Capítulo 3

3. IMPLEMENTACIÓN DE LOS MÓDULOS LÓGICOS.

3. Implementación de los módulos lógicos.

3.1 Descripción general.

En este capítulo se detalla el flujo de ejecución asociado con la validación de cada uno de los módulos lógicos realizables por el PLM08. Se muestra para cada caso el programa genérico en lenguaje ensamblador; donde aparecerán cadenas mudas que habrán de sustituirse con lo requerido por cada módulo, con características individuales que el usuario haya declarado en el programa en SIIL1 asociado.

Para cada caso se muestra un ejemplo ilustrativo de asignación específica a cada una de las cadenas mudas presentes en el esqueleto.

Para hacer un programa que pueda ejecutarse en el microcontrolador, es necesario que esté se apegue a las normas de programación en ensamblador. Para tal efecto es necesario un esqueleto de programación, es decir, un programa genérico en el caso de cada módulo lógico; que es tomado como plantilla para sustituir la información definida por el usuario y que a la vez mantenga la estructura básica de la programación para la correcta ejecución del programa.

A continuación se muestra de manera general los diagramas de flujo y el programa esqueleto genérico para cada uno de los módulos realizables por el PLM08.

3.2 Programa esqueleto general en lenguaje ensamblador asociado con un programa en lenguaje SIIL1.

En la sección 3.1.3 del capítulo 3 se describió la forma genérica que presenta un programa en SIIL1. A continuación se muestra el programa en ensamblador asociado ya que en este capítulo se hará referencia a componentes de este, al ejemplificar con casos particulares la asignación a cadenas mudas requerida por cada modulo realizable por el PLM08.

El programa genérico en cuestión es el siguiente:

```
** Encabezado común
locaux#1 equ $4c
tesppspt equ $4d
tesppspp equ $4e
tesentb equ $4f
gpe0 equ $50
gpe1 equ $52
gps0 equ $54
```

```
gpi0 equ $56
gpi1 equ $58
gpi2 equ $5a
gpi3 equ $5c
gpi4 equ $5e
gpi5 equ $60
gpi6 equ $62
gpi7 equ $64
gpi8 equ $66
gpi9 equ $68
dirbtempos equ $6a
ddrc equ $06
ddrd equ $07
pta equ $00
ptb equ $01
ptc equ $02
ptd equ $03
t1sc equ $20
t1modh equ $23
config1 equ $1f
modtemp equ $4e1f
ini#sp          equ $023F

**** INPROG ******
                org $8000
                ldhx #ini#sp+1
                txs
                bset 0,config1
                clrh
                ldx #locaux#1
otrocl:             clr ,x
                incx
                cpx #gpi9+2
                bne otrocl

                mov #$03,ddrc ;ptc0,ptc1 y ptc2 son salidas.
                mov #$3e,ddrd ;ptd1 a ptd5 son salidas
```

```
; Inicializa parámetros de temporizador para que se de una interrupción
; por sobreflujo cada 10 ms.

                lda #$40
                sta t1sc ;tstop<--0,toie<--1
                ldhx #modtemp
                sthx t1modh
                cli
ciclopp:            lda pta
                sta gpe0
          lda ptb
                sta gpe1
**** Fin de código asociado con sentebcia INPROG *****

**** Código de módulos de usuario  colocados en el subprograma principal
**** ..... ****
**** Fin de código de módulos de usuario colocados en el subprograma principal

**** Código asociado con sentencia FINPP

                lda gps0
                sta ptc
                clc
                rola
                rola
                sta ptd

                mov #$ff,tesppspp
                jmp ciclopp
*** Fin de código asociado con sentencia FINPP ****

**** Código asociado con sentencia INMODI   *****
servovf:            pshh
                lda t1sc
                bclr 7,t1sc ;TOF<--0
**** Fin de código asociado con sentencia INMODI *****

*** Código asociado con módulos de usuario colocados en el subprograma temporizado
*** .....***
```

*** Fin de código de módulos del SPT

**** Código asociado con sentencia FINMODI

```
                clrh
                ldx gpe0
guardates:          lda ,x
                sta $01,x
                incx
                incx
                cpx dirbtempos
                bne guardates
                mov #$ff,tesppspt
                pulh
                rti
```

*** Fin de código asociado con sentencia FINMODI ****

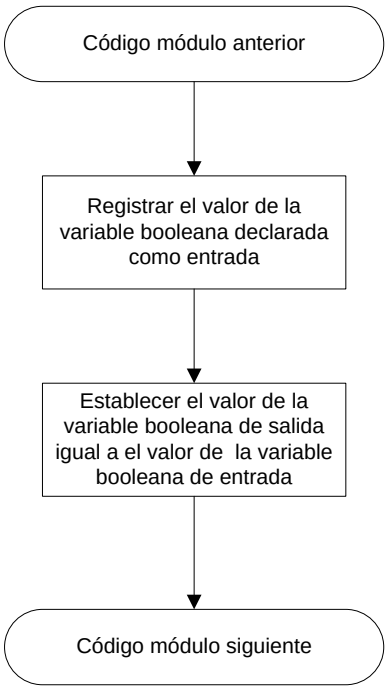
**** Colocación de vectores de usuario de reset y TOF ***

```
                org $d7f2
                dw servovf

                org $d7fe
                dw $8000
```

3.2.1 Flujo de ejecución y programa esqueleto genérico para el módulo seguidor.

Para el módulo lógico seguidor, el flujo de ejecución es el siguiente:



Para el módulo seguidor el programa ensamblador genérico tiene la siguiente forma:

```
                brclr bitent,grupobitent,seg#00nm
                brset bitent,grupobitent,seg#10nm
seg#00nm:       bclr bitsal,grupobitsal
                bra seg#20nm
seg#10nm:       bset bitsal,grupobitsal
seg#20nm:       nop
```

Las cadenas mudas para este esqueleto son: **bitent**, **grupobitent**, **bitsal**, **grupobitsal** y **nm**.

A continuación se muestra, en un ejemplo específico la asignación a las cadenas mudas. Supóngase que se tiene la siguiente sentencia SIIL1 asociada con un seguidor lógico:

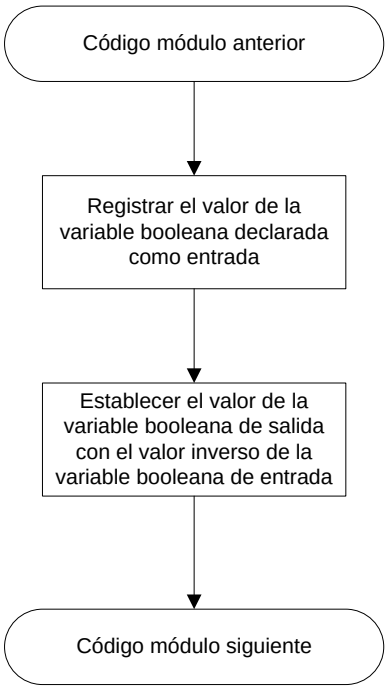
SEG#1 E12,S01;

Como se ve; se trata de un seguidor lógico cuya entrada es el bit 2 del grupo 1 de entradas y cuya salida es el bit 1 del grupo 0 de salidas. De acuerdo con el programa general mostrado en la sección 3.2 es fácil ver que la asignación a las cadenas mudas debe ser la siguiente:

bitent= 2
bitsal=1
grupobitent=gpe1
grupobitsal=gps0
nm =01

3.2.2 Flujo de ejecución y programa esqueleto genérico para el módulo inversor.

Para el módulo lógico inversor, el flujo de ejecución es el siguiente:



Para el módulo inversor el programa ensamblador genérico tiene la siguiente forma:

```
                brclr bitent,grupobitent,inv#00nm
                brset bitent,grupobitent,inv#10nm
inv#00nm:       bset bitsal,grupobitsal
                bra inv#20nm
inv#10nm:       bclr bitsal,grupobitsal
inv#20nm:       nop
```

Las cadenas mudas para este esqueleto son: **bitent**, **grupobitent**, **bitsal**, **grupobitsal** y **nm**.

A continuación se muestra, en un ejemplo específico la asignación de cadenas mudas.

Supóngase que se tiene la siguiente sentencia SIIL1 asociada con un inversor lógico.

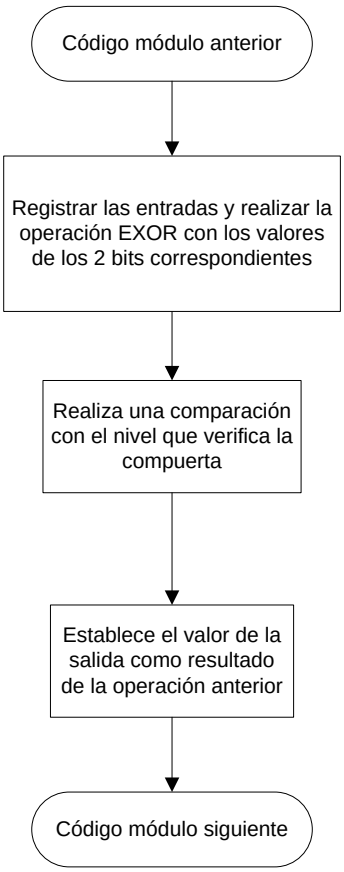
NOT#2 E13,S02;

Como se ve; se trata de un inversor lógico cuya entrada es el bit 3 del grupo 1 de entradas y cuya salida es el bit 2 del grupo 0 de salidas. De acuerdo con el programa general mostrado en la sección 3.2 es fácil ver que la asignación de cadenas mudas debe ser la siguiente:

bitent = 3
bitsal = 2
grupobitent = gpe1
grupobitsal = gps0
nm = 02

3.2.3 Flujo de ejecución y programa esqueleto genérico de los módulos de compuertas lógicas AND, OR, NAND y NOR de dos entradas realizables con el PLM08.

Para el módulo de una compuerta lógica de dos entradas, el flujo de ejecución es el siguiente:



Para el caso de módulo para una compuerta lógica AND2 el programa ensamblador genérico tiene la siguiente forma:

```
                clr tesentb
                brclr bitent1,grupobitent1,and2#00nm
                bset 0,tesentb
and2#00nm:      brclr bitent2,grupobitent2,and2#10nm
```

```
                    bset 1,tesentb
and2#10nm:      lda tesentb
                    eor #$XX
                    cmp #$YY
                    beq and2#20nm
                    bclr bitsal,grupobitsal
                    bra and2#30nm
and2#20nm:      bset bitsal,grupobitsal
and2#30nm:      nop
```

Las cadenas mudas para este esqueleto son: **bitent1**,**grupobitent1**,**bitent2**,**grupobitent2**, **bitsal**, **grupobitsal**, **XX** y **nm**.

A continuación se muestra, en un ejemplo específico la asignación de cadenas mudas.

Supóngase que se tiene la siguiente sentencia SIIL1 asociada con una compuerta AND de dos entradas:

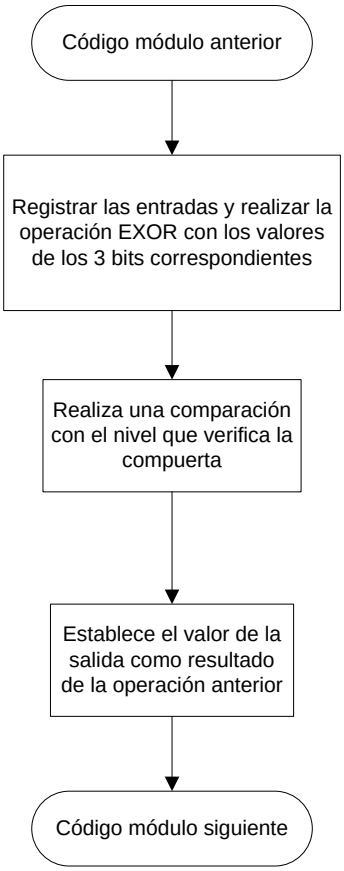
AND2#1 E00,E13,S01,10;

Como se ve; se trata de un compuerta lógica AND de 2 entradas, cuya primera entrada es el bit 0 del grupo 0 de entradas; su segunda entrada es el bit 3 del grupo 1 de entradas y cuya salida es el bit 1 del grupo 0 de salidas. De acuerdo con el programa general mostrado en la sección 3.2 es fácil ver que la asignación de cadenas mudas debe ser la siguiente:

bitent1 = 0
bitent2 = 3
bitsal = 1
grupobitent1 = gpe0
grupobitent2 = gpe1
grupobitsal = gps0
nm = 01
XX = 01

3.2.4 Flujo de ejecución y programa esqueleto genérico de los módulos de compuertas lógicas AND, OR, NAND y NOR de tres entradas realizables con el PLM08.

Para el módulo de una compuerta lógica de tres entradas, el flujo de ejecución es el siguiente:



Para el caso de una compuerta lógica 0R3 el programa ensamblador genérico es de la siguiente forma:

```
                clr tesentb
                brclr bitent1,grupobitent1,or3#00nm
                bset 0,tesentb
or3#00nm:       brclr bitent2,grupobitent2,or3#10nm
                bset 1,tesentb
or3#10nm:       brclr bitent3,grupobitent3,or3#20nm
```

```
                bset 2,tesentb
or3#20nm:       lda tesentb
                eor #$XX
                cmp #$YY
                beq or3#30nm
                bset bitsal,grupobitsal
                bra or3#40nm
or3#30nm:       bclr bitsal,grupobitsal
or3#40nm:       nop
```

Las cadenas mudas para este esqueleto son: **bitent1**,**grupobitent1**, **bitent2**, **grupobitent2**, **bitent3**, **grupobitent3, bitsal**, **grupobitsal, XX** y **nm**.

A continuación se muestra, en un ejemplo específico la asignación de cadenas mudas.

Supóngase que se tiene la siguiente sentencia SIIL1 asociada con una compuerta 0R de tres entradas:

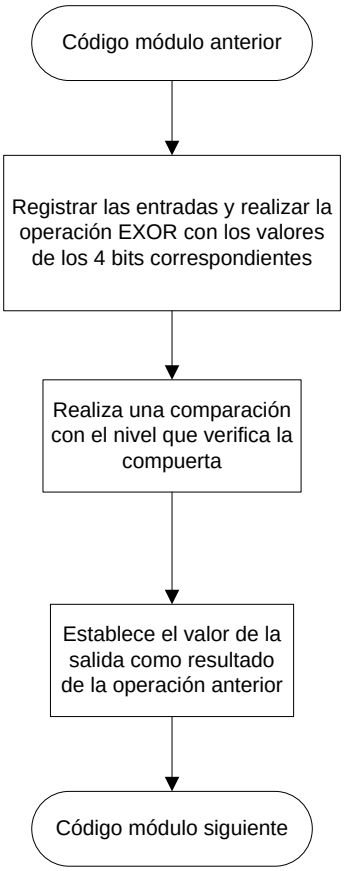
OR3#1 E07,I06,I15,S04,011;

Como se ve; se trata de un compuerta lógica 0R de 3 entradas, cuya primera entrada es el bit 7 del grupo 0 de entradas; su segunda entrada es el bit 6 del grupo 0 de variables intermedias; su tercera entrada es el bit 5 del grupo 1 de variables intermedias y cuya salida es el bit 4 del grupo 0 de salidas. De acuerdo con el programa general mostrado en la sección 3.2 es fácil ver que la asignación de cadenas mudas debe ser la siguiente:

bitent1 = 7
bitent2 = 6
bitent3 = 5
bitsal = 4
grupobitent1 = gpe0
grupobitent2 = gpi0
grupobitent3 = gpi1
grupobitsal = gps0
nm = 01
XX = 04

3.2.5 Flujo de ejecución y programa esqueleto genérico de los módulos de compuertas lógicas AND, OR, NAND y NOR de cuatro entradas realizables con el PLM08.

Para el módulo lógico de una compuerta lógica de cuatro entradas, el flujo de ejecución es el siguiente:



Para el caso de una compuerta lógica NOR4 el programa ensamblador genérico es de la siguiente forma:

```
                clr tesentb
                brclr bitent1,grupobitent1,nor4#00nm
                bset 0,tesentb
nor4#00nm:      brclr bitent2,grupobitent2,nor4#10nm
                bset 1,tesentb
nor4#10nm:      brclr bitent3,grupobitent3,nor4#20nm
                bset 2,tesentb
nor4#20nm:      brclr bitent4,grupobitent4,nor4#30nm
                bset 3,tesentb
nor4#30nm:      lda tesentb
```

```
                eor #$XX
                cmp #$YY
                beq nor4#40nm
                bclr bitsal,grupobitsal
                bra nor4#50nm
nor4#40nm:      bset bitsal,grupobitsal
nor4#50nm:      nop
```

Las cadenas mudas para este esqueleto son: **bitent1**,**grupobitent1**, **bitent2**, **grupobitent2**, **bitent3**, **grupobitent3**, **bitent4**, **grupobitent4**, **bitsal**, **grupobitsal, XX** y **nm**.

A continuación se muestra, en un ejemplo específico la asignación de cadenas mudas.

Supóngase que se tiene la siguiente sentencia SIIL1 asociada con una compuerta NOR de cuatro entradas:

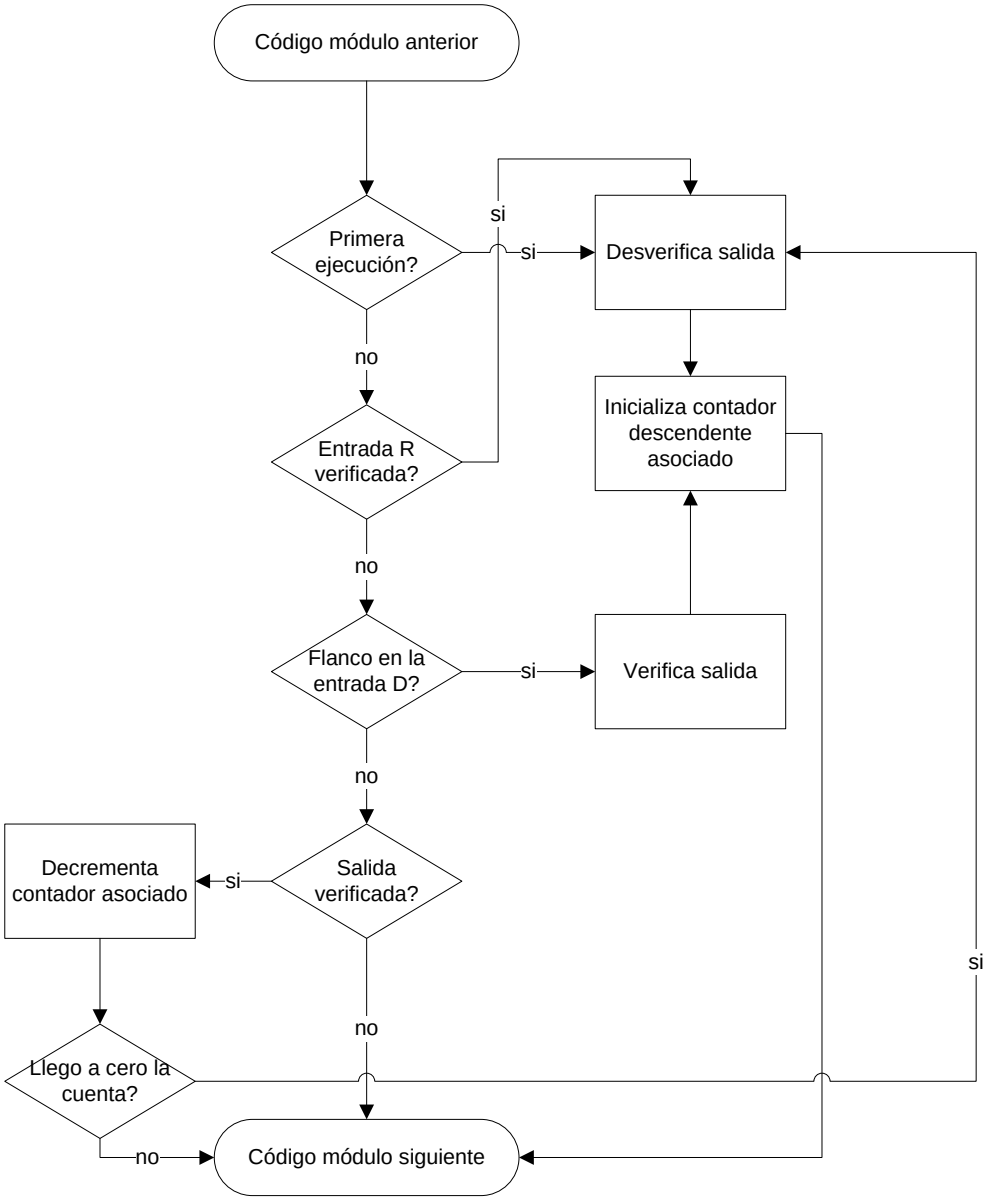
NOR4#3 E11,E03,I25,I34,S07,0111;

Como se ve; se trata de un compuerta lógica N0R de 4 entradas, cuya primera entrada es el bit 1 del grupo 1 de entradas; su segunda entrada es el bit 3 del grupo 0 de entradas; su tercera entrada es el bit 5 del grupo 2 de variables intermedias, su cuarta entrada es el bit 4 del grupo 3 de variables intermedias y cuya salida es el bit 7 del grupo 0 de salidas. De acuerdo con el programa general mostrado en la sección 3.2 es fácil ver que la asignación de cadenas mudas debe ser la siguiente:

bitent1 = 1
bitent2 = 3
bitent3 = 5
bitent4 = 4
bitsal = 7
grupobitent1 = gpe1
grupobitent2 = gpe0
grupobitent3 = gpi2
grupobitent4 = gpi3
grupobitsal = gps0
nm = 03
XX = 08

3.2.6 Flujo de ejecución y programa esqueleto genérico del módulo lógico que realiza temporizadores monodisparo (TEMPOC).

Para el módulo lógico TEMPOC, el flujo de ejecución es el siguiente:



Para el módulo lógico TEMPOC el programa ensamblador genérico tiene la siguiente forma:

```
                lda tesppspt
                bne npptc#nm
resettc#nm:     bclr bitsal,grupobitsal
inicontc#nm:    mov #$FF,dirbtempos+nn
                ldhx #$FFFF
                sthx dirbtempos+nnn
                bra salidatc#nm
npptc#nm:       brclr bitrst,grupobitrst,resettc#nm
                lda grupoedis+1
                and #$XX
                sta locaux#1
                lda grupoedis
                and #$YY
                cmp locaux#1
                beq siguetc3#0
                bhi versaltc#0
siguetc3#nm:    brclr bitsal,grupobitsal,salidatc#nm
                ldhx dirbtempos+nnn
                aix #$ff
                sthx dirbtempos+nnn
                cphx #$0000
                bne salidatc#nm
                lda dirbtempos+nn
                beq resettc#nm
                dec dirbtempos+nn
salidatc#nm:    bra sigblotc#nm
versaltc#nm:    bset bitsal,grupobitsal
                bra inicontc#nm
sigblotc#nm:    nop
```

Las cadenas mudas para este esqueleto son: **bitsal**, **grupobitsal**, **nn**, **nm, nnn**, **FFFF**,**FF**, **XX**, **YY**, **bitrst**, **grupobitrst** y **grupoedis**.

A continuación se muestra, en un ejemplo específico la asignación de cadenas mudas.

Supóngase que se tiene la siguiente sentencia SIIL1 asociada con un temporizador one shot (TEMPOC).

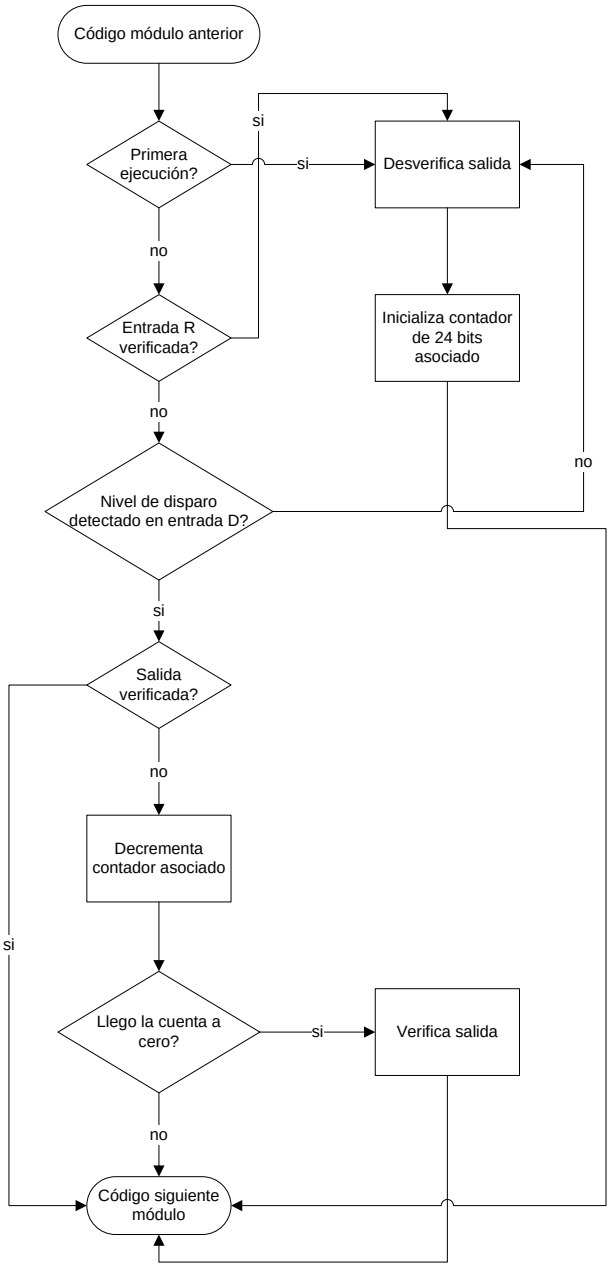
TEMPOC#2 E07,E06,S01,00:00:10.00,111;

Como se ve; se trata de un temporizador One shot, cuyo bit de disparo es el bit 7 del grupo 0 de entradas, su bit de reset es el bit 6 del grupo 0 de entradas; su salida es el bit 1 del grupo 0 de salidas. La duración del pulso temporizado es de 10 segundos. Se dispara por flanco de subida, se reestablece por nivel bajo y su nivel de verificación es 1 lógico. De acuerdo con el programa general mostrado en la sección 3.2 se deduce que la asignación de cadenas mudas debe ser la siguiente:

bitsal = 1
bitrst = 6
grupobitsal = gps0
grupobitrst = gpe0
grupoedis = Grupo del bit de disparo = gpe0
nn = 3 Enmascaramiento de bits (número de temporizador -1) *3
nnn = 4 Enmascaramiento de bits (nn +1)
FFFF = 03E8 Tiempo de 10 s.
FF = 00
XX = 80 Enmascaramiento del bit de disparo E07.
YY = 80 Enmascaramiento del bit de disparo E07.

3.2.7 Flujo de ejecución y programa esqueleto genérico del módulo lógico que realiza temporizadores con retardo a la activación (On Delay) y con retardo a la desactivación. (Off Delay) (TEMPOD).

Para el módulo lógico TEMPOD, el flujo de ejecución es el siguiente:



Para el módulo lógico TEMPOD el programa ensamblador genérico tiene la siguiente forma:

```
                lda tesppspt
                bne npptc#nm
resettc#nm:     bclr bitsal,grupobitsal
inicontc#nm:    mov #$FF,dirbtempos+nn
                ldhx #$FFFF
                sthx dirbtempos+nnn
                bra salidatc#nm
npptc#nm:       brclr bitdsp,grupobitdsp,resettc#nm
                brclr bitrest,grupobitrest,resettc#nm
                brset bitsal,grupobitsal,salidatc#nm
                ldhx dirbtempos+nnn
                aix #$FF
                sthx dirbtempos+nnn
                ldhx dirbtempos+nnn
                cphx #$FFFF
                beq versaltc#nm
                bne salidatc#nm
                dec dirbtempos+nn
                ldhx dirbtempos+nnn
                cphx #$0000
                bne salidatc#nm
                lda dirbtempos+nn
                beq versaltc#nm
salidatc#nm:    bra sigblotc#nm
versaltc#nm:    bset bitsal,grupobitsal
sigblotc#nm:    nop
```

Las cadenas mudas para este esqueleto son: **bitsal**, **grupobitsal**, **nn**, **nnn**,**FF**, **FFFF**, **bitdsp**, **grupobitdsp**, **bitrest**, **grupobitrest,nm**.

A continuación se muestra, en un ejemplo específico la asignación de cadenas mudas.

Supóngase que se tiene la siguiente sentencia SIIL1 asociada con un temporizador del tipo retardo a la activación ó retardo a la desactivación.

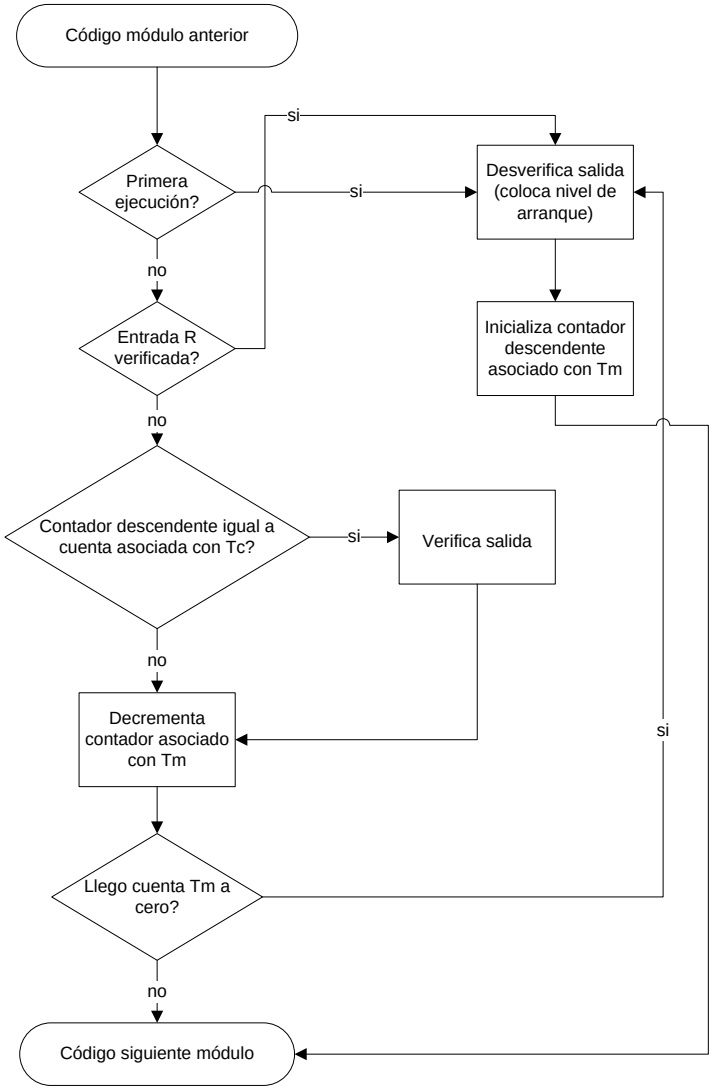
TEMPOD#3 E17,E06,S02,00:00:10.00,01;

Como se ve; se trata de un temporizador con retardo a la desactivación, cuyo bit de disparo es el bit 7 del grupo 1 de entradas, su bit de restablecimiento es el bit 6 del grupo 0 de entradas; su salida es el bit 2 del grupo 0 de salidas. La duración del pulso temporizado es de 10 segundos. Se restablece por nivel alto. De acuerdo con el programa general mostrado en la sección 3.2 se deduce que la asignación de cadenas mudas debe ser la siguiente:

bitsal = 2
bitrest = 6
bitdsp = 7
grupobitsal = gps0
grupobitrest = gpe0
grupobitdsp = gpe1
nn = 6 Enmascaramiento de bits (número de temporizador -1) *3
nnn = 7 Enmascaramiento de bits (nn +1)
FFFF = 03E8 Tiempo de 10 s.

3.2.8 Flujo de ejecución e implementación del módulo lógico que realiza temporizadores astables (TEMPO E).

Para el módulo lógico TEMPOE, el flujo de ejecución es el siguiente:



Para el módulo lógico que realiza temporizadores astables (TEMPOE) el programa ensamblador genérico tiene la siguiente forma:

```
                lda tesppspt
                bne npptc#nm
resettc#nm:     bclr bitsal,grupobitsal
etiqueta nm:    bset bitsal,grupobitsal
inicontc#nm:    mov #$FF,dirbtempos+nn
```

```
                    ldhx #$FFFF
                    sthx dirbtempos+nnn
                    bra salidatc#nm
npptc#nm:           brclr bitrest,grupobitrest,resettc#nm
                    lda dirbtempos+nn
                    cmp #$00
                    bne decrementa#nm
                    ldhx dirbtempos+nnn
                    cphx #$ffff
                    bne decrementa#nm
                    bclr bitsal,grupobitsal
decrementa#nm: ldhx dirbtempos+nnn
                    aix #$FF
                    sthx dirbtempos+nnn
                    ldhx dirbtempos+nnn
                    cphx #$AAAA
                    beq etiquetanm
                    bne salidatc#nm
                    dec dirbtempos+nn
                    ldhx dirbtempos+nnn
                    cphx #$0000
                    bne salidatc#nm
                    lda dirbtempos+nn
                    beq etiquetanm
salidatc#nm:        bra sigblotc#nm
sigblotc#nm:        nop
```

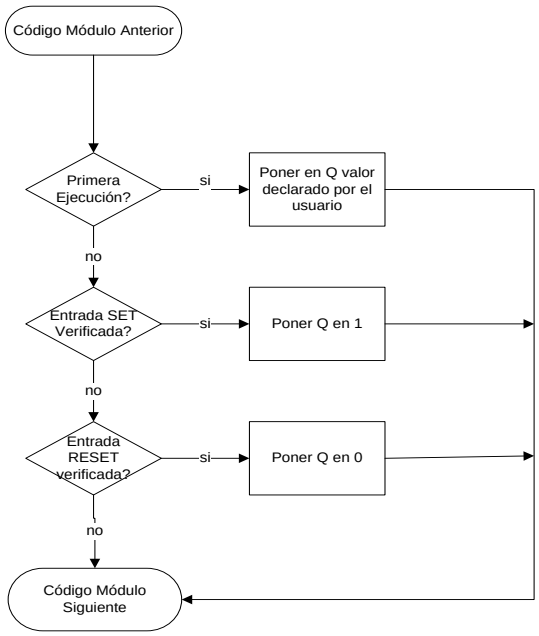
Las cadenas mudas para este esqueleto son: **bitsal**, **grupobitsal**, **nn**, **nnn**,**FF**, **FFFF**, **AAAA**, **bitrest**, **grupobitrest**,**nm**.

A continuación se muestra, en un ejemplo específico la asignación de cadenas mudas.
Supóngase que se tiene la siguiente sentencia SIIL1 asociada con un temporizador del tipo astable.
TEMPOE#5 E07,S00,00:00:00.02,00:00:00.01,10;
Como se ve; se trata de un temporizador astable, cuyo bit de restablecimiento es el bit 7 del grupo 0 de entradas y su bit de salida es el bit 0 del grupo 0 de salidas. La duración del pulso Tm es 10s y la duración del pulso Tc es 7 s. Se restablece por nivel bajo y arranca en nivel cero. De acuerdo con el programa general mostrado en la sección 3.2 se deduce que la asignación de cadenas mudas debe ser la siguiente:
bitsal = 0

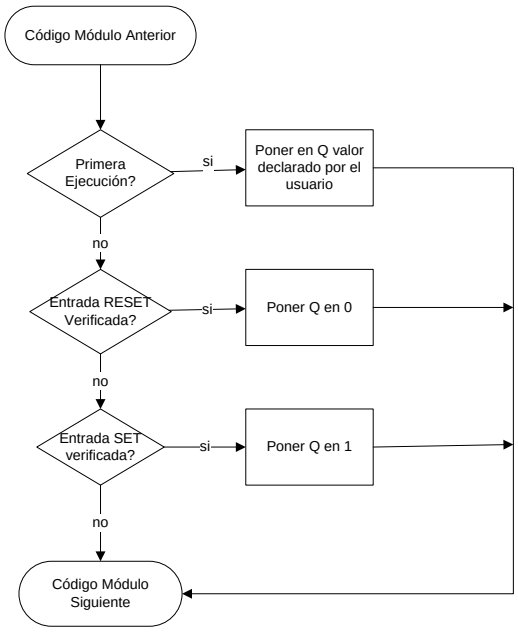
bitrest = 7
grupobitsal = gps0
grupobitrest = gpe0
nn = 12 Enmascaramiento de bits (número de temporizador -1) *3
nnn = 13 Enmascaramiento de bits (nn +1)
FFFF = 03E8 Tiempo de 10 s (Tm).
AAAA = 02BC Tiempo de 7 s (Tc).

3.2.9 Flujo de ejecución e implementación del módulo lógico que realiza flip-flops asíncronos R-S (FFARS).

Para el módulo lógico FFARS, el flujo de ejecución es el siguiente:



Flujo de ejecución del tramo de código empleado para realizar un módulo tipo latch (FFARS), cuando la entrada S tiene prioridad.



Flujo de ejecución del tramo de código empleado para realizar un módulo tipo latch (FFARS), cuando la entrada R tiene prioridad.

Para el módulo lógico FFARS el programa ensamblador genérico tiene la siguiente forma:

```
                lda tesppspt
                bne npptc#nm
                bset bitQ,grupobitQ
                bra salidatc#nm
npptc#nm:       brset bitS,grupobitS,pon1sal#nm
                brset bitR,grupobitR,pon0sal#nm
                bra salidatc#nm
pon1sal#nm:     bset bitsal,grupobitsal
                bra salidatc#nm
pon0sal#nm:     bset bitsal,grupobitsal
                bra salidatc#nm
salidatc#nm:    nop
```

Las cadenas mudas para este esqueleto son: **bitQ**, **grupobitQ**, **bitS**, **grupobitS**, **bitR**, **grupobitR**, **bitsal**, **grupobitsal**.

A continuación se muestra, en un ejemplo específico la asignación de cadenas mudas.

Supóngase que se tiene la siguiente sentencia SIIL1 asociada con un módulo que realiza un flip-flop asíncrono R-S.

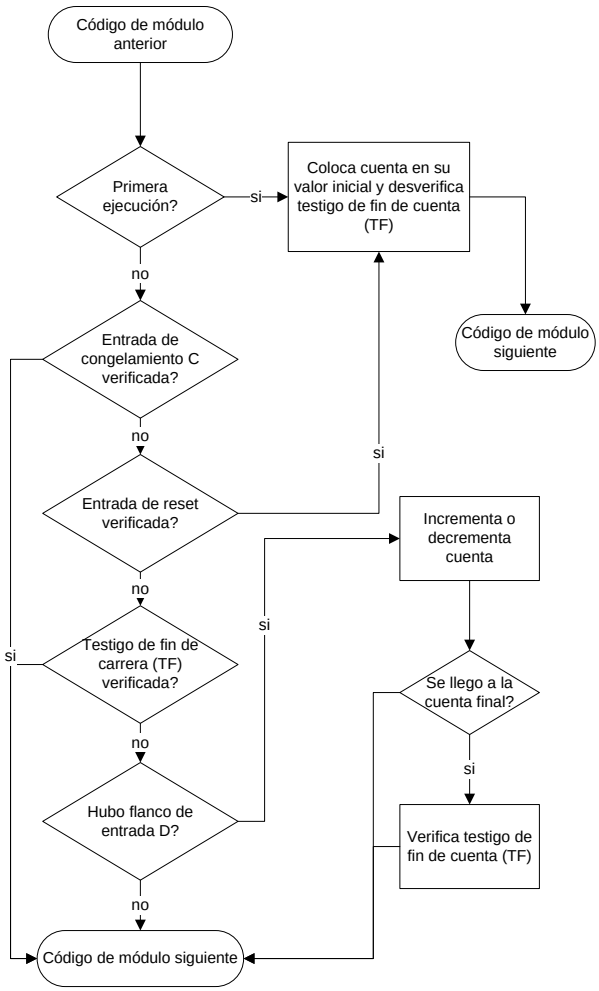
FFARS#22 E007,E006,S005,0001;

Como se ve; se trata de flip-flop R-S, cuyo bit de entrada S es el bit 7 del grupo 0 de entradas, su bit de reset R es el bit 6 del grupo 0 de entradas y su bit de salida Q es el bit 5 del grupo 0 de salidas. El nivel de verificación de la entradas R y S son bajos; se tiene prioridad para la variable booleanas de entrada RESET y la salida se inicializa en nivel 1 lógico. De acuerdo con el programa general mostrado en la sección 3.2 se deduce que la asignación de cadenas mudas debe ser la siguiente:

bitQ = 5
bitR = 6
bitS = 7
bitsal = 5
grupobitQ = gps0
grupobitR = gpe0
grupobitS = gpe0
grupobitsal = gps0

3.2.10 Flujo de ejecución e implementación del módulo lógico que realiza contadores de eventos.

Para el módulo lógico CONTADOR DE EVENTOS, el flujo de ejecución es el siguiente:



Para el módulo lógico CONTADOR DE EVENTOS el programa ensamblador genérico tiene la siguiente forma:

```
                lda tesppspt
                bne npptc#nm
resettc#nm:     bclr bitsal,grupobitsal
inicontc#nm:    ldhx #$FFFF
                sthx dirbconts+nn
                ldhx #$AAAA
                sthx dirbconts+nnn
                bra salidatc#nm
npptc#nm:       brset bitent,grupobitent,salidatc#nm
                brclr bitrst,grupobitrst,resettc#nm
                brset bitsal,grupobitsal,salidatc#nm
                lda grupoedis+1
                and #$40
                sta locaux#1
                lda grupoedis
                and #$40
                cmp locaux#1
                beq salidatc#nm
                bhi siguetc3#nm
                bra salidatc#nm
siguetc3#nm:    ldhx dirbconts+nn
                aix #$FF
                sthx dirbconts+nn
                ldhx dirbconts+nn
                cphx dirbconts+nnn
                beq versaltc#nm
salidatc#nm:    bra sigblotc#nm
versaltc#nm:    bset bitsal,grupobitsal
                bra inicontc#nm
sigblotc#nm:    nop
```

Las cadenas mudas para este esqueleto son: **bitsal**, **grupobitsal**, **FFFF**, **AAAA**, **nn**, bhi, **nnn**, **bitent**, **grupobitent**, **bitrst**,**nm**,**grupobitrst** y **grupoedis**.

A continuación se muestra, en un ejemplo específico la asignación de cadenas mudas.

Supóngase que se tiene la siguiente sentencia SIIL1 asociada con un módulo que realiza un contador de eventos.

CONTA#14 E016,E015,E014,S007,00010,00005,00001;

Como se ve; se trata de contador de eventos, cuyo bit de entrada D es el bit 6 del grupo 1 de entradas, para su entrada de congelamiento C es el bit 5 del grupo 1 de entradas, su bit de reset R es el bit 4 del grupo 1 de entradas y su salida es el bit 7 del grupo de salidas 0.

Sus características: se modifica la cuenta de bajada en la entrada de disparo D, el nivel de verificación de la entrada C es bajo, el nivel de verificación de la entrada R es alto, es un contador descendente y el nivel de verificación de la salida es alto. Su cuenta comienza en 10 y termina en 5 .De acuerdo con el programa general mostrado en la sección 3.2 se deduce que la asignación de cadenas mudas debe ser la siguiente:

bitent = 6
bitrst = 4
bitsal = 7
grupobitent = gpe1
grupobitrst = gpe1
grupobitsal = gps0
nn = 39 Enmascaramiento de bits (número de temporizador -1) *3
nnn = 41 Enmascaramiento de bits (nn +2)
FFFF = 000A Cuenta inicial.
AAAA = 0005 Cuenta final.
grupoedis = Grupo del bit de congelamiento = gpe1
Nota: El bit de congelamiento se encuentra enmascarado en el detector de salto de flanco.

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.