Entorno de Desarrollo uVision 4 (I)

Introducción al entorno de desarrollo KEIL µVision® 4 para la familia de microcontroladores LPC17xx

Versión 3.0

Sistemas Electrónicos Digitales

Universidad de Alcalá
Departamento de Electrónica

1 Introducción al documento

Este documento presenta una introducción al manejo del entorno de desarrollo µVision© de Keil©, una herramienta software para el desarrollo de proyectos basados en microcontroladores. Entre otras cosas, nos permite compilar, simular, depurar y cargar el código en nuestro microcontrolador (LPC1768, en el caso que nos ocupa).

Se hace una introducción a este entorno de desarrollo a través de ejemplos sencillos que ilustran la creación de un proyecto, su simulación, la utilización del analizador lógico para viasualizar señales o variables en el dominio del tiempo, uso de puertos y breakpoints.

2 Índice de contenidos

3 ¿Qué es KEIL µVISION?

El entorno de desarrollo µVision© de Keil© es una herramienta IDE (Entorno de Desarrollo Integrado) profesional para el desarrollo de aplicaciones basadas en microcontrolador. Este entorno engloba una serie de componentes mediante los que se pueden llevar a cabo los procesos inherentes al desarrollo y depuración de aplicaciones. La figura 1 muestra los módulos que integran este entorno de desarrollo.

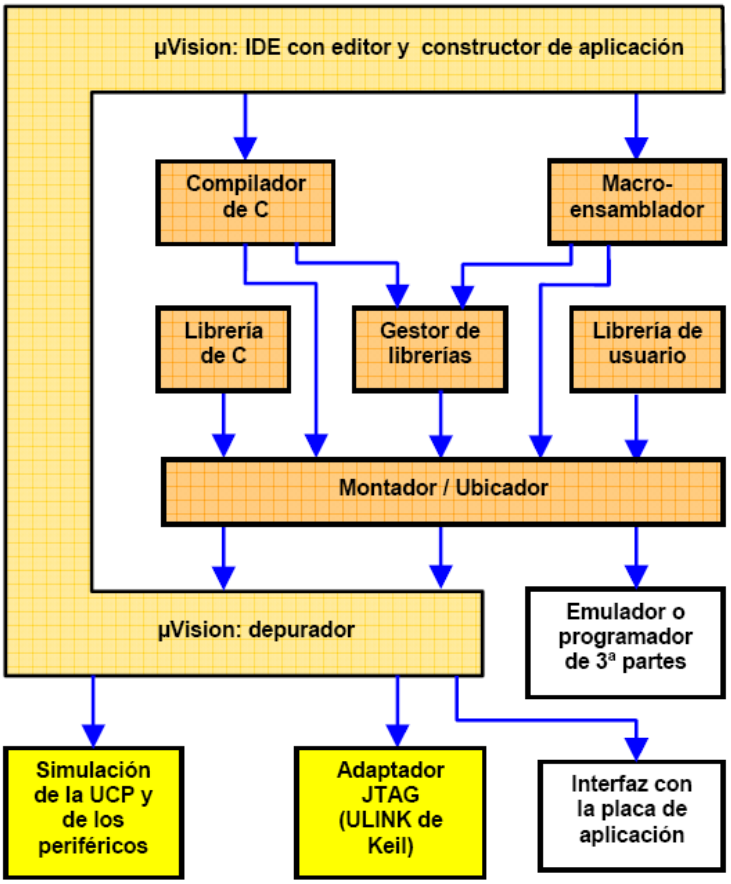


Figura 1. Componentes del sistema µVision de Keil.

La interfaz de este entorno nos permite definir las características de nuestro proyecto, como el tipo de procesador que utilizaremos, tipo de optimización a la hora de compilar, tipos de ficheros al ensamblar (o compilar) y montar, etc. µVision nos permite, asimismo, depurar el código desarrollado. Para ello, dispone del modo depuración, mediante el cual vuelca el código en la placa de desarrollo o placa de aplicación final, ejecutándose el código en esta y comunicándose con µVision. Para ello, el PC y la placa de desarrollo se conectan mediante un cable de conversión USB a JTAG. También es posible realizar la depuración sin un soporte físico o placa de desarrollo que ejecute

realmente el código. Para ello, µVision integra un potente simulador con el que se puede simular el funcionamiento del procesador y todos los periféricos que éste integre.

Los ficheros fuente creados con el IDE de µVision se pasan al compilador o al macroensamblador para ser procesados y obtener los ficheros objeto reubicables, que luego pasan al módulo montador para obtener el fichero ejecutable. Los ficheros ejecutables (.HEX) se utilizan, por ejemplo, para programar la memoria Flash del microcontrolador.

El gestor de librerías permite la utilización de librerías de módulos objeto previamente generadas con el compilador o el macroensamblador. µVision también incluye una serie de librerías, por ejemplo, las que incluyen ciertas funciones matemáticas, etc.

4 Creación de un proyecto

La figura 2 esquematiza de una manera muy general los pasos que debemos seguir en el diseño de un programa para resolver un determinado problema con un microcontrolador.

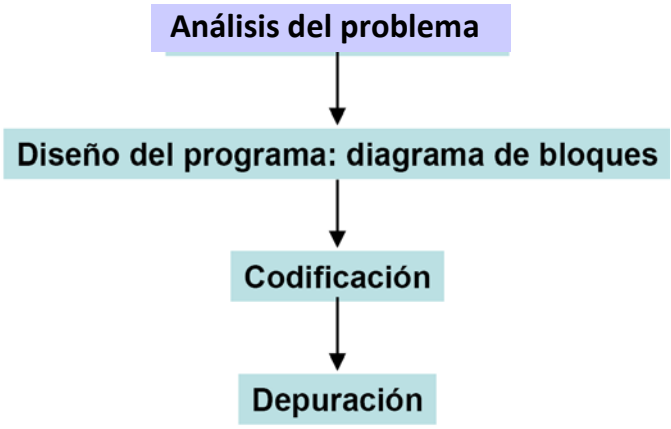


Figura 2. Pasos en el diseño de un programa para microcontrolador.

A continuación, se ilustrará cómo crear un proyecto en µVision con el programa fuente que previamente hemos escrito con un editor de texto o desde el propio editor de µVision, luego se compilará este programa y se simulará para irlo depurando, si fuera necesario. Por último, descargaremos el programa ejecutable depurado sobre la memoria flash del microcontrolador ARM.

Comenzamos creando un nuevo proyecto, siguiendo los pasos indicados en la figura 3. Si ya se hubiese creado un proyecto anteriormente, entonces habría que abrirlo. A

continuación se abre una ventana donde escribimos el nombre del proyecto (Ejemplo1) y lo guardamos (fig 3B). Luego se abrirá otra ventana (fig 3C) en la que hay que seleccionar el microcontrolador que se vaya a utilizar (LPC1768). Para ello se selecciona el fabricante y se hace clic en el símbolo '+' a su izquierda, con objeto de que aparezcan todos los modelos que corresponden a ese fabricante. En la siguiente ventana (fig 3D) que se abre se pregunta si se desea copiar al proyecto un fichero con una plantilla para iniciar el código que se vaya a escribir (responder SI).

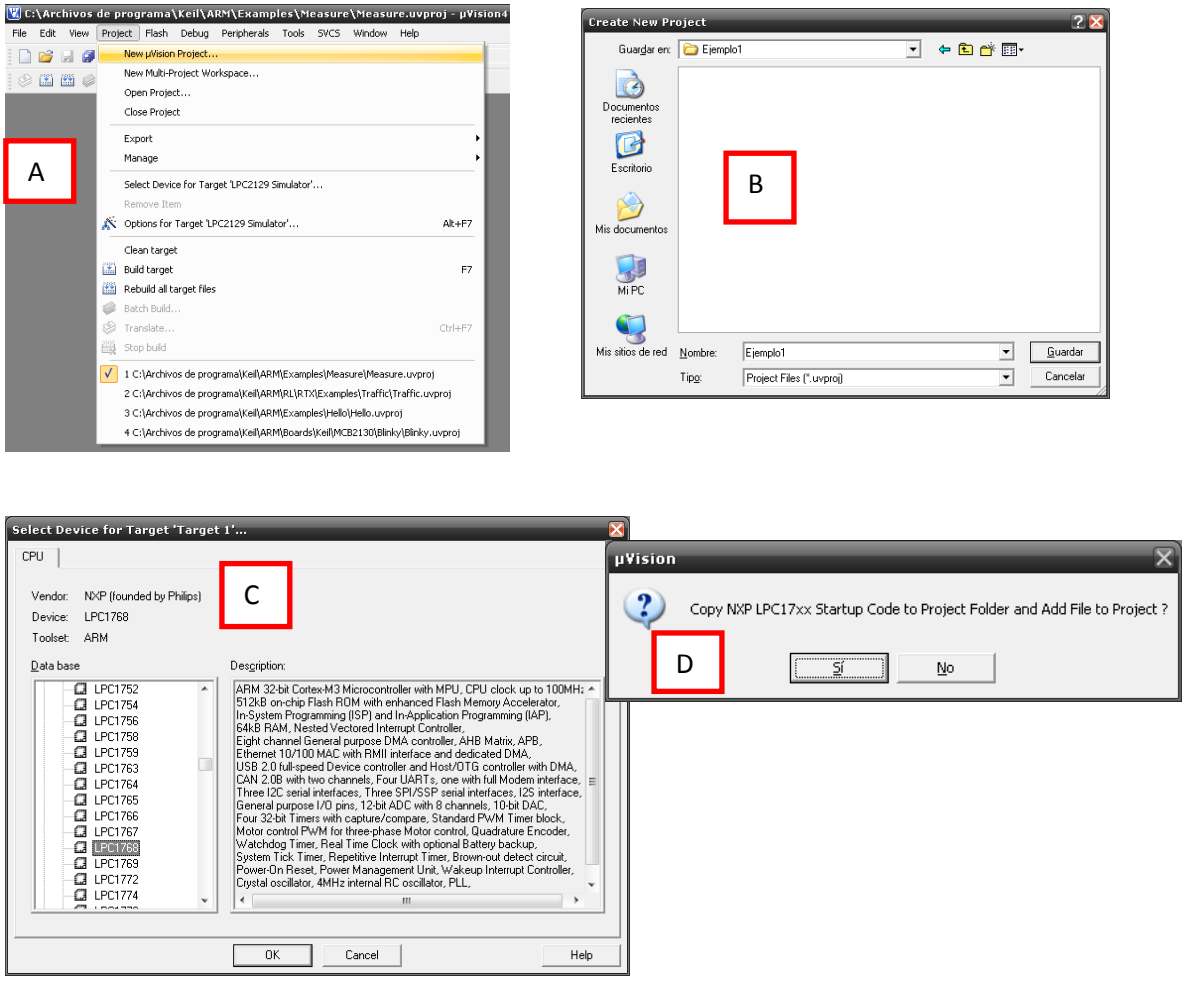


Figura 3. Ventanas para crear un nuevo proyecto, seleccionar el fabricante y el dispositivo.

Este fichero que sirve como plantilla para iniciar el código se llama *startup_LPC17xx.s* y tiene las siguientes características:

- Es un módulo en ensamblador propio de cada microcontrolador
- Se ejecuta a partir de la dirección apuntada por el vector de reset
- Inicializa los punteros de pila (SP) para cada modo de funcionamiento
- Inicializa la tabla de vectores de interrupción

- Llama a la función *SystemInit()* que se encuentra en el fichero *system_LPC17xx.c*. La función *SystemInit()* básicamente configura las señales de reloj de la CPU y de los distintos periféricos.
- Llama a la función *main.c* de nuestro proyecto.

4.1 Configuración del entorno

Seleccionar el compilador adecuado en la ventana de Configuración del Entorno (figura 4). Para ello, tenemos dos opciones:

1. Compilador de ARM: Versión de evaluación restringida a 32Kbytes de código. Para códigos más amplios es necesario comprar la licencia. Esta es la opción que seleccionaremos.
2. Compilador GCC de GNU: Versión libre.

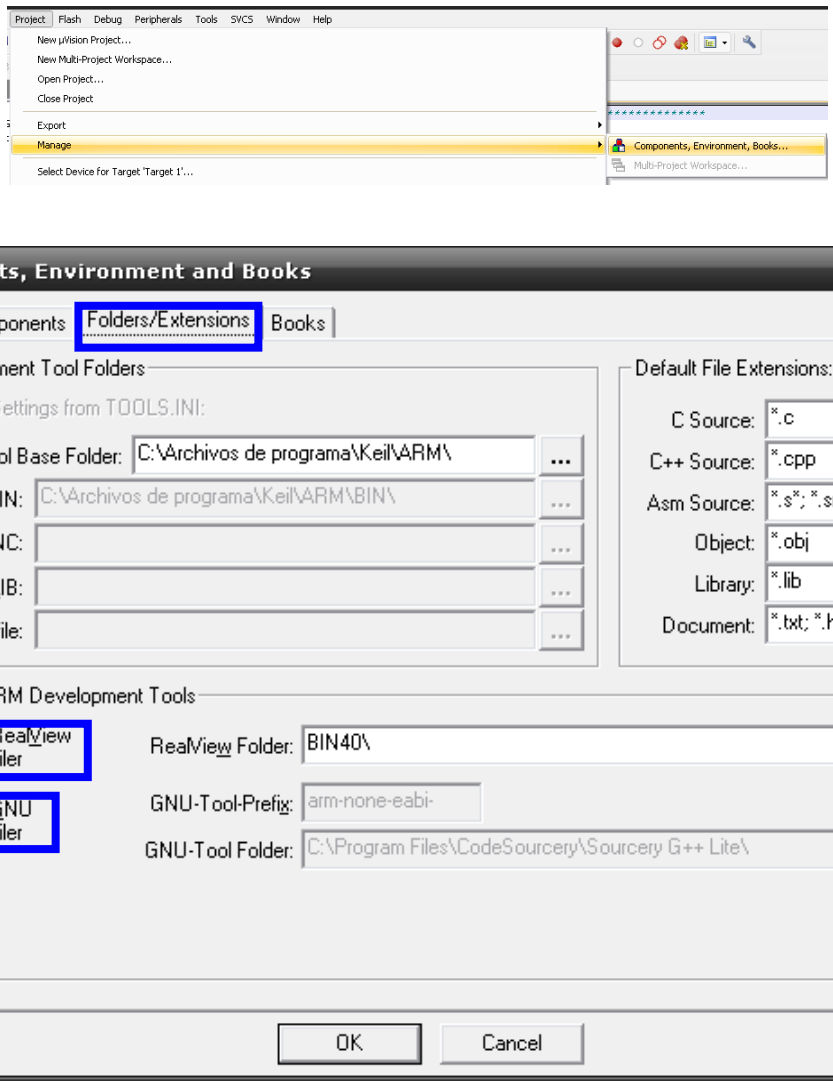


Figura 4. Ventana de configuración de entorno.

4.2 Configuración de opciones

Renombrar la entrada Target 1 como LPC1768 (haciendo click sobre ella, tras estar previamente seleccionada). Hacer click con el botón derecho del ratón sobre la entrada **LPC1768** para que aparezca su menú contextual y seleccionar **Options for Target 'LPC1768'…** (figura 5).

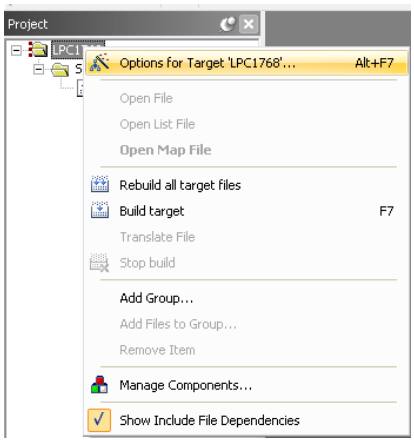


Figura 5. Menú contextual de LPC1768.

Una vez aparezca la ventana 'Options for Ejemplo1' configurar las opciones que aparecen a continuación (figuras 6A, 6B y 6C). Seleccionamos la velocidad de cristal que viene por defecto (12 MHz), ya que será ese el valor del cristal de nuestra tarjeta en el resto de prácticas.

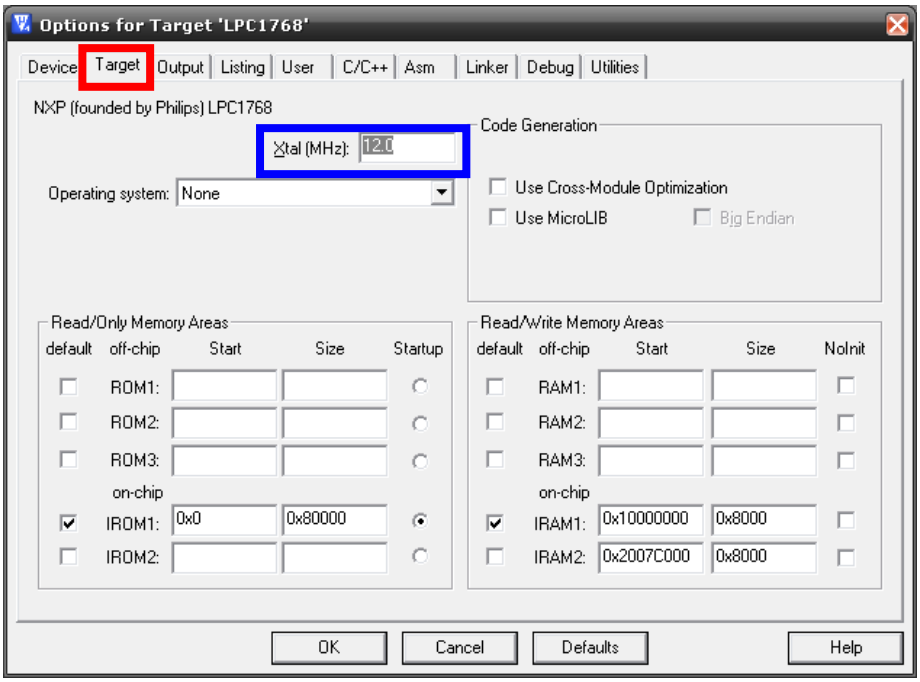


Figura 6A. Opciones para la pestaña Target.

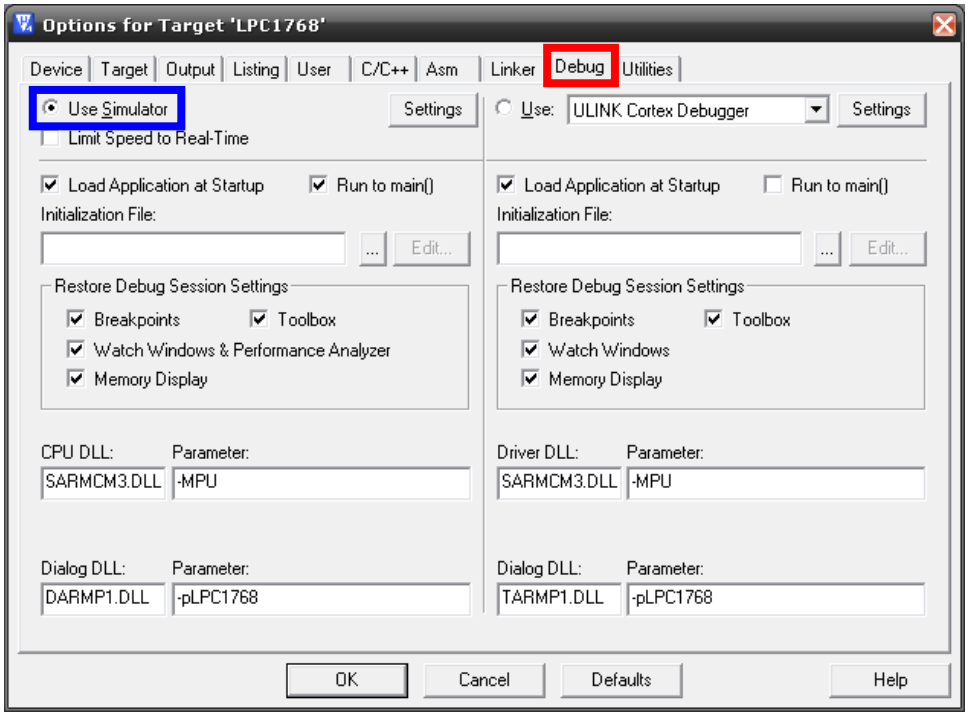


Figura 6B. Opciones para la pestaña Debug. Observar que esta práctica sólo será simulada, y no volcaremos el código sobre ninguna placa externa.

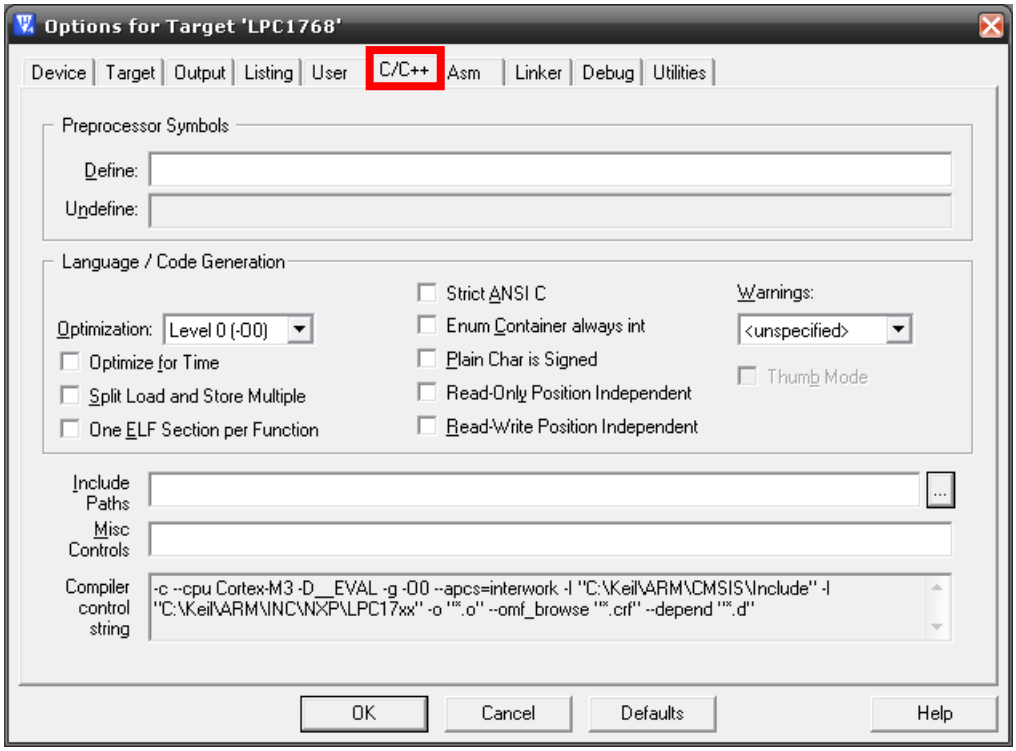


Figura 6C. Opciones para la pestaña C/C++.

4.3 Agregar ficheros al proyecto

Renombrar la entrada del árbol de proyecto *Source Group 1* como *StartUp*. Además de contener esta entrada de proyecto al fichero *startup_LPC17xx.s*, incluiremos otro fichero llamado *system_LPC17xx.c* necesario para la configuración inicial del microcontrolador. Para ello, proceder como se ilustra en las figuras 7 y 8 (*system_LPC17xx.c* se encuentra en la ruta C:\Keil\ARM\Startup\NXP\LPC17xx\, pero es muy importante copiar este fichero en el mismo directorio que nuestro proyecto e incluirlo desde ahí, para así no correr el riesgo de modificar el original). Ahora nos situamos sobre la entrada LPC1768 del árbol de proyecto, hacemos click con el botón derecho y seleccionamos *Add Group…*. Se creará una entrada llamada *New Group* que renombraremos como *SourceFiles*. Del mismo modo que hemos incluido el fichero *system_LPC17xx.c* a la entrada *StartUp*, incluiremos nuestro fichero fuente *main.c*, pero en la entrada *SourceFiles* que acabamos de crear (figura 10). Es necesario saber dónde hemos almacenado previamente los ficheros que deseemos agregar para así poder localizarlos. NOTA: podemos editar previamente y guardar el fichero main.c con el listado de código que aparece en el siguiente listado.

```
void CalculaCuadrado(int *origen, int *destino, int n)
{
   int i;
   for(i=0;i<n;i++)
      *(destino+i)=(*(origen+i))*(*(origen+i));
}

main()
{
  static char var1;
  int Tabla1[5]={2,3,9,12,15};
  int TablaCuadrados[5];
  static int i;

  var1=0x12;
  CalculaCuadrado(Tabla1, TablaCuadrados, 5);
  while(1)
  {
     i=0;
     i=1;
  }
}
```

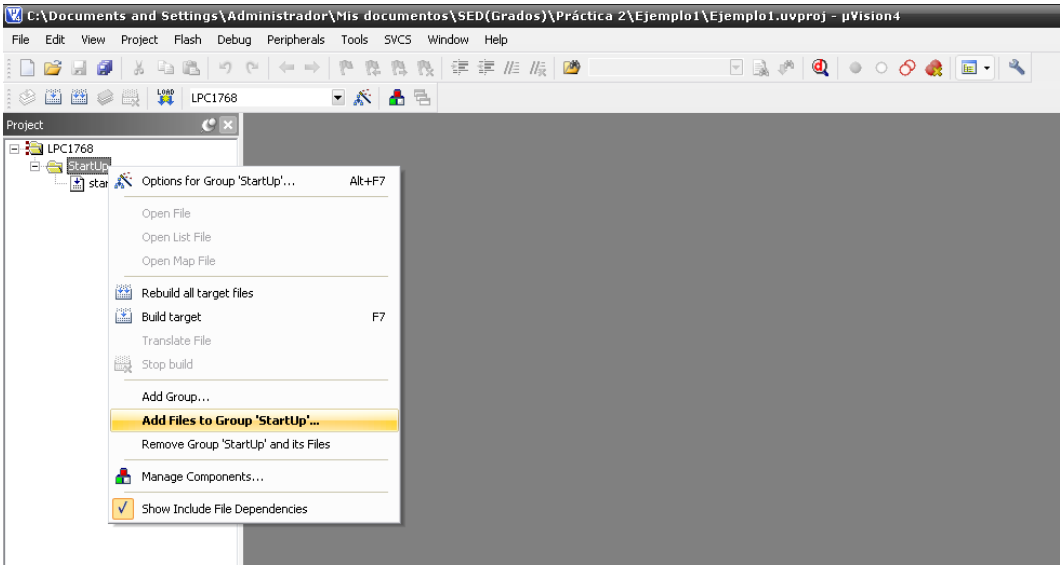


Figura 7. Menú contextual de la entrada 'StartUp'.

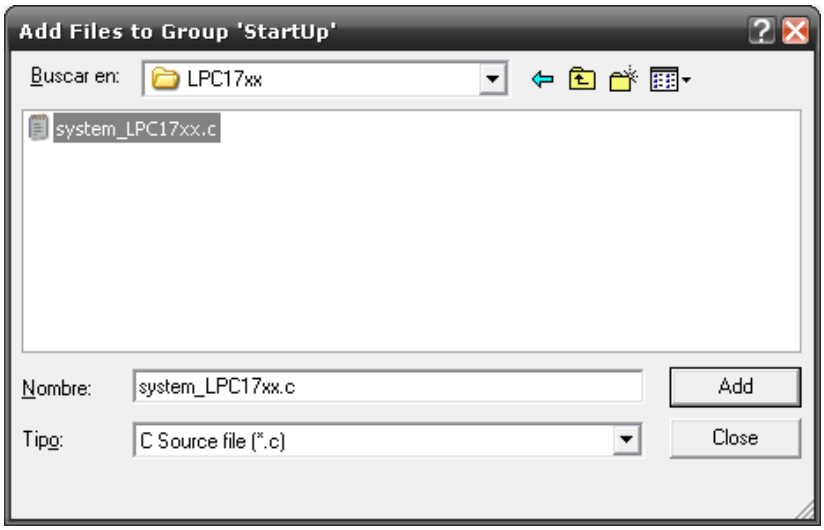


Figura 8. Fichero system_LPC17xx.c a ser agregado al proyecto.

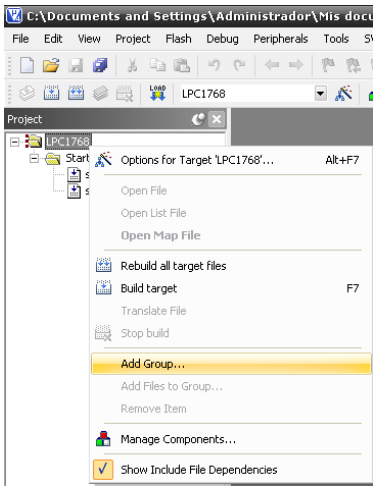


Figura 9. Agregamos grupo SourceFiles a la entrada LPC1768.

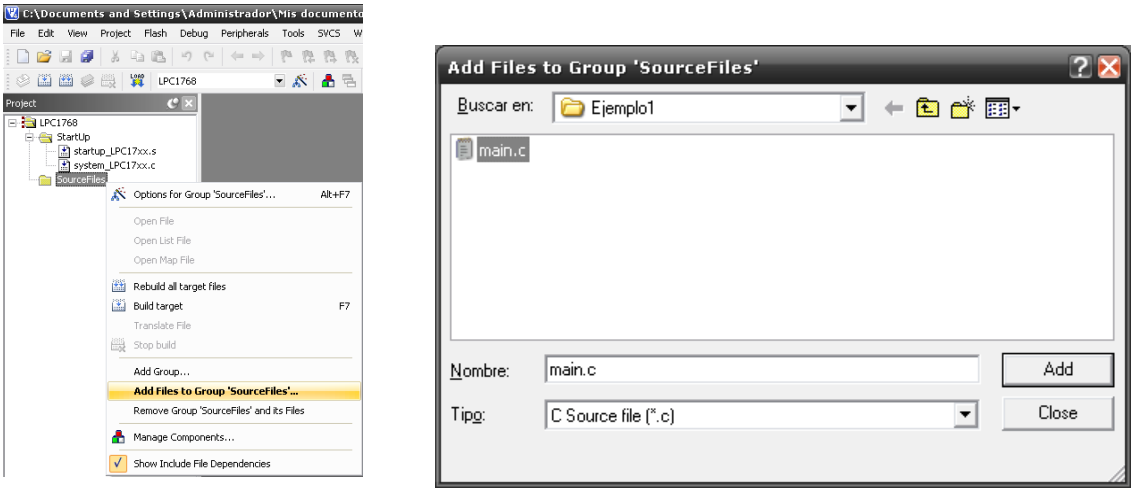


Figura 10. Agregamos main.c a la entrada SourceFiles.

En la figura 11 se muestra la estructura completa de nuestro proyecto ejemplo y el resultado de su compilación y enlazado mediante el comando Build (F7) o haciendo click sobre cualquiera de los iconos . Obsérvese la ventana inferior *Build Output* que nos indica si se ha producido algún error durante la compilación y también nos informa sobre el tamaño que ocupa nuestro programa.

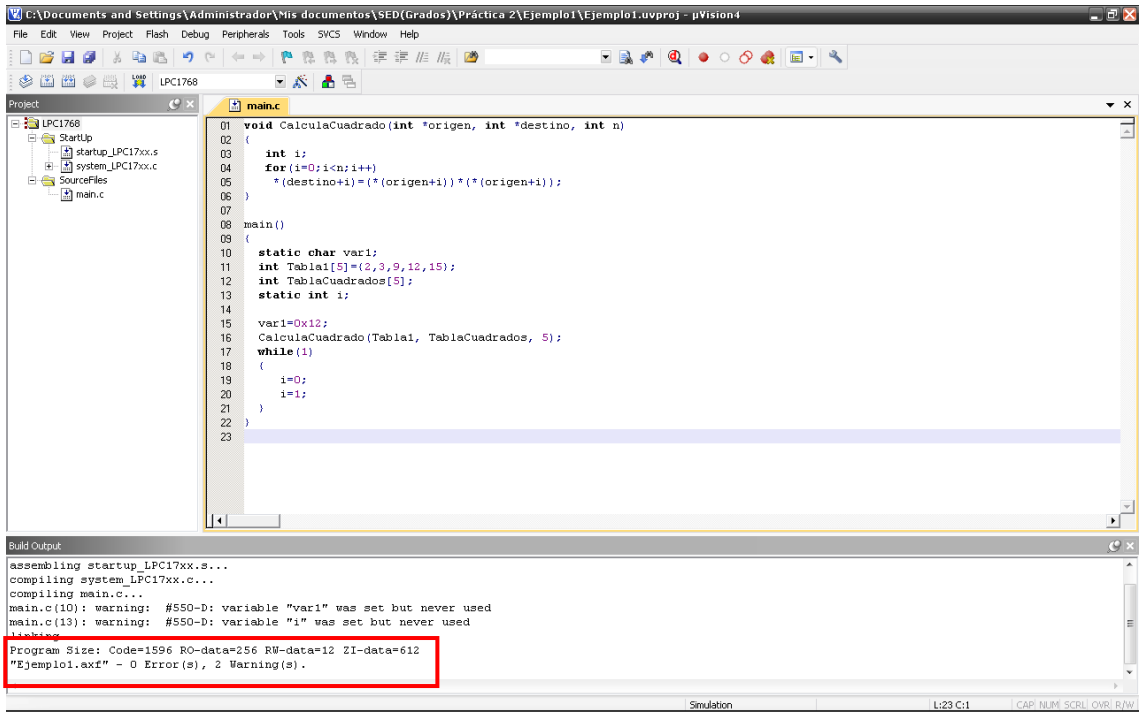


Figura 11. Compilado y "*linkado*" del proyecto (ZI-data: Zero Initialized Data, RO-data son constantes. Tamaño total de RAM = RW data + ZI data. Tamaño total de ROM = Code + RO data).

4.4 Simulación

Ahora tenemos la posibilidad de ejecutar el programa en el simulador que incluye µVision o de cargarlo a nuestra placa de desarrollo externa para que corra físicamente en el microcontrolador. De momento, haremos uso del simulador y veremos algunas de sus posibilidades interesantes. Para ello, arrancar el simulador haciendo click sobre el botón (F5) y aparecerá la ventana de la figura 11. Estos son algunos comandos que nos resultarán de utilidad:

- Ejecutaremos el programa paso a paso haciendo un click sobre (F11) por cada instrucción.
- Si deseamos que se ejecute con un simple click toda una función, sin necesidad de ejecutar paso a paso cada una de las instrucciones que la componen, hacemos click sobre (F10).
- Para ejecutar todo el código hasta la línea donde hemos colocado el cursor, click sobre el icono (Ctrl.+F10).
- Para hacer reset y llevar la ejecución al comienzo del código, .
- Si lo que queremos es ejecutar todo el código de una vez, (F5) y para detenerlo.

A continuación se comentan algunas observaciones sobre las ventanas que aparecen en la figura 11:

- En la ventana *Disassembly* aparece el código traducido a ensamblador a partir del código fuente en c de las funciones *startup_LPC17xx.s* y *main.c*.
- En la zona de la izquierda aparece la ventana *Registers*, con el nombre y valor actual de todos los registros del microcontrolador. Se observa una marca grisácea sobre el/los registros que se ven modificados cada vez que ejecutamos un paso con Step(F11).
- Se puede apreciar una flecha o cursor de color amarillo que apunta siempre a la instrucción que se va a ejecutar (tanto en la ventana de ensamblador como en la de código c).
- También se muestra el tiempo transcurrido desde que se inició la ejecución del programa hasta el comienzo de la instrucción actual (parte inferior derecha).
- En la ventana *Locals* (parte inferior derecha) vemos el valor actual que van tomando las variables de nuestro programa.

En la ventana *Locals* también podemos comprobar que la variable *Tabla1* se ha almacenado a partir de la dirección de memoria 0x1000025C. Para mostrar el contenido de esta zona de memoria hacer click sobre la pestaña *Memory 1* e introducir en el campo *Address:* el valor 0x1000025C. Colocar el cursor delante de la linea *CalculaCuadrado(Tabla1, TablaCuadrados, 5)* y ejecutar hasta la posición del cursor (Ctrl+F10). En este momento ya está inicializada la variable *Tabla1* y en la figura 12 se puede ver el contenido de la zona de memoria donde se ha almacenado esta variable.

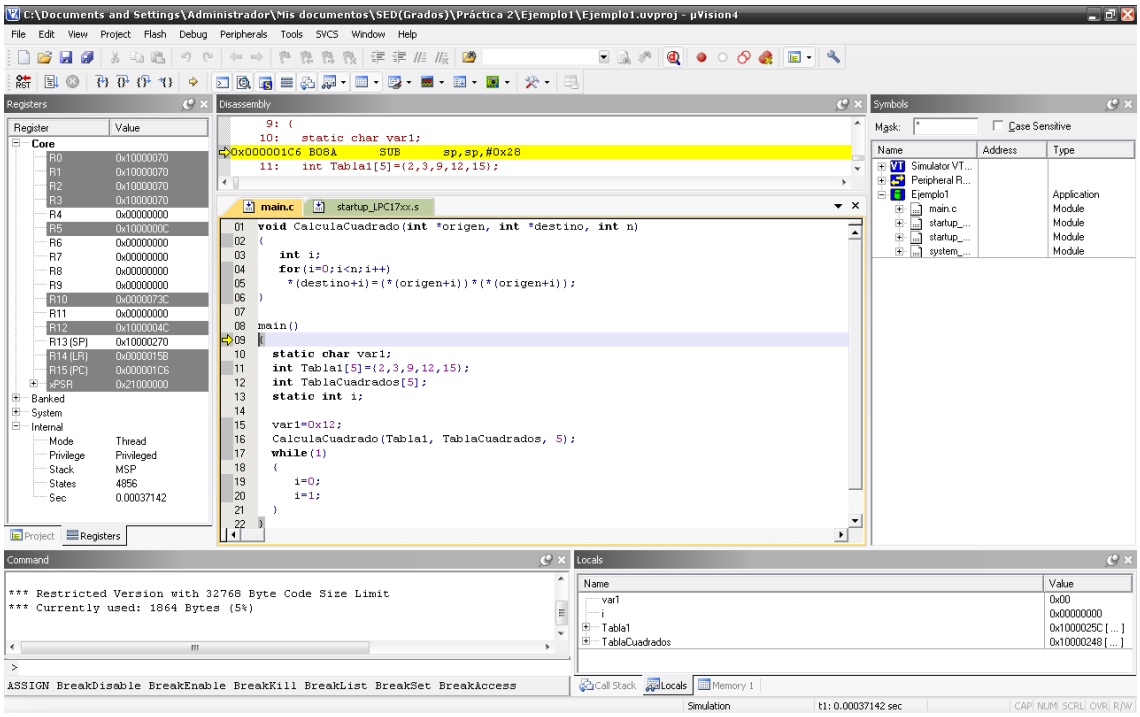


Figura 11. Ventana inicial del simulador.

Resulta interesante observar que cada valor de *Tabla1* ocupa 4 bytes porque se declaró como array de enteros. También se puede comprobar que el byte menos significativo se almacena en la parte más baja de memoria. Podemos modificar a mano el valor de una posición de memoria haciendo doble click sobre ella y editando su contenido.

Se puede visualizar el contenido de una variable determinada simplemente situando el cursor sobre ella en la ventana *Symbols* o en la ventana del código fuente (en este último caso, la variable debe pertenecer a la función que se esté ejecutando en ese momento).

Observar dónde se almacena el array *TablaCuadrados[5]* y comprobar como se actualiza mientras el programa se ejecuta paso a paso. Hacer lo mismo con la variable *var1*. **IMPORTANTE:** Para visualizar la variable *var1* es necesario hacer dos cosas: definirla como estática (*static char var1*) y en las opciones del compilador seleccionar *Optimization: Level 0*. Si no se selecciona este nivel de optimización, tampoco se ejecutará el bucle *while*, porque el compilador ve una cadena de operaciones sin mucho sentido: *i=0; i=1;* que las elimina del código compilado. Para visualizar la variable *var1* podemos hacer click con el botón derecho sobre ella y seleccionar *Add 'var1' to…→Watch1*, o bien escribiendo *&var1* en el campo *Address* de la ventana *Memory 1*.

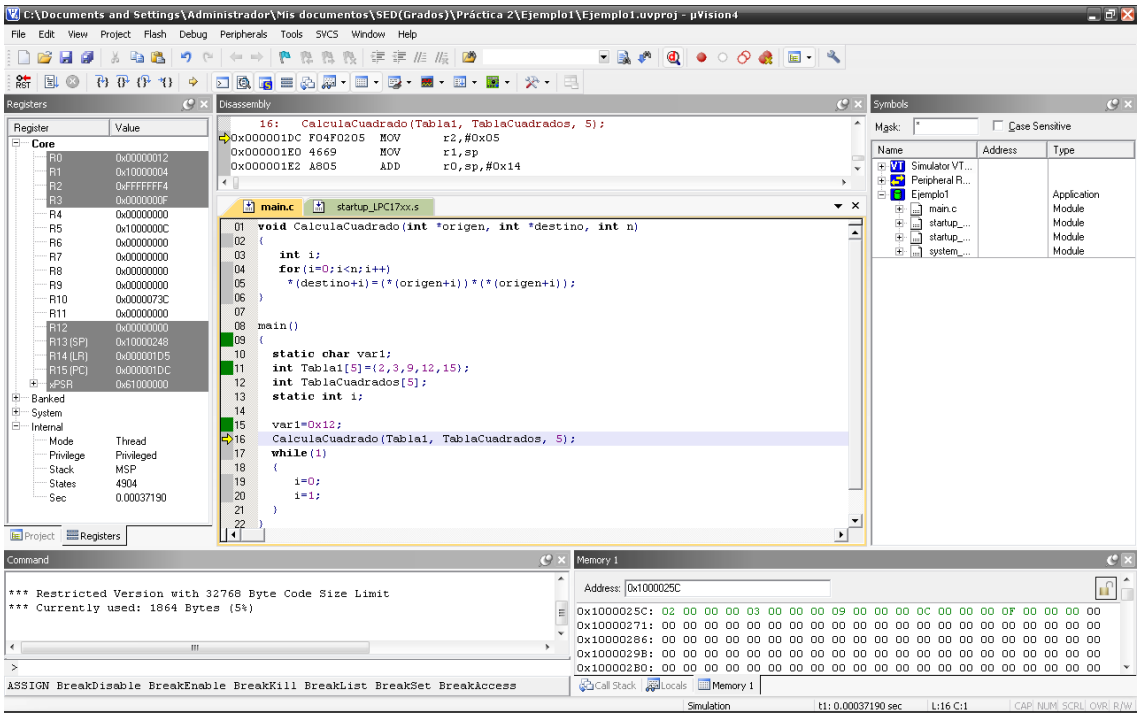


Figura 12. Contenido de la memoria donde se ha almacenado la variable Tabla1.

4.5 Utilización del analizador lógico

El simulador dispone de un analizador lógico que permite ver variables y señales de puertos en el dominio del tiempo. Para visualizar las variables en el analizador, estas deben ser estáticas o globales; no locales. A continuación se muestran los pasos para visualizar con el analizador lógico la variable *i* que se actualiza constantemente en el bucle *while*:

- Hacer click sobre el icono  para visualizar el analizador lógico.
- Introducir la variable *i* en el analizador, simplemente arrastrándola y soltándola en la ventana del analizador. Con un conjunto de variables, procederíamos del mismo modo, arrastrándolas y soltándolas de una en una.
- Ajustar la escala vertical haciendo click en el botón *Setup…* de la ventana del analizador y rellenar con *0x0* y *0x1* las cajas de texto *Min:* y *Max:*, respectivamente. Aceptar y cerrar haciendo click en el botón *Close*. Nota: también se puede ajustar la escala vertical abriendo el menú contextual sobre el analizador y seleccionando la opción 'Bit'.
- Ajustar la escala de tiempo a 20 ns/div (*Grid:* 20 ns) haciendo varios clicks sobre el botón *In* del *Zoom*.
- Habilitar la casilla *Signal Info* para obtener información al colocar el puntero sobre la señal.

- Ejecutar paso a paso hasta entrar en el bucle *while* y observar la señal cuadrada que aparece en la ventana del analizador lógico a medida que cambia el valor de la variable *i* en el bucle *while*.
- Hacer click sobre cualquier flanco de la señal para colocar una marca y luego colocar el puntero del ratón sobre otro flanco para visualizar la diferencia de tiempos.

NOTA: otra forma de especificar directamente un bit de una variable en el analizador lógico, por ejemplo el bit 0 de la variable *i*, sería la siguiente: hacer click en  de la ventana del analizador lógico (figura 15) e introducir el texto *i.0*.

En la figura 13 aparece un ejemplo que muestra una porción de la señal sobre el analizador configurado con una escala de 0.1us/div. Es importante resaltar que el microcontrolador ofrece varias posibilidades como fuente de reloj para su funcionamiento. Tras el reset, queda seleccionado como reloj por defecto la salida de un oscilador RC de 4MHz que posee internamente, pero la selección de un cristal de 12 MHz (figura 6A), junto a los ficheros que hemos incluído para el arranque en la entrada StartUp del árbol de proyecto nos garantizan que la velocidad de reloj interna en la CPU sea de 100 MHz.

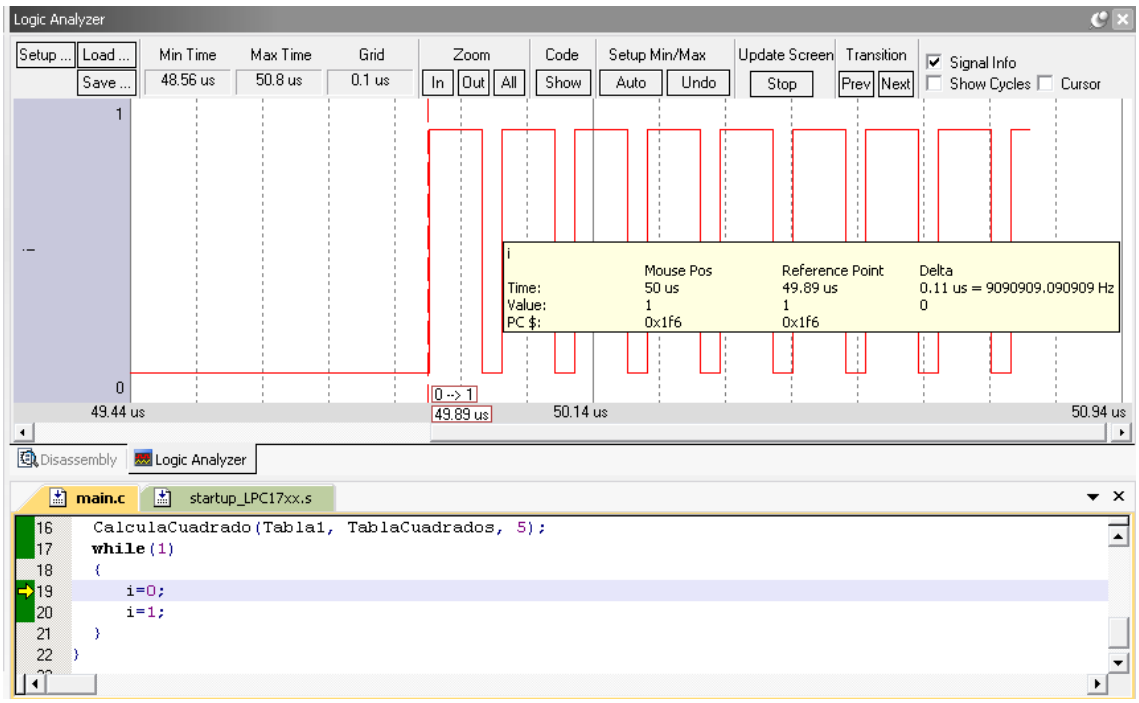


Figura 13. Ejemplo que visualiza la variable i sobre el analizador lógico.

5 Ejemplo de utilización de los puertos

En este caso se presenta un ejemplo que hace uso del puerto 0 para activar cuatro pines del mismo de manera secuencial, seleccionando el sentido de activación de acuerdo al

valor de una variable que podrá ser modificada de forma interactiva en el simulador. Las señales serán visualizadas mediante el analizador lógico.

Comenzar cerrando el proyecto actual (*Project→Close Project*) y creando uno nuevo como ya se ha explicado en los primeros apartados de esta guía. Llamar *Ejemplo2* al nuevo proyecto y almacenarlo en una carpeta diferente, ya que también llamaremos *main.c* al fichero fuente que usará el nuevo proyecto, pero será uno diferente que debemos tener preparado en la nueva carpeta (no es necesario que sea definitivo, puesto que siempre podemos editarlo en la ventana del entorno de µVision).

5.1 Código fuente

```
#include <LPC17xx.H>     /* Definiciones para el LPC17xx */
#define CONST_25ms_100MHz 500000

int32_t sentido;

//Función que establece el tiempo que una salida del puerto
//permanecerá activada.
void Retardo(int32_t tiempo)
{
  int32_t i;

  //El bucle for tarda 25 ms a 100MHz.
  for(i=0;i<tiempo;i++);
}

int main(void)
{
  int32_t n;

  // Los pines P0.[0..3]  configurados como salidas,
  //           P0.[4..31] configurados como entradas.
  LPC_GPIO0->FIODIR=0x0000000F;

  // Se ponen a cero los pines P0.[0..3].
  LPC_GPIO0->FIOCLR=0x0000000F;

  while(1){
    if(sentido) {
      n=n<<1;
        if(n==(1<<4)) n=1;
        }
      else{
      n=n>>1;
        if(n==0) n=(1<<3);
        }
     // Activar la salida que corresponda.
    LPC_GPIO0->FIOPIN=(n<<0);
    //Retardo de 25 ms.
    Retardo(CONST_25ms_100MHz);
      }
}
```

5.2 Visualización de puertos en el analizador lógico

Una vez compilado el projecto sin errores arrancar el simulador, mostrar la ventana del analizador lógico y configurarlo para visualizar los cuatro bits de menor peso del puerto 0 P0.[3..0]. Para poder visualizar estos cuatro bits, primero buscamos el puerto 0 (*FIO0PIN*) haciendo click en el símbolo '+' de la entrada *Peripheral Registers* (ventana *Symbols*) para expandirla (figura 14). Como *FIO0PIN* hace referencia conjunta a todos los bits del puerto 0, arrastraremos y soltaremos hasta cuatro veces *FIO0PIN* a la ventana del analizador lógico y a continuación seleccionaremos un bit diferente en cada variable *FIO0PIN*. La figura 15 ilustra este proceso para seleccionar el bit de menor peso (b0) de la primera variable *FIO0PIN*, configurando el valor 0x00000001 en el campo *And Mask* y estableciendo un rango de amplitud *Min/Max* de 0x0-0x1. Siguiendo un proceso similar, configuramos el resto de variables. Así, para seleccionar el bit b3 en la *FIO0PIN* inferior escribimos 0x00000008 en su campo *And Mask*.

NOTA: otra forma de especificar directamente un pin de un puerto en el analizador lógico, por ejemplo el pin 18 del puerto 1, sería la siguiente: hacer click en  de la ventana del analizador lógico (figura 15) e introducir el texto FIO1PIN.18, o bien PORT1.18 (esta última sintaxis sólo es válida usando el simulador, pero no lo reconocería el emulador JLINK (unknown pin)).

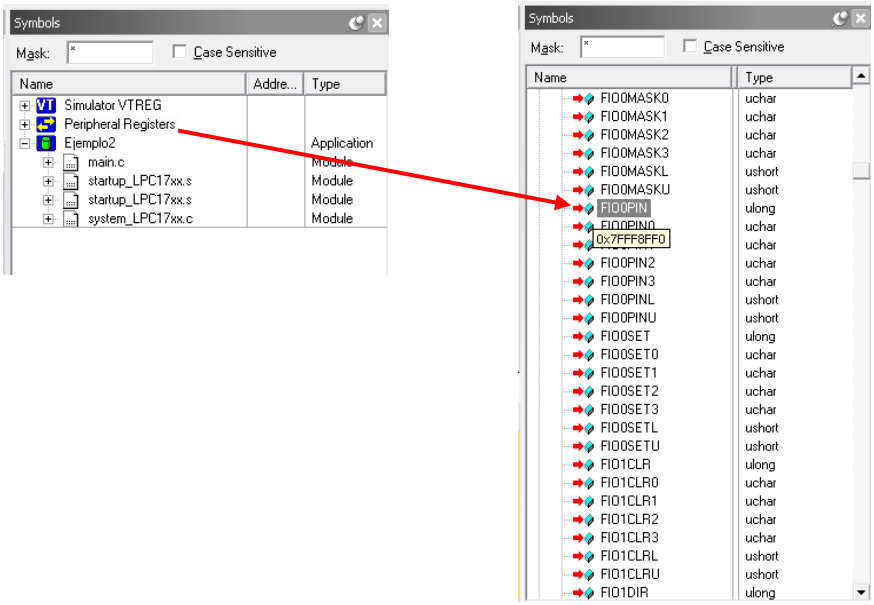


Figura 14. Localización del puerto P0 en la ventana Symbols.

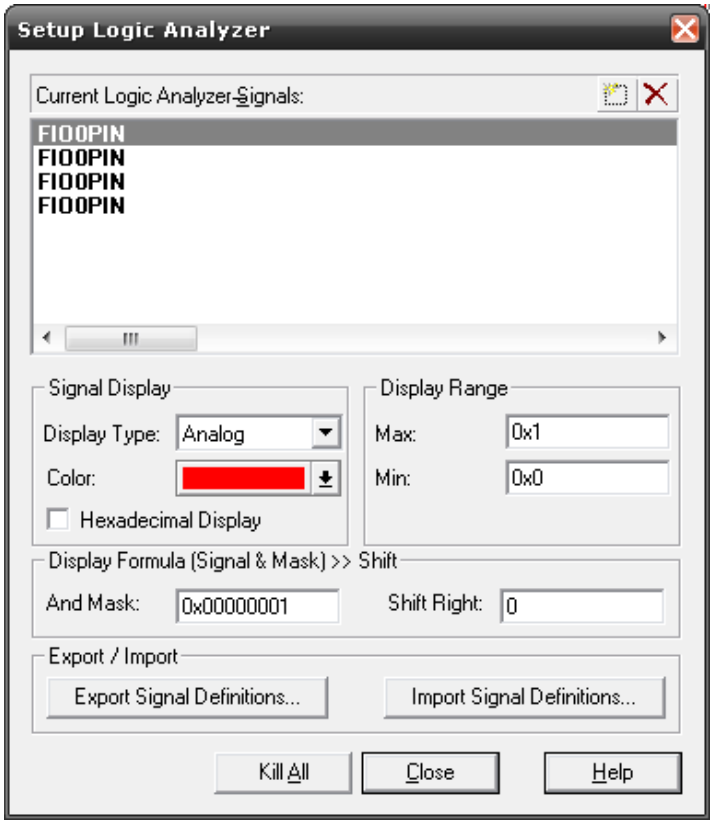


Figura 15. Selección del bit b0 del puerto P0 y configuración de su escala.

5.3 Modificación interactiva de variables

Con objeto de poder interactuar con el programa, para modificar la variable *sentido* en tiempo de ejecución y que cambie el sentido de desplazamiento en la activación de los pines P0.[0..3], crearemos una ventana compuesta de dos botones: '*right*' y '*left*', que al hacer click sobre ellos asignarán los valores '*sentido=0*' y '*sentido=1*', respectivamente. Para ello, comenzamos visualizando la ventana Command haciendo click sobre el icono , si no estaba ya visualizada en el interface gráfico del simulador.

Escribir a continuación del símbolo de prompt (>) de la ventana Comand el comando *define button "right","sentido=0"* y pulsar enter (figura 16). A continuación teclear *define button "left","sentido=1"* y pulsar enter. El resultado es la ventana con dos botones que aparece en la figura 17. Tener presente que esto sólo tiene sentido en el simulador. En una aplicación corriendo sobre una placa con un microcontrolador real se podría controlar el sentido de activación, leyendo el valor que se introduzca por otro pin de un puerto determinado.

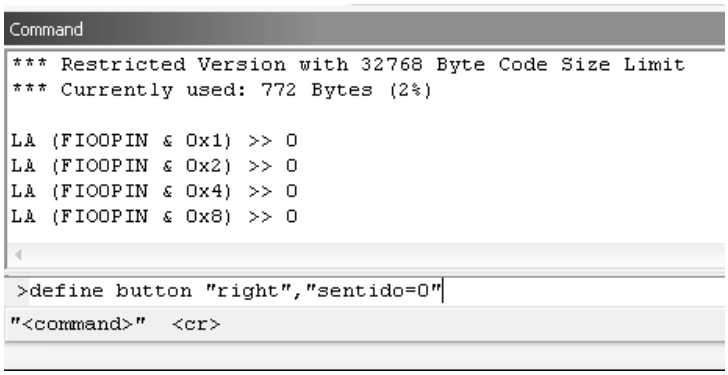


Figura 16. Creación de botones mediante la ventana de comandos.

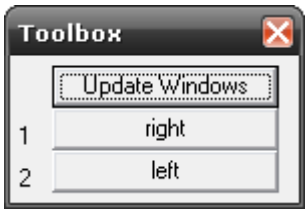


Figura 17. Ventana con dos botones para interactuar con el programa.

Ejecutar el programa paso a paso y ver cómo evolucionan las señales en el analizador lógico. Experimentar pulsando los botones *right* y *left* y ver cómo afectan a las señales en el analizador. Reflexionar sobre los cambios que se producen. En la figura 18 se visualiza una porción de las señales cuando la variable *sentido* vale cero.

Figura 18. Señales P0.[0..3] cuando la variable *sentido* vale cero.

Otra posibilidad de visualizar las señales del puerto P0 es mostrando su ventana a través del menú *Peripherals→GPIO Fast Interface→Port 0*. La ventana que aparece es la de

la figura 19. Ejecutar el programa paso a paso y observar como se activan progresivamente los pines P0.[0..3].

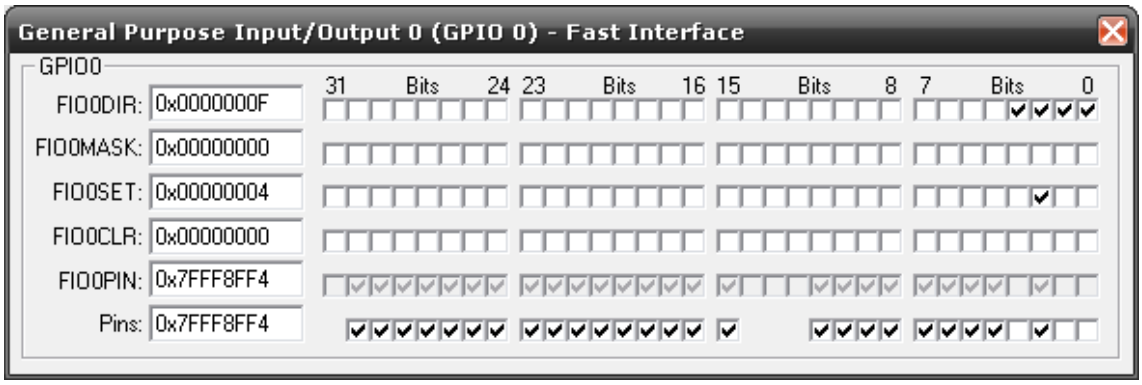


Figura 19. Ventana de visualización del puerto P0.

6 Utilización de Breakpoints

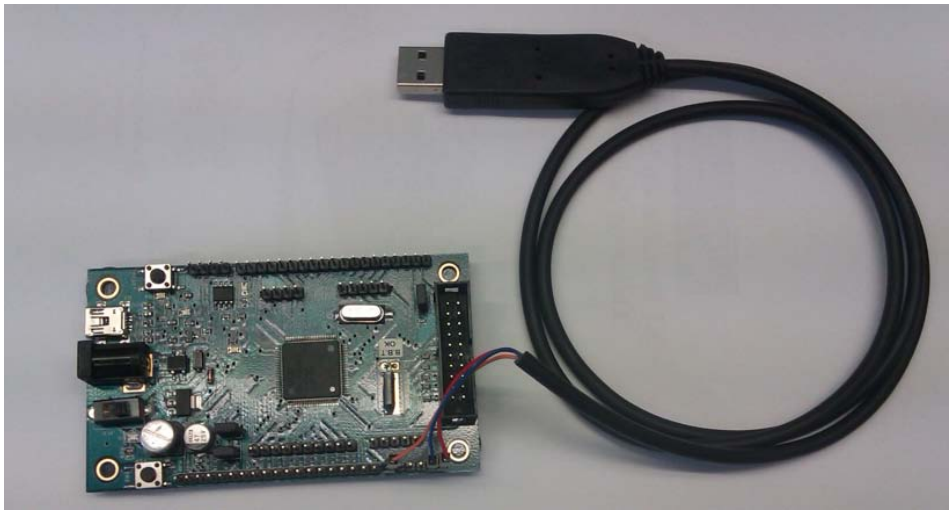
Los breakpoints o puntos de ruptura o parada son marcas que podemos colocar delante de las líneas de código para ayudarnos a depurarlo durante la simulación. Evitan, por ejemplo, tener que ejecutar paso a paso una porción de código que lleva mucho tiempo y sabemos a priori que funciona correctamente. De este modo, nos permiten ejecutar el programa y que éste quede detenido cuando se encuentra una línea que hemos marcado en el simulador con un breakpoint. A partir de ahí, podemos seguir ejecutando paso a paso para depurar cuidadosamente el código que sigue o seguir ejecutando el programa hasta que se encuentre el siguiente breakpoint, etc. Para insertar un breakpiont, situarse en una línea de código y hacer click en **Insert/Remove Breakpoint (F9)**.

Existen muchas otras posibilidades de utilizar breakpionts a través de la ventana de comandos, por ejemplo, condicionalmente. Se recomienda consultar el comando *BreakSet* en la ayuda del entorno µVision, pestaña 'índice'. Ω

7 Carga del programa a la tarjeta mediante el modo ISP

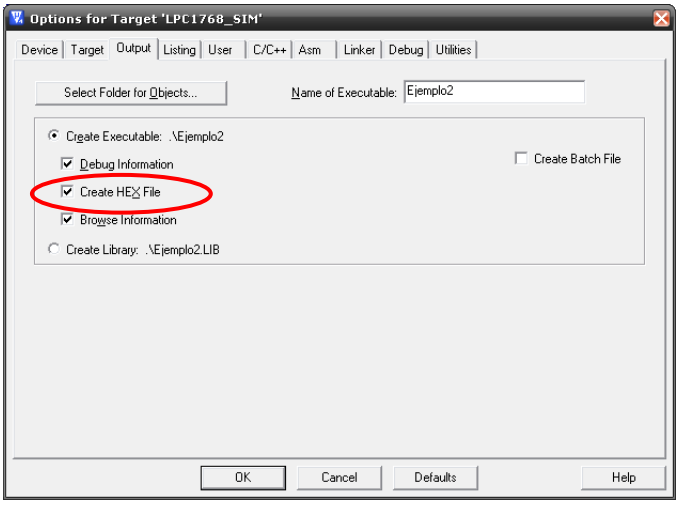
Una de las formas de cargar el programa a la tarjeta de desarrollo *Blueboard* es mediante comunicación serie asíncrona a través de la UART0 (modo ISP) y el programa FlashMagic de NXP, que ha de estar instalado en el PC. Para ello, los pasos que hay que seguir son:

1. Conectar el cable USB-serie (GND, TX, RX) a las líneas de la UART0 (**TxD0**, **RxD0**) y a la masa (**GND**) del **conector J3** de la tarjeta, tal como muestra la foto.

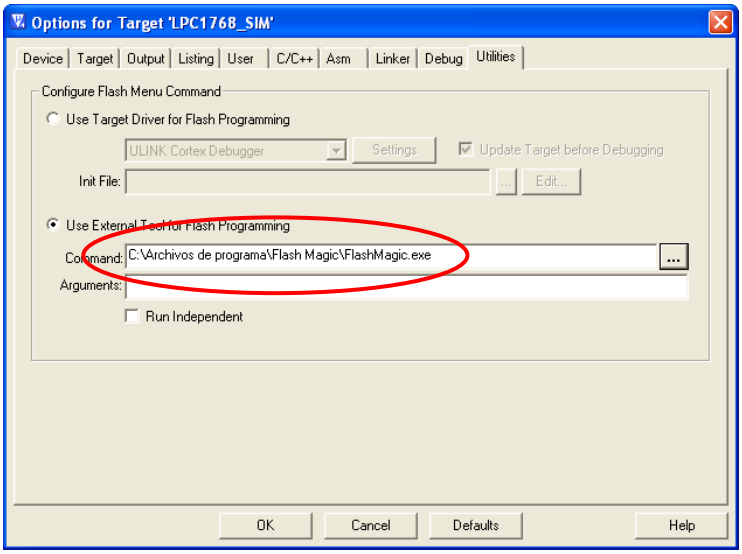


2. **Entrar en modo ISP**. Se lleva a cabo manteniendo pulsado un breve instante de tiempo el pulsador **SW2** (mantiene P2.10 a masa), tras hacer un reset mediante el pulsador SW1. **NOTA:** Al no estar soldada de fábrica la resistencia R27 (0 ohmios) tenemos que hacer un puente con un cable (hembra-hembra) entre el pin P2.10 y masa (GND) de la tarjeta.

3. No olvidar Compilar el proyecto con la opción “create HEX file”, en la pestaña **Output** del menú de opciones.

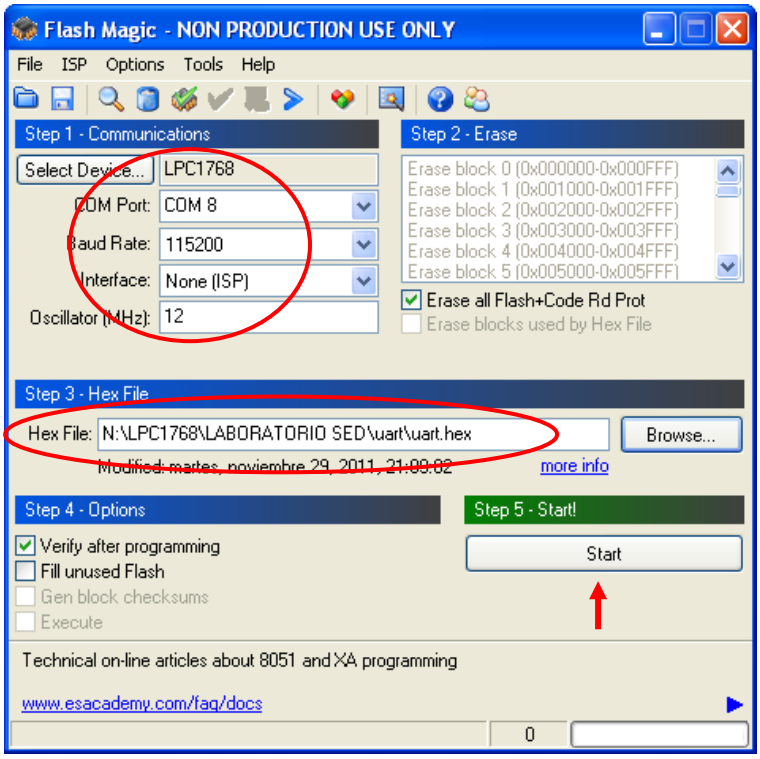


4. Pinchar desde Keil el icono **LOAD**. Previamente es necesario configurar en la pestaña **Utilities**, el path de acceso al ejecutable (.exe) FlashMagic donde se instaló.



5. Se abrirá el programa FlashMagic. Configurar el puerto COMx que se haya detectado, y seleccionar el Baud Rate a 115200 baudios. Comprobar el path del archivo .HEX de carga.

6. Pinchar en **Start** para que se cargue el programa en la memoria flash del LPC1768. Observar la barra inferior derecha de la ventana que muestra el proceso de la transferencia.



7. Para ejecutar el programa basta con pulsar el botón de reset de la tarjeta (SW1).

8. **IMPORTANTE: No olvidar cerrar Flash Magic, para tener de nuevo el control sobre el entorno Keil, y poder lanzar la ejecución del programa que acabamos de cargar en la Blueboard desde el PC.**