# Capítulo 5

5. EJEMPLO DE APLICACIÓN.

5.1 Descripción general del problema de aplicación.
5.2 Diseño del programa que controla la banda transportadora.
5.3 Programa en lenguaje SIIL1 que da solución al problema de la banda transportadora.
5.4 Generación del programa en lenguaje ensamblador usando el software GEN_ENS_PLM08.
5.5 Ejecución del programa en lenguaje ensamblador usando el software PUMMA_08+.
5.5.1 Inicialización del sistema.
5.5.2 Carga y ejecución del programa que resuelve el ejemplo de la banda transportadora.

## 5.1 Descripción general del problema de aplicación.

Para este capítulo, se presenta un ejemplo que utiliza la mayoría de módulos lógicos realizables por el PLM08 para resolver una aplicación real. El problema a resolver es controlar una banda transportadora que debe detenerse cada determinado tiempo para una parte de su proceso industrial.

Con motivos de ejemplificar esto se describe a continuación el problema a resolver. Se tiene una banda transportadora con un contenedor que debe detenerse en tres ocasiones para atender diversas etapas de un proceso. En la primera estación debe detenerse por 5 segundos; en la segunda estación debe esperar que un despachador le suministre 5 objetos y cumplido esto, esperar por 5 segundos más antes de ir a la tercera estación. En esta última estación debe esperar por cinco segundos más. Todo el proceso debe comenzar de nuevo cuando se alimenta la banda transportadora con un nuevo contenedor.

A continuación se muestra en la figura 5.1, la distribución de las estaciones de proceso en la banda transportadora a controlar mediante el PLM08.

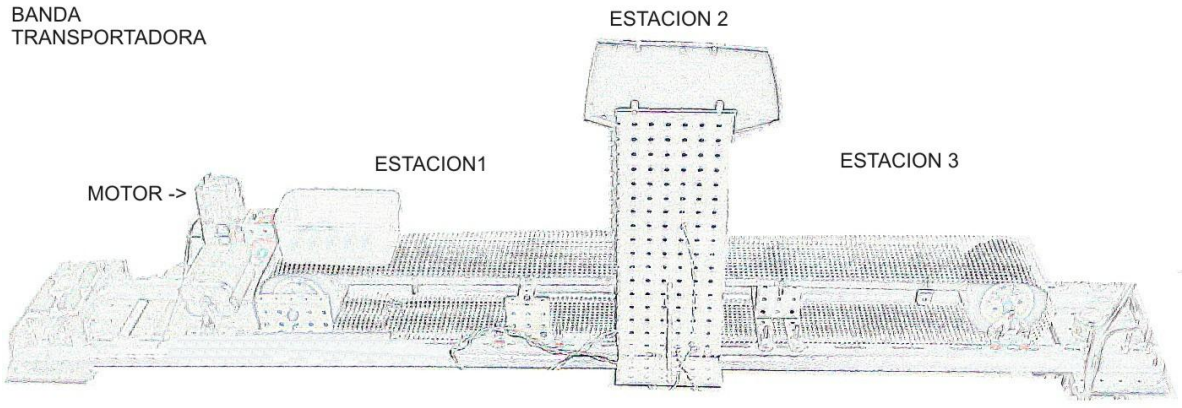


Figura 5.1 Imagen de la banda transportadora y sus distintas etapas.

## 5.2 Diseño del programa que controla la banda transportadora.

Como se mencionó en la sección 1.6 del capítulo 1, “Diseño de hardware”, el PLM08 cuenta con 16 entradas optoacopladas y 8 salidas a relevadores que son las variables booleanas de entrada y salida, respectivamente. También se cuenta con variables booleanas intermedias, estas variables sirven como conexión interna entre los módulos lógicos realizables por el PLM08.

Entonces se tiene que el problema básico es controlar el motor que mueve la banda, y hacer tres estaciones para los procesos que nos pide atender el problema; estas estaciones están temporizadas para durar un cierto tiempo definido por el usuario o esperar el resultado de un proceso.

La forma de diseñar el programa es definiendo qué se busca en cada etapa, para así luego resolverlo.

| ETAPAS DEL PROGRAMA | QUÉ SE BUSCA |
|---|---|
| ESTACIÓN 1 | El motor debe hacer parar la banda en la estación 1 durante 5 segundos. |
| ESTACIÓN 2 | El motor debe esperar que 5 objetos sean depositados en la bandeja de carga; y una vez hecho esto esperar por 5 segundos más antes de continuar a la siguiente estación. |
| ESTACION 3 | El motor debe esperar 5 segundos en esta estación y se debe dar restablecimiento al contador de la estación 2 para hacer el proceso cíclico. |

Como se puede apreciar, el principal objetivo del programa es controlar el motor. Por lo mismo; está claro que nuestra variable de salida más importante es la que dispara el motor. Para el ejemplo esta es la variable booleana de salida S02.

A continuación se muestra en la figura 6.2 la solución esquemática al control de la banda transportadora, usando compuertas y módulos lógicos realizables por el PLM08.

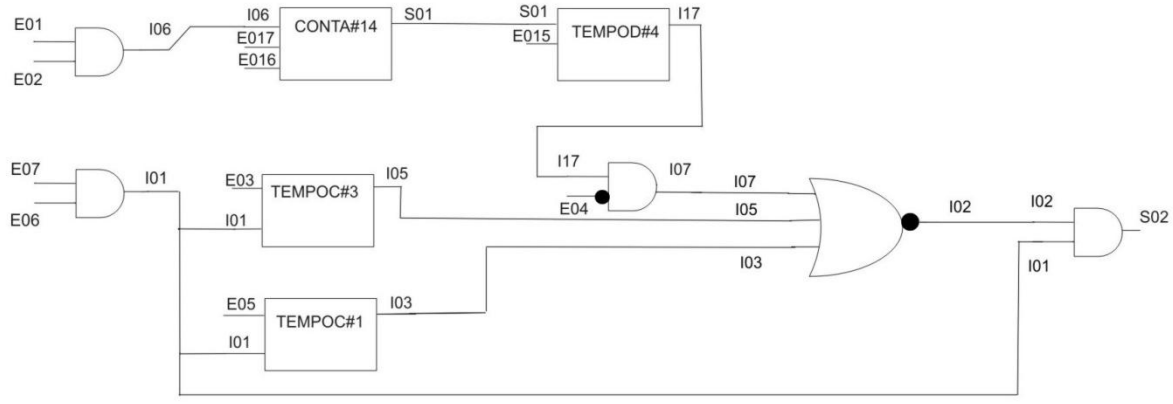


Figura 5.2 Solución esquemática al problema de la banda transportadora.

Para dar las señales de disparo de los temporizadores monodisparo tipo dos, TEMPOC#1 y TEMPOC#3 respectivamente, se utilizaron apagadores magnéticos (reed switch) conectados a un nivel de referencia. Para disparar el contador de eventos se utilizó un fototransistor, conectado a la entrada E02.

Las entradas E07 y E06 sirven para dar un control de habilitación al sistema, porque sólo cuando ambas están en nivel alto, la variable booleana intermedia I01 está también en nivel alto; lo que es una de las condiciones para hacer válida a la variable booleana S02 que maneja el motor.

Se utiliza la variable booleana de salida S01, para prender un *led*, que testifica que la cuenta en la estación 2 se ha verificado de manera satisfactoria y también habilita un módulo con retardo a la activación que hace esperar al contenedor 5 segundos más en la estación 2 antes de continuar a la estación 3.

Para restablecer el contador de eventos, se utilizó una conexión entre la variable de restablecimiento E016 y la variable booleana de entrada E04, de manera que al declararse en nivel alto esta última se da la señal que regresa el contador a su estado inicial.

Por último, el motor que mueve la banda es de 6VCD y se controla mediante la señal que da la variable booleana de salida S02, al verificarse en nivel alto alimenta con un voltaje de 5VCD a un relevador MD5 de la marca SunHold que habilita o apaga la conexión entre el motor y su fuente de alimentación.

**5.3 Programa en lenguaje SIIL1 que da solución al problema de la banda transportadora.**

A continuación se muestra el programa en lenguaje SIIL1 que resuelve el problema de la banda transportadora y que es la implementación de lo mostrado en la figura 5.2

```
CONFIG1;

 INPROG;
 AND2#1 E07,E06,I01,11;
 AND2#2 I01,I02,S02,11;
 NOR3#1 I03,I05,I07,I02,111;
 AND2#3 E02,E01,I06,11;
 AND2#4 E04,I17,I07,01;
 FINPP;
```

```
INMODI;
TEMPOC#1 E05,I01,I03,00:00:05.00,011;
CONTA#14 I06,E017,E016,S01,00005,00000,01001;
TEMPOD#4 S01,E015,I17,00:00:05.00,10;
TEMPOC#3 E03,I01,I05,00:00:05.00,011;
FINMODI;
```

Con la finalidad de brindar al lector un repaso, se analizan las características de las compuertas lógicas y módulos temporizados contenidos en este programa en cuanto a sus características generales.

AND2#1 E07,E06,I01,11;

Compuerta lógica AND de dos entradas, sin preinversión. Variables booleanas de entrada E07 y E06 respectivamente. La variable booleana de salida es la variable intermedia I01.

AND2#2 I01,I02,S02,11;

Compuerta lógica AND de dos entradas, sin preinversión. Variables booleanas de entrada I01 y I02 respectivamente. La variable booleana de salida es S02.

NOR3#1 I03,I05,I07,I02,111;

Compuerta lógica NOR de tres entradas, sin preinversión. Variables booleanas de entrada I03, I05 y I07 respectivamente. La variable booleana de salida es la variable intermedia I02.

AND2#3 E02,E01,I06,11;

Compuerta lógica AND de dos entradas, sin preinversión. Variables booleanas de entrada E02 y E01 respectivamente. La variable booleana de salida es la variable intermedia I06.

AND2#4 E04,I17,I07,01;

Compuerta lógica AND de dos entradas, con preinversión en su primer VBE. Variables booleanas de entrada E04 e I17 respectivamente. La variable booleana de salida es la variable intermedia I07.

TEMPOC#1 E05,I01,I03,00:00:05.00,011;

Temporizador monodisparo tipo dos; variable de disparo E05, variable de restablecimiento I01 y variable de salida I03. Se dispara por flancos de bajada; el temporizador es restablecido por nivel bajo y el nivel de verificación de la salida es alto. Su tiempo de duración es 5 segundos.

CONTA#14 I06,E017,E016,S01,00005,00000,01001;

Contador de eventos. Variable de disparo I06, variable que activa el congelamiento de cuenta es E017, variable de restablecimiento es E016. La salida del contador se encuentra en la variable booleana de salida S01. Sus características son: modifica su cuenta para flanco de bajada en la entrada de disparo D; nivel de verificación de la entrada de congelamiento es alto; nivel de la entrada de restablecimiento es alto; su cuenta es descendente; nivel de verificación de la salida es alto, su cuenta inicial es 5 y su cuenta final llega a 0.

TEMPOD#4 S01,E015,I17,00:00:05.00,10;

Temporizador con retardo a la activación o a la desactivación. Variable de entrada D al temporizador es S01. Variable de entrada de restablecimiento E015. Variable de salida del temporizador es la variable booleana I17. Su duración es de 5 segundos. Presenta retardo a la activación y es restablecido por nivel alto.

TEMPOC#3 E03,I01,I05,00:00:05.00,011;

Temporizador monodisparo tipo dos; variable de disparo E03, variable de restablecimiento I01 y variable de salida I05. Se dispara por flancos de bajada; el temporizador es restablecido por nivel bajo y el nivel de verificación de la salida es alto. Su tiempo de duración es 5 segundos.

Este programa debe ser guardado bajo la extensión *.sil para su uso posterior. Para este caso se le llamó *ejemplocapitulo6.sil*

**5.4 Generación del programa en lenguaje ensamblador usando el software GEN_ENS_PLM08.**

Como se mencionó en la sección 4.3 del capítulo anterior, la guía rápida del usuario nos da los pasos a seguir para obtener el programa en lenguaje ensamblador.

Abrimos el archivo de interés, en este caso *ejemplocapitulo6.sil* como se muestra en la figura 5.3.

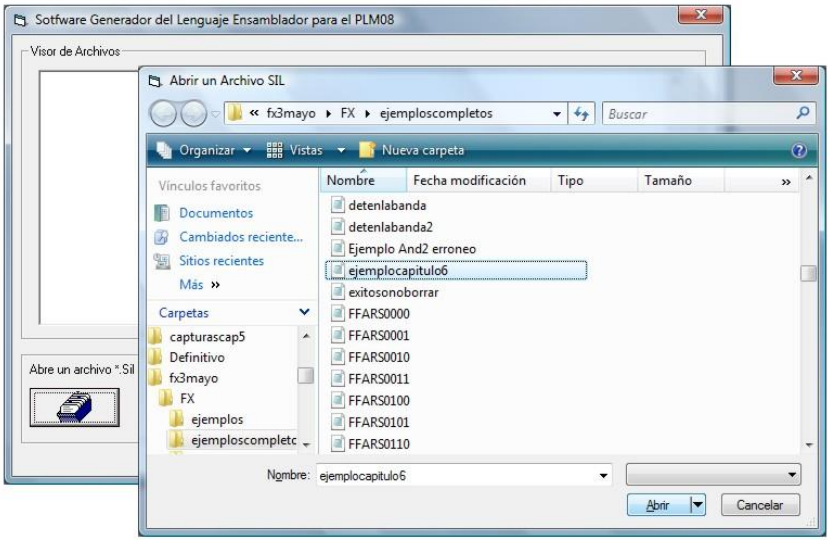


Figura 5.3 Apertura del archivo correspondiente a la solución de control de la banda transportadora.

Se depura el código en busca de errores, y si no se tienen se procede a generar el código en lenguaje ensamblador. Como se muestra en las figuras 5.4 y 5.5 respectivamente.

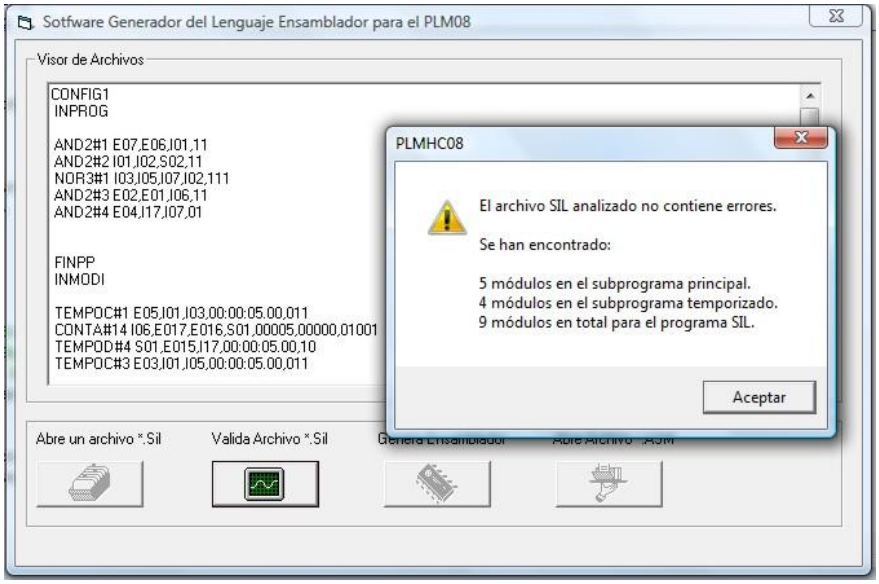


Figura 5.4 Resultado de la depuración del código en busca de errores. Para el archivo *ejemplocapitulo6.sil*

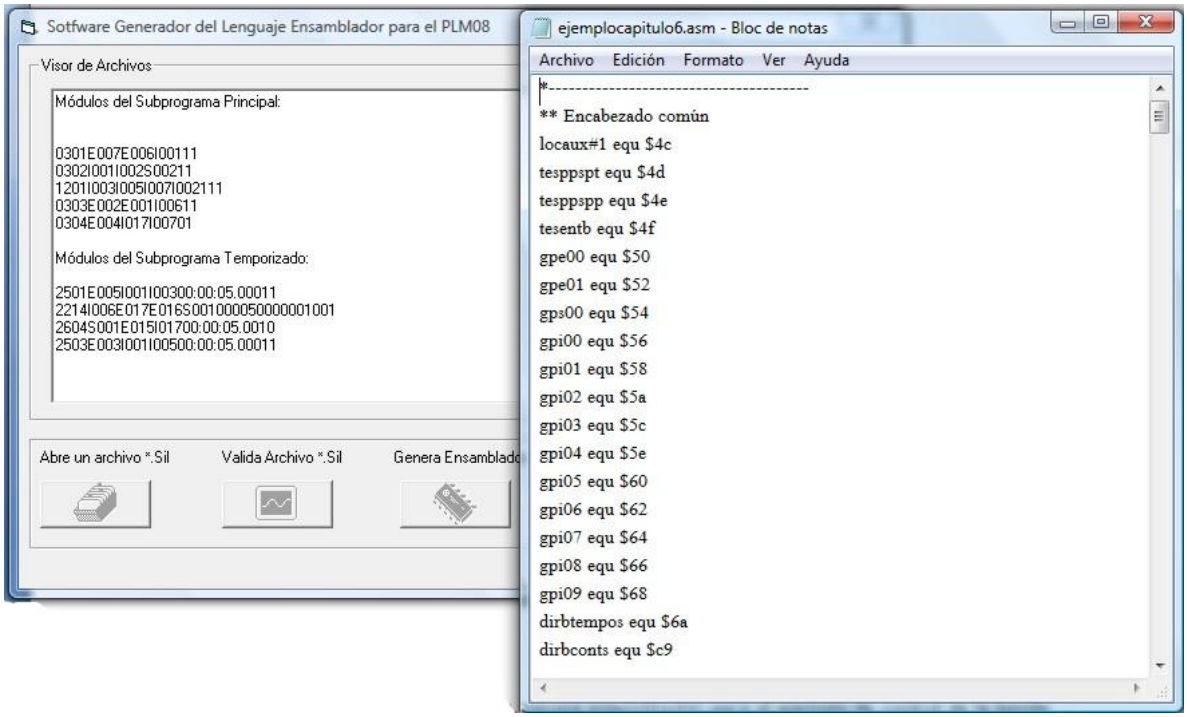


Figura 5.5 Generación del código en lenguaje ensamblador; para el ejemplo de control de la banda transportadora.

El programa generado por el software GEN_ENS_PLM08 se transcribe a continuación:

```
*----------------------------------------
** Encabezado común
locaux#1 equ $4c
tesppspt equ $4d
tesppspp equ $4e
tesentb equ $4f
gpe00 equ $50
gpe01 equ $52
gps00 equ $54
gpi00 equ $56
gpi01 equ $58
gpi02 equ $5a
gpi03 equ $5c
gpi04 equ $5e
gpi05 equ $60
gpi06 equ $62
gpi07 equ $64
gpi08 equ $66
gpi09 equ $68
dirbtempos equ $6a
dirbconts equ $c9
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 #gpi09+2
                bne otrocl

                mov #$07,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
                coma
                sta gpe00
                lda ptb
                coma
                sta gpe01
**** Fin de código asociado con sentencia INPROG *****
*---------------------------------------
*---------------------------------------
```

*Inicio del Ciclo de Módulos del Programa Pricipal

```
                clr tesentb
                brclr 7,gpe00,and2#0001
                bset 0,tesentb
and2#0001:      brclr 6,gpe00,and2#1001
                bset 1,tesentb
and2#1001:      lda tesentb
                eor #$00
                cmp #$03
                beq and2#2001
                bclr 1,gpi00
                bra and2#3001
and2#2001:      bset 1,gpi00
and2#3001:      nop

                clr tesentb
                brclr 1,gpi00,and2#0002
                bset 0,tesentb
and2#0002:      brclr 2,gpi00,and2#1002
                bset 1,tesentb
and2#1002:      lda tesentb
                eor #$00
                cmp #$03
                beq and2#2002
                bclr 2,gps00
                bra and2#3002
and2#2002:      bset 2,gps00
and2#3002:      nop

                clr tesentb
                brclr 3,gpi00,nor3#0001
                bset 0,tesentb
nor3#0001:      brclr 5,gpi00,nor3#1001
                bset 1,tesentb
nor3#1001:      brclr 7,gpi00,nor3#2001
                bset 2,tesentb
nor3#2001:      lda tesentb
                eor #$00
                cmp #$00
                beq nor3#3001
```

```
            bclr 2,gpi00
            bra nor3#4001
nor3#3001:      bset 2,gpi00
nor3#4001:      nop

            clr tesentb
            brclr 2,gpe00,and2#0003
            bset 0,tesentb
and2#0003:      brclr 1,gpe00,and2#1003
            bset 1,tesentb
and2#1003:      lda tesentb
            eor #$00
            cmp #$03
            beq and2#2003
            bclr 6,gpi00
            bra and2#3003
and2#2003:      bset 6,gpi00
and2#3003:      nop

            clr tesentb
            brclr 4,gpe00,and2#0004
            bset 0,tesentb
and2#0004:      brclr 7,gpi01,and2#1004
            bset 1,tesentb
and2#1004:      lda tesentb
            eor #$02
            cmp #$03
            beq and2#2004
            bclr 7,gpi00
            bra and2#3004
and2#2004:      bset 7,gpi00
and2#3004:      nop

*---------------------------------------
*---------------------------------------
**** Código asociado con sentencia FINPP
                lda gps00
                sta ptc
                clc
                rora
                rora
                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 *****
*---------------------------------------
*---------------------------------------
*Inicio del Ciclo de Módulos del Programa Temporizado

            lda tesppspt
            bne npptc#01
resettc#01:     bclr 3,gpi00
inicontc#01:    mov #$00,dirbtempos+0
            ldhx #$01F4
            sthx dirbtempos+1
            bra salidatc#01
npptc#01:       brclr 1,gpi00,resettc#01
            lda gpe00+1
            and #$20
            sta locaux#1
            lda gpe00
            and #$20
            cmp locaux#1
            beq siguetc3#01
            blo versaltc#01
siguetc3#01:    brclr 3,gpi00,salidatc#01
            ldhx dirbtempos+1
            aix #$ff
            sthx dirbtempos+1
            cphx #$0000
            bne salidatc#01
            lda dirbtempos+0
```

```
            beq resettc#01
            dec dirbtempos+0
salidatc#01:    bra sigblotc01
versaltc#01:    bset 3,gpi00
            bra inicontc#01
sigblotc01:     nop



            lda tesppspt
            bne npptc#14
resettc#14:     bclr 1,gps00
inicontc#14:    ldhx #$0005
            sthx dirbconts+39
            ldhx #$0000
            sthx dirbconts+41
            bra salidatc#14
npptc#14:       brset 7,gpe01,salidatc#14
            brset 6,gpe01,resettc#14
            brset 1,gps00,salidatc#14
            lda gpi00+1
            and #$40
            sta locaux#1
            lda gpi00
            and #$40
            cmp locaux#1
            beq salidatc#14
            blo siguetc3#14
            bra salidatc#14
siguetc3#14:    ldhx dirbconts+39
            aix #$FF
            sthx dirbconts+39
            ldhx dirbconts+39
            cphx dirbconts+41
            beq versaltc#14
salidatc#14:    bra sigblotc#14
versaltc#14:    bset 1,gps00
            bra inicontc#14
sigblotc#14:    nop
```

```
            lda tesppspt
            bne npptc#04
resettc#04:     bclr 7,gpi01
inicontc#04:    mov #$00,dirbtempos+9
            ldhx #$01F4
            sthx dirbtempos+10
            bra salidatc#04
npptc#04:       brclr 1,gps00,resettc#04
            brset 5,gpe01,resettc#04
            brset 7,gpi01,salidatc#04
            ldhx dirbtempos+10
            aix #$ff
            sthx dirbtempos+10
            ldhx dirbtempos+10
            cphx #$ffff
            beq versaltc#04
            bne salidatc#04
            dec dirbtempos+9
            ldhx dirbtempos+10
            cphx #$0000
            bne salidatc#04
            lda dirbtempos+9
            beq versaltc#04
salidatc#04:    bra sigblotc#04
versaltc#04:    bset 7,gpi01
sigblotc#04:    nop



            lda tesppspt
            bne npptc#03
resettc#03:     bclr 5,gpi00
inicontc#03:    mov #$00,dirbtempos+6
            ldhx #$01F4
            sthx dirbtempos+7
            bra salidatc#03
npptc#03:       brclr 1,gpi00,resettc#03
            lda gpe00+1
            and #$08
            sta locaux#1
```

```
            lda gpe00
            and #$08
            cmp locaux#1
            beq siguetc3#03
            blo versaltc#03
siguetc3#03:    brclr 5,gpi00,salidatc#03
            ldhx dirbtempos+7
            aix #$ff
            sthx dirbtempos+7
            cphx #$0000
            bne salidatc#03
            lda dirbtempos+6
            beq resettc#03
            dec dirbtempos+6
salidatc#03:    bra sigblotc03
versaltc#03:    bset 5,gpi00
            bra inicontc#03
sigblotc03:     nop


*---------------------------------------
*---------------------------------------
**** Código asociado con sentencia FINMODI
                clrh
                ldx #gpe00
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
*---------------------------------------
```

Este código en lenguaje ensamblador puede ser grabado directamente en el microcontrolador 68HC908GP32 a fin de utilizar las prestaciones de la tarjeta MINICON_08A y la tarjeta de entradas y salidas, para lograr el control efectivo de la banda transportadora.

## 5.5 Ejecución del programa en lenguaje ensamblador usando el software PUMMA_08+.

Se presenta a continuación una breve guía del software manejador PUMMA_08+; diseñado por el M.I Antonio Salvá Calleja y cuya información completa de uso se encuentra disponible en internet en la dirección http://dctrl.fi-b.unam.mx/~salva/MUAIDA08b_enero_2010.pdf para su ejecución con la tarjeta de desarrollo MINICON_O8A .Ver figura 5.6.

De manera general, se muestra como ejecutar el programa en lenguaje ensamblador generado por el software GEN_ENS_PLM08.

### 5.5.1 Inicialización del sistema.


Figura 5.6 Tarjeta de desarrollo MINICON_08A

Primero debe polarizarse la tarjeta MINICON-08A. Para este caso se recomienda el eliminador de baterías ELI-100, fabricado y distribuido por la empresa STEREN®; seleccionando preferentemente un voltaje comprendido entre los 7 y 9 volts.

Para inicializar PUMMA_08+ para su operación utilizando una PC, está debe contar con un puerto serie disponible. En el caso de una computadora *notebook*, se debe usar un adaptador USB-SERIE, y se debe haber instalado el driver de éste previo a la primera ejecución del software.

Los pasos a seguir para la inicialización son:

1. Si la PC se encuentra ejecutando alguna aplicación que use el puerto serie que el usuario piensa emplear para fines de PUMMA_08+, está deberá ser terminada y cerrada.
2. Energizar la tarjeta MINICON_08A .Después de esto se deberá observar un parpadeo del LED1, testificandosé así que la tarjeta de desarrollo puede ser manejada por PUMMA_08+.
3. Empleando un cable RS-232C conectar la tarjeta de desarrollo al puerto serie disponible en la PC.
4. Instalar en la PC el software manejador PUMMA_08A.
5. Ejecutar PUMMA_08A y dar el número de puerto serie disponible cuando esté lo pida. En caso de que el puerto serie dado por el usuario no esté disponible, PUMMA_08+ despliega el mensaje “Puerto serie ocupado o no existente” y se debe volver al paso 1.
6. Definir 9600 bps como baudaje cuando el software manejador lo pida. Figura 5.7.
7. Después de la confirmación mencionada en el paso 5, se pedirá al usuario definir el microcontrolador presente en la tarjeta de desarrollo, en el caso de la tarjeta MINICON_08A, se debe seleccionar la opción 2 que corresponde al microcontrolador 68HC908GP32. Figura 6.8.

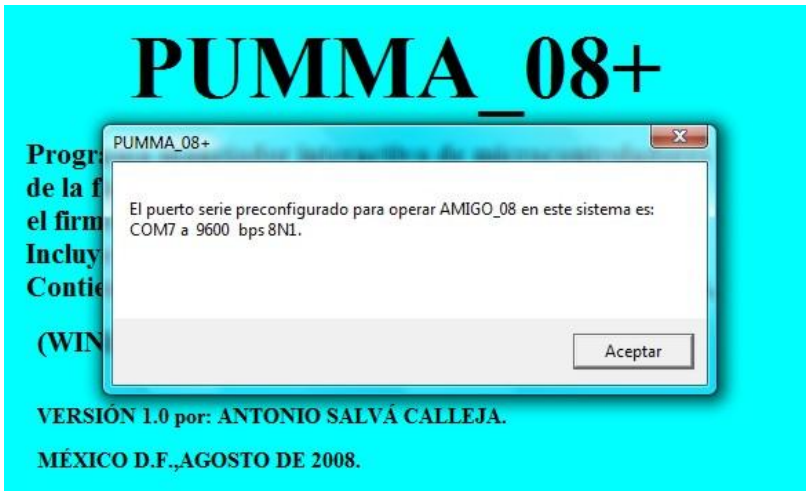


Figura 5.7 Confirmación de PUMMA_08A acerca del puerto serie a emplear en el enlace.

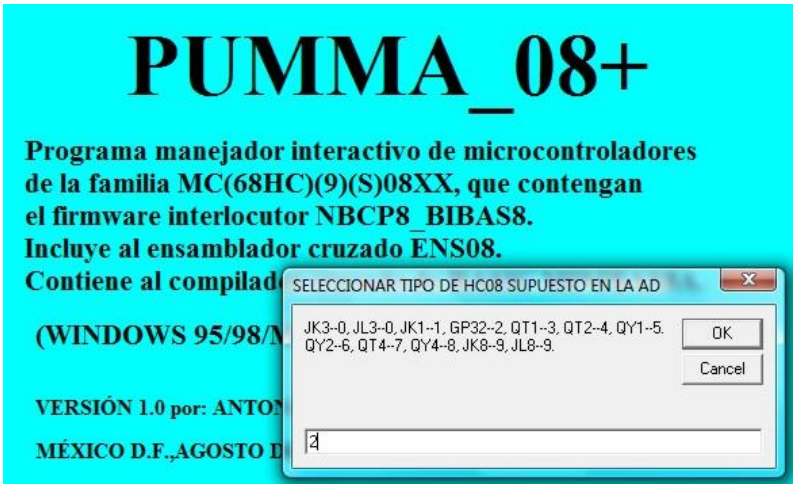


Figura 5.8 Dialogo de PUMMA_08A para definir el tipo de microcontrolador presente en la tarjeta de desarrollo.

Después de que el usuario ha definido el microcontrolador presente en la tarjeta de desarrollo, debe aparecer la ventana de edición de PUMMA_08+. Bajo estas condiciones el sistema se haya listo para trabajar. Como se muestra en la figura 5.9.

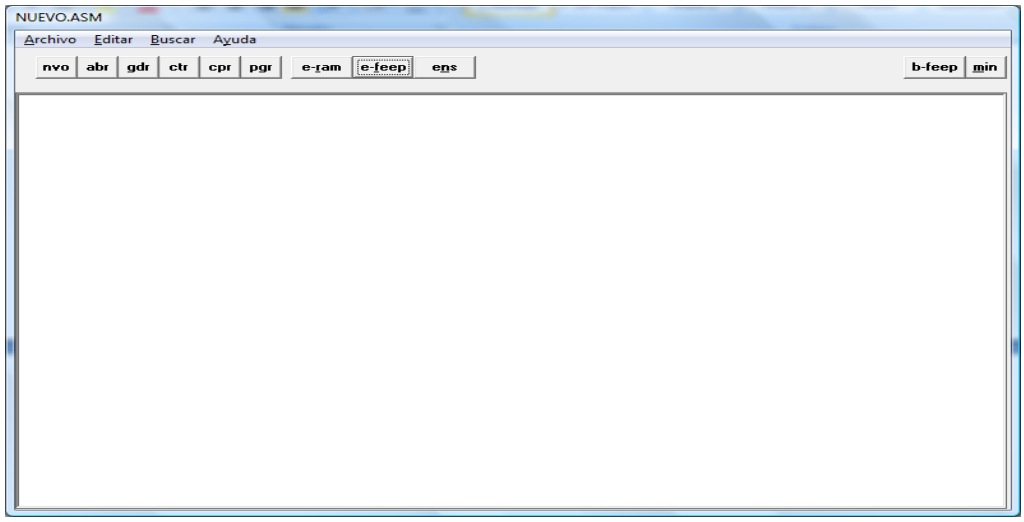


Figura 5.9 Ventana de edición de PUMMA_08+

### 5.5.2 Carga y ejecución del programa que resuelve el ejemplo de control de la banda transportadora.

Ya habilitada la ventana de edición de PUMMA_08+. Simplemente debe cargarse el programa generado por GEN_ENS_PLM08 como se muestra en la figura 5.10.

Para lograr la carga exitosa del programa, se utiliza el botón *abr* ubicado en la barra principal de la ventana de edición del programa PUMMA_08+ y se busca la ruta a la carpeta denominada *salidas*, en la ubicación raíz del programa GEN_ENS_PLM08. En esta carpeta se guardan los archivos con extensión .ASM que pueden ejecutarse con la tarjeta de desarrollo MINICON_08A.

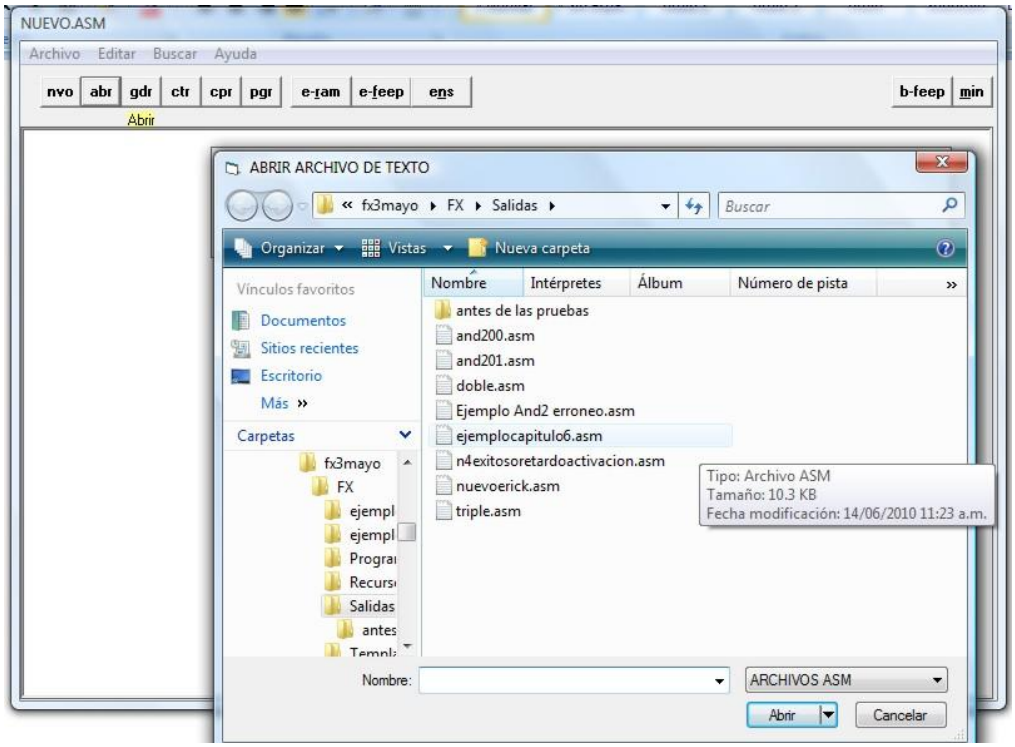


Figura 5.10 Programa generado por el software GEN_ENS_PLM08 cargado a la ventana de edición del software manejador y desarrollo PUMMA_08+.

El siguiente paso es guardar el archivo con su nombre definitivo, usando el botón *gdr* ubicado en la barra principal del programa PUMMA_08+ (figura 5.11).

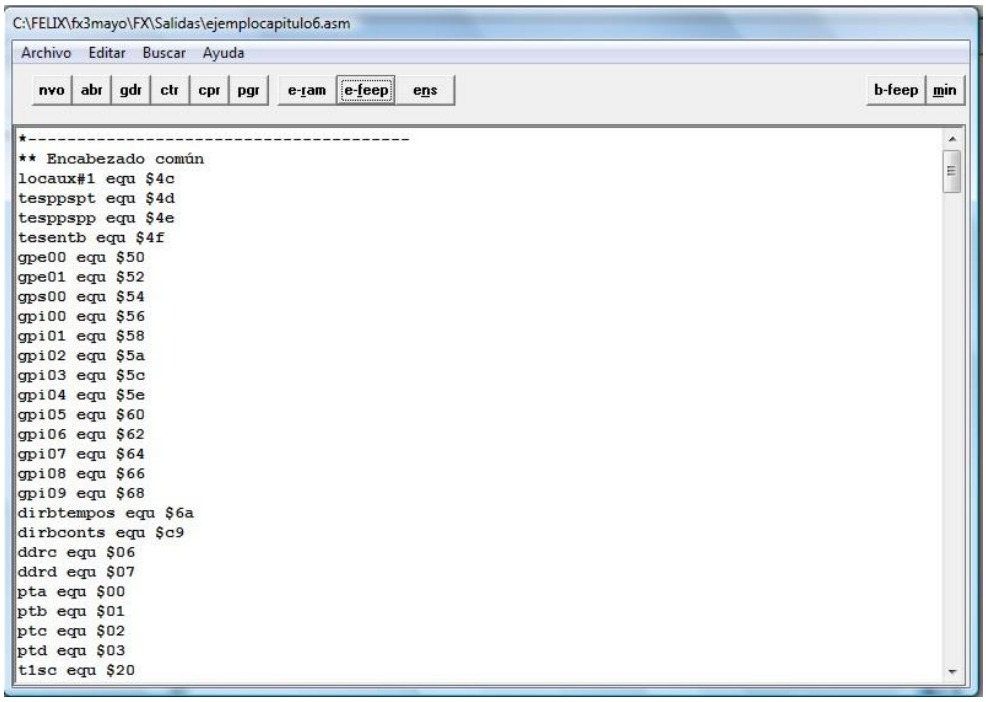


Figura 5.11 Archivo generado por el programa GEN_ENS_PLM08 cargado en el programa PUMMA_08+.

Después se debe borrar cualquier programa previo, que se pueda encontrar grabado en el microcontrolador. Para hacer esto, se presiona dentro de la ventana de edición de PUMMA_08+ el botón *b-feep*. Este botón se halla localizado, en la parte superior derecha de la ventana de edición como se muestra en la figura 5.12


Figura 5.12 Botón para borrar la memoria del microcontrolador 68HC908GP32.

Es importante destacar que para cualquier accionamiento sobre la tarjeta de desarrollo, el LED1 debe presentar un continuo proceso de encendido y apagado; lo cual testifica que el receptor de comandos está activo. Al ejecutarse algún programa del usuario, esté toma el control de la tarjeta de desarrollo y el LED1 entonces se apagará. Bajo esta circunstancia la tarjeta de desarrollo no puede ser manejada hasta oprimir el botón correspondiente de reset.

Una vez borrado cualquier programa previo; basta con oprimir el botón e-feep (ejecución inmediata en memoria FLASH), que se localiza en la parte media del menú de la ventana de edición como se muestra en la figura 5.13.


Figura 5.13 botón e-feep

Este botón ensambla el programa presente en la ventana del editor y si no hay errores de sintaxis, este se carga y ejecuta de manera inmediata en la tarjeta de desarrollo. En caso de haber errores, al primero de estos se detiene el proceso de ensamble y se da un reporte al usuario. El programa fuente siempre debe estar en lenguaje ensamblador, para evitar errores de ejecución.

Referencias:


- 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 – "Ambiente integrado para desarrollo y aprendizaje con microcontroladores de la familia 68HC908 de Freescale". Manual de usuario básico. División de Ingeniería Eléctrica, Facultad de Ingeniería, UNAM, enero de 2010.