Guía de Microcontroladores

Introducción

Este documento tiene como objetivo de para comenzar con el desarrollo de programas para micro-controladores. Por motivos prácticos se utilizan líneas microcontroladores PIC de 8 bits de Microchip, ya que por su popularidad existe una amplia cantidad de herramientas disponibles para el desarrollo.
Esta guía explica la instalación de las herramientas necesarias para resolver la práctica e introduce algunos ejemplos para demostrar cómo se utilizan. Es importante tener en cuenta que este documento es solo un complemento de la teoría y de las explicaciones de práctica, y que se requiere un mínimo de conocimiento para entender y resolver los ejercicios prácticos.

Instalación de herramientas

1. Instalación del compilador C para microcontroladores

Hi-Tech Pic C Compiler es un compilador C que permite generar código de máquina para ejecutar en microncontroladores PIC de la empresa Microchip. Pasos para instalar:

a. Descargar la versión Lite para Windows del compilador C desde la url: http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en542849

b. Ejecutar el asistente de instalación. Es importante durante la instalación seleccionar las opciones de "modo lite" (figura 1) y de "agregar a las variables de entorno" (figura 2). Recuerde la ruta de instalación porque la necesitaremos luego.

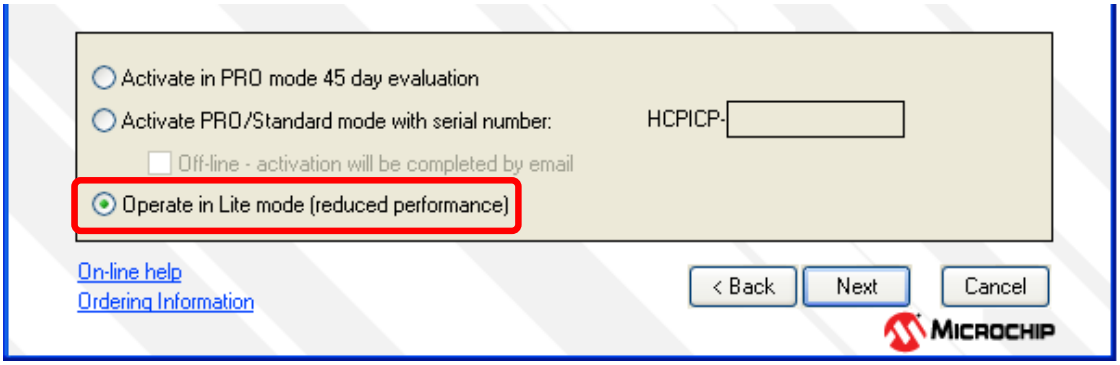

Figura 1

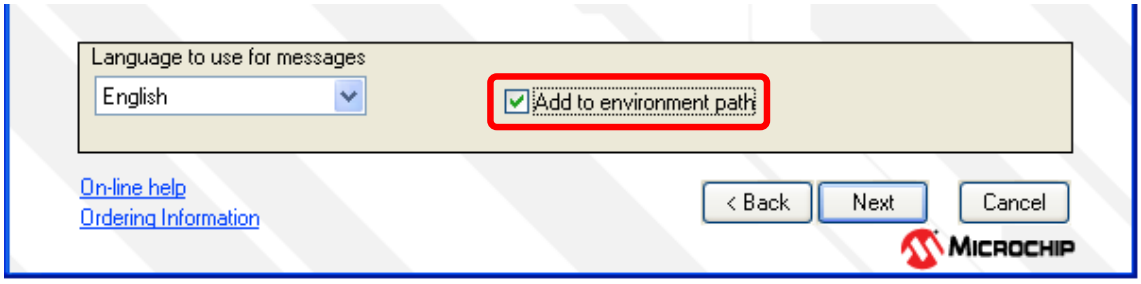

Figura 2

2. Instalación del entorno de simulación

Proteus es un software desarrollado por Labcenter Electronics que permite el diseño de placas de circuitos impresos y la simulación de dispositivos electrónicos y microprocesadores. Los pasos a seguir para instalar es te software son los siguientes:

2.1. Descargar la versión de demostración de este entorno de simulación desde: http://www.labcenter.com/download/prodemo_download.cfm

2.2. Ejecutar el asistente de instalación. Es importante durante la instalación seleccionar las opciones de "Simulación VSM" que instala el soporte para la simulación de micro-controladores (figura 3).

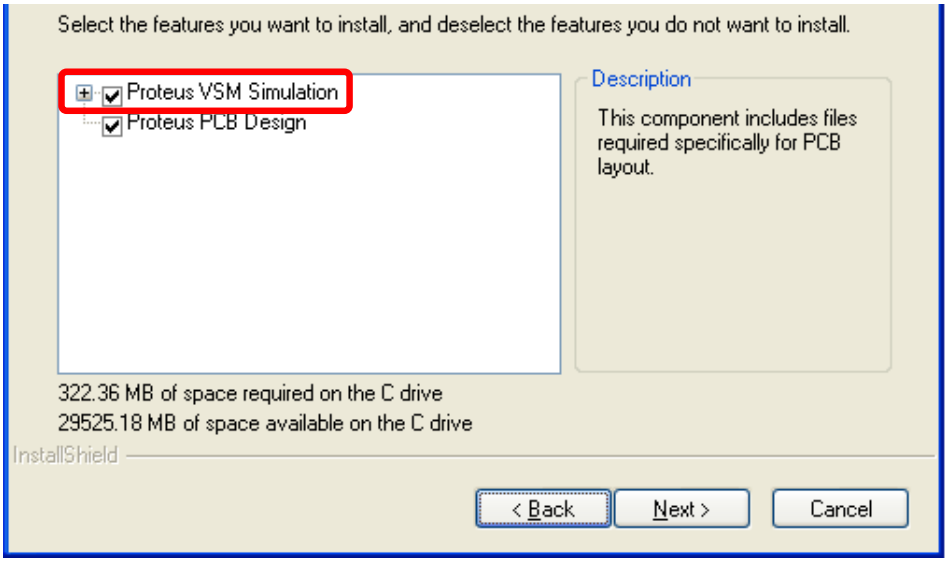

Figura 3.

Configuración de herramientas

3. Configuración del entorno

Para comenzar con la configuración del entorno de simulación ejecute la aplicación recién instalada "Isis Demo". A continuación abra la ventana de configuración seleccionando la opción del menú "Source➔Define Code Generation Tools" (figura 4).

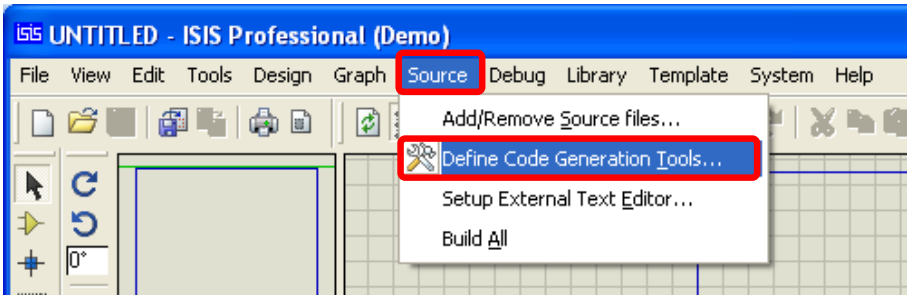

Figura 4.

En la ventana de configuración (figura 5) debe crear una nueva configuración para el compilador C instalado previamente. Los pasos para configurar son:

a. Presione el botón “New” para crear la nueva configuración.
b. Seleccione el compilador PICC.EXE ubicado en la carpeta BIN donde fue instalado el compilador de Hi-Tech C.
c. Establecer la extensión de los archivos fuente con el valor “C”.
d. Establecer la extensión de los archivos de salida con el valor “COF”.
e. Establecer los parámetros que se enviarán al compilador con el valor “%1 –O%2”:
    1. “%1” representa el nombre del archivo principal a compilar.
    2. “–O%2” indica que cual es el nombre del archivo de salida al compilador. Este es el nombre es el mismo que el principal pero con la extensión establecida en el punto anterior.
f. Para finalizar la configuración presione el botón “OK”.

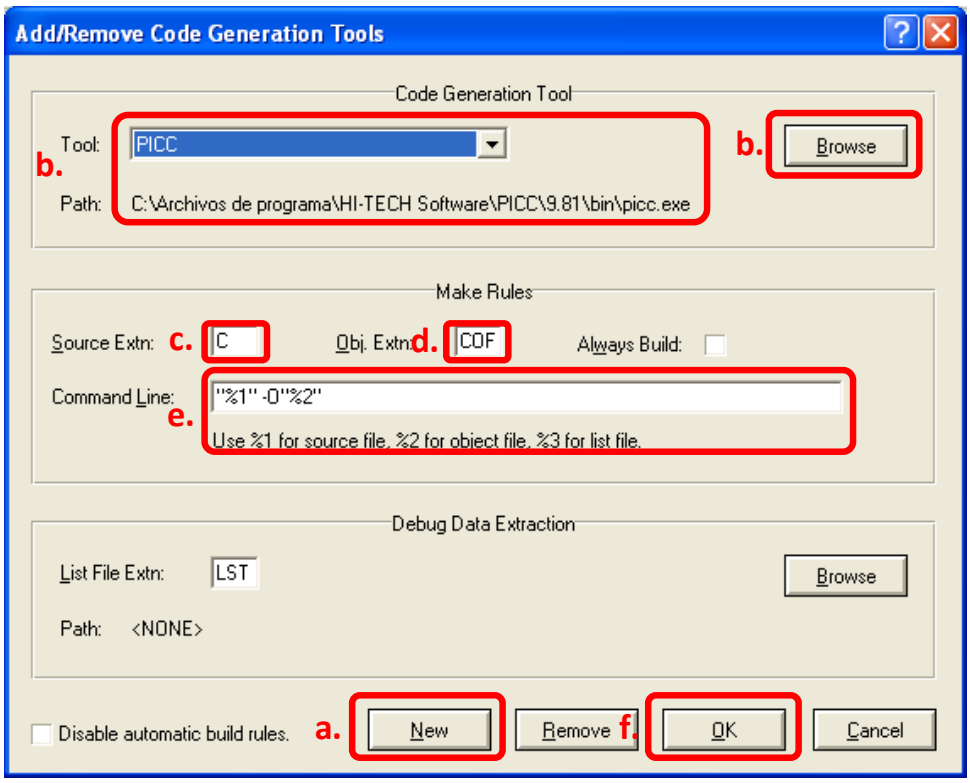

Figura 5.

4. Configuración opcional del editor de archivos

El entorno de simulación provee de un editor de texto interno, pero es posible establecer un editor externo. Para configurar el editor por defecto selección la opción del menú “Source➔Setup External Text Editor” (figura 6). Un vez abierta la ventana de configuración, simplemente presione el botón “Browse” y seleccione la aplicación que desea usar como editor de texto. Luego presione presione “OK” para finalizar.

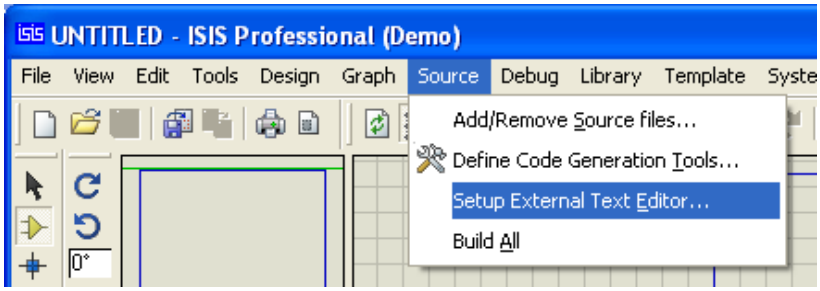

Figura 6.

Prueba de Configuración e Interacción con el entorno

5. Primer Programa

Los pasos a seguir para crear una simulación dentro del entorno son:

a. Abrir o crear un diseño de un circuito electrónico que contenga un microcontrolador.
b. Escribir un programa en C y compilarlo para ejecutarlo dentro del microcontrolador.
c. Iniciar la simulación para interactuar con el microcontrolador.

5.1. Circuito electrónico

Para empezar con el primer programa abriremos el uno de los diseños de ejemplos que vienen con el entorno de simulación:

a. Seleccione la opción del menú **"File➔Open Design"** para abrir la ventana se selección.
b. Ingrese a la carpeta **"VSM for PICMICRO"** donde se encuentran los ejemplos para los microcontroladores PIC de Microchip.
c. Ingrese a la carpeta **"VSM for PIC16"** donde se encuentran los microcontroladores de 8 bits de la línea **PIC16**.
d. Ingrese a la carpeta **"PIC Doorbell"** y seleccione el archivo **"Doorbell.DSN"**. En este diseño (figura 7) puede observarse:
    a. Un microcontrolador **PIC 16F84A** (ver figura 8 para las características), donde se ejecutará nuestro programa compilado.
    b. Dos interruptores conectados a los pines **RA0** y **RA1** que se mapean con los 2 bits menos significativos del registro **PORTA** del microcontrolador.
    c. Un parlante conectado a los pines **RB0** a **RB3** que se mapean con los 4 bits menos significativos del registro **PORTB** del microcontrolador.
    d. Dos leds conectados a los pines **RB4** y **RB5** que se mapean con los bits 4 y 5 del registro **PORTB** del microcontrolador.
    e. Un oscilador conectado a los pines **OSC1** y **OSC2** del microcontrolador que se utilizan para establecer la frecuencia de reloj a 1Mhz. Este circuito (2 capacitores y un cristal de 1Mhz) genera los pulsos que permiten sincronizar todos los dispositivos y la ejecución de las instrucciones.

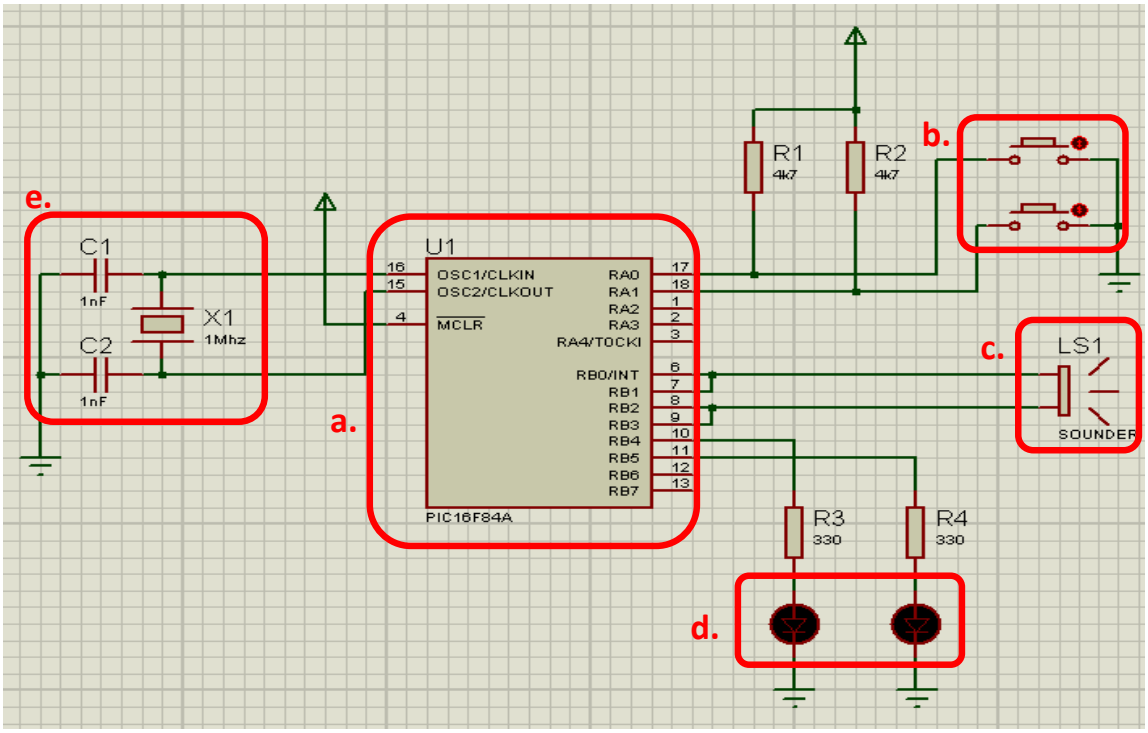

Figura 7. Esquema “Doorbell.DSN” con microcontrolador PIC 16F84A.

- Conjunto de instrucciones reducidas, solo 35.
- Ejecución de 1 instrucción por ciclo de reloj, instrucciones de salto 2 ciclos.
- Velocidad máxima de reloj de 20 Mhz, 200 ns por instrucción.
- Memoria de programa de 1024 instrucciones de 14 bits con un ciclo de vida de 10.000 escrituras.
- 68 bytes de RAM de 8 bits de ancho.
- 64 bytes de EEPROM con ciclo de vida de 10.000.000 lecturas/escrituras y retención por 40 años.
- Control individual de dirección de 13 pines de E/S.
- Temporizador/Contador de 8 bits con preescaler programable.
- Pila de 8 niveles de profundidad.
- Fuentes de interrupción:
  - por cambio de estado de pin RB0/INT.
  - por desborde de temporizador TMR0.
  - por cambio de estado de pines <7:4> del puerto PORTB
  - por finalización de escritura de memoria EEPROM.
- Voltaje de alimentación de 2.0V a 5.5V
- Corriente de 25 mA por pin (sink/source).
- Bajo consumo:
  - Menos de 2mA a 5V con reloj de 4Mhz
  - Menos de 0.5uA a 2V con reloj de 32Khz.
  - Soporte de modo SLEEP.

Figura 8. Características del microcontrolador PIC 16F84A.

5.2. Programa simple y prueba de compilación

Una vez cargado o diseñado el esquema del circuito electrónico seleccionamos el archivo C con el programa principal que compilaremos para que corra en el microcontrolador. Los pasos a realizar son:

a. Seleccione la opción del menú **"Source➔Add/Remove Source files"** para abrir la ventana de configuración (figura 9).
b. Seleccione el modelo del microcontrolador que almacenará el programa que se va a compilar en la opción **"Target Processor"**. En nuestro caso es el **"PIC16F84A"**. Tenga en cuenta que en el diseño puede haber mas de un microcontrolador y debe configurar el programa en cada uno.
c. Seleccione el compilador a utilizar en la opción **"Code Generation Tool"**. En nuestro caso utilizamos el compilador para C PICC.
d. Presione el botón **"NEW"** para abrir la ventana de diálogo para seleccionar el archivo principal. Si no tenemos uno previamente creado, seleccione la carpeta donde desea ubicarlo y escriba un nombre para que el diálogo cree un archivo en blanco. En nuestro caso utilizamos el nombre **"prueba.c"**.
e. Establecer los flags adicionales que se enviaran al compilador PICC con el valor "--chip=16f84a".
f. Presione el botón **"Ok"** para cerrar la ventana y finalizar la configuración.

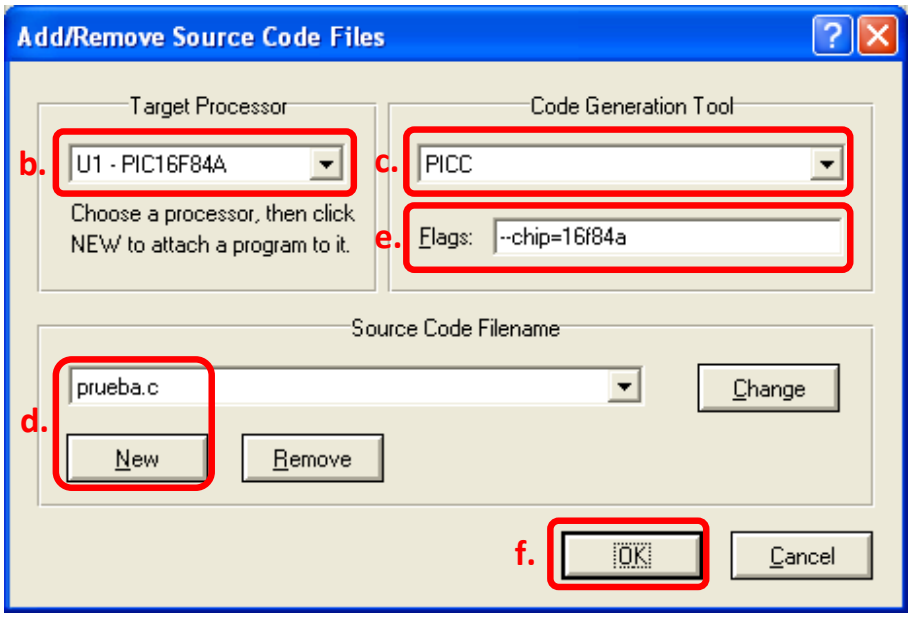

Figura 9.

g. El archivo seleccionado queda disponible como opción del menú **"Source➔Prueba.c"** (figura 9). Haga click sobre este para abrirlo con el editor de configurado. Si no se configuró uno lo abrirá con el editor de texto interno.
h. En el editor de texto escriba el cuerpo vacio de un programa en C. Recuerde los programas que normalmente compilamos en los microprocesadores, eventualmente terminan. A diferencia de estos, los programas normalmente compilados para micro controladores no terminan por lo que el programa principal es un bucle infinito (figura 10).

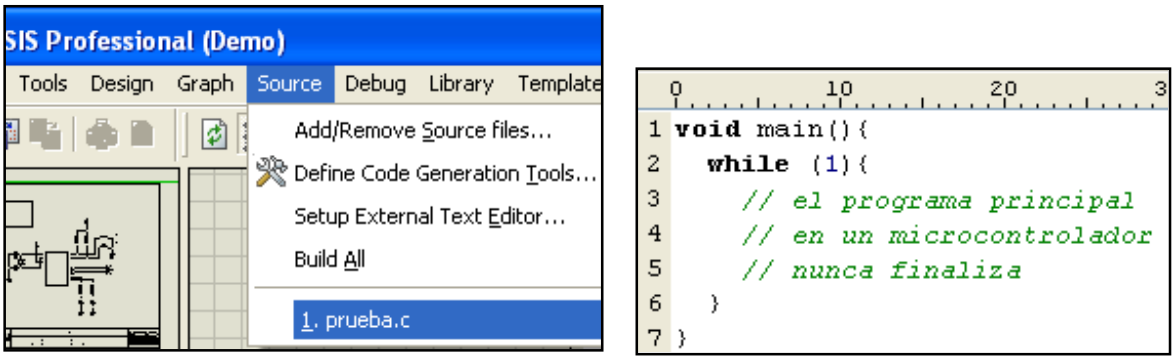

Figura 10.

i. Compilar el programa (o todos los programas si hay más de un microcontrolador) seleccionando la opción "Source➔Build All". Luego de la compilación se muestra una ventana (figura 11) con el resultado de la compilación con las siguientes secciones:
   a. Línea de comando ejecutada con los parámetros. En caso de haber errores los describe en esta sección.
   b. Resumen con información sobre la memoria como espacio que ocupa el programa, los datos, la memoria eeprom y los bits de configuración.
   c. Resultado de la compilación. En caso de ser incorrecto informa la cantidad de errores.

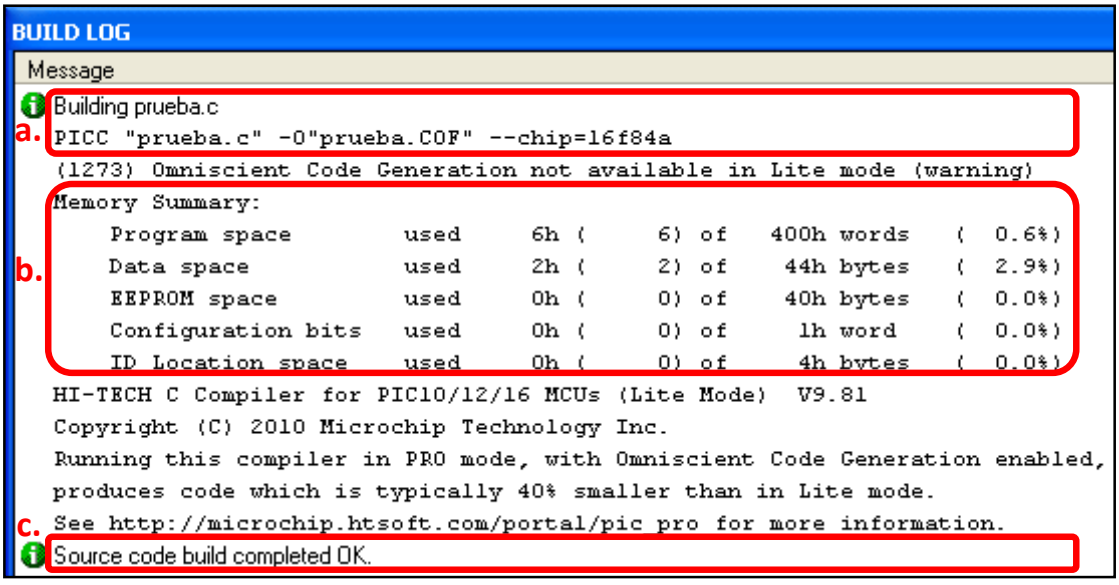

Figura 11.

5.3. Programa básico y simulación

Esquema de un programa

Para empezar a codificar un programa, la primera línea que deberá tener es: "#include <htc.h>". Esta incluye el archivo donde se encuentran algunas funciones y macros importantes junto con la definición de todas las constantes que representan registros, bits de registros y valores comunes que estos pueden tomar.

En las primeras líneas de la función del programa se configuran e inicializan todos los registros que controlan los módulos de hardware del microcontrolador que se desean utilizar. Luego de la inicialización se desarrollan en un bucle infinito las instrucciones que resuelven y cumplen con el objetivo del programa.

Como los microcontroladores con los que trabajaremos pueden generar interrupciones, existe una directiva "interrupt" para definir una función que será invocada cada vez que se produzca un evento de interrupción programado.

Un programa simple

En esta sección se propone, sobre el esquema "Doorbell.dsn", realizar un programa simple en C, compilarlo y simularlo para observar su funcionamiento. Como ejercicio se plantea realizar un programa que haga parpadear 1 vez por segundo el led conectado al PIN RB4. En otras palabras este led permanecerá 500 ms (0.5 segundos) apagado y 500 ms encendido.

El pseudo-código del programa sería:

1. *Inicializar el bit 4 del puerto B como salida para configurar la electrónica asociada al pin RB4*
2. *Repetir:*
3. *Invertir el bit 4 de PORTB:*
4. *Esperar 500 ms*

En la primera línea del pseudo-código, la configuración del puerto **B** (registro **PORTB**) para el **PIC 16F84A** consiste en indicar para cada bit si se lo va a utilizar como entrada o como salida. Esto se hace a través del registro **TRISB** donde cada bit de este determina la dirección de cada bit de **PORTB**. Si en un bit de **TRISB** se establece un **"1"** entonces el mismo bit de **PORTB** se configura la electrónica para que funcione como entrada digital. Si en cambio se establece un **"0"** entonces se configura para que funcione como salida digital.

En la línea 3, se invierte el cuarto bit del puerto **B** del microcontrolador. La modificación de un bit de este registro provoca la modificación del bit asociado del puerto **B**. Por ejemplo, si se escribe un **"1"** en un bit se envía un nivel alto (ej: 5V) al pin asociado del puerto **B**. Si bien **PORTB** es un registro de 8 bits y normalmente su lectura y escritura se hace de a byte, es posible leer y escribir solo un bit utilizando las constantes **RBx**, donde x es el número de bit.

En la línea 4, se realiza una espera de medio segundo. En el archivo **htc.h** se incluyen las funciones **__delay_ms** y **__delay_us**. La primera función espera un parámetro con la cantidad de milisegundos a esperar, mientras de la segunda espera la cantidad de microsegundos a esperar. Como estas funciones se basan en la espera de ciclos de instrucción, para sincronizarlas con el tiempo es necesario declarar la constante **_XTAL_FREQ** con la frecuencia del oscilador/reloj de forma de relacionar la cantidad de instrucciones que se ejecutan por segundo.

A continuación se muestran cuatro soluciones al ejercicio planteado. Observar bien las diferencias entre cada solución.

```c
#include <htc.h>

// constante requerida para funciones
// de espera de tiempo
#define _XTAL_FREQ  1000000 //1MHz osc

void main(){
  // configura en binario el bit4 de
  // registro PORTB(RB4) como salida,
  // los demas bits como entradas
  // bits   76543210
  TRISB = 0b11101111;
  while (1){
    // invierte bit4, xor de los bits
    // donde la mascara binaria es 0
    PORTB = PORTB ^ 0b00010000;
    // espera 500 milisegundos
    __delay_ms(500);
  }
}
```

```c
#include <htc.h>

// constante requerida para funciones
// de espera de tiempo
#define _XTAL_FREQ  1000000 //1MHz osc

void main(){
  // configura en binario el bit4 de
  // registro PORTB(RB4) como salida,
  // los demas bits como entradas
  // bits   76543210
  TRISB = 0b11101111;
  while (1){
    // pone en 1 bit4
    PORTB = PORTB | 0b00010000;
    // espera 500 milisegundos
    __delay_ms(500);
    // pone en 0 bit4
    PORTB = PORTB & 0b11101111;
    // espera 500 milisegundos
    __delay_ms(500);
  }
}
```

```c
#include <htc.h>

// constante requerida para funciones
// de espera de tiempo
#define _XTAL_FREQ  1000000 //1MHz osc

void main(){
  // configura en binario el bit4 de
  // registro PORTB(RB4) como salida,
  // los demas bits como entradas
  // bits   76543210
  TRISB = 0b11101111;
  while (1){
    // pone en 1 bit4
    RB4=! RB4;
    // espera 500 milisegundos
    __delay_ms(500);
  }
}
```

```c
#include <htc.h>

// constante requerida para funciones
// de espera de tiempo
#define _XTAL_FREQ  1000000 //1MHz osc

void main(){
  // configura en binario el bit4 de
  // registro PORTB(RB4) como salida,
  // los demas bits como entradas
  // bits   76543210
  TRISB = 0b11101111;
  while (1){
    // pone en 1 bit4
    RB4=1;
    // espera 500 milisegundos
    __delay_ms(500);
    // pone en 0 bit4
    RB4=0;
    // espera 500 milisegundos
    __delay_ms(500);
  }
}
```

Simulación y ejecución del programa

Ya sabemos que podemos compilar un programa desde la opción **"Source➔Build All"**. Una vez que el programa este compilado estamos en condiciones de simularlo para interactuar y/o ver los resultados.

En la parte inferior de la ventana principal (figura 11) o desde el menú "Debug" se encuentran los comandos que permiten controlar la simulación. Los comandos que se pueden ejecutar son:

a. Comenzar una simulación: inicia la simulación. Si fuera necesario compila el programa si este fue modificado.
b. Pausar una simulación: detiene temporalmente una simulación.
c. Avanzar una "paso" una simulación: avanza una simulación pausada 50 milisegundos.
d. Detener una simulación: finaliza una simulación.

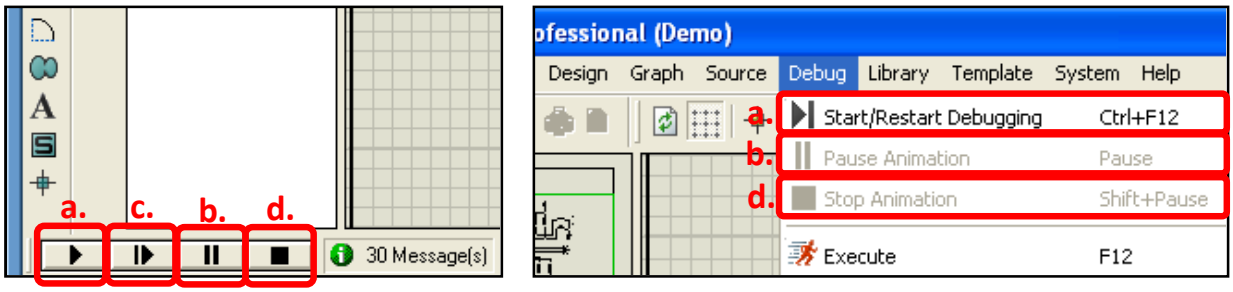

Figura 11.

Para comenzar la simulación de nuestro programa seleccionamos el comando desde el menú "Debug➔Start/Restart Debugging". En la barra inferior de la ventana principal se muestra el tiempo de simulación y observaremos que en el intervalo de un segundo el led conectado al pin RB4 enciende y apaga una vez.