APLICACIÓN BASADA EN REDES DE PETRI PARA LA PROGRAMACIÓN AUTOMÁTICA DE PLCs EN PROCESOS BATCH

Jorge Eliecer Calvo Giraldo
Mitchel Esteban Collazos Agredo

ANEXOS

Director:
PhD. **Carlos Alberto Gaviria López**

Universidad del Cauca
Facultad de Ingeniería Electrónica y Telecomunicaciones
Departamento de Electrónica, Instrumentación y Control
Ingeniería en Automática Industrial
Popayán
2015

TABLA DE CONTENIDO

ANEXO A

A.1 Fase cargar_Agua

A.1.1 Lógica Running para la fase *Cargar_Agua*

La condición para dar inicio a este estado es que la válvula se debe abrir antes de que la bomba sea encendida. Esta lógica se muestra en la figura A.1

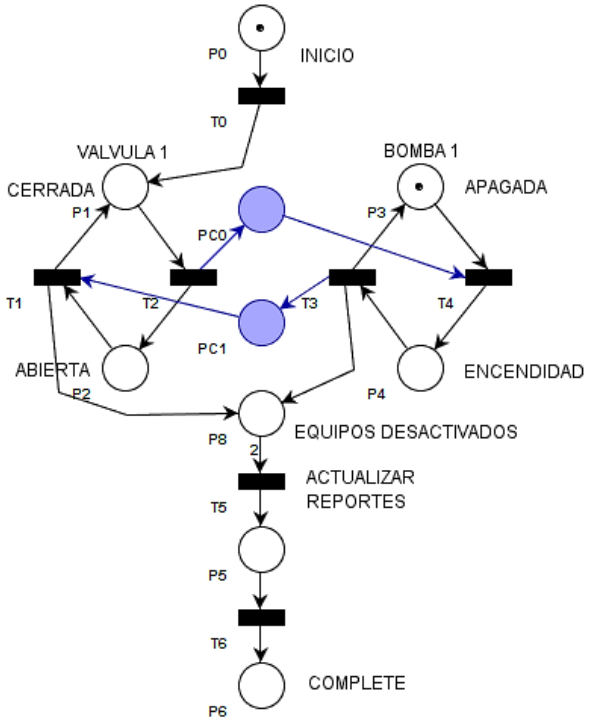


Figura A.1 Lógica Running – *Cargar_Agua*

Una vez la bomba se haya encendido, y la cantidad requerida de agua fue transferida la bomba debe apagarse y posteriormente la válvula deberá cerrarse. Estas condiciones se definen como:

$$v4 - v2 \leq 0$$

$$v1 - v3 \leq 0$$

Por otro lado, el código de programación de esta fase está representado en la figura A.2. Sin embargo, debido a la complejidad de este código, solo es presentada una parte de él.

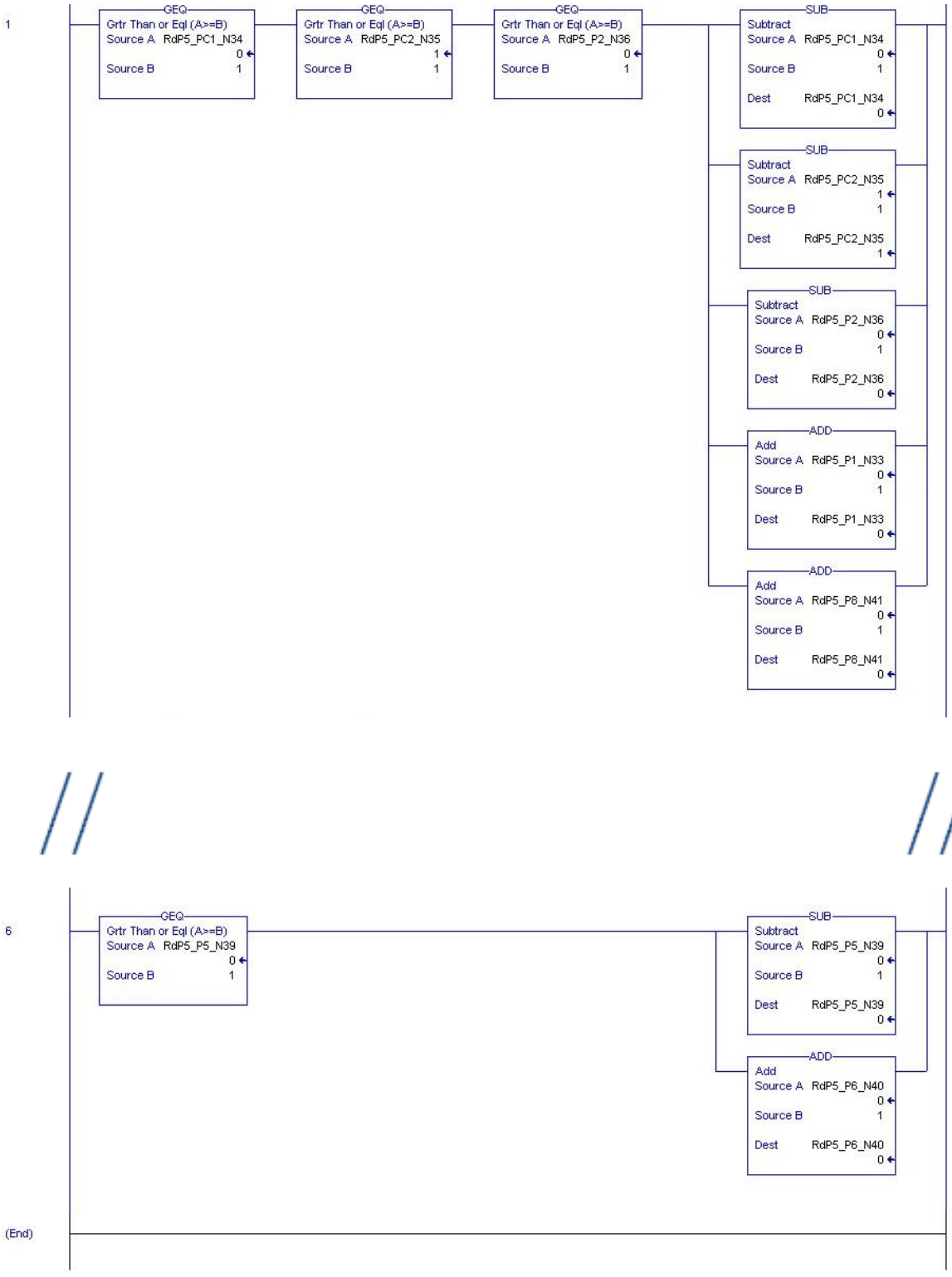


Figura A.2 Programa Ladder para Running – *Cargar_Agua*

A.2 Fase *cargar_EDA*

A.2.1 Logica Running

Las condiciones que se deben cumplir para un correcto funcionamiento de la lógica son:

- La válvula se abre cuando el indicador de peso indica la cantidad de material deseada.
- La bomba enciende cuando la válvula está abierta.
- La válvula se cierra cuando la bomba se apaga.

Lo anterior se puede evidenciar en la figura A.3.

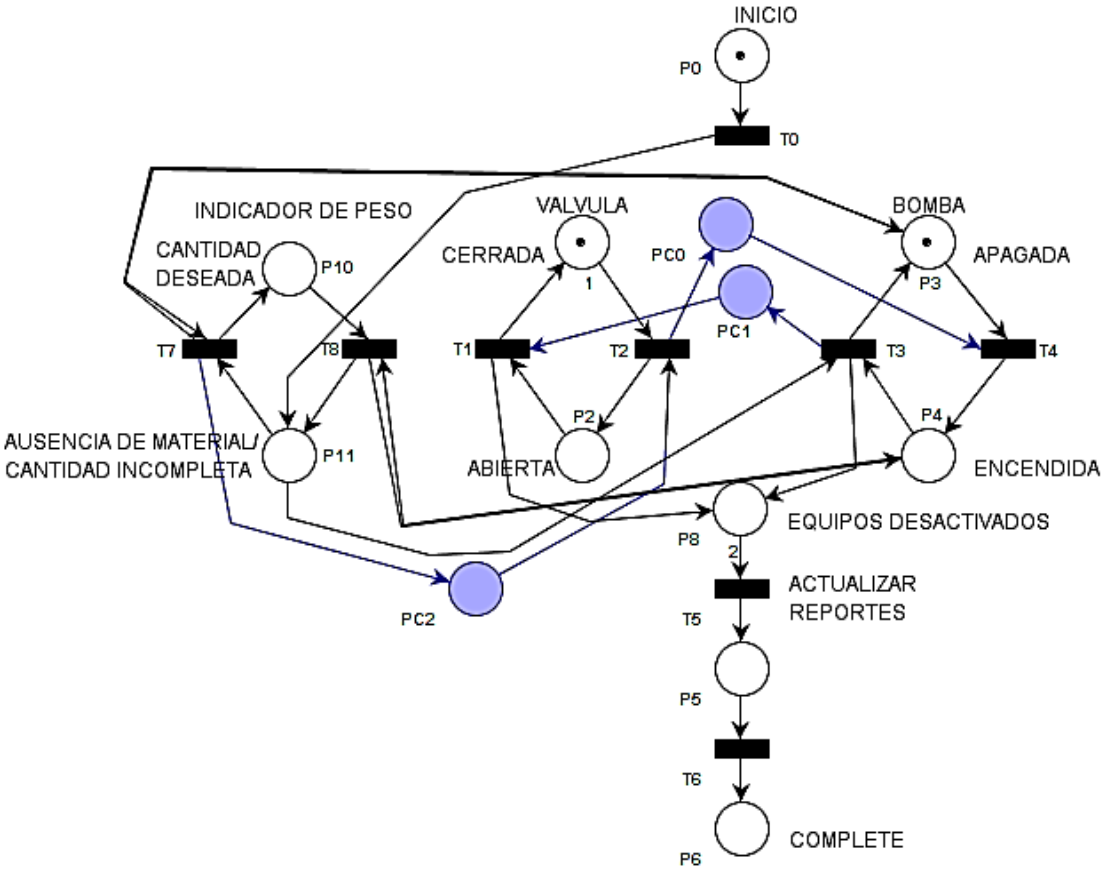


Figura A.3 Lógica Running - *Cargar_EDA*

De forma análoga, la evidencia del código de programación de esta fase está representada en la figura A.4.

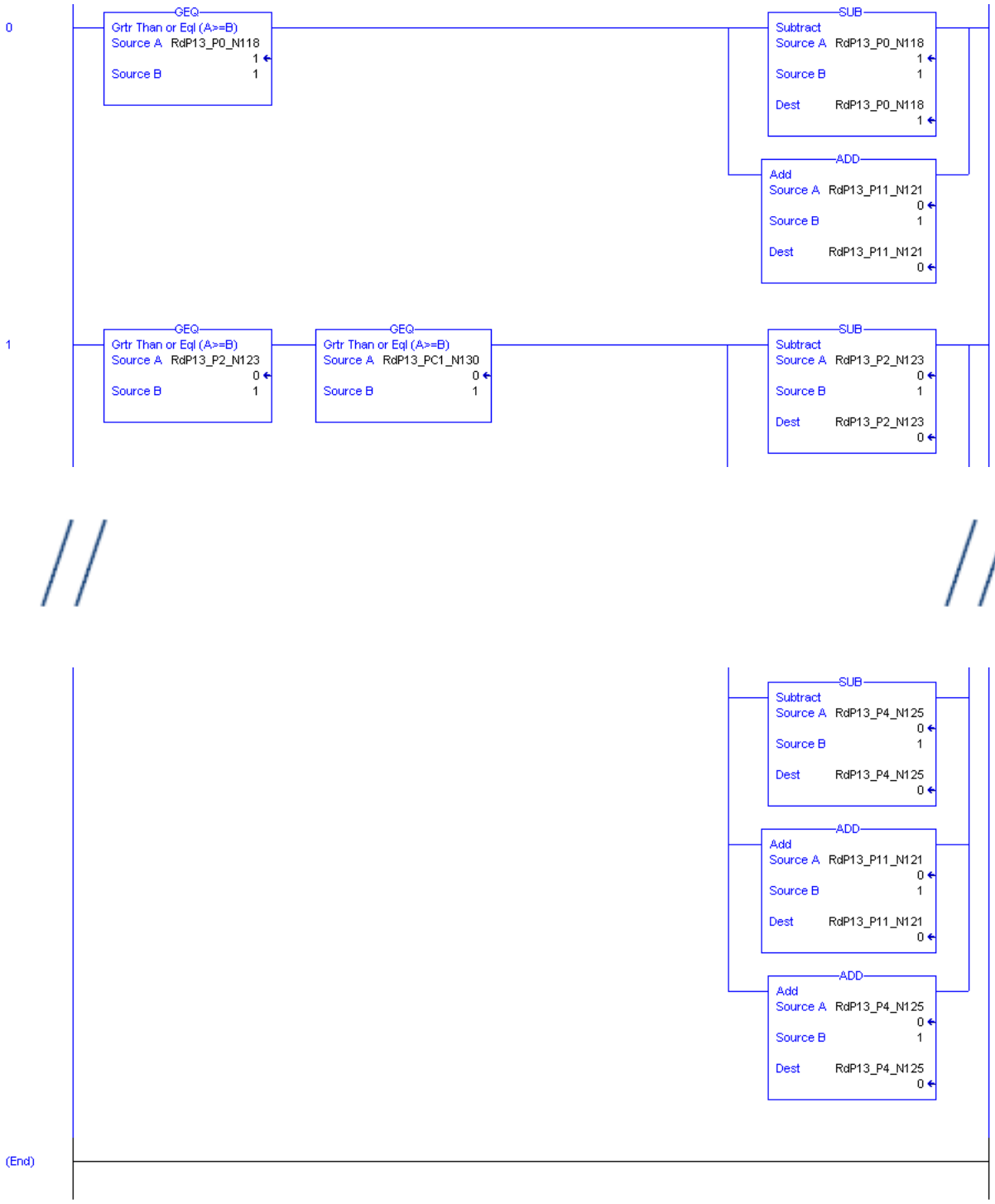


Figura A.4 Programa Ladder para Running – *Cargar_EDA*

A.2.2 Lógica Holding

Condición para el diseño de la lógica *Holding*:

- La válvula cierra cuando la bomba se apaga. Ver figura A.5

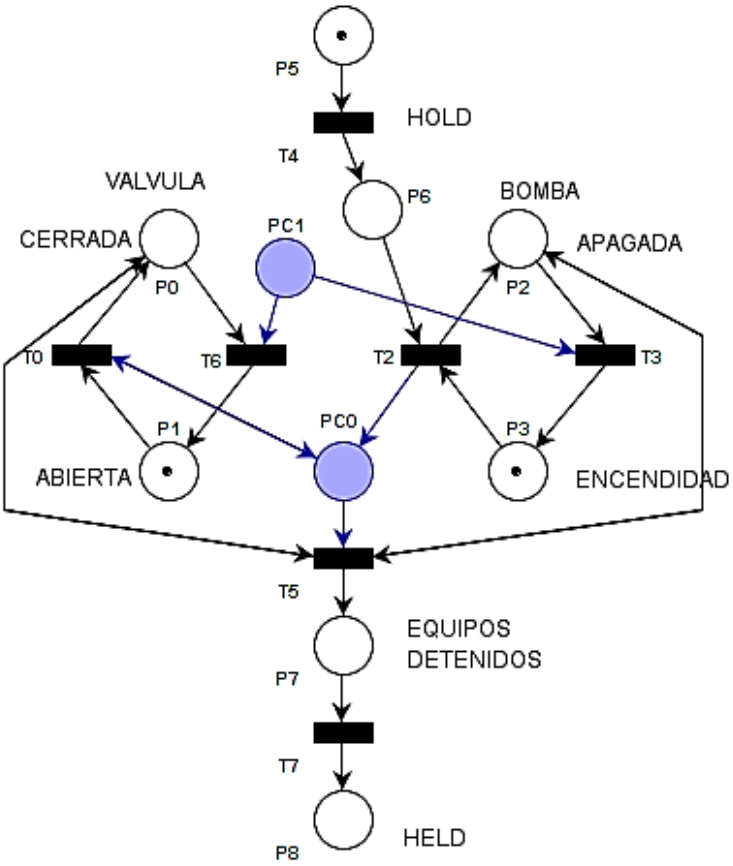


Figura A.5 Lógica Holding - *Cargar_EDA*

La evidencia del código de programación de esta fase está representada en la figura A.6.

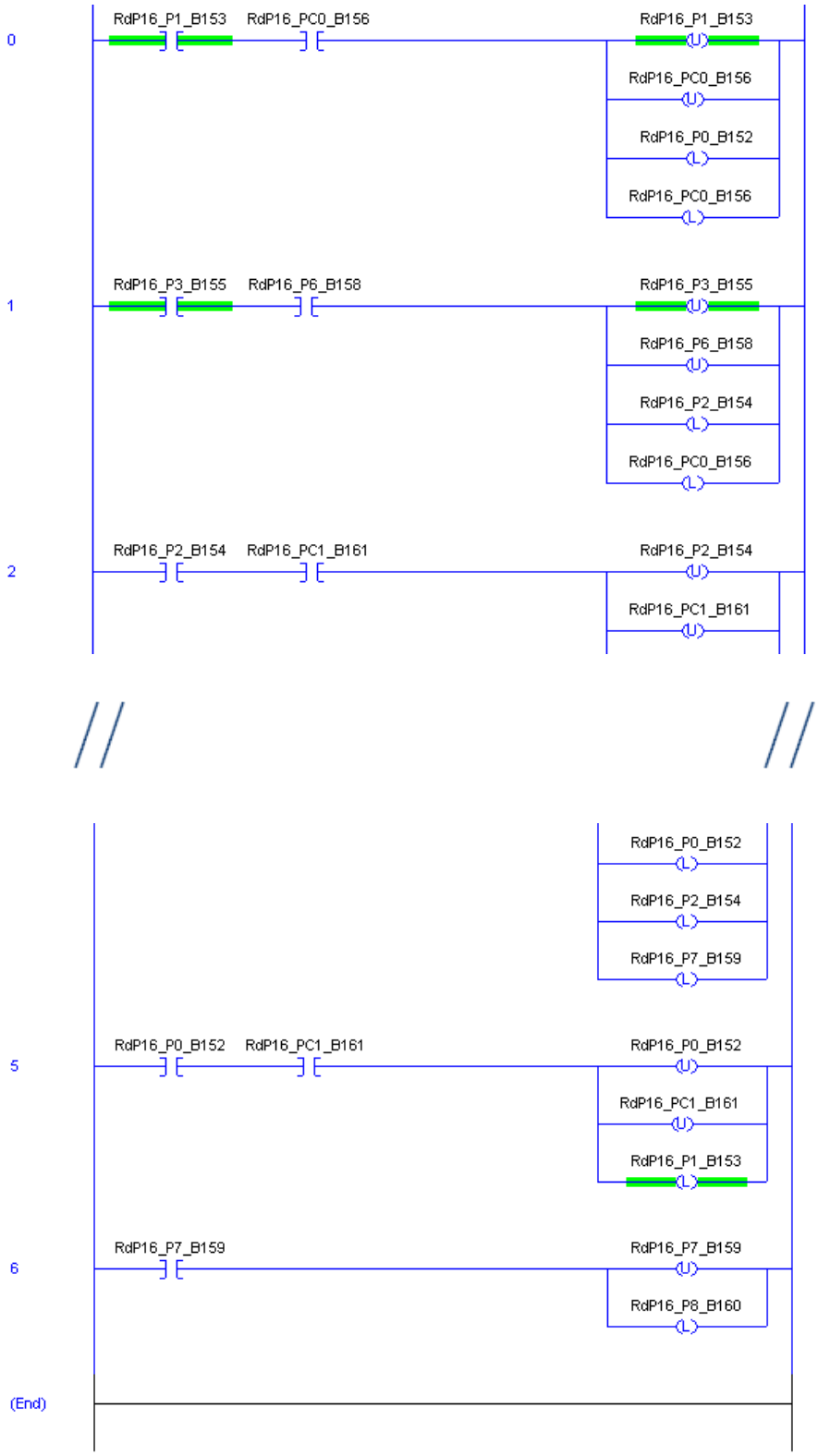


Figura A.6 Programa Ladder para Holding – *Cargar_EDA*

A.2.3 Lógica Stopping

La red de Petri para el diseño de la lógica *Stopping* es similar a la del estado *Holding*, con la diferencia de que se agrega una nueva etapa para la actualización de reportes, ver figura A.7.

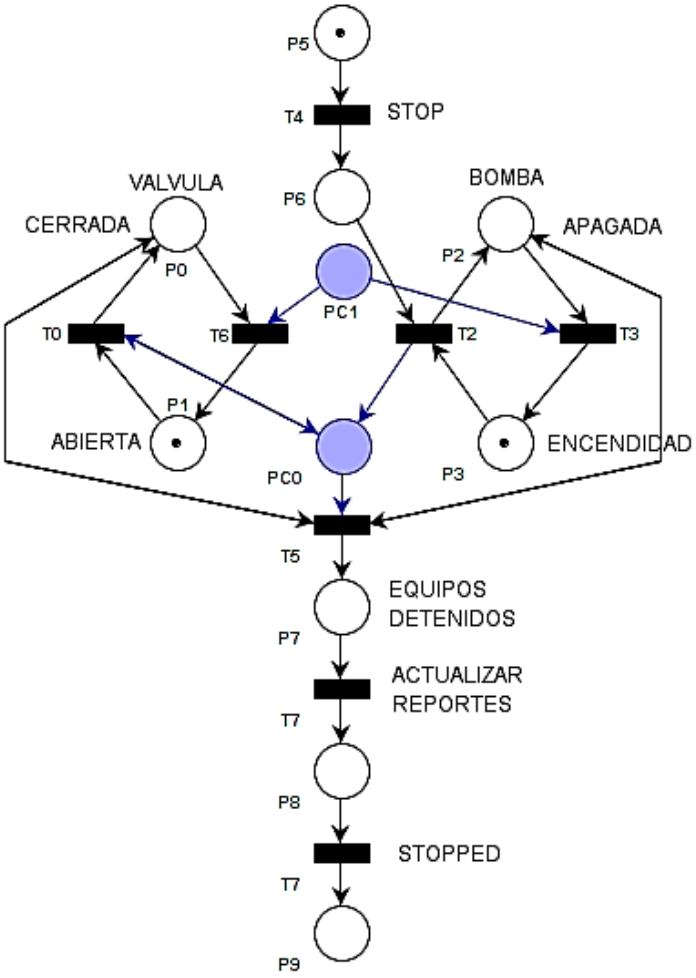


Figura A.7 Lógica Stopping - *Cargar_EDA*

La evidencia del código de programación de esta fase está representada en la figura A.8.

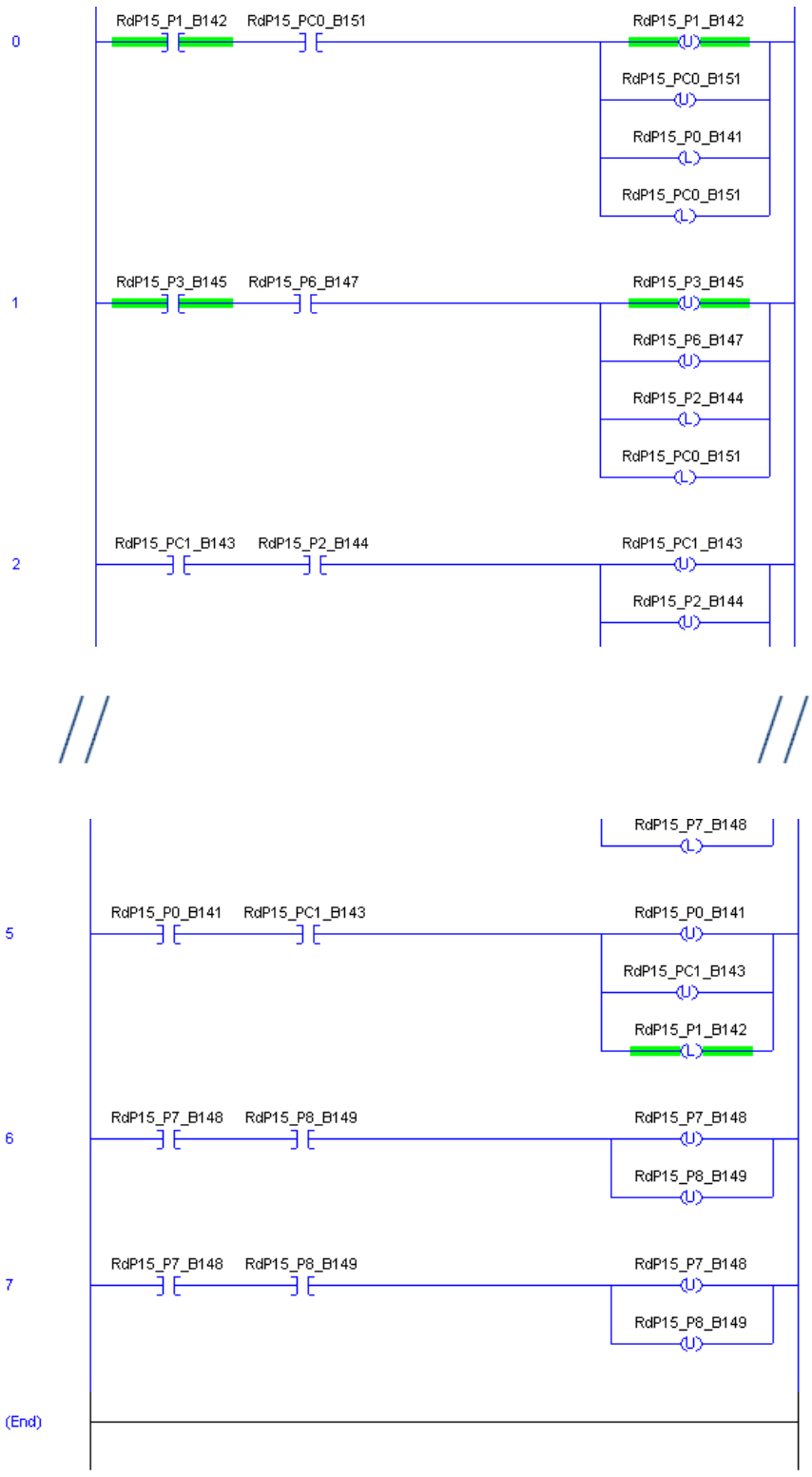


Figura A.8 Programa Ladder para Stopping – *Cargar_EDA*

A.2.4 Lógica Aborting

Esta lógica es descrita en la figura A.9

Las fases ***Cargar_CS2***, ***Cargar_NaOH***,y ***Transferir*** requieren los mismos módulos de control y su lógica requiere las mismas condiciones de funcionamiento que la fase ***Cargar_EDA***, por tanto su lógica puede ser replicada.

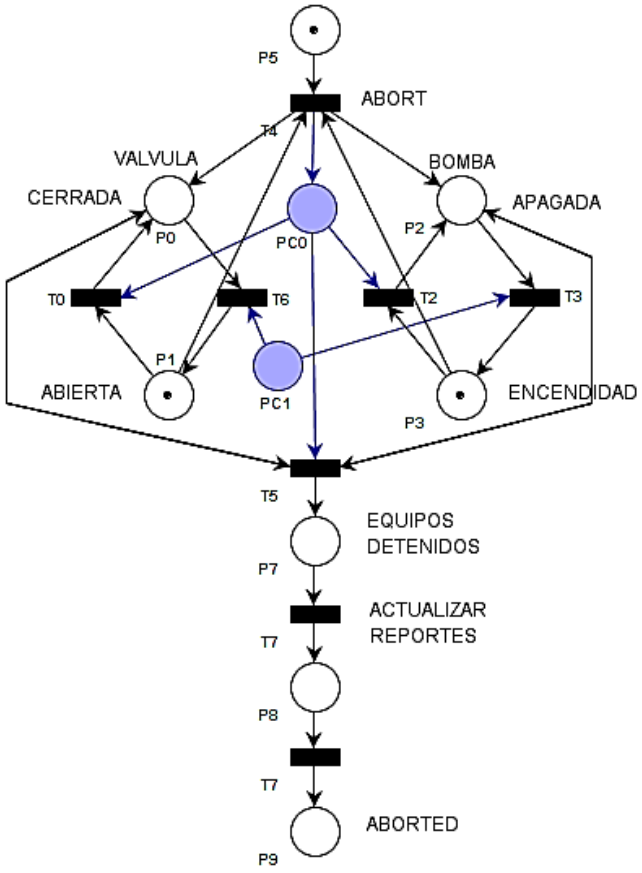


Figura A.9 Lógica Aborting - *Cargar_EDA*

La evidencia del código de programación de esta fase está representada en la figura A.10.

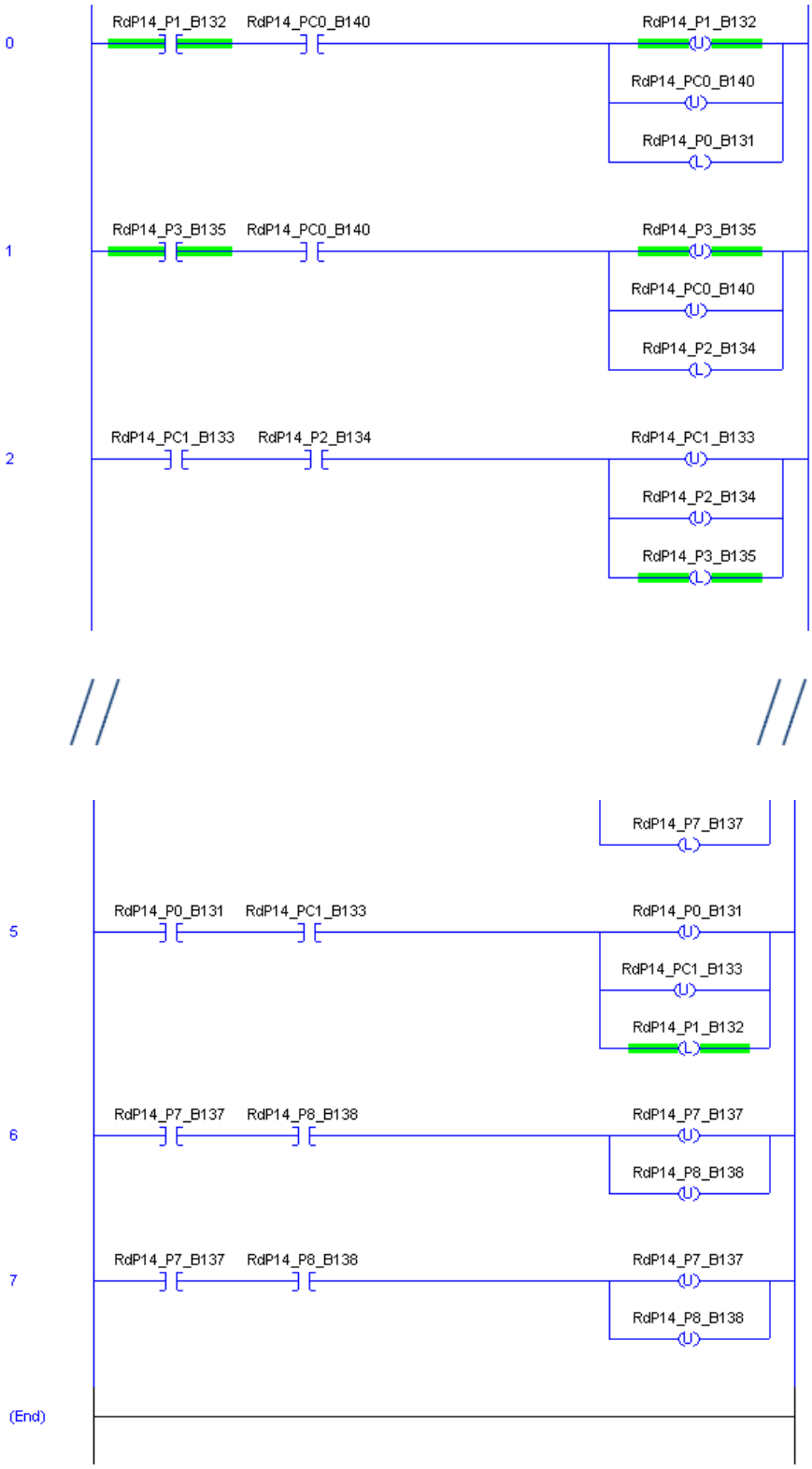


Figura A.10 Programa Ladder para Aborting – *Cargar_EDA*

A.3 Fase *Agitar*

El diseño de la lógica para esta fase resulta mucho más sencillo que la anterior debido a que solo cuenta con un módulo de control para su ejecución, en este caso un agitador. En la figuras A.11 a A.18 se describe la logica para cada uno de los estados de la fase.

A.3.1 Lógica Running

Para el inicio del agitador, el transmisor de nivel deberá marcar un nivel normal de material.

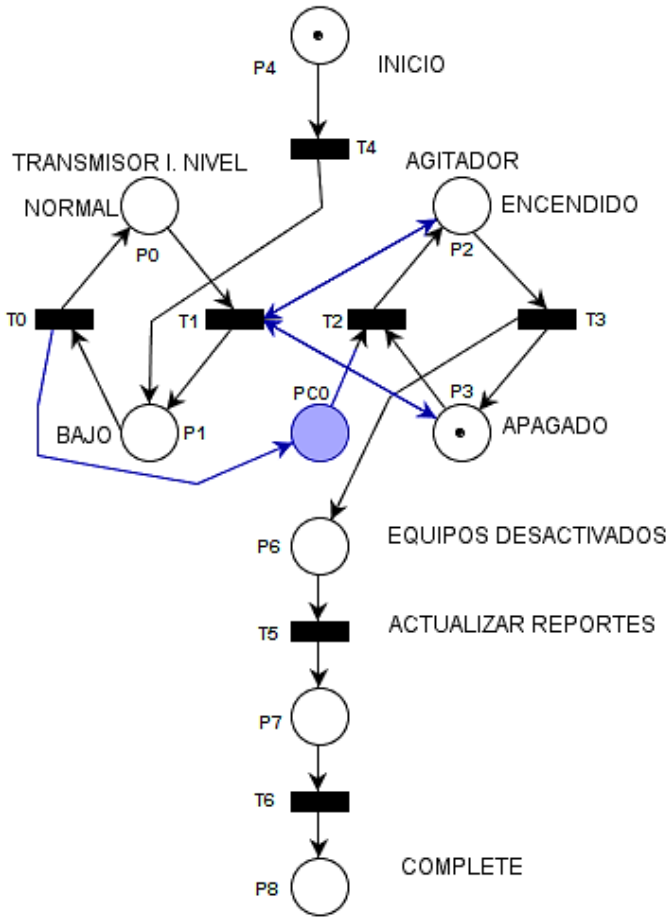


Figura A.11 Lógica Running – *Agitar*

La evidencia del código de programación de esta fase está representada en la figura A.12.

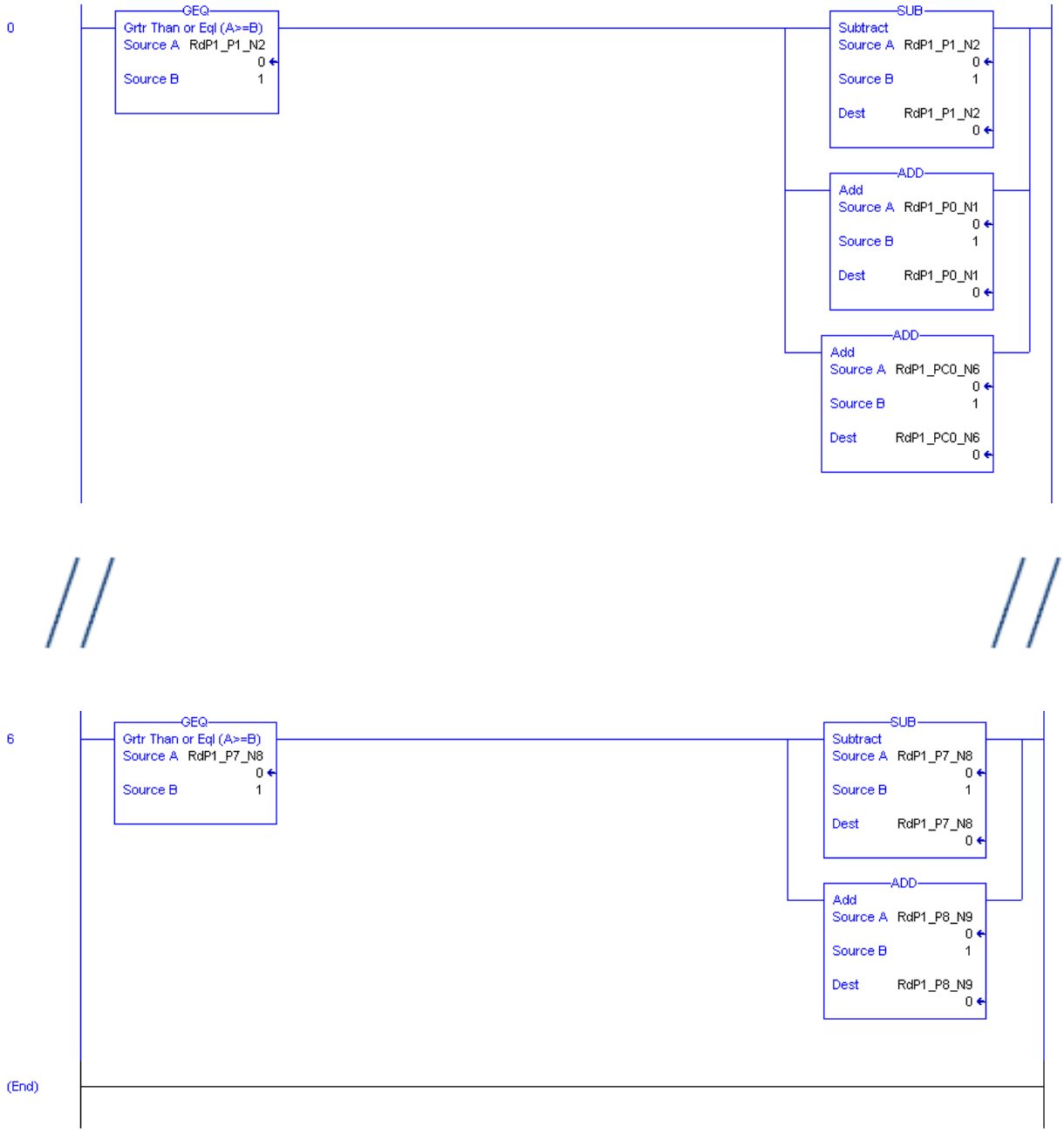


Figura A.12 Programa Ladder para Running – *Agitar*

A.3.2 Lógica Holding

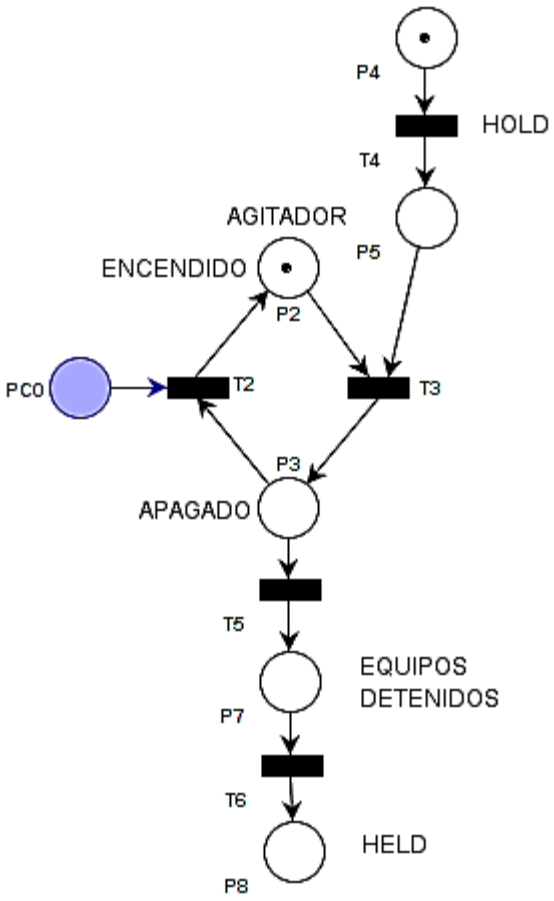


Figura A.13 Lógica Holding – ***Agitar***

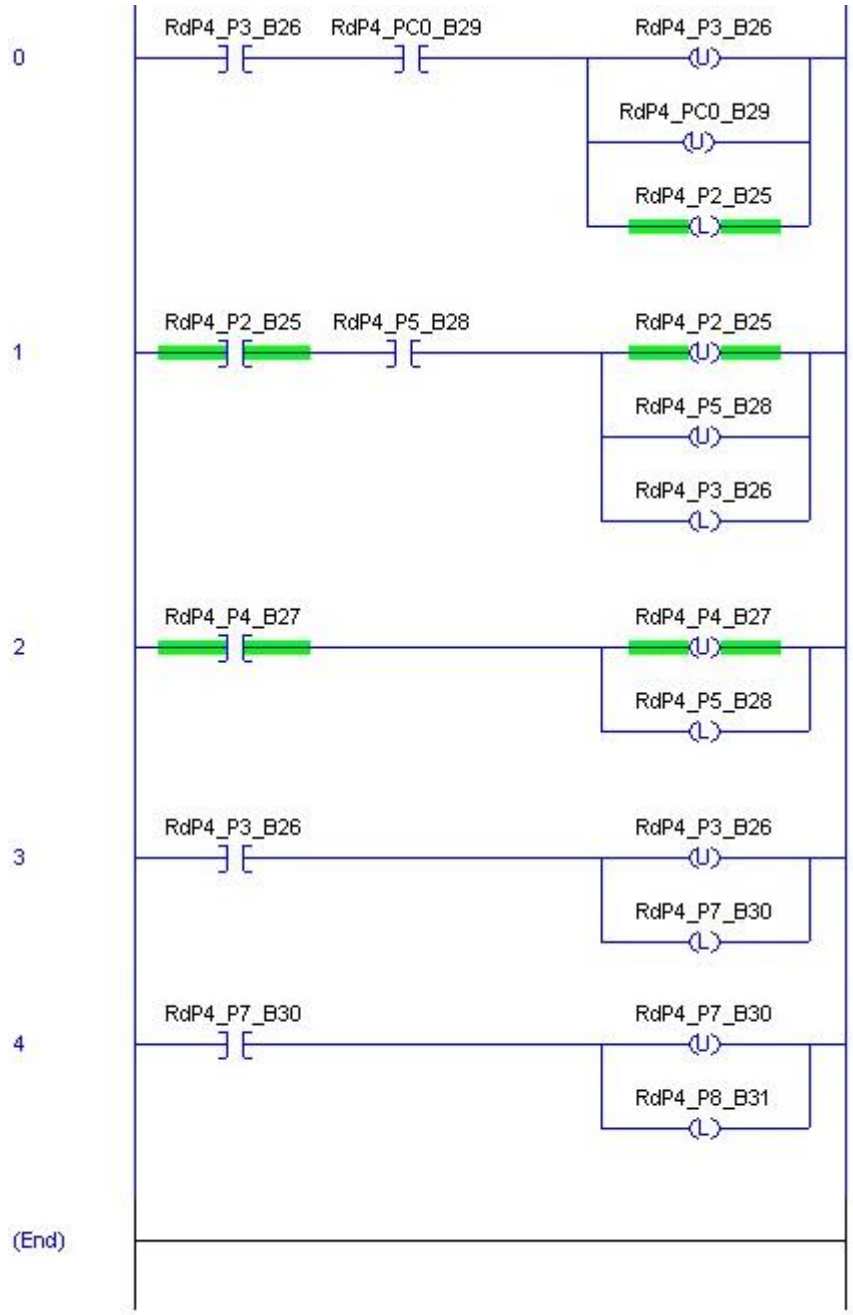


Figura A.14 Programa Ladder para Holding – *Agitar*

A.3.3 Lógica Stopping

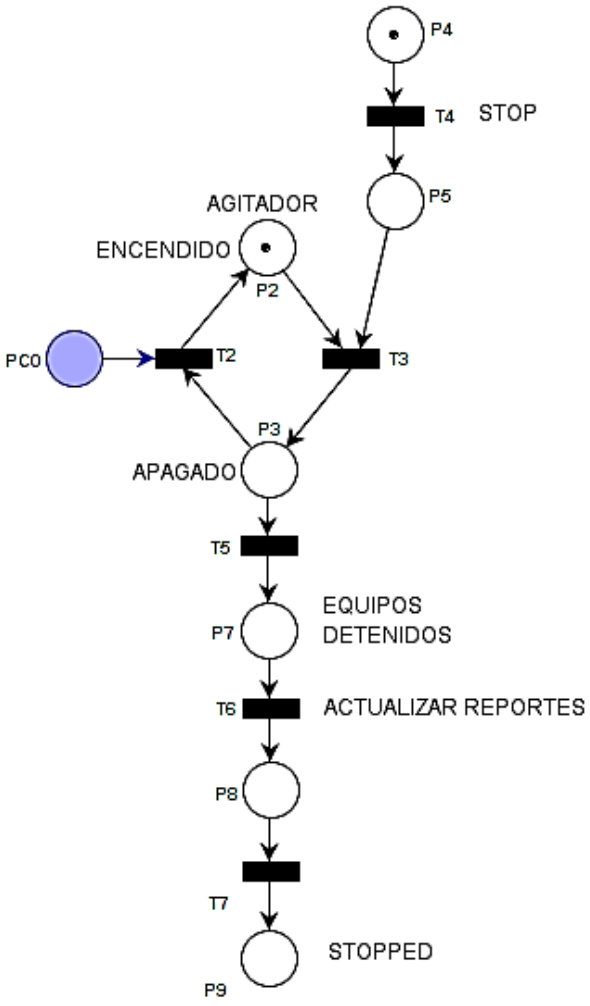


Figura A.15 Logica Stopping – *Agitar*

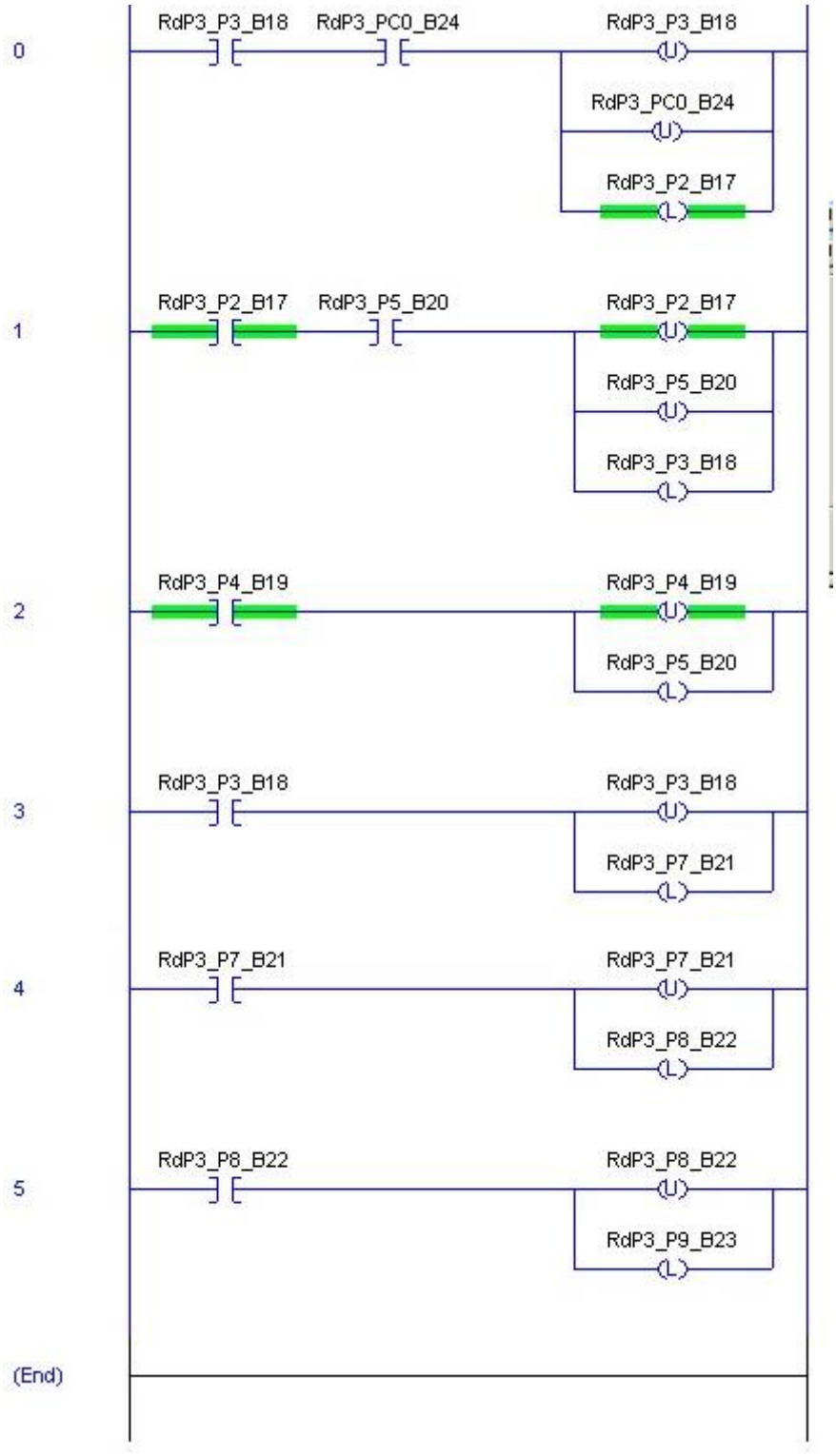


Figura A.16 Programa Ladder para Stopping – *Agitar*

A.3.4 Lógica Aborting

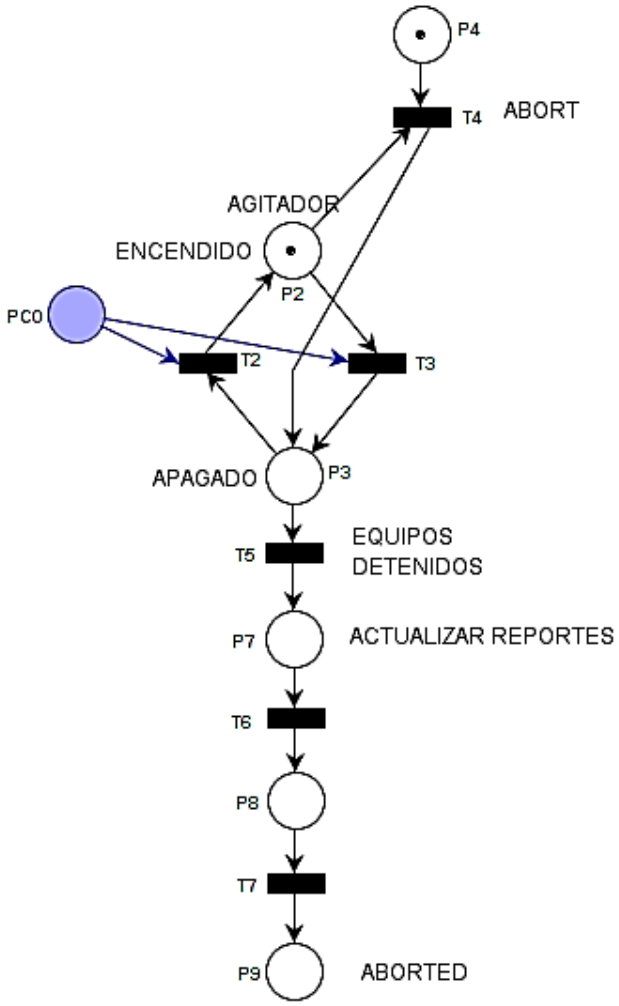


Figura A.17 Lógica Aborting – *Agitar*

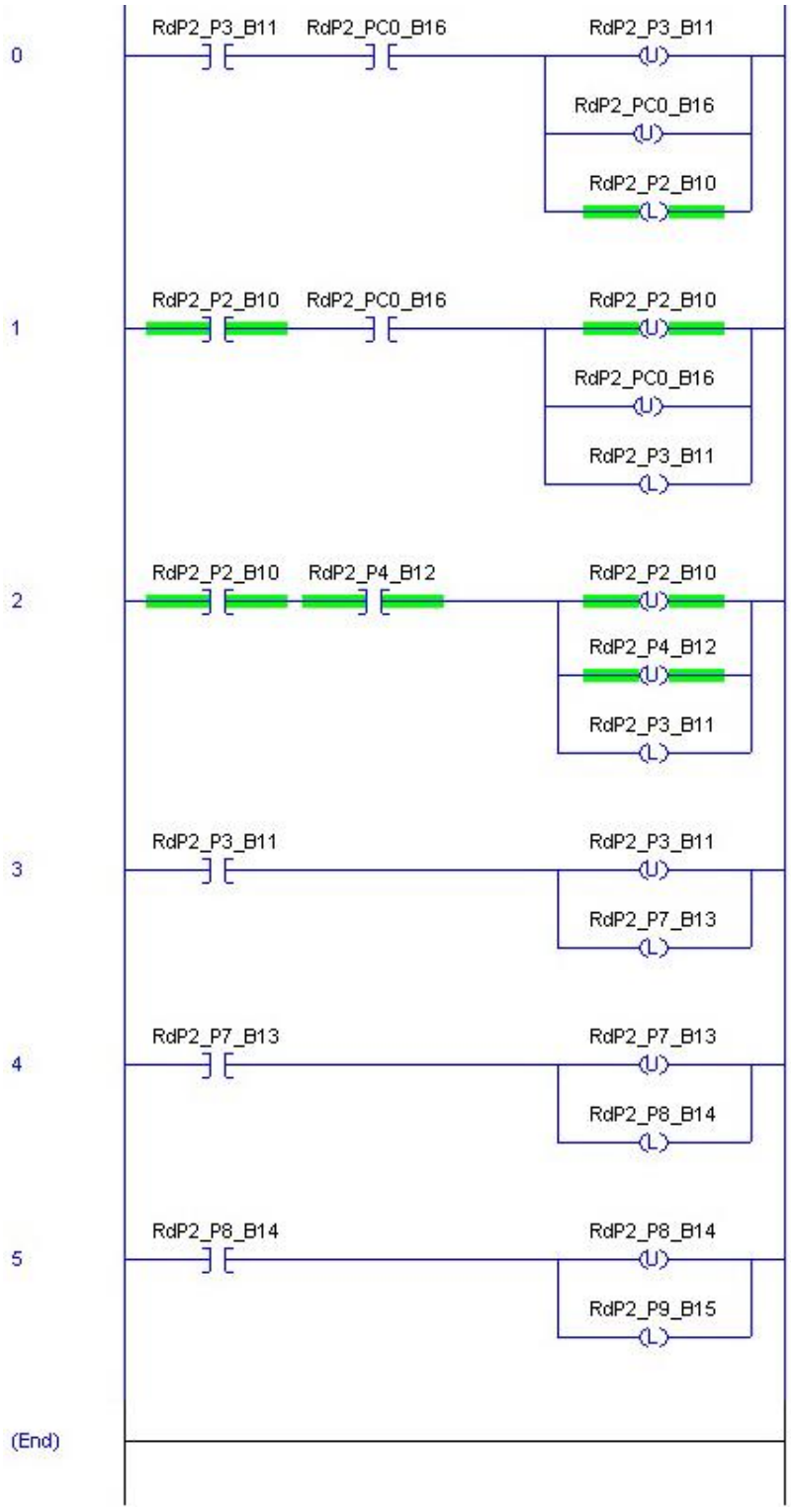


Figura A.18 Programa Ladder para Aborting – *Agitar*

ANEXO B

B.1 Paso de una red de Petri simple a un proyecto en L5K

En este anexo se encuentra detallado la imagen visual de la herramienta software y el paso de una red de Petri simple a un proyecto L5K de RsLogix5000.

Desde la ventana principal se debe de crear un nuevo proyecto en blanco e importar un proyecto nuevo en RsLogix5000 con una tarea de programación en blanco (ver Figura B.1). No es obligatoria la sincronización con el editor de equipos de FactoryTalk Batch.

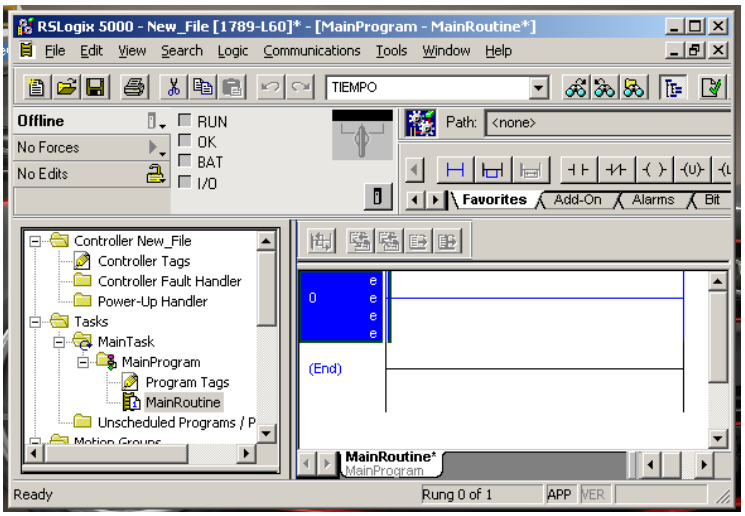


Figura B.19 Archivo en blanco generado desde RsLogix5000

En la Figura B.2 se prosigue con la importación de una red de Petri diseñada desde PIPE (también es posible realizar el mismo procedimiento con un archivo de extensión de tipo Snoopy).

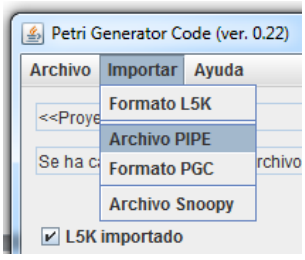


Figura B.2 Importar red de Petri realizada en PIPE

La red de Petri diseñada en la Figura B.3 representa el procedimiento de una máquina de estados. La información detallada de esta red de Petri se podrá consultar en la ventana de información de la Figura B.4.

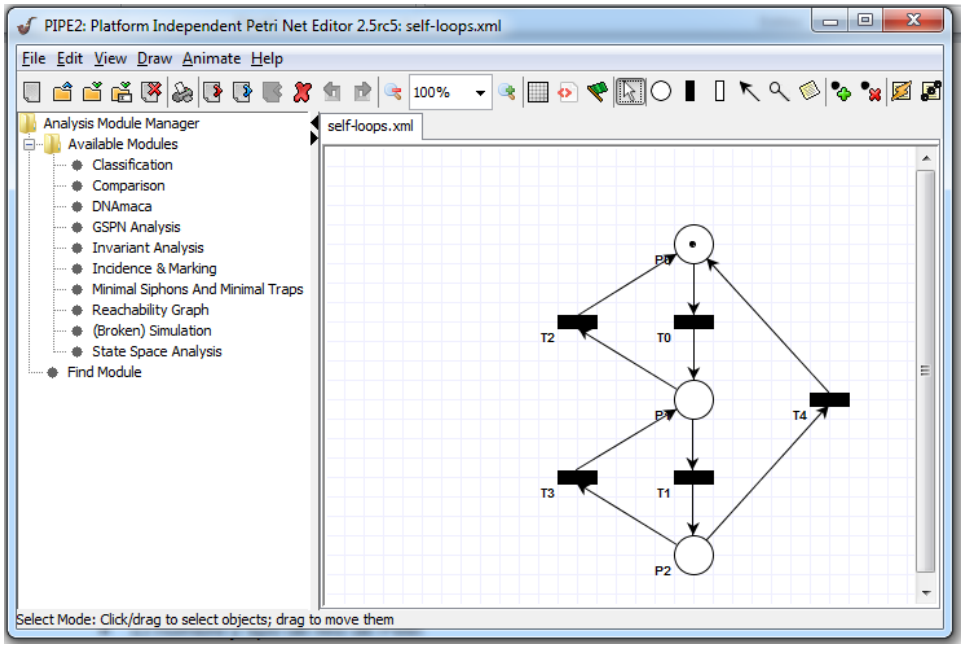


Figura B.3 Red de Petri realizada en PIPE

La ventana de información de la figura B.4 contiene la siguiente información:

- El nombre y tipo de red de Petri
- El ID con el cual se identifican los lugares y las transiciones de la red de Petri
- Las matrices de
    - incidencia combinada,
    - incidencia hacia adelante,
    - incidencia hacia atrás,
    - arcos de inhibición
    - arcos de lectura
- La marca inicial de los lugares
- La existencia de transiciones temporizadas
- El número de estados encontrados por el algoritmo de alcanzabilidad
- Clasificación de seguridad de la RdP

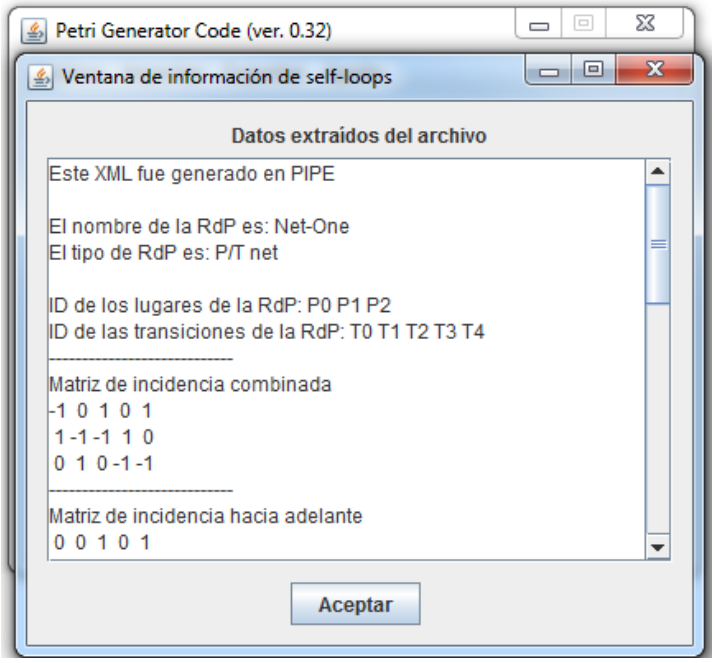


Figura B.4 Información de la red de Petri realizada en PIPE

Al realizar click sobre Generar Programa se accederá a una interfaz gráfica de usuario (GUI). La GUI contiene dos ventanas a la izquierda desde las cuales se podrán visualizar los lugares y transiciones de la red de Petri cargada en PGC (ver Figura B.5).

En la parte superior derecha se podrá asignar una o varias acciones a un lugar conectadas por una instrucción AND. La lectura de una acción sería la siguiente “Cuando exista una marca en P1, las salidas 1 y 2 del módulo de E/S se energizarán”.

De manera análoga, en la parte inferior derecha se realizará la asignación de una o varias condiciones lógica a un lugar conectadas por una instrucción AND o por una instrucción AND NOT. La lectura de una condición lógica sería la siguiente “Para disparar una transición T1, se requiere que la entrada 0 esté activada y la entrada 2 esté desactivada, ambas del módulo de E/S”.

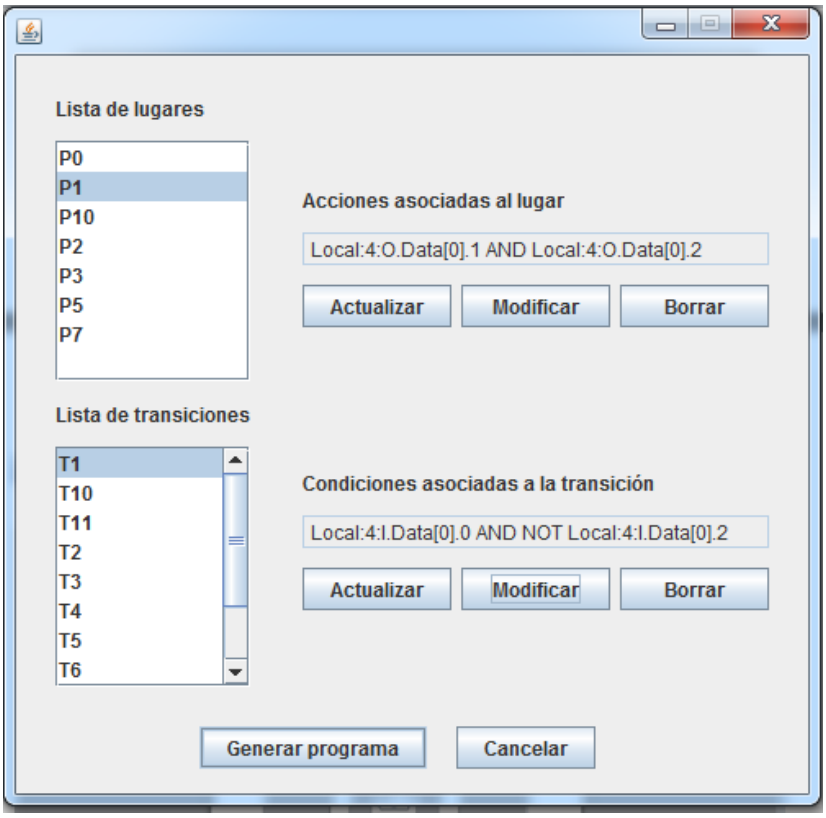


Figura B.5 Asignación de condiciones y acciones a la red de Petri

Luego de acceder a la opción de Generar programa. Aparecerá una ventana desde la cual se podrá seleccionar una opción: LLD o SFC (ver Figura B.6)

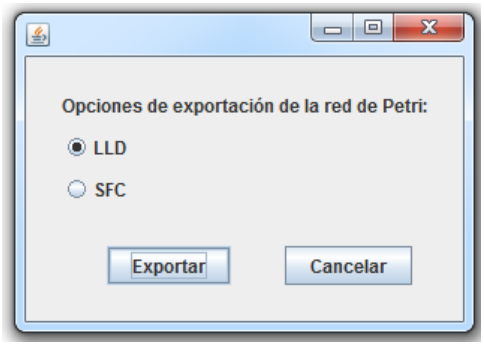


Figura B.6 Opciones de generación del código de programación en IEC 61131-3

Se podrán verificar los resultados de la exportación en la Figuras 3.31 del Capítulo 3 de la monografía.

ANEXO C

C.1 Implementación de redes de Petri con snoopy para la aplicación PGC

La herramienta de diseño Snoopy ha sido adaptada dentro como aplicación auxiliar para el diseño de redes de Petri jerárquicas con extensiones, o para la animación y simulación de eventos simultáneos en redes de Petri. Por tal razón, se ofrece como alternativa al diseño de redes de Petri en PIPE/CRP en caso de que así se requiera. Dentro de las opciones de esta plataforma, se deberá utilizar la opción “Extended Petri Net” para la creación de redes de Petri con las características ya mencionadas. Sin embargo, también es válido el uso de “Petri Net” para crear redes de Petri jerárquicas (sin extensiones).

A continuación se hace una descripción del uso de este tipo de redes de Petri en Snoopy:

- ***Desarrollo de redes de Petri con estructuras jerárquicas***

El diseño de redes de Petri por jerarquías propone no solo la construcción de redes más pequeñas, sino también de estructuras que son encapsuladas y reutilizadas en la construcción de otras redes de Petri. Esta característica en particular, simplifica en grandes proporciones la construcción y diseño de extensos modelos dinámicos por medio de la distribución de su comportamiento. Para esto, se utilizarán dos elementos gráficos adicionales a las redes de Petri ordinarias: macro nodos y nodos fusionados, los cuales están disponibles en la plataforma de Snoopy.

Macro nodos

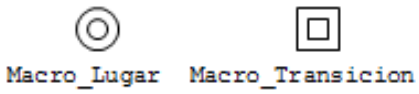


Figura C.1 Representación gráfica de un macro lugar y de una macro transición.

Los Macro nodos se dividen en Macro transiciones y Macro lugares, tal como están representados en la figura C.1. Cada Macro nodo tendrá asociado así mismo una subred, la cual podrá estar conformada por lugares, transiciones, Macro lugares y Macro transiciones conectados entre sí, por medio de arcos. Un macro lugar se conecta a transiciones, mientras que una macro transición se conecta a lugares. Por el contrario, no será posible conectar macro lugares con macro transiciones.

En la visualización de una subred, estarán diferenciados aquellos elementos que son propios de la subred con aquellos elementos externos a ella (pertenecientes a la red de Petri superior). De esta manera, se representan de un color distinto los lugares y transiciones externas con referencia a los elementos internos de la subred, tal como se puede ver en la figura C.2 para representar las conexiones entre estos elementos gráficos.

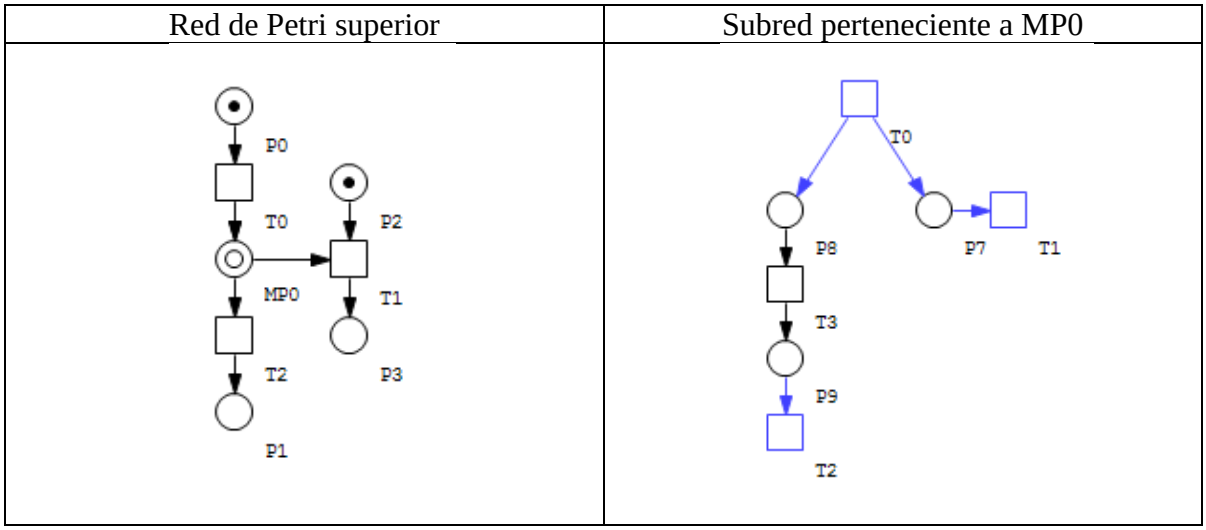


Figura C.2 Red de Petri diseñada con macro lugares.

Nodos fusionados

Para reutilizar los mismos elementos de una red de Petri, se pueden agrupar como nodos fusionados (en inglés llamados: fusion nodes o logical nodes) los cuales se representan de color gris en una red de Petri. Los nodos fusionados podrán ser lugares fusionados o transiciones fusionadas tal como están representados en la figura C.3. Varios nodos fusionados con el mismo nombre serán idénticos (pasan a ser copias de un único nodo), con lo cual podrán ocupar varios lugares en una red de Petri.

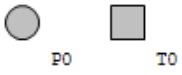


Figura C.3 Representación gráfica de un lugar fusionado y de una transición fusionada.

En la figura C.4 se encuentra dibujada una red de Petri con inicio en P0, con dos posibles secuencias de ejecución de dos procesos distintos: MT1 y MT2 (MT: Macro Transición.) La red de Petri culmina en P1.

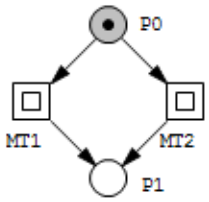


Figura C.4 Red de Petri con ejecución de dos procesos distintos MT1 y MT2.

Cada macro Transición tiene asociada una subred como se puede visualizar en la figura C.5. La subred de la izquierda perteneciente a MT1 indica una secuencia del proceso que podría disparar la transición T4 en caso de existir un fallo en la ejecución de MT1, lo que posiciona una marca en el lugar "Fallo". De igual forma, la ejecución de la subred de la derecha perteneciente a MT2 indica una secuencia del proceso que disparar la transición T6 en caso de existir un fallo en la ejecución de MT2. Análogamente, esto posiciona una marca en el lugar "Fallo".

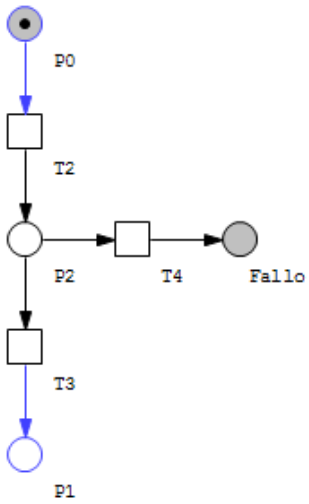
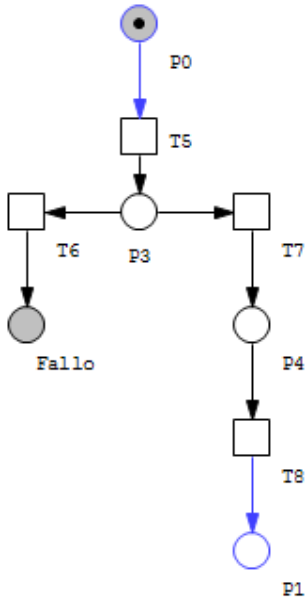
Subred asociada a MT1	Subred asociada a MT2
	

Figura C.5 Subredes asociadas a los procesos MT1 y MT2.

En ambos casos, la secuencia activación de Fallo es la misma, ya que podría indicar la ejecución de una rutina genérica en estos casos tal como está indicado en la figura C.6, como por ejemplo activar una luz roja de alarma o detener actuadores. En este caso, el

proceso quedará detenido hasta que sea enviada una señal de reanudación del proceso por medio del disparo de T1.

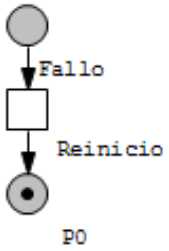


Figura C.6 Secuencia de detección y reparación de fallos de los procesos MT1 y MT2.

Si desde la aplicación PGC se abre el XML asociado a esta red de Petri, se podrá obtener la información indicada en la figura C.7.

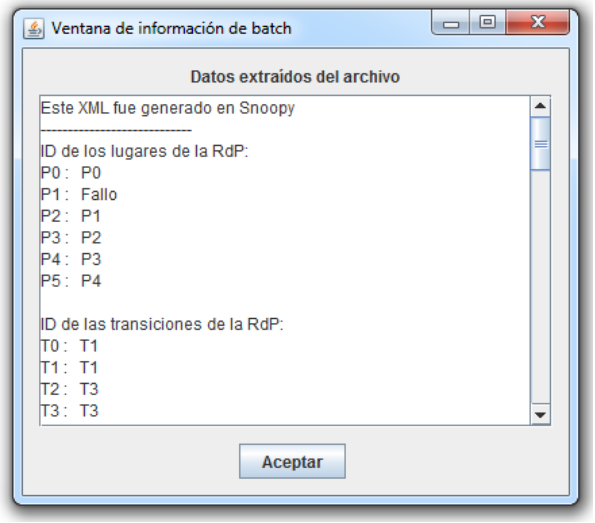


Figura C.7 Ventana de información de la secuencia descrita en Snoopy.

Posteriormente a su transformación por medio de PGC, el código Ladder generado por el modelo dinámico está indicado en la figura C.8.

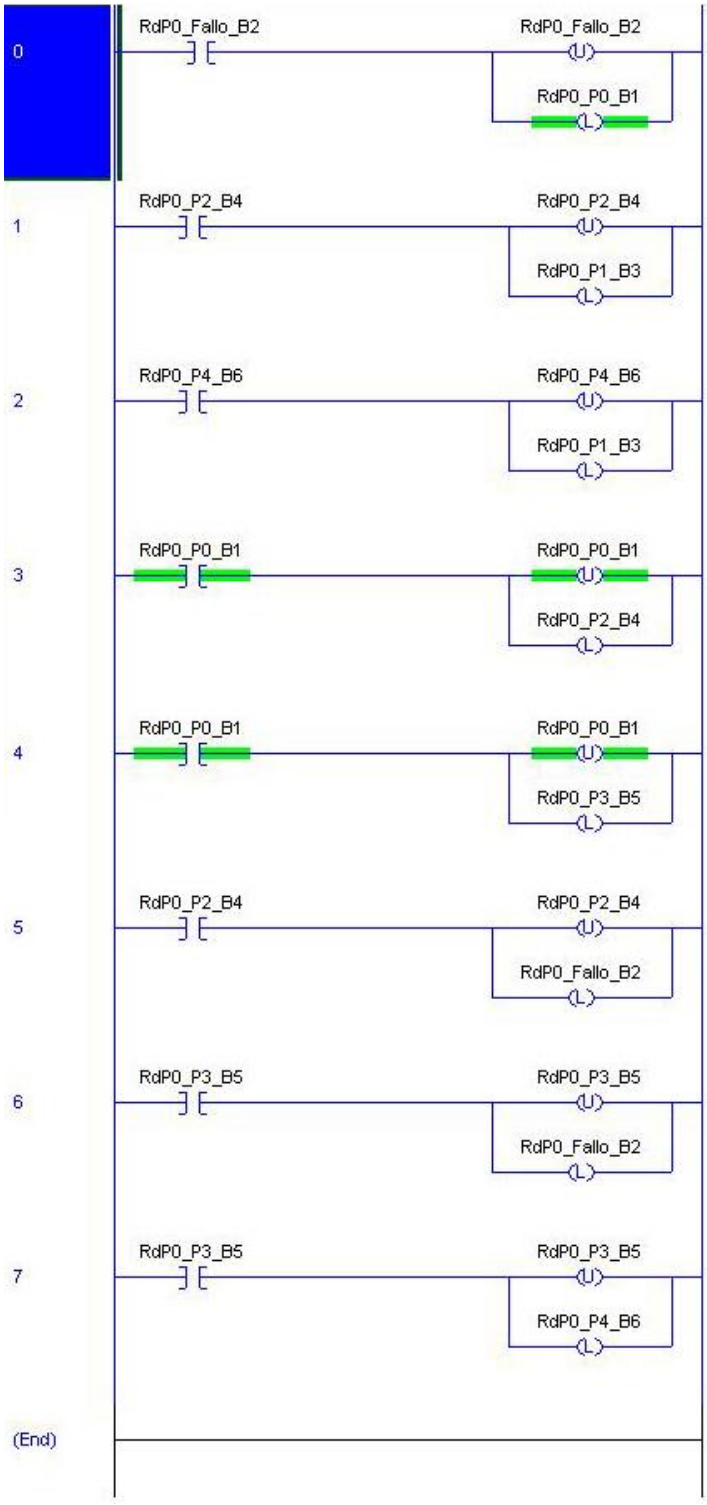


Figura C.8 Código Ladder de la secuencia descrita en Snoopy.

Estos nuevos elementos gráficos son detectados por el usuario mas no por el algoritmo de programación, y por tanto, la generación de código Ladder no se ve afectada por la inclusión de estos nuevos elementos gráficos. Esto ocurre ya que el algoritmo de lectura creado para Snoopy permite obtener las conexiones que existan entre transiciones y lugares, y no toma en cuenta la existencia de las subredes que existan a nivel gráfico, por lo cual estos, elementos adicionales son invisibles a ella.

Como el lector podrá percibir, estos elementos gráficos contribuyen al diseño de componentes reutilizables en sistemas complejos (por ejemplo, con el uso de subredes que posean exactamente las mismas entradas y salidas definidas), con lo cual, estos gráficos impactan de manera positiva al usuario diseñador de las redes de Petri.

Finalmente, el procedimiento para realizar el paso a SFC no está contemplado ya que excede la capacidad del algoritmo en tanto que las posiciones x,y dibujadas en un plano pueden llegar a repetirse llegando a generar conflictos en la generación del código.