Práctica Número 1:

"Conceptos del Microcontrolador MC9S08JM60 (NXP-Freescale) y su Entorno de Programación CodeWarrior"

Objetivo

Comprender los elementos fundamentales del microcontrolador y su entorno de programación.

Material y equipo para la práctica

1 Multímetro.
1 PC.
1 Tarjeta DEMOJM60.
1 Microcontrolador MC9S08JM60.

Cuestionario Preliminar

1.- Enuncie que es un microcontrolador.
2.- Enuncie que es un sistema embebido.
3.- Cuáles son los tipos de arquitectura con los que cuentan los microcontroladores y realice un esquema básico de cada una de ellas.
4.- Mencione que es el lenguaje máquina y el lenguaje ensamblador.
5.- Qué es un lenguaje de alto nivel, mencione algunos ejemplos.
6. - Qué significa: Bit, Byte, WordDouble y Word.
7.- ¿Qué es una memoria?
8.- ¿Qué es una memoria ROM, RAM, Flash?
9.- Enuncie que es el contador de programa (PC).
10.-Tener a la mano en papel o digital el conjunto de Instrucciones del microcontrolador HCS08. (Consultar http://www.nxp.com/docs/en/data-sheet/MC9S08JM60.pdf, tabla 7.2, p.p. 108 a 116)

Introducción

La situación actual en el campo de los microcontroladores se ha producido gracias al desarrollo de la tecnología de fabricación de los circuitos integrados. Este desarrollo ha permitido construir las centenas de miles de transistores en un chip, esto fue una condición previa para la fabricación de un microprocesador. Las primeras microcomputadoras se fabricaron al añadirles periféricos externos, tales como: memoria, líneas de entrada/salida, temporizadores y otros.

El incremento posterior de la densidad de integración permitió crear un circuito integrado que contenía tanto al procesador como periféricos. Así es cómo fue desarrollada la primera microcomputadora en un solo chip, denominada más tarde microcontrolador.

Los principiantes en electrónica creen que un microcontrolador es igual a un microprocesador. Esto no es cierto, difieren uno del otro en muchos sentidos. La primera y la más importante diferencia es su funcionalidad, para utilizar al microprocesador en una aplicación real se debe de conectar con componentes, tales como memoria o componentes y buses de transmisión de datos.

Aunque el microprocesador se considera una máquina de computación poderosa, no está preparado para la comunicación con los dispositivos periféricos que se le conectan. Para que se comunique con algún periférico, se deben utilizar los circuitos especiales. Así era en el principio y esta práctica sigue vigente en la actualidad.

Funcionamiento

1.-Explicación teórica de las características fundamentales de la tarjeta DEMOJM de FREESCALE, figura 3.1.

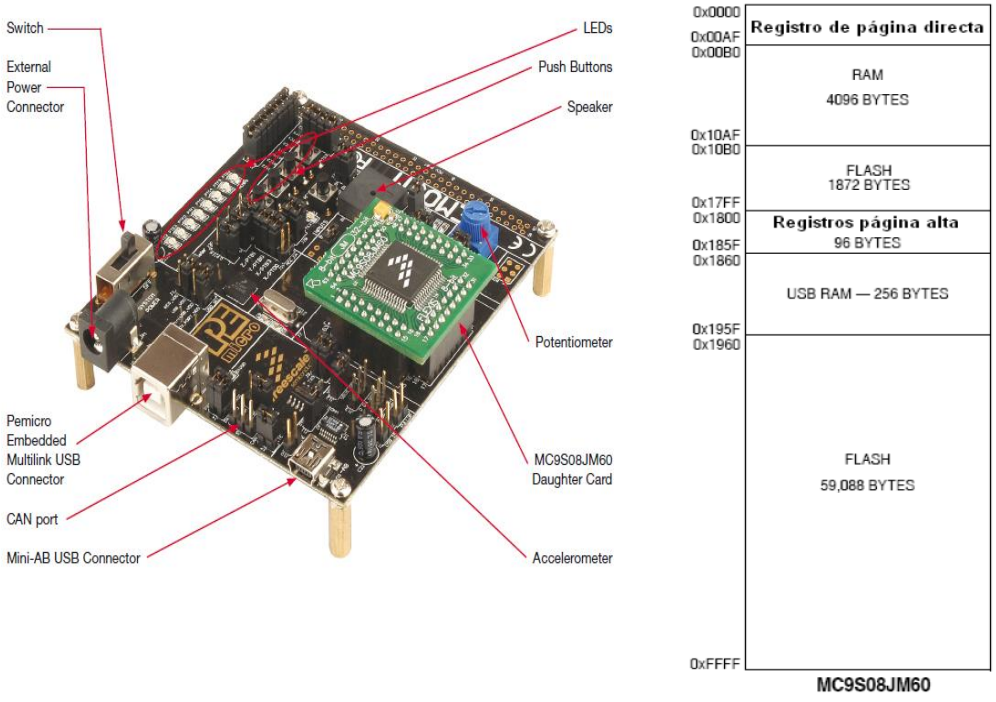

Figura 3.1 Tarjeta DEMOJM y mapa de memoria del MC9S08JM60.

2.- Ejecute los siguientes pasos para realizar el primer programa en Ensamblador (se debe de tener el programa CodeWarrior instalado en la pc):

Nota: Se puede descargar de: http://www.nxp.com/products/developer-resources/software-development-tools/codewarrior-development-tools/codewarrior-development-suites/codewarrior-development-suite-special:CW-SUITE-SPECIAL

Estas prácticas están diseñadas para usar el Ambiente de Desarrollo Integrado (IDE) clásico, sin embargo se recomienda descarga la versión de Eclipse IDE para la familia **S08**.

Una vez hecho esto, aparecerá la pantalla en la figura 3.3.

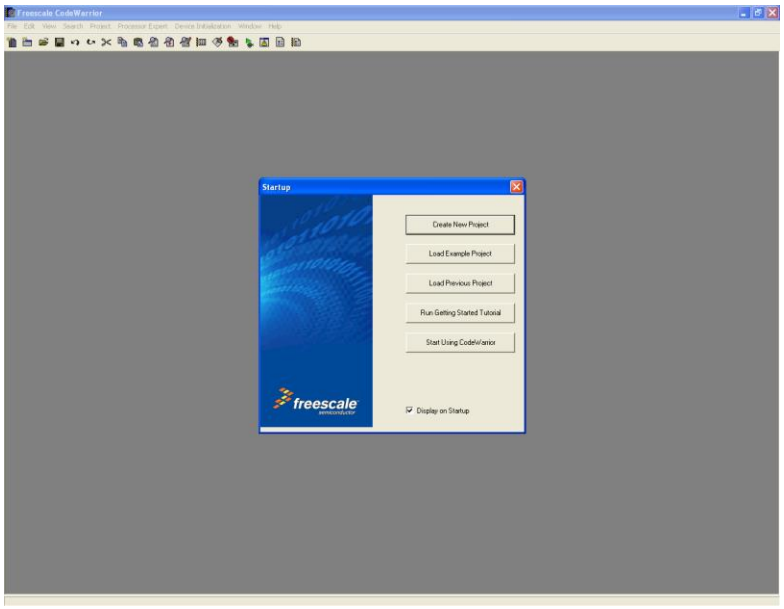

Figura 3.3 Ventana inicial del CodeWarrior IDE.

Ya que ha seleccionado la opción ***"Create New Project o Crear Nuevo Proyecto",*** aparecerá un nuevo cuadro figura 3.4. Aquí podrá seleccionar el tipo de arquitectura que desea utilizar, las cual son: *HC08, HCS08, RS08, ColdFire V1,* y *Flexis.* Desglose el árbol para encontrar el microcontrolador que empleará, en este caso ***MC9S08JM60***, este se encuentra en el grupo de los pertenecientes a la arquitectura ***HCS08*** en la familia ***JM***.

También podrá seleccionar el tipo de conexión que quiera utilizar, como lo son: ***Full Chip Simulation*** o ***P&E Multilink/Cyclone Pro***.

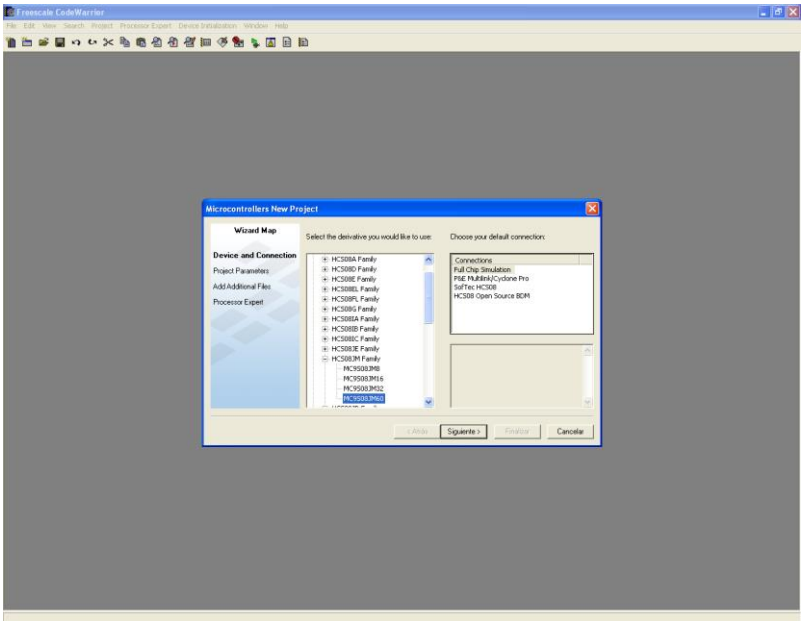

Figura 3.4 Selección del microcontrolador y tipo de conexión.

Full Chip Simulation:

En este tipo no se requiere de una conexión al microcontrolador (simulación en frío) y toda la depuración se realiza en el mismo PC. Sin embargo, solo es recomendable para secciones de código en las cuales solo interfieren datos y no hardware externo.

P&E Multilink/Cyclone Pro:

Para este tipo el código máquina es programado en el microcontrolador y su ejecución se realizará directamente en la maquina final. Esta es la opción recomendada porque permite el 100% de interacción con el hardware. La simulación es completamente real, y la lectura y escritura sobre los pins del microcontrolador se realizan de la misma forma.

Después de presionar el botón *siguiente* en el cuadro anterior, aparecerán las opciones correspondientes a *parámetros del proyecto,* mostrado en la figura 3.5. Aquí podrá usted seleccionar el tipo de lenguaje con el que desea programar así como el nombre del proyecto y el lugar donde será guardado el archivo, en este caso el proyecto se escribirá en ***Ensamblador*** y se

nombrara *Proyecto_1,* por default la opción de lenguaje C se encuentra activa, así que hay que desactivarla y seleccionar ***Relocable Assembly***

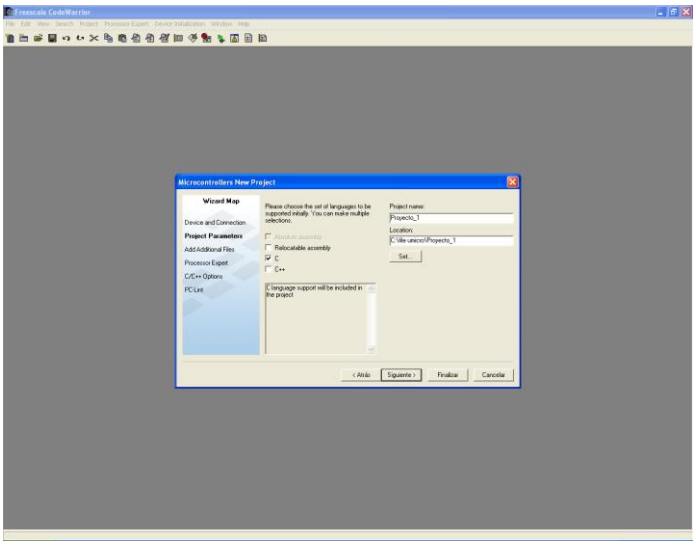

Figura 3.5 Selección de parámetros del proyecto.

Después de presionar el botón *siguiente* aparecerá el cuadro mostrado en la figura 3.6, con el cual podremos adicionar a nuestro proyecto programas existentes o subrutinas ya escritas. El botón ***Add*** es para adjuntar el archivo seleccionado en el árbol izquierdo y el botón ***Remove*** sirve para remover el archivo.

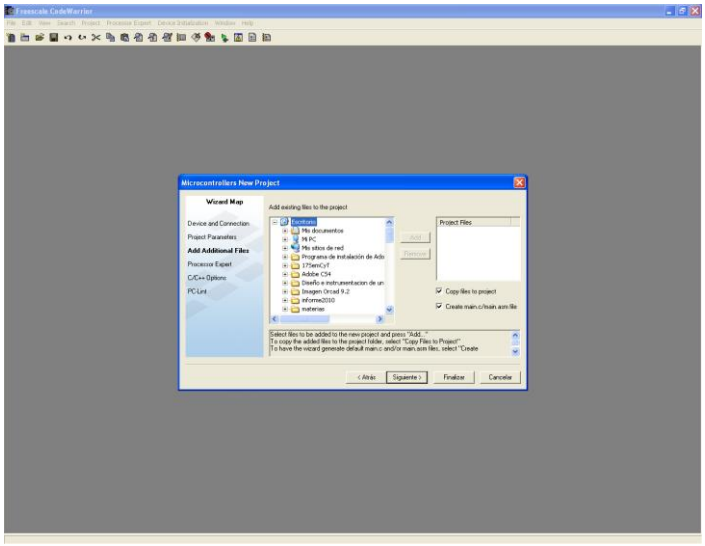

Figura 3.6 Adjuntar archivos adicionales.

Presione el botón ***siguiente*** para que aparezca el cuadro mostrado en la figura 3.7, donde podrá seleccionar ***None*** *"para realizar una programación pura en* ***lenguaje ensamblador"***, y posteriormente presionar el botón ***Finalizar*** para concluir la creación de un nuevo proyecto con el asistente ***CodeWarrior***.

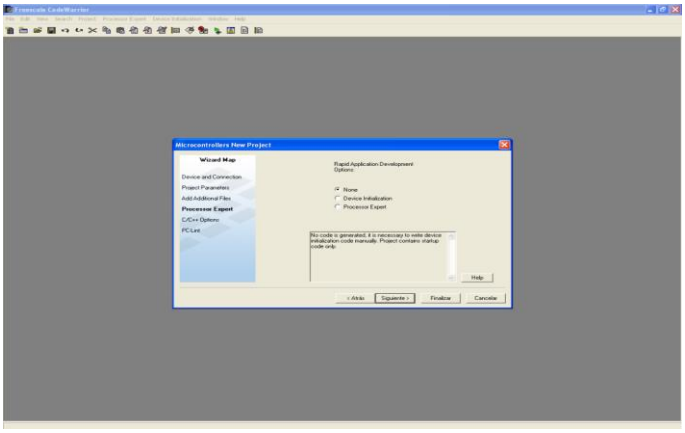

Figura 3.7 Activación del Processor Expert.

Ya inicializado un nuevo proyecto se abrirá la ventana mostrada en la figura 3.8, que será el entorno donde trabajará la mayor parte del tiempo.

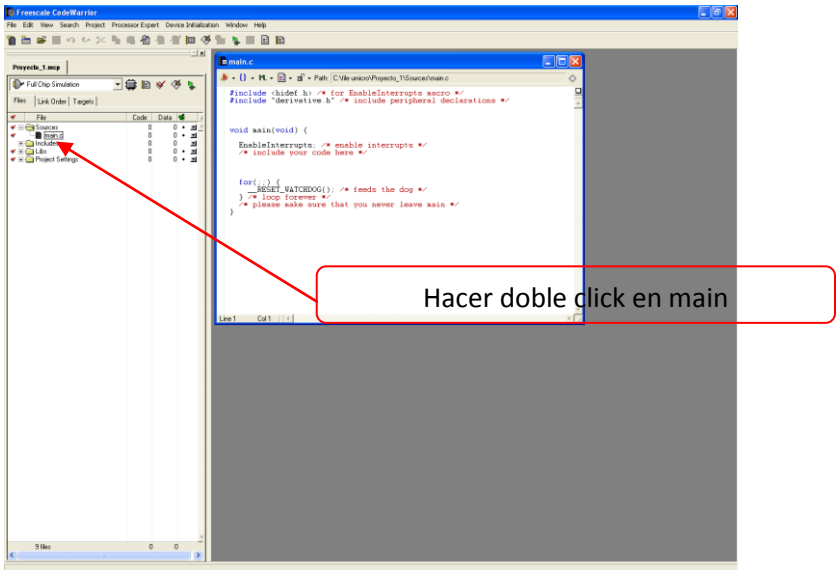

Figura 3.8 Ventana principal del CodeWarrior.

Desarrollo

3. Una vez abierta y maximizada la ventana de trabajo, se introducirá el siguiente código que realiza el encendido y apagado del led PTC4 de la DEMOJM con una demora de tiempo, para poder observar la transición de encendido y apagado.

El código generado por CodeWarrior se representa en color purpura y no es necesario escribirlo. El texto en color verde son comentarios y el texto en negritas es el código en ensamblador que debe escribir!

```
; Include derivative-specific definitions
            INCLUDE 'derivative.inc'
; export symbols
            XDEF _Startup, main
            ; we export both '_Startup' and 'main' as symbols. Either can
            ; be referenced in the linker .prm file or from C/C++ later on
            XREF ___SEG_END_SSTACK    ; symbol defined by the linker for the end of the stack
; variable/data section
MY_ZEROPAGE: SECTION  SHORT          ; Insert here your data definition
; code section
MyCode:     SECTION
main:
_Startup:
            LDHX   #___SEG_END_SSTACK ; initialize the stack pointer
            TXS
            CLI                                 ; enable interrupts
; Common initialization of the write once registers
; SOPT1: COPT=0,STOPE=0
      LDA     #$13
      STA     SOPT1
; Port configuration
; 0 <- Input, 1 -> Output
; PTDDS: PTDDS7=0,PTDDS6=0,PTDDS5=0,PTDDS4=0,PTDDS3=0,PTDDS2=0,PTDDS1=0,PTDDS0=0
      CLRA
      STA     PTDDS
; PTDDD: PTDDD2=1
      BSET    $04,PTCDD
 mainLoop:
            ; Insert your code here
            NOP
            BSET 4,PTCD_PTCD4
            BSR  RETARDO
            BCLR 4,PTCD_PTCD4
            BSR  RETARDO
```

```
            feed_watchdog
            BRA     mainLoop

RETARDO:    CLRX
LOOP2:      CLRA
LOOP:       CBEQA #$FF,ETIQUETA
            INCA
            BRA    LOOP

ETIQUETA:   CBEQX #$F3,FIN
            INCX
            BRA    LOOP2
FIN:        RTS
```

Se muestra el diagrama de flujo para encender y apagar un led con demora de tiempo por software:

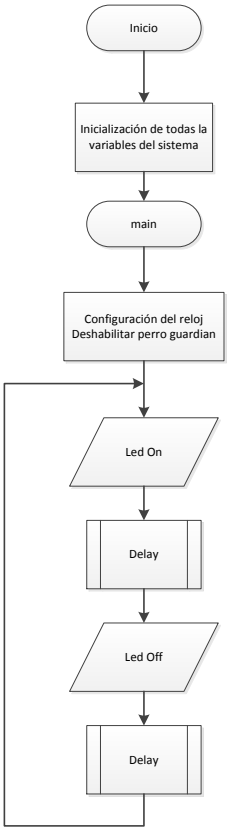

Figura 3.9 Diagrama de flujo de rutina para encender y apagar un led.

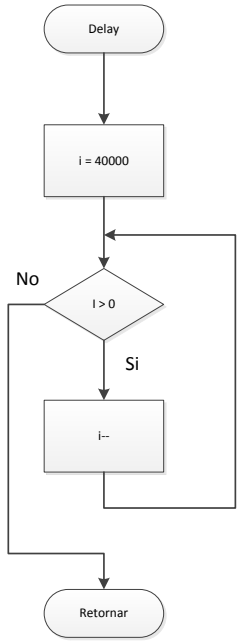

Figura 3.10 Diagrama de flujo para realizar retardos por medio de software.

4. Una vez escrito el programa, se compila usando **Ctrol. B**

Si existen errores, nuevamente revisar la sintaxis del programa.

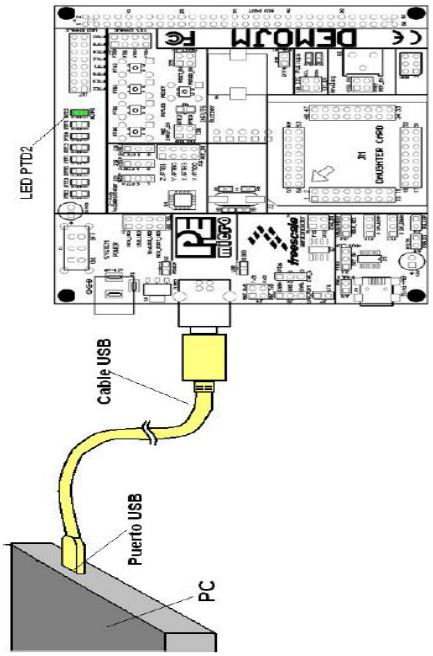

Figura 3.11 Conexión de la DEMOJM a la PC por medio del cable USB.

6. Transferir el programa que realizamos al microcontrolador es de la siguiente manera, figura 3.12.

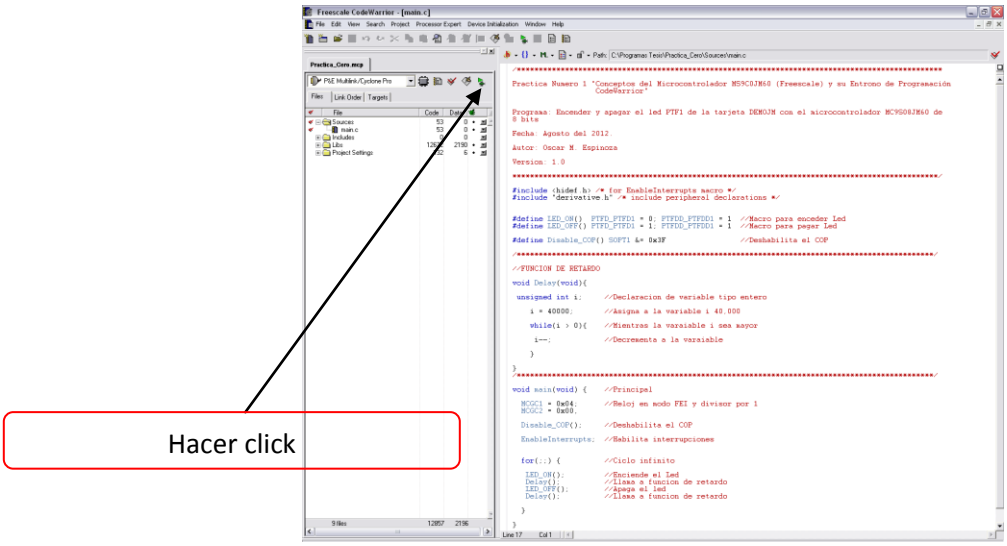

Figura 3.12 Icono Debug para transferir el programa al microcontrolador.

Despues de presionar el icono del ***Debug*** aparecerá la siguiente ventana, anterirormente a este paso se debe conectar la DEMOJM a la PC y encender la tarjeta.

7. Seleccionar ***Connect(Reset),*** figura 3.13.

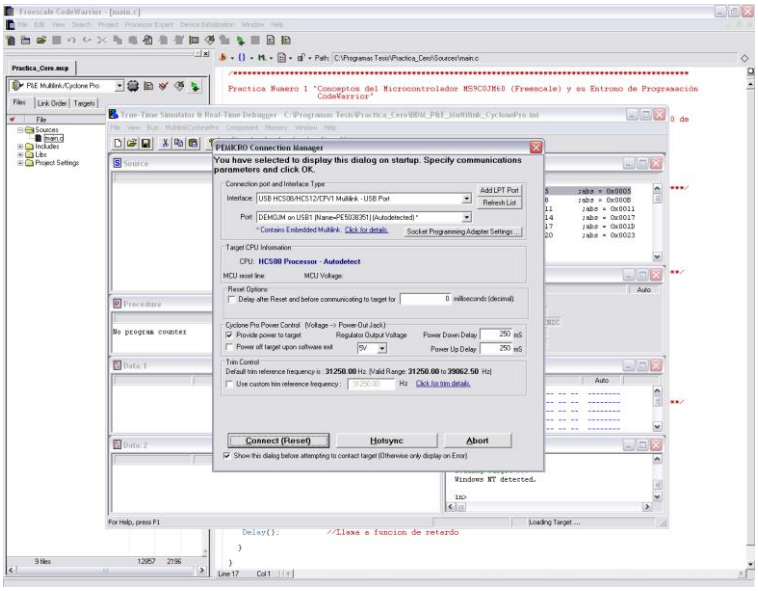

Figura 3.13 Ventana de que indica la comunicación, PC a la tarjeta DEMOJM.

8. Presionar ***Yes***, figura 3.14.

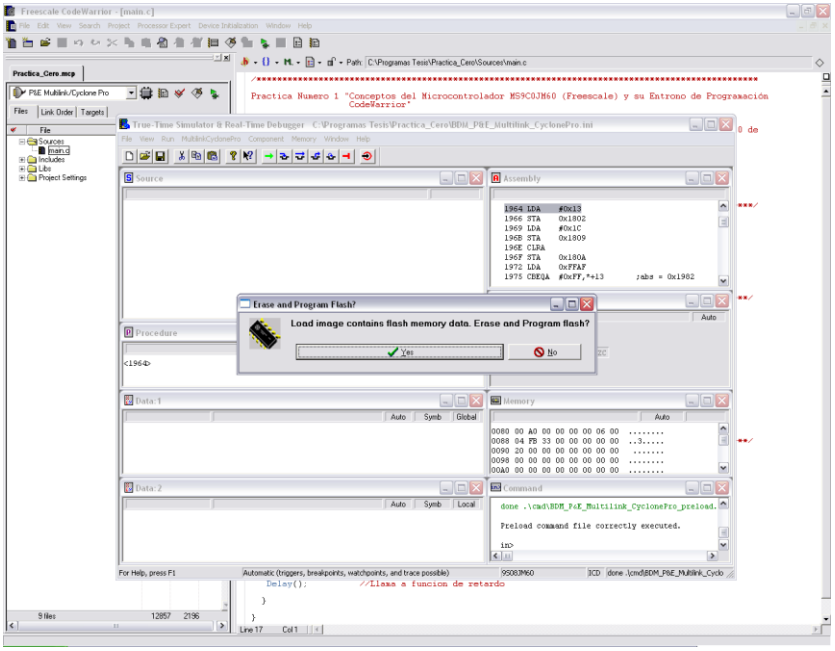

Figura 3.14 Programa transferido a la memoria del microcontrolador.

9. Al dar clic en ***Start*** el programa se ejecutará en tiempo real, observar el comportamiento, figura 3.15.

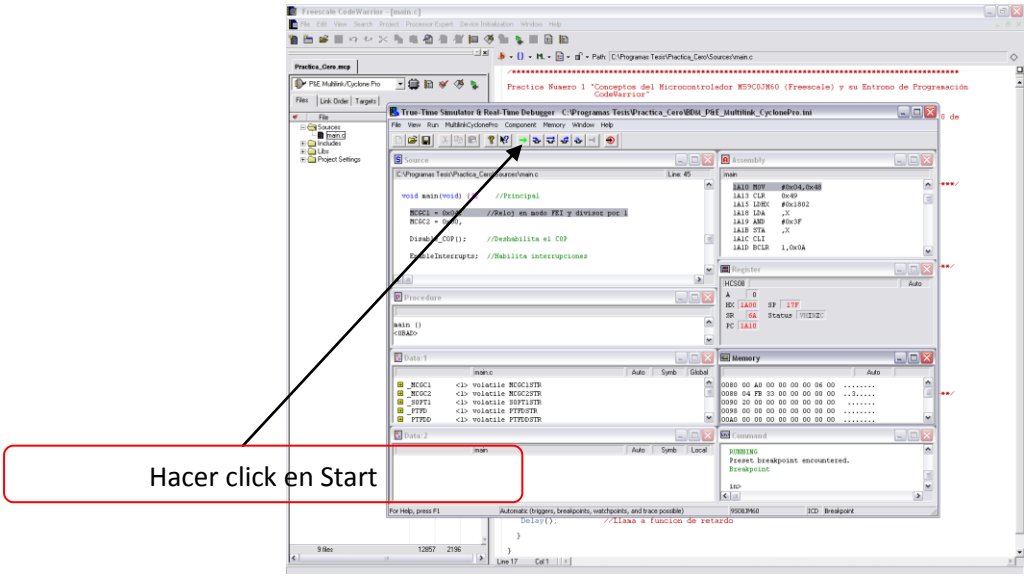

Figura 3.15 Para iniciar, la ejecución del programa en tiempo real en la memoria del microcontrolador.

El menú de botones para la depuración es el siguiente:

 **Run**: Realiza la ejecución desde la función main(). En el caso de una depuración con conexión "in circuit", la ejecución en el microcontrolador se hará en tiempo real y de forma continua. El procesamiento puede ser parado en cualquier momento mediante el botón "**halt**". 

 **Step-In:** Realiza la ejecución de una línea simple de C. En el caso del llamado a una función, lo realiza y se posiciona en la primera línea de la función. Este botón es útil para evaluar el comportamiento de algún procedimiento de forma detallada.

 **Step-Over:** Realiza la ejecución de una línea de C, sin embargo, si la misma corresponde a un llamado de función, esta será invocada y su ejecución no será intervenida. La función se ejecuta en tiempo real y la computadora detiene su ejecución al regresar de la línea invocada.

 **Step-Out:** Realiza ejecución hasta que el procesador abandone el procedimiento o función actual. Es útil cuando se ingresa a ejecución detallada "Step-In" de una función.

 **Single-Step**: Ejecuta la instrucción en ensamblador que está siendo apuntada por el contador de programa (PC). Este botón es útil cuando se requiere conocer el comportamiento del software a nivel de ensamblador y sus efectos sobre las variables, la memoria y los puertos de entrada y salida.

 **Halt:** Obliga al microcontrolador a detenerse en la posición en la que se encuentra en contador de programa (PC) y a actualizar las subventanas del depurador.

 **Reset:** Genera un restablecimiento al microcontrolador, obligando al contador de programa (PC) a posicionarse en la función de inicio Startup (), o bien, a la apuntada por su vector de RESET (0x0000_0004).

Referencias.

http://www.nxp.com/products/sensors/touch-sensors/flexis-jm-demonstration-board:DEMOJM