AUTOMATIZACIÓN INDUSTRIAL

PRÁCTICA 2

Esta práctica consta de tres ejercicios a desarrollar.

Ejercicio 1.- Automatización de un cruce de semáforos

Este ejercicio trata de ordenar el tráfico de forma automatizada, de un cruce de semáforos como muestra la Figura 1.

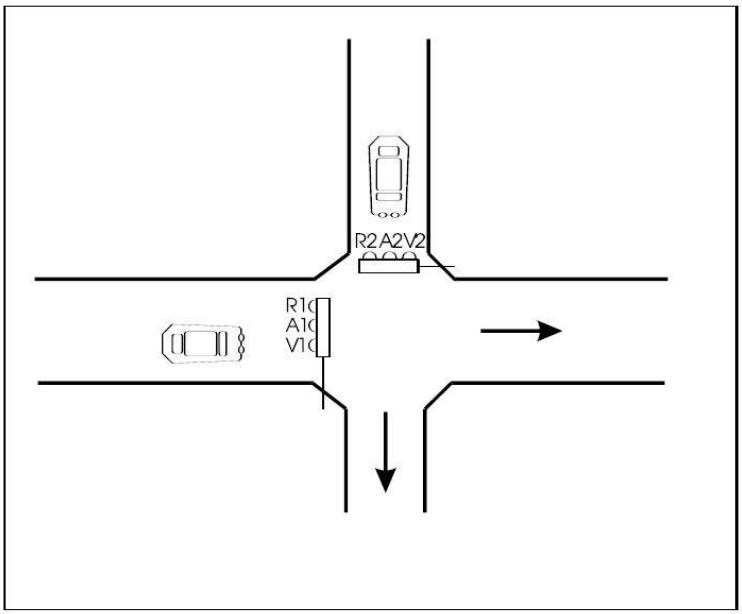

Figura 1.- Esquema del cruce de semáforos

El funcionamiento del sistema deberá cumplir las siguientes especificaciones:

- Se ha de disponer de los colores típicos por cada uno de semáforos que serán identificados como Rojo1, Amarillo1, Verde1, Rojo2, Amarillo2, Verde2 que se ubicarán en las direcciones 1.0, 1.1, 1.2, 1.8, 1.9 y 1.10 respectivamente.
- El sistema estará controlado por dos entradas 0.0 y 0.1
- Si ninguna de las entradas está activa, el sistema permanecerá en reposo con todas las luces apagadas.
- Si se activa la primera entrada, pero no la segunda entrada, el sistema estará en el modo de funcionamiento 1 en el que se harán ciclos de 1 minuto de la forma:
    - 30 segundos con Verde1 y Rojo2
    - 5 segundos con Amarillo 1 y Rojo2
    - 20 segundos con Rojo1 y Verde2
    - 5 segundos con Rojo1 y Amarillo2
- Si tras tres ciclos de repetición del ciclo descrito anteriormente, la entrada 1 sigue activa se insertará un ciclo adicional en el que durante 30 segundos las luces Amarillo1 y Amarillo2 estarán parpadeando de forma alternativa
- Si se activa la segunda entrada, el sistema pasará a funcionar de forma que las luces Amarillo1 y Amarillo2 estarán parpadeando de forma alternativa hasta que se desactive la segunda entrada.

Resolver mediante una red de Petri

(Al validar la práctica, cambiar la referencia de todas las temporizaciones a tres segundos)

Ejercicio 2.- Máquina expendedora de caramelos

Una máquina expendedora de caramelos, puede aceptar monedas de 5 y 10 céntimos de Euro. Se expenden 2 tipos de caramelos A o B, de precios 15c y 20c respectivamente.

El mecanismo de expulsión de los caramelos se realizada mediante un pulsador activado por el cliente una vez alcanzado el precio correspondiente.

Diseñar un modelo representativo de un algoritmo de detección de pago y expedición de caramelo. Dotar al sistema de una opción de devolución de monedas en caso de desestimar la

compra.

Deberán activarse las siguientes salidas:

- Un LED (salida 1.0) que indique máquina dispuesta para admisión de monedas.
- Una barra de leds que indique el dinero acumulado a medida que se va introduciendo.
    - Un led (salida 1.1) 5c
    - Dos leds (salidas 1.1 y 1.2) 10c
    - Tres leds (salidas 1.1, 1.2 y 1.3) 15c
    - Cuatro leds (salidas 1.1, 1.2, 1.3 y 1.4) 20c
    - Cinco leds (salidas 1.1, 1.2, 1.3, 1.4 y 1.5) parpadeando, mayor de 20c
- Dos LEDs (salidas 1.6 y 1.7) que indiquen respectivamente caramelo A o B listo para expulsión.
- Activación de expulsión de caramelo A o B (salidas 1.8 y 1.9) respectivamente. Simúlese un tiempo de expulsión de 3 segundos del caramelo, hasta que la máquina está dispuesta para una nueva operación.
- La máquina sólo funciona con importe exacto, por lo que una vez alcanzados o superados los 20c, la máquina no admite más monedas. Si el importe alcanzado es de 20c, podremos extraer el caramelo B o recuperar las monedas. Si el importe alcanzado supera los 20c (15c + moneda de 10c) sólo podremos recuperar las monedas

Consideraciones a tener en cuenta:

El flag de puesta en marcha del sistema es una señal de sólo lectura que el PLC activa al iniciar el programa y que permanece a 1 durante el primer ciclo de programa. Una vez pasado el primer ciclo del programa, el PLC pone a 0 dicho flag. Es imposible ver en el CX-Programmer dicho flag a 1, ya que la duración del ciclo del PLC será del orden de milisegundos. Dicho flag tiene un símbolo predefinido que se denomina P_First_Cycle.

Operaciones con números enteros.

Podemos operar con número enteros direccionando palabras de memoria, como con los contadores y temporizadores. En el ejemplo que nos ocupa bastará con las opciones de suma y puesta a cero. Si la máquina devolviese cambio, añadiríamos la resta. Para todos esto utilizaremos una dirección de la zona de DM (data memory) que direccionaremos directamente como palabra. Como ejemplo, utilizaremos el D300, es decir, la palabra número 300. Si en vez de utilizar un PLC CJM utilizamos un CQM1H la denominación será DM300.

Puesta a cero

La puesta a cero es el procedimiento más sencillo de todos. Consiste en mover el valor numérico cero a la dirección seleccionada, en este caso la D300. Para ello basta con utilizar la función mover datos (mnemónico MOV).

Tal como muestra la Figura 2, la función MOV es un bloque que tiene una entrada y no tiene salida. La entrada sirve para determinar cuándo se ejecuta la acción de movimiento de datos. En el ejemplo de la figura, la dirección D300 se pondrá a cero cuando se inicie el programa (inicialización de la variable haciendo uso del flag de inicio, P_First_Cycle) y cada vez que la dirección de memoria 10.0 se active.

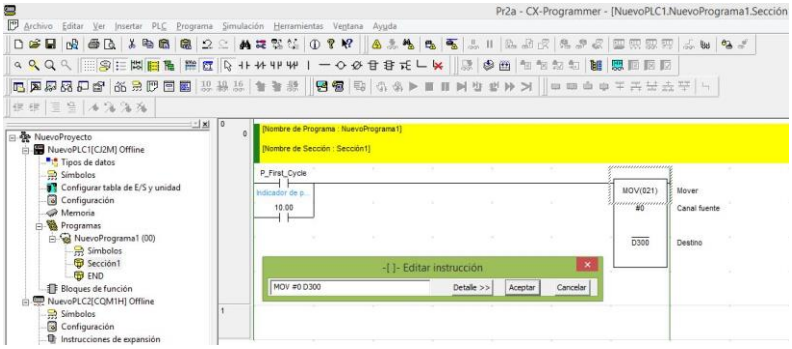

Figura 2.- Uso de la función MOV

La definición de la función es muy sencilla, se compone del mnemónico, los datos origen y la dirección de destino. Por esta razón, en la definición mostrada en la figura anterior podemos apreciar el mnemónico MOV como primer dato. A continuación #0 (valor 0) y seguidamente D300, dirección 300 del Data Memory. Si queremos copiar el contenido de una palabra de memoria en otra dirección distinta, lo haríamos utilizando las direcciones de memoria correspondientes. Por ejemplo, MOV D100 D300 copiará el contenido de la palabra 100 a la palabra 300.

Muy importante, si se nos olvida el símbolo # junto con la constante, en realidad estaremos utilizando un puntero a memoria, no una constante. Si definimos la orden como MOV 300 D300, estaremos moviendo la palabra 300 de la zona de memoria de entrada salida común (CIO common input output) a la palabra 300 de la zona de memoria de datos. El símbolo #significa que lo que viene a continuación es una constante codificada en BCD, mientras que el símbolo & indica que lo que viene a continuación es una constante codificada en binario natural. Así, si expresamos #12 la trama binaria equivalente, lo que verá el PLC será 00010010. Si expresamos &12, la trama que verá el PLC será 1100. Hay que tener en cuenta que para números de una cifra, las representaciones en binario natural y en BCD coinciden.

Suma

La función que utilizaremos para sumar depende del PLC que estemos utilizando. Si utilizamos un PLC CJM2 utilizaremos el mnemónico + (suma binaria) o +B (suma en BCD) y para un CQM1H utilizaremos el mnemónico ADD (suma en BDC) o ADB (suma en binario). Excepto el mnemónico, el resto coincide para ambos casos. La función tiene un terminal de entrada que al activarse hace que se sume, no tiene salida y la definición consiste en mnemónico, dato1, dato2 y dirección del resultado. Así, una definición del tipo “+ D300 #2 D300” significa que vamos a sumar el contenido de D300 con la constante 2 y el resultado se almacenará en la dirección

D300. Hay un pequeño detalle a tener en cuenta, mientras que la entrada siga activa, se seguirá produciendo la suma, por lo que hay que hacer que la acción sea impulsional, es decir, que la señal que pongamos en la entrada sea la detección de un flanco.

Si estamos utilizando un PLC CJ2M, es muy sencillo, tal como muestra la Figura 3

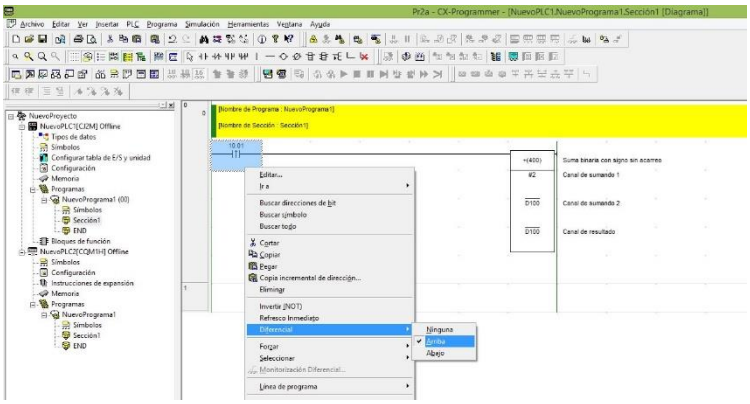

Figura 3.- Detección de flanco por diferencial (PLC CJ2M)

Así, la acción suma sólo se llevará a cabo una vez, cuando la señal pase de 0 a 1 gracias a la detección de flanco.

Si utilizamos un PLC de otra gama, como los CQM1H, nuestra tarea será un poco más laboriosa.

En primer lugar, cambian los nombres de las direcciones de memoria y de la función. Ahora el mnemónico de la función no será "+" sino "ADD" y la dirección D300 será DM300. De modo que para definir la función suma, en vez de utilizar la sentencia "+ D300 #2 D300" utilizaremos "ADD DM300 #2 DM300".

La mayor diferencia es la detección de flanco, si en un autómata de la gama CQM1H o de la gama C200 activamos el diferencial como lo hemos visto antes no nos validará la ecuación, hay que utilizar la función DIFU (differentiate up) para el flanco de subida y DIFD (differentiate down) para el flanco de bajada.

En primer lugar, ponemos el contacto abierto 0.0, y seguidamente utilizamos la función differentiate up de la forma "DIFU 10.00", con esto cerramos la primera ecuación, tal como muestra la Figura 4. De este modo, la detección del flanco de subida en la entrada 0 se produce

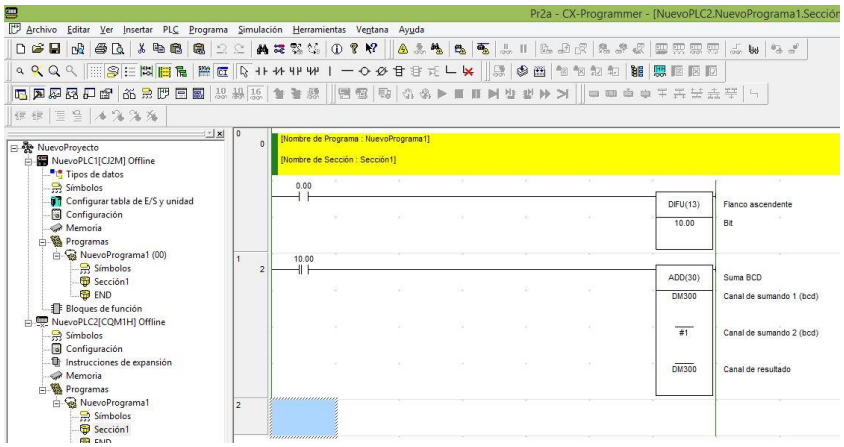

Figura 4.- Detección de flanco y suma en un PLC de la serie CQM1H

Seguidamente, insertamos la segunda ecuación con un contacto abierto referenciando la dirección 10.0 (la de detección del flanco). Como esa dirección es el resultado de la función DIFU, sin que nosotros hagamos nada el icono representativo aparece alterado, de modo que tiene una doble barra a la izquierda. Es el indicativo de que funciona por flanco de subida. A continuación, insertamos la función ADD con la sentencia "ADD DM300 #1 DM300". Con esto conseguimos que cada vez que se produzca un cambio de 0 a 1 en la entrada 0.0, el valor de la dirección DM300 se incremente en una unidad.

Resta

La resta es exactamente igual que la suma, es decir:

- Las funciones deben activarse mediante la detección de flancos, directa en el caso de un PLC CJ2M o a través de differentiate up en el caso de un PLC CQM1H.
- Para el caso de un PLC CJ2M la función tiene el mnemónico "-". Así, definiríamos la función como a "- D300 #1 D300". Si el autómata es de la serie CQM1H el mnemónico es "SUB", con los que la definición quedaría como "SUB DM300 #1 DM300".

Lo que sí es necesario tener en cuenta es el orden de los operandos, así como en la suma da igual, en la resta en primer operando es el minuendo y el segundo el sustraendo. Al igual que en la suma, el tercer parámetro es la dirección de almacenamiento del resultado de la operación

La Figura 5 muestra un ejemplo donde se combinan suma y resta en un PLC del tipo CQM1H

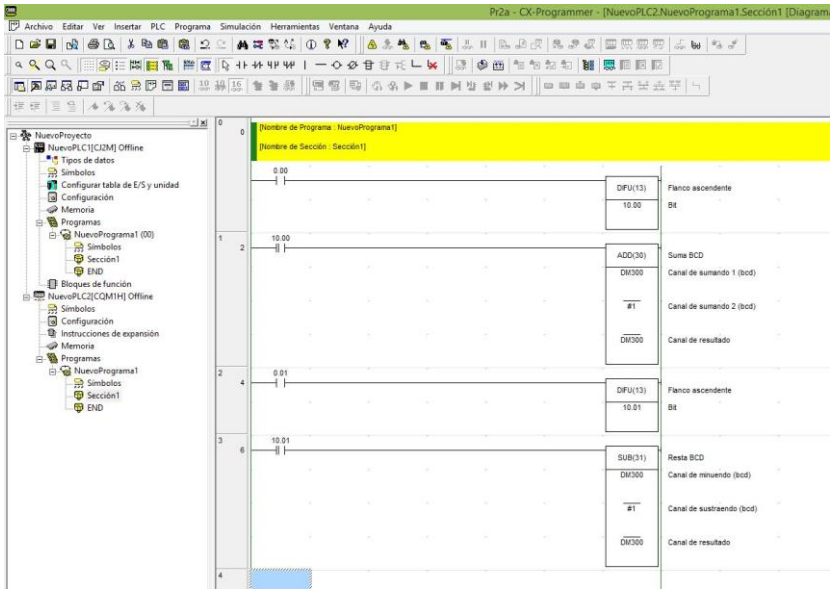

Figura 5.- Suma y resta en un PLC de la gama CQM1H

Con la información suministrada, ya se está en condiciones de resolver el ejercicio. Resolver mediante la programación de la red de Petri adecuada.

Ejercicio 3.- Automatización de un cruce de semáforos

Este ejercicio trata de gestionar el tráfico de una red de ferrocarril en la que nos encontramos dos vías en sentido inverso y un túnel compartido, de modo que por el túnel no puede haber dos trenes circulando en sentido contrario. El esquema se muestra en la **¡Error! No se encuentra el origen de la referencia.**.

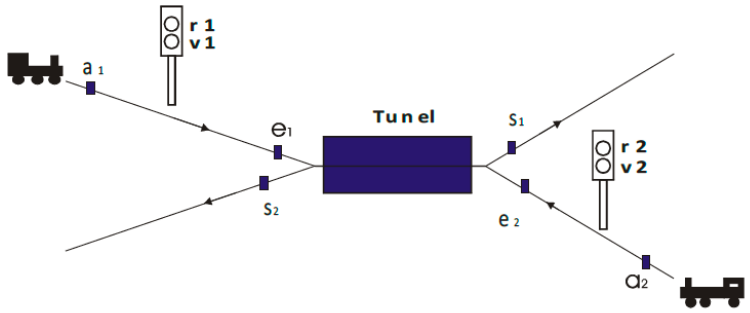

Figura 6.- Esquema de las vías con el túnel compartido

Las señales a_1, e_1 y s_1 corresponden a las señales de aproximación, entrada al túnel y salida del túnel para la vía 1, mientras que las señales con el subíndice 2 tienen el mismo significado, pero para la segunda vía.

El sistema debe controlar el acceso al túnel. Para ello dispondrá de las señales de salida v_1, v_2 (verde) y r_1, r_2 (rojo) que autorizan (verde) y prohíben (rojo), respectivamente la entrada al túnel en el sentido correspondiente.

Se desea que la salida del sistema sea tal que se activen de forma simultánea r1 y r2 en los supuestos siguientes:

- Si no hay tren en espera o recorriendo el túnel (el automatismo no sirve más que para autorizar una vía de paso: seguridad intrínseca)
- Si hay un tren que recorre el túnel

Un tren que espera en sentido 1 tiene preferencia sobre un tren que espera en el sentido 2. Supóngase que puede haber un número ilimitado de trenes que esperan en cada sentido.

Resolver mediante redes de Petri generalizadas (no binarias)