Elementos básicos de programación de autómatas

Sistemas Automáticos

Curso 2010-2011

1. Introducción

A continuación se ofrece una breve descripción, ilustrada con ejemplos, de las operaciones básicas de todo Autómata Programable:

- Funciones lógicas
- Consideraciones sobre el emisor
- Biestables
- Temporizadores

Para esta introducción se ha elegido el lenguaje gráfico de contactos, también conocido como lenguaje de escalera, *ladder diagram* o LD. Es el lenguaje más intuitivo para personal familiarizado con esquemas eléctricos, y fácilmente aplicable a problemas sencillos.

2. Funciones lógicas

Los ejemplos del apartado 2.1 muestran la implementación de las funciones lógicas básicas AND y OR usando el lenguaje de programación de contactos (LD). Para todos ello se han supuesto sensores normalmente abiertos. Obsérvese que la condición AND se corresponde con una configuración de contactos en serie, colocándose éstos en paralelo en el caso de la OR. Asimismo se muestra la solución en LD para una función más compleja. La resolución de un problema práctico sobre lógica combinacional se describe en el ejemplo del apartado 2.2.

2.1. Ejemplos de funciones lógicas

$$Q = A \cdot B$$

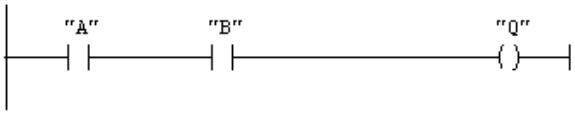

Figura 1: Implementación en LD de la función AND

$$Q = A + B$$

Figura 2: Implementación en LD de la función OR

$$Q = (A \cdot \overline{B} + C) \cdot D$$

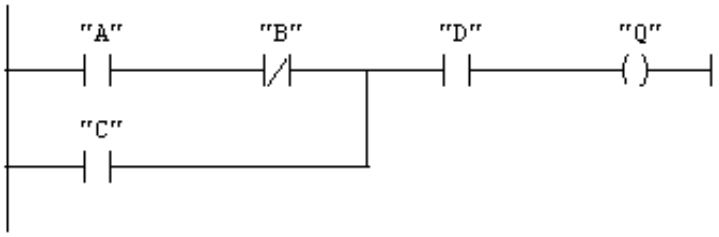

Figura 3: Implementación en LD de una función lógica

2.2. Ejemplo de aplicación de lógica combinacional

Activación y desactivación de la iluminación de un local, mediante el accionamiento de tres interruptores de configuración normalmente abiertos. Supóngase la sala de un museo en la figura 4. Se quiere que la iluminación no esté activada cuando se encuentre vacía. Para ello, cuando se entra en la sala, se pulsa el interruptor de la puerta por la que se acceda para que se encienda la luz. Cuando se abandone la

sala se debe accionar el interruptor correspondiente a la puerta por la que se sale, para apagar la luz (suponiendo que no quede nadie dentro).

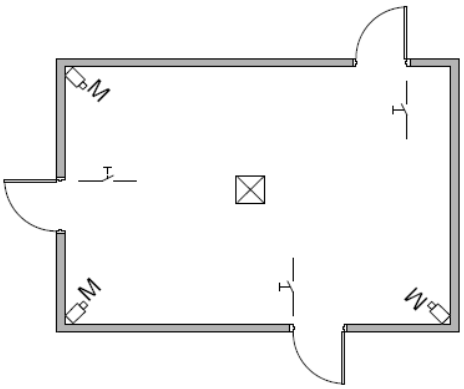

Figura 4: Habitación del museo

La función lógica correspondiente a la tabla de verdad que aparece en la figura 5 está expresada en la ecuación (1). La solución a la implementación de tres interruptores para la iluminación de una habitación se refleja en la figura 5.

$$Q = \overline{A} \cdot \overline{B} \cdot C + \overline{A} \cdot B \cdot \overline{C} + A \cdot \overline{B} \cdot \overline{C} + A \cdot B \cdot C \tag{1}$$

A	B	C	Q
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	0
1	0	0	1
1	0	1	0
1	1	0	0
1	1	1	1

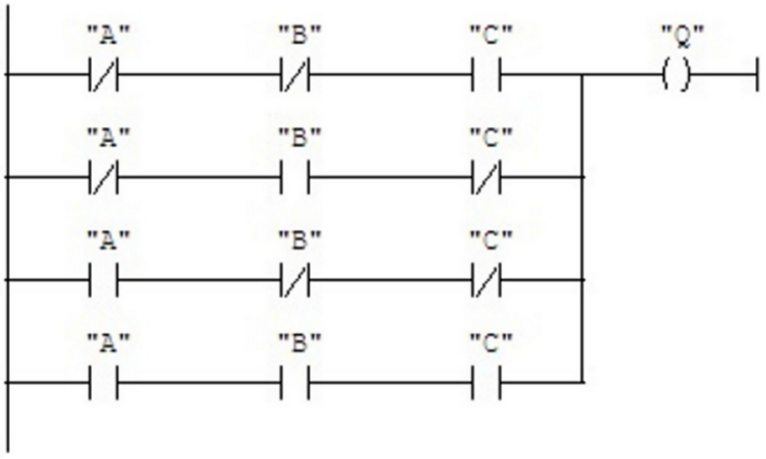

Figura 5: Tabla de verdad y solución en LD

Si bien puede ser útil el empleo de tablas de verdad, no lo es tanto su simplificación (por Karnaugh u otros medios). No es tan importante la minimización del número de 'puertas lógicas' como lo sería en un circuito implementado en hardware. En la mayor parte de las situaciones con tecnología programada es preferible una

mayor claridad e inteligibilidad del algoritmo y/o un menor tiempo de desarrollo, que el ahorro de unas líneas de código.

3. Consideraciones sobre el emisor

A la hora de programar es necesario tener en cuenta la lógica con que están implementados los sensores. Hay que pensar que el autómata no ve más allá de las bornas de sus tarjetas de entrada. Por lo tanto habrá que adecuar el programa según los elementos emisores de señal (sensores) sean de nivel activo alto o bajo.

En los ejemplos anteriores se ha supuesto que todos los sensores eran de nivel activo alto. Pero no siempre es así, ni tan siquiera es lo más habitual. En muchos casos la selección de sensores de nivel activo alto o bajo está condicionada por cuestiones de seguridad.

A continuación se muestra un ejemplo. Se pretende gestionar el encendido y apagado de una bombilla en función de la información obtenida de los sensores Sa y Sb cuyo estado de reposo se observa en el cableado de las entradas de la figura 6. La ley de control es la siguiente: *la bombilla sólo debe lucir cuando, simultáneamente, Sa esté activo y Sb no lo esté.*

Evidentemente la solución programada será diferente según cual sea la configuración de los sensores. Para resolver el problema conviene fijarse en qué señal aparece en los bornes del autómata (lo que en realidad ve en sus entradas), según el estado de detección de los sensores y la ley de control.

En la situación de la figura 6, Sa es de nivel activo alto, o normalmente abierto (en adelante NA) y Sb es de nivel activo bajo, o normalmente cerrado (en adelante NC). Entonces, si Sa está activo aparecerá un nivel de tensión alto en la entrada %I0.0 y si Sb no detecta, por ser de nivel activo bajo aparecerá también un nivel de tensión alto en la entrada %I0.1. Así pues, la condición que habrá que programar para que se encienda la bombilla es que %I0.0 esté a 1 y %I0.1 esté a 1.

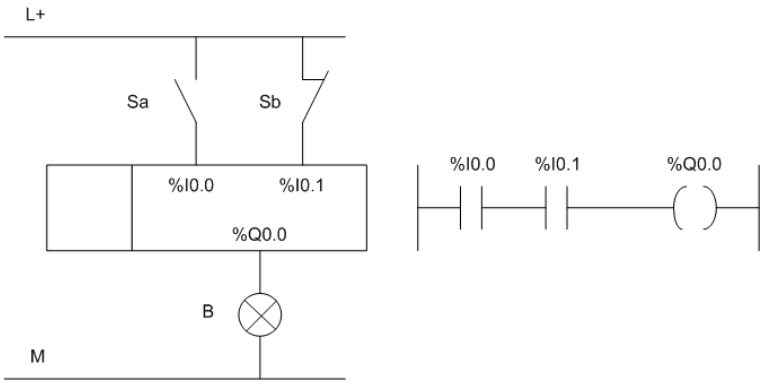

Figura 6: La bombilla luce si Sa detecta y Sb no

Para evitar problemas, es necesario hablar con propiedad. Se puede decir que

los sensores conectados al AP son NA o NC, pero cuando se habla de los elementos del programa es peligroso emplear esa denominación. En este caso es más apropiado hablar de *consultas de estado* y decir que si %I0.0 está en estado alto y %I0.1 está también en estado alto se activará la salida %Q0.0.

4. Biestables

Los biestables permiten mantener el estado de una variable aún cuando sus entradas se desactiven. Son muy usados cuando se trabaja con pulsadores para memorizar las órdenes.

Se representan de forma similar a la asignación, pero con una S (set) o una R (reset) dentro.

En el programa de la figura 7, una pulso en %I0.0 pondrá la salida %Q0.0 a uno. Aunque la entrada vuelva a cero la salida seguirá a uno, hasta que no haya orden de reset. Esto ocurrirá cuando se active la entrada %I0.1, e inmediatamente se apagará la salida.

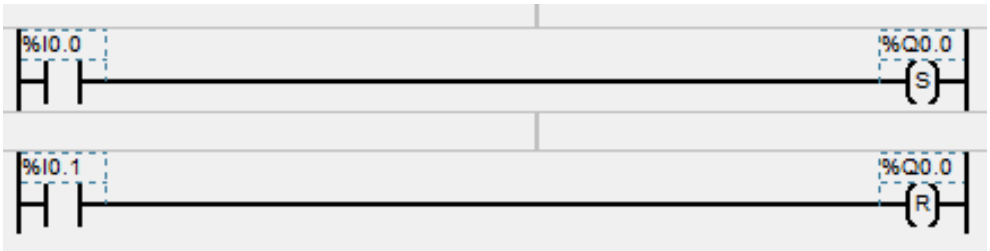

Figura 7: Instrucciones Set y Reset

Si coincidiesen en el tiempo los pulsos de set y de reset, tendrá prioridad la última instrucción que se ejecute. En el ejemplo de la figura 7 la salida quedaría a cero, pues la última instrucción es el reset. Si se quisiera dar prioridad a la puesta a uno (que no suele ser la condición más segura), se escribiría la instrucción de set a continuación de la de reset.

Por último, queda comentar que las instrucciones de set y de reset no tienen que ir necesariamente seguidas (puede haber otras entremedio), ni por pares (puede haber un set y varios o ningún reset, y viceversa).

4.1. Biestable con realimentación de la salida

Otra forma muy empleada, sobre todo en los esquemas de contactos eléctricos, de conseguir la función de un biestable es mediante la realimentación del estado de la salida. Las salidas, así como las variables internas (marcas), se pueden consultar además de asignar. En la figura 8 se puede ver un ejemplo de un esquema.

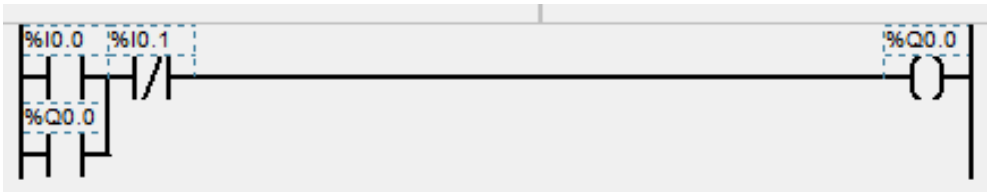

Figura 8: Biestable construido con consulta de la salida

Si %I0.0 pasa a estado alto y %I0.1 se encuentra en estado bajo se activa la salida %Q0.0. Una vez activada la salida, la consulta de su estado permite que ésta se mantenga activa, aunque %I0.0 pase a estado bajo, siempre y cuando %I0.1 se siga en estado bajo. En el momento en que %I0.1 pase a estado alto se desactivará la salida, independientemente del estado del resto de las variables.

La entrada %I0.0 actúa como señal de Set y la %I0.1 de Reset. El que se realicen consultas de estado alto o bajo en ellas dependerá del nivel activo de los sensores que tengan cableados.

4.2. Ejemplo del uso de biestables

La figura 9 muestra un depósito llenado por una electrobomba que se pretende controlar mediante dos sensores de nivel (NA). El estado de disponibilidad o no de servicio lo proporciona un interruptor de funcionamiento. La electrobomba se pondrá en marcha cuando el nivel descienda por debajo del mínimo y se apagará cuando: bien se alcance el máximo, bien salte el térmico (NC) que protege el motor eléctrico de la bomba, o bien sea desconectada mediante el interruptor. Si la electrobomba está en funcionamiento deberá lucir una lámpara indicadora. Si salta el térmico se encenderá una lámpara de aviso.

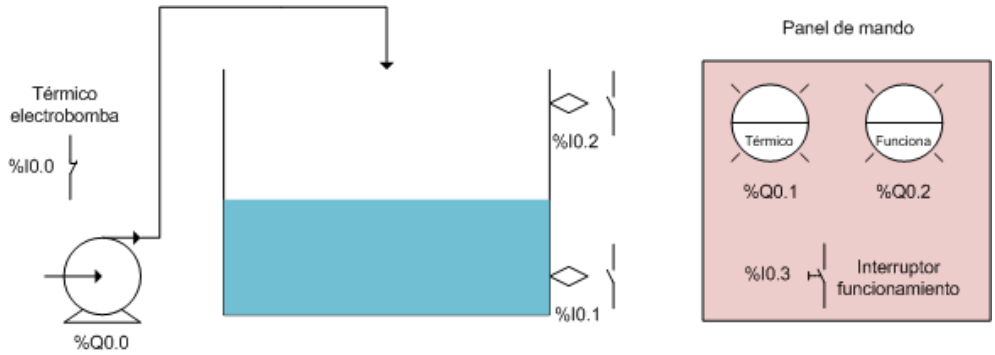

Figura 9: Depósito que se pretende controlar

la solución al ejemplo planteado se puede ver en la figura 10.

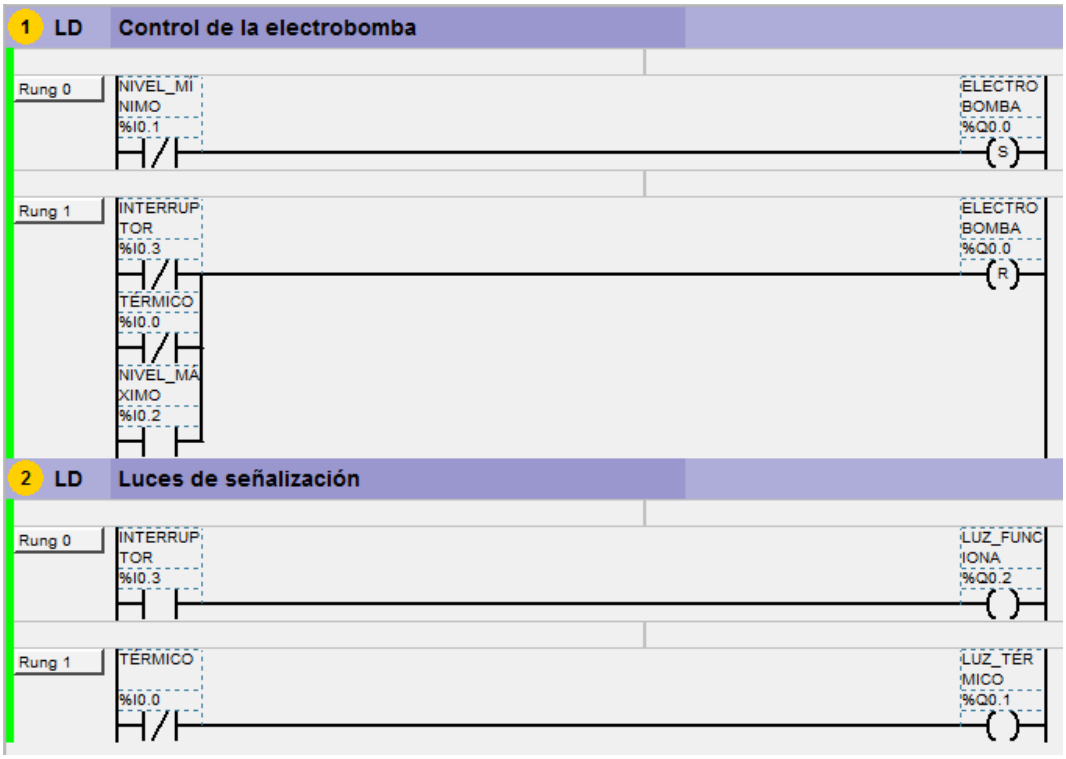

Figura 10: Programa en LD para el control de nivel del depósito

5. Temporizadores

Para gestionar la activación o desactivación de determinadas acciones en función del tiempo transcurrido desde un determinado evento se usarán los bloques *temporizadores*.

En el autómata, un temporizador se denotará: **%TMi**, donde 'i' es un número (e.g. %TM5). Son objetos que se almacenan en la memoria interna. No se corresponden por tanto con entradas ni salidas del autómata.

Twido dispone de 3 tipos de temporización:

- TON: Retraso a la conexión. La salida se activa después que la entrada, con un retraso configurable. Cuando la entrada vuelve a cero, la salida se pone a cero inmediatamente.
- TOF: Retraso a la desconexión. Cuando la entrada se pone a uno, la salida se pone inmediatamente a uno. Cuando la entrada vuelve a cero, la salida vuelve a cero también, pero más tarde, con un retraso configurable.
- TP: Temporizador de impulso. Cuando la entrada se pone a uno, la salida se pone también a uno, durante un tiempo fijo, limitado y configurable. Una vez que la salida está a uno, no depende del valor de la entrada, da igual que se

ponga a cero, que la salida permanecerá a uno hasta el final de la temporización. Si se produce un nuevo impulso mientras aún está temporizando el anterior, el nuevo será ignorado.

La activación de los temporizadores TON y TP se produce cuando ocurre un flanco ascendente (la señal en el ciclo anterior valía cero y en el actual uno) en su patilla de entrada. El temporizador TOF por el contrario se activa con el flanco descendente de la entrada.

A continuación, en la figura 11 se puede ver la evolución de la salida de cada temporizador para una misma entrada y un mismo tiempo configurado (T).

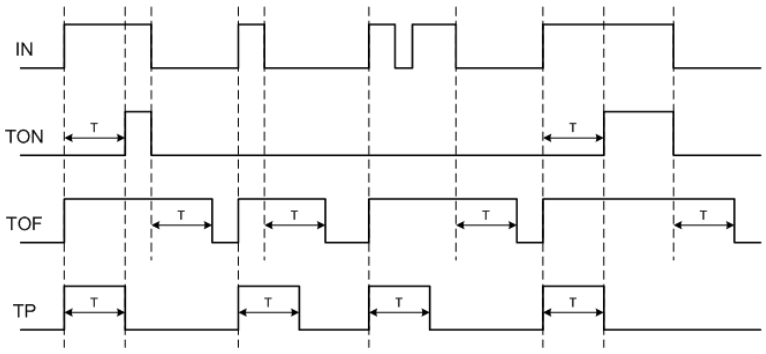

Figura 11: Cronogramas de los diferentes temporizadores

5.1. Ejemplo del uso de temporizadores

Se pretende controlar una cinta de transporte mediante un interruptor que gestione su arranque y parada. La cinta debe comenzar a funcionar 30 segundos después que se active el interruptor. Además, se debe de hacer sonar una bocina durante los 10 segundos previos al arranque de la cinta.

La solución al ejemplo planteado aparece en la figura 13. El cronograma asociado se puede ver en la figura en la figura 12.

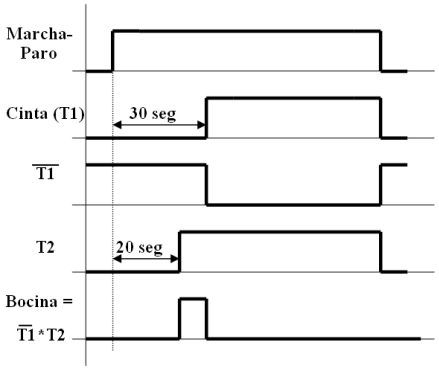

Figura 12: Cronograma de funcionamiento de la cinta y la bocina

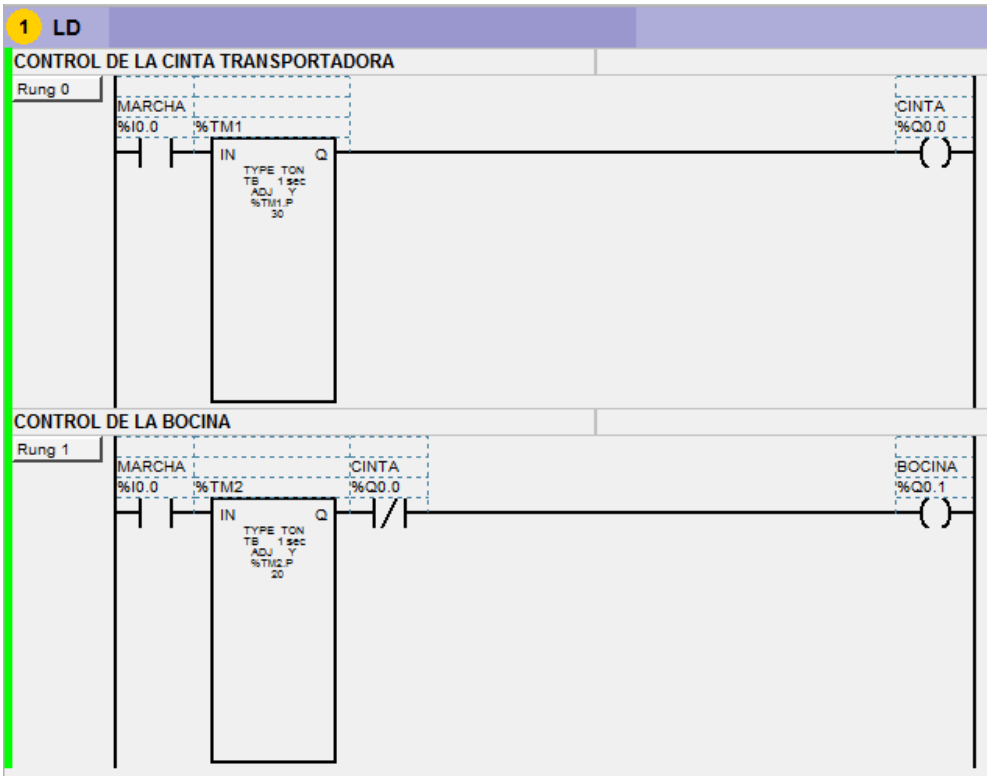

Figura 13: Solución en LD al problema de control de la cinta transportadora

6. Para saber más

Puede descargar el manual: *TwidoSuite V2.2. Guía de programación*
http://www.global-download.schneider-electric.com/85257578007E5C8A/all/1FD7B0C140308E4F8825757800476557/$File/35013228k01000.pdf