



**FACULTAD DE INGENIERÍA UNAM
DIVISIÓN DE EDUCACIÓN CONTINUA**



**DIPLOMADO EN
MICROCONTROLADORES (SISTEMAS
EMBEBIDOS)**

**CA 207 MÓDULO I
MICROCONTROLADORES 8 BITS**

TEMA:

EL MPLAB

AUTOR: FERNANDO REMIRO DOMÍNGUEZ
INSTRUCTOR: ING. DANIEL MARTÍNEZ GUZMÁN

**DEL 27 DE SEPTIEMBRE AL 1° DE OCTUBRE DE 2004
PALACIO DE MINERÍA**



DIVISIÓN DE
EDUCACIÓN
CONTINUA

Programa 2004

Autor: Fernando Remiro Domínguez
Profesor de Sistemas Electrónicos del
IES Juan de la Cierva
<http://teletel.terra.es/personal/fremiro/>

EL MPLAB

El MPLAB es un software que junto con un emulador y un programador de los múltiples que existen en el mercado, forman un conjunto de herramientas de desarrollo muy completo para el trabajo y/o el diseño con los microcontroladores PIC desarrollados y fabricados por la empresa Arizona Microchip Technology (AMT).

El MPLAB incorpora todas las utilidades necesarias para la realización de cualquier proyecto y, para los que no dispongan de un emulador, el programa permite editar el archivo fuente en lenguaje ensamblador de nuestro proyecto, además de ensamblarlo y simularlo en pantalla, pudiendo ejecutarlo posteriormente en modo paso a paso y ver como evolucionarían de forma real tanto sus registros internos, la memoria RAM y/o EEPROM de usuario como la memoria de programa, según se fueran ejecutando las instrucciones. Además el entorno que se utiliza es el mismo que si se estuviera utilizando un emulador.

En las siguientes líneas se pretende ayudar a todos aquellos que se enfrentan por primera vez a este programa, tanto en su instalación como en la utilización de esta potente herramienta que nos proporciona Arizona Microchip Technology. En el CD-ROM que se adjunta con este curso se encuentran las versiones 4.12.00 y la 4.99.07 del programa; éstas versiones y las nuevas que van saliendo cada poco tiempo y que incorporan nuevos tipos de microcontroladores, se pueden obtener de forma gratuita en la página web www.microchip.com, el la cual se encuentra una amplia información sobre todos los dispositivos que fabrica AMT.

De las dos versiones, nosotros vamos a centrarnos en la V.12.00, por

ser esta la que menos recursos de software y hardware necesita para trabajar con ella, además está pensada para trabajar con las herramientas de desarrollo MPLAB-ICD y el PICSTART que se encuentran ya muy difundidas, mientras que la versión V.99.07 está pensada para trabajar con el MPLAB-ICE 2000 soportado en NT, esta versión presenta algunas modificaciones en las ventanas de configuración del programa respecto a la anterior, además para su correcto funcionamiento es necesario disponer de la gama alta de los sistemas operativos que se indican en el siguiente apartado además del hardware más potente.

INSTALACIÓN DEL PROGRAMA

Los requerimientos mínimos para la instalación de los programas son:

- Procesador 386, 486 o Pentium
- Windows 3.1/ 95/ 98, Windows NT 3.51/4.0, Windows 2000, MACOS 7.0, o Unix compatible OS
- 16 MB de memoria RAM para sistema con Windows 95.
- 24 MB de RAM para Windows NT systems.
- 32 MB para sistemas con Windows 2000.
- Unidad de CD-ROM.
- Navegador (3.0 HTML)

Se recomienda por AMT:

- Procesador Pentium
- 32 MB de memoria RAM

El CD-ROM de Microchip requiere para su navegación de un programa *HTML*. Para los equipos con Windows 95/98/NT se recomienda utilizar Internet Explorer versión 5.0 o el Netscape Navigator versión 4.0, además hará falta para la lectura de los numerosos documentos en formato *pdf* el programa Adobe Acrobat Reader versión 3.0 o 4.0.

Estos programas pueden obtenerse gratuitamente en las correspondientes web:

Microsoft Internet Explorer :
www.microsoft.com
 Netscape Navigator:
www.netscape.com
 Adobe Acrobat Reader:
www.adobe.com
 Winzip: <http://www.pkware.com> o en
<http://www.winzip.com>

Al introducir el CD-ROM en la unidad correspondiente en los sistemas que tengan instalado Windows 95/98, Windows 2000 o NT y que tenga habilitada la opción *autorun*, aparece en pantalla el mensaje de la Figura 1, activaremos el botón de *Yes*.

En caso de que no aparezca este mensaje al arrancar el CD-ROM, buscar con el explorador de Windows el archivo `\index.txt.htm` para el formato de texto o el `\index.htm` para el formato gráfico, al ejecutar este archivo aparecerá una pantalla como la que se muestra en la Figura 2, que es similar a la que tiene Microchip en su página web.

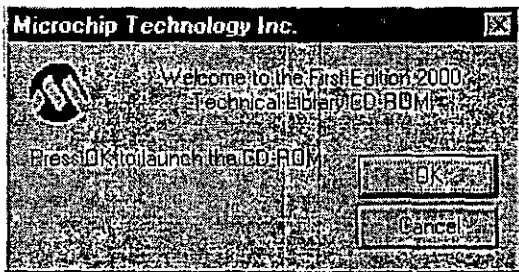


Figura 1.- Mensaje al arrancar el CD-ROM

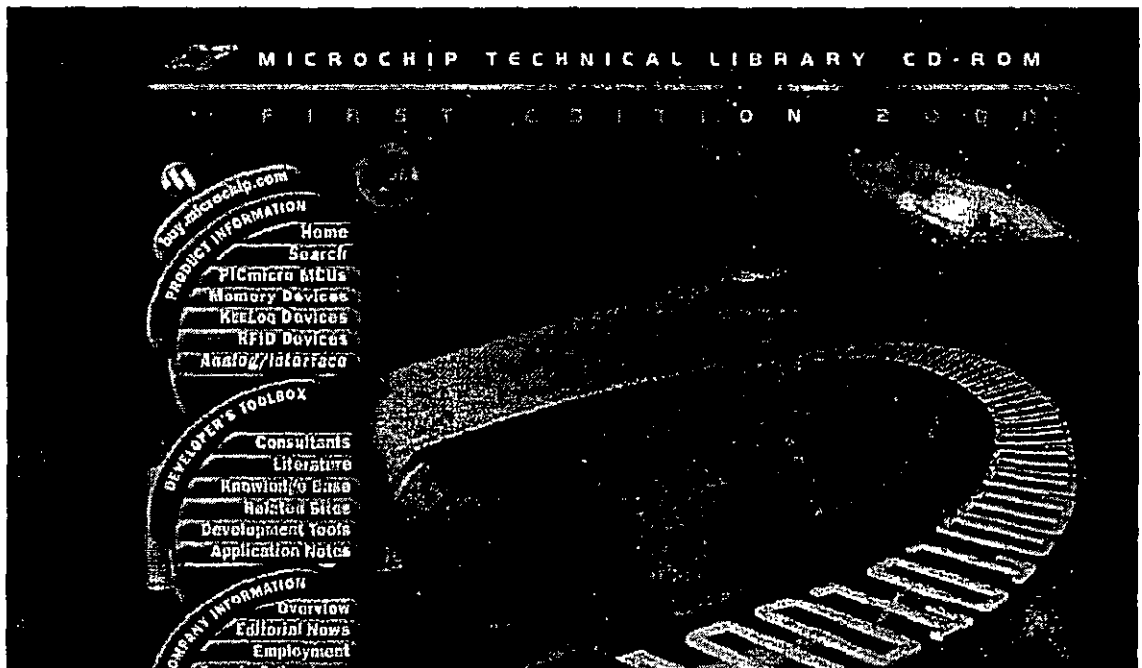


Figura 2.- Pantalla de presentación del CD-ROM y de la página web de Microchip

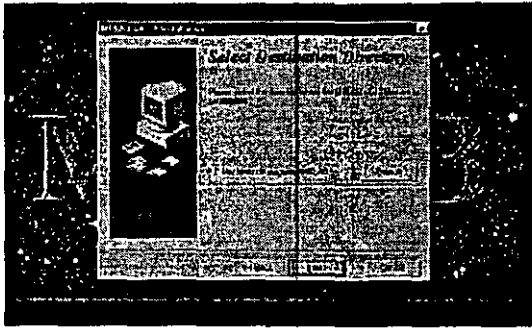


Figura 6.- Selección del subdirectorio donde se instalará el programa MPLAB

Una vez finalizada la instalación que puede tardar unos minutos, se puede pasar a ejecutar el programa MPLAB, es recomendable que si se va a utilizar mucho este programa, lo cual esperamos, se haga un acceso directo a dicho programa con lo que tendremos en el escritorio de Windows un icono como el de la Figura 7.



Figura 7.- Icono de acceso directo al programa MPLAB

Antes de seguir adelante, recomendamos crear una carpeta con el explorador de Windows por ejemplo *C:\Archivos de programas\MPLAB\Trabajo*, dentro de la cual posteriormente iremos creando todos nuestro proyectos.

COMO EMPEZAR.

Cuando se pulsa el icono del MPLAB aparece una pantalla como la que se muestra en la Figura 8.



Figura 8.- Escritorio del MPLAB

Lo primero que haremos es seleccionar el modo de trabajo como simulador y el tipo de microcontrolador con el que queremos trabajar. Para ello se selecciona el botón de *Options* de la barra del control que aparece en el escritorio y del menú desplegable la opción *Development Mode*, con lo que aparece la pantalla de la Figura 9 en la que se activa el modo *MPLAB-SIM simulator* y el microcontrolador con el que se desea trabajar, que en nuestro caso será el *PIC16F84*, por último, pulsamos el botón de *Reset* para aceptar los cambios.

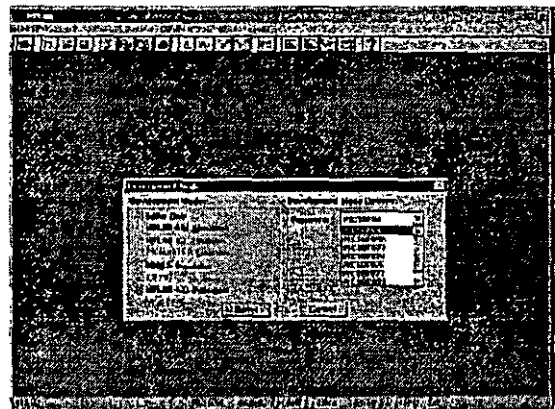


Figura 9.- Selección de la opción de trabajo como simulador y el tipo de microcontrolador

Los iconos que aparecen en la barra de herramientas, son funciones que se encuentran incluidas en el menú de control, pero como en todos los programas de Windows se incluyen para manejar de forma más cómoda el programa. Seguidamente comentaremos

que significa cada uno de los iconos de la barra de herramientas que aparece en esta pantalla, mas adelante veremos que

hay más barras de herramientas que pueden ser conmutadas

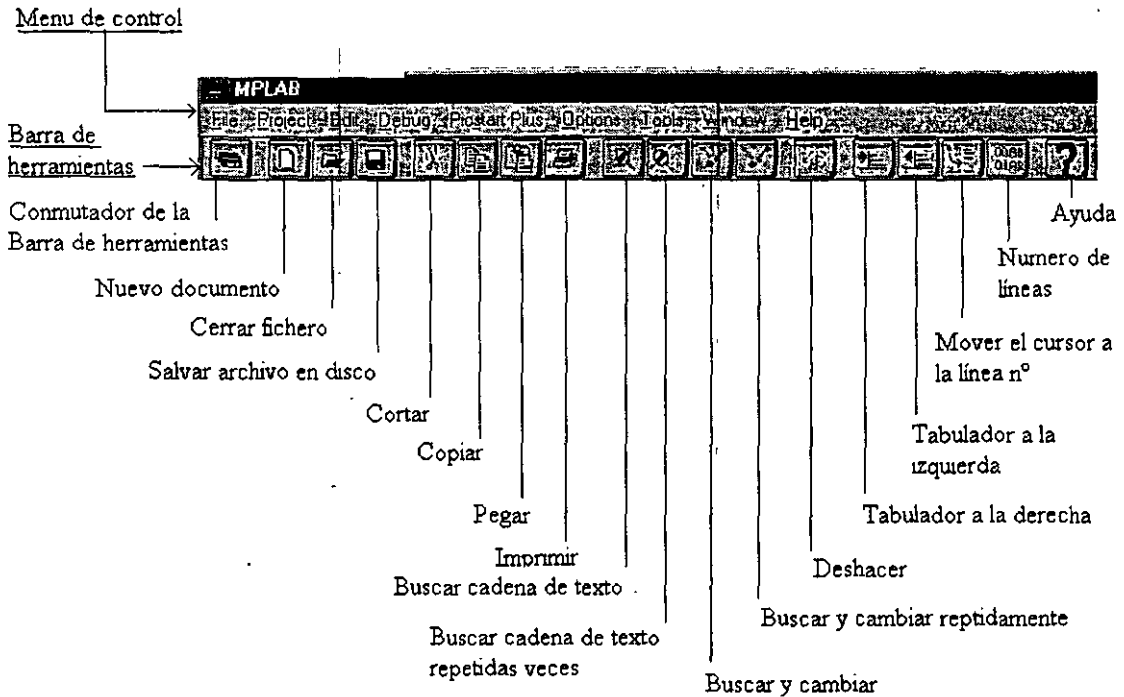


Figura 10.- Barra de herramientas de edición

NUESTRO PRIMER PROYECTO

Bueno, pues ya estamos en condiciones de crear nuestro primer proyecto, para ello comenzamos por activar en el menú de control la opción *File>New* o bien activamos el icono de *crear nuevo documento* en la barra de herramientas. El programa contestará con el cuadro de diálogo de la Figura 11.

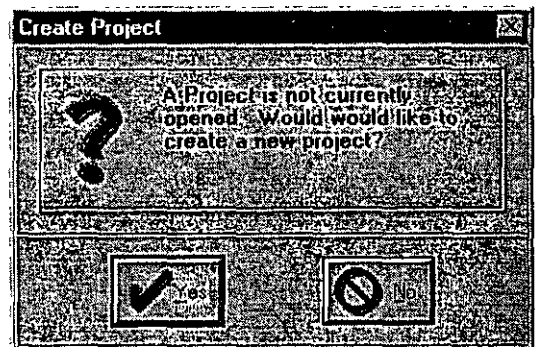


Figura 11.- No hay ningún proyecto abierto ¿Quiere crear un nuevo proyecto?

Activamos el botón de **Yes** y aparece un cuadro de diálogo como el de la Figura 12 en el que se nos pide el nombre del proyecto que tendrá extensión ***.pjt**, como este es nuestro primer proyecto le llamaremos **ejer1.pjt** y lo guardaremos en la carpeta de **trabajo** que habíamos creado anteriormente.

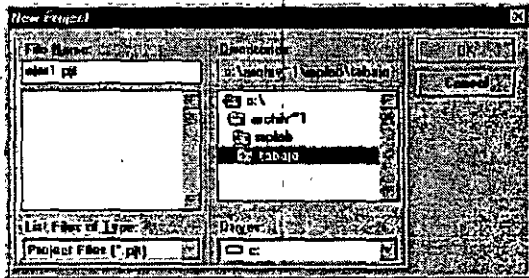


Figura 12.- Creación de un nuevo proyecto

El programa devuelve el cuadro de diálogo de la Figura 13

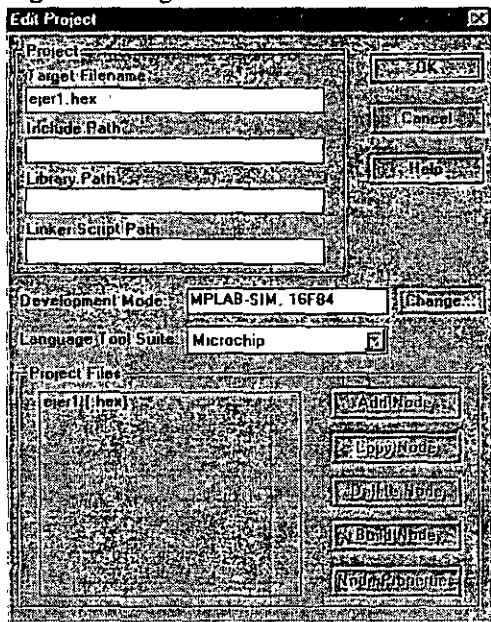


Figura 13.- Propiedades de edición del proyecto

Activamos el botón de **OK** y estamos en condiciones de empezar a escribir nuestro primer proyecto al aparecer una pantalla como la de la Figura 14

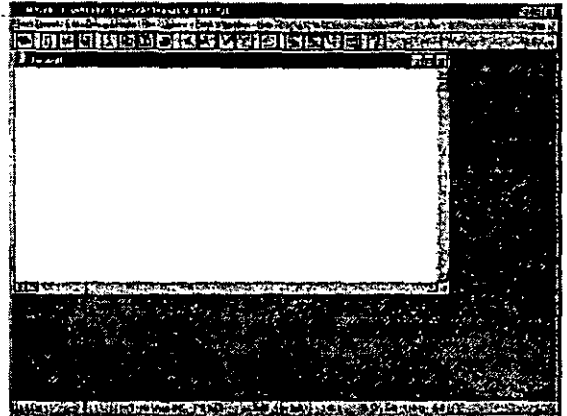


Figura 14.- Apertura del documento para comenzar a escribir nuestro proyecto

EL EDITOR

Comencemos por lo tanto a escribir en lenguaje ensamblador nuestro primer programa que llamaremos **ejer1.asm** y que se muestra en la Figura 15. El programa realiza la suma en binario de dos números ($7+8 = 15$) y para escribirlo usamos el editor de textos. La extensión ***.asm** es la que deben llevar todos los programas escritos en ensamblador.

Deberemos de tener en cuenta que la primera columna del editor está reservada para las **etiquetas** que son expresiones alfanuméricas escogidas por el usuario que definen valores de posiciones de memoria. Estas deben empezar siempre por una letra. Además se debe de tener en cuenta que no pueden usarse expresiones que ya utiliza el ensamblador tales como:

- Instrucciones
- Directivas del propio ensamblador
- Nombres de registros especiales (SFR)
- Nombre de cada uno de los bit de los registros especiales.

En las siguientes columnas, se puede comenzar a escribir el nemónico de la instrucción o las directivas del ensamblador. Por último hay que decir que se pueden y se deben añadir comentarios que son elementos indispensables en muchos casos para seguir el razonamiento de los programas sin perderse,

para ello cuando el MPLAB encuentra un “;”(punto y coma) no se genera código máquina.

En todos estos campos los espacios en blanco no son significativos y las líneas en blanco tampoco.

Para una mejor legibilidad del programa, se recomienda acceder a cada campo utilizando el tabulador.

El uso de mayúsculas y minúsculas en los programas obedece a una serie de reglas o normas de estilo, comunes entre los programadores en ensamblador, que si bien no son obligatorias, facilitan la

lectura del código fuente. Estas reglas son:

- Las directivas del ensamblador se escriben en mayúsculas
- Los nombres de las variables se escriben en mayúsculas.
- Los nemónicos de las instrucciones se escriben en minúsculas
- El programa se escribe utilizando los tabuladores para definir las distintas columnas, tales como etiquetas, comienzo de líneas de programa y columna donde empiezan los comentarios separados por un “;”(punto y coma).

```

MPLAB - C:\ARCHIV~1\MPLAB\TABAJD\EJER1.PJT - [c:\archiv~1\mplab\tabajo\ejer1.asm]
File Project Edit Debug Project Options Tools Window Help
*****
; Programa Ejer1.ASM                               Fecha : 1 - Marzo - 2000
; Este programa suma dos valores inmediatos (en este caso 7+8) el resultado se deposita
; en la posición 0x10
; Revisión : 0.0
; Velocidad del Reloj: 4 MHz                       Programa para PIC16C84 y PIC16F84
; Perro Guardian : deshabilitado                 Reloj Instrucción: 1 MHz = 1 uS
; Protección del código : OFF                   Tipo de Reloj : XT
*****
LIST      p=16F84          ;Tipo de procesador
RESULTADO EQU      0x10    ;Define la posición del resultado
          ORG      0x00    ;Vector de Reset
          goto    INICIO
          ORG      0x05    ;Salva el vector de interrupción
INICIO    movlw   0x07     ;Carga 1er. sumando en W
          addlw  0x08     ;Suma el 2º sumando
          movwf  RESULTADO ;Almacena el resultado
STOP     sleep          ;Poner ponerse a dormir
          |
          END           ;Fin del programa fuente

```

Figura 15.- Nuestro primer programa ejer1.asm

Cuando terminemos de escribir el programa seleccionamos **File>Save** con lo que aparece el cuadro de diálogo de la Figura 16, donde le damos el nombre a nuestro programa **ejer1.asm**, dentro de nuestra carpeta **Trabajo**.

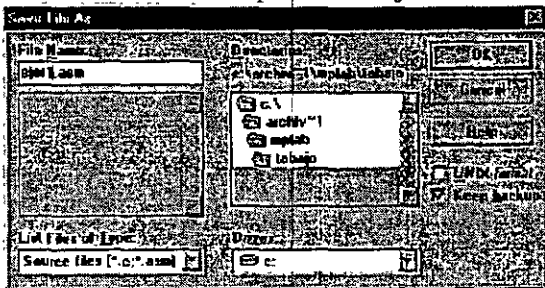


Figura 16.- Salvar el programa **ejer1.asm** en la carpeta de trabajo

El siguiente paso será volver a editar nuestro proyecto seleccionando en el menú de control **project>edit project**, lo que provoca que aparezca el menú de la Figura 17.

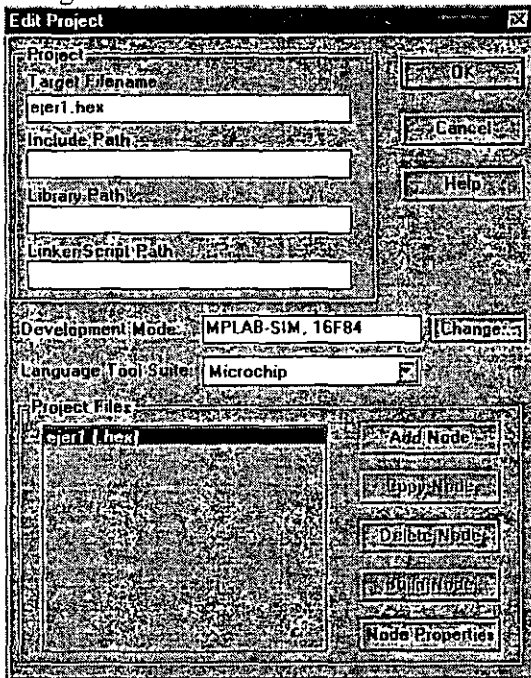


Figura 17.- Pantalla de edición del proyecto

Pulsamos sobre **ejer1.asm**, y se activa el botón de **Node Properties**, que hasta el momento aparecía de color gris, si lo activamos aparece el cuadro de diálogo de la Figura 19, donde están reflejadas todas las propiedades del nodo actual. Sin modificar ninguna de estas propiedades se pulsa el botón de **OK** para continuar, lo que nos lleva de nuevo a la pantalla de la Figura 17. Ahora seleccionamos el botón **Add Node** (añadir elementos al nodo), lo que provoca que aparezca un nuevo cuadro de diálogo como el de la Figura 18, en el que seleccionaremos el archivo **ejer1.asm**

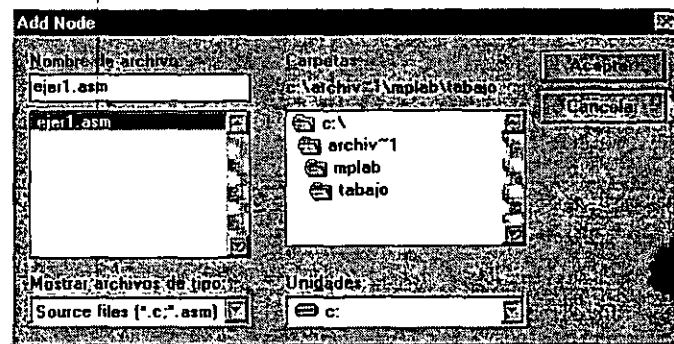


Figura 18.- Nombre del archivo a incluir en el proyecto **ejer1.asm**

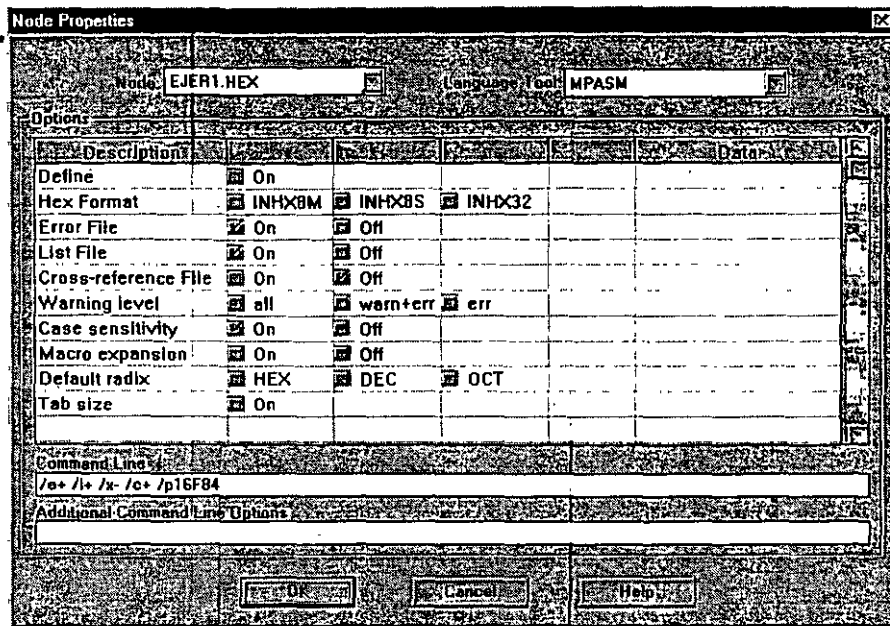



Figura 19.- Propiedades del nodo de nuestro proyecto donde se seleccionan los ficheros y formatos que se obtendrán al ensamblar el programa.

Pulsamos el botón de *Aceptar* y se vuelve a la pantalla de la Figura 17 en la que ha aparecido el fichero *ejer1[.asm]* junto al fichero *ejer1[.hex]* que aparecía antes en el campo de *Project files*. Seguidamente pulsamos el botón de *OK*, lo que nos llevará de vuelta a la pantalla de la Figura 15. Para ensamblar el programa seleccionamos en el menú de control la opción *Project>Build All* (también podríamos haber pulsado el botón correspondiente de la barra de herramientas del simulador , como luego veremos), y si no se han cometido errores al introducir los códigos, aparece una pantalla como la de la Figura 20, lo que nos indica que el programa se ha ensamblado con éxito y ya estamos en condiciones de iniciar la simulación del programa. Si por el contrario, se han detectado errores, en dicha pantalla será mostrado el error; si se hace doble clic sobre la línea que muestra el error, el cursor saltará directamente a la línea de código donde se encuentra el error. Una vez subsanados los errores habrá que volver a compilar el programa.

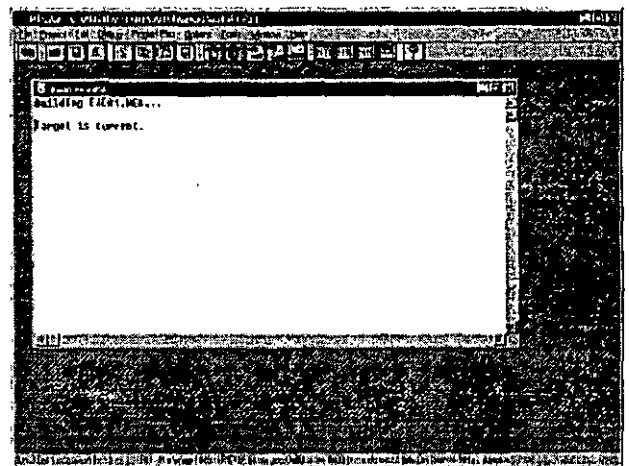


Figura 20.- Pantalla del MPLAB una vez ensamblado correctamente el programa fuente

LA BARRA DE MENÚS

Seguidamente analizaremos las distintas posibilidades de la barra de menú del MPLAB, si bien ya hemos utilizado algunas de las posibles opciones que presenta la barra de herramientas, ahora analizaremos estas una por una.

1.- Windows:

Al activar esta opción de la barra de menú, aparece el menú desplegable de la Figura 21.

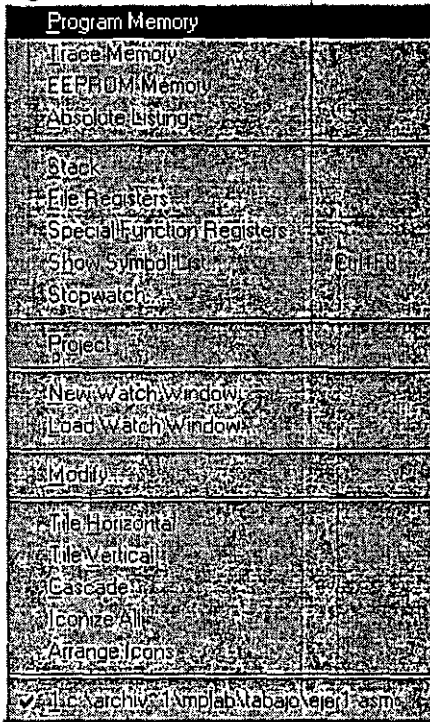


Figura 21.- Menú desplegado de la opción Windows de la barra de herramientas.

1.1.- Program Memory : Al seleccionar esta opción aparece la pantalla de la Figura 22 en la que se puede apreciar las posiciones de memoria que ocupa cada una de las instrucciones, el código de operación de cada instrucción y la posición de memoria que se le ha dado a cada etiqueta.

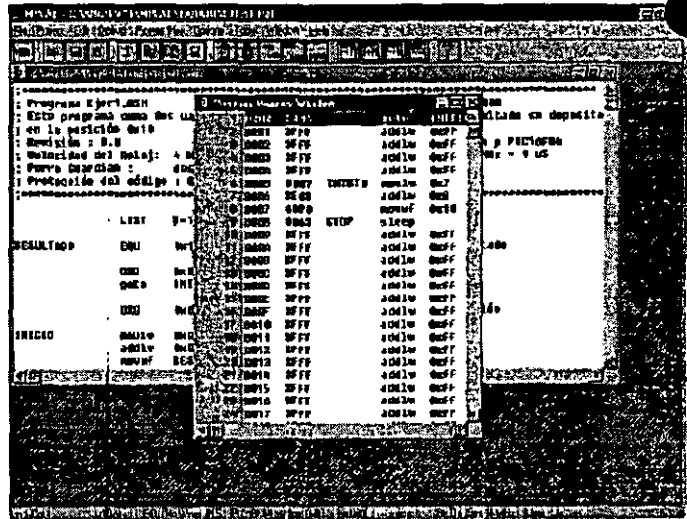


Figura 22.- Pantalla de la ventana de Memoria de programa.

Si hacemos clic sobre la barra de *menú del sistema*, activando el icono que hay en la parte superior izquierda de esta ventana, aparece el menú desplegable de la Figura 23, en el que se puede seleccionar entre tres formas de ver la memoria de programa:

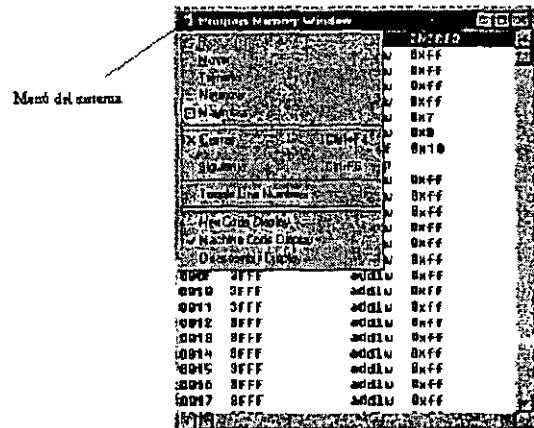


Figura 23.- Despliegue de opciones del menú de sistema.

- **Hex Code Display:** representa la memoria de programa con los datos en hexadecimal. Esta opción es muy útil al usar el programador del dispositivo y comprobar si se grabaron bien los datos. La pantalla que se obtiene es la que se muestra en la Figura 24.

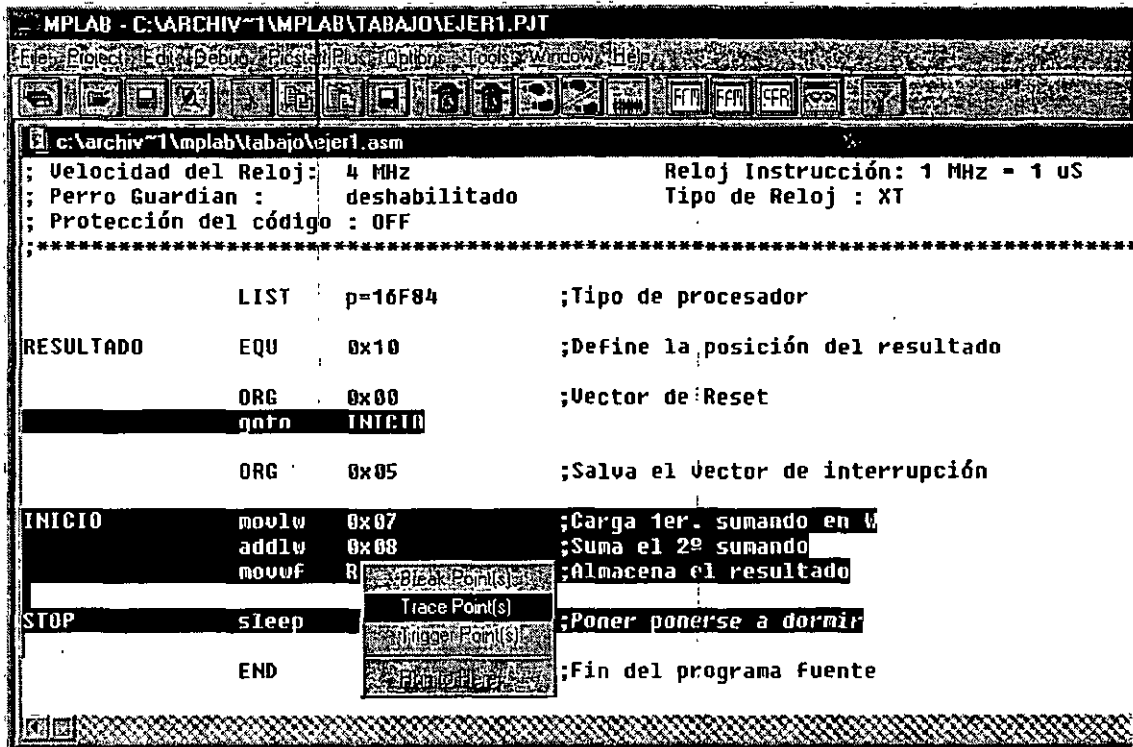




Figura 25.- Selección de las líneas de programa para cargar el buffer de traza.

Al seleccionar la opción *Trace Point(s)* aparecen resaltadas las líneas en color verde. Seguidamente se activa el icono del semáforo verde  (*Run*), lo que hará ejecutar la simulación en “tiempo real” (no olvidemos que en el simulador emula el funcionamiento del microcontrolador y es mucho más lento que este), y después de unos segundos, si activa-

mos el icono del semáforo rojo  *Halt the processor*, se detiene la ejecución del programa. Si ahora se activa dentro de la opción *Window>Trace Memory*, se pueden ver la traza obtenida, que en nuestro caso en la que se muestra en la Figura 26.

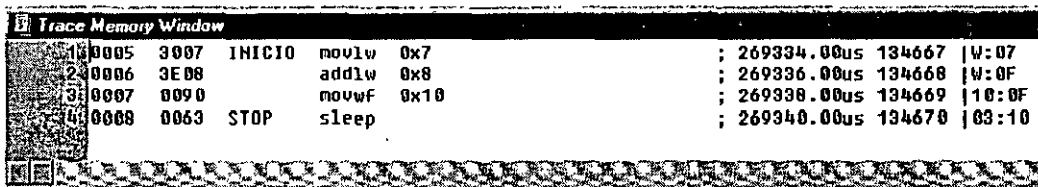


Figura 26.- Traza de memoria obtenida en el programa con los valores marcados en la Figura 25

El simulador muestra en esta ventana el valor del tiempo que tarda en ejecutar cada línea de programa y también cualquier variación sobre los registros al ejecutarse el código de instrucción.

1.3.- EEPROM Memory: Si el dispositivo emulado tiene EEPROM o memoria

Flash, como es el caso del PIC16C84 o el 16F84 respectivamente, el contenido de la memoria EEPROM puede verse seleccionando la opción *Window>EEPROM*.

La memoria de EEPROM no puede modificarse a través de esta ventana. Para ello hay que utilizar el menú de dialogo al que se accede seleccionando

Window>Modify..., que se describe más adelante.

1.4.- Absolute Listing: La Ventana de "Listado de Programa", realmente nos presenta el archivo de nuestro proyecto con extensión **.lst*, donde se puede ver

el archivo generado por el ensamblador o compilador. El listado muestra el código fuente en modo absoluto con el código objeto generado, tal y como se puede ver en la Figura 27.

```

c:\archiv\Tmplab\abajo\ejer1.lst
MPASM 02.30 Released          EJER1.ASM    2-28-2000  10:28:18          PAGE 1

LOC OBJECT CODE      LINE SOURCE TEXT
VALUE

00001 ;*****
00002 ; Programa Ejer1.ASM                      Fecha : 1 - Marzo
00003 ; Este programa suma dos valores inmediatos (en este caso 7+8) el
00004 ; en la posición 0x10
00005 ; Revisión : 0.0                          Programa para PIC1
00006 ; Velocidad del Reloj: 4 MHz              Reloj Instrucción:
00007 ; Perro Guardian : deshabilitado         Tipo de Reloj : X1
00008 ; Protección del código : OFF
00009 ;*****
00010
00011          LIST      p=16F84          ;Tipo de procesador
00012
00000010      00013 RESULTADO EQU      0x10          ;Define la posición del re
00014
0000          00015          ORG      0x00          ;Vector de Reset
0000 2805      00016          goto    INICIO
00017
0005          00018          ORG      0x05          ;Salva el vector de interr
00019
0005 3007      00020 INICIO      movlw   0x07          ;Carga 1er. sumando en W
00021

```

Figura 27.- Archivo ejer1.lst

Además al final de este archivo aparece la información de las etiquetas utilizadas en el programa, en que línea de programa se encuentran, la memoria utilizada, la memoria libre además de los errores, *warnings* y mensajes reportados por el ensamblador.

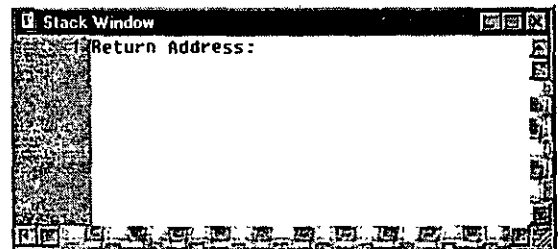


Figura 28.- Ventana de la Pila

1.5.- Stack: El contenido de la pila puede verse al seleccionar la opción **Window>Stack**.

Los contenido de la pila puede mostrarse con o sin número de línea. El formato de presentación se selecciona a través del menú del sistema. Si la Pila se desborda, el MPLAB indica con su rebo-samiento con el mensaje *underflow*.

Al menú del sistema se accede pulsando el botón de la esquina de la ventana.

1.6.- File Registers : La lista de registros de propósito general (GPR) del micro-controlador, que son de memoria SRAM, se pueden ver seleccionando la opción **Window>File**. Esta ventana al desplegarse presenta una lista con todos los registros de proposito general del dispositivo emulado, tal y como se muestra en la Figura 29.

su vez el movimiento giratorio de una leva o de un árbol unido al dispositivo de disparo. Si la corriente absorbida por el receptor supera el valor de reglaje del relé, las biláminas se deformarán lo bastante como para que la pieza a la que están unidas las partes móviles de los contactos se libere del tope de sujeción. Este movimiento causa la apertura brusca del contacto del relé intercalado en el circuito de la bobina del contactor y el cierre del contacto de señalización. El rearme no será posible hasta que se enfrien las biláminas. Relés térmicos LR2-D de Telemecanique (1) La norma IEC 947-4 sustituye el concepto de "relé diferencial" por el de "relé sensible a una pérdida de fase".

Compensación de la temperatura ambiente

La curvatura que adoptan las biláminas no sólo se debe al calentamiento que provoca la corriente que circula en las fases, sino también a los cambios de la temperatura ambiente. Este factor ambiental se corrige con una bilámina de compensación sensible únicamente a los cambios de la temperatura ambiente y que está montada en oposición a las biláminas principales. Cuando no hay corriente, la curvatura de las biláminas se debe a la temperatura ambiente. Esta curvatura se corrige con la de la bilámina de compensación, de forma tal que los cambios de la temperatura ambiente no afecten a la posición del tope de sujeción. Por lo tanto, la curvatura causada por la corriente es la única que puede mover el tope provocando el disparo.

Los relés térmicos compensados son insensibles a los cambios de la temperatura ambiente, normalmente comprendidos entre $-40\text{ }^{\circ}\text{C}$ y $+60\text{ }^{\circ}\text{C}$.

Reglaje

Los relés se regulan con un pulsador que modifica el recorrido angular que efectúa el extremo de la bilámina de compensación para liberarse del dispositivo de sujeción que mantiene el relé en posición armada. La rueda graduada en amperios permite regular el relé con mucha precisión. La corriente límite de disparo está comprendida entre 1,05 y 1,20 veces el valor indicado.

Detección de una pérdida de fase

Este dispositivo provoca el disparo del relé en caso de ausencia de corriente en una fase (funcionamiento monofásico). Lo componen dos regletas que se mueven solidariamente con las biláminas. La bilámina correspondiente a la fase no alimentada no se deforma y bloquea el movimiento de una de las dos regletas, provocando el disparo.

Los receptores alimentados en corriente monofásica o continua se pueden proteger instalando en serie dos biláminas que permiten utilizar relés sensibles a una pérdida de fase. Para este tipo de aplicaciones, también existen relés no sensibles a una pérdida de fase.

Clases de disparo

Los relés térmicos se utilizan para proteger los motores de las sobrecargas, pero durante la fase de arranque deben permitir que pase la sobrecarga temporal que provoca el pico de corriente, y activarse únicamente si dicho pico, es decir la duración del arranque, resulta excesivamente larga. La duración del arranque normal del motor es distinta para cada aplicación; puede ser de tan sólo unos segundos (arranque en vacío, bajo par resistente de la máquina arrastrada, etc.) o de varias decenas de segundos (máquina arrastrada con mucha inercia), por lo que es necesario contar

dialogo de la opción *Modify* (Figura 40) en la que aparecerá ya la dirección del registro seleccionado.

1.8.- Show Symbol List :

Esta ventana muestra un listado de los símbolos, es decir variables y etiquetas, utilizadas en el código fuente del programa.

Estos símbolos están en el archivo *.COD de nuestro proyecto. En la Figura 32 se muestra el listado de símbolos de nuestro programa.

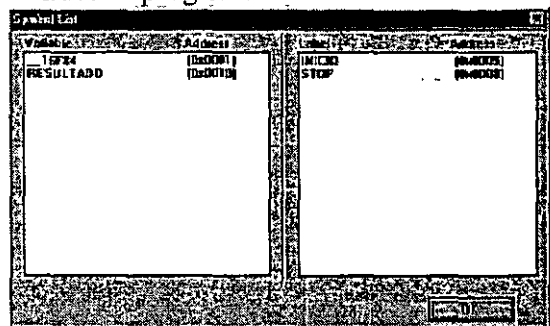


Figura 32.- Ventana del listado de símbolos utilizados en nuestro programa fuente.

1.9.- Stopwatch and Clock Frequency :

Para calcular el tiempo de ejecución de nuestro programa o de una subrutina, podemos contar el número de instrucciones que se realizan y multiplicarlo por 4 veces la frecuencia de la señal de reloj (tiempo de un ciclo máquina) o por 8 en el caso de que las instrucciones sean de salto. Esto en algunas ocasiones es engorroso, pero el MPLAB con esta opción de cronómetro nos permite medir tiempo de ejecución de las instrucciones de nuestro programa sin equivocaciones.

El cronómetro calcula el tiempo basándose en la frecuencia del reloj del microcontrolador PIC que estamos simulando, para ello previamente debemos fijar la frecuencia del oscilador empleado. Esto se realiza haciendo los siguientes pasos: Activamos desde el menú *Options>Processor Setup>Clock frequency* tal y como se muestra en la Figura 33

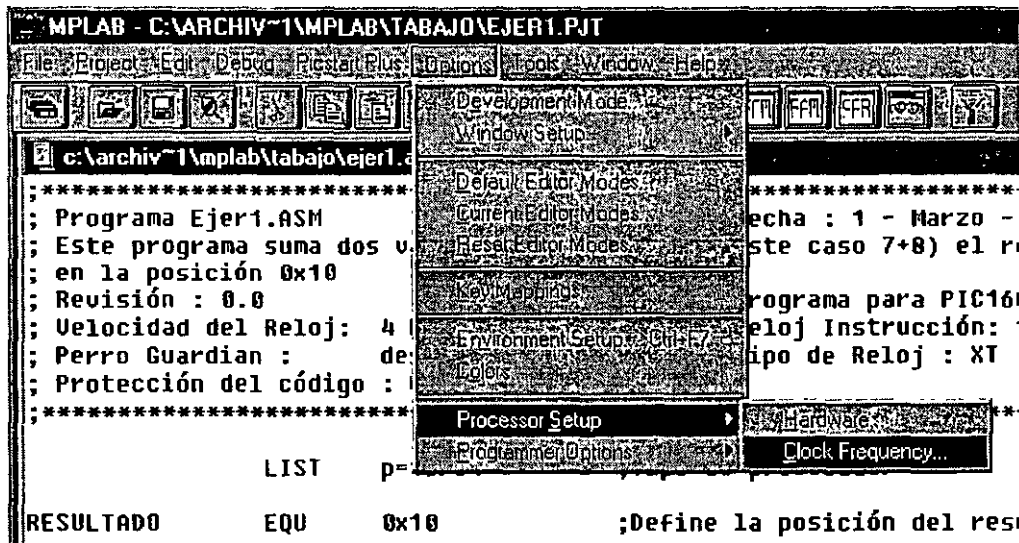


Figura 33.- Camino a seguir para definir la frecuencia del microcontrolador *Options>Processor Setup>Clock frequency*

Inmediatamente se abre un cuadro de dialogo como la de la Figura 34, donde se fija la frecuencia del reloj.

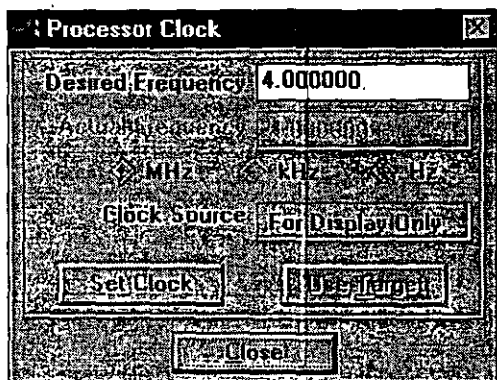


Figura 34.- Definición de la frecuencia de oscilador del microcontrolador.

Después se activa la opción *Windows>StopWatch*, con esto conseguimos tener siempre abierta la ventana que muestra el tiempo transcurrido y los ciclos máquina empleados en la ejecución de cada instrucción, como puede verse en la Figura 35.

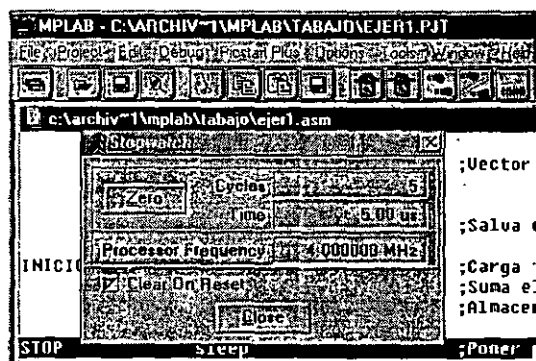


Figura 35.- Cronometro para contar el tiempo que tarda en ejecutarse un programa o parte de él.

1.10.- Project Windows : La Ventana del Proyecto sólo está disponible cuando hay un proyecto abierto. Presenta la lista de archivos que actualmente hay en dicho proyecto. Si el proyecto se ha ensamblado o compilado la ventana del proyecto muestra una lista de todos los archivos incluidos en el proyecto.

Por otra parte, la ventana del Proyecto sólo presenta el archivo del proyecto principal. Un doble clic en cualquier archivo resaltado en la ventana

del proyecto, abrirá dicho archivo para su revisión.

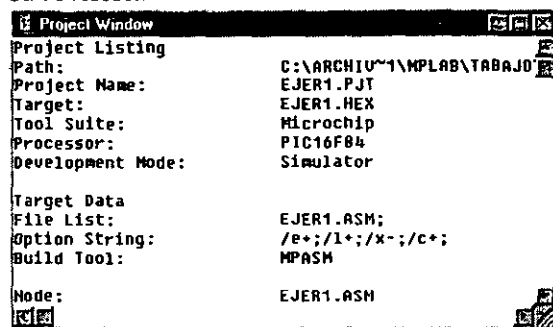


Figura 36.- Ventana de Proyecto

1.11.- Watch Windows :

MPLAB permite supervisar los contenidos de los registros del archivo a través de una ventana temporal. Para abrir una ventana temporal, se selecciona *Window>Watch Windows*. El programa responde con un cuadro de diálogo como el de la Figura.37

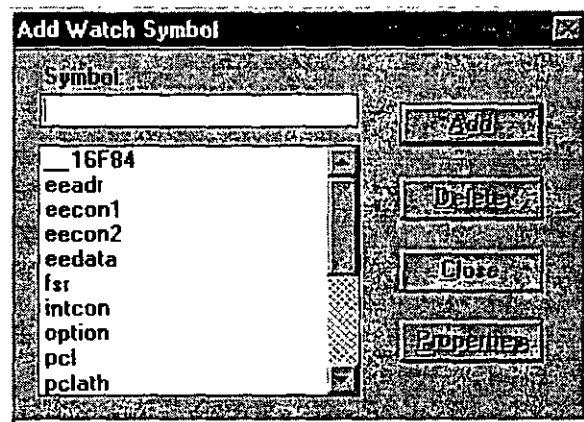


Figura 37.- Cuadro de diálogo de los símbolos de la ventana temporal

Para agregar los registros a visualizar, poner el ratón encima de uno de ellos pulsar el botón de la izquierda, seguidamente activar el botón de *Add*. También se pueden anular los símbolos poniéndose sobre ellos y pulsando el botón izquierdo del ratón y seguidamente el botón de *Delete*. Cuando estén todos los registros seleccionados pulsar el botón de *Close* y aparecerá una ventana, en este primer caso, *Watch_1*, como puede

verse en la Figura 38, en la que se ven los símbolos (etiquetas y variables) seleccionados

Address	Symbol	Value
10	RESULTADO	H'0F'
200	w	H'0F'
02	pc1	H'00'
0A	pc1ath	H'00'

Figura 38.- Ventana Watch_1

Para ver y cambiar las propiedades de un símbolo, hay que pulsar el botón de propiedades que aparece en el cuadro de diálogo de la Figura 37, al hacerlo aparece un nuevo cuadro de diálogo como el de la Figura 39

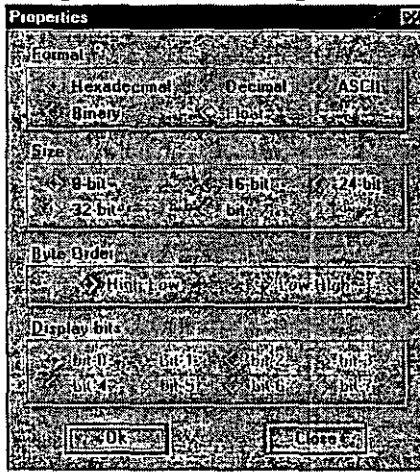


Figura 39.- Cuadro de diálogo de las propiedades de los datos de las ventanas temporales

El contenido de la ventana del reloj puede desplegarse mostrando o sin mostrar los números de la línea. Para elegir el formato deseado se hace a través del menú del sistema (pulsando el icono de la parte superior izquierda de la ventana). El menú del sistema también se usa para revisar la información en la ventana temporal.

La ventana de diálogo permite seleccionar el formato en que se presentan los símbolos:

- Format.-** Determina qué tipo de numeración se desea visualizar.
- Size.-** Determina cuántos bytes serán incluidos en la visualización del número:

hexadecimal, decimal, binario, ASCII o float.

•**Byte Order.-** Determina el orden de visualización de cada byte, disponible sólo para los números de 16 bits.

•**Display Bits.-** Determina en qué momento se visualiza el bit seleccionado al activarse.

1.12.- Modify: Al activar la opción **Window>Modify...** aparece el cuadro de diálogo **Modify** como el que se muestra en la Figura 40. En este cuadro se permite leer y escribir una posición de memoria o el rango de una posición de memoria. **Modify** puede trabajar en las áreas de memoria siguientes:

- Data
- Stack
- Program
- EEPROM (Si tiene)

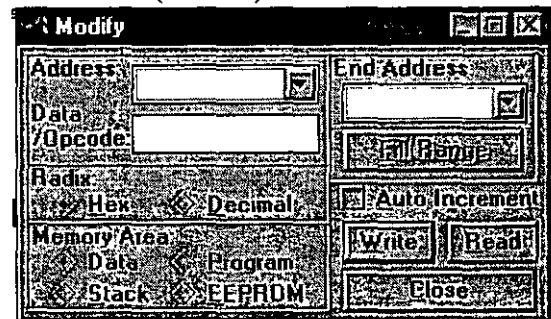


Figura 40.- Cuadro de diálogo de la opción Modify

Como resumen a todo lo que hemos contado hasta el este momento, podemos decir que el comando **Windows**, puede presentar una visión de todos los registros del microcontrolador en cada momento y podemos tener al final una pantalla en la que visualicemos según nos interese las ventanas mas adecuadas para el seguimiento de nuestra aplicación, como puede ser la de la Figura 41.

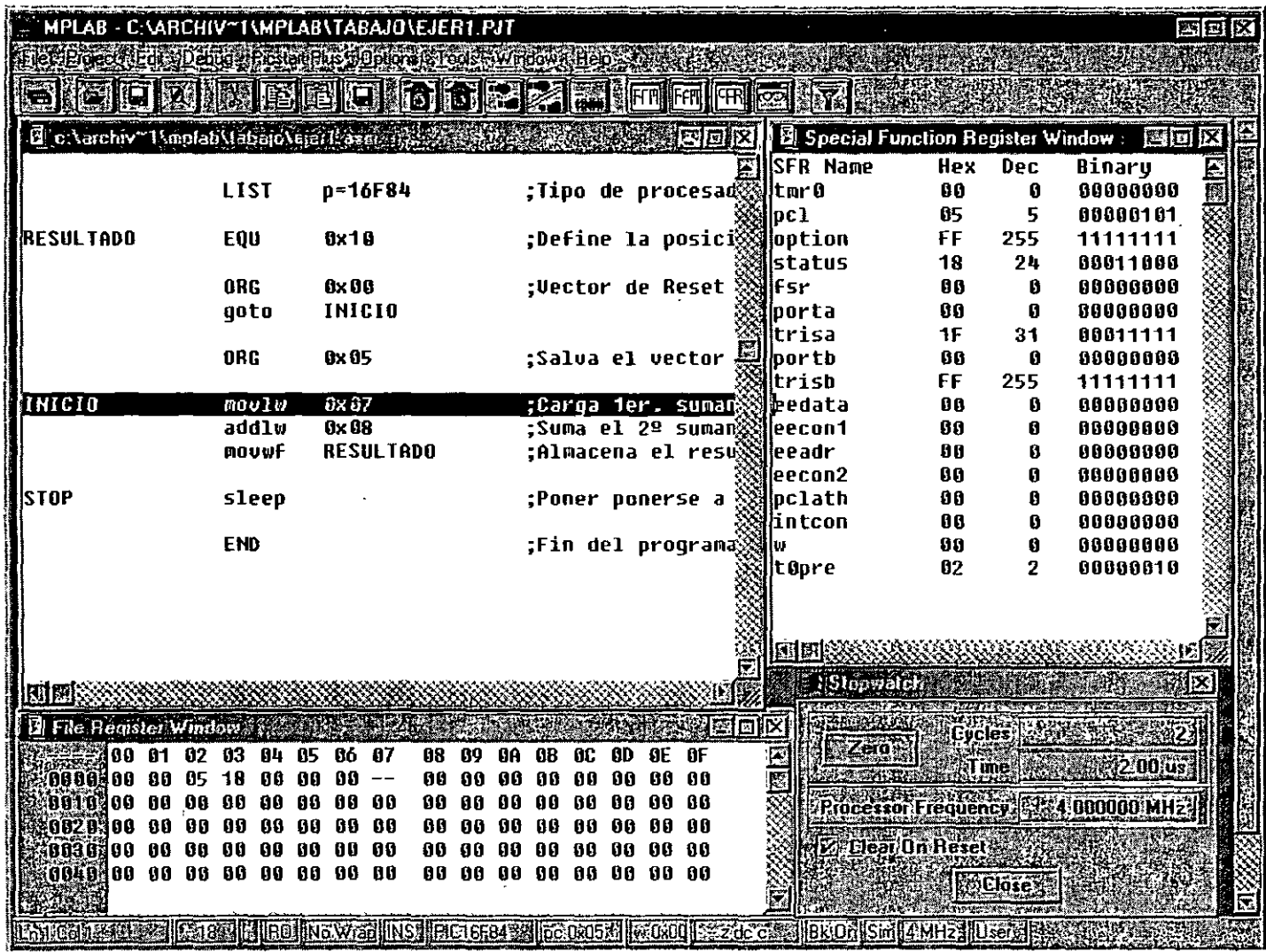


Figura 41.- Presentación de algunas ventanas de forma simultánea en pantalla



**FACULTAD DE INGENIERÍA UNAM
DIVISIÓN DE EDUCACIÓN CONTINUA**



Departamento de Cursos Abiertos

**DIPLOMADO EN
MICROCONTROLADORES (SISTEMAS
EMBEBIDOS)**

**CA 207 MÓDULO I
MICROCONTROLADORES 8 BITS**

TEMA:

ENSAMBLADOR IASM11

INSTRUCTOR: ING. DANIEL MARTÍNEZ GUZMÁN

**DEL 27 DE SEPTIEMBRE AL 1° DE OCTUBRE DE 2004
PALACIO DE MINERÍA**



DIVISIÓN DE
EDUCACIÓN
CONTINUA



APENDICE B ENSAMBLADOR IASM11

Para ensamblar un programa a ser ejecutado por el microcontrolador HC11 en cualquier versión de este; se pueden recurrir a dos maneras distintas:

- a) Editar el programa en lenguaje ensamblador, en el editor de texto de su preferencia y guardarlo en su disco de trabajo con extensión ASM o ASC.

Una vez hecho lo anterior, desde el sistema operativo se pueden emplear los ensambladores ASMHC11, IASM11 o AS11; con el siguiente formato:

AS11 Nombre_programa.ASM/ASC

Si no existe ningún error, se genera el programa en lenguaje de máquina con extensión S19.

- b) Empleando el programa ensamblador IASM11

Este programa contiene internamente la posibilidad de editar, ensamblar y poder realizar comunicación con el microcontrolador, tiene un ambiente accesible para su manejo; el despliegue que se presenta es muy similar a un editor de texto, como se muestra en la figura 1.

NOMANE.ASM	Line 1	Col 1	Byte 0	Insert	Indent
F1-Help F2-Save F3-Load F4-Assemble F5-Exit F7-Comm F9-DOS shell F10-Menu					

Figura 1.

Las teclas de función se describen a continuación;

- F1 Pasa al sistema de ayuda.
- F2 Salvar el programa que se está editando actualmente.
- F3 Cargar un programa.
- F4 Ensamblar el programa actual .
- F5 Salir del ensamblador.
- F7 Cargar y ejecutar un programa previamente ensamblado.
- F9 Shell al sistema operativo.

F10 Pasar a otro sistema de comandos; los comandos que se obtendrán al seleccionar esta función son los siguientes:

Edit	File	Assemble	Communicate	Help
------	------	----------	-------------	------

Edit Regresa al modo de edición

File Manejo de archivos

Assemble Se configura el formato del programa que se obtendrá al seleccionar la función de ensamblar; las posibles opciones con las que cuenta este comando son:

Assemble	F4
Object	off
Listing	off
Debug Map	off
Cycle cntr	off
Macros	Hide
Includes	Hide

Communicate Sirve para configurar los parámetros de comunicación, como son la velocidad de transmisión, la longitud de palabra, bits de inicio y de paridad.

Help Comando para invocar la ayuda.

Para invocar a los diferentes comandos, se puede realizar tecleando la letra mayúscula que corresponde al comando o llegar a él con las flechas del cursor.

Formato de la instrucción:

Columna

No. 1



ETIQUETA INSTRUCCIÓN OPERANDOS ;COMENTARIOS

Es importante que las etiquetas inicien en la columna número uno y las instrucciones propias del ensamblador en cualquier posición.

El procedimiento para ensamblar un programa es el siguiente:

- a) Se edita el programa;
- b) Con la tecla F10, se pasa a otro sistema de comandos;

- c) Seleccionar el comando de ensamblar;
- d) Configurar el formato del o de los programas que se requieran; en el **Objeto** seleccionar con ENTER el programa **S19**;
- e) Con la tecla ESC se sale de ese comando hasta regresar a la pantalla principal;
- f) Se presiona la tecla F4 para ensamblar el programa;
- g) Si no existió ningún error, se generan los programas seleccionados en el submenú de ensamblado.

Con el programa S19 se puede pasar a simular el funcionamiento del programa, en su caso, con la seguridad de su correcta operación, se puede bajar y ejecutar directamente en el microcontrolador.

APENDICE C.
SIMULADOR AVSIM11

Con el programa AVSIM11 se pueden simular los programas realizados para los microcontroladores de la familia 68HC11 en sus versiones A8, A0 y A2, tanto en modo single chip como en modo expandido; se pueden simular con otras versiones del HC11, pero considerando algunas restricciones, como son la cantidad de puertos que se tienen disponibles, así como las direcciones asignadas para ellos en el mapa de memoria del microcontrolador.

El primer desplegado que presenta este programa, nos sirve para seleccionar la versión del microcontrolador, con el cuál se simulará un programa previamente editado y ensamblado; tal como se muestra en la figura 1.

AVSIM11 Simulator/Debugger

Serial #KA06137 Licensed by Avocet Systems, Inc.

Copyright (C) 1987,1988 by Ken Anderson Software
All Rights Reserved

Microcomputer Configurations

Single-Chip Mode

A:68HC11A8

B:68HC11A0 (no EEPROM)

C:68HC811A2

Expanded Mode

D:68HC11A8 + 68HC24 PRU

E:68HC11A0 + 68HC24 PRU

F:68HC811A2 + 68HC24 PRU

Choose a Configuration for simulation:

Figura 1.

Una vez aceptada la opción del microcontrolador, se desplegará una pantalla similar a la que presenta en la figura 2, misma en la que se pueden identificar los siguientes elementos:

- a) En la parte izquierda (LABEL OPERATION), es donde se ubicará el código del programa a simular; es decir indicará la dirección asignada para cada instrucción y su mnemónico.
- b) En la parte superior se ubican los acumuladores A y B, indicados por la etiqueta CPU REGISTERS.
- c) A la derecha de la posición de los registros, se ubica el registro de banderas (FLAGS), identificando las banderas SHNZVIX; la bandera de acarreo © se encuentra del lado izquierdo de los acumuladores.
- d) A un costado derecho del registro de banderas se encuentran los indicadores del modo de simulación, como la velocidad y otros parámetros.

- e) Debajo de los acumuladores se encuentran el Stack Pointer (SP) y el Program Counter (PC), los cuales se irán modificando de acuerdo a los valores actuales, de igual forma se identifican los registros de Índice Y y X.
- f) Se dispone de dos ventanas para observar e introducir valores, indicados como Memory Space; es posible seleccionar cualesquiera de las ellas, así como la dirección donde se desea que inicie el desplegado.
- g) Del lado derecho en su parte media se encuentran los registros asociados a los puertos seriales (SCI y SPI).
- h) En la parte inferior derecha, se encuentran los puertos paralelos A, B, C, D y E; lugar donde se podrá visualizar el valor actual que contienen estos puertos.

```

LABEL OPERATION      68HC11A8  AVSIM11 Simulator/Debugger          VI.50
$0000 ERROR          CPU REGISTERS      FLAGS      SCL SPD DSP SKP CURSOR
$0001 ERROR          C A:ACCUMULATOR  SHNZV IX   OFF HI ON OFF      MENU
$0002 ERROR          0 00000000:00:q      1000 11    PINS           Cycles:
$0003 ERROR          B: 00000000:00:      COP:0000   IXAB           SPI SPDR:00:_
$0004 ERROR          addr      data RTI:0000      1100           TDR:00:_
$0005 ERROR          PC:0000 »    00 00 00 00 00 00 00 00 00           SPSR:00000000
$0006 ERROR          SP:0000 »    00 00 00 00 00 00 00 00 00           SCI SCDR:00:_
$0007 ERROR          00 00 00 00 00 00 00 00 00           TDR:00:_*00:_
$0008 ERROR          X :0000 »    00 00 00 00 00 00 00 00 00           SCCR2:00000000
$0009 ERROR          Y :0000 »    00 00 00 00 00 00 00 00 00           SCSR:11000000
$000A ERROR          TMR:0000 I:0000 0000 0000      M1:00000000   M2:0000      F2:0000
$000B ERROR          PA:00      O:FFFF FFFF FFFF  F1:00000000   PORTS          ddr
$000C ERROR          Memory Space          A           IOOOOIII
$000D ERROR          1000 00 00 03 00 00 00 00 00           00:           :00000000
$000E ERROR          1008 00 00 00 00 00 00 00 00           B :           OOOOOOOO
$000F ERROR          1010 00 00 00 00 00 00 FF FF           00:           :00000000
$0010 ERROR          1018 FF FF FF FF FF FF FF FF           C           I I I I I I I I
$0011 ERROR          Memory Space          00:           :00000000
$0012 ERROR          1020 00 00 00 00 00 00 00 00           D           I I I I I I
$0013 ERROR          1028 04 00 00 00 00 00 C0 00           00:           : 000000
$0014 ERROR          1030 00 00 00 00 00 00 00 00           E           I I I I I I I I
$0015 ERROR          1038 00 10 00 00 05 01 00 0F           00:           :00000000

>Select Command - or use arrow keys
Dump      Expression  commandFile      Help IO      Load      --space-- ESC to screen

```

Figura 2.

El total de comandos con los cuales se puede trabajar son los siguientes

Dump	Expression	commandFile	Help	Io	Load
Memory	Patch Quit	Reset	Set	View	eXecute

Cada uno de estos comandos contienen generalmente otro sistema de subcomandos; existen dos formas de acceder a ellos, con la letra mayúscula asociada al comando o con las flechas del cursor seguido de la tecla enter.

El procedimiento para simular un programa es el siguiente.

Una vez con el programa ensamblado se ingresa al simulador seleccionando la versión del microcontrolador a su elección.

1. Cargar el programa

Loads - Program Enter filename: **Nombre_programa.S19**

Introducir el nombre de su programa, es necesario indicar la extensión.

El programa se visualizará a partir de la dirección de inicio, esta dirección se indica en el contador de programa PC.

2. Selección de la ventana para visualizar y en su caso introducir los datos

Dump Area: 1 2 (1 la ventana superior, 2 ventana inferior)
 seleccionar en el siguiente comando Absolute
 Enter Expression in Memory Address Space: **\$Direc_Inicio**
 Se indica a la dirección donde se comenzará a verse la ventana.

3. Configuración de la ventana seleccionada para que trabaje como memoria RAM

Set Memory-map r a m d o m _ A c c e s s
 Enter Expression in Memory Address Space: **\$Direc_Inicio**
 Enter Expression in Memory Address Space: **\$Direc_Final**

En esta parte se indica la dirección inicial y la final para configurar el rango de memoria RAM.

4. Ejecución de la simulación

- F1** Ejecución del programa en forma continua
- F10** Ejecución de una sola instrucción
- F9** Ejecución por pasos hacia atrás --

5. Salir del simulador

Quit Exit

El simulador cuenta con un sistema de ayuda, para invocar se selecciona el comando Help.



**FACULTAD DE INGENIERÍA UNAM
DIVISIÓN DE EDUCACIÓN CONTINUA**



**DIPLOMADO EN
MICROCONTROLADORES (SISTEMAS
EMBEBIDOS)**

**CA 207 MÓDULO I
MICROCONTROLADORES 8 BITS**

TEMA:

MICROCONTROLADOR MC68HC11

INSTRUCTOR: ING. DANIEL MARTÍNEZ GUZMÁN

**DEL 27 DE SEPTIEMBRE AL 1° DE OCTUBRE DE 2004
PALACIO DE MINERÍA**



DIVISIÓN DE
EDUCACIÓN
CONTINUA



Programa 2004

APENDICE A **MICROCONTROLADOR MC68HC11**

Se consideran para el curso la posible utilización de las versiones de esta familia conocidas como E9 y F1, por lo que en el caso de existir alguna diferencia se mencionarán en su oportunidad.

Características generales MC68HC11E9

El MC68HC11E9 es un semiconductor de alta densidad de óxido de metal complementario (HCMOS), unidad microcontroladora (MCU). Esta MCU tiene un voltaje de operación bajo, de alta velocidad, con un bus multiplexado de una velocidad nominal de 2MHZ.

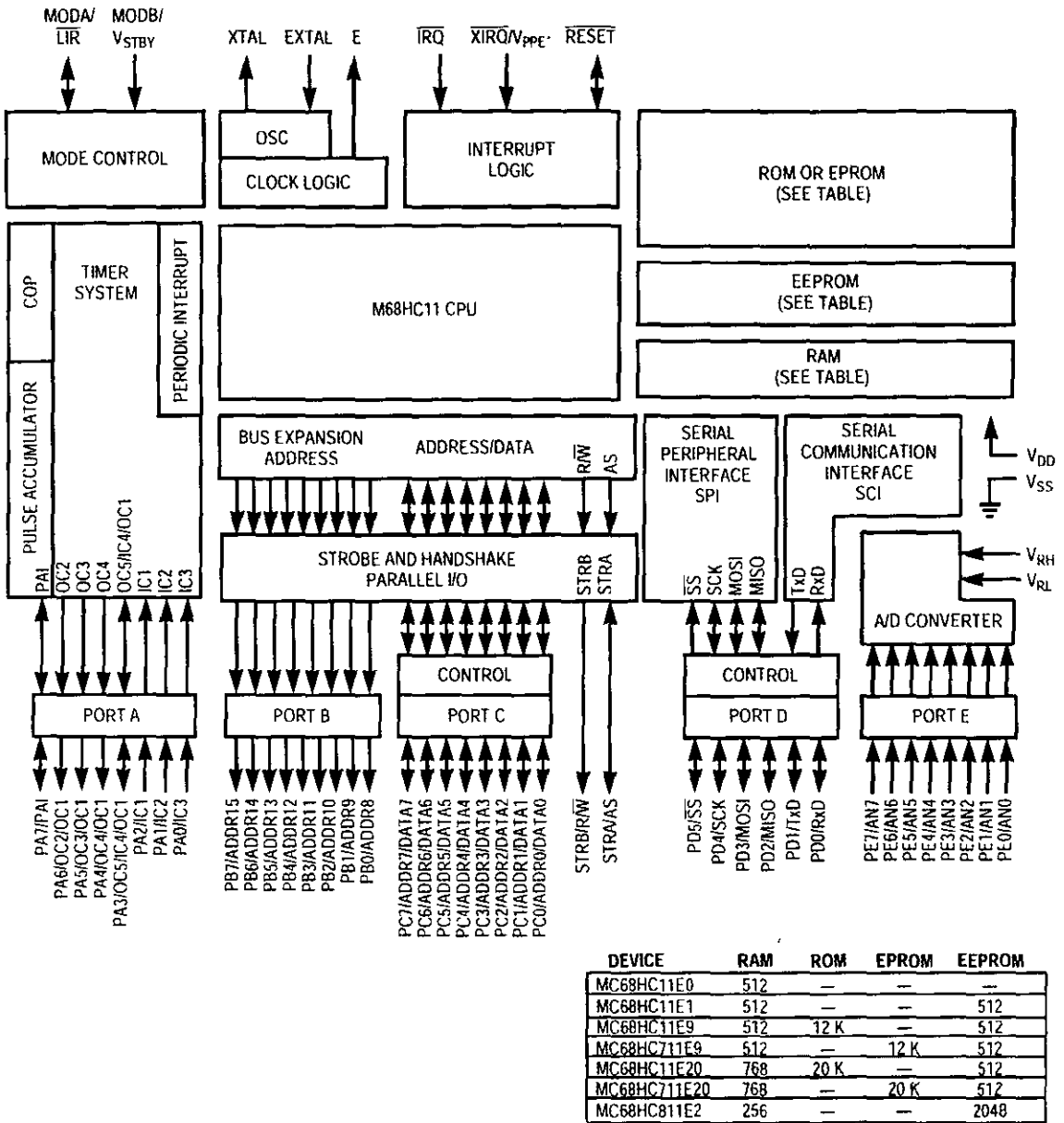
Las características especiales de este microcontrolador son:

- ◆ Cinco puertos paralelos, con doble función cada uno de ellos (A,B,C,D,E)
- ◆ Expansión de sistemas de tiempo, con cuatro etapas preescalables
- ◆ Se resalta el no retorno a cero (NRZ)
- ◆ Interface de Comunicación Serial (SCI)
- ◆ Convertidor Analógico Digital de ocho canales, con ocho bits de resolución
- ◆ Bloque protector de mecanismo para EEPROM y CONFIG
- ◆ Bus multiplexado
- ◆ Empaquetado de 58 pines
- ◆ Tope para el ahorro de energía y modos de espera
- ◆ 64 Kbytes de memoria direccionable
- ◆ Interface de Comunicación con Periféricos (SPI)
- ◆ 512 Bytes de EEPROM
- ◆ 512 Bytes de RAM
- ◆ 12 Kbytes de ROM, con el Buffalo programado
- ◆ Circuito de Interrupción real
- ◆ Acumulador de pulsos
- ◆ Captura de entrada
- ◆ Comparación de salida

Características generales MC68HC11F1

- ◆ 512 Bytes de EEPROM
- ◆ 1024 Bytes de RAM interna
- ◆ Sistema temporizador
 - ◆ Tres funciones de captura de entrada (IC)
 - ◆ Cuatro funciones de comparación de salida (OC)
 - ◆ Cuarta o quinta función de IC y OC respectivamente, seleccionada por software
- ◆ Función de chip selects
- ◆ Circuito de interrupción en tiempo real
- ◆ Acumulador de pulsos de ocho bits
- ◆ Interfaz serial con periféricos (SPI)
- ◆ Interfaz de comunicación serial (SCI)
- ◆ Convertidor Analógico Digital
 - ◆ Ocho canales
 - ◆ Ocho bits de resolución

General Description



* V_{PPE} applies only to devices with EPROM/OTPROM.

Figure 1-1. M68HC11 E-Series Block Diagram

- ◆ Bus de datos y direcciones no multiplexado
- ◆ Encapsulado PLCC de 68 pines
- ◆ Siete puertos paralelos con doble función cada uno de ellos (A, B, C, D, E, F, G)

Modos de operación de HC11

Emplea dos pines, para seleccionar el modo de operación, el MODA y el MODB, básicamente existen dos modos de operación normales, que son el single chip y el expandido, y dos modos de operación especiales, el bootstrap y el test. La selección del modo de operación es de acuerdo a los valores codificados en estos pines.

MODA	MODB	Modo seleccionado
0	1	Normal Single Chip
1	1	Normal Expandido Multiplexado
0	0	Especial Bootstrap
1	0	Especial Test

Single Chip

Solo se dispone de la memoria interna del microcontrolador. Los puertos B y C, así como las terminales STRA y STRB están disponibles como entrada y salida paralelas de propósito general; todo el software necesario para controlar el MCU está contenido en los recursos internos.

Expandido

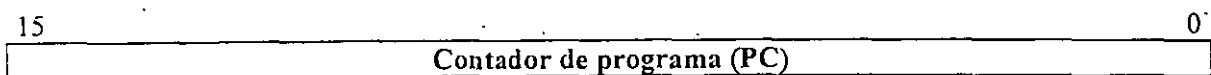
En el modo de operación expandido, permite acceder a la memoria externa dentro de la totalidad de los 64 Kbytes de espacio direccionable, el espacio incluye al espacio de memoria interna del modo single chip, los puertos B y C se emplean para realizar la expansión de memoria, el puerto B formará la parte alta del bus de direcciones y el puerto C tendrá la función multiplexada para la parte baja del bus de direcciones y el bus de datos, además incluye las señales AS, E y R/W.

Test

Permite el acceso privilegiado a recursos internos del MCU, este es una variación del modo expandido y es normalmente empleado para pruebas internas de producción por el fabricante.

Bootstrap

Es una variación especial del modo single chip, este permite dar entrada a programas en la RAM interna. Al seleccionarse este modo de operación, durante el restablecimiento una pequeña ROM de bootstrap se hace presente en el mapa de memoria. Esta contiene un pequeño programa el cuál inicializa la interface serial asincrona de comunicación (SCI), lo cuál permite al usuario cargar programas dentro de la RAM interna, finalizando esto, el control lo tendrá el programa recién cargado. Este es el modo de operación en el cual se trabaja con mayor frecuencia.



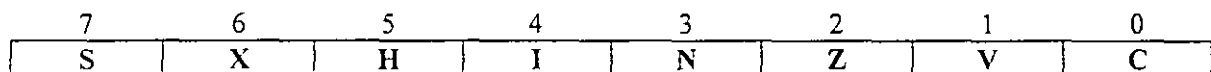
Apuntador a la pila (SP)

El stack pointer SP es un registro de 16 bits que contiene la dirección de la siguiente ubicación libre en el stack. El stack es configurado como una secuencia de últimos en entrar, primeros en salir (LIFO), lee registros de escritura que permiten datos importantes para ser almacenados durante interrupciones y llamados a subrutinas.



Registro de banderas (CCR)

El registro de banderas es un registro de 8 bits, en el cuál cinco bits son utilizados para indicar los resultados de la instrucción recién ejecutada, dos bits mascarables de interrupciones y un bit de paro. Estas banderas pueden ser chequeadas individualmente por el programa y realizar una tarea específica como resultado de su estado; cada bit se explica a continuación.



(C) Acarreo (Carry/Borrow)

El bit C indica si en la última operación realizada, ya sea lógica o aritmética existió un acarreo, esta bandera actúa también como una bandera de error para las operaciones de multiplicación y división, así mismo se afecta durante las operaciones de rotación de los registros.

(V) Desbordamiento (Overflow)

El bit V indica si ocurrió un sobreflujo como resultado de una operación.

(Z) Cero (Zero)

Se fija cuando el resultado de la última operación lógica, aritmética o manipulación de datos es cero.

(N) Negativo (Negative)

El bit N indica si el resultado de la última operación aritmética, lógica o manipulación de datos es negativo, es decir, refleja el estado del bit más significativo (MSB) de un resultado. Un número es positivo cuando el MSB es cero y es negativo cuando MSB es uno.

(H) Medio Acarreo (Half Carry)

Indica el acarreo existente en una operación aritmética entre los bits tercero y cuarto; es de utilidad cuando se desea hacer el ajuste a decimal del resultado de una suma.

(I) Interrupción de Máscara

Este bit puede ser activado por software y hardware para habilitar todas las fuentes de interrupción de máscara.

(X) Interrupción de Máscara

El bit X se activa solamente por hardware (RESET o XIRQ) y limpiado solamente por el programa de instrucciones de transferencia de A hasta CCR o retornando de interrupciones (RTI).

(S) Inhabilitador de Paro

Este bit permite habilitar o deshabilitar la instrucción de stop.

Modos de Direccionamiento

El MCU cuenta con seis modos de direccionamiento, inmediato, directo, extendido, indexado, inherente y relativo, que son utilizados para obtener o almacenar un dato, de o hacia alguna localidad de memoria.

Modo de Direccionamiento Inmediato (INM)

En el modo de direccionamiento inmediato, el argumento actual está contenido en el byte(s) que siguen a la instrucción.

Formato: Instrucción #Dato

Ejemplo:

LDAA #SA0 ; Carga el dato SA0 en hexadecimal al acumulador A
LDAB #%01100011 ; Carga el dato binario indicado al acumulador B

Modo de Direccionamiento Directo (DIR)

En el modo de direccionamiento directo, el byte menos significativo de la dirección efectiva de la instrucción aparecerá en el byte siguiente al opcode. El byte más significativo de la dirección efectiva es tomado como \$00. Este modo de direccionamiento hace referencia al espacio de memoria comprendido entre \$0000 y \$00FF, es conocido como direccionamiento página cero.

Formato: Instrucción \$00Dirección

Ejemplo:

LDAA #SA0 ; Carga el dato indicado al acumulador A
STAA \$50 ; Envía el contenido de A a la dirección \$50

Modo de Direccionamiento Extendido (EXT)

La dirección efectiva de la instrucción aparece explícitamente en los dos bytes siguientes al opcode. Por lo tanto se puede hacer referencia al área total del microcontrolador es decir de la dirección \$0000 a la \$FFFF.

Formato: Instrucción \$Dirección

Ejemplo:

LDAA #\$01FF ;Carga el dato indicado al acumulador A
STAA \$1004 ;Envía el contenido del acumulador A en la dirección \$1004

Modo de Direccionamiento Indexado (INDX, INDY)

Los registros de índice X o Y son empleados para hacer referencia a la dirección efectiva donde se obtendrá o almacenará el dato. La dirección efectiva es variable y depende del contenido actual del registro de índice a emplear y de un OFFSET sin signo, contenido en la instrucción.

Formato: Instrucción \$Offset,Reg_índice

Ejemplo:

LDX #\$1000 ;Inicializa el registro de índice X
LDAB \$31,X ;Carga el contenido de la dirección \$1031 en B

Modo de Direccionamiento Inherente (INH)

En este modo de direccionamiento, toda la información necesaria para ejecutar una instrucción, está contenida en el código de operación.

Formato: Instrucción

Ejemplo:

INX ;Incrementa el contenido del registro X
ABA ;Suma el contenido de los acumuladores A y B

Modo de Direccionamiento Relativo (REL)

Su utilidad se encuentra en los saltos o brincos relativos, comúnmente ejecutados después de una comparación o lectura del CCR; para lo cual es posible la toma de decisiones de bifurcación. Las instrucciones de bifurcación generan dos bytes, uno para el opcode y el otro para el offset relativo. Si la condición de bifurcación es verdadera, el contenido de los 8 bits del byte siguiente al opcode (offset) son sumados al contenido del contador de programa para formar la dirección efectiva de la bifurcación, de otra manera, el control pasa a la instrucción siguiente a esta.

Formato: Instrucción Etiqueta

Ejemplo:

LDAA \$100A ;Carga en A el contenido de la dirección \$100A
CMPA #\$F0 ;Compara A con el dato #\$F0
BEQ igual ;Si Z=1 brinca a igual
BNE diferente ;Si no es igual brinca a diferente

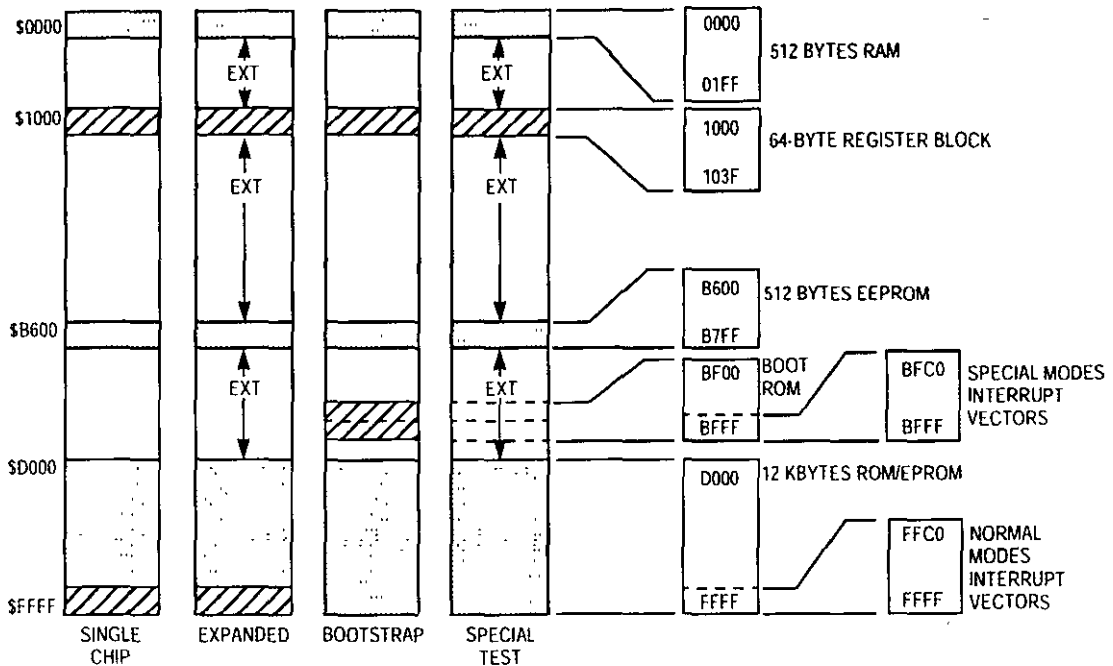
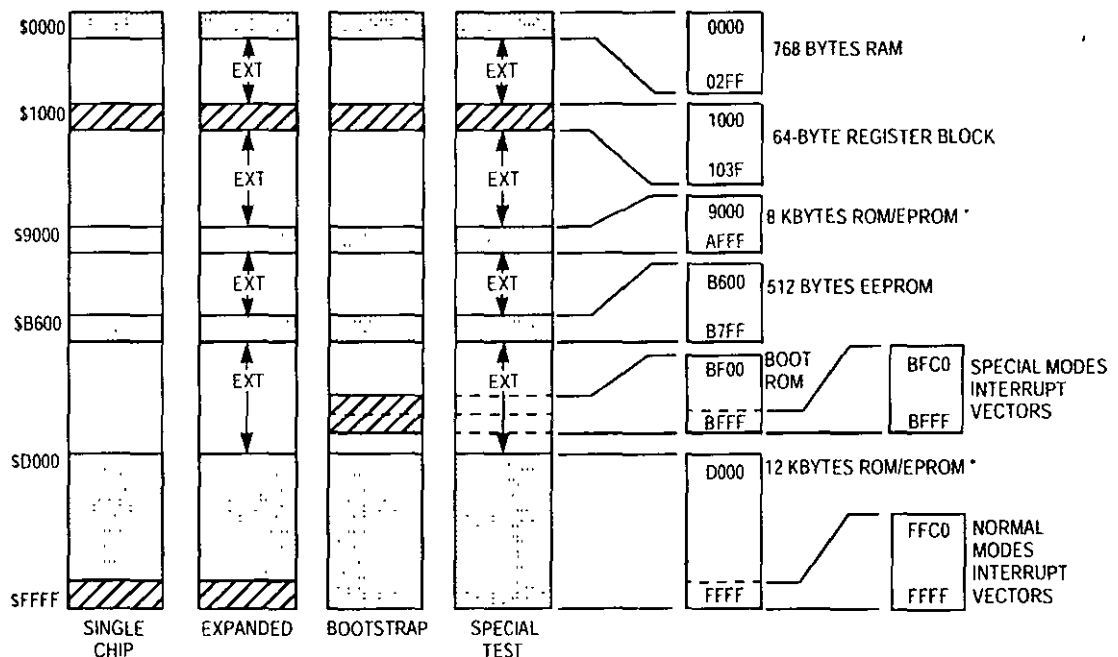


Figure 4-4. Memory Map for MC68HC(7)11E9



* 20 Kbytes ROM/EPROM are contained in two segments of 8 Kbytes and 12 Kbytes each.

Figure 4-5. Memory Map for MC68HC(7)11E20

Puertos Paralelos

En lo que se refiere al sistema de puertos paralelo existen diferencias sustanciales entre las diferentes versiones del HC11,, en el caso concreto de las versiones E9 se cuentan con cinco puertos paralelos y en la versión F1 se disponen de siete puertos paralelos, cuyas características se describen a continuación.

Puertos Paralelos Microcontrolador MC68HC11E9

En la serie E del microcontrolador MC68HC11, se disponen de 5 puertos paralelos, y un total de 38 pines de entrada y salida; a continuación se presentan estos:

Puerto	Pines de entrada	Pines de salida	Pines Bidireccionales	Funcion alterna
Puerto A	3	3	2	Temporizador
Puerto B	-	8	-	Parte alta del bus de direcciones
Puerto C	-	-	8	Parte baja del bu de direcciones/bus de datos
Puerto D	-	-	6	SCI y SPI
Puerto E	8	-	-	Convertidor A/D

Puerto A

Al puerto A no le afecta el modo de operación, viene configurado para esta versión del microcontrolador de la siguiente manera, tres pines de salida, tres de entrada y dos que se pueden seleccionar como entrada o salida.

PORTA Datos de Puerto A \$1000

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0
PAI	OC2	OC3	OC4	IC4/OC5	IC1	IC2	IC3

Puerto B

En single chip y bootstrap todos los pines del puerto B son salidas de propósito general; en el modo expandido y test, forman la parte alta del bus de direcciones.

PORTB Datos del Puerto B \$1004

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
ADR15	ADR14	ADR13	ADR12	ADR11	ADR10	ADR9	ADR8

Puerto C

En single chip y bootstrap, el puerto C se puede configurar para que trabajen sus pines como salida o entrada, dependiendo de la selección realizada en el registro DDRD. En el modo expandido y test, el puerto C tiene la función multiplexada para la parte baja del bus de direcciones y el bus de datos.

DDRC Registro de Dirección de Datos del Puerto C \$1007

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0

DDC(7:0) Dirección de datos para el puerto C

0 = Bit seleccionado como entrada

1 = Bit seleccionado como salida

PORTC Datos del puerto C \$1003

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0

ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0
DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0

Puerto D

Al puerto D o le afecta el modo de operación, pero únicamente se dispone de 6 bits de propósito general de entrada o salida, seleccionados por la configuración del registro DDRD; tiene su doble función con los subsistemas SCI y SPI.

DDRD Registro de Dirección de Datos del puerto D \$1009

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	-	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0

PORTD Datos del puerto D \$1008

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		PD5	PD4	PD3	PD2	PD1	PD0

PD5	PD4	PD3	PD2	PD1	PD0
SS	SCK	MOSI	MISO	TxD	RxD

Puerto E

El puerto E es de 8 bits de entrada general, no le afecta el modo de operación, comparte su función con el convertidor analógico digital.

PORTE				Datos del Puerto E				\$100A							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0

Puertos Paralelos Microcontrolador MC68HC11F1

En la serie F1 del microcontrolador MC68HC11, se disponen de 7 puertos paralelos, y un total de 54 pines de entrada y salida; a continuación se presentan estos:

Puerto	Pines de entrada	Pines de salida	Pines Bidireccionales	Función alternativa
Puerto A	-	-	8	Temporizador
Puerto B	-	8	-	Parte alta del bus de direcciones
Puerto C	-	-	8	Bus de datos
Puerto D	-	-	6	SCL y SPI
Puerto E	8	-	-	Convertidor Analógico/Digital
Puerto F	-	8	-	Parte baja del bus de direcciones
Puerto G	-	-	8	Chipselects

Puerto A

Al puerto A no le afecta el modo de operación, se configura en sus ocho bits como se desea trabajar en este puerto.

PORTA				Datos de Puerto A				\$1000							
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PA7	PA6	PA5	PA4	PA3	PA2	PA1	PA0	PAI	OC2	OC3	OC4	IC4/OC5	IC1	IC2	IC3

DDRA Registro de Datos del Puerto A \$1001

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DDA7	DDA6	DDA5	DDA4	DDA3	DDA2	DDA1	DDA0

DDA(7:0) Dirección de datos para el puerto C

- 0 = Bit seleccionado como entrada
- 1 = Bit seleccionado como salida

Puerto B

En single chip y bootstrap todos los pines del puerto B son salidas de propósito general; en el modo expandido y test, forman la parte alta del bus de direcciones.

PORTB	Datos del Puerto B	\$1004					
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PB7	PB6	PB5	PB4	PB3	PB2	PB1	PB0
ADR15	ADR14	ADR13	ADR12	ADR11	ADR10	ADR9	ADR8

Puerto C

En single chip y bootstrap, el puerto C se puede configurar para que trabajen sus pines como salida o entrada, dependiendo de la selección realizada en el registro DDRC. En el modo expandido y test, el puerto C realizará la función del bus de datos.

DDRC	Registro de Dirección de Datos del Puerto C	\$1007					
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DDC7	DDC6	DDC5	DDC4	DDC3	DDC2	DDC1	DDC0

DDC(7:0) Dirección de datos para el puerto C

- 0 = Bit seleccionado como entrada
- 1 = Bit seleccionado como salida

PORTC	Datos del puerto C	\$1006					
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PC7	PC6	PC5	PC4	PC3	PC2	PC1	PC0
ADR7	ADR6	ADR5	ADR4	ADR3	ADR2	ADR1	ADR0
DATA7	DATA6	DATA5	DATA4	DATA3	DATA2	DATA1	DATA0

Puerto D

Al puerto D o le afecta el modo de operación, pero únicamente se dispone de 6 bits de propósito general de entrada o salida, seleccionados por la configuración del registro DDRD; tiene su doble función con los subsistemas SCI y SPI.

DDRD	Registro de Dirección de Datos del puerto D	\$1009					
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
-	-	DDD5	DDD4	DDD3	DDD2	DDD1	DDD0

PORTD	Datos del puerto D	\$1008					
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
		PD5	PD4	PD3	PD2	PD1	PD0
		PD5	PD4	PD3	PD2	PD1	PD0
		SS	SCK	MOSI	MISO	TxD	RxD

Puerto E

El puerto E es de 8 bits de entrada general, no le afecta el modo de operación, comparte su función con el convertidor analógico digital.

PORTE	Datos del Puerto E	\$100A					
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PE7	PE6	PE5	PE4	PE3	PE2	PE1	PE0
AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0

Puerto F

En el modo de operación bootstrap los ocho bits trabajarán únicamente como salida, en modo expandido formará la parte baja del bus de direcciones (ADR15:8).

PORTF	Datos del Puerto F	\$1005					
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
PF7	PF6	PF5	PF4	PF3	PF2	PF1	PF0
ADR15	ADR14	ADR13	ADR12	ADR11	ADR10	ADR9	ADR8

Puerto G

En single chip y bootstrap, el puerto C se puede configurar para que trabajen sus pines como salida o entrada, dependiendo de la selección realizada en el registro DDRG. En el modo expandido y test, los pines (7:4) pueden ser usados para la función de chip select.

DDRG	Registro de Dirección de Datos del Puerto G	\$1003					
Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DDG7	DDG6	DDG5	DDG4	DDG3	DDG2	DDG1	DDG0

DDG(7:0) Dirección de datos para el puerto G

0 = Bit seleccionado como entrada

1 = Bit seleccionado como salida



**FACULTAD DE INGENIERÍA UNAM
DIVISIÓN DE EDUCACIÓN CONTINUA**



**DIPLOMADO EN
MICROCONTROLADORES (SISTEMAS
EMBEBIDOS)**

**CA 207 MÓDULO I
MICROCONTROLADORES 8 BITS**

TEMA:

DESCRIPCIÓN GENERAL

INSTRUCTOR: ING. DANIEL MARTÍNEZ GUZMÁN

**DEL 27 DE SEPTIEMBRE AL 1º DE OCTUBRE DE 2004
PALACIO DE MINERÍA**



DIVISION DE
EDUCACIÓN
CONTINUA



Programa 2004

APENDICE D LM18293/L293 MANEJADOR DE POTENCIA

Descripción general

El LM18293 es un circuito integrado diseñado para manejar motores hasta de 1 A. Entre las aplicaciones típicas, incluye manejo de cargas inductivas como solenoides, relevadores, motores de corriente directa y motores de paso, emplea internamente los transistores de potencia y utiliza un buffer para señales de nivel bajo.

En el circuito de aplicación., se presenta el patigrama de este dispositivo, contiene cuatro entradas para ingresar señales de control de los motores, acepta niveles estándares de lógica TTL y DTL, para realizar su interfaz; dos señales de habilitación para controlar la velocidad, que también acepta la misma lógica. Cada habilitador controla dos canales; cuando el pin de habilitación está desactivado (cero lógico), las salidas correspondientes se encontraran con lógica de tres estados; si el pin no está conectado (flotando), el circuito funcionará como si estuviera habilitado.

Se cuenta con dos pines para proporcionar el voltaje; el pin 8 proporciona la potencia del motor y el pin 16 suministra un voltaje independiente del anterior, que polariza los circuitos internos.

El chip está incluido en un diseño DIP de 16 pines, el dispositivo es capaz de operar con voltajes máximos de 36 volts.

La figura mostrara la forma de conectar dos motores y controlar al mismo tiempo el sentido de giro; en este caso podrán girar ambos en sentido horario y antihorario.

Características

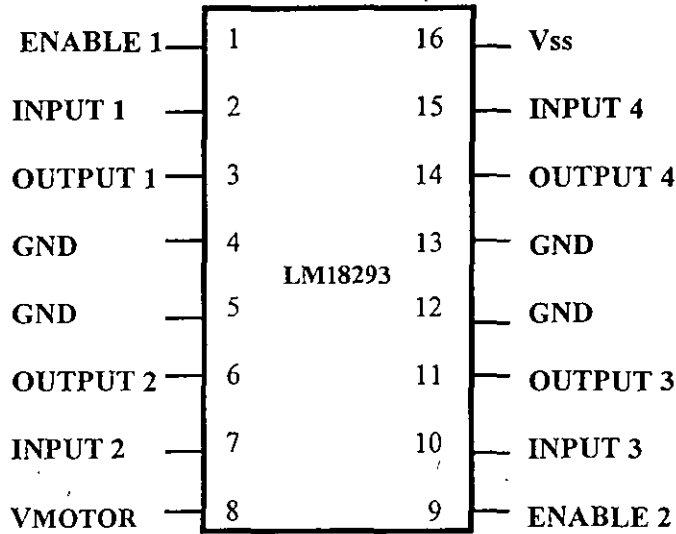
- Salida por canal de 1 Amper
- Reemplazo directo por el circuito integrado L293B y L293D
- Empaquetado DIP de 16 pines
- Protección térmica contra sobrecargas
- Cero lógico hasta 1.5 volts
- Alta inmunidad al ruido

Máximos rangos de voltaje

* Voltaje para las cargas	(Vs)	36 volts
* Voltaje de la fuente lógica	(Vss)	36 volts
* Voltaje de entrada	(Vi)	7 volts
* Habilitación de voltaje	(Ve)	7 volts
* Corriente de salida		2 amperes

Características eléctricas

$V_s = 24V$, $V_{ss} = 5V$, $T = 25^{\circ}C$, $L = 0.4V$, $H = 3.5 V$.



Asignación de pines.

Control de los motores de corriente directa

Para controlar los motores se utiliza el circuito integrado L293D, que como se describió anteriormente, nos presenta la capacidad de controlar los dos motores.

Como se muestra en la figura D-1, y de acuerdo a las características de este circuito, se requieren de cuando menos cuatro señales de control, las cuales serán otorgadas por el microcontrolador. Para controlar un motor se requiere de una señal que entregará el comando de dirección del motor, es decir; hacia donde deseamos que gire (derecha o izquierda), y otra señal que nos proporcionará la velocidad de giro del motor. Para el control del otro motor, se necesita de la misma cantidad de señales.

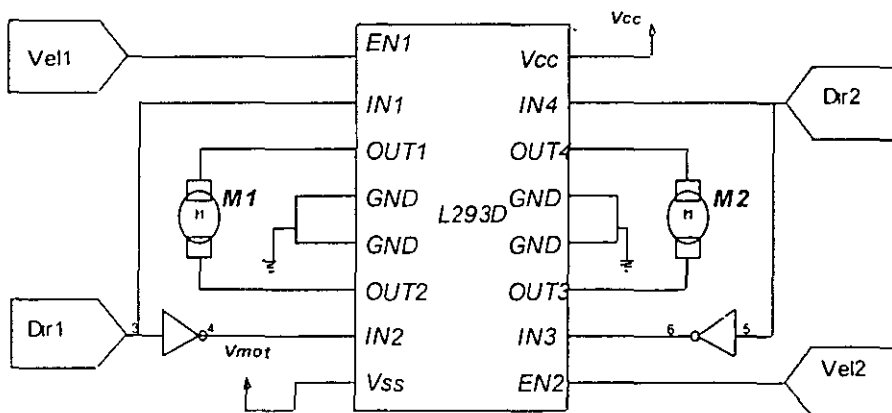


Figura D-1. Circuito de control de los motores de CD.

El circuito de la figura D-2, muestra el agregado de una etapa de acoplamiento y aislamiento, entre las señales que provienen del microcontrolador y la etapa de control de los motores; para tener este tipo de control se requieren de seis señales, cuatro para la dirección y dos para la velocidad de giro de los motores.

Así mismo, otra ventaja que presenta este circuito es contar con un pin asignado exclusivamente para suministrar el voltaje de polarización de los motores, el cuál puede tener un valor entre 0.2 hasta 32 volts, como se indica en la hoja de especificaciones del L293D en el apéndice C.

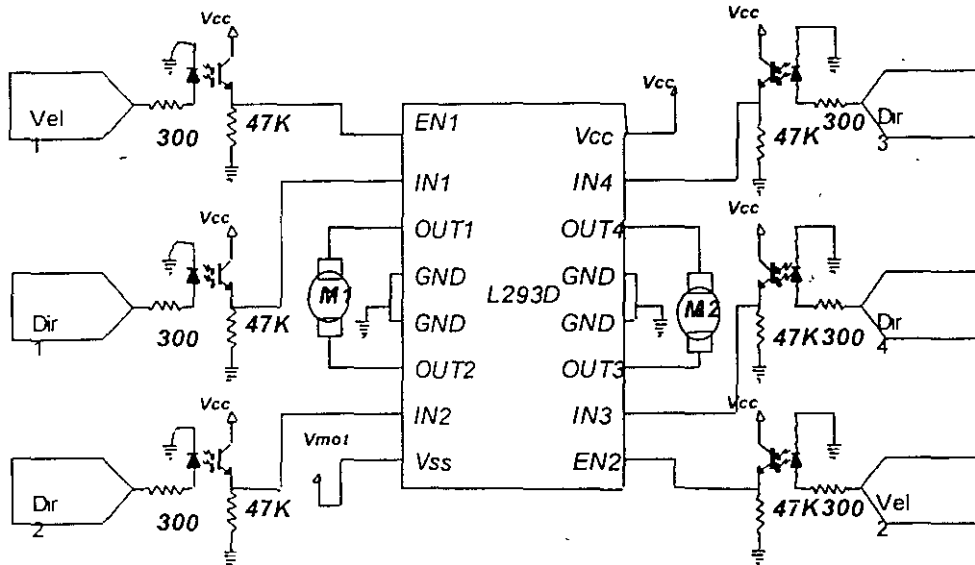


Figura D-2. Circuito de control de los motores empleando optoacopladores.

Es posible utilizar únicamente cuatro señales para controlar los motores , esto es un pin para controlar la dirección del motor, por lo tanto una señal deberá estar en alto y la otra en bajo, conectando a la salida un inversor (7404) se obtendrá lo anterior, la otra señal controlara el habilitador y lo mismo sucederá con el otro motor.