Capítulo 2

2. DESCRIPCIÓN Y SINTAXIS DE LOS MÓDULOS LÓGICOS QUE PUEDE IMPLEMENTAR EL PLM08.

2.1 Descripción general de los módulos lógicos.

Los módulos lógicos (ML) que puede realizar el PLM pueden ser representados a nivel de "caja negra" como se muestra en la figura 2.1, donde se muestra un módulo lógico, que representa "n" entradas y "m" salidas; "m" y "n" varían de acuerdo con el tipo de función que un determinado módulo lógico realice; así por ejemplo, para una compuerta AND de tres entradas "n" y "m" serían tres y uno respectivamente.

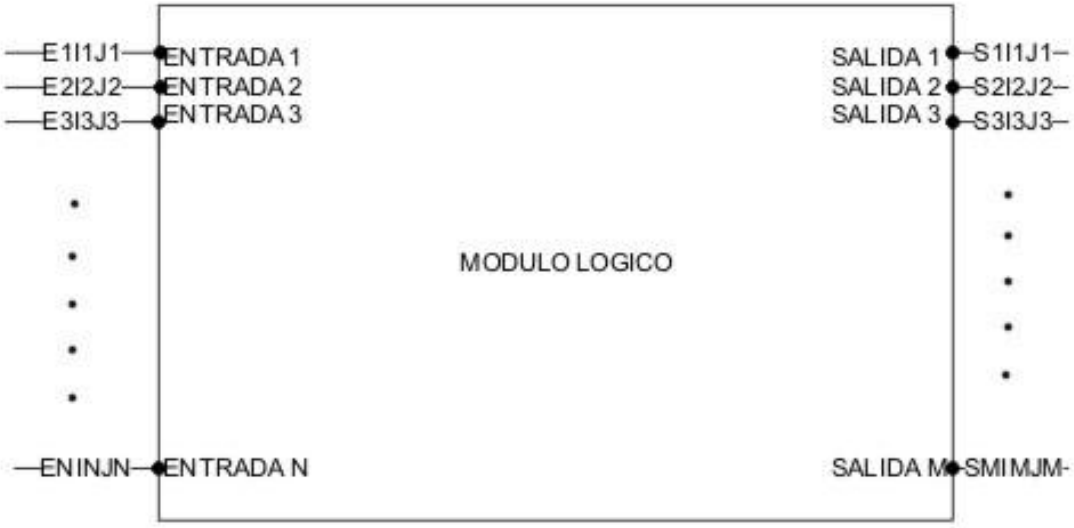


Figura 2.1 Representación genérica de un módulo lógico de "N" entradas y "M" salidas.

Los módulos lógicos propios del PLM08 son:

- Compuertas and, nand, or, nor, de dos, tres y cuatro entradas.
- Inversores y seguidores lógicos
- 3 tipos diferentes de temporizador
- Dos tipos de contadores de eventos
- Flip-Flops asícronos

Las variables asociadas con el PLM08 se denotan con una letra (“e” para entrada, “s” para salida e “i” para intermediaria) seguida por dos números que designan al grupo y al número de variable implicada. Así, “E10” denotaría la variable binaria cero del grupo de entradas uno; “I37” designaría a la variable binaria siete del grupo tres de variables intermediarias y “S04” denotaría a la variable binaria cuatro del grupo de salidas cero.

2.1.1 Programación del PLM08.

Una determinada aplicación de automatización realizada con el PLM08, requerirá del concurso de diversos módulos lógicos interconectados.

Características básicas del lenguaje SIIL1.

Al desarrollar una aplicación con el PLM, cada módulo lógico debe ser declarado por medio de uno o varios renglones de texto que expresan en alguna forma el tipo de módulo y características del mismo, de esta forma al conjunto de módulos requeridos le corresponderá un renglón contenido en un archivo de texto, el cuál es procesado por una computadora (PC) mediante software que genera el código objeto que deberá ejecutar el procesador central del PLM08, que para el prototipo es una tarjeta basada en el microcontrolador 68HC908GP32. Además de las declaraciones asociadas con los módulos lógicos, se requiere de otras instrucciones que no están propiamente relacionadas con un determinado módulo, pero son necesarias para delimitaciones de ejecución del programa objeto en el procesador del dispositivo.

Al conjunto de instrucciones mencionadas en el párrafo anterior se le denomina *programa fuente en lenguaje SIIL1*, asociado con la aplicación que se desea realice el PLM08. La forma sintáctica de las declaraciones asociadas caben en un renglón, y se ilustra a continuación:

CODM#N E1,…En,…S1,…Sm, D1,…Dq, CADBI;

Donde:

- CODM, es una cadena de caracteres que simboliza la función efectuada por el módulo.
- N, es el número asociado con el módulo, ya que todos los módulos lógicos de un mismo tipo de una aplicación deben ser enumerados.
- E1, a En son las designaciones de las n variables de entrada.
- S1, a Sm son las designaciones asociadas con las m salidas.

- D1, a Dq son datos auxiliares que pudieran ser requeridos por algunos módulos. Estos podrían ser entre otros; el tiempo asociado con la duración de un pulso generado por un temporizador o si un contador es ascendente o descendente. Hay módulos que no requieren de estas especificaciones, como las compuertas lógicas. Para los módulos que si requieren de estos datos, "q" es un número que está comprendido entre cero y tres.
- CADBI, es una cadena formada por unos y ceros, que específica diversas características de funcionamiento, como podrían ser: que entradas a una compuerta van a tener negación implícita, a qué tipo de flanco responde una entrada de algún otro tipo de modulo, etc.

Cabe señalar que el primer carácter de la instrucción nunca deberá estar en la primera columna y que al final de la misma siempre ha de colocarse el carácter ";". Todo texto a la derecha del carácter ";" no es tomado en cuenta por el software generador del código objeto, de esta manera el usuario podrá colocar comentarios en el programa fuente; si se desea tener un renglón completo como comentario, simplemente se coloca en la primera columna del mismo ya sea el carácter ";" o bien el carácter "*"

Para fines ilustrativos, a continuación se presenta la declaración en SIIL1 correspondiente a una compuerta AND de dos entradas que se desea sean las variables físicas E01 y E12; se requiere que las entradas no tengan preinversión y que la salida sea la variable intermediaria I02, además a esta compuerta se le designa el número 1; la forma sintáctica asociada es:

```
AND2#1 E01,E12,I02,11;
```

2.1.2 Secuencia de ejecución de un programa en lenguaje SIIL1.

Al correr un programa en SIIL1 en el procesador del PLM08, el código asociado con cada módulo lógico es ejecutado cíclicamente siguiendo la siguiente secuencia:

1. Se copian en buffer de entrada (BE) en RAM el estado que guardan los puertos asociados con las 16 variables físicas VBE.
2. Se ejecuta uno a uno el código asociado con cada uno de los ML que el usuario haya declarado en el programa fuente correspondiente, actualizándose un buffer de salida (BS).
3. Se copia el estado del BS en el puerto físico asociado con las VBS.
4. Se regresa al paso uno.

Existen módulos que requieren que el período de repetición de la ejecución de un código asociado sea constante (10ms), tal es el caso por ejemplo de los temporizadores, para hacer esto posible el código asociado es colocado en una rutina de servicio de interrupción que es invocada con una periodicidad de 10ms, empleándose para ello facilidades de temporización con que cuenta el microcontrolador *68HC908GP32.*

En consecuencia el código asociado con un programa en SIIL1 está dividido en dos partes, una de ellas es la que se ejecuta de acuerdo con los cuatro pasos descritos en el párrafo anterior, a esta parte se le llama subprograma principal (SPP), la otra parte está constituida por el código cuya ejecución es temporizada y se denomina subprograma temporizado (SPT).

2.1.3 Formato de un programa fuente en lenguaje SIIL1.

De acuerdo a lo explicado anteriormente, un programa fuente en SIIL1, deberá estar integrado por una serie de sentencias, varias de ellas serán declaraciones asociadas con los módulos lógicos que la aplicación requiera y otras serán simplemente delimitadores del código fuente de los subprograma principal y temporizado. En general el formato de un programa fuente en SIIL1 deberá presentar la siguiente forma:

INPROG; delimitador del inicio del SPP.

Sentencias asociadas con declaraciones de módulos que deben estar en el SPP

FINPP; delimitador de fin de SPP

INMODI; delimitador del inicio del SPT
Sentencias asociadas con declaraciones de módulos que deben estar en el SPT.

FINMODI; delimitador de fin del SPT.

En la figura 2.2 se muestra un sistema lógico integrado por dos módulos: una compuerta and de dos entradas cuya salida es la señal de disparo de un temporizador monodisparo (one shot) que genera un pulso verificado en bajo de dos minutos de duración; esto al presentarse una transición de bajo a alto en la entrada de disparo, las entradas a la compuerta son las variables binarias de

entrada E00 y E01; la salida binaria del PLM donde se genera el pulso es S00; nótese el uso de la variable intermediaria I00 como enlace entre la salida de la compuerta y la entrada del temporizador, además del empleo de la entrada binaria E03 como señal de restablecimiento del temporizador.

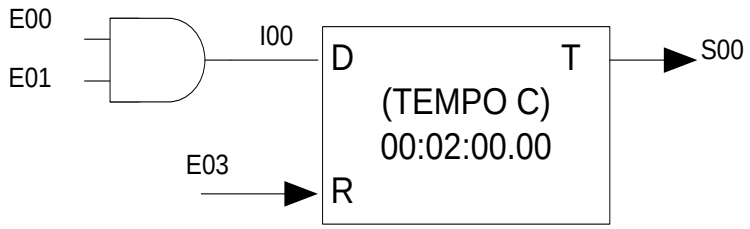


Figura 2.2 Sistema lógico disparo temporizador monodisparo (TEMPOC).

2.2 Descripción del módulo de seguidor lógico.

Este módulo simplemente pone la variable booleana (VB) declarada como salida en el nivel lógico que exista en la VB declarada como entrada al mismo, en la figura 2.3 se ilustra en forma genérica este módulo lógico (ML), debiendo el mismo ser declarado en el subprograma principal, siendo la sintaxis para declararlo lo siguiente:

SEG#nm TeEiiEj, TsSiiSj;

Donde:

nm, denota el número de seguidor, esto definido por el usuario en dos caracteres.

Te, podrá ser la letra “e”, “s” o “i” mayúscula o minúscula dependiendo esto de que la variable de entrada al seguidor sea una VBE, VBS o VBI.

Eii, denota el número de grupo que corresponda a la VB declarada como entrada al seguidor.

Ej, denota el número de bit dentro del grupo Eii, asociado a la variable de entrada al seguidor.

Ts, podrá ser la letra “e”, “s” o “i” mayúscula o minúscula dependiendo esto de que la variable de salida al seguidor sea una VBS o VBI.

Sii, denota el número de grupo que corresponda a la VB declarada como salida al seguidor.

Sj, denota el número de bit dentro del grupo Sii, asociado a la variable de salida al seguidor.

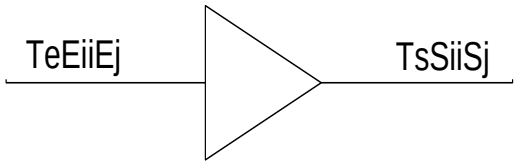


Figura 2.3 Representación genérica del seguidor lógico realizado por el PLM08.

A continuación se muestra un ejemplo sobre como declarar un módulo seguidor lógico, en un programa fuente en SIIL1.

Ejemplo 2.1

Se desea realizar con el PLM08 un seguidor lógico al que se le asigne el número 4, requiriéndose que la entrada y salida al mismo sean respectivamente las VB E03 e I24; la declaración sintáctica sería:

SEG#4 E03,I24;

2.3 Descripción del módulo de inversor lógico.

Este módulo simplemente pone la VB declarada como salida en el nivel lógico opuesto que exista en la VB declarada como entrada al mismo, en la figura 2.4 se ilustra en forma genérica este ML, debiendo el mismo ser declarado en el subprograma principal, siendo la sintaxis para declararlo lo siguiente:

NOT#nm TeEiiEj,TsSiiSj;

Donde:

nm, denota el número de inversor, esto definido por el usuario.

Te, podrá ser la letra “e”, “s” o “i” mayúscula o minúscula dependiendo esto de que la variable de entrada al inversor sea una VBE, VBS o VBI.

Eii, denota el número de grupo que corresponda a la VB declarada como entrada al inversor.

Ej, denota el número de bit dentro del grupo Eii, asociado a la variable de entrada al inversor.

Ts, podrá ser la letra “e”, “s” o “i” mayúscula o minúscula dependiendo esto de que la variable de salida al inversor sea una VBS o VBI.

Sii, denota el número de grupo que corresponda a la VB declarada como salida al inversor.

Sj, denota el número de bit dentro del grupo Sii, asociado a la variable de salida al inversor.

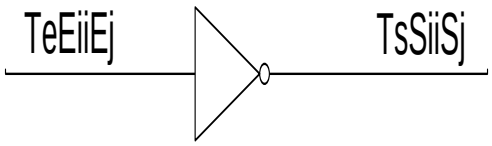


Figura 2.4 Representación genérica del inversor lógico realizado por el PLM08.

A continuación se muestra un ejemplo sobre como declarar un módulo inversor lógico, en un programa fuente en SIIL1.

Ejemplo 2.2

Se desea realizar con el PLM08 un inversor lógico al que se le asigne el número 7, requiriéndose que la entrada y salida al mismo sean respectivamente las VB E12 e I67; la declaración sintáctica sería:

NOT#7 E12,I67;

2.4 Descripción de los módulos lógicos que realizan compuertas lógicas de dos entradas.

El PLM08 puede realizar cuatro tipos de compuertas lógicas de dos entradas y estas son de tipo AND, OR, NAND, NOR, teniéndose además la capacidad de preinversión en las entradas que el usuario desee. En la figura 2.5 se ilustra en forma genérica este ML, debiendo el mismo ser declarado en el subprograma temporizado, siendo la sintaxis para declararlo la siguiente:

COMP#nm Te0E0iiE0j,Te1E1iiE1j,TsSiiSj,AB;

Donde:

COMP, es una cadena que puede ser AND2,OR2,NAND2,NOR2 esto de acuerdo al tipo de compuerta que se desee realizar.

nm, denota el número de compuerta, esto definido por el usuario, para cada uno de los cuatro tipos de compuertas posibles se ha de llevar una numeración independiente.

Te0, podrá ser la letra “e”,”s” o “i” mayúscula o minúscula dependiendo esto de que la variable de entrada E0 a la compuerta sea una VBE, VBS o VBI.

E0ii, denota el número de grupo que corresponda a la VB declarada como entrada E0 a la compuerta.

E0j, denota el número de bit dentro del grupo E0ii, asociado a la variable de entrada E0.

Te1, podrá ser la letra “e”,”s” o “i” mayúscula o minúscula dependiendo esto de que la variable de entrada E1 a la compuerta sea una VBE, VBS o VBI.

E1ii, denota el número de grupo que corresponda a la VB declarada como entrada E1 a la compuerta.

E1j, denota el número de bit dentro del grupo E1ii, asociado a la variable de entrada E1.

Ts podrá ser la letra ”s” o “i” mayúscula o minúscula dependiendo esto de que la variable de salida S de la compuerta sea una VBS o VBI.

Sii, denota el número de grupo que corresponda a la VB declarada como salida de la compuerta.

Sj, denota el número de bit dentro del grupo Sii asociado a la variable de salida de la compuerta.

A, es un dígito binario, que habrá de ser cero, si se desea que la entrada “E1” tenga preinversión, en otro caso el dígito “A” deberá ser uno.

B, es un digito binario, que habrá de ser cero, si se desea que la entrada “E0” tenga preinversión, en otro caso el dígito “B” deberá ser uno.

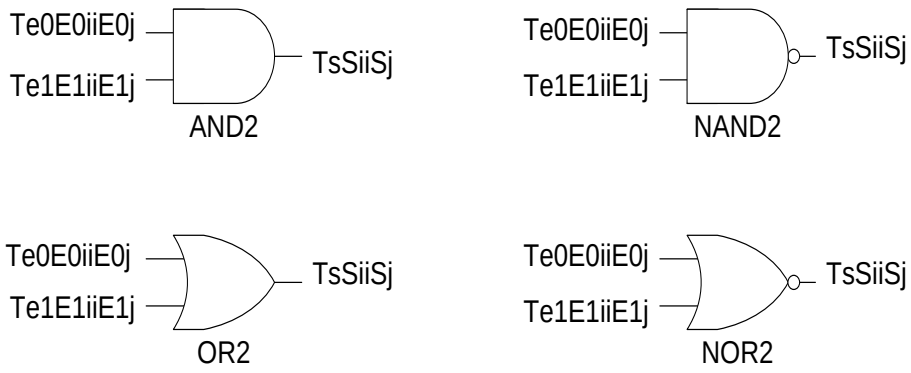


Figura 2.5 Representación genérica de las compuertas de dos entradas realizadas por el PLM08.

A continuación se muestra un ejemplo sobre como declarar un módulo que realiza una compuerta de dos entradas, en un programa fuente en SIIL1.

Ejemplo 2.3 Se desea realizar con el PLM08 una compuerta AND de dos entradas, para la cual se desea que las entradas E0 y E1 y la salida S sean respectivamente las VB E01,I24 y S13, requiriéndose que la entrada E0 tenga preinversión y que el número de asignación sea 4, véase la figura 2.6; en este caso se deberá usar la siguiente sintaxis:

```
AND2#4 E01,I24,S13,10;
```

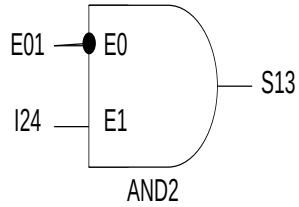


Figura 2.6 Ejemplo de compuerta AND de dos entradas con preinversión en la entrada E0.

2.5 Descripción de los módulos lógicos realizan compuertas lógicas de tres entradas.

El PLM08 puede realizar cuatro tipos de compuertas de tres entradas y estas son del tipo AND, NAND, OR, NOR teniéndose además la capacidad de preinversión en las entradas que el usuario desee. En la figura 2.7 se ilustra de forma genérica este módulo lógico, debiendo el mismo ser declarado en el subprograma principal, siendo la sintaxis para declararlo la siguiente:

COMP#nm Te0E0iiE0j,Te1E1iiE1j,Te2E2iiE2j, TsSiiSj,ABC;

Donde:

COMP, es una cadena que puede ser AND3, OR3, NAND, NOR3, esto de acuerdo al tipo de compuerta que se desee realizar.

nm, denota el número de compuerta, esto definido por el usuario, para cada uno de los cuatro tipos de compuertas posibles se ha de llevar una numeración independiente.

Te0 podrá ser la letra “e”,”s” o “i” mayúscula o minúscula dependiendo esto de que la variable de entrada E0 a la compuerta sea una VBE, VBS o VBI.

E0ii, denota el número de grupo que corresponda a la VB declarada como entrada E0 en la compuerta.

E0j, denota el número de bit dentro del grupo E0ii, asociado a la variable de entrada E0.

Te1, podrá ser la letra “e”,”s” o “i” mayúscula o minúscula dependiendo esto de que la variable de entrada E1 a la compuerta sea una VBE, VBS o VBI.

E1ii, denota el número de grupo que corresponda a la VB declarada como entrada E1 a la compuerta.

Ej1, denota el número de bit dentro del grupo E1ii, asociado con la variable de entrada E1.

Te2, podrá ser la letra “e”,”s” o “i” mayúscula o minúscula dependiendo esto de que la variable de entrada E2 a la compuerta sea una VBE, VBS o VBI.

E2ii, denota el número de grupo que corresponda a la VB declarada como entrada E2 a la compuerta.

E2j, denota el número de bit dentro del grupo E2ii, asociado con la variable de entrada E2.

Ts, podrá ser la letra “s” o “i” mayúscula o minúscula dependiendo esto de que la variable de salida “S” de la compuerta sea una VBS o VBI.

Sii, denota el número de grupo que corresponda a la VB declarada como salida de la compuerta.

Sj, denota el número de bit dentro del grupo Sii, asociado a la variable de salida de la compuerta.

A, es un dígito binario, que habrá de ser cero, si se desea que la entrada “E2” tenga preinversión, en otro caso el dígito “A” deberá ser uno.

B, es un dígito binario que habrá de ser cero, si se desea que la entrada “E1” tenga preinversión, en otro caso el dígito “B” deberá ser uno.

C, es un dígito binario, que habrá de ser cero si se desea que la entrada “E0” tenga preinversión, en otro caso el dígito “C” deberá ser uno.

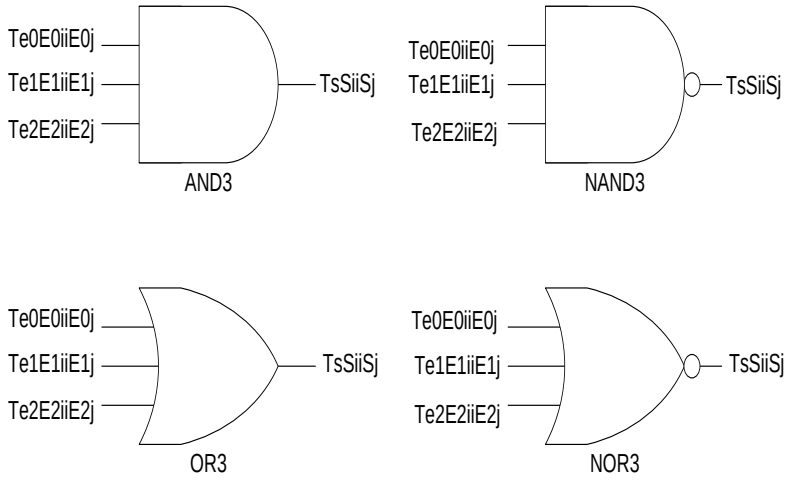


Figura. 2.7 Representación genérica de una compuerta de tres entradas realizada por el PLM08.

A continuación se muestra un ejemplo sobre como declarar una compuerta de tres entradas, en un programa SIIL1.

Ejemplo 2.4 Supóngase que se necesita realizar con el PLM08 una compuerta NOR de tres entradas, para la cual se desea que las entradas E0,E1 y E2 y la salida S sean respectivamente las VB E14, I03, E17 y S17, requiriéndose que la entrada E1 tenga preinversión y que el número de asignación sea 7, véase la figura 2.8; en este caso se deberá usar la sintaxis:

NOR3#7 E14,I03,E17,S17,101;

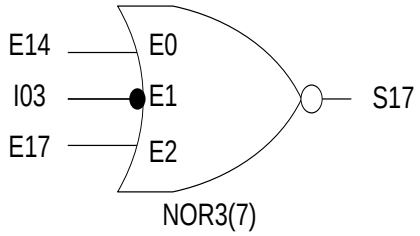


Figura 2.8 Compuerta NOR de tres entradas con preinversión en la entrada E1.

2.6 Descripción de los módulos lógicos realizan compuertas lógicas de cuatro entradas.

El PLM08 puede realizar cuatro tipos de compuertas de cuatro entradas y estas son del tipo AND, NAND, OR, NOR teniéndose además la capacidad de preinversión en las entradas que el usuario desee. En la figura 2.9 se ilustra de forma genérica este módulo lógico, debiendo el mismo ser declarado en el subprograma principal, siendo la sintaxis para declararlo la siguiente:

COMP#nm Te0E0iiE0j, Te1E1iiE1j, Te2E2iiE2j, Te3E3iiE3j, TsSiiSj,ABCD;

Donde:

COMP, es una cadena que puede ser AND4, OR4, NAND4, NOR4, esto de acuerdo al tipo de compuerta que se desee realizar.

nm, denota el número de compuerta, esto definido por el usuario, para cada uno de los cuatro tipos de compuertas posibles se ha de llevar una numeración independiente.

Te0, podrá ser la letra “e”,”s” o “i” mayúscula o minúscula dependiendo esto de que la variable de entrada E0 a la compuerta sea una VBE, VBS o VBI.

E0ii denota el número de grupo que corresponda a la VB declarada como entrada E0 en la compuerta.

E0j, denota el número de bit dentro del grupo E0ii, asociado a la variable de entrada E0.

Te1, podrá ser la letra “e”,”s” o “i” mayúscula o minúscula dependiendo esto de que la variable de entrada E1 a la compuerta sea una VBE, VBS o VBI.

E1ii, denota el número de grupo que corresponda a la VB declarada como entrada E1 a la compuerta.

E1j, denota el número de bit dentro del grupo E1ii, asociado con la variable de entrada E1.

Te2, podrá ser la letra “e”,”s” o “i” mayúscula o minúscula dependiendo esto de que la variable de entrada E2 a la compuerta sea una VBE, VBS o VBI.

E2ii, denota el número de grupo que corresponda a la VB declarada como entrada E2 a la compuerta.

E2j, denota el número de bit dentro del grupo E2ii, asociado con la variable de entrada E2.

Te3, podrá ser la letra “e”,”s” o “i” mayúscula o minúscula dependiendo esto de que la variable de entrada E3 a la compuerta sea una VBE, VBS o VBI.

E3ii, denota el número de grupo que corresponda a la VB declarada como entrada E3 a la compuerta.

E3j, denota el número de bit dentro del grupo E3ii, asociado con la variable de entrada E3.

Ts, podrá ser la letra “s” o “i” mayúscula o minúscula dependiendo esto de que la variable de salida “S” de la compuerta sea una VBS o VBI.

Sii, denota el número de grupo que corresponda a la VB declarada como salida de la compuerta.

Sj, denota el número de bit dentro del grupo Sii, asociado a la variable de salida de la compuerta.

A, es un dígito binario, que habrá de ser cero, si se desea que la entrada “E3” tenga preinversión, en otro caso el dígito “A” deberá ser uno.

B, es un dígito binario que habrá de ser cero, si se desea que la entrada “E2” tenga preinversión, en otro caso el dígito “B” deberá ser uno.

C, es un dígito binario, que habrá de ser cero si se desea que la entrada “E1” tenga preinversión, en otro caso el dígito “C” deberá ser uno.

D, es un dígito binario, que habrá de ser cero si se desea que la entrada “E0” tenga preinversión, en otro caso “D” deberá ser uno.

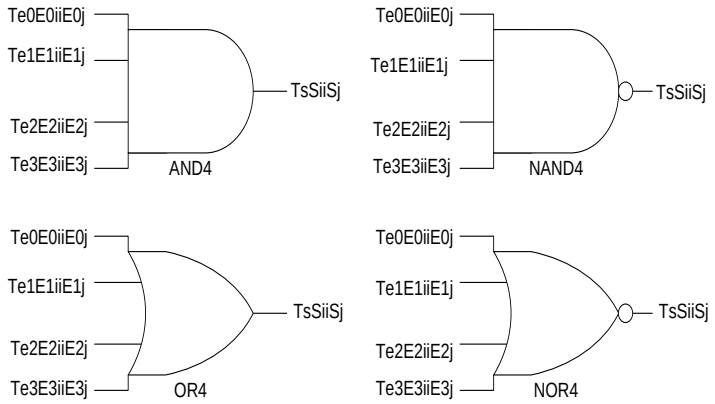


Figura. 2.9 Representación genérica de una compuerta de cuatro entradas realizada por el PLM08.

A continuación se muestra un ejemplo sobre como declarar una compuerta de cuatro entradas, en un programa SIIL1.

Ejemplo 2.5 Supóngase que se necesita realizar con el PLM08 una compuerta NAND de cuatro entradas, para la cual se desea que las entradas E0, E1, E2 y E3 y la salida S sean respectivamente las VB E12, E14, I16, E06 y S03, requiriéndose que la entrada E1 y E0 tengan preinversión y que el número de asignación sea 8, en este caso se deberá usar la sintaxis:

NAND4#8 E12,E14,I16,E06,S03,1100;

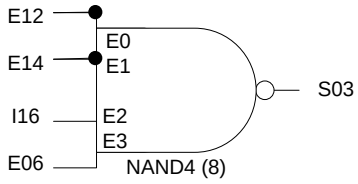


Figura 2.10 Ejemplo de compuerta NAND de cuatro entradas con preinversión en las entradas E0 y E1, la declaración sintáctica correspondiente es:

NAND4 #8 E12,E14,I16,E06,S03,1100;

2.7 Descripción del módulo lógico que realiza temporizadores monodisparo (TEMPOC).

De acuerdo con la señalización de entrada correspondiente, el PLM08 puede realizar dos tipos de temporizadores monodisparo, aquí se describe lo concerniente al temporizador monodisparo tipo dos, mostrándose respectivamente en las figuras 2.11 y figura 2.12 la representación como bloque de este módulo lógico y el diagrama de tiempos asociados con el mismo.

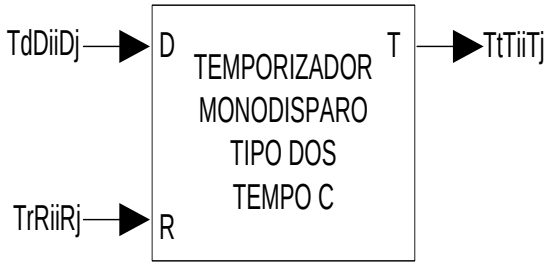


Figura 2.11 Representación genérica de un temporizador monodisparo de tipo dos.

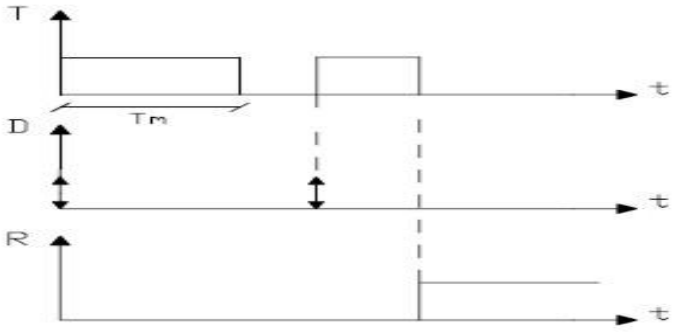


Figura 2.12 Diagrama de tiempos asociados con un temporizador monodisparo de tipo dos, (RESET verificado en alto.

El intervalo de tiempo correspondiente lo especifica el usuario con la declaración sintáctica correspondiente, pudiendo el mismo estar comprendido entre 10 ms y 47 horas con 22 minutos y 36.2 segundos, como se aprecia en la figura 3.12 el disparo puede ser por flanco de subida o flanco de bajada, teniéndose además otra entrada denominada “R” (RESET), al verificarse coloca a este módulo en su condición de espera de disparo, con su salida no verificada. Tiene capacidad de volver a ser disparado.

La entrada R corresponde al nivel, y al verificarse se desverifica la salida y se restablece a cero el contador de tiempo asociado.

Este módulo debe declararse en el subprograma temporizado, la sintaxis correspondiente es:

TEMPOC#nm TdDiiDj,TrRiiRj,TtTiiTj,HH:MM:SS.CS,ABC;

Donde:

nm, denota el número de temporizador, esto definido por el usuario.

Td, podrá ser la letra “e”,”s” o “i” mayúscula o minúscula dependiendo esto de que la variable de entrada D al temporizador sea una VBE, VBS o VBI:

Dii, denota el número de grupo que corresponda a la VB declarada como entrada “D” al temporizador.

Dj, denota el número de bit dentro del grupo Dii, asociado a la variable de entrada “D”.

Tr, podrá ser la letra “e”,”s” o “i” mayúscula o minúscula dependiendo esto de que la variable de restablecimiento “R” al temporizador sea una VBE, VBS o VBI.

Rii, denota el número de grupo que corresponda a la VB declarada como entrada “R” al temporizador.

Rj, denota el número de bit dentro del grupo Rii, asociado a la variable de entrada “R”.

Tt, podrá ser la letra “s” o “i” mayúscula o minúscula, dependiendo esto de que la variable de salida “T” del temporizador sea una VBS o VBI.

Tii, denota el número de grupo que corresponda a la VB declarada como salida del temporizador.

Tj, denota el número de bit dentro del grupo Tii, asociado a la variable de salida del temporizador.

HH, denota un par de dígitos que especifican el número de horas en el tiempo Tm.

MM, denota un par de dígitos que especifican el número de minutos en el tiempo Tm.

SS, denota un par de dígitos que especifican los segundos en Tm.

CS, denota un par de dígitos que especifican las centésimas de segundo en Tm.

A, es un dígito binario, que habrá de ser cero, si se desea que el temporizador se dispare para flancos de bajada en la entrada de disparo “D”, en otro caso el dígito “A” deberá ser uno.

B, es un dígito binario, que habrá de ser cero, si se desea que el temporizador sea restablecido por nivel alto, en otro caso el dígito “B” deberá ser uno.

C, es un dígito binario, que habrá de ser cero, si se desea que el nivel de verificación de la salida (T) sea bajo, en otro caso el dígito “C” deberá ser uno.

A continuación se muestra un ejemplo sobre como declarar un módulo temporizador monodisparo de tipo dos, en u programa fuente en SIIL1.

Ejemplo 2.6 Supóngase se desea realizar con el PLM08 un temporizador monodisparo de tipo dos, de manera que las entradas de disparo y restablecimiento sean respectivamente las entradas físicas E00 y E03, requiriéndose que la salida T sea la VBS S03, es necesario que el pulso de salida sea verificado en bajo y tenga una duración de treinta segundos, el disparo debe ser por flanco de bajada y el restablecimiento debe ser por nivel bajo; la declaración sintáctica correspondiente podría ser, suponiendo que se le asigna a este temporizador el número dos:

TEMPOC#2 E00,E03,S03,00:00:30,00,010;

En la figura 2.13 se muestra una representación como bloque de este temporizador.

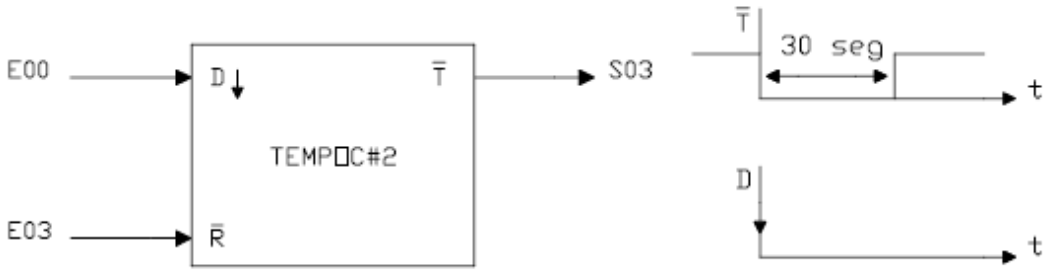


Figura 2.13 Representación como bloque de un temporizador tipo C

2.8 Descripción del módulo lógico que realiza temporizadores con retardo a la activación (On Delay) y con retardo a la desactivación (Off Delay) (TEMPOD).

El PLM08 puede realizar temporizadores con retardo a la activación (RA) o con retardo a la desactivación (RD), esto se logra a partir de un solo módulo lógico. En las figuras 2.14 y 2.15 se muestran respectivamente la representación como bloques de este módulo y los diagramas de tiempo asociados.

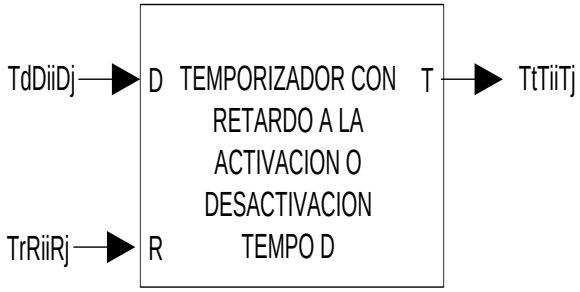


Figura 2.14. Representación genérica de un temporizador que puede operar con retardo a la activación o a la desactivación.

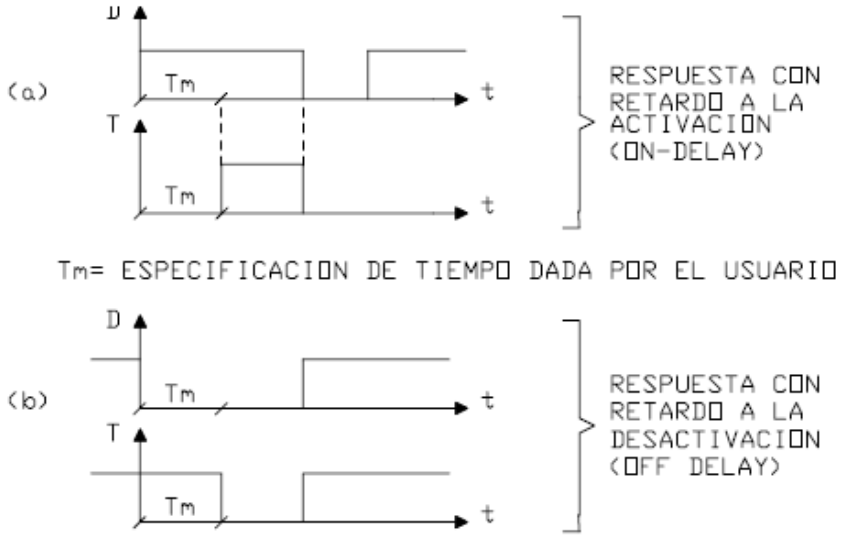


Figura 2.15. Diagrama de tiempos asociados con un temporizador con capacidad de retardo a la activación (a) o a la desactivación (b). Al verificarse la entrada de restablecimiento (R) la salida pasa a su nivel no verificado (cero para on-delay, uno para off-delay) inicializándose el contador descendente asociado.

El intervalo de tiempo correspondiente lo especifica el usuario en la declaración sintáctica correspondiente, pudiendo el mismo estar comprendido entre 10ms y 47 horas con 22 minutos y 36.2 segundos; la entrada R responde al nivel, y al verificarse se desverifica la salida y se inicializa el contador asociado.

Este módulo debe declararse en el subprograma temporizado, la sintaxis correspondiente es:

TEMPOD#nm TdDiiDj,TrRiiRj,TtTiiTj, HH:MM:SS.CS,AB;

Donde:

nm, denota el número de temporizador, esto definido por el usuario.

Td, podrá ser la letra “e”,”s” o “i” mayúscula o minúscula dependiendo esto de que la variable de entrada D al temporizador sea una VBE,VBS o VBI.

Dii, denota el número de grupo que corresponda a la VB declarada como entrada “D” al temporizador.

Dj, denota el número de bit dentro del grupo Dii, asociado a la variable de entrada “D”.

Tr, podrá ser la letra “e”, “s” o “i” mayúscula o minúscula, dependiendo esto de que la variable de entrada de restablecimiento (R) al temporizador sea una VBE,VBS o VBI.

Rii, denótale número de grupo que corresponda a la VB declarada como entrada “R” al temporizador.

Rj, denota el número de bit del grupo Rii, asociado a la variable de entrada “R”.

Tt, podrá ser la letra “s” o “i” mayúscula o minúscula, dependiendo esto de que la variable de salida “T” del temporizador sea una VBS o VBI.

Tii, denota el número de grupo que corresponda a la VB declarada como salida del temporizador.

Tj, denota el número de bit dentro del grupo Tii, asociado a la variable de salida del temporizador.

HH, denota un par de dígitos que especifican el número de horas en el tiempo Tm.

MM, denota un par de dígitos que especifican el número de minutos en el tiempo Tm.

SS, denota un par de dígitos que especifican los segundos en el tiempo Tm.

CS, denota un par de dígitos que especifican las centésimas de segundo en el tiempo Tm.

A, es un dígito binario, que habrá de ser cero, si se desea que el temporizador presente retardo a la desactivación (off-delay), en otro caso (A=1), el temporizador presentará retardo a la activación (on-delay).

B, es un dígito binario, que habrá de ser cero si se desea que el temporizador sea restablecido por nivel alto, en otro caso el dígito "B" deberá ser uno.

El siguiente ejemplo ilustra como declarar temporizadores con retardo a la activación y a la desactivación, en un programa fuente en SIIL1.

Ejemplo 2.7 Supóngase que se desea realizar con el PLM08 dos temporizadores, uno con retardo a la activación y el otro con retardo a la desactivación. Para el primero se requiere que el retardo a la activación sea de 7 segundos, y la entrada "R" sea verificada en bajo asignándosele el número 3, las entradas de disparo y restablecimiento han de ser las VB E02 y E03, la salida debe ser la VB S04.

Para el segundo temporizador se requiere que presente un retardo a la desactivación de 10 segundos, con restablecimiento en nivel bajo, asignándosele el número 4, las entradas de disparo y restablecimiento han de ser las VB E04 y E05, la salida debe ser la VB S05. En la figura 2.16 se muestran las representaciones como bloque de este ejemplo y sus diagramas de tiempo asociados.

La declaración correspondiente a estos temporizadores es la siguiente:

TEMPOD#3 E02,E03,S04,00:00:07,00,11;

TEMPOD#4 E04,E05,S05,00:00:10,00,01;

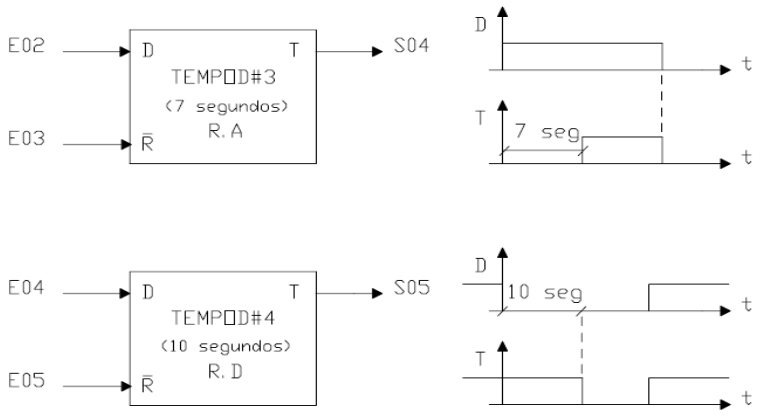


Figura 2.16 Diagramas de tiempo y representaciones como bloques, de los dos temporizadores on delay off delay.

2.9 Descripción del módulo lógico que realiza temporizadores astables (TEMPOE).

EL PLM08 puede realizar temporizadores astables que generan señales cuadradas con ciclo de trabajo que puede ser fijado por el usuario, en las figuras 2.17 y 2.18, se muestran respectivamente la representación como bloque de este módulo y los diagramas de tiempo asociados, el nivel de arranque puede ser uno o cero, esto definido por el usuario.

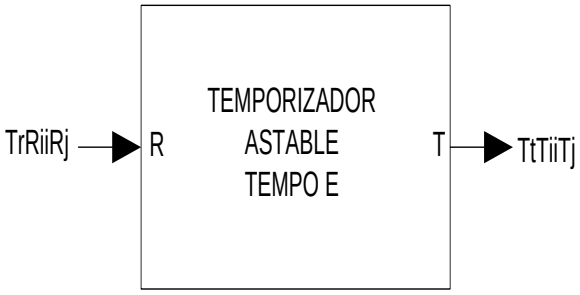
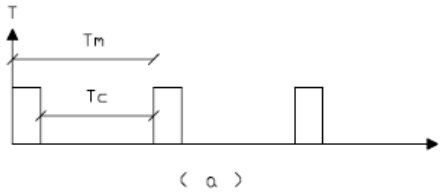


Figura 2.17. Representación genérica de un temporizador astable realizable con el PLM08.



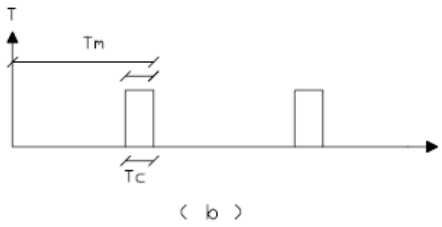


Figura 2.18 Diagramas de tiempo asociados con un temporizador astable con arranque en uno (a) y con arranque en cero (b).

Los tiempos Tc y Tm pueden estar comprendidos entre 10ms y 47 horas con 22 minutos y 36.2 segundos, debiendo siempre el tiempo Tc ser menor que el tiempo Tm; la entrada R responde al nivel, ya que al verificarse se coloca en la salida del nivel de arranque y se inicializa el contador asociado.

Este módulo debe declararse en el subprograma temporizado, la sintaxis correspondiente es:

TEMPOE#nm TrRiiRj,TtTiiTj,HHm:MMm:SSm.CSm, HHc:MMc:SSc.CSc,AB;

Donde:

nm, denota el número de temporizador, esto definido por el usuario.

Tr, podrá ser la letra “e”,”s” o “i” mayúscula o minúscula, dependiendo esto de que la variable de entrada de restablecimiento (R) al temporizador sea una VBE, VBS o VBI.

Rii, denota el número de grupo que corresponda a la VB declarada como entrada “R” al temporizador.

Rj, denota el número de bit dentro del grupo Rii, asociado a la variable de entrada “R”.

Tt, podrá ser la letra “s” o “i” mayúscula o minúscula, dependiendo esto de que la variable de salida “ T “ del temporizador sea una VBS o VBI.

Tii, denota el número de grupo que corresponda a la VB declarada como salida del temporizador.

Tj, denota el número de bit dentro del grupo Tii, asociado a la variable de salida del temporizador.

HHm, denota un par de dígitos que especifican el número de horas en el tiempo Tm.

MMm, denota un par de dígitos que especifican el número de minutos en el tiempo Tm.

SSm, denota un par de dígitos que especifican el número de segundos en el tiempo Tm.

CSm, denota un par de dígitos que especifican las centésimas de segundo en el tiempo Tm.

HHc, denota un par de dígitos que especifican el número de horas en el tiempo Tc.

MMc, denota un par de dígitos que especifican el número de minutos en el tiempo Tc.

SSc, denota un par de dígitos que especifican el número de segundos en el tiempo Tc.

CSc, denota un par de dígitos que especifican las centésimas de segundo en el tiempo Tc.

A, es un dígito binario, que habrá de ser cero, si se desea que el temporizador se restablezca por nivel alto, en otro caso A deberá ser igual a uno para que el temporizador se restablezca por nivel bajo.

B, es un dígito binario, que habrá de ser cero, si se desea que el temporizador arranque en cero, en otro caso contrario este dígito deberá ser uno.

El siguiente ejemplo muestra como declarar temporizadores astables con arranque en uno y en cero, en un programa fuente SIIL1.

Ejemplo 2.8 Supóngase que se desea realizar con el PLM08 dos temporizadores astables, uno con arranque en uno y el otro con arranque en cero. Para el primero se requiere que los tiempos Tm y Tc sean respectivamente 2s y 250ms, el nivel de restablecimiento ha de ser bajo y la VB asociada debe ser E06, la salida debe ser la VB S06, asignándosele a este temporizador el número cinco.

Para el segundo temporizador de este ejemplo, que requiere que los tiempos Tm y Tc sean respectivamente 1s y 300ms, el nivel de restablecimiento ha de ser bajo siendo E01 la VB asociada, la salida debe ser la VB S07, asignando a este temporizador el número 6.

La declaración para estos temporizadores es la siguiente:

TEMPOE#5 E06,S06,00:00:02.00,00:00:00.25,11;

TEMPOE#6 E01,S07,00:00:01.00,00:00:00.30,10;

En la figura 2.19 se muestran las representaciones como bloques de los temporizadores de este ejemplo y sus diagramas de tiempo asociados.

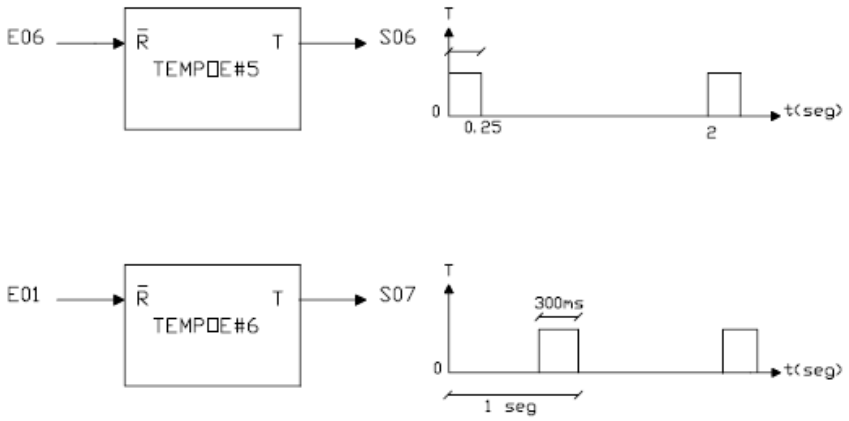


Figura 2.19 Diagramas de tiempo y representaciones como bloques, de los temporizadores del ejemplo 2.8.

2.10 Descripción del módulo lógico que realiza flip-flops asíncronos (FFARS).

EL PLM08 puede realizar módulos tipo latch, que en la nomenclatura del mismo se denominan como flip-flops asíncronos R-S (FFARS), teniéndose para este módulo lógico, la capacidad de predefinir el nivel de verificación de las entradas “S” y “R”, y además de poder predefinir el nivel que ha de tener la salida cuando ambas entradas se verifican y el valor que se desea tome la misma al iniciar el programa SIIL1 que ejecuta el PLM08 en un momento dado, en la figura 2.20 se ilustra en forma genérica este módulo lógico, debiendo el mismo ser declarado en el subprograma principal, siendo la sintaxis para declararlo la siguiente:

FFARS#nm TsSiiSj,TrRiiRj,TqQiiQj,ABCD;

Donde:

nm, representa en dos caracteres el número que el usuario asigno a este módulo.

Ts, podrá ser la letra “e”, “s” o “i” mayúscula o minúscula dependiendo esto de que la variable de entrada SET (S) al flip-flop sea una VBE, VBS o VBI.

Sii, denota el número de grupo que corresponda a la VB declarada como entrada “S” al flip-flop.

Sj, denota el número de bit del grupo Sii, asociado a la variable de entrada “S”.

Tr, podrá ser la letra “e”,”s”o “i” mayúscula o minúscula dependiendo esto de que la variable de entrada RESET (R) al flip-flop sea una VBE, VBS o VBI.

Rii, denota el número de grupo que corresponda a la VB declarada como entrada “R” al flip-flop.

Rj, denota el número de bit dentro del grupo Rii, asociado a la variable de entrada “R”.

Tq, podrá ser la letra “s” o “i” mayúscula o minúscula dependiendo esto de que la variable de salida “Q” del flip-flop sea una VBS o VBI.

Qii, denota el número de grupo que corresponda a la VB declarada como salida del flip-flop.

Qj, denota el número de bit dentro del grupo Qii, asociado a la variable de salida del flip-flop.

A, es un dígito binario, que habrá de ser cero, si se desea que el nivel de verificación de la entrada “S” sea bajo, en otro caso el dígito “A” deberá ser uno.

B, es un dígito binario que habrá de ser cero, si se desea que el nivel de verificación de la entrada "R" sea bajo, en otro caso el dígito "B" deberá ser uno.

C, es un dígito binario, que habrá de ser uno si se desea que la VB de entrada SET tenga prioridad, en otro caso (prioridad para la VB de entrada RESET), "C" deberá ser cero. El hecho de que la entrada SET tenga prioridad implica que si ambas entradas SET y RESET se verifican simultáneamente la salida Q será uno lógico, por otro lado, prioridad para la entrada RESET significa que al verificarse ambas entradas del flip-flop la salida Q será puesta en cero lógico.

D, es un dígito binario, que habrá de ser cero, si se desea que la salida Q se inicialice en cero lógico, en otro caso "D" deberá ser uno.

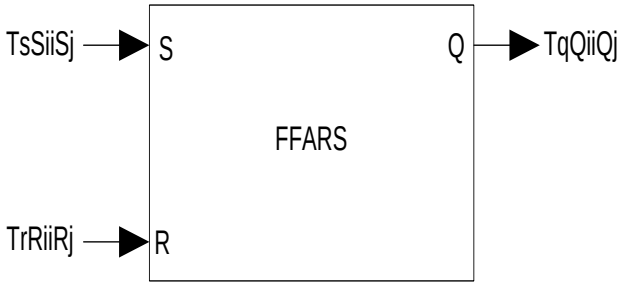


Figura. 2.20 Representación genérica del módulo lógico que realiza el flip-flop asíncrono R-S.

A continuación se muestra un ejemplo sobre como declarar un módulo que realice un módulo tipo latch , en un programa fuente SIIL1.

Ejemplo 2.9 Supóngase que se desea realizar con el PLM08 un módulo tipo latch, para el cual se desea que la entrada S tenga prioridad, deseándose que las entradas S,R y la salida Q sean respectivamente las VB E13,E14 y S06, requiriéndose que ambas entradas S y R tengan verificación en bajo y que el estado inicial de la salida Q sea uno, además que a este latch se le asigne el número 22, vease la figura 2.21 , en este caso la sintaxis es :

FFARS#22 E13,E14,S06,0011;

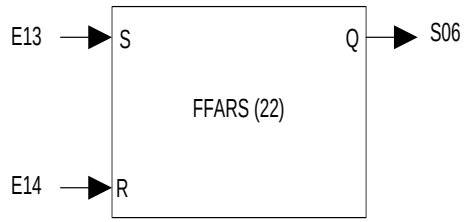


Figura 2.21. Ejemplo de FFARS realizado con el PLM08.

2.11 Descripción del módulo lógico que realiza contadores de eventos.

EL PLM08 puede realizar módulos contadores de eventos ascendentes o descendentes, siendo los valores de la cuenta comprendidos en un determinado intervalo, pudiendo el usuario definir tanto la cuenta inicial (CUENTA I) como la cuenta final (CUENTA F), este módulo lógico tiene tres entradas y una salida, véase la figura 2.22 , las entradas son: entrada "D" sensible a flancos que hacen que se modifique la cuenta, entrada de restablecimiento (RESET) que al verificarse hace que la cuenta retorne a su valor inicial desverificándose la salida (TF), y entrada de congelamiento que al verificarse hace que el contador conserve la cuenta sin responder a los niveles lógicos presentes en las otras dos entradas; la salida de este módulo lógico (TF) se verifica cuando la cuenta ha llegado a su valor final.

Para este módulo lógico se tiene la capacidad de predefinir los límites del intervalo de cuenta, el tipo de flanco que incrementa o decrementa la cuenta, el nivel de verificación de las entradas de RESET y congelamiento, el nivel de verificación de la salida testigo de cuenta final y el tipo de cuenta (ascendente o descendente) a efectuar.

Este módulo debe declararse en el subprograma temporizado, siendo la sintaxis para declararlo la siguiente:

CONTA#nm TdDiiDj,TcCiiCj,TrRiiRj,TfFiiFj,ctaInic,ctaFin,ABCDE;

Donde:

nm, denota el número de contador de eventos, definido por el usuario.

Td, podrá ser la letra “e”,”s” o “i” mayúscula o minúscula dependiendo de que la variable de entrada D al contador sea una VBE, VBS o VBI.

Dii, denota el número de grupo que corresponda a la VB declarada como entrada “D” al contador.

Dj, denota el número de bit dentro del grupo Dii, asociado a la variable de entrada “D”.

Tc, podrá ser la letra “e”,”s” o “i” mayúscula o minúscula dependiendo de que la variable de entrada de congelamiento “C” al contador sea una VBE, VBS o VBI.

Cii, denota el número de grupo que corresponda a la VB declarada como entrada “C” al contador.

Cj, denota el número de bit dentro del grupo Cii, asociado a la variable de entrada “C”.

Tr, podrá ser la letra “e”,”s” o “i” mayúscula o minúscula dependiendo de que la variable de entrada RESET (R) al contador sea una VBE, VBS o VBI.

Rii, denota el número de grupo que corresponda a la VB declarada como entrada “R” al contador.

Rj, denota el número de bit dentro del grupo Rii, asociado a la variable de entrada “R”.

Tf, podrá ser la letra “s” o “i” mayúscula o minúscula dependiendo de que la variable de salida “TF” al contador sea una VBS o VBI.

Fii,denota el número de grupo que corresponda a la VB declarada como salida del contador.

Fj, denota el número de bit dentro del grupo Fii asociado a la variable de salida del contador.

ctaInic, denota el valor de la cuenta inicial, debiendo el mismo estar comprendido entre 0 y 65535, debiendo este valor ser menor que el correspondiente a la cuenta final, si el contador es ascendente, en otro caso el valor declarado para la cuenta inicial deberá ser mayor que la cuenta final.

ctaFin, denota el valor de la cuenta final y está comprendido entre 0 y 65535.

A, es un dígito binario, que habrá de ser cero, si se desea que se modifique la cuenta para flancos de bajada en la entrada de disparo “D”, en otro caso el dígito “A” deberá ser uno.

B, es un dígito binario, que habrá de ser cero, si se desea que el nivel de verificación de la entrada “C” sea bajo, en otro caso el dígito “B” deberá ser uno.

C, es un dígito binario que habrá de ser uno si se desea que el nivel de verificación de la entrada “R” sea bajo, en otro caso “C” deberá ser cero.

D, es un dígito binario, que habrá de ser uno, si se desea que la cuenta sea ascendente, en otro caso el dígito “D” deberá ser cero.

E, es un dígito binario, que habrá de ser cero, si se desea que el nivel de verificación de la salida “TF” sea bajo, en otro caso el dígito “E” deberá ser uno.

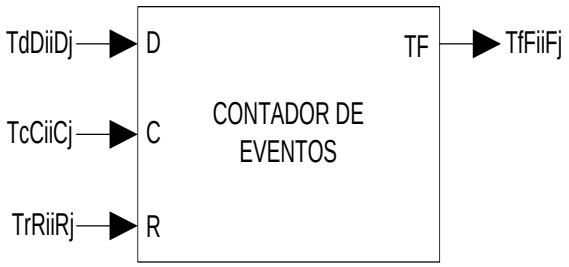


Figura 2.22. Representación genérica de un módulo contador de eventos realizable con el PLM08.

A continuación se muestra un ejemplo sobre como declarar un módulo que realice un módulo tipo contador de eventos, en un programa fuente SIIL1.

Ejemplo 2.10 Supóngase que se necesita realizar con el PLM08 un módulo contador de eventos ascendente, con un intervalo de cuenta comprendido entre cero y siete y testificación de cuenta en nivel bajo, se requiere que los niveles de verificación de las entradas “R” y “C” sean en alto y que la entrada de disparo “D” sea sensible a flancos de bajada, se desea que las entradas D,C,R y la salida TF sean respectivamente las VB E17,I14,E12 y S01, además que este contador tenga el número 14, véase la figura 2.23, por lo cual la sintaxis correspondiente es:

```
CONTA#14 E17,I14,E12,S01,0,7,01010;
```

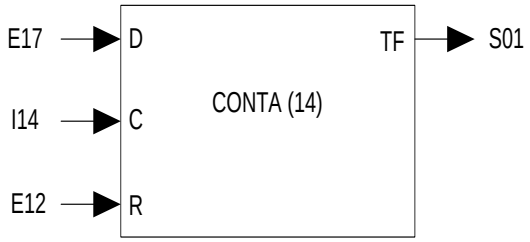


Figura 2.23 Ejemplo de módulo contador de eventos ascendente realizado con el PLM08 descrito anteriormente.

En la figura 2.24 se ilustran los diagramas de tiempo asociados con el contador de eventos aquí ejemplificado. Cabe señalar que para la correcta operación del contador de eventos, se requiere que el intervalo de tiempos entre dos flancos consecutivos sea mayor a 10ms.

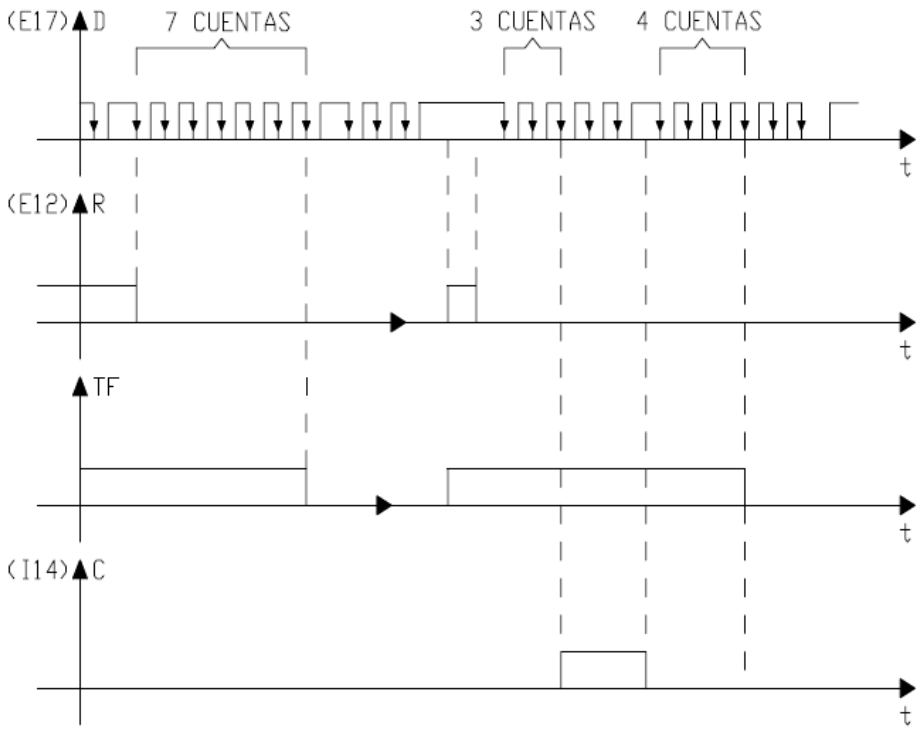


Figura 2.24. Diagramas de tiempo asociados con el temporizador mostrado en la figura 2.23.

Referencias:

- Salvá Calleja Antonio – “Programador Lógico Modular”- México, D.F .Tesis de Maestría, División de Estudios de Posgrado, Facultad de Ingeniería, UNAM, febrero de 1999.

- Altamirano Yépez Luis Antonio, Dehesa Castillejos Erick Abraham, Hernandez Reyes Maricarmen –“Desarrollo de software de simulación para el PLM (Programador lógico modular). Tesis de licenciatura, Facultad de Ingeniería, UNAM, 2003.

- Salvá Calleja Antonio, Sánchez Esquivel Victor Manuel, Salcedo Ubilla María Leonor, Ramírez Gutierrez José Luis – “Manual de usuario del PLM2”. Controlador lógico programable para auxilio didáctico. Facultad de Ingeniería, UNAM, 2006.