ANEXO D

SOFTWARE DE DESARROLLO
"CODEWARRIOR DEVELOPMENT STUDIO"

Este documento posibilitará al estudiante comprender y utilizar el ambiente de desarrollo CodeWarrior IDE para la creación, depuración y descarga de programas de los sistemas basadas en microcontroladores Freescale, y particularmente permitirá:

- Entender las partes que componen a CodeWarrior.
- Utilizar el simulador para probar un programa.
- Comprobar las opciones que ofrece el Depurador incluido en CodeWarrior.
- Utilizar las diferentes opciones para programar un Archivo.

Introducción.

Existen en el mercado varias opciones para poder compilar es decir convertir del lenguaje C al lenguaje maquina los programas que son desarrollados por los diseñadores de firmware, pero sin embargo las opciones que cada uno de estos paquetes nos ofrecen son limitadas y no siempre aprovechan las ventajas que ofrece la arquitectura del microcontrolador o del microprocesador utilizado y siempre son de un costo elevado, es por esto que es altamente recomendable utilizar el paquete de desarrollo que cada empresa que crea los microcontroladores nos propone.
En el caso de Freescale Semiconductor, se ofrece CodeWarrior Development Studio, este IDE permite a los usuarios pasar a través de todas las etapas de desarrollo desde el manejo concepto hasta programar la memoria del microcontrolador.

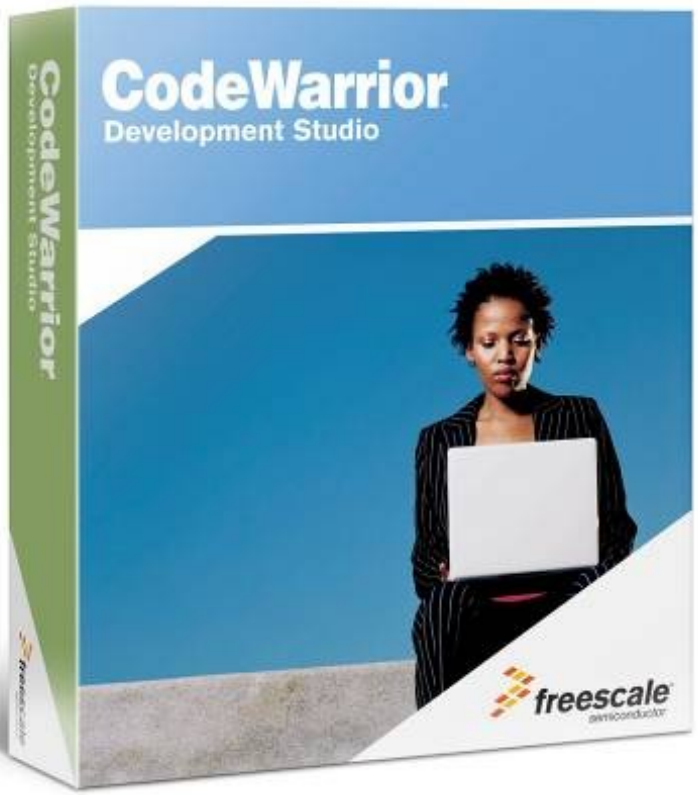


Fig. 1.1 Ambiente de Desarrollo CodeWarrior. Creado por Freescale.

Algunas de las ventajas que nos provee CodeWarrior son:

- Utiliza un sistema basado en Proyectos que nos hace facil el manejo de multiples archivos.
- Creación automatica de Templates para iniciar el proyecto, archivos de encabezado tambien llamados header files para cada uno de los productos de Freescale.
- Dos excelentes herramientas para disminuir la curva de aprendizaje de los perifericos como son:
    - Device Initialization Tool,
    - Processor Expert.
- Cuenta con Editor y compilador para lenguaje C, C++, y lenguaje ensamblador.
- Linker entre el Compilador y el sistema de Descarga.
- Simulador, con varias herramientas para ayudar a la depuracion del sistema sin necesidad de contar con el microcontrolador fisicamente.
- El Editor cuenta con herramientas que permiten la comparación entre archivos, fácil cambio entre componentes (llamados targets).
- Permite el manejo de Control de Versiones con programas como CVS, o Subversion
- Fácil interface con otros programas com Matlab.

- Creación de Librerias.
- Manejo de Sistemas Operativos embebidos.

Compilador.

Como se menciono en el punto anterior los compiladores por lo regular son caros y es un costo que el desarrollador del producto debe absorber sin embargo el ambiente de Desarrollo CodeWarrior existen dos versiones una es gratuita CodeWarrior IDE with Project Wizard con la cual se puede realizar lo siguiente:

- Manejo de Proyectos con hasta 32 archivos.
- Se puede compilar código ensamblador en forma ilimitada.
- Compilador de C con 50 opciones de optimización.
- Es posible compilar hasta 16Kbytes de código en C o C++ para dispositivos de las familias HC08 y HC(S)08/RS08.
- Descarga rápida del programa a Memoria Flash con dos opciones:
    - HC08: Via MON08
    - HC(S)08/RS08: Via BDM
- Simulador completo para todo el chip.
- UNIS Device Initialization tool para generar código de inicialización tanto para el CPU como para los perificos de las familias HC08, HC(S)08/RS08.
- UNIS Processor Expert™ con componentes para las familias HC08, HC(S)08.

Archivo PRM.

Como se menciono en el capitulo 3 el lugar donde queden almacenadas las variables y constantes es siempre una prioridad CodeWarrior utiliza un archivo que es manejado por el Linker para ubicar las zonas de memoria de nuestro microcontrolador. Este archivo es llamado Archivo PRM.

El archivo PRM como se muestra en la figura siguiente cuenta con dos zonas principales en la primera llamada SEGMENTS y la segunda llamada PLACEMENT. En la zona de SEGMENTS se definen las distintas zonas del mapa de memoria del componente eligiendo el tipo de memoria sea RAM o ROM esto le permite al programa que descarga y hace el programado de las distintas memorias donde es RAM y donde es ROM.

Ejemplo. En el siguiente ejemplo se ve como se define un segmento de RAM y y de ROM para el S08QG8. La memoria que se etiqueto como ROM utiliza el espacio desde 0xE000 y hasta la dirección 0xFFAD, y la memoria RAM se elige de la 0x0060 hasta la 0x00FF.

Solución

```
SEGMENTS
     ROM   =  READ_ONLY    0xE000     to    0xFFAD;
     RAM    =  READ_WRITE 0x0060     to    0x00FF;
END
```

Este archivo es generado automáticamente por CodeWarrior una vez que se utiliza el Project Wizard para crear el espacio de trabajo en base a uno de los microcontroladores.

Pero nos falta la segunda zona la zona de PLACEMENT en esta parte el linker informa al compilador y al programa de descarga donde va el código, donde van las variables y donde van las constantes dentro del mapa que se creo en la parte de SEGMENT.

Ejemplo. Se desea que de manera automática las variables se coloquen en RAM y el código y constantes se guarden en ROM .

Solucion.

```
PLACEMENT

     DEFAULT_RAM          INTO       RAM;
     DEFAULT_ROM          INTO       ROM;

END
```

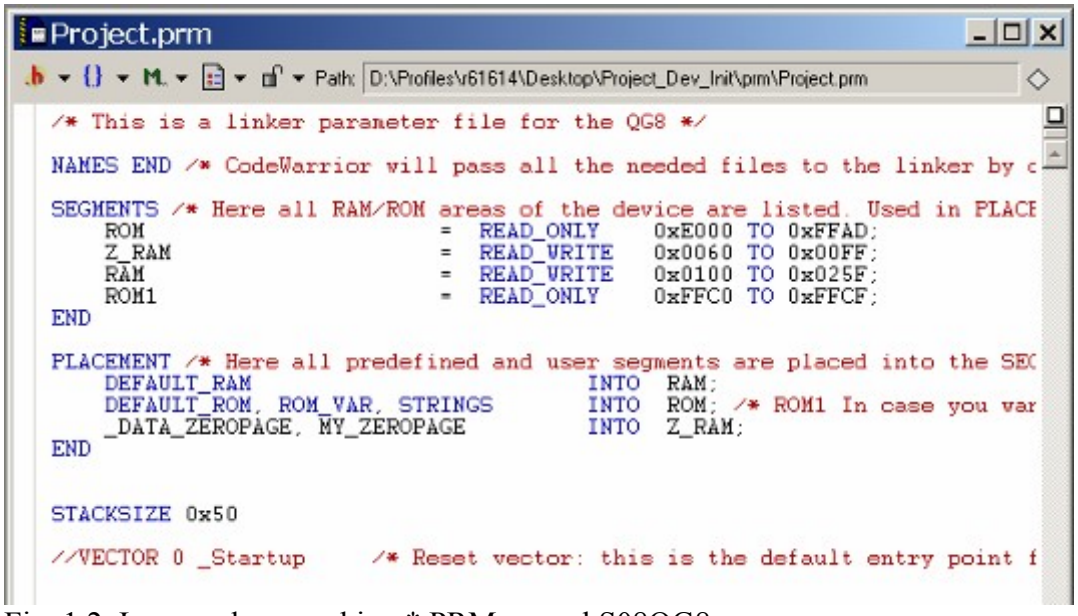


Fig. 1.2 Imagen de un archivo *.PRM para el S08QG8

DIRECTIVAS PRAGMA

Las directivas PRAGMA nos permiten informar al compilador de acciones que tiene que tomar durante el proceso de compilación, en este caso, solo nos vamos enfocar en como podemos elegir donde ubicar una variable o constante en diferentes zonas de memoria definidas por el archivo PRM que anteriormente vimos.

Las directivas PRAGMA que nos permiten esto son:

- DATA_SEG. Indica el segmento para guardar una variable o dato. Este segmento debe estar declarado en el archivo PRM. Las constantes también pueden indicarse el lugar donde serán guardadas con esta directiva.
- CODE_SEG. Indica el segmento para el siguiente código. Y por lo tanto se debe tratar como tal.
- CONST_SEG. Indica que el dato siguiente es reservado y debe mantenerse constante.

A continuación se muestran ejemplos sobre el uso de las directivas,

Ejemplo. En el siguiente ejemplo las funciones f y h son definidas en diferentes segmentos utilizando #pragma CODE_SEG. Se puede observar que tanto la definición de la funciona así como el código de la función se deben de definir con la directiva. Tambien debe hacerse notar que tanto CODIGO1 como CODIGO2 son segmentos que estan definidos en el archivo PRM.

```
/* Definición de las Funciones */
#pragma CODE_SEG CODIGO1
extern void f(void);

#pragma CODE_SEG CODIGO2
extern void h(void);

#pragma CODE_SEG DEFAULT


/* Codigo de las Funciones */
#pragma CODE_SEG CODIGO1
void f(void){ /* mientras que f esta en CODIGO1
h(); /* h esta en el Segmento CODIGO2*/
}

#pragma CODE_SEG MY_CODIGO2
void h(void){
f();
}
```

```
#pragma CODE_SEG DEFAULT
```

Ejemplo. En el siguiente ejemplo las variables a y b son definidas en diferentes segmentos utilizando #pragma DATA_SEG. Observe que después de elegir el segmento para cada una de las variables se regresa al segmento DEFAULT .

```
/* Definición de Variables */
#pragma DATA_SEG SHORT_MEMORY
int b;
#pragma DATA_SEG CUSTOM_MEMORY
int j;
#pragma DATA_SEG DEFAULT
```

COMPILACION CONDICIONAL.

En muchas ocasionas deseamos hacer un código lo suficientemente robusto que nos permita que sea reutilizado para nuevos diseños o simplemente una actualización del producto que estamos lanzando, y mas por los cambios tan rápidos que nuestro mundo globalizado nos demanda.

Una alternativa para poder de alguna manera controlar que una parte de código sea compilada para ciertos casos y en casos distintos no, son las directivas para compilación condicional.

Estas directivas son:

- `#if`
- `#ifdef`
- `#ifndef`
- `#elif`
- `#else`
- `#endif.`

Estas directivas se basan en el funcionamiento tal cual de una sentencia IF-ELSE de código con la diferencia que su uso es solo por el compilador y no por el código que se esta ejecutando.

Ejemplo. En el siguiente ejemplo se desea utilizar el mismo código generado tanto para el MC9S08QG8 como para el MC9S08QG4 aunque son de la misma familia existen algunos registros que no se encuentran en la misma dirección por tanto usando la siguiente compilación condicional se puede elegir entre la librería del QG8 y la librería del QG4. El define es la condicion del IFDEF por tanto en este caso se selecciona QG8

Solución.

```
#define QG8  /*Aquí se elige la librería */


#ifdef QG8

#include <MC9S08QG8.h>  /* Libreria para el QG8

#else
#include <MC9S08QG4.h>  /* Libreria para el QG4
#endif
```

MACROS.

Su uso elimina la realización de tareas repetitivas, automatizándolas. Básicamente, se trata de un grupo de comandos de una aplicación, organizados según un determinado juego de instrucciones y cuya ejecución puede ser pedida de una sola vez para realizar la función que se desea. La directiva DEFINE nos sirve para poder crearlas, bajo la siguiente sintaxis.

```
#define   <ETIQUETA de la MACRO>   <ACCIONES DE LA MACRO>
```

Ejemplo. En este ejemplo se crean las macros LED_ENCENDIDO y LED_BICOLOR en la primera nos permite apagar o prender el LED que esta conectado al bit 5 del puerto B. Y haciendo uso de la macro LED_BICOLOR podemos de una manera muy simple y automazida cambiar el color del LED_BICOLOR observa que se ejecutan dos lineas de código y en el programa principal solo lo mandas llamar como si fuera una función.

```
#define OFF    0
#define ON     1
#define LED_ENCENDIDO(status)        PTB_PTB5 = status;
#define  LED_BICOLOR(control)           PTB_PTB6 =!control;
                                 PTB_PTB6=control;


void main (void)
{
PTB_PTBD=0xff;
LED_ENCENDIDO(ON);
LED_BICOLOR(OFF);
for(;;);
}
```

ESTRUCTURA DE ARCHIVOS.

El orden es uno de los elementos que al programar se vuelve necesario es por esta razón entre otras que se requiere organizar el código que hacemos en diferentes archivos, CodeWarrior al utilizar su Proyect Wizard para empezar un proyecto genera como se muestra en la figura siguiente, una estructura de archivos básica. Esta estructura contiene las siguientes carpetas

- SOURCES en la cual se deben agregar todos los archivos con extensión *.C es decir todos los que contienen el código fuente de todas y cada una de las funciones diseñadas para el programa.
- INCLUDES en esta carpeta deben añadirse los archivos *.H que contienen las definiciones de las variables asi como los prototipos de las funciones de los archivos *.C
- PROJECT SETTINGS. Esta carpeta contiene las propiedades del proyecto completo. Dentro de esta carpeta se encuentra :
    - STARTUP_CODE. Que nos da el código de inicialización del STACK y otras partes del CPU para ejecutar codigo ANSI C.
    - LINKER FILES. Dentro de esta carpeta se encuentra el archivo PRM con las definiciones necesarias para el linker.
    - Archivo BURNER.BBL. Contiene definiciones que seran utilizadas por el HIWARE (programa de Descarga) para la programación de la memoria FLASH y EEPROM.
    - Archivo *.MAP. Contiene información generada después de la compilación, nos indica el tamaño de cada una de las funciones, el tamaño y posición donde estan almacenadas las variables, etc.

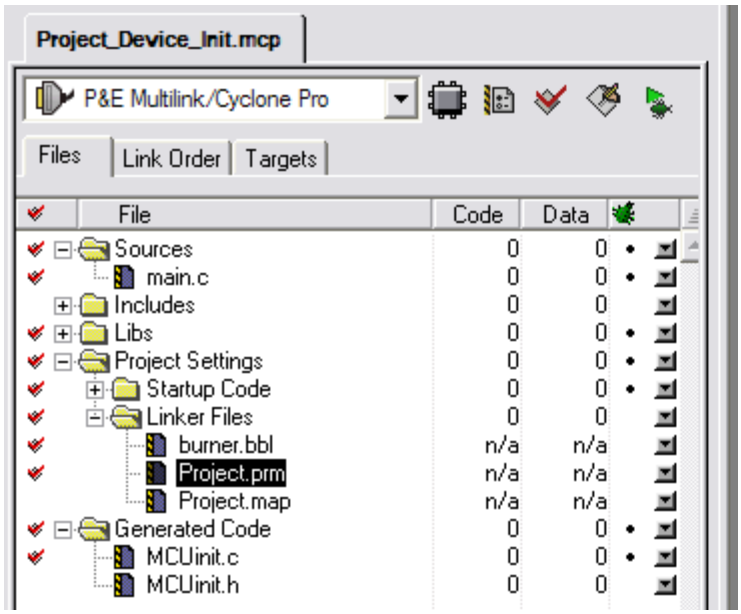


Fig. 5.3 Estructura de un Programa Generado por el Project Wizard para el S08QG8.

Descarga.

La etapa intermedia una vez que el programa ha sido compilado es descargarlo a la memoria del Microcontrolador para poder hacer depuración del mismo. Para esto cabe aclarar que por conexiones se denomina en CodeWarrior a las distintas formas que existen para conectarnos con el MCU para tanto programar la memoria como para poder hacer depuración. En el caso del microcontrolador que hemos estado usando el MC9S08QG8 este microcontrolador pertenece a la famila HC(S)08 de Freescale el cual utiliza el modulo de BDM para poder hacer estas funciones, La tarjeta que DEMOQG8 tiene entonces un BDM desarrollado por la empresa P&E. Por lo cual utiliza una conexión , P&E Multilink / Cyclone Pro, en la siguiente figura se muestran las opciones disponibles en CodeWarrior.

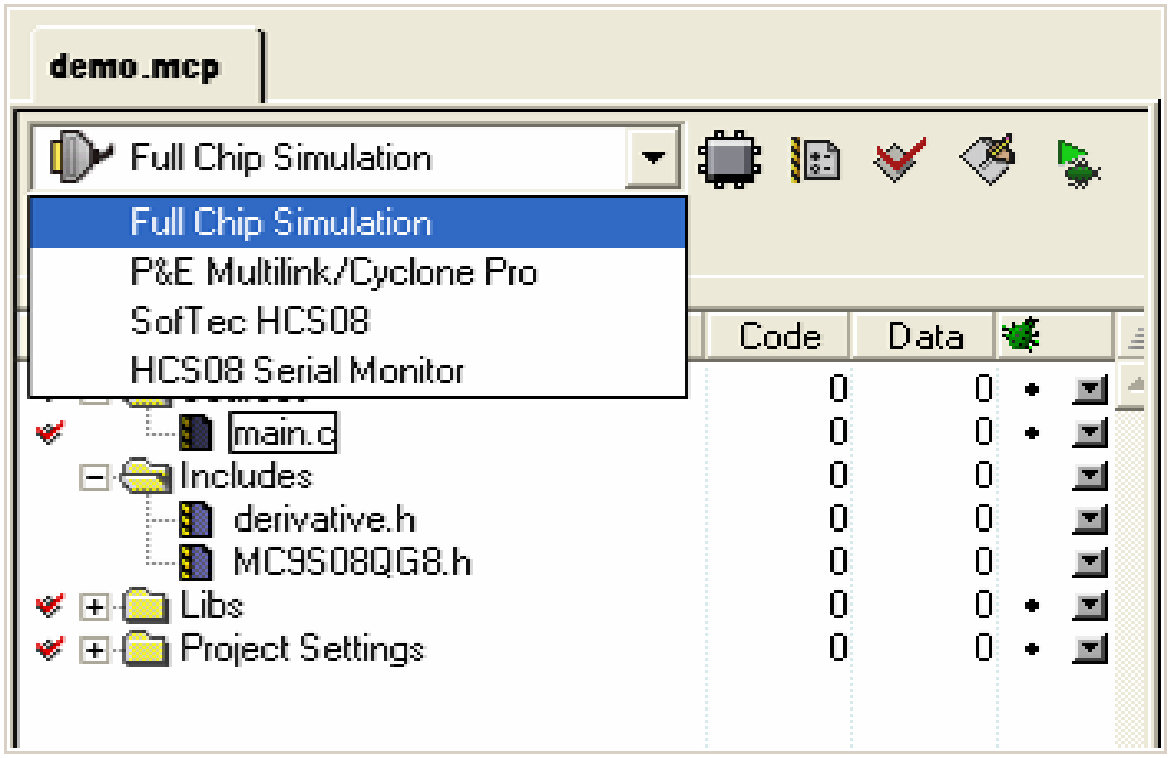


Fig. 5.4 Conexiones Disponibles en CodeWarrior.

Para poder entonces descargar el programa debemos realizar una conexión entre nuestro TARGET (tarjeta o microcontrolador) y nuestra PC, en la fig 5.5 se muestra la ventana que aparece una vez que se ha eligido y en la cual se elige la interfase de conexión, misma que se utilizo durante la explicación en el capitulo 2.

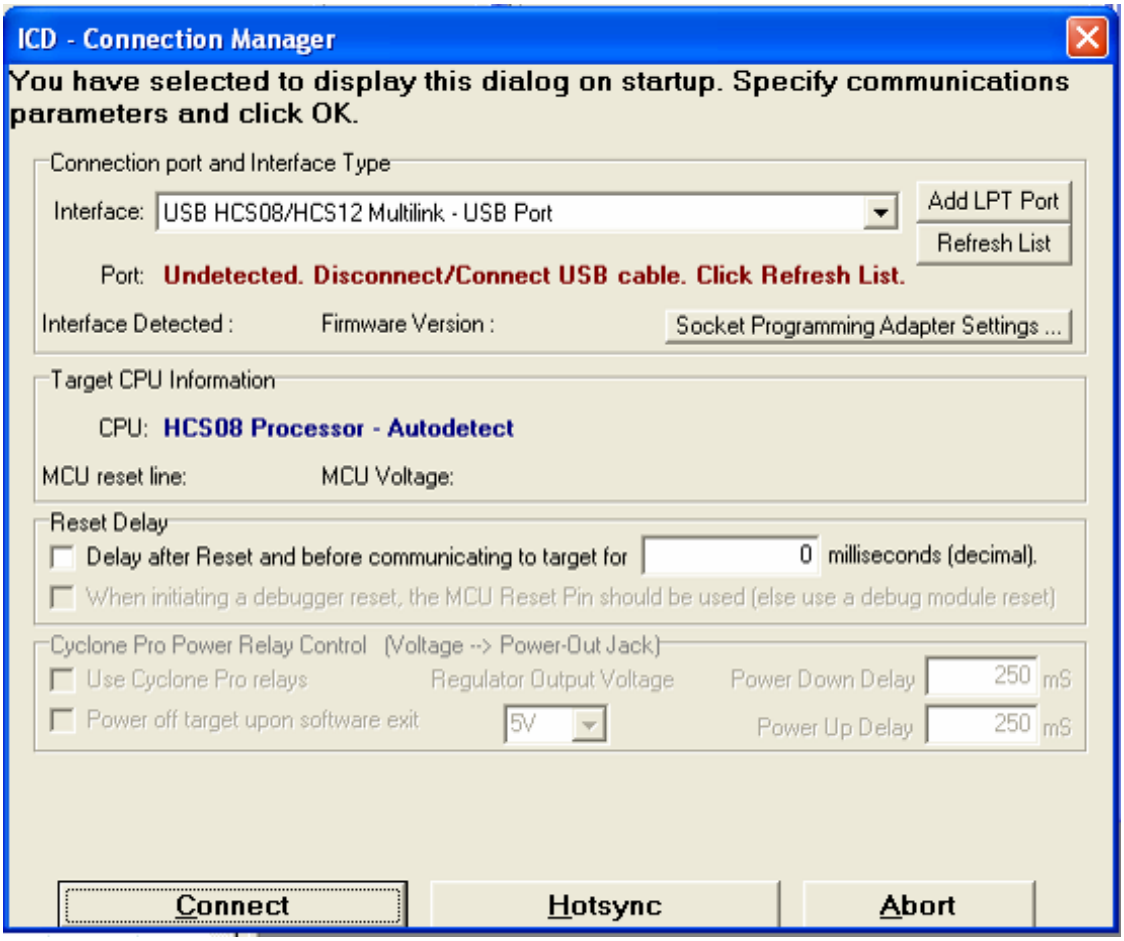


Fig. 5.5 Ventana de Programación del Dispositivo.

Depuración.

Una de las herramientas que mas nos sirven para poder corregir errores de programación (bugs) son las herramientas de depuración en CodeWarrior es crear BREAKPOINTS y WATCHPOINTS a continuación se da una explicación de cada uno de ellos. En la figura 5.6 Se muestra una imagen de la herramienta de Depuración HIWAVE que viene dentro del Ambiente de Desarrollo de CodeWarrior.

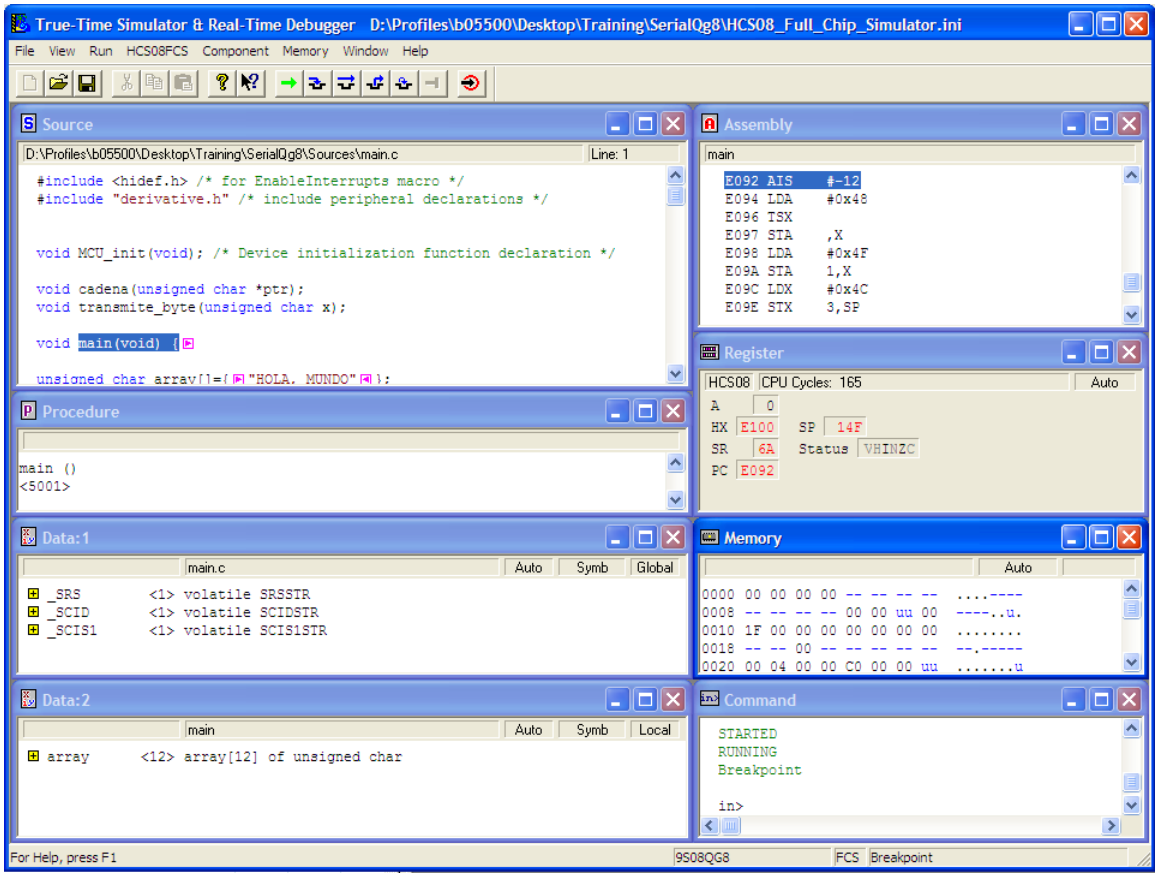


Fig. 5.6 Imagen de la Herramienta de Depuración Hiwave.

BREAKPOINT. Es un comando que permite detener la ejecución del programa en una línea de código indicada. Para fijar un BREAKPOINT basta con solo dar clic derecho sobre la línea de código donde se desea fijar el BREAKPOINT. Como se muestra en la figura 5.7 la flecha con color rojo indica que se ha fijado un BREAKPOINT de manera que al correr el programa en forma continua al encontrar la flecha roja el programa de depuración dentiene la ejecución en esa línea y se puede observar el resultado que hasta ese momento se lleva.

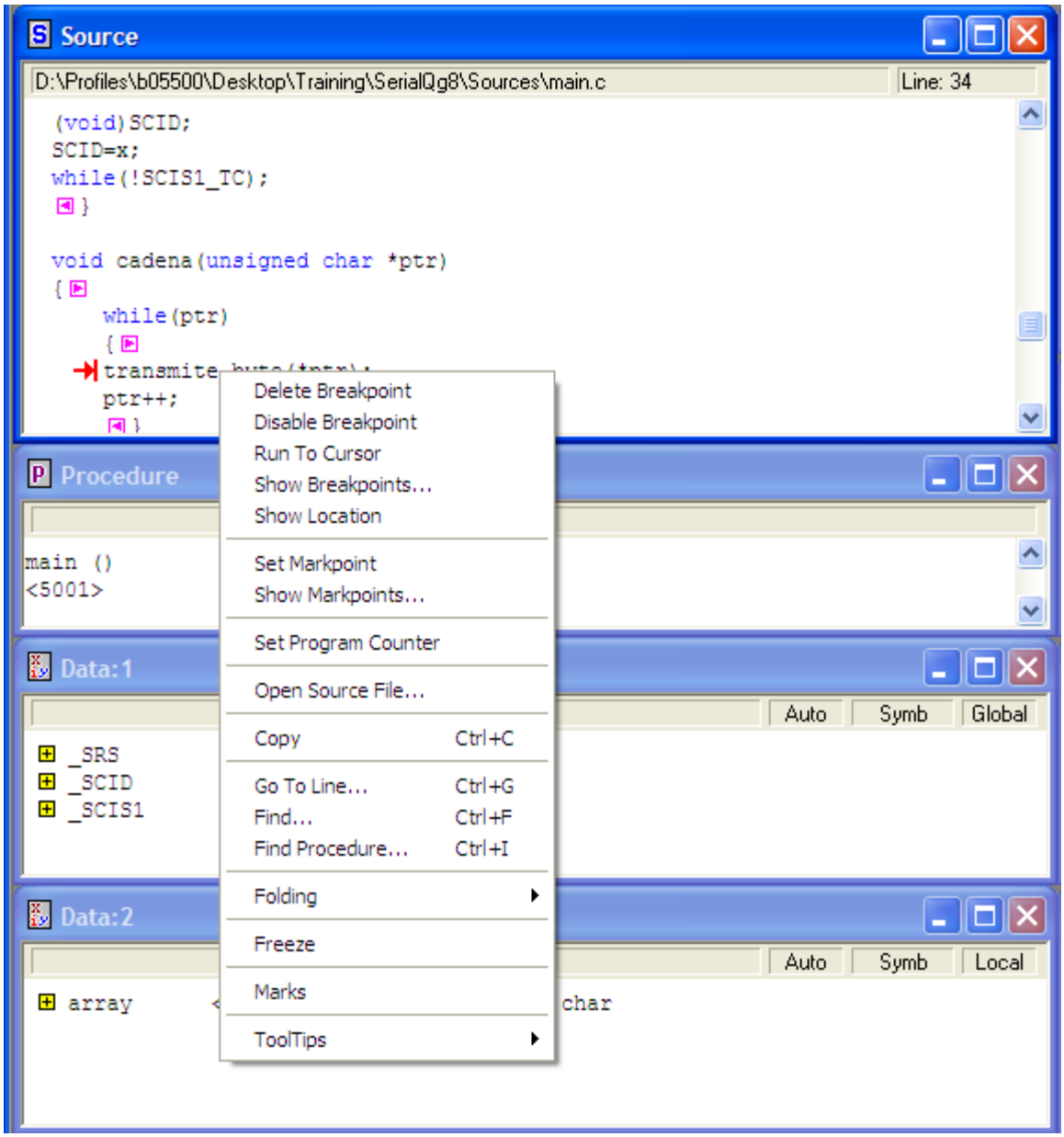


Fig. 5.7 Ventana de Código con un BREAKPOINT.

WATCHPOINT. Es un BREAKPOINT en una localidad de memoria específica. Como se ve en la fig. 5.8 Se agrega un WATCHPOINT al registro SCID del puerto serial, para observar en que momento cambia y es modificado. Al igual que un BREAKPOINT haciendo clic con botón derecho aparece el menú donde se puede elegir agregar WATCHPOINT.

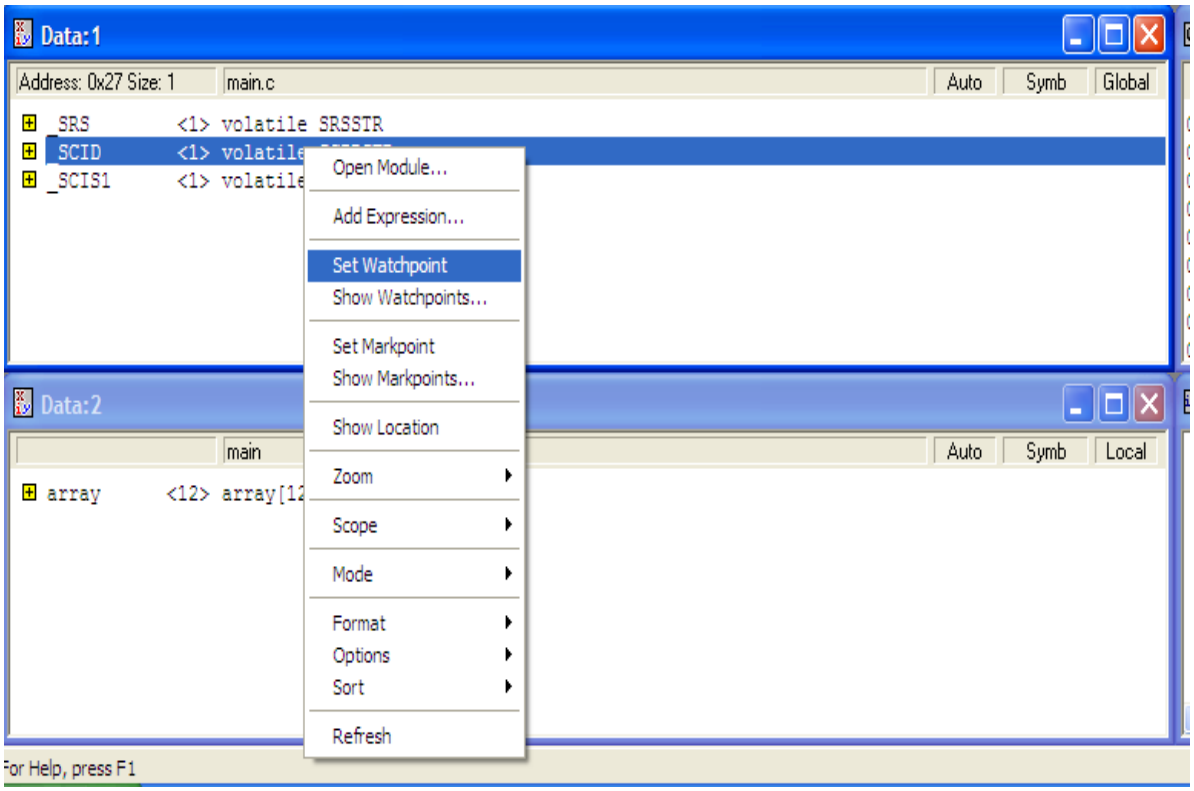


Fig. 5.8 Ventana de Datos donde se agrego un WATCHPOINT.

Funciones Especiales

A continuación en listamos tres funciones extras con las que cuenta el ambiente de Desarrollo de CodeWarrior:

Función de Actualización CodeWarrior Updater.

Con esta herramienta es posible mantener su ambiente de desarrollo con las últimas actualizaciones que Freescale ofrece, algo importante es que actualiza tanto al compilador con las nuevas librerías disponibles tanto para los microcontroladores actuales como para los próximos lanzamientos que van existir. La fig. 5.9 Muestra las pantallas del CodeWarrior Updater.

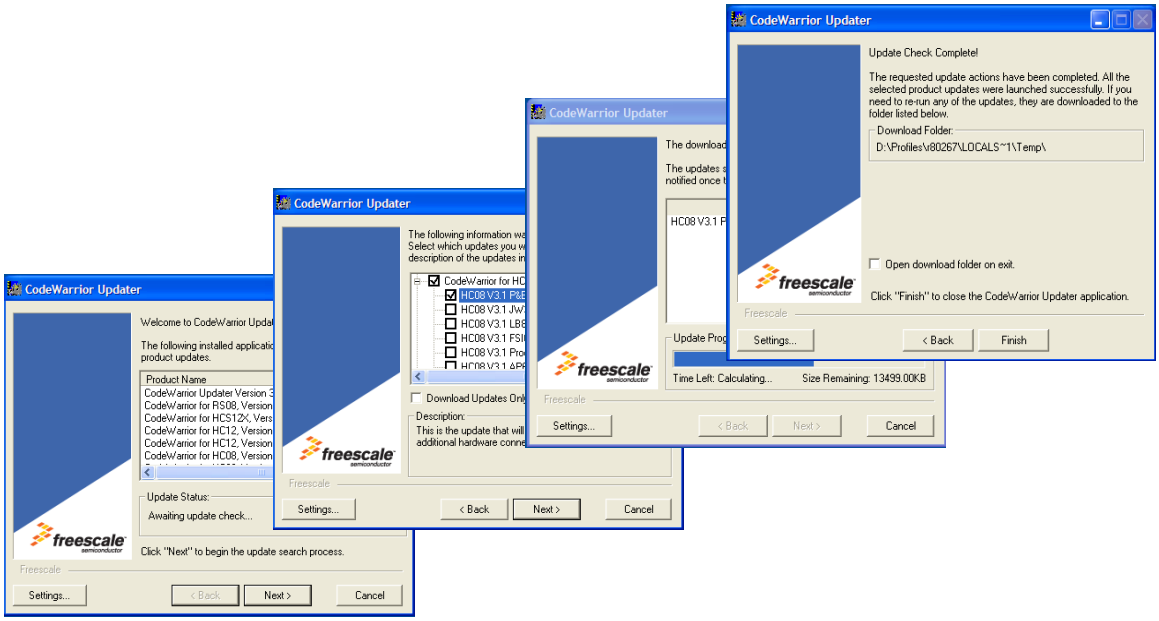


Fig. 5.9 Ventanas del CodeWarrior Updater

Función de Compresión del Proyecto Pack and GO .

Con esta opción es posible crear un archivo ZIP de todo el proyecto que actualmente se este desarrollando. Es muy útil ya que guarda por completo las rutas del proyecto. O se puede seleccionar que solo se guarde el código fuente, dejando los demás archivos no disponibles.

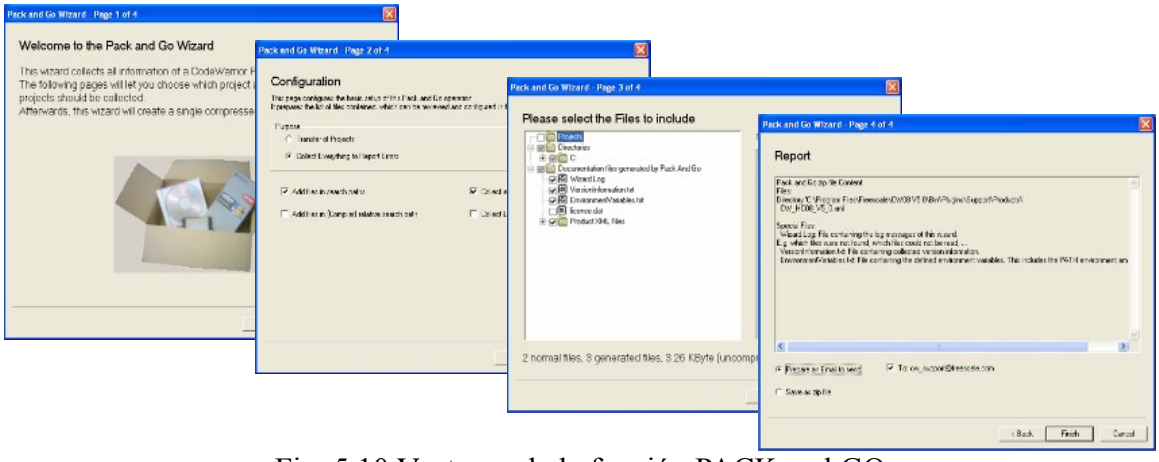


Fig. 5.10 Ventanas de la función PACK and GO

Herramientas de Depuración.

Existen algunas herramientas que ofrecen compañías socias de Freescale para poder hacer Depuración en Línea o en vivo, entre estas se encuentran el P&E Multilink BDM que es la figura 5.11 y en la figura 5.12 se observa un P&E Cyclone Pro que permite programar OFFLINE así como ONLINE.

Fig. 5.11 Multilink BDM

Fig. 5.12 Cyclone PRO

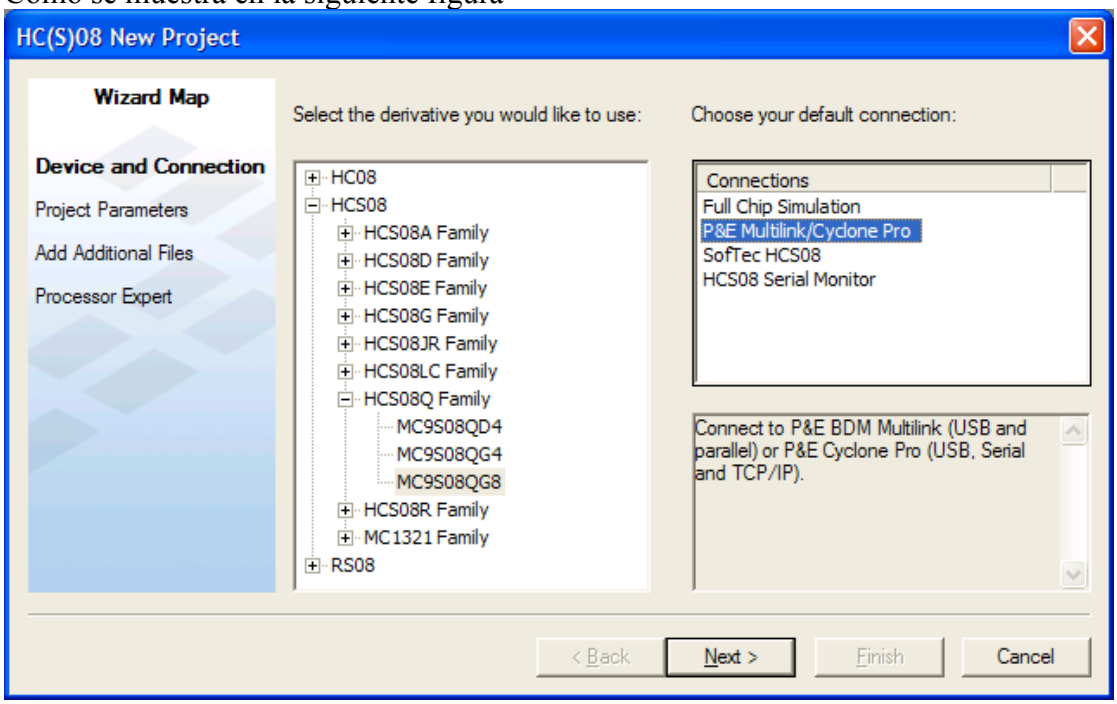
Iniciando el uso del CodeWarrior

El CodeWarrior es una herramienta integrada de desarrollo para los microcontroladores, la cual nos permite desde editar los archivos de los programas, compilar en C, programar y depurar los programas creados para el microcontrolador. En la siguiente sección se vera como empezar un nuevo proyecto en CodeWarrior y hacer el programa para el prendido y apagado de los LED de la tarjeta DEMO9S08QG8. Estamos considerando que previamente ha sido instalado en la computadora el CodeWarrior.

¿Cómo crear un proyecto en el CodeWarrior ?

- Iniciando un proyecto Nuevo
- Del menu nuevo proyecto “File->New project”
  - Selecciona la familia de microcontroladores HCS08
  - Selecciona la familia HCS08Q Familia
  - Selecciona el microcontrolador MC9S08QG8
  - Selecciona la conexión
    - P&E Multilink/

Como se muestra en la siguiente figura

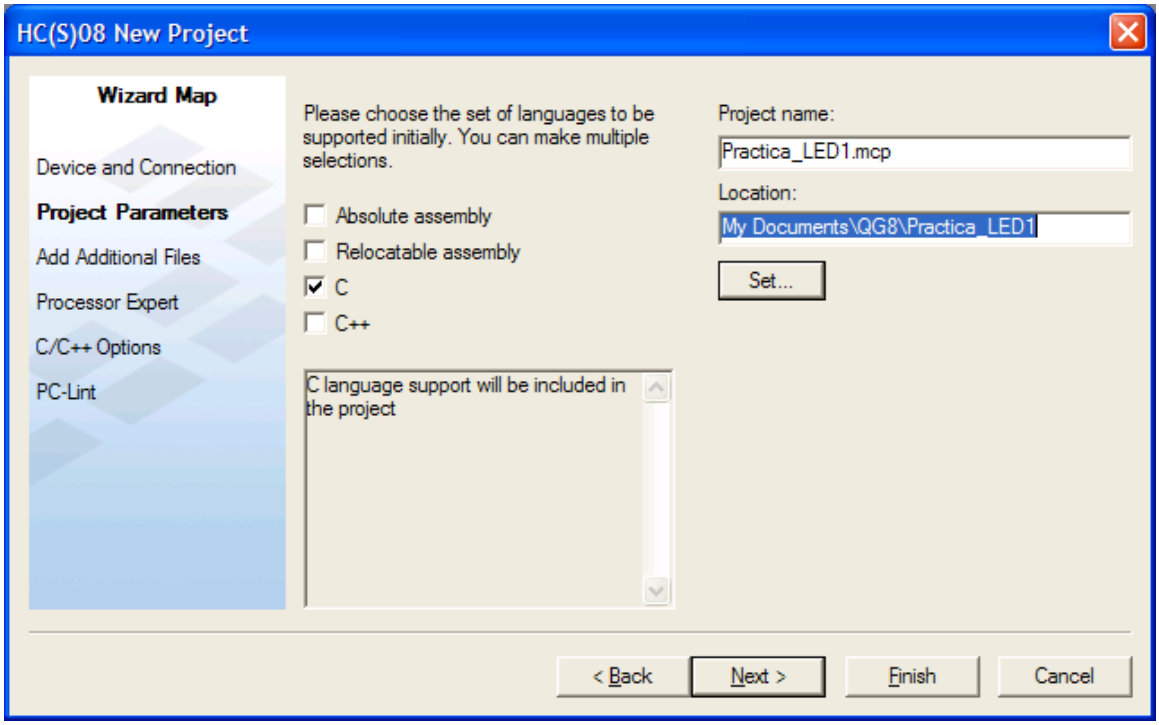


Establece el nombre del proyecto

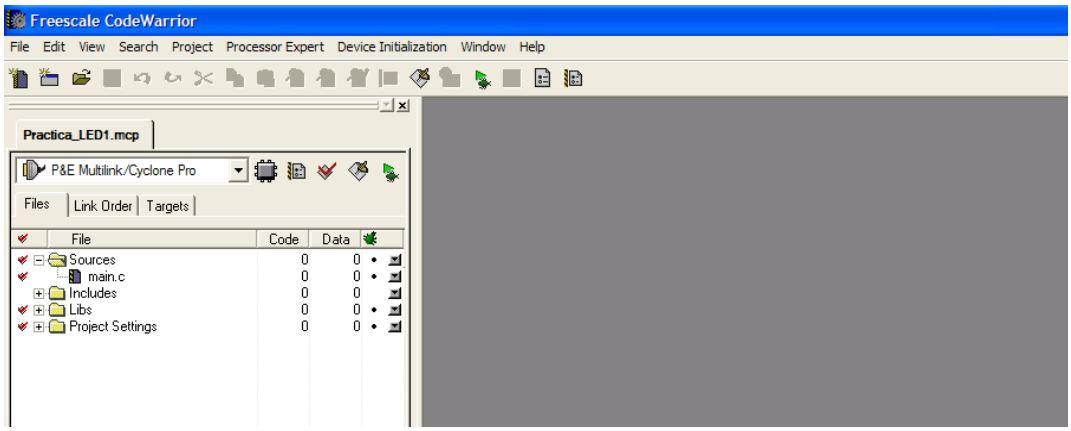
En el cuadro de texto "*Project Name*:"

y la ruta donde se guardará la información del proyecto en el cuadro de texto *"Location:"*

Seleccionamos el Lenguaje a utilizar, para este ejemplo solo marcamos la casilla de C, y hacemos clic en el botón terminar *"Finish"*

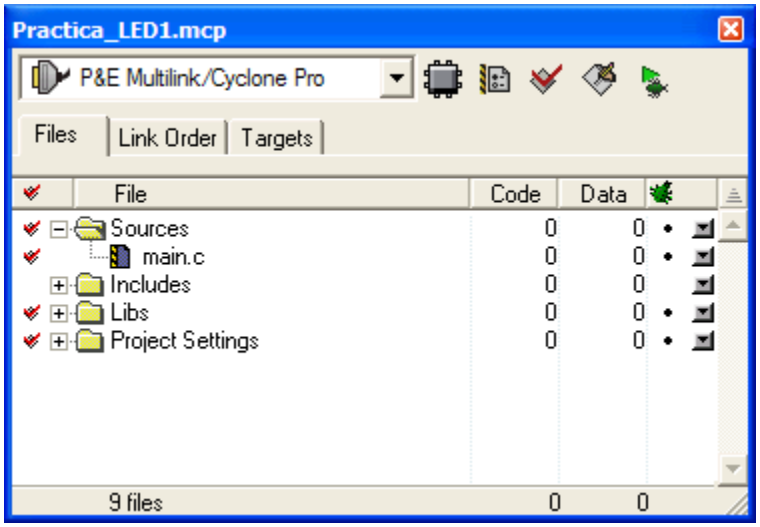

HC(S)08 New Project
Wizard Map
Device and Connection
Project Parameters
Add Additional Files
Processor Expert
C/C++ Options
PC-Lint
Please choose the set of languages to be supported initially. You can make multiple selections.
Absolute assembly
Relocatable assembly
C
C++
C language support will be included in the project
Project name:
Practica_LED1.mcp
Location:
My Documents\QG8\Practica_LED1
Set...
< Back
Next >
Finish
Cancel

Un vistazo sobre el proyecto generado



Al finalizar el proceso de creación del proyecto aparece la ventana del proyecto como se muestra

Una vista mas detallada a la ventana del proyecto



En esta ventana se muestra el nombre del proyecto en el título: Práctica_LED1.mcp

Botones de Herramientas del proyecto:

Los botones de herramientas que se va a utilizar durante los ejercicios de este libro se encuentran agrupados a un lado de la persiana de conexión 

4  Hacer el proyecto (Compilar y enlazar). Make.

Esta herramienta realiza el proceso de compilación de nuestro programa en “C” y de enlace el cual genera el archivo de salida y los mapas necesarios para después programar y depurar nuestra aplicación.

5  Depuración. Debug.

Esta herramienta integra la función de compilar y enlazar “Make” si es que no se ha compilado nuestro código y ejecuta el programa de depuración, el cual a su vez graba el programa en nuestro microcontrolador y permite la depuración del código.

Archivo main().c

En este archivo es donde escribiremos nuestro código, e incluye la función main() la cual llamada después de un proceso de inicialización denominado startup(), encargado de inicializar las variables como es requerido por el estándar ANSI de C.

```
void MCU_init(void); /* Device initialization function declaration */

void main(void) {

  /* Uncomment this function call after using Device Initialization
     to use the generated code */
  /* MCU_init(); */

  EnableInterrupts; /* enable interrupts */

  /* include your code here */



  for(;;) {
    __RESET_WATCHDOG(); /* feeds the dog */
  } /* loop forever */
  /* please make sure that you never leave this function */
}
```

Archivos de Encabezado (archivos.h)

En los programas en lenguaje “C” estos archivos de encabezado se utilizan para definir constantes, establecer el prototipo de las funciones, se verá mas detalle de las funciones en el capítulo 3.

En el caso del archivo generado por el CodeWarrior se incluyen las siguientes referencias a los archivos.h

```
#include <hidef.h>          /* for EnableInterrupts macro */
#include "derivative.h"       /* incluye las declaraciones de los periféricos */
#include <MC9S08QG8.h> /* Definicion de los registros y bits del microcontrolador */
```

En el siguiente recuadro se muestra una parte del archivo MC9S08QG8.h

```
/*** PTBD - Port B Data Register; 0x00000002 ***/
typedef union {
  byte Byte;
  struct {
    byte PTBD0       :1;                                       /* Port B Data
Register Bit 0 */
    byte PTBD1       :1;                                       /* Port B Data
Register Bit 1 */
    byte PTBD2       :1;                                       /* Port B Data
Register Bit 2 */
    byte PTBD3       :1;                                       /* Port B Data
Register Bit 3 */
    byte PTBD4       :1;                                       /* Port B Data
Register Bit 4 */
    byte PTBD5       :1;                                       /* Port B Data
Register Bit 5 */
    byte PTBD6       :1;                                       /* Port B Data
Register Bit 6 */
    byte PTBD7       :1;                                       /* Port B Data
Register Bit 7 */
  } Bits;
} PTBDSTR;
extern volatile PTBDSTR _PTBD @0x00000002;
#define PTBD                            _PTBD.Byte
```

Ejercicio Práctico

Control de encendido y apagado de un LED

En esta práctica se realizará todo el proceso de cómo modificar el proyecto generado por el CodeWarrior para hacer uso de los puertos de entrada y salida del microcontrolador realizando con ello un control de encendido y apagado de los LEDs existentes en la tarjeta DEMO9S08QG8.

Los pasos a seguir son:

1.- Crear un nuevo proyecto con el CodeWarrior con el nombre: practica_led.c y abrir el archivo main.c

2.- Inicializar el puerto B bit 6 y bit 7 como salidas, de acuerdo con la Tabla 1. Los LEDs están conectados en el puerto B PTB6 y PTB7. esto se hace escribiendo un uno en el registro de dirección del puerto B PTBDD.

3.- En el lazo principal del programa encender y apagar los LED
Esto es realizado escribiendo en el registro de datos del Puerto B. PTBD.

Ejemplo del código

```
void main()
{
 /*inicialización del puerto */
 EnableInterrupts; /* enable interrupts */
 /*include your code here */

 PTBDD_PTBDD6 = 1;/*Puerto PTB6 como salida*/
 PTBDD_PTBDD7 = 1;/*Puerto PTB7 como salida*/

 for(;;)
 {
  PTBD_PTBD6 = 1;   /* Apaga el LED en PTB6 */
  PTBD_PTBD7 = 1;   /* Apaga el LED en PTB7 */
  PTBD_PTBD6 = 0;   /* Enciende el LED en PTB6 */
  PTBD_PTBD7 = 0;   /* Enciende el LED en PTB7 */
 __RESET_WATCHDOG(); /* feeds the dog */
 }
}
```

4.- Compilar el programa 

Al terminar tu código en C haz clic en el botón “MAKE”, con lo cual se inicializará el proceso de compilación en cual tomo el código en C y verificará que no haya errores de escritura en el código, o nos advertirá sobre ciertas funciones que efectuó el compilador, si el código no tiene errores de sintaxis este será traducido a un lenguaje para que pueda ser programado y entendido por el microcontrolador.

5.- Programar el microcontrolador 

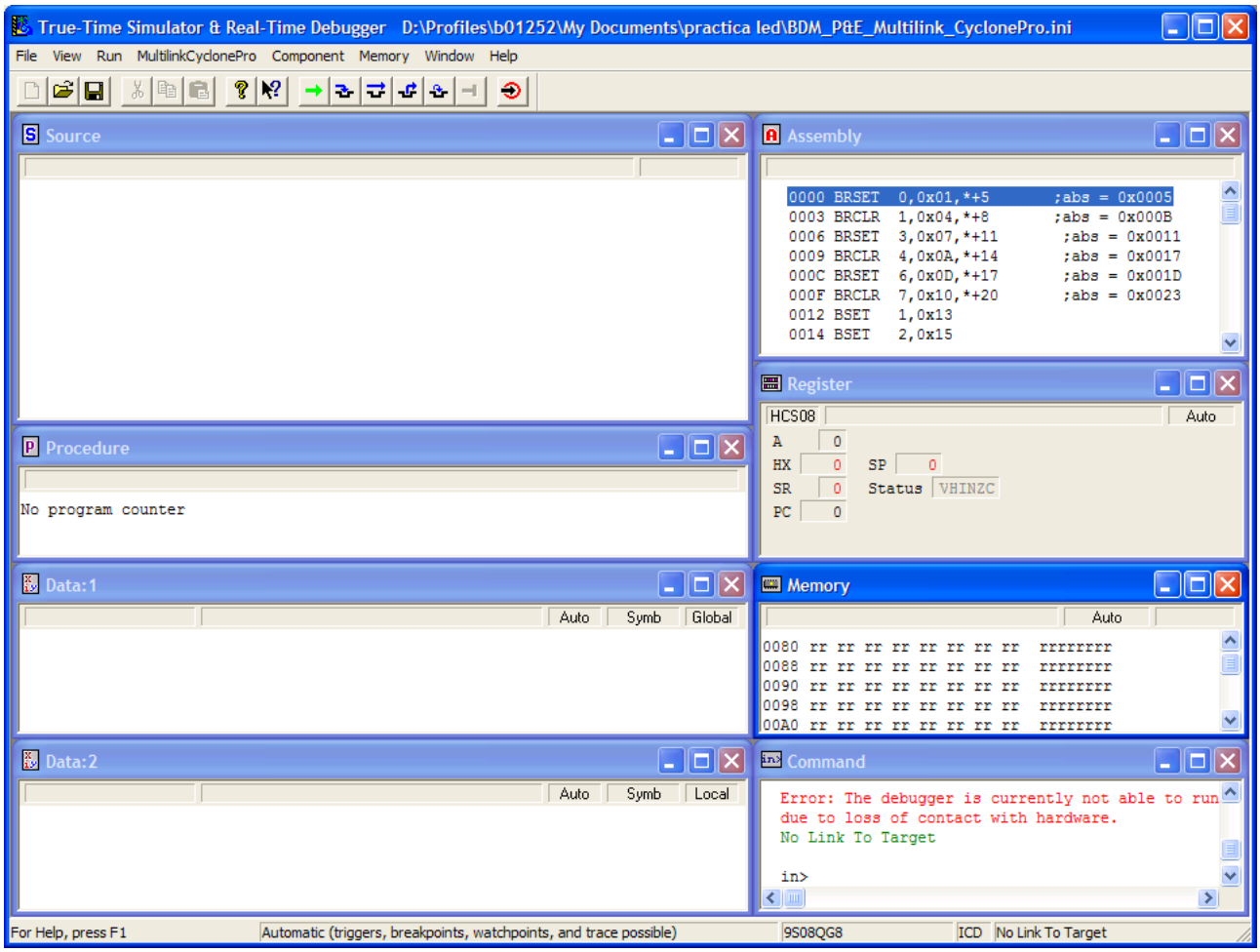
Antes de proceder con esto verifica que la tarjeta DEMO9S08QG8 este conectada a la computadora con el cable USB. Los led de USB PWR y PWR_OUT deben estar encendidos, así como el led verde VDD.
Sino no esta encendido el led verde VDD verifica que los jumpers PWR_SEL este colocado en la posición VB.

La programación se realiza mediante la herramienta Debug  y lo hace en forma automática cuando damos clic en el botón DEBUG. Durante este proceso se indican el estado de programación de la memoria del microcontrolador (memoria Flash)

Las operaciones que realiza el Debugger son:

- ✓ Borra la memoria.
- ✓ Se baja el código a la memoria FLASH (se programa el microcontrolador)
- ✓ Se verifica el programa
- ✓ Se ejecuta el programa y para automáticamente en la función main().

Al terminar el proceso de programación del microcontrolador tenemos la siguiente pantalla:



6.- Correr paso a paso la aplicación

Para correr la aplicación paso a paso o instrucción por instrucción, se utiliza el botón 

6.1.- Observar como se modifica la memoria del microcontrolador
6.2.- Observar el encendido y apagado de los LED

7.- Para ejecutar el programa se utiliza el botón 
Nota. Si se ejecuta el programa los LED aparecerán que están todo el tiempo encendidos, debido a que apagan y prenden muy rápidamente.

Puertos de Entrada – Utilizando los botones de la tarjeta

Objetivo: Para verificar que los botones son leídos adecuadamente, se hará que cuando se presione el boton 1 (SW1) uno encienda el LED1 y cuando se presione el botón 2(SW2) encienda el LED2

De acuerdo con la tabla de jumpers y configuración de la tarjeta DEMO9S08QG8, los botones están conectados a los pines PTA2 y PTA3 del microcontrolador y cada vez que son presionados leen un cero. La practica consiste en base al programa anterior agregar la parte de leer los botones

Los pasos a seguir son.

1.- configurar el puerto como entrada PTA2, y PTA3
PTADD_PTADD2=0;
PTADD_PTADD3=0;

2.- habilitar las resitencias de pull-enable para PTA2 y PTA3
PTAPE_PTAPE2= 1;
PTAPE_PTAPE3= 1;

3.- Actuar sobre los LED si el botón en presionado
En este caso el valor del led toma directamente el valor del botón correspondiente

PTBD_PTBD6 = PTAD_PTAD2;
PTBD_PTBD7 = PTAD_PTAD3;

Ejemplo del código

```
void main()
{
 /*inicialización del puerto */
 EnableInterrupts; /* enable interrupts */
 /*include your code here */

 PTBDD_PTBDD6 = 1;/*Puerto PTB6 como salida*/
 PTBDD_PTBDD7 = 1;/*Puerto PTB7 como salida*/

 PTADD_PTADD2=0;  /*Puerto PTA2 como entrada*/
 PTADD_PTADD3=0;  /*Puerto PTA3 como entrada*/

 PTAPE_PTAPE2=1;  /*Puerto PTA2 como resistencia de Pull-up habilitada*/
 PTAPE_PTAPE3=1;  /*Puerto PTA3 como resistencia de Pull-up habilitada*/
```

```
 for(;;)
 {
  PTBD_PTBD6 =  PTAD_PTAD2; /*El LED1 toma el valor del SW1*/
  PTBD_PTBD7 =  PTAD_PTAD2; /*El LED2 toma el valor del SW2*/

__RESET_WATCHDOG(); /* feeds the dog */
 }
}
```

4.- Compila el programa

Ejecuta tu programa – corre tu programa sin estar conectado a la Computadora.

Una vez programado el microcontrolador, el programa queda en forma permanente en la memoria FLASH. Por lo que para correr la aplicación será solamente necesario conectar la fuente de alimentación.

NOTA el COP. El microcontrolador se configura después de reset para que opere el COP, por lo que si tu aplicación no esta preparada para utilizar esta función debes deshabilitarlo como se muestra en el siguiente código.

SOPT1_COPE = 0; /*Deshabilita la función de COP*/

```
void main()
{
 /*inicialización del puerto */
 OPT1_COPE = 0; /*Deshabilita la función de COP*/

 EnableInterrupts; /* enable interrupts */
 /*include your code here */

 PTBDD_PTBDD6 = 1;/*Puerto PTB6 como salida*/
 PTBDD_PTBDD7 = 1;/*Puerto PTB7 como salida*/

 for(;;)
 {
  PTBD_PTBD6 = 1;    /* Apaga el LED en PTB6 */
  PTBD_PTBD7 = 1;    /* Apaga el LED en PTB7 */
  PTBD_PTBD6 = 0;    /* Enciende el LED en PTB6 */
  PTBD_PTBD7 = 0;    /* Enciende el LED en PTB7 */
 __RESET_WATCHDOG(); /* feeds the dog */
 }
}
```

4.- Compilar el programa 

Al terminar tu código en C haz clic en el botón “MAKE”, con lo cual se inicializará el proceso de compilación en cual tomo el código en C y verificará que no haya errores de escritura en el código, o nos advertirá sobre ciertas funciones que efectuó el compilador, si el código no tiene errores de sintaxis este será traducido a un lenguaje para que pueda ser programado y entendido por el microcontrolador.

5.- Programar el microcontrolador 

6.- Ejecutar el programa  .

Los LEDs estaran apagados hasta que sea presionado alguno de los botones.