

Practica 2

Controladores lógicos programables

Diego Armando Becerra Iñiguez

5-A

18/02/19

**Introducción**

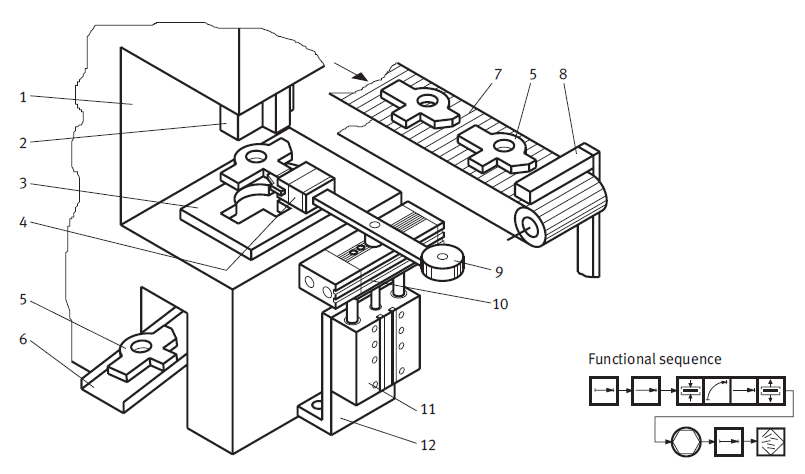
En esta práctica mostraremos la alimentación a una prensa de desbarbado. A continuación, mostraremos el enunciado seguido con la imagen del ejercicio.

El dispositivo de manipulación recoge las fundiciones orientadas desde el transportador utilizando una pinza de doble mandíbula y las posiciones uno a la vez sobre la placa de apertura y debajo del émbolo. Después del desbarbado,

las piezas de trabajo son alimentadas por la fuerza gravitacional en un recipiente colector. El brazo giratorio está equipado con un contrapeso para evitar cargas excéntricas que llevaría a un exceso de desgaste de la guía.

Las posiciones finales están equipadas con cilindros de amortiguación hidráulicos. Esta secuencia de movimiento podría, por supuesto, también puede lograrse mediante el uso de otras configuraciones de accionamientos neumáticos, dispositivos de manipulación con coordenadas cartesianas utilizando ejes lineales.

**Desarrollo**



0.5

0.4

0.3

0.3

0.2

0.2

0.1

0.1

0.0

0.0

**Ahora se pondrá cada parte del mecanismo**

1 prensa

2 émbolo de desbarbado

3 placa de apertura

4 pinza de sujeción

5 objeto Deburr Ed

(casting)

6 Conducto de salida

7 cinta transportadora (conveyor)

8 Tope

9 contrapeso

10 unidad giratoria

11 corredera elevadora

12 soporte de montaje

**Seguido de sus sensores y actuadores.**

|  |  |
| --- | --- |
| **SENSORES** | **ACTUADORES** |
| I00=PIEZA DETECTADA | 00=MOVER BANDA |
| I01=BANDA DETENIDA | 01=DETENER BANDA |
| I02=BRAZO POSICIONADO 0 GRADOS | 02=GIRAR 0 GRADOS |
| I03=PINZA ABIERTA | 03=ABRIR PINZA |
| I04=PINZA CERRADA | 04=BAJAR CORREDERA |
| I05=PIEZA TOMADA | 05=CERRAR PINZA |
| I06=CORREDERA ARRIBA | 06=SUBIR CORREDERA |
| I07=GIRADO A 90 GRADOS | 10=MOVER 90 GRADOS |
| I10=PIEZA EN PLACA | 11=BAJAR EMBOLO |
| I11=EMBOLO FINAL DE CARRERA | 12=SUBIR EMBOLO |
| I12=EMBOLO ARRIBA |  |
| l13=CORREDERA ABAJO |  |
| L14=PIEZA EN BANDA |  |

**Entendamos que es el lenguaje ladder**

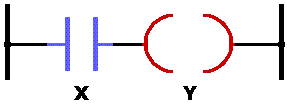
El diagrama de escalera fue uno de los primeros lenguajes utilizados para programar PLCs debido a su similitud con los diagramas de relés que los técnicos ya conocían.

Este lenguaje permite representar gráficamente el circuito de control de un proceso, con ayuda de símbolos de contactos normalmente cerrados (N.C.) y normalmente abiertos (N.A.), [relés](https://unicrom.com/rele-relay-relevador-interruptor-operado-magneticamente/), temporizadores, contadores, registros de desplazamiento, etc.. Cada uno de estos símbolos representa una variable lógica cuyo estado puede ser verdadero o falso.

En el **diagrama de escalera**, la fuente de energía se representa por dos “rieles” verticales, y las conexiones horizontales que unen a los dos rieles, representan los circuitos de control. El riel o barra del lado izquierdo representa a un conductor con voltaje positivo y el riel o barra de lado derecho representa tierra o masa.

El programa se ejecuta de arriba hacia abajo y de izquierda a derecha. Observar el diagrama anterior, donde se muestra el circuito para el accionamiento de un motor. Este motor se activa cuando el interruptor SW se cierra y permite el paso de [corriente](https://unicrom.com/corriente-electrica/) del riel del lado izquierdo al riel del lado derecho a través de él.

Acordarse que el riel izquierdo es el conductor con [voltaje](https://unicrom.com/tension-electrica/) y el riel o barra derecha está a tierra. En el siguiente diagrama se gráfica la representación del anterior diagrama en lenguaje de escalera.



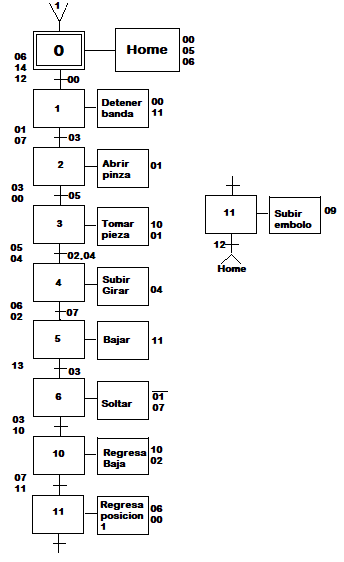
* “X” representa el interruptor normalmente abierto y se representa con esa letra por que es una entrada. “Y” representa al motor que se desea activar y se representa con esa letra por que es una salida.
* Cuando se activa X, se completa el circuito entre el riel izquierdo y el riel derecho a través del motor (Y), que se pone en funcionamiento.
* Cuando se desactiva X, se abre el circuito entre el riel izquierdo y el riel derecho y el [motor](https://unicrom.com/motor-de-corriente-alterna-o-motor-ac/) deja de funcionar.

**¿Qué es el GRAFCET?**

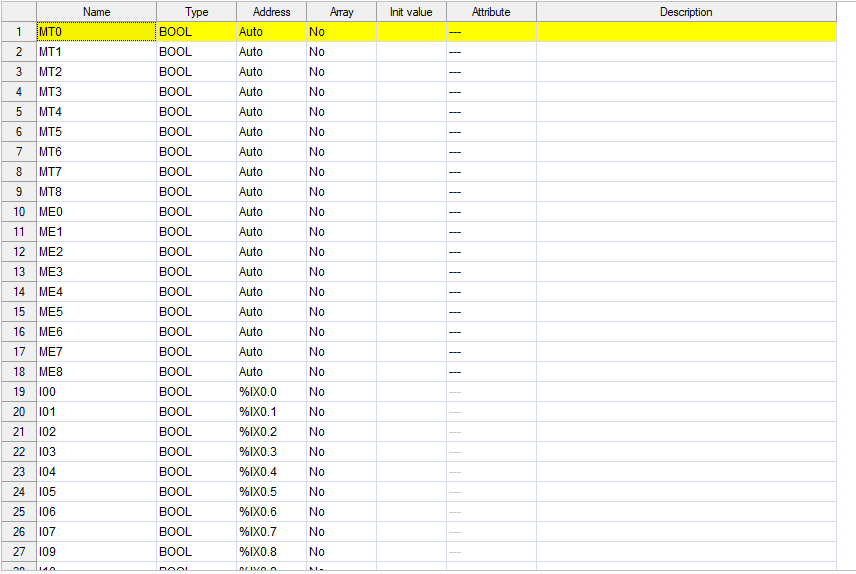
significa **diagrama de control con etapas y transiciones**, nació en [1977](https://es.wikipedia.org/wiki/1977" \o "1977), fuente del trabajo de la Asociación Francesa para la Cibernética Económica y Técnica (AFCET), en principio como síntesis teórica de las diferentes herramientas existentes por aquel entonces (organigrama, organifase, diagramas de Girard, red de Petri, etc...). Le otorgó su actual forma en [1979](https://es.wikipedia.org/wiki/1979" \o "1979) la Agencia Nacional para el Desarrollo de la Producción Automatizada (ADESA) francesa. Normalizada en Europa como (EN61131) e internacionalmente en [1992](https://es.wikipedia.org/wiki/1992" \o "1992) como (norma CEI 1131).

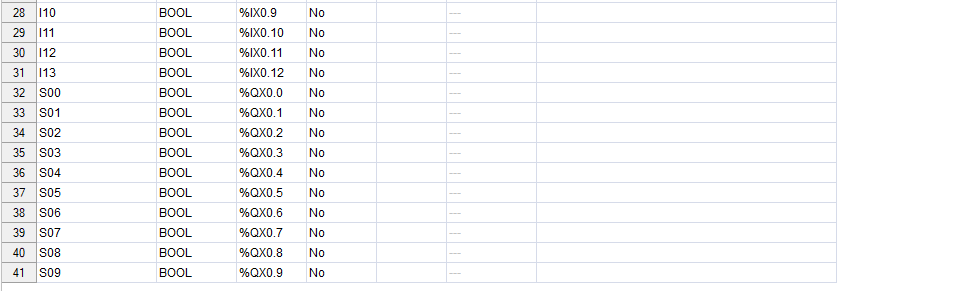
el GRAFCET es un modelo de representación gráfica, de los sucesivos comportamientos de un sistema lógico, predefinido por sus entradas y salidas. También es un [grafo](https://es.wikipedia.org/wiki/Grafo" \o "Grafo), o diagrama funcional normalizado, que permite hacer un modelo del proceso a automatizar, contemplando entradas, acciones a realizar, y los procesos intermedios que provocan estas acciones. Inicialmente fue propuesto para documentar la etapa secuencial de los sistemas de control de procesos a eventos discretos. No fue concebido como un lenguaje de programación de autómatas, sino un tipo de grafo para elaborar el modelo pensando en la ejecución directa del automatismo o programa de autómata. Varios fabricantes en sus autómatas de gama alta hacen este paso directo, lo que lo ha convertido en un potente lenguaje gráfico de programación para [autómatas](https://es.wikipedia.org/wiki/Aut%C3%B3mata_programable" \o "Autómata programable), adaptado a la resolución de sistemas secuenciales. En la actualidad no tiene una amplia difusión como lenguaje, puesto que la mayoría de los autómatas no pueden programarse directamente en este lenguaje, a diferencia del [lenguaje Ladder](https://es.wikipedia.org/wiki/Lenguaje_Ladder" \o "Lenguaje Ladder). Pero se ha universalizado como herramienta de modelado que permite el paso directo a programación, también con Ladder.

Entendiendo estos 2 conceptos comenzaremos con nuestro GRAFCET.

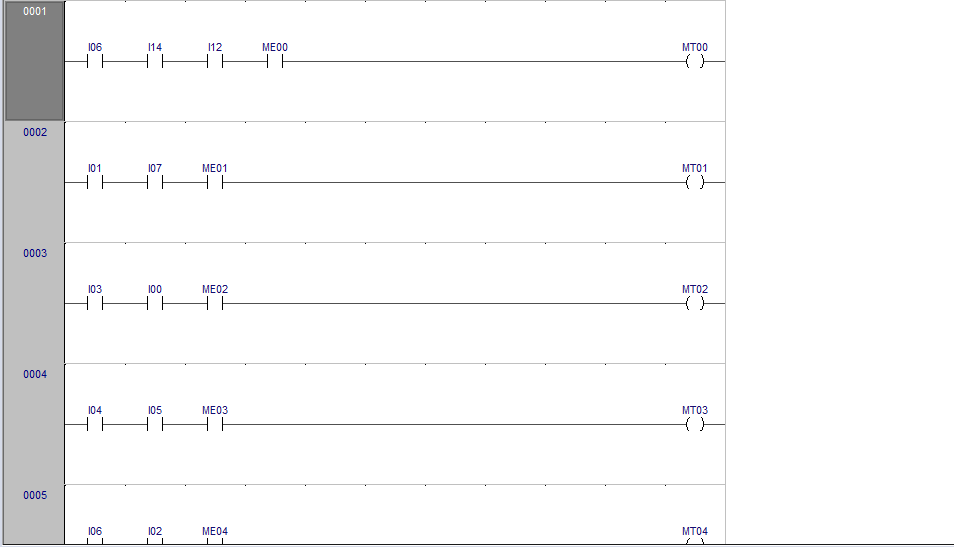


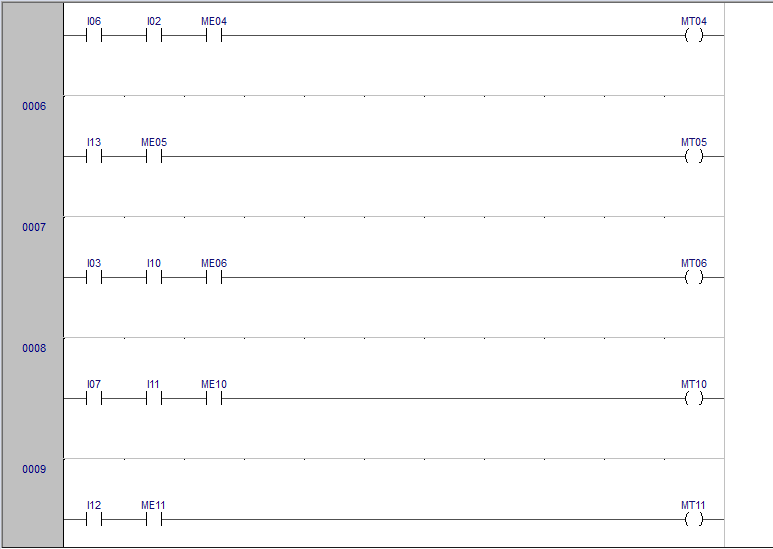
Teniendo nuestro GRAFCET podemos proceder a realizar nuestra programación en escalera, por lo cual comenzaremos a ubicar nuestras variables.

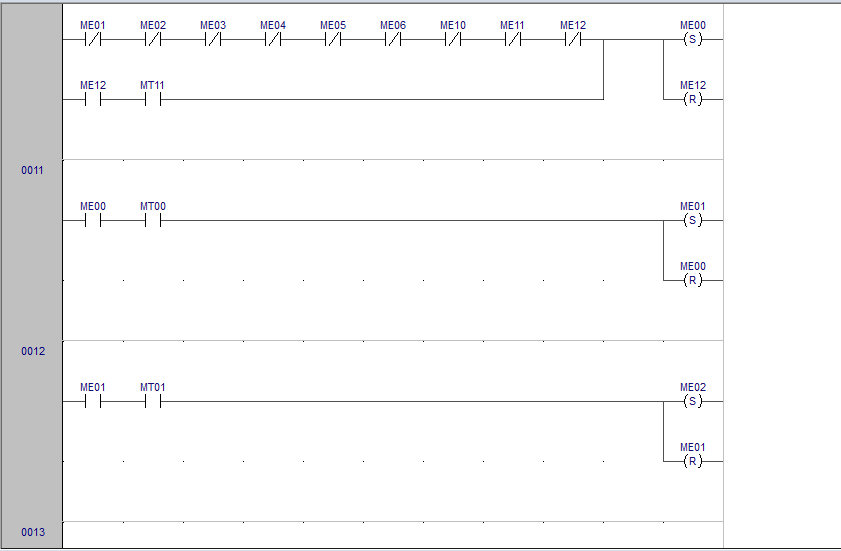


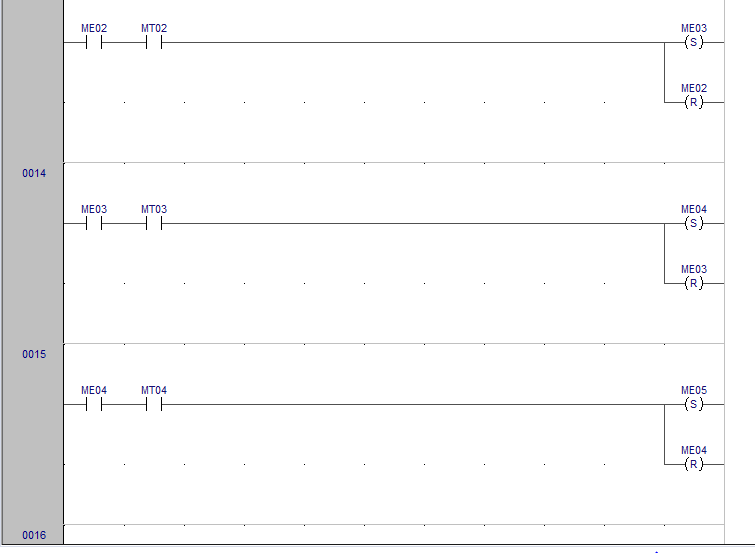


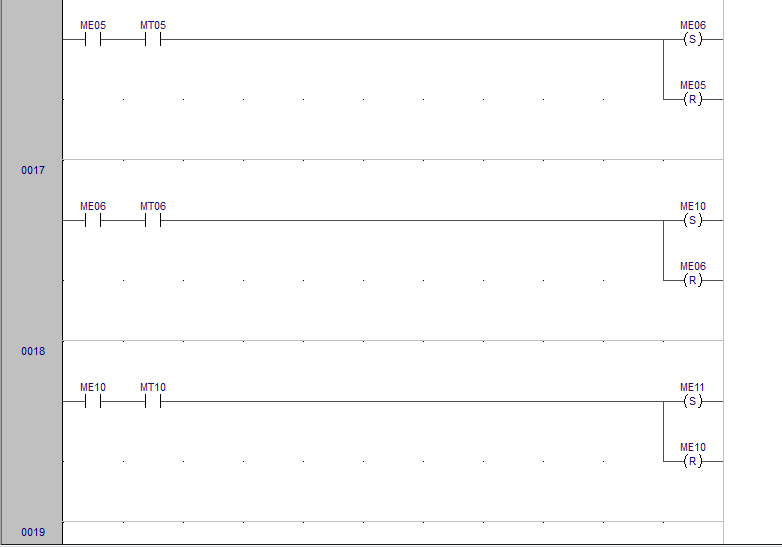
Después de declarar comenzamos la programación.

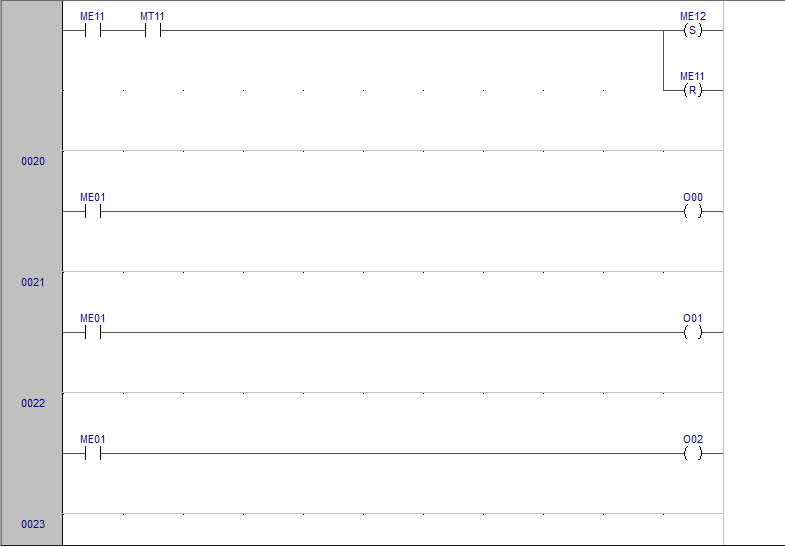


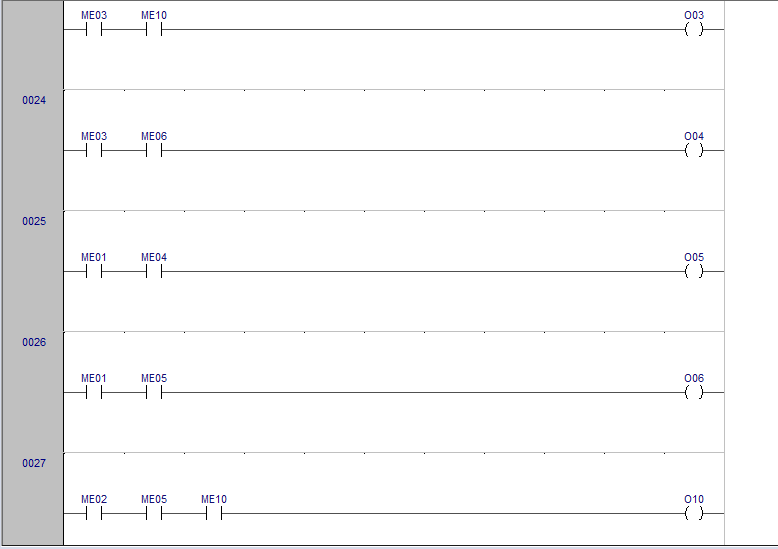


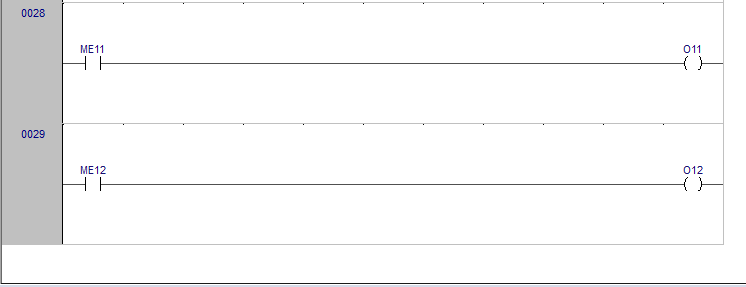






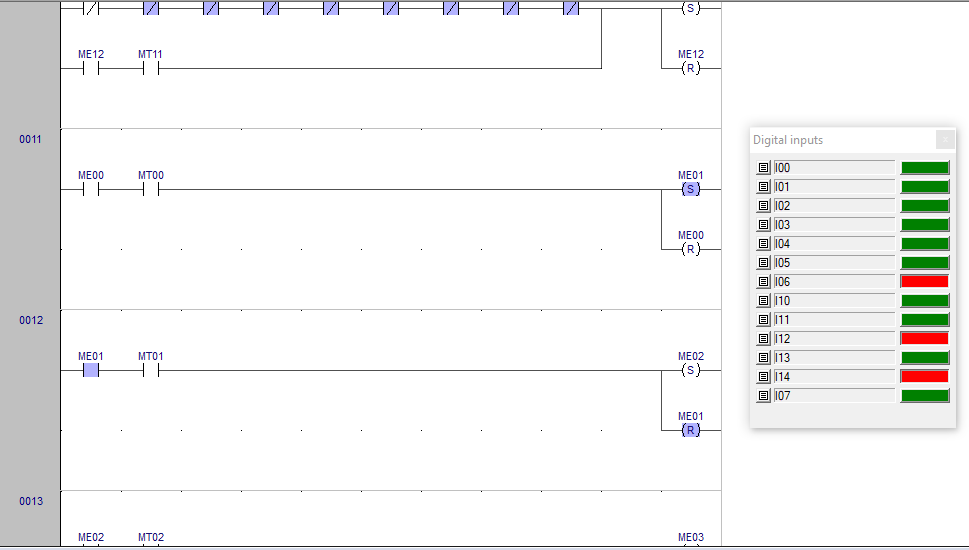




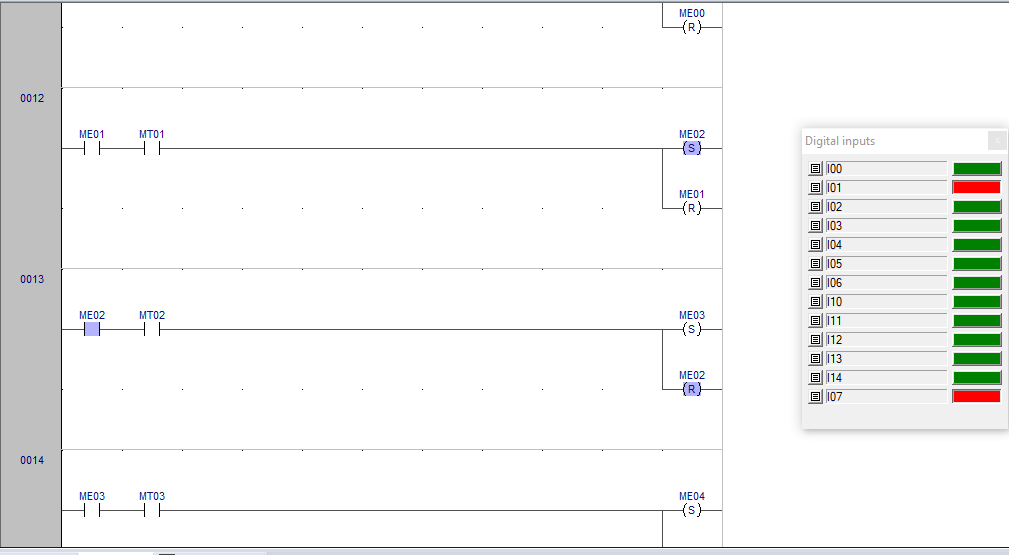


Aquí terminamos nuestra programación, ahora lo comprobaremos con la simulación.

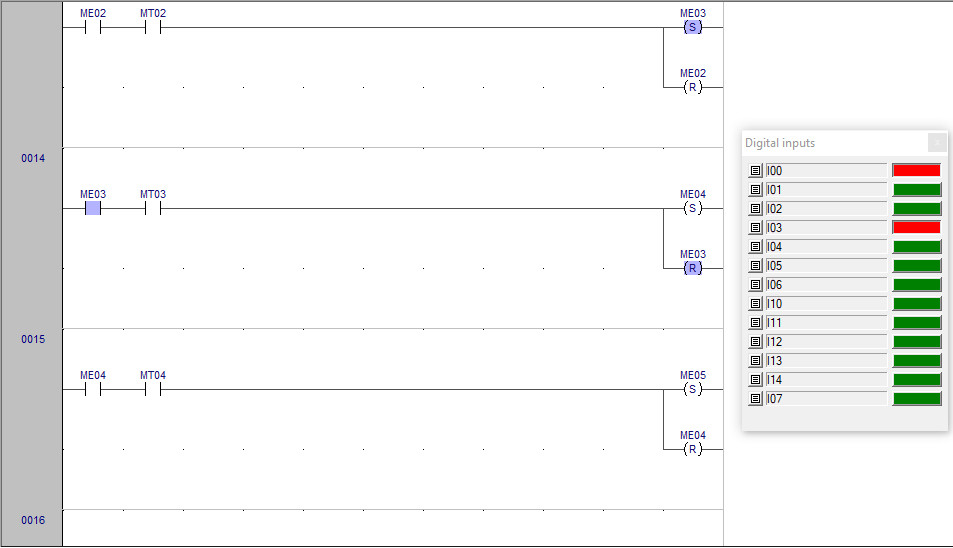
De acuerdo a nuestro GRAFCET para poder entrar en la etapa principal tenemos que encender el sensor **06,14,12**



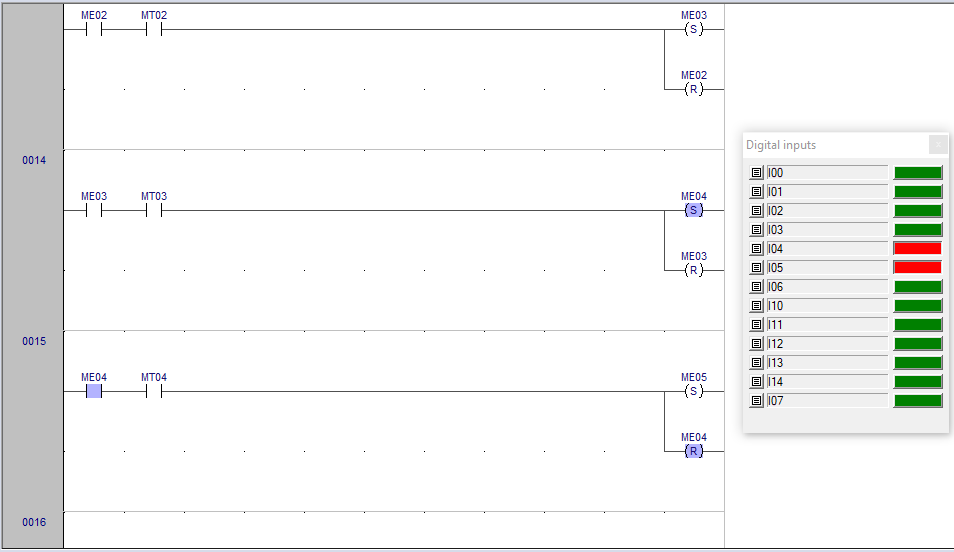
En este caso se cumple por lo cual pasa a la transición 1 y se guarda en la memoria de estado, esto ira en aumento es decir se pasara a la transición 2, la memoria de estado cambiara al 2do, etc.



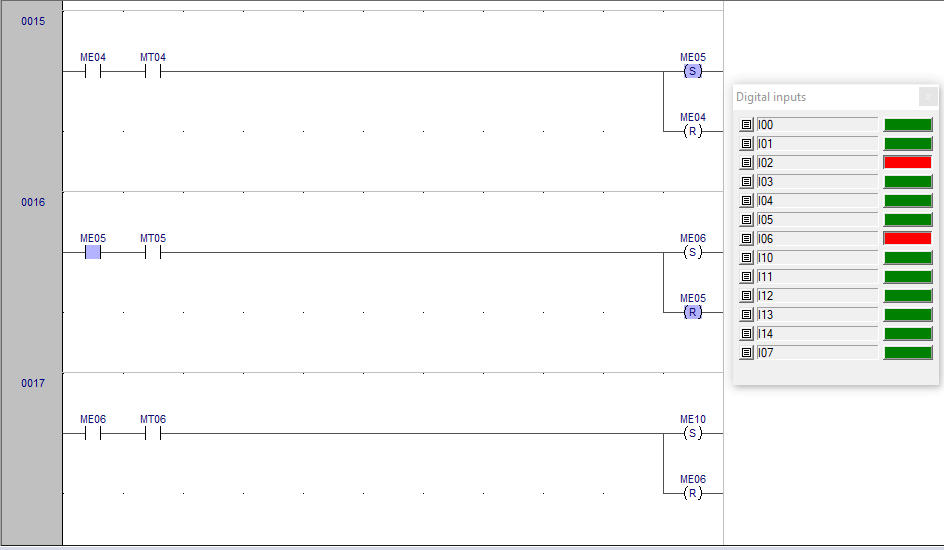
Aquí se dice que los sensores 01 y 07 tienen que estar encendidos para pasar a la siguiente transición el cual se efectúa perfectamente



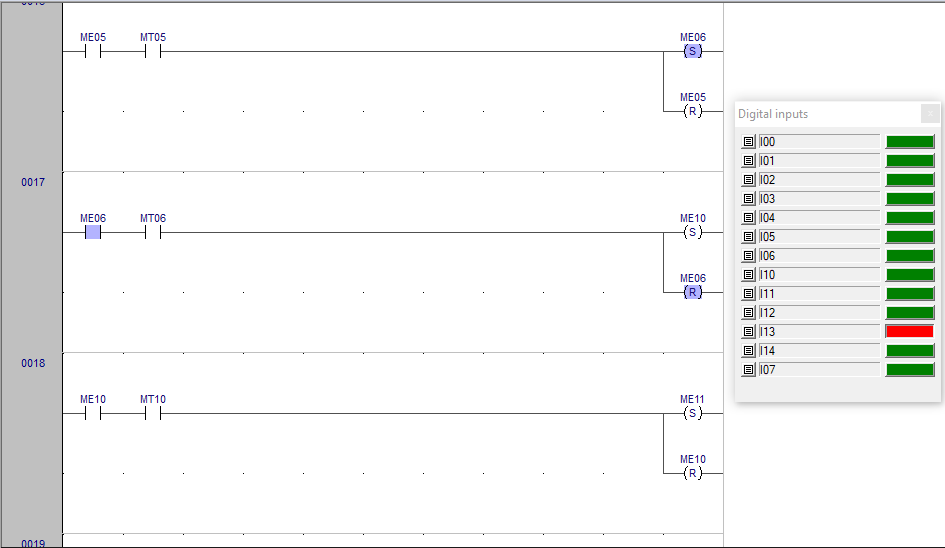
Aquí los sensores son **03,00** para pasar a la transición 3



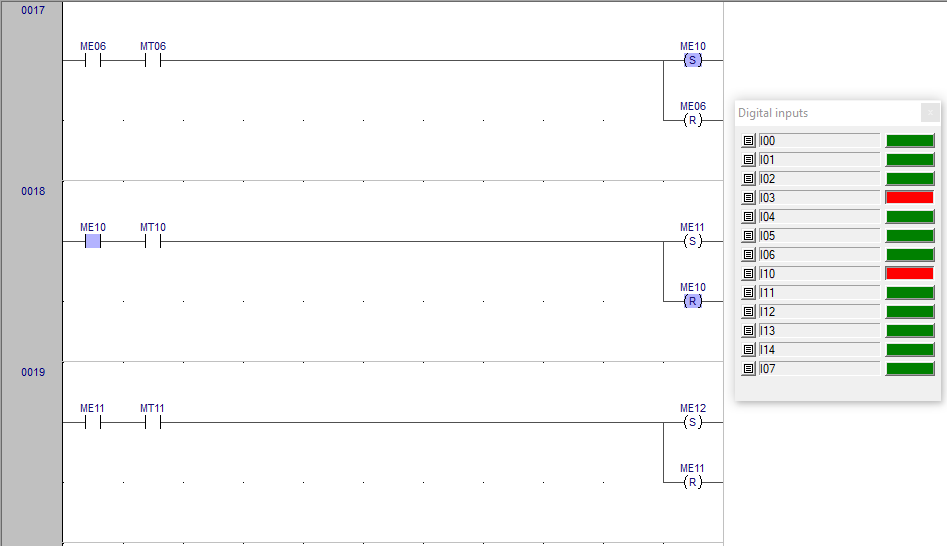
Sensores **04,05** pasando a la memoria de estado



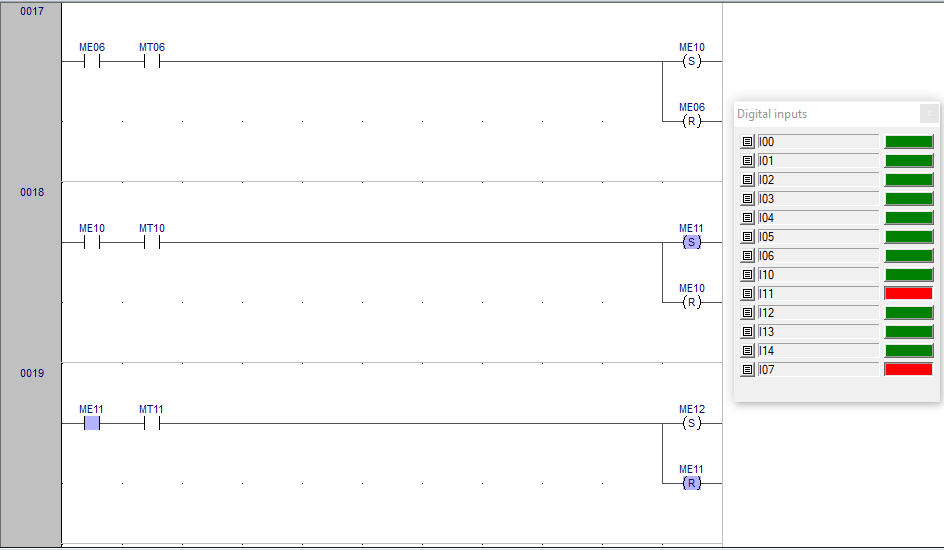
Sensores **06,02** memoria de estado 5



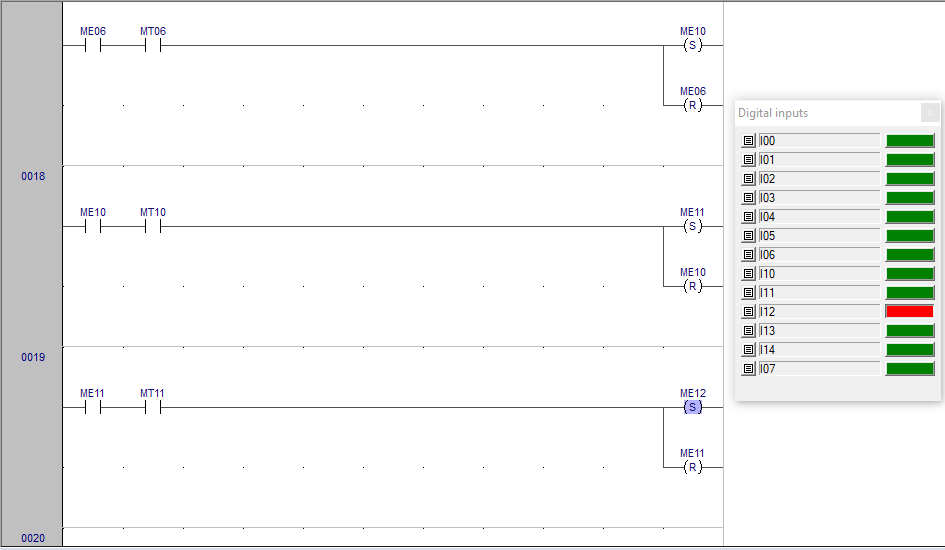
Sensor **13** memoria de estado 6



Sensores **03,10** memoria de estado 10



Sensores **11,07** memoria de estado 11



Sensores **12** memoria de estado 12

Con esto acabaríamos con el ciclo y regresaría a home tal como nuestro diagrama

**Conclusión**

Poco a poco este estilo de programación se comienza a manejar mejor, ya que no es muy fácil de entender si no se tiene una mínima de noción del mismo, por ende, este estilo de programación es bien pagado sobre todo en las empresas de automotriz.

Evidencia

