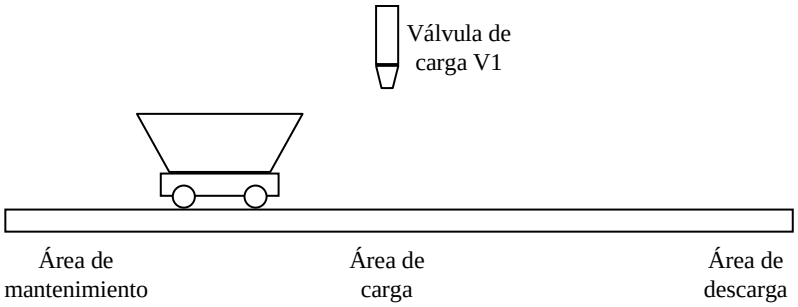
Problema: Vagoneta

Se pretende automatizar el sistema de transporte de material de la figura:



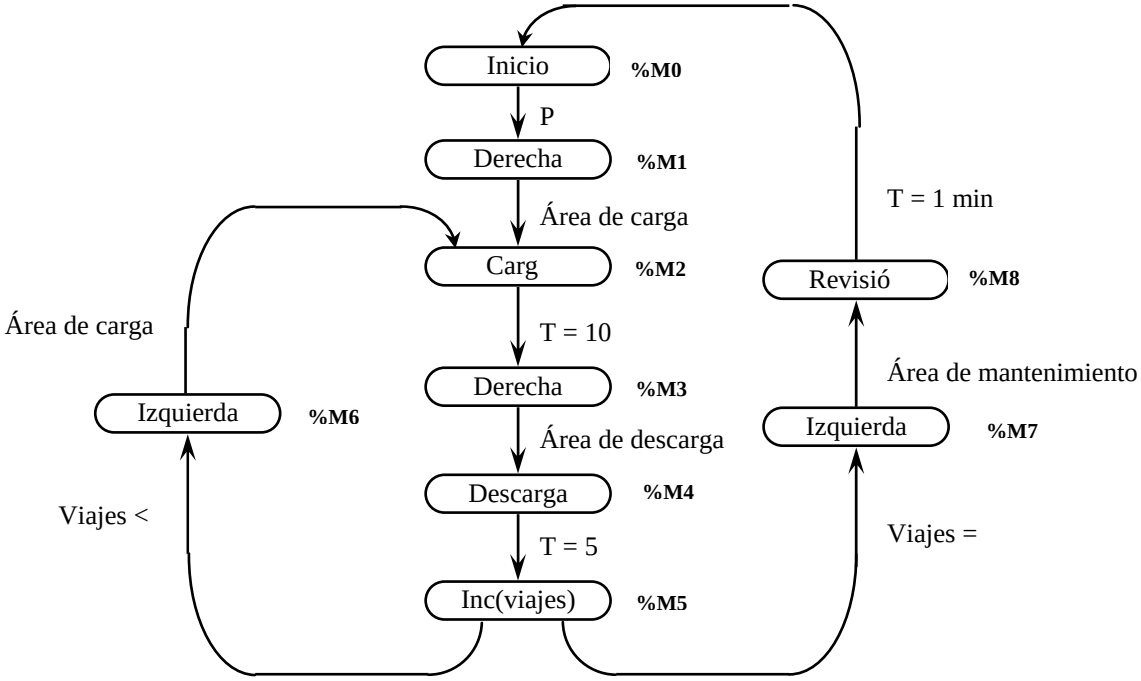
El funcionamiento del sistema es el siguiente:

- En el estado inicial la vagoneta se encuentra en el área de mantenimiento.
- El sistema se activa mediante un pulsador **P**.
- Se pone en marcha la vagoneta hacia la derecha (**Mov.Dcha.**) hasta llegar a la zona de carga (se detecta mediante un sensor) y se detiene.
- Se abre la válvula de carga **V1**, durante 10 segundos, tiempo empleado en llenar la vagoneta.
- Una vez llena se desplaza hacia la zona de descarga donde vacía su contenido en 5 segundos.
- Vuelve a la zona de carga y repite el proceso 5 veces.
- Concluida la quinta descarga, vuelve a la zona de mantenimiento (**Mov.Izqda.**) para una inspección de la vagoneta; la revisión dura 1 minuto.
- Terminada la revisión se puede repetir el ciclo actuando sobre el pulsador.
- Durante el proceso permanecerá encendido un piloto indicando el estado activo.

Se pide: Programar en contactos el controlador del automatismo, partiendo del diagrama de estados.

Solución al Problema: Vagoneta

El diagrama de estados seguido en la solución, resuelto en los ejercicios anteriores anterior, es el siguiente:



El entorno de programación que utilizaremos es el PL7 Junior. Antes de programar en lenguaje de contactos, debemos asociar a posiciones de memoria del autómata las entradas, las salidas y los estados.

Para asignar variables de memoria, tendremos en cuenta como realiza el direccionamiento de memoria el autómata. Para los estados emplearemos variables de memoria booleanas, designadas por %M0, %M1, etc. Para las entradas y salidas, hay que tener en cuenta la configuración de entradas y salidas del autómata. En los autómatas de prácticas las entradas “todo o nada” están en el primer módulo, luego las direccionaremos con %I1.0, %I1.1, etc. Las salidas están situadas en el segundo módulo, luego las direccionaremos como %Q2.0, %Q2.1, etc.

Siguiendo este criterio realizamos las siguientes asignaciones, teniendo en cuenta las entradas y salidas físicas donde se han cableado las entradas y salidas en el autómata, por ejemplo suponemos:

Entradas al autómata:

%I1.0: Pulsador de puesta en marcha
%I1.10: Sensor área de carga
%I1.13: Sensor área de descarga
%I1.8: Sensor área de mantenimiento

Salidas del autómata:

%Q2.0: Motor hacia la derecha
%Q2.1: Motor hacia la izquierda

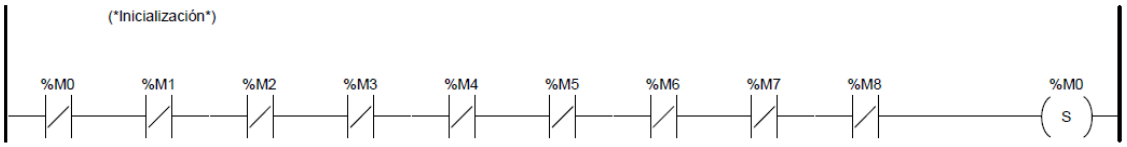
%Q2.8: Indicador de actividad
%Q2.2: Led de revisión
%Q2.7: Led de descarga
%Q2.4: Led de carga.

Estados:

%M0: Estado inicial
%M1: Movimiento derecha hacia la zona de carga
%M2: Cargando
%M3: Movimiento derecha hacia zona de descarga
%M4: Descargando
%M5: Reposo. Contar viajes
%M6: Movimiento izquierda hacia zona de carga
%M7: Movimiento izquierda hacia zona de mantenimiento
%M8: Revisión

A continuación se procede a realizar el programa en lenguaje de contactos. En el programa se distinguen tres partes:

1.- Inicialización: El programa debe inicializarse en el estado %M0. Para ello se aprovecha que cuando se activa el programa, todas las variables valen cero. Por lo tanto, si no está en ningún estado, pongo a 1 con un set la variable %M0.



2.- Programar la secuencia de estados: A continuación, debemos de programar la secuencia que sigue el diagrama de estados. De forma genérica, si estoy en un estado y se cumple la condición de cambio, reseteo el estado anterior y pongo a uno el estado siguiente correspondiente a esa transición. Entre estados utilizamos bobinas de set y reset ya que entro en un estado con un set, y permanezco en el hasta que se dé la correspondiente condición de cambio. Así, si estoy en el estado %M0 (inicial) y se actica el pulsador de puesta en marcha %I1.0, salgo del estado %M0 con un reset y activo el estado %M1 (Movimiento a la derecha).

Se procede de igual forma con el resto de estados:

En el siguiente segmento de programa, para hacer el cambio de estado se utiliza un temporizador. Ahora, si estoy en el estado %M2 y el temporizador me indica que he estado 10

segundos en dicho estado (10 segundos cargando), salgo del estado %M1 con un reset y activo el estado %M2.

%TMO: Modo de trabajo: TON
Tiempo Base (TB)= 10 ms (Cuanto menor sea, mayor precisión al contar).
Valor de preselección: %TM0.P= 1000 (Cuenta 1000 veces la base de tiempos: 1000*10ms=10000ms=10s).

Procedemos de igual manera para ir del estado %M3 al %M4:

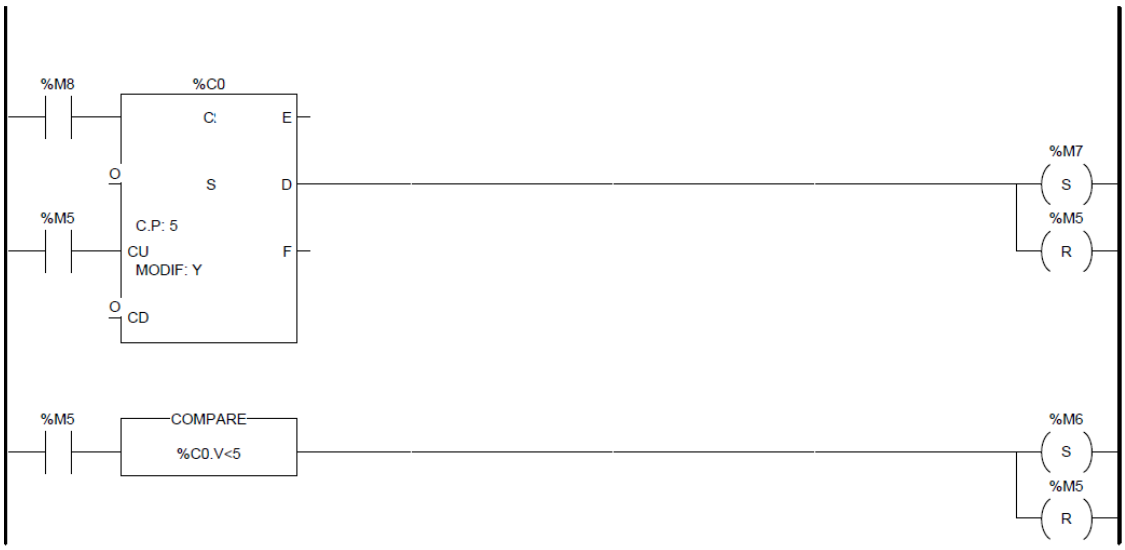
En el estado %M4, en el que estamos descargando, debemos permanecer 5 segundos, con lo que emplearemos otro temporizador:

%TM1: Modo de trabajo: TON
Tiempo Base (TB)= 10 ms (Cuanto menor sea, mayor precisión al contar).
Valor de preselección: %TM0.P= 500 (Cuenta 500 veces la base de tiempos: 500*10ms=5000ms=5s).

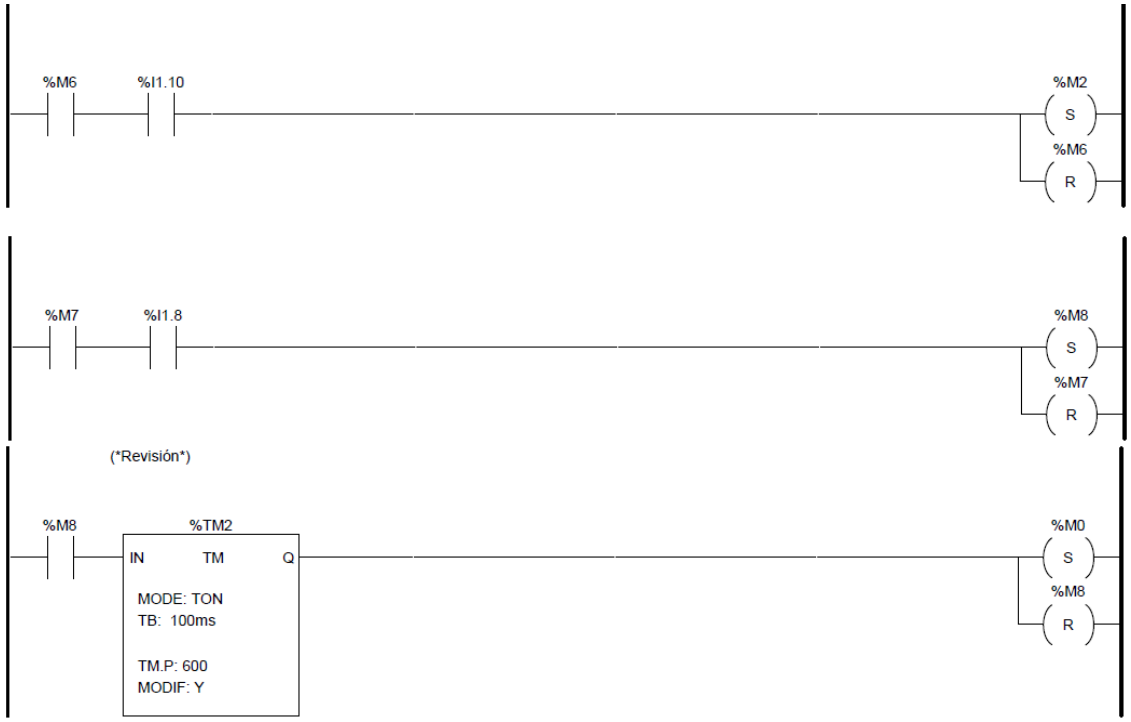
En el estado %M5 incrementamos un contador, que también debe ser configurado previamente:

%C0: Valor de preselección: %C0.P=5

Una vez estamos en el estado %M5, tenemos dos posibles alternativas de evolución. Si el contador vale 5, la patilla %C0.D vale 1, con lo que si estamos en %M5 y el contador vale 5, reseteo el estado %M5 y activo el estado %M7. Si por el contrario la cuenta vale menos de 5, indicado por el valor de la cuenta almacenado en la palabra %C0.V y para lo que utilizo un comparador, reseteo el estado %M5 y activo el estado %M6



De esta forma completamos los segmentos de programa que faltan para recorrer todo el diagrama de estados y contemplar todos los posibles cambios de estado:



3.- Activación de salidas: Una vez programados todos los cambios de estado, deben programarse la activación de las salidas. La forma que utilizaremos es la siguiente:

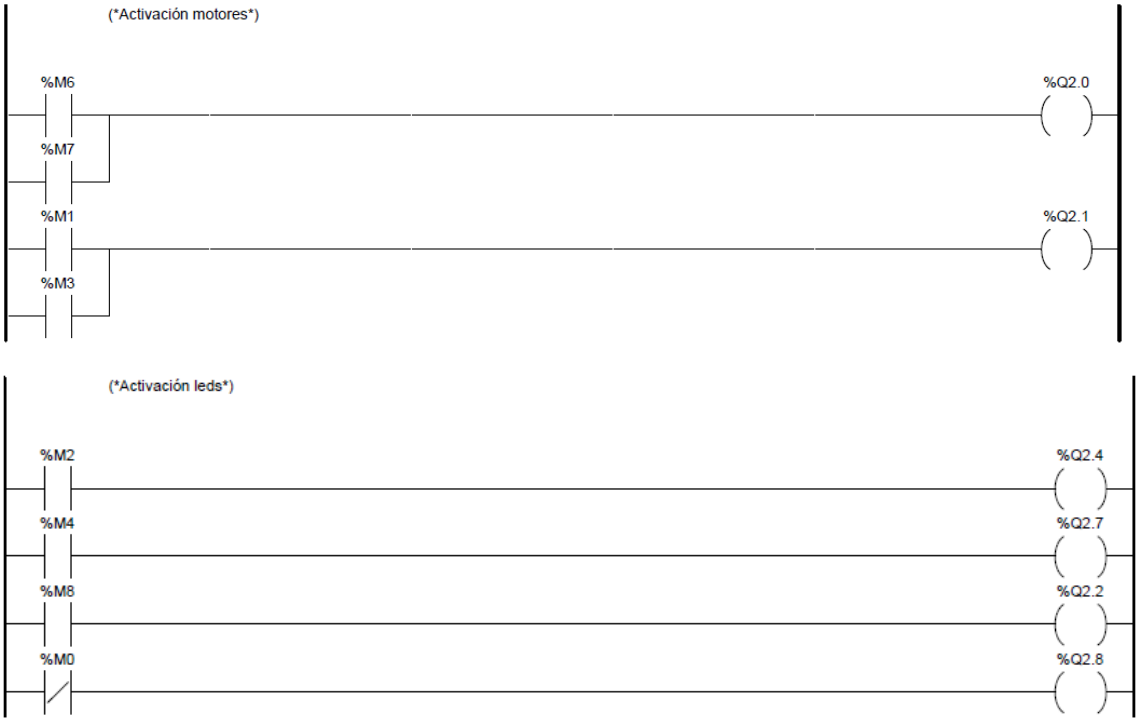
- Situamos todas las salidas a activar y las colocamos en la zona de acciones (derecha).
- A continuación miramos en que estados deben estar activas y los ponemos en la zona de condiciones (izquierda) mediante una operación lógica “or”.

Notas:

- Las salidas NO DEBEN aparecer más de una vez, pues podrían recibir órdenes contradictorias.

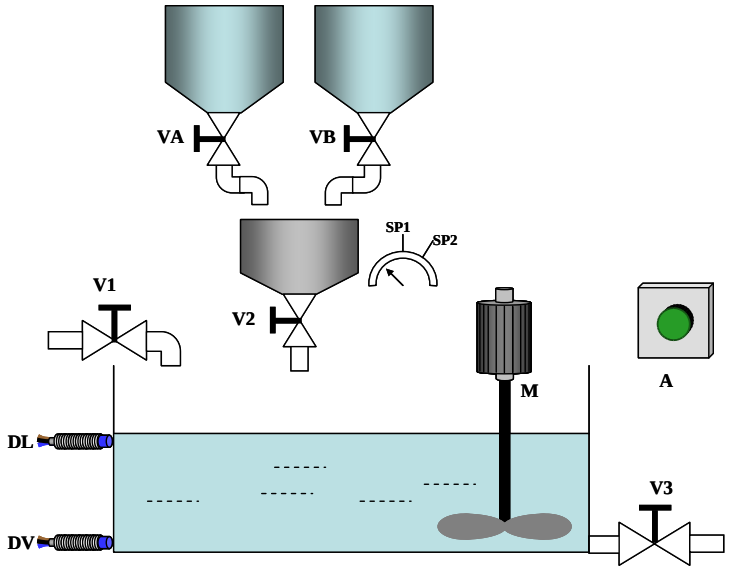
- Los estados se ponen en paralelo, para que se active en cualquiera de los estados que deba activarse.

3.- Activación de salidas:



Ejercicio: MEZCLADORA

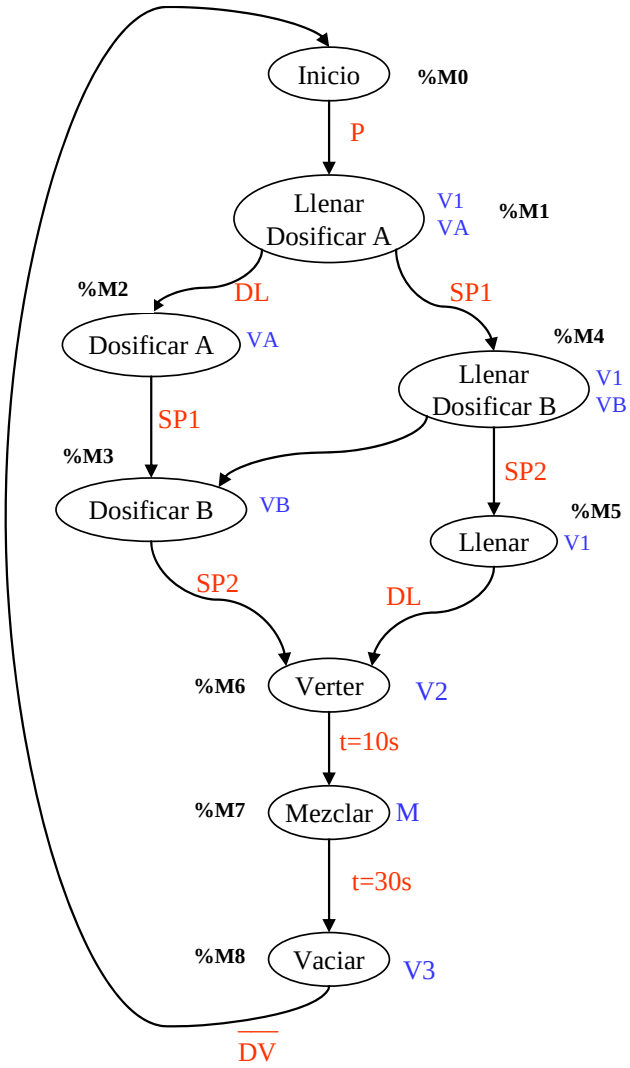
Se quieren mezclar 2 productos con agua. Se llena el depósito de agua abriendo la válvula V1. La dosificación de los dos productos se realiza con una tolva acumulativa, se vierte el producto A sobre la tolva hasta que se alcanza un peso SP1 y a continuación se añade el producto B para conseguir el peso total de los 2 productos, SP2. Se abre la válvula de la tolva durante 10 segundos para dejar caer el contenido. Se realiza el proceso de mezclado durante 30 segundos accionando el agitador y se vacía el depósito para poder iniciar un nuevo ciclo. El proceso se activa con un interruptor P.



Se pide: Programar en contactos el controlador del automatismo, partiendo del diagrama de estados.

Solución 1: Diagrama de estados: Salidas=f(Entradas)

El diagrama de estados seguido en la solución, resuelto en los ejercicios anteriores, es el siguiente:



Como en el ejercicio anterior, el entorno de programación que utilizaremos es el PL7 Junior. Antes de programar en contactos, debemos asociar a posiciones de memoria del autómata las entradas, las salidas y los estados.

En este caso consideramos:

Entradas al autómata:

P: %I1.0
DL: %I1.1
DV: %I1.2
SP1: %I1.3
SP2: %I1.4

Salidas del autómata:

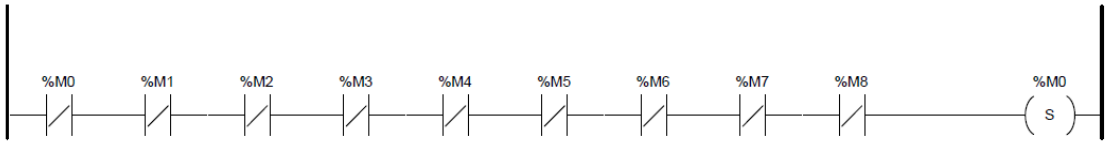
V1: %Q2.0
V2 : %Q2.1
V3 : %Q2.2
VA : %Q2.3
VB: %Q2.4
M: %Q2.5

Estados:

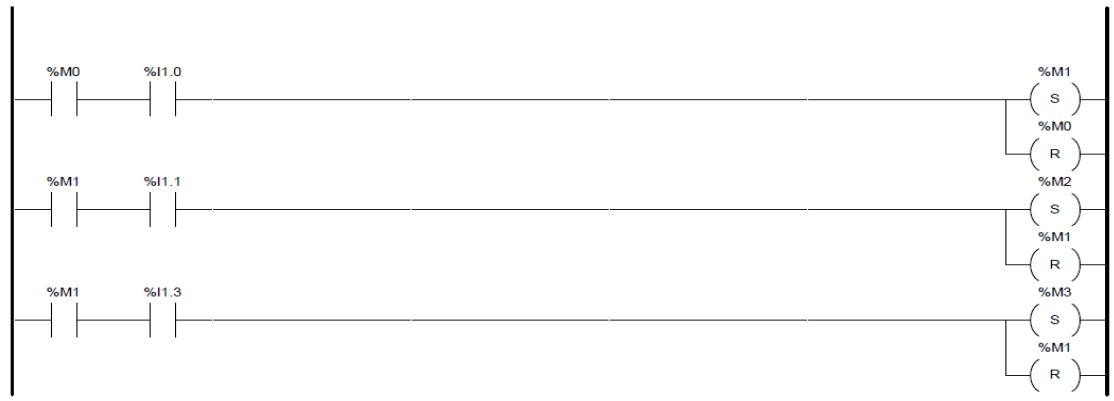
%M0: Estado inicial
%M1: Llenar Dosificar A
%M2: Dosificar A
%M3: Dosificar B
%M4: Llenar y dosificar B
%M5: Llenar
%M6: Verter
%M7: Mezclar
%M8: Vaciar

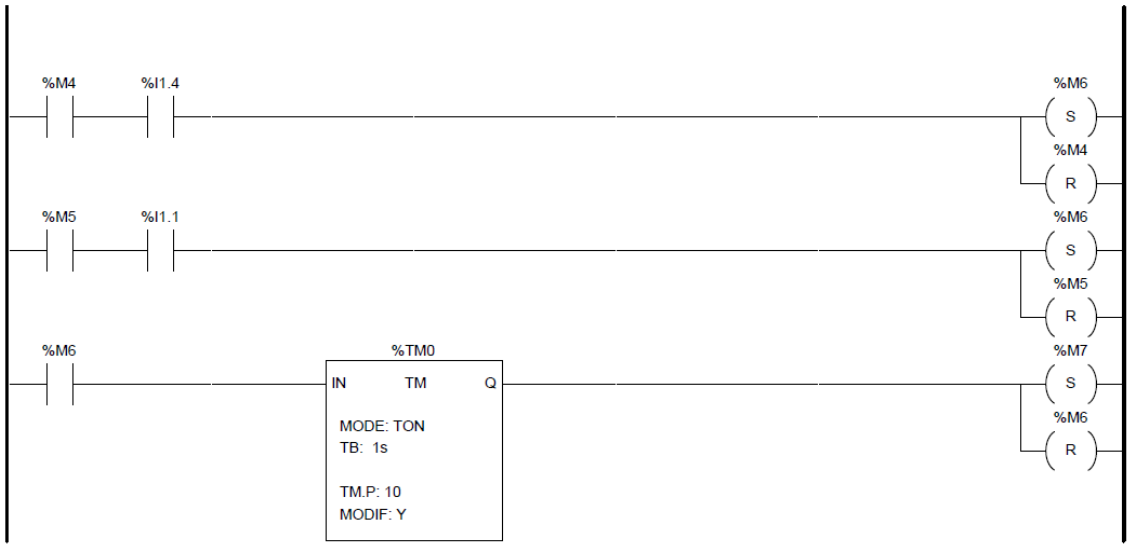
El programa de contactos en este caso queda como:

1.- Inicialización:

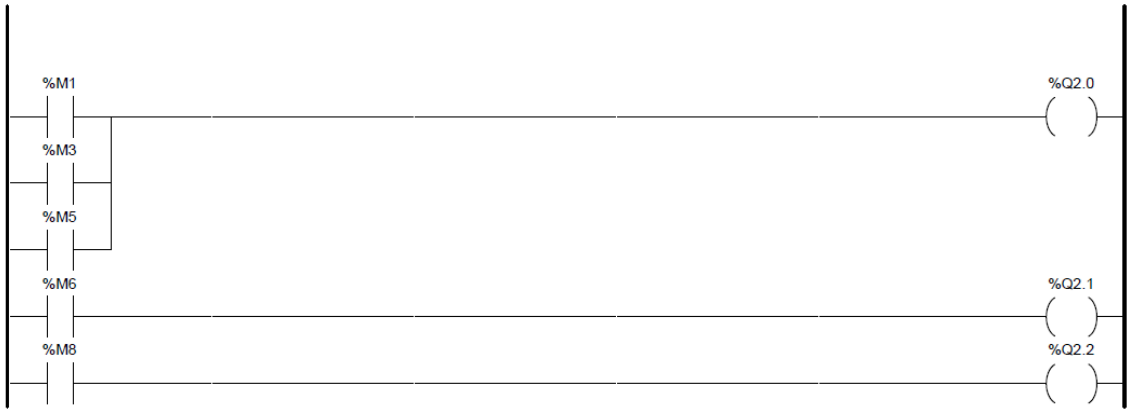


2.- Programar la secuencia de estados:



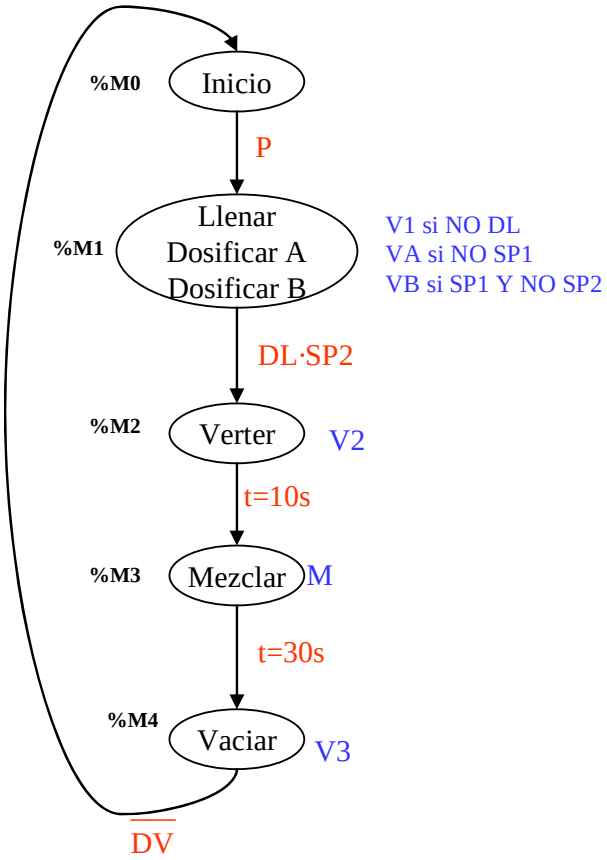


3.- Activación de salidas:



Solución 2: Diagrama de estados: Salidas=f(Entradas, Memoria)

El diagrama de estados seguido en la solución, resuelto en los ejercicios anteriores, es el siguiente:



Como ya se comentó en la explicación de este diagrama de estados en los ejercicios anteriores, el diagrama de estados se simplifica, pero en la asignación de las salidas se deberá tener en cuenta las diversas posibilidades que hay dentro de cada estado.

Como en el ejercicio anterior, el entorno de programación que utilizaremos es el PL7 Junior. Antes de programar en contactos, debemos asociar a posiciones de memoria del autómata las entradas, las salidas y los estados. En este caso consideramos:

Entradas al autómata:

P: %I1.0
DL: %I1.1
DV: %I1.2
SP1: %I1.3
SP2: %I1.4

Salidas del autómata:

V1: %Q2.0
V2 : %Q2.1
V3 : %Q2.2
VA : %Q2.3
VB: %Q2.4
M: %Q2.5

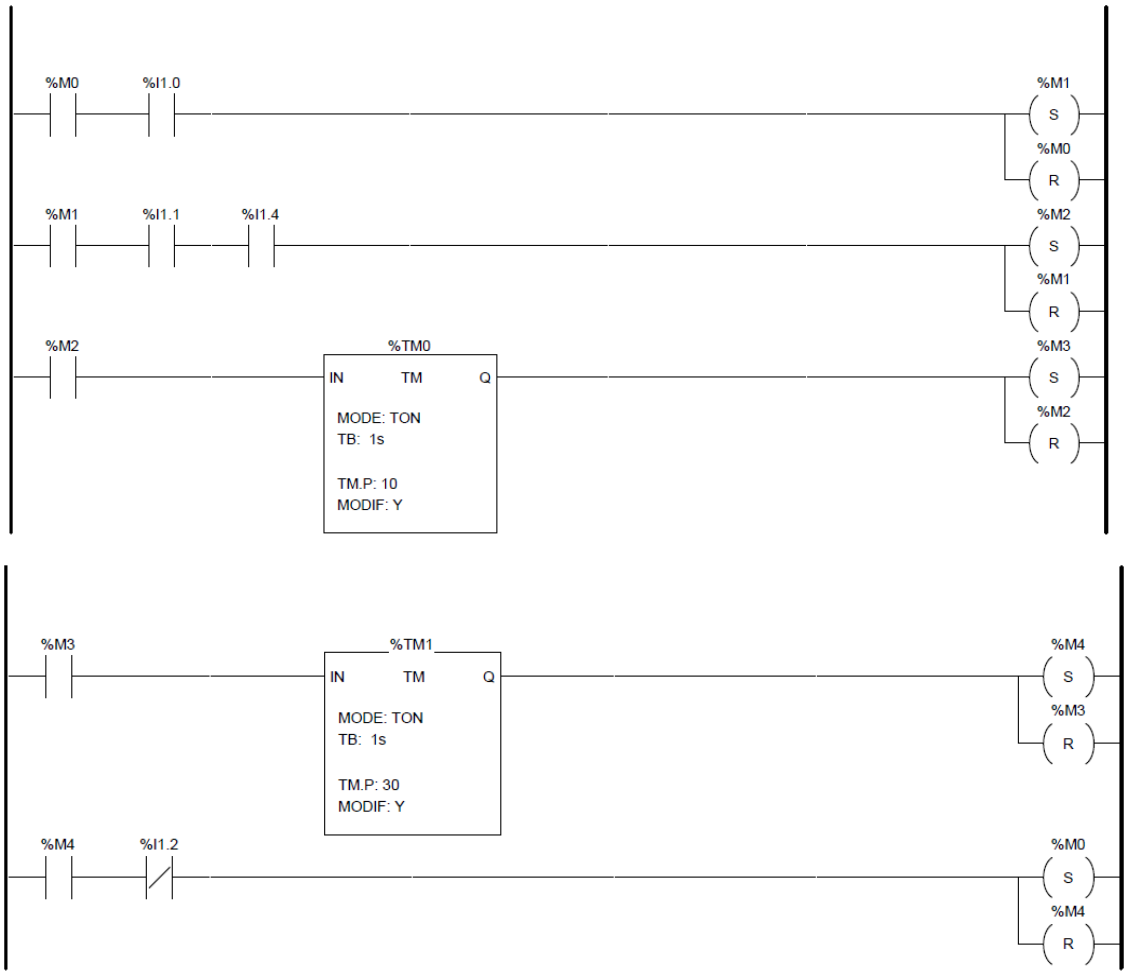
Estados:

%M0: Estado inicial
%M1: Llenar Dosificar A Dosificar B
%M2: Verter
%M3: Mezclar
%M4: Vaciar

El programa de contactos en este caso queda como:

1.- Inicialización:

2.- Programar la secuencia de estados:



3.- Activación de salidas:

En este caso vemos que las salidas, además de depender del estado, dependen de variables de entrada.