UNIVERSIDAD NACIONAL AUTONOMA DE MEXICO

FACULTAD DE INGENIERIA
División de Estudios de Posgrado

PROGRAMADOR LOGICO MODULAR

T E S I S
Presentada por

ANTONIO SALVA CALLEJA

para obtener el grado de
MAESTRO EN INGENIERIA
(E L E C T R I C A)

Director de Tesis:
M. en C. Luis Marcial Hernández Ortega

México, D. F. **1999**

A la memoria de mis padres

Máximo Salvá Contreras

María Teresa Calleja de Salvá

A mis hermanos

Alberto y Jorge

A la Facultad de Ingeniería de la

Universidad Nacional Autónoma de México,

por estos años de libertad para aprender, enseñar y hacer

AGRADECIMIENTOS

Al maestro Luis Marcial Hernández Ortega, por su atinada guía, comprensión y apoyo para la realización de este trabajo.

Gracias Marcial.

Al Ing. Víctor Manuel Sánchez Esquivel, por su valiosa colaboración en el desarrollo de la versión preliminar de la circuitería del PLM.

Gracias Víctor.

A la Ing. Fabiola Guzmán Peña, quien alambró parte de la circuitería de la versión preliminar del PLM.

Gracias Fabiola.

Al Ing. Agustín Soto Urrutia, por su entusiasta participación en el armado y prueba de las tarjetas de periféricos que integran el PLM.

Gracias Agustín.

A la Ing. Lidia Botello Acosta, quien alambró circuitos experimentales, que sirvieron para efectuar diversas pruebas previas, relacionadas con el hardware y software del PLM.

Gracias Lidia.

A la señorita María Leonor Salcedo Ubilla, quien realizó los dibujos de esta tesis.

Gracias Leonor.

Cualquier empresa, no importando lo complicado que sea el realizarla, es siempre la suma de esfuerzos constantes y simples.

ÍNDICE

ÍNDICE

ÍNDICE

ÍNDICE

ÍNDICE

INTRODUCCIÓN

Los procesos industriales, para fines de su control y/o monitoreo, pueden subdividirse en varios subprocesos individuales que tienen asociados a ellos variables fisicas tales como: temperatura, presión, nivel. Cada subproceso individual es típicamente controlado mediante un sistema de lazo cerrado, analógico o digital.

En la práctica los diversos subprocesos requieren de un arbitraje lógico que regule secuencias de eventos entre ellos, un ejemplo de esta clase de secuencias es: una bomba que suministre un reactivo, que ha de combinarse químicamente con otra substancia en una autoclave. Esta bomba, deberá funcionar sólo por un tiempo determinado y a condición de que el otro reactivo ya se encuentre presente a una determinada temperatura; tanto la temperatura como el nivel requeridos son controladas individualmente por sendos lazos.

Se intuye la necesidad de una instancia de control que podría ser una compuerta lógica AND combinada con un temporizador; las entradas de la compuerta serían variables booleanas cuyo estado testificaría simplemente si la temperatura es adecuada o no y si el nivel de la segunda substancia en la autoclave es conveniente o no; la salida de la compuerta podría disparar a un sistema de temporización que mantuviera en operación la bomba de suministro del primer reactivo el tiempo requerido.

El arbitraje lógico mencionado en el párrafo anterior es denominado *control lógico o secuencial* y para llevarlo a cabo se requiere de lo siguiente:

1.- Sensores booleanos de diversas condiciones de proceso, que testifiquen el que variables asociadas con los diversos subprocesos locales se encuentren o no en un determinado rango.

2.- Sistema lógico, conformado típicamente por compuertas lógicas, flip-flops, temporizadores, etc, el cual procesa las señales proporcionadas por los sensores booleanos.

3.- Actuadores lógicos cuyas entradas son a su vez salidas del sistema lógico mencionado en el párrafo anterior.

Por lo anterior, en la industria es muy frecuente la necesidad de llevar a cabo el control secuencial, de eventos relacionados con bloques funcionales asociados a un determinado proceso productivo, ejemplos de esto pueden apreciarse en la industria automotriz, de alimentos y petroquímica solo por mencionar algunas.

Hasta la década de los setentas el sistema lógico asociado, con un sistema de control secuencial, era realizado por elementos físicos fijos, que realizaban las compuertas requeridas mediante relevadores o compuertas electrónicas que se interconectaban físicamente de acuerdo con lo que un determinado proceso requiriera en un momento dado. En caso de que hubiera que hacer modificaciones al sistema lógico había que realambrar o incluso rehacer completamente el hardware requerido, esto consumía mucho tiempo y dinero; en la figura I.1 se muestra un esquema posible de alambrado para la situación de control lógico mencionada en el primer párrafo.

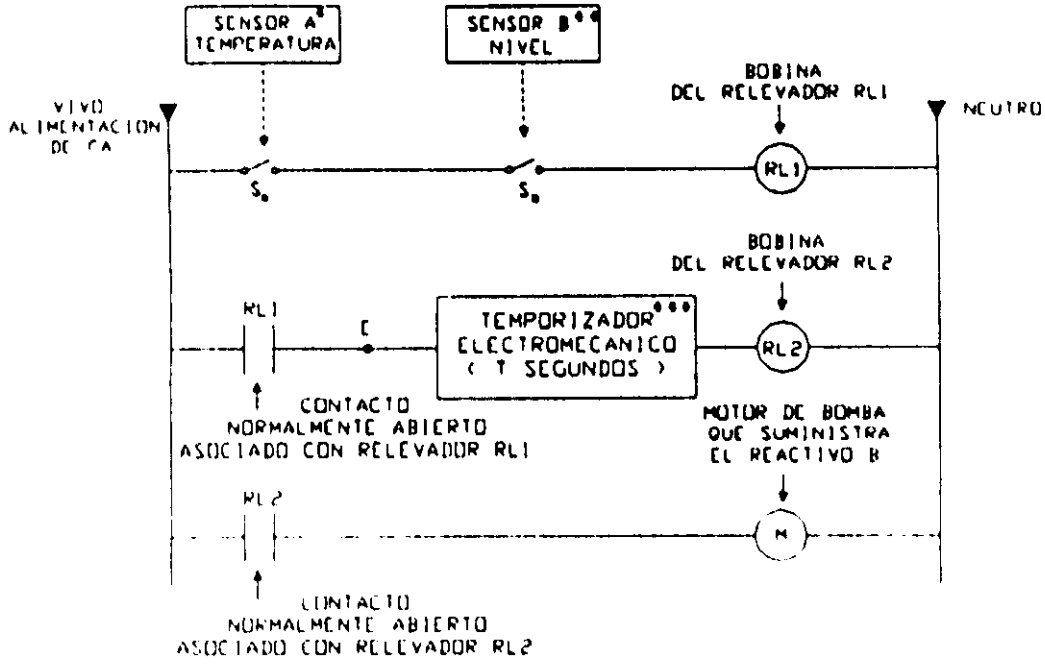


Figura I.1.- Esquema de alambrado de la situación de control lógico que requiere que una bomba suministre un reactivo B, a una autoclave por un tiempo T, sólo si el otro reactivo, denominado A ya se encuentre presente con una temperatura Tb. Véanse notas aclaratorias en la siguiente página.

* *El sensor A cierra el interruptor s_a cuando la temperatura del reactivo A, presente en la autoclave, está comprendida entre T_1 y T_2, se supone que la temperatura requerida está comprendida en ese rango.*
** *El sensor B cierra el interruptor s_b cuando el reactivo A se encuentre presente en la autoclave con el volumen requerido.*
****El temporizador electromecánico energiza la bobina del relevador rl2, por un tiempo T, cuando en su entrada (E) pasa el potencial del vivo del suministro eléctrico.*
En la industria, a un esquema como el mostrado en la figura anterior, se le llama *diagrama de escalera* dada la obvia similitud visual.

Otra manera de realizar la situación de control de la figura I.2 seria empleando componentes electrónicos integrados, esto se muestra en la figura I.2.

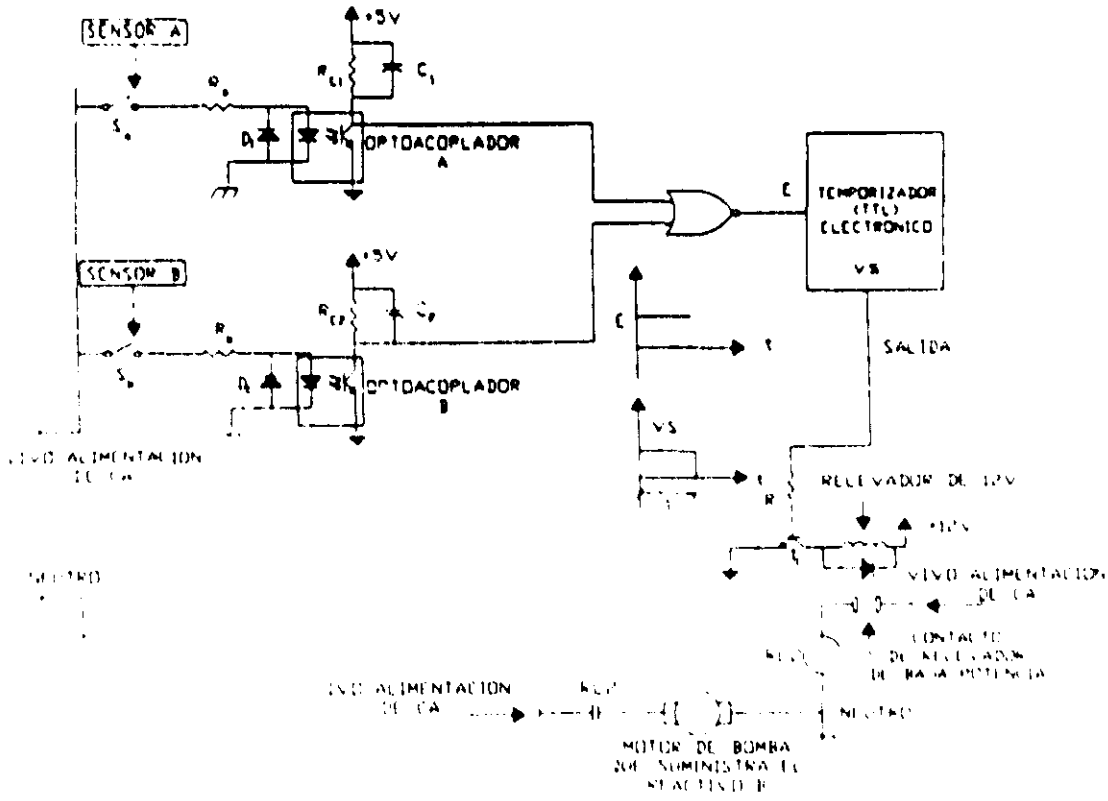


Figura I.2.- Esquema que realiza la situación de control de la figura I.1, empleando componentes electrónicos

En la figura I.2 se aprecia que a diferencia del esquema de la figura I.1 la lógica requerida es efectuada a un nivel de baja potencia y las componentes físicas involucradas ocupan menos espacio, aunque se sigue teniendo el problema de realambrado cuando hubiera que hacer cambios en la lógica de control.

Otra forma de resolver el problema sería substituir la electrónica TTL por una computadora monotablilla que tuviera un puerto de entrada y un puerto de salida, la ventaja de esta solución sería que hacer cambios en la lógica de control, implicaría sólo cambiar el programa que se ejecuta en la computadora monotablilla, sin que sea necesario hacer modificaciones al hardware. En la figura I.3 se muestra como se realizaría esta solución.

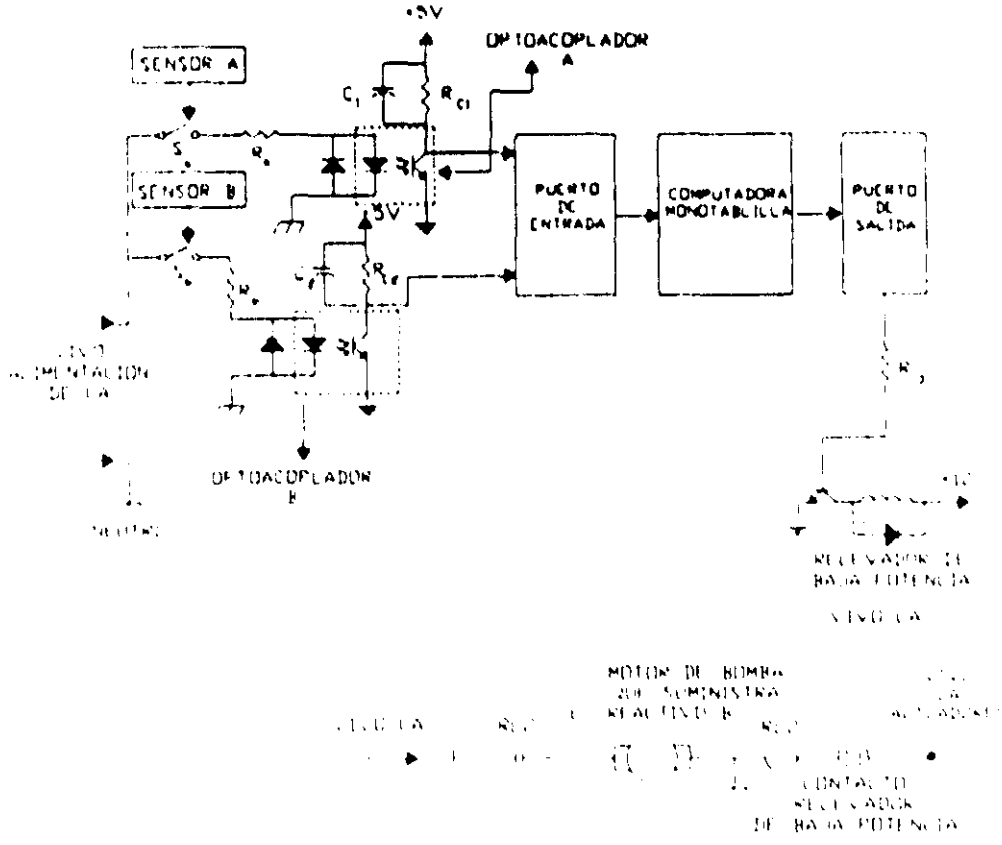


Figura I.3.- Solución a la situación de control lógico de la figura I.1 empleando una computadora monotablilla.

En las tres soluciones presentadas se destaca el hecho de que tanto los sensores como los actuadores requeridos no cambian; sin embargo, cabe señalar que frecuentemente en la práctica los niveles lógicos empleados, tanto para las entradas que reciben las señales de los sensores como para las salidas que disparan los actuadores, son 24 Volts para el uno lógico y cero Volts para el cero lógico.

En campo, el número de variables de entrada y salida requeridas en un momento dado, es mayor que las implicadas en el sencillo ejemplo discutido aquí, aunque es fácil ver que tal requerimiento puede ser solventado por cualquier computadora monotablilla basada en algún microcontrolador o microprocesador comercial.

Por lo anterior puede pensarse en un sistema genérico para control lógico que contuviera un optoacoplador por cada entrada booleana y sendos relevadores de baja potencia por cada salida booleana, ligado todo esto con una computadora monotablilla que ejecutara un programa que validara el control lógico requerido para un proceso dado; tal sistema genérico deberá presentar, para que el usuario tenga acceso a las entradas y salidas, lo siguiente:

N terminales de entrada para conectar sensores booleanos.

Una terminal de entrada para el neutro de la alimentación de C.A., o bien el negativo de la fuente de 24 volts

M terminales de salida asociadas cada una con uno de los contactos del relevador de salida correspondiente y una terminal del lado de las salidas para conexión del vivo de la línea de C.A., o la terminal positiva de la fuente de 24 volts; el uso de el suministro de alterna o la fuente de directa dependerá de si el uno lógico, tanto para sensores como para actuadores es verificado con 24 volts o con 120 Volts de C.A.

En la figura 1.4 se muestra un esquema a bloques simplificado de un sistema genérico para control lógico basado en una computadora monotablilla

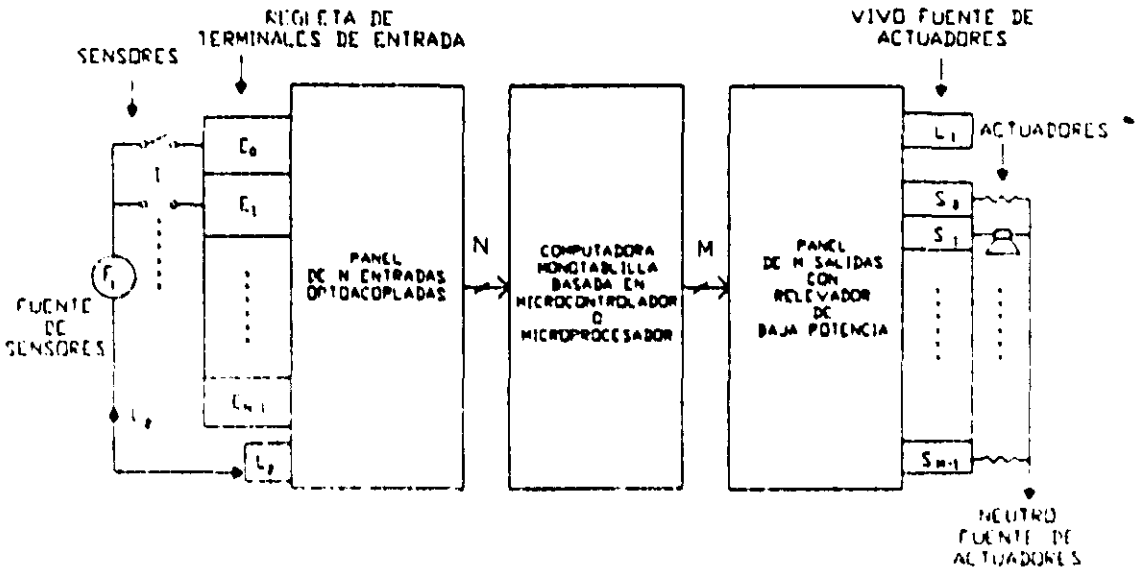


Figura 1.4.- Sistema genérico para control lógico basado en una computadora monotablilla

Aún cuando el sistema genérico de control lógico mencionado en el párrafo anterior, presenta muchas ventajas respecto a las soluciones de lógica alambrada, todavía hay que considerar que requiere además de un programador experto que domine aspectos tanto de hardware como de software relacionados con el microcontrolador o microprocesador que sea el núcleo de la computadora monotablilla implicada.

En consecuencia, es deseable contar con un lenguaje de programación que genere código para la computadora monotablilla, de modo que el usuario final no se las tenga que ver con detalles técnicos de la arquitectura y funcionamiento del microcontrolador correspondiente, sino solo con la manera en que debe declarar los *módulos lógicos* que su aplicación requiera en un momento dado.

Un sistema que conjunte el hardware descrito como sistema genérico de control lógico aunado con software que facilite al usuario final el desarrollo de aplicaciones es conocido en la industria como CONTROLADOR LÓGICO PROGRAMABLE, (PLC por sus siglas en inglés).

El objetivo del proyecto de tesis, reportado en esta presentación escrita, fue el diseñar

y construir un prototipo experimental denominado PROGRAMADOR LÓGICO MODULAR (PLM), que realiza bloques funcionales denominados módulos lógicos, que son usualmente requeridos en el control lógico de procesos; tales módulos manejarían entradas y salidas binarias y se implantan mediante tramos de código ejecutable por una computadora monotablilla basada en el microcontrolador 68HC11F1. La programación del PLM se realiza con el auxilio de un Software de Interpretación de Instrucciones Lógicas (SIIL1), que corre en una computadora de tipo PC. El diseño del lenguaje de programación del PLM y el software de traducción es parte importante de este trabajo.

CAPÍTULO 1

ESTRUCTURA Y FUNCIONAMIENTO BÁSICO DEL PROGRAMADOR LÓGICO MODULAR

En este capítulo se describe de una manera general la estructura a bloques del Programador Lógico Modular (PLM); la organización y nomenclatura asociada con las variables booleanas que maneja; las características a nivel de "caja negra", de los módulos lógicos que puede realizar el dispositivo y el formato sintáctico de las instrucciones para declararlos en SIIL1 (lenguaje propio del PLM para utilización por parte del usuario final).

Se presenta una descripción del formato que un programa en SIIL1 debe tener de manera que el mismo pueda ser procesado en una PC para obtener código objeto, listo para ser cargado y ejecutado en el PLM.

El capítulo concluye con la metodología a seguir para la realización de una aplicación de control lógico empleando el PLM.

1-1 ESTRUCTURA BÁSICA DEL PROGRAMADOR LÓGICO MODULAR

El PLM, es un dispositivo orientado a la realización de diversos bloques funcionales típicos de aplicaciones de control lógico, como podrían ser: compuertas lógicas, temporizadores, contadores de eventos y secuenciadores de estados; en la nomenclatura del PLM se le llama módulo lógico a un bloque de los mencionados anteriormente, realizado virtualmente por software ejecutable en el microcontrolador 68HC11F1, que gobierna el funcionamiento del dispositivo.

Los módulos lógicos que el PLM puede realizar son:

a) Compuertas AND de dos, tres y cuatro entradas.

b) Compuertas OR de dos, tres y cuatro entradas.

c) Compuertas NAND de dos, tres, y cuatro entradas.

d) Compuertas NOR de dos, tres y cuatro entradas.

e) Compuertas XOR de dos, tres y cuatro entradas.

f) Compuertas XOR negada de dos, tres y cuatro entradas.

g) Inversores y seguidores lógicos.

h) Cinco tipos diferentes de temporizador.

i) Dos tipos de contadores de eventos.

j) Secuenciadores de estado de uno a ocho bits.

k) Flip-Flops asíncronos.

Cabe señalar aquí que las compuertas XOR y XOR negada se denominan respectivamente como EOR y EORN en la terminología del PLM; además de que para todas las compuertas existe la posibilidad de negación para cualquiera de las entradas, contribuyendo esto a disminuir el número de módulos requeridos en una determinada aplicación.

El PLM puede operar de dos modos denominados respectivamente autónomo y esclavo. Al operar de manera autónoma el PLM puede realizar un sistema de control lógico, ejecutando el código correspondiente que se encontrará residente como firmware en una EPROM contenida en el mismo, esta idea se ilustra en la figura 1.1; cuando el PLM opera en modo esclavo el mismo se encontrará ligado vía serie con una computadora de tipo PC donde puede correrse software que permitirá probar y depurar los programas que requiera el PLM para realizar una determinada aplicación de control lógico, véase la figura 1.2.

Figura 1.1 PLM operando en forma autónoma

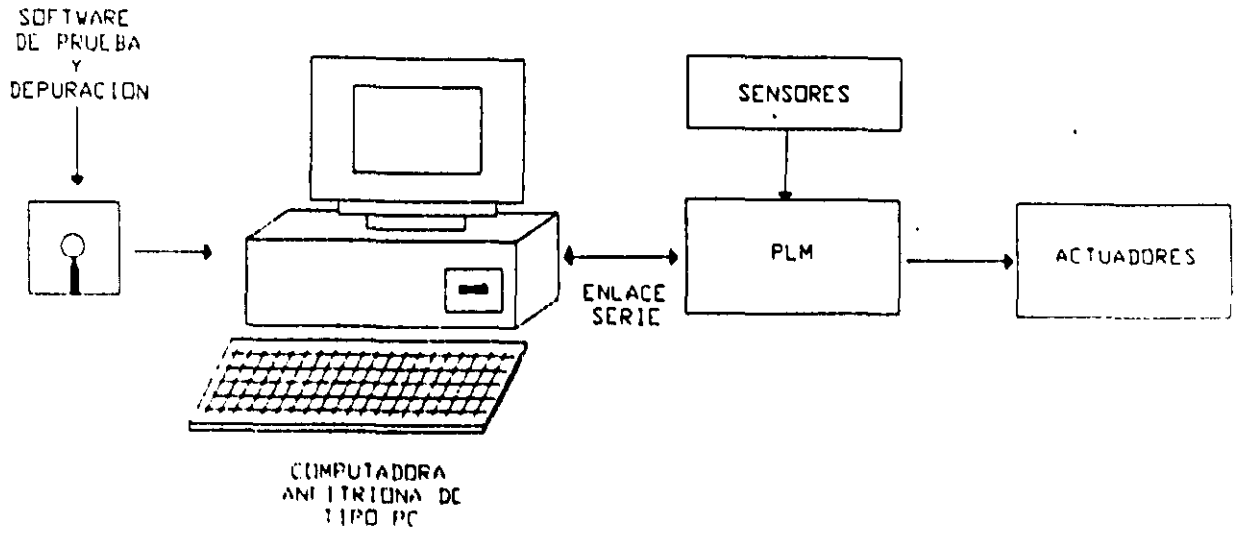


Figura 1.2 PLM operando en modo esclavo

La estructura a bloques del PLM se muestra en la figura 1.3, como se aprecia en la figura el dispositivo cuenta con 32 entradas y 16 salidas booleanas y está conformado por los siguientes cinco bloques funcionales:

1) Computadora Central (CC)
2) Bloque de Entradas (BE)
3) Bloque de Salidas (BS)
4) Bloque de Comando Local y Despliegue (BCLD)
5) Fuente de Alimentación (FA)

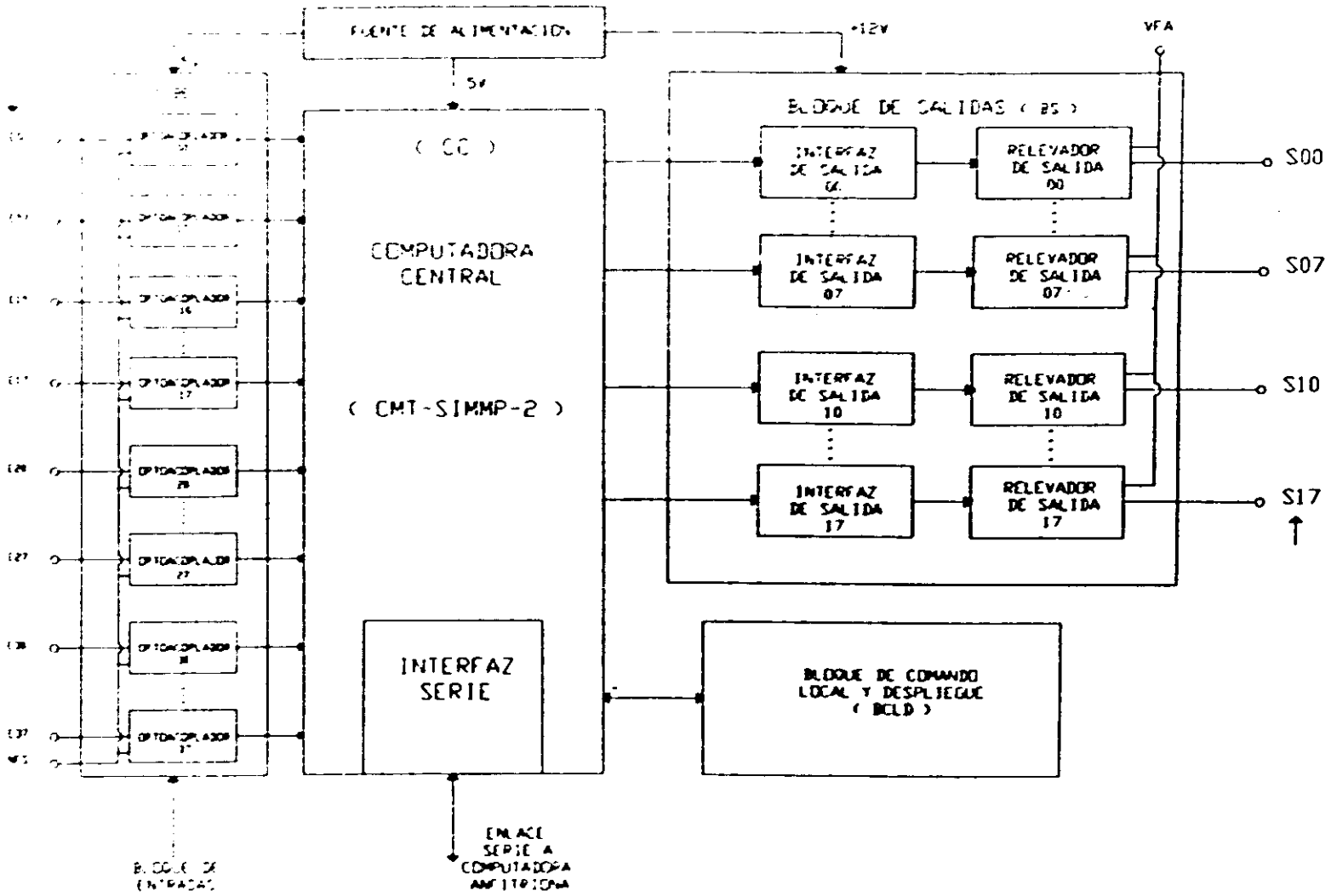


FIGURA 1.4 ESTRUCTURA A BLOQUES DEL PROGRAMADOR LOGICO MODULAR (PLM)

A continuación se describe, en lo general, el funcionamiento de cada uno de los bloques del PLM.

1-1-1 Computadora Central (CC)

La computadora central del PLM está realizada por la computadora monotablilla (CMT) SIMMP-2, cuya CPU es el microcontrolador 68HC11F1 fabricado por la compañía Motorola, la CMT SIMMP-2 puede operar en cualquiera de los cuatro modos en los que puede funcionar el 68HC11 y cuenta con facilidades que permiten que la misma opere de manera autónoma o bien controlada vía serie por una computadora anfitriona de tipo PC. La CMT SIMMP-2 fue desarrollada por el autor de esta tesis y tiene las siguientes características principales:

1) Capacidad para operar en cualquiera de los cuatro modos asociados con el 68HC11.

2) Firmware interlocutor que permite enlazarla vía serie a una computadora PC, donde se ejecuta un manejador hexadecimal (programa pumma.exe) mediante el cual se puede entre otras cosas hacer lo siguiente:

a) Cargar desde la PC, programas en lenguaje de máquina del microcontrolador para su ejecución en el mismo.

b) Lectura desde la PC, de la memoria contenida en la tarjeta.

c) Compatibilidad con herramientas de software asociadas con el 68HC11, permitiendo esto la ejecución en la tarjeta de programas originalmente escritos en lenguaje C o ensamblador, lográndose esto mediante la importación del archivo S19 correspondiente que haya sido generado por el software de ensamble o compilación respectivo.

d) Capacidad para configurar diversos mapas de memoria al operar en modo expandido

e) Programador integrado de memorias EPROM, mediante el cual pueden programarse EPROM's usando el propio manejador hexadecimal y hardware contenido en la tarjeta

La CMT SIMMP-2 como computadora central del PLM opera en el modo expandido del 68HC11, contándose en este caso con seis puertos, cuatro de entrada y dos

de salida, con los que se realizan a nivel de la CC las entradas y salidas con que cuenta el PLM; para más información acerca de la arquitectura, mapas de memoria y modos de configuración de la computadora monotablilla empleada, puede consultarse el apéndice A.

1-1-2 Bloque de Entradas (BE)

Esta parte está conformada por 32 entradas optoacopladas, el PLM reconoce un nivel de uno lógico, para una determinada entrada, cuando nominalmente se presente un voltaje de 24 volts medido entre la terminal correspondiente y el punto NFS (neutro de la fuente de sensores), en otro caso el nivel tomado será cero lógico. En realidad para los niveles de uno y cero lógico en las entradas corresponden sendos intervalos de voltaje, de esto se hablará en el capitulo 2 de este trabajo.

Como se observa en la figura 1.3, las 32 entradas están agrupadas en cuatro grupos de ocho entradas cada uno, esto se debe a que la información en el microcontrolador empleado está organizada en bytes.

Las entradas son denotadas empleando tres caracteres, el primero puede ser la letra "e" como mayúscula o minúscula, el segundo es un número comprendido en un rango de cero a 3 que indica el grupo, y finalmente el tercer caracter puede ser un número comprendido entre cero y siete que indica el bit de entrada correspondiente; así por ejemplo, la entrada correspondiente al bit 3 del grupo 2 puede ser indicada como "E23"; para cada grupo de entradas corresponde un puerto físico con una determinada dirección en el mapa de puertos de la CC, de esto se hablará un poco más en el capitulo 2.

1-1-3 Bloque de Salidas (BS)

Este bloque está realizado por dos puertos de salida de la CC, de modo que para cada uno de sus bits se cuenta con una interfaz a un relevador de baja potencia de contactos normalmente abiertos.

Todas las terminales comunes de contactos de los 16 relevadores están conectadas al punto VFA (vivo de la fuente de actuadores) en tanto que para cada relevador el otro contacto está conectado con su correspondiente terminal de salida asociada.

Las salidas se denotan con tres caracteres, el primero es la letra "s" como mayúscula o minúscula, el segundo es un número que puede ser cero o uno indicando esto el grupo al que pertenece la salida en cuestión, finalmente el tercer caracter es un número comprendido entre cero y siete que denota el número de bit de salida correspondiente; así por ejemplo, la salida 4 del grupo uno puede indicarse como "S14"

Al verificarse el nivel de uno lógico para una determinada salida habrá continuidad eléctrica entre las terminales VFA y la propia correspondiente con la salida, en otro caso no habrá continuidad eléctrica, la máxima corriente permisible, para disparo del actuador correspondiente, es 500 mA.

1-1-4 Bloque de Comando Local y Despliegue (BCLD)

Desde el punto de vista del usuario final este bloque está constituido por tres componentes, uno de ellos es la Unidad Desplegadora (UD) que maneja dos renglones de 16 caracteres, otro es un panel que contiene cinco postes que habilitan sendas entradas binarias auxiliares, cuatro botones para comando local y dos pares de postes donde se podrían colocar puentes (jumpers) que configurarían la manera en que el PLM respondería a una reinicialización del programa del usuario; el tercer componente del BCLD es un reloj de tiempo real, que puede servir simplemente como testigo de la hora o como base de tiempo para una función especial del dispositivo; que permite generar disparos a otros módulos lógicos para horarios predeterminados por el usuario en el programa en SIIL1, hecho de acuerdo con las necesidades de una determinada aplicación.

Para la implantación de la unidad desplegadora se utilizó el desplegado alfanumérico inteligente AND491 fabricado por la corporación electrónica PURDY; en lo que toca al reloj de tiempo real se empleó el chip MM58274N fabricado por NATIONAL que es una componente pensada para interconectarse con un microcontrolador o microprocesador.

Los botones y postes para entradas auxiliares o colocación de puentes, están validados por dos puertos de entrada (denotados como PAUXA y PAUXB) adicionales a los que forman parte de la arquitectura de la CMT SIMMP-2. Los cuatro botones están denotados como BAXA, BAXB, BAXC y BAXD; los tres primeros son usados para poner el reloj de tiempo real a una hora determinada, el último es usado como auxiliar en una de

las instrucciones que manejan la UD; los postes que presentan las cinco entradas auxiliares están denotados como EA1, EA2, EA3 y EA4 y EA5; los puentes sirven para configurar tipos de respuesta al restablecimiento y se denominan Ja y Jb. En la figura 1.2 se muestra un esquema a bloques del BCLD y en la tabla 1.1 se resumen, en lo general, las funciones de los botones de comando local, los puentes y las entradas binarias auxiliares; en el capitulo 3, al tratar los módulos lógicos que emplean el BCLD, se aborda con detalle lo descrito de manera resumida en la tabla 1.1.

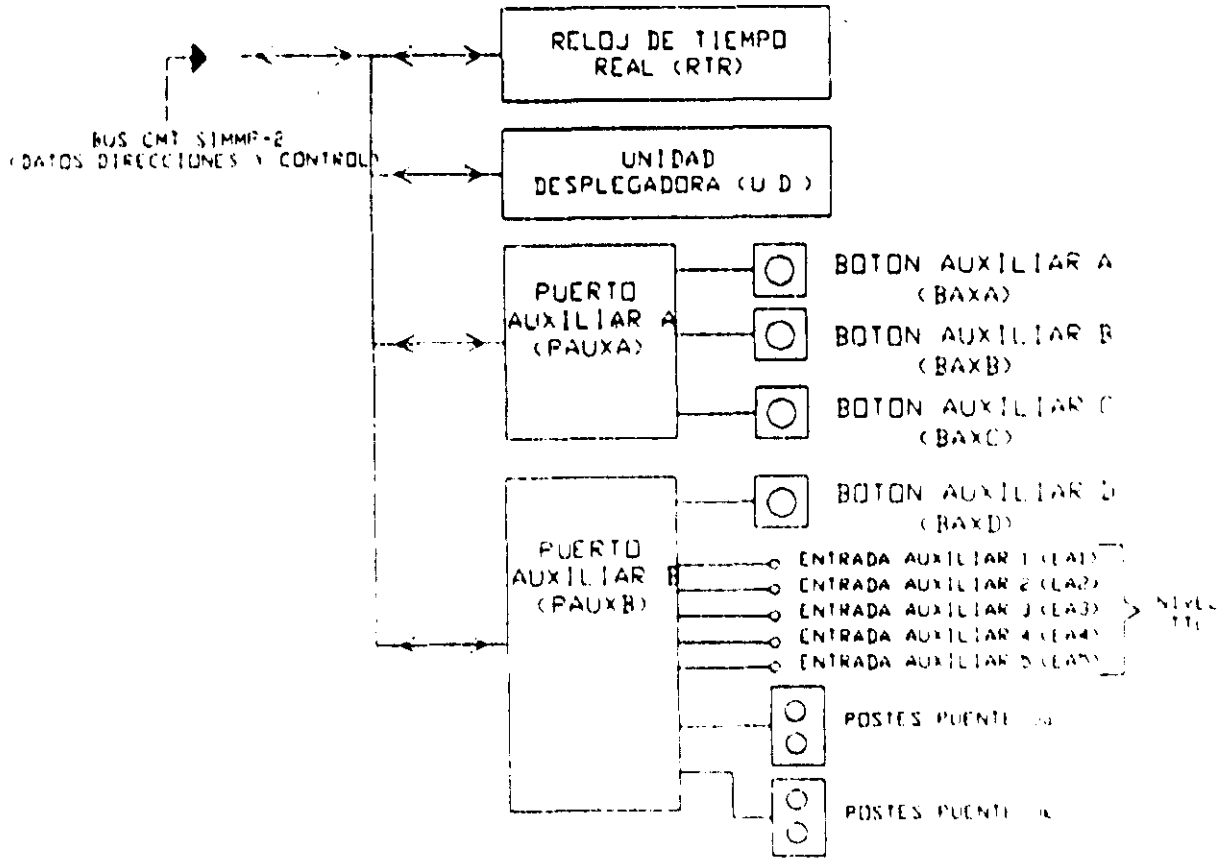


Figura 1.4 - Esquema a bloques del Bloque de Comando Local y Despliegue del PLM

Tabla 1.1 Resumen de funciones asociadas con instancias del BCLD, del PLM.

Instancia del Bloque de comando local y despliegue (BCLD)	Uso en el PLM desde el punto de vista del usuario final
Botones BAXA, BAXB y BAXC	Ajuste y puesta a tiempo del reloj de tiempo real (RTR)
Botón BAXD	Este botón se emplea para desplegar secuencialmente mensajes priorizados en la UD
Entradas auxiliares EA1 a EA5	Reservadas para funciones futuras que pudieran requerir botones o puentes
Puente Ja	Con Ja no colocado, al reinicializar el programa del usuario el reloj de tiempo real se pone en ceros (00:00:00), en otro caso el RTR conserva la hora al reinicializar el programa del usuario
Puente Jb	Con Jb no colocado, al reinicializar el programa del usuario, se ponen en cero todas las variables booleanas que use la aplicación, en otro caso las variables conservan el valor que tenían antes de la reinicialización

1-1-5 Fuente de alimentación (FA)

El PLM requiere para su funcionamiento de dos fuentes de voltaje, una de 12 volts y otra de 5 volts, la primera polariza únicamente a los relevadores del bloque de salidas y requiere de una capacidad de corriente de un Ampere; el requerimiento de corriente de la segunda fuente mencionada es de 500 mA; cabe señalar aquí que para el primer prototipo del PLM, reportado en esta tesis, la fuente de alimentación se implantó empleando una fuente comercial para laboratorio de electrónica.

1-2 VARIABLES BOOLEANAS EN EL PLM

Las entradas y salidas de los módulos lógicos que pueden ser realizados con el PLM son variables booleanas que son clasificadas como: variables booleanas de entrada (VBE), variables booleanas de salida (VBS) y variables booleanas intermediarias (VBI).

Dado que la información en el microcontrolador 68HC11 está organizada en bytes, las variables mencionadas aquí están aglutinadas en conjuntos (grupos) de ocho variables de

un mismo tipo; esto es, hay cuatro grupos de variables booleanas de entrada, dos grupos de variables booleanas de salida y 21 grupos de variables booleanas intermediarias; a continuación se describen conceptos asociados con cada uno de los tipos de variables binarias del dispositivo.

1-2-1 Variables Booleanas de Entrada (VBE)

Este tipo de variables están asociadas con sendas terminales de entrada siendo cada una de ellas optoacoplada a la CC, cada terminal de entrada puede recibir una señal lógica de voltaje (0-24 volts) proveniente de algún sensor, que sea parte del sistema de control lógico, que se requiera implantar en un momento dado.

El primer prototipo del PLM está pensado para manejar 32 VBE's y como se ha mencionado anteriormente cada VBE se especifica con tres caracteres, el primero es una letra "e" mayúscula o minúscula, el segundo es un número del cero al tres que denota el grupo al que pertenece la VBE y el tercero es un número del cero al siete que define el bit asociado del puerto de entrada relacionado con el grupo de entradas de que se trate ; así por ejemplo, la quinta variable del grupo de entradas dos se podría denotar como E25.

1-2-2 Variables Booleanas de Salida (VBS)

Existen para el PLM 16 variables booleanas de salida, cada una de ellas está asociada con un relevador de baja potencia cuyos contactos se cierran al presentar la variable de salida correspondiente el nivel de uno lógico, al ser cero lógico el valor en cuestión tales contactos permanecen abiertos; las VBS están aglutinadas en dos grupos de ocho salidas cada uno; de esta forma, se emplean tres caracteres para denotar a una VBS, el primero es la letra "s" mayúscula o minúscula, el segundo es un número que puede ser cero o uno que denota el número de grupo de salida y el tercero es un dígito del cero al siete que define el bit asociado con el puerto de salida físico de la CC relacionado; así por ejemplo, la salida dos del grupo de salidas cero se podría definir como S02.

1-2-3 Variables Booleanas Intermediarias (VBI)

Este tipo de variables son manejadas internamente y no tienen entradas o salidas fisicas asociadas, su función consiste en servir de enlace entre módulos lógicos cuando esto sea necesario; por ejemplo, supóngase que se tiene una situación de control lógico que requiere de varias compuertas lógicas, puede suceder que las salidas de algunas de ellas sean variables requeridas como entrada de otras, el emplear variables fisicas de salida para habilitar esta circunstancia no sería conveniente dado que su número es limitado, de ahí la necesidad de contar con las VBI; desde luego que una variable booleana, que sea entrada de un módulo y salida de otro, puede ser una salida fisica si esto es necesario, lo que obviamente no es permitido es el hecho de que una variable sea simultáneamente salida de más de un módulo.

Las VBI están aglutinadas en 21 grupos de ocho variables cada uno, de esta forma se puede contar dentro del PLM con 168 variables de este tipo; la notación empleada para designarlas emplea cuatro caracteres, el primero es la letra "i" mayúscula o minúscula, el segundo y tercero representan a un número comprendido entre cero y veinte que denota el número de grupo al que pertenece la VBI y el cuarto es un dígito del cero al siete que define el número de VBI dentro del grupo; así por ejemplo, la VBI cuatro del grupo doce de VBI's se denominaría I124.

1-3 DESCRIPCIÓN GENERAL DE LOS MÓDULOS LÓGICOS

Los módulos lógicos (ML) que puede realizar el PLM constituyen los bloques funcionales elementales para la realización de aplicaciones de control lógico y pueden ser representados a nivel de "caja negra" como se muestra en la figura 1.5, donde se muestra un ML que presenta "m" entradas y "n" salidas; m y n varian de acuerdo con el tipo de función que un determinado ML realice; así por ejemplo, para una compuerta AND de tres entradas m y n serian tres y uno respectivamente; en cambio, un secuenciador de estados con palabras de cuatro bits requerirá tres entradas y cinco salidas; para mayor detalle acerca de los módulos secuenciadores pude consultarse el capitulo tres de esta tesis.

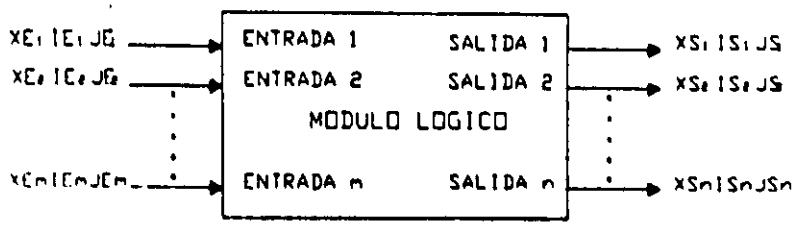


Figura 1.5 Representación de bloque de un módulo lógico de m entradas y n salidas

En la figura 1.5 XEk representaría a un caracter que podría ser cualquiera de las letras e ,i o s mayúsculas o minúsculas dependiendo esto del tipo de varaiable (VBE, VBI o VBS) asociada con la entrada k-esima del ML; IEk y JEk serían respectivamente los números asociados con el grupo y el número de bit correspondientes con la variable k-esima de entrada; XSk representaría a un caracter que podría ser cualquiera de las letras i o s mayúsculas o minúsculas dependiendo esto del tipo de variable (VBI o VBS) asociada con la salida k-esima del ML; ISk y JSk serían respectivamente los números asociados con el grupo y el número de bit correspondientes con la variable k-esima de salida. Por ejemplo, una compuerta NAND de tres entradas con negación en una de ellas se muestra en la figura 1.6, las entradas son respectivamente las variables E01, I45 (entrada negada) y E13, la salida es la variable S02.

Figura 1.6.- Representación de un Módulo Lógico que realiza una compuerta NAND de tres entradas con negación en una de ellas, nótese que las entradas pueden ser de grupos diferentes.

Para todos los ML que validan compuertas lógicas se tiene que los mismos responden al nivel que presenten sus entradas; sin embargo, algunos de los ML que no son compuertas están diseñados de modo que responden a flancos que se presenten en una o

varias de sus entradas; en la figura 1.7 se muestra un ML que se encuentra en este caso, se trata de un temporizador de tipo monodisparo (one-shot) que presentará en su salida (variable S14) un pulso verificado alto de una duración determinada, cada vez que en la entrada de disparo (variable E12) se manifieste un flanco de bajada, en lo que toca a la otra entrada (variable E02) el ML responde al nivel, cuando el mismo es alto el temporizador está habilitado, en caso de que el nivel sea bajo se retorna la salida a su nivel no verificado no respondiendo el módulo a los disparos hasta que la entrada mencionada retorne a el nivel alto.

A nivel de los esquemas de bloques asociados con los ML la sensibilidad a flancos de una entrada es denotada mediante una flecha vertical cuyo sentido indica el tipo de flanco asociado, véase la figura 1.7.

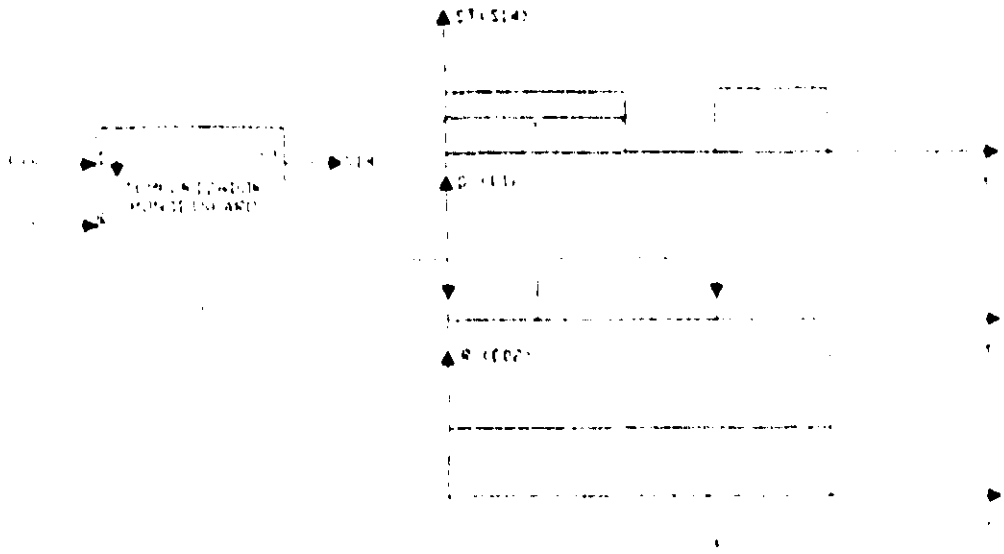


Figura 1.7 a) Representación en forma de bloque de un temporizador monodisparo (one-shot), nótese la flecha vertical indicando que el disparo es por flanco de bajada.
b) Diagrama de tiempos correspondiente al temporizador mostrado.

1-4 FORMAS SINTÁCTICAS ASOCIADAS CON LOS MÓDULOS LÓGICOS

Cada uno de los módulos lógicos, requeridos en una determinada aplicación, deben haber sido declarados secuencialmente en un archivo de texto, para que el mismo sea

procesado por una computadora anfitriona empleando un programa denominado SIIL1 (Software de Interpretación de Instrucciones Lógicas), que genera el código objeto que ha de ejecutar la CC del PLM de modo que los ML requeridos queden realizados; además de las declaraciones asociadas con los módulos, en el archivo mencionado se requieren colocar otras *instrucciones*, no relacionadas directamente con algún ML, pero necesarias para la ejecución adecuada del programa que ha de ejecutarse en el PLM, de esto se hablará más adelante.

Al conjunto de instrucciones, mencionadas anteriormente, escritas en secuencia en un archivo de texto, se le denomina *programa fuente en lenguaje SIIL1* asociado con la aplicación que ha de realizar el PLM. A excepción de los módulos que requieren datos adicionales que ha de proporcionar el usuario, la forma sintáctica de las declaraciones asociadas con los mismos requiere de un solo renglón en el archivo de texto a procesar, esta forma se ilustra a continuación:

CODM#N ENT$_1$, ENT$_2$, ..., ENTm, SAL$_1$, SAL$_2$, ..., SALn, DA$_1$,...DAq, ...,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 ML de un mismo tipo que use una aplicación, deben ser numerados.

ENT$_1$ a ENTm son las designaciones asociadas con las m variables de entrada que el módulo requiera.

SAL$_1$ a SALn son las designaciones asociadas con las n salidas que pudiera tener el módulo.

DA$_1$ a Daq 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 bien el número de estados que ha de presentar un secuenciador, etc. Hay módulos que no requieren de estas especificaciones, tal es el caso de 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 especifica diversas características de funcionamiento como podrían ser: que entradas a una compuerta van a tener negación implícita, a que tipo de flanco responde una entrada de algún otro tipo de módulo, etc. En el capítulo tres de esta tesis se trata en detalle para cada módulo las características que en cada caso define cada caracter (1 ó 0), que integra la cadena CADBI que corresponda.

Cabe señalar que el primer caracter de la instrucción nunca deberá estar en la primera columna y que al final de la misma siempre ha de colocarse el caracter ";".

Como ejemplo de estructura sintáctica, a continuación se muestra la instrucción asociada con la declaración de la compuerta NAND de tres entradas mostrada en la figura 1.6.

NAND3#1 E01, I45, E13, S02,101;

En la instrucción anterior CODM es la palabra NAND3 que denota el hecho de que se trata de una compuerta NAND de tres entradas; por ejemplo, si se hubiera tratado de una compuerta NAND de cuatro entradas CODM hubiera sido NAND4.

Nótese además que la cadena binaria asociada (CADBI) consta de tres bits, ya que la compuerta es de tres entradas, indicándose que la entrada I45 deberá tener negación implícita colocando un cero en la posición que corresponde a tal entrada; así, si se requiriera que tuvieran negación implícita las dos primeras entradas (E01 e I45) CADBI sería 001.

Se aprecia también en la declaración anterior que se le ha asignado el número uno a la compuerta NAND en cuestión.

1-5 FORMATO DE UN PROGRAMA EN SIIL1.

1-5-1 Características generales de la ejecución de un programa en SIIL1 en la CC del PLM

Al correr un programa en SIIL1 en la CC del PLM, el código asociado con cada ML es ejecutado cíclicamente siguiendo la siguiente secuencia:

1.- Se copian en un buffer de entrada (BE) en RAM el estado que guardan los cuatro puertos asociados con las 32 entradas 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, tomándose del BE las entradas que cada módulo requiera, las salidas que se fueran generando son colocadas en un buffer de salida (BS) en RAM; si el ML emplea una o varias VBI como entradas los valores asociados con las mismas son tomados de un buffer intermediario (BI) en RAM, en caso de que haya en el ML una o varias salidas de tipo VBI los valores correspondientes son escritos en el BI.

3.- Se copia el estado del BS en los puertos físicos asociados con las VBS.

4.- Se regresa al paso uno.

De lo anterior se aprecia que el código asociado con cada módulo es ejecutado repetitivamente, variando el intervalo de repetición de acuerdo con el número módulos que contenga el programa; esto es, a mayor número de módulos crece el periodo de repetición.

Existen módulos que requieren que el periodo de repetición de la ejecución de su código asociado sea constante (10 ms), 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 10 ms, empleándose para ello facilidades de temporización con que cuenta la CC del PLM.

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 un párrafo anterior, a esta parte se le llama *subprograma principal*, la otra parte está constituida por el código cuya ejecución es temporizada y se denomina *subprograma temporizado*. En la figura 1.8 se ilustra esta idea. Cabe señalar aquí que todo programa en SIIL1 debe tener un subprograma principal; sin embargo, puede haber programas que no contengan un subprograma temporizado.

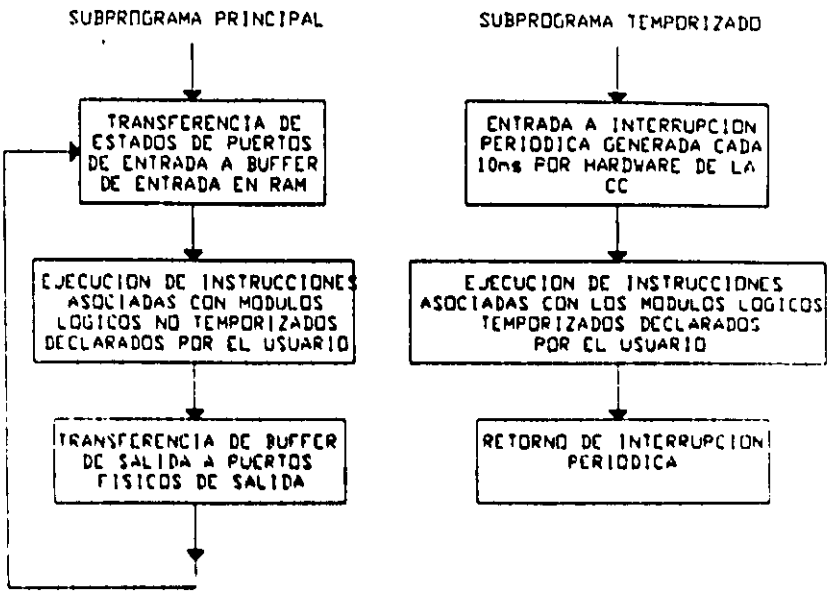


Figura 1.8.- Ejecución, en la CC del PLM, de los dos subprogramas que integran un programa en SIIL1.

1-5-2 Forma de un programa fuente en SIIL1

El programa fuente en SIIL1 asociado con una aplicación está constituido por declaraciones, que pueden ser comandos o instrucciones; los comandos son indicaciones tales como inicio o fin del subprograma principal, tipo de mapa de memoria empleado en la CC, inicio o fin de instrucciones asociadas con el subprograma temporizado; las instrucciones son declaraciones asociadas con características que han de tener los módulos empleados por la aplicación y deben respetar la sintaxis descrita en párrafos anteriores. En lo general un programa fuente en SIIL1 está integrado por la siguiente secuencia de declaraciones:

1.- Comando que indica el tipo de configuración de funcionamiento, la sintaxis asociada es CONFIGN, donde N es un número entero que puede ser uno, dos, o tres; de esta manera, existen a la fecha de elaboración de este trabajo de tesis, tres posibles configuraciones de funcionamiento para el PLM. En la tabla 1.2 se resumen las características principales de funcionamiento asociadas con cada configuración.

2.- Comando que marca el inicio del subprograma principal, la sintaxis correspondiente en este caso es INPROG

3.- Instrucciones asociadas con los módulos que se desea integren al subprograma principal.

4.- Comando que marca el fin del subprograma principal, la sintaxis en este caso es FINPP.

5.- Comando que marca el inicio del subprograma temporizado, la sintaxis asociada es INMODI.

6.- Instrucciones asociadas con los módulos que han de integrar al subprograma temporizado.

7.- Comando que indica el fin del subprograma temporizado, la sintaxis en este caso es FINMODI.

Los siete componentes del programa fuente han de ser colocados respetando el orden anterior; al igual que en el caso de las instrucciones asociadas con los módulos, los comandos deberán ser concluidos con el caracter ";". Todo texto colocado a la derecha del caracter ";" no es tomado en cuenta por el programa que genera el código objeto, de esta manera pueden adicionarse comentarios al programa fuente.

Tabla 1.2 Resumen de características de funcionamiento del PLM asociadas con las diferentes configuraciones de funcionamiento.

Configuración	Entradas	Salidas	Máximo tamaño del programa (kb)
1	32	16	7.5
2 *	8	8	7.5
3	32	16	24

* Esta configuración se empleó para la prueba de los módulos con lógica nivel TTL.

1-6 METODOLOGÍA A SEGUIR PARA ESTRUCTURAR UNA APLICACIÓN DE CONTROL LÓGICO EMPLEANDO EL PLM

Toda aplicación de control lógico puede integrarse empleando tres conjuntos de elementos funcionales denominados como:

1.- Elementos sensores, los cuales son dispositivos que presentan en su salida un nivel lógico que testifica un determinado evento como podrían ser por ejemplo el paso de un producto por una banda transportadora, el fin del recorrido de un embolo, el que un operador oprima

un botón, etc. Los sensores pueden estar constituidos desde por simples interruptores hasta por bloques que basan su funcionamiento en componentes electrónicos. Frecuentemente en la industria los niveles lógicos empleados son cero y 24 volts.

2.- Elementos lógicos, los cuales son dispositivos que realizan funciones booleanas cuyas entradas son las variables presentadas por los sensores, siendo sus salidas variables lógicas que comandan a elementos actuadores, de modo que fisicamente se realicen los eventos que la aplicación requiera.

3.- Elementos actuadores, que son dispositivos que actúan directamente sobre el proceso y son comandados por las salidas que generan los elementos lógicos mencionados en el párrafo anterior, ejemplos de actuadores podrían ser resistencias eléctricas que suministren calor, motores eléctricos que muevan elementos mecánicos de diversa índole, etc.

El papel del PLM en la realización de un sistema de control lógico es el de implantar los elementos lógicos que la aplicación requiera, empleando para ello a los módulos lógicos que el mismo puede realizar virtualmente.

A continuación se describe el proceso a seguir para que el conjunto de módulos lógicos necesarios en un control lógico tomen forma en el PLM, se supone que el PLM debe operar en modo esclavo y debe estar convenientemente enlazado con una computadora anfitriona de tipo PC, desde luego que el desarrollo completo debe contemplar lo relacionado con los sensores y actuadores; sin embargo, aquí se habla únicamente acerca de lo concerniente al PLM que es el objeto de este trabajo de tesis. En síntesis los pasos a seguir son los siguientes:

1 - Definir los módulos lógicos que la aplicación requiera, especificando para cada uno las variables de entrada y salida empleadas respetando el hecho de que una variable no puede ser salida de más de un módulo; de lo anterior escribir el programa fuente asociado siguiendo los lineamientos dados en el tema 1-5, empleando para ello a un editor de texto convencional que se ejecute en la computadora anfitriona.

2 - Guardar el texto generado en el paso anterior en un archivo con un nombre escogido por el usuario y con la extensión SIL.

3 - Ejecutar en la computadora anfitriona el Software de Interpretación de Instrucciones Lógicas (programa SIIL1.EXE) tomando como archivo de entrada el generado en el paso

anterior; en caso de haber errores de sintaxis en las declaraciones mencionadas en el paso uno se mostrarán en pantalla los mismos.

En caso que no haya errores se indicará esto, generándose además un archivo binario con el mismo nombre dado por el usuario en el paso dos y con la extensión BLM; este último archivo contendrá el código ejecutable por el PLM correspondiente a la aplicación que se esté desarrollando.

4.- Si se detectaron errores en el paso anterior corregirlos en el editor de texto y regresar al paso dos. Si no hubo errores de sintaxis proceder al paso cinco.

5.- Transferir para ejecución, a la memoria RAM del PLM, el código contenido en el archivo BLM generado en el paso tres, esto puede hacerse empleando el manejador hexadecimal PUMMA propio de la tarjeta SIMMP-2, véase el apéndice A.

6.- En caso de que el programa no opere correctamente en el PLM hacer las cambios necesarias, a nivel de los módulos lógicos empleados por la aplicación, y regresar al paso uno. Si el programa opera correctamente proceder al siguiente paso.

7 - Desenergizar el PLM, colocar una EPROM borrada en la base correspondiente, así como los puentes J4 y J5 en la CC (tarjeta SIMMP-2), energizar el PLM, oprimir y soltar el botón de restablecimiento del mismo.

8 - Programar la EPROM con el código objeto contenido en el archivo BLM generado en el paso tres. Para efectuar esto se puede emplear el manejador hexadecimal PUMMA, propio de la tarjeta SIMMP-2, véase el apéndice A.

9 - Quitar los puentes J4 y J5 y validar el modo autónomo de operación, esto último se hace colocando el puente J11 en la CC (CMT SIMMP-2).

10.- Oprimir y soltar el botón de restablecimiento del PLM, al hacer esto deberá ejecutarse en forma autónoma el programa del usuario, de esta manera el sistema de control lógico diseñado funcionará siendo realizado por el PLM.

1-7 EJEMPLO 1.1

Para aclarar algunos de los conceptos descritos en este capítulo, se muestra aquí como se programaría el PLM para realizar la situación de control lógico mencionada en la introducción de esta tesis. En la figura 1.9 se muestra un esquema a bloques que emplea

módulos propios del PLM, para realizar la situación de control lógico mostrada en la figura I.1, se supone que la bomba que suministra el reactivo B debe operar durante treinta segundos.

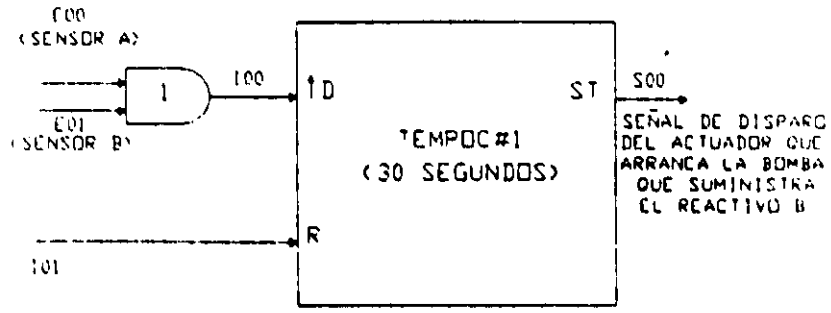


Figura 1.9 Esquema a bloques de una posible realización de la situación de control lógico, mostrada en la figura 1.1, empleando el PLM.

El temporizador requerido en este ejemplo es de tipo mono disparo (one shot) como el ilustrado en la figura 1.7, para claridad en este ejemplo, la sintaxis genérica asociada con un ML que realice un temporizador como el requerido se muestra a continuación:

TEMPOC #N DISPARO, HABILITACIÓN, SALIDA, DURACIÓN, ABC;

Donde:

N representa el número de temporizador.

DISPARO denota la variable booleana que dispara al temporizador.

HABILITACIÓN denota a la variable booleana asociada con la habilitación y restablecimiento del temporizador.

SALIDA denota a la variable booleana asociada con la salida del temporizador

DURACIÓN denota el tiempo, que ha de especificarse en horas minutos segundos y centésimas de segundo de acuerdo con el formato 00:00:00.00; por ejemplo, si se desea que el pulso dure dos horas cuarenta y cinco minutos con diez segundos, esto se especificará como 02:45:10.00, en caso de que el tiempo dure ya sea menos de una hora o de un minuto se deberá denotar con ceros los espacios correspondientes a las horas y/o minutos según sea el caso

A es un caracter que podrá ser cero si se desea que el disparo sea por flanco de bajado o uno si se desea que el disparo sea por flanco de subida.

B es un caracter que podrá ser cero si se desea que la habilitación sea por nivel alto y el restablecimiento sea por nivel bajo; en caso de que se requiera que la habilitación sea por nivel bajo y el restablecimiento por nivel alto, B deberá ser cero. Para que el temporizador responda a los disparos la habilitación del mismo deberá estar verificada, en caso de verificarse el restablecimiento mientras se verifica el pulso de salida, el mismo regresará a su nivel no verificado, véase la figura 1.7b.

C es un caracter que podrá ser cero si se desea que el pulso de salida tenga verificación en bajo y uno en caso de que se desee que tal verificación sea en alto.

Nótese en la figura 1.9 el empleo de dos variables booleanas intermediarias, aclarándose aquí que el valor por defecto de una variable booleana del PLM es cero, siendo esta la causa de que el nivel requerido para la señal de habilitación sea bajo, apreciándose el empleo, como delimitadora, de la variable intermediaria I01.

El programa correspondiente en SIIL1 es:

```
                Programa asociado con el ejemplo 1.1
        CONFIG1; declaración de configuración de funcionamiento
        INPPROG; declaración de inicio de subprograma principal
La siguiente línea corresponde a la instrucción asociada con el módulo que realiza
la compuerta lógica requerida
        AND2#1 E00, E01, I00, 111; compuerta and número 1
        FINPP; declaración de fin de subprograma principal
        INMODI; declaración de inicio de subprograma temporizado
        TEMPOC#1 I00, I01, S00, 00:00:30.00, 101; temporizador
        FINMODI; declaración de fin de subprograma temporizado
```

Para mayores detalles acerca de la sintaxis asociada con los módulos de este ejemplo o bien con el funcionamiento de los mismos, puede consultarse el capítulo tres de esta tesis

En la figura 1.10 se muestra el conexionado al PLM de los sensores y actuadores asociados con el ejemplo aquí descrito.

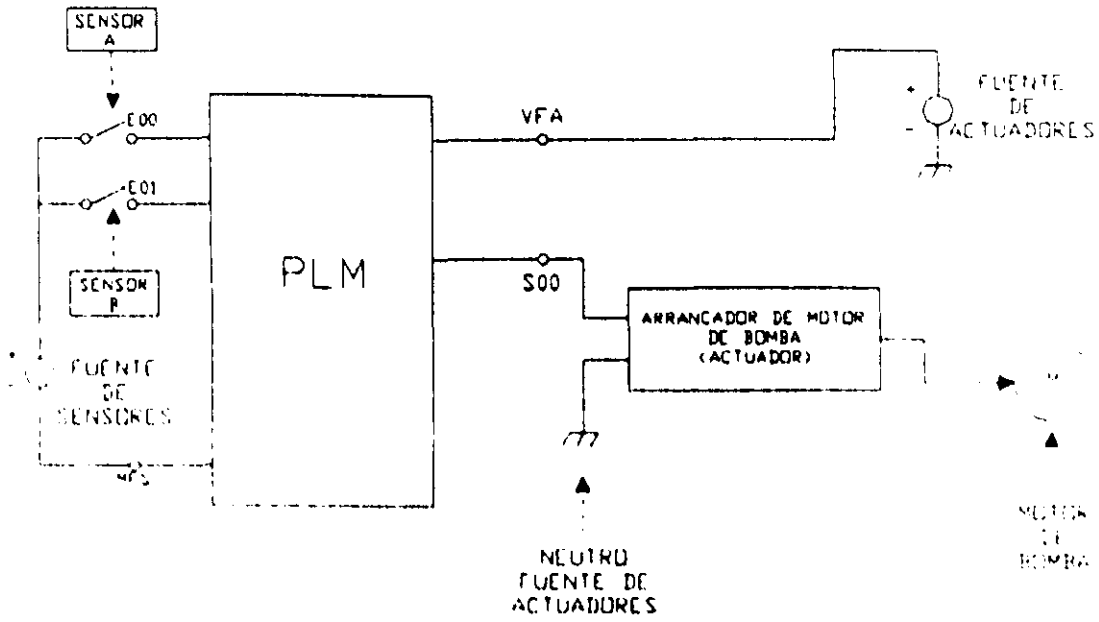


Figura 1.10 .- Conexionado de los sensores y actuadores, asociados con la situación de control lógico mostrada en la figura I.1, cuando la misma es realizada por el PLM, de acuerdo al diagrama de bloques mostrado en la figura 1.9

CAPÍTULO 2
HARDWARE DEL PLM

En este capitulo se describe la circuiteria asociada con cuatro de los cinco bloques funcionales que integran al PLM, iniciando con la computadora central, continuando con los circuitos de optoacoplamiento asociados con cada una de las 32 entradas fisicas, pasando por los circuitos de cada una de las 16 salidas fisicas, concluyendo con los circuitos correspondientes al bloque de comando local y despliegue (BCLD); el quinto bloque (fuente de alimentación) no se detalla por las razones mencionadas en el capitulo anterior.

2-1 COMPUTADORA CENTRAL (CC)

La computadora central está realizada por la tarjeta SIMMP-2 que es una computadora monotablilla basada en el microcontrolador 68HC11F1, fabricado por Motorola, en la figura 2.1 se muestra este chip. Como ya se ha mencionado en el capitulo anterior, la CMT SIMMP-2 puede operar en cualquiera de los cuatro modos asociados con este microcontrolador y como CC del PLM opera en modo expandido, pudiéndose en este caso validar los mapas de memoria requeridos por cada una de las configuraciones de funcionamiento del PLM, véase la tabla 1.2 del capitulo 1.

En las figura 2.2 se muestra una fotografia de la tarjeta SIMMP-2, apreciándose un diagrama de bloques de la misma en la figura 2.1 del apéndice A. En ese mismo apéndice se detallan diversos aspectos relacionados con los bloques funcionales de la tarjeta.

En este capitulo se detallan circuitos y conceptos de la CMT SIMMP-2 relacionados con el funcionamiento de la misma como CC del PLM. Parte de la información aqui descrita es complementada en el apendice A.

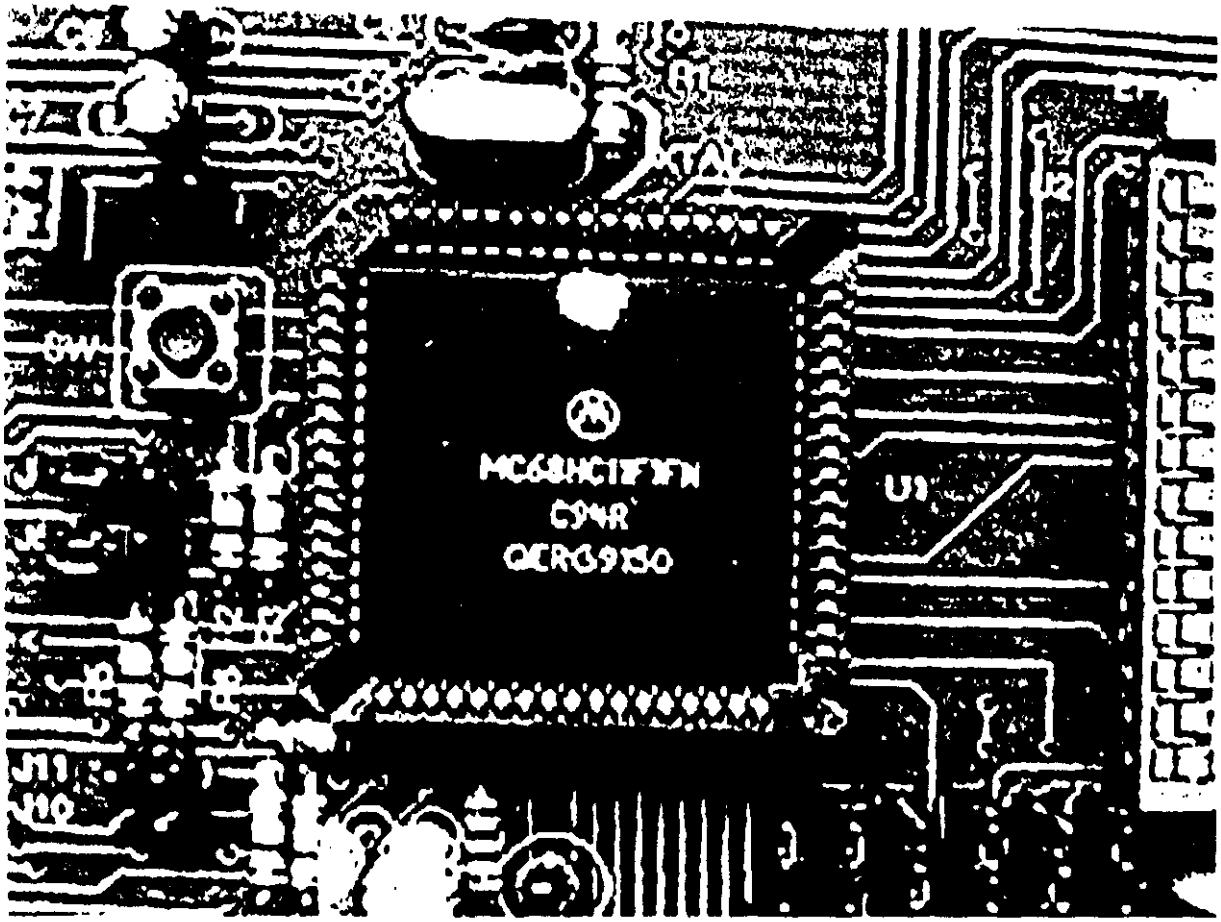


Figura 2.1 - Chip microcontrolador 68HC11F1 (CPU de tarjeta SIMMP-2)

2-1-1 Microcontrolador 68HC11F1

Este componente es una computadora digital completa contenida en un solo circuito integrado, el chip contiene además periféricos de mucha utilidad en instrumentación y control, las componentes funcionales del 68HC11F1 son a grandes rasgos las siguientes

1 - Tres puertos bidireccionales con capacidad de definir por software, el comportamiento, como entrada o salida, asociado con cada uno de sus bits, estos puertos se denotan con las letras A, C, D y G. Al operar en modo expandido se pierden el puerto C que pasa a ser el bus de datos y las lineas del *nibble* alto del puerto G que pudieran ser empleadas para paginación de memoria y puertos

2 - Dos puertos de salida denotados con las letras B y F, que se pierden al operar en modo expandido, ya que pasan a ser respectivamente los bytes alto y bajo del bus de direcciones

3.- Un puerto de entrada, denotado por la letra E cuyas lineas pasan a ser los ocho canales de entrada analógica asociados con el convertidor analógico digital contenido en el chip, cuando el mismo es habilitado.

4.- Un convertidor analógico digital de ocho bits y ocho canales, que puede ser habilitado y leido por software.

5.- Un puerto serie asincrono, que puede operarse en los baudajes más comunmente empleados, al usarse este medio de comunicación entre computadoras.

6.- Un puerto serie sincrono, que puede ser usado para comunicarse con periféricos que intercambian información empleando un formato serie.

7.- Un temporizador, que puede entre otras funciones, generar interrupciones periódicas a intervalos de repetición dependientes de la frecuencia de reloj asociada.

De los bloques y caracteristicas descritas anteriormente, las más importantes para el funcionamiento de la CMT SIMMP-2 como CC del PLM son: el puerto serie asincrono, el temporizador y la capacidad de paginación de memoria y puertos; para más información acerca del 68HC11F1 se pueden consultar las hojas de datos técnicos asociados con el mismo que son proporcionados por la compañía Motorola.

2-1-2 Paginación de Puertos en la CMT SIMMP-2

La lógica empleada para la paginación de puertos de la CMT SIMMP-2 se muestra en la figura 2.3, la misma está conformada por un decodificador de tres a ocho (74LS138) cuyas salidas Y0 a Y7, verificadas en bajo, validan sendas lineas de paginación de puerto con un intervalo asociado de 128 direcciones, asi la linea Y0 decodifica al intervalo de direcciones 1800- 187F, la linea Y1 decodifica al intervalo 1880- 18FF, la linea Y2 decodifica al intervalo 1900- 197F, y asi sucesivamente hasta la linea Y7 que decodifica al intervalo 1B80- 1BFF.

Las lineas de entrada al decodificador son:

1.- CSIO2, que es una linea de control del microcontrolador que decodifica un intervalo de 2k direcciones comprendidas en el intervalo 1800-1FFF, esto se logra gracias a firmware que es parte de las rutinas de inicialización de la CMT SIMMP-2 al operar en modo expandido

2 - Lineas de dirección del microcontrolador comprendidas de la A7 a la A10, actuando sobre entradas de control del decodificador

Del esquema mostrado en la figura 2.3, es fácil comprobar los intervalos de direcciones asociados con cada línea de salida del sistema de paginación de puertos. En la nomenclatura propia de la tarjeta SIMMP-2 las líneas de paginación de puerto Y0 a Y7 son también denotadas como CSP0 a CSP7.

Para más información acerca de el mapa de puertos que valida el sistema de paginación de puertos puede consultarse el apendice A.

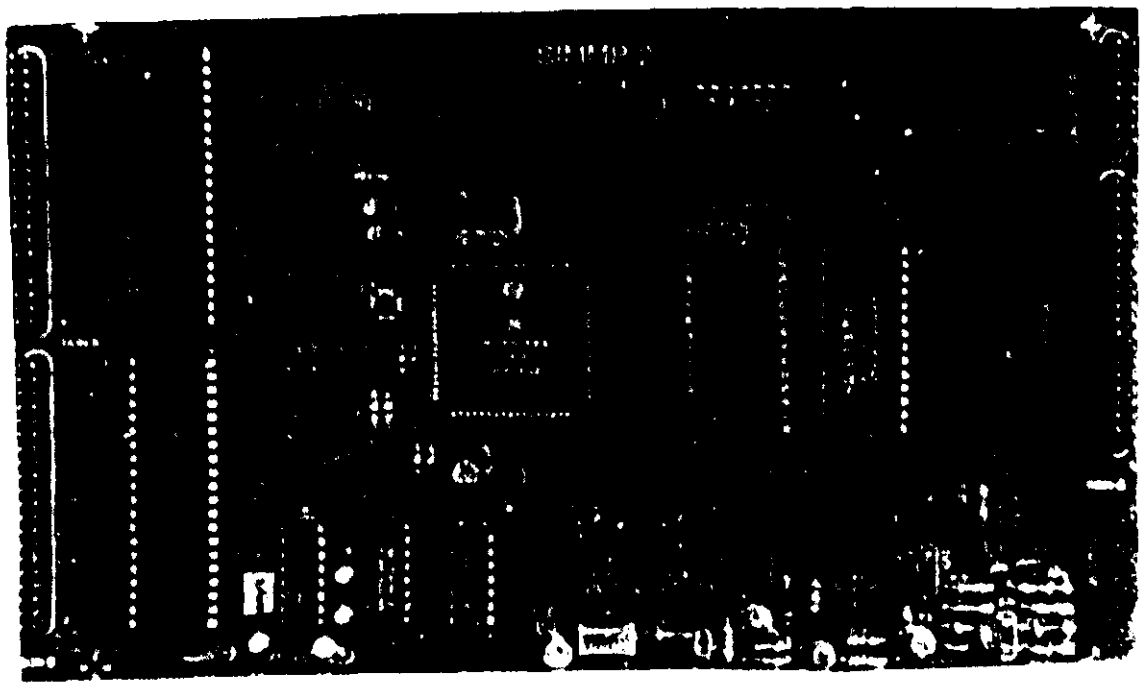


Figura 2.2 Tarjeta SIMMP-2, empleada para realizar la computadora central del PLM

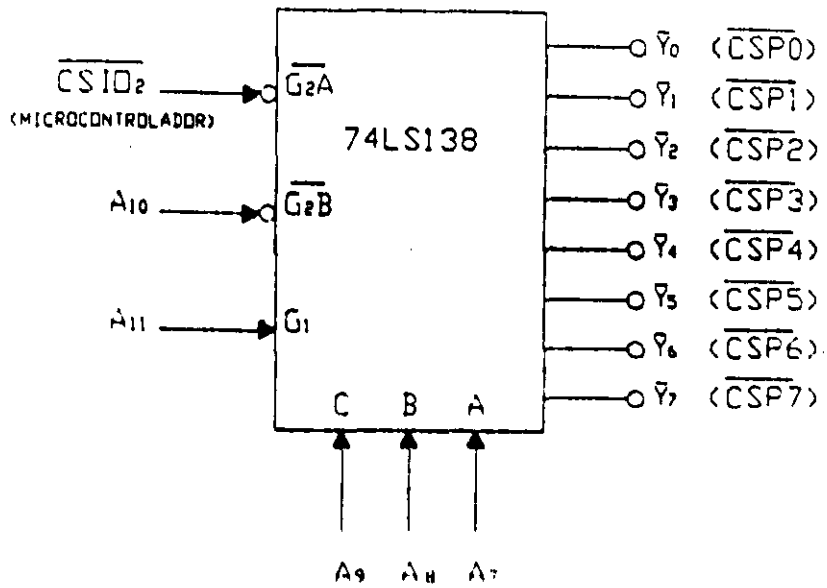


Figura 2.3.- Sistema lógico para paginación de puertos contenido en la CMT SIMMP-2

2-1-3 Lógica M-I (Motorola - INTEL)

La tarjeta SIMMP-2 incorpora circuitería lógica que genera señalización de lectura escritura compatible con periféricos no fabricados por Motorola, tales como el 82C55 fabricado por INTEL, que es un puerto triple programable y el chip de reloj de tiempo real MM58274 fabricado por National. La tarjeta SIMMP-2 contiene dos chips 82C55 que son empleados para realizar las entradas y salidas binarias del PLM. Además el BCLD del mismo emplea el chip de reloj aquí mencionado.

En la figura 2.4 se muestra el sistema lógico generador de señalización de lectura escritura de tipo INTEL.

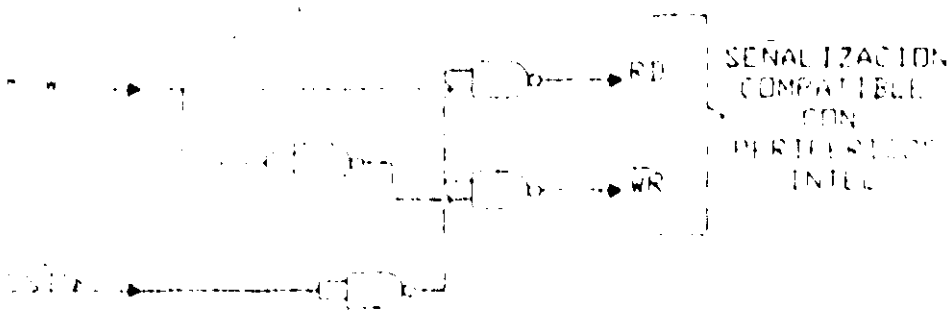


Figura 2.4 - Sistema lógico contenido en la CMT SIMMP-2 para generar señalización de lectura escritura compatible con periféricos INTEL.

2-1-4 Puertos paralelos adicionales de la CMT SIMMP-2

La tarjeta SIMMP-2 cuenta con dos conjuntos de puertos paralelos denominados como CTPP1 y CTPP2, cada uno de ellos está constituido por un chip 82C55 que es un componente de uso común en la industria y como es conocido agrupa a tres puertos que pueden programarse como entradas o salidas mediante software.

De esta manera potencialmente se cuenta dentro de la tarjeta hasta con seis puertos paralelos adicionales, cuyas terminales se encuentran presentes en postes para conexionado de cable plano presentes en la tarjeta, para mayor detalle acerca de esto último puede consultarse el apéndice A y la hoja de datos del 82C55 proporcionada por INTEL, aquí basta mencionar que para fines de la CC del PLM el chip es operado en modo cero programándose como entradas los puertos A y B del mismo y como salida el puerto C correspondiente, de esta forma se logran las 32 entradas y 16 salidas requeridas.

Las direcciones de puerto asociadas pueden verse en la figura 2.11 del apéndice A, a continuación se indica que puertos corresponden con cada uno de los grupos de entradas y salidas del PLM.

Entradas 00 a 07 ----------Puerto A de CTPP1 dirección $1800
Entradas 10 a 17 ----------Puerto B de CTPP1 dirección $1801
Entradas 20 a 27 ----------Puerto A de CTPP2 dirección $1880
Entradas 30 a 37 ----------Puerto B de CTPP2 dirección $1881
Salidas 00 a 07 ----------Puerto C de CTPP1 dirección $1802
Salidas 10 a 17 ----------Puerto C de CTPP2 dirección $1882

En la figura 2.5 se muestra en forma genérica el conexionado, al bus de la tarjeta SIMMP-2, de los chips 82C55 que realizan los conjuntos triples de puertos CTPP1 y CTPP2, empleados por la CC del PLM para validar los puertos físicos de entrada y salida requeridos por el mismo.

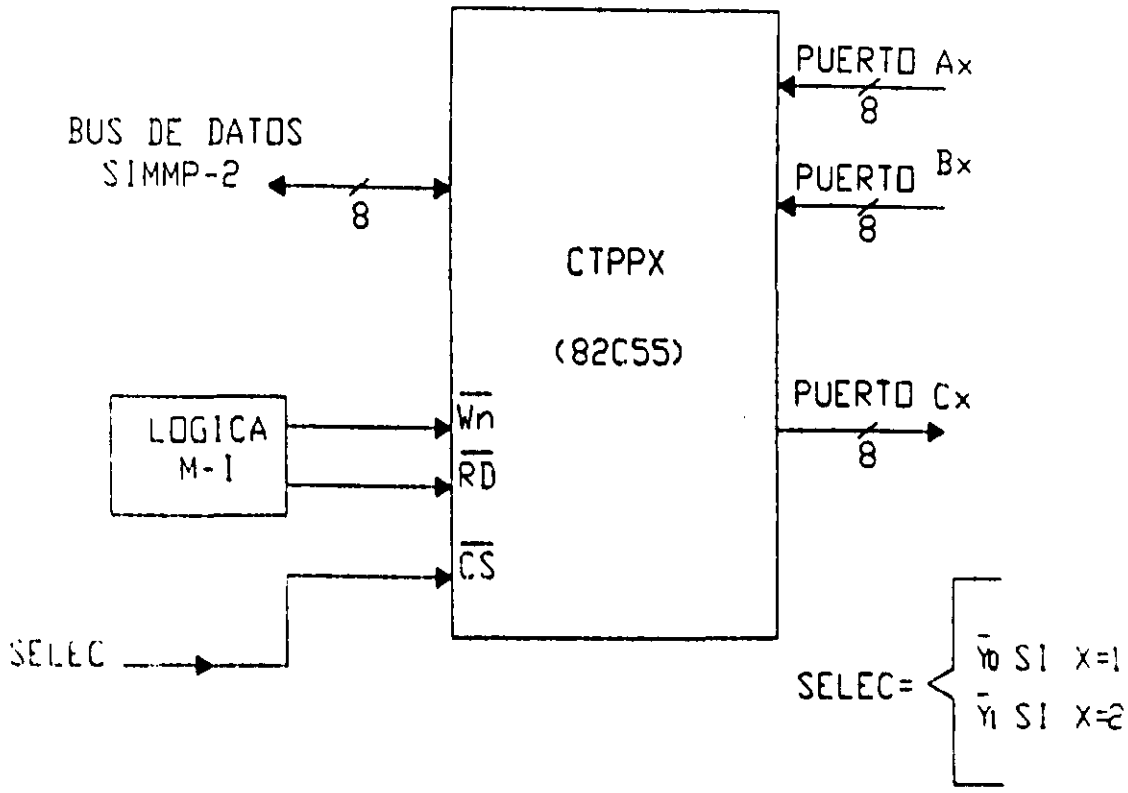


Figura 2.5.- Conexionado de los chips 82C55 al bus de la tarjeta SIMMP-2

2-1-5 Memoria EPROM

La tarjeta SIMMP-2 puede recibir para programación a EPROM's de 2kb a 64kb, para ejecutar código desde las mismas se recomienda emplear memorias cuyo tamaño este comprendido entre 8kb y 64kb. Para fines de la CC del PLM, de acuerdo con las configuraciones de funcionamiento del mismo, vigentes a la fecha de escribir esta tesis, las EPROM's requeridas son de 8kb para las configuraciones 1y 2 y 32kb para la configuración 3.

En la figura 2.6 se muestra el conexionado de la memoria EPROM al bus de la terjeta SIMMP-2

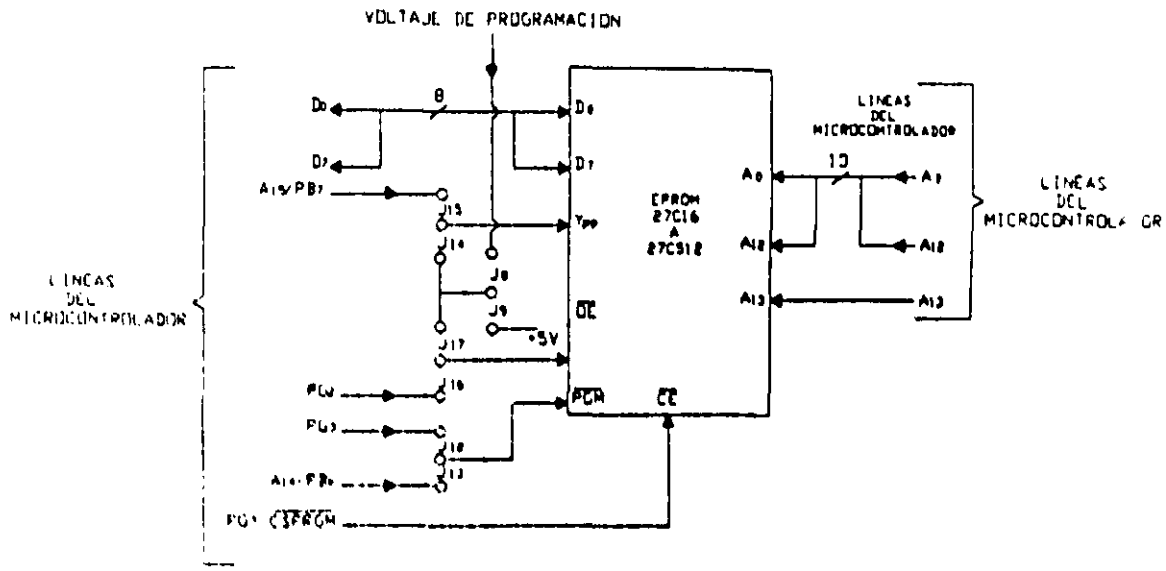


Figura 2.6.- Conexionado de la EPROM al bus de la tarjeta SIMMP-2

Los puentes que se aprecian en la figura 2.6 son requeridos por el hardware y software de programación de las EPROM's, además es conveniente mencionar aquí el hecho de que el firmware de restablecimiento de la tarjeta SIMMP-2 reside en la memoria EEPROM interna del microcontrolador, encargándose el mismo de inicializar las líneas PG2 y PG3 de la figura 2.6, para que se pueda ejecutar código desde la memoria EPROM colocada.

La disposición de puentes para programación y/o lectura de diferentes EPROM's, puede consultarse en el apendice A.

2-1-6 Memoria RAM

La tarjeta SIMMP-2 puede recibir memorias RAM estáticas de 8kb y 32kb, que tamaño es requerido en un momento dado, depende de el tipo de mapa de memoria con el que se desee trabajar, en la figura 2.7 se muestra el conexionado de la RAM al bus de la tarjeta.

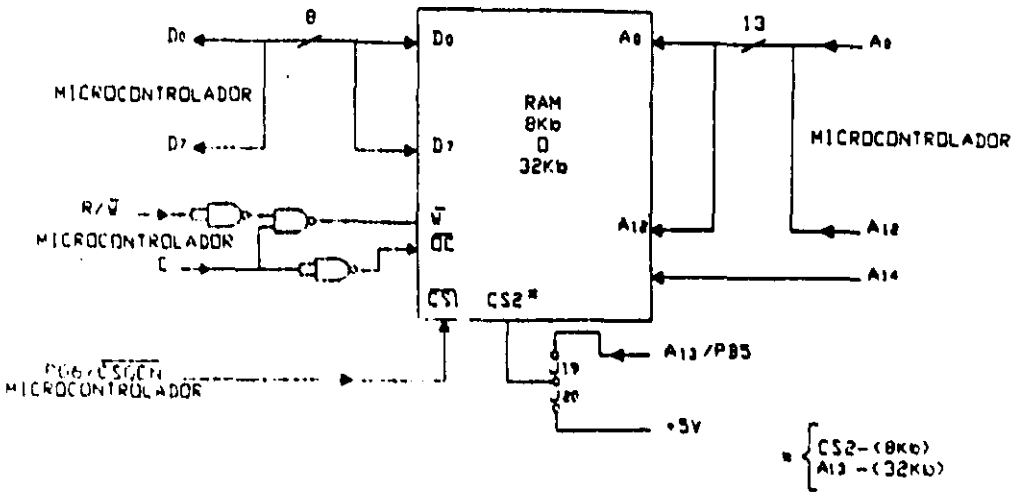


Figura 2.7.- Conexionado genérico de la memoria RAM al bus de la tarjeta SIMMP-2

Los diversos mapas de memoria en que puede operar la CMT SIMMP-2 se configuran mediante colocación de puentes, al operar en modo expandido pueden configurarse seis mapas de memoria diferentes; en el apendice A se indica, para cada mapa de memoria, la configuración de puentes requerida.

Para fines de la CC del PLM y las configuraciones de funcionamiento del mismo los mapas de memoria requeridos se muestran en la tabla 2.1.

Tabla 2.1 Mapas de memoria de la CMT SIMMP-2 empleados por las configuraciones de funcionamiento del PLM.

Configuración de Funcionamiento PLM	Mapa Empleado.	Tamaño de Memorias.	Colocación de Puertos.
1 y 2	EA	8 Kb RAM 8Kb EPROM	J3, J9, J12, J14, J16, J20, J22
3	EB	32 Kb RAM 32 Kb EPROM	J3, J9, J13, J14, J16 J19, J22

2-1-7 Circuitería para programación de memorias EPROM

La tarjeta SIMMP-2 cuenta con un bloque funcional que permite la programación de memorias EPROM, en la figura 2.8 se aprecia el mismo, para fines del PLM únicamente interesaría programar EPROM's de 8kb y 32kb para las configuraciones de funcionamiento del mismo a la fecha de escritura de esta tesis.

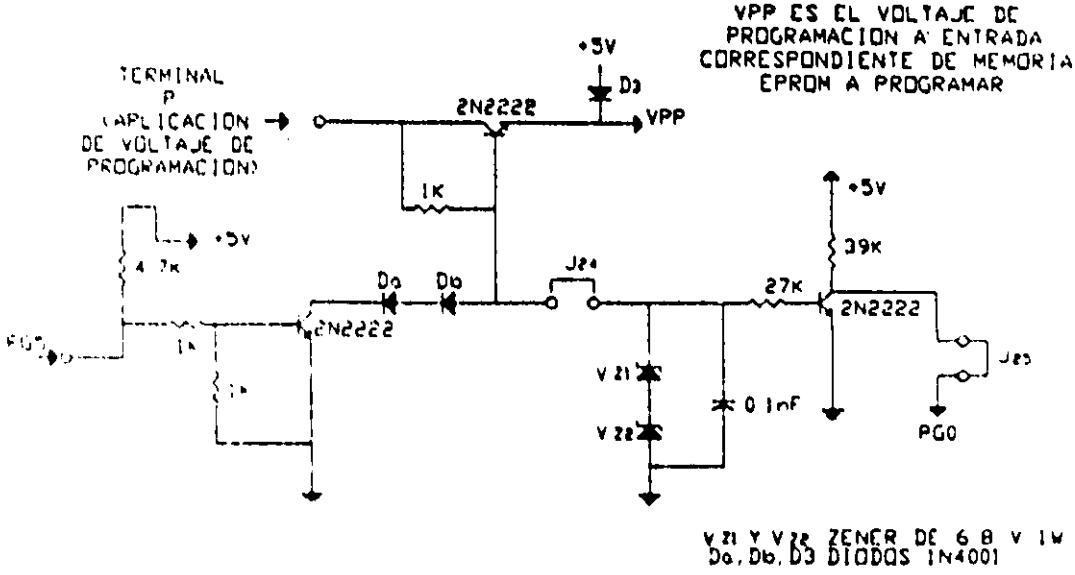


Figura 2.8.- Circuito en la CMT SIMMP-2 que soporta físicamente la programación de memorias EPROM.

Como se aprecia en la figura 2.8 la conmutación del voltaje de programación es arbitrada electrónicamente por el propio microcontrolador, empleando para ello a las líneas PG5 (programada como salida) y PG0 (programada como entrada); los diodos ZENER fijan el voltaje de programación Vpp en 12.5 volts (puente J24 colocado) siempre que el voltaje en la terminal "P" sea de 14 o más volts, en caso de que el voltaje de programación requerido sea mayor que este valor, se deberá quitar el puente J24 y aplicar el voltaje requerido más un volt por la terminal "P".

En el apéndice A se explican los pasos a seguir, por parte del usuario final, para programar EPROM's con la tarjeta SIMMP-2 empleando herramientas de software propias de la misma.

2-2 CIRCUITOS DE OPTOACOPLAMIENTO DE ENTRADA

Para cada una de las 32 entradas del PLM, existe un circuito de optoacoplamiento que cambia los niveles lógicos de entrada (0-24volts) a niveles TTL propios de los puertos de entrada de la CC, además de proporcionar aislamiento eléctrico entre la circuitería asociada con los sensores y la CC, que opera a cinco volts; en la figura 2.9 se muestra el circuito genérico de optoacoplamiento empleado.

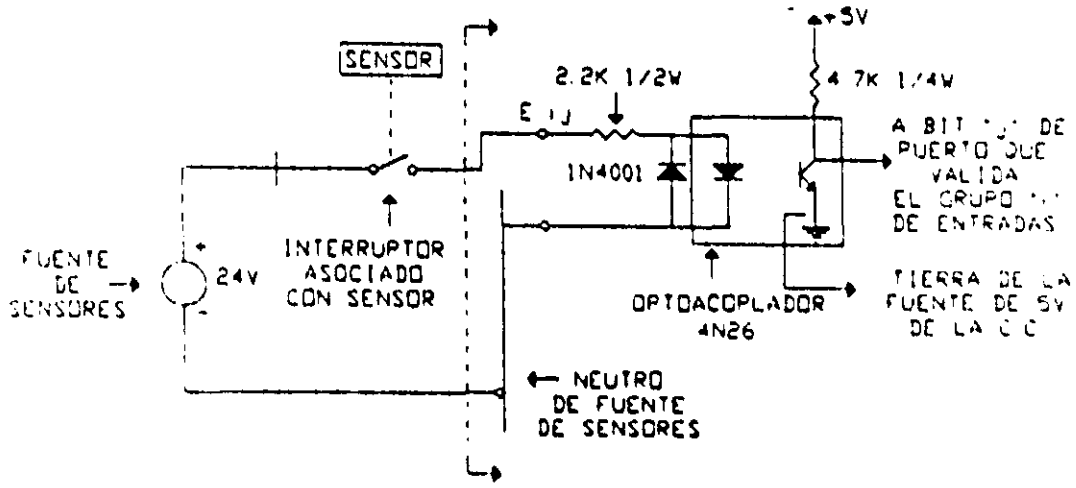


Figura 2.9.- Circuito de optoacoplamiento genérico empleado para cada una de las 32 entradas del PLM

Cabe señalar que se supone que la fuente de sensores debe tener una regulación adecuada, de modo que el voltaje suministrado por la misma no baje más de un 10% de su valor nominal (24 volts), en el peor caso (32 entradas conectadas con sus correspondientes interruptores de sensor cerrados), se aprecia además que hay una inversión lógica en cada circuito de optoacoplamiento, esto es compensado por software al momento de que la CC lee cada uno de los puertos asociados con cada grupo de entradas.

Físicamente los 32 optoacopladores están agrupados en dos tarjetas, alambradas a mano, conteniendo cada una de ellas 16 circuitos de entrada; en la figura 2.10 se muestra la fotografía de una de estas tarjetas, se aprecian en la misma transistores adicionales (32, dos por cada entrada), empleándose los mismos para realizar por hardware la inversión mencionada en el párrafo anterior y para suministrar corriente a sendos LEDS testigo asociados con cada entrada, la experiencia demostró que se podía hacer la inversión por software y que el LED testigo no es indispensable, aunque estas

primeras tarjetas de entradas pueden utilizarse para validar el primer prototipo del PLM, tomándose realmente en cuenta el circuito simple de la figura 9 al reproducir el PLM.

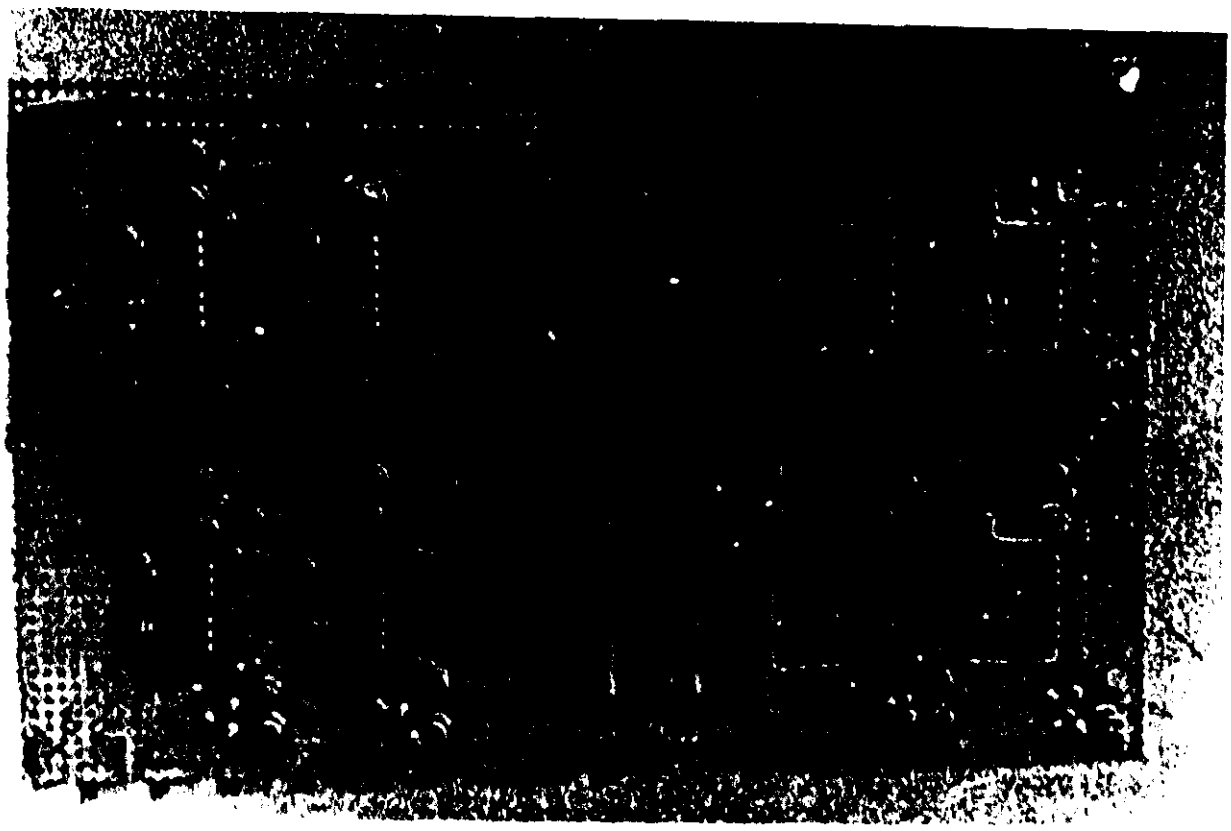


Figura 2.10 - Tarjeta de entrada del PLM que realiza 16 entradas del mismo (E00 a E17)

2-3 CIRCUITOS DE INTERFAZADO A RELEVADORES DE SALIDA

Las salidas del PLM se manifiestan físicamente mediante la apertura (salida cero lógico) o cerradura (salida uno lógico) de los contactos de sendos relevadores de baja potencia, para ello se emplearon relevadores con voltaje de activación de doce volts, para la bobina de los mismos, si bien la máxima corriente que pueden soportar los contactos es del orden de seis Amperes, se recomienda para el primer prototipo del PLM, que dicha corriente no exceda los 500 mA, esto se debe a que la tarjeta que agrupa los 16 relevadores asociadas con las salidas físicas fue tambien alambrada a mano, habiendose empleado conductores de bajo calibre para este fin.

En la figura 2.11 se observa el circuito de interfazado para las salidas del PLM y en la figura 2.12 se muestra una fotografía de la tarjeta que agrupa las 16 salidas del PLM empleada en el primer prototipo del mismo.

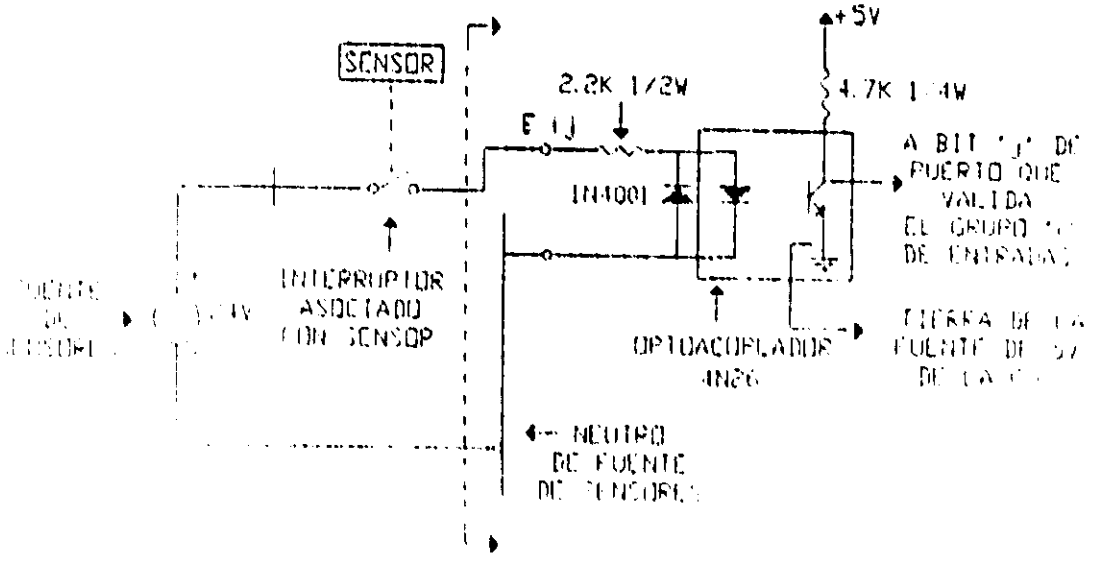


Figura 2.11.- Circuito de interfazado para la salida física Sij del PLM

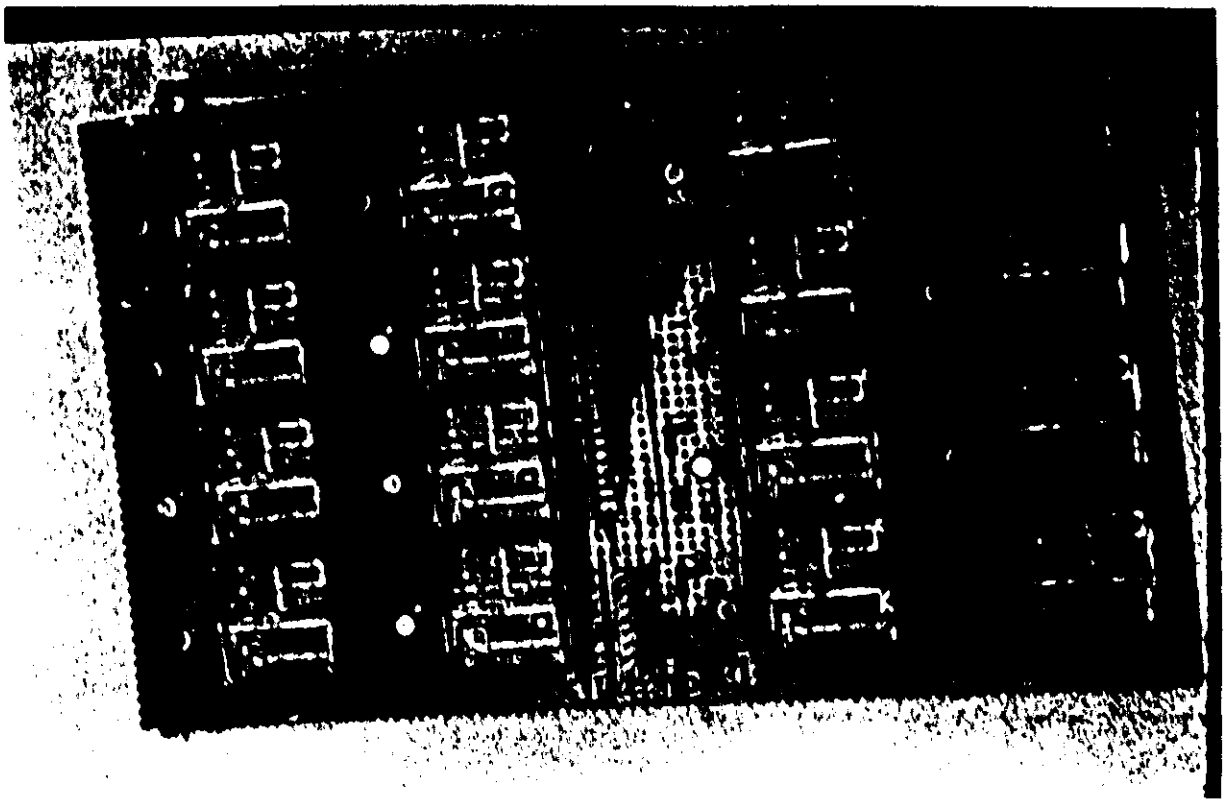


Figura 2.12 Fotografía que muestra la tarjeta de salidas del PLM

2-4 BLOQUES FUNCIONALES ASOCIADOS CON EL BCLD

En el tema 1-1-4 del capítulo uno se describe, a nivel del usuario final, la forma en que está integrado el Bloque de Comando Local y Despliegue (BCLD), en las figuras 2.13 (a) y 2.13 (b) se muestra el esquema electrónico del mismo apreciándose en la primera el subpaginador de puertos requerido y el reloj de tiempo real empleado, en la segunda figura se muestra el conexionado de los dos puertos auxiliares y la Unidad de Despliegue.

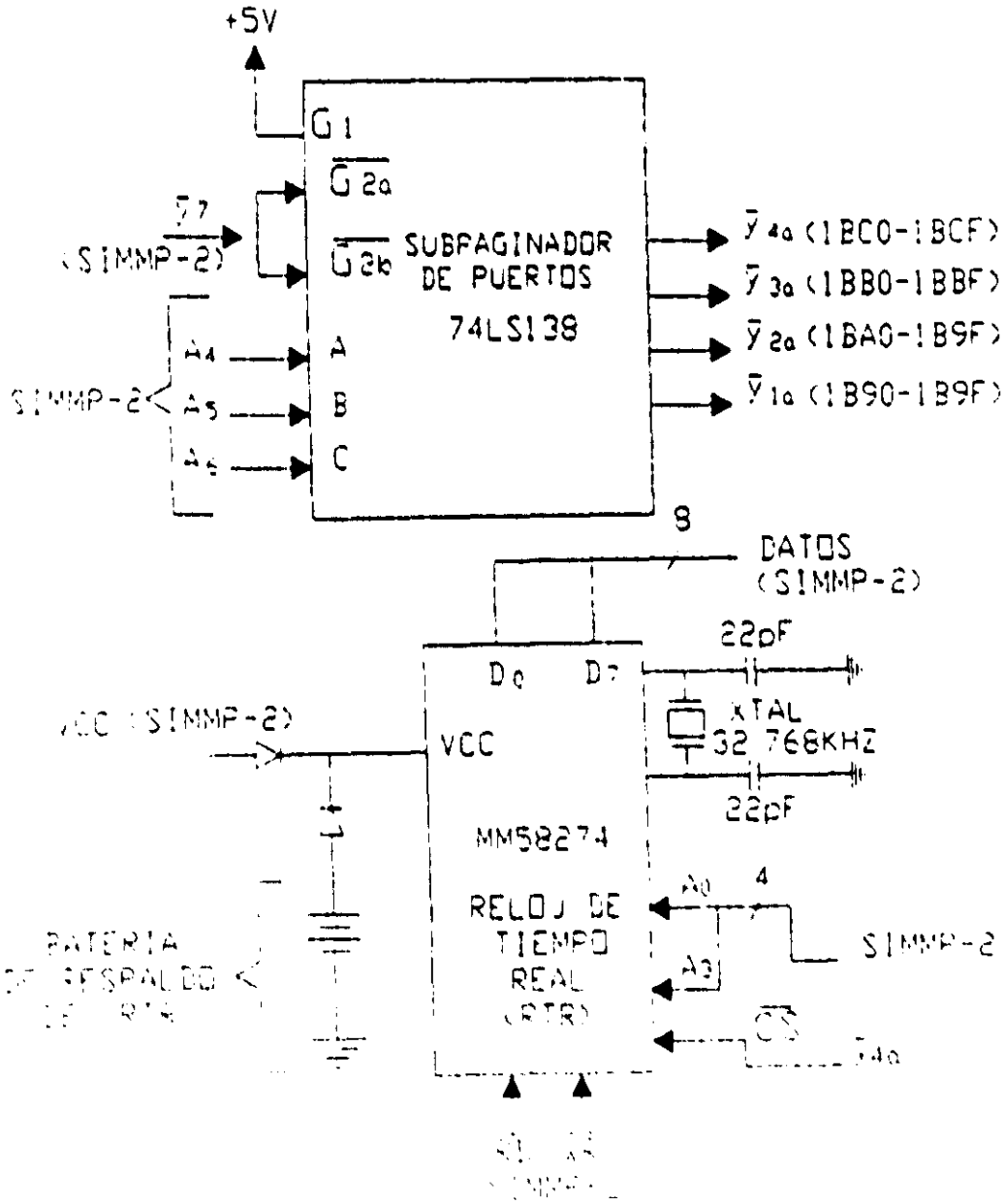


Figura 2.13 (a) Conexionado de el subpaginador de puertos y el reloj de tiempo real del BCLD.

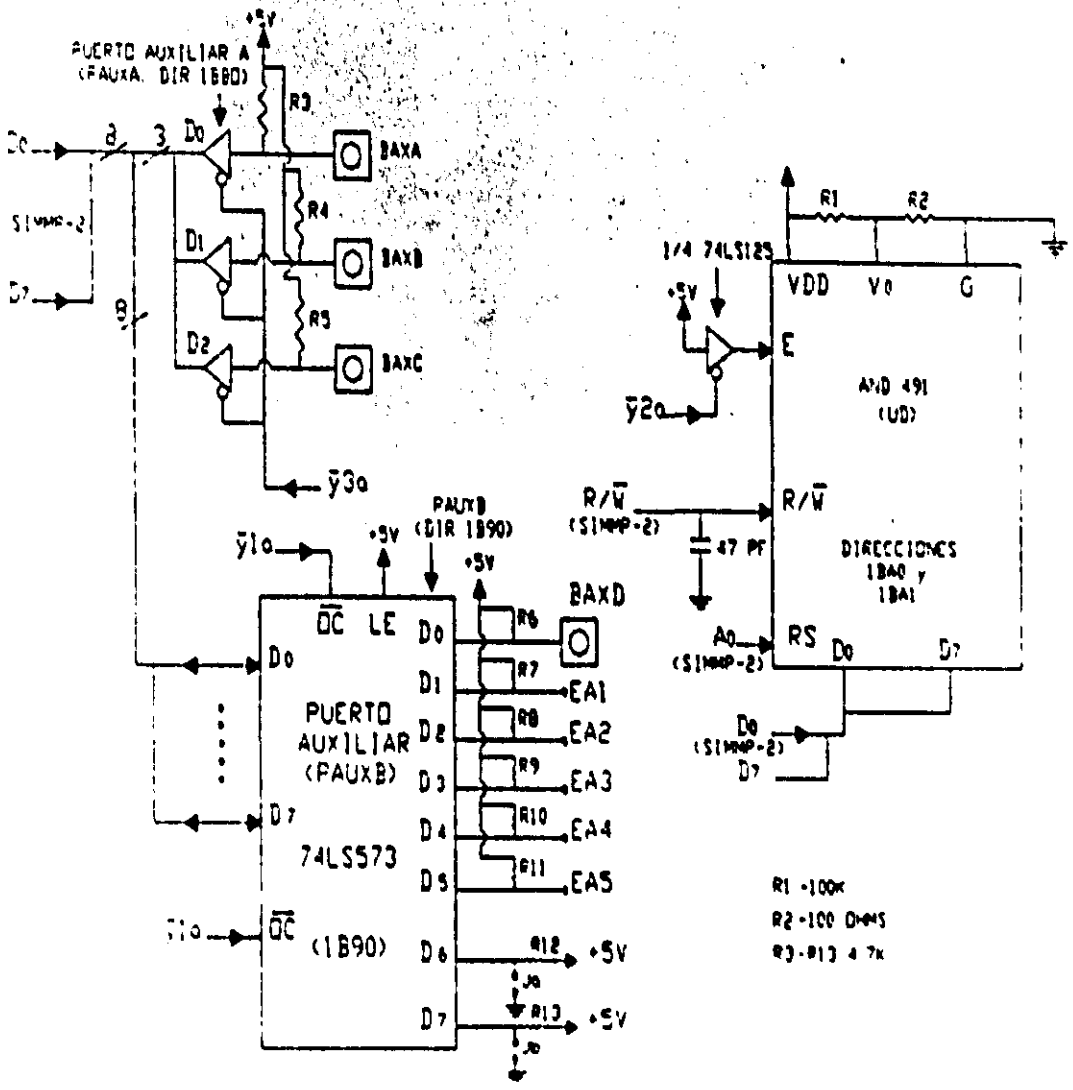


Figura 2.13 (b) Conexionado de los puertos auxiliares y Unidad Desplegadora del BCLD.

En la figura 2.14 se muestra una primera versión del BCLD alambrada en *proto-board* para fines de prueba.

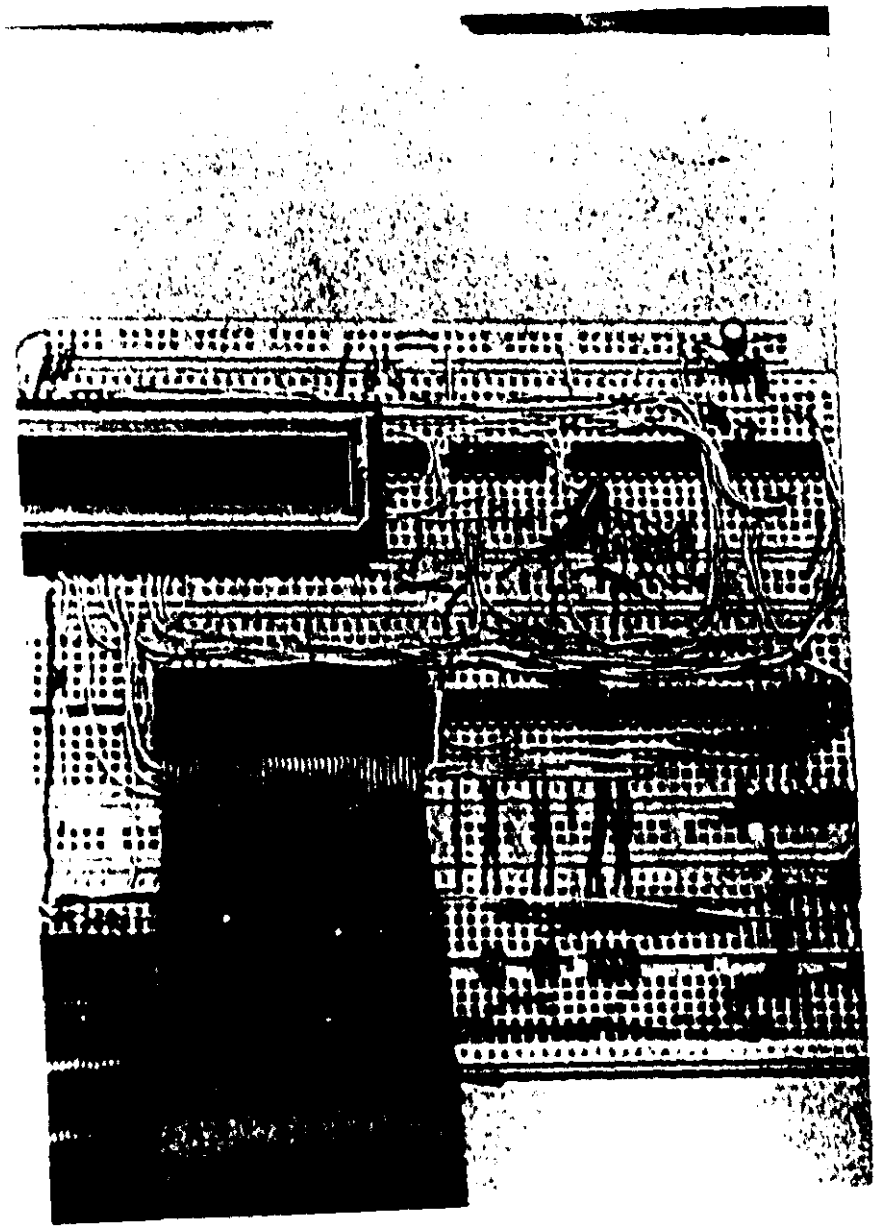


Figura 2.14 - Fotografía del BCLD alambrado para fines experimentales

A continuación se describen los componentes funcionales del BCLD

2-4-1 Subpaginador de puertos

Este bloque del BCLD esta constituido por un decodificador de tres a ocho (74LS138), cada una de las salidas del mismo validan sendas señales de habilitacion que se verifican en bajo para un intervalo de 16 direcciones, vease la figura 2.10 para mejor comprension de lo anterior, puede confrontarse la informacion de la tabla de funcionamiento del decodificador con las señales de entrada al mismo mostradas en las

figuras 2.13 (a) y (b) y con la descripción del mapa de puertos y señales de habilitación asociadas para la tarjeta SIMMP-2 descrita en el apéndice A.

2-4-2 Puerto Auxiliar A (PAUXA)

El puerto auxiliar A es un puerto de entrada de tres bits, implantado con tres de los cuatro seguidores binarios con salida de tercer estado, contenidos en el chip 74LS125, la dirección asociada con este puerto puede ser cualquiera de las comprendidas en el intervalo 1BB0 a 1BBF, las entradas correspondientes están asociadas con los botones auxiliares A, B y C que son empleados por el software que controla la puesta a tiempo del reloj de tiempo real, esto último se explicará en el capítulo tres de esta tesis.

2-4-3 Puerto Auxiliar B

Este puerto de entrada está implantado por el circuito integrado 74LS573, que es un *latch* octal con salida de tercer estado, en la figura puede apreciarse que las direcciones asociadas con este puerto son las comprendidas en el intervalo 1B90 a 1B9F, observándose que el bit menos significativo está ligado con el botón auxiliar D, que se usa para desplegar secuencialmente mensajes priorizados, que pudieran haber sido programados por el usuario en el programa fuente en SIIL1 de acuerdo con necesidades propias de la aplicación que el PLM estuviera realizando en un momento dado, para más detalles acerca de esto último puede consultarse, en el capítulo tres, el formato y funcionamiento de la instrucción "mensajero". Las líneas de entrada uno a la cinco de este puerto están ligadas con las entradas auxiliares EA1 a EA5, por último, las entradas seis y siete están asociadas con los puentes Ja y Jb respectivamente, en la tabla 1.1 se describe la finalidad de estos últimos.

2-4-4 Unidad Desplegadora (UD)

La unidad desplegadora tiene una capacidad de dos renglones de 16 caracteres cada uno y está realizada por el desplegado AND491 fabricado por la corporación Purdy Electronics, en la figura 2.13 (b) se aprecia el conexionado de la UD a la tarjeta SIMMP-2, en el capítulo tres se detallan aspectos de software que corre en el microcontrolador de la CC, al ejecutarse instrucciones de SIIL1 que empleen la UD del PLM

2-4-5 Reloj de Tiempo Real (RTR)

El reloj de tiempo real del PLM es implantado con el chip MM58274 fabricado por National; este chip está pensado para interfazarse con microcontroladores o microprocesadores, siendo la base de tiempo del mismo generada mediante el auxilio de un cristal externo de 32768 Hertz, este CI tiene capacidad para manejar un formato de tiempo en horas - minutos - segundos - décimas de segundo, además de manejar dias, meses y años. en la figura 2.13 (a) se puede observar el conexionado del RTR a la tarjeta SIMMP-2, apreciándose ahí el empleo de una batería de respaldo de modo que el RTR pueda seguir operando aún con el PLM desenergizado.

En el capítulo tres se detallan aspectos de software que corre en el microcontrolador de la CC, al ejecutarse instrucciones de SIIL1 que usen el RTR.

2-5 INTEGRACIÓN FÍSICA DE LAS COMPONENTES DEL PLM

Para este primer prototipo del PLM se agruparon en forma paralela las tarjetas que lo integran, empleando para ello tornillos sin fin (husillos) para así poder emplear tuercas, arandelas de hule y rondanas convencionales para sujetar las tarjetas, los postes están fijados a una base metálica que forma parte de una caja con tapa, la misma se colocó en un tablero donde se colocaron las tiras de terminales correspondientes a las terminales fisicas de entrada y salida.

En las figuras 2.15 y 2.16, se muestran fotografías donde se observan respectivamente, el arreglo de tarjetas que realizan los bloques funcionales del PLM y el mismo arreglo colocado en la caja empleada como gabinete, en la figura 2.17 se aprecia el PLM en su caja fijada en el tablero, observándose las tiras de terminales correspondientes con las entradas y salidas fisicas.

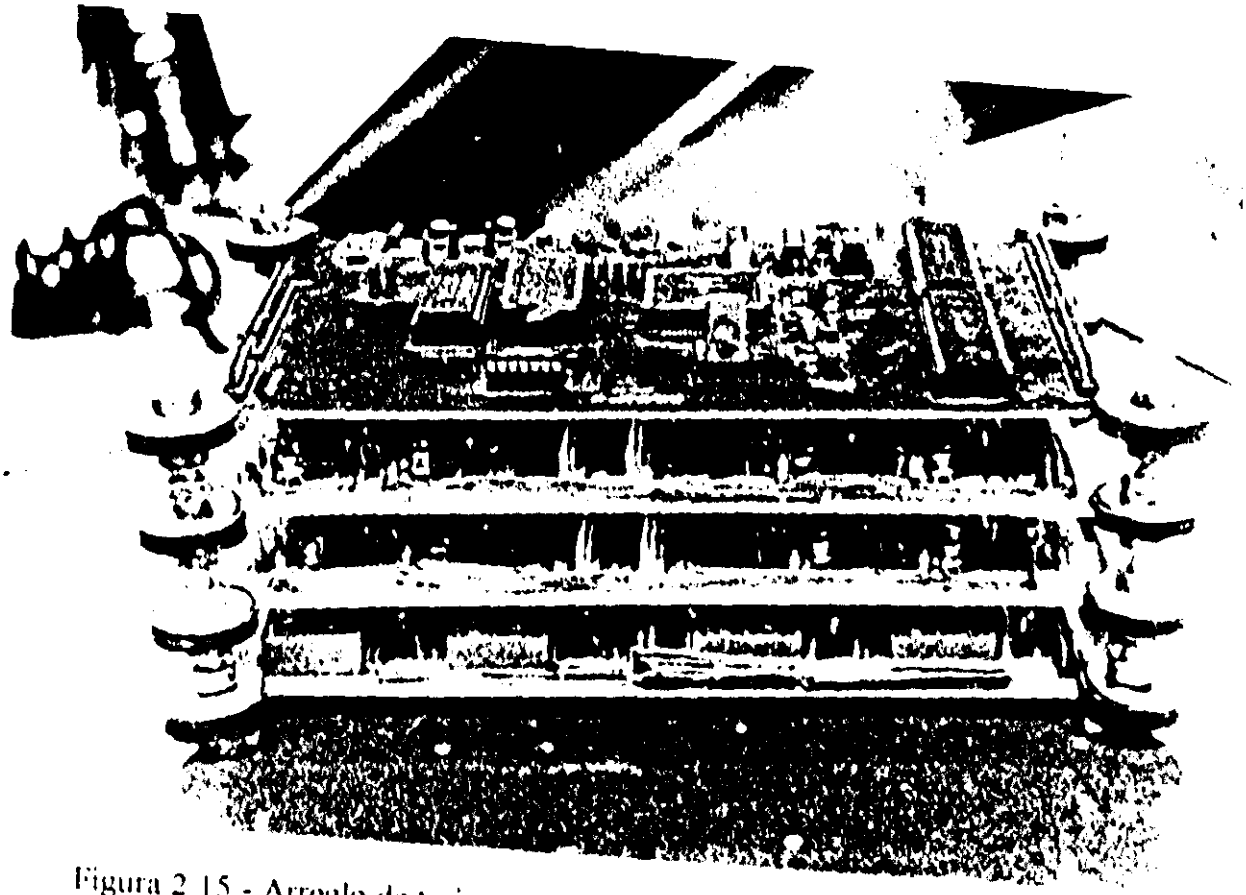


Figura 2.15 - Arreglo de tarjetas que realizan los bloques funcionales del PLM

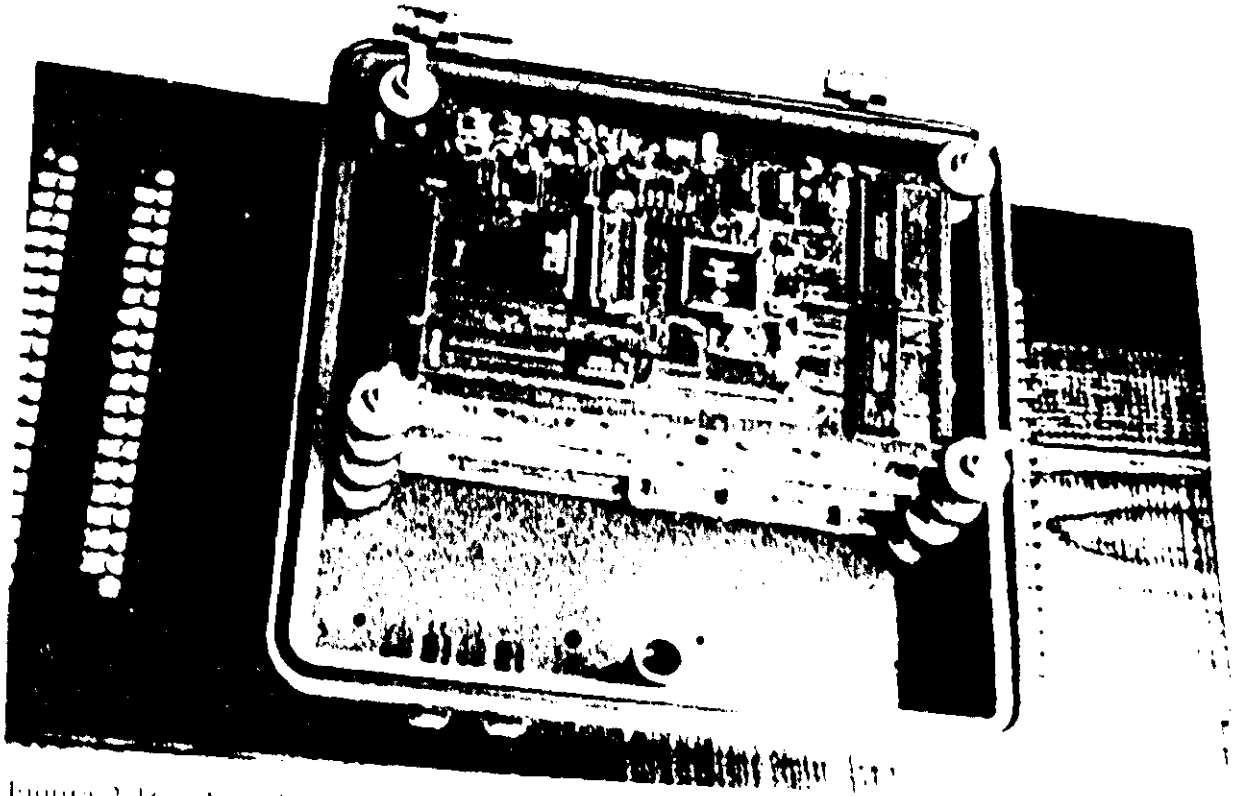


Figura 2.16 - Arreglo de tarjetas que realizan los bloques funcionales del PLM [illegible] en la caja empleada como gabinete

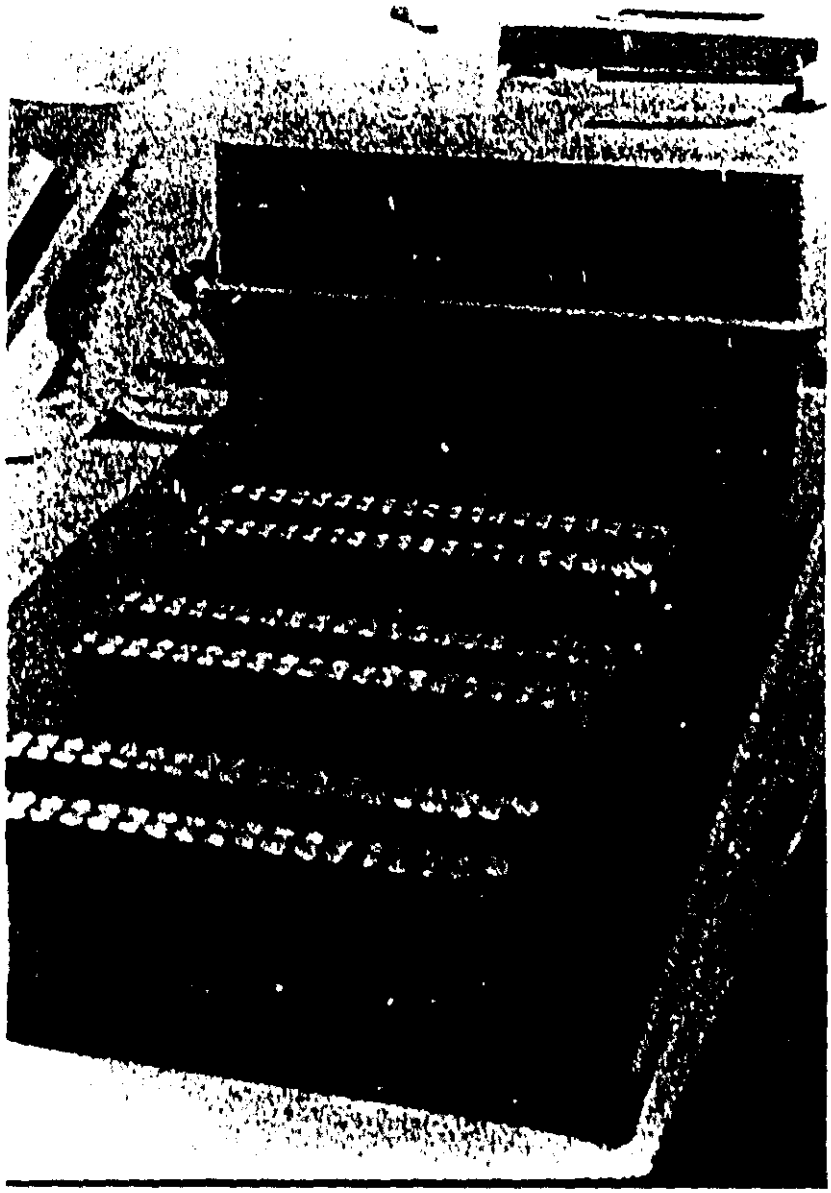


Figura 2.17 - PLM en su caja fijada a un tablero que contiene también a las terminales de entrada y salida

CAPÍTULO 3
MÓDULOS REALIZABLES POR EL PLM

En este capítulo se habla de la forma en que se diseñó el software correspondiente a cada uno de los módulos lógicos y auxiliares realizables por el PLM; iniciando con una descripción del flujo de los subprogramas principal y temporizado, al ejecutarse los mismos en la CC; continuando con la explicación del código asociado con cada uno de los módulos lógicos; siguiendo con lo propio relacionado con los módulos auxiliares; concluyendo con un ejemplo de programación en SIIL1.

3-1 ESTRUCTURA DE UN PROGRAMA EN SIIL1

Un programa en SIIL1, al ejecutarse en la CC del PLM, está dividido en dos subprogramas denominados subprograma principal y subprograma temporizado, el primero podrá contener tanto el código correspondiente a módulos que realizan compuertas lógicas y Flip-Flops como el correspondiente a módulos auxiliares, además de código de inicialización, de lectura y escritura de buffers y de salto a la posición del inicio del ciclo de barrido (scan); en lo que toca al subprograma temporizado el mismo es invocado por interrupción cada 10 ms, empleándose para ello al canal OC2 del temporizador interno del microcontrolador, siendo el código asociado el correspondiente a los módulos que realizan temporizadores, secuenciadores, contadores de eventos y módulos auxiliares, además de haber código propio del manejo de la interrupción OC2 y de actualización de buffers testigo de estado anterior.

Los módulos auxiliares mencionados en el párrafo anterior, están asociados con acciones relacionadas con el manejo de la Unidad Desplegadora (UD) y el reloj de tiempo real, algunos deberán ser colocados en el subprograma principal y otros deberán declararse en el subprograma temporizado, más adelante al tratar a cada uno de ellos en particular, se explicará lo que proceda.

Para cada tramo de código en un programa en SIIL1, correspondiente a una determinada acción efectuada por el mismo (módulo lógico, módulo auxiliar, código de inicialización, etc.), existe un código esqueleto normalizado (CEN) que es empleado por el

software que hace la traducción del programa fuente en SIIL1 al código objeto a ejecutarse en la CC del PLM, colocando para ello los bytes que correspondan en el CEN y localizando el mismo en la posición de memoria que corresponda.

Cada CEN se generó empleando un ensamblador de código fuente 68HC11, de modo que el código hexadecimal generado se origine en la dirección 0000, de esta manera cada CEN tiene asociada una tabla de asignación de bytes (TAB), donde se especifica que bytes han de ser modificados por el software de traducción de modo que el módulo asociado tenga las características especificadas originalmente por el usuario en el programa fuente; de esta manera, en cada uno de los CEN que maneja el software de traducción, el valor de los bytes que ha de cambiar el mismo podrá parecer absurdo, ya que dichos bytes tendrán su valor correcto después de que el software de traducción los haya reemplazado para obtener el tramo de código objeto requerido por el módulo que el usuario haya declarado en el programa fuente en SIIL1, que corresponda con la aplicación que este desarrollándose.

A continuación se describe la conformación en memoria RAM de los buffers que un programa en SIIL1 requiere al ejecutarse, y se explica el flujo de ejecución de los subprogramas principal y temporizado, detallándose en cada caso los CEN que correspondan.

3-2 DESCRIPCIÓN DE BUFFERS EN RAM

Al ejecutarse un programa en SIIL1 en la CC del PLM, el mismo requerirá varias localidades de RAM, para almacenamiento temporal de estados lógicos o valores numéricos que el propio programa genera y emplea, por cada grupo de variables booleanas (VBE, VBS, o VBI) se asigna en RAM un par de bytes colocados en direcciones consecutivas, en el primero de ellos es donde se copian los valores presentes de los bytes que reflejan el estado de las VB correspondientes al grupo, en el segundo byte se almacena el estado que presentaba el grupo antes de salir de la rutina de servicio de interrupción OC2 (subprograma temporizado), siendo esta información empleada por los módulos que tienen entradas sensibles a flancos, de esta forma, dado que el origen del buffer de las VB está en una dirección par, los valores presentes de las VB serán almacenados en direcciones pares, y los valores anteriores correspondientes quedarán guardados en direcciones impares.

Otras instancias del PLM que requieren el empleo de buffers en RAM son los temporizadores, los contadores de eventos, los módulos que manejan el reloj de tiempo real y la unidad desplegadora; a continuación se detalla la posición que ocupan en RAM los buffers asociados con las VB, los temporizadores y los contadores de eventos, lo concerniente a los buffers usados por el RTR y la UD es tratado al abordar los módulos correspondientes más adelante en este capítulo.

3-2-1 Buffers asociados con las variables booleanas

Los buffers asignados a las variables booleanas están colocados a partir de la dirección 0100H, En las tabla 3.1A se muestran las asignaciones de direcciones para los buffers de las variables booleanas del PLM y en la tabla 3.1B se muestran los valores de parámetros que aparecen en la tabla 3.1A que varían con la configuración de funcionamiento.

Tabla 3.1A Asignación de direcciones en RAM asociadas con buffers de almacenamiento de variables booleanas del PLM

Grupo	DAVAC*	DAVAN**
Grupo I de VBE's	2I+256+BUFEN	DAVAC+1
Grupo I de VBS's	2I+256+BUFSAL	DAVAC+1
Grupo I de VBI,s	2I+256+BUFI	DAVAC+1

* DAVAC= Dirección de almacenamiento de valor actual de grupo.
** DAVAN= Dirección de almacenamiento de valor anterior de grupo, esto se refiere al valor como byte que el grupo tenía durante la última ejecución del subprograma temporizado.

Tabla 3.1B Asignación de valores de parámetros que aparecen en la tabla 3.1A de acuerdo con la configuración de funcionamiento del PLM

Configuración	BUFEN	BUFSAL	BUFI
1 y 3	00	08	12 (0CH)
2	00	02	04

3-2-2 Buffers asociados con temporizadores y contadores de eventos

Los buffers correspondientes a los temporizadores inician a partir de la dirección 0200H y los asociados con los contadores de eventos se originan a partir de la dirección 0300H.

En cuanto al buffer que corresponde a los temporizadores, se asignan en el mismo tres bytes por cada temporizador, siendo la dirección base correspondiente dada por la siguiente expresión:

$$DIRBTN=512+3N \qquad (3.1)$$

Donde:

DIRBTN=dirección base de trío de bytes asociado con el temporizador número N.

N=número de temporizador.

En los temas de este capítulo que tratan los CEN que corresponden a los temporizadores, se explica el uso que tienen los trios de bytes mencionados aquí.

En lo que toca al buffer asociado con los contadores de eventos, se asignan dos bytes para cada uno de ellos, siendo la correspondiente dirección base:

$$DIRBCN=768+2N \qquad (3.2)$$

Donde:

DIRBCN=Dirección base del par de bytes asociado con el contador de eventos N.

N= número asociado al contador de eventos.

En los temas de este capítulo que tratan los CEN que corresponden a los contadores de eventos, se explica el uso que tienen los pares de bytes mencionados aquí.

3-3 FLUJO DE EJECUCIÓN DEL SUBPROGRAMA PRINCIPAL

El código del subprograma principal está dividido en dos partes denominadas respectivamente como INPLM3 y LPP (Lazo Programa Principal), en la primera se inicializan aspectos tales como la programación de puertos físicos, puesta a cero de buffers, carga de información relacionada con enrutamiento de interrupciones, carga de registros asociados con el canal OC2 del temporizador interno y colocación de testigos de primer paso por el LPP y subprograma temporizado, estos testigos son empleados en tiempo de ejecución, para inicializar condiciones lógicas en módulos que así lo requieran, por ejemplo, si el pulso de salida de un temporizador es verificado en alto, la salida correspondiente al

mismo deberá ser inicializada con un nivel bajo y esta acción ha de ser efectuada la primera vez que se ejecute el código correspondiente a tal temporizador, en la CC del PLM.

En lo que toca al código correspondiente al LPP a continuación se describen las acciones llevadas a cabo por el mismo.

Acciones efectuadas por el LPP

1.- Se copian en un buffer de entrada (BE) en RAM el estado que guardan los cuatro puertos asociados con las 32 entradas físicas VBE. El tramo de código asociado con esta acción se denomina LECBUF3.

2.- Se ejecuta uno a uno el código asociado con cada uno de los ML que el usuario haya declarado en el subprograma principal, tomándose del BE las entradas que cada módulo requiera, las salidas que se fueran generando son colocadas en un buffer de salida (BS) en RAM; si el ML emplea una o varias VBI como entradas los valores asociados con las mismas son tomados de un buffer intermediario (BI) en RAM, en caso de que haya en el ML una o varias salidas de tipo VBI los valores correspondientes son escritos en el BI. El código correspondiente a esta parte, estará constituido por la unión de los tramos de código correspondientes a cada uno de los ML que integren el subprograma principal, colocados en el orden en que los mismos hayan sido declarados.

3.- Se copia el estado del BS en los puertos físicos asociados con las VBS. El tramo de código correspondiente a esta acción se denomina ESCBUF1.

4.- Se regresa al paso uno. El tramo de código correspondiente a esta parte se denomina SALTO.

En la figura 3.1 se muestra un diagrama que muestra el flujo de ejecución del subprograma principal.

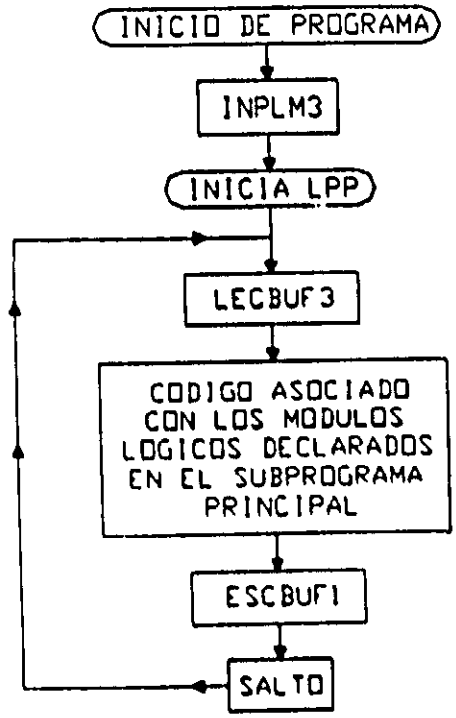


Figura 3.1.- Flujo de ejecución del subprograma principal, correspondiente a un determinado programa en SIIL1 ejecutándose en la CC del PLM.

A continuación se describen los CEN correspondientes a los tramos de código asociados con los bloques que aparecen en la figura 3.1.

3-3-1 Descripción del CEN asociado con el tramo de código INPLM3

El CEN correspondiente al tramo de código INPLM3 se muestra a continuación:

CEN ASOCIADO CON INPLM3

```
Carga seudovector de canal OC2 del temporizador interno del 68HC11
0000 867E                                   LDAA #$7E
0002 97DC                                   STAA $DC
0004 CEC000                                 LDX #$C000
0007 DFDD                                   STX $DD
0009 8692                                   LDAA #$92; INICIALIZA PPI's
000B B71803                                 STAA $1803
000E B71883                                 STAA $1883
  Inicializa registros asociados con canal OC2 del temporizador interno
                                    del 68HC11
0011 8640                                   LDAA #$40
0013 B71020                                 STAA $1020
0016 B71022                                 STAA $1022
0019 B71023                                 STAA $1023
        Inicializa RTR (MM58274), colocado en direccion base 1BC0
001C CE1BC0                                 LDX #$1BC0
```

```
001F 8601                                   LDAA #$01
0021 A700                                   STAA $00,X
0023 A70F                                   STAA $0F,X
                    Checa puente Ja
0025 B61B90                                 LDAA $1B90
0028 8440                                   ANDA #$40
002A 2703                                   BEQ CHECJB
Pone en RTR hora y fecha por defecto (00:00:00 del viernes 1 de enero de
1998).
002C BDFF90                                 JSR HORADEF
                    Checa puente Jb
002F B61B90         CHECJB:                 LDAA $1B90
0032 8480                                   ANDA #$80
0034 270B                                   BEQ HH1
Inicia a ceros buffers de E/I/S, temporizadores y contadores
0036 CE0100                                 LDX #$0100
0039 6F00           RETOR:                  CLR $00,X
003B 08                                     INX
003C 8C0400                                 CPX #$0400
003F 26F8                                   BNE RETOR
0041 CE0100         HH1:                    LDX #$0100
        Pone testigo de primer paso por subprograma temporizado
0044 86FF                                   LDAA #$FF
0046 A7F1                                   STAA $F1,X
      Pone testigo de primer paso por lazo de subprograma principal
0048 A7F2                                   STAA $F2,X
```

En la figura 3.2 se muestra a bloques el flujo de ejecución de INPLM3 y en la tabla 3.2 se aprecia la asignación de bytes correspondiente.

Tabla 3.2 Asignación de bytes asociada con el CEN INPLM3

Valor numérico (VN) generado por el software de traducción	Bytes en CEN INPLM3 a los que hay que asignar el valor numérico VN
Byte alto de dirección de inicio de rutina de servicio de interrupción OC2	B5
Byte bajo de dirección de inicio de rutina de servicio de interrupción OC2	B6

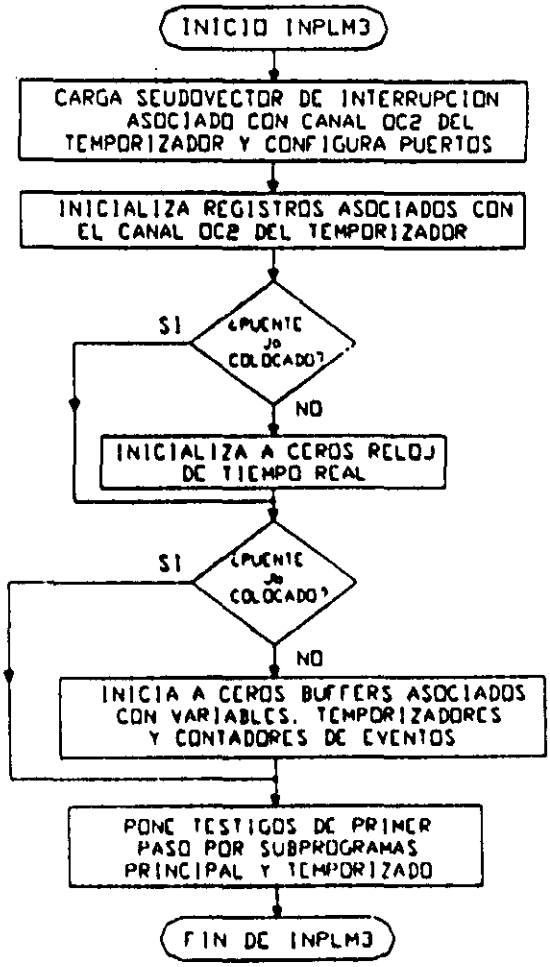


Figura 3.2 Flujo de ejecución de INPLM3

3-3-2 Descripción del CEN asociado con el tramo de código LECBUF3

Para el tramo de código LECBUF3 el programa ensamblado es el siguiente:

CEN asociado con LECBUF3

```
0000 FC1800          LECBUF:     LDD $1800
0003 43                          COMA
0004 53                          COMB
0005 B70100                      STAA $0100
0008 F70102                      STAB $0102
000B FC1880                      LDD $1880
000E 43                          COMA
000F 53                          COMB
0010 B70104                      STAA $0104
0013 F70106                      STAB $0106
```

El CEN LECBUF3 tiene un tiempo de ejecución asociado de 17 microsegundo- (fe=2MHz) y sirve para las configuraciones de funcionamiento uno y tres y no tiene TAB, esto es, el software de traducción coloca al mismo en la posición que le corresponda sin

hacer ningún cambio en sus bytes, para la configuración de funcionamiento dos el CEN a emplear se denomina LECBUF2 y se muestra a continuación:

CEN asociado con LECBUF2

```
0000 B61B00          LECBUF:          LDAA $1B00
0003 B70100                           STAA $0100
```

Al igual que LECBUF3, LECBUF2 no requiere tener una TAB; el tiempo de ejecución asociado con LECBUF2 es de 4 microsegundos (fe=2MHZ).

3-3-3 Código asociado con los módulos declarados en el subprograma principal

El código correspondiente a los módulos declarados en el subprograma principal (CMLPP), está constituido por una sucesión de tramos de código (TC), uno por cada módulo declarado en el subprograma principal; cada TC es armado por el software de traducción (SIIL1.EXE), a partir de los CEN de cada módulo implicado de acuerdo con lo declarado por el usuario en cada caso en el subprograma principal, empleándose para ello la TAB que corresponda.

Más adelante en este capítulo se hablará de los CEN asociados con cada uno de los módulos (lógicos y auxiliares), que a la fecha de elaboración de este trabajo, pueden ser realizados por el PLM.

3-3-4 Descripción del CEN asociado con el tramo de código ESCBUF1

Para el tramo de código ESCBUF1 el programa ensamblado es el siguiente:

CEN asociado con ESCBUF1

```
0000 B60108          ESCBUF: LDAA $0108
0003 B71802                  STAA $1802
0006 B6010A                  LDAA $010A
0009 B71882                  STAA $1882
```

El TC ESCBUF1 es empleado para las configuraciones de funcionamiento uno y tres, y al igual que para el TC LECBUF1, ESCBUF1 no requiere de una TAB, para la configuración de funcionamiento dos el TC correspondiente se denomina ESCBUF2 y su CEN asociado se muestra a continuación:

CEN asociado con el TC ESCBUF2

```
0000 B60102          ESCBUF:          LDAA $0102
0003 B71B80                           STAA $1B80
```

ESCBUF2 tampoco requiere de una TAB; los tiempos de ejecución asociados con los TC ESCBUF1 y ESCBUF2 son respectivamente 8 y 4 microsegundos (fe=2MHz).

3-3-5 Descripción del CEN asociado con el tramo de código SALTO

Para el tramo de código SALTO el programa ensamblado es el siguiente:

CEN asociado con el TC SALTO

```
 Checa si se ha ejecutado el primer paso por el LPP del subprograma
temporizado
 0000 A6F2                                  LDAA $F2,X
 Si se ha ejecutado el primer paso por el LPP habilita interrupciones, si
el programa en SIIL1 contiene subprograma temporizado
 0002 2703                                  BEQ SALTON
 El siguiente byte habrá de ser 01H si no hay subprograma temporizado o
0EH en otro caso
 0004 01                                    NOP
 Pone a cero byte testigo de primer paso por LPP
 0005 6FF2                                  CLR $F2,X
 Salta a dirección de colocación del TC LECBUFX, X=1 ó 2
 0007 7EC040                SALTON:         JMP $C040
```

El tiempo de ejecución del TC que se obtiene a partir del CEN SALTO es de 9 microsegundos para la primera ejecución del mismo, y de 5 microsegundos en otro caso. La tabla de asignación de bytes correspondiente al CEN SALTO se muestra a continuación:

Tabla 3.3 Asignación de bytes asociada con el CEN SALTO

Valor numérico (VN) generado por el software de traducción	Bytes en CEN INPLM3 a los que hay que asignar el valor numérico VN
VN=01 si no hay subprograma temporizado VN=0EH si hay subprograma temporizado	B4
Byte alto de dirección de inicio de TC LECBUFX, X=1 ó 2	B8
Byte bajo de dirección de inicio de TC LECBUFX, X=1 ó 2	B9

3-4 FLUJO DE EJECUCIÓN DEL SUBPROGRAMA TEMPORIZADO

El código correspondiente al subprograma temporizado, es ejecutado cada 10 ms y bajo la perspectiva del microcontrolador 68HC11, constituye la rutina de servicio de interrupción OC2 y el mismo está integrado por cinco tramos de código que son los siguientes:

1 - Entrada a rutina de servicio de interrupción OC2, este TC actualiza umbral de comparación del canal OC2 del temporizador interno del microcontrolador y pone a cero la bandera correspondiente (OC2F), el CEN asociado con este TC se denomina ENRUS.

2.- Código asociado con los ML que hayan sido declarados en el subprograma temporizado, que deberá corresponder con módulos cuyo código requiere ser ejecutado cada 10 ms (v.g. temporizadores) y/o contienen entradas sensibles a flancos de subida o bajada.

3.- Código que copia buffer de valor presente de todos los grupos de variables booleanas en buffer testigo de valor anterior correspondiente. Este tramo de código, en su versión normalizada, se denomina ACT.

4.- Código que copia a RAM el estado de los dos puertos de entrada auxiliares (PAUXA y PAUXB), este tramo de código, en su versión normalizada, se denomina ACTBOT.

5.- Código de retorno de interrupción, denominado RETRUS.

En la figura 3.3 se muestra a bloques el flujo de ejecución del subprograma temporizado.

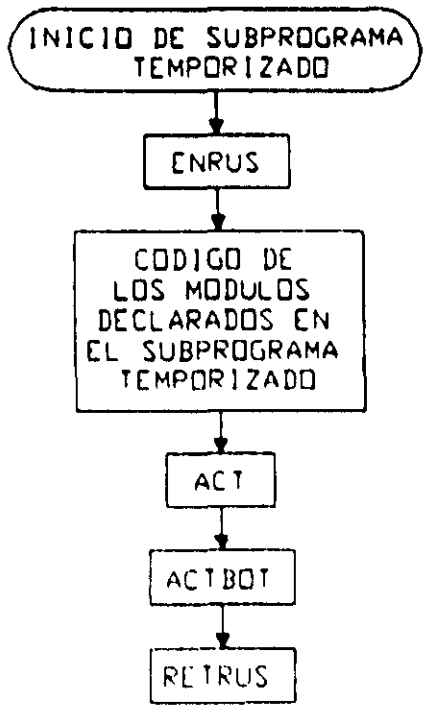


Figura 3.3 flujo de ejecución del subprograma temporizado, correspondiente a un programa en SIIL1 ejecutándose en el PLM.

A continuación se describe los CEN asociados con los TC que integran el subprograma temporizado.

3-4-1 Descripción del CEN asociado con el tramo de código ENRUS

Para el tramo de código ENRUS el programa ensamblado es el siguiente:

CEN asociado con el TC ENRUS

```
0000 FC1018                     LDD $1018
0003 C34E20                     ADDD #$4E20
0006 FD1018                     STD $1018
0009 8640                       LDAA #$40
000B B71023                     STAA $1023
000E CE0100                     LDX #$0100
```

ENRUS no requiere de tabla de asignación de bytes y el tiempo de ejecución asociado es de 11.5 microsegundos.

3-4-2 Código asociado con los módulos declarados en el subprograma temporizado

El código correspondiente a los módulos declarados en el subprograma temporizado (CMST), está constituido por una sucesión de tramos de código (TC), uno por cada ML declarado en el subprograma temporizado; cada TC es armado por el software de traducción (SIIL1.EXE), a partir de los CEN de cada módulo implicado de acuerdo con lo declarado por el usuario en cada caso en el subprograma temporizado, empleándose para ello la TAB que corresponda.

3-4-3 Descripción del CEN ACT

La función del TC, que se obtiene a partir del CEN ACT, consiste en copiar el estado de todos los grupos de VB en RAM, esto antes de salir de la rutina de servicio OC2 (subprograma temporizado) ya que esta información es requerida, por el código asociado con módulos con entradas sensibles a flancos en ejecuciones posteriores del subprograma temporizado.

El programa ensamblado correspondiente con el CEN ACT es:

```
0000 3C                         PSHX
0001 A600           AA:         LDAA $00,X
0003 A701                       STAA $01,X
0005 08                         INX
0006 08                         INX
Compara con dirección tope +2 de VBI's
0007 8C0134                     CPX #$0136
000A 26F5                       BNE AA
000C 38                         PULX
```

Para las configuraciones de funcionamiento 1 y 3, el tiempo de ejecución para el TC asociado es de 432 microsegundos, para la configuración 2 es de 361 microsegundos, en la tabla 3.4 se muestra la TAB correspondiente.

Tabla 3.4 Asignación de bytes asociada con el CEN ACT

Valor numérico (VN) generado por el software de traducción	Byte en CEN ACT al que hay que asignar el valor numérico VN
VN=54 (36H) para las configuraciones 1 y3 VN=46 (2EH) para la configuración 2	B9

3-4-4 Descripción del CEN ACTBOT

La función del TC, que se obtiene a partir del CEN ACTBOT, consiste en copiar en RAM el estado de los puertos de entrada auxiliares, esto antes de salir de la rutina de servicio OC2 (subprograma temporizado), ya que esta información es requerida, por el código asociado con módulos que pudieran emplear uno o varios de los botones auxiliares, esto en ejecuciones posteriores del subprograma temporizado.

El programa ensamblado correspondiente con el CEN ACTBOT es:

```
0000 B61BB0          LDAA $1BB0
0003 A7CE            STAA $CE,X
0005 B61B90          LDAA $1B90
0008 A7CF            STAA $CF,X
```

El tiempo de ejecución asociado con el TC obtenido a partir del CEN ACTBOT es de 8 microsegundos, y el mismo no requiere de tabla de asignación de bytes.

3-4-5 Descripción del CEN RETRUS

La función del TC que se obtiene a partir del CEN RETRUS, consiste en poner a cero el testigo de primer paso por el subprograma temporizado y ejecutar la instrucción de retorno de interrupción (RTI) del microcontrolador, este CEN no requiere de una TAB y consta de solo tres bytes, el programa ensamblado correspondiente con el CEN RETRUS es:

CEN asociado con el TC RETRUS

```
0000 6FF1            CLR $F1,X
0002 3B              RTI
```

El tiempo de ejecución correspondiente es en este caso de 9 microsegundos.

3-5 DESCRIPCIÓN DE LOS CEN ASOCIADOS CON MÓDULOS LÓGICOS DECLARDOS POR EL USUARIO

A partir de los ML declarados por el usuario tanto en el subprograma principal como en el temporizado, el software de traducción (SIIL1.EXE) genera los TC CMLPP y CMST como se ha esbozado en los temas 3-3-3 y 3-4-2, siendo además estos TC colocados en su posición dentro del programa objeto, para hacer esto SIIL1.EXE emplea para cada ML un CEN asociado con el mismo, a continuación se describe el funcionamiento de los módulos lógicos que puede realizar el PLM a la fecha de escritura de esta tesis, detallándose para cada caso le estructura del CEN que corresponda.

3-5-1 Descripción del módulo de seguimiento lógico y su CEN asociado

Este ML simplemente pone en la VB declarada como salida el nivel lógico que exista en la VB declarada como entrada al mismo, en la figura 3.4 se ilustra en forma genérica este ML, debiendo el mismo ser declarado en el subprograma principal, siendo la sintaxis para declararlo la siguiente:

SEG#N XEIEJE,XSISJS;

Donde:

N denota el número de seguidor, esto definido por el usuario.

XE 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.

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

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

XS podrá ser la letra "s" o "i" mayúscula o minúscula dependiendo esto de que la variable de salida al seguidor sea una VBS o VBI.

IS denota el número de grupo que corresponda a la VB declarada como salida del seguidor

JS denota el número de bit dentro del grupo IS, asociado a la variable de salida del seguidor

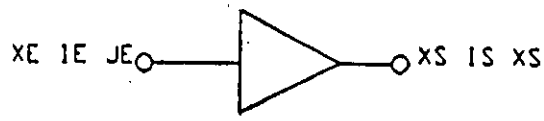


Figura 3.4 Representación genérica del seguidor lógico realizado por el PLM

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

Ejemplo 3.1

Se desea realizar con el PLM 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

Descripción del CEN asociado con el módulo de seguimiento lógico

El CEN asociado con el módulo de seguimiento lógico es:

```
0000 A600                          LDAA $00,X
0002 8401                          ANDA #$01
0004 2705                          BEQ ALFA1
0006 1C0001                        BSET $00,X,01
0009 2003                          BRA ALFA2
000B 1D0001       ALFA1:           BCLR $00,X,01
000E 01           ALFA2:           NOP
```

En la tabla 3.5A se muestra la TAB asociada con el CEN del seguidor lógico y en la tabla 3.5B se aprecia el valor que deben tomar, de acuerdo con el tipo de VB que corresponda a la entrada y salida del seguidor, parámetros que aparecen en la tabla 3.5A. En la tabla 3.5C se muestra el valor que puede tomar el tiempo de ejecución de este módulo, bajo las diferentes condiciones lógicas que se pueden presentar a su entrada, esto considerando que la frecuencia de la señal de reloj E del microcontrolador es de 2 MHz, que es el caso de la CC del PLM.

Tabla 3.5A Asignación de bytes asociada con los CEN del seguidor e inversor lógicos

Valor numérico (VN) generado por el software de traducción	Byte en CEN del seguidor o inversor al que hay que asignar el valor numérico VN
VN=2IE+BUFE	B1
VN=2^JE	B3
VN=2IS+BUFS	B7 y B12
VN=2^JS	B8 y B13

Tabla 3.5B Asignación de valores de los parámetros BUFE y BUFS de la tabla 3.5A

Caracter XE	**BUFE**
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Caracter XS	**BUFS**
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)

Tabla 3.5C Tiempo de ejecución (Te) del código correspondiente al seguidor e inversor.

Entrada (XEIEJE)	**Te en microsegundos (fe=2Mhz)**
0	9
1	10.5

3-5-2 Descripción del módulo de inversión lógica y su CEN asociado

Este ML simplemente pone en la VB declarada como salida el nivel lógico opuesto al que exista en la VB declarada como entrada al mismo, en la figura 3.5 se ilustra en forma genérica este ML, debiendo el mismo ser declarado en el subprograma principal, siendo la sintaxis para declararlo la siguiente:

NOT#N XEIEJE,XSISJS

Donde:

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

XE 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.

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

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

XS podrá ser la letra "s" o "i" mayúscula o minúscula dependiendo esto de que la variable de salida al inversor sea una VBS o VBI.

IS denota el número de grupo que corresponda a la VB declarada como salida del inversor.

JS denota el número de bit dentro del grupo IS, asociado a la variable de salida del inversor.

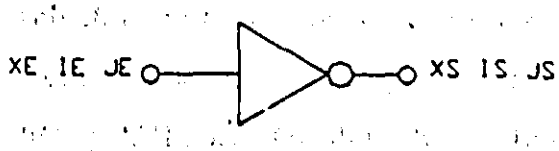


Figura 3.5 Representación genérica del inversor lógico realizado por el PLM

A continuación se muestra un ejemplo sobre como declarar un módulo de inversión lógica, en un programa fuente en SIIL1.

Ejemplo 3.2

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

NOT#7 E12, I67

Descripción del CEN asociado con el módulo de inversión lógica

El CEN asociado con el módulo de inversión lógica es:

```
0000 A600                LDAA $00,X
0002 8401                ANDA #$01
0004 2705                BEQ ALFA1
0006 1D0001              BCLR $00,X,01
0009 2003                BRA ALFA2
000B 1C0001      ALFA1:  BSET $00,X,01
000E 01          ALFA2:  NOP
```

En la tabla 3.5A se muestra la TAB asociada con el CEN del inversor lógico y en la tabla 3.5B se aprecia el valor que deben tomar, de acuerdo con el tipo de VB que corresponda a la entrada y salida del inversor, parámetros que aparecen en la tabla 3.5A. En la tabla 3.5C se muestra el valor que puede tomar el tiempo de ejecución de este módulo, bajo las diferentes condiciones lógicas que se pueden presentar a su entrada, esto considerando que la frecuencia de la señal de reloj E del microcontrolador es de 2 MHz, que es el caso de la CC del PLM.

3-5-3 Descripción de los ML que realizan compuertas de dos entradas y CEN asociado

El PLM puede realizar seis tipos de compuertas lógicas de dos entradas y estas son de tipo: AND, OR, NAND, NOR, OR EXCLUSIVA y OR EXCLUSIVA NEGADA, teniéndose además la capacidad de preinversión en las entradas que el usuario desee. En la Figura 3.6 se ilustra en forma genérica este ML, debiendo el mismo ser declarado en el subprograma temporizado, siendo la sintaxis para declararlo la siguiente:

COMP#N X0I0J0, X1I1J1, XSISJS, AB

Donde:

COMP es una cadena que puede ser AND2, OR2, NAND2, NOR2, EOR2 o EORN2, esto de acuerdo al tipo de compuerta que se desee realizar.

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

X0 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.

I0 denota el número de grupo que corresponda a la VB declarada como entrada E0 a la compuerta.

J0 denota el número de bit dentro del grupo I0, asociado a la variable de entrada E0.

X1 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.

I1 denota el número de grupo que corresponda a la VB declarada como entrada E1 a la compuerta.

J1 denota el número de bit dentro del grupo I1, asociado a la variable de entrada E1.

XS 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.

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

JS denota el número de bit dentro del grupo IS, 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 dígito 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 3.6 Representación genérica de una compuerta de dos entradas realizada por el PLM

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 3.3

Se desea realizar con el PLM 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 3.7; en este caso, se deberá usar la siguiente sintaxis:

AND2#4 E01, I24, S13, 10;

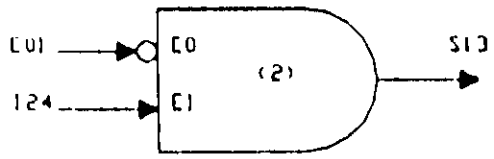


Figura 3.7.- Ejemplo de compuerta AND de dos entradas con preinversión en la entrada E1, la declaración sintáctica correspondiente es: AND2#4 E01, I24, S13, 10

Descripción del CEN asociado con los ML que realizan compuertas de dos entradas

En la figura 3.8 se muestra el flujo de ejecución del tramo de código correspondiente a una compuerta de dos entradas.

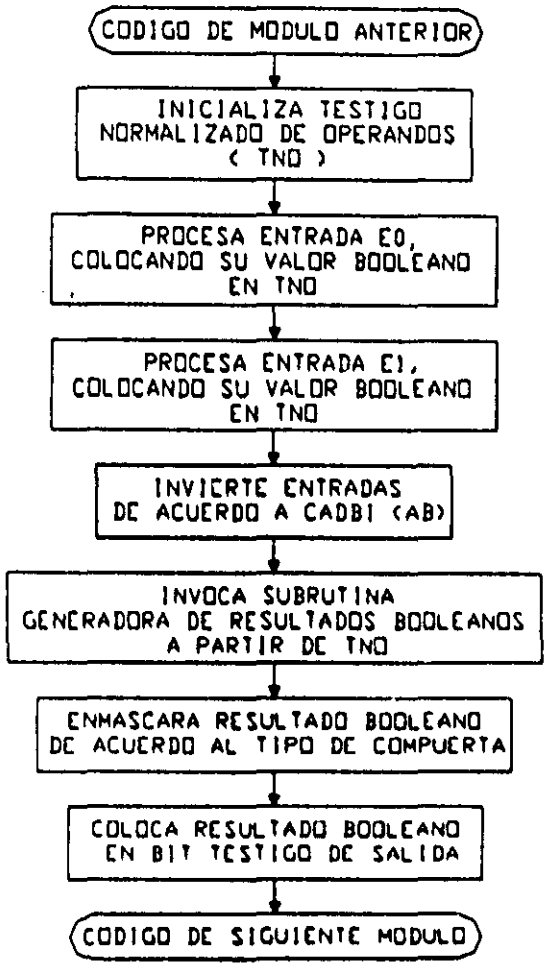


Figura 3.8 Flujo de ejecución del tramo de código empleado para realizar una compuerta de dos entradas

El software de traducción emplea un CEN genérico denominado OPER2UN, para obtener el TC requerido por una determinada compuerta de dos entradas declarada por el usuario en el subprograma principal; el CEN OPER2UN es:

CEN OPER2UN

```
Inicializa testigo normalizado de operandos (TNO)
0000 8664                                       LDAA #$64
0002 A7F0                                       STAA $F0,X
Procesa entrada E0
0004 A600                                       LDAA $00,X
0006 8401                                       ANDA #$01
0008 2703                                       BEQ ALFA1
000A 1CF001                                     BSET $F0,X,01
Procesa entrada E1
000D A600                 ALFA1:                LDAA $00,X
000F 8402                                       ANDA #$02
0011 2703                                       BEQ ALFA2
0013 1CF002                                     BSET $F0,X,02
```

```
 Invierte entradas de acuerdo con cadena CADBI
 0016 8600                ALFA2:                  LDAA #$00
 0018 A8F0                                        EORA $F0,X
 001A A7F0                                        STAA $F0,X
 Invoca subrutina generadora de resultados booleanos (SGRB)
 001C BDFF80                                      JSR $FF80
 Enmascara resultado booleano de acuerdo a tipo de compuerta colocando el
mismo en bit testigo de salida
 001F 8401                                        ANDA #$01
 0021 2705                                        BEQ ALFA5
 0023 1C0001                                      BSET $00,X,01
 0026 2003                                        BRA ALFA6
 0028 1D0001              ALFA5:                  BCLR $00,X,01
 002B 01                  ALFA6:                  NOP
```

La subrutina generadora de resultados booleanos (SGRB) está cargada en memoria fija (EEPROM) de la CC a partir de la dirección FF80H, el código en ensamblador para la misma es el siguiente:

Subrutina SGRB

```
ORG $FF80
PSHX
PSHB
LDAB $F0,X
LDAA #$FF
XGDX
LDAA $00,X
PULB
PULX
RTS
```

Para generar el resultado booleano, la subrutina SGRB usa tablas cargadas en memoria fija (EEPROM) de la CC, empleándose la localidad de RAM cuya dirección es 01F0H para almacenar el testigo normalizado de operandos (TNO).

En la tabla 3.6A se muestra la TAB asociada con el CEN OPER2UN y en la tabla 3.6B se aprecia el valor que deben tomar, de acuerdo con el tipo de VB que corresponda a la entrada y salida de la compuerta, parámetros que aparecen en la tabla 3.6A. El tiempo de ejecución para una compuerta de dos entradas variará de acuerdo con las entradas a la misma, en la tabla 3.6C se muestra el valor máximo que puede tomar este tiempo, para los distintos tipos de compuertas realizables por el PLM, esto considerando que la frecuencia de la señal de reloj E del microcontrolador es de 2 MHz, que es el caso de la CC del PLM.

Tabla 3.6A Asignación de bytes asociada con el CEN OPER2UN, empleado para obtener el código requerido por los ML que realizan compuertas de dos entradas.

Valor numérico (VN) generado por el software de traducción	Byte en CEN genérico para realizar compuertas de dos entradas, al que se debe asignar el valor numérico VN
VN=2I0+BUF0	B5
VN=2^J0	B7
VN=2I1+BUF1	B14
VN=2^J1	B16
VN=2IS+BUFS	B36, B41
VN=2^JS	B37, B42
VN=1 (AND), 2 (OR), 4 (NAND), 8 (NOR), 16 (EOR), 32 (EORN)	B32
VN=3-X, X=valor numérico de CADBI	B23

Tabla 3.6B Valores de los parámetros BUF0, BUF1, y BUFS de la tabla 3.6A

Carácter X0	**BUF0**
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Carácter X1	**BUF1**
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Carácter XS	**BUFS**
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)

Tabla 3.6C Máximo tiempo de ejecución (TEM) del código correspondiente a una compuerta de dos entradas realizada por el PLM.

Tipo de compuerta de dos entradas	TEM en microsegundos (fc=2Mhz)
AND	52.5
OR	52.5
NAND	51
NOR	51
EOR	51
EORN	52.5

3-5-4 Descripción de los ML que realizan compuertas de tres entradas y CEN asociado

El PLM puede realizar seis tipos de compuertas lógicas de tres entradas y estas son de tipo: AND, OR, NAND, NOR, OR EXCLUSIVA y OR EXCLUSIVA NEGADA,

teniéndose además la capacidad de preinversión en las entradas que el usuario desee. En la figura 3.9 se ilustra en forma genérica este ML, debiendo el mismo ser declarado en el subprograma principal, siendo la sintaxis para declararlo la siguiente:

COMP#N X0I0J0, X1I1J1, X2I2J2, XSISJS, ABC

Donde:

COMP es una cadena que puede ser AND3, OR3, NAND3, NOR3, EOR3 o EORN3, esto de acuerdo al tipo de compuerta que se desee realizar.

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

X0 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.

I0 denota el número de grupo que corresponda a la VB declarada como entrada E0 a la compuerta.

J0 denota el número de bit dentro del grupo I0, asociado a la variable de entrada E0.

X1 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.

I1 denota el número de grupo que corresponda a la VB declarada como entrada E1 a la compuerta.

J1 denota el número de bit dentro del grupo I1, asociado a la variable de entrada E1.

X2 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.

I2 denota el número de grupo que corresponda a la VB declarada como entrada E2 a la compuerta.

J2 denota el número de bit dentro del grupo I2, asociado a la variable de entrada E2

XS 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.

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

JS denota el número de bit dentro del grupo IS, 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.

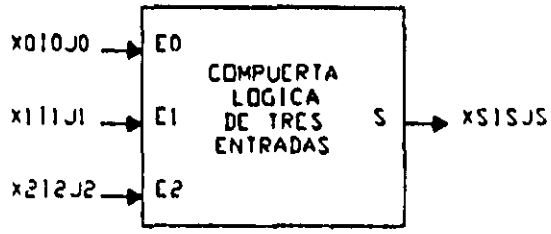


Figura 3.9 Representación genérica de una compuerta de tres entradas realizada por el PLM

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

Ejemplo 3.4

Supóngase que se necesita realizar con el PLM 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 3.10; en este caso, se deberá usar la siguiente sintaxis:

NOR3#7 E14, I03, E17, S17, I01;

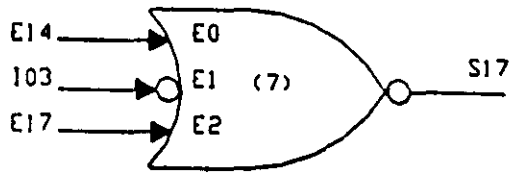


Figura 3.10.- Ejemplo de compuerta NOR de tres entradas con preinversión en la entrada E1, la declaración sintáctica correspondiente es:NOR3#7 E14, I03, E17, S17, 101

Descripción del CEN asociado con los ML que realizan compuertas de tres entradas

En la figura 3.11 se muestra el flujo de ejecución del tramo de código correspondiente a una compuerta de tres entradas.

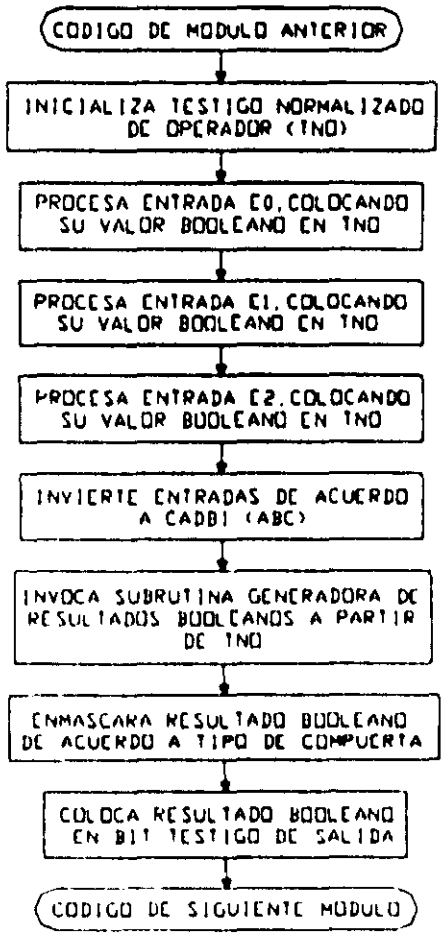


Figura 3.11 Flujo de ejecución del tramo de código empleado para realizar una compuerta de tres entradas

El software de traducción emplea un CEN genérico denominado OPER3UN, para obtener el TC requerido por una determinada compuerta de tres entradas declarada por el usuario en el subprograma principal; el CEN OPER3UN es:

CEN OPER3UN

```
Inicializa testigo normalizado de operandos (TNO)
 0000 8668                                      LDAA #$68
 0002 A7F0                                      STAA $F0,X
 Procesa entrada E0
 0004 A600                                      LDAA $00,X
 0006 8401                                      ANDA #$01
 0008 2703                                      BEQ ALFA1
 000A 1CF001                                    BSET $F0,X,01
 Procesa entrada E1
 000D A600                      ALFA1:          LDAA $00,X
 000F  8402                                     ANDA #$02
 0011  2703                                     BEQ ALFA2
 0013  1CF002                                   BSET $F0,X,02
 Procesa entrada E2
 0016 A600                      ALFA2:          LDAA $00,X
 0018 8401                                      ANDA #$01;
BT(25)=V5, V1=2^J2
 001A 2703                                      BEQ ALFA4
 001C 1CF004                                    BSET $F0,X,04
 Invierte entradas de acuerdo a cadena CADBI
 001F 8600                      ALFA4:          LDAA #$00
 0021 A8F0                                      EORA $F0,X
 0023 A7F0                                      STAA $F0,X
 Invoca subrutina generadora de resultados booleanos (SGRB)
 0025 BDFF80                                    JSR $FF80
 Enmascara resultado booleano de acuerdo a tipo de compuerta colocando el
mismo en bit testigo de salida
 0028 8401                                      ANDA #$01
 002A 2705                                      BEQ ALFA5
 002C 1C0001                                    BSET $00,X,01
 002F 2003                                      BRA ALFA6
 0031 1D0001                    ALFA5:          BCLR $00,X,01
 0034 01                        ALFA6:          NOP
```

La subrutina generadora de resultados booleanos (SGRB) está cargada en memoria fija (EEPROM) de la CC a partir de la dirección FF80H, el código en ensamblador de la misma puede verse en el tema 3-5-3 de este capítulo, donde se explica el CEN OPER2UN

Para generar el resultado booleano, la subrutina SGRB usa tablas cargadas en memoria fija (EEPROM) de la CC, empleándose la localidad de RAM cuya dirección es 01F0H para almacenar el testigo normalizado de operandos (TNO).

En la tabla 3.7A se muestra la TAB asociada con el CEN OPER3UN y en la tabla 3.7B se aprecia el valor que deben tomar, de acuerdo con el tipo de VB que corresponda a la entrada y salida de la compuerta, parámetros que aparecen en la tabla 3.7A. El tiempo de

ejecución para una compuerta de tres entradas variará de acuerdo con las entradas a la misma, en la tabla 3.7C se muestra el valor máximo que puede tomar este tiempo, para los distintos tipos de compuertas realizables por el PLM, esto considerando que la frecuencia de la señal de reloj E del microcontrolador es de 2 MHz, que es el caso de la CC del PLM.

Tabla 3.7A Asignación de bytes asociada con el CEN OPER3UN, empleado para obtener el código requerido por los ML que realizan compuertas de tres entradas.

Valor numérico (VN) generado por el software de traducción	Byte en CEN genérico para realizar compuertas de tres entradas, al que se debe asignar el valor numérico VN
VN=210+BUF0	B5
VN=2^J0	B7
VN=211+BUF1	B14
VN=2^J1	B16
VN=212+BUF2	B23
VN=2^J2	B25
VN=21S+BUFS	B45, B50
VN=2^JS	B46, B51
VN=1 (AND), 2 (OR), 4 (NAND), 8 (NOR), 16 (EOR), 32 (EORN)	B41
VN=·7-X, X=valor numérico de CADBI	B32

Tabla 3.7B Valores de los parámetros BUF0, BUF1, BUF2 y BUFS de la tabla 3.7A

Caracter X0	**BUF0**
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Caracter X1	**BUF1**
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Caracter X2	**BUF2**
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Caracter XS	**BUFS**
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)

Tabla 3.7C Máximo tiempo de ejecución (TEM) del código correspondiente a una compuerta de tres entradas realizada por el PLM.

Tipo de compuerta de dos entradas	TEM en microsegundos (fe=2Mhz)
AND	60.5
OR	60.5
NAND	59
NOR	59
EOR	59
EORN	60.5

3-5-5 Descripción de los ML que realizan compuertas de cuatro entradas y CEN asociado.

El PLM puede realizar seis tipos de compuertas lógicas de cuatro entradas y estas son de tipo: AND, OR, NAND, NOR, OR EXCLUSIVA y OR EXCLUSIVA NEGADA, teniéndose además la capacidad de preinversión en las entradas que el usuario desee. En la figura 3.12 se ilustra en forma genérica este ML, debiendo el mismo ser declarado en el subprograma principal, siendo la sintaxis para declararlo la siguiente:

COMP#N X0I0J0, X1I1J1, X2I2J2, X3I3J3, XSISJS, ABCD

Donde:

COMP es una cadena que puede ser AND4, OR4, NAND4, NOR4, EOR4 o EORN4, esto de acuerdo al tipo de compuerta que se desee realizar.

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

X0 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.

I0 denota el número de grupo que corresponda a la VB declarada como entrada E0 a la compuerta.

J0 denota el número de bit dentro del grupo I0, asociado a la variable de entrada E0.

X1 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.

I1 denota el número de grupo que corresponda a la VB declarada como entrada E1 a la compuerta.

J1 denota el número de bit dentro del grupo I1, asociado a la variable de entrada E1

X2 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.

I2 denota el número de grupo que corresponda a la VB declarada como entrada E2 a la compuerta.

J2 denota el número de bit dentro del grupo I2, asociado a la variable de entrada E2.

X3 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.

I3 denota el número de grupo que corresponda a la VB declarada como entrada E3 a la compuerta.

J3 denota el número de bit dentro del grupo I3, asociado a la variable de entrada E3.

XS 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.

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

JS denota el número de bit dentro del grupo IS, 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 el dígito "D" deberá ser uno.

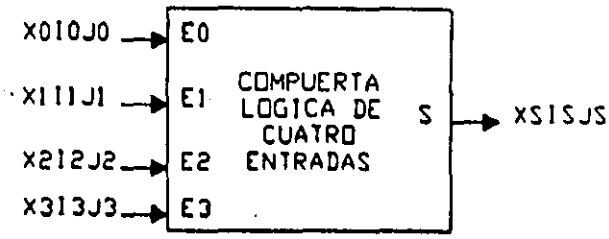


Figura 3.12 Representación genérica de una compuerta de cuatro entradas realizada por el PLM

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

Ejemplo 3.5

Supóngase que se necesita realizar con el PLM 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 las entradas E1 y E0 tengan preinversión y que el número de asignación sea 8, véase la figura 3.13; en este caso, se deberá usar la siguiente sintaxis:

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

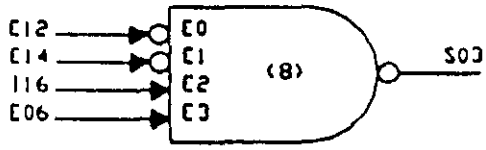


Figura 3.13.- Ejemplo de compuerta NAND de cuatro entradas con preinversión en las entradas E0 y E1, la declaración sintáctica correspondiente es:
NAND4#8 E12, E14, I16, E06, S03, 1100

Descripción del CEN asociado con los ML que realizan compuertas de cuatro entradas

En la figura 3.14 se muestra el flujo de ejecución del tramo de código correspondiente a una compuerta de cuatro entradas.

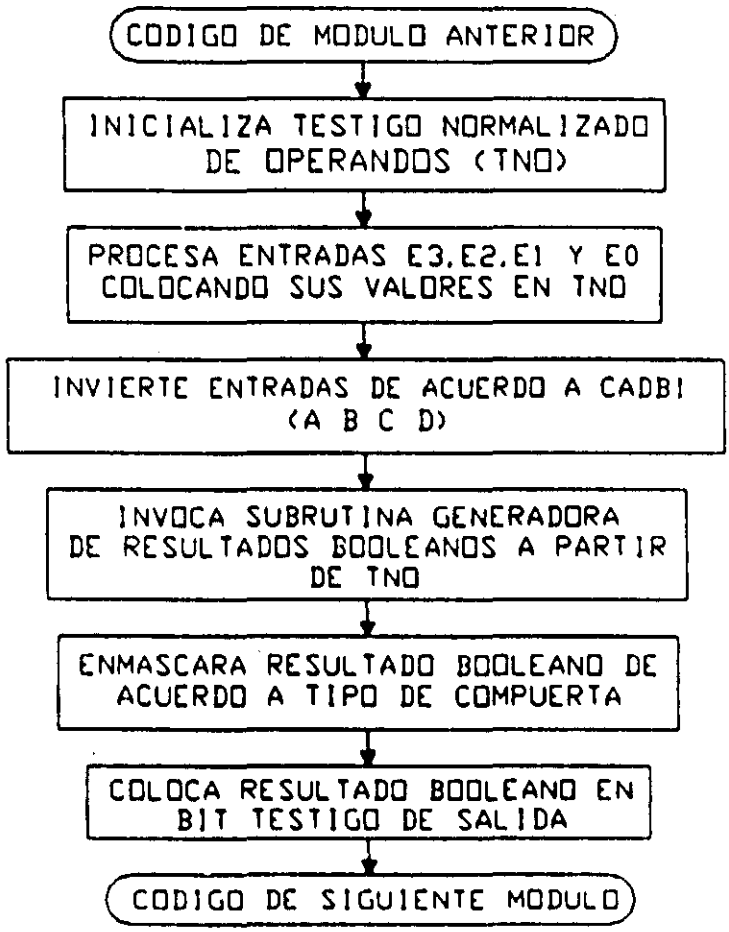


Figura 3.14 Flujo de ejecución del tramo de código empleado para realizar una compuerta de cuatro entradas

El software de traducción emplea un CEN genérico denominado OPER4UN, para obtener el TC requerido por una determinada compuerta de cuatro entradas declarada por el usuario en el subprograma principal; el CEN OPER4UN es:

CEN OPER4UN

```
Inicializa testigo normalizado de operandos (TNO)
0000 8670                                   LDAA #$70
0002 A7F0                                   STAA $F0,X
Procesa entrada E0
0004 A600                                   LDAA $00,X
0006 8401                                   ANDA #$01
0008 2703                                   BEQ ALFA1
000A 1CF001                                 BSET $F0,X,01
Procesa entrada E1
000D A600                ALFA1:             LDAA $00,X
000F 8402                                   ANDA #$02
```

```
0011 2703                                          BEQ ALFA2
0013 1CF002                                        BSET $F0,X,02
Procesa entrada E2
0016 A600              ALFA2:                      LDAA $00,X
0018 8401                                          ANDA #$01
001A 2703                                          BEQ ALFA3
001C 1CF004                                        BSET $F0,X,04
Procesa entrada E3
001F A600              ALFA3:                      LDAA $00,X
0021 8402                                          ANDA #$02
0023 2703                                          BEQ ALFA4
0025 1CF008                                        BSET $F0,X,08
Invierte entradas de acuerdo a CADBI (ABCD)
0028 8600              ALFA4:                      LDAA #$00
002A A8F0                                          EORA $F0,X
002C A7F0                                          STAA $F0,X
Invoca subrutina generadora de resultados booleanos (SGRB)
002E BDFF80                                        JSR $FF80
Coloca resultado booleano en bit testigo de salida
0031 8401                                          ANDA #$01
0033 2705                                          BEQ ALFA5
0035 1C0001                                        BSET $00,X,01
0038 2003                                          BRA ALFA6
003A 1D0001            ALFA5:                      BCLR $00,X,01
003D 01                ALFA6:                      NOP
```

La subrutina generadora de resultados booleanos (SGRB) está cargada en memoria fija (EEPROM) de la CC a partir de la dirección FF80H, el código en ensamblador de la misma puede verse en el tema 3-5-3 de este capítulo, donde se explica el CEN OPER2UN

Para generar el resultado booleano, la subrutina SGRB usa tablas cargadas en memoria fija (EEPROM) de la CC, empleándose la localidad de RAM cuya dirección es 01F0H para almacenar el testigo normalizado de operandos (TNO).

En la tabla 3.8A se muestra la TAB asociada con el CEN OPER4UN y en la tabla 3.8B se aprecia el valor que deben tomar, de acuerdo con el tipo de VB que corresponda a la entrada y salida de la compuerta, parámetros que aparecen en la tabla 3.8A. El tiempo de ejecución para una compuerta de cuatro entradas variará de acuerdo con las entradas a la misma, en la tabla 3.8C se muestra el valor máximo que puede tomar este tiempo, para los distintos tipos de compuertas realizables por el PLM, esto considerando que la frecuencia de la señal de reloj E del microcontrolador es de 2 MHz, que es el caso de la CC del PLM

Tabla 3.8A Asignación de bytes asociada con el CEN OPER4UN, empleado para obtener el código requerido por los ML que realizan compuertas de cuatro entradas.

Valor numérico (VN) generado por el software de traducción	Byte en CEN genérico para realizar compuertas de cuatro entradas, al que se debe asignar el valor numérico VN
VN=2I0+BUF0	B5
VN=2^J0	B7
VN=2I1+BUF1	B14
VN=2^J1	B16
VN=2I2+BUF2	B23
VN=2^J2	B25
VN=2I3+BUF3	B32
VN=2^J3	B34
VN=2IS+BUFS	B54, B59
VN=2^JS	B55, B60
VN=1 (AND), 2 (OR), 4 (NAND), 8 (NOR), 16 (EOR), 32 (EORN)	B50
VN=15-X, X=valor numérico de CADBI	B41

Tabla 3.8B Valores de los parámetros BUF0, BUF1, BUF2 y BUFS de la tabla 3.8A

Caracter X0	**BUF0**
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Caracter X1	**BUF1**
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Caracter X2	**BUF2**
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Caracter X3	**BUF3**
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Caracter XS	**BUFS**
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)

Tabla 3.8C Máximo tiempo de ejecución (TEM) del código correspondiente una compuerta de cuatro entradas realizada por el PLM.

Tipo de compuerta de dos entradas	TEM en microsegundos (fe=2Mhz)
AND	68.5
OR	68.5

Tipo de compuerta de dos entradas	TEM en microsegundos (fe=2MHz)
NAND	67
NOR	67
EOR	67
EORN	68.5

3-5-6 Descripción de los ML que realizan Flip-Flops R-S asíncronos y CEN asociado

El PLM puede realizar módulos tipo *latch*, que en la nomenclatura del mismo se denominan como Flip-Flops asíncronos R-S (FFARS), teniéndose para este ML, la capacidad de predefinir el nivel de verificación de las entradas "S" y "R", además de poder predefinir también el nivel que ha de tener la salida cuando ambas entradas se verifican y el valor que se desea que tome la misma al iniciar el programa en SIIL1 que ejecuta el PLM en un momento dado; en la figura 3.15 se ilustra en forma genérica este ML, debiendo el mismo ser declarado en el subprograma principal, siendo la sintaxis para declararlo la siguiente:

FFARS#N XSISJS, XRIR1JR, XQIQJQ, ABCD;

Donde:

N denota el número de Flip-Flop, esto definido por el usuario.

XS podrá ser la letra "e", "s" o "i" mayúscula o minúscula dependiendo esto de que la variable de entrada EST (S) al Flip-Flop sea una VBE, VBS o VBI.

IS denota el número de grupo que corresponda a la VB declarada como entrada "S" al Flip-Flop.

JS denota el número de bit dentro del grupo IS, asociado a la variable de entrada "S".

XR podrá ser la letra "e", "s" o "i" mayúscula o minúscula dependiendo esto de que la variable de entrada RESET (R) al Flip-Flop una VBE, VBS o VBI.

IR denota el número de grupo que corresponda a la VB declarada como entrada "R" al Flip-Flop.

JR denota el número de bit dentro del grupo IR, asociado a la variable de entrada "R"

XQ 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.

IQ denota el número de grupo que corresponda a la VB declarada como salida del Flip-Flop

JQ denota el número de bit dentro del grupo IQ, 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 el dígito "D" deberá ser uno.

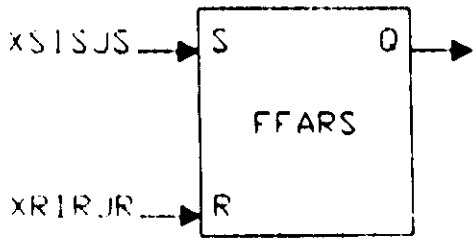


Figura 3.15 Representación genérica del ML que realiza un Flip-Flop asincrono R-S

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

Ejemplo 3.6

Supóngase que se necesita realizar con el PLM 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, deseándose ademas que a

este latch se le asigne el número 22, véase la figura 3.16; en este caso, se deberá usar la siguiente sintaxis:

FFARS#22 E13, E14, S06, 0011;

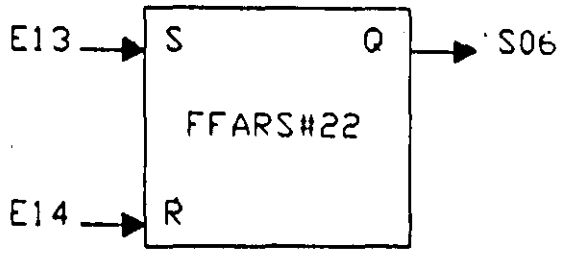


Figura 3.16.- Ejemplo de módulo tipo latch (FFARS) realizado con el PLM, se desea que la entrada S tenga prioridad, requiriéndose que el nivel de verificación de las dos entradas sea bajo y que el estado inicial de la salida sea uno lógico, la declaración sintáctica correspondiente es: FFARS#22 E13, E14, S06, 0011

Descripción del CEN asociado con los módulos de tipo Flip-Flop asíncrono

En la figuras 3.17 se muestra el flujo de ejecución de un módulo tipo latch realizable por el PLM, cuando la entrada S es la que tiene la prioridad y en la figura 3.18 se muestra lo propio cuando la prioridad la tiene la entrada R.

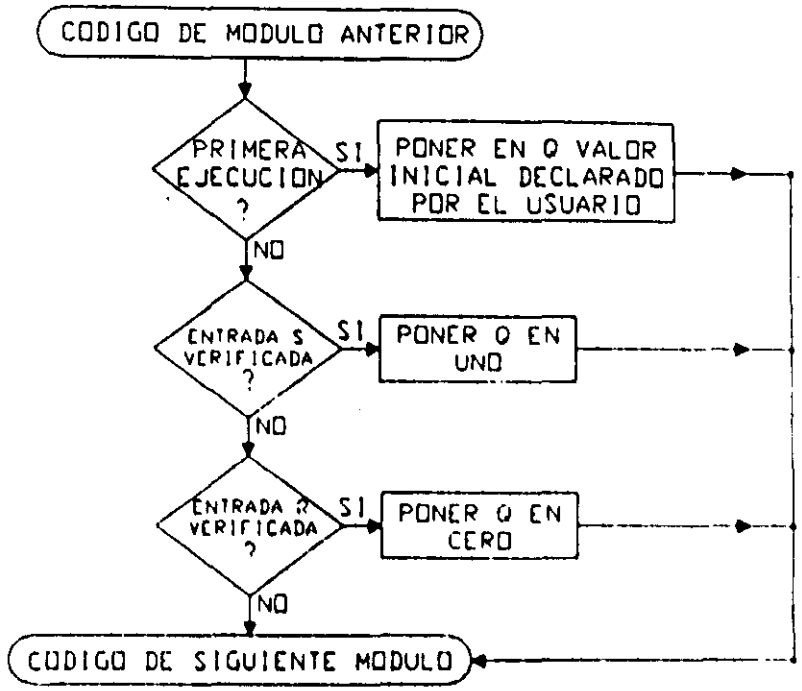


Figura 3.17 Flujo de ejecución del tramo de código empleado para realizar un módulo tipo latch (FFARS), cuando la entrada S tiene prioridad.

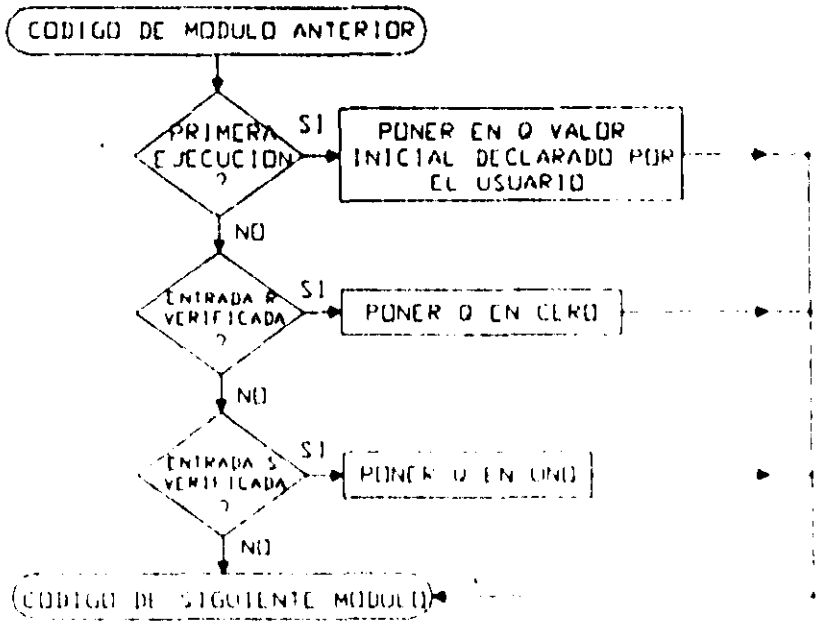


Figura 3.18 Flujo de ejecución del tramo de código empleado para realizar un módulo tipo latch (FFARS), cuando la entrada R tiene prioridad.

El software de traducción emplea un CEN genérico denominado FFARSXX, para obtener el TC requerido por un determinado Flip-Flop asincrono declarado por el usuario en el subprograma principal, el CEN FFARSXX es

CEN FFARSXX

```
Checa por primera ejecución de este código
0000 A6F1                          LDAA $F2,X
0002 2705                          BEQ NOPP
Inicializa Q la primera vez que se ejecuta este código, aqui se muestra
el caso de inicialización en cero
0004 1D0002          INIC:         BCLR $00,X,02
0007 2014                          BRA SALIDA
Checa nivel de entrada S (SET)
0009 A600            NOPP:         LDAA $00,X
000B 8404                          ANDA #$04
000D 2605                          BNE RES
000F 1C0002                        BSET $00,X,02
0012 2009                          BRA SALIDA
Checa nivel de entrada R (RESET)
0014 A600            RES:          LDAA $00,X
0016 8401                          ANDA #$01
0018 2603                          BNE SALIDA
001A 1D0002                        BCLR $00,X,02
001D 01              SALIDA:       NOP
```

El código ensamblado mostrado anteriormente, corresponde al caso de que la prioridad la tenga la entrada S, en otro caso debe haber un intercambio de instrucciones, colocándose primero el código correspondiente al chequeo de la entrada R debiendo ser este seguido por el código que checa la entrada S, esta situación de intercambio esta consignada en la TAB asociada con este CEN (FFARSXX).

En la tabla 3.9A se muestra la TAB asociada con el CEN FFARSXX y en la tabla 3.9B se aprecia el valor que deben tomar, de acuerdo con el tipo de VB que corresponda a las entradas y salida del Flip-Flop, parámetros que aparecen en la tabla 3.9A. El tiempo de ejecución para un Flip-Flop asíncrono varía de acuerdo a distintas condiciones de funcionamiento del mismo, siendo 17 microsegundos su valor máximo, en la tabla 3.9C se muestran los distintos valores que puede tomar la ejecución del código correspondiente a un Flip-Flop bajo diferentes condiciones de funcionamiento del mismo, esto considerando que la frecuencia de la señal de reloj E del microcontrolador es de 2 MHz, que es el caso de la CC del PLM.

Tabla 3.9A Asignación de bytes asociada con el CEN FFARSXX, empleado para obtener el código requerido por los ML que realizan Flip-Flops asíncronos.

Valor numérico (VN) generado por el software de traducción	Byte en CEN genérico para realizar Flip-Flops, al que se debe asignar el valor numérico VN
VN=242 si el Flip-Flop es declarado en el subprograma principal, en otro caso VN=241	B1

Valor numérico (VN) generado por el software de traducción	Byte en CEN genérico para realizar Flip-Flops, al que se debe asignar el valor numérico VN
VN=28 (D=1), 29 (D=0)	B4
VN=2IS+BUFS (C=1), 2IR+BUFR (C=0)	B10
VN=2^JS (C=1), 2^JR (C=0)	B12
VN=2IS+BUFS (C=0), 2IR+BUFR (C=1)	B21
VN=2^JS (C=0), 2^JR (C=1)	B23
VN=2IQ+BUFQ	B5, B16, B27
VN=2^JQ	B6, B17, B28
VN=VERSET* (C=1), VERES* (C=0)	B13
VN=VERES* (C=1), VERSET* (C=0)	B24
VN=28 (C=1), 29 (C=0)	B15
VN=29 (C=1), 28 (C=0)	B26

* VERSET=38 si A=0, 39 si A=1

* VERES=38 si B=0, 39 si B=1

Tabla 3.9B Valores de los parámetros BUFS, BUFR y BUFQ de la tabla 3.9A

Caracter XS	**BUFS**
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Caracter XR	**BUFR**
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Caracter XQ	**BUFQ**
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)

Tabla 3.9C Tiempo de ejecución (Te) en microsegundos del código correspondiente un latch (Flip-Flop asíncrono R-S), bajo distintas condiciones de funcionamiento

Condición de funcionamiento	Tiempo de ejecución (Te) (fc=2Mhz)
Primera ejecución del código	9.5
Ninguna entrada verificada	13.5
Entrada prioritaria verificada	14
Entrada no prioritaria verificada	17

3-5-7 Descripción del ML que realiza un contador de eventos y CEN asociado

El PLM 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 (CUENTAI) como la final (CUENTAF), este ML tiene tres entradas y una salida, véase la figura 3.19, 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 (TF) se verifica cuando la cuenta ha llegado a su valor final.

Para este ML se tiene la capacidad de predefinir los limites 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 declarase en el subprograma temporizado, siendo la sintaxis para declararlo la siguiente:

CONTA#N XDIDJD, XCICJC, XRIRJR, XFIFJF,CUENTAI, CUENTAF, ABCDE;

Donde:

N denota el número de contador de eventos, esto definido por el usuario.

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

ID denota el número de grupo que corresponda a la VB declarada como entrada "D" al contador.

JD denota el número de bit dentro del grupo ID, asociado a la variable de entrada "D".

XC podrá ser la letra "e", "s" o "i" mayúscula o minúscula, dependiendo esto de que la variable de entrada de congelamiento (C) al contador sea una VBE, VBS o VBI.

IC denota el número de grupo que corresponda a la VB declarada como entrada "C" al contador.

JC denota el número de bit dentro del grupo IC, asociado a la variable de entrada "C"

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

IR denota el número de grupo que corresponda a la VB declarada como entrada "R" al contador.

JR denota el número de bit dentro del grupo IR, asociado a la variable de entrada "R".

XF podrá ser la letra "s" o "i" mayúscula o minúscula, dependiendo esto de que la variable de salida "TF" del contador sea una VBS o VBI.

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

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

CUENTAI denota al valor de la cuenta inicial, debiendo el mismo estar comprendido entre cero 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.

CUENTAF denota el valor de la cuenta final, debiendo el mismo estar comprendido entre cero 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.

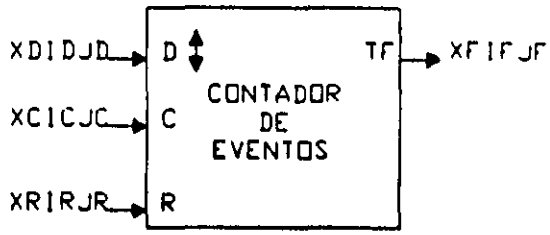


Figura 3.19 Representación genérica de un módulo contador de eventos realizable en el PLM

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

Ejemplo 3.7

Supóngase que se necesita realizar con el PLM un módulo contador de eventos ascendente, con intervalo de cuenta comprendido entre cero y siete y testificación de fin 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, descándose además que a este contador se le asigne el número 14, véase la figura 3.20; en este caso, se deberá usar la siguiente sintaxis:

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

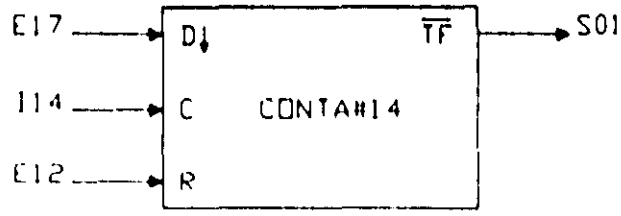


Figura 3.20.- Ejemplo de módulo contador de eventos ascendente realizado con el PLM, descrito en el ejemplo 3.4.

En la figura 3.21 se ilustran diagramas de tiempo asociados con el contador de eventos aquí ejemplificado. Cabe señalar aquí, que para la correcta operación del contador de eventos, se requiere que el intervalo de tiempo entre dos flancos consecutivos sea mayor de 10 ms.

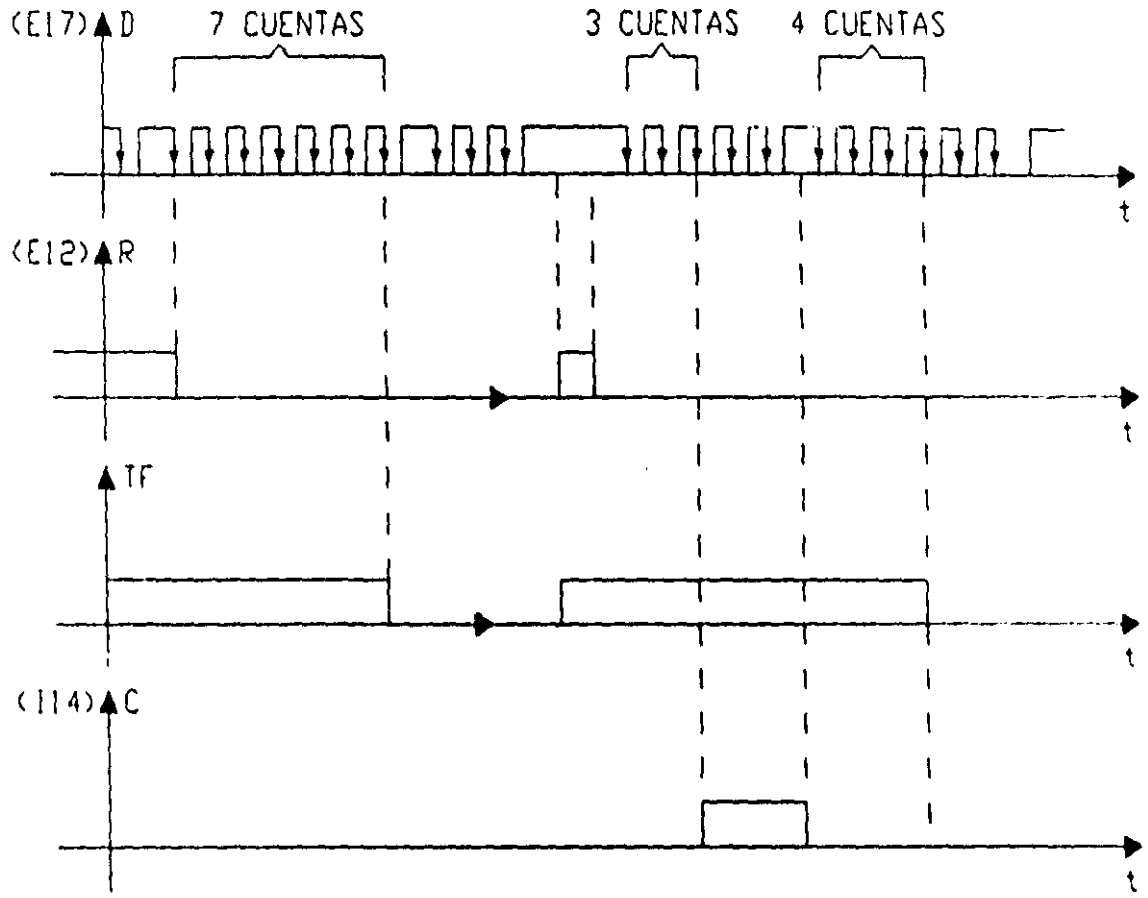


Figura 3.21.- Diagramas de tiempo asociados con el temporizador mostrado en la figura 3.20

Descripción del CEN asociado con los módulos contadores de eventos

En la figuras 3.22 se muestra el flujo de ejecución de un módulo contador de eventos realizable por el PLM.

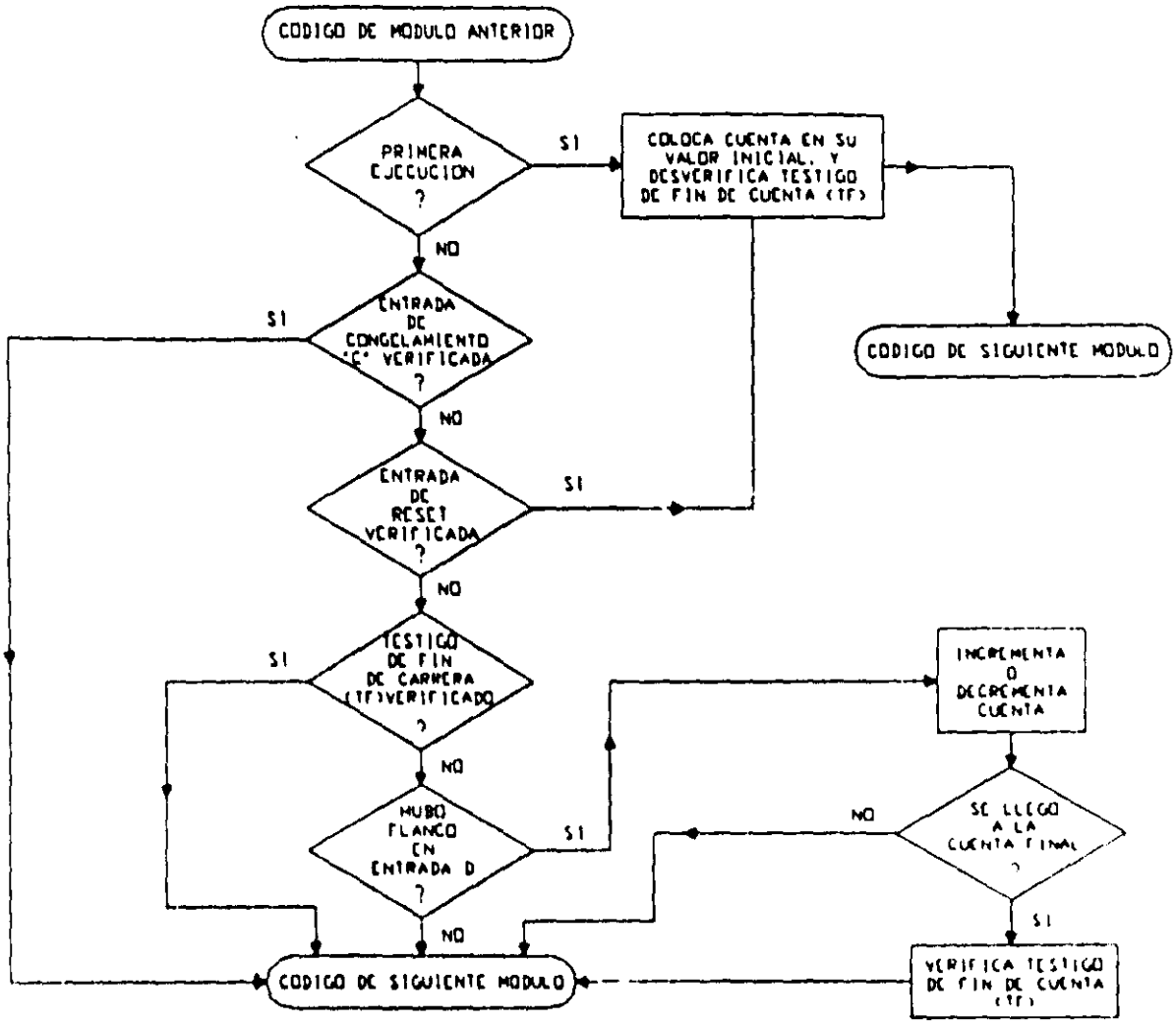


Figura 3.22.- Flujo de ejecución del tramo de código empleado para realizar un módulo contador de eventos.

El software de traducción emplea un CEN genérico denominado CONTAXX, para obtener el TC requerido por un determinado contador de eventos declarado por el usuario en el subprograma temporizado; el CEN CONTAXX es:

CEN CONTAXX

```
 Checa por primera ejecución de este código
 0000 A6F1                              LDAA $F1,X
 0002 270B                              BEQ NOPP
 Accionamiento ejecutable en la primera ejecución o al verificarse la
entrada de RESET
 0004 1D0002          RESET:            BCLR $00,X,02
 0007 CC0000                            LDD #$0000
 000A FD0300                            STD $0300
 000D 2035                              BRA SALIDA
 Checa entrada "C" de congelamiento
 000F A600            NOPP:             LDAA $00,X
 0011 8404                              ANDA #$04
```

```
0013 272F                              BEQ SALIDA
Checa nivel de entrada de reset
0015 A600                              LDAA $00,X
0017 8401                              ANDA #$01
0019 27E9                              BEQ RESET
Checa nivel de testigo de fin de cuenta (TF)
001B A640                              LDAA $40,X
001D 8402                              ANDA #$02
001F 2623                              BNE SALIDA
Checa por flanco en entrada D
0021 A600          AAA:                LDAA $00,X
0023 8401                              ANDA #$01
0025 E601                              LDAB $01,X
0027 C401                              ANDB #$01
0029 11                                CBA
002A 2318                              BLS SALIDA
Invoca subrutina de incremento/decremento
002C 8D02                              BSR INCREM
Salta a salida de este módulo
002E 2014                              BRA SALIDA
Subrutina de incremento/decremento
0030 18FE0300      INCREM:             LDY $0300
0034 1808                              INY
0036 18FF0300                          STY $0300
003A 188CAA55                          CPY #$AA55
003E 2603                              BNE TER
0040 1C0201        VERIFTES:           BSET $02,X,01
0043 39            TER:                RTS
0044 A600          SALIDA:             NOP
```

En la tabla 3.10A se muestra la TAB asociada con el CEN CONTAXX y en la tabla 3.10B se aprecia el valor que deben tomar, de acuerdo con el tipo de VB que corresponda a las entradas y salida del contador de eventos, parámetros que aparecen en la tabla 3.10A. El tiempo de ejecución para este ML varía de acuerdo a distintas condiciones de funcionamiento del mismo, en la tabla 3.10C se muestra este tiempo apreciándose que el tiempo máximo de ejecución de este módulo es 50.5 microsegundos, esto considerando que la frecuencia de la señal de reloj E del microcontrolador es de 2 MHz, que es el caso de la CC del PLM.

Tabla 3.10A Asignación de bytes asociada con el CEN CONTAXX, empleado para obtener el código requerido por los ML que realizan contadores de eventos.

Valor numérico (VN) generado por el software de traducción	Byte en CEN genérico para realizar contadores de eventos, al que se debe asignar el valor numérico VN
VN=29 si TF es verificado en alto (E=1) VN=28 si TF es verificado en bajo (E=0)	B4
VN=2IF+BUFF	B5, B28, B65
VN=2^JF	B6, B30, B66

Continuación Tabla 3.10A.

VN= byte alto de CUENTAI	B8
VN= byte bajo de CUENTAI	B9
VN= byte alto de DIRCON*	B11, B50, B56
VN= byte bajo de DIRCON*	B12, B51, B57
VN= 2IC+BUFC	B16
VN= 2^JC	B18
VN= 39 si el congelamiento es en bajo (B=0) VN= 38 si el congelamiento es en alto (B=1)	B19
VN= 39 si el RESET es en bajo (C=1) VN= 38 si el RESET es en alto (C=0)	B25
VN=38 si TF es verificado en alto (E=1) VN=39 si TF es verificado en bajo (E=0)	B31
VN= 2IR+BUFR	B22
VN= 2^JR	B24
VN= 2ID+BUFD	B34
VN= 2^JD	B36, B40
VN= 2ID+1+BUFD	B38
VN= 35, cuenta por flanco de subida (A=1) VN= 36, cuenta por flanco de bajada (A=0)	B42
VN=28 si TF es verificado en alto (E=1) VN=29 si TF es verificado en bajo (E=0)	B64
VN= 8 para cuenta ascendente (D=1) VN= 9 para cuenta descendente (D=0)	B53
VN= byte alto de CUENTAF	B60
VN= byte bajo de CUENTAF	B61

* DIRCON=768+2N, donde N es el número asignado al contador.

Tabla 3.10B Valores de los parámetros BUFD, BUFC, BUFR y BUFF de la tabla 3.10A

Caracter XD	**BUFD**
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Caracter XC	**BUFC**
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Caracter XR	**BUFR**
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Continuación de tabla 3.10B en página 97	

Caracter XF	BUFF
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)

Tabla 3.10C Tiempo de ejecución (Te en microsegundos), del código correspondiente a un contador de eventos, bajo diferentes condiciones en las entradas al mismo

Condición de funcionamiento	Tiempo de ejecución (Te), (fe=2Mhz)
Primera ejecución del código	15
Entrada de congelamiento "C" verificada	10.5
RESET verificado	24
Testigo de fin de cuenta "TF" verificado	19.5
Detección de flanco en entrada "D", sin que esto lleve la cuenta a su valor final	47
Detección de flanco en entrada "D", llevando esto la cuenta a su valor final	50.5
En espera de flanco en entrada "D"	29.5

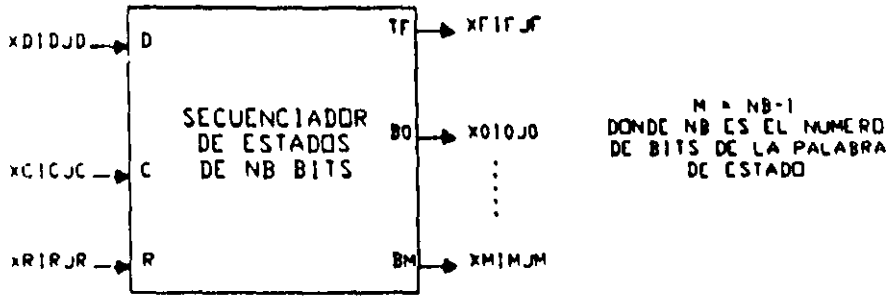
3-5-8 Descripción del ML secuenciador de estados de NBxNE y su CEN asociado

El PLM puede realizar módulos secuenciadores de estados, el número de bits de la palabra de estado puede ir de uno a ocho, tanto el número de estados como los diferentes valores que presenten los mismos son programables por el usuario, la sucesión de estados deseada debe ser declarada por el usuario, en renglones del programa fuente subsecuentes al que contiene la declaración sintáctica correspondiente; de esta manera, la declaración de un secuenciador de estados ocupara mas de un renglón en el programa fuente en SIIL1; las declaraciones de los valores deseados para los estados pueden hacerse en formato binario o hexadecimal. El software de traducción limita a 1000 el número de estados asociados con un secuenciador, esto aún cuando en la mayoría de las aplicaciones prácticas este tope seria sensiblemente menor.

Un secuenciador presentará tres entradas y NB+1 salidas, donde NB es el número de bits en la palabra de estado; la primera entrada es sensible a flancos y se denomina como "D", al detectarse un flanco en ella se coloca en las salidas asociadas el siguiente estado de la lista que el usuario haya declarado, si el estado colocado es el último de la lista se verifica una salida denominada como "TF" (testigo de fin de carrera), la segunda entrada, se denomina "C" (entrada de congelamiento) y al verificarse, hace que el secuenciador permanezca en el estado presente sin responder a las otras dos entradas; la tercera entrada se

denomina "R" (RESET) y al verificarse, hace que el secuenciador presente el estado inicial y desverifique la salida "TF", la primera vez que se ejecuta el código se invoca el accionamiento de RESET; en la figura 3.23 se muestra la representación genérica de un módulo secuenciador de estados.

Figura 3.23.- Representación genérica de un módulo secuenciador de estados de NB bits realizable en el PLM



Para este ML se tiene la capacidad de predefinir el número de bits de la palabra de estado, el número de estados que se han de secuenciar, el tipo de flanco que hace que se presente el siguiente estado, el nivel de verificación de las entradas de RESET y congelamiento, el nivel de verificación de la salida testigo de colocación de estado final; una vez que se coloca el estado final el secuenciador no responde a los flancos, para reiniciar el ciclo hay que verificar la entrada de RESET.

La declaración de un modulo secuenciador ocupará varios renglones en el programa fuente en SIIL1, en el primero de ellos se especificarán los siguientes parámetros: número de bits de la palabra de estado, número de secuenciador, VB seleccionada como entrada de disparo, VB seleccionada como entrada de congelamiento, VB seleccionada como entrada de RESET, VB seleccionada como salida testigo de fin de carrera, las VB seleccionadas como salidas para que las mismas presenten los valores booleanos que correspondan a la palabra de estado, el número de estados que manejará el secuenciador, el tipo de flanco al que responderá la entrada de disparo, el nivel de verificación de la entrada de congelamiento, el nivel de verificación de la entrada de RESET y el nivel de verificación de la salida testigo de fin de carrera.

En los renglones subsecuentes al primero, se deben definir cada uno de los valores sucesivos deseados para la palabra de estado, pudiéndose hacer esto ya sea en formato hexadecimal o binario, el usuario podrá declarar uno o más valores de estados en cada renglón, cada renglón de especificación de valores de estados deberá tener un caracter "#" en la primera columna, para especificar el último renglón de datos el mismo habrá de iniciarse con dos caracteres "#" seguidos, colocándose el primero de ellos en la primera columna.

Este módulo debe declararse en el subprograma temporizado, la sintaxis genérica para declararlo es:

SECNB#N XDIDJD, XCICJC, XRIRJR, XFIFJF, EVBPE, NE, ABCD;

[EF1][ESTADO1], [EF2][ESTADO2], [EF3][ESTADO3], ...[EFq][ESTADOq];

[EFp][ESTADOp], ...[EFr][ESTADOr];

.

.

.

[EFu][ESTADOu],.....[EFne][ESTADOne];

A continuación se explica el significado de las literales que aparecen en el primer renglón de la declaración genérica de un módulo secuenciador.

NB denota el número de bits de la palabra de estado, debiendo el mismo estar comprendido entre uno y ocho, esto definido por el usuario.

N denota el número de secuenciador, esto definido por el usuario.

XD podrá ser la letra "e", "s" o "i" mayúscula o minúscula dependiendo esto de que la variable de entrada D al secuenciador sea una VBE, VBS o VBI

ID denota el número de grupo que corresponda a la VB declarada como entrada "D" al secuenciador.

JD denota el número de bit dentro del grupo ID, asociado a la variable de entrada "D"

XC podrá ser la letra "e", "s" o "i" mayúscula o minúscula, dependiendo esto de que la variable de entrada de congelamiento (C) al secuenciador sea una VBE, VBS o VBI.

IC denota el número de grupo que corresponda a la VB declarada como entrada "C" al secuenciador

JC denota el número de bit dentro del grupo IC, asociado a la variable de entrada "C".

XR podrá ser la letra "e", "s" o "i" mayúscula o minúscula, dependiendo esto de que la variable de entrada RESET (R) al secuenciador sea una VBE, VBS o VBI.

IR denota el número de grupo que corresponda a la VB declarada como entrada "R" al secuenciador.

JR denota el número de bit dentro del grupo IR, asociado a la variable de entrada "R".

XF podrá ser la letra "s" o "i" mayúscula o minúscula, dependiendo esto de que la variable de salida "TF" del secuenciador sea una VBS o VBI.

IF denota el número de grupo que corresponda a la VB declarada como salida del secuenciador.

JF denota el número de bit dentro del grupo IF, asociado a la variable de salida del secuenciador.

EVBPE es un vector de NB elementos separados por comas, que especifica que variables booleanas se desea que sean cada uno de los bits de la palabra de estado, por lo tanto, EVBPE presentará la siguiente forma:

XPIPJP,.....XLILJL,.....X0I0J0

con P=NB-1 y:

XL (L=0, 1,.....NB-1), podrá ser la letra "s" o "i" mayúscula o minúscula, dependiendo esto de que el bit L de la palabra de estado del secuenciador sea una VBS o VBI.

IL (L=0, 1,.....NB-1), denota el número de grupo que corresponda a la VB declarada como bit L de la palabra de estado del secuenciador.

JL (L=0, 1,.....NB-1), denota el número de bit dentro del grupo IL, asociado a con el bit L de la palabra de estado

NE denota el número de estados que se desea presente el secuenciador y el mismo deberá ser mayor o igual que dos.

A es un digito binario, que habrá de ser cero, si se desea que se coloque el estado siguiente para flancos de bajada en la entrada de disparo "D", en otro caso el digito "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 cero, si se desea que el nivel de verificación de la salida testigo de fin de carrera (TF) sea bajo, en otro caso el dígito "D" deberá ser uno.

En lo que toca a los renglones donde se declaran los valores que se desea tomen los estados que presentará el secuenciador, a continuación se explica el significado de los términos genéricos que los mismos contienen:

EFi (i=1, 2,.....NE), deberá ser la letra "B" si se desea que la especificación del estado "i" sea en formato binario, en caso de que se desee que la misma sea en formato hexadecimal EFi deberá ser la letra H; si el usuario escoge el formato binario deberá escribir la palabra de estado correspondiente limitándose al número de bits de la misma, por otro lado si el formato escogido fue el hexadecimal, el usuario deberá escribir un byte, en caso de que la longitud de la palabra de estado sea menor a ocho bits el valor binario de los bits no usados será irrelevante (don't care).

La literal "q", denota el número correspondiente al último estado declarado en el primer renglón de especificación de valores de estados.

La literal "p", denota el número correspondiente al primer estado declarado en el segundo renglón de especificación de valores de estados.

La literal "r", denota el número correspondiente al último estado declarado en el segundo renglón de especificación de valores de estados.

La literal "u", denota el número correspondiente al primer estado declarado en el último renglón de especificación de valores de estados.

Las literales "ne", denotan el número correspondiente al último estado de la secuencia deseada

ESTADOi (i=1, 2, NE), denota el valor deseado para el estado i de la secuencia que se desea presente el secuenciador

Nótese que el número de renglones a emplear, para especificar la secuencia de estados deseada, es variable y se deja al usuario libertad para acomodar en el programa

fuente esta información, ya que se podrá especificar desde un solo estado en cada renglón hasta los que puedan contenerse en el ancho de la pantalla.

A continuación se muestra un ejemplo sobre como declarar un módulo secuenciador de estados, en un programa fuente en SIIL1.

Ejemplo 3.8

Supóngase que se necesita realizar con el PLM un módulo secuenciador de estados de tres bits por doce estados, al cual se le asignará el número nueve; se desea que las entradas de disparo, congelamiento y RESET sean respectivamente las VB E02, I00, y E00, para la salida testigo de fin de carrera ha de usarse la VB S17, las salidas asociadas con los tres bits de la palabra de estado en orden decreciente de significancia, deberán ser las VB I34, S03 y S01; se requiere que la entrada de disparo responda a flancos de bajada, las entradas de congelamiento y RESET y la salida TF sean verificadas en bajo; la lista de estados a secuenciar se muestra a continuación en formato binario:

Estado 01 000
Estado 02 010
Estado 03 110
Estado 04 111
Estado 05 001
Estado 06 010
Estado 07 111
Estado 08 000
Estado 09 001
Estado 10 010
Estado 11 110
Estado 12 111

Para fines ilustrativos, se usará el formato binario para declarar los primeros cinco estados de la lista, empleándose el formato hexadecimal para los demás, colocándose ceros en las posiciones de bit irrelevantes, una posible forma para declarar este secuenciador es la siguiente:

```
                SEC3#9 E02, I00, E00, S17, I34, S03, S01, 12, 0010;
#    B000, B010, B110, B111;
#    B001, H02, H07, H00, H01, H02,
##   H06, H07;
```

En la figura 3.24 se ilustra la representación en forma de bloque para el secuenciador de este ejemplo.

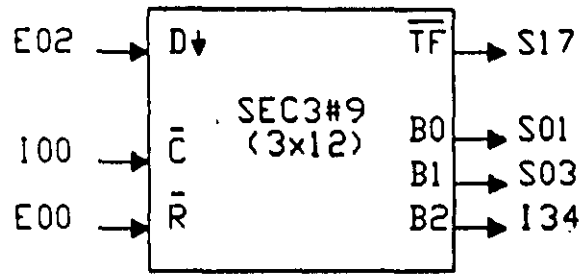


Figura 3.24.- Módulo lógico que realiza el secuenciador de estados de 3x12 del ejemplo 3.8

Descripción del CEN asociado con el ML secuenciador de estados

En la figuras 3.25 se muestra el flujo de ejecución de un módulo secuenciador de estados, el software de traducción, coloca los datos acerca de los estados a secuenciar, a partir de la dirección siguiente al final del código ejecutable obtenido a partir del CEN que corresponde a este tipo de módulo, de esta manera, el código ejecutable y los datos sobre los estados a colocar integran una sola lista de bytes que es generada por el software de traducción, colocando esta a partir de la dirección de memoria que proceda, cabe señalar aquí que el software se traducción denomina DIRBM a esta dirección de colocación, en la figura 3.26 se ilustra la estructura que tendría la lista de bytes asociada con un módulo secuenciador de estados realizable por el PLM.

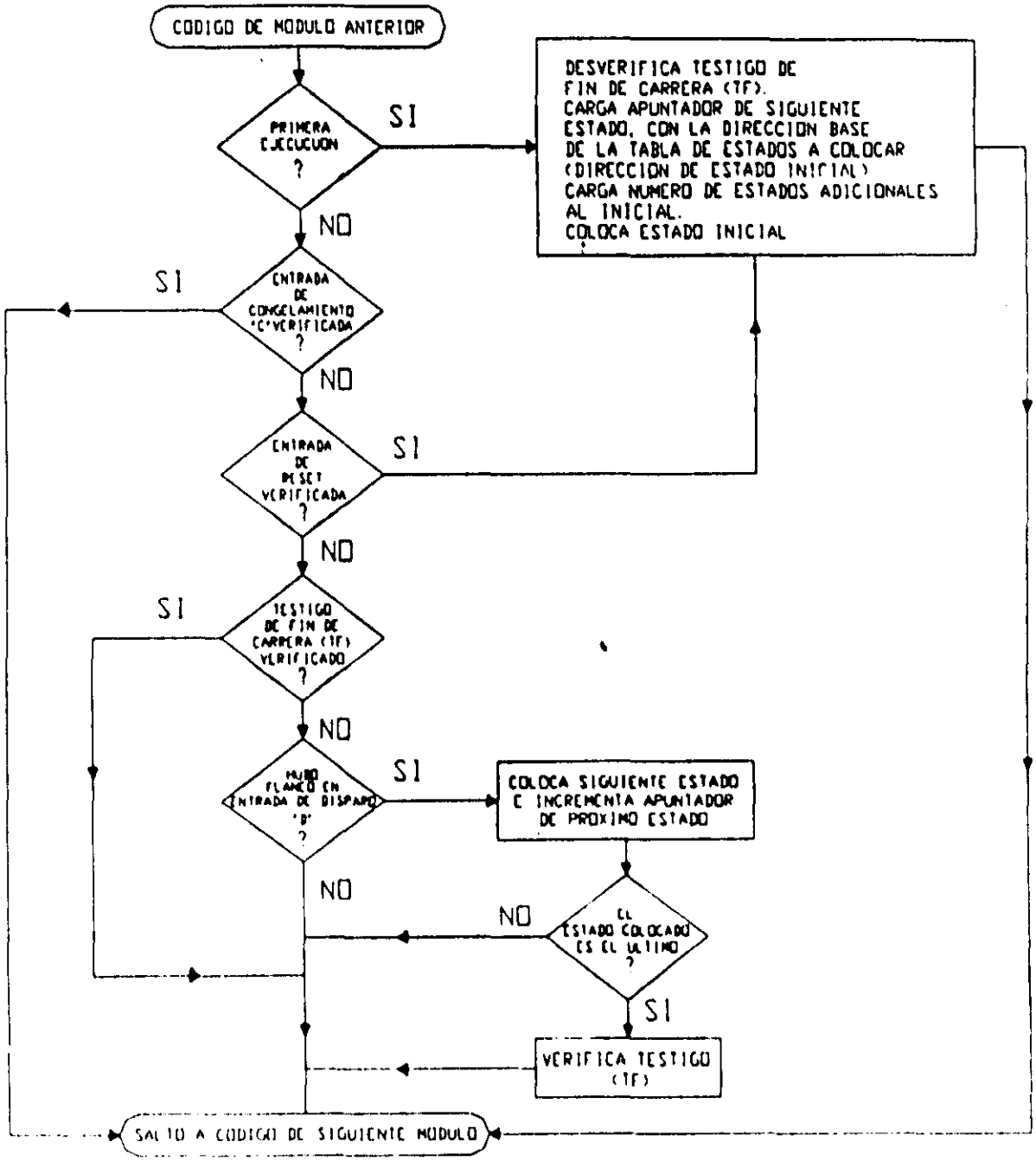


Figura 3.25.- Flujo de ejecución del tramo de código empleado para realizar un módulo secuenciador de estados.

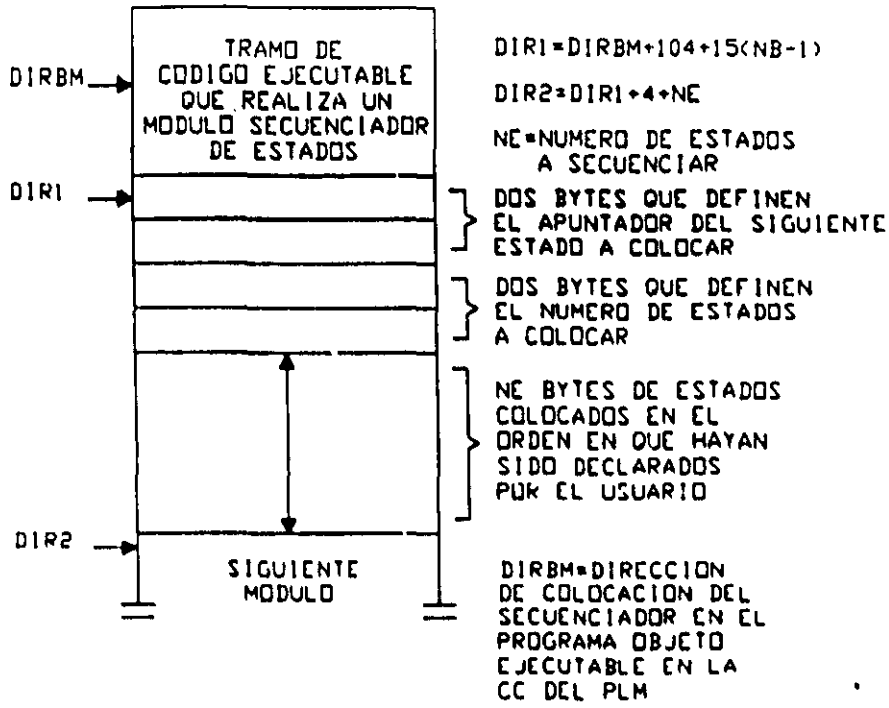


Figura 3.26.- Tramo de código y datos asociados con estados a secuenciar, correspondientes a un secuenciador de estados de NB bits por NE estados.

Para generar el TC asociado con un determinado secuenciador de estados, el software se traducción emplea un CEN diferente para cada tamaño diferente de palabra de estado (NB), sin embargo todos ellos tienen en común la parte del código que monitorea las entradas, manifestándose diferencias sólo en la suburutina que coloca el estado siguiente ya que el tamaño de la misma varía de acuerdo el número de bits de la palabra de estado. Con el objeto de no caer en redundancias se describirá aquí un CEN genérico denominado como SECNBXNE, a partir del cual se obtendría el TC asociado con los secuenciadores de estados, que pudieran haber sido declarados por el usuario en el subprograma temporizado; el CEN SECNBXNE es:

CEN SECNBXNE

```
Checa por primera ejecución de este código
0000 A6F1                                   LDAA $F1,X
0002 2713                                   BEQ NOPP
Accionamiento para la primera ejecución de este código
0004 1D0002          RESET:                 BCLR $00,X,02
0007 CCC1AA                                 LDD #$C'AA
000A FDC150                                 STD $C150
000D CC00F0                                 LDD #$00F0
0010 FDC152                                 STD $C152
```

```
Pone estado inicial
0013 8D39                                   BSR PONESTADO
0015 2030                                   BRA SALIDA
Checa entrada de congelamiento
0017 A600          NOPP:                    LDAA $00,X
0019 8404                                   ANDA #$04
001B 272A                                   BEQ SALIDA
Checa entrada de restablecimiento (RESET)
001D A600                                   LDAA $00,X
001F 8401                                   ANDA #$01
0021 27E1                                   BEQ RESET
Checa testigo de fin de carrera (TF)
0023 A640                                   LDAA $40,X
0025 8402                                   ANDA #$02
0027 261E                                   BNE SALIDA
Checa si hubo flanco en entrada de disparo (D)
0029 A600          AAA:                     LDAA $00,X
002B 8401                                   ANDA #$01
002D E601                                   LDAB $01,X
002F C401                                   ANDB #$01
0031 11                                     CBA
0032 2313                                   BLS SALIDA
Pone siguiente estado al detectarse que hubo flanco en entrada D
0034 8D18                                   BSR PONESTADO
Actualiza testigo de estados por colocar
0036 18FEC152                               LDY $C152
003A 1809                                   DEY
003C 18FFC152                               STY $C152
0040 2702                                   BEQ VERIFTES
0042 2003                                   BRA SALIDA
Verifica testigo de fin de carrera después que se ha colocado el último
estado
0044 1C0201        VERIFTES:                BSET $02,X,01
Enruta hacia código de siguiente módulo
0047 7EC200        SALIDA:                  JMP $C200
004A 01010101      Bytes de ajuste
Aquí inicia la subrutina de colocación de estado, mostrándose el caso de
que el mismo sea de ocho bits, en caso de que el número de bits sea
menor, varios de los tramos de código mostrados no aparecerán

004E 18FEC150  PONESTADO: LDY $C150
Coloca bit 0
0052 18A600        SIGUE0:                  LDAA $00,Y
0055 8401                                   ANDA #$01
0057 2705                                   BEQ CLR0
0059 1C4001                                 BSET $40,X,01
005C 2003                                   BRA SIGUE1
005E 1D4001        CLR0:                    BCLR $40,X,01
Coloca bit 1
0061 18A600        SIGUE1:                  LDAA $00,Y
0064 8402                                   ANDA #$02
0066 2705                                   BEQ CLR1
0068 1C4002                                 BSET $40,X,02
006B 2003                                   BRA SIGUE2
006D 1D4002        CLR1:                    BCLR $40,X,02
Coloca bit 2
0070 18A600        SIGUE2:                  LDAA $00,Y
0073 8404                                   ANDA #$04
0075 2705                                   BEQ CLR2
0077 1C4004                                 BSET $40,X,04
```

```
007A 2003                              BRA SIGUE3
007C 1D4004          CLR2:             BCLR $40,X,04
Coloca bit 3
007F 18A600          SIGUE3:           LDAA $00,Y
0082 8408                              ANDA #$08
0084 2705                              BEQ CLR3
0086 1C4010                            BSET $40,X,10
0089 2003                              BRA SIGUE4
008B 1D4010          CLR3:             BCLR $40,X,10
Coloca bit 4
008E 18A600          SIGUE4:           LDAA $00,Y
0091 8410                              ANDA #$10
0093 2705                              BEQ CLR4
0095 1C4020                            BSET $40,X,20
0098 2003                              BRA SIGUE5
009A 1D4020          CLR4:             BCLR $40,X,20
Coloca bit 5
009D 18A600          SIGUE5:           LDAA $00,Y
00A0 8420                              ANDA #$20
00A2 2705                              BEQ CLR5
00A4 1C4040                            BSET $40,X,40
00A7 2003                              BRA SIGUE6
00A9 1D4040          CLR5:             BCLR $40,X,40
Coloca bit 6
00AC 18A600          SIGUE6:           LDAA $00,Y
00AF 8440                              ANDA #$40
00B1 2705                              BEQ CLR6
00B3 1C4080                            BSET $40,X,80
00B6 2003                              BRA SIGUE7
00B8 1D4080          CLR6:             BCLR $40,X,80
Coloca bit 7
00BB 18A600          SIGUE7:           LDAA $00,Y
00BE 8480                              ANDA #$80
00C0 2705                              BEQ CLR7
00C2 1C4080                            BSET $40,X,80
00C5 2003                              BRA TERMINA
00C7 1D4080          CLR7:             BCLR $40,X,80
Actualiza apuntador de estado siguiente
00CA 1808            TERMINA:          INY
00CC 18FFC150                          STY $C150
00D0 39                                RTS
```

En la tabla 3.11A se muestra la TAB asociada con el CEN SECNBXNE y en la tabla 3.11B se aprecia el valor que deben tomar, de acuerdo con el tipo de VB que corresponda a las entradas y salidas del secuenciador de estados, parámetros que aparecen en la tabla 3.11A. El tiempo de ejecución para este ML varía de acuerdo a distintas condiciones de funcionamiento del mismo, en la tabla 3.11C se muestra este tiempo apreciándose que el tiempo máximo de ejecución de este módulo es 135 microsegundos presentándose para un secuenciador con estados de ocho bits, esto considerando que la frecuencia de la señal de reloj E del microcontrolador es de 2 MHz, que es el caso de la CC del PLM.

Tabla 3.11A Asignación de bytes asociada con el CEN SECNBXNE, empleado para obtener el código requerido por los ML que realizan secuenciadores de estados.

Valor numérico (VN) generado por el software de traducción	Byte en CEN genérico para realizar secuenciadores de estados, al que se debe asignar el valor numérico VN
VN=29 si TF es verificado en alto (D=1) VN=28 si TF es verificado en bajo (D=0)	B4
VN=2IF+BUFF	B5, B36, B69
VN=2^JF	B6, B38, B70
VN= byte alto de DIR1 + 4 *	B8
VN= byte bajo de DIR1 + 4 *	B9
VN= byte alto de DIR1 *	B11, B80, BL (L = 101 + 15(NB-1))
VN= byte bajo de DIR1 *	B12, B81, BJ (J = L + 1)
VN= 2IC+BUFC	B24
VN= 2^JC	B26
VN= 39 si el congelamiento es en bajo (B=0) VN= 38 si el congelamiento es en alto (B=1)	B27
VN= 39 si el RESET es en bajo (C=1) VN= 38 si el RESET es en alto (C=0)	B33
VN=38 si TF es verificado en alto (D=1) VN=39 si TF es verificado en bajo (D=0)	B39
VN= 2IR+BUFR	B30
VN= 2^JR	B32
VN= 2ID+BUFD	B42
VN= 2^JD	B44, B48
VN= 2ID+1+BUFD	B46
VN= 35, disparo por flanco de subida (A=1) VN= 36, disparo por flanco de bajada (A=0)	B50
VN=28 si TF es verificado en alto (D=1) VN=29 si TF es verificado en bajo (D=0)	B68
VN= byte alto de DIR1 + 2 *	B17, B56, B62
VN= byte bajo de DIR1 + 2 *	B18, B57 B63
VN= byte alto de NE - 1	B14
VN= byte bajo de NE - 1	B15
VN= byte alto de DIR2 *	B72
VN= byte bajo de DIR2 *	B73
VN= 2IM + BUFM (M= 0, 1, ...NB-1)	BL (L=90+15M), BP (P=95+15M)
VN= 2^JM	BL (L=91+15M), BP (P=96+15M)

* DIR1 = DIRBM + 104 + 15(NB - 1).
DIR2= DIR1 + 4 + NE.
DIRBM= Dirección base de colocación en memoria del código del secuenciador, esta variable es manejada y definida por el software de traducción (SIIL1.EXE).

Tabla 3.11B Valores de los parámetros BUFD, BUFC, BUFR, BUFM (M=0, 1, 2,...NB-1) y BUFF de la tabla 3.11A

Caracter XD	**BUFD**
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Caracter XC	**BUFC**
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Caracter XR	**BUFR**
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Caracter XF	**BUFF**
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Caracter XM (M=0, 2, ...NB-1)	**BUFM**
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)

Tabla 3.11C Tiempo de ejecución (Te en microsegundos), del código correspondiente a un secuenciador de estados, bajo diferentes condiciones en las entradas al mismo

Condición de funcionamiento	**Tiempo de ejecución (Te), (fe=2Mhz)**
Primera ejecución del código	29.5 + 8.5NB + 1.5NUBAL *
Entrada de congelamiento "C" verificada	11
RESET verificado	42 + 8.5NB + 1.5NUBAL *
Testigo de fin de carrera "TF" verificado	20
Detección de flanco en entrada "D", sin que esto lleve al estado final	53 + 8.5NB + 1.5NUBAL *
Detección de flanco en entrada "D", llevando esto al estado final	55 + 8.5NB + 1.5NUBAL *
En espera de flanco en entrada "D"	28.5

* NUBAL= número de bits en alto en el estado a colocar.

3-5-9 Descripción del ML temporizador monodisparo (one shot) del primer tipo y su CEN asociado.

De acuerdo con la señalización de entrada correspondiente, el PLM puede realizar dos tipos de temporizadores de tipo monodisparo, aquí se describe lo concerniente al temporizador mono disparo de tipo uno, mostrándose respectivamente en las figuras 3.27 y 3.28 la representación como bloque de este ML y el diagrama de tiempos asociado con el mismo.

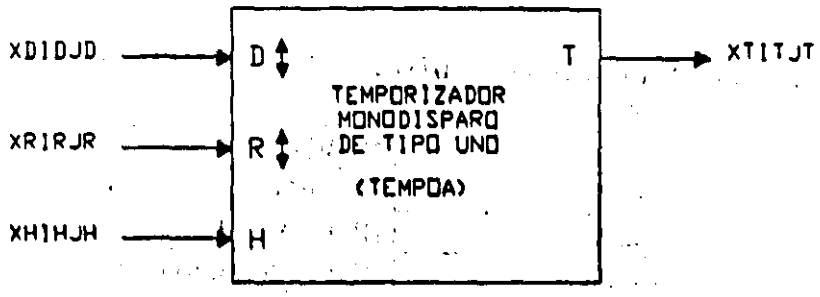
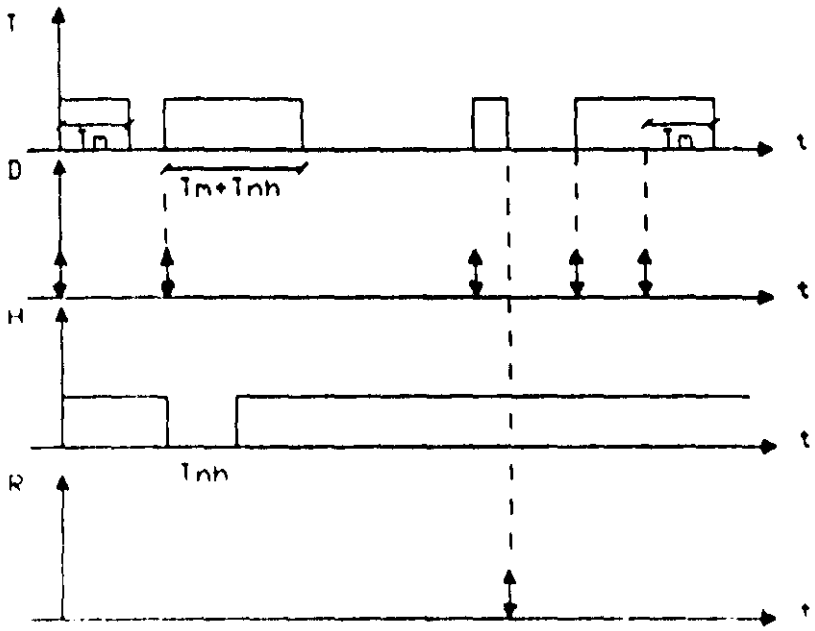


Figura 3.27 Representación genérica de un temporizador monodisparo de tipo uno.

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, como se aprecia en la figura 3.28 tanto el disparo como el



Tm= DURACION DEL PULSO ESPECIFICADA POR EL USUARIO

Figura 3.28 Diagrama de tiempos asociado con un temporizador monodisparo de tipo uno.

restablecimiento (RESET), pueden ser por flanco de subida o bajada, teniéndose además otra entrada denominada como H (habilitación). En la figura 3.28 se aprecia que este temporizador tiene capacidad de redisparo.

Mediante la entrada H se habilita el funcionamiento del temporizador, en caso de que la misma se verifique este ML responde a las otras dos entradas normalmente, si H no se verifica no habrá respuesta a las entradas, si tal verificación ocurre durante el intervalo de verificación de la salida se suspende la cuenta de tiempo asociada, permaneciendo verificada la salida.

La entrada R responde a flancos, que al detectarse desverifican la salida y restablecen a cero el contador de tiempo asociado.

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

TEMPOA#N XDIDJD, XRIRJR, XHIHJH, XTITJT, HH:MM:SS.CS,ABCD;

Donde:

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

XD 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.

ID denota el número de grupo que corresponda a la VB declarada como entrada "D" al temporizador.

JD denota el número de bit dentro del grupo ID, asociado a la variable de entrada "D".

XH podrá ser la letra "e", "s" o "i" mayúscula o minúscula, dependiendo esto de que la variable de entrada de congelamiento (H) al temporizador sea una VBE, VBS o VBI.

IH denota el número de grupo que corresponda a la VB declarada como entrada "H" al temporizador.

JH denota el número de bit dentro del grupo IH, asociado a la variable de entrada "H".

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

IR denota el número de grupo que corresponda a la VB declarada como entrada "R" al temporizador.

JR denota el número de bit dentro del grupo IR, asociado a la variable de entrada "R"

XT 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.

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

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

HH denota un par de dígitos que especifican el número de horas en el tiempo Tm, siendo 47 el valor máximo aceptado.

MM denota un par de dígitos que especifican el número de minutos en 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 se restablezca para flancos de bajada en la entrada de restablecimiento "R", 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 "H" sea alto, en otro caso, "C" deberá ser cero.

D 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 "D" deberá ser uno.

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

Ejemplo 3.9

Supóngase que se desea realizar con el PLM un temporizador monodisparo de tipo uno, de manera que las entradas de disparo, restablecimiento y habilitación sean respectivamente las entradas físicas E00, E01 y E02, requiriéndose que la salida T sea la VBS S02, es necesario que el pulso de salida sea verificado en bajo y tenga una duración de dos minutos con treinta segundos, tanto el disparo como el restablecimiento deben ser por flanco de bajada y la señal de habilitación (H) debe ser verificada en alto, la declaración sintáctica correspondiente podría ser, suponiendo que se le asigna a este temporizador el número uno

TEMPOA#1 E00, E01, E02, S02, 00 02 30 00, 0010,

En la figura 3.29 se muestra una representación como bloque del temporizador de este ejemplo.

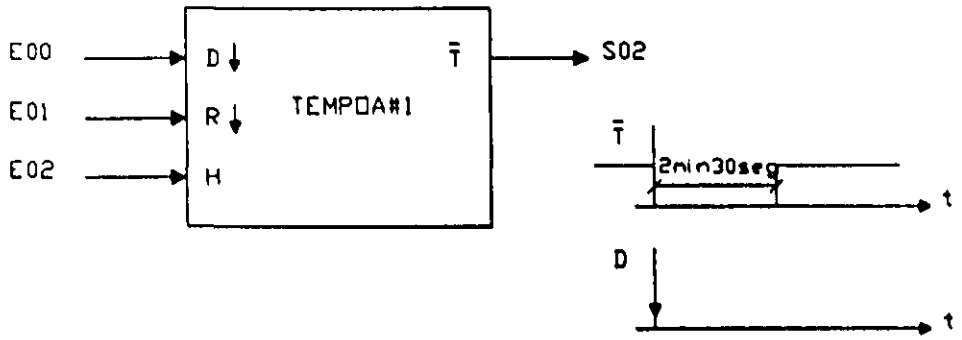


Figura 3.29 Representación como bloque del temporizador monodisparo de tipo uno del ejemplo 3.9

Descripción del CEN asociado con el temporizador monodisparo de tipo uno

El flujo de ejecución de este ML se muestra en la figura 3.30, el tiempo máximo de ejecución de este módulo es de 59 μs (fc=2 Mhz) y el mismo se presenta cuando la entrada "H" se verifica, no habiendo flancos en las entradas "D" y "R", estando la salida verificada habiéndose llegado a cero en el contador descendente asociado con el temporizador.

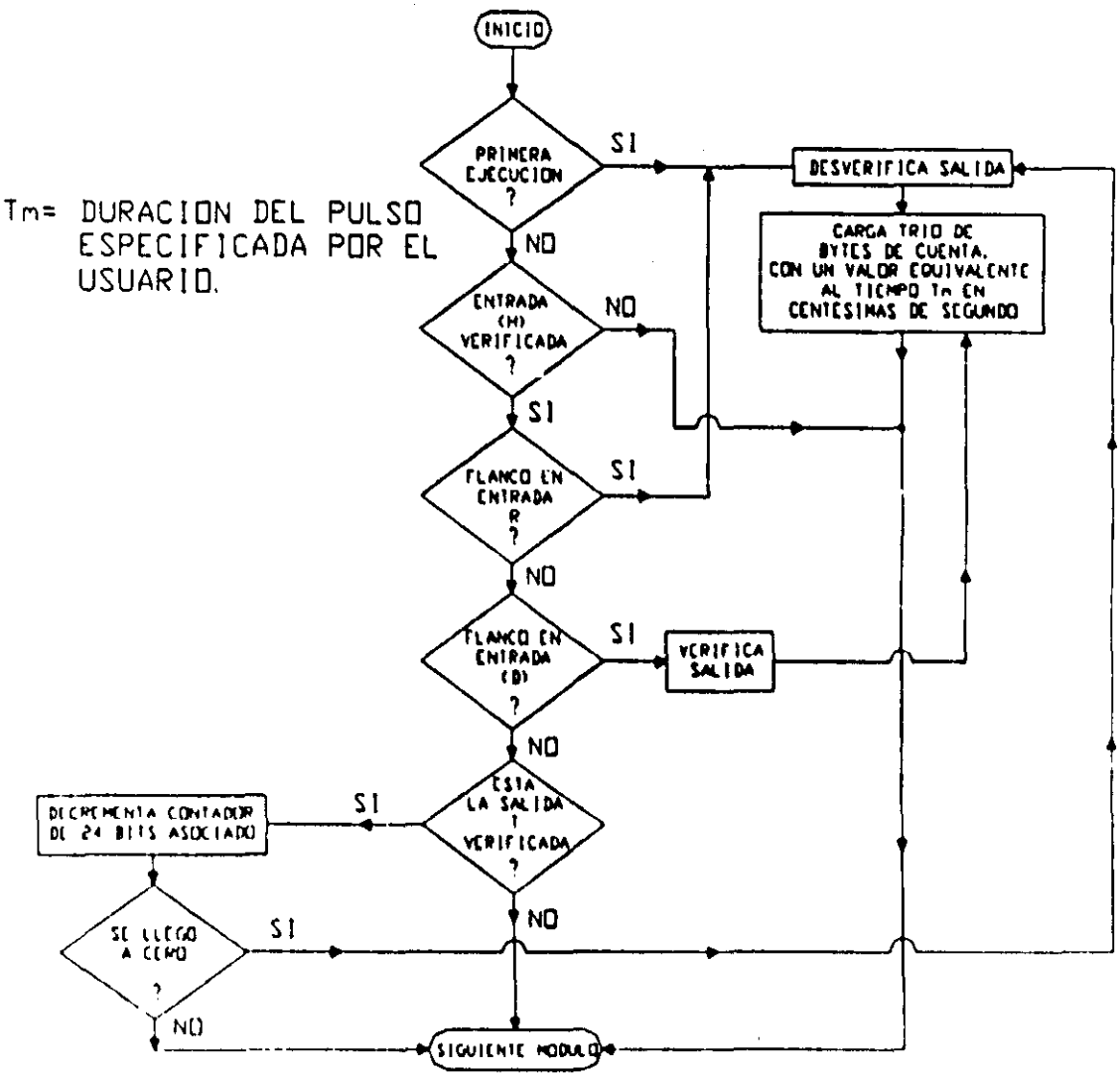


Figura 3.30 Flujo de ejecución del temporizador monodisparo de tipo uno

A continuación se muestra el CEN asociado con este módulo

CEN DEL TEMPORIZADOR MONODISPARO DE TIPO UNO (TEMPOA)

```
0000 A6F1                          LDAA $F1,X
0002 2712                          BEQ NOPP
                INICIALIZACIÓN
0004 1D0001          RESET:        BCLR $00,X,01;BSET PARA PULSO DE SALIDA
                                         VERIFICADO EN BAJO
0007 8620            INCON:        LDAA #$20;BYTE ALTO DE CONTADOR
0009 B70200                        STAA $0200;CARGA BYTE ALTO DE CONTADOR
000C 18CE2122                      LDY #$2122;
0010 18FF0201                      STY $0201;CARGA BYTES INTERMEDIO Y BAJO
                                         DE CONTADOR
0014 203A                          BRA SALIDA
          CHEQUEO DE NIVEL DE ENTRADA DE HABILITACION
0016 A600            NOPP:         LDAA $00,X;CARGA VALOR PRESENTE DE GRUPO
```

```
                                        DE ENTRADA DE HABILITACIÓN
0018 8401                       ANDA #$01;ENMASCARA BIT DE ENTRADA DE
                                        HABILITACIÓN
001A 2734                       BEQ SALIDA;BNE PARA VERIFICACION DE
                                        HABILITACIÓN EN BAJO
              CHEQUEO DE FLANCO EN ENTRADA DE RESET
001C A600                       LDAA $00,X;CARGA VALOR PRESENTE DE GRUPO
                                        ENTRADA DE RESET
001E 8401                       ANDA #$01;ENMASCARA BIT PRESENTE DE
                                        ENTRADA DE RESET
0020 E601                       LDAB $01,X;CARGA VALOR ANTERIOR DE GRUPO
                                        DE ENTRADA
0022 C401                       ANDB #$01;ENMASCARA BIT DE ENTRADA
                                        ANTERIOR DE RESET
0024 11                         CBA
0025 2702                       BEQ SIGUE2
0027 2EDB                       BGT RESET;BLT PARA RESET POR FLANCO DE
                                        BAJADA
          CHEQUEO DE FLANCO EN ENTRADA DE DISPARO
0029 A600           SIGUE2:     LDAA $00,X;CARGA VALOR PRESENTE DE GRUPO
                                        DE ENTRADA DE DISPARO
002B 8401                       ANDA #$01;ENMASCARA BIT PRESENTE DE
                                        DISPARO
002D E601                       LDAB $01,X;CARGA VALOR ANTERIOR DE GRUPO
                                        GRUPO DE ENTRADA DE DISPARO
002F C401                       ANDB #$01;ENMASCARA BIT ANTERIOR DE
                                        DISPARO
0031 11                         CBA
0032 2702                       BEQ SIGUE3
0034 2E1C                       BGT VERIFSAL;BLT PARA DISPARO POR FLANCO
                                        DE BAJADA
0036 A600           SIGUE3:     LDAA $00,X;
0038 8401                       ANDA #$01;ENMASCARA BIT DE SALIDA
003A 2714                       BEQ SALIDA;BNE PULSO DE SALIDA
                                        VERIFICADO EN BAJO
003C 18FE0201                   LDY $0201
0040 1809                       DEY
0042 18FF0201                   STY $0201
0046 2608                       BNE SALIDA
0048 B60200                     LDAA $0200
004B 27B7                       BEQ RESET
004D 7A0200                     DEC $0200
0050 2005           SALIDA:     BRA SIGBLO
0052 1C0001         VERIFSAL:   BSET $00,X,01;BCLR PARA PULSO DE SALIDA
                                        VERIFICADO EN BAJO
0055 20B0                       BRA INCON
0057 01             SIGBLO:     NOP
```

En la tabla 3.12A se muestra la TAB asociada con el CEN TEMPOA y en la tabla 3.12B se aprecia el valor que deben tomar, de acuerdo con el tipo de VB que corresponda a las entradas y salida de este temporizador, parámetros que aparecen en la tabla 3.12A.

Tabla 3.12A Asignación de bytes asociada con el CEN TEMPOA, empleado para obtener el código requerido por los ML que realizan temporizadores monodisparo de tipo uno.

Valor numérico (VN) generado por el software de traducción	Byte en CEN genérico para realizar temporizadores monodisparo de tipo uno, al que se debe asignar el valor numérico VN
VN=29 si T es verificado en alto (D=1) VN=28 si T es verificado en bajo (D=0)	B4
VN=2IT+BUFT	B5
VN=2^JT	B6
VN= BAC*	B8
VN= byte alto de DIRCON**	B10
VN= byte bajo de DIRCON**	B11
VN= BIC***	B14
VN= BBC	B15
VN= Byte alto de DIRCON + 1	B18
VN= Byte bajo de DIRCON + 1	B19
VN= 2IH + BUFH	B23
VN= 2^JH	B25
VN= 39 si H se verifica en alto (C=1) VN= 38 si H se verifica en bajo (C=0)	B26
VN= 2IR + BUFR	B29
VN= 2^JR	B31
VN= 2IR + 1 +BUFR	B33
VN= 2^JR	B35
VN= 46 si RESET por flanco de subida, B=1 VN= 45 si RESET por flanco de bajada, B=0	B39
VN= 2ID + BUFD	B42
VN= 2^JD	B44
VN= 2ID + BUFD + 1	B46
VN= 2^JD	B48
VN= 46 si disparo por flanco de subida A=1 VN= 45 si disparo por flanco de bajada A=0	B52
VN= 2IT + BUFT	B55
VN= 2^JT	B57
VN= 39 si salida verificada en alto (D=1) VN= 38 si salida verificada en bajo (D=0)	B58
VN = Byte alto de DIRCON + 1	B62
VN = Byte bajo de DIRCON + 1	B63
VN = Byte alto de DIRCON + 1	B68
VN = Byte bajo de DIRCON + 1	B69
VN = Byte alto de DIRCON	B73

Continuación Tabla 3.12A en la siguiente página.

VN = Byte bajo de DIRCON	B74
VN = Byte alto de DIRCON	B78
VN = Byte bajo de DIRCON	B79
VN = 28 si salida verificada en alto (D=1) VN = 29 si salida verificada en bajo (D=0)	B82
VN = 2IT + BUFT	B83
VN = 2^JT	B84

* BAC = Byte alto de cuenta asociada (CA)

CA = valor en centésimas de segundo del tiempo de duración del pulso de salida.

** DIRCON= DIRBCT + 3N (DIRBCT= dirección base de contadores de temporizadores). N=número de temporizador asignado por el usuario.

*** BIC = Byte intermedio de cuenta asociada (CA)

**** BBC = Byte bajo de cuenta asociada (CA)

Tabla 3.12B Valores de los parámetros BUFD, BUFR, BUFH, y BUFT de la tabla 3.12A

Caracter XD	**BUFD**
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Caracter XR	**BUFR**
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Caracter XH	**BUFH**
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Caracter XT	**BUFT**
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)

3-5-10 Descripción del ML temporizador monodisparo (one shot) del segundo tipo y su CEN asociado.

De acuerdo con la señalización de entrada correspondiente, el PLM puede realizar dos tipos de temporizadores de tipo monodisparo, aqui se describe lo concerniente al temporizador mono disparo de tipo dos, mostrándose respectivamente en las figuras 3 31 y 3 32 la representación como bloque de este ML y el diagrama de tiempos asociado con el mismo.

Figura 3.31 Representación genérica de un temporizador monodisparo de tipo dos.

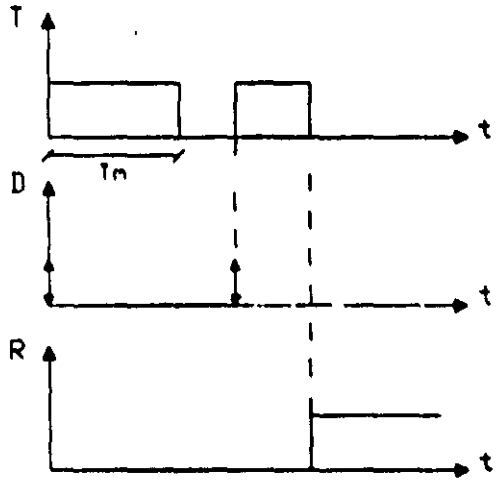


Figura 3.32 Diagrama de tiempos asociado con un temporizador monodisparo de tipo dos, (RESET verificado en alto).

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, como se aprecia en la figura 3.32 el disparo puede ser por flanco de subida o bajada, teniéndose además otra entrada denominada "R" (RESET), que al verificarse coloca a este módulo en su condición de espera de disparo, con su salida no verificada. Al igual que el temporizador monodisparo de tipo uno, este temporizador tiene capacidad de redisparo.

La entrada R responde 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#N XDIDJD, XRIRJR, XTITJT, HH:MM:SS.CS,ABC;

Donde:

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

XD 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.

ID denota el número de grupo que corresponda a la VB declarada como entrada "D" al temporizador.

JD denota el número de bit dentro del grupo ID, asociado a la variable de entrada "D".

XR 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.

IR denota el número de grupo que corresponda a la VB declarada como entrada "R" al temporizador.

JR denota el número de bit dentro del grupo IR, asociado a la variable de entrada "R".

XT 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.

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

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

HH denota un par de dígitos que especifican el número de horas en el tiempo Tm, siendo 47 el valor máximo aceptado.

MM denota un par de dígitos que especifican el número de minutos en 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 un programa fuente en SIIL1.

Ejemplo 3.10

Supóngase que se desea realizar con el PLM un temporizador monodisparo de tipo dos, de manera que las entradas de disparo y restablecimiento sean respectivamente las entradas fisicas 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 3.33 se muestra una representación como bloque del temporizador de este ejemplo.

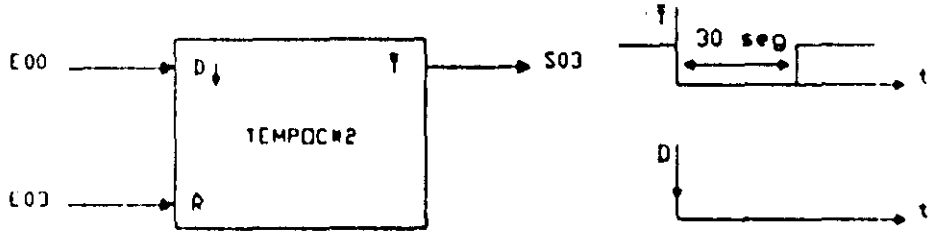


Figura 3.33 Representación como bloque del temporizador monodisparo del ejemplo 3.10

Descripción del CEN asociado con el temporizador monodisparo de tipo dos

El flujo de ejecución de este ML se muestra en la figura 3.34, el tiempo máximo de ejecución de este módulo es de 51 μs (fe=2 Mhz) y el mismo se presenta cuando no habiendo flanco en la entrada "D", estando la entrada de "R" no verificada y la salida "T" verificada se ha llegado al final de la cuenta regresiva asociada.

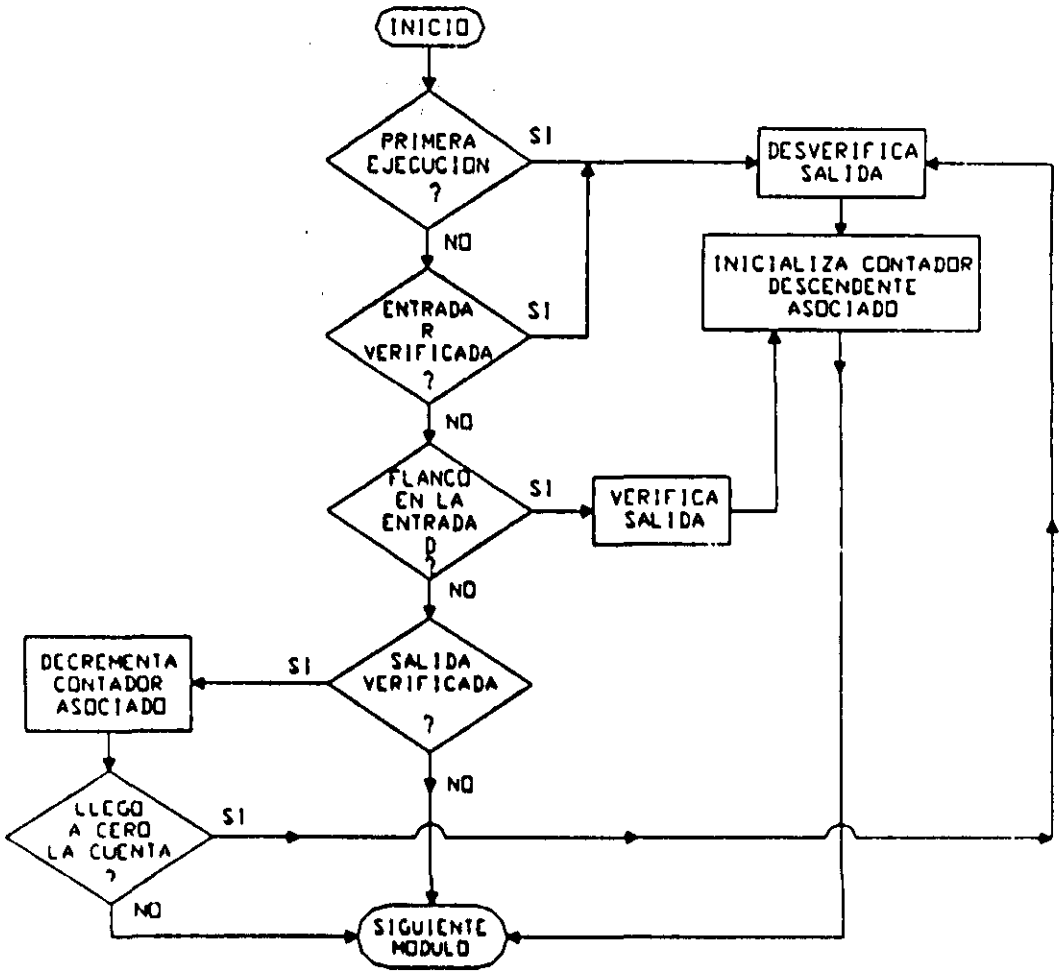


Figura 3.34 Flujo de ejecución del temporizador monodisparo de tipo dos (TEMPOC).

A continuación se muestra el CEN asociado con este módulo

CEN DEL TEMPORIZADOR MONODISPARO DE TIPO DOS (TEMPOC)

```
0000 A6F1                      LDAA $F1,X
0002 2712                      BEQ NOPP
   ACCIONAMIENTO AL ENTRAR A ESTE CÓDIGO POR PRIMERA VEZ
0004 1D0001           RESET:   BCLR $00,X,01;BSET PARA PULSO DE SALIDA
                                      VERIFICADO EN BAJO.
0007 8620             INCON:   LDAA #$20
0009 B70200                    STAA $0200;CARGA BYTE ALTO DE CONTADOR
000C 18CE2122                  LDY #$2122;
0010 18FF0201                  STY $0201;CARGA BYTES INTERMEDIO Y BAJO DE
                                      CONTADOR
0014 2020                      BRA SALIDA
   CHEQUEO DE NIVEL DE ENTRADA DE RESTABLECIMIENTO (RESET)
0016 A600             NOPP:    LDAA $00,X;CARGA VALOR PRESENTE DE GRUPO
                                      ENTRADA DE RESET
0018 8401                      ANDA #$01;ENMASCARA BIT DE ENTRADA DE
                                      RESET
001A 27E8                      BEQ RESET;BNE PARA VERIFICACION DE RESET
                                      EN NIVEL ALTO
        CHEQUEO DE FLANCO EN ENTRADA DE DISPARO
```

```
001C A600                    LDAA $00,X;CARGA VALOR PRESENTE DE GRUPO DE
                                       ENTRADA DE DISPARO
001E 8401                    ANDA #$01;ENMASCARA BIT PRESENTE DE DISPARO
0020 E601                    LDAB $01,X;CARGA VALOR ANTERIOR DE GRUPO DE
                                       ENTRADA DE DISPARO
0022 C401                    ANDB #$01;ENMASCARA BIT ANTERIOR DE DISPARO
0024 11                      CBA
0025 2702                    BEQ SIGUE3
0027 221C                    BHI VERIFSAL;BLO PARA DISPARO POR FLANCO DE
                                          BAJADA
0029 A600          SIGUE3:   LDAA $00,X;
002B 8401                    ANDA #$01;ENMASCARA BIT DE SALIDA
002D 2714                    BEQ SALIDA;BNE PULSO DE SALIDA VERIFICADO
                                       EN BAJO
002F 18FE0201                LDY $0201
0033 1809                    DEY
0035 18FF0201                STY $0201
0039 2608                    BNE SALIDA
003B B60200                  LDAA $0200
003E 27C4                    BEQ RESET
0040 7A0200                  DEC $0200
0043 2005          SALIDA:   BRA SIGBLO
0045 1C0001        VERIFSAL: BSET $00,X,01;BCLR PARA PULSO DE SALIDA
                                           VERIFICADO EN BAJO
0048 20BD                    BRA INCON
004A 01            SIGBLO:   NOP
```

En la tabla 3.13A se muestra la TAB asociada con el CEN TEMPOC y en la tabla 3.13B se aprecia el valor que deben tomar, de acuerdo con el tipo de VB que corresponda a las entradas y salida de este temporizador, parametros que aparecen en la tabla 3.13A.

Tabla 3.13A Asignación de bytes asociada con el CEN TEMPOC, empleado para obtener el código requerido por los ML que realizan temporizadores monodisparo de tipo dos.

Valor numérico (VN) generado por el software de traducción	**Byte en CEN genérico para realizar temporizadores monodisparo de tipo dos, al que se debe asignar el valor numérico VN**
VN=29 si T es verificado en alto (C=1) VN=28 si T es verificado en bajo (C=0)	B4
VN=2IT+BUFT	B5
VN=2^JT	B6
VN= BAC*	B8
VN= byte alto de DIRCON**	B10
VN= byte bajo de DIRCON**	B11
VN= BIC***	B14
VN= BBC	B15
VN= Byte alto de DIRCON + 1	B18
VN= Byte bajo de DIRCON + 1	B19
VN= 2IR + BUFR	B23

```
001C A600                    LDAA $00,X;CARGA VALOR PRESENTE DE GRUPO DE
                                       ENTRADA DE DISPARO
001E 8401                    ANDA #$01;ENMASCARA BIT PRESENTE DE DISPARO
0020 E601                    LDAB $01,X;CARGA VALOR ANTERIOR DE GRUPO DE
                                       ENTRADA DE DISPARO
0022 C401                    ANDB #$01;ENMASCARA BIT ANTERIOR DE D'3PARO
0024 11                      CBA
0025 2702                    BEQ SIGUE3
0027 221C                    BHI VERIFSAL;BLO PARA DISPARO POR FLANCO DE
                                          BAJADA
0029 A600          SIGUE3:   LDAA $00,X;
002B 8401                    ANDA #$01;ENMASCARA BIT DE SALIDA
002D 2714                    BEQ SALIDA;BNE PULSO DE SALIDA VERIFICADO
                                       EN BAJO
002F 18FE0201                LDY $0201
0033 1809                    DEY
0035 18FF0201                STY $0201
0039 2608                    BNE SALIDA
003B B60200                  LDAA $0200
003E 27C4                    BEQ RESET
0040 7A0200                  DEC $0200
0043 2005          SALIDA:   BRA SIGBLO
0045 1C0001        VERIFSAL: BSET $00,X,01;BCLR PARA PULSO DE SALIDA
                                           VERIFICADO EN BAJO
0048 20BD                    BRA INCON
004A 01            SIGBLO:   NOP
```

En la tabla 3.13A se muestra la TAB asociada con el CEN TEMPOC y en la tabla 3.13B se aprecia el valor que deben tomar, de acuerdo con el tipo de VB que corresponda a las entradas y salida de este temporizador, parámetros que aparecen en la tabla 3.13A.

Tabla 3.13A Asignación de bytes asociada con el CEN TEMPOC, empleado para obtener el código requerido por los ML que realizan temporizadores monodisparo de tipo dos.

Valor numérico (VN) generado por el software de traducción	Byte en CEN genérico para realizar temporizadores monodisparo de tipo dos, al que se debe asignar el valor numérico VN
VN=29 si T es verificado en alto (C=1) VN=28 si T es verificado en bajo (C=0)	B4
VN=2IT+BUFT	B5
VN=2^JT	B6
VN= BAC*	B8
VN= byte alto de DIRCON**	B10
VN= byte bajo de DIRCON**	B11
VN= BIC***	B14
VN= BBC	B15
VN= Byte alto de DIRCON + 1	B18
VN= Byte bajo de DIRCON + 1	B19
VN= 2IR + BUFR	B23

ContinuaciónTabla 3.13A	
VN= 2^JR	B25
VN= 39 si R se verifica en bajo (B=1) VN= 38 si R se verifica en alto (B=0)	B26
VN= 2ID + BUFD	B29
VN= 2^JD	B31
VN= 2ID + 1 +BUFD	B33
VN= 2^JD	B35
VN= 34 si disparo por flanco de subida, A=1 VN= 37 si disparo por flanco de bajada, A=0	B39
VN= 2IT + BUFT	B42
VN= 2^JT	B44
VN= 39 si salida verificada en alto (C=1) VN= 38 si salida verificada en bajo (C=0)	B45
VN = Byte alto de DIRCON + 1	B49
VN = Byte bajo de DIRCON + 1	B50
VN = Byte alto de DIRCON + 1	B55
VN = Byte bajo de DIRCON + 1	B56
VN = Byte alto de DIRCON	B60
VN = Byte bajo de DIRCON	B61
VN = Byte alto de DIRCON	B65
VN = Byte bajo de DIRCON	B66
VN = 28 si salida verificada en alto (C=1) VN = 29 si salida verificada en bajo (C=0)	B69
VN = 2IT + BUFT	B70
VN = 2^JT	B71

* BAC = Byte alto de cuenta asociada (CA)

CA = valor en centésimas de segundo del tiempo de duración del pulso de salida.

** DIRCON= DIRBCT + 3N (DIRBCT= dirección base de contadores de temporizadores)
N=número de temporizador asignado por el usuario.

*** BIC = Byte intermedio de cuenta asociada (CA)

**** BBC = Byte bajo de cuenta asociada (CA)

Tabla 3.13B Valores de los parámetros BUFD, BUFR, BUFH, y BUFT de la tabla 3.13A

Caráeter XD	BUFD
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Caracter XR	**BUFR**
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Caracter XT	**BUFT**

Continuación Tabla 3.13B.

Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)

3-5-11 Descripción del ML que realiza temporizadores con retardo a la activación (on-delay) o con retardo a la desactivación (off-delay).

El PLM puede realizar temporizadores con retardo a la activación (RA) o a la desactivación (RD), esto se logra a partir de un solo ML. En las figuras 3.35 y 3.36 se muestran respectivamente la representación como bloque de este módulo y los diagramas de tiempo asociados.

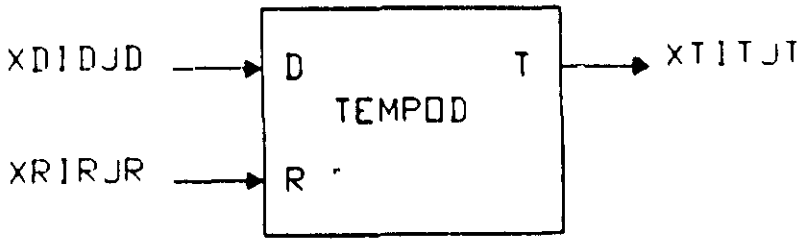


Figura 3.35 Representación genérica de un temporizador que puede operar con retardo n la activación o desactivación.

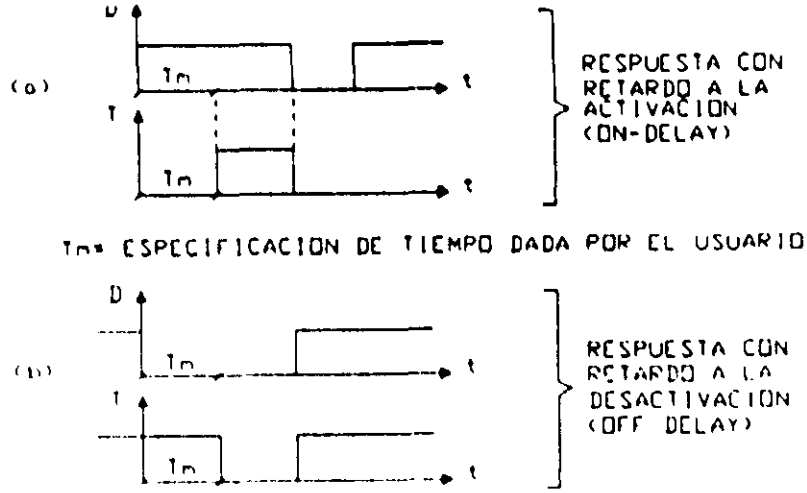


Figura 3.36 Diagrama de tiempos asociado con un temporizador con capacidad de retardo a la activación (a) o a la desactivación (b). Al verificarse la entrada de restablecimiento (R) la salida pasa a su nivel no verificado (cero para on-delay, uno para off-delay) inicializándose el contador descendente 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#N XDIDJD, XRIRJR, XTITJT, HH:MM:SS.CS,AB;

Donde:

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

XD 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.

ID denota el número de grupo que corresponda a la VB declarada como entrada "D" al temporizador.

JD denota el número de bit dentro del grupo ID, asociado a la variable de entrada "D".

XR 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.

IR denota el número de grupo que corresponda a la VB declarada como entrada "R" al temporizador.

JR denota el número de bit dentro del grupo IR, asociado a la variable de entrada "R".

XT 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.

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

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

HH denota un par de digitos que especifican el número de horas en el tiempo Tm, siendo 47 el valor máximo aceptado.

MM denota un par de digitos que especifican el número de minutos en Tm.

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

CS denota un par de digitos que especifican las centésimas de segundo en 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 3.11

Supóngase que se desea realizar con el PLM 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, siendo necesario que 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 de este ejemplo, 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.

Una declaración para estos temporizadores podría ser la siguiente:

```
TEMPOD#3 E02, E03, S04, 00:00:07.00, 11;
TEMPOD#4 E04, E05, S05, 00:00:10.00, 01;
```

En la figura 3.37 se muestran las representaciones como bloques de los temporizadores de este ejemplo y sus diagramas de tiempo asociados.

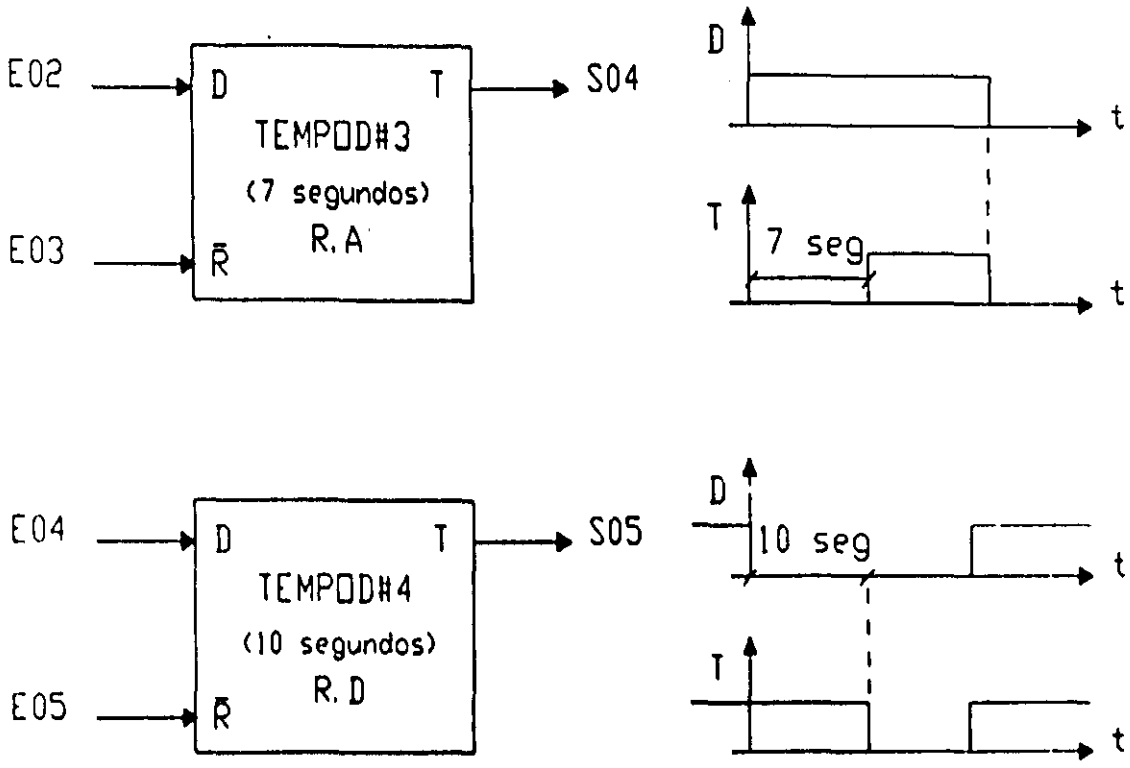


Figura 3.37 Diagramas de tiempo y representaciones como bloques, de los dos temporizadores del ejemplo 3.11

Descripción del CEN asociado para temporizadores con retardo a la activación o desactivación

El flujo de ejecución de este ML se muestra en la figura 3.38, el tiempo máximo de ejecución de este módulo es de 34.5 μs (fe=2 Mhz).

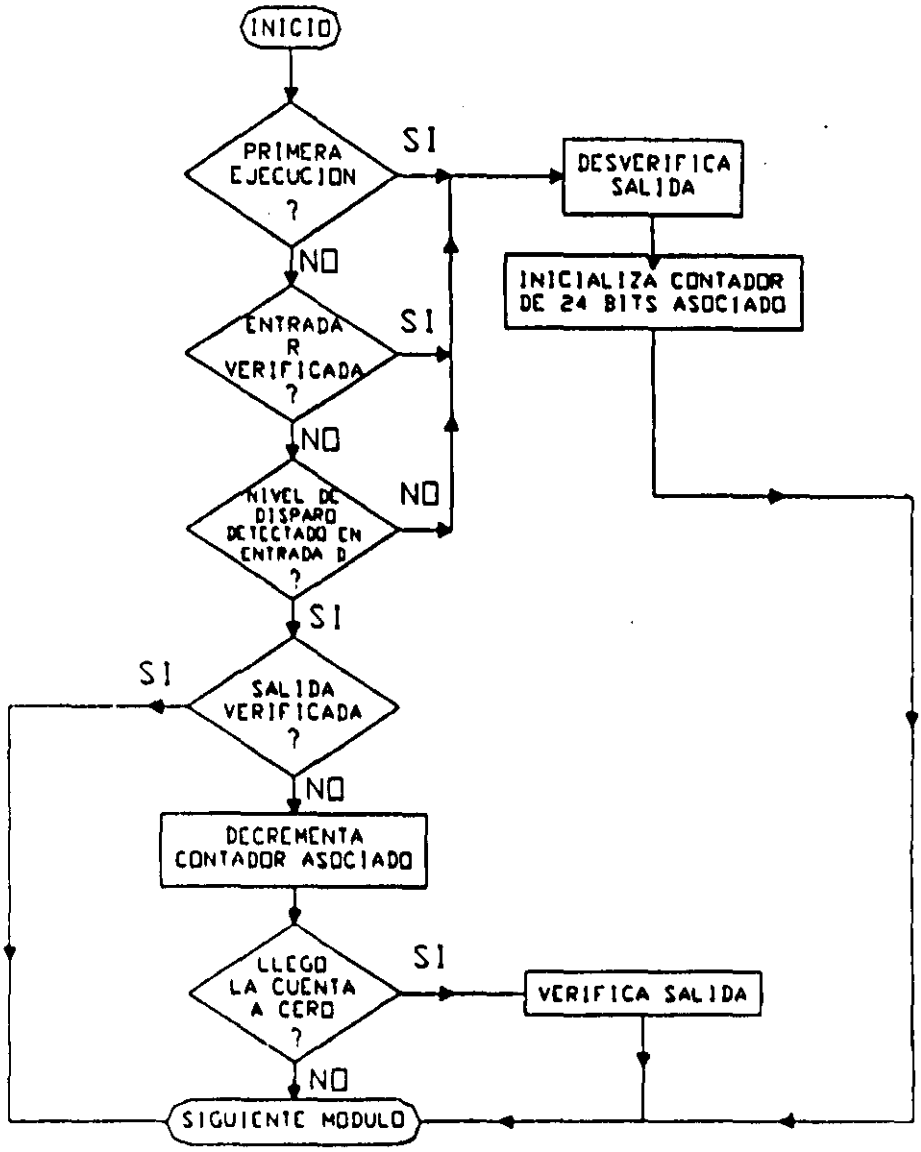


Figura 3.38 Flujo de ejecución del temporizador con capacidad de retardo a la activación o desactivación (TEMPOD).

A continuación se muestra el CEN asociado con este módulo.

CEN DEL TEMPORIZADOR CON RETARDO A LA ACTIVACIÓN O DESACTIVACIÓN (TEMPOD)

```
0000 A6F1                       LDAA $F1,X
0002 2712                       BEQ NOPP
  ACCIONAMIENTO AL ENTRAR A ESTE CÓDIGO POR PRIMERA VEZ
0004 1D0001           RESET:    BCLR $00,X,01;BSET PARA OFF DELAY
0007 8620             INCON:    LDAA #$20;BYTE ALTO DE CONTADOR
0009 B70200                     STAA $0200;CARGA BYTE ALTO DE CONTADOR
000C 18CE2122                   LDY #$2122;
```

```
0010 18FF0201                    STY $0201;CARGA BYTES INTERMEDIO Y BAJO
                                          DE CONTADOR
0014 202B                        BRA SALIDA
  CHEQUEO DE NIVEL DE ENTRADA DE RESTABLECIMIENTO (RESET)
0016 A600          NOPP:         LDAA $00,X;CARGA VALOR PRESENTE DE GRUPO
                                          DE ENTRADA DE RESET
0018 8401                        ANDA #$01;ENMASCARA BIT DE ENTRADA DE
                                          RESET
001A 27E8                        BEQ RESET;BNE PARA RESET EN NIVEL
                                          ALTO
        CHEQUEO DE NIVEL DE ENTRADA DE DISPARO
001C A600                        LDAA $00,X;CARGA VALOR PRESENTE DE GRUPO
                                          DE ENTRADA DE DISPARO
001E 8401                        ANDA #$01;ENMASCARA BIT PRESENTE DE
                                          DISPARO
0020 27E2                        BEQ RESET;BNE OFF DELAY
0022 A600          SIGUE3:       LDAA $00,X;
0024 8401                        ANDA #$01;ENMASCARA BIT DE SALIDA
0026 2619                        BNE SALIDA;BEQ RETARDO A LA
                                          DESACTIVACIÓN (OFF-DELAY)
0028 18FE0201                    LDY $0201
002C 1809                        DEY
002E 18FF0201                    STY $0201
0032 260D                        BNE SALIDA
0034 B60200                      LDAA $0200
0037 2705                        BEQ VERIFSAL
0039 7A0200                      DEC $0200
003C 2003                        BRA SALIDA
003E 1C0001        VERIFSAL:     BSET $00,X,01;BCLR PARA OFF DELAY
0041 01            SALIDA:       NOP
```

En la tabla 3.14A se muestra la TAB asociada con el CEN TEMPOD y en la tabla 3.14B se aprecia el valor que deben tomar, de acuerdo con el tipo de VB que corresponda a las entradas y salida de este temporizador, parámetros que aparecen en la tabla 3.14A.

Tabla 3.14A Asignación de bytes asociada con el CEN TEMPOD, empleado para obtener el código requerido por los ML que realizan temporizadores con retardo a la activación (on-delay) o desactivación (off-delay).

Valor numérico (VN) generado por el software de traducción	Byte en CEN genérico para realizar temporizadores con retardo a la activación o desactivación al que se le asignará el valor VN
VN=29, retardo a la activación (A=1) VN=28, retardo a la desactivación (A=0)	B4
VN=2IT+BUFT	B5
VN=2^JT	B6
VN= BAC*	B8
VN= byte alto de DIRCON**	B10
VN= byte bajo de DIRCON**	B11
VN= BIC***	B14
VN= BBC	B15

Continuación Tabla 3.14A.

VN= Byte alto de DIRCON + 1	B18
VN= Byte bajo de DIRCON + 1	B19
VN= 2IR + BUFR	B23
VN= 2^JR	B25
VN= 39 si R se verifica en bajo (B=1) VN= 38 si R se verifica en alto (B=0)	B26
VN= 2ID + BUFD	B29
VN= 2^JD	B31
VN= 39, retardo a la activación (A=1) VN= 38, retardo a la desactivación (A=0)	B32
VN= 2IT + BUFT	B35
VN= 2^JT	B37
VN= 38, retardo a la activación (A=1) VN= 39, retardo a la desactivación (A=0)	B38
VN = Byte alto de DIRCON + 1	B42
VN = Byte bajo de DIRCON + 1	B43
VN = Byte alto de DIRCON + 1	B48
VN = Byte bajo de DIRCON + 1	B49
VN = Byte alto de DIRCON	B53
VN = Byte bajo de DIRCON	B54
VN = Byte alto de DIRCON	B58
VN = Byte bajo de DIRCON	B59
VN= 28, retardo a la activación (A=1) VN= 29, retardo a la desactivación (A=0)	B62
VN = 2IT + BUFT	B63
VN = 2^JT	B64

* BAC = Byte alto de cuenta asociada (CA)

CA = valor en centésimas de segundo del intervalo de tiempo implicado (Tm).

** DIRCON= DIRBCT + 3N (DIRBCT= dirección base de contadores de temporizadores).
N=número de temporizador asignado por el usuario.

*** BIC = Byte intermedio de cuenta asociada (CA)

**** BBC = Byte bajo de cuenta asociada (CA)

Tabla 3.14B Valores de los parámetros BUFD, BUFR y BUFT de la tabla 3.14A

Caracter XD	**BUFD**
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Caracter XR	**BUFR**
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)

Continuación Tabla 3.14B.

Caracter XT	BUFT
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)

3-5-12 Descripción del ML que realiza temporizadores astables.

El PLM puede realizar temporizadores astables que generan señales cuadradas con ciclo de trabajo que puede ser fijado por el usuario, en las figuras 3.39 y 3.40 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 3.39 Representación genérica de un temporizador astable realizable en el PLM.

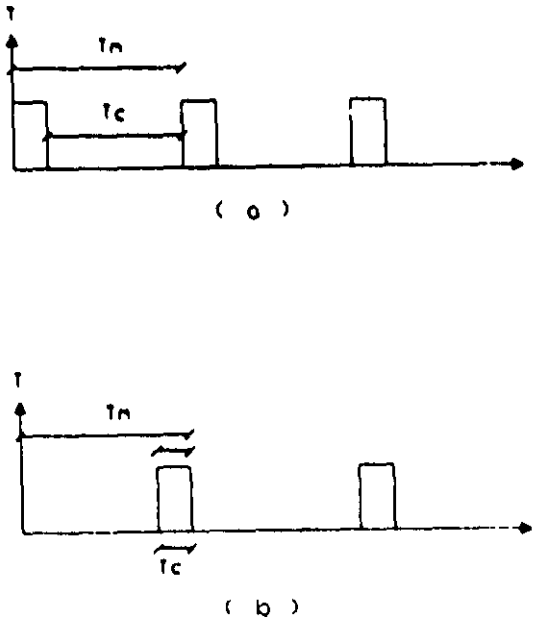


Figura 3.40 Diagramas de tiempos asociados con un temporizador astable con arranque en uno (a) y con arranque en cero (b).

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, y al verificarse se coloca en la salida el nivel de arranque y se inicializa el contador asociado.

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

TEMPOE#N XRIRJR, XTITJT, HHm:MMm:SSm.CSm, HHc:MMc:SSc.CSc, AB;

Donde:

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

XR 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.

IR denota el número de grupo que corresponda a la VB declarada como entrada "R" al temporizador.

JR denota el número de bit dentro del grupo IR, asociado a la variable de entrada "R".

XT 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.

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

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

HHm denota un par de dígitos que especifican el número de horas en el tiempo Tm, siendo 47 el valor máximo aceptado.

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

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

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

HHc denota un par de dígitos que especifican el número de horas en el tiempo Tc, siendo 47 el valor máximo aceptado.

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

SSc denota un par de dígitos que especifican los segundos en Tc.

CSc denota un par de dígitos que especifican las centésimas de segundo en 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=1), el temporizador se restablecerá 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 el dígito "B" deberá ser uno.

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

Ejemplo 3.12

Supóngase que se desea realizar con el PLM 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 dos segundos y 250 ms, 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.

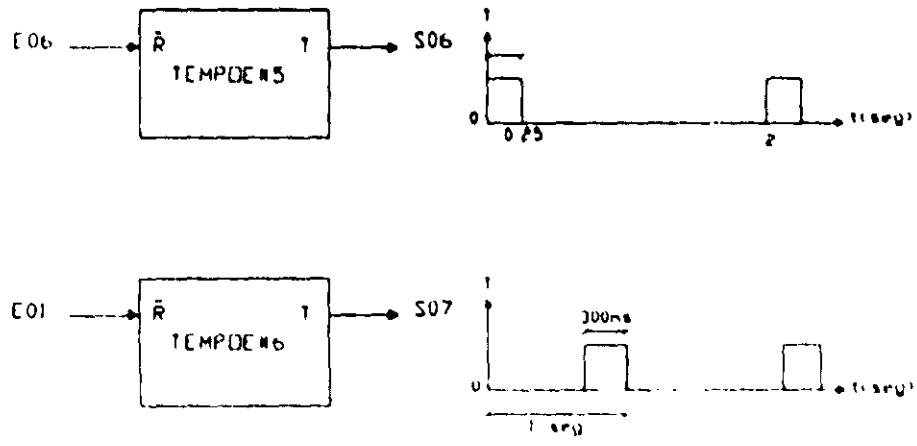
Para el segundo temporizador de este ejemplo, se requiere que los tiempos Tm y tc sean respectivamente un segundo y 300 ms, el nivel de restablecimiento ha de ser bajo siendo E07 la VB asociada, la salida debe ser la vb S07, asignándosele a este temporizador el número seis.

Una declaración para estos temporizadores podría ser la siguiente:

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

TEMPOE#6 E07, S07, 00:00:01.00, 00:00:00.30, 10;

En la figura 3.41 se muestran las representaciones como bloques de los temporizadores de este ejemplo y sus diagramas de tiempo asociados.



dos Figura 3.41 Diagramas de tiempo y representaciones como bloques, de los temporizadores del ejemplo 3.12

Descripción del CEN asociado para temporizadores astables

El flujo de ejecución de este ML se muestra en la figura 3.42, el tiempo máximo de ejecución de este módulo es de 51 µs (fe=2 Mhz).

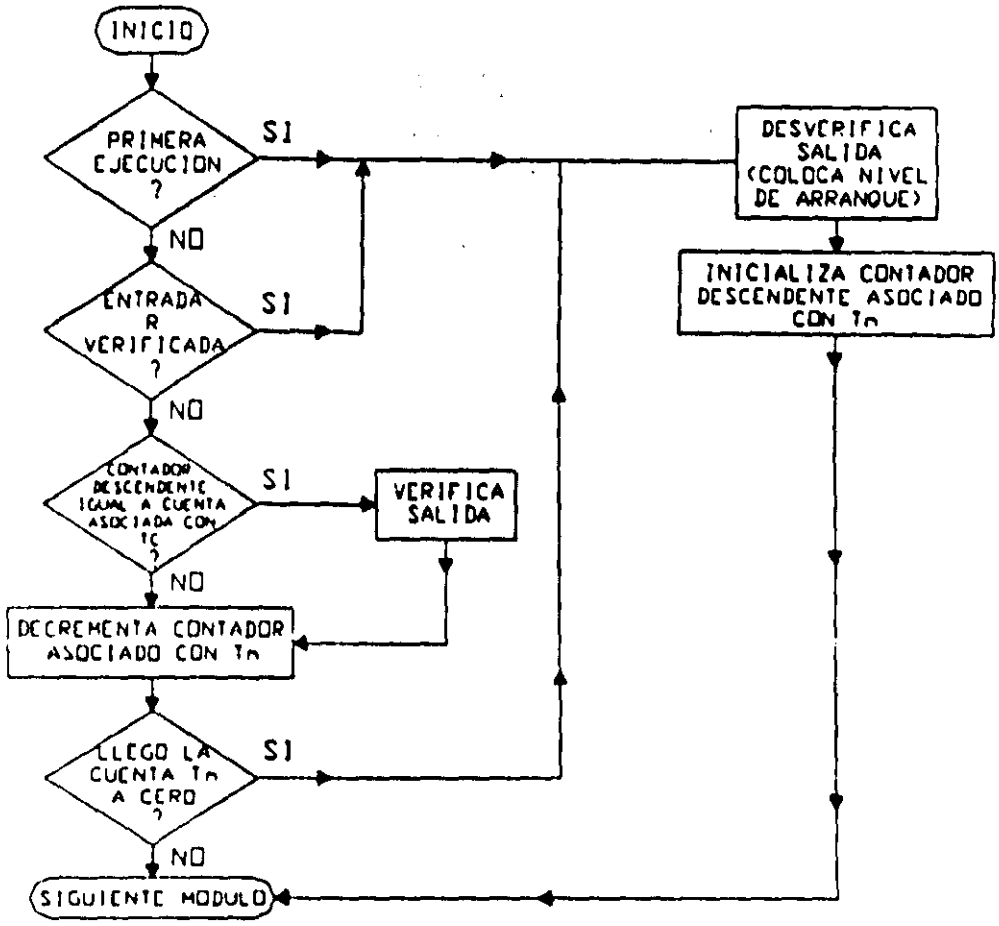


Figura 3.42 Flujo de ejecución del temporizador astable realizable por el PLM (TEMPOE).

A continuación se muestra el CEN asociado con este módulo

CEN DEL TEMPORIZADOR ASTABLE (TEMPOE)

```
0000 A6F1                       LDAA $F1,X
0002 2712                       BEQ NOPP
ACCIONAMIENTO AL ENTRAR Al ENTRAR A ESTE CÓDIGO LA PRIMERA VEZ
0004 1D0001          RESET:     BCLR $00,X,01;BSET PARA ARRANQUE EN
                                          NIVEL ALTO
0007 8620            INCON:     LDAA #$20;BYTE ALTO DE CONTADOR
0009 B70200                     STAA $020C:CARGA BYTE ALTO DE CONTADOR
                                          EN RAM
000C 18CE2122                   LDY #$2122
0010 18FF0201                   STY $0201;CARGA BYTES INTERMEDIO Y BAJO
                                        DE CONTADOR
0014 2033                       BRA SALIDA
```

```
 CHEQUEO DE NIVEL DE ENTRADA DE RESTABLECIMIENTO (RESET)
0016 A600            NOPP:     LDAA $00,X;CARGA VALOR PRESENTE DE GRUPO
                                        DE RESTABLECIMIENTO (RESET)
0018 8401                      ANDA #$01;ENMASCARA BIT DE ENTRADA DE
                                        RESTABLECIMIENTO
001A 27E8                      BEQ RESET;BNE PARA VERIFICACION DE
                                        DE RESET EN NIVEL ALTO
   ************  COMPARAR Y DECREMENTAR ************
001C B60200                    LDAA $0200
001F 8101                      CMPA #$01;COMPARA CON BYTE ALTO DE
                                        TIEMPO DE CONMUTACIÓN (Tc)
0021 2716                      BEQ COMPARA;COMPARACION DE BYTE
                                        INTERMEDIO Y BAJO DE CUENTA
0023 18FE0201        DECREM:   LDY $0201
0027 1809                      DEY
0029 18FF0201                  STY $0201
002D 261A                      BNE SALIDA
002F B60200                    LDAA $0200
0032 27D0                      BEQ RESET
0034 7A0200                    DEC $0200
0037 2010                      BRA SALIDA
0039 FC0201          COMPARA:  LDD $0201
003C 1A8303E8                  CPD #$03E8;COMPARA BYTES INTERMEDIO Y
                                        BAJO DE TIEMPO Tc
0040 2702                      BEQ VERIFSAL
0042 20DF                      BRA DECREM
0044 1C0001          VERIFSAL: BSET $00,X,01;BCLR PARA ARRANQUE EN
                                        NIVEL ALTO
0047 20DA                      BRA DECREM
0049 01              SALIDA:   NOP
```

En la tabla 3.15A se muestra la TAB asociada con el CEN TEMPOE y en la tabla 3.15B se aprecia el valor que deben tomar, de acuerdo con el tipo de VB que corresponda a las entradas y salida de este temporizador, parámetros que aparecen en la tabla 3.15A.

Tabla 3.15A Asignación de bytes asociada con el CEN TEMPOE, empleado para obtener el código requerido por los ML que realizan temporizadores astables.

Valor numérico (VN) generado por el software de traducción	Byte en CEN genérico para realizar temporizadores astables al que se le asignará el valor VN
VN=29, arranque en cero (B=0) VN=28, arranque en uno (B=1)	B4
VN=21T+BUFT	B5
VN=2^JT	B6
VN= BAC*	B8
VN= byte alto de DIRCON**	B10
VN= byte bajo de DIRCON**	B11
VN= BIC***	B14
VN= BBC	B15
VN= Byte alto de DIRCON + 1	B18

Tabla 3.15A Continuación.

VN= Byte bajo de DIRCON + 1	B19
VN= 2IR + BUFR	B23
VN= 2^JR	B25
VN= 39 si R se verifica en bajo (A=1) VN= 38 si R se verifica en alto (A=0)	B26
VN= Byte alto de DIRCON**	B29
VN= Byte bajo de DIRCON**	B30
VN= Byte alto de cuenta Tc	B32
VN = Byte alto de DIRCON + 1	B37
VN = Byte bajo de DIRCON + 1	B38
VN = Byte alto de DIRCON + 1	B43
VN = Byte bajo de DIRCON + 1	B44
VN = Byte alto de DIRCON	B48
VN = Byte bajo de DIRCON	B49
VN = Byte alto de DIRCON	B53
VN = Byte bajo de DIRCON	B54
VN = Byte alto de DIRCON + 1	B58
VN = Byte bajo de DIRCON + 1	B59
VN= Byte intermedio de cuenta Tc	B62
VN= Byte bajo de cuenta Tc	B63
VN= 28, arranque en cero (B=0) VN= 29, Arranque en uno (B=1)	B68
VN = 2IT + BUFT	B69
VN = 2^JT	B70

* BAC = Byte alto de cuenta asociada (CA)

CA = valor en centésimas de segundo del intervalo de tiempo implicado (Tm).

** DIRCON= DIRBCT + 3N (DIRBCT= dirección base de contadores de temporizadores).
N=número de temporizador asignado por el usuario.

*** BIC = Byte intermedio de cuenta asociada (CA)

**** BBC = Byte bajo de cuenta asociada (CA)

Tabla 3.15B Valores de los parámetros BUFR y BUFT de la tabla 3.15A

Caracter XR	**BUFR**
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Caracter XT	**BUFT**
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)

3-5-13 Descripción del ML que realiza temporizadores con capacidad para generar N pulsos a intervalos de tiempo especificados por el usuario.

El PLM puede realizar temporizadores denominados de tipo *multipulso* que generan N pulsos de anchura fija, a intervalos de tiempo especificíficos, tanto el número N como los intervalos de tiempo implicados son definidos por el usuario, una vez que ha transcurrido el flanco que lleva a la verificación del último pulso la salida permanece en ese nivel, verificándose otra salida del módulo denominada "testigo de fin de carrera", este módulo cuenta con dos entradas, una es para restablecimiento y la otra es una entrada denominada "entrada de congelamiento", al verificarse esta última el estado de la salida de pulsos permanece invariable, en las figuras 3.43 y 3.44 se muestran respectivamente la representación como bloque de este módulo y los diagramas de tiempo asociados, el nivel de verificación de los pulsos es definido por el usuario.

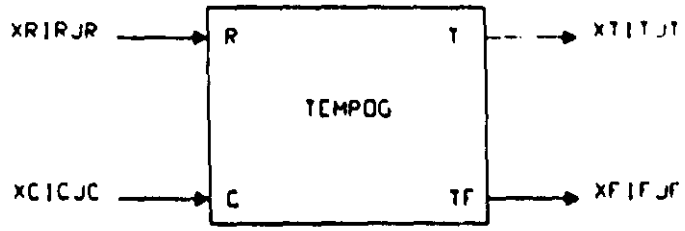


Figura 3.43 Representación genérica de un temporizador multipulso realizable en el PLM.

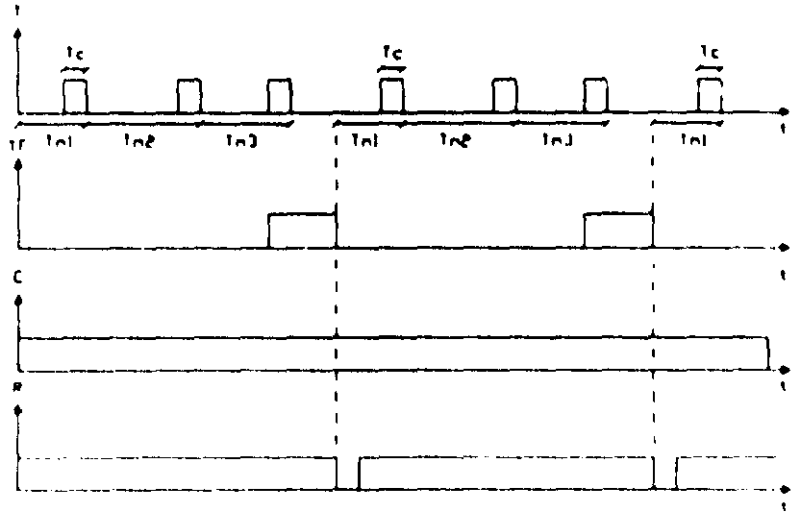


Figura 3.44 Diagramas de tiempos asociados con un temporizador multipulso, para esta ilustración el nivel de verificación de las dos entradas del módulo es bajo, el nivel de verificación, tanto de los pulsos como del testigo de fin de carrera es alto y el número de pulsos es tres.

Los tiempos Tc y Tmi (i = 1, 2, 3, ...N), pueden estar comprendidos entre 10ms y 47 horas con 22 minutos y 36.2 segundos, debiendo siempre el tiempo Tc ser menor que todos los tiempos Tmi; la entrada R responde al nivel, y al verificarse se coloca en la salida el nivel de arranque y se inicializa el contador asociado, dado que para este módulo el usuario debe declarar los N intervalos de tiempo implicados, la declaración de este módulo involucra varios renglones en un programa fuente en SIIL1.

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

TEMPOG#N XRIRJR, XCICJC, XTITJT, XFIFJF, NP, HH:MM:SS.CS, ABCDE;
Renglón de datos uno;
Renglón de datos dos;
.
.
.
.
último renglón de datos;

Donde:
N denota el número de temporizador, esto definido por el usuario.

NP denota el número de pulsos a generar.

XR 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.

IR denota el número de grupo que corresponda a la VB declarada como entrada "R" al temporizador.

JR denota el número de bit dentro del grupo IR, asociado a la variable de entrada "R".

XC podrá ser la letra "e", "s" o "i" mayúscula o minúscula, dependiendo esto de que la variable de entrada de congelamiento (C) al temporizador sea una VBE, VBS o VBI.

IC denota el número de grupo que corresponda a la VB declarada como entrada "C" al temporizador.

JC denota el número de bit dentro del grupo IC, asociado a la variable de entrada "C"

XT 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.

IT denota el número de grupo que corresponda a la VB declarada como salida del temporizador

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

XF podrá ser la letra "s" o "i" mayúscula o minúscula, dependiendo esto de que la variable de salida "TF" del temporizador sea una VBS o VBI.

IF denota el número de grupo que corresponda a la VB declarada como salida testigo de fin de carrera del temporizador.

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

HH denota un par de dígitos que especifican el número de horas en el tiempo Tc, siendo 47 el valor máximo aceptado.

MM denota un par de dígitos que especifican el número de minutos en Tc.

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

CS denota un par de dígitos que especifican las centésimas de segundo en 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=1), el temporizador se restablecerá por nivel bajo.

B es un dígito binario, que habrá de ser cero, si se desea que el arranque del tren de pulsos generado sea en bajo (pulsos verificados en 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 entrada de congelamiento sea bajo, en otro caso el dígito "C" deberá ser uno.

D es un dígito binario, que habrá de ser cero si se desea que la salida testigo de fin de carrera sea verificada en bajo, en otro caso el dígito "D" deberá ser uno.

E es un dígito binario, que habrá de ser cero si se desea deshabilitar la entrada de congelamiento, en otro caso (operación normal), el dígito "E" deberá ser uno; es importante señalar aquí que aún cuando la entrada de congelamiento sea deshabilitada, en la posición que corresponda a esta entrada en la declaración del módulo, deberá colocarse la especificación de una VB, de no hacerse esto el programa traductor de SIIL1 a código ejecutable por la CC del PLM, indicará un error de sintaxis.

En los renglones de datos habrán de colocarse, separados por comas, las especificaciones de tiempo que correspondan a cada uno de los Tmi (i = 1, 2, 3, ,N) implicados, al hacer esto el usuario es libre de colocar en cada renglón el número de

especificaciones de tiempo (Tmi) que le acomode, cada renglón de datos, excepto el último, deberá iniciar con el caracter "#" en la primera columna seguido por un espacio, el último renglón de datos deberá iniciar con dos caracteres "#" en la primera y segunda columna seguidos por un espacio, todos los renglones de datos deberán finalizar con el caracter ";".

Cabe señalar aquí, que en caso de que alguna especificación de tiempo Tmi tenga pares de ceros a la izquierda el usuario podrá, si lo desea, prescindir de ellos en la cadena correspondiente, así, un intervalo Tmi de siete segundos, podrá especificarse como 00:00:07.00 ó 00:07.00 ó 07.00.

El siguiente ejemplo ilustra como declarar temporizadores multidisparo en SIIL1.

Ejemplo 3.13

Supóngase que se desea realizar con el PLM un temporizador multidisparo que genere seis pulsos, es necesario que el arranque sea en cero (pulsos verificados en alto), se requiere que el tiempo Tc sea dos segundos, el nivel de restablecimiento ha de ser bajo y la VB asociada debe ser E06, la entrada de congelamiento "C" habrá de ser la VB E07 con un nivel de verificación en bajo, la salida de pulsos "T" deberá ser la VB S06 y la salida "TF" habrá de ser la VB S07 con verificación en alto, asignándosele a este temporizador el número siete.

Los seis intervalos de tiempo (Tmi) implicados deberán presentarse en el orden que se indican a continuación: treinta segundos, un minuto con treinta segundos, veinte segundos, cinco segundos con 250 ms, dos minutos y un minuto con dos segundos.

La declaración de este temporizador multidisparo podría ser:

TEMPOG#7 E06, E07, S06, S07, 6, 00:00:02.00, 10011;

30.00, 01:30.00, 20.00, 05.25;

02.00.00, 01:02.00,

En la figura 3.45 se muestra la representación como bloque del temporizador de este ejemplo y su diagramas de tiempo asociado.

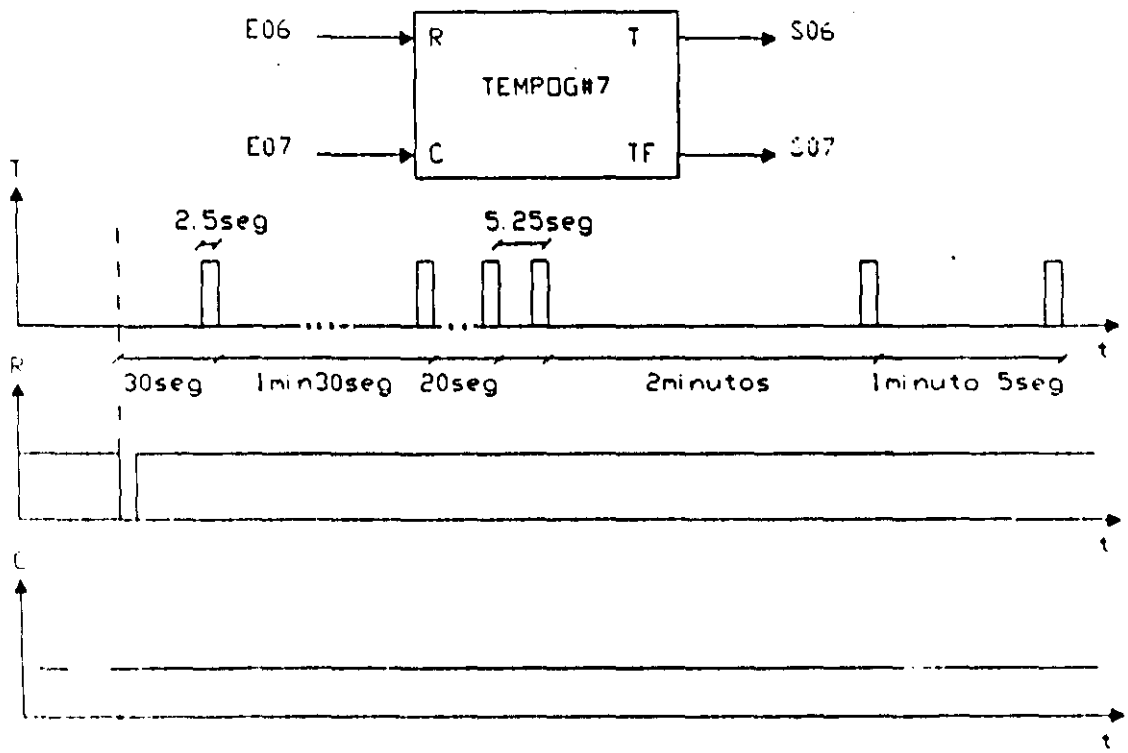


Figura 3.45 Diagrama de tiempo y representación como bloque, del temporizador multidisparo del ejemplo 3.13.

Descripción del CEN asociado para temporizadores multipulso (TEMPOG)

El flujo de ejecución de este ML se muestra en la figura 3.46, el tiempo máximo de ejecución de este módulo es de 92 μs (fc=2 Mhz).

Dado que este módulo requiere de un buffer de datos con los valores de los tiempos Tmi, el código objeto correspondiente al mismo lo deberá contener, en la figura 3.47 se muestra la estructura que en la memoria de la CC del PLM tendría el tramo de código correspondiente a un temporizador multidisparo, nótese que el buffer de datos correspondiente está colocado inmediatamente después del código ejecutable correspondiente.

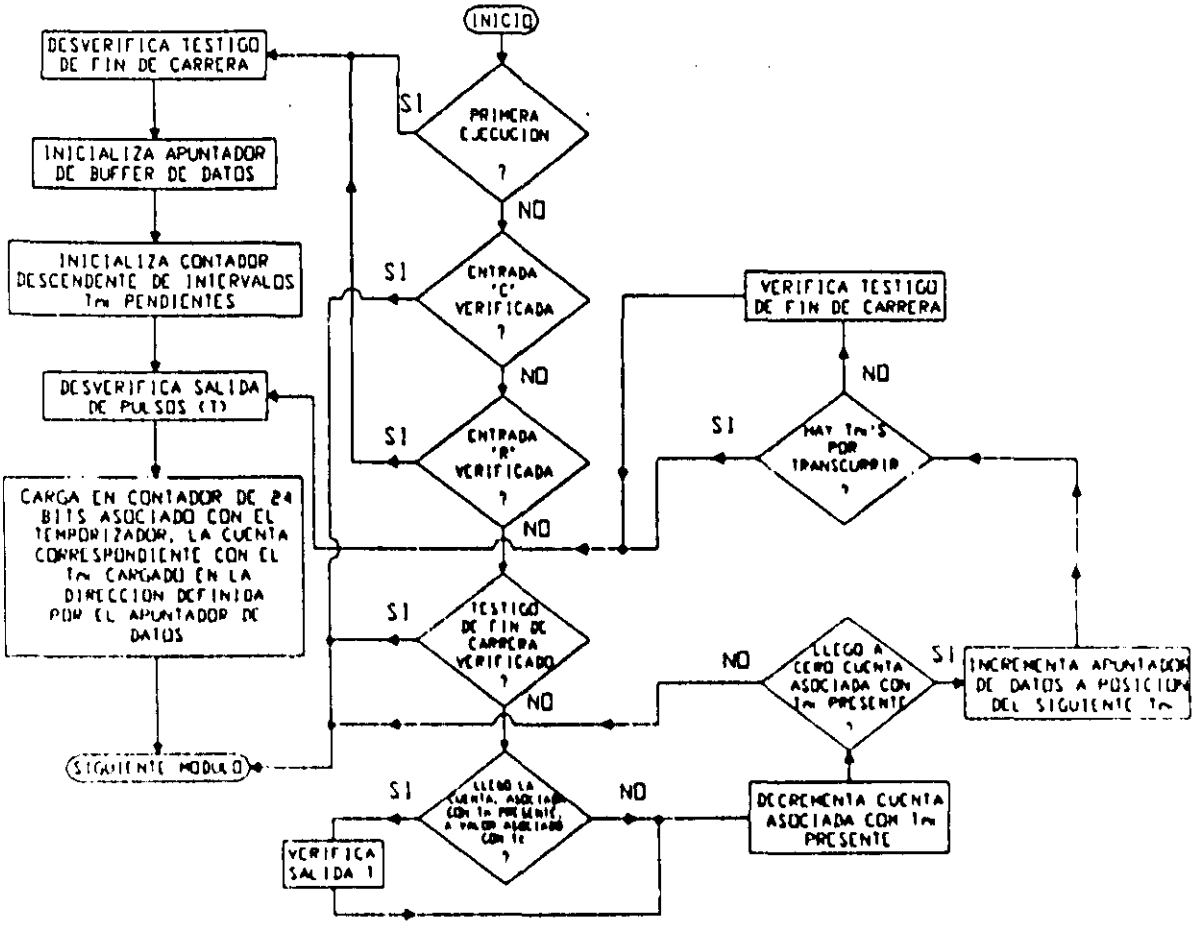


Figura 3.46 Flujo de ejecución del temporizador multipulso realizable por el PLM (TEMPOG).

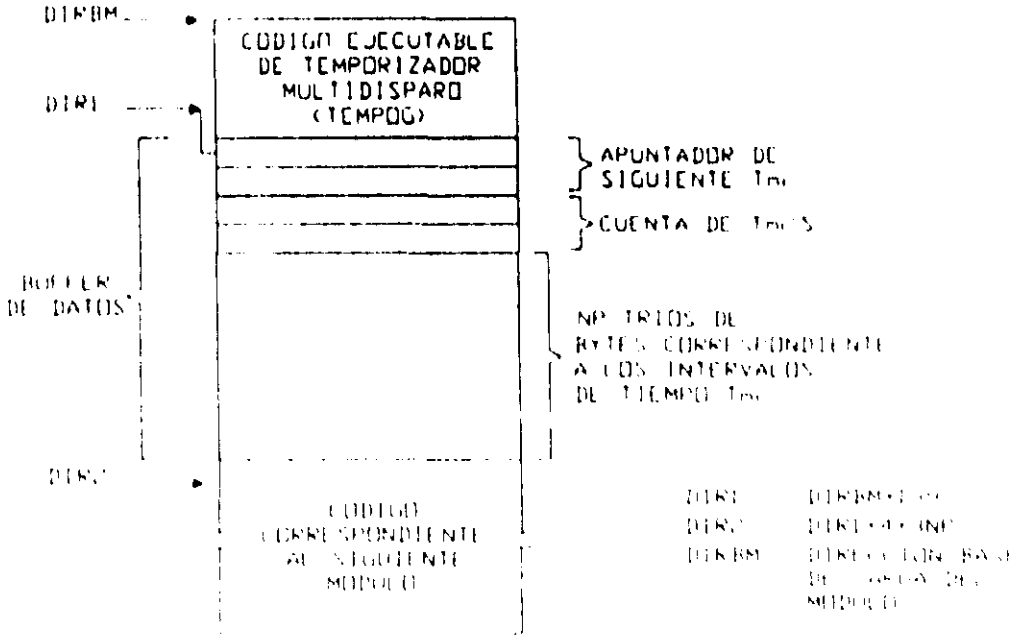


Figura 3.47 Estructura del código ejecutable correspondiente a un temporizador multipulso

A continuación se muestra el CEN asociado con este módulo

CEN DEL TEMPORIZADOR MULTIPULSO (TEMPOG)

```
0000 A6F1                          LDAA $F1,X
0002 2724                          BEQ NOPP
ACCIONAMIENTO AL ENTRAR A ESTE CÓDIGO LA POR PRIMERA VEZ
0004 1D0002          RESET:        BCLR $00,X,02;BSET PARA TESTIGO DE FIN
                                     DE CARRERA VERIFICADO BAJO
0007 CCC1AA                        LDD #$C1AA;CARGA D CON $DIR1 + 4
000A FDC150                        STD $C150;CARGA APUNTADOR INICIAL EN
                                     BUFER DE DATOS ASOCIADO
000D CC00F0                        LDD #$00F0;CARGA D CON NUMERO DE TIEMPOS
                                     Tmi
0010 FDC152                        STD $C152;CARGA NUMERO DE TIEMPOS Tmi
                                     EN BUFFER DE DATOS ASOCIADO
0013 1D0001          RESETF:       BCLR $00,X,01;BSET PARA ARRANQUE EN
                                     NIVEL ALTO
0016 18FEC150        INCON:        LDY $C150;CARGA IY CON APUNTADOR
                                     INICIAL DE DATOS DE TIEMPOS
                                     Tmi
001A 18A600                        LDAA $00,Y;COPIA EN A BYTE ALTO DE
                                     CUENTA ASOCIADA CON Tmi
001D B70200                        STAA $0200
0020 18EC01                        LDD $01,Y;COPIA EN D LOS DOS BYTES MENOS
                                     SIGNIFICATIVOS DE CUENTA
                                     ASOCIADA CON TMi
0023 FD0201                        STD $0201
0026 2060                          BRA SALIDA
      CHEQUEO DE NIVEL DE ENTRADA DE CONGELAMIENTO
0028 A600            NOPP:         LDAA $00,X;CARGA VALOR PRESENTE DE GRUPO
                                     DE ENTRADA DE CONGELAMIENTO
002A 8404                          ANDA #$04;ENMASCARA BIT DE ENTRADA DE
                                     CONGELAMIENTO
002C 275A                          BEQ SALIDA;BNE PARA CONGELAMIENTO
                                     VERIFICADO EN ALTO
      CHEQUEO DE NIVEL DE ENTRADA DE RESTABLECIMIENTO (RESET)
002E A600                          LDAA $00,X;CARGA VALOR PRESENTE DE GRUPO
                                     DE ENTRADA "R"
0030 8401                          ANDA #$01;ENMASCARA BIT DE ENTRADA "R"
0032 27D0                          BEQ RESET;BNE PARA RESET EN ALTO
    CHEQUEO DE VERIFICACION DEL TESTIGO DE FIN DE CARRERA
0034 A640                          LDAA $40,X;LEE GRUPO DE TESTIGO DE FIN
                                     DE CARRERA
0036 8402                          ANDA #$02;ENMASCARA BIT TESTIGO DE FIN
                                     DE CARRERA
0038 264E                          BNE SALIDA;BEQ PARA TESTIGO DE FIN DE
                                     FIN DE CARRERA VERIFICADO
                                     EN NIVEL BAJO
    ************** COMPARAR Y DECREMENTAR **************
003A B60200                        LDAA $0200
003D 8101                          CMPA #$01;COMPARA CON BYTE ALTO DE
                                     TIEMPO DE CONMUTACIÓN Tc
003F 2716                          BEQ COMPARA; COMPARACION DE BYTE
                                       INTERMEDIO Y BAJO DE TIEMPO
                                       DE CONMUTACIÓN Tc
0041 18FE0201        DECREM:       LDY $0201
0045 1809                          DEY
0047 18FF0201                      STY $0201
```

```
004B 263B                       BNE SALIDA
004D B60200                     LDAA $0200
0050 2710                       BEQ CAMBIODETM
0052 7A0200                     DEC $0200
0055 2031                       BRA SALIDA
0057 FC0201          COMPARA:   LDD $0201
005A 1A8303E8                   CPD #$03E8;COMPARA BYTES INTERMEDIO Y
                                          BAJO DE TIEMPO DE CONMUTACIÓN
                                          Tc
005E 2723                       BEQ VERIFSAL
0060 20DF                       BRA DECREM
0062 18FEC150        CAMBIODETM:LDY $C150;CARGA IY CON APUNTADOR DE
                                        SIGUIENTE TRIO DE BYTES
                                        ASOCIADO CON TIEMPO Tmi
0066 1808                       INY
0068 1808                       INY
006A 1808                       INY
006C 18FFC150                   STY $C150
0070 18FEC152                   LDY $C152;CARGA IY CON NUMERO DEL Tmi
                                          RECIEN TRANSCURRIDO
0074 1809                       DEY
0076 18FFC152                   STY $C152;ACTUALIZA CONTADOR DESCENDENTE
                                          DE TIEMPOS Tmi TRANSCURRIDOS
007A 2702                       BEQ VERIFTES
007C 2095                       BRA RESETF
007E 1C0201          VERIFTES:  BSET $02,X,01;BCLR PARA TESTIGO DE FIN
                                            DE CARRERA VERIFICADO BAJO
0081 2090                       BRA RESETF
0083 1C0001          VERIFSAL:  BSET $00,X,01;BCLR PARA ARRANQUE EN
                                            NIVEL ALTO
0086 20B9                       BRA DECREM
0088 7EC200          SALIDA:    JMP $C200;SALTO A SIGUIENTE MÓDULO
                                          (DIR2), A TRAVES DEL BUFER DE
                                          DATOS ASOCIADO
```

En la tabla 3.16A se muestra la TAB asociada con el CEN TEMPOG y en la tabla 3.16B se aprecia el valor que deben tomar, de acuerdo con el tipo de VB que corresponda a las entradas y salida de este temporizador, parámetros que aparecen en la tabla 3.16A

Tabla 3.16A Asignación de bytes asociada con el CEN TEMPOG, empleado para obtener el código requerido por los ML que realizan temporizadores multidiparo.

Valor numérico (VN) generado por el software de traducción	Byte en CEN genérico para realizar temporizadores multidiparo al que se le asignará el valor VN
VN=29, salida TF verificada en alto (D= 1) VN=28, salida TF verificada en bajo (D= 0)	B4
VN=2IF+BUFF	B5
VN=2^JF	B6
VN= Byte alto de DIR1* + 4	B8
VN= byte bajo de DIR1 + 4	B9
VN= byte alto de DIR1	B11

Tabla 3.16A Continuación	
VN= Byte bajo de DIR1	B12
VN= Byte alto de NP	B14
VN= Byte bajo de NP	B15
VN= Byte alto de DIR1 + 2	B17
VN= Byte bajo de DIR1 + 2	B18
VN=29, arranque en cero, pulsos en verificados en alto, (B = 0) VN=28, arranque en uno, pulsos verificados en bajo, (B = 1)	B19
VN= 2IT + BUFT	B20
VN= 2^JT	B21
VN= Byte alto de DIR1	B24
VN= Byte bajo de DIR1	B25
VN= Byte alto de de DIRCON**	B30
VN = Byte bajo de DIRCON	B31
VN = Byte alto de DIRCON + 1	B36
VN = Byte bajo de DIRCON + 1	B37
VN = 2IC+BUFC	B41
VN = 2^JC	B43
VN = 39, congelamiento en bajo, (C=0) VN = 38,congelamiento en alto, (C=1)	B44
VN = 2IH+ BUFH	B47
VN = 2^JH	B49
VN = 39, restablecimiento en bajo, (A=1) VN = 38, restablecimiento en alto, (A=0)	B50
2IF + BUFF	B53
2^JF	B55
VN = 39, TF verificado en bajo, (D=0) VN = 38, TF verificado en alto, (D=1)	B56
VN = Byte alto de DIRCON	B59
VN = Byte bajo de DIRCON	B60
VN = Byte alto de cuenta Tc	B62
VN = Byte alto de DIRCON + 1	B67
VN = Byte bajo de DIRCON + 1	B68
VN = Byte alto de DIRCON + 1	B73
VN = Byte bajo de DIRCON + 1	B74
VN = Byte alto de DIRCON	B78
VN = Byte bajo de DIRCON	B79
VN = Byte alto de DIRCON	B83
VN = Byte bajo de DIRCON	B84
VN = Byte alto de DIRCON + 1	B88
VN = Byte bajo de DIRCON + 1	B89

Continuación de Tabla 3.16A	
VN= Byte intermedio de cuenta Tc	B92, (último renglón de tabla
VN= Byte bajo de cuenta Tc	B93
VN = Byte alto de DIR1	B100
VN = Byte bajo de DIR1	B101
VN = Byte alto de DIR1	B110
VN = Byte bajo de DIR1	B111
VN = Byte alto de DIR1 + 2	B114
VN = Byte bajo de DIR1 +.2	B115
VN = Byte alto de DIR1 + 2	B120
VN = Byte bajo de DIR1 + 2	B121
VN = 29, TF verificado en bajo, (D=0) VN = 28, TF verificado en alto, (D=1)	B126
VN = 2IF + BUFF	B127
VN = 2^JF	B128
VN=28, arranque en cero, pulsos en verificados en alto, (B = 0) VN=29, arranque en uno, pulsos verificados en bajo, (B = 1)	B131
VN = 2IT + BUFT	B132
VN = 2^JT	B133
VN = Byte alto de DIR2***	B137
VN = Byte bajo de DIR2	B138

* DIR1 = DIRBM + 139 (La variable DIRBM es generada por el Software de traducción)
** DIRCON= DIRBCT + 3N (DIRBCT= dirección base de contadores de temporizadores).
*** DIR2 = DIR1 + 4 + 3NP

Tabla 3.16B Valores de los parámetros BUFR, BUFC, BUFT y BUFF de la tabla 3.16A

Caracter XR	BUFR
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Caracter XC	BUFC
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Caracter XT	BUFT
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)
Caracter XF	BUFF
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)

3-5-14 Descripción del ML que realiza temporizadores con capacidad de generación de pulsos en instantes de acuerdo al estado del reloj de tiempo real (RTR).

El PLM puede realizar temporizadores denominados de tipo *GPRTR*, con capacidad para generar pulsos en instantes preestablecidos de acuerdo con el RTR, la duración de los mismos es aproximadamente de 100 ms, de acuerdo con la forma en que se especifiquen los instantes en que se desea se produzcan los pulsos los temporizadores GPRTR pueden ser de seis clases a saber:

1.- Temporizador GPRTR de clase uno, en este caso se especifican los instantes de disparo con un par de dígitos, que indican los segundos en los que se desea que aparezcan los pulsos; por ejemplo, 43 sería una especificación de disparo para el segundo cuarenta y tres de cada minuto.

2.- Temporizador GPRTR de clase dos, en este caso se especifican los instantes de disparo con dos pares de dígitos, que indican los minutos y segundos en los que se desea que aparezcan los pulsos; por ejemplo, 37:15 sería una especificación de disparo para el minuto treinta y siete con quince segundos de cada hora.

3.- Temporizador GPRTR de clase tres, en este caso se especifican los instantes de disparo con tres pares de dígitos, que indican en que hora, minuto y segundo se desea que aparezcan los pulsos; por ejemplo, 13:23:10 sería una especificación de disparo para las trece horas con veintitrés minutos y diez segundos.

4.- Temporizador GPRTR de clase cuatro, en este caso se especifican los instantes de disparo con un par de caracteres que indican un dia de la semana seguidos por tres pares de dígitos, de esta forma cada especificación indica en que hora minuto y segundo y día de la semana se desea que aparezcan los pulsos; por ejemplo, VI/00:12:08 sería una especificación de disparo para las cero horas con doce minutos y ocho segundos de un dia viernes.

El par de caracteres empleados para denotar a cada uno de los dias de la semana es: "LU" para el lunes, "MA" para el martes, "MI" para el miércoles, "JU" para el jueves, "VI" para el viernes, "SA" para el sábado y "DO" para el domingo.

5.- Temporizador GPRTR de clase cinco, en este caso se especifican los instantes de disparo con cinco pares de dígitos, que indican en que número de dia, mes, hora, minuto y segundo se desea que aparezcan los pulsos, por ejemplo, 08/05/12 30 02 sería una especificación de disparo para las doce horas con treinta minutos y dos segundos de un dia ocho de mayo

6.- Temporizador GPRTR de clase seis, en este caso se especifican los instantes de disparo con seis pares de dígitos, que indican en que día de mes, mes, año, hora, minuto y segundo se desea que aparezcan los pulsos; por ejemplo, 03/31/02/15:12:13 sería una especificación de disparo para las quince horas con doce minutos y trece segundos del treinta y uno de marzo del año dos mil dos.

Los caracteres delimitadores "/" y ":" pueden ser remplazados por cualquier otro, ya que el software de traducción no los toma en cuenta para fines de la obtención de la representación binaria asociada con cada una de las especificaciones de disparo, correspondientes a un temporizador de tipo GPRTR.

Este temporizador es deshabilitado siempre que el usuario tenga oprimido el botón auxiliar "A" (BAXA), el cual es empleado para habilitar el poder poner a tiempo el RTR mediante los botones auxiliares BAXB y BAXC, presentes en el BCLD del PLM, para más información acerca de como colocar a tiempo el RTR puede consultarse en este trabajo el módulo que lo maneja que es denominado como RTRX.

En la figura 3.48 se muestra la representación genérica de un temporizador de tipo GPRTR.

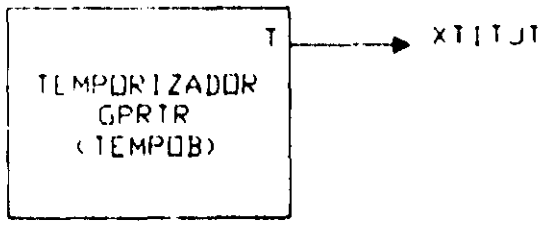


Figura 3.48 Representación genérica de un temporizador de tipo GPRTR realizable en el PLM.

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

TEMPOB#N XTITJT, CLASE, NT, A;

Renglón de datos uno;
Renglón de datos dos;

último renglón de datos.

Donde:

N denota el número de temporizador, esto definido por el usuario; cabe señalar aquí el hecho de que la numeración para módulos GPRTR es independiente de la correspondiente a la correspondiente con los otros cinco temporizadores que puede realizar el PLM.

CLASE es un dígito que denota la clase de temporizador GPRTR que se desea realizar.

NT denota el número de pulsos a generar, el cual es limitado a cincuenta por módulo.

XT 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.

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

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

A es un dígito binario, que habrá de ser cero, si se desea que los pulsos de salida sean verificados en bajo, en otro caso (pulsos de salida verificados en alto) "A" deberá ser uno.

En los renglones de datos habrán de colocarse, separados por comas, las especificaciones de disparo deseadas, al hacer esto el usuario es libre de colocar en cada renglón el número de especificaciones de disparo que le acomode, cada renglón de datos, excepto el último, deberá iniciar con el caracter "#" en la primera columna seguido por un espacio, el último renglón de datos deberá iniciar con dos caracteres "#" en la primera y segunda columna seguidos por un espacio, todos los renglones de datos deberán finalizar con el caracter ";".

El siguiente ejemplo ilustra como declarar temporizadores GPRTR en SIIL1.

Ejemplo 3.14

Supóngase que se desea realizar con el PLM un temporizador de tipo GPRTR que genere cinco pulsos verificados en alto, los primeros dos habrán de presentarse los días lunes

a las ocho de la mañana en punto y a las once horas con cinco minutos, los otros tres se deberán presentar los días miércoles jueves y sábado a las cinco de la tarde; la salida "T" deberá ser la VBS S12, asignándosele el número nueve; el temporizador GPRTR obviamente seria de clase cuatro.

La declaración de este temporizador GPRTR podría ser:

TEMPOB#9 S12, 4, 5, 1;

LU/08:00:00, LU/11:05:00, MI/17:00:00;

JU/17:00:00, SA/17:00:00;

En la figura 3.49 se muestra la representación como bloque del temporizador de este ejemplo.

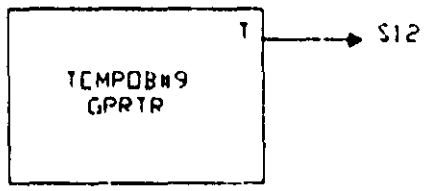


Figura 3.49 Diagrama de tiempo y representación como bloque, del temporizador GPRTR del ejemplo 3.14.

Descripción del CEN asociado para temporizadores GPRTR (TEMPOB)

El flujo de ejecución de este ML se muestra en la figura 3.50, el tiempo máximo de ejecución en microsegundos (Tmtb) para este módulo, suponiendo que fc = 2MHz, está dado por la siguiente ecuación:

$$Tmtb = 232 + (30 + 8CLASE)NT \quad \ldots\ldots (3.3)$$

Dado que este módulo requiere de un buffer de datos con los valores de los tiempos en los que se desea aparezcan los pulsos de salida, el código objeto correspondiente al mismo lo deberá contener, en la figura 3.51 se muestra la estructura que en la memoria de la CC del PLM tendría el tramo de código correspondiente a un temporizador GPRTR, notese que el buffer de datos correspondiente esta colocado inmediatamente despues del codigo ejecutable correspondiente.

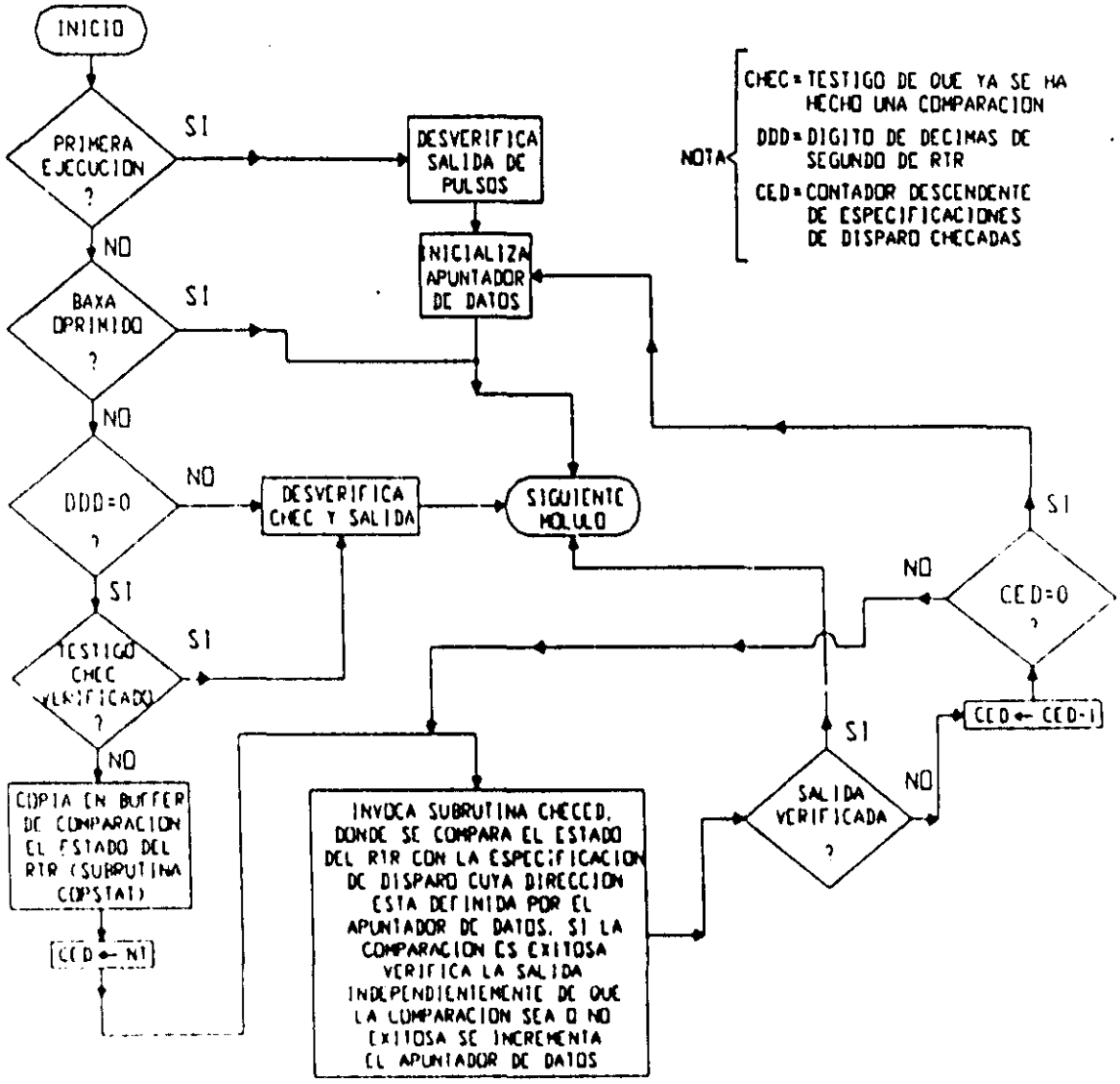


Figura 3.50 Flujo de ejecución del temporizador GPRTR realizable por el PLM (TEMPOB)

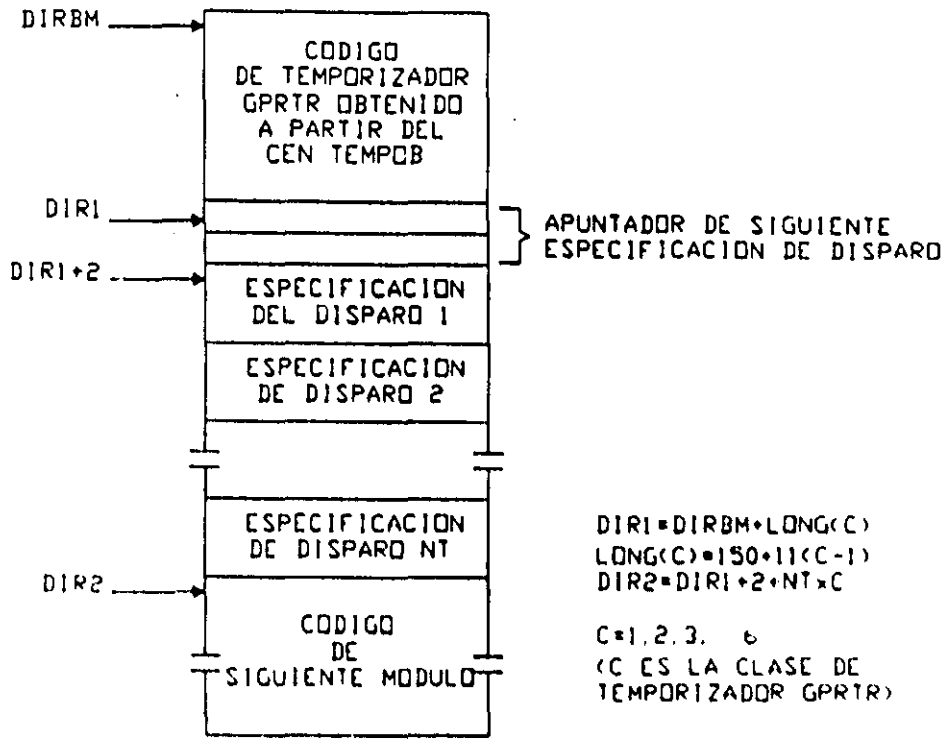


Figura 3.51 Estructura del código ejecutable correspondiente a un temporizador GPRTR.

El CEN asociado con este módulo es en esencia el mismo para cada una de las clases (1 a 6) que se pueden presentar; sin embargo, debido a que la longitud de las especificaciones de disparo varía desde un byte para la clase uno, hasta seis bytes para la clase seis, la subrutina CHECED (véase flujo de ejecución en figura 3.50), tendrá una secuencia de instrucciones que variará de acuerdo con la clase de temporizador GPRTR para el cual se va a obtener el código objeto.

A continuación se muestra el CEN asociado con este módulo cuando la clase es cuatro, posteriormente se muestra la subrutina CHECED para cada una de las otras cinco clases que se pueden presentar.

CEN DEL TEMPORIZADOR GPRTR DE CLASE CUATRO (TEMPOB4)

```
0000 A6F2              INIRTRPD:   LDAA $F2,X
0002 270B                          BEQ NOPPP
                CODIGO AL ENTRAR LA PRIMERA VEZ
0004 1D0201            RESETT:     BCLR $02,X,01;BSET PARA PULSO DE SALIDA
                                                 VERIFICADO EN BAJO
0007 CC00B9            RR:         LDD #DISPAROS+2;CARGA D CON APUNTADOR DE
                                                   PRIMERA ESPECIFICACION
                                                   DE DISPARO
```

```
000A FD00B7                         STD DISPAROS;CARGA EN BUFFER ASOCIADO
                                                APUNTADOR DE PRIMERA
                                                ESPECIFICACIÓN DE DISPARO
000D 202E                           BRA SALABB
000F B61BB0           NOPPP:        LDAA $1BB0
0012 8401                           ANDA #$01
0014 2727                           BEQ SALABB;SALTA A SALIDA SI EL BOTON DE
                                               PONER EL RELOJ ESTÁ EN BAJO.
0016 B61BC1                         LDAA $1BC1
0019 840F                           ANDA #$0F
001B 270A                           BEQ ACCION
     DESVERIFICA SALIDA Y TESTIGO DE QUE YA SE PASO A SUB CHECED
001D 1D0201                         BCLR $02,X,01;BSET PAR PULSO DE SALIDA
                                                  VERIFICADO BAJO
0020 8600                           LDAA #$00
0022 B70040                         STAA TESTIGO
0025 2016                           BRA SALABB
0027 B60040           ACCION:       LDAA TESTIGO
002A 2611                           BNE SALABB;SALE SI YA SE HAN COMPARADO
                                              ESPECIFICACIONES DE DISPARO
   CHECA ESPECIFICACIONES DE DE DISPARO CONTRA ESTADO DE RTR
002C 8D57             CHECAESDI: BSR COPSTAT;SE INVOCA SUBRUTINA DE
                                              COPIADO DE ESTADO DEL RTR
                                              EN BUFFER AUXILIAR
002E C603                           LDAB #$03;CARGA B CON NÚMERO DE
                                             ESPECIFICACIONES DE DISPARO
                                             (NT)
0030 8D0F             SIGESP:       BSR CHECED;SE INVOCA SUBRUTINA QUE
                                              VERIFICA LA SALIDA SI LA
                                              ESPECIFICACIÓN DE DISPARO
                                              COINCIDE CON EL TIEMPO DEL
                                              RTR
0032 A602                           LDAA $02,X;COPIA ESTADO DE GRUPO AL QUE
                                              PERTENECE LA SALIDA
0034 8401                           ANDA #$01;ENMASCARA BIT DE SALIDA
0036 2605                           BNE SALABB;BEQ SI LA SALIDA SE VERIFICA
                                              EN BAJO
0038 5A                             DECB
0039 26F5                           BNE SIGESP
003B 20CA                           BRA RR
                              SALTA A DIR2
003D 7ECCAA           SALABB:       JMP $CCAA
0040 00               TESTIGO:      FCB $00
                SUBRUTINA QUE CHECA SIGUIENTE ESPECIFICACIÓN
                      DE DISPARO CONTRA ESTADO DE RTR.
0041 36               CHECED:       PSHA
  VERIFICA TESTIGO DE QUE SE HA ENTRADO A SUBRUTINA CHECED
0042 86FF                           LDAA #$FF
0044 B7CCFF                         STAA $CCFF;LA DIRECCIÓN REAL AQUÍ
                                               REQUERIDA DEBE SER DIRBM+ $40
                                               (TESTIGO CHEC)
0047 18FE00B7                       LDY DISPAROS;CARGA Y CON APUNTADOR DE
                                                 INICIO DE SIGUIENTE
                                                 ESPECIFICACIÓN DE
                                                 DISPARO. (LDY $DIR1)
004B 18A600           CHECSEG:      LDAA $00,Y
004E A1C2                           CMPA $C2,X
0050 2622                           BNE NOFSEG
0052 1808                           INY
0054 18A600           CHECMIN:      LDAA $00,Y
```

```
0057 A1C3                       CMPA $C3,X
0059 261B                       BNE NOFMIN
005B 1808                       INY
005D 18A600           CHECHOR:  LDAA $00,Y
0060 A1C4                       CMPA $C4,X
0062 2614                       BNE NOFHOR
0064 1808                       INY
0066 18A600           CHECDS:   LDAA $00,Y
0069 A1C5                       CMPA $C5,X
006B 260D                       BNE NOFDS
006D 1808                       INY
006F 1C0201           VERIFSAL: BSET $02,X,01;BCLR SI LA SALIDA SE
                                          VERIFICA EN BAJO
0072 200B                       BRA SALSUB
0074 1808             NOFSEG:   INY
0076 1808             NOFMIN:   INY
0078 1808             NOFHOR:   INY
007A 1808             NOFDS:    INY
007C 1D0201           DESVERSAL: BCLR $02,X,01;BSET SI LA SALIDA SE
                                          VERIFICA EN BAJO
007F 18FF00B7         SALSUB:   STY DISPAROS;(STY $DIR1)
0083 32                         PULA
0084 39                         RTS
SUBRUTINA COPSTAT;COPIA EL STATUS DEL RTR EN BUFFER AUXILIAR
(X+$C2)<----SEGUNDOS,(X+$C3)<----MINUTOS,.....,(X+$C8)<----AÑO
0085 3C               COPSTAT:  PSHX
0086 18CE1BC2                   LDY #$1BC2;CARGA Y CON DIRECCIaN BASE+2
                                          DE RTR,(DIR DE SEGS)
008A 18A601           ALFA:     LDAA $01,Y
008D 840F                       ANDA #$0F
008F C60A                       LDAB #$0A
0091 3D                         MUL
0092 18A600                     LDAA $00,Y
0095 840F                       ANDA #$0F
0097 1B                         ABA
0098 A7C2                       STAA $C2,X
009A 1808                       INY
009C 1808                       INY
009E 08                         INX
009F 188C1BC8                   CPY #$1BC8
00A3 2601                       BNE BETA
00A5 08                         INX
00A6 188C1BCE         BETA:     CPY #$1BCE
00AA 26DE                       BNE ALFA
00AC 18A600                     LDAA $00,Y;COPIA EN A DIA DE LA SEMANA
                                          CONTENIDO EN RTR.
00AF 840F                       ANDA #$0F
00B1 38                         PULX
00B2 A7C5                       STAA $C5,X;COPIA EN BUFFER AUXILIAR DIA
                                          DE LA SEMANA TOMADO DE RTR.
00B4 6AC7                       DEC $C7,X;AJUSTA TESTIGO DE NUMERO DE
                                          MES EN BUFFER AUXILIAR,DE MODO
                                          QUE ENERO=0,...DICIEMBRE=11
00B6 39                         RTS
                      DISPAROS:
```

La forma que toma la subrutina CHECED para los temporizadores GPRTR de clase diferente de cuatro, se muestra a continuación.

Subrutina CHECED para el temporizador GPRTR de clase uno

```
SUBRUTINA QUE CHECA SIGUIENTE ESPECIFICACIÓN DE DISPARO CONTRA ESTADO DE
RTR PARA EL TEMPORIZADOR GPRTR DE CLASE UNO
CHECED:     PSHA
     VERIFICA TESTIGO DE QUE SE HA ENTRADO A SUBRUTINA CHECED
            LDAA #$FF
            STAA $CCFF;LA DIR AQUÍ ESPECIFICADA ES DIRBM+$40 (TESTIGO
CHEC)
            LDY DISPAROS;CARGA IY CON APUNTADOR DE INICIO DE SIGUIENTE
                        ESPECIFICACIÓN DE DISPARO,(LDY $DIR1).
CHECSEG:    LDAA $00,Y
            CMPA $C2,X
            BNE NOFSEG
            INY
VERIFSAL:   BSET $02,X,01;BCLR SI LA SALIDA SE VERIFICA EN BAJO
            BRA SALSUB
NOFSEG:     INY
DESVERSAL:  BCLR $02,X,01;BSET SI LA SALIDA SE VERIFICA EN BAJO

SALSUB:     STY DISPAROS; (STY $DIR1)
            PULA
            RTS
```

Subrutina CHECED para el temporizador GPRTR de clase dos

```
     SUBRUTINA QUE CHECA SIGUIENTE ESPECIFICACIÓN DE DISPARO CONTRA
ESTADO DE RTR PARA EL TEMPORIZADOR GPRTR DE CLASE DOS
CHECED:     PSHA
     VERIFICA TESTIGO DE QUE SE HA ENTRADO A SUBRUTINA CHECED
            LDAA #$FF
            STAA $CCFF;LA DIR AQUÍ ESPECIFICADA ES DIRBM+$40 (TESTIGO
CHEC)
            LDY DISPAROS;CARGA Y CON APUNTADOR DE INICIO DE SIGUIENTE
ESPECIFICACIÓN DE DISPARO,(LDY $DIR1).
CHECSEG:    LDAA $00,Y
            CMPA $C2,X
            BNE NOFSEG
            INY
CHECMIN:    LDAA $00,Y
            CMPA $C3,X
            BNE NOFMIN
            INY
VERIFSAL:   BSET $02,X,01;BCLR SI LA SALIDA SE VERIFICA EN BAJO
            BRA SALSUB
NOFSEG:     INY
NOFMIN:     INY
DESVERSAL:  BCLR $02,X,01;BSET SI LA SALIDA SE VERIFICA EN BAJO

SALSUB:     STY DISPAROS; (STY $DIR1)
            PULA
            RTS
```

Subrutina CHECED para el temporizador GPRTR de clase tres

```
     SUBRUTINA QUE CHECA SIGUIENTE ESPECIFICACIÓN DE DISPARO CONTRA
ESTADO DE RTR PARA TEMPORIZADORES GPRTR DE CLASE TRES
CHECED:     PSHA
     VERIFICA TESTIGO DE QUE SE HA ENTRADO A SUBRUTINA CHECED
            LDAA #$FF
            STAA $CCFF;LA DIR AQUÍ ESPECIFICADA ES DIRBM+$40 (TESTIGO
CHEC)
            LDY DISPAROS;CARGA Y CON APUNTADOR DE INICIO DE SIGUIENTE
ESPECIFICACIÓN DE DISPARO,(LDY $DIR1).
CHECSEG:    LDAA $00,Y
            CMPA $C2,X
            BNE NOFSEG
            INY
CHECMIN:    LDAA $00,Y
            CMPA $C3,X
            BNE NOFMIN
            INY
CHECHOR:    LDAA $00,Y
            CMPA $C4,X
            BNE NOFHOR
            INY
VERIFSAL:   BSET $02,X,01;BCLR SI LA SALIDA SE VERIFICA EN BAJO
            BRA SALSUB
NOFSEG:     INY
NOFMIN:     INY
NOFHOR:     INY
DESVERSAL:  BCLR $02,X,01;BSET SI LA SALIDA SE VERIFICA EN BAJO

SALSUB:     STY DISPAROS; (STY $DIR1)
            PULA
            RTS
```

Subrutina CHECED para el temporizador GPRTR de clase cinco

```
     SUBRUTINA QUE CHECA SIGUIENTE ESPECIFICACIÓN DE DISPARO CONTRA
ESTADO DE RTR.PARA TEMPORIZADORES GPRTR DE CLASE CINCO
CHECED:     PSHA
     VERIFICA TESTIGO DE QUE SE HA ENTRADO A SUBRUTINA CHECED
            LDAA #$FF
            STAA $CCFF;LA DIR AQUÍ ESPECIFICADA ES DIRBM+$40 (TESTIGO
CHEC)
            LDY DISPAROS;CARGA Y CON APUNTADOR DE INICIO DE SIGUIENTE
ESPECIFICACIÓN DE DISPARO,(LDY $DIR1).
CHECSEG:    LDAA $00,Y
            CMPA $C2,X
            BNE NOFSEG
            INY
CHECMIN:    LDAA $00,Y
            CMPA $C3,X
            BNE NOFMIN
            INY
CHECHOR:    LDAA $00,Y
            CMPA $C4,X
            BNE NOFHOR
            INY
CHECDM:     LDAA $00,Y
```

```
            CMPA $C6,X
            BNE NOFDM
            INY
CHECME:     LDAA $00,Y
            CMPA $C7,X
            BNE NOFME
            INY
VERIFSAL:   BSET $02,X,01;BCLR SI LA SALIDA SE VERIFICA EN BAJO
            BRA SALSUB
NOFSEG:     INY
NOFMIN:     INY
NOFHOR:     INY
NOFDM:      INY
NOFME:      INY
DESVERSAL:  BCLR $02,X,01;BSET SI LA SALIDA SE VERIFICA EN BAJO

SALSUB:     STY DISPAROS; (STY $DIR1)
            PULA
            RTS
```

Subrutina CHECED para el temporizador GPRTR de clase seis

```
     SUBRUTINA QUE CHECA SIGUIENTE ESPECIFICACIÓN DE DISPARO CONTRA
ESTADO DE RTR PARA TEMPORIZADORES GPRTR DE CLASE SEIS
CHECED:     PSHA
     VERIFICA TESTIGO DE QUE SE HA ENTRADO A SUBRUTINA CHECED
            LDAA #$FF
            STAA $CCFF;LA DIR AQUÍ ESPECIFICADA ES DIRBM+$40 (TESTIGO
CHEC)
            LDY DISPAROS;CARGA Y CON APUNTADOR DE INICIO DE SIGUIENTE
ESPECIFICACIÓN DE DISPARO,(LDY $DIR1).
CHECSEG:    LDAA $00,Y
            CMPA $C2,X
            BNE NOFSEG
            INY
CHECMIN:    LDAA $00,Y
            CMPA $C3,X
            BNE NOFMIN
            INY
CHECHOR:    LDAA $00,Y
            CMPA $C4,X
            BNE NOFHOR
            INY
CHECDM:     LDAA $00,Y
            CMPA $C6,X
            BNE NOFDM
            INY
CHECME:     LDAA $00,Y
            CMPA $C7,X
            BNE NOFME
            INY
CHECAA:     LDAA $00,Y
            CMPA $C8,X
            BNE NOFAA
            INY
VERIFSAL:   BSET $02,X,01;BCLR SI LA SALIDA SE VERIFICA EN BAJO
            BRA SALSUB
NOFSEG:     INY
NOFMIN:     INY
```

```
NOFHOR:     INY
NOFDM:      INY
NOFME:      INY
NOFAA:      INY
DESVERSAL:  BCLR $02,X,01;BSET SI LA SALIDA SE VERIFICA EN BAJO

SALSUB:     STY DISPAROS; (STY $DIR1)
            PULA
            RTS
```

En la tabla 3.17A se muestra la TAB asociada con los CEN TEMPOBC (C = 1, 2, ...6) y en la tabla 3.17B se aprecia el valor que deben tomar, de acuerdo con el tipo de VB que corresponda a la salida de este temporizador, parámetros que aparecen en la tabla 3.17A.

Tabla 3.17A Asignación de bytes asociada con los CEN TEMPOBC (C = 1, 2,....6), empleados para obtener el código requerido por los ML, que realizan temporizadores que generan pulsos, en tiempos de acuerdo al reloj de tiempo real (RTR), del PLM.

Valor numérico (VN) generado por el software de traducción	Byte en CEN genérico para realizar temporizadores que generan pulsos de acuerdo con el estado del RTR, al cual se le asignará el valor VN
VN=29, salida verificada en alto (A= 1) VN=28, salida verificada en bajo (A= 0)	B4
VN=2IT+BUFT	B5
VN=2^JT	B6
VN= Byte alto de DIR1* + 2	B8
VN= byte bajo de DIR1 + 2	B9
VN= byte alto de DIR1	B11
VN= Byte bajo de DIR1	B12
VN=29, salida verificada en alto (A= 1) VN=28, salida verificada en bajo (A= 0)	B29
VN= 2IT + BUFT	B30
VN= 2^JT	B31
VN= Byte alto de DIRBM + 64	B35
VN = Byte bajo de DIRBM + 64	B36
VN = Byte alto de DIRBM + 64	B40
VN = Byte bajo de DIRBM + 64	B41
VN= NT (número de especificaciones de disparo)	B47
VN = 2IT + BUFT	B51
VN = 2^JT	B53
VN = 38, salida verificada en alto, (A=0) VN = 39, salida verificada en bajo, (A=1)	B54

Continuación de Tabla 3.17A	
VN = Byte alto de DIR2**	B62
VN = Byte bajo de DIR2	B63
VN= Byte alto de DIRBM + 64	B69
VN = Byte bajo de DIRBM + 64	B70
VN = Byte alto de DIR1	B73
VN = Byte bajo de DIR1	B74
VN=28, salida verificada en alto (A= 1) VN=29, salida verificada en bajo (A= 0)	B(X1), X1 = 84 + 9(CLASE - 1)
VN = 2IT + BUFT	B(X1+1)
VN = 2^JT	B(X1+2)
VN=29, salida verificada en alto (A= 1) VN=28, salida verificada en bajo (A= 0)	B(X2), X2 = 91 + 11(CLASE – 1)
VN = 2IT + BUFT	B(X2+1)
VN = 2^JT	B(X2+2)
VN = Byte alto de DIR1	B(X2+5)
VN = Byte bajo de DIR1	B(X2+6)

* DIR1 = DIRBM + 150 + 11(CLASE - 1), (La variable DIRBM es generada por el Software de traducción)

** DIR2 = DIR1 +2 + CLASExNT, (CLASE = 1, 2, ...6; véase definición de sintaxis

Tabla 3.17B Valores del parámetro BUFT de la tabla 3.17A

Caracter XT	BUFT
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)

3-6 MÓDULOS AUXILIARES (MA).

Existen módulos para el PLM denominados como auxiliares, que sirven para manejar facilidades que no son propiamente funciones lógicas, para varios de ellos la declaración correspondiente no requiere de operandos, algunos de estos módulos se deberán colocar en el subprograma principal y otros en el temporizado.

Cada uno de los MA tiene desde luego un CEN y una TAB que son empleados por el software de traducción, a continuación se detalla lo concerniente con estos módulos

3-6-1 Descripción del MA con capacidad para generar texto estático y/o dinámico en la UD del PLM.

El PLM puede realizar módulos denominados de tipo *MENSAJERO*, con capacidad para generar texto, predefinido por el usuario, dicho texto podria ser el alusivo a una condición de alarma, o bien cualquier otro tipo de mensaje que el usuario deseara desplegar, en la figura 3.52 se muestra la pantalla de la UD, ahí se aprecia que la misma puede

desplegar dos renglones de dieciséis caracteres cada uno; la posición de cada caracter se especifica por un par de números que denotan en que renglón y columna se encuentra el mismo.

Al texto a desplegar se le denomina mensaje, teniendo este módulo la capacidad para desplegar mensajes fijos y/o mensajes móviles, este último será visible en una ventana delimitada por dos columnas, cuyo número es definido por el usuario y se desplazará entrando por la columna que define el extremo derecho de la ventana, saliendo por el extremo izquierdo de la misma; el usuario puede especificar en la declaración correspondiente, en cual renglón de la UD se desplegarán cada uno de los dos tipos de mensaje aqui mencionados, el tamaño en caracteres del mensaje móvil es definido por el usuario, a cada uno de los mensajeros involucrados en una determinada aplicación se le ha de asignar un número, el cual es limitado a 32 por el software de traducción.

Dado el reducido tamaño de la pantalla de la UD, este módulo está diseñado de modo que en una aplicación que involucre más de un mensajero, sólo esté activo uno a la vez, por defecto el mensajero que es activo al iniciar la ejecución de un programa en el PLM, es el número cero, existen dos módulos con capacidad para cambiar el número de mensajero activo y estos son el módulo MANDESP y el módulo ALARMA, los cuales serán explicados más adelante.

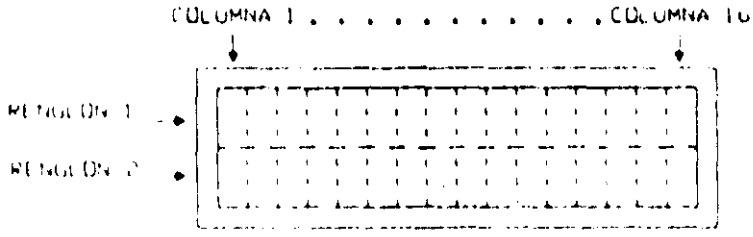


Figura 3.52 Pantalla fisica de la Unidad Desplegadora (UD) del PLM

El módulo mensajero es deshabilitado siempre que el usuario tenga oprimido el botón auxiliar "A" (BAXA), el cual es empleado para habilitar el poder poner a tiempo el RTR mediante los botones auxiliares BAXB y BAXC, presentes en el BCLD del PLM, para más información acerca de como colocar a tiempo el RTR puede consultarse en este trabajo el módulo que lo maneja que es denominado como RTRX

Este módulo debe declararse como parte del subprograma temporizado, la sintaxis para declararlo es la siguiente:

MENSAJERO#N "MENSAJE FIJO", CI, TV, P, ABCD;
Renglón de datos uno
Renglón de datos dos
.
.
.
.
último renglón de datos

Donde:
N denota el número de mensajero, esto definido por el usuario, el valor máximo permitido es 32.

MENSAJE FIJO es una cadena de un máximo de dieciséis caracteres, que es colocada a partir de la columna uno del renglón que el usuario especifique, nótese el empleo de comillas para delimitar el texto del mensaje fijo, si el usuario deseara que no haya mensaje fijo, denotaría esto con dos comillas seguidas.

CI denota el número de columna, que delimita el extremo izquierdo de la ventana donde se desplegará el mensaje móvil.

TV representa el tamaño en columnas del mensaje móvil.

P es un número comprendido entre 1 y 255, que representa el intervalo en centésimas de segundo entre dos posiciones subsecuentes del mensaje móvil.

A es un dígito binario, que habrá de ser uno, si se desea que el mensaje fijo se despliegue en el renglón uno, en otro caso "A" deberá ser cero.

B es un dígito binario, que deberá ser uno, si se desea que el mensaje móvil se despliegue en el renglón uno, en otro caso "B" deberá ser cero.

C es un dígito binario, que deberá ser cero, si se desea que el mensajero opere sólo si el mismo es el módulo desplegador activo, si "C" es uno, el mensajero opera independientemente del hecho de que el mismo sea el módulo desplegador activo, desde luego que el usuario sería responsable de que en el último caso no se produjera una colisión entre dos mensajeros, que intentarán escribir texto en una misma zona de la pantalla de la UD

D es un dígito binario, que deberá ser cero, si se desea que únicamente se despliegue el mensaje fijo, si "D" se pone en uno la operación será normal, desplegándose tanto el mensaje fijo como el móvil.

En los renglones de datos habrá de colocarse el texto del mensaje móvil, al hacer esto el usuario es libre de colocar en cada renglón el número de caracteres que le acomode, cada renglón de datos, excepto el último, deberá iniciar con el caracter "#" en la primera columna seguido por un espacio, el último renglón de datos deberá iniciar con dos caracteres "#" en la primera y segunda columna seguidos por un espacio, a diferencia de los otros módulos que involucran declaraciones de datos, al final de cada renglón de datos no deberá colocarse el caracter ";".

En caso de que un renglón de datos terminara con una palabra completa, deberá colocarse el caracter "|" como último caracter del mismo, de no hacerse esto, en el mensaje móvil aparecerán, sin espaciamiento entre ellas, la última palabra del renglón en cuestión y la primera del renglón subsecuente, por cada caracter "|" colocado al final de un renglón de datos aparecerá, en el mensaje móvil, un espacio entre el último caracter de un renglón de datos y el primero del siguiente, los caracteres espacio que queden en alguna posición intermedia, en los renglones que declaran el contenido del texto móvil, se colocan de manera normal.

El siguiente ejemplo ilustra como declarar módulos mensajeros en SIIL1.

Ejemplo 3.15

Supóngase que como parte de una aplicación del PLM se desea realizar un módulo de tipo mensajero, el cual deberá activarse al iniciar la ejecución en el PLM del programa escrito para el caso, debiendo ser el texto fijo el siguiente "PROG1", el texto móvil debe ser "Se inició exitosamente la ejecución de PROG1 en el PLM, mientras aparezca este mensaje, no se ha generado ninguna condición de alarma"

Tanto el texto fijo como el móvil deberán aparecer en el renglón uno, la ventana para el mensaje móvil deberá estar comprendida entre las columnas 7 y 16 (TV = 10), siendo 25 centésimas de segundo el intervalo deseado entre posiciones subsecuentes del mensaje

móvil; debiendo el mensajero operar sólo cuando el número que se le asigne coincida con el del módulo desplegador activo (C = 0).

Dado que se requiere que el mensajero inicie su operación al ejecutarse el programa de la aplicación, se le deberá asignar el número cero; por lo tanto, la declaración de este mensajero podría ser:

MENSAJERO#0 "PROG1", 7, 10, 25, 1101;

Se inició exitosamente la ejecución de PROG1|

en el PLM, mientras aparezca este mensaje, no se ha generado ninguna condición de|

alarma

En la figura 3.53 se muestra el aspecto que tendría la pantalla de la UD, 150 centésimas de segundo después de iniciarse la ejecución del programa que contenga el mensajero del ejemplo 3.15.

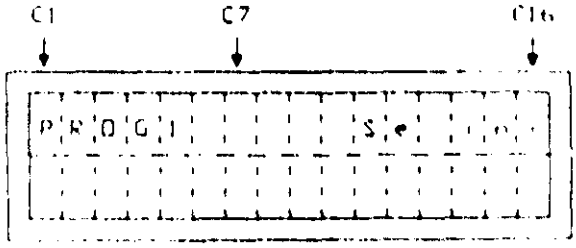


Figura 3.53 Pantalla de la UD, 150 centésimas de segundo después de iniciarse la ejecución del programa que contenga el mensajero del ejemplo 3.15.

Descripción del CEN asociado con módulos mensajeros (MENSAJE1)

El flujo de ejecución de este ML se muestra en la figura 3.54, el tiempo máximo de ejecución en microsegundos (Tmns) para este módulo, suponiendo que fe = 2MHz, está dado por la siguiente ecuación:

$$Tmns = 87.5 + 10.5xTV \quad \ldots\ldots(3.4)$$

Dado que este módulo requiere de un buffer de datos con los códigos ASCII de los caracteres que conforman el mensaje móvil, el código objeto correspondiente al mismo lo deberá contener, en la figura 3.55 se muestra la estructura que en la memoria de la CC del PLM tendría el tramo de código correspondiente a un módulo de tipo mensajero, notese que

el buffer de datos correspondiente está colocado inmediatamente después del código ejecutable correspondiente.

Es importante señalar aquí, el hecho de que el código objeto asociado con este módulo, coloca los códigos ASCII asociados con los caracteres a desplegar en la UD en un buffer en RAM denominado BD, el cual está comprendido de la dirección 01D0 a la 01EF, correspondiendo el dato existente en la dirección 01D0 con el código ASCII del carácter a desplegarse en la columna uno del renglón superior de la UD, el código ASCII correspondiente con el carácter a desplegarse en la segunda columna de la UD será el dato contenido en la dirección 01D1 y así sucesivamente, debiendo el código ASCII del carácter a desplegarse en la columna dieciséis del segundo renglón de la UD estar almacenado en la dirección 01EF; el BD es empleado no solo por los módulos de tipo mensajero, sino también por módulos cuya acción es escribir algo en la UD, tales como los módulos observadores del estado de contadores de eventos y el módulo auxiliar que maneja el reloj de tiempo real (RTR), los cuales serán tratados más adelante en este trabajo.

Para copiar el contenido del BD en la UD se emplea un módulo auxiliar denominado DESP que debe colocarse en el subprograma principal, tratándose lo concerniente con el mismo más adelante.

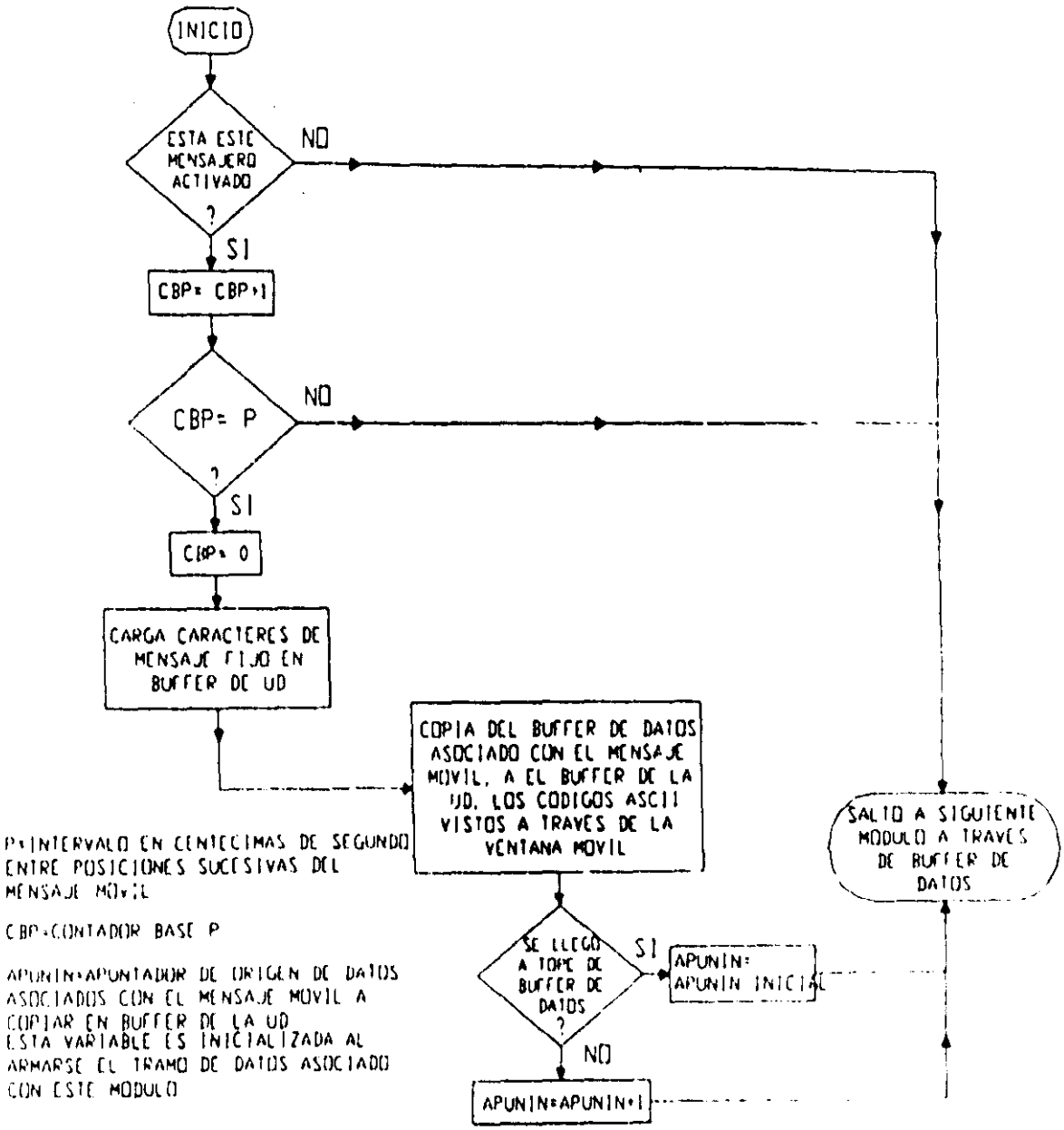


Figura 3.54 Flujo de ejecución del módulo mensajero realizable por el PLM (MENSAJERO1).

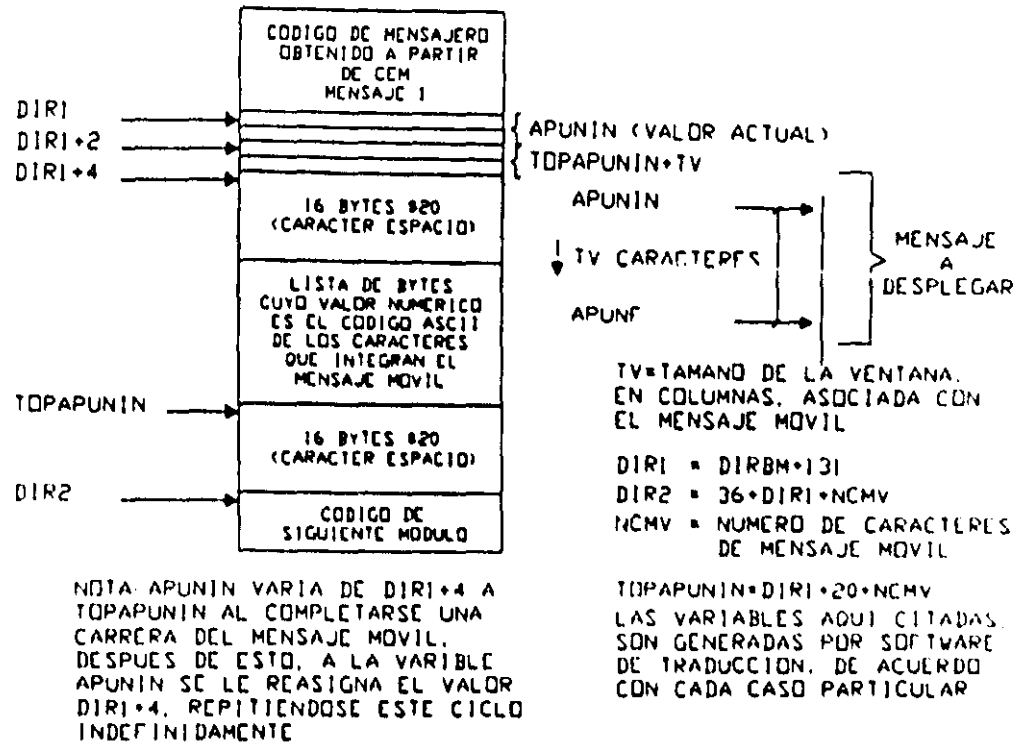


Figura 3.55 Estructura del código ejecutable correspondiente a un módulo mensajero.

El CEN asociado con este módulo se denomina como MENSAJE1 y es mostrado a continuación:

CEN DEL MÓDULO DESPLEGADOR DE TEXTO EN LA UD (MENSAJE1).

```
0000 A6FE        LDAA $FE,X;CHECA SI ESTE MENSAJE ESTA
                            ACTIVADO
0002 8104        CMPA #$04;COMPARA CON NUMERO ASOCIADO
                           CON ESTE MENSAJERO
0004 267A        BNE FINRS;ENRUTA HACIA FINRS (SALIDA)
0006 6CCB        INC $CB,X
0008 A6CB        LDAA $CB,X
000A 8164        CMPA #$64
000C 2672        BNE FINRS
000E 6FCB        CLR $CB,X
0010 8620        LDAA #$20
0012 A7D0        STAA $D0,X;ALMACENA PRIMER CARACTER DE
                            MENSAJE FIJO EN RENGLON 1
0014 8620        LDAA #$20
0016 A7D1        STAA $D1,X;ALMACENA SEGUNDO CARACTER DE
                            MENSAJE FIJO EN RENGLON 1
0018 8620        LDAA #$20
001A A7D2        STAA $D2,X;ALMACENA TERCER CARACTER DE
                            MENSAJE FIJO EN RENGLON 1
001C 8620        LDAA #$20
001E A7D3        STAA $D3,X;ALMACENA CUARTO CARACTER DE
                            MENSAJE FIJO EN RENGLON 1
0020 8620        LDAA #$20
0022 A7D4        STAA $D4,X;ALMACENA QUINTO CARACTER DE
```

```
                                    MENSAJE FIJO EN RENGLON 1
0024 8620                   LDAA #$20
0026 A7D5                   STAA $D5,X;ALMACENA SEXTO CARACTER DE
                                    MENSAJE FIJO EN RENGLON 1
0028 8620                   LDAA #$20
002A A7D6                   STAA $D6,X;ALMACENA SEPTIMO CARACTER DE
                                    MENSAJE FIJO EN RENGLON 1
002C 8620                   LDAA #$20
002E A7D7                   STAA $D7,X;ALMACENA OCTAVO CARACTER DE
                                    MENSAJE FIJO EN RENGLON 1
0030 8620                   LDAA #$20
0032 A7D8                   STAA $D8,X;ALMACENA NOVENO CARACTER DE
                                    MENSAJE FIJO EN RENGLON 1
0034 8620                   LDAA #$20
0036 A7D9                   STAA $D9,X;ALMACENA DECIMO CARACTER DE
                                    MENSAJE FIJO EN RENGLON 1
0038 8620                   LDAA #$20
003A A7DA                   STAA $DA,X;ALMACENA ONCEAVO CARACTER DE
                                    MENSAJE FIJO EN RENGLON 1
003C 8620                   LDAA #$20
003E A7DB                   STAA $DB,X;ALMACENA DOCEAVO CARACTER DE
                                    MENSAJE FIJO EN RENGLON 1
0040 8620                   LDAA #$20
0042 A7DC                   STAA $DC,X;ALMACENA TRECEAVO CARACTER DE
                                    MENSAJE FIJO EN RENGLON 1
0044 8620                   LDAA #$20
0046 A7DD                   STAA $DD,X;ALMACENA CATORCEAVO CARACTER
                                    DE MENSAJE FIJO EN RENGLON 1
0048 8620                   LDAA #$20
004A A7DE                   STAA $DE,X;ALMACENA QUINCEAVO CARACTER
                                    DE MENSAJE FIJO EN RENGLON 1
004C 8620                   LDAA #$20
004E A7DF                   STAA $DF,X;ALMACENA DIECISEISAVO
                                    CARACTER DE MENSAJE FIJO EN
                                    RENGLON 1

0050 C610          INICIOP: LDAB #$10
0052 18FECAAA               LDY $CAAA;CARGA Y CON APUNTADOR APUNIN
                                    PRESENTE (CONTENIDO EN DIR1)
0056 3C                     PSHX
0057 CE01E0                 LDX #$01E0
005A 18A600        QK:      LDAA $00,Y
005D A700                   STAA $00,X
005F 08                     INX
0060 1808                   INY
0062 5A                     DECB
0063 26F5                   BNE QK
0065 38                     PULX
0066 18BCCBBB               CPY $CBBB;COMPARA "Y" CON TOPAPUNIN+TV
                                    (CONTENIDO DE DIR1+2)
006A 260A                   BNE INCAP
006C 18CEC111               LDY #$C111;CARGA APUNTADOR INICIAL DE
                                    MENSAJERO (DIR1+4)
0070 18FFCA11               STY $CA11;ALMACENA APUNTADOR INICIAL EN
                                    DIR1
0074 200A                   BRA FINRS
0076 18FECAAA      INCAP:   LDY $CAAA;CARGA Y CON APUNTADOR APUNIN
                                    PRESENTE (DIRECCION DIR1)
007A 1808                   INY
007C 18FFCAAA               STY $CAAA;ALMACENA EN DIR1
0080 7EDDDD        FINRS:   JMP $DDDD; SALTA A DIR2
```

En la tabla 3.18 se muestra la TAB asociada con los CEN MENSAJE1.

Tabla 3.18 Asignación de bytes asociada con el CEN MENSAJE1, empleado para obtener el código requerido por los ML, que despliegan mensajes en la UD del PLM.

Valor numérico (VN) generado por el software de traducción	Byte en CEN genérico para realizar módulos de tipo mensajero, al cual se le asignará el valor VN
VN= 166, si C = 0 VN= 32, si C = 1	B0
VN= 254, si C = 0 VN= 4, si C = 1	B1
VN= N (número de mensajero)	B3
VN= 128 + N	B7
VN= 128 + N	B9
VN= P (permanencia en centésimas de segundo)	B11
VN= 128 + N	B15
VN= código ASCII del i-ésimo caracter del mensaje fijo	BX1, X1 = 17 + 4(i-1) i = 1, 2, 3, ...16
VN= 207 + i, si A = 1 VN= 223 + i si A = 0	BX2, X2 = 19 + 4(i-1) i = 1, 2, 3, ...16
VN= 198, si D = 1 VN= 32, si D = 0	B80
VN= TV, (número de caracteres de ventana)	B81
VN= Byte alto de DIR1	B84
VN= Byte bajo de DIR1	B85
VN= 223 + CI, (mensaje móvil en renglón uno de UD), B= 0 VN= 207 + CI, (Mensaje móvil en renglón dos de UD), B= 1	B89
VN= Byte alto de DIR1* + 2	B104
VN= Byte bajo de DIR1 + 2	B105
VN= Byte alto de DIR1 + 4	B110
VN= Byte bajo de DIR1 + 4	B111
VN= Byte alto de DIR1	B114
VN= Byte bajo de DIR1	B115
VN= Byte alto de DIR1	B120
VN= Byte bajo de DIR1	B121
VN= Byte alto de DIR1	B126
VN= Byte bajo de DIR1	B127
VN= Byte alto de DIR2**	B129
VN= Byte bajo de DIR2	B130

* DIR1 = DIRBM + 131, (La variable DIRBM es generada por el Software de traducción)

** DIR2 = DIR1 +36 + número de caracteres de cuerpo del mensaje móvil

3-6-2 Descripción del MA con capacidad para generar mensajes de alarma en la UD del PLM.

El PLM puede realizar módulos denominados de tipo *ALARMA*, con capacidad para generar mensajes en la UD, este módulo cuenta con una sola entrada, que al verificarse, hace que el texto asociado con un determinado módulo mensajero sea desplegado en la UD. En la figura 3.56 se muestra la representación como bloque de este tipo de módulo.

Cada módulo de alarma tendrá asociado el número de módulo mensajero, que contiene el texto alusivo al la misma, el orden de colocación de la declaración correspondiente en el programa fuente será el orden de prioridad del mensaje de alarma asociado; de esta manera, el primer módulo de alarma colocado tendrá máxima prioridad, mientras que el último tendrá la prioridad mínima, esto quiere decir que si se verifica más de una entrada de alarma en sendos módulos de este tipo, se desplegará en la UD únicamente el mensaje asociado con el módulo de mayor prioridad, cuando la entrada de alarma asociada con este último módulo se desverifica se despliega el mensaje asociado con el módulo de alarma de mayor prioridad cuya entrada permanezca verificada.

Cuando no están verificadas las entradas asociadas, con cada uno de los módulos de alarma declarados en un programa en SIIL1, el texto que se despliega es el correspondiente con el módulo mensajero número cero; así, el texto que indique que no se ha dado ninguna condición de alarma, habrá de ser el asociado con un módulo mensajero, al que se deberá asignar el número cero; de este modo, al diseñar un sistema de mensajes de alarma apoyándose en los módulos mensajeros y de alarma, siempre deberá existir un módulo mensajero con el número cero.

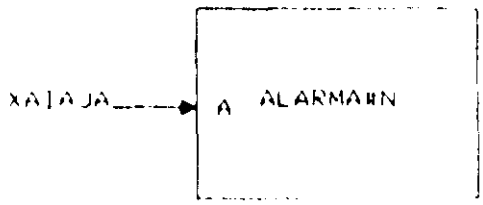


Figura 3.56 Representación como bloque, del módulo activador de mensajes de alarma

Este módulo debe declararse como parte del subprograma principal, la sintaxis para declararlo es la siguiente:

ALARMA#NMG XAJAIA, A;

Donde:
NMG denota el número de módulo de tipo alarma, que debe coincidir con el número de módulo mensajero asociado con el mensaje a desplegar, cuando se verifique la variable definida como entrada al módulo de alarma.
XA podrá ser la letra "e", "s" o "i" mayúscula o minúscula, dependiendo esto de que la variable de entrada de activación de mensaje de alarma (A), sea una VBE, VBS o VBI.
IA denota el número de grupo que corresponda a la VB declarada como entrada "A".
JA denota el número de bit dentro del grupo IA, asociado a la variable de entrada "A".
A es un dígito binario, que habrá de ser uno, si se desea que el nivel de verificación de la entrada sea alto, en otro caso "A" deberá ser cero.

El siguiente ejemplo ilustra como programar un sistema de mensajes de alarma en SIIL1, empleando para ello a módulos mensajeros y de alarma.

Ejemplo 3.16

Supóngase que como parte de una aplicación del PLM, se desea realizar un sistema de mensajes de alarma con tres mensajes testigo, todos ellos móviles sin mensaje fijo y desplegados en el primer renglón de la UD a partir de la columna uno, con un tamaño de ventana de 10 caracteres, de modo que el mensaje de mayor prioridad tenga el siguiente texto: "El motor del compresor principal se ha detenido, si no se reinicia su operación, habrá que parar la planta por 24 horas"; el texto del siguiente mensaje en orden de prioridad deberá ser: "La temperatura del interior de la autoclave 4 no es la adecuada, checar el controlador asociado."; finalmente, el texto del mensaje de mínima prioridad ha de ser "La pintura en el dosificador A, se ha agotado, por lo tanto, el suministro de pintura se ha conmutado al dosificador B ", se requiere que el mensaje la condición de no alarma sea: "Operación normal, no se ha dado ninguna condición de alarma", debiendo el mismo ser móvil, desplegado al igual que los mensajes de alarma, en el renglón uno a partir de la

columna uno con un tamaño de ventana de 10 caracteres; se desea que el intervalo entre posiciones subsecuentes de los mensajes sea de 30 centésimas de segundo.

Las VB que testificarían las condiciones de alarma serían en orden de prioridad las siguientes: E23, E12, e I12, siendo las dos primeras verificadas en alto y la última en bajo, a los módulos mensajeros se les deberá asignar, de acuerdo al orden de prioridad los números 3, 5, y 7.

Este ejemplo implica a varios módulos, cuatro mensajeros y tres activadores de mensajes de alarma, de acuerdo a lo explicado en párrafos anteriores los módulos activadores de alarma de este ejemplo se deben colocar en el subprograma principal mientras que los módulos mensajeros deben ser parte del subprograma temporizado a continuación se muestra una posible forma de hacer las declaraciones correspondientes:

Declaraciones de los módulos de tipo de alarma, (deben estar en el subprograma principal).

ALARMA#3 E23, 1; Prioridad 1

ALARMA#5 E12, 1; Prioridad 2

ALARMA#7 I12, 0; Prioridad 3

Declaraciones de los módulos mensajeros, (deben estar en el subprograma temporizado).

MENSAJERO#5 "", 1, 10, 30, 1001; Mensaje asociado con la señal de alarma E12.

La temperatura del interior de la autoclave 4 no es la adecuada, |

checar el controlador asociado.

MENSAJERO#3 "", 1, 10, 30, 1001; Mensaje asociado con la señal de alarma E23.

El motor del compresor principal se ha detenido, si no se reinicia su operación, |

habrá que parar la planta por 24 horas

MENSAJERO#7 "", 1, 10, 30, 1001; Mensaje asociado con la señal de alarma I12.

La pintura en el dosificador A, se ha agotado; por lo tanto, el suministro de pintura |

se ha conmutado al dosificador B.

MENSAJERO#0 "", 1, 10, 30, 1001; Mensaje testigo de no condición de alarma.

Operación normal, no se ha dado ninguna condición de alarma

Nótese que las declaraciones de los módulos mensajeros, pueden ser hechas no necesariamente en el que corresponda, a la prioridad de los mensajes que portan.

Para apreciar como sería un programa completo en SIIL1, que contuviera un sistema de mensajes de alarma, se puede ver el ejemplo 3.18 al final de este capítulo.

Descripción del CEN ALARMA, asociado con módulos activadores de mensajes de alarma.

El flujo de ejecución de este ML se muestra en la figura 3.57, el tiempo máximo de ejecución en microsegundos (Tmns) para el mismo es 26 µs, suponiendo que fe = 2MHz.

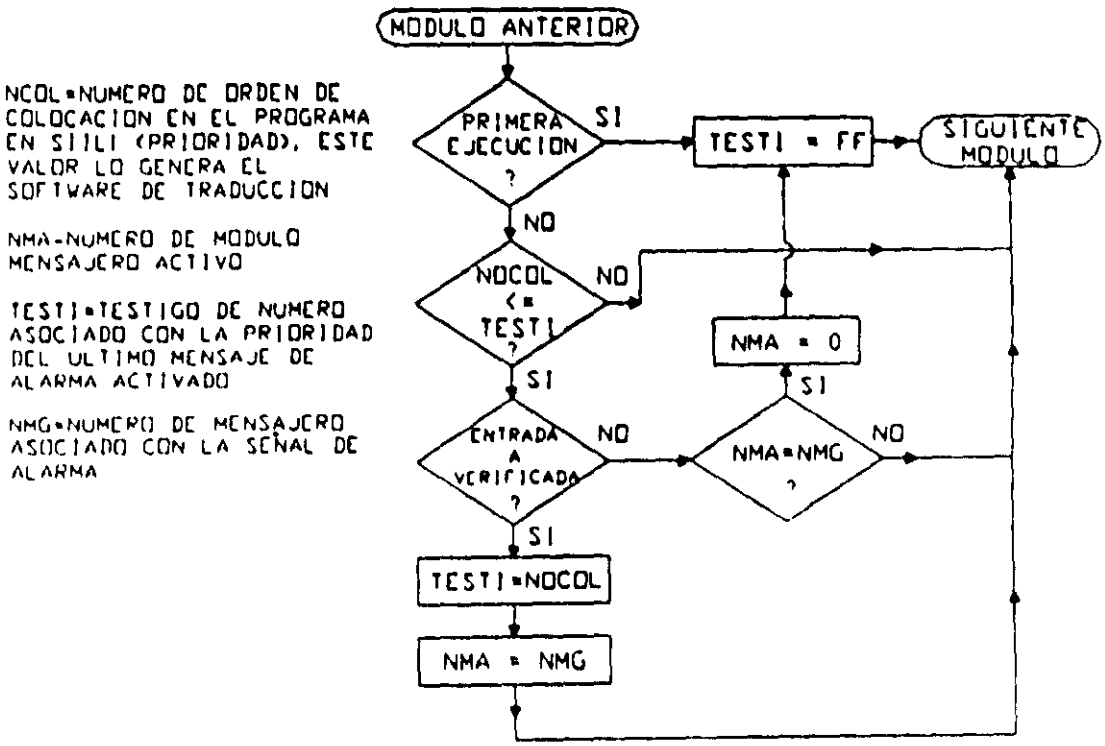


Figura 3.57 Flujo de ejecución del módulo activador de mensajes de alarma.

El CEN asociado con este módulo se denomina como ALARMA y es mostrado a continuación:

CEN DEL MÓDULO DISPARADOR DE MENSAJES DE ALARMA.

```
0000 A6F2                          LDAA $F2,X
0002 2706                          BEQ NOPP
          ACCIONAMIENTO EN PRIMERA EJECUCIÓN
0004 86FF              PRIOMAX:    LDAA #$FF
0006 A7BF                          STAA $BF,X;PONE TESTIGO DE PRIORIDAD DE
                                              "ÚLTIMA ALARMA VERIFICADA" EN
                                               255 (TESTI)
0008 201D                          BRA SALIDA
000A 860A              NOPP:       LDAA #$0A;CARGA A CON NÚMERO DE ORDEN DE
                                             COLOCACION (NOCOL)
```

```
000C A1BF                          CMPA $BF,X;COMPARA CON TESTI (X+BF)
000E 2217                          BHI SALIDA
      PASA A VER SI ENTRADA DECLARADA ESTÁ VERIFICADA
0010 A600                          LDAA $00,X
0012 8401                          ANDA #$01
0014 260A                          BNE ENTVERI;BEQ PARA ENTRADA VERIFICADA
                                               EN BAJO
0016 8602              ENTNOVERI:  LDAA #$02;CARGA A CON NÚMERO DE MENSAJE
                                               ASOCIADO CON ESTA ALARMA (NMG)
0018 A1FE                          CMPA $FE,X;COMPARA CON NÚMERO DE MENSAJE
                                               ACTUAL EN EL DESPLEGADO
001A 260B                          BNE SALIDA
001C 6FFE                          CLR $FE,X;DESACTIVA MENSAJE ACTUAL
001E 20E4                          BRA PRIOMAX
0020 CC0A02            ENTVERI:    LDD #$0A02;CARGA A CON NÚMERO DE ORDEN
                                               DE COLOCACIÓN (NOCOL) Y B CON
                                               NÚMERO DE MENSAJE ASOCIADO
                                               CON ESTA ALARMA
0023 A7BF                          STAA $BF,X;CARGA TESTI CON PRIORIDAD DE
                                               ESTÁ ALARMA (NOCOL)
0025 E7FE                          STAB $FE,X;ACTIVA MENSAJE ASOCIADO CON
                                               ESTA ALARMA
                       SALIDA:
```

En la tabla 3.19A se muestra la TAB asociada con los CEN ALARMA.

Tabla 3.19A Asignación de bytes asociada con el CEN ALARMA, empleado para obtener el código requerido por los ML, que activan mensajes de alarma en la UD del PLM.

Valor numérico (VN) generado por el software de traducción	Byte en CEN genérico para realizar módulos de tipo alarma, al cual se le asignará el valor VN
VN= NOCOL*	B11, B33
VN= 2IA + BUFA	B17
VN= 2^JA	B19
VN= 38, entrada se verifica en alto, A=1 VN= 39, entrada se verifica en bajo, A=0	B20
VN= NMG**	B23, B34

* NOCOL = número de orden de colocación del módulo de alarma en el programa fuente en SIIL1 (prioridad).

** NMG = número de módulo mensajero asociado con la alarma.

Tabla 3.19B Valores del parámetro BUFA, de la tabla 3.19A

Caracter XA	BUFA
Letra "e" mayúscula o minúscula	BUFEN (definido en tabla 3.1B)
Letra "s" mayúscula o minúscula	BUFSAL (definido en tabla 3.1B)
Letra "i" mayúscula o minúscula	BUFI (definido en tabla 3.1B)

3-6-3 Descripción del módulo observador del estado de contadores de eventos.

El PLM puede realizar módulos que hacen que el estado de un determinado contador de eventos sea desplegado en la UD, definiendo el usuario el número de contador, el renglón y la columna donde se mostrará la cuenta, así como también el número de dígitos a emplear; el software de traducción limita a ochenta el número de contadores de eventos permisibles.

Este módulo debe declararse como parte del subprograma principal, la sintaxis para declararlo es la siguiente:

OBSCE#N CI, ND, R;

Donde:
N denota el número de módulo observador de contador de eventos, que a su vez es igual al que corresponda, con el contador cuyo estado se desea desplegar.
CI denota el número de columna en la UD, a partir de la cual se mostrará la cuenta.
ND especifica el número de dígitos significativos a emplear.
R es un número que habrá de ser uno si se desea que la cuenta sea desplegada en el renglón superior de la UD, en otro caso R deberá ser dos.

El siguiente ejemplo ilustra como declarar un módulo observador de contador de eventos.

Ejemplo 3.17

Supóngase que como parte de una aplicación del PLM, se desea desplegar el estado de un contador de eventos cuyo número de asignación es tres, requiriéndose que la cuenta sea mostrada en el renglón inferior de la UD a partir de la quinta columna, mostrándose sólo los tres dígitos menos significativos de la cuenta.

La declaración correspondiente es:

OBSCE#3 5, 3,2;

Descripción de los CEN asociados con módulos observadores de contadores de eventos.

El flujo de ejecución de este ML, cuando se desea desplegar cinco dígitos de la cuenta a observar, se muestra en la figura 3.58, el tiempo máximo de ejecución en microsegundos (Tmoce) asociado varía con el número de dígitos significativos a mostrar y con el hecho de que la cuenta sea menor que 10000 o no; así, si la cuenta es menor que 10000 y suponiendo que que fe = 2MHz, Tmoce está dado por la siguiente ecuación:

$$Tmce = Z(X) \qquad (3.4)$$

Donde:

X representa el número de dígitos significativos a emplear, (X = 1, 2, 3, 4, 5).

Z = 95.5 si X = 1

Z = 98.5 si X = 2

Z = 136 si X = 3

Z = 136 si X = 4

Z = 176 si X = 5

En caso de que la cuenta sea mayor que 9999 para fe = 2MHz los correspondientes tiempos máximos de ejecución son:

$$Tmce = Q(X) + 9.5DMSC \qquad (3.5)$$

Donde:

X representa el número de dígitos significativos a emplear, (X = 1, 2, 3, 4, 5).

Q = 91.5 si X = 1

Q = 94.5 si X = 2

Q = 132 si X = 3

Q = 132 si X = 4

Q = 172 si X = 5

DMSC es un número comprendido entre uno y seis, que representa el valor del dígito más significativo de la cuenta, recuérdese que la cuenta máxima que un contador del PLM puede manejar es 65535.

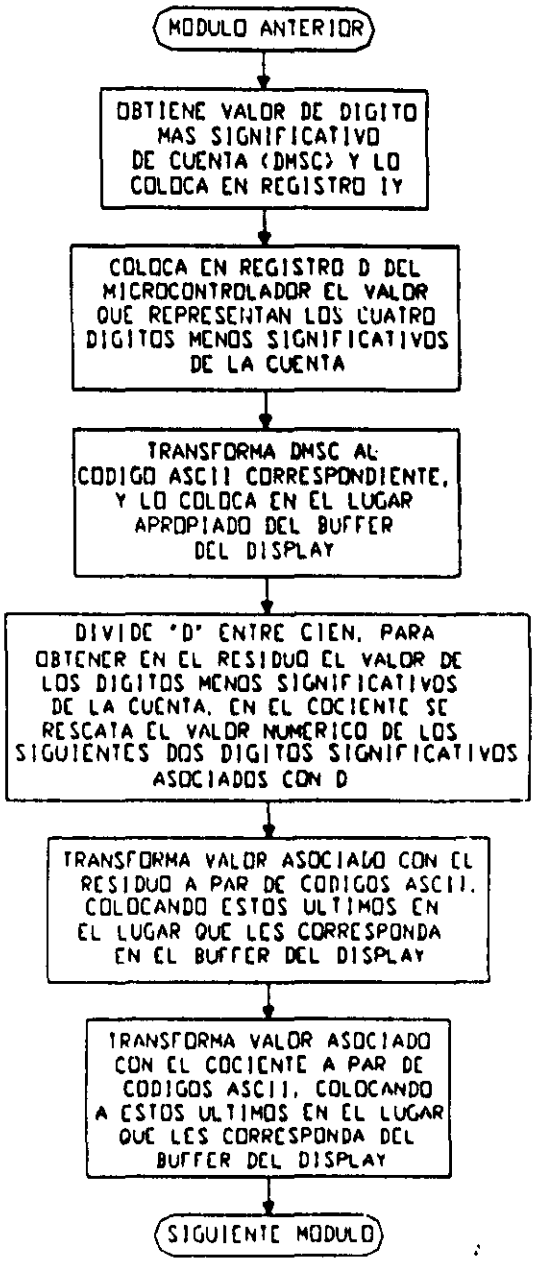


Figura 3.58 Flujo de ejecución del módulo activador de mensajes de alarma.

Dependiendo del número de dígitos significativos a emplear, este módulo tiene asociado cinco diferentes CEN que si bien en lo general presentan la misma estructura, en cada caso particular tienen variantes en la parte del código que concierne con la escritura de los dígitos al buffer de la UD, estos CEN se denominan OBSCEDX, donde *X* representa el número de dígitos significativos a emplear, el software de traducción toma el CEN adecuado dependiendo esto del valor del parámetro ND que el usuario haya indicado en la declaración

correspondiente, a continuación se muestra el detalle del CEN asociado con este módulo cuando se requiere desplegar cinco dígitos para un determinado contador.

CEN DEL MÓDULO DESPLEGADOR DE ESTADO DE CONTADOR DE EVENTOS. CON CINCO DÍGITOS (OBSCED5).

```
0000 2064              BRA INDESCE
0002 00010203    TABCD: FCB 00,01,02,03,04,05,06,07,08,09,10,11,12,
     04050607                13,14,15
     08091011
     12131415
0012 16171819          FCB 16,17,18,19,20,21,22,23,24,25,26,27,
     20212223              28,29,30
     24252627
     282930
0021 31323334          FCB 31,32,33,34,35,36,37,38,39,40,41,42,
     35363738              43,44,45
     39404142
     434445
0030 46474849          FCB 46,47,48,49,50,51,52,53,54,55,56,57,
     50515253              58,59,60
     54555657
     585960
003F 61626364          FCB 61,62,63,64,65,66,67,68,69,70,71,72,
     65666768              73,74,75
     69707172
     737475
004E 76777879          FCB 76,77,78,79,80,81,82,83,84,85,86,87,
     80818283              88,89,90
     84858687
     888990
005D 91929394          FCB 91,92,93,94,95,96,97,98,99
     95969798
     99
0066 18CE0000    INDESCE:  LDY #$0000
006A FC0200                LDD $0200;CARGA D CON DATO DE CUENTA
                                CONTENIDO EN DIRECCIaN BASE
                                DE CONTADOR (DIRCON)
006D 1A832710    CPD:      CPD #$2710;COMPARA CON 10000
0071 2507                  BLO NOMA
0073 1808                  INY
0075 832710                SUBD #$2710
0078 20F3                  BRA CPD
LLEGA A ETIQUETA NOMA CON DIGITO MAS SIGNIFICATIVO EN "IY" Y RESTO EN D.
007A 8D09        NOMA:     BSR DESPLE
007C 188F                  XGDY
007E 17                    TBA
007F 8D22                  BSR CONASCII
0081 E7D0                  STAB $D0,X;ESCRIBE DIGITO MAS
                                SIGNIFICATIVOVO EN DISPLAY
0083 203F                  BRA SALABB
SUBRUTINA DESPLE, QUE ESCRIBE EN BUFFER DE DISPLAY, EL VALOR DE RESTO DE
 CUENTA CONTENIDA EN D<10000d
0085 183C        DESPLE:   PSHY
0087 3C                    PSHX
0088 CE0064                LDX #$0064
008B 02                    IDIV
008C 3C                    PSHX
008D 1838                  PULY;COCIENTE EN IY
```

```
008F 38                          PULX;X EN SU VALOR ($0100)
0090 17                          TBA
0091 8D10                        BSR CONASCII
0093 A7D3                        STAA $D3,X
0095 E7D4                        STAB $D4,X;ESCRIBE PAR DE DIGITOS MENOS
                                          SIGNIFICATIVOS
0097 188F                        XGDY;PASA COCIENTE A D
0099 17                          TBA;PASA COCIENTE AL ACC A.
009A 8D07                        BSR CONASCII
009C A7D1                        STAA $D1,X
009E E7D2                        STAB $D2,X;ESCRIBE PAR DE DIGITOS MAS
                                          SIGNIFICATIVOS
00A0 1838                        PULY
00A2 39                          RTS
SUBRUTINA CONASCII, DE CONVERSIÓN A PAR DE ASCII'S A PARTIR DE BYTE
HEXADECIMAL,SE ENTRA EL BYTE HEX EN EL ACUMULADOR A Y SE RETORNA CON EL
PAR DE ASCCI'S QUE REPRESENTAN AL NÚMERO (N<=99), EN EL ACUMULADOR D
00A3 183C              CONASCII: PSHY
00A5 A7CB                        STAA $CB,X
00A7 6FCA                        CLR $CA,X
00A9 CC0002                      LDD #TABCD
00AC E3CA                        ADDD $CA,X
00AE 188F                        XGDY
00B0 18A600                      LDAA $00,Y
00B3 16                          TAB
00B4 C40F                        ANDB #$0F
00B6 CB30                        ADDB #$30
00B8 84F0                        ANDA #$F0
00BA 0C                          CLC
00BB 46                          RORA
00BC 46                          RORA
00BD 46                          RORA
00BE 46                          RORA
00BF 8B30                        ADDA #$30
00C1 1838                        PULY
00C3 39                          RTS
                       SALABB:
```

Los CEN a emplearse cuando el número de dígitos a emplear es diferente de cinco se muestran a continuación:

CEN OBSCED1, asociado con el módulo observador de contadores de eventos, cuando de requiere desplegar sólo el dígito menos significativo de la cuenta.

```
            BRA INDESCE
;
TABCD:FCB $00,$01,$02,$03,$04,$05,$06,$07,$08,$09,$10,$11,$12,$13,$14,$15
            FCB 16,$17,$18,$19,$20,$21,$22,$23,$24,$25,$26,$27,$28,$29,$30
            FCB 31,$32,$33,$34,$35,$36,$37,$38,$39,$40,$41,$42,$43,$44,$45
            FCB 46,$47,$48,$49,$50,$51,$52,$53,$54,$55,$56,$57,$58,$59,$60
            FCB 61,$62,$63,$64,$65,$66,$67,$68,$69,$70,$71,$72,$73,$74,$75
            FCB 76,$77,$78,$79,$80,$81,$82,$83,$84,$85,$86,$87,$88,$89,$90
            FCB $91,$92,$93,$94,$95,$96,$97,$98,$99

INDESCE:     LDY #$0000
             LDD $0200;CARGA D CON DATO DE CUENTA CONTENIDO EN DIRECCIÓN
```

```
                    BASE DE CONTADOR (DIRCON)
CPD:        CPD #$2710;COMPARA CON 10000
            BLO NOMA
            INY
            SUBD #$2710
            BRA CPD

LLEGA A ETIQUETA NOMA CON DIGITO MAS SIGNIFICATIVO EN "Y" Y RESTO EN D.
NOMA:       BSR DESPLE
            BRA SALABB
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP

SUBRUTINA DESPLE, QUE ESCRIBE EN BUFFER DE DISPLAY EL VALOR DE RESTO DE
CUENTA CONTENIDA EN D<10000d
DESPLE:     PSHY
            PSHX
            LDX #$0064
            IDIV
            PSHX
            PULY;COCIENTE EN Y
            PULX;X EN SU VALOR ($0100)
            TBA
            BSR CONASCII
            NOP
            NOP
            STAB $D0,X;ESCRIBE DIGITO MENOS SIGNIFICATIVO
            BRA SALSUB
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
SALSUB:     PULY
            RTS

SUBRUTINA DE CONVERSIÓN A PAR DE ASCII'S A PARTIR DE BYTE HEXADECIMAL, SE
ENTRA CON EL BYTE HEX EN EL ACUMULADOR A Y SE RETORNA CON EL PAR DE
ASCII'S QUE REPRESENTAN AL NÚMERO (N<=99) EN EL ACUMULADOR D.
CONASCII:   PSHY
            STAA $CB,X
            CLR $CA,X
            LDD #TABCD
            ADDD $CA,X
            XGDY
            LDAA $00,Y
            TAB
            ANDB #$0F
            ADDB #$30
            ANDA #$F0
            CLC
            RORA
            RORA
```

```
            RORA
            RORA
            ADDA #$30
            PULY
            RTS
SALABB:
```

CEN OBSCED2, asociado con el módulo observador de contadores de eventos, cuando de requiere desplegar sólo los dos dígitos menos significativos de la cuenta.

```
            BRA INDESCE
TABCD:FCB $00,$01,$02,$03,$04,$05,$06,$07,$08,$09,$10,$11,$12,$13,$14,$15
          FCB $16,$17,$18,$19,$20,$21,$22,$23,$24,$25,$26,$27,$28,$29,$30
          FCB $31,$32,$33,$34,$35,$36,$37,$38,$39,$40,$41,$42,$43,$44,$45
          FCB $46,$47,$48,$49,$50,$51,$52,$53,$54,$55,$56,$57,$58,$59,$60
          FCB $61,$62,$63,$64,$65,$66,$67,$68,$69,$70,$71,$72,$73,$74,$75
          FCB $76,$77,$78,$79,$80,$81,$82,$83,$84,$85,$86,$87,$88,$89,$90
          FCB $91,$92,$93,$94,$95,$96,$97,$98,$99

INDESCE:    LDY #$0000
            LDD $0200;CARGA D CON DATO DE CUENTA CONTENIDO EN DIRECCIÓN
                      BASE DE CONTADOR (DIRCON)
CPD:        CPD #$2710;COMPARA CON 10000
            BLO NOMA
            INY
            SUBD #$2710
            BRA CPD

LLEGA A ETIQUETA NOMA CON DIGITO MAS SIGNIFICATIVO EN "Y" Y RESTO EN D.
NOMA:       BSR DESPLE
            BRA SALABB
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP

SUBRUTINA DESPLE, QUE ESCRIBE EN BUFFER DE DISPLAY EL VALOR DE RESTO DE
CUENTA CONTENIDA EN D<10000d
DESPLE:     PSHY
            PSHX
            LDX #$0064
            IDIV
            PSHX
            PULY;COCIENTE EN Y
            PULX;X EN SU VALOR ($0100)
            TBA
            BSR CONASCII
            STAA $D0,X
            STAB $D1,X;ESCRIBE PAR DE DIGITOS MENOS SIGNIFICATIVOS
            BRA SALSUB
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
```

```
            NOP
SALSUB:     PULY
            RTS


SUBRUTINA CONASCII, DE CONVERSIÓN A PAR DE ASCII'S A PARTIR DE BYTE
HEXADECIMAL,SE ENTRA CON EL BYTE HEX EN EL ACUMULADOR A Y SE RETORNA CON
EL PAR DE ASCCI'S QUE REPRESENTAN AL NÚMERO (N<=99) EN EL ACUMULADOR D
CONASCII:   PSHY
            STAA $CB,X
            CLR $CA,X
            LDD #TABCD
            ADDD $CA,X
            XGDY
            LDAA $00,Y
            TAB
            ANDB #$0F
            ADDB #$30
            ANDA #$F0
            CLC
            RORA
            RORA
            RORA
            RORA
            ADDA #$30
            PULY
            RTS
SALABB:
```

CEN OBSCED3, asociado con el módulo observador de contadores de eventos, cuando de requiere desplegar sólo los tres dígitos menos significativos de la cuenta.

```
            BRA INDESCE
TABCD:FCB $00,$01,$02,$03,$04,$05,$06,$07,$08,$09,$10,$11,$12,$13,$14,$15
          FCB $16,$17,$18,$19,$20,$21,$22,$23,$24,$25,$26,$27,$28,$29,$30
          FCB $31,$32,$33,$34,$35,$36,$37,$38,$39,$40,$41,$42,$43,$44,$45
          FCB $46,$47,$48,$49,$50,$51,$52,$53,$54,$55,$56,$57,$58,$59,$60
          FCB $61,$62,$63,$64,$65,$66,$67,$68,$69,$70,$71,$72,$73,$74,$75
          FCB $76,$77,$78,$79,$80,$81,$82,$83,$84,$85,$86,$87,$88,$89,$90
          FCB $91,$92,$93,$94,$95,$96,$97,$98,$99

INDESCE:    LDY #$0000
            LDD $0200;CARGA D CON DATO DE CUENTA CONTENIDO EN DIRECCION
                      BASE DE CONTADOR (DIRCON)
CPD:        CPD #$2710;COMPARA CON 10000
            BLO NOMA
            INY
            SUBD #$2710
            BRA CPD

LLEGA A ETIQUETA NOMA CON DIGITO MAS SIGNIFICATIVO EN "Y" Y RESTO EN D.
NOMA:       BSR DESPLE
            BRA SALABB
            NOP
            NOP
            NOP
            NOP
            NOP
```

```
            NOP
            NOP

SUBRUTINA DESPLE, QUE ESCRIBE EN BUFFER DE DISPLAY EL VALOR DE RESTO DE
CUENTA CONTENIDA EN D<10000d
DESPLE:     PSHY
            PSHX
            LDX #$0064
            IDIV
            PSHX
            PULY;COCIENTE EN Y
            PULX;X EN SU VALOR ($0100)
            TBA
            BSR CONASCII
            STAA $D1,X
            STAB $D2,X;ESCRIBE PAR DE DIGITOS MENOS SIGNIFICATIVOS
            XGDY;PASA COCIENTE A D
            TBA;PASA COCIENTE AL ACC A.
            BSR CONASCII
            NOP
            NOP
            STAB $D0,X;ESCRIBE DIGITO MAS SIGNIFICATIVO
            PULY
            RTS

SUBRUTINA CONASCII, DE CONVERSIÓN A PAR DE ASCII'S A PARTIR DE BYTE
HEXADECIMAL,SE ENTRA CON EL BYTE HEX EN EL ACUMULADOR A Y SE RETORNA CON
EL PAR DE ASCCI'S QUE REPRESENTAN AL NÚMERO (N<=99) EN EL ACUMULADOR D.
CONASCII:   PSHY
            STAA $CB,X
            CLR $CA,X
            LDD #TABCD
            ADDD $CA,X
            XGDY
            LDAA $00,Y
            TAB
            ANDB #$0F
            ADDB #$30
            ANDA #$F0
            CLC
            RORA
            RORA
            RORA
            RORA
            ADDA #$30
            PULY
            RTS
SALABB:
```

CEN OBSCED4, asociado con el módulo observador de contadores de eventos, cuando de requiere desplegar sólo los cuatro dígitos menos significativos de la cuenta.

```
            BRA INDESCE
TABCD:FCB $00,$01,$02,$03,$04,$05,$06,$07,$08,$09,$10,$11,$12,$13,$14,$15
      FCB $16,$17,$18,$19,$20,$21,$22,$23,$24,$25,$26,$27,$28,$29,$30
      FCB $31,$32,$33,$34,$35,$36,$37,$38,$39,$40,$41,$42,$43,$44,$45
      FCB $46,$47,$48,$49,$50,$51,$52,$53,$54,$55,$56,$57,$58,$59,$60
      FCB $61,$62,$63,$64,$65,$66,$67,$68,$69,$70,$71,$72,$73,$74,$75
```

```
      FCB $76,$77,$78,$79,$80,$81,$82,$83,$84,$85,$86,$87,$88,$89,$90
      FCB $91,$92,$93,$94,$95,$96,$97,$98,$99

INDESCE:    LDY #$0000
            LDD $0200;CARGA D CON DATO DE CUENTA CONTENIDO EN DIRECCIÓN
                      BASE DE CONTADOR (DIRCON)
CPD:        CPD #$2710;COMPARA CON 10000
            BLO NOMA
            INY
            SUBD #$2710
            BRA CPD

LLEGA A ETIQUETA NOMA CON DIGITO MAS SIGNIFICATIVO EN "Y" Y RESTO EN D.
NOMA:       BSR DESPLE
            BRA SALABB
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP

SUBRUTINA DESPLE, QUE ESCRIBE EN BUFFER DE DISPLAY EL VALOR DE RESTO DE
CUENTA CONTENIDA EN D<10000d
DESPLE:     PSHY
            PSHX
            LDX #$0064
            IDIV
            PSHX
            PULY;COCIENTE EN Y
            PULX;X EN SU VALOR ($0100)
            TBA
            BSR CONASCII
            STAA $D2,X
            STAB $D3,X;ESCRIBE PAR DE DIGITOS MENOS SIGNIFICATIVOS
            XGDY;PASA COCIENTE A D
            TBA;PASA COCIENTE AL ACC A.
            BSR CONASCII
            STAA $D0,X
            STAB $D1,X;ESCRIBE PAR DE DIGITOS MAS SIGNIFICATIVOS
            PULY
            RTS

SUBRUTINA CONASCII, DE CONVERSIÓN A PAR DE ASCII'S A PARTIR DE BYTE
HEXADECIMAL,SE ENTRA CON EL BYTE HEX EN EL ACUMULADOR A Y SE RETORNA CON
EL PAR DE ASCII'S QUE REPRESENTAN AL NÚMERO (N<=99) EN EL ACUMULADOR D.
CONASCII:   PSHY
            STAA $CB,X
            CLR $CA,X
            LDD #TABCD
            ADDD $CA,X
            XGDY
            LDAA $00,Y
            TAB
            ANDB #$0F
            ADDB #$30
            ANDA #$F0
            CLC
            RORA
```

```
            RORA
            RORA
            RORA
            ADDA #$30
            PULY
            RTS
SALABB:
```

En la tabla 3.20 se muestra la TAB asociada con los CEN OBSCEDX

Tabla 3.20 Asignación de bytes asociada con el CEN OBSCEDX (X = 1, 2, ...5), empleado para obtener el código requerido por los ML, que despliegan el estado de contadores de eventos, en la UD del PLM.

Valor numérico (VN) generado por el software de traducción	Byte en CEN genérico para realizar módulos de tipo observador de contador de eventos, al cual se le asignará el valor VN
VN= Byte alto de DIRCON*	B107
VN= Byte bajo de DIRCON*	B108
VN= Byte alto de DIRBM + 2	B170
VN= Byte bajo de DIRBN + 2	B171
VN= DIRCOL1**	B150, para X = 1 B148, para X = 2 B159, para X = 3 B157, para X = 4 B130, para X = 5
VN= DIRCOL2 (ver nota sobre DIRCOL1)	B150, para X = 2 B148, para X = 3 B159, para X = 4 B157, para X = 5
VN= DIRCOL3 (ver nota sobre DIRCOL1)	B150, para X = 3 B148, para X = 4 B159, para X = 5
VN= DIRCOL4 (ver nota sobre DIRCOL1)	B150, para X = 4 B148, para X = 5
VN= DIRCOL5 (ver nota sobre DIRCOL1)	B150, para X = 5

* DIRCON = DIRBCE + NCE, DIRBCE = dirección base de buffer de contadores de eventos, NCE = número de contador de eventos cuyo estado se va a desplegar.

** DIRCOL1 = 207 + 16(R – 1) + CI, R = número de renglón donde se desplegará el estado del contador de eventos, CI = columna en la UD, a partir de la cual se observará la cuenta asociada.

DIRCOL2 = DIRCOL1 + 1
DIRCOL3 = DIRCOL1 + 2
DIRCOL4 = DIRCOL1 + 3
DIRCOL5 = DIRCOL1 + 4

3-6-4 Descripción del módulo manejador del reloj de tiempo real (RTR).

Para habilitar el poder poner a tiempo el RTR del PLM, existen dos tipos de módulos auxiliares denominados como RTRA y RTRC; la diferencia entre ambos radica en el hecho de que para el primero el estado del RTR (hora y fecha) es visible en la UD, ocupando la fecha las columnas nueve a la dieciséis del primer renglón y la hora las mismas columnas en el renglón dos, véase la figura 3.59; mientras que para el MA RTRC el estado del RTR no es visible en la UD; ambos MA contienen código que permite el poder poner a tiempo el RTR, empleando para ello únicamente los tres botones (BAXA, BAXB, y BAXC), ligados al puerto auxiliar A del bloque de comando local y despliegue (BCLD) del PLM.

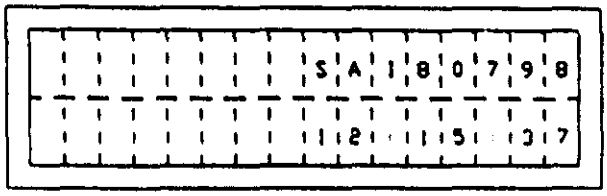


Figura 3.59 Aspecto que tendría el despliegue del estado del RTR en la UD, cuando la fecha y hora son los indicados, al ejecutarse en el PLM un programa que contenga la declaración del MA RTRA.

En la figura 3.60 se muestra la representación como bloque de los MA RTRA y RTRC, estos módulos deben ser declarados en el subprograma temporizado y la sintaxis asociada es:

RTRX;

Donde:
X será la letra A, si se desea que el estado del RTR sea visible en la UD, en otro caso X deberá ser la letra C.

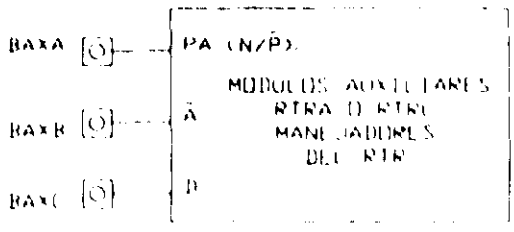


Figura 3.60 Representación como bloque de los módulos auxiliares RTRA y RTRC

Estos módulos deben ser declarados una sola vez en un programa en SIIL1; de este modo, al ejecutarse el mismo quedarán habilitadas las facilidades de despliegue y puesta a tiempo del RTR; para poner a tiempo el RTR se deben llevar a cabo los siguientes pasos:

1.- Oprimir el botón PA (BAXA), esto hace que aparezca en la UD el estado del RTR al instante de la opresión sin mostrarse avance en el tiempo, en todos los siguientes pasos salvo el último, el botón PA deberá permanecer oprimido.

2.- Oprimir el botón "A" (BAXB), esto hace que aparezcan sucesivamente en la posición de los caracteres que denotan el año, los siguientes pares de caracteres: **SE**, indicando que al oprimir el botón "D" los dígitos indicadores de los segundos avanzan en forma natural; **MI**, indicando que al oprimir el botón "D" los dígitos indicadores de los minutos avanzan en forma natural; **HO**, indicando que al oprimir el botón "D" los dígitos indicadores de las horas avanzan en forma natural; **DS**, indicando que al oprimir el botón "D" los caracteres indicadores del día de la semana avanzan en forma natural; **DM**, indicando que al oprimir el botón "D" los dígitos indicadores del día del mes avanzan en forma natural; **ME**, indicando que al oprimir el botón "D" los dígitos indicadores del mes avanzan en forma natural; repitiéndose lo anterior en forma cíclica mientras el botón "A" permanezca oprimido.

Por defecto, el avance que queda habilitado al oprimirse el botón "PA" es el correspondiente al par de dígitos que denota el año, no habiendo testificación de avance como las indicadas en el párrafo anterior.

3.- Soltar el botón "A" cuando en las columnas quince y dieciséis del renglón uno de la UD, aparezcan los caracteres que denotan al par de dígitos que se desea ajustar; por ejemplo, si se desea ajustar los dígitos que indican "horas", el botón A deberá soltarse cuando en la esquina superior derecha de la UD aparezcan el par de letras "HO", véase la figura 3.61.

4.- Oprimir el botón "D" (BAXC), hasta que el par de dígitos seleccionado llegue al valor deseado.

5.- Repetir los pasos dos, tres y cuatro, para cada par de dígitos indicadores del estado del RTR que se desee ajustar.

6.- Soltar el botón "PA", esto hace que la hora y fecha indicada por el usuario sea copiada al RTR.

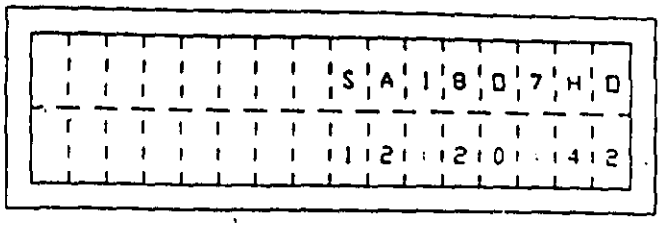


Figura 3.61 Aspecto de la pantalla de la UD, indicando que el par de dígitos indicadores de las "horas", se incrementarán al oprimirse el botón "D".

Descripción del CEN asociado con los módulos RTRA y RTRC.

El flujo de ejecución de estos MA se muestra en la figura 3.62 y es para ambos en esencia el mismo, salvo por el hecho de que el software de traducción deshabilita para el MA RTRC, una invocación a la subrutina COPRD que copia el estado del RTR al buffer de la UD, véanse la dirección 0366H (posición de la etiqueta DESPREL), en el CEN RTRX mostrado más adelante y el último renglón de la tabla 3.21.

El tiempo de ejecución de estos módulos, cuando no se están accesando las facilidades para poner a tiempo el RTR (operación normal), es de 114.5 μs para el MA RTRA y de 22.5 μs para el MA RTRC, con fe = 2MHz.

El código del CEN RTRX se muestra a continuación:

```
CEN RTRX, EMPLEADO PARA HABILITAR LOS MA RTRA Y RTRC

0000 207E                          BRA INIRTRPD
0002 53454D49 TESPDD:FCB $53,$45,$4D,$49,$48,$4F,$44,$53,$44,$4D,$4D,$45
     484F4453
     444D4D45
000E 4C554D41 DIASEM:FCB $4C,$55,$4D,$41,$4D,$49,$4A,$55,$56,$49,$53,$41
                      $44,$4F
     4D494A55
     56495341
     444F
001C 00010203 TABCD:FCB $00,$01,$02,$03,$04,$05,$06,$07,$08,$09,$10,$11
                     $12,$13,$14,$15
     04050607
     08091011
     12131415
002C 16171819        FCB $16,$17,$18,$19,$20,$21,$22,$23,$24,$25,$26
                     $27,$28,$29,$30
     20212223
     24252627
     282930
003B 31323334        FCB $31,$32,$33,$34,$35,$36,$37,$38,$39,$40,$41
                     $42,$43,$44,$45
     35363738
```

```
     39404142
     434445
004A 46474849        FCB $46,$47,$48,$49,$50,$51,$52,$53,$54,$55,$56
                         $57,$58,$59,$60
     50515253
     54555657
     585960
0059 61626364        FCB $61,$62,$63,$64,$65,$66,$67,$68,$69,$70,$71
                         $72,$73,$74,$75
     65666768
     69707172
     737475
0068 76777879        FCB $76,$77,$78,$79,$80,$81,$82,$83,$84,$85,$86
                         $87,$88,$89,$90
     80818283
     84858687
     888990
0077 91929394        FCB  $91,$92,$93,$94,$95,$96,$97,$98,$99
     95969798
     99

0080 A6F1             INIRTRPD:  LDAA $F1,X
0082 270B                        BEQ NOPP
                   CODIGO AL ENTRAR LA PRIMERA VEZ
0084 8606                        LDAA #$06
0086 A7C0                        STAA $C0,X;PONE APUNTADOR DE PAR DE
                                           DIGITOS EN (AÑOS).
0088 6FC1                        CLR $C1,X;PONE A CERO CONTADOR BASE
                                           CINCUENTA,EMPLEADO PARA
                                           GENERAR CADECIA DE .5 SEGUNDOS
008A 6FC9                        CLR $C9,X;PONE A CERO TESTIGO DE QUE SE
                                           HA "PUESTO EL RELOJ"
008C 7E0377                      JMP SALAB;SALTA A SIGUIENTE MÓDULO

008F B61BB0           NOPP:      LDAA $1BB0;LEE PUERTO AUXILIAR A (PAUXA)
0092 8401                        ANDA #$01
0094 2602                        BNE DDESPREL
0096 2003                        BRA CADEN
0098 7E02EE           DDESPREL:  JMP SALIDA
009B 6CC1             CADEN:     INC $C1,X
009D A6C1                        LDAA $C1,X
009F 8132                        CMPA #$32
00A1 2602                        BNE DDFF1
00A3 2003                        BRA DDFF
00A5 7E0377           DDFF1:     JMP SALAB
00A8 A6C9             DDFF:      LDAA $C9,X
00AA 81FF                        CMPA #$FF
00AC 2602                        BNE WWW
00AE 200A                        BRA WW
00B0 BD023D           WWW:       JSR COPSTAT
00B3 BD026F                      JSR COPRD
00B6 86FF                        LDAA #$FF
00B8 A7C9                        STAA $C9,X
00BA 6FC1             WW:        CLR $C1,X
00BC B61BB0                      LDAA $1BB0;LEE PUERTO AUXILIAR A (PAUXA)
00BF 8407                        ANDA #$07
00C1 8104                        CMPA #$04;PREGUNTA SI BAXB ESTA OPRIMIDO
                                           Y BAXC SUELTO (INC APUN)
00C3 2702                        BEQ III
00C5 2003                        BRA II
```

```
00C7 7E021C          III:     JMP INCAPUPDD
00CA 8102            II:      CMPA #$02;PREGUNTA SI BAXC ESTÁ OPRIMIDO
                                       Y BAXB SUELTO (INC DIGITOS)
00CC 2703                     BEQ INCPDD
00CE 7E02EE                   JMP SALIDA
                  CODIGO DE INCPDD
00D1 A6C0            INCPDD:  LDAA $C0,X
00D3 8100                     CMPA #$00
00D5 2730                     BEQ INCSEG
00D7 8101                     CMPA #$01
00D9 273E                     BEQ INCMIN
00DB 8102                     CMPA #$02
00DD 274C                     BEQ INCHOR
00DF 8103                     CMPA #$03
00E1 275A                     BEQ INCDSEM
00E3 8104                     CMPA #$04
00E5 2702                     BEQ UUU
00E7 2003                     BRA UU
00E9 7E0172          UUU:     JMP INCDMES
00EC 8105            UU:      CMPA #$05
00EE 2702                     BEQ MMM
00F0 2003            MM:      BRA INCA
00F2 7E020A          MMM:     JMP INCMES
00F5 A6C8            INCA:    LDAA $C8,X
00F7 4C                       INCA
00F8 8164                     CMPA #$64
00FA 2601                     BNE NO100
00FC 4F                       CLRA
00FD A7C8            NO100:   STAA $C8,X
00FF BD01ED                   JSR CONASCII
0102 EDDE                     STD $DE,X
0104 7E02EE                   JMP SALIDA
0107 A6C2            INCSEG:  LDAA $C2,X
0109 4C                       INCA
010A 813C                     CMPA #$3C
010C 2601                     BNE NO60S
010E 4F                       CLRA
010F A7C2            NO60S:   STAA $C2,X
0111 BD01ED                   JSR CONASCII
0114 EDEE                     STD $EE,X
0116 7E02EE                   JMP SALIDA
0119 A6C3            INCMIN:  LDAA $C3,X
011B 4C                       INCA
011C 813C                     CMPA #$3C
011E 2601                     BNE NO60M
0120 4F                       CLRA
0121 A7C3            NO60M:   STAA $C3,X
0123 BD01ED                   JSR CONASCII
0126 EDEB                     STD $EB,X
0128 7E02EE                   JMP SALIDA
012B A6C4            INCHOR:  LDAA $C4,X
012D 4C                       INCA
012E 8118                     CMPA #$18
0130 2601                     BNE NO24H
0132 4F                       CLRA
0133 A7C4            NO24H:   STAA $C4,X
0135 BD01ED                   JSR CONASCII
0138 EDE8                     STD $E8,X
013A 7E02EE                   JMP SALIDA
013D A6C5            INCDSEM: LDAA $C5,X
```

```
013F 4C                         INCA
0140 8108                       CMPA #$08
0142 2602                       BNE NOFINS
0144 8601                       LDAA #$01
0146 A7C5           NOFINS:     STAA $C5,X
0148 8D05                       BSR ASCIIDS
014A EDD8                       STD $D8,X
014C 7E02EE                     JMP SALIDA
```

SUBRUTINA ASCIIDS, QUE OBTIENE EL PAR DE ASCII'S INDICADORES DE ABREVIATURA DE DIA DE LA SEMANA;SE ENTRA CON NÚMERO DE DÍA EN ACUMULADOR A Y SE RETORNA CON CADENA DE CARACTERES INDICADORES DE DÍA, EN EL REGISTRO D

```
014F 4A             ASCIIDS:    DECA
0150 0C                         CLC
0151 49                         ROLA;SE MULTIPLICA POR DOS EL NÚMERO DE
                                     DÍA MENOS UNO
0152 A7CB                       STAA $CB,X
0154 6FCA                       CLR $CA,X
0156 CC000E                     LDD #DIASEM
0159 E3CA                       ADDD $CA,X
015B 188F                       XGDY;CARGA EN Y DIRECCIÓN DE TABLA "DIA
                                     DE LA SEMANA"
□
015D 18EC00                     LDD $00,Y;COPIA EN D EL PAR DE ASCII'S
                                          QUE DENOTAN AL DÍA DE LA SEMANA
□
0160 39                         RTS
```

SUBRUTINA ASCIISPDD, EMPLEADA PARA LA OBTENCIÓN DEL PAR DE ASCII'S, INDICADORES DE QUE PAR DE DIGITOS RESPONDERAN A PULSACIONES DEL BOTÓN DE INCREMENTO DE DIGITOS;SE ENTRA CON EL ESTADO DEL APUNTADOR DE PAR DE DIGITOS EN EL ACUMULADOR A,RETORNANDOSE CON LA CADENA DE DOS CARACTERES INDICADORES (SE,MI,HO,DS,ETC) EN EL REGISTRO D

```
0161 0C             ASCIISPDD:  CLC
0162 49                         ROLA;SE MULTIPLICA POR DOS EL NÚMERO DE
                                     PAR DE DIGITOS (PDD)
0163 A7CB                       STAA $CB,X
0165 6FCA                       CLR $CA,X
0167 CC0002                     LDD #TESPDD
016A E3CA                       ADDD $CA,X
016C 188F                       XGDY;CARGA EN Y DIRECCIÓN DE TABLA "DIA
                                     DE LA SEMANA"
016E 18EC00                     LDD $00,Y;COPIA EN D EL PAR DE ASCII'S
                                          QUE DENOTAN AL DÍA DE LA SEMANA
0171 39                         RTS

0172 A6C6           INCDMES:    LDAA $C6,X
0174 4C                         INCA
0175 8D0F                       BSR DIATOPE
0177 A1CB                       CMPA $CB,X
0179 2602                       BNE NOFINM
017B 8601                       LDAA #$01
017D A7C6           NOFINM:     STAA $C6,X
017F 8D6C                       BSR CONASCII
0181 EDDA                       STD $DA,X
0183 7E02EE                     JMP SALIDA
```

SUBRUTINA DIATOPE,RETORNA EN LA DIRECCIÓN X+$CB EL TOPE DE DIAS DEL MES MAS 1

```
0186 36          DIATOPE:  PSHA
0187 A6C7                  LDAA $C7,X
0189 8100                  CMPA #$00
018B 273E                  BEQ ENE
018D 8101                  CMPA #$01
018F 273C                  BEQ FEB
0191 8102                  CMPA #$02
0193 2742                  BEQ MAR
0195 8103                  CMPA #$03
0197 2740                  BEQ ABR
0199 8104                  CMPA #$04
019B 273E                  BEQ MAY
019D 8105                  CMPA #$05
019F 273C                  BEQ JUN
01A1 8106                  CMPA #$06
01A3 273A                  BEQ JUL
01A5 8107                  CMPA #$07
01A7 2738                  BEQ AGO
01A9 8108                  CMPA #$08
01AB 2736                  BEQ SEP
01AD 8109                  CMPA #$09
01AF 2734                  BEQ OCT
01B1 810A                  CMPA #$0A
01B3 2732                  BEQ NOV
01B5 810B                  CMPA #$0B
01B7 2730                  BEQ DIC
01B9 8620        MESDE31:  LDAA #$20
01BB A7CB        QK:       STAA $CB,X
01BD 202C                  BRA VVV;SALTA A SALIDA DE LA SUBRUTINA
01BF 861F        MESDE30:  LDAA #$1F
01C1 20F8                  BRA QK
01C3 861E        MESDE29:  LDAA #$1E
01C5 20F4                  BRA QK
01C7 861D        MESDE28:  LDAA #$1D
01C9 20F0                  BRA QK
01CB 20EC        ENE:      BRA MESDE31
01CD A6C8        FEB:      LDAA $C8,X
01CF 8403                  ANDA #$03;CHECA SI EL AÑO ES BISIESTO
01D1 2602                  BNE NOBIS
01D3 20EE                  BRA MESDE29
01D5 20F0        NOBIS:    BRA MESDE28
01D7 20E0        MAR:      BRA MESDE31
01D9 20E4        ABR:      BRA MESDE30
01DB 20DC        MAY:      BRA MESDE31
01DD 20E0        JUN:      BRA MESDE30
01DF 20D8        JUL:      BRA MESDE31
01E1 20D6        AGO:      BRA MESDE31
01E3 20DA        SEP:      BRA MESDE30
01E5 20D2        OCT:      BRA MESDE31
01E7 20D6        NOV:      BRA MESDE30
01E9 20CE        DIC:      BRA MESDE31
01EB 32          VVV:      PULA
01EC 39                    RTS
```

SUBRUTINA CONASCII, EMPLEADA PARA LA OBTENCIÓN DEL PAR DE ASCII'S QUE REPRESENTAN A UN NÚMERO N (N<=99);SE ENTRA CON EL BYTE HEX, CUYO VALOR ES N,EN EL ACUMULADOR A Y SE RETORNA CON EL PAR DE ASCCI'S QUE REPRESENTAN AL NÚMERO N EN EL ACUMULADOR D

```
01ED A7CB           CONASCII:  STAA $CB,X
01EF 6FCA                      CLR $CA,X
01F1 CC001C                    LDD #TABCD
01F4 E3CA                      ADDD $CA,X
01F6 188F                      XGDY
01F8 18A600                    LDAA $00,Y
01FB 16                        TAB
01FC C40F                      ANDB #$0F
01FE CB30                      ADDB #$30
0200 84F0                      ANDA #$F0
0202 0C                        CLC
0203 46                        RORA
0204 46                        RORA
0205 46                        RORA
0206 46                        RORA
0207 8B30                      ADDA #$30
0209 39                        RTS

020A A6C7           INCMES:    LDAA $C7,X
020C 4C                        INCA
020D 810C                      CMPA #$0C
020F 2601                      BNE NOFINA
0211 4F                        CLRA
0212 A7C7           NOFINA:    STAA $C7,X
0214 4C                        INCA
0215 8DD6                      BSR CONASCII
0217 EDDC                      STD $DC,X
0219 7E02EE         SS:        JMP SALIDA

021C 6CC0           INCAPUPDD: INC $C0,X
021E A6C0                      LDAA $C0,X
0220 8107                      CMPA #$07
0222 2602                      BNE NOTOPE
0224 6FC0                      CLR $C0,X
0226 8106           NOTOPE:    CMPA #$06
0228 270A                      BEQ TESA
022A A6C0                      LDAA $C0,X
022C BD0161                    JSR ASCIISPDD
022F EDDE                      STD $DE,X;CARGA PAR DE ASCII'S,
                                        INDICADORES DE QUE PAR DE
                                        DIGITOS, RESPONDERÁ A
                                        PULSACIONES DEL BOTÓN
                                        DE INCREMENTO.
0231 7E02EE                    JMP SALIDA
0234 A6C8           TESA:      LDAA $C8,X
0236 8DB5                      BSR CONASCII
0238 EDDE                      STD $DE,X
023A 7E02EE                    JMP SALIDA
                  TERMINA CODIGO DE INCAPUPDD

SUBRUTINA COPSTAT,COPIA EL STATUS DEL RTR EN BUFFER AUXILIAR
(X+$C2)<--SEGUNDOS,(X+$C3)<----MINUTOS,.....,(X+$C8)<----AÑO
023D 3C             COPSTAT:   PSHX
023E 18CE1BC2                  LDY #$1BC2;CARGA Y CON DIRECCIÓN BASE+2
                                          DEL RTR,(DIR DE SEGS).
0242 18A601         ALFA:      LDAA $01,Y
0245 840F                      ANDA #$0F
0247 C60A                      LDAB #$0A
0249 3D                        MUL
```

```
024A 18A600                     LDAA $00,Y
024D 840F                       ANDA #$0F
024F 1B                         ABA
0250 A7C2                       STAA $C2,X
0252 1808                       INY
0254 1808                       INY
0256 08                         INX
0257 188C1BC8                   CPY #$1BC8
025B 2601                       BNE BETA
025D 08                         INX
025E 188C1BCE         BETA:     CPY #$1BCE
0262 26DE                       BNE ALFA
0264 18A600                     LDAA $00,Y;COPIA EN A DÍA DE LA SEMANA
                                          CONTENIDO EN RTR.
0267 840F                       ANDA #$0F
0269 38                         PULX
026A A7C5                       STAA $C5,X;COPIA EN BUFFER AUXILIAR DIA
                                          DE LA SEMANA TOMADO DE RTR.
026C 6AC7                       DEC $C7,X;AJUSTA TESTIGO DE NÚMERO DE
                                          MES EN BUFFER AUXILIAR,DE MODO
                                          QUE ENERO=0,....DICIEMBRE=11
026E 39                         RTS

SUBRUTINA COPRD,COPIA DIRECTAMENTE EL ESTADO DEL RTR A DISPLAY AND491
0270 36               COPRD:    PSHA
0270 863A                       LDAA #$3A
0272 A7ED                       STAA $ED,X
0274 A7EA                       STAA $EA,X;PONE DELIMITADORES ENTRE
                                          HORAS,MINUTOS Y SEGUNDOS
0276 B61BC2                     LDAA $1BC2
0279 840F                       ANDA #$0F
027B 8B30                       ADDA #$30
027D A7EF                       STAA $EF,X;COPIA A DISPLAY DIGITO DE
                                          SEGUNDOS
027F B61BC3                     LDAA $1BC3
0282 840F                       ANDA #$0F
0284 8B30                       ADDA #$30
0286 A7EE                       STAA $EE,X;COPIA A DISPLAY DIGITO QUE
                                          REPRESENT A DECENAS DE
                                          SEGUNDOS
0288 B61BC4                     LDAA $1BC4
028B 840F                       ANDA #$0F
028D 8B30                       ADDA #$30
028F A7EC                       STAA $EC,X;COPIA A DISPLAY DIGITO DE
                                          MINUTOS
0291 B61BC5                     LDAA $1BC5
0294 840F                       ANDA #$0F
0296 8B30                       ADDA #$30
0298 A7EB                       STAA $EB,X;COPIA A DISPLAY DIGITO QUE
                                          REPRESENT A DECENAS DE MINUTOS
029A B61BC6                     LDAA $1BC6
029D 840F                       ANDA #$0F
029F 8B30                       ADDA #$30
02A1 A7E9                       STAA $E9,X;COPIA A DISPLAY DIGITO DE
                                          HORAS
02A3 B61BC7                     LDAA $1BC7
02A6 840F                       ANDA #$0F
02A8 8B30                       ADDA #$30
02AA A7E8                       STAA $E8,X;COPIA A DISPLAY DIGITO QUE
```

```
                                    REPRESENT A DECENAS DE HORAS
02AC B61BC8                 LDAA $1BC8
02AF 840F                   ANDA #$0F
02B1 8B30                   ADDA #$30
02B3 A7DB                   STAA $DB,X;COPIA A DISPLAY DIGITO DE
                                    UNIDADES PARA DÍA DE MES
02B5 B61BC9                 LDAA $1BC9
02B8 840F                   ANDA #$0F
02BA 8B30                   ADDA #$30
02BC A7DA                   STAA $DA,X;COPIA A DISPLAY DIGITO QUE
                                    REPRESENT A DECENAS DE DÍA DE
                                    MES
02BE B61BCA                 LDAA $1BCA
02C1 840F                   ANDA #$0F
02C3 8B30                   ADDA #$30
02C5 A7DD                   STAA $DD,X;COPIA A DISPLAY DIGITO DE
                                    UNIDADES (MES)
02C7 B61BCB                 LDAA $1BCB
02CA 840F                   ANDA #$0F
02CC 8B30                   ADDA #$30
02CE A7DC                   STAA $DC,X;COPIA A DISPLAY DIGITO QUE
                                    REPRESENT A DECENAS (MES)
02D0 B61BCC                 LDAA $1BCC
02D3 840F                   ANDA #$0F
02D5 8B30                   ADDA #$30
02D7 A7DF                   STAA $DF,X;COPIA A DISPLAY DIGITO DE
                                    UNIDADES (AÑOS)
02D9 B61BCD                 LDAA $1BCD
02DC 840F                   ANDA #$0F
02DE 8B30                   ADDA #$30
02E0 A7DE                   STAA $DE,X;COPIA A DISPLAY DIGITO QUE
                                    REPRESENT A DECENAS (AÑOS)
02E2 B61BCE                 LDAA $1BCE;COPIA EN A NÚMERO DE DÍA DE LA
                                    SEMANA
02E5 840F                   ANDA #$0F
02E7 BD014F                 JSR ASCIIDS
02EA EDD8                   STD $D8,X
02EC 32                     PULA
02ED 39                     RTS
```

CODIGO QUE PREGUNTA SI BAXA NO ESTÁ OPRIMIDO, PARA EN SU CASO COPIAR A RTR NUEVA HORA Y FECHA, SI EL TESTIGO DE "PONER EL RELOJ" (TPREL) ESTÁ VERIFICADO.

```
02EE B61BB0         SALIDA: LDAA $1BB0
02F1 8401                   ANDA #$01
02F3 276F                   BEQ ASALAB;SALE SI CONTINUA OPRIMIDO
                                BOTON BAXA (N/PNEG)
02F5 A6C9                   LDAA $C9,X
02F7 81FF                   CMPA #$FF
02F9 266B                   BNE DESPREL;PASA A COPIAR DIGITOS DE
                                RELOJ A DISPLAY
```

COPIA HORA Y FECHA ESPECIFICADA POR EL USUARIO A RTR

```
02FB A6C8                   LDAA $C8,X
02FD BD01ED                 JSR CONASCII
0300 EDDE                   STD $DE,X;ACTUALIZA AÑO, POR SI EL USUARIO
                                DEJO TESTIGO DE PAR DE DIGITOS
                                ANTES DE LEVANTAR NIVEL DE
```

```
                                    BOTON BAXA (N/PNEG)
0302 6FC9                   CLR $C9,X;DESVERIFICA TESTIGO DE QUE EL
                                    USUARIO DESEA CAMBIAR HORA.
0304 BD0186                 JSR DIATOPE
0307 A6CB                   LDAA $CB,X
0309 4A                     DECA
030A A1C6                   CMPA $C6,X
030C 2502                   BLO AJUSTA
030E 2005                   BRA NOPRO
0310 BD01ED         AJUSTA: JSR CONASCII
0313 EDDA                   STD $DA,X;CAMBIA DIA DE MES EN DISPLAY,SI
                                    EL USUARIO ESPECIFICO EL MISMO
                                    ERRONEAMENTE
0315 A6EF           NOPRO:  LDAA $EF,X
0317 B71BC2                 STAA $1BC2;COPIA A RTR DIGITO DE UNIDADES
                                    (SEGUNDOS)
031A A6EE                   LDAA $EE,X
031C B71BC3                 STAA $1BC3;COPIA A RTR DIGITO DE LAS
                                    DECENAS (SEGUNDOS)
031F A6EC                   LDAA $EC,X
0321 B71BC4                 STAA $1BC4;COPIA A RTR DIGITO DE UNIDADES
                                    (MINUTOS)
0324 A6EB                   LDAA $EB,X
0326 B71BC5                 STAA $1BC5;COPIA A RTR DIGITO DE LAS
                                    DECENAS (MINUTOS)
0329 A6E9                   LDAA $E9,X
032B B71BC6                 STAA $1BC6;COPIA A RTR DIGITO DE UNIDADES
                                    (HORAS)
032E A6E8                   LDAA $E8,X
0330 B71BC7                 STAA $1BC7;COPIA A RTR DIGITO DE LAS
                                    DECENAS (HORAS)
0333 A6DB                   LDAA $DB,X
0335 B71BC8                 STAA $1BC8;COPIA A RTR DIGITO DE UNIDADES
                                    (DÍA DEL MES)
0338 A6DA                   LDAA $DA,X
033A B71BC9                 STAA $1BC9;COPIA A RTR DIGITO DE LAS
                                    DECENAS (DÍA DEL MES)
033D A6DD                   LDAA $DD,X
033F B71BCA                 STAA $1BCA;COPIA A RTR DIGITO DE UNIDADES
(MES)
0342 A6DC                   LDAA $DC,X
0344 B71BCB                 STAA $1BCB;COPIA A RTR DIGITO DE LAS
                                    DECENAS (MES)
0347 A6DF                   LDAA $DF,X
0349 B71BCC                 STAA $1BCC;COPIA A RTR DIGITO DE UNIDADES
                                    (AÑO)
034C A6DE                   LDAA $DE,X
034E B71BCD                 STAA $1BCD;COPIA A RTR DIGITO DE LAS
                                    DECENAS (AÑO)
0351 A6C5                   LDAA $C5,X;COPIA EN A DÍA DE LA SEMANA
                                    ESPECIFICADO POR EL USUARIO
0353 B71BCE                 STAA $1BCE;PONE NÚMERO DE DÍA DE LA
                                    SEMANA EN RTR
0356 3C                     PSHX
0357 C608                   LDAB #$08
0359 8620                   LDAA #$20
035B A7D8           BORRAR: STAA $D8,X
035D A7E8                   STAA $E8,X
035F 08                     INX
0360 5A                     DECB
```

```
0361 26F8                       BNE BORRAR
0363 38                         PULX
0364 2011           ASALAB:     BRA SALAB

0366 BD026F         DESPREL:    JSR COPRD;COPIA ESTADO DE RTR A DISPLAY
0369 200C                       BRA SALAB
036B 3C                         PSHX;ESTE CODIGO BORRA EL ESPACIO OCUPADO
                                    POR LA FECHA EN EL DISPLAY
036C C608                       LDAB #$08
036E 8620                       LDAA #$20
0370 A7D8           BORRAR1:    STAA $D8,X
0372 08                         INX
0373 5A                         DECB
0374 26FA                       BNE BORRAR1
0376 38                         PULX
                    SALAB:
```

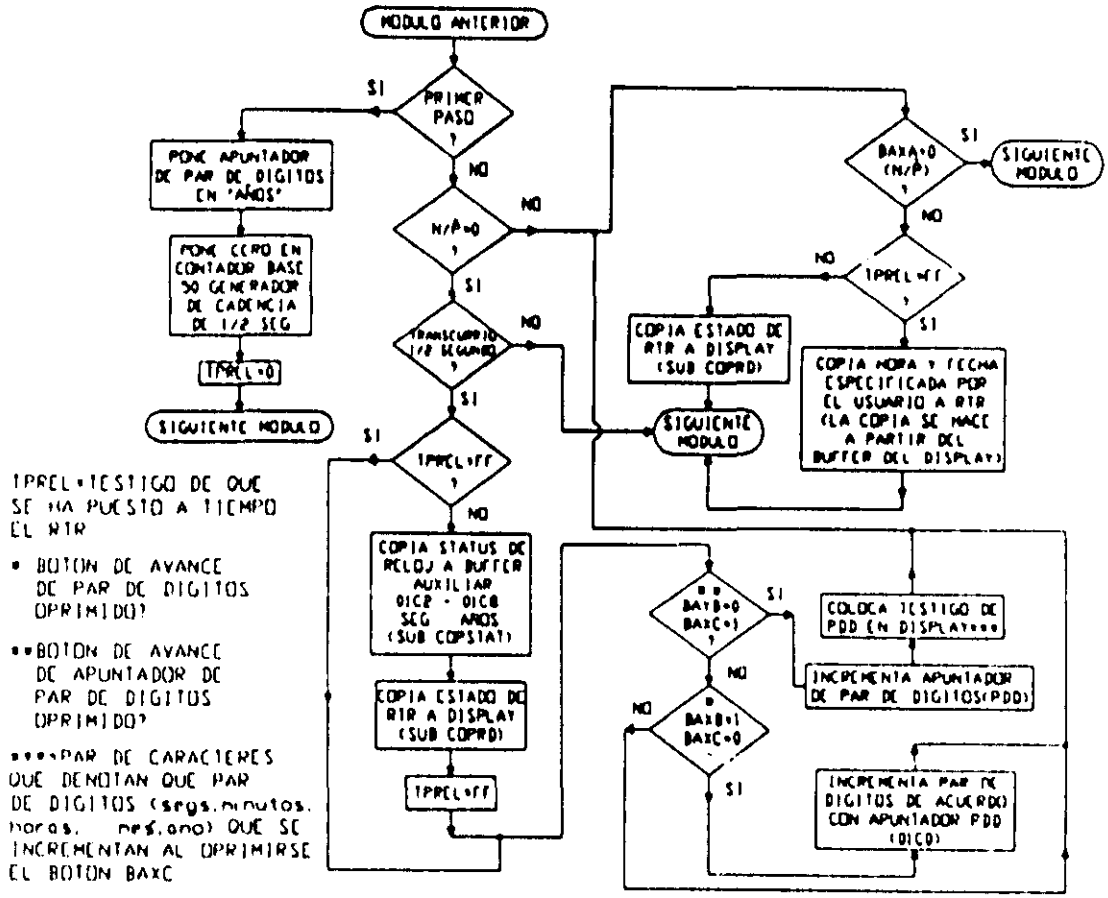


Figura 3.62 Flujo de ejecución de los módulos auxiliares RTA y RTRC, que habilitan el despliegue local y puesta a tiempo del RTR del PLM.

En la tabla 3.21 se muestra la TAB asociada con los MA RTRA y RTRC

Tabla 3.21 Asignación de bytes asociada con el CEN RTRX, empleado para obtener el código requerido por los MA RTRA y RTRC, que habilitan el despliegue del estado del RTR y las facilidades de puesta a tiempo del mismo.

Valor numérico (VN) generado por el software de traducción	Byte en CEN RTRX, al cual se le asignará el valor VN, al obtenerse el código objeto asociado con los MA RTRA y RTRC.
VN= Byte alto de DIRBM + 887	B141, B166
VN= Byte bajo de DIRBM + 887	B142, B167
VN= Byte alto de DIRBM + 540	B200
VN= Byte bajo de DIRBN + 540	B201
VN= Byte alto de DIRBM + 370	B234
VN= Byte bajo de DIRBM + 370	B235
VN= Byte alto de DIRBM + 522	B243
VN= Byte bajo de DIRBM + 522	B244
VN= Byte alto de DIRBM + 750	B153, B207, B261, B279, B297, B315, B333, B388, B538, B562, B571
VN= Byte bajo de DIRBM + 750	B154, B208, B262, B280, B298, B316, B334, B389, B539, B563, B572
VN= Byte alto de DIRBM + 623	B180, B871
VN= Byte bajo de DIRBM + 623	B181, B872
VN= Byte alto de DIRBM + 493	B256, 274, B292, B310, B766, B785
VN= Byte bajo de DIRBM + 493	B257, 275, B293, B311, B767, B786
VN= Byte alto de DIRBM + 573	B172
VN= Byte bajo de DIRBM + 573	B173
VN= Byte alto de DIRBM + 557	B557
VN= Byte bajo de DIRBM + 558	B558
VN= Byte alto de DIRBM + 335	B744
VN= Byte bajo de DIRBM + 335	B745
VN= Byte alto de DIRBM + 390	B773
VN= Byte bajo de DIRBM + 390	B774
VN= Byte alto de DIRBM + 2	B360
VN= Byte bajo de DIRBM + 2	B361
VN= Byte alto de DIRBM + 14	B343
VN= Byte bajo de DIRBM + 14	B344
VN= Byte alto de DIRBM + 28	B498
VN= Byte bajo de DIRBM + 28	B499
VN= 1 (sólo para el MA RTRC)	B870, B871, B872

3-6-5 Descripción del MA DESP, que copia a la UD el contenido de su buffer asociado.

Siempre que se empleen módulos que hagan uso de la UD, se deberá declarar en el subprograma principal un MA denominado como DESP, esto habrá de hacerse una sola vez. Este MA copia cíclicamente el contenido del buffer de la UD al desplegador fisico, la sintaxis para declarar a este módulo es la siguiente:

DESP;

Descripción los CEN asociados con el MA DESP.

Para armar el código correspondiente al MA DESP, el software de traducción emplea dos CEN denominados como DESP e INCESPLM, quedando el mismo integrado por la conjunción de ambos, no habiendo para este MA TAB asociada. El CEN DESP copia del buffer de la UD (direcciones 01D0H a 01EFH) a la unidad desplegadora física (AND491), estando su código diseñado de modo que el tiempo de espera que se requiere para escribir un dato o comando a la UD no "cuelgue" el tiempo de ejecución del lazo del programa principal (LPP), esto se logra gracias al hecho de que este MA queda colocado en el LPP, y cuando se requiere escribir un dato o comando en la UD física, el código pregunta si esto es posible, en caso afirmativo lo hace, en otro caso salta al siguiente módulo, reintentándose la escritura la siguiente vez que se ejecuta el código y así sucesivamente.

El código INCESPLM opera propiamente únicamente en la primera vuelta del LPP, inicializando la RAM de caracteres especiales de la UD.

El máximo tiempo de ejecución para el módulo DESP es de 70.5 μs bajo condiciones normales, (fc = 2MHz y primera vuelta de LPP ejecutada). El flujo de ejecución correspondiente a los dos CEN que integran este módulo se muestra en las figuras 3.63 y 3.64.

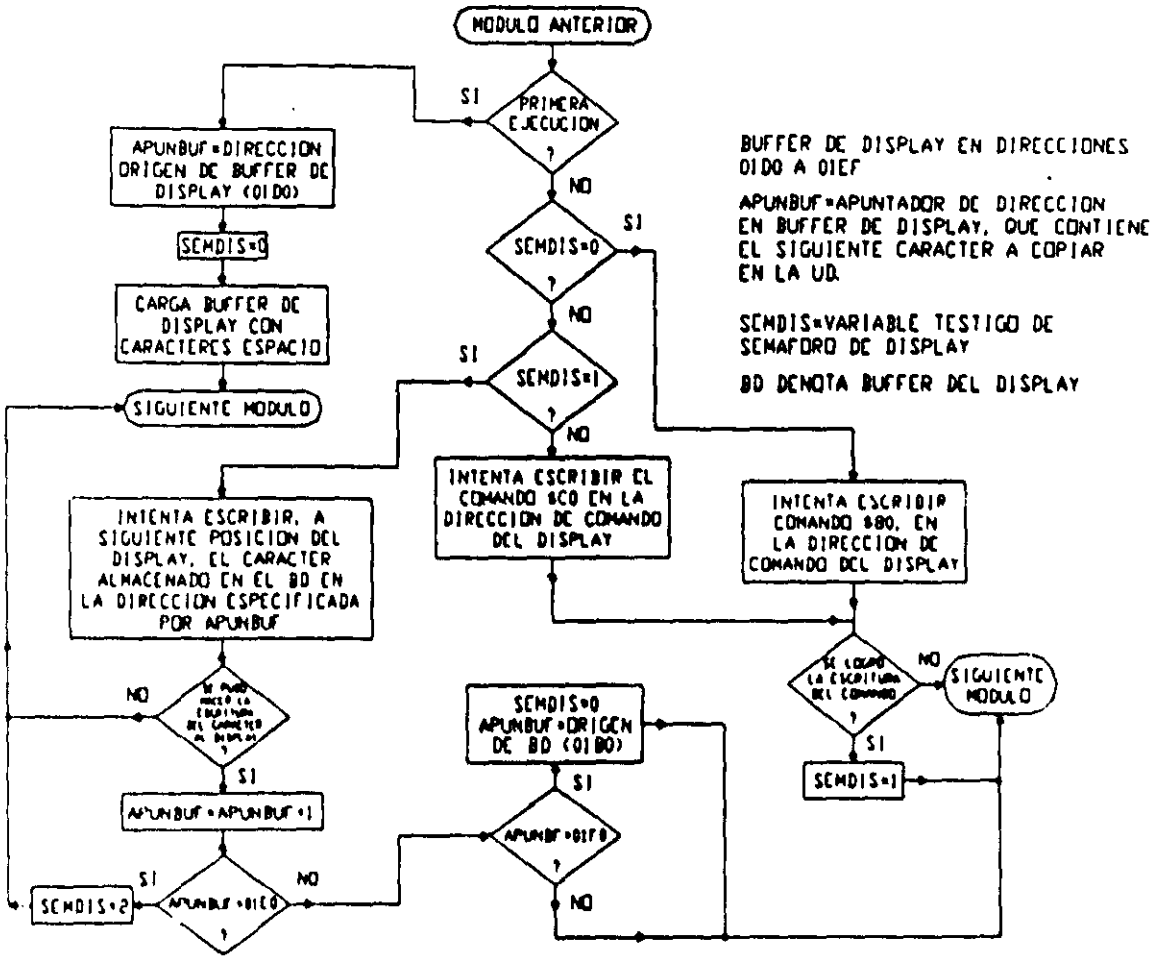


Figura 3.63 Flujo de ejecución del CEN DESP, (primer tramo de código del MA DESP).

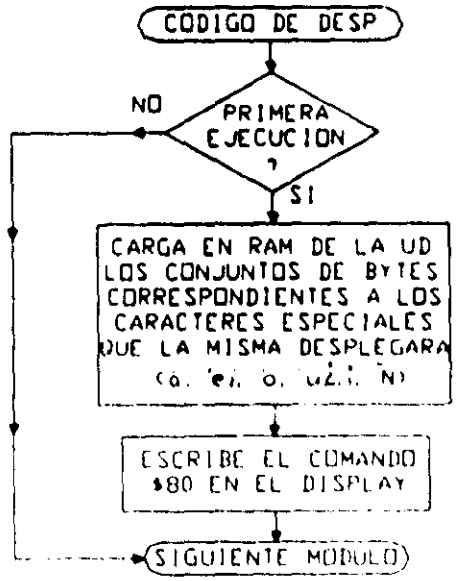


Figura 3.64 Flujo de ejecución del CEN INCESPLM, (segundo tramo de código del MA (DESP).

El código correspondiente a los **CEN DESP** e **INCESPLM** se muestra a continuación:

```
CEN DESP,EMPLEADO JUNTO CON EL CEN INCESPLM,PARA HABILITAR EL MA DESP


DESPLEGADOR DE DOS RENGLONES EN DISPLAY AND491 EN DIRECCIÓN BASE 1BA0H,
A PARTIR DE BUFFER EN RAM

□
BUFER DE DISPLAY EN DIRECCIONES 01D0 A 01EF
□
RENGLON1 01D0 A 01DF,RENGLON2 01E0 A 01EF
□

□
 0000 A6F1                      LDAA $F2,X
□
 0002 271A                      BEQ NOPP
 0004 CC01D0                    LDD #$01D0
 0007 EDFC                      STD $FC,X;CARGA APUNTADOR DE DIR BASE DE
                                         BUFER DE DISPLAY
 0009 6FFB                      CLR $FB,X
 000B 18CE0100                  LDY #$0100
 000F 8620                      LDAA #$20
 0011 18A7D0          WE:       STAA $D0,Y;CARGA DE BUFER DE DISPLAY CON
                                         CARACTERES ESPACIO
 0014 1808                      INY
 0016 188C0120                  CPY #$0120
 001A 26F5                      BNE WE
 001C 2070                      BRA SALIDAB

 001E A6FB            NOPP:     LDAA $FB,X
 0020 8100                      CMPA #$00
 0022 270A                      BEQ ESC80
 0024 8101                      CMPA #$01
 0026 270C                      BEQ ESCREN
 0028 8102                      CMPA #$02
 002A 270C                      BEQ ESCC0
 002C 2060                      BRA SALIDAB
 002E 8680            ESC80:    LDAA #$80
 0030 8D0C                      BSR ESCOM
 0032 205A                      BRA SALIDAB
 0034 8D1B            .ESCREN:  BSR ESCDAT
 0036 2056                      BRA SALIDAB
 0038 86C0            ESCC0:    LDAA #$C0
 003A 8D02                      BSR ESCOM
 003C 2050                      BRA SALIDAB
 003E 37              ESCOM:    PSHB
 003F 36                        PSHA
 0040 F61BA0          ASD:      LDAB $1BA0
 0043 C480                      ANDB #$80
 0045 2607                      BNE SALIDA1
 0047 B71BA0                    STAA $1BA0
 004A 8601                      LDAA #$01
 004C A7FB                      STAA $FB,X
 004E 32              SALIDA1:  PULA
 004F 33                        PULB
```

```
0050 39                          RTS
0051 183C          ESCDAT:       PSHY
0053 37                          PSHB
0054 36                          PSHA
0055 F61BA0        ASDD:         LDAB $1BA0
0058 C480                        ANDB #$80
005A 262D                        BNE SALIDA2
005C 1AEEFC                      LDY $FC,X
005F 18A600                      LDAA $00,Y
0062 B71BA1                      STAA $1BA1
0065 1808                        INY
0067 1AEFFC                      STY $FC,X
006A 188C01E0                    CPY #$01E0
006E 2715                        BEQ ACT02
0070 188C01F0                    CPY #$01F0
0074 2702                        BEQ ACT00
0076 2011                        BRA SALIDA2
0078 8600          ACT00:        LDAA #$00
007A A7FB                        STAA $FB,X
007C 18CE01D0                    LDY #$01D0
0080 1AEFFC                      STY $FC,X
0083 2004                        BRA SALIDA2
0085 8602          ACT02:        LDAA #$02
0087 A7FB                        STAA $FB,X
0089 32            SALIDA2:      PULA
008A 33                          PULB
008B 1838                        PULY
008D 39                          RTS
008E 01            SALIDAB:      NOP
```

CEN INCESPLM, EMPLEADO JUNTO CON EL CEN DESP, PARA HABILITAR EL MA DESP

INICIALIZADOR DE RAM CG DE DISPLAY AND 491, LOS CARACTERES ESPECIALES SON:
00-á, 01-é, 02-í, 03-ó, 04-ú, 05-¿, 06-¡, 07-Ñ

```
0000 A6F2                        LDAA $F2,X
0002 271A                        BEQ NOPP
INICIO DE INICIALIZACION DE RAM CG
0004 8640                        LDAA #$40
0006 8D18                        BSR ESCOM
0008 3C                          PSHX
0009 CE003A                      LDX #DATOS
000C 18CE0040                    LDY #$0040
0010 A600            JKL         LDAA $00,X
0012 8D19                        BSR ESCDAT
0014 08                          INX
0015 1809                        DEY
0017 26F7                        BNE JKL
0019 38                          PULX
    FIN DE INICIALIZACION DE RAM CG
001A 8680                        LDAA #$80
001C 8D02                        BSR ESCOM
001E 205A            NOPP        BRA DATOS+$40;SALTA A SIGUIENTE MODULO
```

```
0020 37                  ESCOM:     PSHB
□
0021 F61BA0              ASD:       LDAB $1BA0
0024 C480                           ANDB #$80
0026 26F9                           BNE ASD
0028 B71BA0                         STAA $1BA0
002B 33                             PULB
002C 39                             RTS
002D 37                  ESCDAT:    PSHB
002E F61BA0              ASDD:      LDAB $1BA0
0031 C480                           ANDB #$80
0033 26F9                           BNE ASDD
0035 B71BA1                         STAA $1BA1
0038 33                             PULB
0039 39                             RTS
003A 02040E010F110F00 DATOS: FCB $02,$04,$0E,$01,$0F,$11,$0F,$00,
0042 02040E111F100E00        FCB $02,$04,$0E,$11,$1F,$10,$0E,$00,
004A 02040C0404040E00        FCB $02,$04,$0C,$04,$04,$04,$0E,$00,
0052 02040E1111110E00        FCB $02,$04,$0E,$11,$11,$11,$0E,$00,
005A 0204111111130D00        FCB $02,$04,$11,$11,$11,$13,$0D,$00,
0062 0400040811110E00        FCB $04,$00,$04,$08,$11,$11,$0E,$00,
006A 0400000404040400        FCB $04,$00,$00,$04,$04,$04,$04,$00,
0072 0E11191513111100        FCB $0E,$11,$19,$15,$13,$11,$11,$00
```

3-6-6 Descripción del MA MANDESP, que permite observar para verificación, el texto asociado con módulos de tipo mensajero.

El PLM cuenta con un MA que permite observar cíclicamente el texto asociado con los módulos de tipo mensajero, que hayan sido declarados como parte del programa en SIIL1, correspondiente a una determinada aplicación, para habilitar esto este módulo debe declarase como parte del subprograma temporizado; así, al estarse ejecutando el programa y oprimirse sucesivamente el botón BAXD del BCLD se mostrarán, uno a la vez, los diferentes textos asociados con los módulos de tipo mensajero que se hubieren declarado, especificando el usuario el número máximo de módulo mensajero implicado, los textos irán apareciendo en orden ascendente, al oprimirse BAXD cuando se esté desplegando el mensaje cuyo módulo asociado tiene el valor máximo definido, se regresa al despliegue del mensaje correspondiente al módulo mensajero número cero, repitiéndose el ciclo siempre que el usuario continúe oprimiendo el botón BAXD.

El propósito de este MA es proporcionar al usuario un medio para verificar la operación de los diversos módulos de tipo mensajero, que el mismo este empleando en una determinada aplicación, una vez hecho esto el usuario podrá, si lo desea, eliminar del programa la declaración de este MA; la sintaxis para declarar a este módulo es:

MANDESP N;

Donde:

N representa el número máximo asociado con módulos de tipo mensajero que contenga el programa.

Descripción del CEN asociado con el MA MANDESP.

Para armar el código correspondiente al MA MANDESP, el software de traducción emplea un CEN denominado MANDESP, el tiempo máximo de ejecución del mismo es 26 μs, con fe = 2MHz; el flujo de ejecución correspondiente se muestra en la figura 3.65.

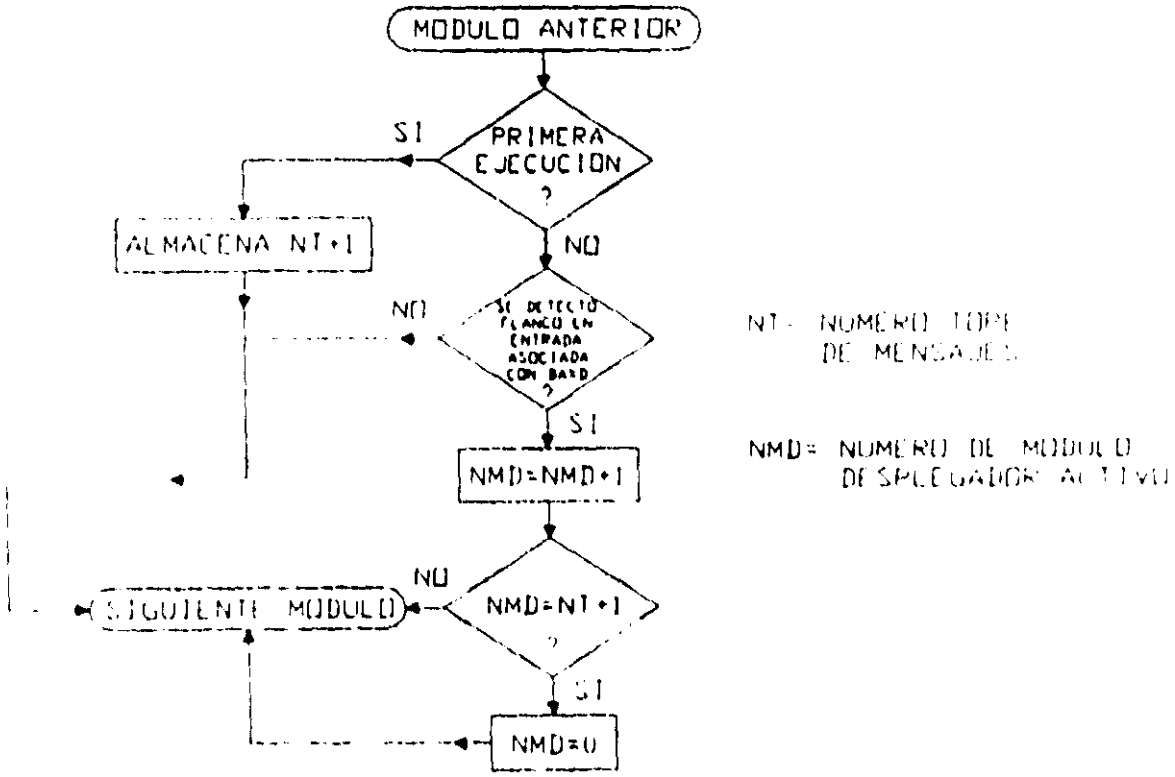


Figura 3.65 Flujo de ejecución del CEN MANDESP

El código correspondiente al CEN MANDESP se muestra a continuación

CODIGO DEL CEN MANDESP

```
0000 A6F1          LDAA $F1,X
0002 2706          BEQ NOPP
ACCIONAMIENTO EN LA PRIMERA EJECUCION (INICIALIZACION)
```

```
0004 860A                          LDAA #$0A;
0006 A7FF                          STAA $FF,X;ALMACENA NÚMERO MAXIMO DE
                                             MODULO MENSAJERO + 1
0008 201B                          BRA SIGBLO
  CHEQUEO DE FLANCO EN ENTRADA DE DISPARO
000A B61B90          NOPP:         LDAA $1B90;CARGA A CON VALOR PRESENTE DE
                                             GRUPO DE ENTRADA DE DISPARO
000D 8401                          ANDA #$01;ENMASCARA BIT PRESENTE DE
                                             DISPARO
□
000F E6CF                          LDAB $CF,X;CARGA VALOR ANTERIOR DE GRUPO
                                             DE ENTRADA DE DISPARO
□
0011 C401                          ANDB #$01;ENMASCARA BIT ANTERIOR DE
                                             DISPARO
□
0013 11                            CBA
□
0014 270F                          BEQ SIGBLO
□
0016 2D02                          BLT INCNUMOD;
□
0018 200B                          BRA SIGBLO
□
001A A6FE            INCNUMOD:     LDAA $FE,X
001C 4C                            INCA
001D A7FE                          STAA $FE,X
001F A1FF                          CMPA $FF,X;COMPARA A CON NUMERO MAXIMO
                                             DE MODULO MENSAJERO + 1
0021 2602                          BNE SIGBLO
0023 6FFE                          CLR $FE,X
0025 01              SIGBLO:       NOP
```

En la tabla 3.22 se muestra la TAB asociada con el MA MANDESP.

Tabla 3.22 Asignación de bytes asociada con el CEN MANDESP, empleado para obtener el código requerido por el MA MANDESP, que habilita el despliegue cíclico de los textos asociados con los módulos de tipo mensajero empleados en una aplicación.

Valor numérico (VN) generado por el software de traducción	Byte en CEN MANDESP, al cual se le asignará el valor VN, al obtenerse el código objeto asociado con el MA MANDESP.
VN= número tope de módulo mensajero + 1	B5

3-7 EJEMPLO DE PROGRAMACIÓN.

Con el objeto de aclarar ideas acerca de lo tratado en este capítulo, se muestra aquí un ejemplo de programación en SIIL1, cuyo propósito es meramente didáctico, los módulos que contiene no son parte de los requerimientos de un determinado control lógico y su inclusión en el programa, es sólo con el propósito de ilustrar la potencialidad del PLM, para realizar bloques funcionales de utilidad en la automatización de procesos.

Cabe recordar aquí que un programa en SIIL1, está estructurado no solo por las declaraciones que correspondan a los módulos lógicos y auxiliares que el mismo contenga, sino también por los comandos: CONFIGN, INPROG, FINPP, INMODI, y FINMODI; que especifican la configuración de funcionamiento y delimitan a los subprogramas principal y temporizado del mismo, véase la sección 1-5-2 de esta tesis.

Ejemplo 3.18

Se desea realizar con el PLM los siguientes cuatro bloques funcionales:

1.- Función lógica combinacional de cuatro entradas, siendo las mismas las VBE E30, E31, E32 y E33, la salida del bloque debe ser la VBS S17 y deberá verificarse en alto para los mintérminos de entrada 0000, 0110, y 1111.

En la figura 3.66 se ilustra un posible arreglo de compuertas que pueden realizar la función combinacional aquí requerida, nótese el empleo de variables booleanas intermediarias

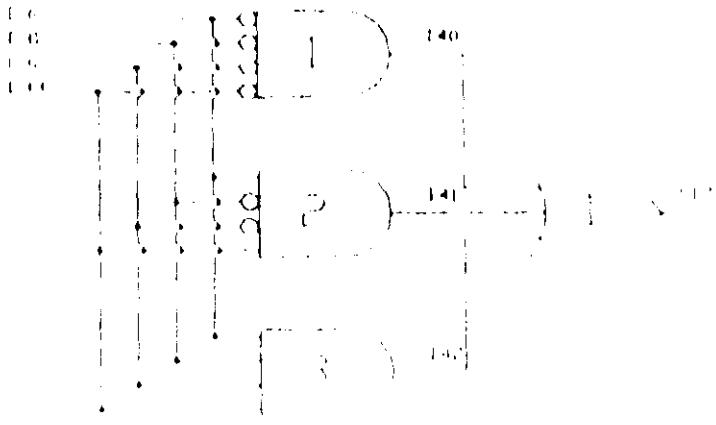


Figura 3.66 Arreglo de compuertas lógicas y variables asociadas, para realizar la función combinacional del ejemplo 3.18

A continuación se muestra el tramo de código SIIL1, asociado con el arreglo de compuertas mostradas en la figura 3.66, véanse las secciones 3-5-4 y 3-5-5 de esta tesis.

```
AND4#1 E30,E31,E32,E33,I40,0000;
AND4#2 E30,E31,E32,E33,I41,0110;
AND4#3 E30,E31,E32,E33,I42,1111;
  OR3#1 I40,I41,I42,S17,111;
```

2.- Sistema de mensajes de alarma que opere de la siguiente forma:

No debe haber texto fijo, el texto móvil debe apreciarse en el primer renglón de la UD a partir de la primera columna, con un tamaño de ventana de diez caracteres y una cadencia de 30 centésimas de segundo entre posiciones sucesivas del mismo, los mensajes de alarma deseados son tres y deben activarse al tenerse un nivel alto en las entradas E23 y E12 y cada que falten quince minutos para la hora en el RTR; a continuación se describe el texto deseado para cada uno de ellos.

Al oprimirse un botón normalmente abierto ligado con la entrada E23, deberá aparecer en la UD el siguiente mensaje móvil: "Se ha oprimido el botón ligado con la entrada E23 del PLM".

Al oprimirse un botón normalmente abierto ligado con la entrada E12, deberá aparecer en la UD el siguiente mensaje móvil: "Se ha oprimido el botón ligado con la entrada E12 del PLM".

Cada que falten quince minutos para la hora en el RTR, deberá aparecer por veinte segundos en la UD, el siguiente texto móvil: "Faltan quince minutos para la hora."

Se supone que el orden de prioridad de los mensajes debe ser el que se uso para describirlos en los párrafos anteriores, siendo el mensaje de no condición de alarma el siguiente: "Operación normal, no se ha dado ninguna condición de alarma."; en la figura 3.67 se ilustra un posible arreglo de módulos que puede emplearse para realizar el sistema de alarmas aquí requerido, nótese el empleo de dos temporizadores, uno de tipo GPRTR (TEMPOB#1) y el otro de tipo monodisparo (TEMPOC#3) además del empleo de módulos de tipo mensajero y alarma

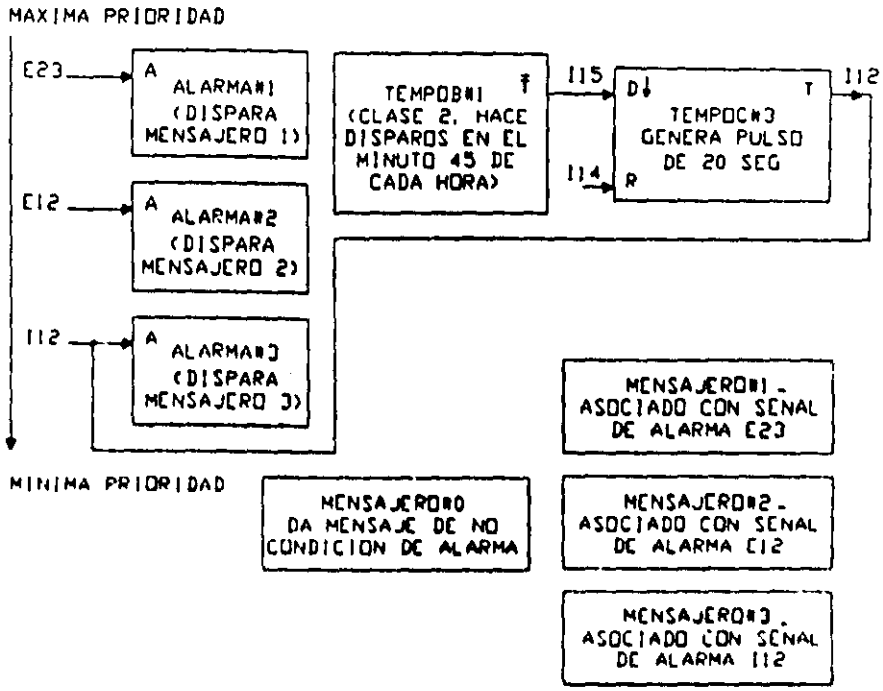


Figura 3.67 Arreglo de módulos empleado para realizar el bloque funcional dos (sistema de alarmas) del ejemplo 3.18.

Si a cada módulo de tipo alarma, se le asigna como número de mensajero asociado, el valor del orden de prioridad, el código asociado con el sistema de alarmas requerido, que debe ser parte del subprograma principal, es el que se muestra a continuación, véanse las secciones 3-5-14 y 3-6-2 de esta tesis.

```
    ALARMA#1 E23, 1; Prioridad 1
    ALARMA#2 E12, 1; Prioridad 2
    ALARMA#3 I12, 1; Prioridad 3
    TEMPOB#1 I15, 2, 1, 0;
## 45:00;
```

A continuación se muestra el código asociado con el sistema de alarmas, que debe ser parte del subprograma temporizado, véanse las secciones 3-5-10 y 3-6-1 de esta tesis.

```
   MENSAJERO#2 "", 1, 10, 30, 1001; Mensaje asociado con la señal de
;                                   alarma E12.
# Se ha oprimido el botón ligado con |
## la entrada E23 del PLM.
   MENSAJERO#1 "", 1, 10, 30, 1001; Mensaje asociado con la señal de
;                                   alarma E23.
# Se ha oprimido el boton ligado con la |
## entrada E23 del PLM.
```

```
   MENSAJERO#3 "", 1, 10, 30, 1001; Mensaje asociado con la señal de
;                                   alarma I12.
## Faltan quince minutos para la hora.
MENSAJERO#0 "", 1, 10, 30, 1001; Mensaje testigo de no condición de
;                                   alarma.
# Operación normal, no se ha dado ninguna condición de alarma.
              TEMPOC#3 I15,I14,I12,00:00:20.00,001;
```

Nótese en la declaración del temporizador monodisparo empleado (TEMPOC#3) el que la entrada de RESET se verifica en alto y es la VBI I14, que no es empleada como salida de ningún otro módulo en el programa, esto hace que la misma permanezca en cero todo el tiempo, apareciendo para el usuario final este temporizador, como uno que tuviera solo una entrada activa (disparo).

Cabe recordar aquí, que cuando inicia la ejecución de un programa en el PLM, todas las variables booleanas son puestas a cero; por lo tanto, cualquier VBI o VBS permanecerá en cero todo el tiempo, siempre que la misma no sea empleada como salida de algún módulo del programa; este hecho puede ser de utilidad cuando se requiera tener una VB de referencia con un nivel permanente de cero lógico.

3.- Temporizador astable con periodo de un segundo y arranque en cero, con pulsos de 250 ms; la entrada de restablecimiento a este bloque debe ser la VBE E02, con nivel de verificación en bajo; la salida debe ser la VBS S05; en la figura 3.68 se muestra una posible implantación de este bloque, que emplea solamente a un módulo lógico de tipo temporizador astable, véase la sección 3-5-12 de esta tesis, la sintaxis correspondiente es:

```
TEMPOE#2 E02,S05,00 00 01.00,00 00 00.10,10;
```

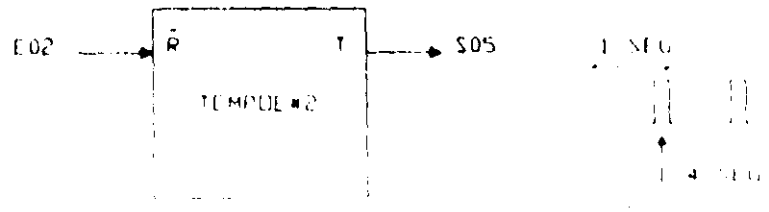


Figura 3.68 Módulo lógico que realiza el bloque funcional tres del ejemplo 3.18

4.- Bloque funcional que genere en la salida S03, una señal repetitiva cuyo periodo básico sea el mostrado en la figura 3.69, con una entrada de RESET verificada en bajo siendo la misma la VBE E03.

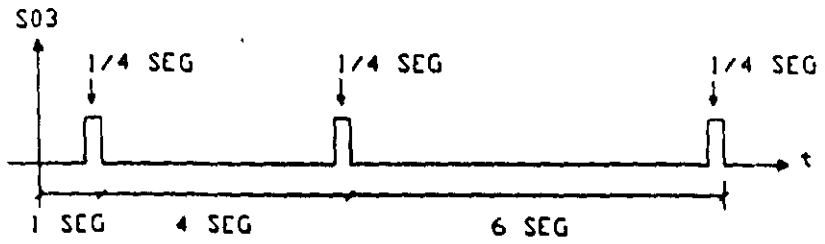


Figura 3.69 Periodo básico de la señal repetitiva que debe aparecer en la salida S03 al activarse en el PLM el bloque funcional 4 del ejemplo 3.18.

Este bloque funcional puede ser implantado con un arreglo de dos módulos lógicos interconectados, uno es un temporizador multipulso (TEMPOG#4) y el otro es un temporizador monodisparo (TEMPOC#5), el primero genera los tres pulsos requeridos en cada periodo básico de la señal requerida en la salida S03, el segundo genera un pulso de 0.5 seg verificado en bajo, cada vez que el temporizador multidisparo ha completado un periodo de la señal a generar, este pulso restablece al primer temporizador, repitiéndose esto indefinidamente, véase la figura 3.70; la sintaxis asociada con los módulos de este bloque funcional, véanse las secciones 3-5-10 y 3-5-13, es la que a continuación se muestra:

```
            TEMPOG#4 I01,I01,S03,I00,3,00:00:00.50,I0010;
##           01.00,04.00,06.00;

            TEMPOC#5 I00,E03,I01,00:00:00.50,110;
```

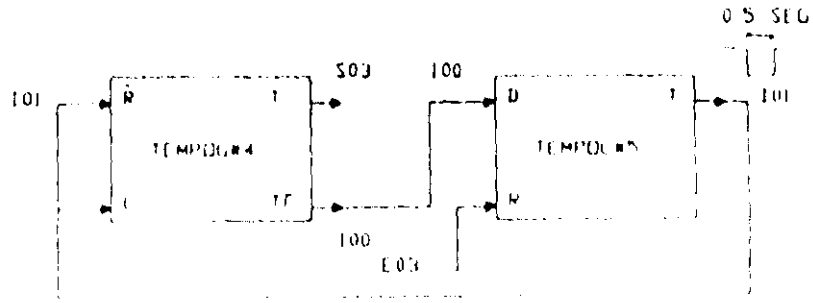


Figura 3.70 Arreglo de módulos lógicos interconectados para realizar el bloque funcional número cuatro del ejemplo 3.18.

El programa en SIIL1 asociado con este ejemplo, se denomina PEJ318 y para ser procesado por el software de traducción, debe ser guardado en un archivo de texto que podría ser denominado como PEJ318.SIL, la secuencia de sentencias que lo integran se muestra a continuación, mostrándose en negritas las declaraciones asociadas; nótese el empleo de los caracteres ";" y "*", que pueden ser empleados para delimitar comentarios.

```
******* PROGRAMA PEJ318 CORRESPONDIENTE AL EJEMPLO 3.18 ***************
*                    POR ANTONIO SALVÁ CALLEJA                          *
*                           JULIO DE 1998                               *


        CONFIG1; COMANDO QUE ESPECIFICA CONFIGURACIÓN DE FUNCIONAMIENTO
;                DEL PLM.
    INPROG; COMANDO QUE DELIMITA EL INICIO DEL PROGRAMA (INICIO DEL SPP).

* INICIO DE SENTENCIAS ASOCIADAS CON EL BLOQUE FUNCIONAL DOS,           *
* QUE DEBEN SER PARTE DEL SUBPROGRAMA PRINCIPAL                         *
              ALARMA#1 E23, 1; Prioridad 1
              ALARMA#2 E12, 1; Prioridad 2
              ALARMA#3 I12, 1; Prioridad 3
              TEMPOB#1 I15, 2, 1, 0;
## 45:00;
* FIN DE SENTENCIAS ASOCIADAS CON EL BLOQUE FUNCIONAL DOS,             *I
* QUE DEBEN SER PARTE DEL SUBPROGRAMA PRINCIPAL                        *

* INICIO DE SENTENCIAS ASOCIADAS CON EL BLOQUE FUNCIONAL UNO           *
    AND4#1 E30,E31,E32,E33,I40,0000;
    AND4#2 E30,E31,E32,E33,I41,0110;
    AND4#3 E30,E31,E32,E33,I42,1111;
      OR3#1 I40,I41,I42,S17,111;
* FIN DE SENTENCIAS ASOCIADAS CON EL BLOQUE FUNCIONAL UNO               *

          DESP; DECLARACIÓN DE MÓDULO AUXILIAR DESP,QUE HABILITA A LA
;               UD.
         FINPP; DECLARACIÓN DELIMITADORA DEL FIN DEL SUBPROGRAMA
;
        INMODI; DECLARACIÓN DELIMITADORA DEL INICIO DEL SUBPROGRAMA
;                           TEMPORIZADO

* INICIO DE SENTENCIAS ASOCIADAS CON EL BLOQUE FUNCIONAL DOS,           *
* QUE DEBEN SER PARTE DEL SUBPROGRAMA TEMPORIZADO                       *
    MENSAJERO#2 "", 1, 10, 30, 1001; Mensaje asociado con la señal de
;                                    alarma E12.
# Se ha oprimido el botón ligado con |
## la entrada E23 del PLM.
    MENSAJERO#1 "", 1, 10, 30, 1001; Mensaje asociado con la señal de
;                                    alarma E23.
```

```
# Se ha oprimido el botón ligado con la |
## entrada E23 del PLM.
    MENSAJERO#3 "", 1, 10, 30, 1001; Mensaje asociado con la señal de
;                                    alarma I12.
## Faltan quince minutos para la hora.
    MENSAJERO#0 "", 1, 10, 30, 1001; Mensaje testigo de no condición de
;                                    alarma.
## Operación normal, no se ha dado ninguna condición de alarma.
; AQUI SIGUE DECLARACIÓN DE TEMPORIZADOR MONODISPARO,ACTIVADO POR EL
; TEMPORIZADOR TIPO GPRTR NÚMERO UNO (TEMPOB#1), EL CUAL A SU VEZ ACTIVA
; EL MENSAJE DE ALARMA QUE INDICA QUE FALTAN QUINCE MINUTOS PARA LA HORA.
        TEMPOC#3 I15,I14,I12,00:00:20.00,001;
* FIN DE SENTENCIAS ASOCIADAS CON EL BLOQUE FUNCIONAL DOS,              *
* QUE DEBEN SER PARTE DEL SUBPROGRAMA TEMPORIZADO                       *

* DECLARACIÓN DE MÓDULO LÓGICO ASOCIADO CON EL BLOQUE FUNCIONAL TRES    *
        TEMPOE#2 E02,S05,00 00 01.00,00 00 00.10,10;

* INICIO DE SENTENCIAS ASOCIADAS CON EL BLOQUE FUNCIONAL CUATRO         *
      TEMPOG#4 I01,I01,S03,I00,3,00:00:00.50,10010;
##          01.00,04.00,06.00;

          TEMPOC#5 I00,E03,I01,00:00:02.10,110;
* FIN DE SENTENCIAS ASOCIADAS CON EL BLOQUE FUNCIONAL CUATRO            *

      MANDESP 3;  DECLARACIÓN DEL MÓDULO AUXILIAR MANDESP,QUE PERMITE AL
;                 USUARIO, PROBAR UNO A UNO LOS CUATRO MÓDULOS MENSAJERO
;                 DE LA APLICACIÓN (0,1,2,3), OPRIMIENDO SUCESIVAMENTE EL
;                 BOTÓN BAXD DEL BCLD.
       FINMODI; DECLARACIÓN QUE DELIMITA EL FINAL DEL SUBPROGRAMA
;               TEMPORIZADO.
```

En la figura 3.71, se muestra un posible conexionado al PLM, para probar el funcionamiento de los cuatro bloques funcionales de este ejemplo.

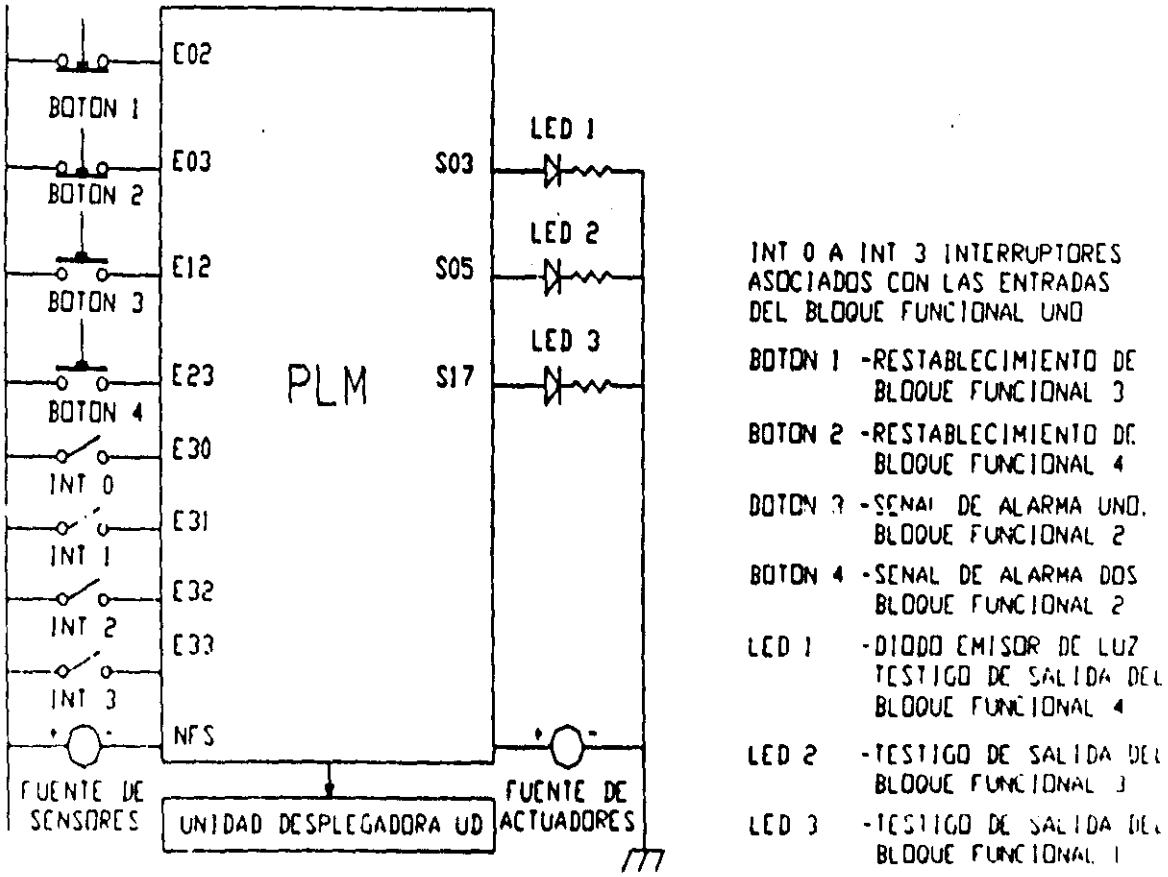


Figura 3.71 Conexionado al PLM para verificar el funcionamiento de los cuatro bloques funcionales del ejemplo 3.18, al ejecutarse en el mismo el programa objeto correspondiente al programa fuente en SIIL1 PEJ318.SIL.

CAPÍTULO 4
SOFTWARE DE TRADUCCIÓN Y DESARROLLO

Este capítulo inicia con una descripción simplificada de la forma en que el software de traducción efectúa la generación del código objeto asociado con un determinado programa fuente, para pasar luego a describir la razón de ser y operación básica del código que se adiciona al programa objeto generado a partir de las declaraciones del usuario, de manera que el PLM pueda recibir comandos de monitoreo desde la computadora anfitriona, esto permite observar en tiempo real en la pantalla de la PC la evolución de diversas variables del PLM, al ejecutar el mismo un programa; el capítulo finaliza con la descripción general de un sistema para desarrollo con el PLM que corre en ambiente windows y para el cual a la fecha se cuenta con una versión preliminar.

4-1 SOFTWARE DE TRADUCCIÓN DE UN PROGRAMA FUENTE

Para ejecutar un programa en SIIL1 en el PLM, el mismo debe ser traducido a código ejecutable por la CC, en principio esto se efectúa ejecutando en la computadora anfitriona (PC) un programa denominado SIIL1.EXE que corre bajo MSDOS, al hacerse esto en primera instancia la PC pide el nombre del archivo de texto que contiene el código fuente a traducir, el cual debe tener la extensión SIL, pasando a procesarse el archivo renglón por renglón, identificándose en cada caso el tipo de instrucción (declaración de módulo o comando) que el mismo contiene y procediéndose en consecuencia, generando el código objeto que corresponda, esto de acuerdo con el CEN que se requiera, llenándose los bytes que sea necesario en concordancia con la TAB asociada con los operandos que la instrucción pudiera contener, en caso de que hubiera un error de sintaxis, la PC emite un pitido y muestra en pantalla un número que denota el tipo de error y el renglón donde el mismo se produjo, incrementándose un contador de errores (NE), procediéndose después de esto con el procesamiento del siguiente renglón del programa hasta la detección del comando que denota el final del subprograma temporizado (FINMODI).

El código asociado con cada renglón se va colocando sucesivamente en una zona reservada de memoria RAM en la PC, desde luego que de acuerdo a los comandos que el programa pudiera contener, se crean los tramos de código correspondientes, por ejemplo, al detectarse en un renglón el comando que denota el final del subprograma

principal, el software de traducción coloca un tramo de código (TC) obtenido a partir del CEN SALTO descrito en el capítulo anterior, asignándose ahí el valor del par de bytes que denotan la dirección donde se origina el TC que copia al buffer asociado en RAM de la CC del PLM, el estado de los puertos de entrada físicos; este último TC seguramente fue obtenido a partir del CEN LECBUF3, que obviamente ya ha sido colocado por el software de traducción; para una mejor comprensión de lo descrito en este párrafo se puede ver la figura 3.1 y el texto explicativo alrededor de ella.

Después que el software de traducción ha colocado el TC asociado con el comando FINMODI, se procede a colocar el último TC del programa, el cual se obtiene a partir de un CEN denominado SMT6 que será descrito más adelante en este capítulo.

El TC mencionado en el párrafo anterior se denominará de aquí en adelante como código de monitoreo (CM), conteniendo el mismo lo siguiente:

1.- Subrutina de monitoreo (SUBMON), con código que determina si se ha recibido vía serie desde la computadora anfitriona, algún comando de monitoreo, que pudiera ser el requerimiento al PLM de información tal como: El estado de un grupo de entradas, la cuenta asociada con un contador de eventos, la colocación de una determinada hora en el RTR, etc. Tal subrutina es invocada antes de ejecutar el código del TC SALTO en el lazo del subprograma principal, véase la figura 3.1, para que esto se logre el software de traducción coloca tres instrucciones NOP (tres bytes 01H) inmediatamente antes de colocar el código del TC SALTO, guardando la dirección donde estos bytes fueron colocados, para que posteriormente, una vez que se ha colocado el código que se requiere para finalizar el subprograma temporizado, cambiar los NOP's por el código de una instrucción de salto a subrutina cuya dirección de colocación será la que corresponde a el último byte del subprograma temporizado más uno, que será la dirección a partir de la cual se coloca el código de monitoreo que inicia con la SUBMON.

2.- Subrutina de servicio de interrupción por recepción que recibe los comandos de monitoreo y los coloca en el buffer de recepción serie (BR), los datos en el mismo son procesados por la SUBMON.

3.- Subrutina de interrupción que transmite a la computadora anfitriona la información que la misma pudiera requerir, la cual está contenida en el buffer de transmisión serie (BT), el cual es actualizado en caso necesario, por el código de la SUBMON, con la información requerida.

Un aspecto importante que se logra gracias al código de monitoreo, es el poder

monitorear en tiempo real desde la PC, la operación del PLM al ejecutar el mismo un determinado programa, la interfaz de software para hacer esto es parte de un sistema para desarrollo con el PLM (SPDPLM) que corre en ambiente windows; de hecho, esto último a fin de cuentas sería lo que emplearía el usuario final para desarrollar aplicaciones con el PLM; la forma de operar el SPDPLM y algunos aspectos relacionados se tratará más adelante en este capítulo.

Así, el código objeto correspondiente a un programa fuente en SIIL1, estará conformado por una lista de bytes integrada por tres componentes que son: Código del subprograma principal, código del subprograma temporizado y código de monitoreo, véase la figura 4.1.

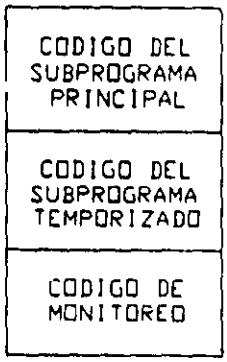


Figura 4.1 Conformación del código objeto correspondiente a un determinado programa en SIIL1.

Una vez que se ha completado el armado del código objeto en la memoria RAM de la computadora anfitriona se procede, en caso de que no haya habido errores de sintaxis, a generar un archivo binario con extensión BLM, que contendrá a la lista hexadecimal mostrada en la figura 4.1, además de el valor de las direcciones inicial y final de colocación del programa en la memoria de la CC del PLM, tal archivo es compatible con el manejador hexadecimal PUMMA, que es la herramienta para desarrollo propia de la CMT SIMMP2, la cual se empleó en primera instancia para bajar a la CC del PLM programas objeto generados por el software de traducción, véase el apéndice A.

En caso de que la cuenta de errores de sintaxis sea diferente de cero, no se genera el archivo binario mencionado en el párrafo anterior, quedando en pantalla la lista de errores encontrados, para que el usuario los corrija empleando un editor ASCII convencional.

Cabe señalar aquí, que en un principio tanto el lenguaje SIIL1 como el software

de traducción eran solamente una idea, por lo que el proceso que se siguió para llegar a tener al SPDPLM fué el siguiente:

1.- Armado "manual" del código objeto de programas pequeños en SIIL1, empleando para ello los CEN y TAB's descritos en el capítulo 3 y al editor hexadecimal contenido en el manejador hexadecimal PUMMA, esto fue laborioso, sin embargo con estos primeros ejemplos de programas en SIIL1 ejecutándose en el PLM se confirmó la viabilidad de la idea original.

2.- Desarrollo del software de traducción (programa SIIL1.EXE), que hace automáticamente lo descrito en el párrafo anterior, probandose además en ambiente MSDOS el código de monitoreo, del que se ha esbozado su funcionamiento en párrafos anteriores.

Cabe señalar aquí que con el programa SIIL1.EXE, el editor de MSDOS y el manejador hexadecimal PUMMA se puede desarrollar cualquier aplicación; sin embargo, dado que es conveniente el tener todas las facilidades para desarrollo en un solo sistema se trabajó en el diseño de una versión preliminar de esto último, que es el SPDPLM.

3.- Desarrollo de una versión preliminar del SPDPLM, apoyándose en el ejecutable mencionado en el párrafo anterior con algunas adecuaciones, y en el lenguaje de programación Visual Basic versión 4. En este sistema se tienen integradas ya varias de las facilidades del manejador PUMMA, que son necesarias para desarrollar aplicaciones con el PLM, una facilidad que a la fecha no se ha integrado al SPDPLM es el software de programación de la EPROM, de modo que para validar una aplicación que corra en forma autónoma, por el momento se tiene que emplear el manejador PUMMA.

4-1-1 Estructura del software de traducción

La primera versión del software de traducción (programa SIIL1.EXE), se desarrolló bajo MSDOS, y su operación se describió a grandes rasgos en párrafos anteriores, en esta sección se describe globalmente su funcionamiento. Los principales pasos al ejecutarse el software de traducción son los siguientes.

1.- Reserva en RAM de segmento de 64k, para armado de programa objeto, y carga a partir del origen del mismo, archivo binario (CODGEN32.BLM) con los CEN asociados con todos los módulos y comandos que maneja el lenguaje SIIL1.

2.- Inicializa a cero contadores de renglón (CRN) y error (NE).

3.- Abre archivo con programa fuente.

4.- Pregunta si se llegó al final del archivo, si este es el caso salta al paso trece.

5.- Toma siguiente renglón del archivo y le suma uno al contador de renglón.

6.- Pregunta si el renglón está conformado únicamente por caracteres espacio, en cuyo caso salta al paso cuatro.

7.- Inicializa a cero variable testigo de tipo de error (TESE).

8.- Invoca subrutina maestra de generación y colocación de código en RAM de PC.

9.- Pregunta si la variable TESE retornó con un valor nulo, en cuyo caso salta al paso cuatro.

10.- Muestra en pantalla el renglón donde se produjo el error y los valores de las variables TESE y CRN.

11.- Suma uno al contador de errores.

12.- Salta al paso cuatro.

13.- Cierra archivo con programa fuente.

14.- Coloca código de monitoreo.

15.- Pregunta si el contador de errores (NE) está en cero, en cuyo caso genera archivo binario con el código objeto colocado en la RAM de la PC.

16.- Finaliza la ejecución.

En la figura 4.2 se ilustra un diagrama de flujo del software de traducción.

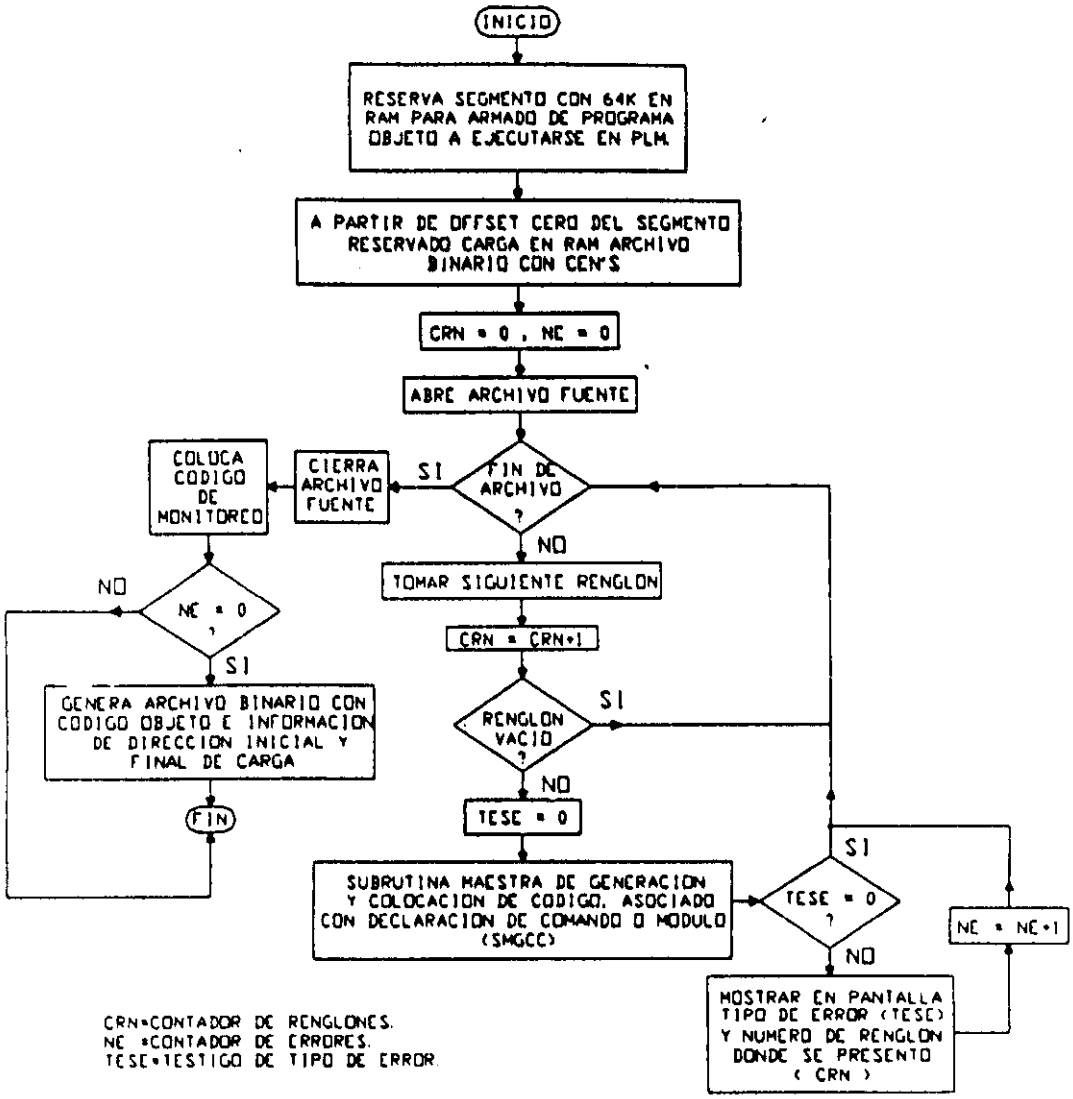


Figura 4.2 Diagrama de flujo del software de traducción.

El archivo binario que contiene todos los CEN que emplea el software de traducción se denomina CODGEN32.BLM y la colocación de cada tramo de código esqueleto en el mismo se muestra en la tabla 4.1; para detalles acerca de los CEN puede consultarse en el capítulo 3, lo concerniente con cada uno de ellos.

Tabla 4.1 Colocación de CEN's en archivo binario CODGEN32.BLM.

Nombre del CEN	Intervalo de direcciones de colocación en decimal
BUFUN	0 – 14
NOTUN	15 – 29
OPER2UN	30 – 73
OPER3UN	74 – 126

Continuación Tabla 4.1

OPER4UN	127 – 188
TEMPOA	189 – 284
TEMPOC	285 – 363
TEMPOD	364 – 429
TEMPOE	430 – 503
TEMPOG	637 – 775
SEC8XN	776 – 984
SEC7XN	985 – 1178
SEC6XN	1179 – 1357
SEC5XN	1358 – 1521
SEC4XN	1522 – 1670
SEC3XN	2301 – 2434
SEC2XN	1671 – 1789
SEC1XN	1790 – 1893
CONTA	1894 – 1966
INPLM2	2089 – 2179
ESCBUF1	2180 – 2191
ESCBUF2	2192 – 2197
LECBUF1	2198 – 2221
LECBUF2	2222 – 2227
SALTO	2228 – 2237
ENRUS	2238 – 2254
RETRUS	2255 – 2257
ACT	2258 – 2270
FFARS	2271 – 2300
DESP	2435 – 2577
MENSAJE1	2578 – 2708
INCESPLM	2709 – 2830
ACTBOT	2831 – 2840
MANDESP1	2841 – 2893

Continuación Tabla 4.1

INPLM3	2894 – 2974
LECBUF3	2975 – 2996
RTRX	2997 – 3883
TEMPOB1	3884 – 4033
TEMPOB2	4034 – 4194
TEMPOB3	4195 – 4366
TEMPOB4	4367 – 4549
TEMPOB5	4550 – 4743
TEMPOB6	4744 – 4948
OBSCED1	4949 – 5144
OBSCED2	5145 – 5340
OBSCED3	5341 – 5536
OBSCED4	5537 – 5732
OBSCED5	5733 – 5928
ALARMA	5929 – 5967
SMT6	5968 – 6566

La subrutina maestra de generación y colocación de código (SMGCC), es una componente importante del software de traducción, la operación global de la misma se describe en la siguiente sección.

4-1-2 Operación de la subrutina maestra de generación y colocación de código (SMGCC)

Esta subrutina es invocada por el software de traducción como se ha descrito en párrafos anteriores, su función consiste en identificar el módulo o comando declarado en el renglón del programa fuente que se esté procesando, para proceder a generar el código objeto requerido empleando para ello al CEN apropiado en cada caso, haciéndose una separación en cuanto al hecho de si el comando es o no de inicialización, empleando para ello a una variable testigo denominada "TESDIR" la cual inicia con un valor cero y se coloca en uno cuando el software de traducción ha detectado el comando INPROG que denota propiamente el inicio del programa, a la fecha los únicos comandos de inicialización son los que denotan la configuración de funcionamiento (CONFIF1, CONFIG2 y CONFIG3) y el inicio del programa

(INPROG).

Los pasos al ejecutarse la subrutina SMGCC son los siguientes:

1.- Obtiene en la cadena INSC$, todo lo que haya la izquierda del caracter ";" en el renglón que se esté procesando.

2.- Convierte a mayúsculas todos los caracteres que fueran letras del alfabeto en la cadena INSC$.

3.- Se pregunta si la variable TESDIR vale uno, si este es el caso se pasa al paso seis.

4.- Se invoca la subrutina reconocedora de comandos de inicialización (SRCI).

5.- Termina la ejecución de la SMGCC con un retorno de subrutina.

6.- Desdobla la cadena INSC$ en el par de cadenas ETIQ$ e INSTRU$, siendo ETIQ$ la cadena que inicia en la primera columna del renglón y finaliza en el primer caracter espacio encontrado, en caso de que el caracter en la primera columna sea un espacio, ETIQ$ será una cadena vacia; la cadena INSTRU$ inicia con el primer caracter no espacio, detectado después del espacio que delimitó el final de la cadena ETIQ$.

7.- Se invoca subrutina de generación de código (SGC).

8.- Se pasa al paso cinco.

En la figura 4.3 se muestra un diagrama de flujo que ilustra el flujo de ejecución de la subrutina maestra de generación y colocación de código (SMGCC).

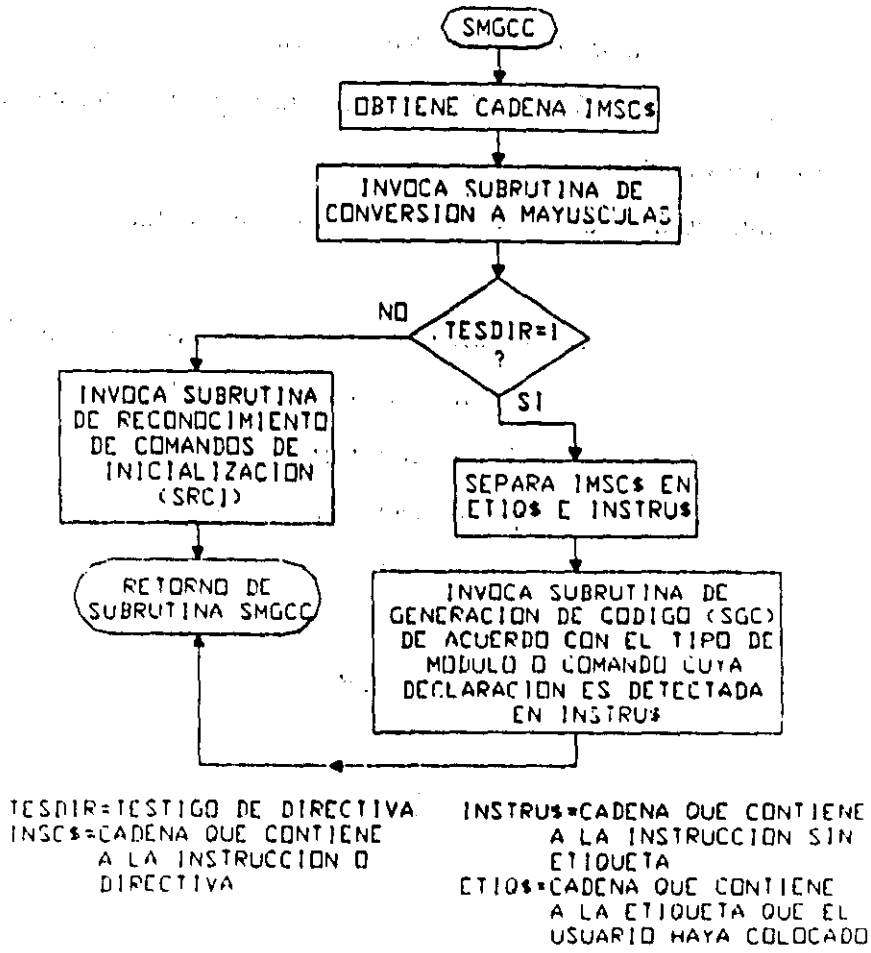


Figura 4.3 Flujo de ejecución de la subrutina maestra de generación y colocación de código (SMGCC).

4-1-3 Operación de la subrutina reconocedora de comandos de inicialización (SRCI)

Al llevarse a cabo la traducción de un programa fuente, el software de traducción supone de origen que las declaraciones contenidas en cada renglón, son comandos de inicialización, esto hasta que se detecte el comando INPROG, la subrutina que procesa los comandos de inicialización se denomina SRCI y los pasos genéricos que se siguen por la misma son:

1.- Rescata en la cadena COIN$, el contenido de la cadena INSC$, libre de caracteres espacio.

2 - Pregunta si la cadena COIN$ es igual al comando CONFIG1, en cuyo caso salta al paso ocho, (reconocimiento de CONFIG1).

3.- Pregunta si la cadena COIN$ es igual al comando CONFIG2, en cuyo caso salta al paso diez, (reconocimiento de CONFIG2).

4.- Pregunta si la cadena COIN$ es igual al comando CONFIG3, en cuyo caso salta al paso doce, (reconocimiento de CONFIG3).

5.- Pregunta si la cadena COIN$ es igual al comando INPROG, en cuyo caso salta al paso INPROG.

6.- Asigna el valor 15 (comando de inicialización desconocido), a la variable testigo de error TESE.

7.- Retorna de la subrutina SRCI.

8.- Efectua las siguientes asignaciones para variables que emplea el software de traducción:

DIRINP = C000H, (dirección inicial de colocación del programa objeto en la CC del PLM).

DIRINPU = DIRINP + 81, (dirección de colocación del primer tramo de código del lazo del subprograma principal).

TOPMEM = DFFFH, (tope de memoria RAM disponible para cargar programa objeto).

BUFEN = 0, (offset sobre la dirección 100H, que marca el origen del buffer de entrada).

BUFSA = 8, (offset sobre la dirección 100H, que marca el origen del buffer de salida).

BUFI = 12, (offset sobre la dirección 100H, que marca el origen del buffer de variables intermediarias).

IEMAX = 3, (número máximo para un grupo de variables de entrada).

ISMAX = 1, (número máximo para un grupo de variables de salida).

IIMAX = 20, (número máximo para un grupo de variables de intermediarias).

9.- Pasa al paso siete.

10.- Efectua las siguientes asignaciones para variables que emplea el software de traducción:

DIRINP = C000H, (dirección inicial de colocación del programa objeto en la CC del PLM).

DIRINPU = DIRINP + 81, (dirección de colocación del primer tramo de código del lazo del subprograma principal).

TOPMEM = DFFFH, (tope de memoria RAM disponible para cargar programa objeto).

BUFEN = 0, (offset sobre la dirección 100H, que marca el origen del buffer de entrada).

BUFSA = 2, (offset sobre la dirección 100H, que marca el origen del buffer de salida).

BUFI = 4, (offset sobre la dirección 100H, que marca el origen del buffer de variables intermediarias).

IEMAX = 0, (número máximo para un grupo de variables de entrada).

ISMAX = 0, (número máximo para un grupo de variables de salida).

IIMAX = 20, (número máximo para un grupo de variables de intermediarias).

11.- Pasa al paso siete.

12.- Efectua las siguientes asignaciones para variables que emplea el software de traducción:

DIRINP = C000H, (dirección inicial de colocación del programa objeto en la CC del PLM).

DIRINPU = DIRINP + 81, (dirección de colocación del primer tramo de código del lazo del subprograma principal).

TOPMEM = DFFFH, (tope de memoria RAM disponible para cargar programa objeto).

BUFEN = 0, (offset sobre la dirección 100H, que marca el origen del buffer de entrada).

BUFSA = 8, (offset sobre la dirección 100H, que marca el origen del buffer de salida).

BUFI = 12, (offset sobre la dirección 100H, que marca el origen del buffer de variables intermediarias).

IEMAX = 3, (número máximo para un grupo de variables de entrada).

ISMAX = 1, (número máximo para un grupo de variables de salida).

IIMAX = 20, (número máximo para un grupo de variables de intermediarias).

13.- Pasa al paso siete.

14.- Asigna el valor uno a la variable TESDIR, (testigo de que ya se reconoció el comando de inicialización INPROG).

15.- Asigna el valor de la variable DIRINP a la variable DIRBM, que especifica la dirección de colocación del siguiente módulo, (DIRBM = DIRINP).

16.- Coloca tramo de código INPLM3.

17.- Actualiza la variable DIRBM sumandole 81, que es la longitud del tramo de código colocado en el paso anterior.

18.- Coloca el tramo de código LECBUF3.

19 Actualiza la variable DIRBM sumandole 22, que es la longitud del tramo de código colocado en el paso anterior.

20.- Pasa al paso siete.

En la figura 4.4 se muestra un diagrama de flujo que ilustra la secuencia de ejecución de la subrutina SRCI aquí descrita.

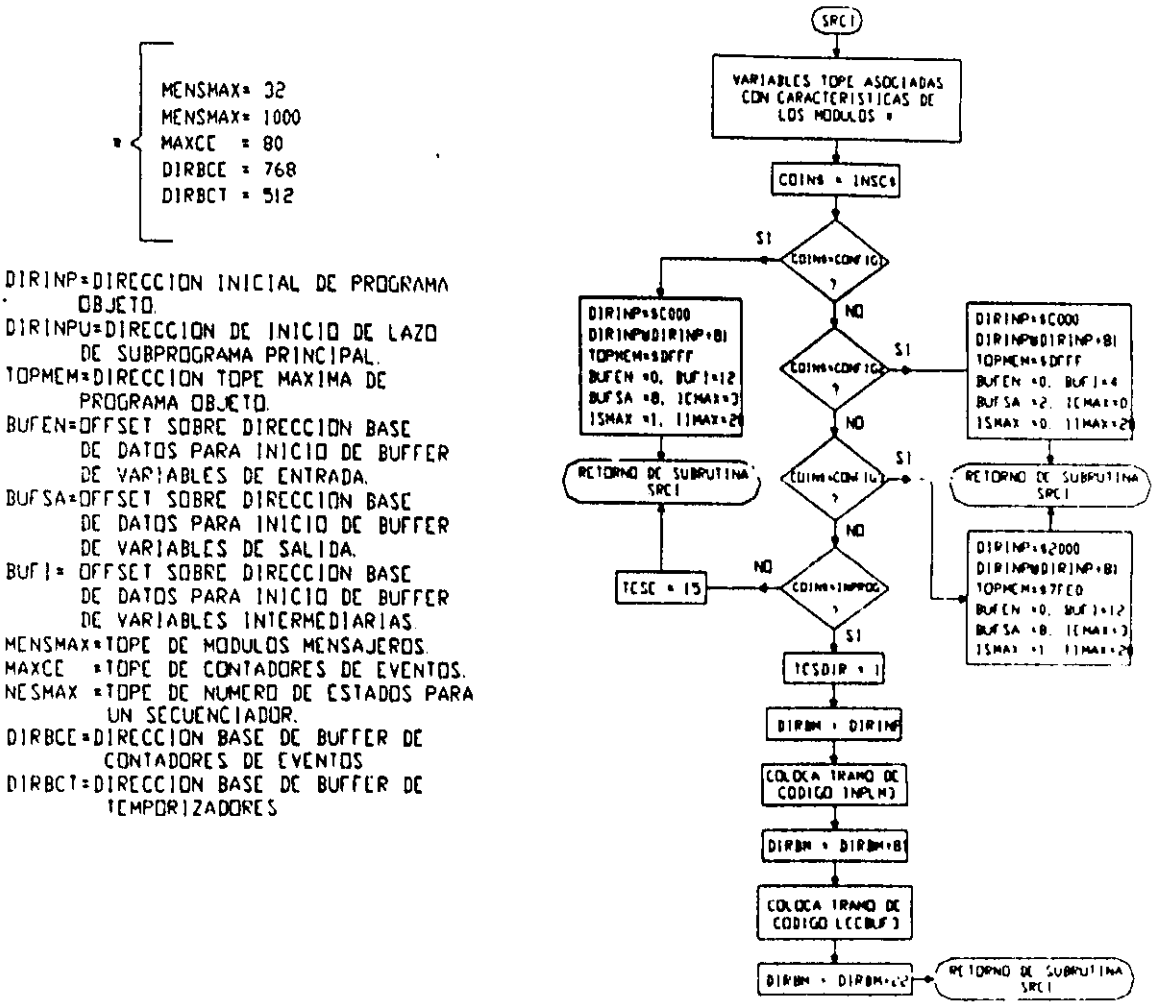


Figura 4.4 Flujo de ejecución de la subrutina reconocedora de comandos de Inicialización (SRCI).

4-1-4 Operación de la subrutina generadora de código (SGC)

Esta subrutina obtiene y coloca el código correspondiente a las instrucciones que no manejan operandos, como pueden ser las declaraciones asociadas con algunos de los módulos auxiliares, o bien comandos delimitadores de inicio y fin de los subprogramas principal y temporizado, que integren un determinado programa en SIIL1

En caso de que el renglón del archivo fuente contenga alguna instrucción que contenga operandos, la subrutina SGC invoca a la subrutina que corresponda al módulo de que se trate, para cada módulo está última subrutina es desde luego diferente; sin embargo, existen para todas ellas diversos aspectos comunes; por lo tanto, para claridad de la descripción que en esta tesis se hace de la operación del software de traducción, en el diagrama de flujo de la subrutina SGC, mostrado en la figura 4.5, se invoca una subrutina genérica de generación de código de módulos que implican operandos en su declaración (SGGCM), la cual será descrita en la sección 4-1-5

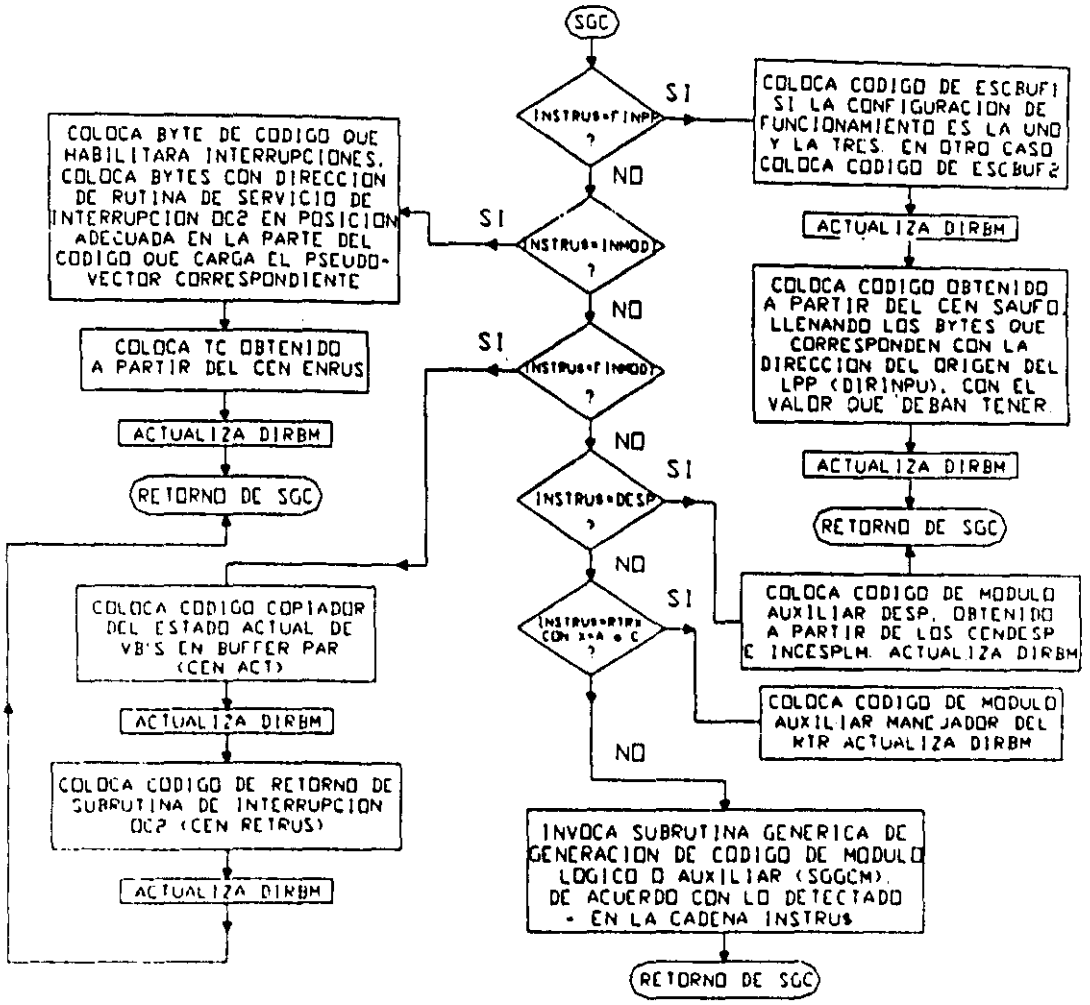


Figura 4.5 Diagrama de flujo de la subrutina generadora de código.

4-1-5 Descripción de la subrutina génerica para generación de código de módulos con operandos (SGGCM)

Esta subrutina se invoca desde la descrita en la sección anterior (SGC), en caso de que el renglón del programa fuente que se esté procesando, contenga la declaración de un módulo que implique operandos; la subrutina aquí descrita es genérica, pero contiene lo que cada subrutina individual empleada para generar el código objeto requerido por los diferentes módulos que emplean operandos (entradas y/o salidas).

Los pasos que sigue esta subrutina son:

1.- A partir de la cadena INSTRU$ que contiene la declaración del módulo, obtiene los operandos.

2.- Pregunta si el número de operandos es congruente con el módulo de que se trate, de no ser esto verdad salta al paso dieciseis.

3.- Obtiene parámetros de identificación (números de grupo y bit), asociados con los operandos que representan variables booleanas.

4.- Obtiene parámetros asociados con operandos que no representan variables booleanas

(v. g. tiempos asociados con temporizadores, cadenas binarias denotadoras de niveles de verificación, etc).

5.- Determina si alguna variable de salida del módulo, que haya sido declarada como VBS, es también salida en el mismo o en otro módulo, en cuyo caso salta al paso dieciocho.

6.- Determina si alguna variable de salida del módulo, que haya sido declarada como VBI, es también salida en el mismo o en otro módulo, en cuyo caso salta al paso veinte.

7.- Determina si alguna de las variables de salida del módulo ha sido declarada como VBE, en cuyo caso salta al paso veintidos.

8.- Carga en el segmento reservado en RAM de la PC a partir del offset especificado por la variable DIRBM el CEN asociado con el módulo declarado en el renglón que se está procesando; se generan los valores VN de la TAB asociada, de acuerdo con los operandos del módulo.

9.- Se asignan en el CEN cargado en el paso anterior, valores numéricos a los bytes especificados por la TAB que corresponda, considerando que el offset real sobre el segmento reservado que corresponda a un determinado byte, es el número que en el CEN tenga dicho byte, más el valor de la variable DIRBM.

10.- Actualiza la variable DIRBM, sumándole el número de bytes que el CEN empleado tenga.

11.- Si el módulo emplea datos adicionales, como es el caso del mensajero o el temporizador multidisparo, se pasa al paso trece.

12.- Se retorna de la subrutina SGGCM.

13.- Se hacen las lecturas de renglones del archivo fuente, necesarias para armar la lista de bytes que conforman el campo de datos asociado con la instrucción, colocándose el mismo, a partir de un offset cuyo valor es DIRBM.

14.- Se actualiza la variable DIRBM, sumándole la longitud en bytes del campo de datos mencionado en el paso anterior.

15.- Se pasa al paso doce.

16.- Asigna el valor 35 a la variable TESE.

17 - Pasa al paso doce.

18 - Asigna el valor 29 a la variable TESE.

19.- Pasa al paso doce.

20 - Asigna el valor 30 a la variable TESE.

21 - Pasa al paso doce.

22.- Asigna el valor 36 a la variable TESE.

23.- Pasa al paso doce.

En la figura 4.6 se muestra un diagrama de flujo que ilustra los pasos anteriores.

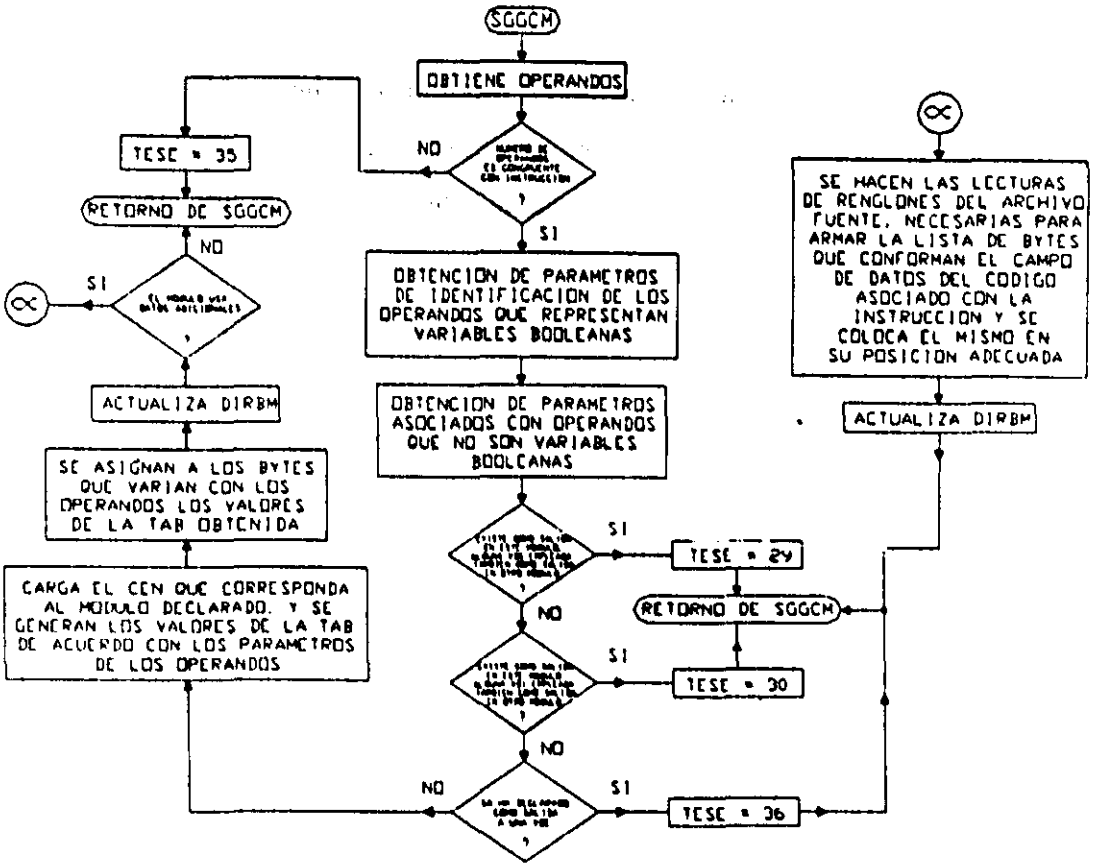


Figura 4.6 Diagrama de flujo de la subrutina SGGCM descrita en esta sección.

En las descripciones genéricas de la operación de las diversas subrutinas que conforman al software de traducción, se aprecia la captura de diversos errores de sintaxis; sin embargo, no se detallan todas las capturas de error posibles ya que las descripciones aquí hechas están simplificadas por razones de claridad; en el apéndice B se muestra la lista completa de los diferentes tipos de errores de sintaxis que a la fecha considera el software de traducción y el valor que se asigna a la variable TESE en cada caso.

4-2 DESCRIPCIÓN DEL CÓDIGO DE MONITOREO

El código de monitoreo está conformado globalmente por tres tramos de código, el primero es una subrutina que procesa los comandos de monitoreo que el PLM puede recibir y que se denomina como SUBMON, siendo la misma invocada cíclicamente en tiempo de ejecución de un programa en el PLM, esto se hace en el LPP justo antes del salto hacia atrás, al tramo de código que copia el estado de las variables físicas de entrada al buffer de entradas, la instrucción de salto a subrutina necesaria es colocada automáticamente por el software de traducción; el segundo es la subrutina de servicio de interrupción asociada con el puerto serie asíncrono del microcontrolador (SCI), a través del cual el PLM recibe de la computadora anfitriona los diversos comandos de monitoreo disponibles a la fecha de elaboración de esta tesis; el tercero es una subrutina denominada ENTREMOTA que permite cargar en el buffer de entradas (BE) bytes previamente cargados en un buffer especial (BES), que el software de traducción ubica inmediatamente después del final del buffer de transmisión.

La subrutina ENTREMOTA permitiría que el SPDPLM contuviera un comando que habilite el simular el cambio de valor en las entradas físicas, para fines de prueba y depuración de programas, para lograr esto se requiere hacer una pequeña modificación en el código objeto de un programa en SIIL1, la cual consistiría en cambiar el tramo de código que copia el estado de las entradas físicas al buffer de entrada (BE) por una invocación a la subrutina ENTREMOTA, los estados simulados de las entradas físicas serían transferidos desde la PC al buffer especial mediante el comando nueve de monitoreo descrito más adelante; cabe señalar aquí el hecho de que la opción de simulación de cambios en entradas físicas, no está por el momento desarrollada en la versión preliminar del SPDPLM.

El flujo de ejecución de la subrutina SUBMON se muestra en la figura 4.7 y el correspondiente a la subrutina de servicio de interrupción del puerto serie se muestra en la figura 4.8, para mayores detalles puede verse lo concerniente en el CEN SMT6 que se muestra más adelante en este trabajo y a partir del cual el software de traducción genera el código de monitoreo.

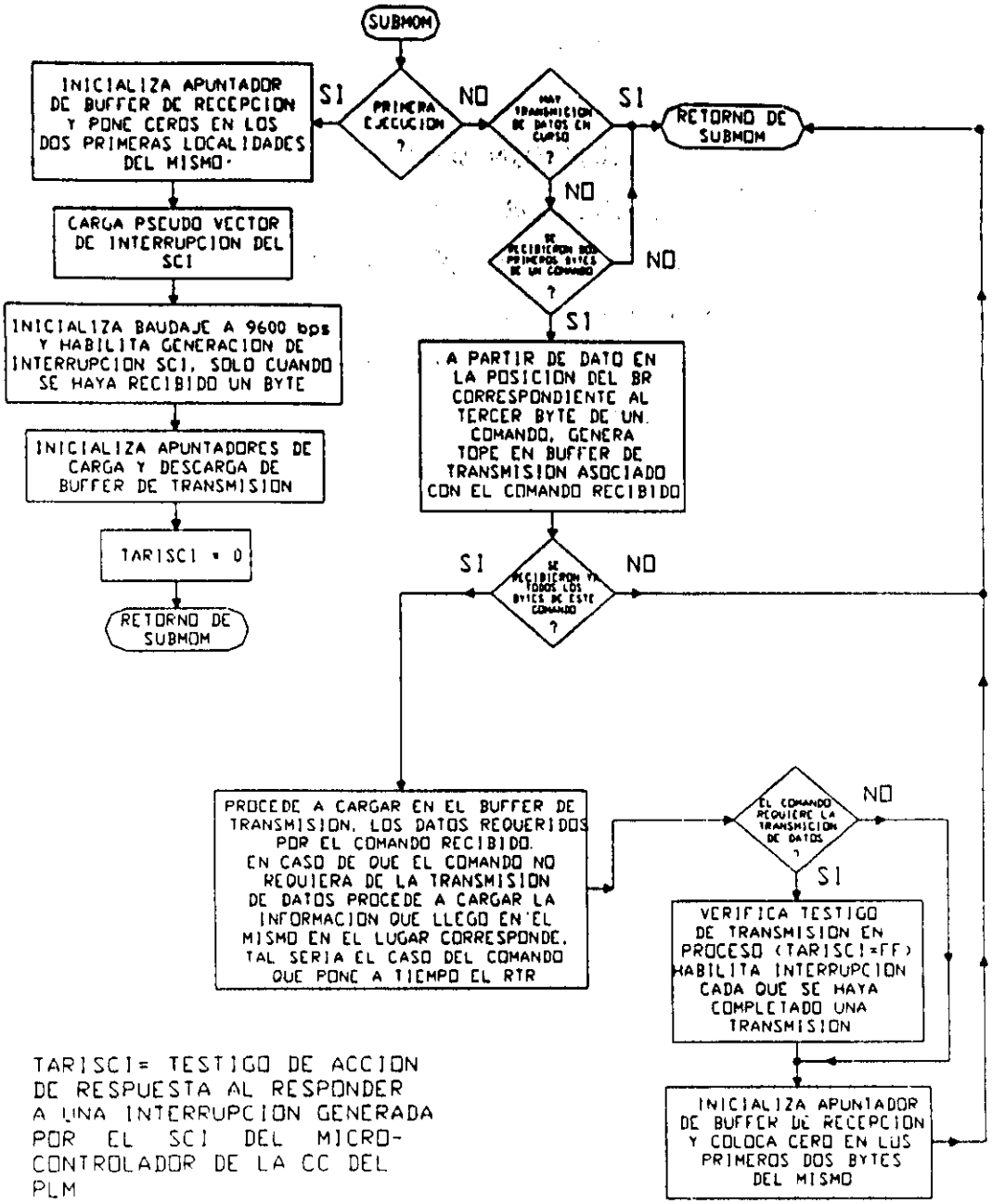


Figura 4.7 Flujo de ejecución de la subrutina SUBMON.

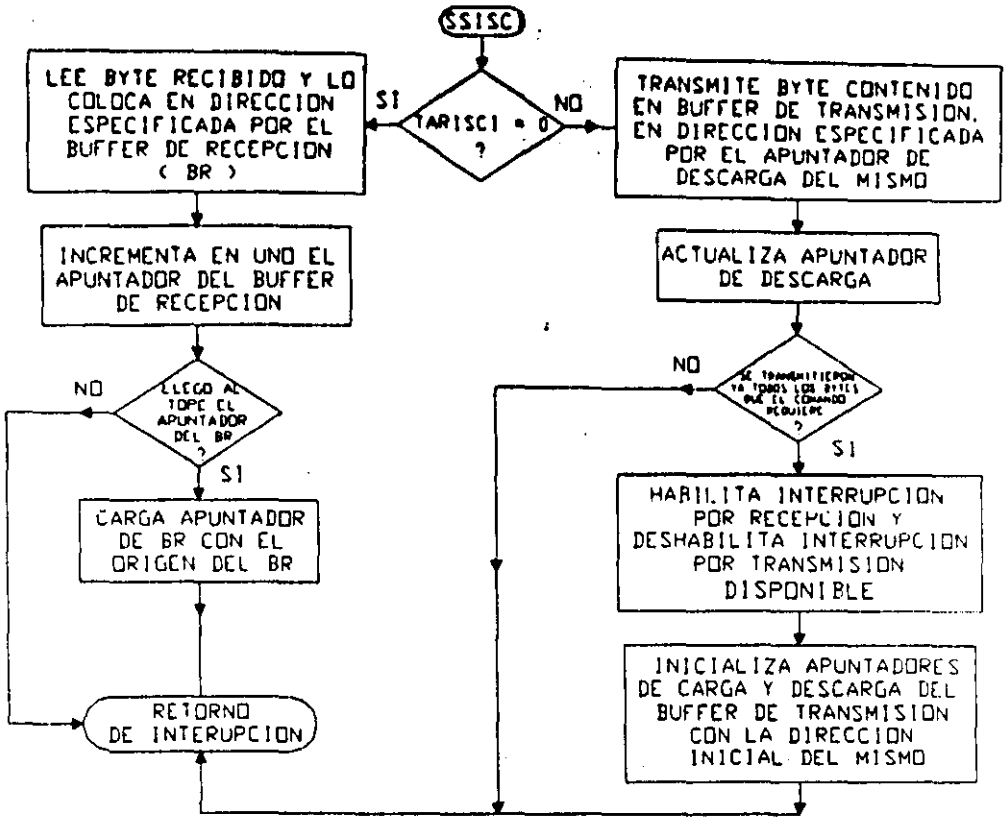


Figura 4.8 Flujo de ejecución de la subrutina de servicio de interrupción del puerto serie.

A continuación se explica globalmente y a detalle, la estructura de los comandos de monitoreo del PLM, disponibles a la fecha de elaboración de este trabajo.

4-2-1 Descripción general de los comandos de monitoreo

Los diferentes comandos de monitoreo con que cuenta el PLM están conformados por secuencias de bytes, que envía la PC al dispositivo al requerir una determinada información, o bien que el PLM efectúe un determinado accionamiento, como podría ser por ejemplo, el poner a tiempo el RTR con información de hora recibida como parte del comando.

Cuando el comando es un requerimiento de información, el PLM contesta al mismo enviando vía serie una cadena de bytes, que representa en alguna forma la información requerida.

Cada comando está formado por tres o más bytes, dependiendo esto de la complejidad del mismo, los primeros tres bytes especifican el número de dispositivo (byte ND), al que se envía el comando, el número de bytes en el comando (byte NBC), y el tipo de comando de que se trata (byte BITC), para los comandos integrados por más

de tres bytes, la información adicional que el mismo contiene, en alguna forma es utilizada por el PLM para efectuar el accionamiento que el comando requiera; en la figura 4.9 se muestra la estructura de un comando de monitoreo bajo el esquema aquí descrito.

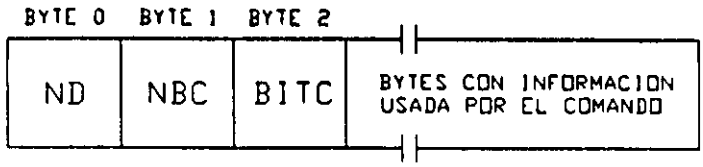


Figura 4.9 Estructura de un comando de monitoreo del PLM

El hecho de que el comando de monitoreo incluya un número de identificación de dispositivo, permite el supervisar la operación de más de un PLM desde la PC anfitriona, cada uno de los PLM implicados tendría un determinado número de identificación, de modo tal que al recibirse en un determinado PLM un comando se checaría si el valor del byte ND del mismo coincide con el propio del PLM, en caso de no haber coincidencia, simplemente se desecha el comando, en otro caso, se procede en consecuencia de acuerdo con el mismo; para las pruebas que se hicieron del SPDPLM se contempló únicamente el tener un solo PLM, forzándose a que el código de monitoreo replique los comandos recibidos sin tomar en cuenta el byte ND, dejándose para futuro las adecuaciones de hardware y software, que permitan el enlazar más de un PLM a la computadora anfitriona.

Cabe señalar aquí, el hecho de que los comandos de monitoreo no contemplan por el momento, con algún medio para detectar posibles errores, esto se debe a que el enlace entre la computadora PC anfitriona y el PLM es directo (MODEM nulo), además de que está primera versión del código de monitoreo (SMT6), se diseñó con el propósito de verificar de una manera ágil la operación de las opciones del SPDPLM, que habilitan el observación en tiempo real de diversas variables del PLM, cuando el mismo ejecuta un determinado programa en SIIL1.

Una posible forma para detectar errores en la recepción de comandos de monitoreo, podría ser el agregar al final de los mismos un byte verificador de la información (BVI), cuyo valor se obtendría efectuando una determinada operación aritmética con el resto de los bytes, de esta manera, al recibirse en el PLM un comando, se efectuaría la misma operación aritmética con los bytes pertinentes del mismo, comparándose el resultado obtenido con el valor del byte BVI recibido, en caso de

discrepancia se tomarían la acción adecuada, que podría ser simplemente desechar el comando o bien requerir de la PC la repetición del envío del mismo; en la figura 4.10 se ilustra la forma que tendrían los comandos de monitoreo si los mismos incluyen un byte verificador de información.

Para detectar errores en la información que el PLM envía a la computadora PC anfitriona al contestar a un determinado comando, podría hacerse lo descrito en el párrafo anterior para los comandos de monitoreo; esto es, las cadenas de bytes correspondientes deberán contar también con un byte BVI, y las rutinas componentes del SPDPLM que reciben tal información, deben ser adecuadas para actuar en consecuencia, al detectarse algún error en la información recibida.

De lo comentado en los dos párrafos anteriores, se desprende que para agregar facilidades que detecten errores de enlace, simplemente habría que rediseñar las rutinas de envío y recepción de datos, que son parte del SPDPLM además de que habría que hacer lo pertinente con el código de monitoreo que corre en el PLM, que desde luego se obtendría a partir de otro CEN diferente al SMT6, que podría llamarse SMT7 para el cual habría una determinada TAB; por lo tanto, también habría que rediseñar la subrutina que el software de traducción emplea para integrar el código de monitoreo al código objeto correspondiente con un determinado programa fuente en SIIL1.

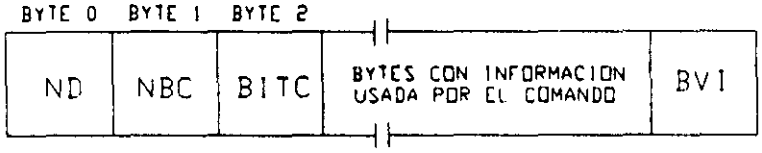


Figura 4.10 Estructura de un comando de monitoreo, cuando el mismo incluye un byte verificador de información (BVI).

4-2-2 Estructura de los diversos comandos de monitoreo manejados por el PLM

A la fecha de elaboración de esta tesis, el PLM puede responder a nueve comandos de monitoreo, la estructura de cada uno de ellos es descrita en esta sección, en todas las siguientes descripciones se asignará el valor uno al byte ND.

Comando de monitoreo uno, (requerimiento de grupo)

Al ser recibido este comando por el PLM, el mismo contesta transmitiendo vía serie el byte que representa el estado de un determinado grupo de variables booleanas, la cadena de bytes asociada es la siguiente:

ND	NBC	BITC	B3
01	04	01	XX

donde XX esta dado por la siguiente ecuación:

$$XX = 2NG + BUFG \qquad (4.1)$$

siendo NG el valor del número de grupo asociado; BUFG es una variable que deberá tomar uno de tres valores de acuerdo con la configuración de funcionamiento y el tipo de VB asociado con el grupo requerido; en la tabla 4.2 se muestran los valores que debe tomar la variable BUFG de la ecuación 4.1 de acuerdo con lo aquí mencionado.

Tabla 4.2 Asignación de valores de la variable BUFG que aparece en la ecuación 4.1 de acuerdo con la configuración de funcionamiento del PLM receptor del comando de monitoreo uno y al tipo de variables booleanas que aglutina el grupo requerido

Configuración del PLM	BUFG si el grupo requerido es de VBE`s	BUFG si el grupo requerido es de VBS`s	BUFG si el grupo requerido es de VBI`s
1 y 3	00	08	12 (0CH)
2	00	02	04

Comando de monitoreo dos, (requerimiento de buffer del buffer de la UD)

Al ser recibido este comando por el PLM, el mismo contesta transmitiendo via serie los 32 bytes contenidos en el buffer de la unidad desplegadora del PLM, la cadena de bytes asociada es la siguiente:

ND	NBC	BITC
01	03	02

Comando de monitoreo tres, (requerimiento del estado de contadores de eventos)

Al ser recibido este comando por el PLM, el mismo contesta transmitiendo via serie el par de bytes que representan la cuenta asociada con un determinado contador de eventos, la cadena de bytes asociada es la siguiente:

ND	NBC	BITC	B3	B4
01	05	03	XX	YY

donde los valores YY y XX están dados por las siguientes ecuaciones

$$XX = \text{byte alto de DIRBCE} + 2NC \qquad (4.2)$$

$$YY = \text{byte bajo de DIRBCE} + 2NC \qquad (4.3)$$

siendo NC el número del contador de eventos requerido y DIRBCE la dirección base del buffer en la memoria del PLM, asociado con los pares de bytes que testifican el estado de los contadores de eventos.

Comando de monitoreo cuatro, (requerimiento de estado de temporizador)

Al ser recibido este comando por el PLM, el mismo contesta transmitiendo vía serie, el trío de bytes que representan el estado de la cuenta de centésimas de segundo, asociada con un determinado temporizador, la cadena de bytes asociada es la siguiente:

ND	NBC	BITC	B3	B4
01	05	04	XX	YY

donde los valores YY y XX están dados por las siguientes ecuaciones:

$$XX = \text{byte alto de DIRBTE} + 3NT \qquad (4.4)$$

$$YY = \text{byte bajo de DIRBTE} + 3NT \qquad (4.5)$$

siendo NT el número del temporizador y DIRBTE la dirección base del buffer en la memoria del PLM, asociado con los tríos de bytes que testifican el estado de los temporizadores.

Comando de monitoreo cinco, (requerimiento de fecha y hora del RTR)

Al ser recibido este comando por el PLM, el mismo contesta transmitiendo vía serie una cadena de catorce bytes que representan la hora y fecha presentes en el RTR, la cadena de bytes asociada es la siguiente:

ND	NBC	BITC
01	03	05

Comando de monitoreo seis, (carga en RTR de hora)

Al recibirse este comando, el PLM carga una determinada hora recibida como parte del propio comando, la cadena de bytes asociada es la siguiente:

ND	NBC	BITC	B3	B4	B5	B6	B7	B8
01	09	06	US	DS	UM	DM	UH	DH

Donde:

US es un byte que representa el digito menos significativo de los segundos a colocar.

DS es un byte que representa el dígito más significativo de los segundos a colocar.

UM es un byte que representa el dígito menos significativo de los minutos a colocar.

DM es un byte que representa el dígito más significativo de los minutos a colocar.

UH es un byte que representa el dígito menos significativo de las horas a colocar.

DH es un byte que representa el dígito más significativo de las horas a colocar.

Comando de monitoreo siete, (carga de hora y fecha en RTR)

Al recibirse este comando, el PLM carga una determinada hora y fecha recibidas como parte del propio comando, la cadena de bytes asociada es la siguiente:

ND	NBC	BITC	B3	B4	B5	B6	B7	B8	B9	B10	B11	B12	B13	B14	B15
01	16	07	US	DS	UM	DM	UH	DH	UDM	DDM	UME	DME	UA	DA	DS

Donde:

US es un byte que representa el dígito menos significativo de los segundos a colocar.

DS es un byte que representa el dígito más significativo de los segundos a colocar.

UM es un byte que representa el dígito menos significativo de los minutos a colocar.

DM es un byte que representa el dígito más significativo de los minutos a colocar.

UH es un byte que representa el dígito menos significativo de las horas a colocar.

DH es un byte que representa el dígito más significativo de las horas a colocar.

UDM es un byte que representa el dígito menos significativo del día del mes a colocar.

DDM es un byte que representa el dígito más significativo del día del mes a colocar.

UME es un byte que representa el dígito menos significativo del mes a colocar.

DME es un byte que representa el dígito más significativo del mes a colocar.

UA es un byte que representa el dígito menos significativo del año a colocar.

DA es un byte que representa el dígito más significativo del año a colocar.

Comando de monitoreo ocho, (transfiere ejecución a dirección específica)

Al recibirse este comando, la ejecución de código por parte de la CC del PLM es transferida a la dirección especificada como parte del mismo, la cadena de bytes correspondiente es la siguiente:

ND	NBC	BITC	B3	B4
01	05	08	BADS	BBDS

Donde:

BADS el valor del byte alto, de la dirección en memoria de la CC del PLM, a donde se ha de transferir la ejecución de código.

BBDS es el valor del byte bajo, de la dirección en memoria de la CC del PLM, a donde se ha de transferir la ejecución de código.

Comando de monitoreo nueve, (carga N bytes a partir de dirección específica)

Al recibirse este comando, se cargan N bytes recibidos como parte del mismo a partir de una determinada dirección de memoria en la CC del PLM, la especificación de esta última es también parte del comando, la cadena de bytes correspondiente es la siguiente:

ND	NBC	BITC	B3	B4	B5 a B(N + 5)
01	NB	09	BADC	BBDC	N bytes a cargar

Donde:

NB es un byte que representa al número de bytes del comando que a su vez es igual al número "N" de bytes a cargar más cinco.

BADC representa al byte alto de la dirección inicial en la CC del PLM, a partir de la cual se han de cargar los N bytes recibidos con el comando.

BBDC representa al byte bajo de la dirección inicial en la CC del PLM, a partir de la cual se han de cargar los N bytes recibidos con el comando.

4-3 CEN SMT6, A PARTIR DEL CUAL EL SOFTWARE DE TRADUCCIÓN ARMA EL CÓDIGO DE MONITOREO.

En esta sección se muestra el CEN SMT6 empleado para generar el código de monitoreo, nótese que el mismo no se origina en la dirección 0000H sino en la 0100H, esto se hizo así debido a el forzamiento que hace el ensamblador empleado, en cuanto al uso del modo de direccionamiento directo para direcciones asociadas con diversas instrucciones, al estar el código correspondiente colocado en la página cero, generándose conflictos, ya que el código real seguramente va estar colocado fuera de la página cero.

El CEN SMT6 es el siguiente:

CEN SMT6

```
CEN ASOCIADO CON LA RRECEPCIÓN DE COMANDOS VÍA SERIE,LOS BYTES QUE SE
RECIBEN SE VAN CARGANDO EN UN BUFFER DE RECEPCIÓN (BR).
APUNTADOR DE BUFFER DE RECEPCIÓN EN DIR X+B0.
APUNTADOR DE CARGA DE BUFFER DE TRANSMISIÓN EN DIR X+B2.
APUNTADOR DE DESCARGA DE BUFFER DE TRANSMISIÓN EN DIR X+B4.
TESTIGO DE ACCIÓN DE RESPUESTA A INTERRUPCIÓN (TARISCI) EN DIR X+B6.
TESTIGO DE DIRECCIÓN TOPE EN BUFFER DE TRANSMISIÓN EN DIR X+B7.

0100                            ORG $0100
0100 2027           SUBMONIT:   BRA SUBMON

AQUÍ INICIA CÓDIGO DE RUTINA DE SERVICIO DE INTERRUPCIÓN
GENERADA POR EL PUERTO SERIE DEL MICROCONTROLADOR (SCI).

0102 B6102E         SERVINTREC: LDAA $102E
0105 CE0100                     LDX #$0100
0108 E6B6                       LDAB $B6,X
010A 2704                       BEQ RECEP;SALTA A LEER BYTE
                                          RECIBIDO,YA QUE (X+B6)=0
010C 8D4D                       BSR TRANSMITE;SALTA A TRANSMITIR
                                              DATOS,YA QUE (X+B6)=FF
010E 2018                       BRA FUERA
0110 F6102F         RECEP:      LDAB $102F;LEE DATO RECIBIDO
0113 1AEEB0                     LDY $B0,X;CARGA "Y" CON APUNTADOR DE
                                          SIGUIENTE DIRECCIÓN
                                          DISPONIBLE,PARA CARGAR UN
                                          BYTE RECIBIDO,EN BUFFER DE
                                          RECEPCIÓN
0116 18E700                     STAB $00,Y;CARGA DATO RECIBIDO EN
                                           BUFFER DE RECEPCIÓN
0119 1808                       INY
011B 188CDF80                   CPY #$DF80;COMPARA "Y" CON TOPE DE
                                           BUFFER DE RECEPCIÓN+1
011F 2604                       BNE NOFINAL;SALTA SI NO SE HA LLEGADO
                                            A TOPE DE BR
0121 18CEDF00                   LDY #$DF00;PREPARA "Y" PARA
                                          REINICIALIZAR APUNTADOR DE
                                          BUFFER DE RECEPCIÓN
0125 1AEFB0         NOFINAL:    STY $B0,X;ACTUALIZA APUNTADOR DE BR
0128 3B             FUERA:      RTI

0129 A6F2           SUBMON:     LDAA $F2,X
012B 272B                       BEQ NOPP
```

```
ACCIONAMIENTO AL ENTRAR A POR PRIMERA VEZ A SUBMON (INICIALIZACIÓN)
012D 18CEDF00                    LDY #$DF00;CARGA D CON DIRECCIÓN DE
                                           .INICIO DE BUFFER DE
                                           RECEPCIÓN
0131 1AEFB0                      STY $B0,X;INICIALIZA APUNTADOR DE BR
0134 186F00                      CLR $00,Y
0137 186F01                      CLR $01,Y
013A 867E                        LDAA #$7E;CARGA PSEUDOVECTOR ASOCIADO
                                           CON INTERRUPCIÓN GENERADA
013C 97C4                        STAA $C4; POR SCI
013E CC0102                      LDD #SERVINTREC
0141 DDC5                        STD $C5;
0143 8630                        LDAA #$30
0145 B7102B                      STAA $102B;CARGA REGISTRO BAUD PARA
                                           9600 BPS
0148 862C                        LDAA #$2C
014A B7102D                      STAA $102D;HABILITA INTERRUPCIONES
                                           POR RECEPCIÓN
014D CCDF80          INICAPSBT:  LDD #$DF80
0150 EDB2                        STD $B2,X
0152 EDB4                        STD $B4,X
0154 6FB6                        CLR $B6,X
0156 201D                        BRA SALIDA2

INVOCACIÓN DE SUBRUTINA RECONOCEDORA DE COMANDOS

0158 8D38            NOPP:       BSR SUBRC
015A 39                          RTS

INICIO DE CÓDIGO DE MÓDULO TRANSMISOR VÍA SERIE

015B 1AEEB4          TRANSMITE:  LDY $B4,X;CARGA Y CON APUNTADOR DE
                                           DESCARGA DE BT
015E 18E600                      LDAB $00,Y;COPIA EN B SIGUIENTE DATO
                                           POR TRANSMITIR
0161 F7102F                      STAB $102F;TRANSMITE DATO
0164 1808                        INY
0166 1AEFB4                      STY $B4,X;ACTUALIZA APUNTADOR DE
                                           DESCARGA
0169 1AACB7                      CPY $B7,X;COMPARA CON DIRECCIÓN
                                           TOPE EN BT,INDICADORA
                                           DE FIN DE DATOS POR
                                           TRANSMITIR
016C 2607                        BNE SALIDA2
016E 862C            FINDAT:     LDAA #$2C
0170 B7102D                      STAA $102D;HABILITA INTERRUPCIONES
                                           SOLO POR RECEPCIÓN
0173 20D8                        BRA INICAPSBT
0175 39              SALIDA2:    RTS

0176 86FF            INICABR:    LDAA #$FF
0178 A7B6                        STAA $B6,X;PONE TESTIGO DE
                                           TRANSMISIÓN EN PROCESO
017A 864C                        LDAA #$4C
017C B7102D                      STAA $102D;HABILITA INTERRUPCIONES
                                           PARA TRANSMITIR CONENIDO
                                           DEL BUFFER DE DATOS
017F 18CEDF00        INICABRS:   LDY #$DF00;CARGA D CON DIRECCIÓN DE
                                           INICIO DE BUFFER DE
                                           RECEPCIÓN
0183 1AEFB0                      STY $B0,X;INICIALIZA APUNTADOR DE BR
```

```
0186 186F00                      CLR $00,Y
0189 186F01                      CLR $01,Y
018C 39              SALIDA3AU:  RTS
018D 01200203        TABLITA:    FCB $01,$20,$02,$03,$0D
     0D
SUBRUTINA RECONOCEDORA DE COMANDOS RECIBIDOS VÍA SERIE
0192 A6B6            SUBRC:      LDAA $B6,X
0194 26F6                        BNE SALIDA3AU;SALE SI HAY
                                               TRANSMISIÓN DE
                                               DATOS EN PROCESO
0196 18CEDF00                    LDY #$DF00;PONE EN IY DIRECCIÓN
                                            ORIGEN DE BR
019A 18A600                      LDAA $00,Y
019D 27ED                        BEQ SALIDA3AU;SALE SI NO SE HA
                                               RECIBIDO PRIMER BYTE
                                               DE COMANDO
019F 18E601                      LDAB $01,Y
01A2 277C                        BEQ SALIDA3;SALE SI NO SE HA
                                            RECIBIDO SEGUNDO BYTE DE
                                            COMANDO
01A4 37                          PSHB
01A5 18E602                      LDAB $02,Y; COPIA EN B BITC
01A8 5A                          DECB
01A9 183C                        PSHY
01AB 18CE018D                    LDY #TABLITA
01AF 183A                        ABY
01B1 18E600                      LDAB $00,Y
01B4 18CEDF80                    LDY #$DF80
01B8 183A                        ABY
01BA 1AEFB7                      STY $B7,X
01BD 1838                        PULY
01BF 33                          PULB
AQUÍ SE TIENE EN A NÚMERO DE DISPOSITIVO Y EN B NÚMERO DE BYTES DEL
COMANDO
01C0 183A                        ABY
01C2 1AACB0                      CPY $B0,X;COMPARA APUNTADOR DE BR
                                           CON ORGBUF + N (NÚMERO DE
                                           DATOS ESPERADOS)
01C5 2659                        BNE SALIDA3;SALE SI HAY MÁS BYTES
                                             POR RECIBIR
01C7 18CEDF00                    LDY #$DF00;CARGA "Y" CON ORIGEN DE
                                            BR
01CB 8104                        CMPA #$04;COMPARA CON NÚMERO DE
                                           DISPOSITIVO ESCLAVO
01CD 2004                        BRA SIFUIYO;BEQ CUANDO YA SE ESTE
                                             TRABAJANDO CON MÁS DE UN
                                             DISPOSITIVO ESCLAVO
01CF 8DAE            NOFUIYO:    BSR INICABRS
01D1 204D                        BRA SALIDA3
01D3 18A602          SIFUIYO:    LDAA $02,Y;COPIA EN A BITC
01D6 8101                        CMPA #$01
01D8 2604                        BNE SS1
01DA 8D45                        BSR REQGE
01DC 2042                        BRA SALIDA3
01DE 8102            SS1:        CMPA #$02
01E0 2604                        BNE SS2
01E2 8D5B                        BSR REQDIS
01E4 203A                        BRA SALIDA3
01E6 8103            SS2:        CMPA #$03
01E8 2605                        BNE SS3
01EA BD0279                      JSR REQECE
```

```
01ED 2031                       BRA SALIDA3
01EF 8104          SS3:         CMPA #$04
01F1 2605                       BNE SS4
01F3 BD0292                     JSR REQETE
01F6 2028                       BRA SALIDA3
01F8 8105          SS4:         CMPA #$05
01FA 2605                       BNE SS5
01FC BD02BD                     JSR REQFYH
01FF 201F                       BRA SALIDA3
0201 8106          SS5:         CMPA #$06
0203 2605                       BNE SS6
0205 BD02DD                     JSR EHRTR
0208 2016                       BRA SALIDA3
020A 8107          SS6:         CMPA #$07
020C 2603                       BNE SS7
020E BD02F3                     JSR EHYFRTR
0211 8108          SS7:         CMPA #$08
0213 2603                       BNE SS8
0215 BD0309                     JSR SALTADOR
0218 8109          SS8:         CMPA #$09
021A 2603                       BNE SS9
021C BD0314                     JSR ENBYTES
021F 01            SS9:         NOP;AQUÍ SEGUIRIAN ENRUTAMIENTOS DE
                                    COMANDOS FUTUROS
0220 39            SALIDA3      RTS
0221 183C          REQGE:       PSHY
0223 18E603                     LDAB $03,Y;COPIA EN B OFFSET SOBRE
                                          DIR BASE DE BUFFER DE
                                          VB`S
0226 3C                         PSHX
0227 1838                       PULY
0229 183A                       ABY
022B 18A600                     LDAA $00,Y;COPIA EN A BYTE TESTIGO
                                          DE GRUPO
022E 1AEEB2                     LDY $B2,X;Y<APUNTADOR DE CARGA DE BT
0231 18A700                     STAA $00,Y
0234 1808                       INY
0236 1AEFB2                     STY $B2,X;ACTUALIZA APUNTADOR DE
CARGA DE BT
0239 1838                       PULY
023B BD0176                     JSR INICABR
023E 39                         RTS
023F 3C            REQDIS:      PSHX
0240 C620                       LDAB #$20
0242 1AEEB2                     LDY $B2,X;Y<PUNTADOR DE CARGA DE BT
0245 A6D0          OTRO:        LDAA $D0,X
0247 8108                       CMPA #$08
0249 2502                       BLO CARACESP
024B 2002                       BRA XXXX
024D 8D19          CARACESP:    BSR AUXREQDIS
024F 18A700        XXXX:        STAA $00,Y
0252 1808                       INY
0254 08                         INX
0255 5A                         DECB
0256 26ED                       BNE OTRO
0258 38                         PULX
0259 1AEFB2                     STY $B2,X;ACTUALIZA APUNTADOR DE
                                          CARG. DE BT
025C BD0176                     JSR INICABR
025F 39                         RTS
0260 A082A1A2      ASCIIEXT:    FCB $A0,$82,$A1,$A2,$A3,$A8,$AD,$A5
```

```
     A3A8ADA5
0268 37           AUXREQDIS:  PSHB
0269 183C                     PSHY
026B 18CE0260                 LDY #ASCIIEXT
026F 16                       TAB
0270 183A                     ABY
0272 18A600                   LDAA $00,Y
0275 1838                     PULY
0277 33                       PULB
0278 39                       RTS
0279 18EC03       REQECE:     LDD $03,Y
027C 188F                     XGDY
027E 18EC00                   LDD $00,Y;COPIA EN D ESTADO DE
                                    CONTADOR REQUERIDO
0281 1AEEB2                   LDY $B2,X;Y<APUNTADOR DE CARGA DE BT
0284 18ED00                   STD $00,Y
0287 1808                     INY
0289 1808                     INY
028B 1AEFB2                   STY $B2,X;ACTUALIZA APUNTADOR DE
                                    CARGA DE BT
028E BD0176                   JSR INICABR
0291 39                       RTS
0292 18EC03       REQETE:     LDD $03,Y
0295 188F                     XGDY
0297 18EC00                   LDD $00,Y;COPIA EN D PAR DE BYTES
                                    MÁS SIGNIFICATIVOS DE
                                    TEMPORIZADOR REQUERIDO
029A 183C                     PSHY
029C 1AEEB2                   LDY $B2,X;Y<AUNTADOR DE CARGA DE BT
029F 18ED00                   STD $00,Y
02A2 1808                     INY
02A4 1808                     INY
02A6 1AEFB2                   STY $B2,X;ACTUALIZA APUNTADOR DE
                                    CARGA DE BT
02A9 1838                     PULY
02AB 18A602                   LDAA $02,Y;COPIA EN A BYTE MENOS
                                    SIGNIFICATIVO DE
                                    TEMPORIZADOR
02AE 1AEEB2                   LDY $B2,X;Y<APUNTADOR DE CARGA DE BT
02B1 18A700                   STAA $00,Y
02B4 1808                     INY
02B6 1AEFB2                   STY $B2,X;ACTUALIZA APUNTADOR DE
                                    CARGA DE BT
02B9 BD0176                   JSR INICABR
02BC 39                       RTS
02BD 3C           REQFYH:     PSHX
02BE C60D                     LDAB #$0D
02C0 1AEEB2                   LDY $B2,X;Y<APUNTADOR DE CARGA DE BT
02C3 CE1BCE                   LDX #$1BCE
02C6 A600         OTRO1:      LDAA $00,X
02C8 840F                     ANDA #$0F
02CA 8B30                     ADDA #$30
02CC 18A700                   STAA $00,Y
02CF 1808                     INY
02D1 09                       DEX
02D2 5A                       DECB
02D3 26F1                     BNE OTRO1
02D5 38                       PULX
02D6 1AEFB2                   STY $B2,X;ACTUALIZA APUNTADOR DE
                                    CARGA DE BT
02D9 BD0176                   JSR INICABR
```

```
02DC 39                         RTS
02DD 3C             EHRTR:      PSHX
02DE 8606                       LDAA #$06
02E0 CE1BC2                     LDX #$1BC2
02E3 18E603         OTRO2:      LDAB $03,Y
02E6 E700                       STAB $00,X
02E8 08                         INX
02E9 1808                       INY
02EB 4A                         DECA
02EC 26F5                       BNE OTRO2
02EE 38                         PULX
02EF BD017F                     JSR INICABRS
02F2 39                         RTS
02F3 3C             EHYFRTR:    PSHX
02F4 860D                       LDAA #$0D
02F6 CE1BC2                     LDX #$1BC2
02F9 18E603         OTRO3:      LDAB $03,Y
02FC E700                       STAB $00,X
02FE 08                         INX
02FF 1808                       INY
0301 4A                         DECA
0302 26F5                       BNE OTRO3
0304 38                         PULX
0305 BD017F                     JSR INICABRS
0308 39                         RTS
0309 BD017F         SALTADOR:   JSR INICABRS
030C CDEE03                     LDX $03,Y;COPIA EN IX DIRECCIÓN A
                                        SALTAR
030F 3C                         PSHX
0310 CE0100                     LDX #$0100;RECUPERA VALOR DE X
0313 39                         RTS;RETORNA A DIRECCIÓN ESPECIFICADA
0314 3C             ENBYTES:    PSHX
0315 CDEE03                     LDX $03,Y;COPIA EN IX DIRECCIÓN DE
                                        CARGA
0318 18E601                     LDAB $01,Y
031B C006                       SUBB #$06
031D 18A605         YY:         LDAA $05,Y
0320 A700                       STAA $00,X
0322 08                         INX
0323 1808                       INY
0325 5A                         DECB
0326 26F5                       BNE YY
0328 38                         PULX
0329 BD017F                     JSR INICABRS
032C 39                         RTS
```

SURUTINA ENTREMOTA,CARGA DATOS EN BUFFER DE ENTRADAS FÍSICAS,A PARTIR DE BUFFER ESPECIAL;LOS DATOS EN ESTE BUFFER SON RECIBIDOS VÍA SERIE DESDE PC ANFITRIONA

```
032D A6F2           ENTREMOTA:  LDAA $F2,X
032F 2711                       BEQ NOPP5
ACCIONAMIENTO EN PRIMERA EJECUCIÓN DE ESTE CÓDIGO
0331 18CEDFC0       CEROSABE:   LDY #$DFC0;CARGA IY CON DIR ORIGEN
                                        DE BUFFER ESPECIAL (BE)
0335 186F00         YUI:        CLR $00,Y
0338 1808                       INY
033A 188CE000                   CPY #$E000; COMPARA CON DIR FINAL DE
                                        BE MÁS UNO
033E 26F5                       BNE YUI
0340 2014                       BRA TERMINA
```

```
0342 3C                 NOPP5:      PSHX
0343 C601                           LDAB #$01;(01 PARA CONFIG2),(04 PARA
                                      CONFIG1 Y CONFIG3)
0345 18CEDFC0                       LDY #$DFC0;IY<DIRINIBE (DIR INICIAL
                                      BUFFER ESPECIAL)
0349 18A600             RJ:         LDAA $00,Y
034C A700                           STAA $00,X
034E 1808                           INY
0350 08                             INX
0351 08                             INX
0352 5A                             DECB
0353 26F4                           BNE RJ
0355 38                             PULX
0356 39                 TERMINA:    RTS
```

En la tabla 4.3 se muestra la TAB asociada con el CEN SMT6.

Tabla 4.3 Asignación de bytes asociada con el CEN SMT6, empleado para obtener el código requerido para monitorear en tiempo real desde una PC la operación del PLM, al ejecutarse en el mismo un programa en SIIL.

Valor numérico (VN) generado por el software de traducción	Byte en CEN genérico SMT6, al que se le debe asignar el valor numérico VN
VN= Byte alto de DIRBM + 377	B235
VN= Byte bajo de DIRBM + 377	B236
VN= Byte alto de DIRBM + 402	B244
VN= Byte bajo de DIRBM + 402	B245
VN= Byte alto de DIRBM + 445	B253
VN= Byte bajo de DIRBM + 445	B254
VN= Byte alto de DIRBM + 477	B262
VN= Byte bajo de DIRBM + 477	B263
VN= Byte alto de DIRBM + 499	B271
VN= Byte bajo de DIRBM + 499	B272
VN= Byte alto de DIRBM + 521	B278
VN= Byte bajo de DIRBM + 521	B279
VN= Byte alto de DIRBM + 532	B285
VN= Byte bajo de DIRBM + 532	B286
VN= Byte alto de DIRBM + 118	B316, B349, B399, B442, B474
VN= Byte bajo de DIRBM + 118	B317, B350, B400, B443, B475
VN= Byte alto de DIRBM + 127	B496, B518, B522, B554
VN= Byte bajo de DIRBM + 127	B497, B519, B523, B555
VN= Byte alto de DIRBM + 2	B63
VN= Byte bajo de DIRBM + 2	B64
VN= Valor numérico de byte a cargarse en registro BAUD del microcontrolador, por defecto VN es 48d, lo que coloca el baudaje en 9600 bps, para fe= 2 MHz	B68
VN= Byte alto de DIRBM + 141	B173
VN= Byte bajo de DIRBM + 141	B174
VN= Byte alto de DIRBM + 352	B365
VN= Byte bajo de DIRBM + 352	B366
VN= Número de identificacion del PLM	B204
VN= Byte alto de DIRINIBR*	B35, B47, B129, B152, B201

```
0342 3C                 NOPP5:      PSHX
0343 C601                           LDAB #$01;(01 PARA CONFIG2),(04 PARA
                                          CONFIG1 Y CONFIG3)
0345 18CEDFC0                       LDY #$DFC0;IY<DIRINIBE (DIR INICIAL
                                          BUFFER ESPECIAL)
0349 18A600             HJ:         LDAA $00,Y
034C A700                           STAA $00,X
034E 1808                           INY
0350 08                             INX
0351 08                             INX
0352 5A                             DECB
035? 26F4                           BNE HJ
0355 38                             PULX
0356 39                 TERMINA:    RTS
```

En la tabla 4.3 se muestra la TAB asociada con el CEN SMT6.

Tabla 4.3 Asignación de bytes asociada con el CEN SMT6, empleado para obtener el código requerido para monitorear en tiempo real desde una PC la operación del PLM, al ejecutarse en el mismo un programa en SIIL1.

Valor numérico (VN) generado por el software de traducción	Byte en CEN genérico SMT6, al que se le debe asignar el valor numérico VN
VN= Byte alto de DIRBM + 377	B235
VN= Byte bajo de DIRBM + 377	B236
VN= Byte alto de DIRBM + 402	B244
VN= Byte bajo de DIRBM + 402	B245
VN= Byte alto de DIRBM + 445	B253
VN= Byte bajo de DIRBM + 445	B254
VN= Byte alto de DIRBM + 477	B262
VN= Byte bajo de DIRBM + 477	B263
VN= Byte alto de DIRBM + 499	B271
VN= Byte bajo de DIRBM + 499	B272
VN= Byte alto de DIRBM + 521	B278
VN= Byte bajo de DIRBM + 521	B279
VN= Byte alto de DIRBM + 532	B285
VN= Byte bajo de DIRBM + 532	B286
VN= Byte alto de DIRBM + 118	B316, B349, B399, B442, B474
VN= Byte bajo de DIRBM + 118	B317, B350, B400, B443, B475
VN= Byte alto de DIRBM + 127	B496, B518, B522, B554
VN= Byte bajo de DIRBM + 127	B497, B519, B523, B555
VN= Byte alto de DIRBM + 2	B63
VN= Byte bajo de DIRBM + 2	B64
VN= Valor numérico de byte a cargarse en registro BAUD del microcontrolador, por defecto VN es 48d, lo que coloca el baudaje en 9600 bps, para fe= 2 MHz	B68
VN= Byte alto de DIRBM + 141	B173
VN= Byte bajo de DIRBM + 141	B174
VN= Byte alto de DIRBM + 352	B365
VN= Byte bajo de DIRBM + 352	B366
VN= Número de identificación del PLM	B204
VN= Byte alto de DIRINIBR*	B35, B47, B129, B152, B201

Continuación Tabla 4.3 Asignación de bytes asociada con el CEN SMT6.

VN= 32, si el enlace de monitoreo es sólo entre una PC y un PLM. VN= 39, si el enlace de monitoreo es entre una PC y varios PLM, cada uno con un número de identificación diferente.	B205
VN= Byte alto de DIRINIBR*	B35, B47, B129, B152, B201
VN= Byte bajo de DIRINIBR*	B36, B48, B130, B153, B202
VN= Byte alto de DIRINIBT**	B29, B78, B182
VN= Byte bajo de DIRINIBT**	B30, B79, B183
VN= Byte alto de DIRINIBE***	B563, B583, B572
VN= Byte bajo de DIRINIBE***	B564, B584, B573
VN= Byte alto de DIRINIBE + long(BE) +1	B572
VN= Byte bajo de DIRINIBE + long(BE) +1	B573
VN= 01, para CONFIG2 VN= 04, para CONFIG1 y CONFIG3	B580

* DIRINIBR = DIRBM + 599 (Drección inicial de buffer de recepción).
** DIRINIBT = DIRINIBR + 128.
*** DIRINIBE = DIRINIBR + 192.
Long(BE) = Longitud de buffer especial, el valor por defecto para este parámetro es 64 bytes.

4-4 DESCRIPCIÓN GLOBAL DEL SPDPLM DESDE EL PUNTO DE VISTA DEL USUARIO FINAL

El sistema para desarrollo con el PLM (SPDPLM), integra en un solo ambiente diversas facilidades de software, las cuales permiten desarrollar aplicaciones y ejemplos de programación alrededor del PLM.. A la fecha se cuenta con una versión preliminar del SPDPLM que fue desarrollada empleando Visual Basic 4 versión profesional, las facilidades más importantes con que cuenta a la fecha el SPDPLM son:

1.- Editor de texto integrado, que el usuario puede emplear para introducir, cambiar y guardar archivos SIL, que contengan programas fuente en SIIL1, para su posterior traducción a código objeto propio de la CC del PLM.

2.- Capacidad para generar el código objeto asociado con un archivo fuente en SIIL1, el archivo objeto queda en un archivo binario con extensión BLM, compatible con el manejador hexadecimal PUMMA. EL SPDPLM emplea para estos fines un programa denominado como SIIL1MVB.EXE que es una versión modificada del programa SIIL1.EXE, descrito en secciones anteriores, las modificaciones consisten fundamentalmente en el hecho de que la comunicación entre el SPDPLM y el programa SIIL1MVB.EXE es mediante archivos auxiliares de texto.

Asi, al ejecutarse SIIL1MVB.EXE el nombre del programa fuente a traducir es tomado de un archivo auxiliar previamente generado por el SPDPLM, con el nombre

del programa capturado de información proporcionada por el usuario y los avisos diversos que en SIIL1.EXE se imprimen directamente en pantalla, pasan a ser colocados en otro archivo auxiliar, que posteriormente emplea el SPDPLM para mostrar el resultado del proceso de traducción, si el usuario así lo desea.

3.- Capacidad de bajar y autoejecutar en la CC del PLM, archivos objeto BLM, que podrían haber sido obtenidos a partir de un programa fuente en SIIL1.

4.- Capacidad para monitorear en tiempo real diversas variables del PLM, al ejecutarse en el mismo un determinado programa en SIIL1.

Para desarrollar con el SPDPLM se requiere para varias de sus facilidades, que el PLM este ligado vía serie a la computadora PC donde se ejecuta el mismo, en la figura 4.11 se ilustra esto.

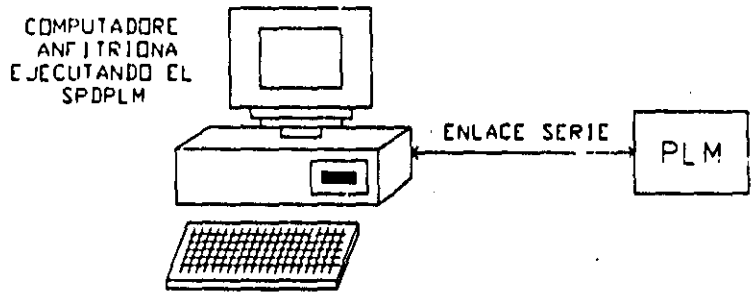


Figura 4.11 Sistema básico para desarrollar, probar y depurar programas ejemplo o de aplicación con el PLM.

A continuación se describen los pasos a seguir para iniciar una sesión de trabajo con el sistema mostrado en la figura 4.11.

1.- Con la computadora anfitriona operando, se energiza el PLM y se oprime el botón de restablecimiento (RESET), de la CC del mismo (CMT SIMMP-2), después de lo cual deberá aparecer en la UD un mensaje que dice: " PLM LISTO PARA RECIBIR PROGRAMA" además de observarse un centelleo rápido en el LED testigo propio de la CMT SIMMP-2.

2.- Estando la PC operando bajo ambiente windows, se entra al SPDPLM apareciendo entonces la ventana de entrada al mismo, en cuya barra de título aparece la leyenda "MANEJADOR Y GENERADOR DE CÓDIGO DEL PLM", debajo de la barra de título aparece una barra de menú que presenta los cuatro menúes siguientes: Archivo, Editar, Ver y Ejecutar.

Una vez que se han llevado a cabo los dos pasos anteriores, se puede proceder a desarrollar programas para ejecutarse en el PLM; en las siguientes secciones se

describen las diversas opciones asociadas con cada uno de los menúes mencionados en el párrafo anterior.

4-4-1 Descripción de las opciones del menú "Archivo" del SPDLM

Las opciones del menú "Archivo" son enumeradas y descritas a continuación:

Abrir archivo BLM

Mediante esta opción, el usuario puede abrir un archivo objeto BLM, para su posterior carga y autoejecución en el PLM, para ello emplearía la opción "Ejecutar archivo BLM presente", que es parte del menú "Ejecutar" que será descrito más adelante en otra sección.

Nuevo archivo SIL

Esta opción permite al usuario poner en blanco la ventana del editor, para iniciar la introducción de un programa fuente en SIIL1; en caso de que previamente se haya hecho alguna modificación a un determinado archivo fuente, que haya estado presente antes de invocar esta opción, el sistema interroga al usuario acerca de si desea guardar en disco el programa modificado, en cuyo caso el sistema procede en consecuencia.

Guardar archivo SIL presente

Mediante esta opción, el usuario puede guardar en disco el archivo fuente SIL, bajo el nombre que previamente se le haya dado.

Guardar archivo fuente SIL como

Esta opción permite al usuario guardar bajo un nombre diferente y en otro subdirectorio, si así lo desea, el archivo SIL presente en la ventana del editor, al invocarse la misma aparecerán los cuadros de dialogo propios del ambiente windows para estos casos.

Obtener código para el PLM desde:

Al invocar esta opción el usuario puede hacer que se genere el archivo objeto BLM que contenga el código asociado con un determinado programa fuente en SIIL1, esta opción contiene dos subopciones que son:

A) Generar archivo objeto a partir del programa fuente presente en la ventana del editor.

B) Generar archivo objeto a partir de un archivo SIL contenido en disco.

Salir

Esta opción cierra el SPDPLM; en caso de que se hayan hecho modificaciones a un determinado archivo fuente y el mismo no haya sido guardado, el sistema interroga

al usuario acerca si desea guardar tal archivo con las modificaciones hechas, actuando en consecuencia dependiendo de la respuesta que el usuario de al cuestionamiento mencionado.

4-4-2 Descripción de las opciones del menú "Editar" del SPDLM

Las opciones del menú "Editar" son enumeradas y descritas a continuación:

Habilitar editor

Al invocarse esta opción se abre la ventana del editor de texto que forma parte del SPDPLM, para que el usuario introduzca un nuevo programa fuente en SIIL1, o bien cargue de disco el archivo algún archivo SIL para trabajar en él.

Deshabilitar editor

Esta opción cierra la ventana del editor de texto del SPDPLM.

4-4-3 Descripción de las opciones del menú "Ver" del SPDLM

Las opciones del menú "Ver" son enumeradas y descritas a continuación:

Ver panel frontal virtual del PLM

Al invocarse esta opción, aparece en la ventana del SPDPLM un marco con mímicos que representan: las 32 variables booleanas de entrada, las 16 variables booleanas de salida, el estado del RTR, y la ventana de la UD.

Cada VB es testificada por el símil de un LED de frente cuadrado de color rojo, si en un momento dado el estado de la VB en cuestión es alto, el LED virtual mencionado dará la apariencia de estar encendido, en otro caso la apariencia será de apagado.

Los pasos que tiene que seguir el usuario, para monitorear en tiempo real la operación del PLM, cuando el mismo ejecuta un determinado programa, podrían ser los siguientes:

1.- Invocar la subopción "Generar código a partir de programa fuente SIL", de la opción "Obtener código para PLM desde:" del menú archivo. Suponiendo que el programa fuente estuviera contenido en un archivo denominado TEMDISP.SIL aparecería el cuadro de dialogo mostrado en la figura 4.12 donde el usuario ha marcado ya el archivo fuente.

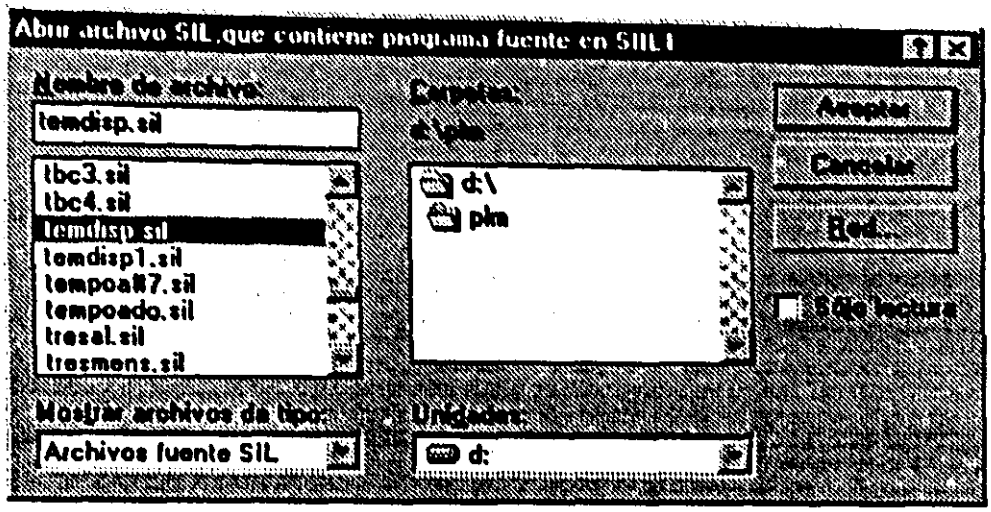


Figura 4.12 Cuadro de dialogo al solicitar al SPDPLM, la generación de código objeto a partir de un determinado programa fuente. que en esta figura es el contenido en el archivo TEMDISP.SIL.

2.- Después de oprimir el botón aceptar en el cuadro de dialogo mencionado en el paso anterior', el SPDPLM generará el archivo objeto correspondiente que para lo aqui mostrado se denominará como TEMDISP.BLM.

3.- Invocar la opción "Ejecutar archivo BLM presente", que es parte del menú ejecutar que se explicará más adelante; al hacerse esto el SPDPLM automáticamente baja al PLM el programa objeto generado, colocándolo en la posición de memoria adecuada, ejecutándose el mismo de inmediato.

3.- Invocar la opción "Ver panel frontal virtual del PLM", esto en el caso de que dicho panel no estuviera habilitado; en la figura 4.13 se muestra el aspecto que para el usuario tendria el Panel frontal del PLM, nótese ahí la forma natural en que aparecen agrupadas las variables de entrada y salida, además del aspecto que tienen los mimicos que muestran el estado de la UD y del RTR.

4.- Oprimir el botón desvanecido que ostenta la leyenda "Habilitar observación en tiempo real", después de lo cual la leyenda del mismo pasará a ser: "Observación en tiempo real habilitada", véase la figura 4.13, donde aparece lo que veria en pantalla el usuario al monitorear la operación del PLM cuando el mismo ejecutara el programa contenido en el archivo fuente TEMDISP.SIL.

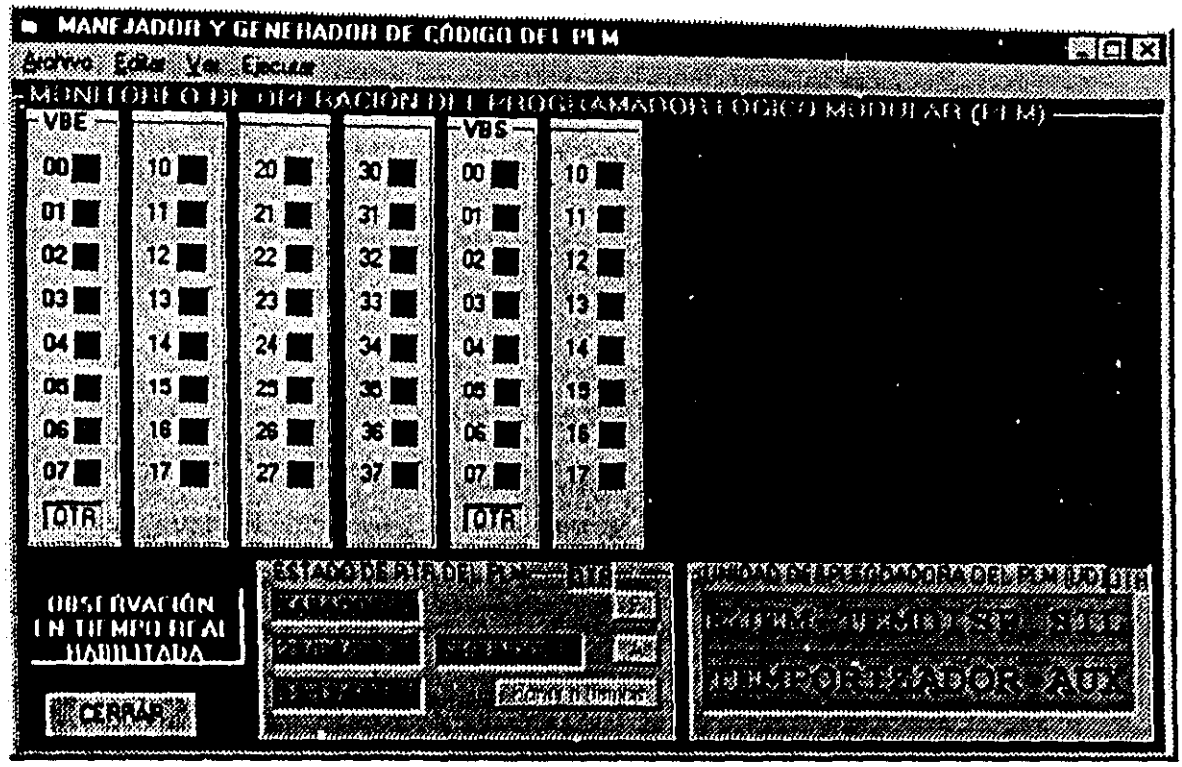


Figura 4.13 Panel frontal del PLM cuando el SPDPLM monitorea en tiempo real la ejecución de un programa, aquí se muestra lo consecuente cuando el programa ejecutándose es el contenido en el archivo TEMDISP.SIL.

Para salir del panel frontal virtual bastará con oprimir el botón "cerrar" contenido en el mismo.

En la figura 4.14 se muestra el archivo TEMDISP.SIL, tal como lo vería el usuario al habilitar la ventana del editor y abrir el archivo mencionado; nótese la existencia de un mensajero con mensaje fijo en el primer renglón, con texto "EJEM-TEMDISP.SIL" y mensaje móvil a desplegarse en el segundo renglón de la UD. Compárese esto con lo mostrado en la figura 4.13.

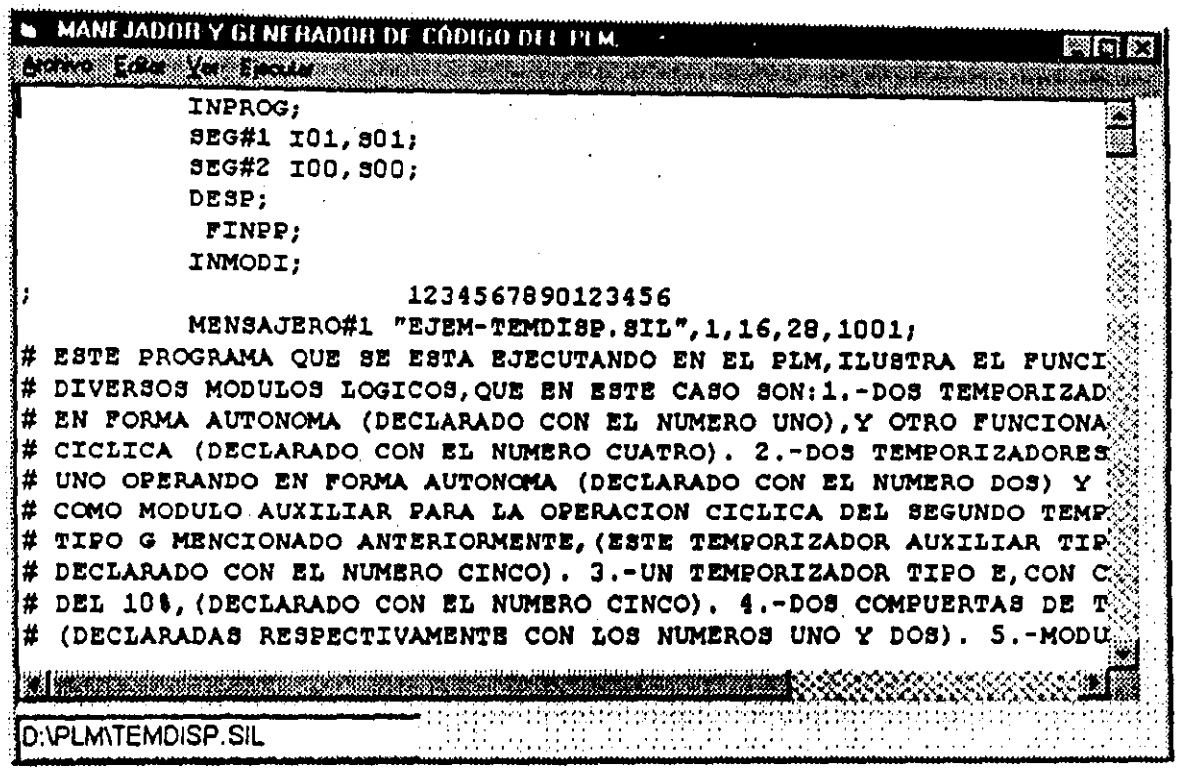


Figura 4.14 Aspecto de la ventana del editor al abrirse el archivo fuente TEMDISP.SIL.

Ver reporte de última generación de código

Al invocarse esta opción se despliega un reporte que contiene información acerca del resultado de la última generación de código, mostrándose la procedencia del código fuente así como también el número de errores de sintaxis detectados y la ubicación y tipo de los mismos, en caso de no haber errores se despliega el tamaño en bytes del código objeto generado.

En la figura 4.15 se muestra el aspecto que tendría este reporte, al invocarse la opción aquí explicada, después de que el SPDPLM ha generado el código correspondiente al programa fuente contenido en el archivo TEMDISP.SIL.

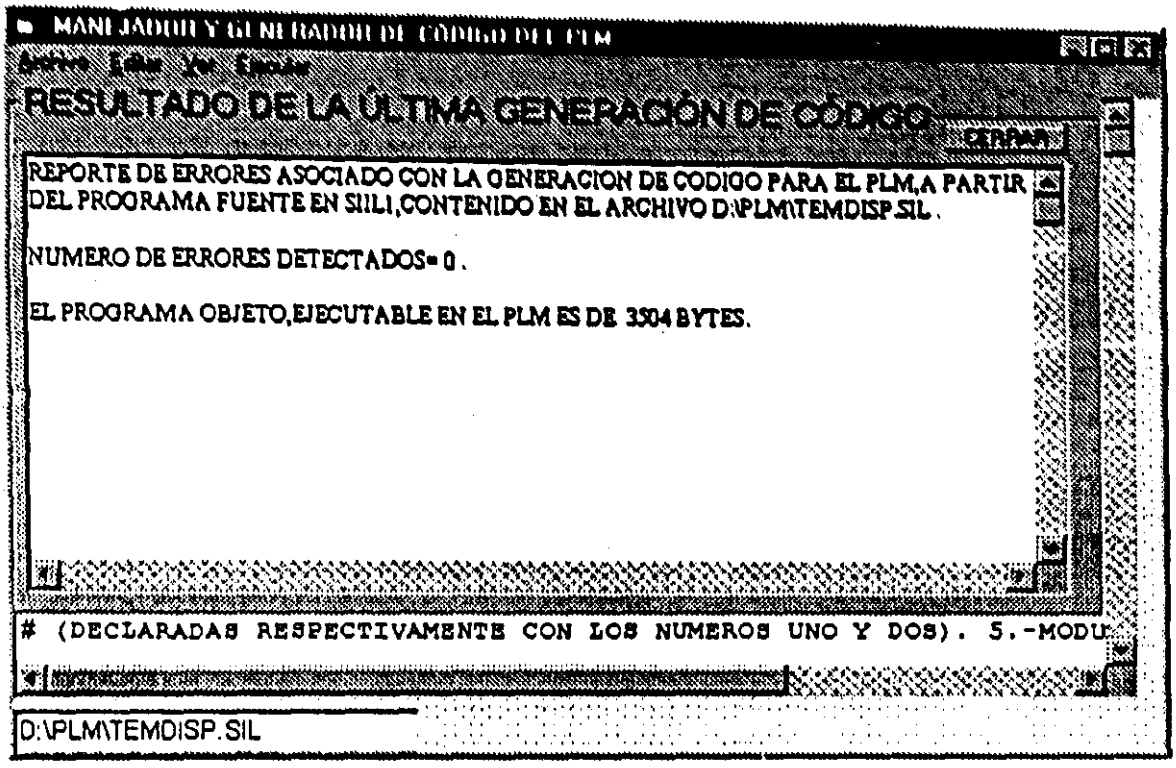


Figura 4.15 Aspecto que presenta al usuario el reporte de la última generación de código para el PLM.

4-4-4 Descripción de las opciones del menú "Ejecutar" del SPDLM

Las opciones del menú "Ejecutar" son enumeradas y descritas a continuación:

Ejecutar archivo BLM presente

Al invocarse esta opción se baja al PLM para su ejecución inmediata, el último archivo objeto BLM que haya sido abierto.

Ejecutar el programa fuente contenido en la ventana del editor

Esta opción genera el código objeto correspondiente con el programa fuente, que estuviera contenido en la ventana del editor, en caso de no detectarse errores de sintaxis el código objeto es bajado al PLM para su ejecución inmediata, en otro caso se muestra automáticamente el reporte de errores que corresponda.

Ejecutar archivo fuente SIL

Al invocarse esta opción aparece un cuadro de dialogo para abrir un archivo SIL, una vez que el usuario a especificado el archivo y oprimido el botón aceptar, se genera el código objeto correspondiente, en caso de no haberse detectado errores de sintaxis se baja al PLM el código generado para su ejecución inmediata, en otro caso se muestra automáticamente el reporte de errores que corresponda.

Ejecutar archivo objeto BLM

Al invocarse esta opción aparece un cuadro de dialogo para abrir un archivo

objeto BLM, una vez que el usuario a especificado el archivo y oprimido el botón aceptar, se baja al PLM el código objeto asociado para su ejecución inmediata.

Parar ejecución

Esta opción detiene en el PLM la ejecución del programa que el mismo esté ejecutando, pasando la CC del mismo a estar en capacidad de recibir una orden generada por el SPDPLM como podrá ser reiniciar la ejecución, ó bien el envío de un programa objeto diferente para su carga en memoria y ejecución inmediata.

Reiniciar ejecución

Al invocarse esta opción, se reinicia la ejecución del programa que se haya interrumpido, al invocarse la opción anterior.

Esto concluye la descripción general de la operación del SPDPLM desde el punto de vista del usuario final.

CAPÍTULO 5
EJEMPLO DE APLICACIÓN

En este capítulo se presenta un ejemplo de aplicación del PLM, la intención del mismo es mostrar el potencial que el dispositivo desarrollado podría tener para automatizar procesos, siendo el alcance del caso aquí mostrado únicamente de tipo ilustrativo.

El capítulo inicia con una descripción general del proceso por automatizar, y la manera con que el mismo interactúa con otro proceso que recibe el producto generado, más adelante se presenta un esquema donde se muestran los sensores y actuadores involucrados y que variables del PLM estarán asociadas en cada caso, para pasar después a describir los diversos módulos empleados en la automatización, concluyendo con la presentación del programa en SIIL1 que debe ejecutar el PLM para realizarla.

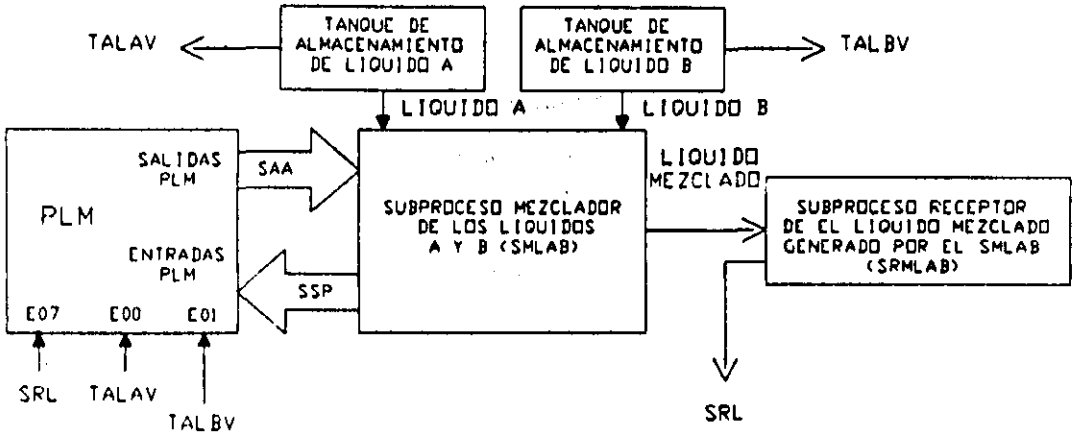
5-1 DESCRIPCIÓN DE UN PROCESO SUSCEPTIBLE DE SER AUTOMATIZADO EMPLEANDO EL PLM

Un determinado proceso industrial, puede dividirse en una cadena de subprocesos eslabonados, uno de esos subprocesos requiere que en un tanque, aquí denotado con la letra C, se mezclen dos líquidos A y B en una proporción volumétrica de uno a dos, antes de mezclar los líquidos A y B los mismos deben ser irradiados por separado con una lampara infrarroja, esto último hace necesario el empleo de dos tanques auxiliares donde se deben colocar las materias primas aquí mencionadas para ser irradiadas, después de lo cual se pasan al tanque de mezclado, una vez que los dos líquidos se encuentran en el tanque mencionado, se debe activar un agitador contenido en el mismo por treinta minutos, después de lo cual, se ha de descargar la mezcla mediante una bomba, de modo que el subproducto obtenido sea empleado por otro subproceso.

Al subproceso mezclador objeto de este ejemplo, se le denominará de aquí en adelante como subproceso de mezclado de líquidos A y B (SMLAB), y al subproceso receptor que recibe el subproducto generado por el SMLAB se le denotará como SRMLAB, en la figura 5.1 se ilustra el eslabonamiento entre estos dos subprocesos, nótese la señal SRL (subproceso receptor listo), mediante la cual se testifica que el SRMLAB está en condiciones de recibir una descarga de subproducto, además en la misma figura, se aprecia la existencia de otras dos señales de entrada al PLM (TALAV y TALBV) que testifican el

hecho de que alguno de los tanques de almacenamiento que suministran la materia prima que emplea el SMLAB, se encuentre vacío.

El nivel de verificación para la señal SRL es alto y el correspondiente a las señales TALAV y TALBV es bajo, nótese que en la figura 5.1 se indican también las entradas del del PLM que se emplearán para captar las tres señales aquí mencionadas.



SAA = SEÑALES DE ACTIVACION DE ACTUADORES ASOCIADOS CON EL SMLAB.

SSP = SEÑALES GENERADAS POR SENSORES LIGADOS AL SMLAB.

SRL = TESTIGO DE SUBPROCESO RECEPTOR LISTO PARA RECIBIR PRODUCTO DEL SMLAB.

TALAV = TESTIGO DE TANQUE DE ALMACENAMIENTO DE LIQUIDO B VACIO

TALBV = TESTIGO DE TANQUE DE ALMACENAMIENTO DE LIQUIDO B VACIO.

Figura 5.1 Esquema global del subproceso de mezclado de dos líquidos A y B (SMLAB) a ser automatizado por el PLM, y su eslabonamiento con el subproceso receptor del subproducto generado.

A cada descarga de producto de la mezcla mencionada en el párrafo anterior se le denomina lote, debiendo suministrarse 20 de ellos en una jornada. Las materias primas aquí representadas por los líquidos A y B están contenidas en dos tanques de almacenamiento denominados como A y B; para cada líquido existe un tanque auxiliar con volúmenes en la proporción requerida, donde se lleva a cabo la irradiación infrarroja mencionada en párrafos anteriores; así, el tanque auxiliar B tendrá un volumen igual al doble del asociado con el tanque auxiliar A; de esta forma, para lograr la proporción volumétrica requerida primero se suben los líquidos A y B a sus respectivos tanques auxiliares, para después

descargarlos al tanque C donde se ha de efectuar el mezclado mediante el agitador mencionado en el párrafo anterior.

En la figura 5.2 se muestra la disposición de las componentes del SMLAB, apreciándose en la misma las variables del PLM asociadas, las cuales serán señales de arranque para los motores de las bombas y el agitador (variables de salida del PLM) y señales suministradas por los sensores de nivel (variables de entrada al PLM). Se supone que los sensores de nivel son booleanos, proporcionando cada uno de ellos un nivel de uno lógico, cuando el nivel del líquido rebasa la posición del mismo, en otro caso el nivel lógico asociado será cero.

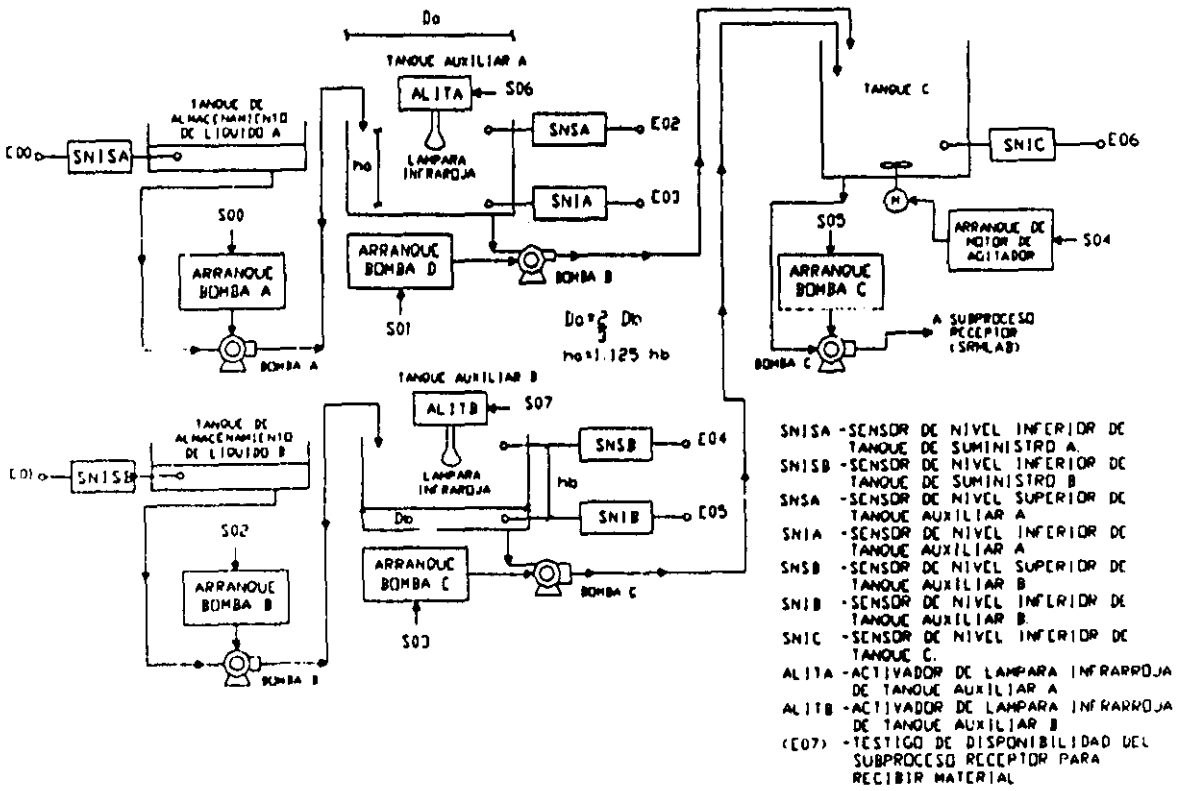


Figura 5.2 Esquema del subproceso de mezclado de dos líquidos A y B que ha de ser automatizado por el PLM.

5.1-1 Descripción general del subproceso de mezclado de los líquidos A y B (SMLAB)

Inicialmente tanto los tanques auxiliares A y B como el tanque de mezclado deben estar vacíos, se requiere que al oprimirse el botón de restablecimiento del PLM, se inicie la rutina lógica que suministrará al subproceso receptor los veinte lotes, la secuencia sería la siguiente:

1.- Se llenan los dos tanques auxiliares empleándose para ello a las bombas A y B, esto requerirá que las señales de arranque correspondientes (S00 y S02) presenten un nivel alto, en caso que un determinado tanque de almacenamiento se encuentre vacío, se deberá poner en cero la señal de activación (S00 ó S02) que corresponda, notificando esta situación al operador mediante un mensaje móvil en la UD.

2.- Una vez que el tanque auxiliar A se ha llenado, el contenido del mismo deberá ser transferido al tanque de mezclado empleando para ello a la bomba D, lo que requerirá que la señal de activación de la misma (S01), presente un nivel de uno lógico, desde luego que el nivel de la salida S00 (activación de la bomba A) deberá pasar a cero.

Durante el curso de llenado y vaciado de los tanques auxiliares, deberán estar activadas las lamparas infrarrojas de cada uno.

3.- Una vez que el tanque auxiliar B se ha llenado, el contenido del mismo deberá ser transferido al tanque de mezclado empleando para ello a la bomba E, lo que requerirá que la señal de activación de la misma (S03), presente un nivel de uno lógico, desde luego que el nivel de la salida S02 (activación de la bomba B) deberá pasar a cero.

4.- Al completarse los dos pasos anteriores se deberá verificar por treinta minutos la señal de activación del motor del agitador (S04).

5.- Al completarse el mezclado del paso anterior (flanco de bajada de S04), se deberá verificar en alto la salida S05, que activa la bomba C, esto sujeto al permisivo SRL (entrada E07) del PLM, transfiriéndose al siguiente subproceso receptor la mezcla de los líquidos A y B obtenida en el paso cuatro.

6.- Si no se han despachado los veinte lotes requeridos, se deberá pasar al paso uno, para iniciar de nuevo el subproceso, en otro caso se deberá notificar al operador que se ha completado la cuota diaria de lotes. Para reiniciar el despacho de otros veinte lotes se deberá oprimir el botón de restablecimiento del PLM.

La realización mediante el PLM, de la rutina lógica descrita a grandes rasgos en párrafos anteriores, puede ser hecha de diversas maneras, aquí se presentará una de ellas, que emplea secuenciadores de estado, temporizadores monodisparo, diversas compuertas lógicas, módulos de tipo alarma y módulos de tipo mensajero.

El proceso de diseño del programa en SIIL1 que valide esta aplicación puede desdoblarse en los siguientes pasos:

1.- Selección de módulos y diseño de interconexión lógica de los mismos, para controlar el llenado y descarga del tanque auxiliar A.

2.- Selección de módulos y diseño de interconexión lógica de los mismos, para controlar el llenado y descarga del tanque auxiliar B.

3.- Selección de módulos y diseño de interconexión lógica de los mismos, para controlar el mezclado en el tanque C y la descarga del mismo a otro subproceso.

4.- Selección de un módulo contador de eventos que lleve la cuenta de los lotes despachados. En conjunción con este contador, se empleará un módulo observador de contador de eventos, para que el operador pueda apreciar en la UD, la cuenta aquí mencionada.

5.- Determinación de los distintos mensajes de alarma o información de flujo del subproceso de mezcla volumétrica arbitrado por el PLM, que podrían ser de alguna utilidad para el operador.

6.- Identificación de las variables booleanas que dispararían los mensajes de alarma mencionados en el paso anterior.

7.- En base a la información asociada con los dos pasos anteriores especificar los módulos mensajeros y de alarma necesarios.

A continuación se detallan los módulos empleados para cada uno de los puntos anteriores así como también el interconexionado entre ellos, para una mejor comprensión es recomendable observar la figura 5.2 conforme se avance en la lectura de las secciones 5-1-2 a 5-1-6.

5-1-2 Módulos empleados para realizar la automatización del llenado y descarga del tanque auxiliar A

Para la realización de la rutina lógica de llenado y vaciado del tanque auxiliar A, se emplearon un secuenciador de estados de 2x3, tres temporizadores monodisoparo y tres compuertas lógicas; en la figura 5.3 se aprecian los módulos empleados y las interconexiones entre ellos, el temporizador número tres (TEMPOC#3) tiene como función el generar la señal de restablecimiento del secuenciador implicado, esto debe acontecer, cada que se ha completado una descarga de subproducto, al subproceso receptor (flanco de bajada del sensor de nivel SNIC (entrada E06 del PLM), de modo que se pueda iniciar un nuevo ciclo de mezclado siempre que no se haya completado la cuota de veinte lotes; por lo tanto, la señal de salida del temporizador número tres (I06), también será empleada por otros módulos implicados en la automatización del subproceso SMLAB.

Los otros dos temporizadores mostrados en la figura 5.3 (TEMPOC#1 y TEMPOC#2), sirven para evitar disparos espurios del secuenciador (SEC2#1), debidos a rebotes en las señales generadas por los sensores de nivel SNIA y SNSA.

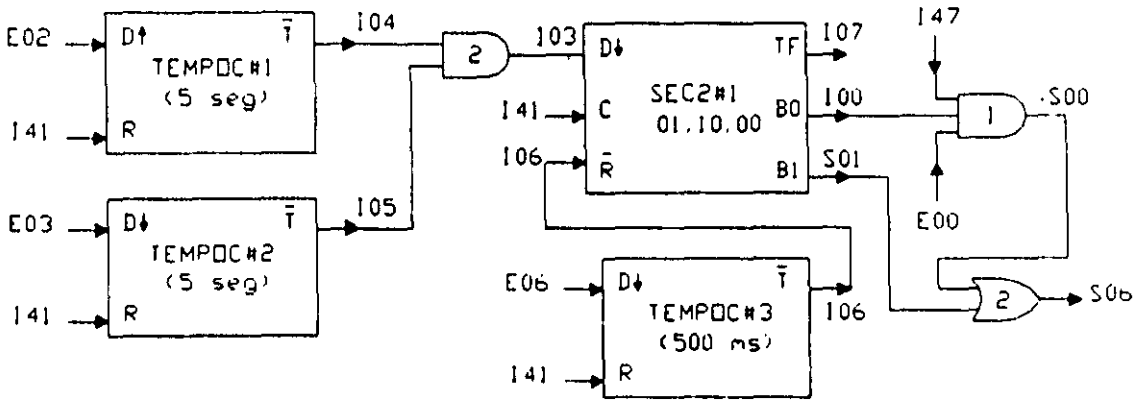


Figura 5.3 Módulos lógicos empleados en la realización de la rutina lógica de llenado y vaciado del tanque auxiliar A.

Configuración del secuenciador de estados SEC2#1 mostrado en la figura 5.3

Las entradas de disparo, congelamiento y restablecimiento de este temporizador son las VBI I03, I41 e I06, siendo el disparo del mismo por flanco de bajada y los niveles de verificación de las entradas de congelamiento y restablecimiento son respectivamente alto y bajo. El testigo de fin de carrera es la VBI I07 y se verifica en nivel alto.

La función de cada uno de los dos bits de salida de este secuenciador se explica a continuación:

El bit menos significativo es la VBI I00, que al presentar un nivel de uno lógico hace que se active la bomba A, siempre que haya líquido en el tanque de almacenamiento A (E00 en alto) y que la VBI I47 presente un nivel alto, lo cual acontecerá siempre que no se hayan despachado los veinte lotes, véase la compuerta AND3#1 en la figura 5.3 y más adelante, lo concerniente con el contador de eventos empleado para contar los lotes despachados.

El otro bit es la VBS S01 que activa a la bomba D la cual descarga al tanque de mezclado el contenido del tanque auxiliar A, una vez que este último se ha llenado.

En la tabla 5.1 se muestra la secuencia de estados requerida para el secuenciador SEC2#1, indicándose ahí también los retroavisos asociados.

Tabla 5.1 secuencia de estados del secuenciador SEC2#1, empleado en la realización de la rutina lógica de llenado y vaciado del tanque auxiliar A.

ESTADO	B1 (S01)	B0 (I00)	Retroaviso para avance	Retorno al estado inicial
1 (inicial)	0	1	Flanco de subida de E02	
2	1	0	Flanco de bajada de E03	
3	0	0		Nivel bajo en VBI I06

De acuerdo con lo explicado en párrafos anteriores, la declaración en SIIL1 del secuenciador SEC2#1 es:

SEC2#1 I03,I41,I06,I07,S01,I00,3,0111;

B01, B10, B00;

Declaraciones en SIIL1 asociadas con las compuertas lógicas y temporizadores implicados en el llenado y vaciado del tanque auxiliar A

De acuerdo con lo mostrado en la figura 5.3, las declaraciones en SIIL1 para cada uno de los temporizadores y compuertas lógicas ahí mostradas, se detalla a continuación:

Para el temporizador monodisparo número uno (TEMPOC#1) la declaración es:

TEMPOC#1 E02, I41, I04, 00:00:05.00, 100;

Para el temporizador monodisparo número dos (TEMPOC#2) la declaración es:

TEMPOC#2 E03, I41, I05, 00:00:05.00, 000;

Para el temporizador monodisparo número tres (TEMPOC#3) la declaración es:

TEMPOC#2 E06, I41, I06, 00:00:00.50, 000;

Para la compuerta and de tres entradas (AND3#1) la declaración es:

AND3#1 I47, E00, I00, S00, 111;

Para la compuerta and de dos entradas (AND2#2) la declaración es:

AND#2 I04, I05, I03, 11;

Para la compuerta or de dos entradas (OR2#2) la declaración es:

OR2#2 S00, S01, S06, 11;

5-1-3 Módulos empleados para realizar la automatización del llenado y descarga del tanque auxiliar B

Para la realización de la rutina lógica de llenado y vaciado del tanque auxiliar B, se emplearon un secuenciador de estados de 2x3, tres temporizadores monodisoparo y tres compuertas lógicas; en la figura 5.4 se aprecian los módulos empleados y las interconexiones entre ellos, nótese que la señal de restablecimiento del secuenciador implicado, es la VBI I06 que es la salida del temporizador monodisparo número tres (TEMPOC#3) mostrado en la figura 5.3, esto debe acontecer, cada que se ha completado una descarga de subproducto, al subproceso receptor (flanco de bajada del sensor de nivel SNIC (entrada E06 del PLM), de modo que se pueda iniciar un nuevo ciclo de mezclado siempre que no se haya completado la cuota de veinte lotes; por lo tanto, la señal de salida del temporizador número tres (106), también será empleada por otros módulos implicados en la automatización del subproceso SMLAB

Los otros dos temporizadores mostrados en la figura 5.4 (TEMPOC#4 y TEMPOC#5), sirven para evitar disparos espurios del secuenciador (SEC2#2), debidos a rebotes en las señales generadas por los sensores de nivel SNIB y SNSB.

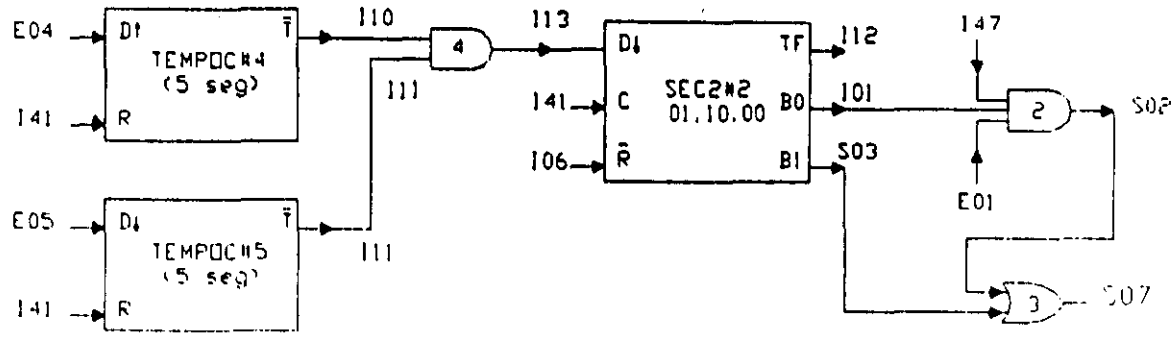


Figura 5.4 Módulos lógicos empleados en la realización de la rutina lógica de llenado y vaciado del tanque auxiliar B.

Configuración del secuenciador de estados SEC2#2 mostrado en la figura 5.4

Las entradas de disparo, congelamiento y restablecimiento de este temporizador son las VBI I13, I41 e I06, siendo el disparo del mismo por flanco de bajada y los niveles de verificación de las entradas de congelamiento y restablecimiento son respectivamente alto y bajo. El testigo de fin de carrera es la VBI I12 y se verifica en nivel alto.

La función de cada uno de los dos bits de salida de este secuenciador se explica a continuación:

El bit menos significativo es la VBI I01, que al presentar un nivel de uno lógico hace que se active la bomba B, siempre que haya líquido en el tanque de almacenamiento B (E01 en alto) y que la VBI I47 presente un nivel alto, lo cual acontecerá siempre que no se hayan despachado los veinte lotes, véase la compuerta AND3#3 en la figura 5.4 y más adelante, lo concerniente con el contador de eventos empleado para contar los lotes despachados.

El otro bit es la VBS S03 que activa a la bomba D la cual descarga al tanque de mezclado el contenido del tanque auxiliar B, una vez que este último se ha llenado.

En la tabla 5.2 se muestra la secuencia de estados requerida para el secuenciador SEC2#2, indicándose ahí también los retroavisos asociados.

Tabla 5.2 secuencia de estados del secuenciador SEC2#2, empleado en la realización de la rutina lógica de llenado y vaciado del tanque auxiliar B.

ESTADO	B1 (S03)	B0 (I01)	Retroaviso para avance	Retorno al estado inicial
1 (inicial)	0	1	Flanco de subida de E04	
2	1	0	Flanco de bajada de E05	
3	0	0		Nivel bajo en VBI I06

De acuerdo con lo explicado en párrafos anteriores, la declaración en SIIL1 del secuenciador SEC2#2 es:

SEC2#2 I13,I41,I06,I07,S01,I00,3,0111;

B01, B10, B00;

Declaraciones en SIIL1 asociadas con las compuertas lógicas y temporizadores implicados en el llenado y vaciado del tanque auxiliar B

De acuerdo con lo mostrado en la figura 5.4, las declaraciones en SIIL1 para cada uno de los temporizadores y compuertas lógicas ahí mostradas, se detalla a continuación:

Para el temporizador monodisparo número cuatro (TEMPOC#4) la declaración es:

TEMPOC#4 E04, I41, I10, 00:00:05.00, 100;

Para el temporizador monodisparo número cinco (TEMPOC#5) la declaración es:

TEMPOC#5 E05, I41, I11, 00:00:05.00, 000;

Para la compuerta and de tres entradas (AND3#3) la declaración es:

AND3#3 I47, E01, I01, S02, 111;

Para la compuerta and de dos entradas (AND2#2) la declaración es:

AND2#2 I10, I11, I13, 11;

Para la compuerta or de dos entradas (OR2#3) la declaración es:

OR2#3 S02, S03, S07, 11;

5-1-4 Módulos empleados para realizar la automatización del mezclado de los líquidos A y B en el tanque C

Para la realización de la rutina lógica que valida el mezclado de los líquidos A y B, que se lleva a cabo en el tanque C, se emplearon un secuenciador de estados de 2x3, cuatro temporizadores monodisoparo y tres compuertas lógicas; en la figura 5.5 se aprecian los módulos empleados y las interconexiones entre ellos, nótese que la señal de restablecimiento del secuenciador implicado, es la VBI I06 que es la salida del temporizador monodisparo número tres (TEMPOC#3) mostrado en la figura 5.3, esto debe acontecer, cada que se ha completado una descarga de subproducto, al subproceso receptor (flanco de bajada del sensor de nivel SNIC (entrada E06 del PLM), de modo que se pueda iniciar un nuevo ciclo de mezclado siempre que no se haya completado la cuota de veinte lotes; por lo tanto, la señal de salida del temporizador número tres (I06), también será empleada por otros módulos implicados en la automatización del subproceso SMLAB.

Los otros tres temporizadores mostrados en la figura 5.5 (TEMPOC#6, TEMPOC#7 y TEMPOC#8), sirven, los dos primeros, para generar los dos retroavisos asociados con el secuenciador (SEC2#3), y el último delimita el tiempo que ha de estar activado el agitador para llevar a cabo el mezclado de los líquidos A y B.

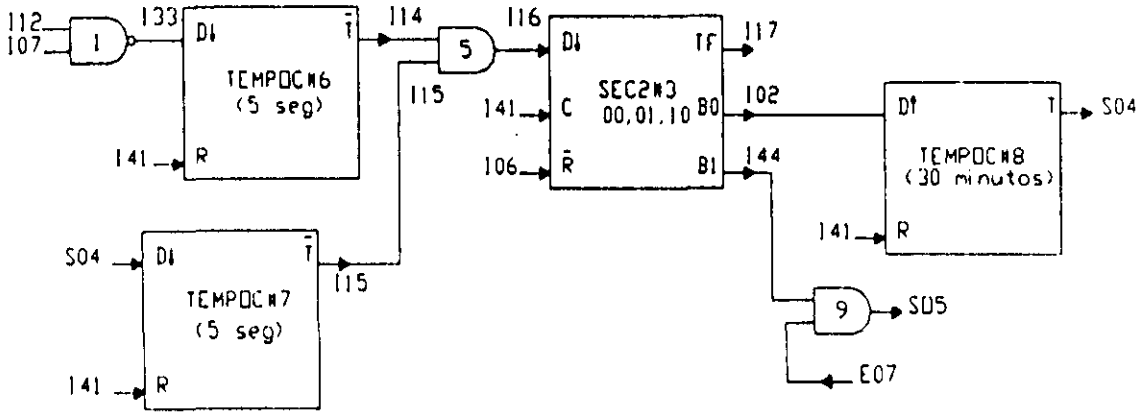


Figura 5.5 Módulos lógicos empleados en la realización de la rutina lógica para efectuar el mezclado de los líquidos A y B en el tanque C.

Configuración del secuenciador de estados SEC2#3 mostrado en la figura 5.5

Las entradas de disparo, congelamiento y restablecimiento de este temporizador son las VBI I16, I41 e I06, siendo el disparo del mismo por flanco de bajada y los niveles de verificación de las entradas de congelamiento y restablecimiento son respectivamente alto y bajo. El testigo de fin de carrera es la VBI I17 y se verifica en nivel alto.

La función de cada uno de los dos bits de salida de este secuenciador se explica a continuación:

El bit menos significativo es la VBI I02, que al presentar un flanco de subida hace que el temporizador monodisparo active por treinta segundos el motor que mueve el agitador.

El otro bit es la VBI I44 que activa a la bomba C, que descarga al subproceso receptor el resultado de la mezcla de los líquidos A y B, esto último siempre y cuando el testigo de que el subproceso receptor está listo (VBE E07 en alto).

En la tabla 5.3 se muestra la secuencia de estados requerida para el secuenciador SEC2#3, indicándose ahí también los retroavisos asociados.

Tabla 5.3 secuencia de estados del secuenciador SEC2#3, empleado en la realización de la rutina lógica para efectuar el mezclado de los líquidos Ay B en el tanque C.

ESTADO	B1 (I44)	B0 (I02)	Retroaviso para avance	Retorno al estado inicial
1 (inicial)	0	0	Flanco de bajada de I33*	
2	0	1	Flanco de bajada de S04**	
3	1	0		Nivel bajo en VBI I06

* La VBI I33 presentará un flanco de bajada, cada vez que se completa el vaciado de los tanques auxiliares A y B al tanque de mezclado C, ya que la misma es la salida de una compuerta NAND (NAND2#1) cuyas dos entradas son los testigos de fin de carrera de los dos secuenciadores que arbitran el llenado y vaciado de dichos tanques auxiliares.

** El flanco de bajada de la VBS S04 marca el fin del intervalo de tiempo que el motor del agitador en el tanque C está activado.

De acuerdo con lo explicado en párrafos anteriores, la declaración en SIIL1 del secuenciador SEC2#3 es:

SEC2#3 I16, I41, I06, I17, S05, I02, 3, 0111;

B00, B01, B10;

Declaraciones en SIIL1 asociadas con las compuertas lógicas y temporizadores implicados en el mezclado, en el tanque C, de los líquidos A y B.

De acuerdo con lo mostrado en la figura 5.4, las declaraciones en SIIL1 para cada uno de los temporizadores y compuertas lógicas ahí mostradas, se detalla a continuación:

Para el temporizador monodisparo número seis (TEMPOC#6) la declaración es:

TEMPOC#6 I33, I41, I14, 00:00:00.50, 000;

Para el temporizador monodisparo número siete (TEMPOC#7) la declaración es:

TEMPOC#7 S04, I41, I15, 00:00:00.50, 000;

Para el temporizador monodisparo número ocho (TEMPOC#8) la declaración es:

TEMPOC#8 I02, I41, S04, 00:30:00.00, 000;

Para la compuerta and de dos entradas (AND2#5) la declaración es:

AND2#5 I14, I15, I16, 111;

Para la compuerta and de dos entradas (AND2#9) la declaración es:

AND2#9 I44, E07, 11;

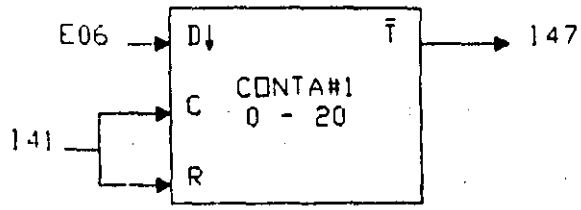
Para la compuerta nand de dos entradas (NAND2#1) la declaración es:

NAND2#1 I12, I07, I33, 11;

5-1-5 Descripción de los módulos empleados para llevar la cuenta de lotes despachados y observar la misma en la UD

Para llevar la cuenta de lotes despachados, se emplea un módulo contador de eventos ascendente con cuenta de cero a veinte, con avance de cuenta por flanco de bajada de la VBE E06 (SENSOR SNIC), siendo las entradas de congelamiento y restablecimiento habilitadadas por la VBI 141, con niveles de verificación alto y bajo respectivamente; para testigo de fin de carrera del módulo contador se seleccionó a la VBI 147 con verificación en bajo; la declaración para el contador de eventos es:

CONTA#1 E06, I41, I41, I47, 0, 20, 01010;



En la figura 5.6 se muestra la representación como bloque del módulo contador de eventos empleado en la automatización del subproceso SMLAB.

Para mostrar en la UD el número de lotes despachados, se empleó un módulo observador de contador de eventos, definido para que la cuenta se despliegue a dos dígitos a partir de la primera columna del renglón superior de la UD, la declaración de este módulo es:

OBSCE#1 1, 2, 1;

5-1-6 Definición de los mensajes a desplegarse en la UD al llevarse a cabo la rutina lógica que valida la automatización del subproceso SMLAB

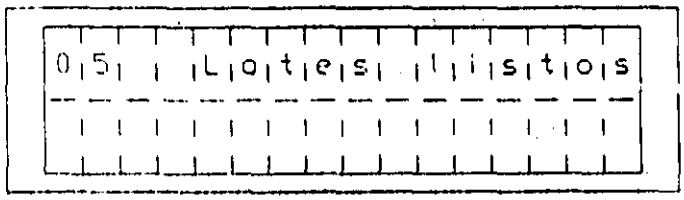
Conforme se vaya llevando a cabo el subproceso de mezclado, en la UD deberán aparecer diversos mensajes que testifiquen entre otros cosas lo siguiente: Condiciones de alarma, o simplemente el aviso al operador acerca de que paso del SMLAB se esta llevando a cabo en un momento dado; o bien el estado del contador de lotes despachados al subproceso reseptor (SRMLAB).

Para el despliegue de la cuenta de lotes despachados, se seleccionó el primer renglón de la UD, donde ese número deberá aparecer a partir de la primera y segunda columna, seguido por un mensaje fijo cuyo texto es " Lotes listos", para realizar esto se usó un módulo observador de contador de eventos ya descrito en la sección anterior; para la implantación del mensaje fijo se empleó un módulo mensajero con mensajes fijo y móvil en el renglón uno de la UD, con deshabilitación del desplegado del mensaje móvil y habilitación de operación independiente del número de módulo mensajero activo, esto último hace que lo desplegado en el renglón uno de la UD sea independiente de los diversos mensajes de alarma y/o aviso que se despliegan en el renglón dos de la UD. La declaración correspondiente al módulo mensajero del que se habla en este párrafo es:

MENSAJERO#13 " Lotes listos",1,11,1,1110;

Este mensaje móvil nunca se desplegará

En la figura 5.7 se ilustra lo que se vería en la UD al ejecutarse el en el PLM el programa correspondiente a la automatización del SMLAB, cuando ya se han despachado cinco lotes al subproceso receptor.



NOTA : EN EL SEGUNDO RENGLON SE DESPLEGARIA UN MENSAJE DE ALARMA Y/O AVISO. VEASE LA TABLA 5 4

Figura 5.7 Despliegue en la UD cuando ya se han despachado cinco lotes al subproceso receptor, al ejecutarse en el PLM el programa asociado con la automatización del SMLAB.

Módulos de tipo alarma y mensajero empleados en la automatización del SMLAB para el despliegue de mensajes de alarma o información acerca del flujo del proceso

En el renglón dos de la UD se desplegarán los distintos mensajes móviles, al ejecutarse el programa que valida la automatización del SMLAB, esto debido a la ocurrencia de diversos eventos, que pueden ser de alarma o simplemente informativos, todos estos mensajes carecen de texto fijo y se moverán con una cadencia de 20 centésimas de segundos entre posiciones sucesivas. En la tabla 5.4 se muestra el texto para cada caso, así como también la VB (denotada como VBD), que dispara cada mensaje y el nivel de verificación de esta última para que esto suceda; el orden de prioridad de los mensajes es el mismo en el que cada uno de ellos aparece en la tabla.

Tabla 5.4 Relación de mensajes de alarma e informativos a desplegarse en el renglón dos de la UD al llevarse a cabo la automatización del SMLAB empleando el PLM.

VBD	NVVD/NM	Texto del mensaje móvil a desplegar al verificarse la VBD asociada
130	Bajo/1	Tanques de almacenamiento de líquidos A y B vacíos.
E00	Bajo/2	Tanque de almacenamiento de líquido A vacío.
E01	Bajo/3	Tanque de almacenamiento de líquido B vacío
147	Bajo/4	Se ha completado el proceso de mezclado de la cuota diaria de lotes. Para reiniciar oprimir botón de reset del PLM.

Continuación Tabla 5.4

S00	Alto/6	Llenado de tanque auxiliar A en curso, (bomba A activada).
I31	Alto/5	Llenado de tanques auxiliares A y B en curso, (bombas A y B activadas).
S00	Alto/6	Llenado de tanque auxiliar A en curso, (bomba A activada).
S02	Alto/7	Llenado de tanque auxiliar B en curso, (bomba B activada).
I32	Alto/8	Descarga de tanques auxiliares A y B en curso, (bombas D y E activadas).
S01	Alto/9	Descarga de tanques auxiliar A en curso, (bomba D activada).
S03	Alto/10	Descarga de tanques auxiliar B en curso, (bomba E activada).
S04	Alto/11	Proceso de mezclado en tanque C activado.
S05	Alto/12	Descarga de tanque mezclador en curso, (bomba C activada).

Nota:
VBD, denota a la variable booleana que dispara cada mensaje.
NVVD, denota el nivel de verificación de la variable boolana para que se dispare el mensaje.
NM, denota el número de módulo mensajero asociado con el texto móvil a desplegar en cada caso.

El mensaje móvil que por defecto se despliega en la UD, cuando ninguna de las VB implicadas en la tabla 5.4, presenta el nivel para que se dispare el mensaje asociado con ella es el siguiente: "Subproceso receptor de lotes despachados, no listo.", y desde luego que el número de mensajero correspondiente debe ser cero; de esta forma, el operador observará este último mensaje en la UD cuando el SMLAB este en el paso que descarga al subproceso receptor el subproducto generado siempre y cuando no se haya dado una condición de alarma, una vez que el SRMLAB indique que está listo para recibir subproducto, en la pantalla de la UD deberá aparecer el texto correspondiente a la verificación en alto de la VBS S05, véase el último renglón de la tabla 5.4.

A continuación se describen las operaciones lógicas empleadas para obtener las VBI I30, I47, I31 e I32; que disparan respectivamente a los mensajes de prioridad uno, cuatro, cinco y ocho; véase la tabla 5.4.

La VBI I30 se obtiene como la salida de una compuerta OR de dos entradas, que son las VBE E00 y E01, la declaración correspondiente es:

OR2#1 E00, E01, I30, 11;

La VBI I47 es simplemente el testigo de fin de carrera del contador de eventos empleado para llevar la cuenta de los lotes despachados.

La VBI I31 se obtiene como la salida de una compuerta AND de dos entradas, que son las VBS S00 y S02, la declaración correspondiente es:

AND2#6 S00, S02, I31, 11;

La VBI I32 se obtiene como la salida de una compuerta AND de dos entradas, que son las VBS S01 y S03, la declaración correspondiente es:

AND2#7 S01, S03, I32, 11;

Al programa en SIIL1 que hace que el PLM realice la automatización del SMLAB se le llamó EJAP11 y el mismo está contenido en el archivo de texto EJAP11.SIL, que se muestra a continuación, ahí mismo se pueden apreciar las declaraciones asociadas con los módulos de tipo alarma y mensajero empleados.

```
**********************************************************************
*Programa EJAP11 empleado para realizar la automatización del proceso  *
*de mezclado de los líquidos A y B, el programa está contenido en el   *
*archivo EJAP11.SIL.                                                   *
*Autor: Antonio Salvá Calleja                                          *
*Fecha: 20 / SEP / 1998                                                *
**********************************************************************
            INPROG;
*****************************************************************
* Declaraciones de módulos activadores de mensajes de           *
* alarma.                                                       *
*****************************************************************
    ALARMA#1 I30,0;
    ALARMA#2 E00,0;
    ALARMA#3 E01,0;
```

```
   ALARMA#4 I47,0;
   ALARMA#5 I31,1;
   ALARMA#6 S00,1;
   ALARMA#7 S02,1;
   ALARMA#8 I32,1;
   ALARMA#9 S01,1;
   ALARMA#10 S03,1;
   ALARMA#11 S04,1;
   ALARMA#12 S05,1;

   NAND2#1 I12,I07,I33,11;Disparo de primer retroaviso del
*                         secuenciador tres,(flanco de bajada
*                         de I33).
     OR2#1 E00,E01,I30,11;Disparo del mensaje de alarma de
*                         prioridad máxima (1).
    AND2#6 S00,S02,I31,11;Disparo del mensaje de alarma de
*                         prioridad (5).
    AND2#7 S01,S03,I32,11;Disparo del mensaje de alarma de
*                         prioridad (8).
    OR2#2 S00,S01,S06,11;Activación de irradiación en tanque auxiliar A.

    OR2#2 S02,S03,S07,11;Activación de irradiación en tanque auxiliar B.

  AND3#1 I47,E00,I00,S00,111;Señal de activación de la bomba
*A (S00),generada de origen por el bit de salida I00 del secuenciador
*de 2x3 número uno,asociado con el control de llenado y vaciado
*del tanque B;la activación de S00 está sujeta a los permisivos:

*I47(testigo de fin de carrrera de contador de lotes despachados).

*E00(testigo de presencia de liquido A en tanque de suministro A).


   AND2#2 I04,I05,I03,11;Señal de disparo del secuenciador de 2x3
*                        número uno,asociado con el control del
*                        llenado y vaciado del tanque auxiliar A.
```

```
  AND3#2 I47,E01,I01,S02,111;Señal de activación de la bomba
*B (S02),generada de origen por el bit de salida I01 del secuenciador
*de 2x3 número dos,asociado con el control de llenado y vaciado
*del tanque B;la activación de S02 está sujeta a los permisivos:

*I47(testigo de fin de carrrera de contador de lotes despachados).

*E01(testigo de presencia de liquido B en tanque de suministro B).


   AND2#4 I10,I11,I13,11;Señal de disparo del secuenciador de 2x3
*                        número dos,asociado con el control del
*                        llenado y vaciado del tanque auxiliar B.

   AND2#5 I14,I15,I16,11;Señal de disparo del secuenciador de 2x3
*                        número tres,asociado con el control del
*                        llenado,vaciado y proceso de mezclado en
*                        el tanque C.


        OBSCE#1 1,2,1;Observador a dos digitos del contador de
*               eventos que lleva la cuenta de lotes despachados.

      AND2#9 E07,I44,S05,11;Activación de bomba C.

      DESP;Activación de Unidad Desplegadora (UD) del PLM.

      FINPP;

      INMODI;


   CONTA#1 E06,I41,I41,I47,0,20,01010;Contador de lotes despachados.

   TEMPOC#1 E02,I41,I04,00:00:05.00,100;Temporizador monodisparo
```

```
* auxiliar para la generación del primer retroaviso,asociado con
* el secuenciador de 2x3 número uno que controla el llenado y vaciado
* del tanque auxiliar A.

  TEMPOC#2 E03,I41,I05,00:00:05.00,000;Temporizador monodisparo
* auxiliar para la generación del segundo retroaviso,asociado con
* el secuenciador de 2x3 número uno que controla el llenado y vaciado
* del tanque auxiliar A.


*******************************************
*  Secuenciador asociado con el tanque A. *
    SEC2#1 I03,I41,I06,I07,S01,I00,3,0111;*
## B01,B10,B00;                          *
*******************************************


  TEMPOC#4 E04,I41,I10,00:00:05.00,100;Temporizador monodisparo
* auxiliar para la generación del primer retroaviso,asociado con
* el secuenciador de 2x3 número dos que controla el llenado y vaciado
* del tanque auxiliar B.

  TEMPOC#5 E05,I41,I11,00:00:05.00,000;Temporizador monodisparo
* auxiliar para la generación del segundo retroaviso,asociado con
* el secuenciador de 2x3 número dos que controla el llenado y vaciado
* del tanque auxiliar B.


*******************************************
*  Secuenciador asociado con el tanque B. *
    SEC2#2 I13,I41,I06,I12,S03,I01,3,0111;*
## B01,B10,B00;                          *
*******************************************

  TEMPOC#6 I33,I41,I14,00:00:00.50,000;Temporizador monodisparo
* auxiliar para la generación del primer retroaviso,asociado con
* el secuenciador de 2x3 número tres que controla el llenado,vaciado
* y proceso de mezclado en el tanque C.
```

```
*******************************************
*  Secuenciador asociado con el tanque C. *
    SEC2#3 I16,I41,I06,I17,I44,I02,3,0111;*
## B00,B01,B10;                           *
*******************************************


  TEMPOC#8 I02,I41,S04,00:30:00.00,111;Temporizador asociado con el
*                                      el motor del agitador del
*                                      tanque C.

  TEMPOC#7 s04,I41,I15,00:00:00.50,000;Temporizador monodisparo
* auxiliar para la generación del segundo retroaviso,asociado con
* el secuenciador de 2x3 número tres que controla el llenado,vaciado
* y proceso de mezclado en el tanque C.

* El siguiente temporizador restablece a los tres secuenciadores,
* al producirse un flanco de bajada,en la señal del sensor del
* nivel inferior del tanque C (SNIC=E06).

          TEMPOC#3 E06,I41,I06,00:00:00.50,000;

* A continuación las declaraciones asociadas con los módulos
* de tipo mensajero empleados.

       MENSAJERO#1 "",1,16,20,0001;
## Tanques de almacenamiento de liquidos A y B vacios.
       MENSAJERO#2 "",1,16,20,0001;
## Tanque de almacenamiento de liquido A vacio.
       MENSAJERO#3 "",1,16,20,0001;
## Tanque de almacenamiento de liquido B vacio.
       MENSAJERO#4 "",1,16,20,0001;
# Se ha completado el proceso de mezclado de la cuota diaria de lotes. |
## Para reiniciar oprimir botón de reset del PLM.
       MENSAJERO#5 "",1,16,20,0001;
## LLenado de tanques auxiliares A y B en curso,(bombas A y B activadas).
       MENSAJERO#6 "",1,16,20,0001;
```

```
## LLenado de tanque auxiliar A en curso,(bomba A activada).
       MENSAJERO#7 "",1,16,20,0001;
## LLenado de tanque auxiliar B en curso,(bomba B activada).
       MENSAJERO#8 "",1,16,20,0001;
## Descarga de tanques auxiliares A y B en curso,(bombas D y E
activadas).
       MENSAJERO#9 "",1,16,20,0001;
## Descarga de tanque auxiliar A en curso,(bomba D activada).
       MENSAJERO#10 "",1,16,20,0001;
## Descarga de tanque auxiliar B en curso,(bomba E activada).
       MENSAJERO#11 "",1,16,20,0001;
## Proceso de mezclado en tanque C activado.
       MENSAJERO#12 "",1,16,20,0001;
## Descarga de tanque mezclador en curso,(bomba C activada).



       MENSAJERO#0 "",1,16,20,0001;
## Subproceso receptor de lotes despachados,no listo.

*                    1234567890123456
       MENSAJERO#13 "    Lotes listos",1,11,1,1110;
## Este mensaje móvil nunca se desplegará

      FINMODI;
```

CONCLUSIONES

El prototipo logrado en este trabajo de tesis, cubre lo originalmente esperado del mismo como resultado de un proyecto académico, a futuro la tecnología desarrollada podría, en su caso, ser transferida a una empresa con capacidad para reproducir el PLM, una vez concluido el proceso de normalización de conformidad con el campo de aplicación en el que se desee comercializar el producto, desde luego que ésto implicaría tiempo y un esfuerzo adicional.

Los campos de aplicación potenciales son de un alto impacto económico, tales como: Industria de alimentos, industria química, industria eléctrica, etc.

El desarrollo presentado en esta tesis involucró aspectos tanto de hardware como de software; en lo que toca al hardware se empleó infraestructura creada previamente por el autor (tarjeta SIMMP-2), y experiencia anterior del mismo, en cuanto al microcontrolador 68HC11 y sistemas digitales.

Desde luego que el PLM pudo haberse desarrollado alrededor de otra plataforma en cuanto al microcontrolador empleado, se trabajó con el 68HC11 simplemente por su popularidad y lo mencionado en el párrafo anterior.

En cuanto a software puede decirse que el esfuerzo fue más intenso, ya que hubo que hacer lo siguiente:

a) Desarrollo del software de traducción, el cual fue escrito en un lenguaje de alto nivel, siendo la primera versión del programa ejecutable bajo MSDOS (programa SIIL1.EXE).
b) Desarrollo de la versión inicial del software para programar aplicaciones con el PLM, el cual corre bajo windows (sistema SPDFLM); para ésto se empleó un lenguaje orientado a objetos (VisualBasic versión 4 profesional).
c) Desarrollo, empleando lenguaje ensamblador del 68HC11, de los tramos de código esqueleto asociados con cada uno de los módulos que puede realizar el PLM.
d) Diseño de las formas sintácticas asociadas con las declaraciones correspondientes a cada uno de los módulos del PLM y lo que se derive de ésto, en cuanto al código esqueleto, que en cada caso corresponda.

e) Aplicación de conceptos relacionados con la forma y aspecto de un programa en lenguaje de máquina para el microcontrolador 68HC11; ésto fue importante en el diseño del armado del código objeto que lleva a cabo el software de traducción.

f) Aplicación de conceptos relacionados con el manejo del puerto serie de una computadora; ésto tanto del lado del microcontrolador, como en lo que toca a la computadora anfitriona (PC); ésto fue importante en el diseño de la infraestructura de software que permite monitorear la operación del PLM desde la computadora anfitriona empleada para desarrollo.

En suma, a groso modo, puede decirse que el esfuerzo para lograr el primer PLM fue un 20 % trabajo con hardware y un 80 % desarrollo de software; el resultado de ésto fue un sistema que cuenta con su propio lenguaje de programación (SIIL1) y que puede realizar bloques funcionales de uso común en el control lógico de procesos.

Desde luego que, tal y como sucede la primera vez que se intenta sacar una idea del papel, existen diversos aspectos del sistema que son susceptibles de ser mejorados y/o ampliados, ésto tanto del lado del hardware como del software.

En cuanto a las mejoras del hardware, una importante sería lograr un dispositivo más compacto, ésto podría hacerse rediseñando circuitos impresos y/o mediante el empleo de componentes de montaje superficial.

En lo que respecta a mejoras y/o adiciones al software, pueden mencionarse entre otras las siguientes:

a) Rediseño del código de monitoreo, ya que si bien el mismo opera adecuadamente para este primer prototipo experimental, para un equipo en campo es necesario que el enlace sea más confiable.

b) Diseño de módulos que realicen compuertas lógicas de más de cuatro entradas.

c) Diseño de módulos que manejen variables analógicas, que pudieran ser empleados en el desarrollo de sistemas de adquisición de datos.

d) Diseño de módulos que realicen lazos de control digital directo.

Las dos últimas mejoras mencionadas, involucrarían el desarrollo del hardware apropiado, y son, potencialmente, líneas de investigación y desarrollo de la División de Ingeniería Eléctrica de la Facultad de Ingeniería de la UNAM.

BIBLIOGRAFÍA

1.- J. Webb, Programmable Logic Controllers
Principles and Applications,
Merrill, 1992.

2.- Maloney T. J.,Electrónica Industrial Moderna,
Prentice Hall, 1996.

3.- J Webb and K. Greshock, Industrial Control Electronics,
Merrill, 1993.

4.- Peter Spasov, Microcontroller Technology. The 68HC11.
Prentice Hall, 1993.

5.- MC68HC11F1 Technical Data.
Motorola Inc.
1990.

6.- M68HC11 Reference Manual.
Motorola Inc.
1991.

7.- Salvá A, Guia de usuario del sistema PUMMA - SIMMP2,
Circulación libre en la FI de la UNAM,
1996.

APÉNDICE A

GUIA DE USUARIO DEL SISTEMA PUMMA-SIMMP-2

Por: Antonio Salvá Calleja

1996

PARTE UNO DE APÉNDICE A

GUÍA DE USO DEL MANEJADOR HEXADECIMAL PUMMA.

El manejador hexadecimal PUMMA es un programa, que al ejecutarse en una computadora de tipo PC (XT, AT, 386, 486,...), hace que dicha computadora y la computadora monotablilla SIMMP-2 conformen un sistema Anfitrión-Destino (Host-Target) donde la CMT SIMMP-2 toma el papel de sistema destino (target) y la PC toma el papel de Anfitrión (host).
El enlace entre ambas computadoras se hace vía puerto serie a 1200 bps y con el formato 8N1. Ver Figura 1.1.
La CMT SIMMP-2 está basada en el microcontrolador MC68HC11 (con un cristal de 8 Mhz) y puede operar en los modos: boot-strap, single-chip, expandido y test.

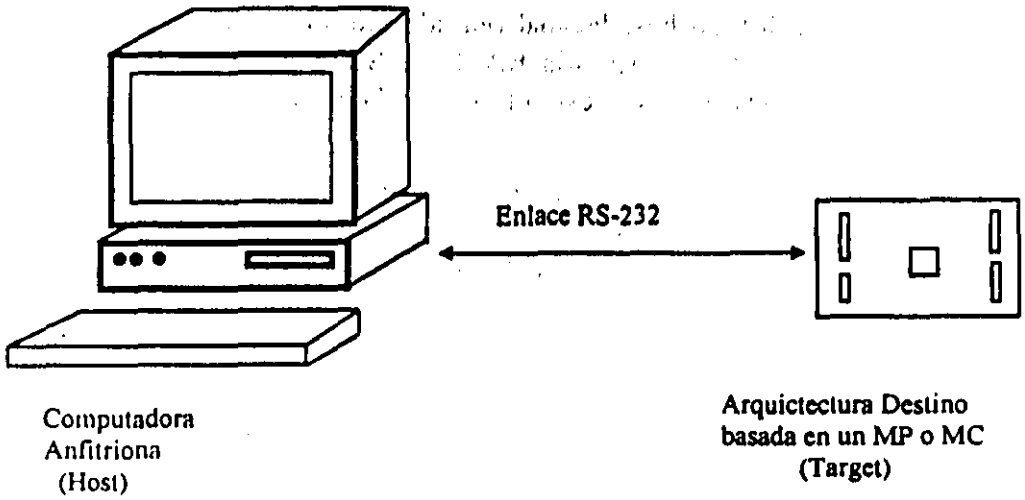


Figura 1.1 .- Esquema básico de un sistema Anfitrión-Destino para desarrollo con microcontrolador o microprocesador.

Nota 1: Para operar la CMT SIMMP-2 en modo test requerirá repaginar la EEPROM interna del 68HC11F1 que la valida.

La CMT SIMMP-2 cuenta entre otras con las siguientes facilidades:

1) Capacidad para configurar diversos mapas de memoria al operar en modo expandido.

2) Capacidad para programar las siguientes memorias EPROM comerciales: 27C16, 27C32, 27C64, 27C128, 27C256, 27C512, 2732, 2764, 27128, 27256 y 27512.

3) Capacidad de validación en el modo expandido de 3 o 6 puertos paralelos adicionales de entrada o salida mediante la inserción de uno o dos chips muy populares en la industria.

4) Capacidad para interfazar con una PC que ejecute el programa PUMMAX.EXE, cuando se opere en cualquiera de los cuatro modos posibles mencionados anteriormente.

La CMT SIMMP-2 es descrita con mayor detalle en otro capítulo de este instructivo.

Si bien PUMMA fue escrito originalmente para manejar la CMT SIMMP-2 desde una PC, puede emplearse con restricciones, para manejar arquitecturas basadas en otras versiones del 68HC11 (v.g. A1, E1, E2, E9, etc.), que operen en modo boot-strap.

En los párrafos siguientes se denominará como PC a la computadora anfitriona donde se ejecuta PUMMA, y se denotará como SD (sistema destino) a la arquitectura 68HC11 ligada a la PC via puerto serie.

A la fecha las facilidades con las que cuenta PUMMA son:

1) Carga en PC de un listado hexadecimal que pudiera representar ya sea código a ejecutarse en el 68HC11 del SD o datos a emplearse en una determinada aplicación.

2) Bajado desde PC a SD de un listado hexadecimal (en adelante denominado como LH), que represente código para ser cargado en memoria RAM del SD a partir de la dirección que previamente haya sido especificada. Una vez cargado en el SD el programa, éste es ejecutado por el SD.

3) Almacenamiento en disco de un LH que represente código o datos para su depuración o empleo posterior. PUMMA guarda en disco a los LH bajo un archivo con un nombre dado por el usuario con la extensión LEM o BLM; cabe señalar aquí que el tamaño de un archivo LEM es aproximadamente cuatro veces el número de bytes del LH correspondiente mientras que ese mismo LH al ser almacenado como un archivo BLM ocupará menos espacio en disco, ya que el tamaño de un archivo BLM es un poco mayor que el número de bytes contenidos en el LH correspondiente.

4) Lectura de disco de un archivo LEM o BLM.

5) Edición hexadecimal de un LH cargado en ese momento por el usuario o recién leído de disco.

6) Capacidad para importar un archivo S19 que haya sido generado por un ensamblador o compilador cruzado para 68HC11. El archivo queda validado como LH para ser de inmediato bajado y ejecutado en el SD o almacenado en disco para su uso posterior.

7) Capacidad para leer información de la memoria del SD.

8) Capacidad para escribir información a la memoria RAM del SD.

9) Capacidad para programar la memoria EEPROM del 68HC11 que valide al SD.

10) Capacidad para programar memorias EPROM comerciales. Esto último requiere que el SD sea la CMT SIMMP-2.

11) Capacidad para cargar bloques de código o datos validados previamente como archivos BLM o LEM), en la memoria RAM del SD.

INICIALIZACIÓN DEL SISTEMA A-D VALIDADO POR PUMMA EN PC

Para operar el sistema A-D se requiere el siguiente hardware:

1) Computadora anfitriona de tipo PC con sistema operativo MS-DOS versión 3.0 en adelante con al menos un puerto serie.

2) Sistema destino validado con un microcontrolador 68HC11. Si el SD es la CMT SIMMP-2, PUMMA permite la operación en forma natural en cualquiera de los modos del 68HC11 (boot-strap, sigle-chip, expandido o test), en caso de que el SD no sea la CMT SIMMP-2 PUMMA operará en forma natural solamente cuando el microcontrolador del SD esté en modo boot-strap.

3) Cable serial con tres hilos.

Nota 2: PUMMA supone que el puerto serie del SD deberá estar validado por el que contiene el 68HC11 del SD (SCI).

La CMT SIMMP-2 se enlaza con la computadora anfitriona (PC) mediante un cable serial validado para el lado que va a la PC con un conector de tipo DB9 hembra y para el lado del SD con un conector MOLEX hembra de tres hilos compatible con el conector serial existente en SIMMP-2.

A la fecha el software que valida a PUMMA del lado de la PC consiste de un programa ejecutable y de archivos LEM y BLM acompañantes que PUMMA emplea para validar las facilidades mencionadas anteriormente, los archivos correspondientes son los siguientes:

1) PUMMAX.EXE (X representa uno o dos caracteres que varían con la versión de PUMMA).

2) CHAPAN.ASC

3) AUXCB.BLM

4) PUMITANN.LEM

5) PUMAS19.LEM

6) PEEPA1.LEM

7) PROGEP5.LEM

8) TSP2C.LEM

9) BORAM.LEM

10)VEREP.LEM

11)RYCS19.LEM

12)RYCES19A.LEM

13)FFFF.BLM

14)PUMMA.AYD

15)RESPALDO.AUX

El archivo ejecutable y los acompañantes LEM, BLM, PUMMA.AYD y CHAPAN.ASC, deberán estar en el mismo subdirectorio o unidad de disco desde donde se ejecute PUMMA. En el caso de que la PC no cuente con reloj calendario permanente, al inicializarla se deberá introducir la fecha y hora correspondientes.

Para inicializar el sistema A-D se debe hacer lo siguiente:

1. Energizar el SD y oprimir su botón de RESET; en el caso de que el SD no sea la CMT SIMMP-2 el SD necesariamente deberá operar en modo boot-strap; si el SD es la CMT SIMMP-2 se podrá operar en cualquiera de los cuatro modos asociados con el 68HC11.
2. Con la PC y el SD enlazados con el cable serial, ver Figura 2.2, ejecutar PUMMA en la PC con el sujetador de mayúsculas validado, (no se debe hacer la ejecución bajo un esquema de multitarea, preferentemente correr el programa desde MS DOS).

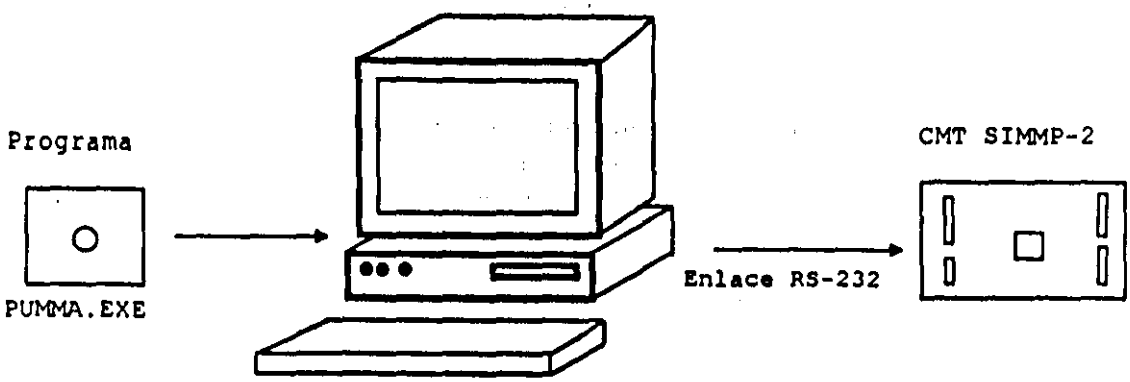


Figura 1.2 .- Sistema A-D para desarrollo alrededor del microcontrolador 68HC11 validado con la CMT SIMMP-2.

3. Al ejecutar PUMMA en la PC aparecerá una pantalla en la que se explica el significado de las siglas PUMMA así como un crédito al autor del programa y la versión y fecha en que fue realizado; en esta misma pantalla se requerirá al usuario que oprima cualquier tecla, seguido a lo cual aparecerá un crédito, por aproximadamente cuatro segundos, al diseñador de la CMT SIMMP-2.
4. Después de lo anterior se requerirá al usuario que especifique, en caso de que la PC tenga dos puertos serie, cual puerto usará (COM1 o COM2), el cual quedará inicializado a 1200 bps y formato 8N1, en caso de que la PC tenga sólo un puerto serie, PUMMA inicializará el mismo y no requerirá del usuario ninguna especificación.
5. En este punto aparecerá un menú en donde se pide al usuario que indique el tipo de microcontrolador que valida al SD. Cabe señalar aquí que PUMMA requiere del usuario, para la especificación de alguna opción , la opresión de una tecla, que puede ser un número o letra, en cuyo caso se indicará ésta denotándola entre paréntesis triangulares.
6. En seguida de lo anterior aparecerá en pantalla el menú de inicialización de la CMT SIMMP-2, que cuenta con las siguientes opciones:

    1) BAJAR AMBIENTE PUMMA A SIMMP-2
    2) PROSEGUIR SIN BAJAR AMBIENTE PUMMA
    3) EJECUTAR SIN ARQUITECTURA 68HC11 DESTINATARIA

En caso de que la CMT SIMMP-2 o SD no tenga validado el ambiente PUMMA el usuario deberá optar por la opción uno de este menú.

Nota 3: La validación del ambiente PUMMA en la CMT SIMMP-2 es apreciada por un continuo encendido y apagado de su LED testigo (LT); en caso de que SIMMP-2 esté operando algún modo que no sea boot-strap, la ausencia de ambiente PUMMA es testificada por un proceso de encendido y apagado del LT con una cadencia lenta, comparada con la correspondiente a la testificación dela presencia del ambiente PUMMA. Si el SD no es la CMT SIMMP-2 el usuario podrá validar un LT conectándolo a el pin PD5 del 68HC11 que valide al SD como se indica en la Figura 1.3, PUMMA puede operar desde luego sin el LT en el SD pero la experiencia ha hecho ver que el LT es muy útil al trabajar con el sistema A-D (PUMMA-SD).

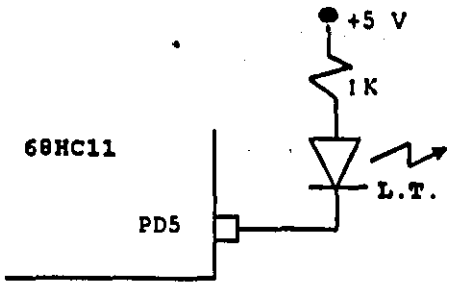


Figura 1.3 .- Conexión de Led Testigo (LT), en la arquitectura destinataria del sistema A-D validado por el programa PUMMAX.EXE.

Al escoger el usuario la opción uno del menú de inicialización de SIMMP-2 aparecerán en la pantalla letreros que testifican la lectura de disco del ambiente PUMMA y su transmisión (bajado al SD), inmediatamente a lo cual se deberá observar en el LT del SD una cadencia de encendido y apagado, indicando esto que el SD tiene validado el ambiente PUMMA, pudiendo establecerse con esto comunicación entre el anfitrión y el sistema destino, posteriormente a lo anterior deberá aparecer en la pantalla un letrero que pide una pequeña espera al usuario después de lo cual aparecerá un letrero indicando el modo en que opera el 68HC11 del SD, requiriendo al usuario la opresión de cualquier tecla, hecho esto aparecerá el menú principal de interfazado PC-SIMMP-2, estando a este punto todo listo para operar el sistema A-D, para aprender, desarrollar o divertirse un rato.

En caso de que el usuario se percate de la presencia en el SD del ambiente PUMMA (observando el encendido-apagado del LT), deberá optar por la opción dos del menú de inicialización de SIMMP-2, siguiendo a esto la misma secuencia de eventos que se suceden cuando opta por la opción uno después que el ambiente PUMMA ha quedado instalado en el SD. Si el usuario baja al SD el ambiente PUMMA estando este presente, se perderá el ambiente PUMMA en el SD debiendo ser necesario reinstalarlo. La reinstalación del ambiente PUMMA en el SD cuando este es la CMT SIMMP-2 y opera en cualquier modo que no sea boot-strap se hace simplemente oprimiendo el botón de RESET, en caso de que el SD opere en modo boot-strap la reinstalación del ambiente PUMMA se hace oprimiendo el botón de RESET del SD para después bajar el ambiente PUMMA, lo anterior se explicará mas adelante cuando se hable de las opciones del menú principal de interfazado.

Si el usuario opta por la opción tres del menú de inicialización de SIMMP-2, se ejecutara PUMMA con las opciones que requieren comunicación con el SD desvalidadas. En este caso

PUMMA indicará al terminar la inicialización que el SD opera en modo boot-strap aunque esto desde luego, no tendrá ningún significado.

Nota 4: En caso de que el SD tenga algún problema no aparecerá en pantalla el requerimiento de espera al usuario apareciendo en la parte inferior derecha de la pantalla un letrero que sugiere al usuario la opresión de la tecla ESC para abortar el lazo de recepción en PC, hecho lo cual aparecerán en pantalla indicaciones al usuario; en caso de no oprimir ESC, después de un tiempo que variara con la velocidad de la PC, aparecerá en pantalla un letrero que dice que el sistema SIMMP-2 está desconectado o con falla, al oprimir el usuario cualquier tecla deberá aparecer el menú principal de interfazado. Es importante señalar aquí que al ejecutarse algunas de las opciones de PUMMA el letrero que sugiere oprimir la tecla ESC para abortar lazo de recepción en PC aparecerá en la parte inferior derecha de la pantalla no debiendo el usuario oprimir ESC, ya que al hacer esto puede dejar inconcluso un proceso propio del sistema A-D (v.g. lectura de memoria a bloques, programación de memoria EPROM, etc), una forma sencilla de saber cuando se debe oprimir ESC es que cuando se ha perdido comunicación con el SD el letrero mencionado en este párrafo aparecerá con una nitidez constante, mientras que cuando aparece al ejecutar PUMMA normalmente una opción se mostrará con un ligero centelleo. Si el SD está validado con una versión del 68HC11 que sólo tenga 256 bytes de memoria RAM interna, al inicializar el sistema A-D PUMMA presentará la misma secuencia de eventos que se presentan cuando no hay comunicación correcta con el SD, sin embargo el usuario puede solventar esto abortando el lazo receptor en PC cuando PUMMA se lo indique, para después desde el menú principal restaurar el ambiente PUMMA en el SD.

A continuación se describen los distintos menúes de opciones con los que cuenta PUMMA para operar el sistema A-D.

MENÚ PRINCIPAL DE INTERFAZADO PC-SIMMP-2

Este menú cuenta con las siguientes opciones:

1. INTRODUCCIÓN DE UN PROGRAMA EN LENGUAJE DE MÁQUINA DEL 68HC11
2. PASAR A MENÚ DE MANEJO DE DISCO
3. BAJAR A SIMMP-2 AMBIENTE RECEPTOR DE PROGRAMAS NP.S19
4. BAJAR A SIMMP-2 PROGRAMA EN MODO BOOT-STRAP
5. PASAR A MENÚ DE MANEJO DE MEMORIA
6. BAJAR A SIMMP-2 PROGRAMA CON ASIGNACION DE DIRECCIONES
7. PASAR A MENÚ DE EDICIÓN
8. BAJAR AMBIENTE PUMMA A SIMMP-2
9. TERMINAR LA SESIÓN

A continuación se describen cada una de las opciones del menú principal.

1.-INTRODUCCIÓN DE UN PROGRAMA EN LENGUAJE DE MÁQUINA DEL 68HC11. Esta opción permite validar en la PC un programa en lenguaje de máquina introduciéndolo como lista hexadecimal (LH), bajo un nombre y direcciones inicial y final especificados por el usuario, desde luego que la LH introducida podría representar también un bloque de datos que podrían ser empleados en una determinada aplicación (v.g. tablas de

calibración, listas alfanuméricas, etc), PUMMA pide al usuario la información requerida por esta opción en el siguiente orden:

A) **NOMBRE DEL PROGRAMA O LH A INTRODUCIR.** Aquí el usuario deberá teclear un nombre de no más de ocho caracteres, seguido de la opresión de la tecla return (<CR>), si el usuario en este punto introduce como nombre del programa a una cadena de un caracter que fuera la letra Q el proceso de introducción del LH es abortado retornando PUMMA de inmediato al menú principal de interfazado.

Nota 5: En esta guía, de aquí en adelante se denotara como NP, al nombre del LH que se esté manejando en un momento dado.

B) **DIRECCIÓN INICIAL (HEX) DEL PROGRAMA O LH.** Aquí el usuario debe teclear la dirección inicial del programa o LH a introducir validada en cuatro dígitos hexadecimales, en caso de que ya sea que la validación no sea dada en cuatro dígitos, o que alguno de ellos no sea un dígito hexadecimal válido, PUMMA hace de nuevo el requerimiento de la dirección inicial, si se deseara abortar el proceso y retornar al menú principal de interfazado, bastará con introducir como dirección inicial a la letra Q seguida de la opresión de la tecla <CR>.

C) **DIRECCIÓN FINAL (HEX) DEL PROGRAMA O LH.** Aquí el usuario debe teclear la dirección final del programa o LH a introducir validada en cuatro dígitos hexadecimales, en caso de que ya sea que la validación no sea dada en cuatro dígitos, o que alguno de ellos no sea un dígito hexadecimal válido, PUMMA hace de nuevo el requerimiento de la dirección final, si se deseara abortar el proceso y retornar al menú principal de interfazado, bastará con introducir como dirección final a la letra Q seguida de la opresión de la tecla <CR>.

D) **INTRODUCCIÓN SECUENCIAL A PC DE LOS BYTES QUE CONFORMAN EL LH.** Aquí el usuario deberá introducir en la PC la secuencia de bytes que conforman el LH, tecleando el par de caracteres que definen cada byte seguidos de la opresión de la tecla <CR>; en caso de que el usuario desee terminar el proceso antes de introducir el byte asociado con la dirección final especificada, bastará con teclear la letra Q seguida por <CR>; si el usuario deseara corregir el último byte introducido deberá teclear R y después <CR> cuando PUMMA pida el byte correspondiente a la siguiente dirección, una vez hecho esto PUMMA volverá a pedir el byte asociado con la dirección que se desea corregir, esta acción podrá repetirse sucesivamente hasta que este proceso de repetición de introducción de bytes llegue a la dirección inicial especificada por el usuario para el LH.

Una vez validado en la PC el programa o LH, el usuario podrá, mediante el empleo de otras opciones de PUMMA efectuar cualquiera de las siguientes acciones:

a) Guardar en disco el LH como un archivo NP.LEM
b) Guardar en disco el LH como un archivo NP.BLM
c) Suponiendo que el LH sea un programa, PUMMA puede bajarlo al SD, colocarlo en las direcciones especificadas, para que de inmediato se ejecute a partir de la dirección inicial asociada con el LH.

d) Suponiendo que el LH sea un programa, PUMMA puede bajarlo al SD, colocarlo a partir de la dirección $0000 de la memoria RAM interna del microcontrolador del SD, para que de inmediato se inicie en el SD, a partir del origen de la RAM interna, la ejecución del programa representado por el LH. Si el SD es la CMT SIMMP-2 esta opción podrá emplearse en cualquiera de los cuatro modos de operación del 68HC11, o en otro caso sólo en el modo boot-strap.

Nota 6: Si la acción (d) se usa sobre la CMT SIMMP-2 operando en algún modo que no sea boot-strap, el ambiente PUMMA deberá estar desvalidado en el SD, y esto sólo se logra desenergizando la CMT SIMMP-2 para después energizarla y de ser necesario oprimir el botón de RESET. Por lo regular esta opción es usada sólo cuando el SD opera en modo boot-strap.

2.-PASAR AL MENÚ DE MANEJO DE DISCO. Esta opción del menú principal conduce al menú de manejo de disco que se explicará mas adelante.

3.-BAJAR A SIMMP-2 AMBIENTE RECEPTOR DE PROGRAMAS NP.S19. Mediante esta opción el usuario podrá validar en el SD un ambiente que recibe byte por byte un archivo S19 que represente a un programa para que éste se ejecute de inmediato a partir de la dirección inicial especificada por el comando ORG en el programa fuente original, cabe señalar aquí que este ambiente receptor de código objeto S19 no checa errores (CRC) y que una vez validado en el SD desde luego que ya no se podrá seguir empleando PUMMA para comunicarse con el SD.

4.-BAJAR A SIMMP-2 PROGRAMA EN MODO BOOT-STRAP. Empleando esta opción el usuario puede bajar al SD un LH que represente un programa usando la secuencia que requiere el programa BOOT-LOADER que se activa después de dar RESET, en los microcontroladores 68HC11 cuando los mismos operan en modo boot-strap, desde luego que el programa que representa el LH es cargado a partir del origen de la RAM interna del microcontrolador, para ejecutarse de inmediato, más información acerca del modo boot-strap puede encontrarse en la referencia 1.

5.-PASAR AL MENÚ DE MANEJO DE MEMORIA. Esta opción hace que PUMMA pase al menú de manejo de memoria, que se explicará mas adelante en cuanto a sus opciones Es importante destacar aquí que al entrar al menú de manejo de memoria se pierde el LH presente en la PC, por lo que si el usuario deseara que esto no sucediera deberá guardar en disco el LH presente antes de pasar al menú de manejo de memoria.

6.-BAJAR A SIMMP-2 PROGRAMA CON ASIGNACIÓN DE DIRECCIONES. Esta opción hace que PUMMA baje al SD un programa representado por el LH presente, colocándolo en las direcciones asociadas correspondientes, para de inmediato ser ejecutado, desde luego que el ambiente PUMMA debe estar presente en el SD para la correcta operación de esta opción.

7.-PASAR A MENÚ DE EDICIÓN. Esta opción hace que PUMMA pase al menú de edición hexadecimal de un LH, cuyas opciones se explicarán mas adelante.

8.-BAJAR AMBIENTE PUMMA A SIMMP-2. Esta opción hace que se reinstale el ambiente PUMMA en el SD suponiendo que el mismo haya sido perdido por alguna razón. Para más detalles ver la nota 3 de esta guía.

9.-TERMINAR LA SESIÓN. Esta opción permite al usuario salir de PUMMA para retornar al sistema operativo de la PC.

A continuación se describen los menúes de manejo de disco, manejo de memoria y edición hexadecimal.

MENÚ DE MANEJO DE DISCO

Las opciones de este menú son:

1.- LEER UN ARCHIVO NP.LEM
2.- LEER UN ARCHIVO NP.S19 CON GENERACIÓN DE ARCHIVO NP.BLM
3.- LEER UN ARCHIVO NP.BLM
4.- CARGAR EN RAM UN ARCHIVO NP.S19 EN DIRECCIONES REALES
5.- LEER DIRECTORIO
6.- GUARDAR EN DISCO EL LH PRESENTE

A continuación se describen cada una de las opciones de este menú.

1.-LEER UN ARCHIVO NP.LEM. Al invocar esta opción el usuario puede leer de disco un LH, que podrá representar código o datos, que previamente haya sido almacenado en disco bajo algún nombre con la extensión LEM. Cuando PUMMA pide el nombre del LH validado bajo el archivo LEM, el usuario deberá teclearlo sin extensión, seguido de la opresión de la tecla <CR>, en caso de que el archivo LEM correspondiente no exista en la unidad de disco o subdirectorio donde esté posicionada la PC, PUMMA indicará esto pudiendo en este momento repetirse el proceso de lectura o abortarse. Si el usuario introduce como nombre del programa a la letra Q el proceso de lectura es abortado. En caso de que el archivo LEM requerido se encuentre en otro subdirectorio o unidad de disco, el usuario deberá teclear antes del nombre la trayectoria correspondiente (PATH), por ejemplo, si se desea leer de disco el archivo CONT.LEM que se encuentra en la unidad de disco B en el subdirectorio SP2, cuando PUMMA pide el nombre del programa el usuario deberá teclear: B:\SP2\CONT

2.-LEER UN ARCHIVO NP.S19 CON GENERACIÓN DE ARCHIVO NP.BLM Mediante esta opción el usuario podrá generar un archivo BLM a partir de un archivo S19, al invocarla, PUMMA pedirá el nombre sin extensión del archivo S19 correspondiente indicando la trayectoria si esto es necesario, si el usuario indica como nombre del archivo la letra Q el proceso es abortado retornando PUMMA al menú de manejo de disco.
En seguida PUMMA pedirá el nombre del archivo BLM a generar pudiendo el usuario indicar la trayectoria correspondiente, en caso de que desee generar el archivo en un subdirectorio o unidad de disco diferente al que contenga el programa PUMMAX.EXE. Si el usuario indica como nombre la letra Q, el proceso es abortado retornando PUMMA al menú de manejo de disco.
En caso de que el usuario desee que el archivo BLM a generar tenga el mismo nombre que el correspondiente al S19 leído, bastará con que oprima la tecla <return>, al pedirle PUMMA el

nombre deseado para el archivo BLM. Para más explicación sobre como denotar trayectorias se sugiere ver la ayuda asociada con las opciones 1, 3 y 6 del menú de manejo de disco.
Después de esto PUMMA indicará sucesivamente en pantalla las direcciones asociadas con cada uno de los bloques que conforman el archivo S19, indicando posteriormente la dirección consignada en el record S9, mediante la opresión de una tecla PUMMA retornará al menú de manejo de disco.
En las direcciones que no existan en el archivo S19 original, PUMMA colocará bytes FF en las mismas direcciones asociadas con el archivo BLM correspondiente.

3.-LEER UN ARCHIVO NP.BLM. Al invocar esta opción el usuario puede leer de disco un LH, que podrá representar código o datos, que previamente haya sido almacenado en disco bajo algún nombre con la extensión BLM. Cuando PUMMA pide el nombre del LH validado bajo el archivo BLM, el usuario deberá teclearlo sin extensión, seguido de la opresión de la tecla <CR>, en caso de que el archivo BLM correspondiente no exista en la unidad de disco o subdirectorio donde esté posicionada la PC, PUMMA indicará esto pudiendo en este momento repetirse el proceso de lectura o abortarse. Si el usuario introduce como nombre del programa a la letra Q el proceso de lectura es abortado.
En caso de que el archivo BLM requerido se encuentre en otro subdirectorio o unidad de disco, el usuario deberá teclear antes del nombre la trayectoria correspondiente (PATH), por ejemplo, si se desea leer de disco el archivo NUMET.BLM que se encuentra en la unidad de disco B en el subdirectorio SPBLM, cuando PUMMA pide el nombre del programa el usuario deberá teclear: B:\SPBLM\NUMET

4.-CARGAR EN RAM UN ARCHIVO NP.S19 EN DIRECCIONES REALES. Esta opción permite al usuario cargar en RAM un archivo NP.S19 en las mismas direcciones indicadas por los records S del archivo; al invocarla, PUMMA pide al usuario indicar mediante la opresión de una tecla si la carga ha de ser con o sin chequeo de errores. Si el usuario desea chequeo de errores deberá oprimir la tecla <E> en otro caso deberá oprimir cualquier tecla que no sea <E>.
Enseguida a lo anterior PUMMA pedirá el nombre sin extensión del archivo S19 a cargar, el usuario podrá indicar la correspondiente trayectoria en caso de ser esto necesario. Si el usuario teclea como nombre del archivo la letra Q el proceso de carga es abortado retornando PUMMA al menú de manejo de disco.

Después de esto PUMMA procede a cargar en RAM el archivo S19 invocado, desplegando en pantalla las direcciones correspondientes a los distintos bloques que lo conforman, una vez que se ha completado la carga, PUMMA despliega en pantalla la dirección correspondiente al record S9, se supone que las direcciones del archivo deben corresponder con direcciones de RAM reales en el sistema destino.
Si la carga es con chequeo de errores y PUMMA encuentra al cargar un record S discrepancia en el byte verificador correspondiente reintentará hasta 20 veces cargarlo nuevamente, si prevalece el error PUMMA desplegará en pantalla el record que no pudo ser cargado exitósamente en el sistema destino y requerirá que el usuario escoja, oprimiendo una tecla, una de tres acciones a seguir:

<A> Abortar el proceso de carga del archivo NP.S19.
<C> Continuar con la carga del los siguientes records del archivo.

<R> Reintentar otras 20 veces la carga del record que no pudo ser cargado.
Cabe señalar que la condición de error al cargar los records pudiera generarse entre otras causas por las dos siguientes:

a) Intento de carga de un archivo S19 que contenga cuando menos una dirección para la cual no exista RAM física en el sistema destino.
b) Falla en el sistema de enlace serial, esto podría suceder cuando el mismo fuera via modem y muy ocasionalmente cuando el enlace fuera sin modem (directo).Si la carga es sin chequeo de errores PUMMA bajará cada record sin tomar en cuanta el byte verificador de cada record, siendo responsabilidad del usuario el que exista RAM física en el sistema destino para todas las direcciones asociadas con el archivo S19.

5.-LEER DIRECTORIO. Mediante esta opción, el usuario puede hacer que PUMMA despliegue en pantalla el nombre de los archivos existentes en el disco contenido en una determinada unidad. PUMMA pide primero los nombres de los archivos a listar, para después solicitar al usuario la unidad de disco correspondiente, en caso de que el usuario deseara desplegar los archivos contenidos en el subdirectorio donde esté posicionada la PC, bastará con teclear la letra que designa a la unidad de disco correspondiente, en caso que los archivos cuyo nombre se desee desplegar estén en otro subdirectorio, el usuario deberá teclear la trayectoria correspondiente (PATH) seguida de la letra que designe a la unidad de disco correspondiente, por ejemplo, si el usuario desea desplegar el nombre de todos los archivos BLM contenidos en el subdirectorio SIMMP del disco contenido en la unidad A, deberá teclear BLM cuando PUMMA pide el nombre de los archivos a listar y cuando PUMMA pide la unidad de disco deberá teclear A:\SIMMP\, inmediatamente a lo cual deberán aparecer en pantalla los nombres de todos los archivos BLM contenidos en el subdirectorio SIMMP del disco contenido en la unidad A, en caso de que no exista ningún archivo de los que se solicitó desplegar su nombre, PUMMA indicará esto en pantalla.

Nota 7: Si la trayectoria indicada contiene uno o varios nombres de subdirectorio inexistentes, PUMMA desplegará en pantalla que no existen archivos cuyo nombre sea el especificado por el usuario, esto no obstante que los archivos en efecto existan en algún subdirectorio válido.

6.-GUARDAR EN DISCO EL LH PRESENTE. Al invocar el usuario esta opción PUMMA indica el nombre bajo el cual el LH sería almacenado en disco, aquí PUMMA pide al usuario la opresión de una tecla para indicar una de las tres siguientes alternativas:

a)Almacenamiento bajo el nombre indicado por PUMMA oprimiendo la tecla M.

b)Almacenamiento bajo otro nombre oprimiendo cualquier tecla que no sea la M.

c)Abortar el proceso de almacenamiento oprimiendo la tecla Q.

En caso de que el usuario haya indicado la opción (b), PUMMA le pedirá el nombre bajo el cual el usuario desea almacenar el LH presente; enseguida de que el usuario teclea el nuevo nombre deseado para el LH o en su caso indica que se desea almacenar el LH bajo el mismo nombre, PUMMA pedirá la designación de la unidad de disco, aquí el usuario deberá teclear la letra correspondiente seguida por la trayectoria correspondiente, en caso de que el almacenamiento

vaya a ser hecho en la unidad de disco donde esté posicionada la PC bastara aquí con teclear la letra que la designe, enseguida a lo anterior PUMMA pedirá al usuario la opresión de una tecla que indique una de las siguientes tres alternativas:

a)Almacenar el LH como archivo BLM mediante la opresión de la tecla B.

b)Almacenar el LH como archivo LEM mediante la opresión cualquier tecla que no sea B.

c)Abortar el proceso de almacenamiento mediante la opresión de la tecla Q.

Nota 8: En caso de que PUMMA indique que el programa se va almacenar bajo un nombre que incluya una trayectoria, el usuario deberá siempre cambiar el nombre ya que de no hacerlo se generará un error en la ejecución de PUMMA, esta condición se produciria cuando el LH presente haya sido leído de disco especificando una determinada trayectoria

MENÚ DE MANEJO DE MEMORIA

Las opciones de este menú son las siguientes:

1.-LEER MEMORIA EN SIMMP-2
2.-CARGAR DATOS EN MEMORIA DE SIMMP-2
3.-PASAR EJECUTAR UN PROGRAMA EN SIMMP-2
4.-PASAR A PROGRAMAR LA EEPROM DEL mC del SIMMP-2 O SD
5.-CARGAR BLOQUES EN SIMMP-2
6.-PASAR A PROGRAMAR LA EPROM EXTERNA DEL SIMMP-2

A continuación se detalla el funcionamiento de cada una de las opciones del menú de manejo de memoria.

1.-LEER MEMORIA EN SIMMP-2. Al invocarse esta opción aparece un menú donde el usuario escoge entre leer una sóla o varias localidades de memoria, si el usuario opta por leer una sóla localidad PUMMA le pedirá la dirección a leer en hexadecimal, después de que el usuario ha introducido la dirección a leer, PUMMA despliega en pantalla la dirección y su contenido en hexadecimal, en caso de que al introducir la dirección el usuario teclee la letra Q seguida de la opresión de la tecla <CR>, el proceso de lectura de una localidad de memoria es abortado.

Si el usuario opta por leer varias localidades de memoria PUMMA pedirá secuencialmente la introducción de las direcciones inicial y final a leer, (la dirección final deberá ser mayor o igual que la especificada como inicial), si al introducir la dirección inicial o final el usuario teclea Q seguido de la opresión de la tecla <CR>, el proceso de lectura de varias localidades de memoria es abortado. Una vez que se han especificado las direcciones inicial y final a leer, PUMMA pide al usuario escoger, mediante la opresión de una tecla, cualquiera de las siguientes tres alternativas:

a) Lectura a pasos, mediante la opresión de cualquier tecla que no sea <C> o <B>.
b) Lectura continua, mediante la opresión de la tecla <C>
c) Lectura de bloques, mediante la opresión de la tecla <B>.

Al elegir el usuario la alternativa (a), PUMMA desplegará secuencialmente el contenido de las direcciones especificadas en forma de lista, requiriéndose que el usuario oprima cualquier tecla,

que no sea <Q>, para que PUMMA despliegue sucesivamente los contenidos respectivos, para abortar el proceso de lectura de memoria bastará con oprimir la tecla <Q>.

Si el usuario elige la alternativa (b), PUMMA desplegará secuencialmente en forma de lista, sin parar, el contenido de las direcciones especificadas, deteniéndose el proceso una vez que se ha desplegado el contenido de la dirección final especificada.
Si el usuario eligió la alternativa (c), PUMMA desplegará después de unos segundos, el contenido de 256 localidades de memoria comenzando por la dirección inmediata anterior a la especificada como inicial que sea múltiplo exacto de 256, la dirección final que haya especificado el usuario no se toma en cuenta, así por ejemplo, si el usuario especifica 012A como dirección inicial y 0257 como dirección final, PUMMA desplegará en pantalla en forma de bloque el contenido de las direcciones que van de 0100 a 01FF. Una vez que un bloque ha sido desplegado, PUMMA pide al usuario la opresión de cualquier tecla que no sea <Q> para proceder a desplegar el subsecuente bloque de contenidos de memoria, si el usuario oprime la tecla <Q> PUMMA aborta el proceso de lectura a bloques retornando de inmediato al menú de manejo de memoria.

2.-CARGAR DATOS EN MEMORIA DE SIMMP-2. Esta opción permite introducir datos validados como bytes en localidades de memoria RAM de la CMT SIMMP-2 o SD, PUMMA le pide al usuario las direcciones inicial y final sobre las que se van a cargar datos, (si el usuario teclea Q seguido de <CR> al validar la dirección inicial o final el proceso de carga de datos a RAM es abortado), una vez hecho lo anterior PUMMA pide secuencialmente al usuario los respectivos bytes a cargar en formato hexadecimal, cuando el usuario termina de introducir los bytes a cargar, PUMMA los coloca en el SD. Si en el proceso de introducción de bytes el usuario desea repetir la introducción del dato a cargar en una localidad bastará con teclear R seguido de <CR> al introducir el contenido de la dirección subsecuente a la que se desea corregir, si el usuario teclea Q al validar el byte a cargar en una localidad, PUMMA carga en RAM del SD los valores especificados por el usuario hasta antes de oprimir la tecla <Q>.

3.-PASAR EJECUTAR UN PROGRAMA EN SIMMP-2. Al elegirse esta opción PUMMA pide al usuario la dirección inicial de un programa previamente cargado en memoria del SD, (si el usuario teclea Q seguido de <CR> el proceso es abortado), una vez hecho esto se pide al usuario la opresión de cualquier tecla seguido a lo cual se inicia la ejecución del programa en el SD, desde luego que el usuario es responsable de que en la dirección especificada realmente inicie un código coherente.

4.-PASAR A PROGRAMAR LA EEPROM DEL mC del SIMMP-2 O SD. Para trabajar con esta opción el microcontrolador del SD deberá operar en modo bootstrap, al invocarla se pide al usuario seleccionar la acción a seguir entre las siguientes dos alternativas:

1) PROGRAMAR LA EEPROM INTERNA CON FUENTE DESDE TECLADO
2) PROGRAMAR LA EEPROM INTERNA CON FUENTE EN DISCO

Al seleccionar una de las dos alternativas anteriores, el usuario observará que se pierde el ambiente PUMMA en el SD, esto es normal; si el usuario aborta el proceso de programación el ambiente PUMMA es automáticamente reinstalado en la PC.

Si el usuario opta por la alternativa uno, PUMMA le pedirá especificar las direcciones inicial y final a programar, (si al introducir la dirección inicial o final el usuario teclea Q y después <CR> se aborta el proceso de programación), en seguida a lo anterior PUMMA pedirá los bytes a programar en cada una de las direcciones del intervalo de programación especificado, si el usuario desea corregir el byte asociado con una determinada dirección bastará con teclear R seguido de <CR> al introducir el byte asociado con la dirección subsecuente, si en esta instancia el usuario teclea Q seguido de <CR> el proceso de toma de bytes es abortado y PUMMA programa en la EEPROM interna los bytes introducidos hasta antes de abortar el proceso. Cabe señalar aquí que el máximo número de bytes que se pueden programar desde teclado es 169.

Después de que se ha efectuado la programación PUMMA reinstala automáticamente el ambiente PUMMA en el SD.

Nota 9: Una vez que se han introducido las direcciones inicial y final, el proceso de programación de la EEPROM interna desde teclado sólo podrá ser abortado después de la introducción del primer byte a programar, en caso de que el usuario teclee Q seguido de <CR> al introducir el primer byte a programar se generará una condición de error en la ejecución de PUMMA que requerirá la reinstalación del ambiente PUMMA en el SD.

En caso de que el usuario haya elegido la alternativa dos, PUMMA le pedirá el nombre sin extensión del archivo a programar, que deberá ser tecleado antecedido, de ser necesario, por la trayectoria correspondiente, (si el usuario teclea como nombre del archivo a la letra Q el proceso de programación con fuente en disco es abortado), enseguida a lo anterior PUMMA pedirá mediante la opresión de una tecla definir si el archivo es BLM o LEM, una vez que el archivo ha sido leído, PUMMA pide al usuario la dirección inicial a programar, (si aquí el usuario teclea Q seguido de <CR> el proceso de programación desde teclado es abortado), después de lo anterior PUMMA indicará la dirección final a programar, posteriormente PUMMA programa el LH validado por el archivo seleccionado particionándolo en bloques de 100 o menos bytes, indicando al usuario en pantalla cual bloque se estaría programando en un momento dado, una vez que se ha concluido el proceso de programación PUMMA reinstala automáticamente el ambiente PUMMA en el SD.

5.-CARGAR BLOQUES EN SIMMP-2. Mediante esta opción el usuario puede cargar en la RAM del SD bloques validados por archivos LEM o BLM, al pedirlo PUMMA el usuario deberá teclear el nombre sin extensión del archivo que contiene al LH a cargar como bloque, antecediendo de ser necesario, la trayectoria correspondiente, en seguida a lo anterior PUMMA pide el tipo de archivo (LEM o BLM), seguido a lo cual, si el archivo existe en disco, en pantalla se despliegan las direcciones inicial y final del LH pudiendo aquí el usuario optar por dos alternativas a saber.

a) Oprimiendo cualquier tecla que no sea <C>, el usuario indica a PUMMA que el LH validado por el archivo leído sea cargado en sus direcciones originales.

b) Oprimiendo la tecla <C>, el usuario indica que desea que el LH sea cargado en direcciones diferentes a las consignadas en el archivo BLM o LEM correspondiente

Si el usuario eligió cambiar las direcciones de carga, PUMMA le pedirá la dirección a partir de donde se desea cargar el LH, aquí el usuario tecleará la nueva dirección inicial, si después de esto el usuario deseara cambiar de nuevo la dirección inicial, bastará con oprimir la tecla <R>, en otro caso el usuario oprimirá cualquier otra tecla para que PUMMA lleve a cabo la carga en RAM del LH, una vez que se ha completado la carga, PUMMA pedirá el nombre del archivo que valida el siguiente bloque a cargar indicando en pantalla el nombre del bloque anterior cargado, así como las direcciones inicial y final donde quedó localizado. El número máximo de bloques a cargar es 1000, si el usuario teclea Q al indicar el nombre del archivo que válida el LH a cargar, el proceso de carga de bloques es abortado, cabe señalar aquí que antes de indicar a PUMMA que cargue un determinado bloque, el usuario deberá verificar que la testificación del ambiente PUMMA esté indicada por el LED testigo (LT) del SD.

6.-PASAR A PROGRAMAR LA EPROM EXTERNA DE LA CMT SIMMP-2. Para trabajar esta opción el microcontrolador del SD deberá operar en modo bootstrap, al ser invocada PUMMA presenta al usuario el suguiente menú:

MENÚ DE PROGRAMACIÓN DE LA EPROM EXTERNA

1.-VERIFICAR QUE LA EPROM ESTÉ COMPLETAMENTE BORRADA
2.-PASAR A PROGRAMAR LA EPROM EXTERNA
3.-PASAR A VERIFICAR LA EPROM EXTERNA
4.-PASAR A LEER LA EPROM EXTERNA

A continuación se describe el accionamiento que se efectuaría al invocar cada una de las opciones del menú de programación de la EPROM externa.

1.-VERIFICAR QUE LA EPROM ESTÉ COMPLETAMENTE BORRADA. Al invocar el usuario esta opción, PUMMA le pide especificar el tipo de memoria EPROM a verificar (e.g. 2716, 2732, 2764, 27128, 27256 o 27512), una vez hecho esto aparecerá en pantalla, con fines recordatorios, un letrero que indica los puentes (jumpers), que deberán estar colocados en la CMT SIMMP-2, de acuerdo al tipo de memoria que se va a verificar, una vez que el usuario valida la verificación transcurrirán unos segundos después de los cuales PUMMA indicará en pantalla si la memoria EPROM externa está completamente borrada, en caso de que exista cuando menos un byte cuyo contenido sea diferente de FFH, PUMMA indicará cuantas localidades de la EPROM están en este caso.

2.-PASAR A PROGRAMAR LA EPROM EXTERNA. Mediante esta opción el usuario puede programar la memoria EPROM externa que se encuentre colocada en la base de 28 pines asociada con la memoria EPROM de la CMT SIMMP-2; al invocarla PUMMA requerirá al usuario la selección, en cuanto a la fuente que originaría la información a programar, de una de las siguientes dos alternativas:

1.- PROGRAMAR LA MEMORIA EPROM DESDE TECLADO

2 - PROGRAMAR LA MEMORIA EPROM DESDE DISCO

Si el usuario selecciona la alternativa uno la fuente de información a programar será un LH que se introduce desde el teclado de la computadora anfitriona junto con las direcciones asociadas, en caso de que se opte por la alternativa dos, el LH a programar se obtendrá de un archivo BLM o LEM, pidiendo PUMMA la dirección inicial a programar en la memoria EPROM; cabe señalar aquí que las direcciones deberán ser especificadas desde el punto de vista de la memoria que se va a programar y no del intervalo de direcciones que estarían asociadas con la misma en el mapa de memoria de la CMT SIMMP-2 operando en modo expandido, así, para programar una memoria 2764, se especificarían direcciones en un intervalo que va de la dirección $0000 a $1FFF.
Una vez que el LH a programar ha sido validado, PUMMA pide al usuario el tipo de memoria EPROM a programar debiendo el usuario especificar únicamente el número que aparece al final de el número de parte del fabricante, por ejemplo, si el usuario desea programar una memoria 27C64 o 2764, deberá al solicitarle PUMMA el tipo de memoria teclear 64 seguido de <CR>; en seguida a lo anterior PUMMA indicará en pantalla los puentes (jumpers) que deben estar colocados en la CMT SIMMP-2 de acuerdo con el tipo de memoria que se seleccionó, después de esto PUMMA pide al usuario información acerca de la magnitud del voltaje (12.5 o mayor que 12.5 Volts) a emplearse al programar, indicando al usuario que puentes en la CMT SIMMP-2 deberán estar colocados, para de inmediato pedirle que conecte el voltaje de programación en la terminal P de la CMT SIMMP-2, oprimiendo aquí cualquier tecla se inicia el proceso de programación y verificación byte por byte del LH correspondiente, una vez terminado esto PUMMA indica en pantalla el número de errores detectados al programar, que será cero en el caso de que la programación haya sido completamente exitosa; una vez que se ha completado el proceso de programación PUMMA pide al usuario que desconecte el voltaje de programación de la terminal P de la CMT SIMMP-2, para retornar de inmediato al menú de programación de la EPROM.

En caso de que la EPROM a programar sea la 2732 o 27512, debido a características especiales de las mismas, el proceso de verificación no puede efectuarse en forma simultánea al de programación, debiendo el usuario hacer la verificación correspondiente a posteriori empleando para ello la opción tres del menú de programación de la EPROM que se explicará mas adelante.

3.-PASAR A VERIFICAR LA EPROM EXTERNA. Mediante esta opción el usuario puede verificar el contenido de la EPROM externa que se encuentre colocada en la base de 28 pines asociada con la memoria EPROM de la CMT SIMMP-2; al invocarla PUMMA requerirá al usuario la selección, en cuanto a la fuente que originaría la información a verificar, de una de las siguientes dos alternativas:

1.- VERIFICAR LA MEMORIA EPROM DESDE TECLADO

2.- VERIFICAR LA MEMORIA EPROM DESDE DISCO

Si el usuario selecciona la alternativa uno la fuente de información a verificar será un LH que se introduce desde el teclado de la computadora anfitriona junto con las direcciones asociadas, en caso de que se opte por la alternativa dos, el LH a verificar se obtendrá de un archivo BLM o LEM, pidiendo PUMMA la dirección inicial a verificar en la memoria EPROM; cabe señalar aquí que las direcciones deberán ser especificadas desde el punto de vista de la memoria que se va a programar y no del intervalo de direcciones que estarían asociadas con la misma en el mapa de memoria de la CMT SIMMP-2 operando en modo expandido, así, para verificar una memoria 2764, se especificarían direcciones en un intervalo que va de la dirección $0000 a $1FFF.

Una vez que el LH a verificar ha sido validado, PUMMA pide al usuario el tipo de memoria EPROM a verificar debiendo el usuario especificar únicamente el número que aparece al final de el número de parte del fabricante, por ejemplo, si el usuario desea verificar una memoria 27C64 o 2764, deberá al solicitarle PUMMA el tipo de memoria teclear 64 seguido de <CR>; en seguida a lo anterior PUMMA indicará en pantalla los puentes (jumpers) que deben estar colocados en la CMT SIMMP-2 de acuerdo con el tipo de memoria que se seleccionó, oprimiendo aquí cualquier tecla se inicia el proceso de verificación byte por byte del LH correspondiente, desplegando PUMMA la dirección a verificar seguida por el byte fuente correspondiente y el byte leído en la EPROM, en caso de haber una discrepancia el proceso se detiene, pudiendo el usuario continuar la verificación oprimiendo cualquier tecla que no sea <Q>, en caso de oprimir <Q> el proceso de verificación es abortado, ya sea que el proceso de verificación se lleve a cabo en su totalidad o que el mismo sea abortado PUMMA indicará al usuario el número de errores detectados al verificar.

4.-PASAR A LEER LA EPROM EXTERNA. Esta opción permite al usuario leer el contenido de la EPROM externa, las direcciones deberán ser especificadas desde el punto de vista de las propias memorias y no de acuerdo con la posición ocupen en el mapa de memoria de la CMT SIMMP-2 operando en modo expandido, por ejemplo, el intervalo de direcciones válido para una memoria 2764 será 0000-1FFF y si se lee un bloque de 256 bytes a partir de la dirección 0000 en el modo expandido dicho bloque podría ser visible en el intervalo de direcciones E000-E0FF, los pasos a seguir para la lectura una vez que la opción ha sido invocada son los mismos que se llevan a cabo al usar la opción de lectura de memoria descrita anteriormente, (ver opción uno del menú de manejo de memoria).

Si el usuario opta por leer la memoria EPROM externa como lista de bytes, al final de la lectura PUMMA le pedirá especificar si desea guardar en disco el tramo leído, en caso que esto asi sea PUMMA pedirá secuencialmente al usuario el nombre que se dará al archivo, unidad de disco y tipo (BLM o LEM).

MENÚ DE EDICIÓN HEXADECIMAL

Esta opción permite editar el listado hexadecimal (LH) presente, dicho LH pudo haber sido introducido por el usuario empleando la opción uno del menú principal, o bien podría haber sido validado, leyendo de disco el archivo BLM, LEM o S19 correspondiente.
Al requerir el usuario pasar al editor hexadecimal, PUMMA presenta las siguientes opciones:

1.-INSERTAR BYTES
2.-BORRAR BYTES
3.-LISTAR
4.-CAMBIAR BYTES
5.-AGREGAR BYTES

A continuación se describe como emplear cada una de las opciones de edición hexadecimal mencionadas anteriormente.

1.-INSERTAR BYTES. Esta opción permite al usuario insertar una cadena de bytes en el LH presente, al invocarla, PUMMA pide las direcciones inicial y final de inserción, que deberán estar dentro del intervalo que valida al LH, después de esto PUMMA requiere al usuario la introducción

de la cadena a insertar, siguiéndose para ello el mismo procedimiento que se lleva a cabo al validar un LH empleando la opción uno del menú principal, en caso de que la dirección inicial de inserción especificada sea menor que la dirección final de inserción, o bien que una de las direcciones o ambas no pertenecieran al intervalo de direcciones que valida al LH, PUMMA indicará al usuario que hay un error, retornando de inmediato al menú de edición para que el usuario pueda repetir el proceso si así lo desea; si al pedir PUMMA la dirección inicial de inserción el usuario oprime <Q> y luego <RETURN>, PUMMA abortará el proceso retornando de inmediato al menú de edición, en caso de que el usuario lleve a cabo el accionamiento anterior al pedir PUMMA la dirección final de inserción, PUMMA supondrá que el usuario requiere insertar un sólo dato consignado en la dirección inicial antes especificada.

2.-BORRAR BYTES. Esta opción permite al usuario borrar una cadena de bytes en el LH presente, al invocarla, PUMMA pide las direcciones inicial y final a borrar, que deberán estar dentro del intervalo que valida al LH, en caso de que la dirección inicial de borrado especificada sea menor que la dirección final de borrado, o bien que una de las direcciones o ambas no pertenecieran al intervalo de direcciones que valida al LH, PUMMA indicará al usuario que hay un error, retornando de inmediato al menú de edición para que el usuario pueda repetir el proceso si así lo desea; si al pedir PUMMA la dirección inicial a borrar el usuario oprime <Q> y luego <RETURN>, PUMMA abortará el proceso retornando de inmediato al menú de edición, en caso de que el usuario lleve a cabo el accionamiento anterior al pedir PUMMA la dirección final a borrar, PUMMA supondrá que el usuario requiere borrar únicamente el byte consignado en la dirección inicial antes especificada.

3.-LISTAR. Esta opción permite al usuario listar el LH presente, al invocarla, PUMMA desplegará en pantalla el siguiente letrero: OPRIMIR CUALQUIER TECLA PARA LISTAR SUCESIVAMENTE de esta manera al oprimirse cualquier tecla que no sea <F>, <Q>, <C>, <R>, o <S>, PUMMA desplegará en pantalla el LH con el que se trabaje en un momento dado, comenzando con la dirección inicial asociada hasta llegar a la dirección final correspondiente, en caso de que al listar sucesivamente se oprima alguna de las teclas mencionadas anteriormente se validarán las facilidades que a continuación se describen:

<S> Salta a dirección especificada por el usuario, que desde luego deberá estar dentro del rango que valida al LH presente.

<F> Salta a la dirección final del LH presente.<R> Regresa la siguiente dirección a listar una posición hacia atrás.

<C> Al invocarse esta alternativa el usuario puede cambiar el dato asociado con la dirección inmediata anterior.

<Q> La opresión de esta tecla al listar aborta el proceso de listado.

4.-CAMBIAR BYTES. Esta opción permite al usuario cambiar una cadena de bytes en el LH presente, al invocarla, PUMMA pide las direcciones inicial y final a cambiar, que deberán estar dentro del intervalo que valida al LH, después de esto PUMMA requiere al usuario la introducción de la cadena que substituirá a la cadena existente en las direcciones especificadas para el cambio, siguiéndose para ello el mismo procedimiento que se lleva a cabo al validar un LH empleando la opción uno del menú principal, en caso de que la dirección inicial de cambio especificada sea menor que la dirección final de cambio, o bien que una de las direcciones o ambas no

pertenecieran al intervalo de direcciones que valida al LH, PUMMA indicará al usuario que hay un error, retornando de inmediato al menú de edición para que el usuario pueda repetir el proceso si así lo desea; si al pedir PUMMA la dirección inicial de cambio el usuario oprime <Q> y luego <RETURN>, PUMMA abortará el proceso retornando de inmediato al menú de edición, en caso de que el usuario lleve a cabo el accionamiento anterior al pedir PUMMA la dirección final de cambio, PUMMA supondrá que el usuario requiere cambiar un sólo dato consignado en la dirección inicial antes especificada.

5.- AGREGAR BYTES. Esta opción permite al usuario agregar bytes adicionales al LH presente, al invocarla, PUMMA pide al usuario la nueva dirección final del LH, para después requerir del usuario los datos adicionales correspondientes.

REFERENCIAS:

1.- Sibigtroth Jim, Rhoades Mike, Longan John.
MC68HC11 Bootstrap Mode. Motorola Semiconductor Application Note. AN1060/D.
Motorola Inc.
1990.

2.- MC68HC11F1 Technical Data.
Motorola Inc.
1990.

3.- M68HC11 Reference Manual.
Motorola Inc.
1991.

PARTE DOS DE APÉNDICE A

GUÍA DE USO DE LA COMPUTADORA MONOTABLILLA SIMMP-2.

La computadora monotablilla (CMT) SIMMP-2 es una arquitectura basada en el microcontrolador 68HC11F1 fabricado por MOTOROLA, la tarjeta incorpora un programador de memorias EPROM y puede ser operada en cualquiera de los cuatro modos de operación del 68HC11, aunque lo recomendable, dadas las características propias del 68HC11F1, es operar la tarjeta en los modos expandido o boot-strap.
Para operar el programador de memorias EPROM la tarjeta necesariamente deberá operar en modo boot-strap, mas adelante se explicará el funcionamiento del mismo.
Al operar la tarjeta SIMMP-2 en los modos expandido o TEST se pueden configurar diferentes mapas de memoria con sus respectivos submapas de puertos, dichas configuraciones se logran cambiando puentes (jumpers), así como también firmware residente en la tarjeta; mas adelante se hablará de como hacer estas configuraciones.

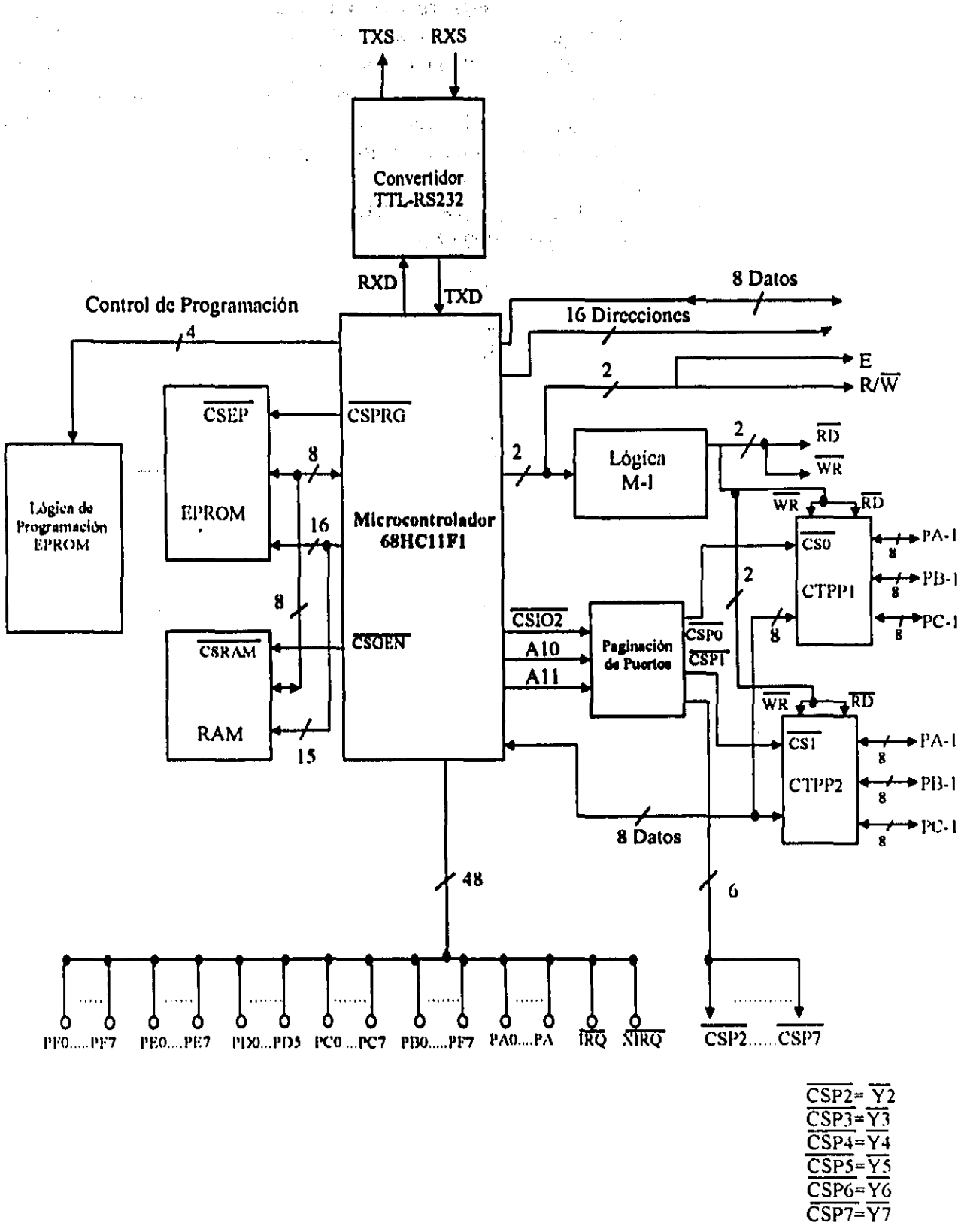
En la Figura 2.1 se muestra el diagrama a bloques de la CMT SIMMP-2, apreciándose que el usuario tiene acceso a casi todas las líneas asociadas ya sea con periféricos propios del microcontrolador o con líneas de control del mismo, el puerto G del microcontrolador no está disponible para el usuario ya que es empleado por la CMT SIMMP-2 para arbitrar la paginación de memoria y puertos y el programador de memorias EPROM; a continuación se describe genéricamente el funcionamiento de los bloques funcionales de la CMT SIMMP-2 que aparecen en la Figura 2.1.

MICROCONTROLADOR 68HC11F1.- Este es el circuito integrado número uno de la CMT SIMMP-2 y es una versión del 68HC11 que cuenta, con las siguientes características adicionales a las que se cuentan en otras versiones del 68HC11 y estas son entre otras las siguientes:

a) Bus de direcciones demultiplexado.
b) Puerto paralelo F de salida, disponible para el usuario sólo en los modos boot-strap o single-chip. Los pines que dan acceso a este puerto validan la parte baja del bus de direcciones al operar el microcontrolador en los modos expandido o TEST.
c) Puerto G bidireccional. Los cuatro bits más significativos de este puerto, pueden ser empleados para controlar la paginación de memoria y puertos cuando el microcontrolador opera en modo expandido o TEST, en cuyo caso quedarían disponibles para el usuario sólo los cuatro bits menos significativos de este puerto. En la tarjeta SIMMP-2 este puerto no está disponible para el usuario debido a que el mismo es empleado para arbitrar parte del funcionamiento de la arquitectura.

CONVERTIDOR TTL-RS 232.- Este bloque funcional está realizado con el chip MAX-232 muy popular en la industria para estos fines; la función del mismo consiste en cambiar los niveles lógicos de voltaje TTL a los niveles RS-232 (+12 volts y -12 volts) y viceversa.

LÓGICA M-I.- Este bloque genera señalización de control de periféricos INTEL para que los mismos puedan ser empleados por la CMT SIMMP-2, es importante aclarar aquí que el periférico INTEL que se deseara conectar no operará correctamente si la velocidad de operación del mismo no es compatible con la correspondiente a la tarjeta SIMMP-2. Las señales de control generadas son WR y RD.



$\overline{CSP2}=\overline{Y2}$
$\overline{CSP3}=\overline{Y3}$
$\overline{CSP4}=\overline{Y4}$
$\overline{CSP5}=\overline{Y5}$
$\overline{CSP6}=\overline{Y6}$
$\overline{CSP7}=\overline{Y7}$

Figura 2.1.- Diagrama de Bloques de la CMT SIMMP-2.

PAGINADOR DE PUERTOS.-Este bloque está realizado con un decodificador de 3 a 8 74LS138 y el mismo genera 8 líneas de paginación de puertos denominadas como CSP0 a CSP7, validando cada una de estas un intervalo de 128 direcciones, en los conectores de la Tarjeta SIMMP-2 el usuario tiene acceso a seis de las ocho líneas mencionadas anteriormente (CSP2 a CSP7), siendo las mismas respectivamente denominadas como Y2 a Y7; a continuación se indica el intervalo de direcciones que verifica cada línea de paginación de puertos.

LÍNEA DE PAGINACIÓN DE PUERTO	LÍNEA EN CONECTORES DE SIMMP-2	INTERVALO DE DIRECCIONES
CSP0	*	$1800 a $17FF
CSP1	*	$1880 a $18FF
CSP2	Y2	$1900 a $197F
CSP3	Y3	$1980 a $19FF
CSP4	Y4	$1A00 a $1A7F
CSP5	Y5	$1A80 a $1AFF
CSP6	Y6	$1B00 a $1B7F
CSP7	Y7	$1B80 a $1BFF

* Estas líneas de paginación de puerto no están disponibles para el usuario ya que el sistema las usa para habilitar los bloques funcionales CTPP1 y CTPP2 de los cuales se hablará mas adelante.

CONJUNTO TRIPLE DE PUERTOS PARALELOS NÚMERO UNO (CTPP1).- Este bloque funcional está realizado por un chip INTEL 82C55 que es un conjunto de tres puertos paralelos programables denotados en la hoja de datos correspondiente como PA, PB, y PC, cada uno de estos puertos tienen un registro de datos asociado; en la notación de la tarjeta SIMMP-2 estos puertos son nombrados como PA-1, PB-1 y PC-1 para evitar confusiones con los puertos que son denotados con las letras A, B, y C en el microcontrolador 68HC11F1. El 82C55 tiene cuatro direcciones asociadas, tres corresponden con los tres puertos mencionados y la cuarta está asociada con un registro de control del chip mediante el cual se puede programar tanto el modo en que va a operar el chip como la naturaleza de entrada o salida de cada uno de los puertos del mismo; las direcciones asociadas en el mapa de puertos de la tarjeta SIMMP-2 son las siguientes:

PUERTO O REGISTRO EN CTPP1	DIRECCIÓN EN SIMMP-2
PA-1	$1800
PB-1	$1801
PC-1	$1802
REGISTRO DE CONTROL	$1803

Dado que el 82C55 usa sólo dos líneas de direccionamiento y cada línea de paginación de puertos en la CMT SIMMP-2 se verifica al invocarse un intervalo de 128 direcciones, las direcciones mencionadas anteriormente estarán repetidas 32 veces esto es: el puerto PA-1 estará en la direcciones 1800, 1804, 1808, hasta la 187C; el puerto PB-1 estará en las direcciones 1801, 1805, 1809, hasta la 187D; el puerto PC-1 estará en las direcciones 1802, 1806, 180A, hasta la 187E; el registro de control de este bloque CTPP1 estará localizado en las direcciones 1803, 1807, 180B, hasta la 187F.

CONJUNTO TRIPLE DE PUERTOS PARALELOS NÚMERO DOS (CTPP2).- Este bloque funcional está realizado por un chip INTEL 82C55 que es un conjunto de tres puertos paralelos programables denotados en la hoja de datos correspondiente como PA, PB, y PC, cada uno de estos puertos tienen un registro de datos asociado; en la notación asociada con la tarjeta SIMMP-2 estos puertos son nombrados como PA-2, PB-2 y PC-2 para evitar confusiones con los puertos que son denotados con las letras A, B, y C en el microcontrolador 68HC11F1. El 82C55 tiene cuatro direcciones asociadas, tres corresponden con los tres puertos mencionados y la cuarta está asociada con un registro de control del chip mediante el cual se puede programar tanto el modo en que va a operar el chip como la naturaleza de entrada o salida de cada uno de los puertos del mismo; las direcciones asociadas en el mapa de puertos de la tarjeta SIMMP-2 son las siguientes:

PUERTO O REGISTRO EN CTPP2	DIRECCIÓN EN SIMMP-2
PA-2	$1880
PB-2	$1881
PC-2	$1882
REGISTRO DE CONTROL	$1883

Dado que el 82C55 usa sólo dos líneas de direccionamiento y cada línea de paginación de puertos en la CMT SIMMP-2 se verifica al invocarse un intervalo de 128 direcciones, las direcciones mencionadas anteriormente estarán repetidas 32 veces esto es: el puerto PA-2 estará en la direcciones 1880, 1884, 1888, hasta la 18FC; el puerto PB-2 estará en las direcciones 1881, 1885, 1889, hasta la 18FD; el puerto PC-2 estará en las direcciones 1882, 1886, 188A, hasta la 18FE; el registro de control de este bloque CTPP2 estará localizado en las direcciones 1883, 1887, 188B, hasta la 18FF.

Para más detalles acerca del funcionamiento del 82C55 se puede consultar las especificaciones técnicas proporcionadas por INTEL.

Las versiones del 82C55 que al emplearse en la CMT SIMMP-2 han sido compatibles con la velocidad de operación de la tarjeta son la 82C55-A2 y la KS82C55A-8CP.

Los bloques CTPP1 y CTPP2 tienen sentido como parte de la arquitectura de la CMT SIMMP-2 cuando ésta opera en modo expandido o TEST, en otro caso no formarían parte del mapa de memoria.

En caso de que la tarjeta CMT SIMMP-2 no cuente con ninguno de los chips 82C55 que realizan los bloques CTPP1 y CTPP2 el usuario podrá emplear registros 74LS373 o 74LS374 o periféricos MOTOROLA para realizar puertos de entrada o salida, mas adelante se explicará como hacer esto para el caso de los chips TTL mencionados.

MEMORIA RAM.- Este bloque es la memoria RAM estática que se emplee en la CMT SIMMP-2 en un momento dado, para ello existe en la tarjeta una base de 28 pines que puede aceptar memorias RAM de 8 o 32 kb, dependiendo de el mapa de memoria en modo expandido que haya escogido el usuario.

MEMORIA EPROM.- Este bloque podrá ser una memoria EPROM de 2k a 64k para fines del programador de memorias EPROM contenido en la tarjeta, si la EPROM en cuestión está colocada para ser parte de el mapa de memoria de la CMT SIMMP-2 operando en modo expandido o TEST necesariamente su tamaño deberá ser de 8k o 32k, dependiendo del mapa de memoria escogido por el usuario.

LÓGICA DE PROGRAMACIÓN DE MEMORIAS EPROM.- Este bloque está integrado por circuitería analógica y digital que realiza fisicamente la programación de memorias EPROM, efectuándose esto mediante el arbitrio del microcontrolador de la tarjeta SIMMP-2 ejecutando software bajado desde una computadora anfitriona vía puerto serie, empleando para ello al manejador hexadecimal PUMMA, para más detalles acerca del uso del programador de memorias EPROM consultar el tema sobre el particular que se expone mas adelante en este capítulo o bien leer lo referente a esto en el capítulo que habla sobre la operación del programa PUMMA.EXE.

LOCALIZACIÓN DE COMPONENTES EN LA CMT SIMMP-2

En la Figura 2.2 se muestra una vista de planta de la CMT SIMMP-2, en dicha figura se aprecia la localización de los diversos componentes que integran la arquitectura de la tarjeta.
En la parte central se aprecia la localización del microcontrolador 68HC11F1 que es el corazón de SIMMP-2.
A la izquierda de la tarjeta aparecen los bloques CTPP1 y CTPP2 observándose a la izquierda de los mismos la ubicación de dos conectores, denominados con las letras A y B respectivamente, dichos conectores dan acceso al usuario a los distintos puertos paralelos programables correspondientes con los bloques mencionados y a algunas líneas importantes del propio microcontrolador tales como XIRQ, IRQ y las líneas asociadas del puerto D del 68HC11F1; en el conector A aparecen líneas de puerto del bloque CTPP1 y en el conector B aparecen líneas de puerto asociadas con el CTPP2.
A la derecha de la tarjeta aparecen otros dos conectores, uno es denominado como conector C y el otro es el conector auxiliar de expansión, en este último el usuario tiene acceso a líneas control, de selección de puerto y de polarización; en el conector C se tiene acceso a líneas de puerto del propio microcontrolador; en la Figura 2.3 se muestra en detalle las líneas asociadas con cada uno de los postes que integran los conectores antes mencionados.
En la parte inferior izquierda se encuentra parte del hardware relacionado con el programador de memorias EPROM.
A la izquierda del microcontrolador se puede observar el botón de restablecimiento manual.
En la parte central inferior de la tarjeta se observa un conector de polarización, el mismo es de cuatro terminales, adjunto al mismo se ve la leyenda(5 0 P 6), indicando esto los postes por donde se han de conectar a la tarjeta SIMMP-2 la fuente de cinco volts (poste con el número cinco), la fuente que proporcionaria el voltaje de programación de las memorias EPROM (poste con la letra P) y la línea común de tierra de ambas fuentes mencionadas (poste con el guarismo cero); al poste con el guarismo seis no se le conecta nada y su uso se reserva para posibles mejoras futuras del programador de memorias EPROM.
En la parte inferior izquierda, entre el circuito que valida al bloque CTPP2 y el circuito MAX-232, se encuentra un conector de tres postes por donde ha de conectarse cable de enlace serie entre SIMMP-2 y una computadora anfitriona, en la Figura 2.4 se muestra el conexionado de el cable de enlace serie.

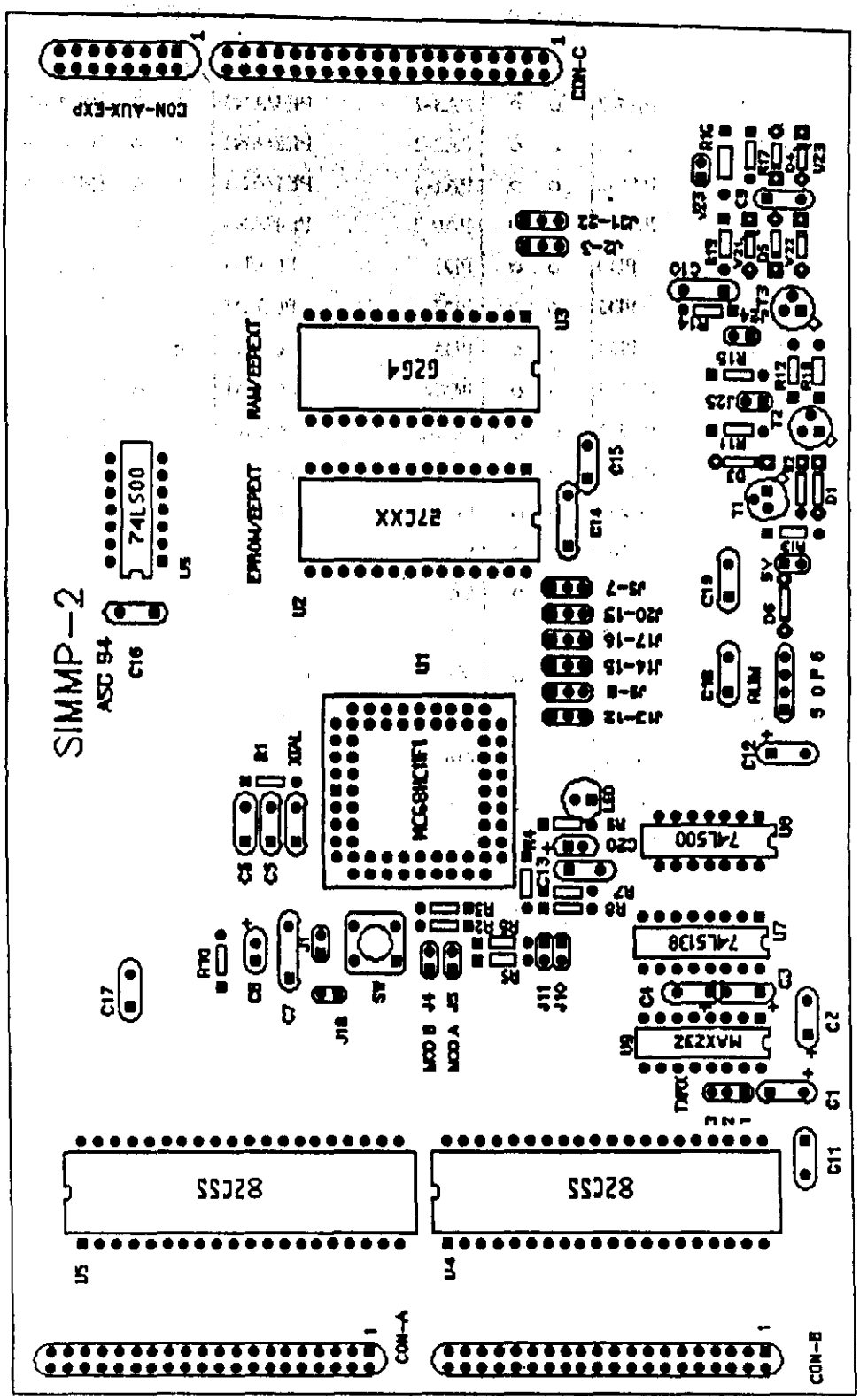


Figura 2.2.- Vista superior de la CMT SIMMP-2

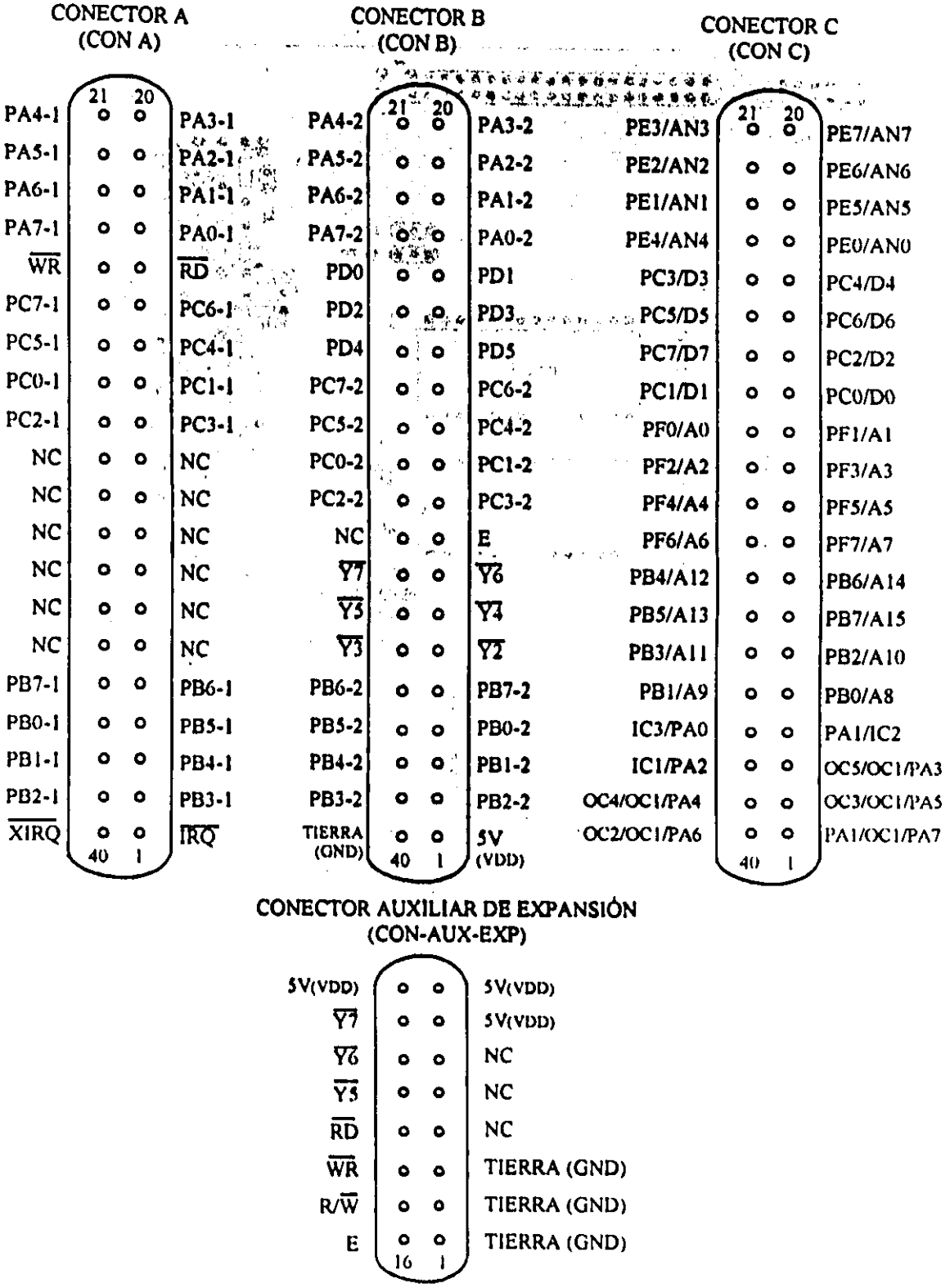


Figura 2.3.- Conectores de la CMT SIMMP-2.

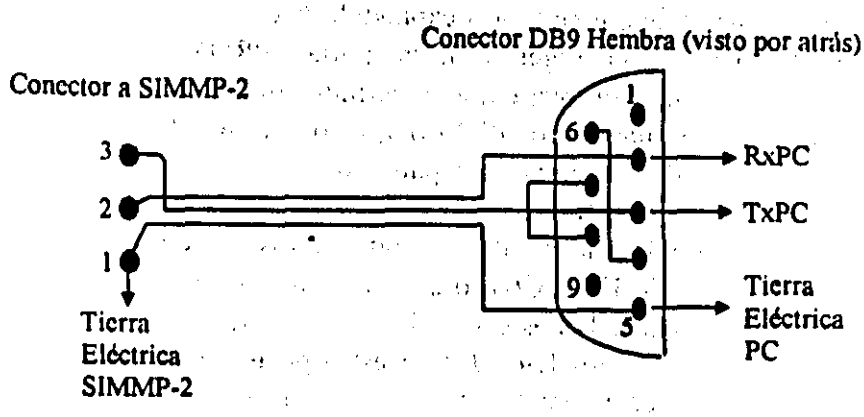


Figura 2.4.- Detalle de conexionado del cable de enlace serie.

Los puentes (jumpers) juegan un papel muy importante ya que con ellos se configuran distintas maneras de operar además de que con ellos se habilitan, para cada caso, las distintas posibilidades con que se cuenta para programar memorias EPROM de acuerdo con su tamaño. A continuación se describe en forma genérica la función de cada uno de los puentes y/o postes asociados a los mismos que existen en la tarjeta SIMMP-2.

a) Postes asociados con el puente J1. Uno de estos postes está conectado con la terminal de restablecimiento (RESET) del microcontrolador y el otro está conectado con el lado negativo (tierra) de la fuente de polarización que alimente a la tarjeta, por lo tanto, el puente J1 nunca deberá ser colocado y la presencia de los postes mencionados facilita al usuario conectar externamente a la tarjeta un botón de restablecimiento o algún circuito especial para estos fines.
b) Trio de Postes asociados con los puentes J2 y J3. Estos están colocados abajo a la derecha de la base de la memoria RAM y permiten la colocación de uno sólo de los puentes J2 o J3 que seleccionan si la memoria EPROM a programar es la 27C16, 27C32 o bien las memorias 27C64, 27C128, 27C256 o 27C512.
c) Postes para el puente J4. A estos postes ha de conectarse un puente, cuando se desea que sea bajo el nivel lógico asociado con la terminal MODB del microcontrolador, en otro caso no ha de conectarse el puente J4.
d) Postes para el puente J5. A estos postes ha de conectarse un puente, cuando se desea que sea bajo el nivel lógico asociado con la terminal MODA del microcontrolador, en otro caso no ha de conectarse el puente J5.
e) Trio de postes asociados con los puentes J6 y J7. A estos postes ha de conectarse ya sea el puente J6 o el puente J7, aunque para la versión actual del programador de memorias EPROM contenido en la tarjeta debe colocarse siempre el puente J6, posibles mejoras futuras del programador de memorias EPROM podrian llegar a requerir la colocación del puente J7 para este trio de postes.
f) Trio de postes asociados con los puentes J8 y J9. En estos postes se coloca el puente J9 para operación normal de lectura de la memoria EPROM, en caso de que se use el programador ha de conectarse a estos postes el puente J8.
g) Postes para el puente J10. Aqui debe conectarse el puente J10, para la versión actual del firmware residente en el microcontrolador de la tarjeta este puente no debe conectarse.

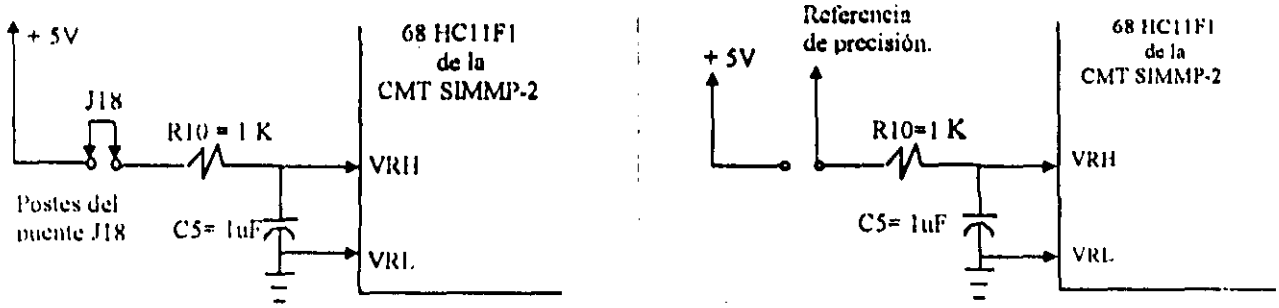
h) Postes para el puente J11. A estos postes se conecta el puente J11; el tener este puente conectado cuando el sistema se opere en modo expandido hace que el microcontrolador salte, después de un restablecimiento, a el origen de la memoria EPROM externa, en otro caso el microcontrolador pasaría a ejecutar firmware que le permite comunicarse con una computadora anfitriona que este ejecutando el manejador PUMMA, esto permite al usuario hacer desarrollo cuando se trabaje la tarjeta SIMMP-2 en modo expandido.

i) Trío de postes asociado con los puentes J12 y J13. A estos postes ha de conectarse el puente J12 o el puente J13; el puente J12 se pondría cuando se tuviera colocada en la base de la EPROM una memoria 27C16, 27C32, 27C64, o 27C128; en caso de que la memoria EPROM sea la 27C256 o la 27C512 el puente J13 es el que debe ser puesto.

j) Trío de postes asociado con los puentes J14 y J15. A estos postes se conecta el puente J14 o el puente J15; el puente J15 se conecta cuando en la base de la memoria EPROM está colocada para programación una memoria 27C512, en otro caso se debe colocar el puente J14.

k) Trío de postes asociado con los puentes J16 y J17. Aquí ha de conectarse el puente J17 cuando se este programando una EPROM 27C32 o 27C512, al programar otro tipo de EPROM se debe colocar aquí el puente J16.

l) Postes asociados con el puente J18. Este puente conecta a la terminal VRH del microcontrolador el mismo voltaje de cinco volts que polariza a la tarjeta; en caso de que el usuario desee una referencia de voltaje más precisa, deberá desconectar el puente J18 y conectar la terminal positiva de la fuente que proporcione la referencia de voltaje a el poste ligado, vía una red RC, con la entrada VRH del microcontrolador; la terminal negativa de la fuente de referencia deberá conectarse con la referencia de cero volts (tierra) de la tarjeta SIMMP-2, ya que en la arquitectura SIMMP-2, la entrada VRL del microcontrolador está conectada a tierra, ver la Figura 2.5. Cabe recordar aquí que el voltaje aplicado en la entrada VRH del microcontrolador, es empleado por el mismo, no sólo como referencia para el convertidor analógico digital sino también como voltaje de entrada para la lógica de programación de la memoria EEPROM interna, por lo tanto, si el usuario desea programar dicha memoria, el puente J18 deberá estar colocado en su lugar. El poste donde el usuario ha de conectar una referencia externa de voltaje a la entrada VRH, previa desconexión del puente J18, es el que se encuentra casi en línea con los postes para el puente J1, véase la Figura 2.6.



a) Cuando dicha referencia es la misma que polariza la tarjeta.

b) Cuando el usuario desea conectar otra referencia de precisión a VRH.

Figura 2.5.- Esquema eléctrico de conexión a la entrada VRH del microcontrolador.

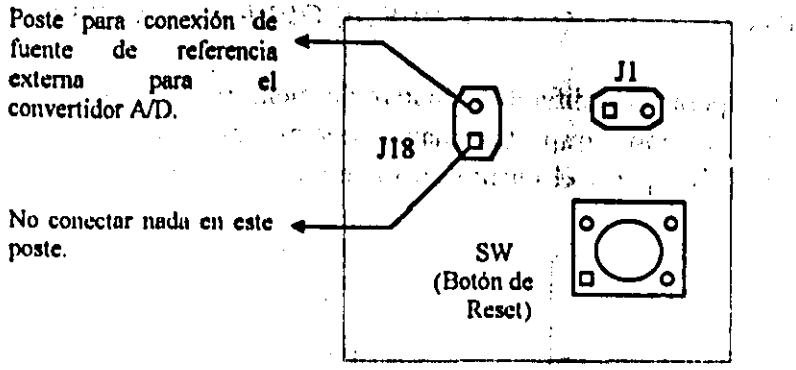


Figura 2.6.- Localización en la CMT SIMMP-2 del poste para conexión de fuente de referencia externa, cuando se desconecta el puente J18.

a) Trío de postes asociado con los puentes J19 y J20. Aquí debe colocarse el puente J19 cuando la memoria RAM colocada en la base correspondiente es de 32k, en caso de que la misma sea de 8k ha de colocarse el puente J20.
b) Trío de Postes asociados con los puentes J21 y J22. Estos están colocados abajo a la derecha de la base de la memoria RAM y permiten la colocación de uno sólo de los puentes J21 o J22 que seleccionan si la memoria EPROM a programar es la 27C16, 27C32 o bien las memorias 27C64, 27C128, 27C256 o 27C512.
c) Postes para el puente J23. El uso del puente J23 se reserva para posibles mejoras futuras del programador de memorias EPROM; por lo tanto, no es necesaria su conexión en la tarjeta SIMMP-2, los postes mencionados pueden ser empleados como puntos de prueba para verificar el funcionamiento del programador de memorias EPROM, ya que en ellos aparece el voltaje de programación cuando se habilita esta facilidad desde la computadora anfitriona que maneje la tarjeta, empleando el manejador PUMMA, en un momento dado.
d) Postes para el puente J24. En estos postes se coloca el puente J24 cuando se desee programar una EPROM cuyo voltaje de programación sea de 12.5 volts; en caso de que la memoria a programar requiera más voltaje para su programación el puente mencionado aquí no debe ser puesto.
e) Postes para el puente J25. Aquí se coloca el puente J25, que deberá estar siempre puesto, para una correcta operación del programador de memorias EPROM y de otras facilidades con las que cuenta la tarjeta SIMMP-2.

MAPAS DE MEMORIA CON LOS QUE PUEDE OPERAR LA CMT SIMMP-2

La CMT SIMMP-2 puede operar con diferentes mapas de memoria; cuando la CMT SIMMP-2 opera en modo single-chip o boot-strap, los mapas correspondientes son los naturales del microcontrolador 68HC11F1, al operar el mismo en los modos mencionados, véanse las Figuras 2.7 y 2.8.

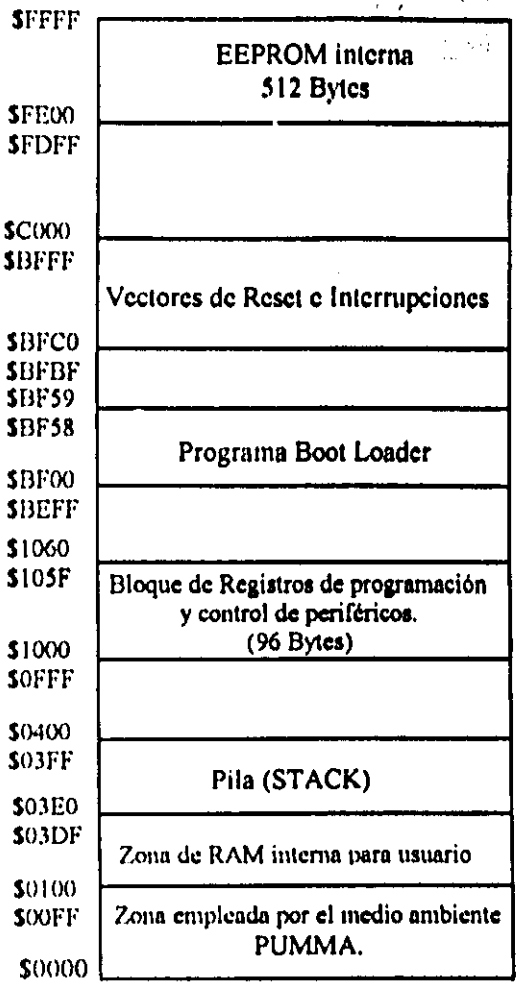


Figura 2.7.- Mapa de memoria de la CMT SIMMP-2 operando en modo Boot-Strap.

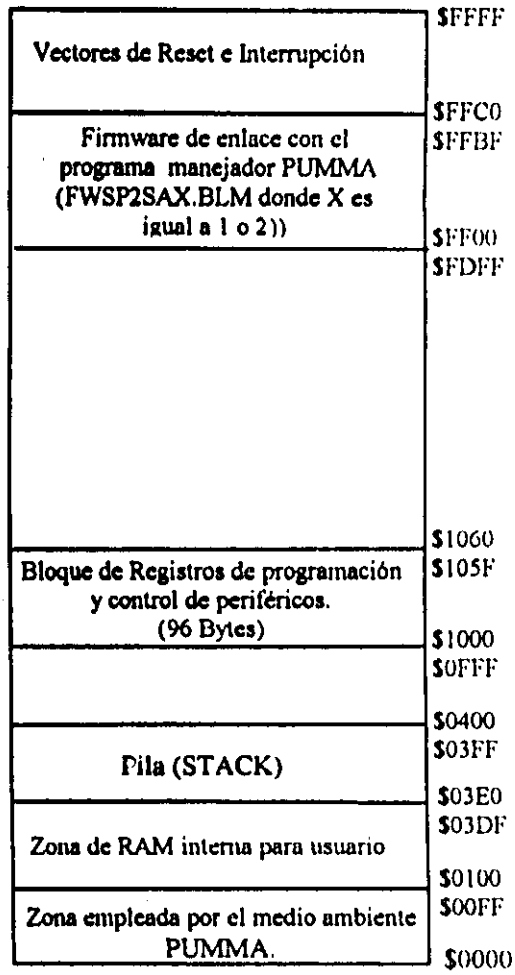


Figura 2.8.- Mapa de memoria de la CMT SIMMP-2 operando en modo Single-Chip.

Al operar en los modos expandido o TEST el microcontrolador 68HC11F1 cuenta con lineas de selección de memoria y puertos que pueden ser configurados por software, esto hace que en estos casos puedan ser configurados diversos mapas de memoria. Cada mapa es configurado por hardware (colocando puentes) y software (programando firmware residente en la tarjeta); para todos estos mapas, existe un submapa de puertos, si se desea conocer más detalles acerca del mismo, consultar el tema sobre este particular, en este mismo capítulo; a continuación se describe la manera de configurar diferentes mapas de memoria cuando la tarjeta SIMMP-2 opere en los

modos expandido o TEST; en la Tabla 2.1 se detalla en forma resumida la configuración de puentes y firmware residente requerido para cada uno de dichos mapas.

Mapa EA

Para este mapa se tiene el siguiente perfil: Operación en modo expandido con 8k de RAM externa, 1k de RAM interna, 7.5k de EPROM y 0.5k de EEPROM interna con firmware SP2EA residente. Para configurar este mapa se deben colocar los siguientes puentes: J3, J8 o J9, J12, J14, J16, J20 y J22, además de que uno de los archivos FWSP2EA1.BLM o FWSP2EA2.BLM deberá haber sido programado en la EEPROM interna a partir de su origen; para el caso del primer archivo mencionado el área de vectores de RESET e interrupción está programada únicamente con los vectores correspondientes a los tres tipos de RESET con los que cuenta el 68HC11 (apuntando estos al origen de la EEPROM interna), dejando al usuario la posibilidad de programar, empleando el manejador PUMMA, los vectores de interrupción que el mismo pudiera llegar a requerir al realizar una determinada aplicación; en lo que toca al segundo archivo mencionado el vector de RESET principal apunta al origen de la EEPROM interna y todos los demás vectores de RESET e interrupción apuntan a la dirección de la página cero, siendo éstas las mismas que las que corresponden a los vectores asociados con el modo boot-strap, lo anterior permitiría al usuario trabajar con el concepto de seudovector de interrupción al programar sus aplicaciones en modo expandido; en la Figura 2.9A se aprecia el mapa de memoria EA; los requerimientos en cuanto a chips de memoria para este mapa son: RAM estática de 8k en la base correspondiente, EPROM de 8k en su respectiva base. Cabe señalar que este mapa es el que viene configurado de origen en la CMT SIMMP-2 con el archivo FWSP2EA1.BLM en la EEPROM.

Mapa EB

Para este mapa se tiene el siguiente perfil: Operación en modo expandido con 28.90625k de RAM externa (29600 bytes), 1k de RAM interna, 31.5k de EPROM y .5k de EEPROM interna con firmware SP2EB residente. Para configurar este mapa se deben colocar los siguientes puentes: J3, J8 o J9, J13, J14, J16, J19 y J22, además de que uno de los archivos FWSP2EB1.BLM o FWSP2EB2.BLM deberá haber sido programado en la EEPROM interna a partir de su origen; para el caso del primer archivo mencionado el área de vectores de RESET e interrupción está programada únicamente con los vectores correspondientes a los tres tipos de RESET con los que cuenta el 68HC11 (apuntando estos al origen de la EEPROM interna), dejando al usuario la posibilidad de programar, empleando el manejador PUMMA, los vectores de interrupción que el mismo pudiera llegar a requerir al realizar una determinada aplicación; en lo que toca al segundo archivo mencionado el vector de RESET principal apunta al origen de la EEPROM interna y todos los demás vectores de RESET e interrupción apuntan a las direcciones de la página cero, siendo estas las mismas que las que corresponden a los vectores asociados con el modo boot-strap, lo anterior permitiría al usuario trabajar con el concepto de seudovector de interrupción al programar sus aplicaciones en modo expandido; en la Figura 2.9B se aprecia el mapa de memoria EB; los requerimientos en cuanto a chips de memoria para este mapa son: RAM estática de 32k en la base correspondiente, EPROM de 32k en su respectiva base.

Mapa EC

Para este mapa se tiene el siguiente perfil: Operación en modo expandido con 28.90625k de RAM externa (29600 bytes), 1k de RAM interna, 8k de EPROM (con direcciones que accesan la misma localidad separadas 8k <espejos>) y .5k de EEPROM interna con firmware SP2EB (el mismo que el correspondiente al mapa EB) residente. Para configurar este mapa se deben colocar los siguientes puentes: J3, J8 o J9, J12, J14, J16, J19 y J22, además de que uno de los archivos FWSP2EB1.BLM o FWSP2EB2.BLM deberá haber sido programado en la EEPROM interna a partir de su origen; para el caso del primer archivo mencionado el área de vectores de RESET e interrupción está programada únicamente con los vectores correspondientes a los tres tipos de RESET con los que cuenta el 68HC11 (apuntando estos al origen de la EEPROM interna), dejando al usuario la posibilidad de programar, empleando el manejador PUMMA, los vectores de interrupción que el mismo pudiera llegar a requerir al realizar una determinada aplicación; en lo que toca al segundo archivo mencionado el vector de RESET principal apunta al origen de la EEPROM interna y todos los demás vectores de RESET e interrupción apuntan a las direcciones de la página cero, siendo estas las mismas que las que corresponden a los vectores asociados con el modo boot-strap, lo anterior permitiría al usuario trabajar con el concepto de seudovector de interrupción al programar sus aplicaciones en modo expandido; en la Figura 2.9B se aprecia el mapa de memoria EB; los requerimientos en cuanto a chips de memoria para este mapa son: RAM estática de 32k en la base correspondiente, EPROM de 8k en su respectiva base.

Mapa TA

Para este mapa se tiene el siguiente perfil: Operación en modo TEST con 8k de RAM externa, 1k de RAM interna, 8k de EPROM y .5k de EEPROM interna con firmware SP2TA residente. Para configurar este mapa se deben colocar los siguientes puentes: J3, J8 o J9, J12, J14, J16, J20 y J22, además de que uno de los archivos FWSP2TA1.BLM o FWSP2TA2.BLM deberá haber sido programado en la EEPROM interna a partir de su origen; para el caso del primer archivo mencionado el área de vectores de RESET e interrupción está programada únicamente con los vectores correspondientes a los tres tipos de RESET con los que cuenta el 68HC11 (apuntando estos al origen de la EEPROM interna), dejando al usuario la posibilidad de programar, empleando el manejador PUMMA, los vectores de interrupción que el mismo pudiera llegar a requerir al realizar una determinada aplicación; en lo que toca al segundo archivo mencionado el vector de RESET principal apunta al origen de la EEPROM interna y todos los demás vectores de RESET e interrupción apuntan a las direcciones de la página cero, siendo estas las mismas que las que corresponden a los vectores asociados con el modo boot-strap, lo anterior permitiría al usuario trabajar con el concepto de seudovector de interrupción al operar el microcontrolador en modo TEST ; en la Figura 2.10A se aprecia el mapa de memoria TA; los requerimientos en cuanto a chips de memoria para este mapa son: RAM estática de 8k en la base correspondiente, EPROM de 8k en su respectiva base.

Para todos los mapas posibles se observa que la página cero (direcciones de la $0000 a la $00FF), está ocupada por lo que se denomina como ambiente PUMMA que es empleado como enlace con la computadora anfitriona cuando se trabaja con el sistema anfitrión-destino para desarrollo; por lo tanto, en tal caso el usuario no deberá modificar localidades de memoria que se encuentren en la página cero antes de las direcciones correspondientes a los seudovectores de interrupción propios del modo boot-strap.

Para los mapas descritos en los párrafos anteriores es importante señalar que cuando se opere en modo TEST el nibble alto del registro CONFIG deberá estar programado con la palabra binaria 1011, en otro caso la dicha palabra deberá ser 1111, el registro CONFIG tiene asociada la dirección $103F y es una localidad EEPROM, para programarlo se puede usar el manejador PUMMA invocando la opción de programación de la EEPROM interna desde teclado, escogiendo como dirección a programar la mencionada anteriormente, para más detalles acerca del registro CONFIG se recomienda consultar el manual técnico sobre el 68HC11 editado por MOTOROLA, en lo que toca a el empleo de PUMMA para programación de la EEPROM interna puede consultarse el capítulo que describe este punto o la ayuda en línea del programa.

Mapa	Puentes en CMT SIMMP-2	Firmware Residente
EA	J3, J8 o J9, J12, J14, J16, J20 y J22	FWSP2EA1 o FWSP2EA2
EB	J3, J8 o J9, J13, J14, J16, J19 y J22	FWSP2EB1 o FWSP2EB2
EC	J3, J8 o J9, J12, J14, J16, J19 y J22	FWSP2EB1 o FWSP2EB2
TA	J3, J8 o J9, J12, J14, J16, J20 y J22	FWSP2TA1 o FWSP2TA2

Tabla 2.1 Configuración de puentes y firmware residente para los mapas de memoria de la CMT SIMMP-2.

Para cada uno de los mapas descritos anteriormente asociados con la operación en modo expandido o TEST, el firmware residente requerido es cargado en la EEPROM interna del microcontrolador, quedando en dicha memoria varias localidades libres para uso del usuario, para los mapas asociados con la operación en modo expandido las correspondientes direcciones van de la $FEA0 a la $FFBF, en lo que toca a el mapa para operación en modo TEST las direcciones de las localidades libres van de la $BEA0 a la $BFBF.

SUBMAPA DE PUERTOS.- El submapa de puertos está definido en un intervalo de 1k (de la dirección $1800 a la $1BFF) dividido en ocho subintervalos de 128 direcciones cada uno, al invocar una dirección de puerto en un subintervalo determinado se verifica en nivel bajo una de ocho líneas de habilitación de puerto, en la Figura 2.11 se muestra el submapa de puertos, para más detalles acerca de las habilitaciones y direcciones asociadas pueden consultarse en este mismo capítulo, los temas sobre el paginador de puertos y los bloques CTPP1 y CTPP2.

SUBMAPA DE PUERTOS ALTERNO.- En todos los mapas de memoria asociados con los modos TEST y expandido se aprecia la existencia de un submapa de puertos alterno, definido en un intervalo de 1k (direcciones de la $1C00 a la $1FFF), si el usuario lo requiriera podría conectar un 74LS138 externo que generaría en sus ocho salidas líneas de salida, habilitaciones de puerto asociadas cada una de ellas con sendos subintervalos de 128 direcciones cada uno, en la Figura 2.12 se muestra como hacer esto, así como también los subintervalos que verificarían cada línea de habilitación; en la práctica se ha visto que con las líneas de habilitación propias del submapa de puertos normal (direcciones de la $1800 a la 1BFF) pueden conectarse, mediante lógica de enlace, diversos puertos externos a la CMT SIMMP-2.

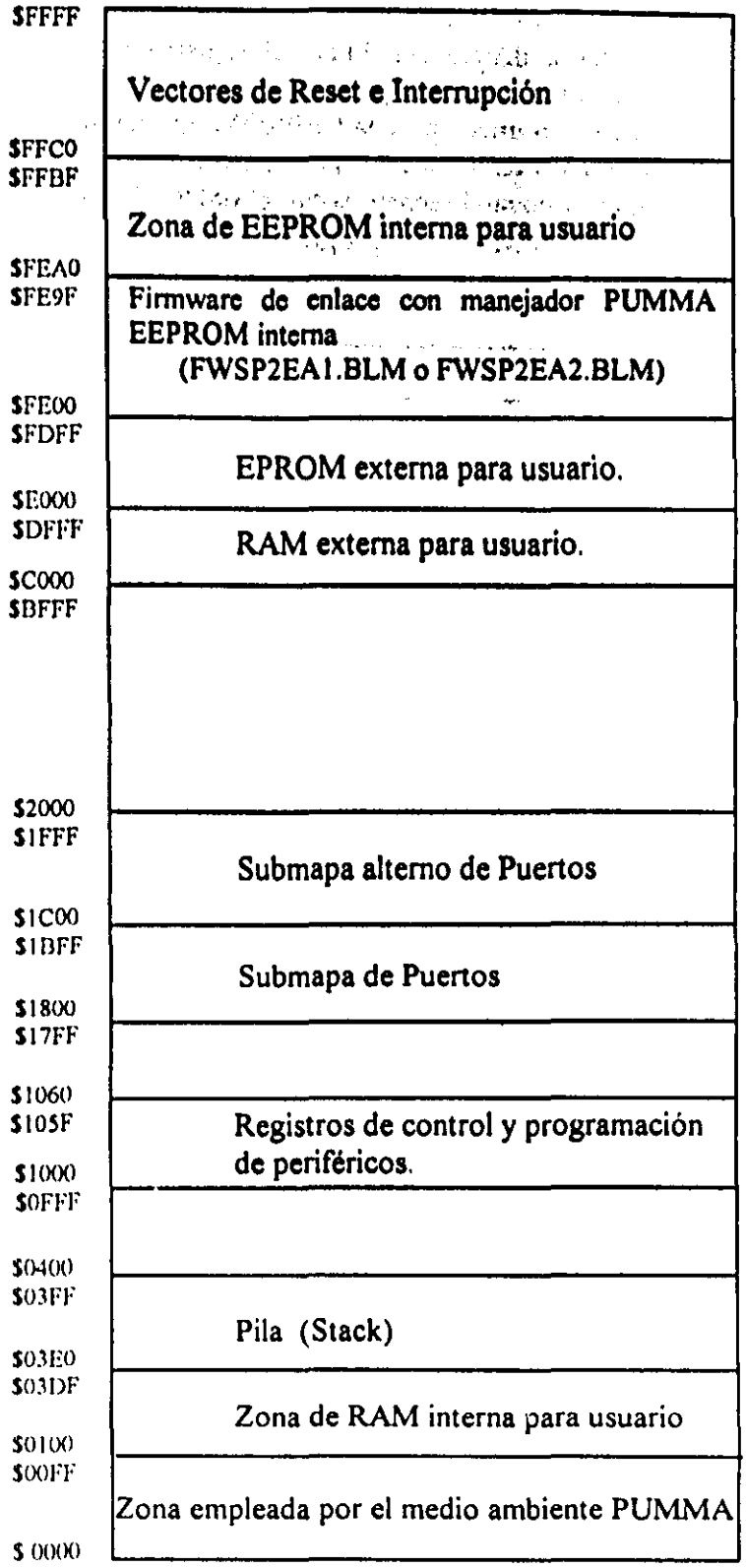


Figura 2.9A.- Mapa de memoria EA (operación en modo expandido)

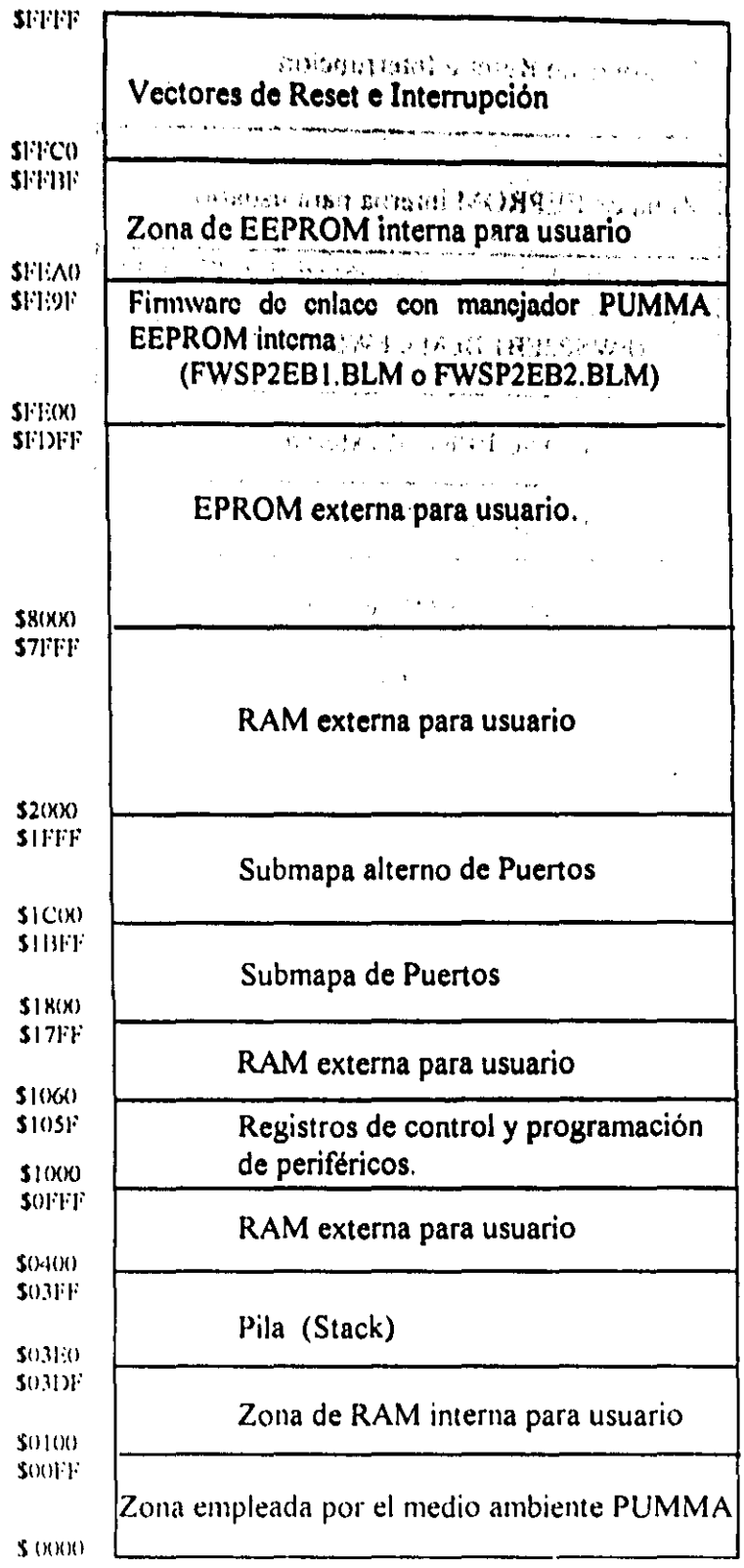


Figura 2.9B.- Mapa de memoria EB (operación en modo expandido)

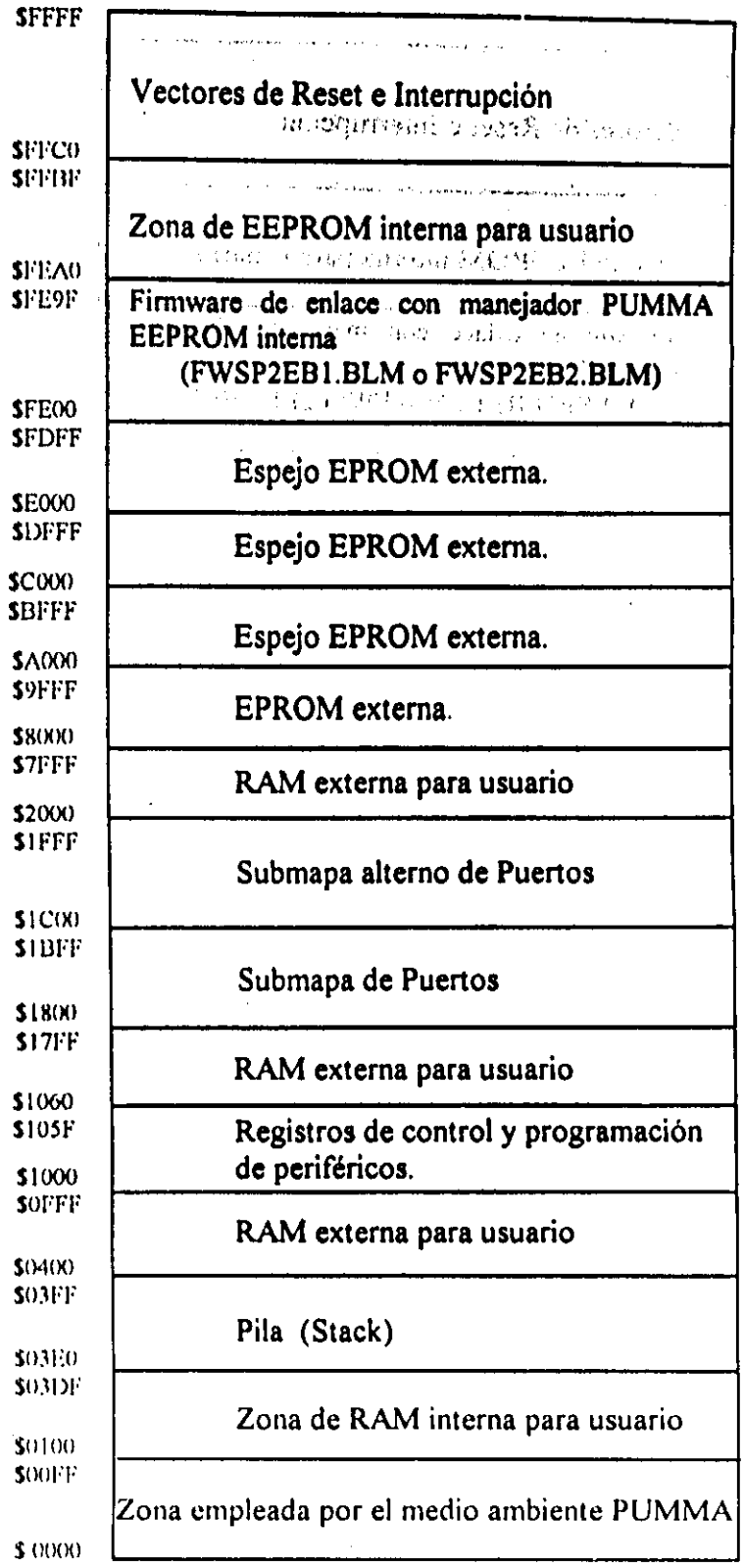


Figura 2.9C Mapa de memoria EC (operación en modo expandido)

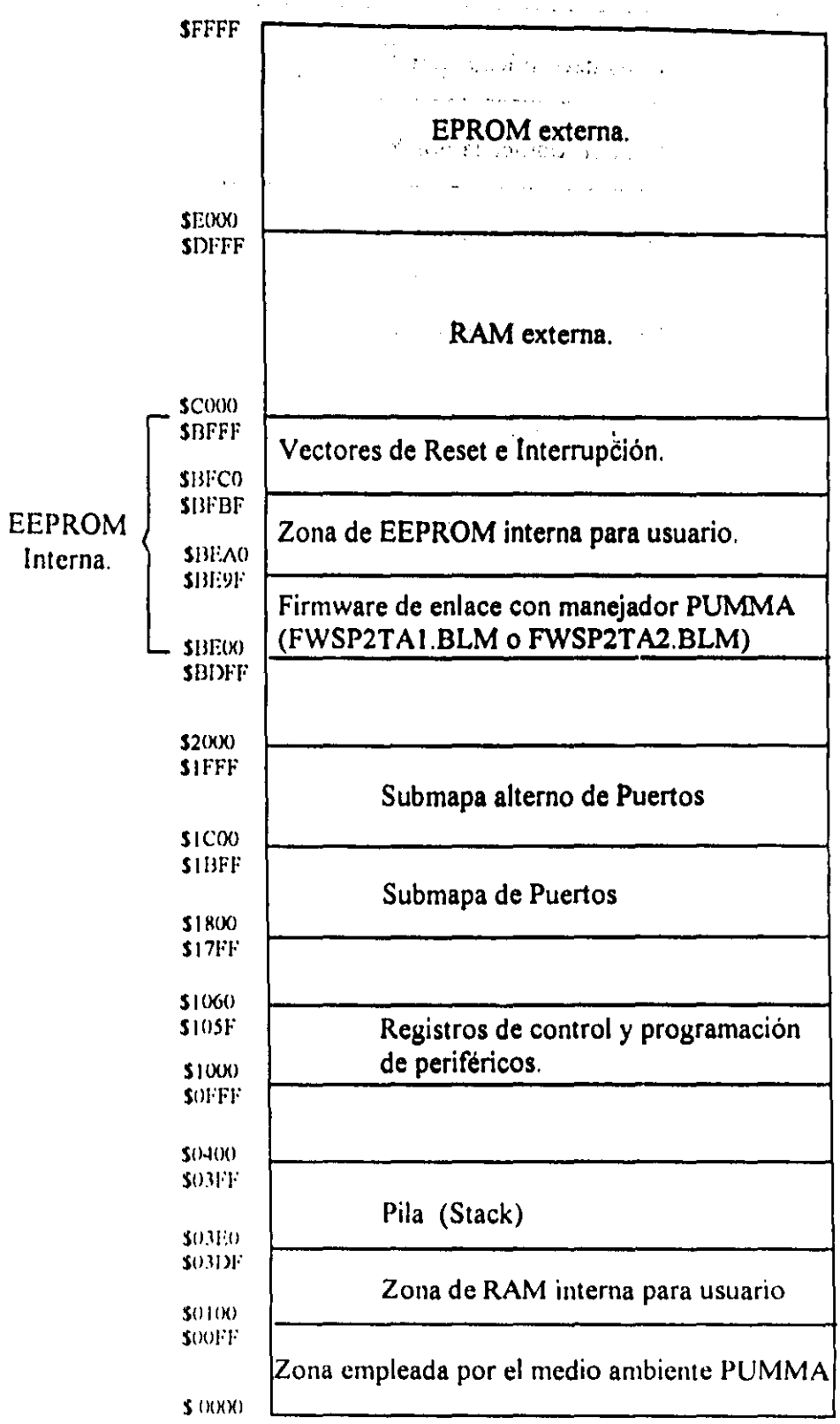


Figura 2.10.- Mapa de memoriaTA (operación en modo TEST).

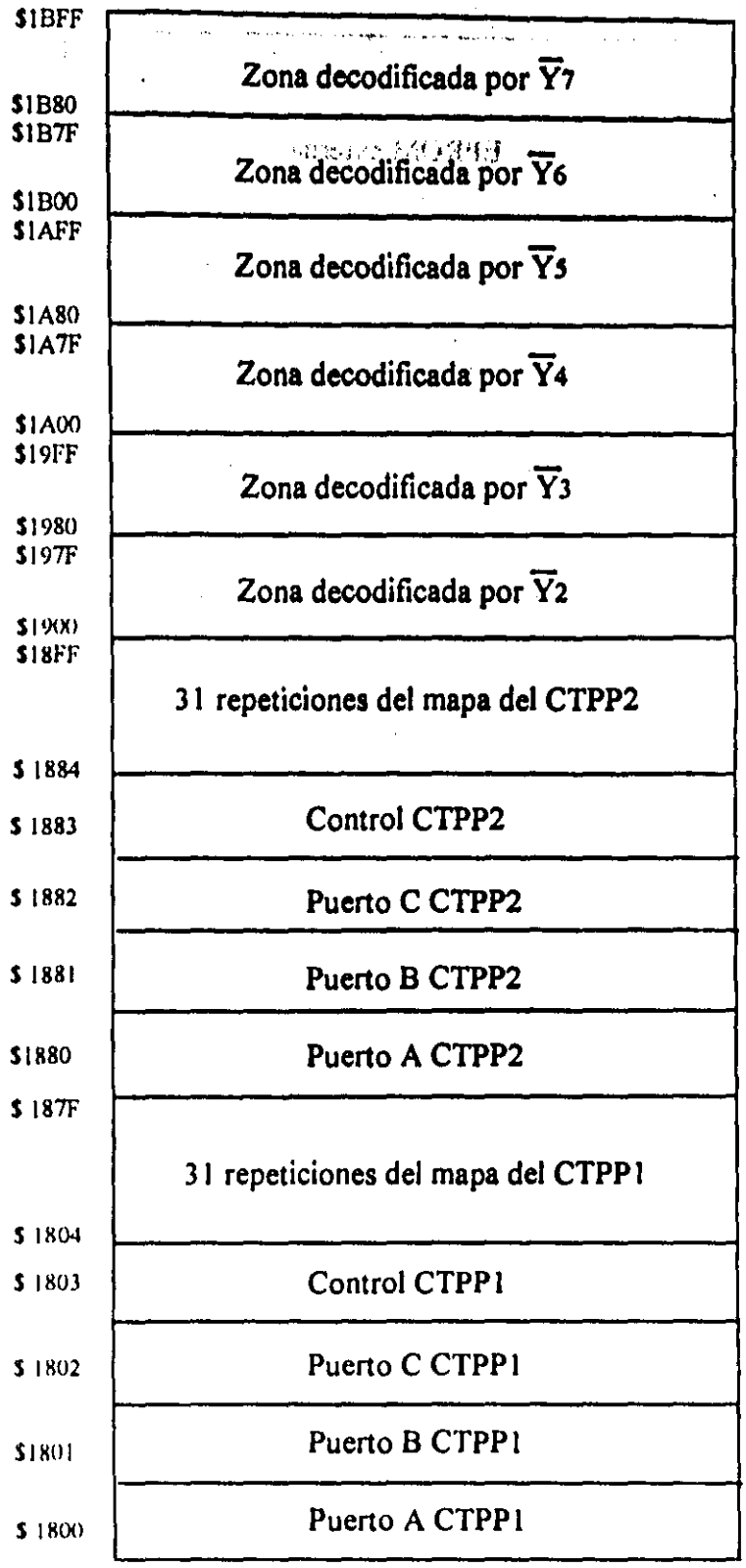


Figura 2.11.- Submapa de puertos de la CMT SIMMP-2 operando en modo expandido o TEST.

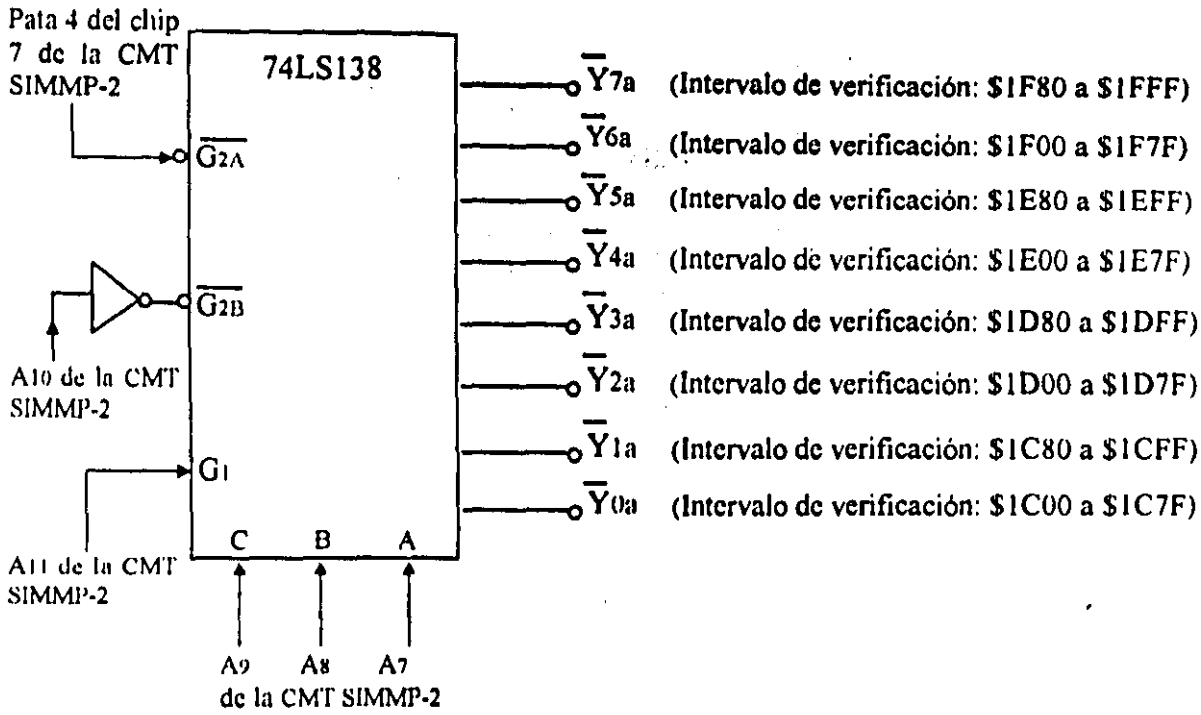


Figura 2.12.- Conexión de un decodificador 74LS138 externo para validar el submapa de puertos alterno con ocho subintervalos de verificación.

Si la tarjeta SIMMP-2 no contara con los chips que validan a los bloques CTPP1 y CTPP2 el usuario podría añadir puertos paralelos de salida cuando se opere en modo expandido o TEST, para ello se podrían emplear chips MOTOROLA que sirvieran para tal fin, tales como el PIA 6821, siempre y cuando los mismos fueran compatibles en velocidad con la CMT SIMMP-2; una manera económica de agregar puertos paralelos de entrada o salida a la arquitectura SIMMP-2 consiste en usar para tal fin a registros 74LS373 o 74LS374 a continuación se describe como hacer esto.

CONEXIÓN DE UN PUERTO DE SALIDA EXTERNO EMPLEANDO UN 74LS374.- En el caso de que la tarjeta SIMMP-2 no contara con los chips que validan a los bloques CTPP1 y CTPP2 el usuario podría añadir puertos paralelos de salida cuando se opere en modo expandido o TEST, para ello se podría emplear un registro 74LS374 que cuenta con señal de captura por flanco de subida (para más detalles consultar un manual de circuitos TTL), en la Figura 2.13 se muestra un esquema de como hacer esto, dado que la línea $\overline{Y7}$ de habilitación se verifica al invocarse direcciones que van de la $1B80 a la $1BFF, el puerto de salida mostrado en la Figura 2.13 tiene como dirección asociada a cualquiera de las contenidas en el intervalo antes mencionado; si se deseara reducir la redundancia de direcciones la señal de captura de dato para el 74LS374 se tendría que obtener de un circuito combinacional cuya entrada fuera la propia señal Y7 y algunas líneas de dirección del microcontrolador, existen desde luego, de acuerdo con la o

las direcciones deseadas para el puerto de salida en cuestión, múltiples maneras de realizar tal lógica de paginación.

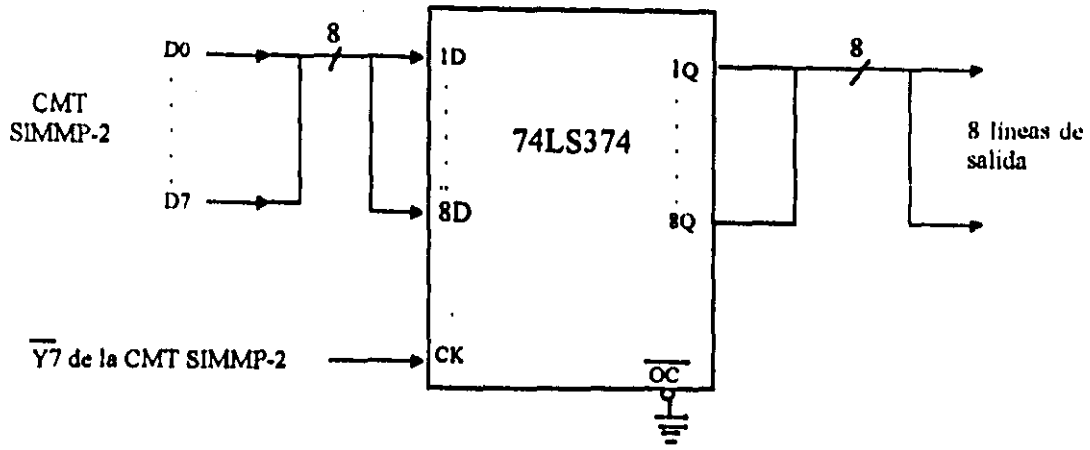


Figura 2.13.- Conexión de un puerto de salida externo a la tarjeta SIMMP-2, empleando un registro 74LS374, validado en cualquiera de las direcciones en el intervalo $1B80 a $1BFF.

CONEXIÓN DE UN PUERTO DE ENTRADA EXTERNO EMPLEANDO UN 74LS373.- En caso de que la tarjeta SIMMP-2 no contara con los chips que validan a los bloques CTPP1 y CTPP2 el usuario podría añadir puertos paralelos de entrada cuando se opere en modo expandido o TEST, para ello se podría emplear un registro 74LS373 que cuenta con señal de habilitación de tercer estado con nivel bajo de verificación (para más detalles consultar un manual de circuitos TTL), en la Figura 2.14 se muestra un esquema de como hacer esto, dado que la línea $\overline{Y6}$ de habilitación se verifica al invocarse direcciones que van de la $1B00 a la $1B7F, el puerto de salida mostrado en la Figura 2.14 tiene como dirección asociada a cualquiera de las contenidas en el intervalo antes mencionado; si se deseara reducir la redundancia de direcciones la señal de habilitación de salida para el 74LS373 se tendría que obtener de un circuito combinacional cuya entrada fuera la propia señal $\overline{Y6}$ y algunas líneas de dirección del microcontrolador, existen desde luego, de acuerdo con la o las direcciones deseadas para el puerto de salida en cuestión, múltiples maneras de realizar tal lógica de paginación.

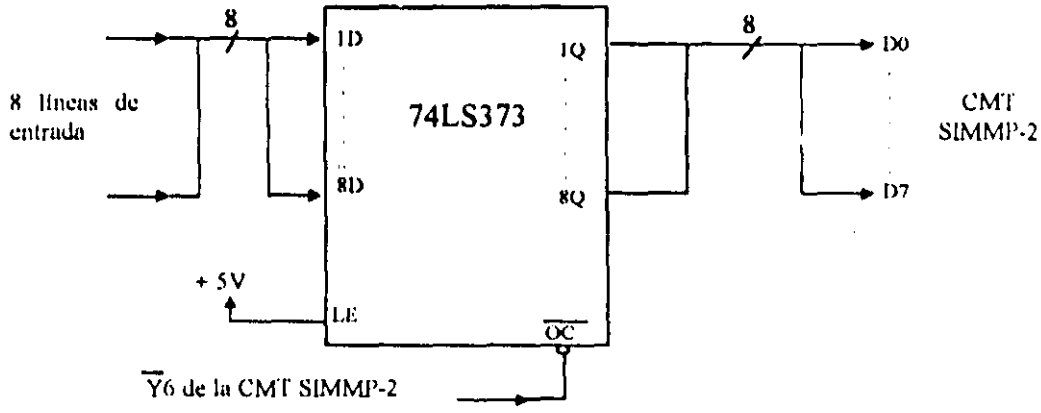


Figura 2.14.- Conexión de un puerto de entrada externo a la tarjeta SIMMP-2, empleando un registro 74LS373, validado en cualquiera de las direcciones en el intervalo $1B00 a $1B7F.

PROGRAMADOR DE MEMORIAS EPROM CONTENIDO EN LA CMT SIMMP-2

La CMT SIMMP-2 incorpora en su arquitectura a un programador de memorias EPROM que requiere para su operación que la CMT SIMMP-2 esté ligada vía serie con una computadora anfitriona que esté ejecutando el manejador hexadecimal PUMMA; la memoria que se desee programar debe ser colocada en la base que para tal fin existe en la tarjeta SIMMP-2, que deberá estar configurada para operación en modo boot-strap, las EPROM que se pueden programar son las siguientes: 2716 o 27C16, 2732 o 27C32, 2764 o 27C64, 27128 o 27C128, 27256 o 27C256 y 27512 o 27C512; cuando se programen memorias 2716, 27C16, 2732 o 27C32 se deberán insertar en la base de modo que la pata uno de la memoria coincida con la pata tres de la base y la pata 12 de la memoria coincida con la pata 14 de la base véase la Figura 2.15.

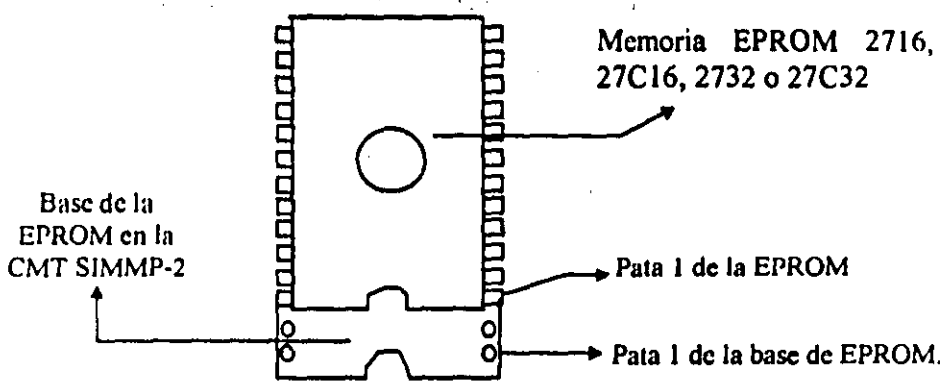


Figura 2.15.- Inserción de memorias EPROM 2716, 27C16, 2732 y 27C32 en la base respectiva de la CMT SIMMP-2.

El programador puede aceptar como origen de la información a colocar en la EPROM ya sea a una lista de bytes que el usuario introduzca desde el teclado de la computadora anfitriona o a la información contenida en un archivo de tipo BLM o LEM que son los que maneja de modo natural el manejador hexadecimal PUMMA (para más información consultar el capitulo referente al manejador PUMMA), si el usuario deseara programar un archivo S19 deberá antes transformarlo a un archivo BLM empleando para ello la opción dos del menú de manejo de disco del programa PUMMA, en caso de que el archivo a programar tenga el formato INTEL-HEX el usuario podrá pasarlo al formato S19, empleando para ello al programa HEXS19.EXE, para su posterior transformación a BLM como se ha descrito anteriormente y proceder a programarlo en la EPROM.

Para las memorias 2716, 27C16, 2764, 27C64, 27128, 27C128, 27256, y 27C256 cada byte es programado y verificado sucesivamente, en lo que toca a las memorias 2732, 27C32, 27512 y 27C512 se debe hacer la programación completa para después proceder a hacer la verificación empleando opciones del manejador PUMMA, esto se debe a características propias de las memorias mencionadas.

En caso de que la memoria sea de tipo 27C16 o 2716 PUMMA preguntará al usuario si la memoria es antigua o de tipo 27C16B ya que la mayoría de las EPROM de 2k antiguas se programan con un pulso con nivel de verificación alto, en caso de que el nivel de verificación del pulso de programación sea bajo el usuario deberá indicar a PUMMA que desea programar una

memoria 27C16B, para saber el tipo de pulso de programación de una memoria de este tipo se recomienda consultar la hoja de especificaciones técnicas de la misma.
En la actualidad el voltaje mas usual para programar memorias EPROM es 12.5 volts aunque para memorias fabricadas hace varios años tal voltaje podría ser mayor (de 21 a 24 volts), esto está contemplado en el diseño del programador de la tarjeta SIMMP-2, pudiendo ser configurado colocando o quitando puentes e indicándolo al manejador PUMMA cuando este lo requiera.
Antes de proceder a la programación de una EPROM el usuario deberá tener dispuesto lo siguiente:

1) Fuente que proporcionará el voltaje de programación de la EPROM, que deberá estar calibrada a un voltaje que exceda en aproximadamente 1.5 volts al voltaje nominal requerido.
2) Fuente de cinco volts para polarizar a la tarjeta SIMMP-2.
3) Si se trata de una EPROM 2716 o 27C16 se deberá saber de antemano el nivel de verificación del pulso de programación.
4) Si la programación ha de hacerse desde disco el archivo BLM correspondiente deberá estar ya generado.

Para cada tipo de EPROM de las manejadas por la CMT SIMMP-2 existe una configuración de puentes para que las mismas puedan ser leídas o programadas, en las Tablas 2.2 y 2.3 se muestran tales configuraciones.

Tabla 2.2.- Configuración de puentes para leer memorias EPROM en la CMT SIMMP-2.

Memoria	Puentes Colocados
27C16 2716	* J2, J9/8, J12, J14, J16, J21
27C32 2732	* J2, J9/8, J12, J14, J16, J22
27C64 2764	* J3, J9/8, J12, J14, J16, J22
27C128 27128	* J3, J9/8, J12,J14, J16, J22
27C256 27256	* J3, J9/8, J13,J14, J16, J22
27C512 27512	* J3, J9/8, J13,J15, J16, J22

* Si está colocado J8, la terminal P de la CMT SIMMP-2 debe estar desconectada, preferentemente colocar J9.

Tabla 2.3.-Configuración de puentes para programar memorias EPROM en la CMT SIMMP-2.

Memoria	Puentes Colocados
27C16 2716	J2, J8, J12, J14, J16, J21
27C32 2732	J2, J8, J12, J14, J16, J22
27C64 2764	J3, J8, J12, J14, J16, J22
27C128 27128	J3, J8, J12, J14, J16, J22
27C256 27256	J3, J8, J13, J14, J16, J22
27C512 27512	J3, J8, J13, J15, J17, J22

En la Figura 2.16 se muestra un arreglo para programar memorias EPROM empleando a la tarjeta SIMMP-2.

A continuación se describen los pasos a seguir para llevar a cabo la programación de una EPROM refiriéndose tanto al arreglo mostrado en la Figura 2.16 como a cosas propias del manejador PUMMA (consultar de ser necesario el capítulo sobre la guía de PUMMA), los pasos a seguir son los siguientes:

a) Configurar la CMT SIMMP-2 para operación en modo boot-strap (puentes J4 y J5 colocados).
b) Con la tarjeta desenergizada colocar en su base la memoria a programar.
c) Energizar con la fuente de cinco volts
d) Oprimir botón de RESET en SIMMP-2.
e) Ejecutar en la computadora anfitriona el manejador PUMMA.

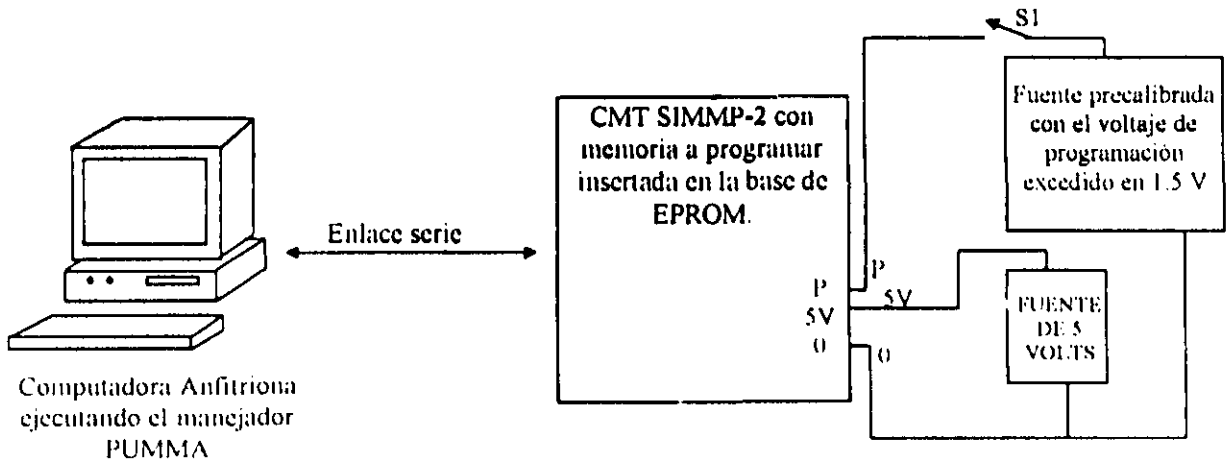


Figura 2.16.- Arreglo para programar memorias EPROM empleando a la tarjeta SIMMP-2.

f) Una vez que PUMMA esté en su menú principal y el ambiente PUMMA esté ejecutándose en la tarjeta SIMMP-2 pasar a el menú de manejo de memoria.

g) Una vez en el menú de manejo de memoria pasar al menú de programación de la EPROM externa. Al entrar a esta opción PUMMA presentará un menú de cuatro opciones siendo estas las siguientes:
   1) Verificar que la EPROM esté completamente borrada.
   2) Pasar a programar la EPROM.
   3) Pasar a verificar lo programado en la EPROM.
   4) Pasar a leer la EPROM.

h) Verificar, antes de proceder a la programación, que la memoria esté completamente borrada de no ser así proceder a borrarla y repetir los pasos anteriores.

i) Invocar la opción (2) del menú de programación.

j) Indicar a PUMMA el origen de la información a programar (bytes introducidos desde el teclado de la PC o un archivo de disco), en caso de que la información a programar sean bytes escritos desde el teclado PUMMA procederá a pedirlos sucesivamente, en otro caso PUMMA pedirá el nombre del archivo BLM o LEM a programar.

l) Indicar a PUMMA la dirección incial de programación de los datos desde el punto de vista de la EPROM y no de su posible localización en un mapa de memoria de un sistema.

m) Indicar a PUMMA el tipo de memoria a programar tecleando los dígitos finales del número de parte de la EPROM, por ejemplo, si se va a programar una memoria 27128 el usuario deberá teclear 128 seguido de la opresión de la tecla return.

n) Indicar a PUMMA el tipo de voltaje de programación y verificar la correcta colocación de los puentes J24 y J25. En caso de que el voltaje de programación sea 12.5 volts J24 y J25 deberán estar colocados, en otro caso deberá estar puesto solamente el puente J25.

ñ) Cerrar el interruptor S1 de la Figura 2.15 y después oprimir cualquier tecla, en seguida a lo anterior PUMMA desplegara un letrero que dice: UN MOMENTO POR FAVOR ESTOY PROGRAMANDO. Después de terminar la programación PUMMA indicará el número de errores al programar debiendo éste ser cero en caso de que la programación haya sido totalmente exitosa, si la memoria que se programó es la 2732, 27C32, 27512 o 27C512 la verificación de la programación debe hacerse a posteriori, empleando para ello la opción tres del menú de programación de la EPROM.

o) Desconectar, abriendo el interruptor S1, el voltaje de programación cuando PUMMA lo indique.

Si el usuario deseara verificar el contenido de una EPROM contra información proporcionada desde teclado o bien contenida en un archivo BLM o LEM, al presentar PUMMA las opciones del menú de programación deberá optar por la opción (3), para seguir los pasos que PUMMA le indique, para más detalles sobre esto se puede consultar el capítulo que trata sobre el manejador PUMMA o la ayuda en línea que contiene el propio programa.

RESPUESTAS AL RESTABLECIMIENTO (RESET) DE LA CMT SIMMP-2

Las distintas formas de respuesta al restablecimiento (RESET) de la CMT SIMMP-2 dependen de características propias del sistema de RESET del 68HC11, del modo de operación del microcontrolador y de colocaciones de puentes en la tarjeta.

Existen para el 68HC11 cuatro posibles maneras en que un restablecimiento (RESET) puede ser invocado y estas son:

a) RESET invocado al detectarse el flanco de subida de la fuente de cinco volts que polariza la tarjeta (POR). Las direcciones de memoria que deben contener al vector correspondiente son la $FFFE y la $FFFF para los modos normales (single-chip y boot-strap), para los modos especiales (boot-strap y TEST) tales direcciones son la $BFFE y la $BFFF.

Si la fuente de alimentación que polariza la tarjeta tuviera un tiempo de levantamiento lento, pudiera llegar a suceder que al energizar la tarjeta no se llevara a cabo correctamente el restablecimiento, originando esto un comportamiento errático del microcontrolador haciéndose necesario el oprimir el botón de RESET para lograr un restablecimiento correcto; para evitar este tipo de problema sobre todo cuando la tarjeta SIMMP-2 vaya a ser parte de un equipo de instrumentación, se podría conectar a el poste de la tarjeta ligado con la terminal de RESET del microcontrolador el circuito de autorrestablecimiento mostrado en la Figura 2.17, para un mejor desempeño del mismo se sugiere desconectar el capacitor C7 de la tarjeta, el circuito mostrado sería exógeno a la tarjeta y parte del hardware del sistema que se estuviera basando en la tarjeta.

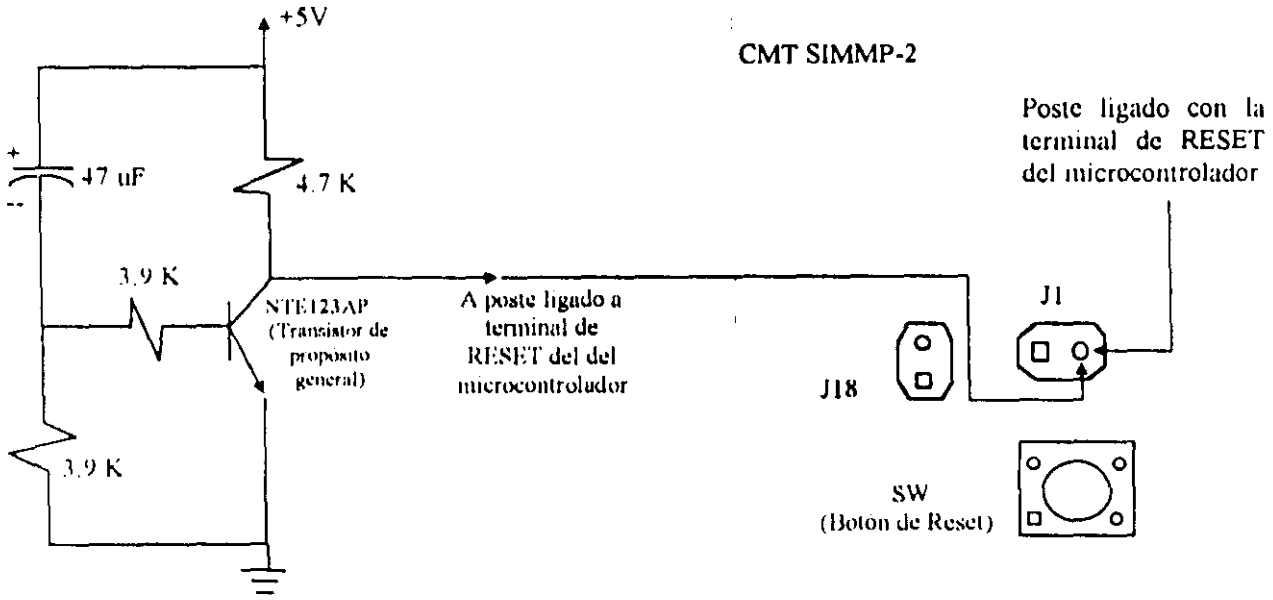


Figura 2.17 - Circuito de autorrestablecimiento que asegura un restablecimiento correcto de la CMT SIMMP-2 aun cuando el tiempo de levantamiento de la fuente de polarización fuera lento.

b) RESET invocado al detectarse una transición de bajo a alto en el pin marcado como RESET en el microcontrolador. Las direcciones de memoria que deben contener al vector correspondiente son la $FFFE y la $FFFF para los modos normales (single-chip y boot-strap), para los modos especiales (boot-strap y TEST) tales dirccciones son la $BFFE y la $BFFF.
c) RESET invocado al detectarse una falla en el funcionamiento de los circuitos de reloj. Las direcciones de memoria que deben contener al vector correspondiente son la $FFFC y la $FFFD para los modos normales (single-chip y boot-strap), para los modos especiales (boot-strap y TEST) tales direcciones son la $BFFC y la $BFFD.
d) RESET invocado al detectar el supervisor de operación correcta (watch-dog) una incorrecta secuencia de ejecución de instrucciones. Las direcciones de memoria que deben contener al vector correspondiente son la $FFFA y la $FFFB para los modos normales (single-chip y boot-strap), para los modos especiales (boot-strap y TEST) tales direcciones son la $BFFA y la $BFFB.

Para una información detallada acerca del funcionamiento de los distintos tipos de RESET mencionados en párrafos anteriores, se sugiere consultar las referencias dos y tres del capítulo que describe el funcionamiento del manejador PUMMA.

Es conveniente señalar aquí, que en caso de que el microcontrolador opere en un modo que no sea boot-strap, para el firmware residente en la CMT SIMMP-2 los tres vectores mencionados anteriormente apuntan a el origen de la EEPROM si tal firmware está definido por archivos BLM cuyo nombre sea terminado por el digito uno, en caso de que el nombre del archivo definitorio termine en el digito dos, el vector correspondiente a los RESET de tipo (a) y (b) apunta a el origen de la EEPROM interna y los otros dos (c) y (d) apuntan a direcciones de la página cero similares a las asociadas con tales vectores cuando el microcontrolador opere en modo boot-strap; de ser necesario el usuario podrá modificar esos vectores empleando el manejador PUMMA.

A continuación se describen las distintas respuestas al restablecimiento que puede presentar la tarjeta SIMMP-2, cuando el microcontrolador de la misma opera en cada uno de los cuatro modos posibles.

Respuesta al restablecimiento operando en modo boot-strap. En este caso al darse el RESET el microcontrolador salta a ejecutar un programa denominado por MOTOROLA como boot-loader, tal programa lleva a cabo las siguientes tareas:

1) Carga el apuntador de pila (stack pointer) con una dirección que apunta al tope superior de la RAM interna.
2) Inicializa el puerto serie interno (SCI) a un baudaje de 7812 bps no paridad y un bit de stop.
3) Genera una transición de uno a cero en la terminal transmisora del puerto serie (TXD).
4) Pasa a esperar la recepción de un byte enviado por una computadora anfitriona ligada via serie al sistema 68HC11, en caso de que tal byte sea $00 se producirá un salto al origen de la memoria EEPROM interna.
5) Si ese primer byte recibido es diferente de $00 y de $FF el baudaje es cambiado a 1200 bps, si el mismo es igual a $FF no se efectúa el cambio mencionado.
6) Los siguientes bytes recibidos son cargados sucesivamente en la RAM interna a partir de la dirección $0000, al detectarse el fin de la transmisión por parte de la computadora anfitriona que esté enviando los bytes al sistema 68HC11, se producirá un salto al origen de la RAM interna autoejecutándose el programa recién cargado en la RAM interna, desde luego que la cadena de bytes recibida debe representar código coherente ya que de no ser asi el comportamiento del sistema será impredecible, debiendo ser necesario restablecerlo.

Existen para las distintas versiones del 68HC11 algunas pequeñas variantes en la lógica de ejecución del programa boot-loader, la descrita anteriormente se apega a el caso del 68HC11F1; para más información acerca del programa boot-loader pueden consultarse las referencias (1) y (3) del capítulo sobre el manejador PUMMA.

El programa boot-loader está contenido en un ROM de 256 bytes (mapeado de la dirección $BF00 a la $BFFF) que contiene además los correspondientes vectores de RESET e interrupción propios del modo boot-strap, tal ROM es programado de fabrica y es visible en el mapa de memoria únicamente al operar en el modo boot-strap.

De lo explicado en los párrafos anteriores se deduce que al restablecer la CMT SIMMP-2 cuando el microcontrolador de la misma opere en modo boot-strap, se tendrán las siguientes dos posibilidades:

a) Se pasará a recibir vía serie un programa que será cargado en la RAM interna a partir de su origen, para ser autoejecutado de inmediato una vez que se detecte el fin de la transmisión.
b) Si se conecta un puente entre la terminal de transmisión y recepción de la CMT SIMMP-2, obviamente sin una computadora anfitriona conectada, se producirá un salto al origen de la EEPROM interna, lográndose con esto la autoejecución autónoma de un programa previamente cargado en tal memoria; el puente mencionado podría ser colocado entre los postes dos y tres de la terminal serie de la CMT SIMMP-2, véase la Figura 2.2.

Si la CMT SIMMP-2 opera en el modo boot-strap y el ambiente PUMMA está ejecutándose en la misma, al restablecerla se perderá el mismo por lo que deberá ser necesario reinstalarlo empleando para ello la opción ocho del menú principal del manejador PUMMA.

Respuesta al restablecimiento operando en modo single-chip.- Cuando el microcontrolador de la tarjeta SIMMP-2 opera en modo single-chip, existen dos posibles respuestas al restablecimiento, determinadas por el puente J11; el firmware correspondiente que deberá estar residente en la CMT SIMMP-2 puede ser el contenido ya sea en el archivo FWSP2SA1.BLM o en el FWSP2SA2.BLM; los dos accionamientos posibles son los siguientes:

a) Generación de un salto a ejecución de código compatible con el manejador PUMMA que se estuviera ejecutando en una computadora anfitriona ligada vía serie con la tarjeta, en caso que se detectara que el ambiente PUMMA está residente en la página cero de RAM se producirá un salto inmediato al mismo, en otro caso se pasara a código que puede recibirlo y autoejecutarlo (para más información sobre esto puede consultarse el capítulo referente a el manejador PUMMA). Para que se produzca este accionamiento se requerirá que el puente J11 no esté colocado.
b) Generación de un salto a la dirección $FEA0, donde el usuario previamente debería haber cargado un programa mediante la opción de programación de la EEPROM interna que es parte del manejador PUMMA; la dirección tope de ese programa no deberá exceder a $FFBF. Para este accionamiento se requiere que el puente J11 esté colocado.

Respuesta al restablecimiento operando en modo expandido.-Cuando el microcontrolador de la tarjeta SIMMP-2 opera en modo expandido, existen dos posibles respuestas al restablecimiento, determinadas por el puente J11; el firmware correspondiente que deberá estar residente en la CMT SIMMP-2 debe ser el adecuado a el mapa de memoria que se esté usando en un momento dado (véase la información referente a los mapas EA, EB y EC), los dos accionamientos posibles son los siguientes:

a) Generación de un salto a ejecución de código compatible con el manejador PUMMA que se estuviera ejecutando en una computadora anfitriona ligada vía serie con la tarjeta, en caso que

se detectara que el ambiente PUMMA está residente en la página cero de RAM se producirá un salto inmediato al mismo, en otro caso se pasará a código que puede recibirlo y autoejecutarlo (para más información sobre esto puede consultarse el capítulo referente a el manejador PUMMA). Para que se produzca este accionamiento se requerirá que el puente J11 no esté colocado.

b) Generación de un salto a la dirección origen de la EPROM, que dependerá del mapa escogido, donde el usuario previamente debería haber cargado un programa mediante la opción de programación de la EPROM externa, que es parte del manejador PUMMA. Para este accionamiento se requiere que el puente J11 esté colocado.

Respuesta al restablecimiento operando en modo TEST.-Cuando el microcontrolador de la tarjeta SIMMP-2 opera en modo TEST, existen dos posibles respuestas al restablecimiento, determinadas por el puente J11; el firmware correspondiente que deberá estar residente en la CMT SIMMP-2 puede ser el contenido ya sea en el archivo FWSP2TA1.BLM o en el FWSP2TA2.BLM; los dos accionamientos posibles son los siguientes:

a) Generación de un salto a ejecución de código compatible con el manejador PUMMA que se estuviera ejecutando en una computadora anfitriona ligada vía serie con la tarjeta, en caso que se detectara que el ambiente PUMMA está residente en la página cero de RAM se producirá un salto inmediato al mismo, en otro caso se pasará a código que puede recibirlo y autoejecutarlo (para más información sobre esto puede consultarse el capítulo referente a el manejador PUMMA). Para que se produzca este accionamiento se requerirá que el puente J11 no esté colocado.

b) Generación de un salto a la dirección origen de la EPROM externa, donde el usuario previamente debería haber cargado un programa mediante la opción de programación de la EPROM que es parte del manejador PUMMA; para este accionamiento se requiere que el puente J11 esté colocado.

En la Tabla 2.4 se resumen los accionamientos al restablecer la CMT SIMMP-2.

Ejemplo de uso del manejador PUMMA para cargar y ejecutar un programa en lenguaje de máquina en la CMT SIMMP-2.

Para el microcontrolador un programa a ejecutarse en el mismo no es sino una lista de palabras binarias de ocho bits, donde cada instrucción elemental estará codificada en uno o varios bytes, para que el programa se ejecute se requiere que la lista de bytes que lo conforma sea cargada en memoria a partir de una determinada dirección, para luego transferir el valor del contador de programa (PC) a la dirección donde haya quedado cargado el código de la primera instrucción; lo anterior puede hacerse empleando el manejador PUMMA, a continuación se ilustra esto por medio de un programa ejemplo sencillo, para una mejor comprensión de este ejemplo se recomienda consultar paralelamente lo referente a las opciones uno, seis y siete del menú principal del manejador PUMMA en el capítulo referente a esto o en la ayuda en línea del manejador.

Modo de Operación	Puente colocado entre las terminales Tx y Rx de SIMMP-2 (ver Figura 2.2)	Puente no colocado entre las terminales Tx y Rx de la SIMMP-2 (ver Figura 2.2)	Puente J11 colocado	Puente J11 no colocado
Boot-Strap	Se genera un salto al origen de la EEPROM interna	Se pasa a firmware de recepción de un programa a colocarse en la página cero.	------	------
Single Chip	Posibilidad no permitida.	----------	Se genera un salto a dirección $FEA0 dentro de la EEPROM interna.	Se pasa a ejecutar firmware compatible con el manejador PUMMA.
Test	Posibilidad no permitida.	----------	Se genera salto al origen de la EPROM externa.	Se pasa a ejecutar firmware compatible con el manejador PUMMA.
Expandido	Posibilidad no permitida.	----------	Se genera salto al origen de la EPROM externa.	Se pasa a ejecutar firmware compatible con el manejador PUMMA.

Tabla 2.4 .- Accionamientos al restablecer la CMT SIMMP-2.

El programa ejemplo, que se denominara como PROGE1 y supuestamente ha de ser cargado a partir de la dirección $0100, es el siguiente:

Programa PROGE1

```
DIREC.  CODIGO  ETIQ      INSTRUC  OPER.   COMENTARIOS
0100    8608              LDAA     #$08    ;Carga de A con$08
0102    8D06    VUELTA:   BSR      RET     ;Invoca retardo de .25 s
0104    4A                DECA             ;Decrementa A
0105    26FB              BNE      VUELTA  ;Saltar si A<>0
0107    7E0000            JMP      $0000   ;Salta a ambiente PUMMA
010A    3C      RET:      PSHX             ;Registro X a pila
010B    CEFFFF            LDX      #$FFFF  ;Carga X con $FFFF
010E    01      REP:      NOP
010F    09                DEX              ;Decrementa X
0110    26FC              BNE      REP     ;Salta a REP si X<>0
0112    38                PULX             ;Reinstala X de pila
0113    39                RTS              ;Retorna de subrutina
```

El programa anterior es una espera de un poco más de dos segundos después de la cual se retorna de inmediato al medio ambiente PUMMA, al ejecutarse este código en la CMT SIMMP-2 se

deberá observar que el LED testigo del ambiente PUMMA, deja de centellear por un tiempo un poco mayor a dos segundos; el programa para el microcontrolador es una lista de bytes que para este ejemplo sería la siguiente:

DIR	CÓDIGO HEX
0100	86
0101	08
0102	8D
0103	06
0104	4A
0105	26
0106	FB
0107	7E
0108	00
0109	00
010A	3C
010B	CE
010C	FF
010D	FF
010E	01
011F	09
0110	26
0111	FC
0112	38
0113	39

La lista anterior es el programa del ejemplo en lenguaje de máquina (código), para cargarlo en la CMT SIMMP-2 el mismo debe ser primero validado en la computadora anfitriona empleando la opción uno del menú principal de PUMMA, después de esto para ejecutarlo en la CMT SIMMP-2 se emplearía la opción seis del mismo menú. Si se desea guardar en disco este programa se puede invocar la opción seis del menú de manejo de disco de PUMMA.

Ejemplo de uso del manejador PUMMA, para cargar y ejecutar en la CMT SIMMP-2, un programa cuyo código haya sido generado por un ensamblador o compilador.

Generalmente en las aplicaciones prácticas, el código requerido es generado a partir de un programa fuente que podría ser una secuencia de instrucciones indicadas mediante cadenas de caracteres, que recuerden la función efectuada por la instrucción, seguidos estos por caracteres que indiquen, si es el caso, la manera como se va a accesar uno de los operandos que requiriera la operación a efectuar, antes de los caracteres que denotan la instrucción podría haber caracteres que etiquetaran la posición de la instrucción para poder hacer transferencias a esa posición desde otras partes del programa si esto fuera necesario; cuando la secuencia de instrucciones que contiene un programa es expresado de la manera descrita anteriormente, el mismo está escrito en lenguaje ensamblador; de acuerdo a lo expresado, cada instrucción del programa es declarada usando un sólo renglón, distinguiéndose en el mismo cuatro campos a saber:

a) Campo de etiqueta. En este campo se coloca una cadena de caracteres que normalmente se usa para marcar la posición que tiene una determinada instrucción dentro del programa,

normalmente el primer caracter de la etiqueta deberá estar colocado en el primera columna de la página de texto donde se estén escribiendo las instrucciones.

b) Campo de instrucción. En este campo se coloca una cadena de caracteres que recuerdan lo que efectúa la instrucción.

c) Campo de operandos. Aquí se colocan, en caso de ser esto necesario, una cadena de caracteres que denotan en alguna forma la manera en que uno de los operandos que emplee la instrucción es accesado.

d) Campo de comentarios. Este campo se inicia siempre con el caracter (;), todo lo que se coloque a la derecha de este caracter no es tomado en cuenta por el programa que traduce el programa fuente a código de máquina del microcontrolador; normalmente este campo es empleado para poner comentarios que pudieran servir de orientación para el programador o a otra persona en la depuración o mejora del propio programa.

Además de lo anterior, el programa fuente en ensamblador contendrá lo que se denomina como directivas que no son propiamente instrucciones del propio microcontrolador o microprocesador en cuestión, sino, indicaciones entre otras como la asignación de valores numéricos además de caracteres o indicaciones de a partir de que dirección de memoria ha de generarse el código de todo el programa o de un tramo del mismo, para mayor información sobre ensambladores para el 68HC11 se puede consultar la referencia uno de este capítulo.

De acuerdo con lo comentado anteriormente la traducción a código de máquina de un programa fuente la hace un programa que se denomina programa ensamblador o simplemente ensamblador, si el mismo es ejecutado en una computadora cuya unidad central de proceso no sea la misma que la correspondiente al procesador del microcontrolador o microprocesador cuyo código de máquina ha de generarse, el mismo es denominado como ensamblador cruzado (cross assembler), tal es el caso de los ensambladores para el 68HC11 que corren en una computadora de tipo PC; existen en el mercado diversos ensambladores para este microcontrolador, habiendo variantes de uno a otro, sin embargo, todos ellos presentan como mínimo las siguientes facilidades:

a) Generación de un archivo con extensión LST, en donde para cada renglón del mismo se coloca del lado izquierdo la dirección a partir de la cual ha de cargarse el código correspondiente a la instrucción seguido por el propio código en hexadecimal, colocándose después de esto la propia instrucción con sus cuatro campos. Por lo general el archivo contendrá también información referente a las direcciones de memoria correspondientes a las distintas etiquetas que se usaran en el programa.

b) Generación de un archivo S19, este tipo de archivo es un estandard en la presentación del código de máquina a ejecutarse en el 68HC11, el mismo consiste de una secuencia de renglones cuyo primer caracter es siempre la letra S seguida por un caracter que podría ser el guarismo uno o nueve, en este último caso esto testifica al último renglón del archivo, después de estos dos caracteres aparece en el renglón una cadena de bytes expresados en notación hexadecimal empleándose dos caracteres ASCII por cada byte, la información contenida en la cadena mencionada es la siguiente:

1) Los primeros dos caracteres denotan un byte que representa el numero (N) de bytes de información que hay en el renglón sin contar a este indicador.

2) Los siguientes cuatro caracteres, denotan dos bytes que representan la dirección a partir de la cual, han de cargarse los bytes que sigan a estos dos mencionados aquí

3) Los siguientes 2(N-3) caracteres, representan a los N-3 bytes que han de cargarse a partir de la dirección especificada en el párrafo anterior.

4) El último par de caracteres representan a un byte verificador, que es el complemento a unos del byte menos significativo de la suma de todos los bytes del renglón.

A continuación se indican los pasos a seguir para ejecutar en la CMT SIMMP-2 un programa escrito originalmente en lenguaje ensamblador, para esta explicación se usará como ejemplo el mismo programa que se empleo anteriormente para explicar la carga y ejecución de un programa en lenguaje de máquina.

Paso 1) Se escribe en un editor de texto el programa fuente, frecuentemente se usa la extensión ASM para el archivo ASCII correpondiente, aunque se podría usar otra diferente.

Para el programa PROG1 se tendría que generar un archivo PROG1.ASM cuyo texto podría ser el siguiente:

```
        ORG   $0100     ;Directiva que denota dirección inicial del programa.
        LDAA  #$08      ;Carga de A con$08
VUELTA: BSR   RET       ;Invoca retardo de .25 s
        DECA            ;Decrementa A
        BNE   VUELTA    ;Saltar si A<>0
        JMP   $0000     ;Salta a ambiente PUMMA
RET:    PSHX            ;Registro X a pila
        LDX   #$FFFF    ;Carga X con $FFFF
REP:    NOP
        DEX             ;Decrementa X
        BNE   REP       ;Salta a REP si X<>0
        PULX            ;Reinstala X de pila
        RTS             ;Retorna de subrutina
```

Paso 2) Se invoca el ensamblador cruzado a emplear, indicándole al mismo si se desea o no generar los archivos LST y S19 correspondientes. Por ejemplo, si se usara el ensamblador cruzado AS11NEW.EXE distribuido por MOTOROLA y que puede ser bajado de la red (freeware), para ensamblar el programa PROG1.ASM con generación de los correspondientes archivos LST y S19, el usuario deberá teclear después del requerimiento de comando del DOS (promt) lo siguiente:

```
AS11NEW PROG1.ASM -L > PROG1.LST
```

El archivo PROG1.LST tendría el siguiente aspecto:

```
0100   8608          LDAA  #$08      ;Carga de A con$08
0102   8D06  VUELTA:  BSR   RET       ;Invoca retardo de .25 s
0104   4A            DECA            ;Decrementa A
0105   26FB          BNE   VUELTA    ;Saltar si A<>0
0107   7E0000        JMP   $0000     ;Salta a ambiente PUMMA
010A   3C    RET:    PSHX            ;Registro X a pila
010B   CEFFFF        LDX   #$FFFF    ;Carga X con $FFFF
010E   01    REP:    NOP
010F   09            DEX             ;Decrementa X
0110   26FC          BNE   REP       ;Salta a REP si X<>0
0112   38            PULX            ;Reinstala X de pila
0113   39            RTS             ;Retorna de subrutina
```

En caso de que el ensamblador detectara algún error, indicaría esto en el archivo LST para que el usuario al revisarlo vea en que linea o líneas del programa se presentó el problema y de esta manera poder hacer las correcciones pertinentes en el archivo ASM correspondiente, para más detalles acerca del funcionamiento de este ensamblador se recomienda consultar su archivo de documentación que puede ser también bajado de la red; el archivo PROG1.S19 correspondiente a este ejemplo tendría el siguiente aspecto:

```
S113010086088D064A26FB7E00003CCEFFFF0109CF
S107011026FC383954
S9030000FC
```

Para ejecutar PROG1 en la CMT SIMMP-2 se deben seguir los siguientes pasos:

1) Inicializar el sistema A-D para desarrollo, con la computadora PC ejecutando el manejador PUMMA y enlazada vía serie con la CMT SIMMP-2 (para más detalles acerca de esto, puede consultarse el capítulo que habla de la guía de PUMMA).
2) Una vez en el menú principal de PUMMA pasar al menú de manejo de disco.
3) Desde el menú de manejo de disco invocar la opción de lectura de un archivo S19, proporcionando aquí el nombre correspondiente sin extensión, para el caso de este ejemplo el usuario deberá teclear PROG1 seguido de la opresión de la tecla <RETURN>.
4) En seguida a lo anterior, PUMMA pide el nombre de el archivo BLM que va a contener el programa en forma de lista (lenguaje de máquina), si el usuario teclea aquí <RETURN> sin escribir ningún nombre el archivo BLM a generar tendrá el mismo nombre que el correspondiente S19, (PROG1.BLM).
5) Después de un tiempo PUMMA desplegará en pantalla los intervalos de direcciones que conforman los distintos bloques que conforman el programa, indicando al final de esto la dirección correspondiente al récord S9, aquí el usuario deberá oprimir cualquier tecla para que PUMMA retorne al menú de manejo de disco.
6) Retornar al menú principal de PUMMA y una vez aquí invocar la opción seis del mismo para que la lista hexadecimal que corresponde al programa sea bajada, colocada en memoria del microcontrolador y ejecutada; al hacer esto el usuario deberá observar el funcionamiento del programa como se describió anteriormente al explicar la metodología para cargar y ejecutar el mismo programa en lenguaje de máquina.

En caso de que el programa fuente haya sido originalmente escrito en lenguaje C, el usuario deberá emplear un compilador cruzado que le genere el correspondiente archivo S19, para la ejecución del programa en la CMT SIMMP-2 podrían seguirse los seis pasos descritos anteriormente para el caso de que el programa fuente haya sido escrito en lenguaje ensamblador; sin embargo, dado que el código que genera un compilador puede ser de un tamaño considerablemente mayor al que se generaría para el caso de un ensamblador, los pasos recomendables a seguir para bajar y ejecutar el programa serían los siguientes:

1) Inicializar el sistema A-D para desarrollo, con la computadora PC ejecutando el manejador PUMMA y enlazada vía serie con la CMT SIMMP-2 (para más detalles acerca de esto, puede consultarse el capítulo que habla de la guía de PUMMA).
2) Una vez en el menú principal de PUMMA pasar al menú de manejo de disco
3) Desde el menú de manejo de disco invocar la opción de carga en memoria RAM de un archivo S19, una vez que PUMMA haya hecho esta tarea, pasar al menú principal del manejador para invocar el menú de manejo de memoria.

4) Desde el menú de manejo de memoria invocar la opción de ejecución a partir de una dirección de memoria, aquí el usuario deberá especificar la dirección inicial del código objeto del programa (esta información de alguna manera la debe proporcionar el compilador), una vez que PUMMA baja a la CMT SIMMP-2 la correspondiente orden el programa deberá iniciar su ejecución.

Cabe señalar aquí que dado el tamaño del código que genera un compilador, es recomendable que la CMT SIMMP-2 opere en modo expandido, además algunos compiladores suponen que los vectores de interrupción son los mismos que los correspondientes al modo boot-strap, por lo que el usuario deberá asegurarse de que el firmware correspondiente así lo consigne, (véase la información referente a los distintos mapas de memoria con los que puede operar la CMT SIMMP-2 en este mismo capítulo); desde luego que deberá haber memoria RAM física en las direcciones que así lo requiera el programa.

En caso de que se requiera que el programa opere en la CMT SIMMP-2 de manera autónoma, el mismo con las asignaciones de direcciones pertinentes (opciones del compilador), deberá ser programado en la EPROM de la tarjeta a partir del origen de la misma, para la ejecución autónoma del programa deberá restablecerse la CMT SIMMP-2 con el puente J11 colocado.

En caso de observarse dificultades al pasar a EPROM el programa que ya ha sido ejecutado correctamente en RAM, podría hacerse lo siguiente:

1) Desde la opción dos del menú de manejo de disco de PUMMA leer el correspondiente archivo S19, para que se genere el archivo BLM respectivo.

2) Programar en la EPROM el archivo BLM generado en el punto anterior, 32 bytes adelante del origen de esta memoria.

3) A partir del origen de la EPROM programar un código que copie de EPROM a RAM la totalidad del código del programa, para después generar un salto a la dirección de RAM donde originalmente se iniciaba el programa cuando el mismo se ejecutaba desde RAM, de esta manera al restablecer la CMT SIMMP-2 con el puente J11 colocado, se copiaría a RAM el programa para su ejecución inmediata en la misma.

Desde luego que las consideraciones, descritas en párrafos anteriores, para ejecución autónoma de un programa escrito originalmente en lenguaje C, serían también aplicables en el caso de que el mismo haya sido escrito en lenguaje ensamblador.

REFERENCIAS:

1) Driscoll. Coughlin. Villanucci.
Data Acquisition and Process Control with the M68HC11 Microcontroller.
Merrill.
1994.

2) Peter Spasov
Microcontroller Technology. The 68HC11.
Prentice Hall.
1993.

APÉNDICE B

LISTA DE ERRORES DE

SINTAXIS PARA EL LENGUAJE SIIL1

LISTA DE ERRORES DE SINTAXIS PARA EL LENGUAJE SIIL1

001 Caracter ";" no encontrado.
002 Instrucción de longitud mayor a 80 caracteres.
003 Operando inexistente en OPR$ (dos o más comas seguidas v.g. xxx,,xx,,)
004 Coma sobrante a la derecha de OPR$ (campo de operandos).
005 Coma sobrante a la izquierda de OPR$ (campo de operandos).
006 Operando faltante (cadena de puros espacios).
007 Etiqueta de más de 20 caracteres.
008 Campo de instrucción (INSTRU$) inexistente.
009 Reservado.
010 Directiva y/o comando de inicialización faltantes.
011 Cadena sobrante a la derecha de una directiva EQU.
012 Directiva EQU con primer caracter no letra.
013 Signo igual no encontrado en directiva EQU.
014 Cadena a la derecha de un signo igual inexistente o de puros espacios.
015 Comando de inicialización (COIN$) inválido.
016 Primer caracter de INSTRU$ no letra.
017 Instrucción inexistente.
018 Declaración FINPP colocada más de una vez.
019 Cadena binaria indicativa inválida.
020 Operando con caracter inicial no letra.
021 Especificación de bit en, grupo de entrada o salida, inválida.
022 Especificación de grupo, en operando XIJ, inválida.
023 Letra en operando real diferente de E, S o I.
024 Carácter delimitador de número de dispositivo no encontrado.
025 Especificación de número de dispositivo inválido.
026 Número de dispositivo previamente existente.
027 Cadena de operandos esperada inexistente OPR$="".
028 Operando que no cuadra con ninguna DIREQU$.
029 Salida S declarada anteriormente.
030 Salida I declarada anteriormente.
031 Entrada E declarada como salida.
032 Configuración inválida detectada al escribir ESCBUFX.BLM .
033 Multideclaración de comando FINPP.
034 Instrucción inexistente.
035 Número de operandos incongruente con instrucción.
036 Entrada física declarada como salida.
037 Cadena binaria (MAIE) de longitud inválida.
038 Grupo de entrada inválido.
039 Grupo de salida inválido.
040 Grupo de VBI inválido.
041 Carácter # no encontrado.
042 Especificación de grupo vacia.

043 Dígito o dígitos inválido en cadena especificadora de tiempo.
044 Cadena indicadora de tiempo (00 00 00.00) inválida.
045 Declaración INMODI antes de FINPP.
046 Multideclaración de INMODI.
047 Declaración FINMODI antes de INMODI.
048 Multideclaración de FINMODI.
049 Número de temporizador fuera de rango.
050 Cadena binaria indicativa inválida.
051 Fin de archivo antes de poder leer tabla de datos.
052 Cadena vacía en renglón de datos de tiempos TM's.
053 Especificación de tiempos inválida (cadena correspondiente de longitud > 11)
054 Cadena especificadora de un TM inconsistente (de longitud inadecuada).
055 Tiempo TM menor de tiempo TC.
056 Número de tiempos TM rebasa el valor NTM.
057 TC mayor o igual a TM en TEMPOE.
058 Renglón leído no corresponde con campo de datos esperado (caracter # en primera columna no encontrado).
059 Número de estados en secuenciador mayor que NESMAX (NESMAX=1000).
060 Número de estados en secuenciador declarado como cero.
061 Declaración no binaria o hexadecimal en "dato estado" asociado con secuenciador.
062 Caracteres Hex definitorios de estado de más de dos dígitos.
063 Dígito no hexadecimal en declaración de estado de secuenciador.
064 Número de dígitos en estado Hex diferente de 2.
065 Longitud de cadena binaria, en especificación de estado, diferente de M en secuenciador SECMXN.
066 Número de estados declarados asociados con un secuenciador es superior a el número de estados especificado en declaración de secuenciador (SECMXN, número de estados en datos > N).
067 Número bits "M" en estado de secuenciador mayor que ocho.
068 Máximo número de contadores de eventos rebasado.
069 Cuenta final o cuenta inicial fuera de rango (>65535) en módulo contador de eventos.
070 CUENTAF<CUENTAI en módulo contador ascendente.
071 CUENTAF>CUENTAI en módulo contador descendente.
072 Multidefinición de instrucción DESP.
073 Máximo número de módulos desplegadores tipo mensajero, rebasado.
074 Tamaño de ventana de despliegue mayor que 16, en módulo desplegador tipo mensajero.

075 Tamaño de ventana fuera de rango en módulo desplegador tipo mensajero.

076 Ventana de tamaño incompatible con el valor de CI, declarado en un módulo desplegador de tipo mensajero.

077 Encabezado, en módulo desplegador de tipo mensajero, de longitud mayor que 16.

078 Tiempo de permanencia de ventana fuera de rango en módulo desplegador de tipo mensajero.

079 Digito o digitos inválidos en cadena especificadora de un valor numérico.

080 Número de tiempos TM inferior al declarado en TEMPOG o TEMPOB.

081 Número de estados leidos en campo de datos asociado con un secuenciador es menor que el declarado en la correspondiente instrucción SECMXN.

082 Multideclaración de instrucción MANDESP.

083 Tope inválido para módulos desplegadores.

084 Multidefinición de RTR.

085 Número de temporizador tipo B fuera de rango.

086 Especificación no válida para clase, en contador tipo B.

087 Especificación de clase fuera de rango en contador tipo B.

088 Especificación de número de especificaciones de disparo, no válida.

089 Número de especificaciones de disparo fuera de rango.

090 Denotación inválida en especificación de disparo de temporizador tipo B.

091 Longitud inválida de cadena especificadora de tiempos de disparo asociado con temporizador tipo B.

092 Día de la semana inválido en especificación de disparo asociado con temporizador tipo B.

093 Día del mes incongruente en especificación de disparo de temporizador tipo B.

094 Número de especificaciones de disparo en temporizador tipo B incongruente con lo declarado en la correspondiente instrucción.

095 Expresión para número de contador de eventos inválida en módulo observador de contador de eventos.

096 Número de contador mayor que 80 en módulo observador de contador de eventos.

097 Expresión no numérica para columna inicial "CI"en módulo observador de contador de eventos.

098 Valor para columna inicial "CI" fuera de rango,en módulo observador de contador de eventos.

099 Expresión no numérica para "ND" (número de digitos) en módulo observador de contador de eventos.

100 Número de dígitos "ND" fuera de rango en módulo observador de contador de eventos.

101 Dígitos de contador de eventos, rebasan límites físicos de la unidad de despliegue.

102 Expresión para número de renglón inválida en módulo observador de contador de eventos.

103 Número de renglón fuera de rango, en módulo observador de contador de eventos.

104 Especificación inválida de mensaje de alarma.

105 Número de mensaje de alarma asociado, mayor que 200.