 (M1)	X7
Salida 7	SET magneto 2 (M2)	X4
Salida 7	RST magneto 2 (M2)	X8
Salida 8	Habilitar impresora 1 (IMP1)	X5
Salida 9	Habilitar impresora 2 (IMP2)	X6
Salida 10	Sirena de alarmas (AS1)	X11 ∨ X12 ∨ X13 ∨ X14 ∨ X15 ∨ X16 ∨ X17 ∨ X18 ∨ X19 ∨ X20 ∨ X21

Tabla 3.6. Relación de las salidas del PLC y las etapas del diagrama Grafcet para el proceso de estampado de latas.

Basándose en la Tabla 3.6 se forma el diagrama lógico de escalera de la sección posterior, el cual se muestra en la Figura 3.27 y Figura 3.28. En este diagrama se utilizaron los nombres relacionados en la descripción de la salida para representarlas en cada escalón del diagrama escalera.

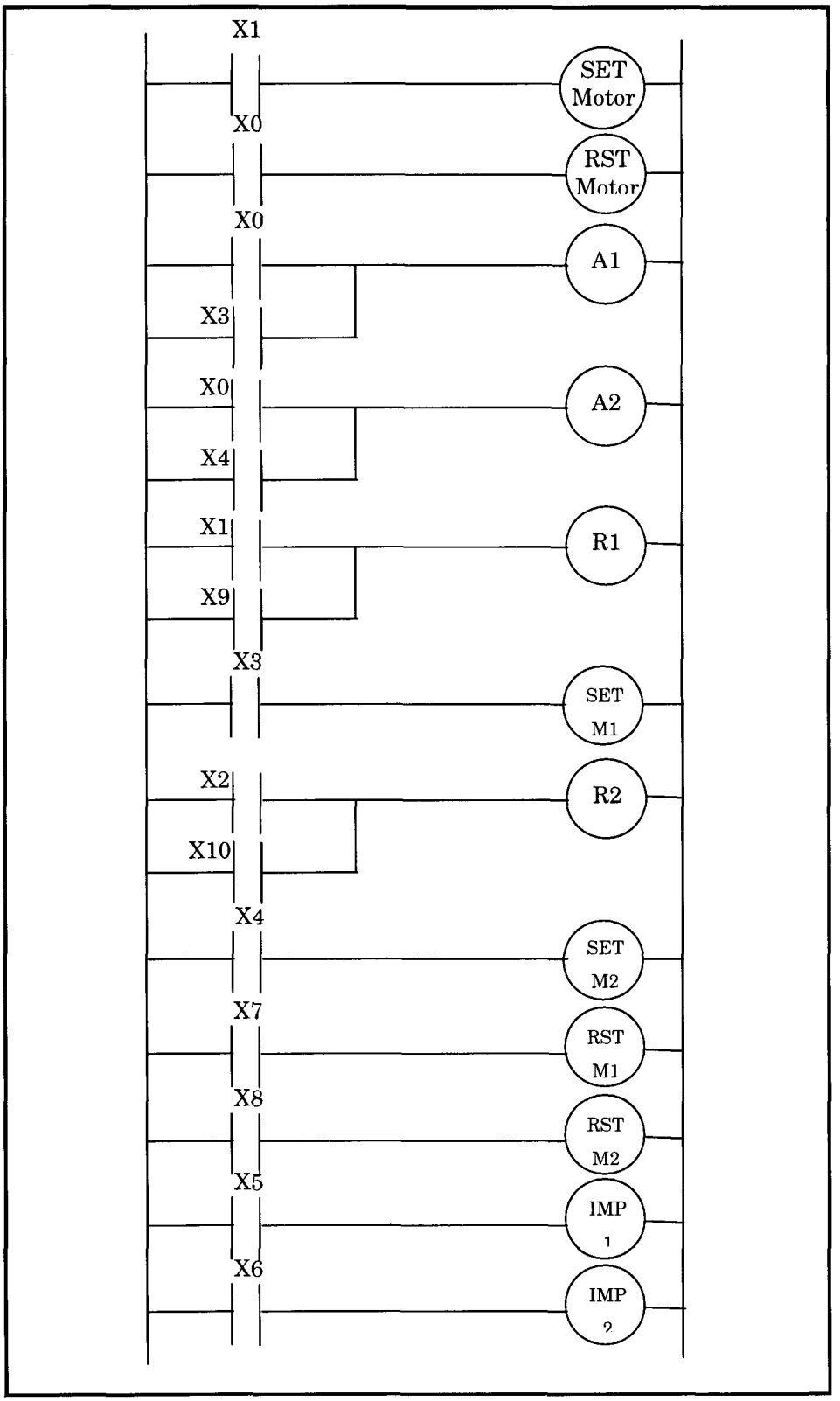


Figura 3.27. Diagrama lógico de escalera de la sección posterior para el proceso de estampado de latas.

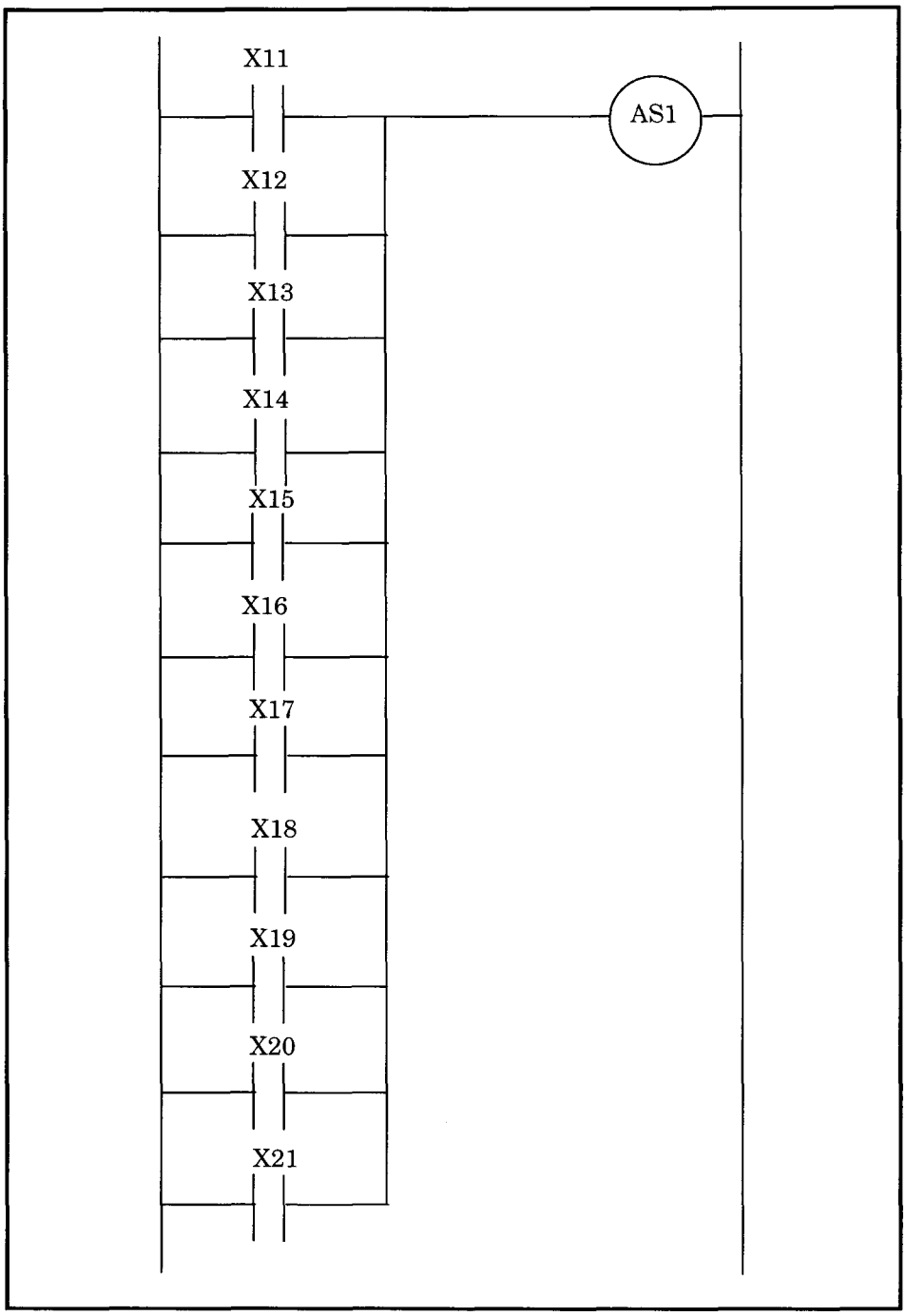


Figura 3.28. Diagrama lógico de escalera de la sección posterior para el proceso de estampado de latas en sus estados prohibidos.

3) Si se desea que al presionar el botón de paro de GRAFCET las salidas sostenidas se apaguen, es necesario añadir al diagrama lógico de escalera

de la sección posterior la condición de paro de GRAFCET activando la función RST de estas salidas. Así, por ejemplo, si Z1 y Z2 son salidas sostenidas y P es la condición de paro de GRAFCET, el diagrama de la Figura 3.29 muestra como al presionar P se activará un RST a las salidas Z1 y Z2. Cabe mencionar que esta acción no es deseable en algunos casos, esto dependerá de la decisión del experto del proceso.

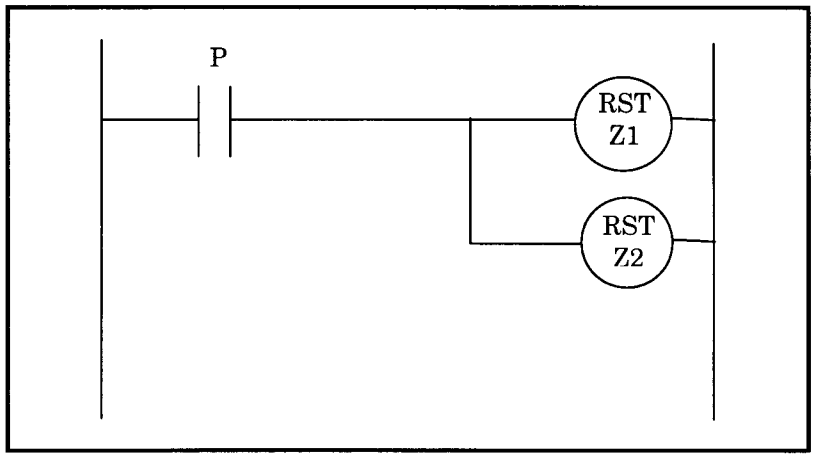


Figura 3.29. Diagrama escalera de la condición P activando el RST de Z1 y Z2.

Para el proceso de estampado de latas esta acción si es deseable así, el diagrama de la Figura 3.30 se adhiere a la sección posterior. La condición de paro definida en este diagrama es la misma que se definió previamente y las únicas salidas sostenidas son:

- El motor de la banda (M).
- El magneto de la estación 1 (M1).
- El magneto de la estación 2 (M2).

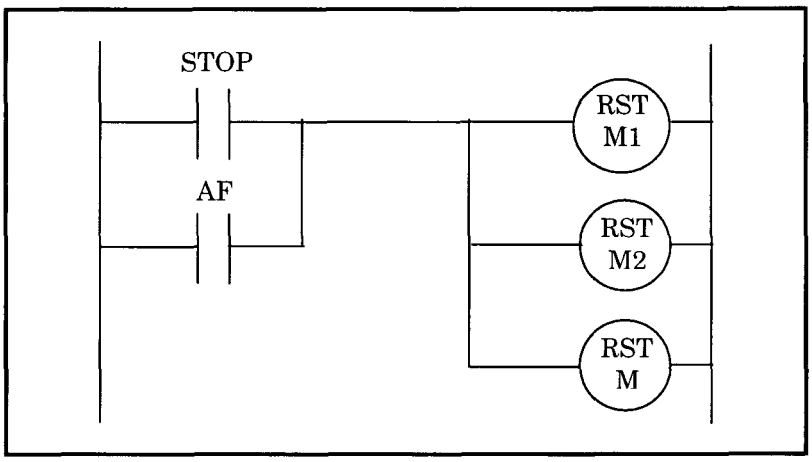


Figura 3.30. Diagrama de escalera para apagar las salidas sostenidas con la condición de paro de GRAFCET para el proceso de estampado de latas.

3.2.6 Comentarios finales.

En el Anexo 4 se muestra un resumen de los resultados obtenidos por etapa para el proceso de estampado de latas utilizando la metodología RIMAnI en GRAFCET.

La metodología RIMAnI en GRAFCET ofrece las siguientes ventajas:

- Considera la ocurrencia de eventos prohibidos en el proceso.
- Permite tomar acciones específicas para cada estado ya sea prohibido o no prohibido.
- Recolecta de una manera más sencilla la información de cada una de las etapas del proceso.
- Divide las estaciones o funciones independientes desde el primer paso.

Desde el punto de vista de control del proceso las metodologías IPTG y RIMAnI en GRAFCET ofrecen una solución al problema de controlar un proceso previamente instrumentado para funcionar automáticamente.

La definición plenamente del modelo autómata discreto en la segunda etapa de la metodología RIMAnI permite a cualquier ingeniero de control conocer la composición y el comportamiento dinámico del programa que se está ejecutando en el PLC, y en consecuencia poder realizar un análisis rápido y eficiente del proceso.

Debido a que el programa de control obtenido para el PLC está totalmente definido como un modelo autómata discreto, las modificaciones futuras a éste son sencillas de documentar, lo cual ofrece que siempre se posea la documentación actualizada del programa.

Capítulo 4

Pruebas Experimentales de las Metodologías IPTG y RIMAnI en GRAFCET

En este capítulo se llevarán a cabo pruebas experimentales en el controlador lógico programable TSX-3705 Modicon Telemecanique, éstas comprenden los resultados obtenidos para el proceso de estampado de latas, en los capítulos 2 y 3.

4.1 Introducción.

Los resultados obtenidos en los capítulos 2 y 3 de este trabajo de investigación pueden ser utilizados para controlar el proceso de estampado de latas propuesto para cada metodología(IPTG y RIMAnI). Para obtener una prueba contundente de la validez de éstos, se realizará la implementación en el controlador lógico programable(PLC) TSX-3705 de la marca Modicon-Telemecanique.

Este PLC puede ser programado en los siguientes lenguajes de programación:

- Diagrama escalera.
- Lista de instrucciones.
- Texto estructurado.
- GRAFCET.**

Así, se utilizará GRAFCET, ya que los resultados obtenidos de las metodologías IPTG y RIMAnI están precisamente en este lenguaje.

** Se manejará GRAFCET (con mayúsculas) como el lenguaje de programación usado en algunos controladores comerciales, y Grafcet (en minúsculas) como la herramienta de modelación de eventos discretos

4.2 Pruebas experimentales para los resultados obtenidos con la metodología IPTG en GRAFCET.

4.2.1 Descripción del controlador lógico programable TSX-3705.

En la Figura 4.1 se muestra el PLC TSX-3705 de la marca Modicon-Telemecanique. Las características eléctricas de este PLC son:

- Tarjeta de 16 entradas binarias y 12 salidas binarias.
- Voltaje de operación de 24 VCD para las entradas y salidas.
- Voltaje de alimentación de 127 VCA @ 60 Hz.
- Batería de respaldo tipo capacitiva.

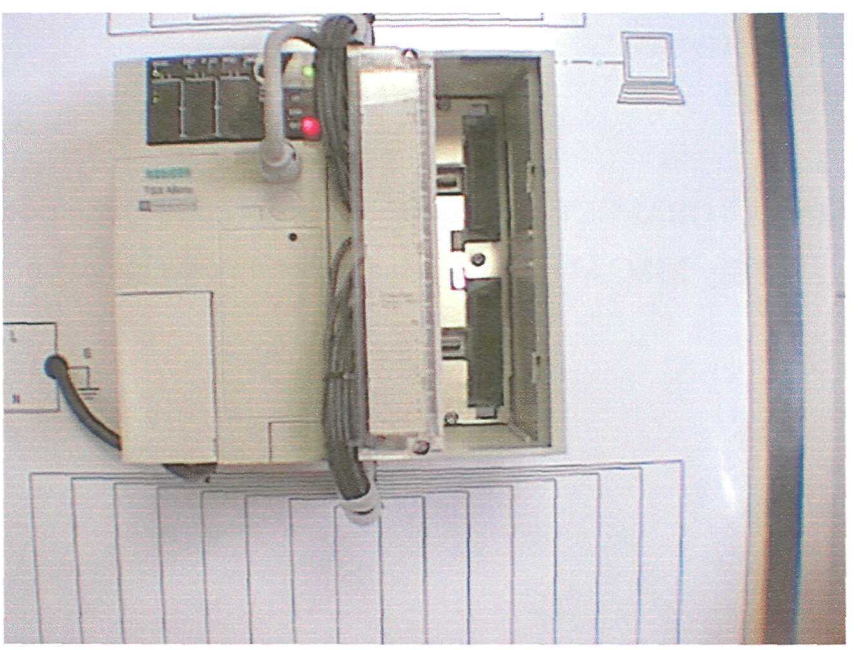


Figura 4.1. PLC TSX-3705 de la marca Modicon-Telemecanique.

El software que utiliza este PLC para llevar a cabo la programación es el PL7 Pro de la marca Modicon-Telemecanique, y sus requerimientos son los siguientes:

- Plataforma en Windows 95 ó 98.
- Capacidad de 200 Mb disponibles en disco duro.
- Memoria RAM de 32 Mb.
- Puerto serial disponible.

Este PLC junto con su computadora se encuentran instalados en un tablero de automatismos lógicos de los laboratorios del Departamento de Mecatrónica y Automatización en el Instituto Tecnológico y de Estudios Superiores Monterrey Campus Monterrey. Por lo que se tramitó el permiso correspondiente para poder utilizar este tablero.

4.2.2 Pruebas experimentales de los resultados obtenidos de la metodología IPTG en GRAFCET.

En el Anexo 2 se expone el listado del programa para las tres secciones, preliminar, Grafcet y posterior.

La Figura 4.2 muestra el tablero de pruebas con las conexiones eléctricas correspondientes.

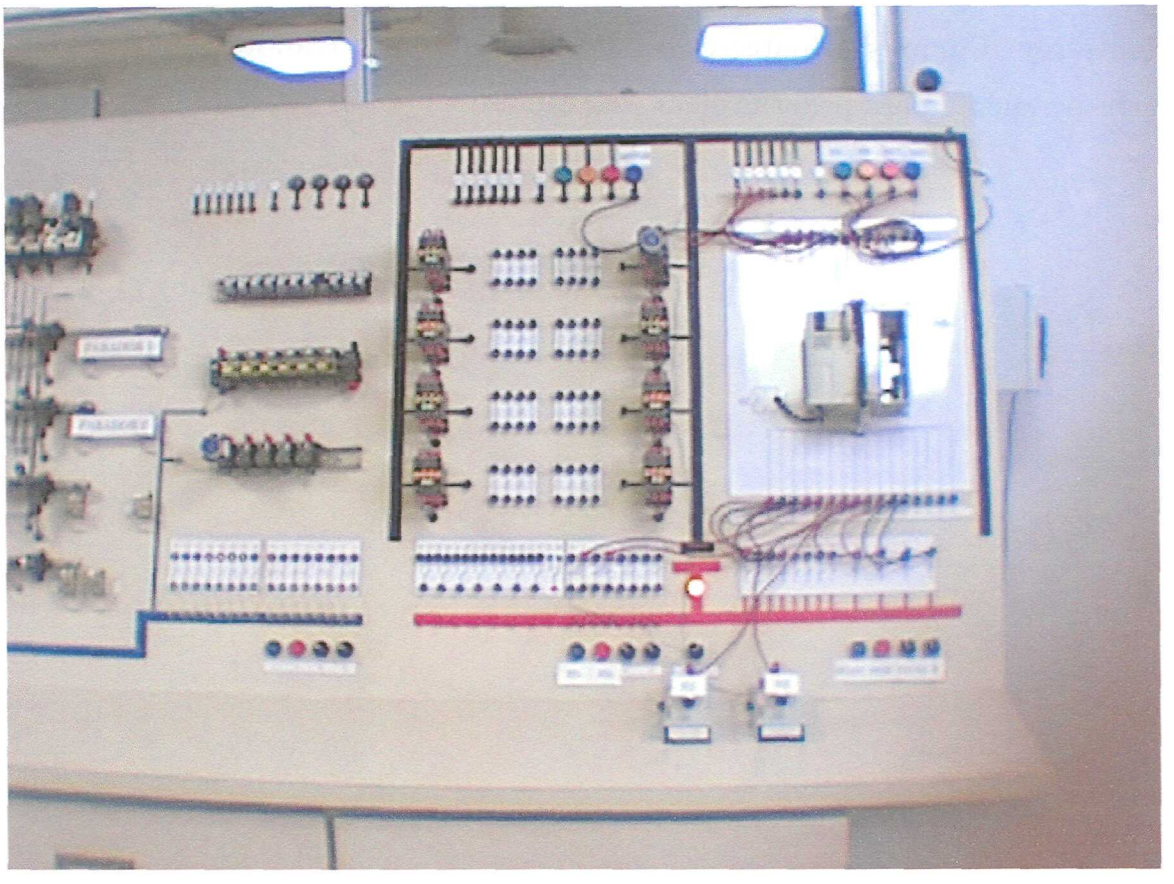


Figura 4.2. Tablero de pruebas con las conexiones eléctricas correspondientes.

La simulación de las señales de entrada FI1 y FI2 se llevó a cabo con dos botones pulsadores, ya que no se cuenta con las impresoras del proceso de estampado de latas. Además, el motor, los magnetos y las impresoras se simularon con focos.

Después de realizar la corrida del programa en el PLC, se concluye:

- Para realizar la programación se tiene un trabajo previo por lo que ésta es más estructurada.
- La programación está documentada.
- El programa controló al sistema conforme a las especificaciones requeridas.
- La relación entre los estados y las salidas del PLC está plenamente documentada.
- El análisis del sistema resulta más sencillo desde el punto de vista que es posible conocer fácilmente el estado del sistema.

En la Figura 4.3 se muestra una foto del proceso de estampado de latas con una lata en la estación 1, también se puede apreciar en la parte superior el encendido de los focos correspondientes al motor, al magneto M1 y a la impresora IMP1.

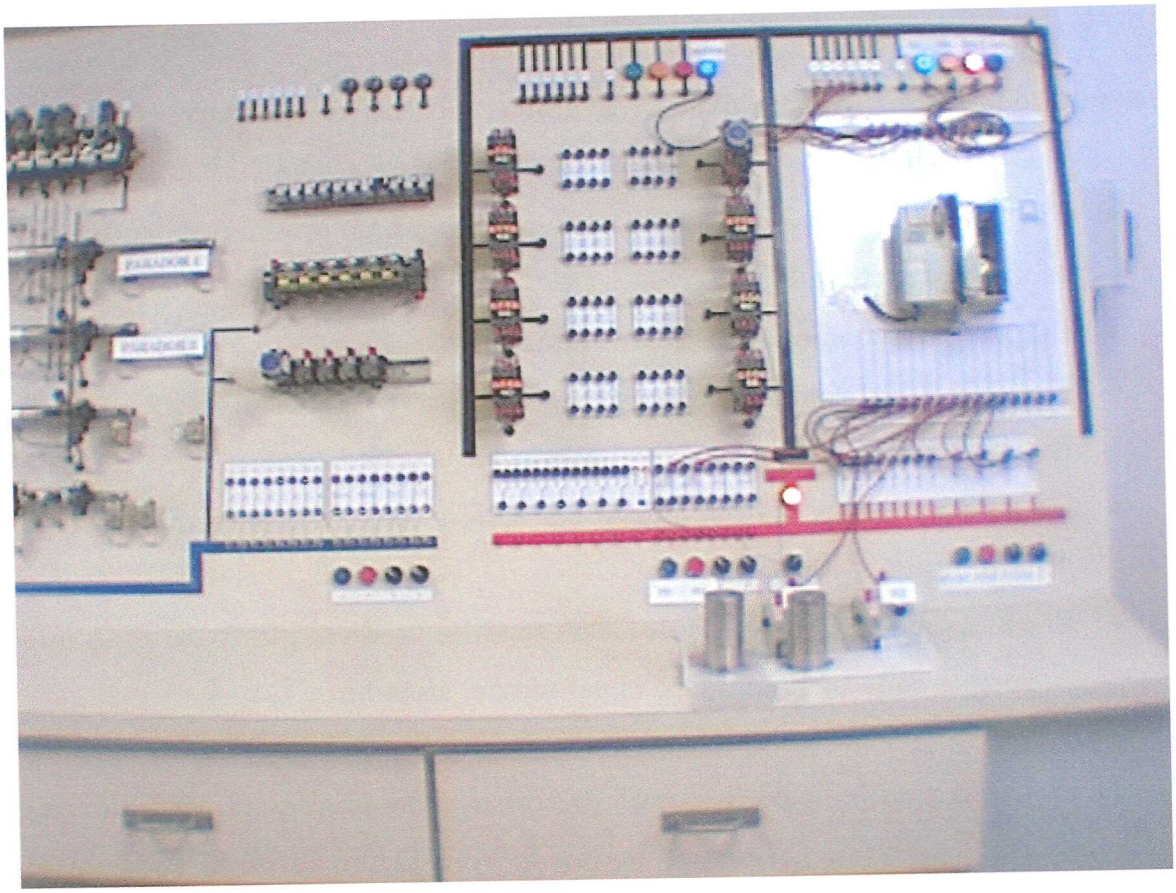


Figura 4.3. Foto del proceso con una lata en la estación 1.

4.3 Pruebas experimentales para los resultados obtenidos con la metodología RIMAnI en GRAFCET.

4.3.1 Pruebas experimentales de los resultados obtenidos de la metodología RIMAnI en GRAFCET.

En el Anexo 3 se expone el listado del programa para las tres secciones, preliminar, Grafcet y posterior.

La Figura 4.4 muestra el tablero de pruebas con las conexiones eléctricas correspondientes.

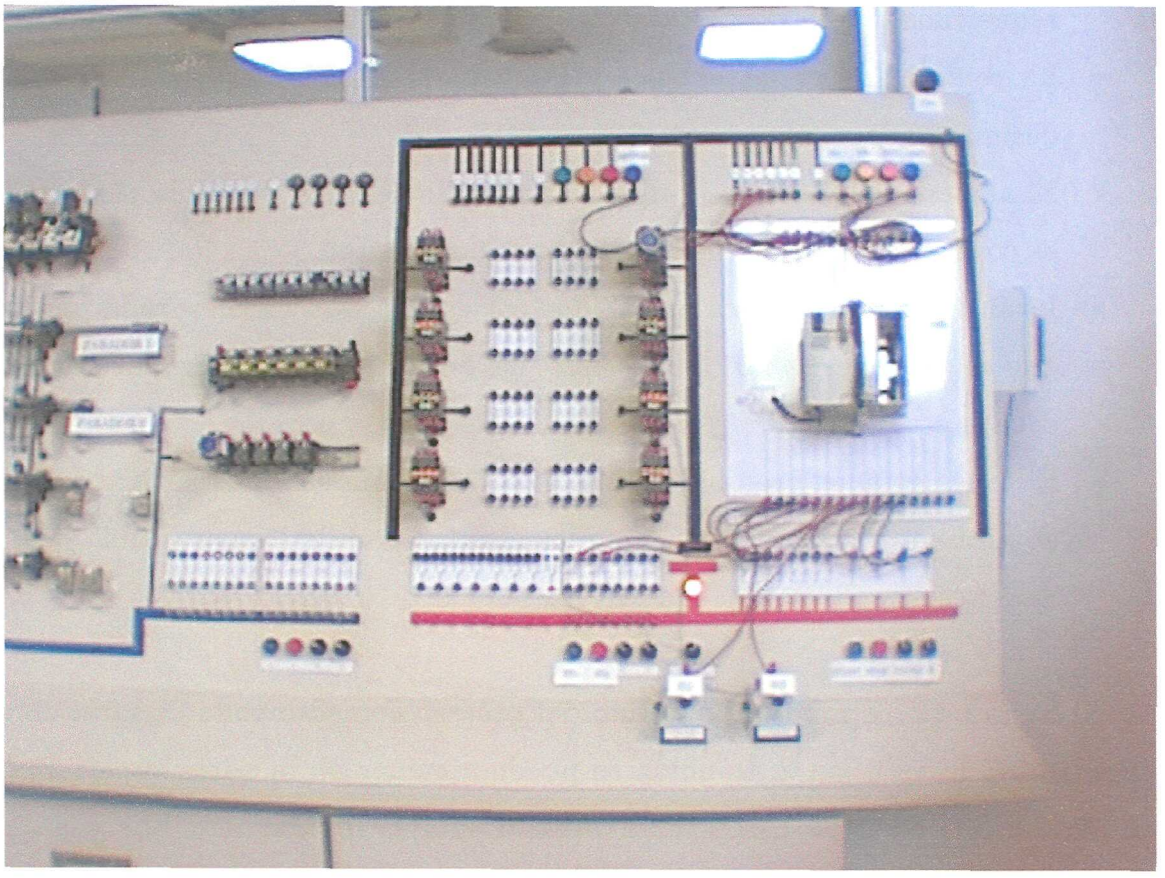


Figura 4.4. Tablero de pruebas con las conexiones eléctricas correspondientes.

La simulación de las señales de entrada FI1 y FI2 se llevó a cabo con dos botones pulsadores, ya que no se cuenta con las impresoras del proceso de estampado de latas. Además, el motor, los magnetos y las impresoras se simularon con focos.

Después de realizar la corrida del programa en el PLC, se concluye:

- Para realizar la programación se tiene un trabajo previo por lo que ésta es más estructurada.
- La programación y el diseño estás documentados.
- El programa controló al sistema conforme a las especificaciones requeridas.
- La relación entre los estados y las salidas del PLC está plenamente documentada.
- El análisis del sistema resulta más sencillo desde el punto de vista que es posible conocer fácilmente el estado del sistema.
- La inclusión de eventos prohibidos por estado está estructurada por lo que es posible tomar la decisión de que acción realizar para cada evento prohibido.
- El tener estados prohibidos hace que el sistema sea más seguro desde el punto de vista que contempla los eventos prohibidos por estado y las acciones a realizar.

En la Figura 4.5 se muestra una foto del proceso de estampado de latas con una lata en la estación 2, también se puede apreciar en la parte superior el encendido de los focos correspondientes al motor, al magneto M2 y a la impresora IMP2.

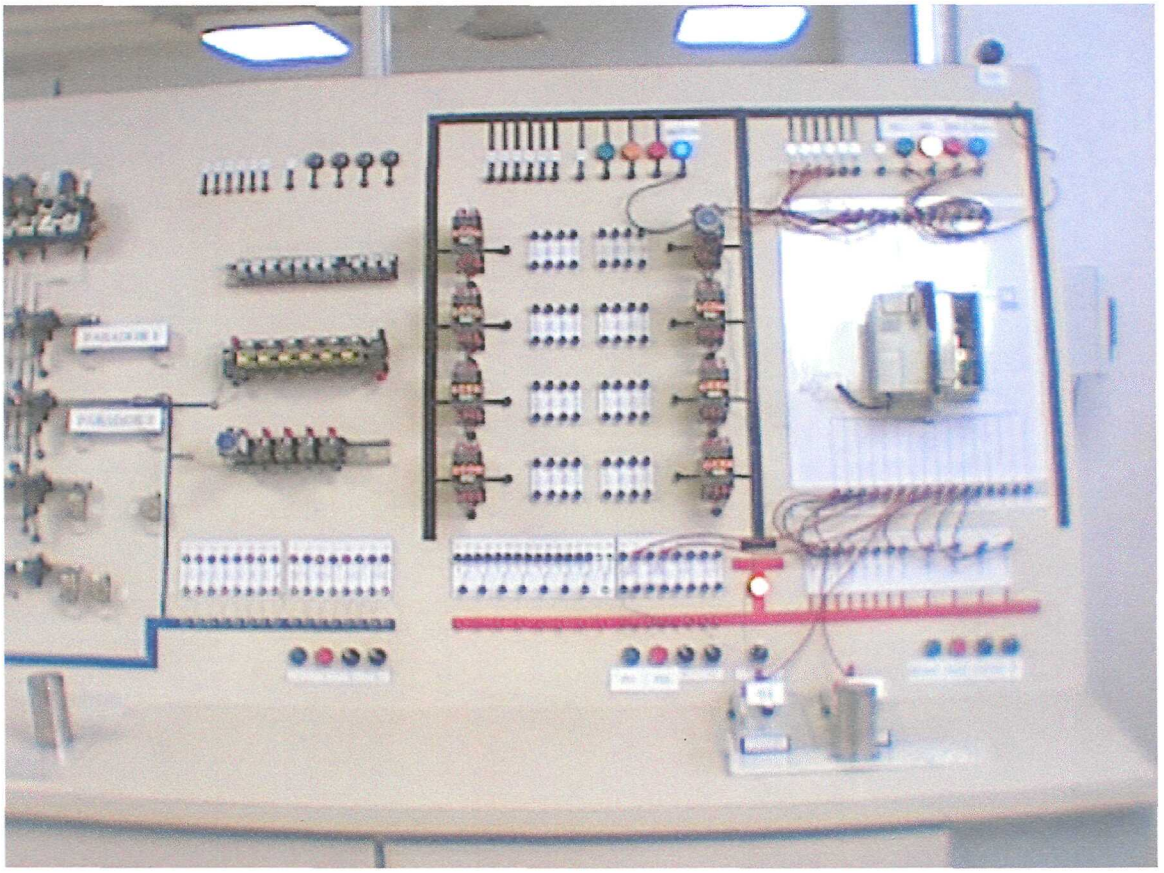


Figura 4.5. Foto del proceso con una lata en la estación 2.

Capítulo 5

Conclusiones y Perspectivas

En este capítulo se expondrán las conclusiones de este trabajo de tesis, así como las investigaciones futuras que pueden ser desarrolladas a partir de los resultados obtenidos.

5.1 Conclusiones.

La primera metodología, denominada IPTG en GRAFCET, es una modificación a la presentada en [14]. La segunda metodología, denominada RIMAnI en GRAFCET, es una aportación con parte académica y de experiencia industrial del autor.

Para la metodología IPTG en GRAFCET se concluye:

- Su uso se recomienda cuando el usuario esté familiarizado con diagramas IDEF0.
- La utilización del modelo IDEF0 permite la integración de todas las actividades en el proceso, así como las entradas y salidas de cada una de éstas.
- El diagrama de flujo de información permite la integración de sensores y simplifica la representación IDEF0 ya que no toma en cuenta las herramientas para realizar cada actividad.
- El modelo SPNC permite obtener una representación en Redes de Petri lo cual facilita el análisis del modelo antes de implementarlo.
- El modelo TPL incluye las salidas del proceso correspondientes a cada plaza del proceso, lo cual permite una visualización mejor de la relación plaza-salida(s).
- La utilización de GRAFCET sustituyendo al diagrama lógico de escalera representa gran ayuda cuando los sistemas son de complejidad mediana o grande.
- La aplicación de esta metodología resuelve la separación que existe entre el proceso instrumentado para trabajar automáticamente y el diseño e implementación de un programa para el controlador de eventos discretos (PLC) correspondiente.

- Ofrece una manera ordenada y sistemática de obtener el programa en un PLC que pueda ser programado en GRAFCET, para controlar adecuadamente el proceso correspondiente.
- Mediante la implementación de los resultados en un PLC real** se muestra como se aplica la metodología desde un punto de vista práctico.

Para la metodología RIMAnI en GRAFCET se concluye:

- La metodología ofrece una manera ordenada, sistemática y sencilla de obtener un modelo autómata discreto del comportamiento dinámico del proceso, para después convertirlo en un programa en GRAFCET.
- La proposición de reglas en cada etapa de la metodología le permite al experto del proceso construir cada una de éstas de una manera sistemática.
- La inclusión de eventos prohibidos por estado le permite al experto del proceso definir acciones correspondientes a cada uno de éstos.
- Con la explotación de la experiencia del experto del proceso el modelo autómata obtenido abarca de una manera completa el comportamiento dinámico del sistema.

En general para las dos metodologías se concluye que:

- El uso de GRAFCET como estructura de programación en un PLC (Controlador lógico programable) permite:

  - Definir las condiciones de inicialización, paro, pausa y de arranque en frío y caliente, para el programa de control.

** Ver el Capitulo 4, punto 4.2.2

- Un seguimiento sencillo del estado del sistema.
- Definir cada salida como condición del estado que la activa y/o otra condición asociada.

- La obtención de un diagrama de estados (diagrama Grafcet) del proceso, resulta de mucha utilidad para análisis y detección de fallas en el proceso.
- La tabla utilizada en las metodologías para la construcción de la sección posterior[††] puede ser utilizada posteriormente para conocer, de una manera sencilla y ordenada, cuáles salidas corresponden a cada estado del sistema.
- El autor recomienda el uso de la metodología RIMAnI ya que:
  - Sólo incluye cuatro etapas (una menos que IPTG).
  - Permite obtener un modelo del proceso aprovechando la experiencia del experto del proceso. Esto se realiza de una manera sistemática y simple.
  - Permite modelar e incluir en el programa de control de una manera sistemática, los eventos prohibidos por estado.
  - Permite tomar acciones en el programa de control, ante la ocurrencia de eventos prohibidos.
  - Permite obtener un modelo autómata discreto del proceso con el cual se pueden realizar diferentes tipos de análisis (deadlocks y liveness).
  - Las modificaciones futuras pueden hacerse sobre los resultados actuales.
- Las metodologías propuestas ofrecen un camino formal y sistemático para obtener un modelo del proceso y su programa de control.

[††] Capítulo 2, punto 2.2.6.3 y Capítulo 3, punto 3.2.5.3

5.2 Perspectivas.

Las perspectivas de este trabajo de investigación son:

- La inclusión de fallas y acciones temporizadas en función de la falla que se presente, tomando a ésta como un evento prohibido pero que no detenga la secuencia del sistema, sino que valore diferentes niveles para estos. Por ejemplo, evento no crítico, evento medianamente crítico, evento crítico.
- La clasificación de las fallas de acuerdo al tipo y tiempo de duración. Además de tomar acciones de seguridad dependiendo del tipo y duración de ésta.
- Las dos metodologías tienen como resultado final la estructura de un programa en GRAFCET, y podrían ser adecuadas para lograr obtener un programa en SFC (Sequential Function Chart).
- La realización de un software especializado que le permita al usuario diseñar sus programas sin necesidad de escribir y dibujar los diagramas de estado, las tablas, etc.
- Las dos metodologías propuestas aplican para sistemas de eventos discretos (SED), por lo que éstas pueden ser usadas como base para su aplicación en el caso de sistemas híbridos.

ANEXO 1

Tablas en blanco, utilizadas por la metodología IPTG en GRAFCET.

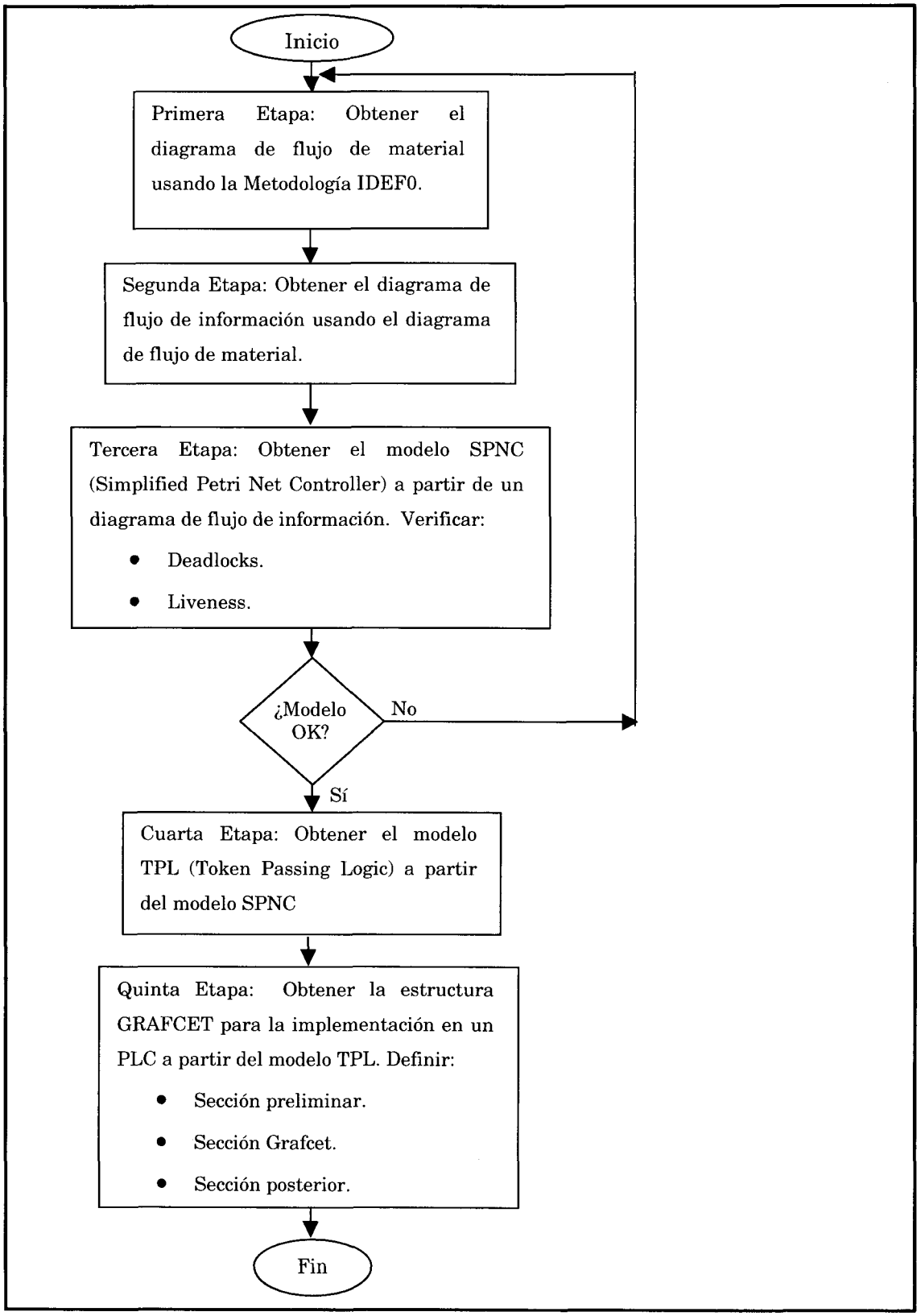
Salidas del PLC	***Descripción de la Salida del PLC***	***Función lógica booleana***

Tabla para relacionar las salidas del PLC con las etapas de la sección Grafcet y sus respectivos retrasos.

ANEXO 2

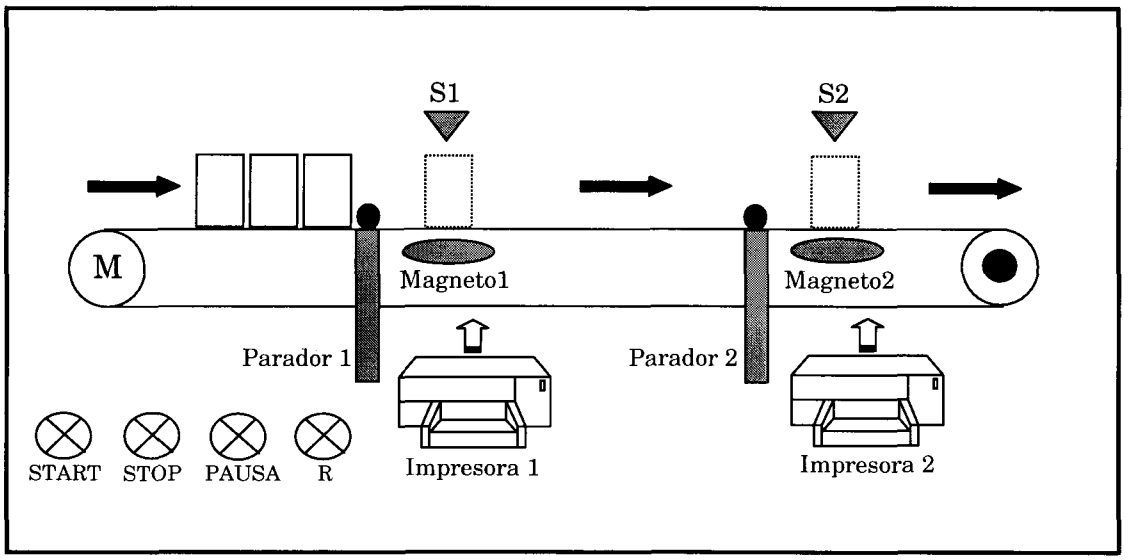
Resumen de los resultados obtenidos por etapa para el proceso de estampado de latas utilizando la metodología IPTG en GRAFCET.

Diagrama de flujo para las etapas de la metodología IPTG en GRAFCET.

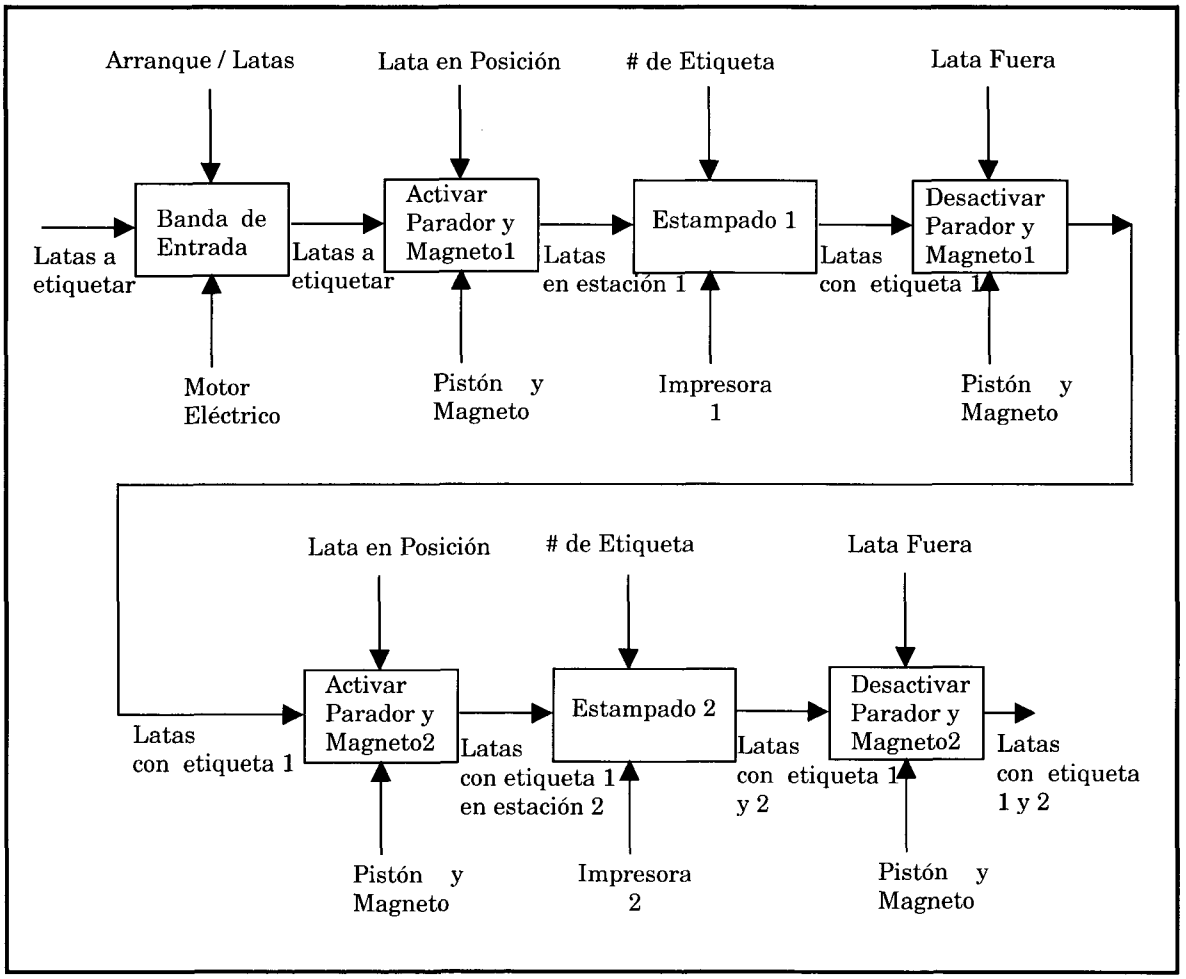


Resumen de los resultados obtenidos por etapa para el proceso de estampado de latas utilizando la metodología IPTG en GRAFCET.

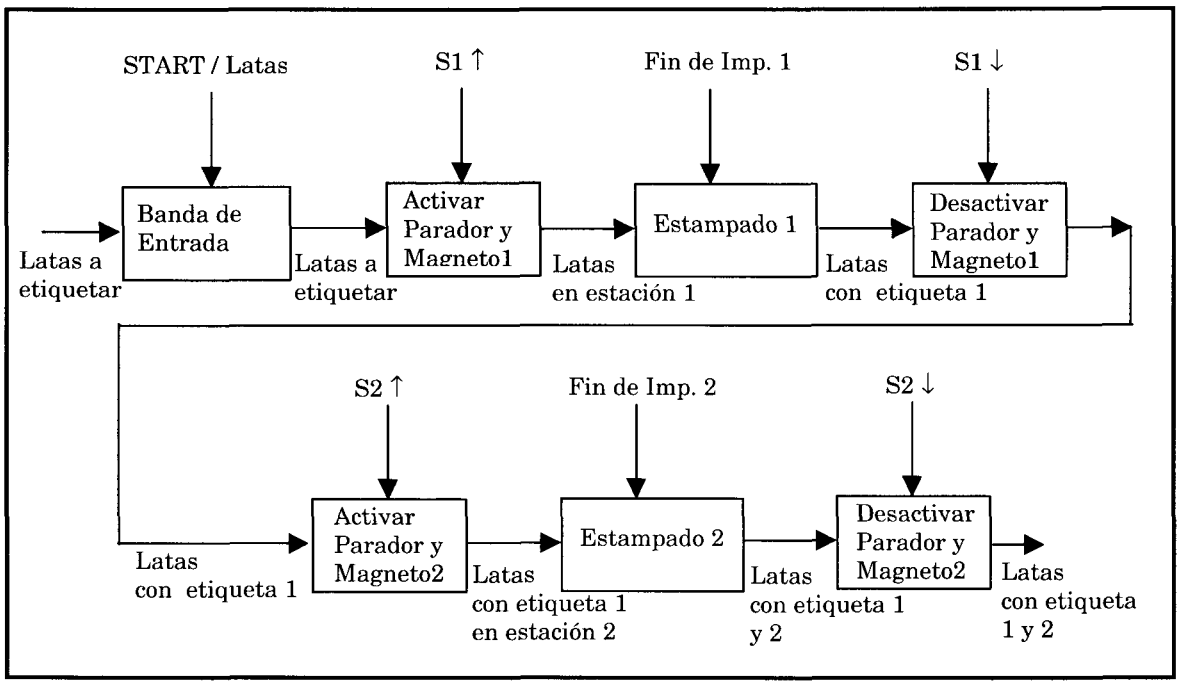
Proceso de Estampado de Latas.



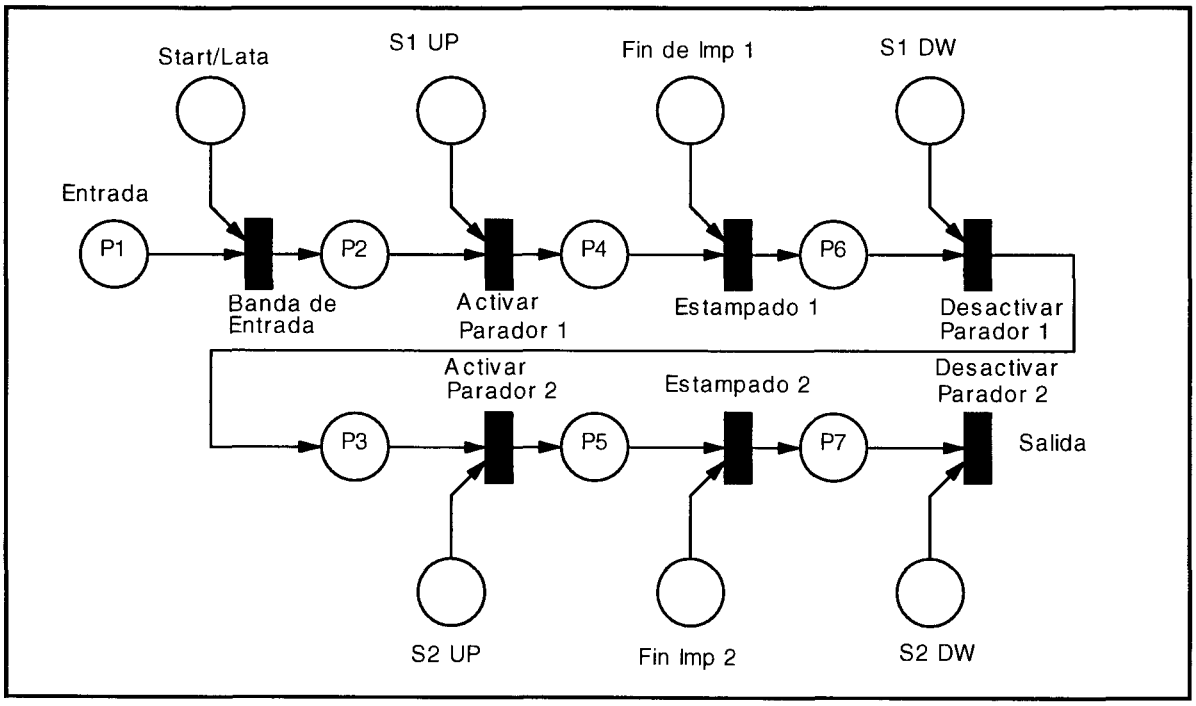
Primera Etapa: Diagrama de flujo de material utilizando **IDEF0**.



Segunda Etapa: Diagrama de flujo de información.

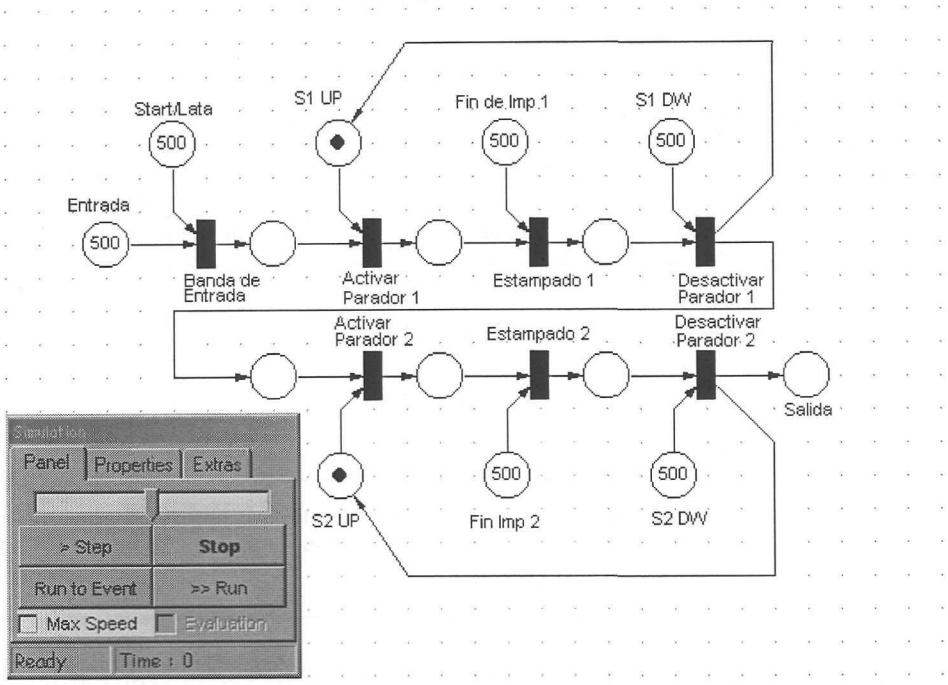


Tercera Etapa: Obtener el modelo SPNC (Simplified Petri Net Controller).

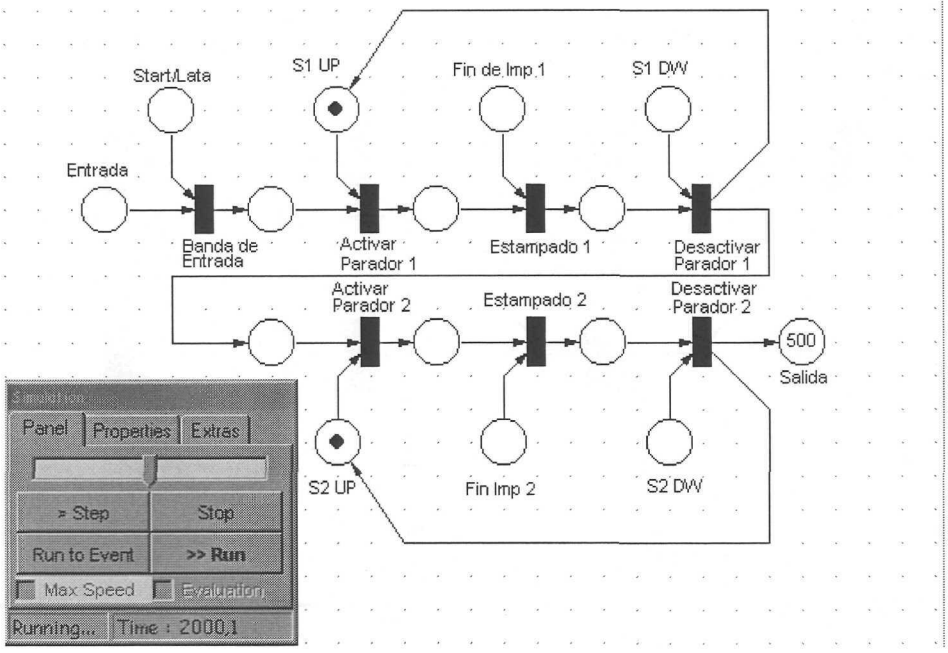


Análisis de modelo SPNC.

Análisis en tiempo t = 0.

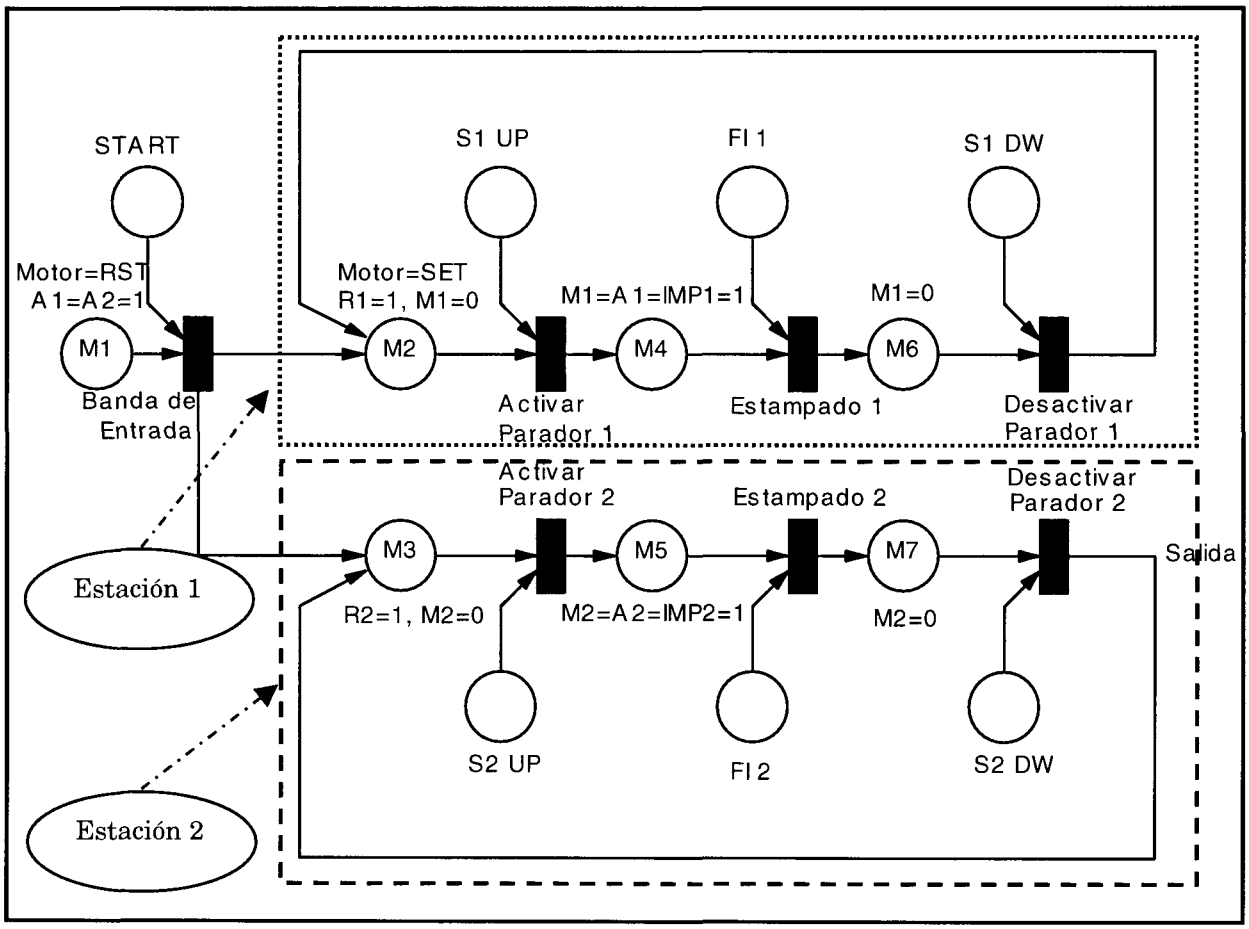


Análisis en tiempo t = 2000.



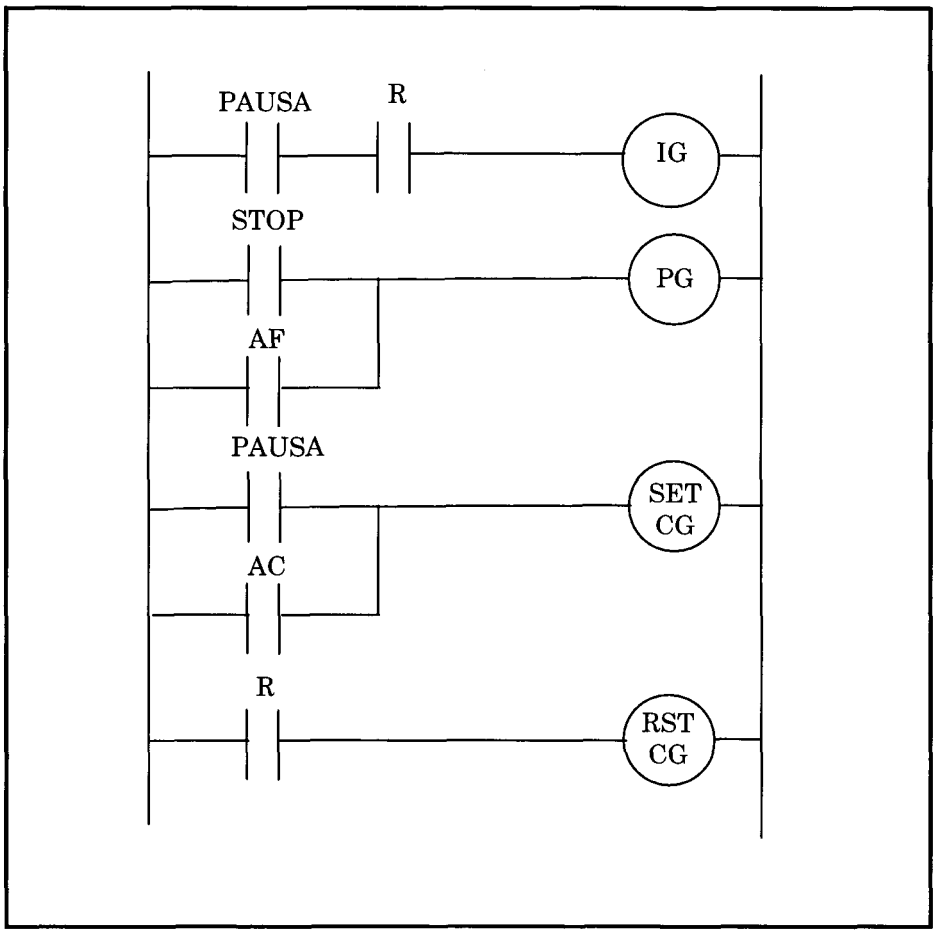
Conclusión: el comportamiento dinámico del modelo SPNC obtenido es aceptable ya que las 500 latas son transportadas a la estación de salida en un tiempo finito.

Cuarta Etapa: Obtener el modelo **TPL** (Token Passing Logic).

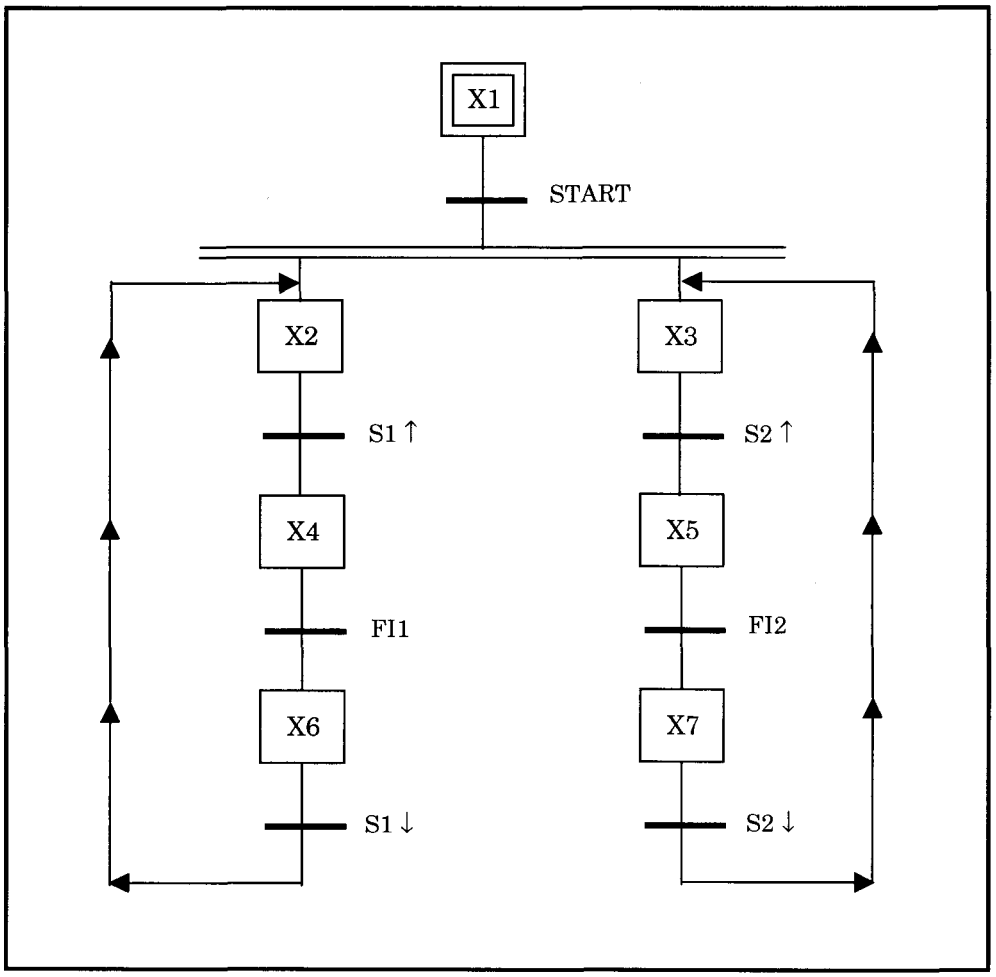


Quinta Etapa: Obtener el programa GRAFCET para la Implementación en un PLC.

Sección preeliminar



Sección Grafcet.

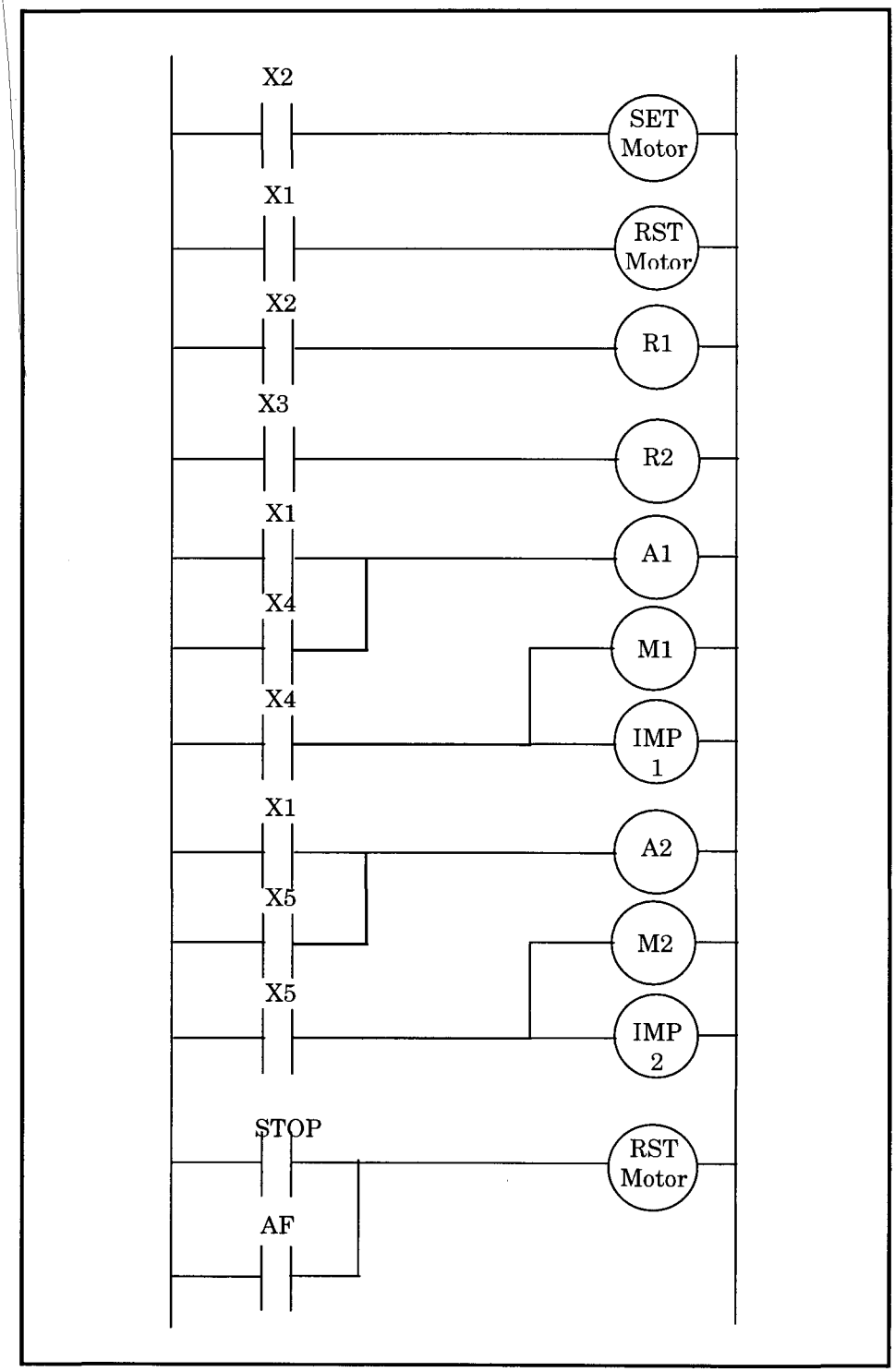


Sección Posterior.

Tabla para relacionar las salidas del PLC con las etapas de la sección Grafcet y sus respectivos retrasos:

Salidas del PLC	*Descripción de la Salida del PLC*	*Función lógica booleana*
Salida 1	SET del Motor	X2
Salida 1	RST del Motor	X1
Salida 2	Activar parador 1 (A1)	X1∨ X4
Salida 3	Retroceder parador 1 (R1)	X2
Salida 4	Activar parador 2 (A2)	X1∨ X5
Salida 5	Retroceder parador 2 (R2)	X3
Salida 6	Activar magneto 1 (M1)	X4
Salida 7	Activar magneto 2 (M2)	X5
Salida 8	Habilitar impresora 1 (IMP1)	X4
Salida 9	Habilitar impresora 2 (IMP2)	X5

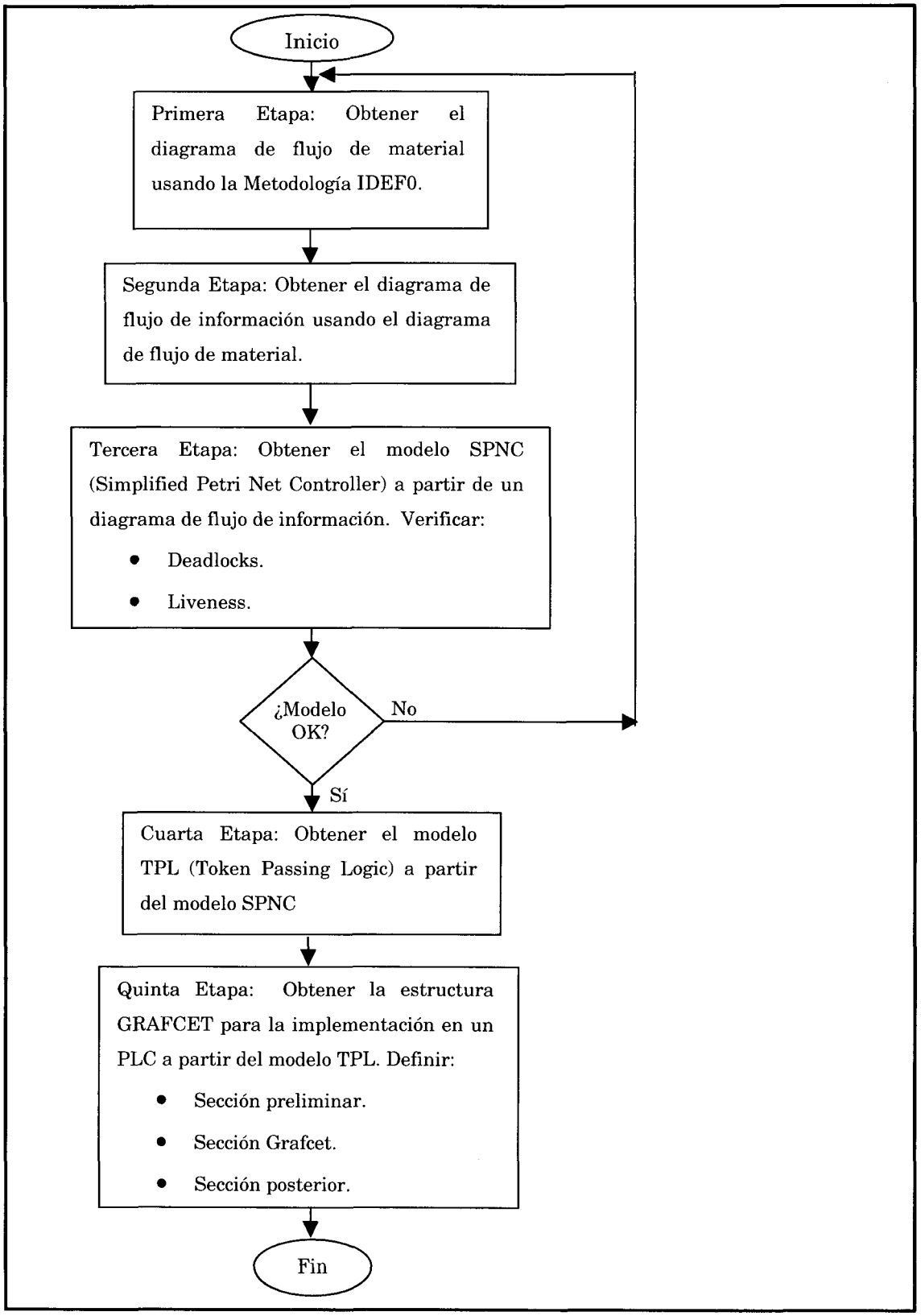
Diagrama escalera para la sección posterior:



ANEXO 2

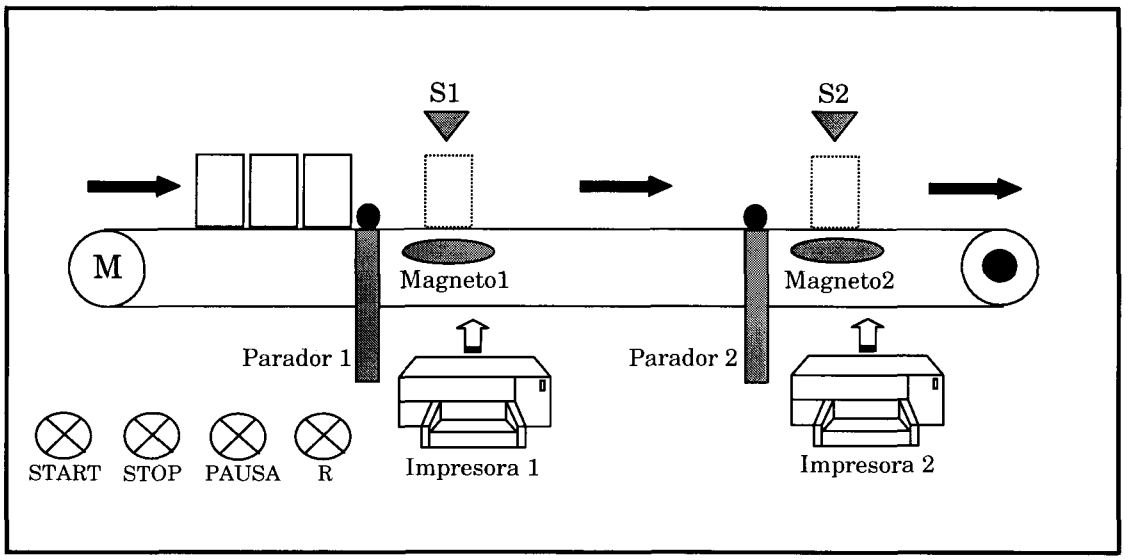
Resumen de los resultados obtenidos por etapa para el proceso de estampado de latas utilizando la metodología IPTG en GRAFCET.

Diagrama de flujo para las etapas de la metodología IPTG en GRAFCET.

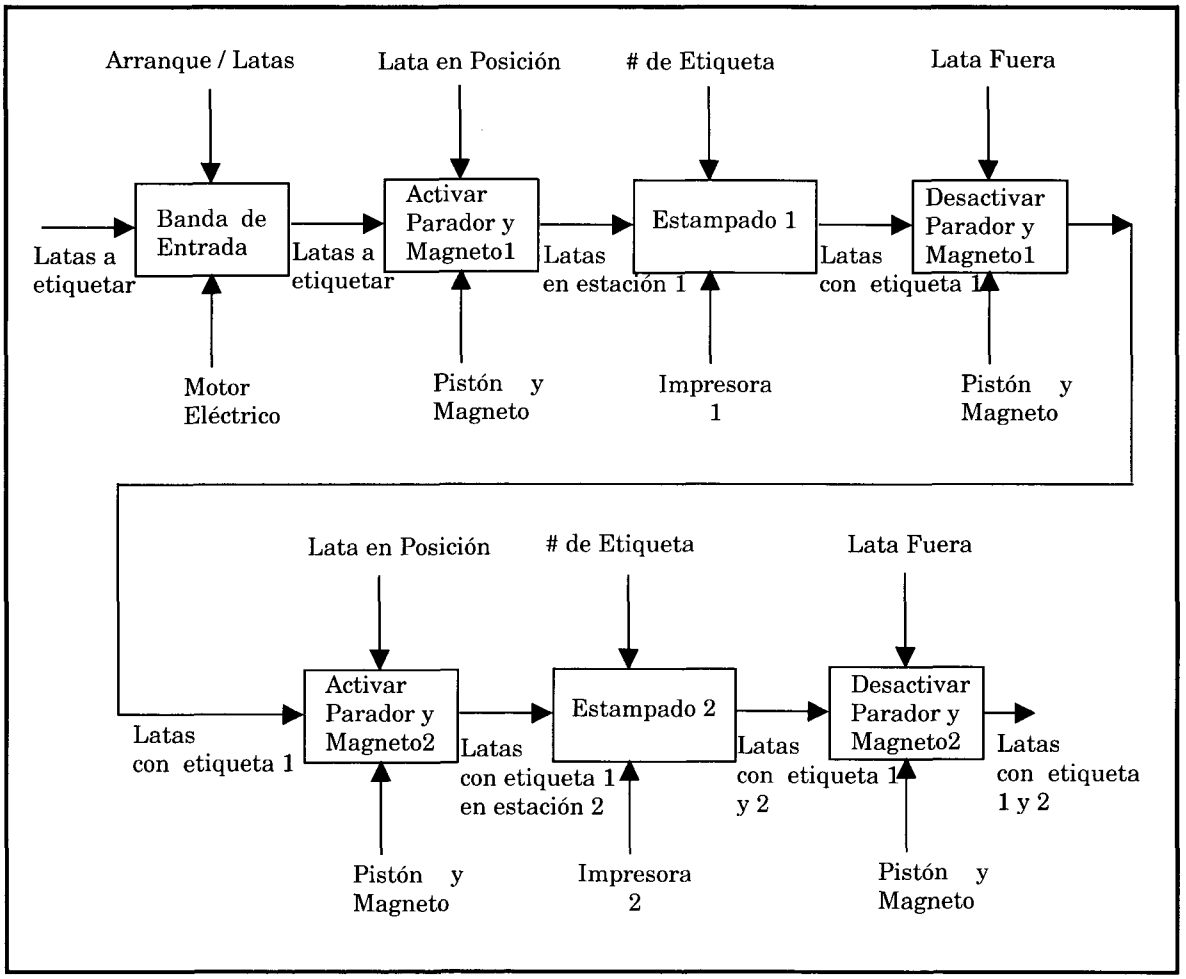


Resumen de los resultados obtenidos por etapa para el proceso de estampado de latas utilizando la metodología IPTG en GRAFCET.

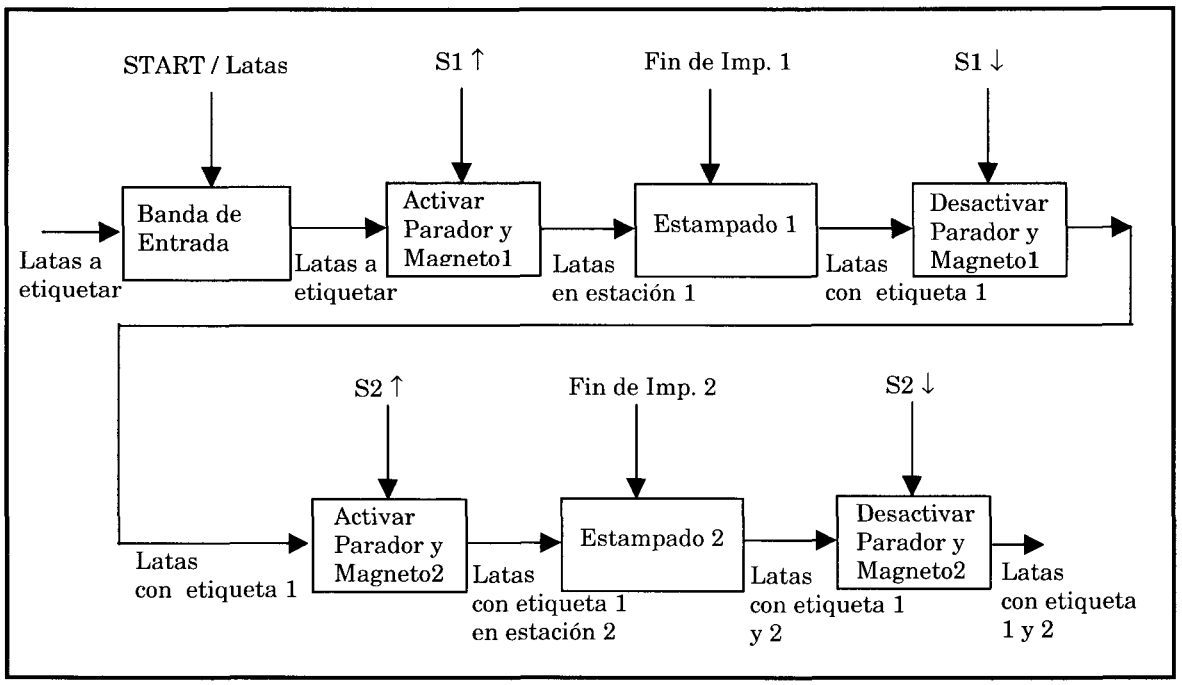
Proceso de Estampado de Latas.



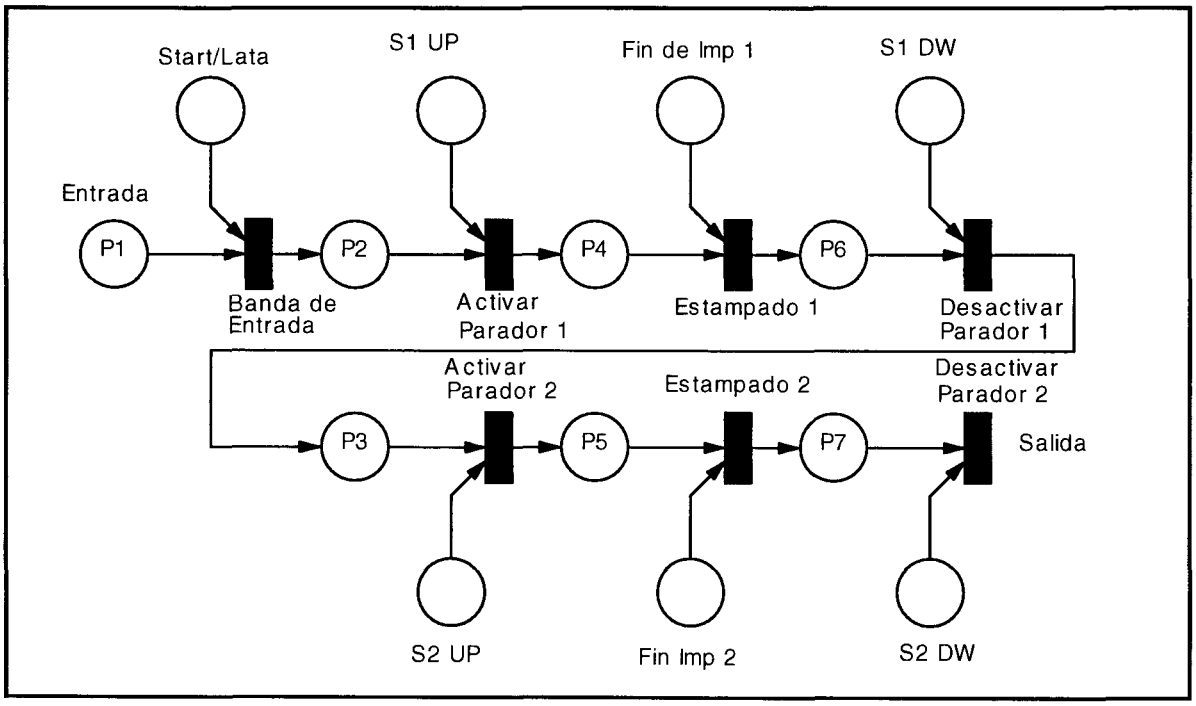
Primera Etapa: Diagrama de flujo de material utilizando **IDEF0**.



Segunda Etapa: Diagrama de flujo de información.

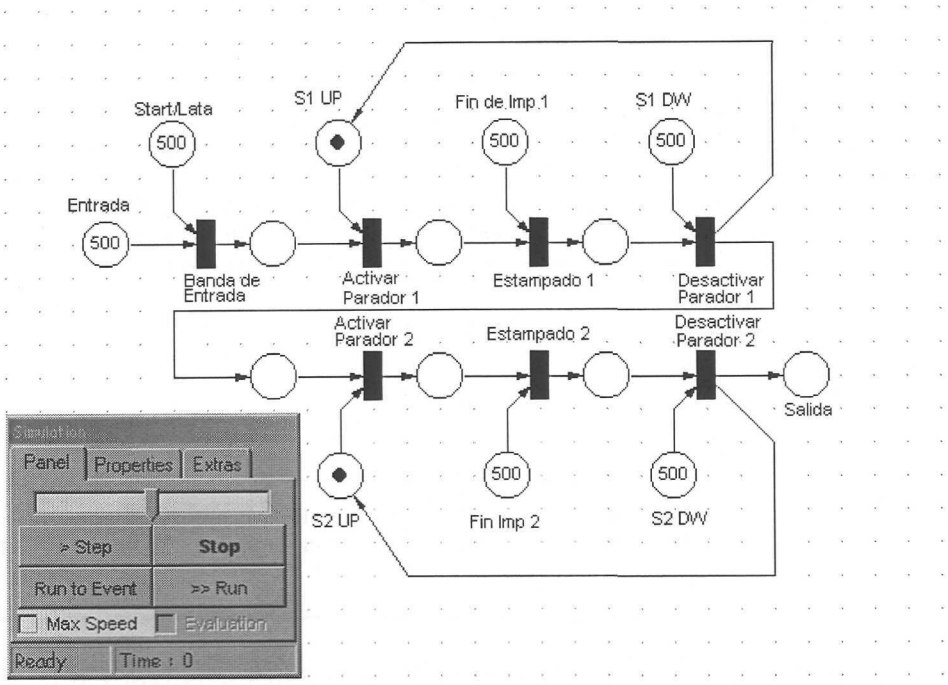


Tercera Etapa: Obtener el modelo SPNC (Simplified Petri Net Controller).

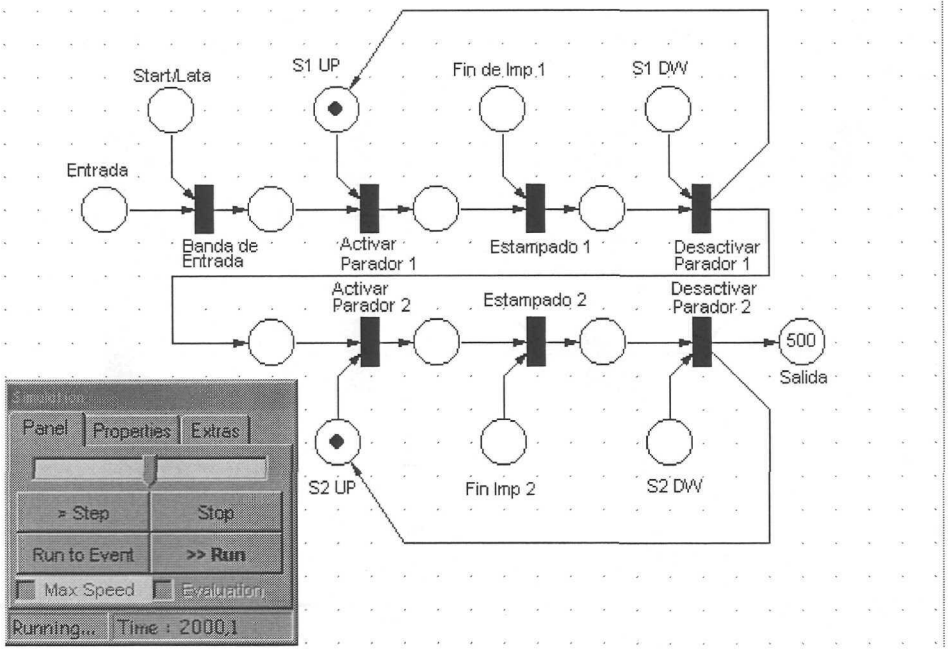


Análisis de modelo SPNC.

Análisis en tiempo t = 0.

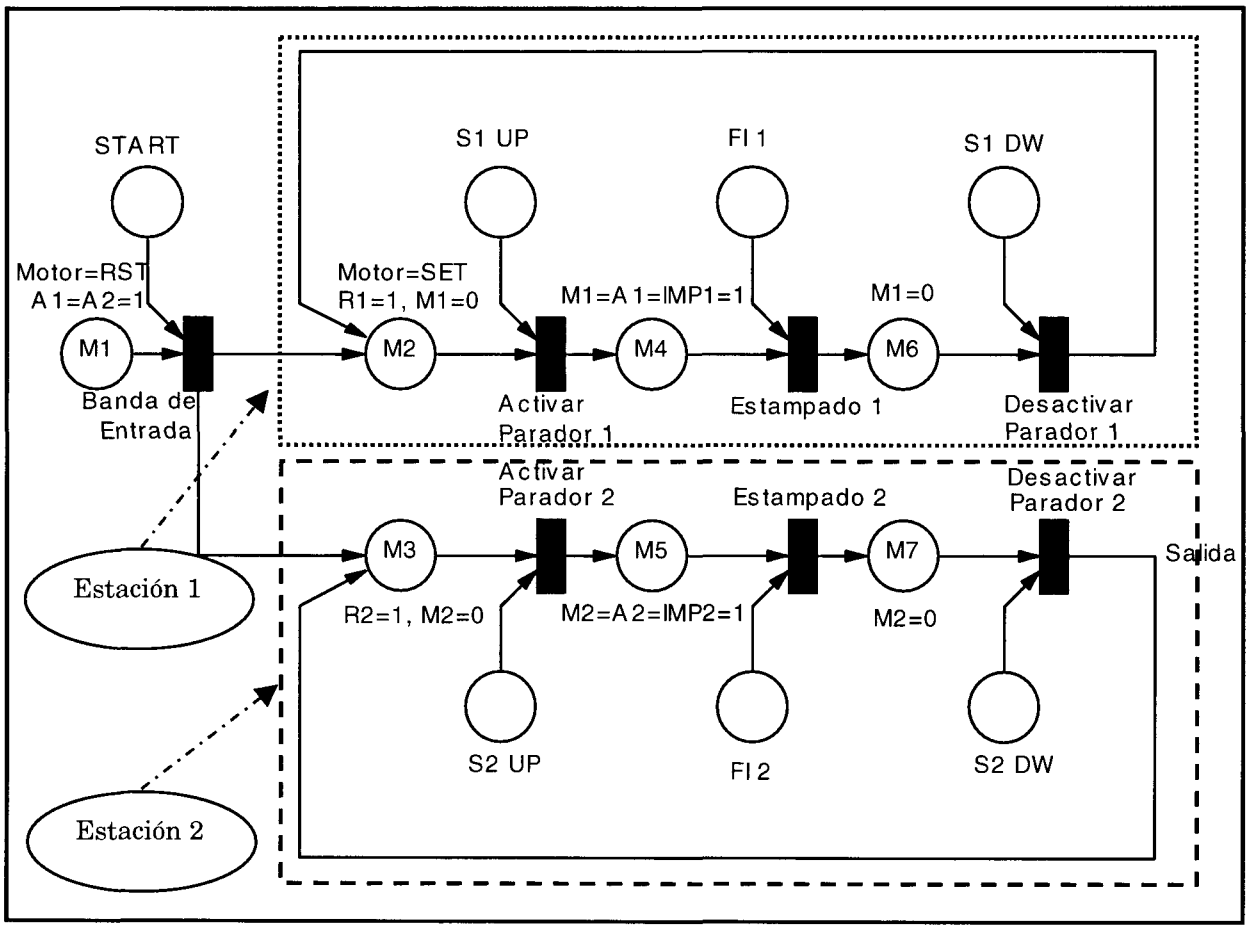


Análisis en tiempo t = 2000.



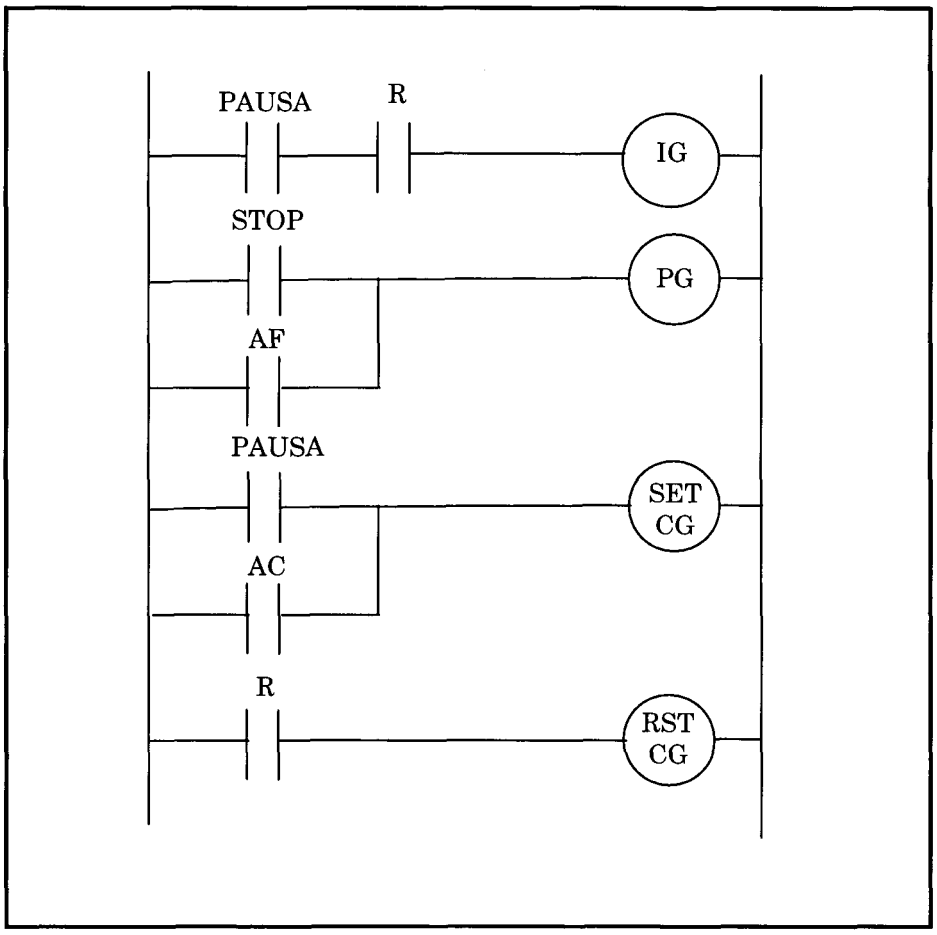
Conclusión: el comportamiento dinámico del modelo SPNC obtenido es aceptable ya que las 500 latas son transportadas a la estación de salida en un tiempo finito.

Cuarta Etapa: Obtener el modelo **TPL** (Token Passing Logic).

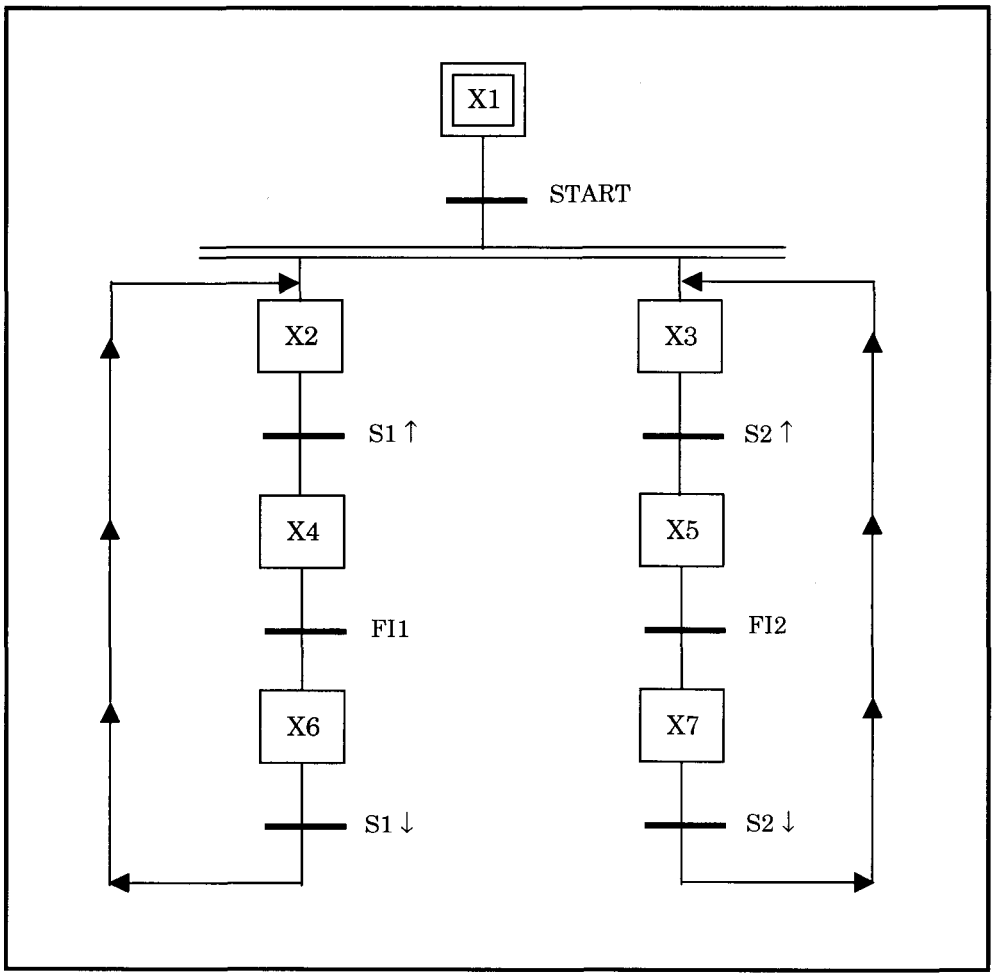


Quinta Etapa: Obtener el programa GRAFCET para la Implementación en un PLC.

Sección preeliminar



Sección Grafcet.

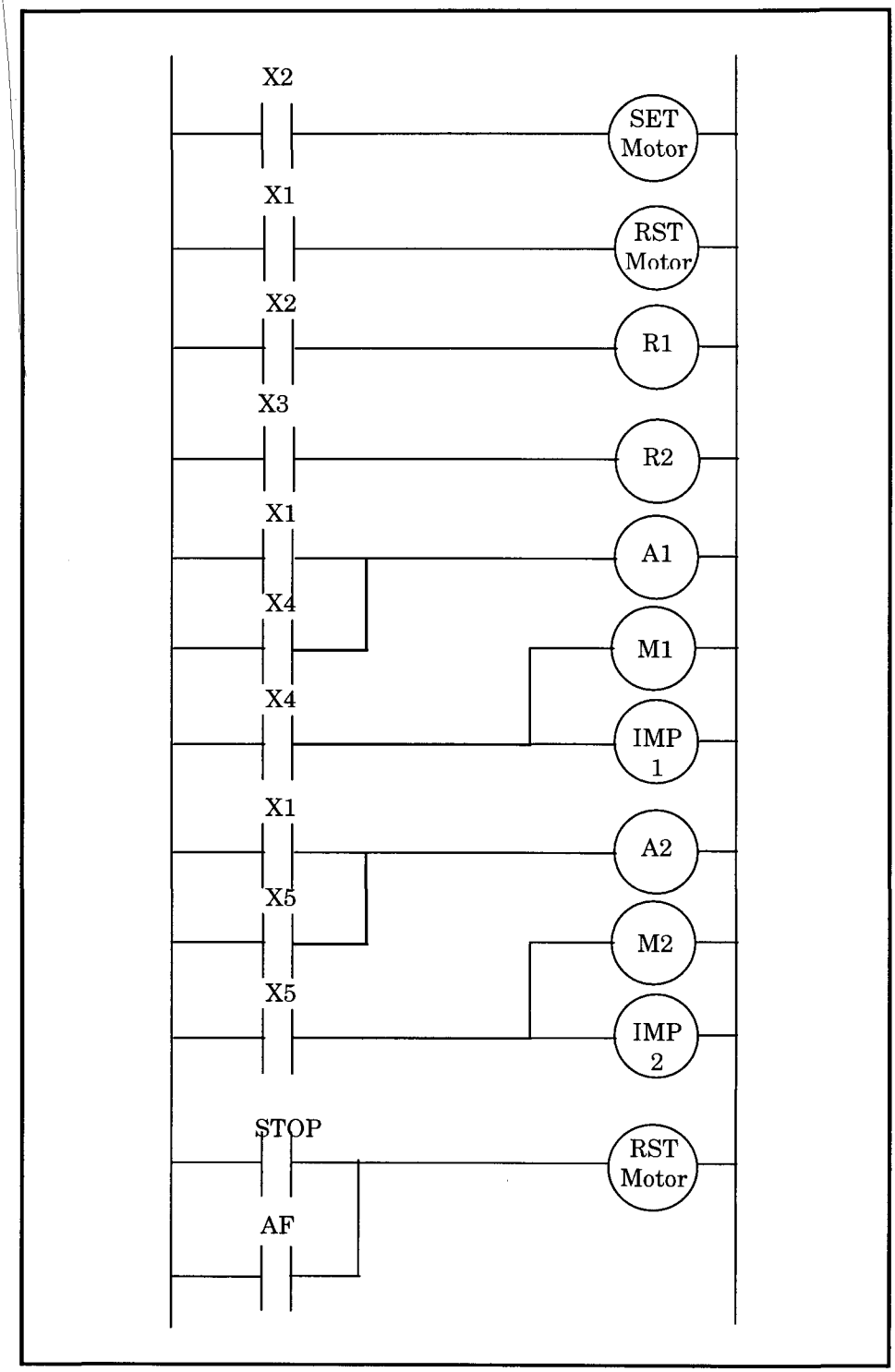


Sección Posterior.

Tabla para relacionar las salidas del PLC con las etapas de la sección Grafcet y sus respectivos retrasos:

Salidas del PLC	*Descripción de la Salida del PLC*	*Función lógica booleana*
Salida 1	SET del Motor	X2
Salida 1	RST del Motor	X1
Salida 2	Activar parador 1 (A1)	X1∨ X4
Salida 3	Retroceder parador 1 (R1)	X2
Salida 4	Activar parador 2 (A2)	X1∨ X5
Salida 5	Retroceder parador 2 (R2)	X3
Salida 6	Activar magneto 1 (M1)	X4
Salida 7	Activar magneto 2 (M2)	X5
Salida 8	Habilitar impresora 1 (IMP1)	X4
Salida 9	Habilitar impresora 2 (IMP2)	X5

Diagrama escalera para la sección posterior:



ANEXO 3

Tablas en blanco, utilizadas por la metodología RIMAnI en GRAFCET.

Nombre del Estado	Descripción del Estado	Salidas a activar con condición booleana y tipo de salida

Tabla que relaciona los estados y las salidas del proceso.

Nombre del Estado	Descripción del Estado	Evento(s) prohibido(s)	Acción a llevar a cabo para cada evento prohibido (tipo y condición booleana)

Tabla que relaciona los estados del proceso y los eventos prohibidos.

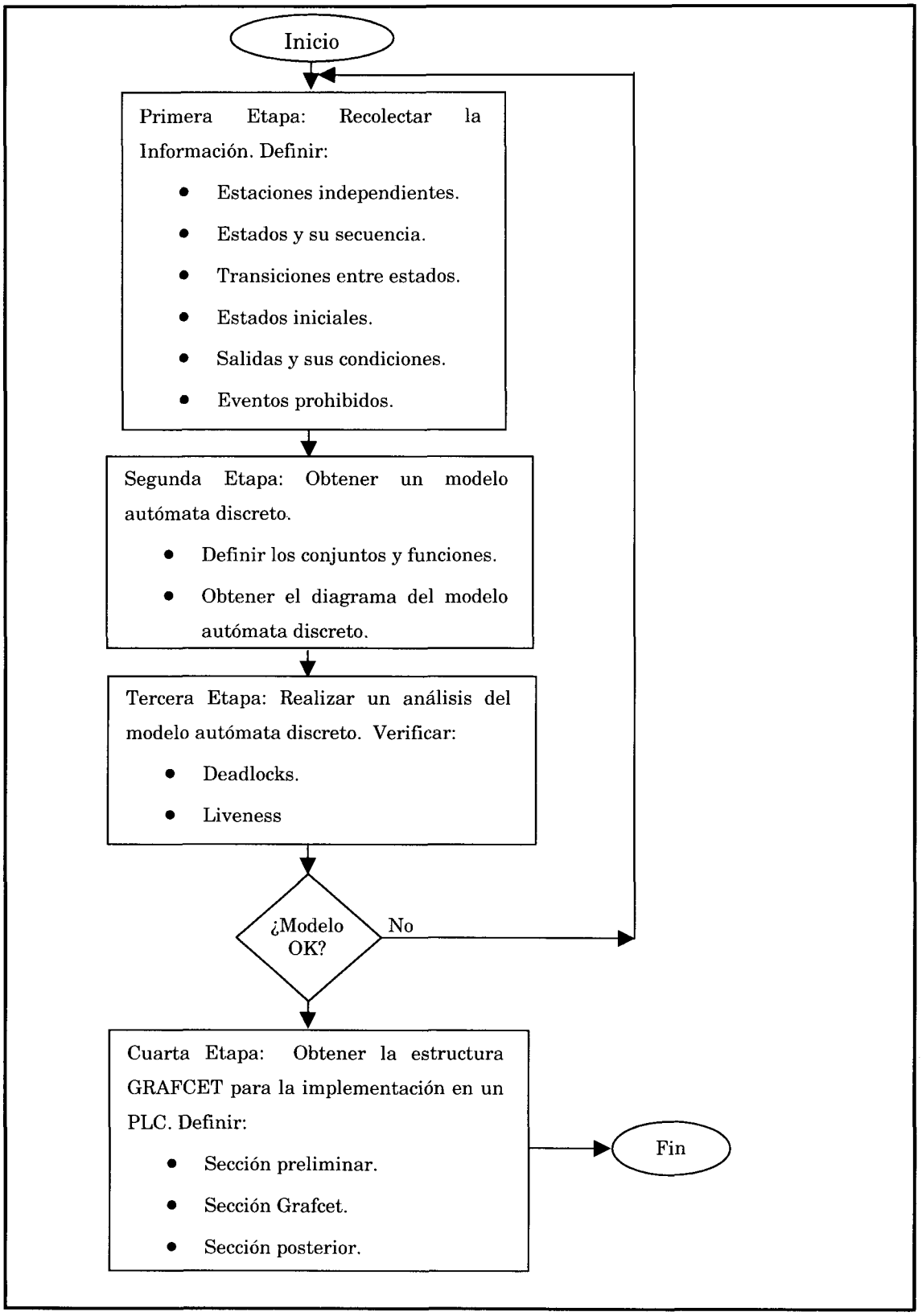
Salidas del PLC	***Descripción de la Salida del PLC***	***Función lógica booleana***

Tabla para relacionar las salidas del PLC con las etapas de la sección Grafcet y sus respectivos retrasos.

ANEXO 4

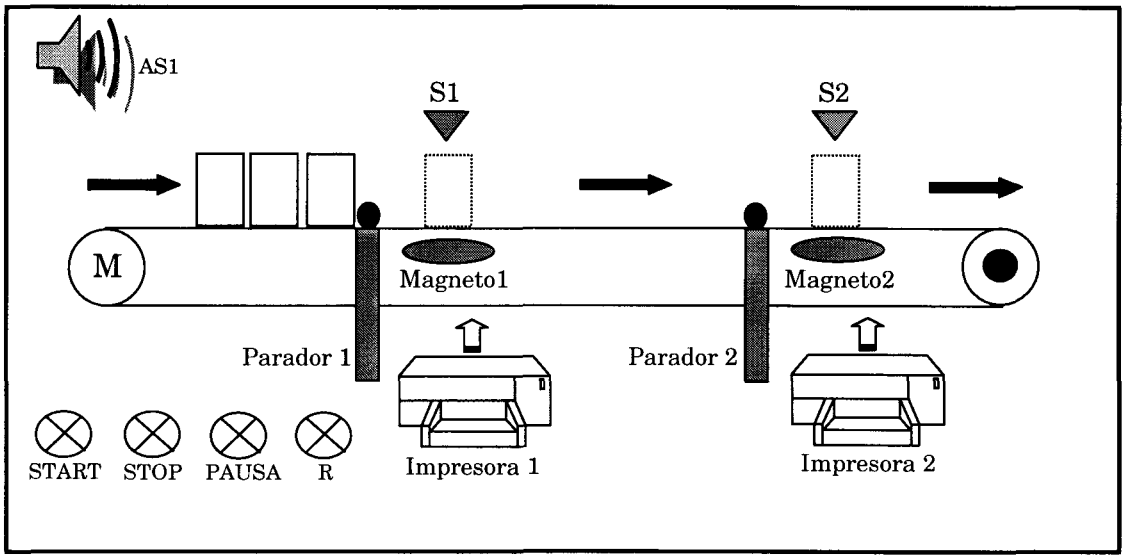
Resumen de los resultados obtenidos por etapa para el proceso de estampado de latas utilizando la metodología RIMAnI en GRAFCET.

Diagrama de flujo para las etapas de la metodología RIMAnI en GRAFCET.

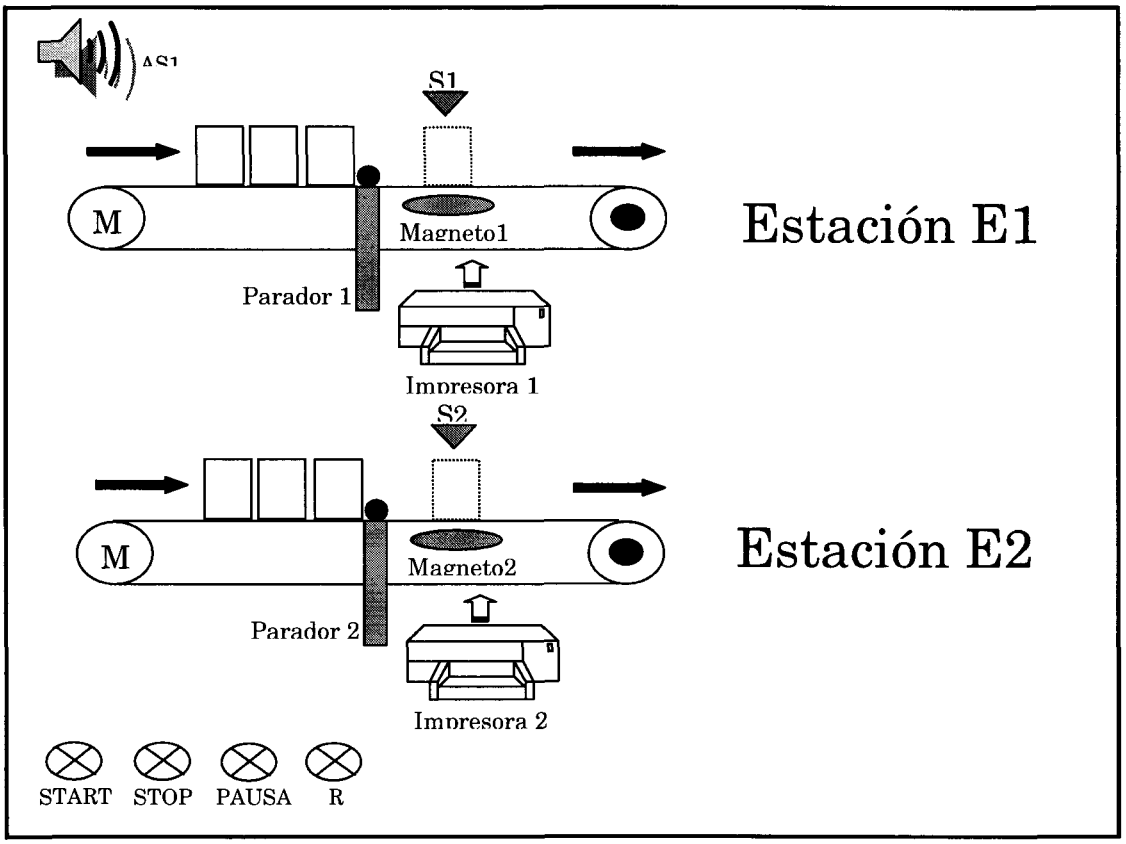


Resumen de los resultados obtenidos por etapa para el proceso de estampado de latas utilizando la metodología RIMAnI en GRAFCET.

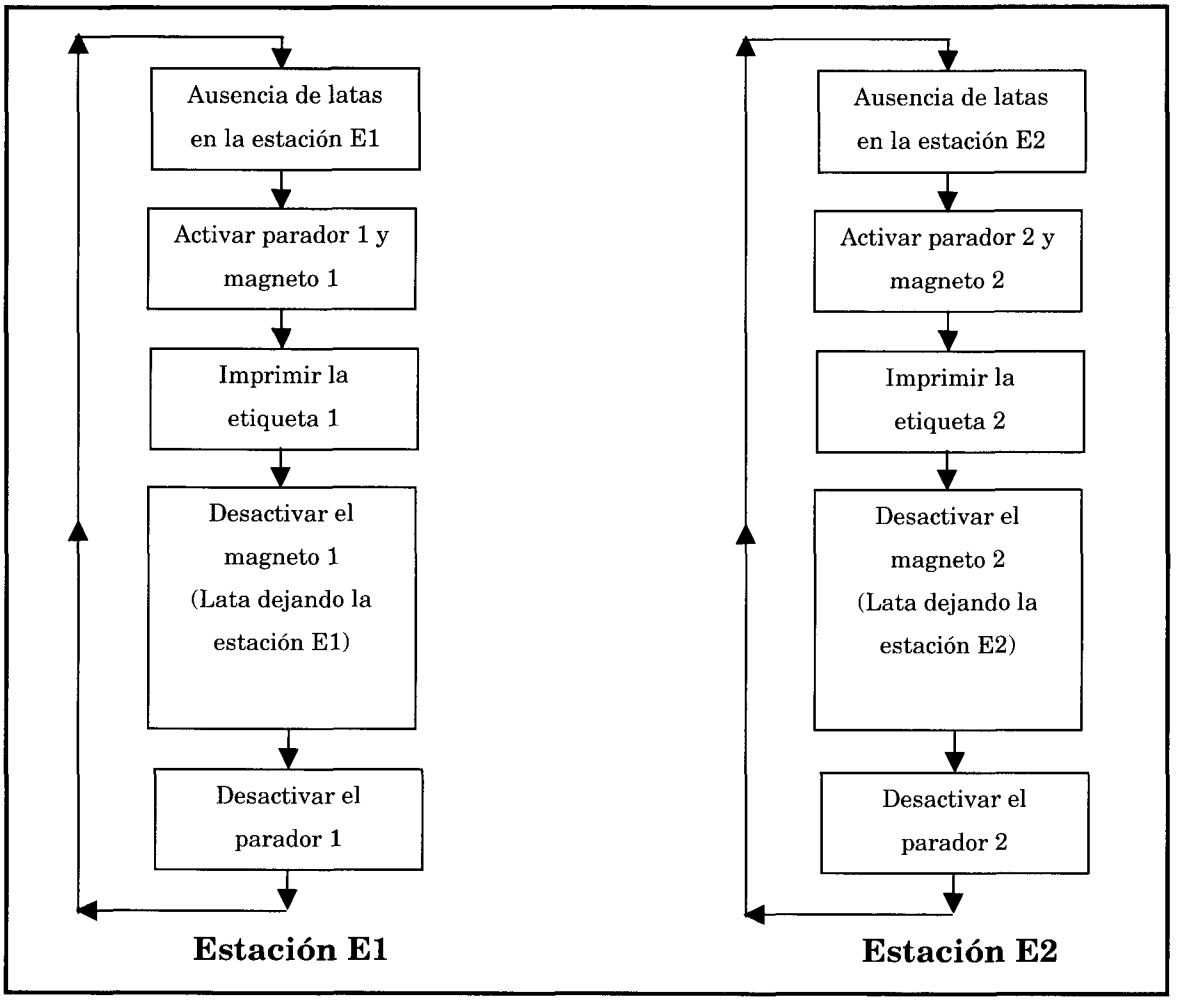
Proceso de Estampado de Latas.



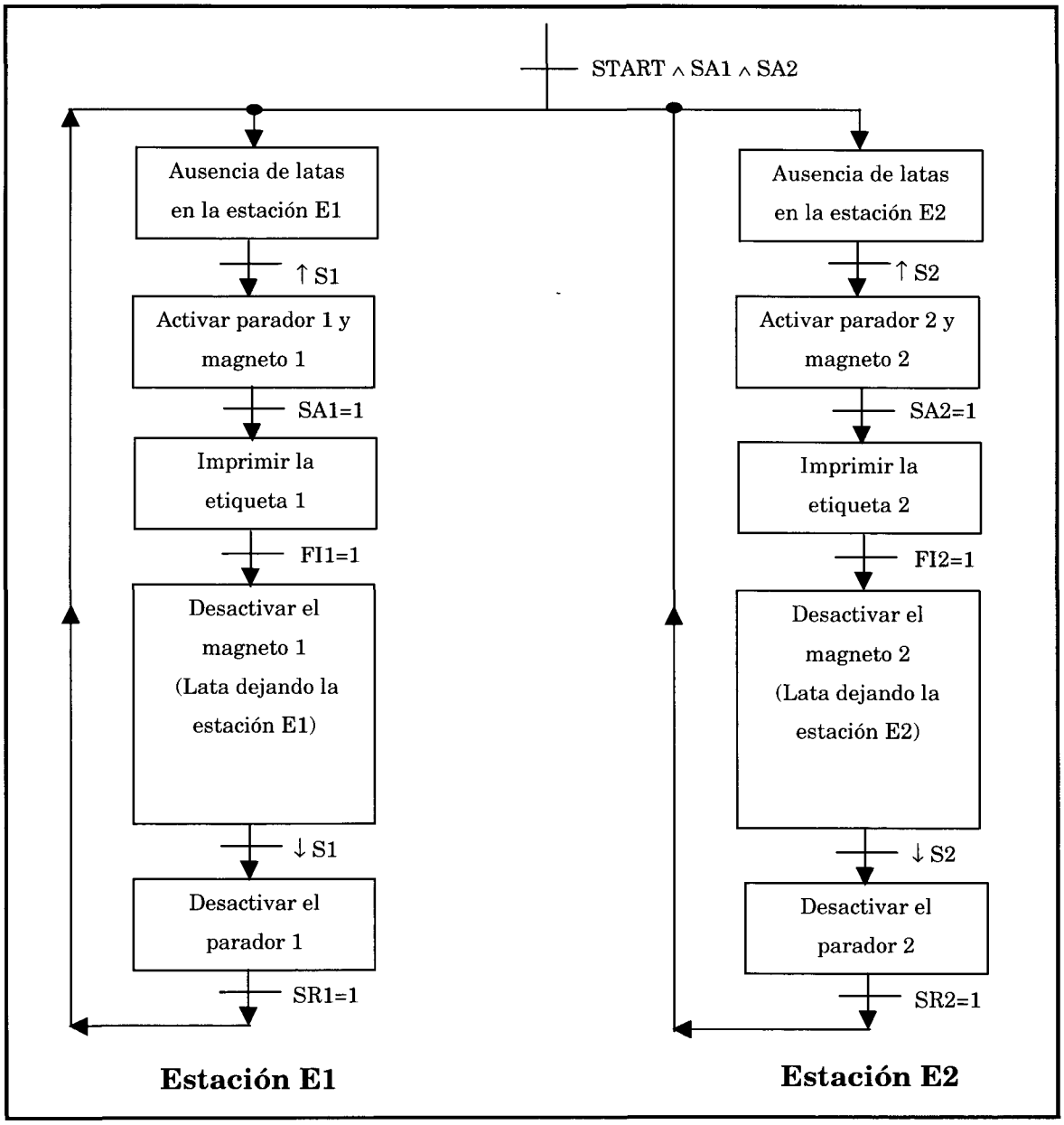
Primera Etapa: Recolectar Información.
Estaciones Independientes.



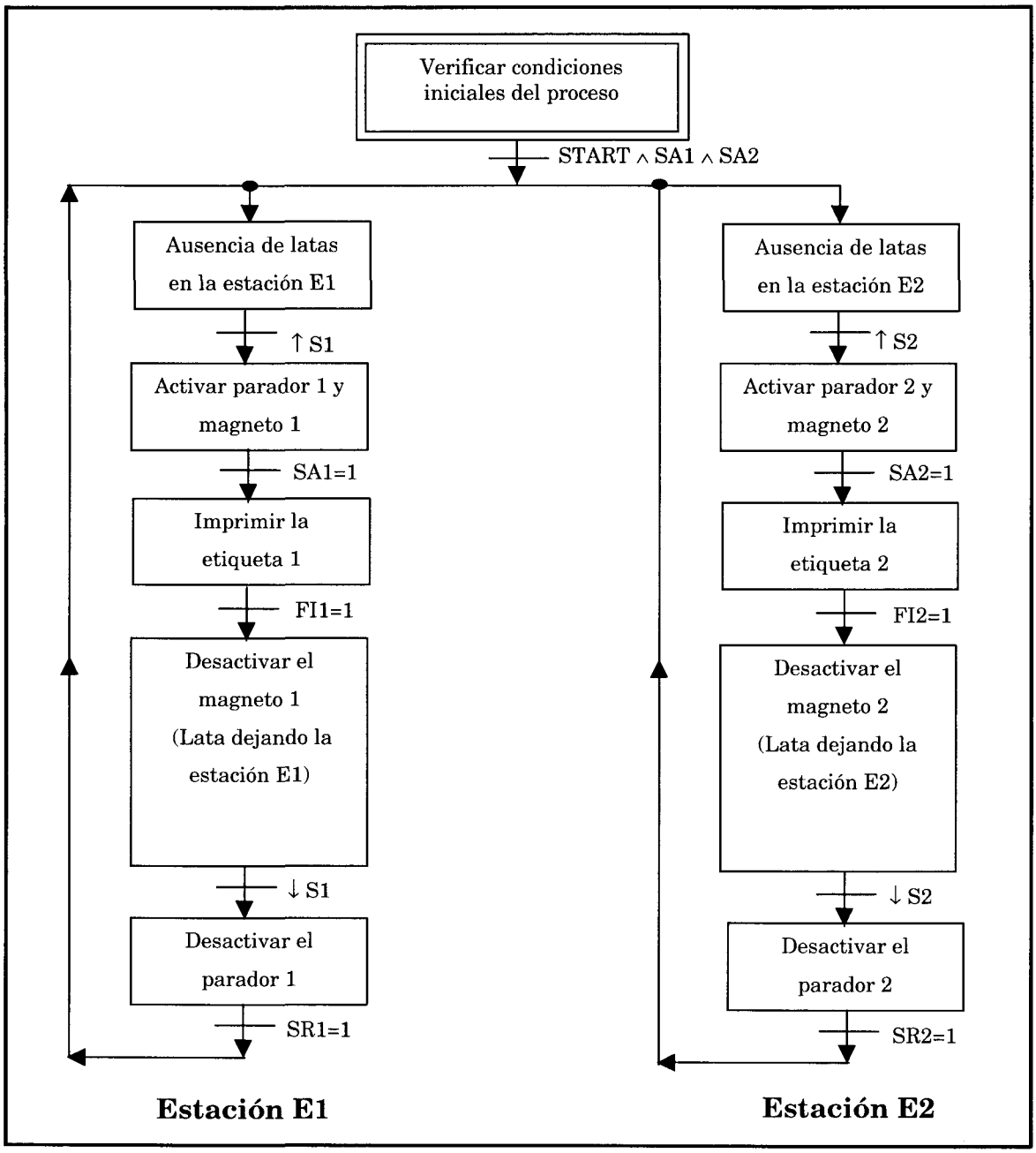
Estados y su secuencia.



Transiciones entre estados.



Estado(s) Inicial(es) de(l) proceso.



Salidas de control del proceso y sus condiciones.

Nombre del Estado	Descripción del Estado	Salidas a activar con condición booleana y tipo de salida
X0	Verificar condiciones iniciales del proceso	• Motor (RST) • A1 • A2
X1	Ausencia de latas en la estación E1	• Motor (SET) • R1
X2	Ausencia de latas en la estación E2	• R2
X3	Activar parador 1 y magneto 1	• A1 • M1(SET)
X4	Activar parador 2 y magneto 2	• A2 • M2 (SET)
X5	Imprimir la etiqueta 1	• IMP1
X6	Imprimir la etiqueta 2	• IMP2
X7	Desactivar el magneto 1 y Lata dejando la estación E1	• M1(RST)
X8	Desactivar el magneto 2 y Lata dejando la estación E2	• M2(RST)
X9	Desactivar el parador 1	• R1
X10	Desactivar el parador 2	• R2

Eventos Prohibidos.

Nombre del Estado	Descripción del Estado	Evento(s) prohibido(s)	Acción a llevar a cabo para cada evento prohibido (tipo y condición booleana)
X0	Verificar condiciones iniciales del proceso	➢ S1 ➢ S2 ➢ SA1∧SR1 ➢ SA2∧SR2 ➢ EI1 ➢ EI2	➢ (S1,S2, FI1, FI2)→ Prender AS1 ➢ (SA1∧SR1, SA2∧SR2,) → Prender AS1
X1	Ausencia de latas en la estación E1	➢ SA1∧SR1	➢ (SA1∧SR1) → Prender AS1
X2	Ausencia de latas en la estación E2	➢ SA2∧SR2	➢ (SA2∧SR2) → Prender AS1
X3	Activar parador 1 y magneto 1	➢ SA1∧SR1 ➢ &S1	➢ (SA1∧SR1) → Prender AS1 ➢ (&S1) → Prender AS1
X4	Activar parador 2 y magneto 2	➢ SA2∧SR2 ➢ &S2	➢ (SA2∧SR2) → Prender AS1 ➢ (&S2) → Prender AS1
X5	Imprimir la etiqueta 1	➢ SA1∧SR1 ➢ &S1	➢ (SA1∧SR1) → Prender AS1 ➢ (&S1) → Prender AS1
X6	Imprimir la etiqueta 2	➢ SA2∧SR2 ➢ &S2	➢ (SA2∧SR2) → Prender AS1 ➢ (&S2) → Prender AS1
X7	Desactivar el magneto 1 y lata dejando la estación E1	➢ SA1∧SR1	➢ (SA1∧SR1) → Prender AS1
X8	Desactivar el magneto 2 y Lata dejando la estación E2	➢ SA2∧SR2	➢ (SA2∧SR2) → Prender AS1
X9	Desactivar el parador 1	➢ SA1∧SR1	➢ (SA1∧SR1) → Prender AS1
X10	Desactivar el parador 2	➢ SA2∧SR2	➢ (SA2∧SR2) → Prender AS1

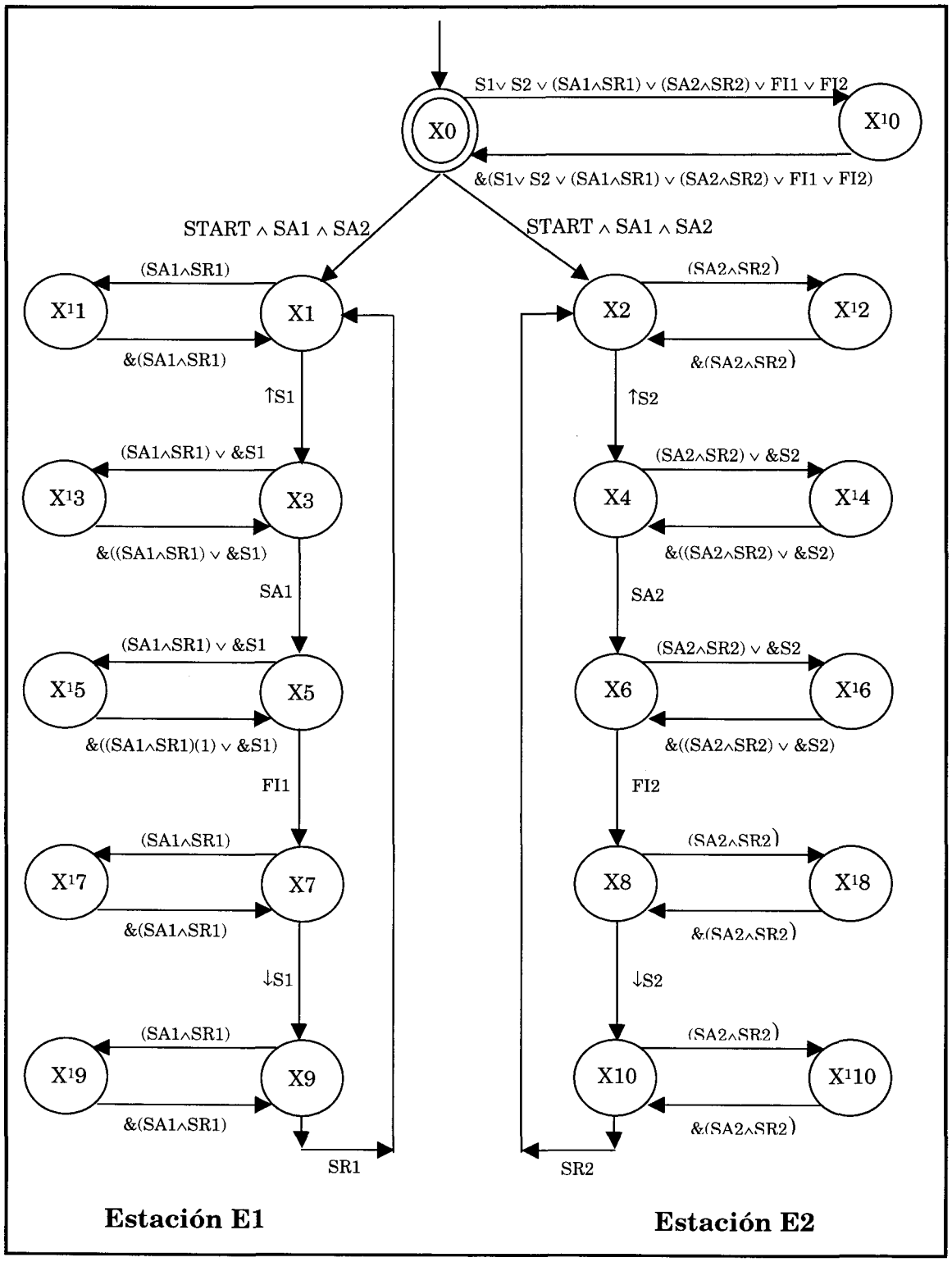
Segunda Etapa: Obtener un **M**odelo Autómata Discreto.

Conjunto y Funciones para el modelo autómata discreto.

- E = {START, S1, S2, SA1, SR1, SA2, SR2, FI1, FI2, T}
  T = {S1, S2, FI1, FI2, &S2, &S1, (SA1∧SR1), (SA2∧SR2)}
- X = {X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X^{1}0, X^{1}1, X^{1}2, X^{1}3, X104, X^{1}5, X^{1}6, X^{1}7, X^{1}8, X^{1}9, X^{1}10}
- X_0 = {X0}
- Las funciones de transición de estados f definidas por:
  - f(X0, START ∧ SA1 ∧ SA2) → X1
  - f(X0, START ∧ SA1 ∧ SA2) → X2
  - f(X1, ↑S1) → X3
  - f(X2, ↑S2) → X4
  - f(X3, SA1) → X5
  - f(X4, SA2) → X6
  - f(X5, FI1) → X7
  - f(X6, SA2) → X8
  - f(X7, ↓S1) → X9
  - f(X8, ↓S2) → X10
  - f(X9, SR1) → X1
  - f(X10, SR2) → X2

  Así como las funciones de transición de eventos prohibidos t definidas por:
  - t(X0, S1∨ S2 ∨ (SA1∧SR1) ∨ (SA2∧SR2) ∨ FI1 ∨ FI2) → X^{1}0
  - t(X^{1}0, &(S1∨ S2 ∨ (SA1∧SR1) ∨ (SA2∧SR2) ∨ FI1 ∨ FI2)) → X0
  - t(X1, (SA1∧SR1)) → X^{1}1
  - t(X^{1}1, &(SA1∧SR1)) → X1
  - t(X2, (SA2∧SR2)) → X^{1}2
  - t(X^{1}2, &(SA2∧SR2)) → X2
  - t(X3, (SA1∧SR1) ∨ &S1) → X^{1}3
  - t(X^{1}3, &((SA1∧SR1) ∨ &S1)) → X3
  - t(X4, (SA2∧SR2) ∨ &S2) → X^{1}4
  - t(X^{1}4, &((SA2∧SR2) ∨ &S2)) → X4
  - t(X5, (SA1∧SR1)(1) ∨ &S1) → X^{1}5
  - t(X^{1}5, &((SA1∧SR1)(1) ∨ &S1)) → X5
  - t(X6, (SA2∧SR2)(1) ∨ &S2) → X^{1}6
  - t(X^{1}6, &((SA2∧SR2)(1) ∨ &S2)) → X6
  - t(X7, (SA1∧SR1)) → X^{1}7
  - t(X^{1}7, &(SA1∧SR1)) → X7
  - t(X8, (SA2∧SR2)) → X^{1}8
  - t(X^{1}8, &(SA2∧SR2)) → X8
  - t(X9, (SA1∧SR1)) → X^{1}9
  - t(X^{1}9, &(SA1∧SR1)) → X9
  - t(X10, (SA2∧SR2)) → X^{1}10
  - t(X110, &(SA2∧SR2)) → X10

Diagrama del modelo autómata discreto.



Tercera Etapa: Analizar el modelo autómata obtenido.

Diagrama general de la simulación en Model Visual Studium (MVS):

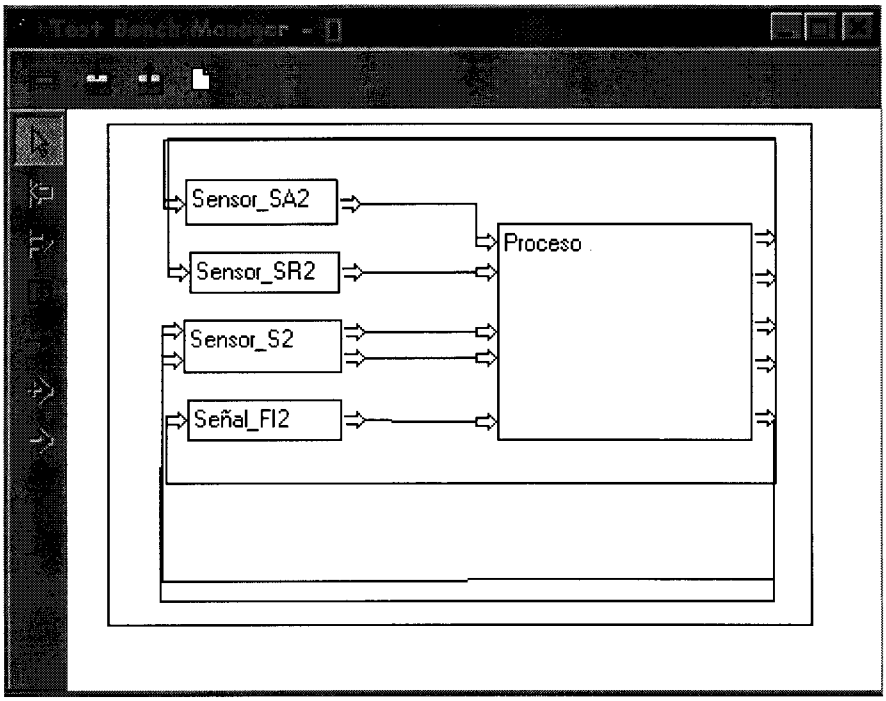


Diagrama de estados para el autómata del proceso:

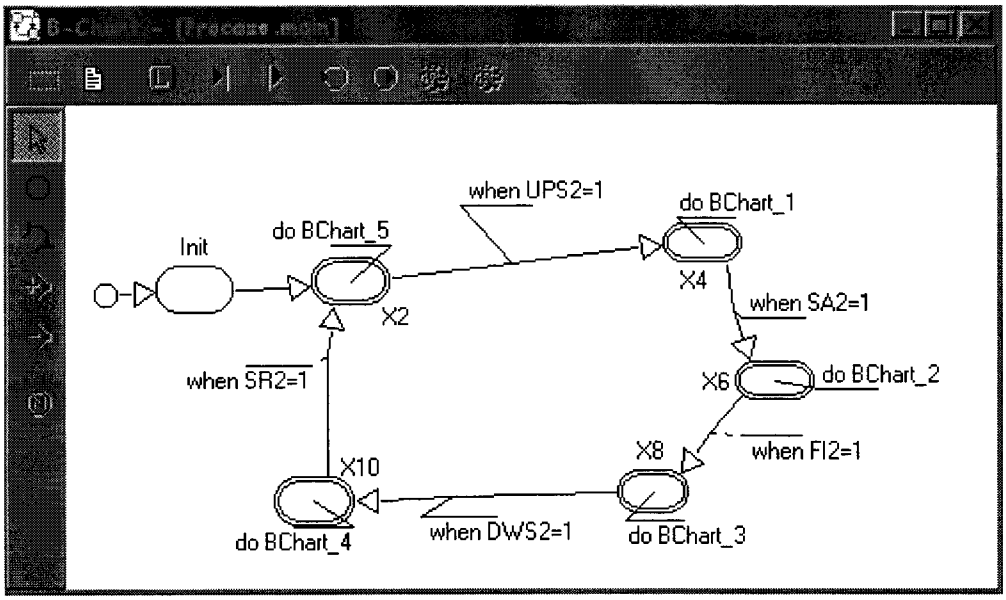


Diagrama de estados para el autómata del sensor SA2:

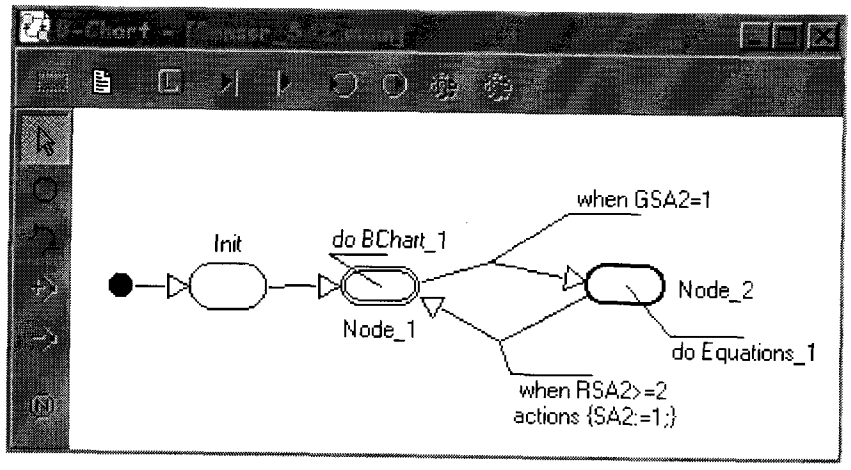


Diagrama de estados para el autómata del sensor SR2:

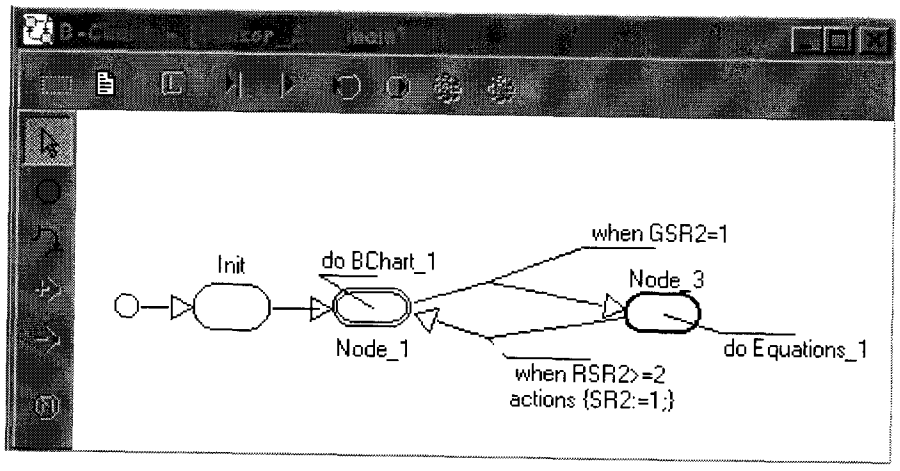


Diagrama de estados para el autómata del sensor S2:

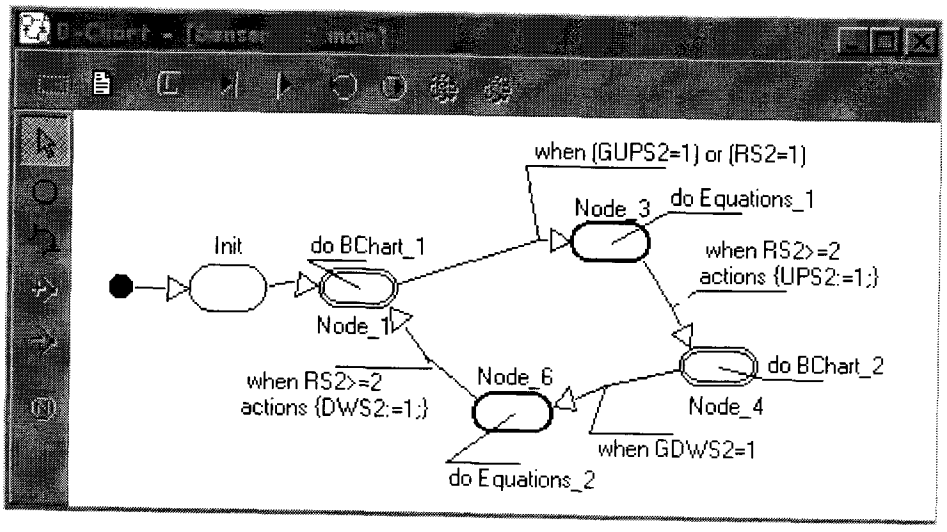


Diagrama de estados para el autómata de la señal FI2:

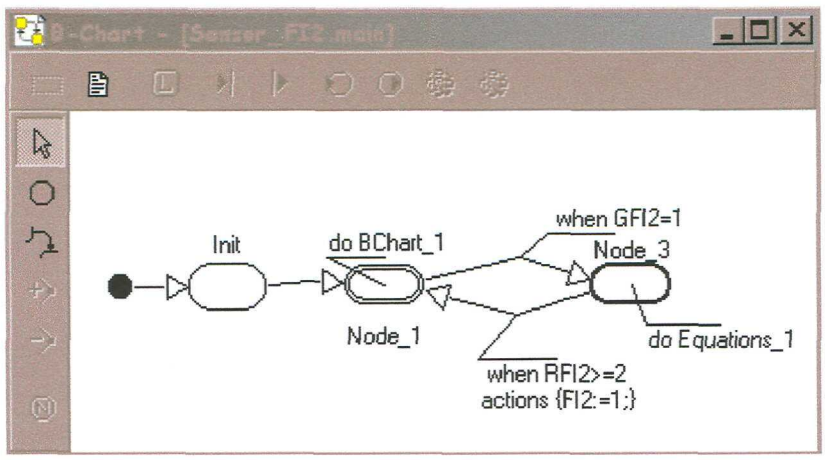
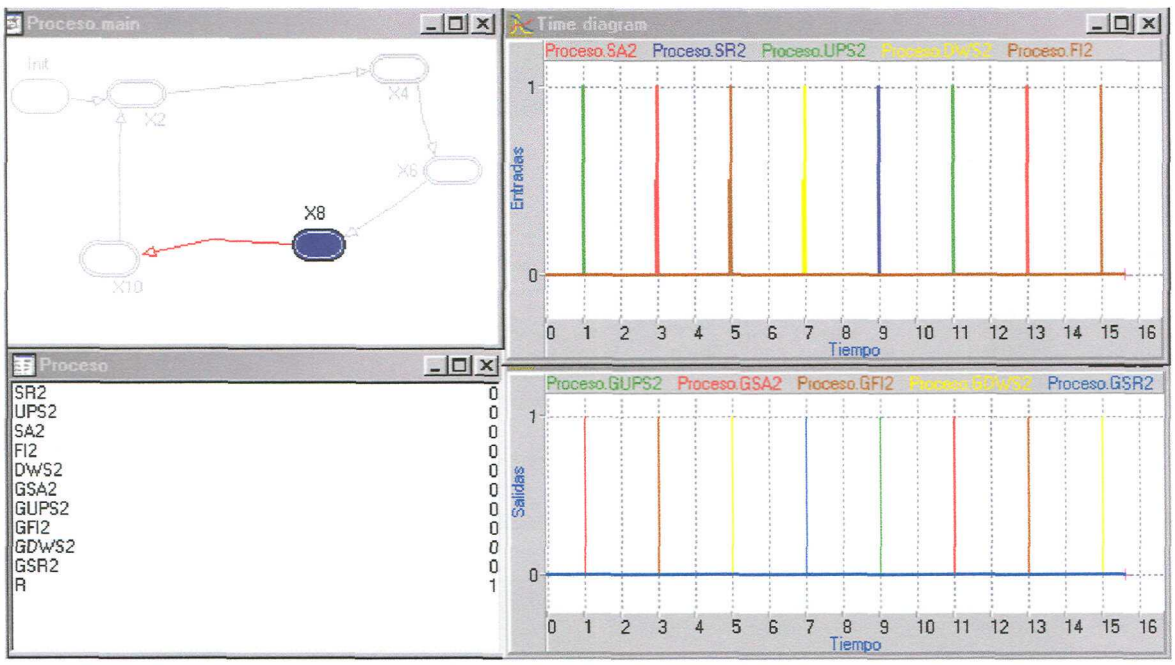


Diagrama de simulación del modelo autómata discreto:

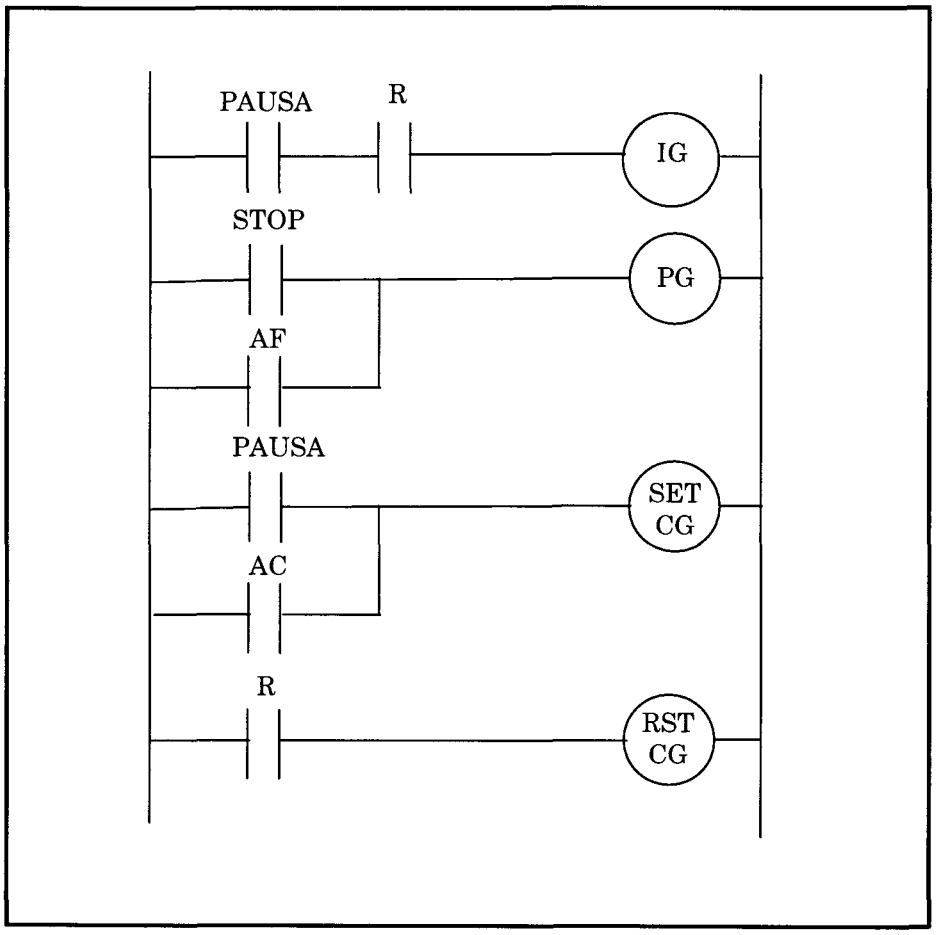


Conclusión del análisis:

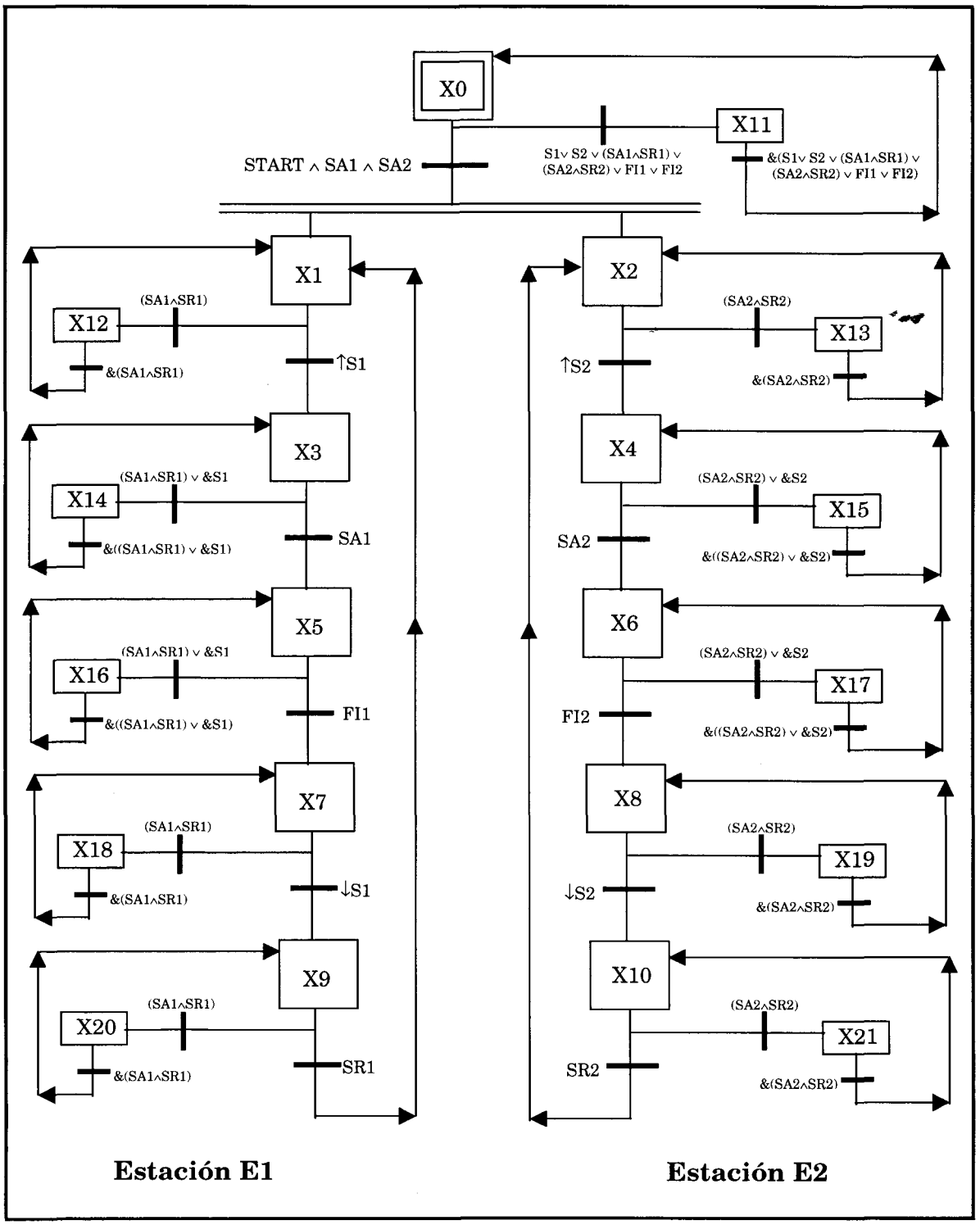
- No presenta deadlocks.
- Tiene buena viveza.

Cuarta Etapa: Obtener el programa GRAFCET para la **I**mplementación en un PLC.

Sección preeliminar.



Sección Grafcet.

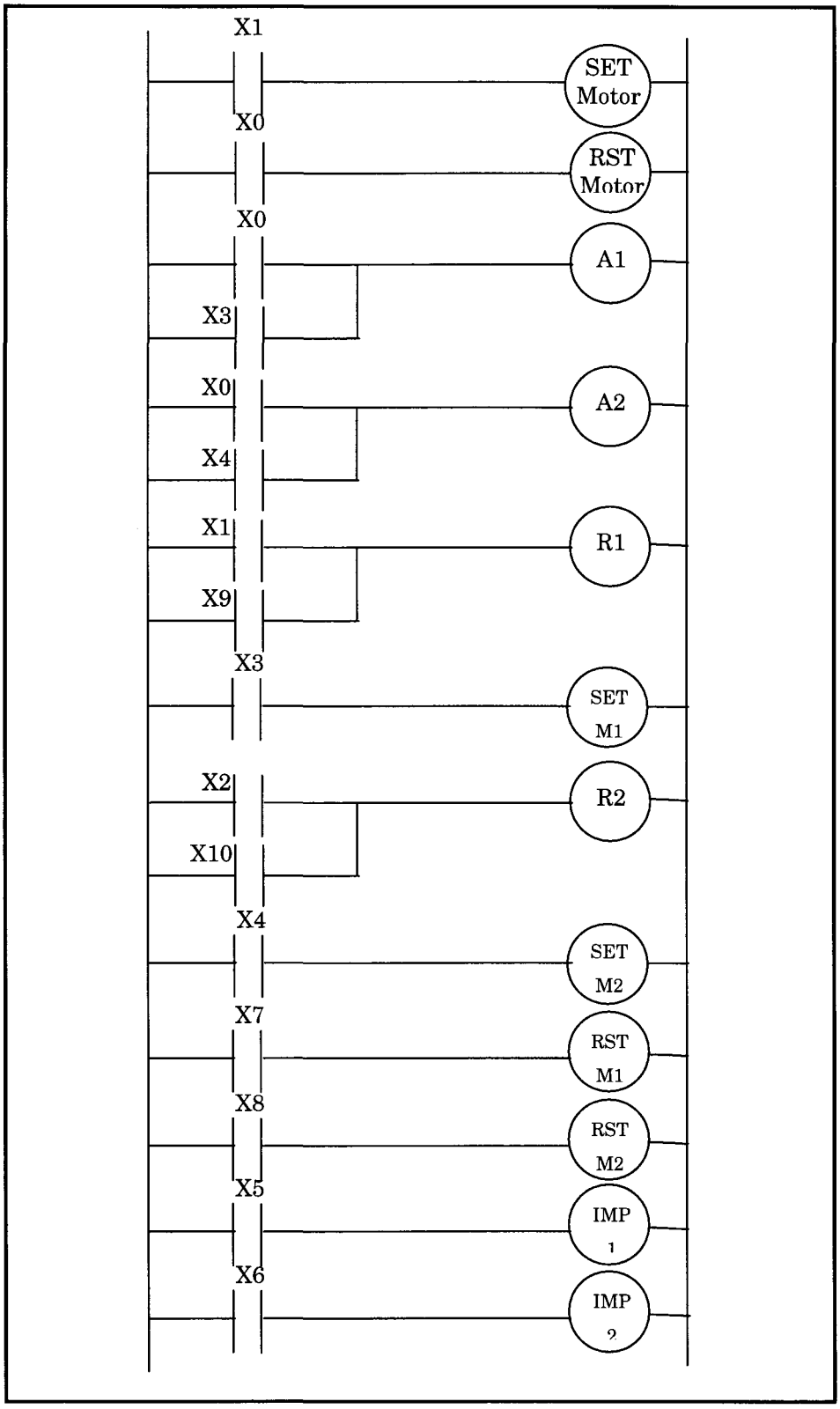


Sección Posterior.

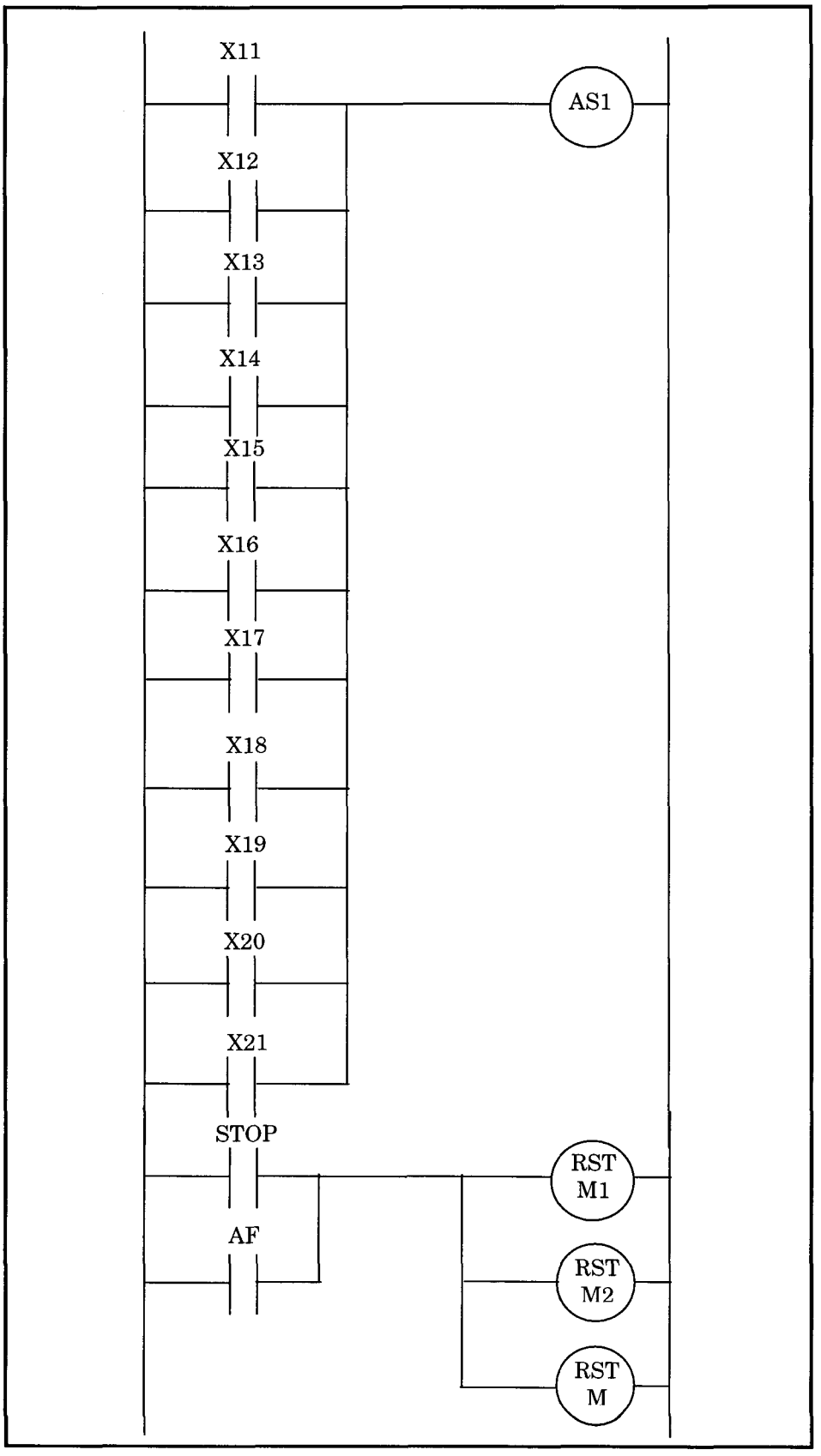
Tabla para relacionar las salidas del PLC con las etapas de la sección Grafcet y sus respectivos retrasos:

Salidas del PLC	***Descripción de la Salida del PLC***	***Función lógica booleana***
Salida 1	SET del Motor	X1
Salida 1	RST del Motor	X0
Salida 2	Activar parador 1 (A1)	X0 ∨ X3
Salida 3	Retroceder parador 1 (R1)	X1 ∨ X9
Salida 4	Activar parador 2 (A2)	X0 ∨ X4
Salida 5	Retroceder parador 2 (R2)	X2 ∨ X10
Salida 6	SET magneto 1 (M1)	X3
Salida 6	RST magneto 1 (M1)	X7
Salida 7	SET magneto 2 (M2)	X4
Salida 7	RST magneto 2 (M2)	X8
Salida 8	Habilitar impresora 1 (IMP1)	X5
Salida 9	Habilitar impresora 2 (IMP2)	X6
Salida 10	Sirena de alarmas (AS1)	X11 ∨ X12 ∨ X13 ∨ X14 ∨ X15 ∨ X16 ∨ X17 ∨ X18 ∨ X19 ∨ X20 ∨ X21

Diagrama escalera para la sección posterior:



Continuación del diagrama escalera de la sección posterior:

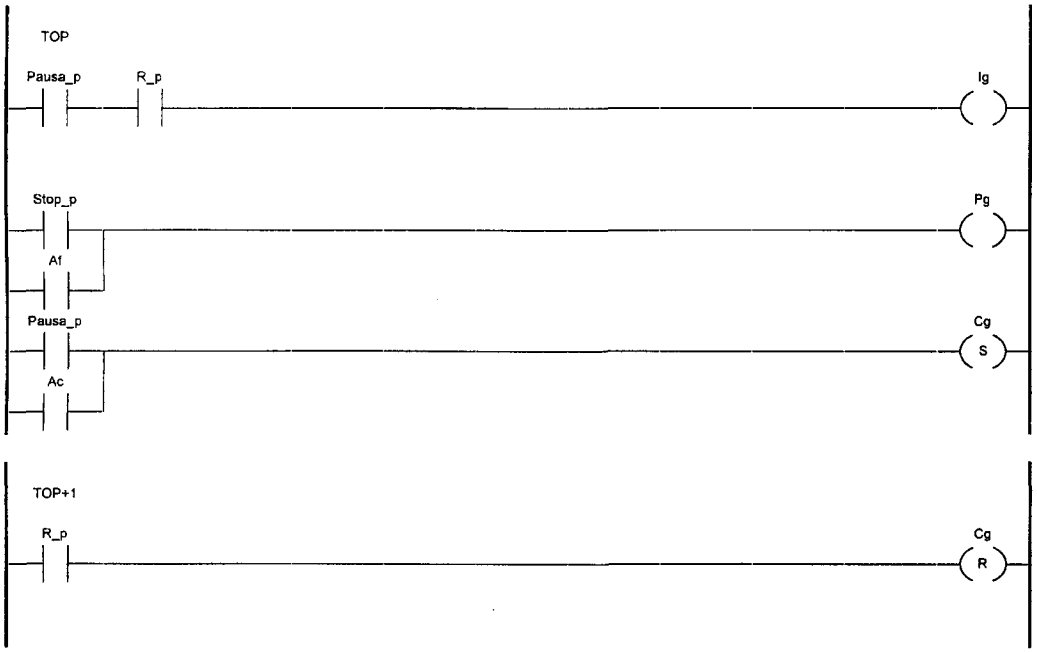


ANEXO 5

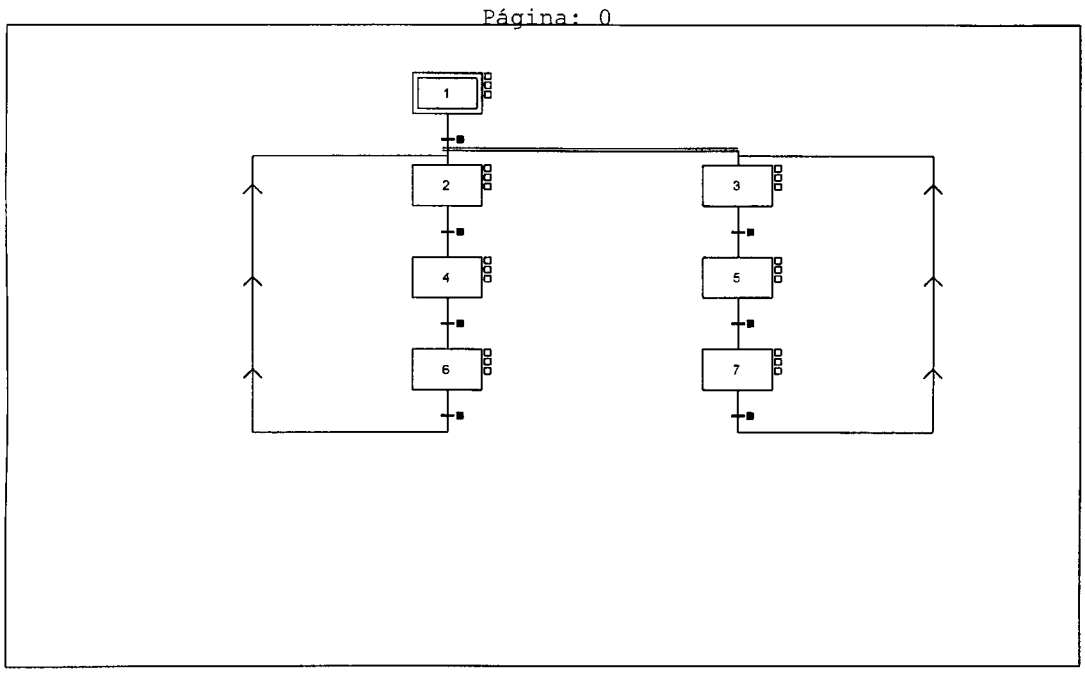
Listado de las secciones preliminar, Grafcet y posterior, obtenidas con la metodología IPTG en GRAFCET para el PLC TSX-3705.

Listado del programa en GRAFCET obtenido para la metodología IPTG.

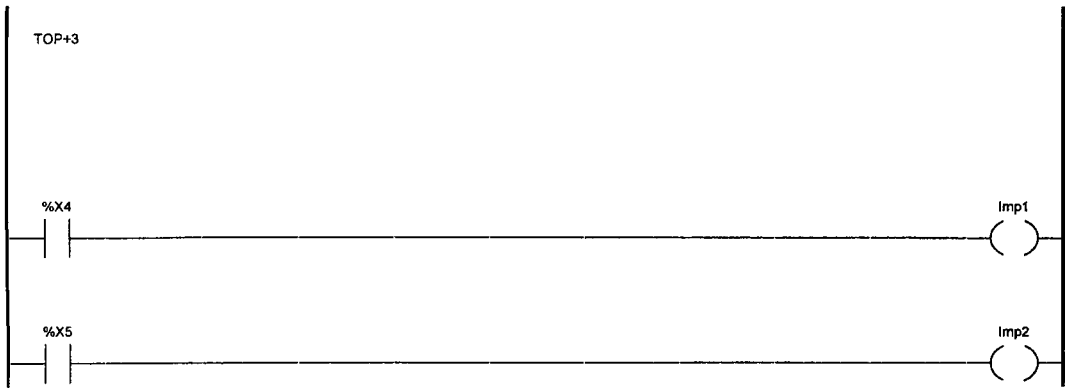
Sección Preliminar:

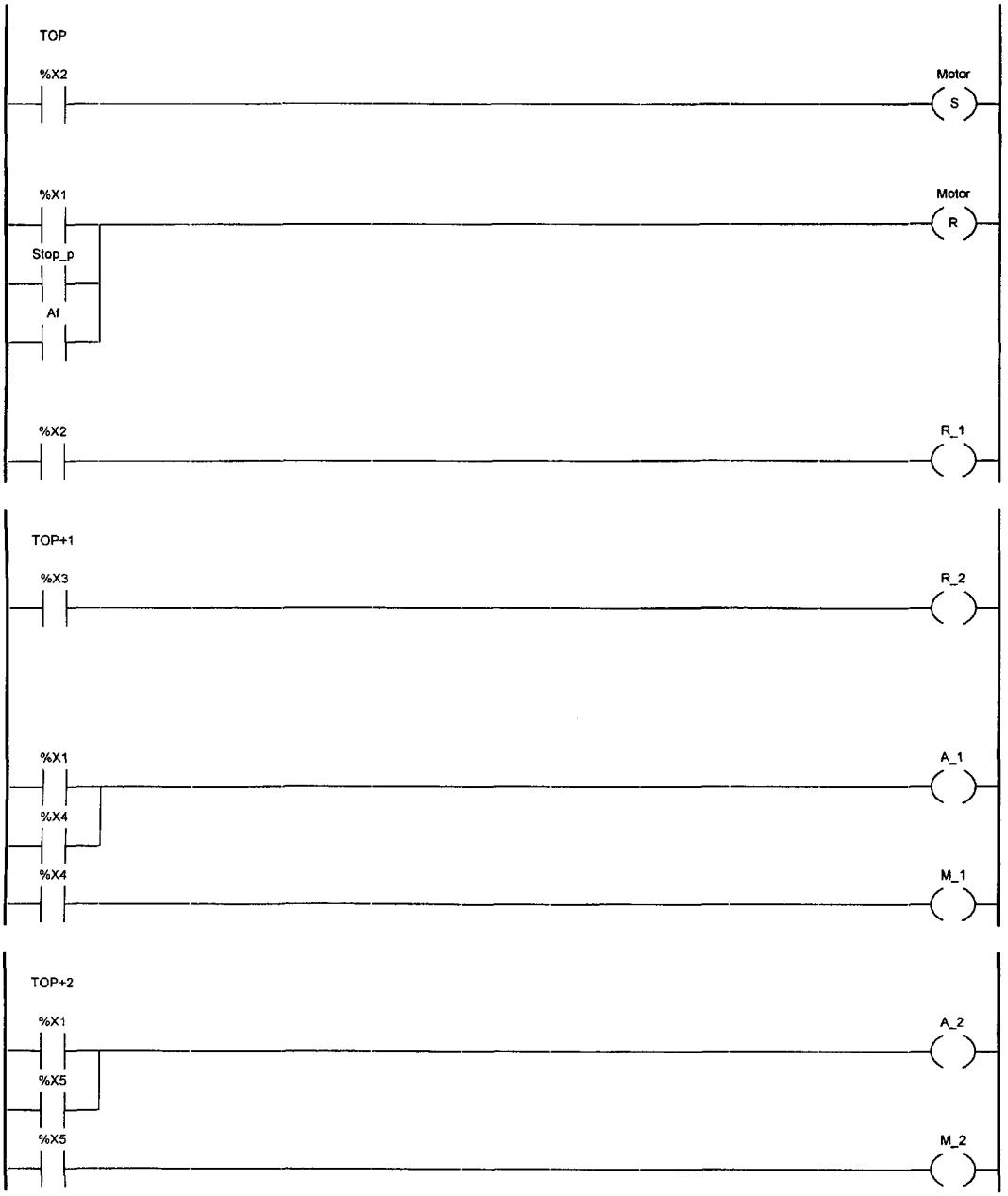


Sección Grafcet:



Sección Posterior:



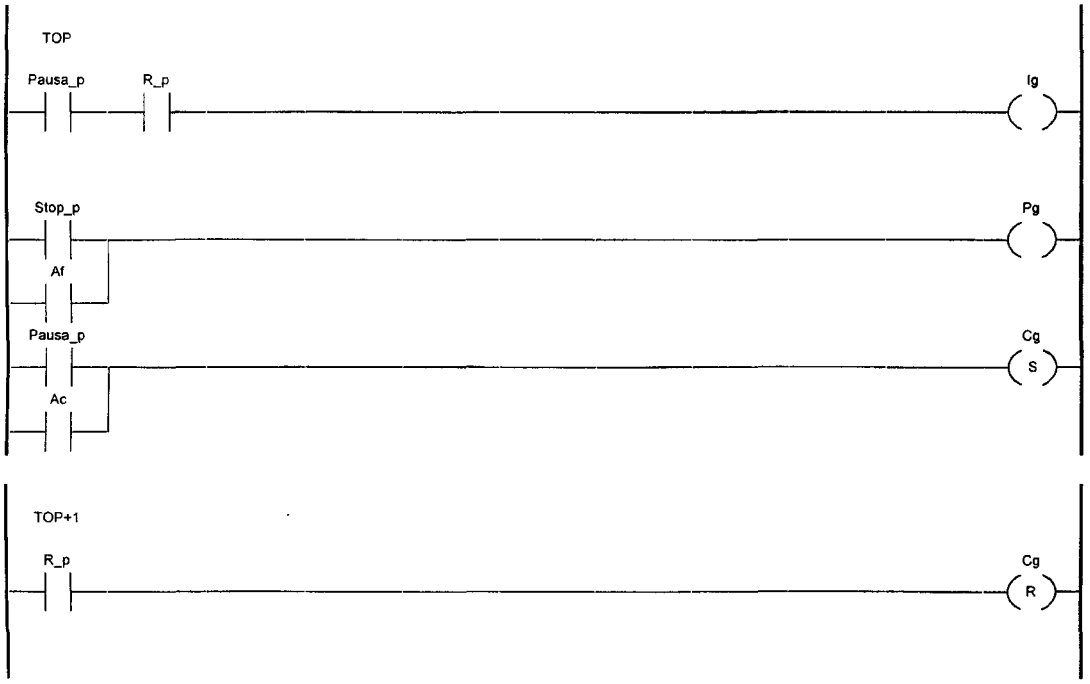

TOP
%X2
Motor
S
%X1
Motor
R
Stop_p
Af
%X2
R_1
TOP+1
%X3
R_2
%X1
A_1
%X4
%X4
M_1
TOP+2
%X1
A_2
%X5
%X5
M_2

ANEXO 6

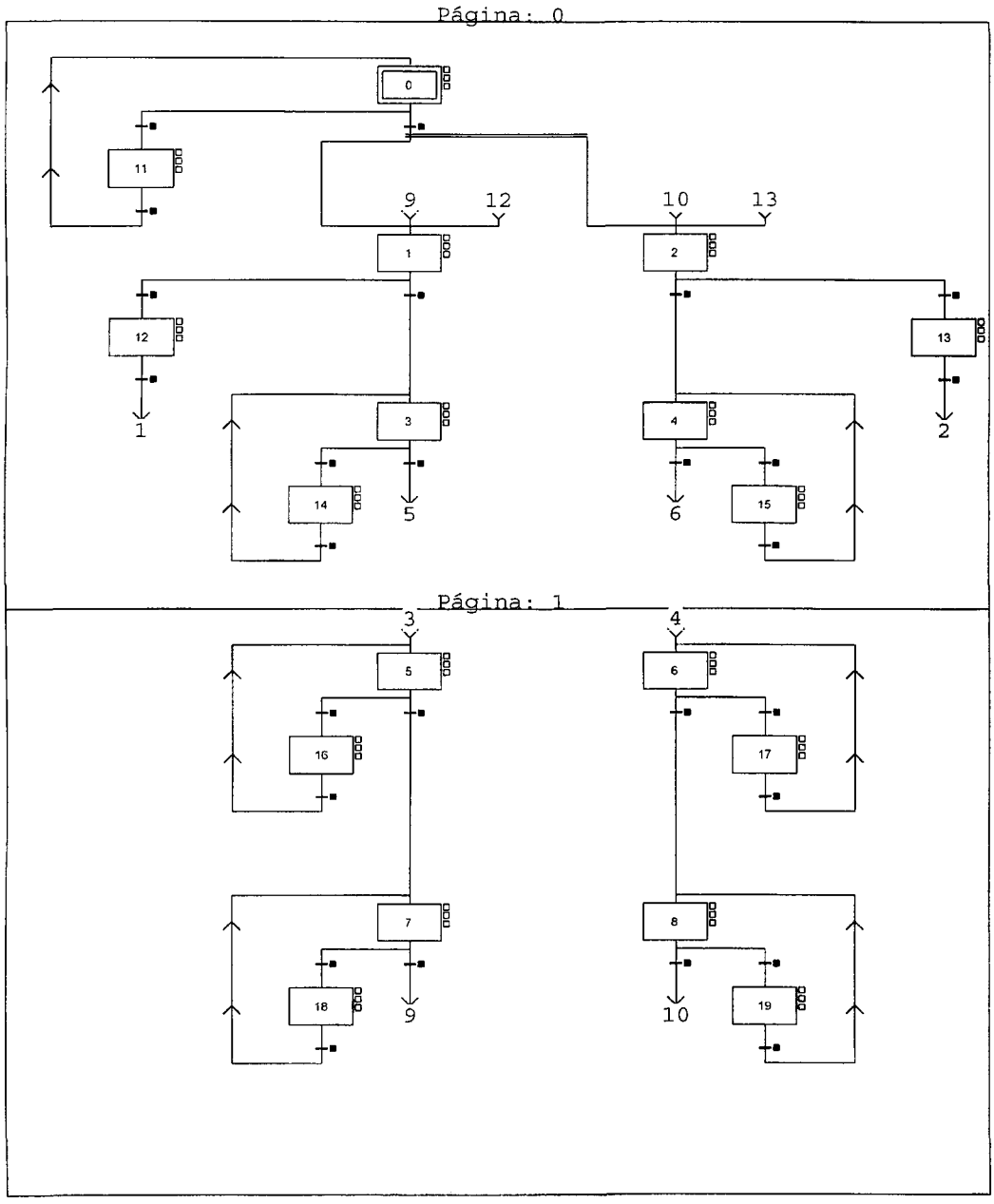
Listado de las secciones preliminar, Grafcet y posterior, obtenidas con la metodología RIMAnI en GRAFCET para el PLC TSX-3705.

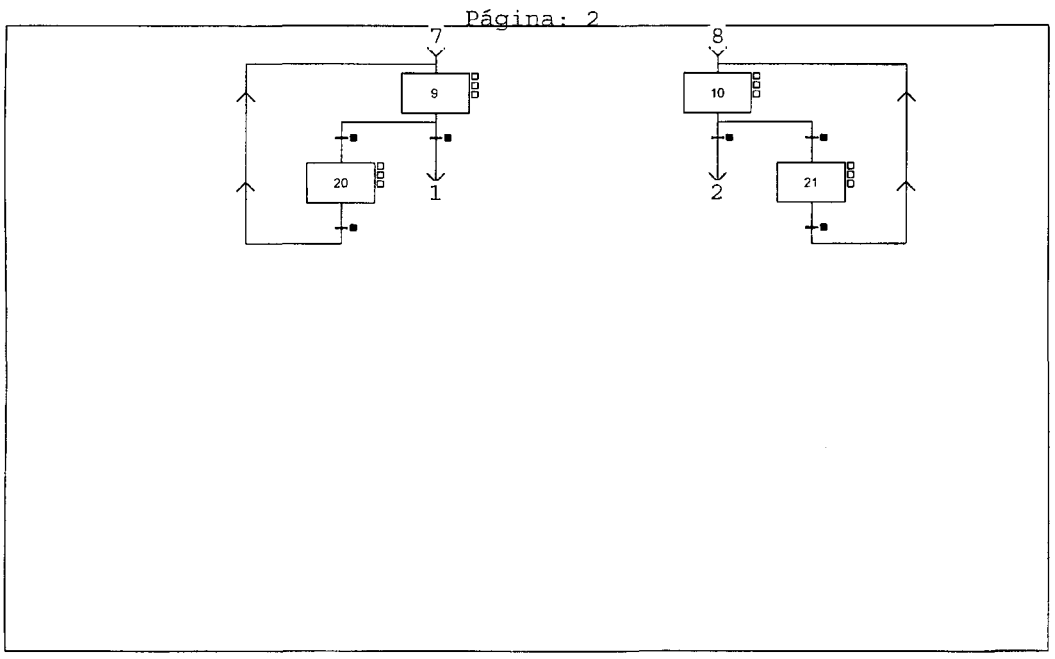
Listado del programa en GRAFCET obtenido para la metodología RIMAnI.

Sección Preliminar:

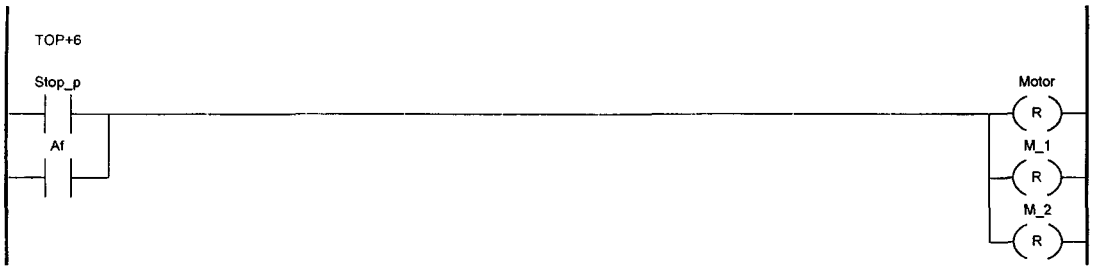


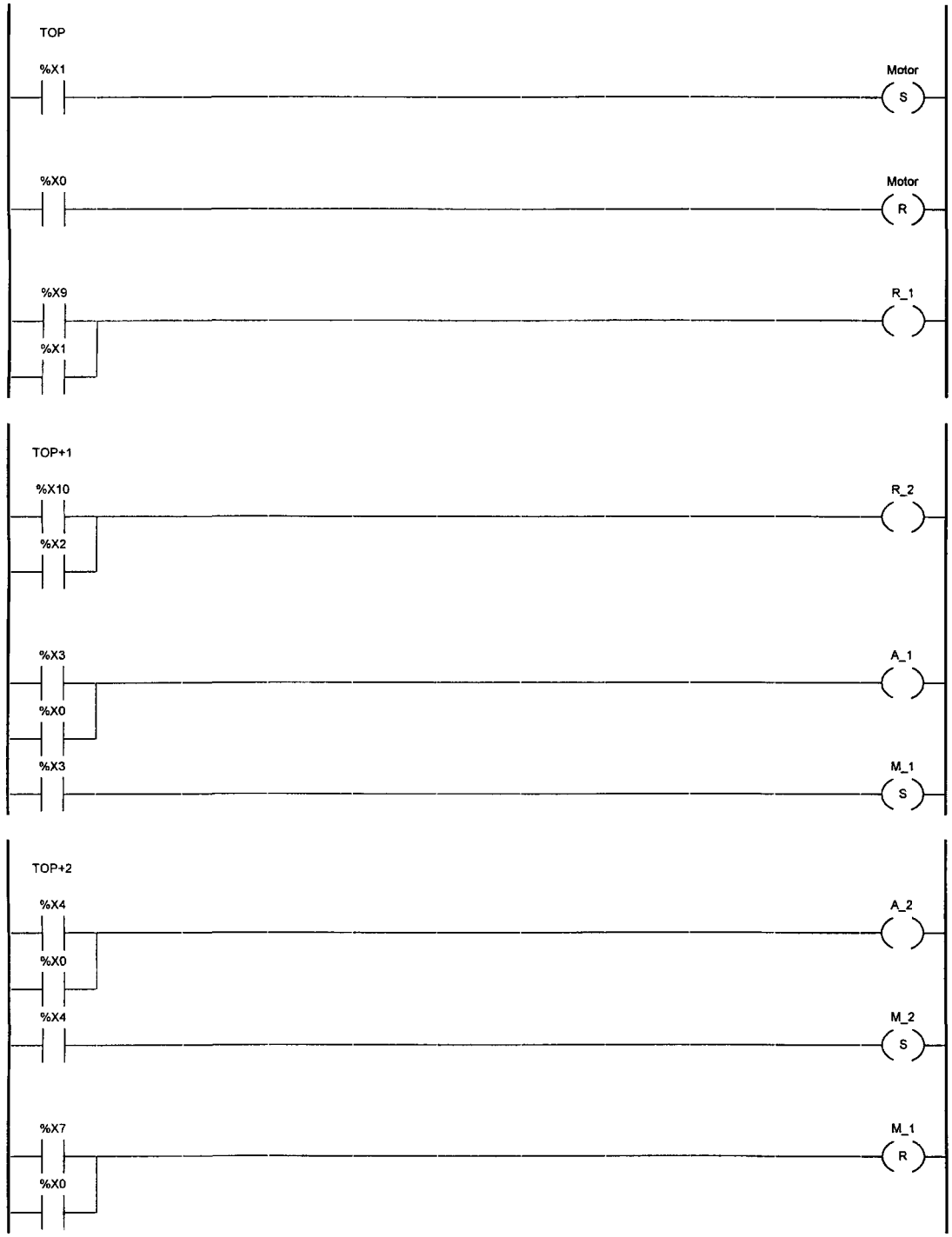
Sección Grafcet:

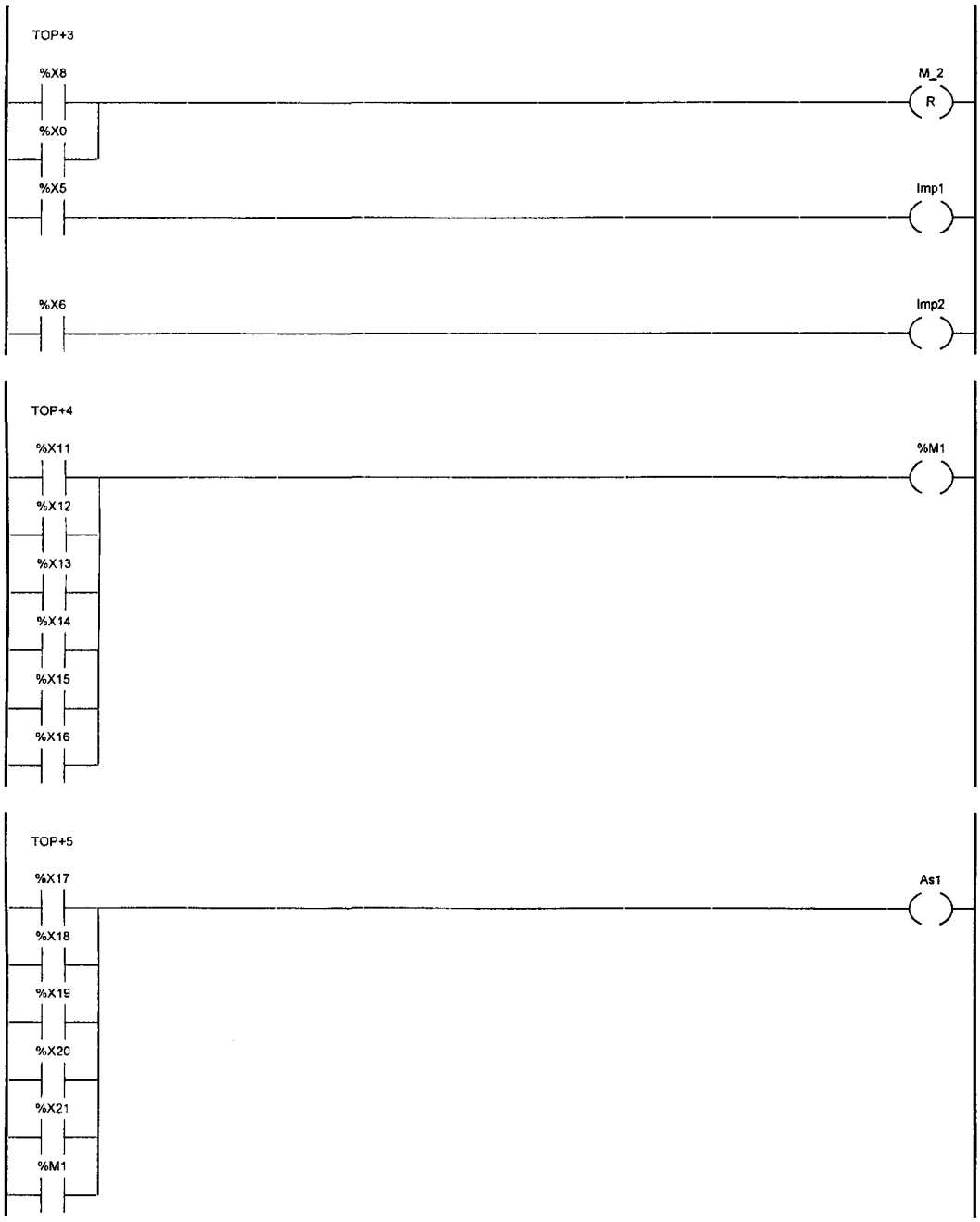




Sección Posterior:




TOP
%X1
Motor
S
%X0
Motor
R
%X9
%X1
R_1
TOP+1
%X10
%X2
R_2
%X3
%X0
A_1
%X3
M_1
S
TOP+2
%X4
%X0
A_2
%X4
M_2
S
%X7
%X0
M_1
R


TOP+3
%X8
%X0
%X5
%X6
M_2
R
Imp1
Imp2
TOP+4
%X11
%X12
%X13
%X14
%X15
%X16
%M1
TOP+5
%X17
%X18
%X19
%X20
%X21
%M1
As1

Bibliografía

Man, and Cybernetics, Vol. 29, No. 3. IEEE Mayo 1999.

[1] Alur R., *Timed Automata,* University of Pennsylvania and Bell Labs.

[2] Brandin B.A., Wonham W.M., *Supervisory Control of Timed Discrete-Event Systems,* IEEE Transactions on Automatic Control, Vol. 39, No. 2, Febrero 1994.

[3] Cassandras C.G., Lafortune S., *Introduction to Discrete Event Systems.* KLUMER, Boston, Dordrecht, London, 1999.

[4] Cassandras C.G., *Discrete Event Systems Modelling and Performance Analysis.* IRWIN, Boston, Massachusetts, 1993.

[5] Charbonnier F., *These Commande supervisée des systèmes à événements discrets.* Instituto Nacional Politécnico de Grenoble, 1996.

[6] Charbonnier F., Alla H., David R., *The Supervised Control of Discrete-Event Dynamic Systems,* IEEE Transactions on Control Systems Technology, Vol. 7, No. 2, Marzo 1999.

[7] David R., Alla H., *Petri Nets & GRAFCET Tools for Modelling Discrete Event Systems.* Prentice Hall NY, London, Toronto Sydney, Tokyo, Singapore, 1992.

[8] David R., *Grafcet: A Powerful Tool for Specification of Logic Controllers,* IEEE Transactions on Control Systems Technology, Vol. 3, No. 3, Septiembre 1995.

[9] David R., Alla H., *Petri Nets for Modelling of Dynamic Systems A Survey,* Automatica Vol. 30, No. 2, 1994.

[10] Department of Automatic Control and Systems Engineering, *Visual Object Oriented Petri Net based Engineering Tool.* Ilmenau University of Technology. Ilmenau, Alemania.

[11] Duraeva O., Dmitrieva E., Tsapko G., *Activity Analysis of Trade Firm by IDEF0 Methodology.* Chair of Automation and Computer Systems, Tomsk Polytechnic University. Tomsk, Rusia. IEEE 2001.

[12] Inihov D. , Kolesov Y., Senichenkov Y. , *Model Vision Studium Version 3.0.17.* Experimental Object Technologies. Rusia, 1999.

[13] Le Parc P., L´Her D., Scharbarg J., Marcé L., *Grafcet Revisited with a Synchronous Data-Flow Language.* IEEE Transactions on Systems, Man, and Cybernetics, Vol. 29, No. 3. IEEE Mayo 1999.

[**14**] Lee J.S., Hsu P.L., *A PLC-Based Design for the Sequence Controller in Discrete Event Systems.* Department of Electrical and Control Engineering. National Chiao-Tung University, Taiwan. IEEE 2000.

[**15**] Mallaband S., *Specification of Real Time Control Systems by Means of Sequential Function Charts,* Bass Brewers Ltd., Reino Unido.

[**16**] Porras A., Montanero A.P., *Autómatas Programables,* Mc Graw Hill, México 1990.

[**17**] Ramadge, P.J. Wonham W.M., *Supervisory Control of a Class of Discrete Event Processes,* SIAM J. Control and Optimization, Vol. 25, No. 1, Enero 1987.

[**18**] Ramadge P.J., Wohham W.M., *The Control of Discrete Event Systems,* Proceedings of the IEEE, Vol. 77, No. 1, Enero 1989.

[**19**] Romeral J., Balcells J., *Autómatas Programables,* Alfaomega Marcombo, México 1998.

[**20**] Sava T., Alla H., *These Commande par supervision des systèmes à événements discrets temporisés.* Instituto Nacional Politécnico de Grenoble, 2001.

[**21**] Sava T., Alla H., *An Approach for Supervised Control of Timed Discrete Event Systems.* Instituto Nacional Politécnico de Grenoble, 2001.

[**22**] Uzam M., Jones A.H., Ajlouni N., *Conversión of Petri Net Controllers for Manufacturing Systems into Ladder Logic Diagrams,* Intelligent Machinery Division University of Salford, IEEE, 1996.