

## Anexo IV. Programación en C con Keil $\mu$ Vision®5

### IV.1. Introducción

Este anexo presenta una introducción al manejo del entorno de desarrollo Keil  $\mu$ Vision®5, una herramienta software para el desarrollo de proyectos en lenguaje C basados en microcontroladores. Entre otras cosas, permite compilar, simular, depurar y cargar el código en un chip microcontrolador (LPC1768, en este caso).

A continuación, se hará una introducción a este entorno de desarrollo a través de un ejemplo sencillo que ilustra la creación de un proyecto, su simulación y depuración para visualizar variables, así como el empleo de los *breakpoints*.

### IV.2. Creación de un proyecto en C

La Figura IV.1 esquematiza de una manera muy general los pasos que se deben seguir en el diseño de un programa para desarrollar una determinada aplicación con un microcontrolador.

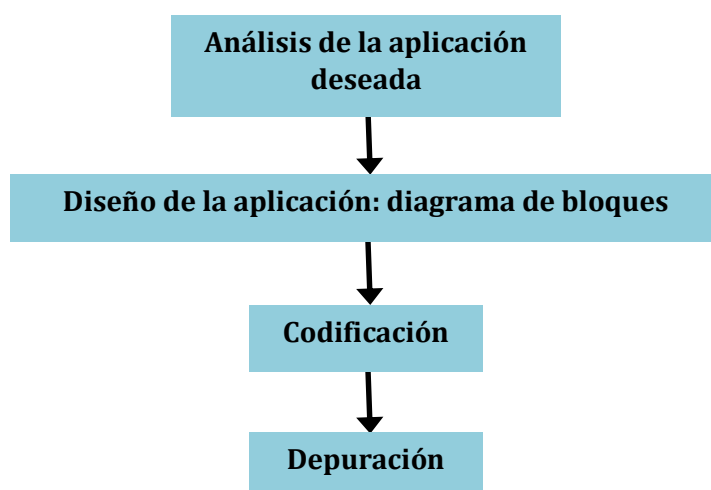


Figura IV.1. Pasos en el diseño de un programa para microcontrolador.

A continuación, se ilustra cómo crear un proyecto en Keil  $\mu$ Vision®5 con el programa fuente que previamente se ha escrito con un editor de texto o desde el propio editor de Keil  $\mu$ Vision®5, luego se compilará este programa y se simulará para poderlo depurar y probar su funcionamiento.

Se comienza creando un nuevo proyecto desde las opciones de menú *Project -> New  $\mu$ Vision Project* (Figura IV.2).

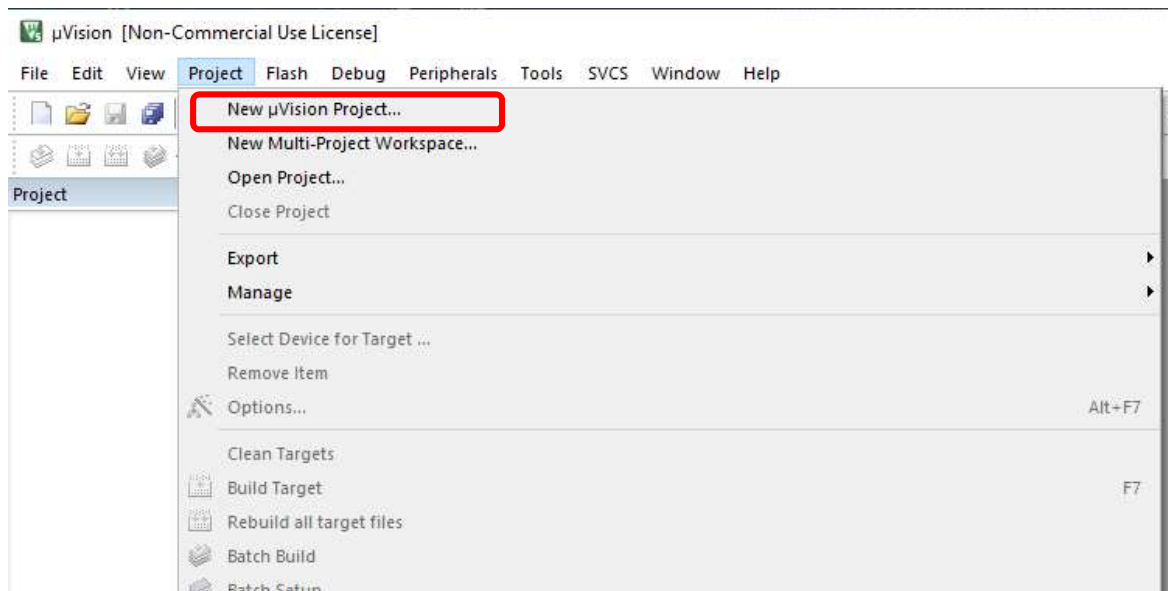


Figura IV.2. Creación de un nuevo proyecto.

En la ventana que se abre (Figura IV.3) se debe indicar el nombre del proyecto. Es recomendable crear una carpeta para almacenar todos los ficheros del proyecto. Este paso se puede llevar a cabo desde esta misma ventana.

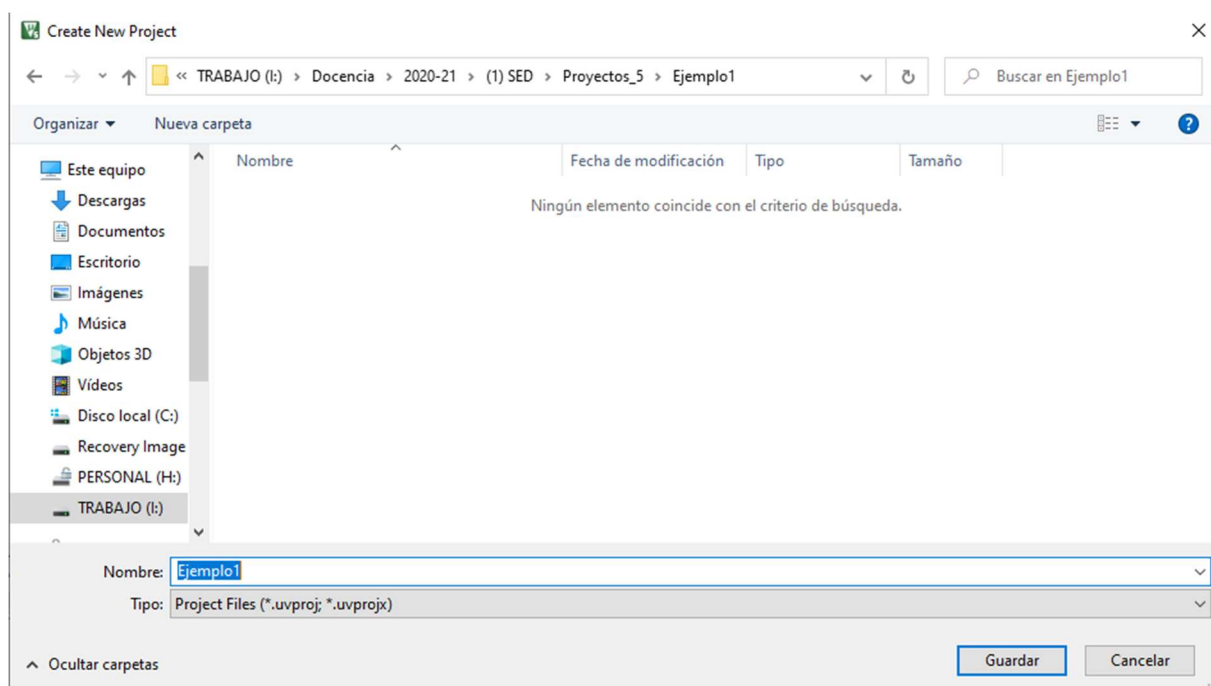


Figura IV.3. Elección de la carpeta del proyecto y nombre

Al hacer clic sobre *Guardar* se muestra la ventana que permite elegir el dispositivo que se va a programar (Figura IV.4). En el laboratorio será el LPC1768, del fabricante NXP. Desplegando el árbol que aparece en la parte izquierda de la ventana se selecciona el dispositivo LPC1768 y se hace clic en *OK*.

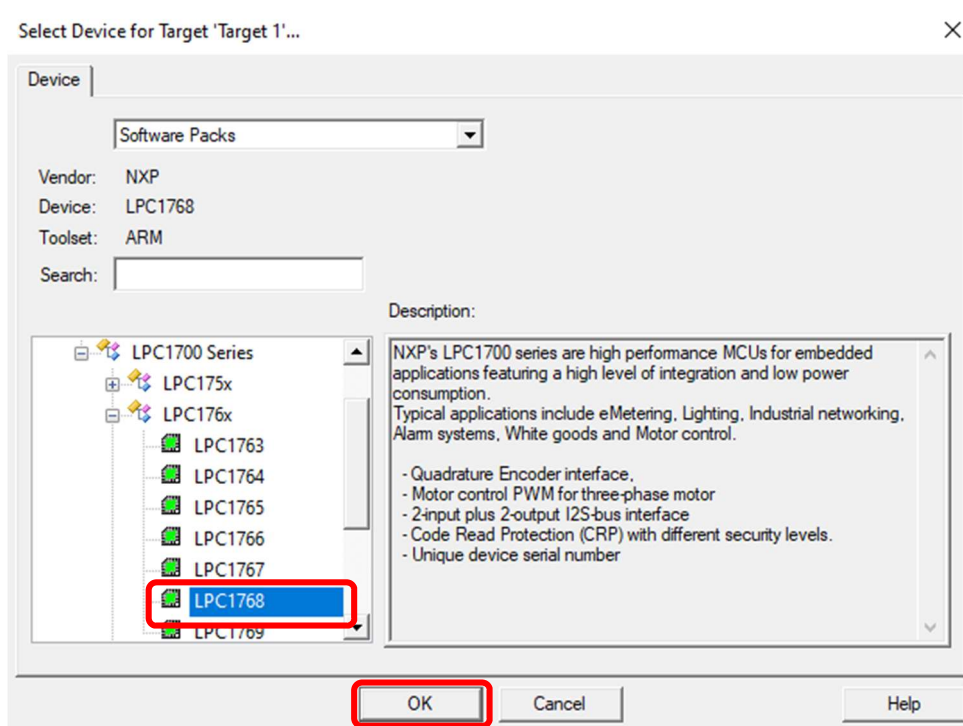


Figura IV.4. Selección del dispositivo que se desea programar.

Se muestra una siguiente ventana en la que se pueden elegir qué librerías se quieren incorporar al proyecto (Figura IV.5). Se debe desplegar la librería *CMSIS* y seleccionar únicamente *CORE*, y de la misma manera se despliega *Device* y se selecciona *Startup*. Se hace clic en el botón *OK*.

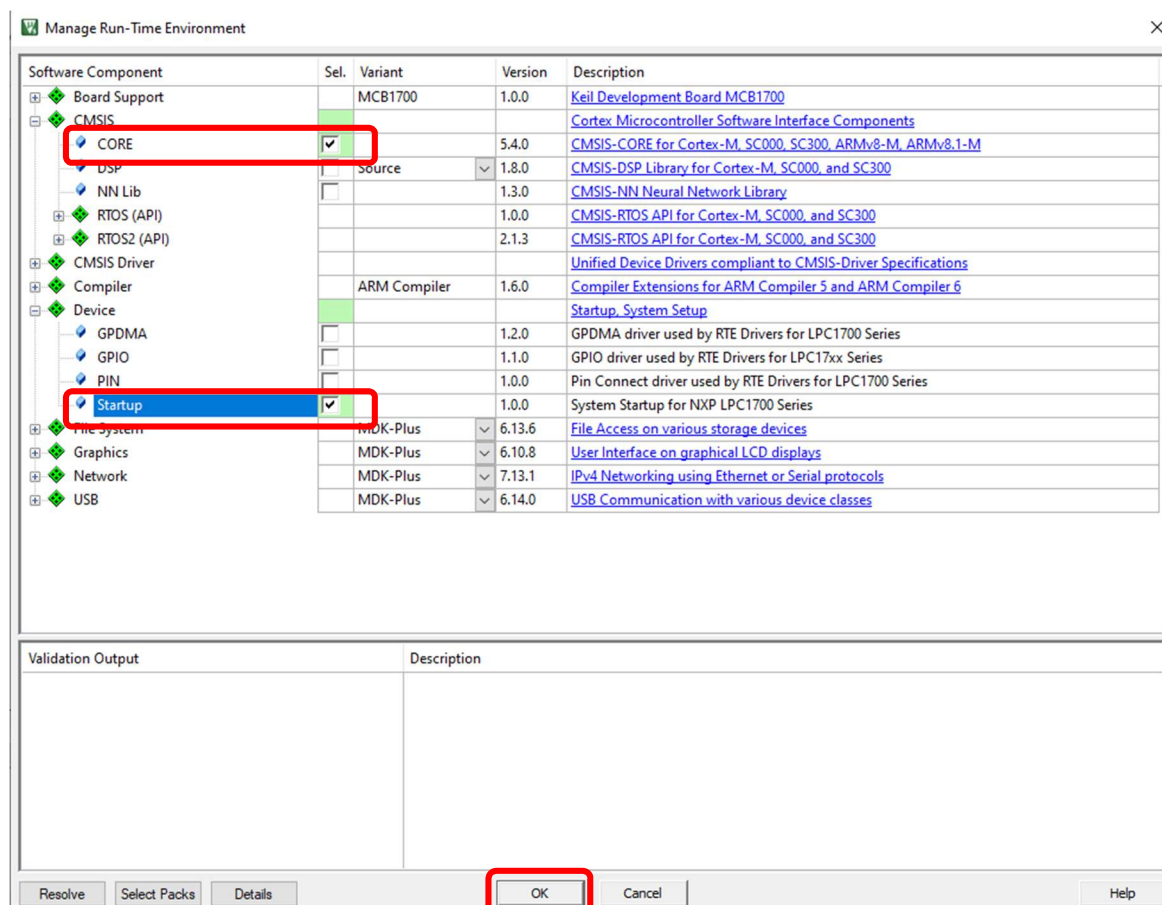


Figura IV.5. Selección de las librerías a incorporar al proyecto.

Una vez creado el proyecto en la ventana *Project* se puede ver un árbol de ficheros en la parte izquierda, como se muestra en la Figura IV.6.

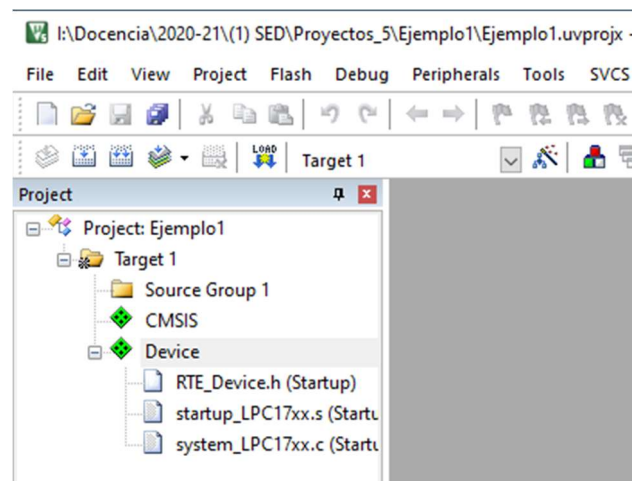


Figura IV.6. Aspecto del árbol del proyecto inicial.

Los ficheros de *Startup* que se han añadido (*startup\_LPC17xx.s* y *startup\_LPC17xx.c*) es similar a un sistema operativo, donde se realiza una configuración inicial del microcontrolador, que permite al usuario trabajar con un nivel mayor de abstracción. Una vez realizada esta configuración se llama a la función `main()`, que es la función principal del programa en C.

#### IV.2.1. Configuración del entorno

A continuación, se selecciona el compilador adecuado en la ventana de Configuración del Entorno. A esta ventana se accede desde el menú de *Project -> Manage -> Projects items....* En la ventana que se abre se selecciona la solapa *Folders/Extensions* (Figura IV.7).

Existe la opción de elegir entre dos compiladores posibles:

1. **Compilador ARM:** Es el compilador de ARM; en este caso se va a trabajar con una versión libre y gratuita de evaluación restringida a 32Kbytes de código. Para códigos más amplios es necesario comprar la licencia. Esta es la opción que se selecciona en este ejemplo.
2. **Compilador de GNU:** Versión libre.

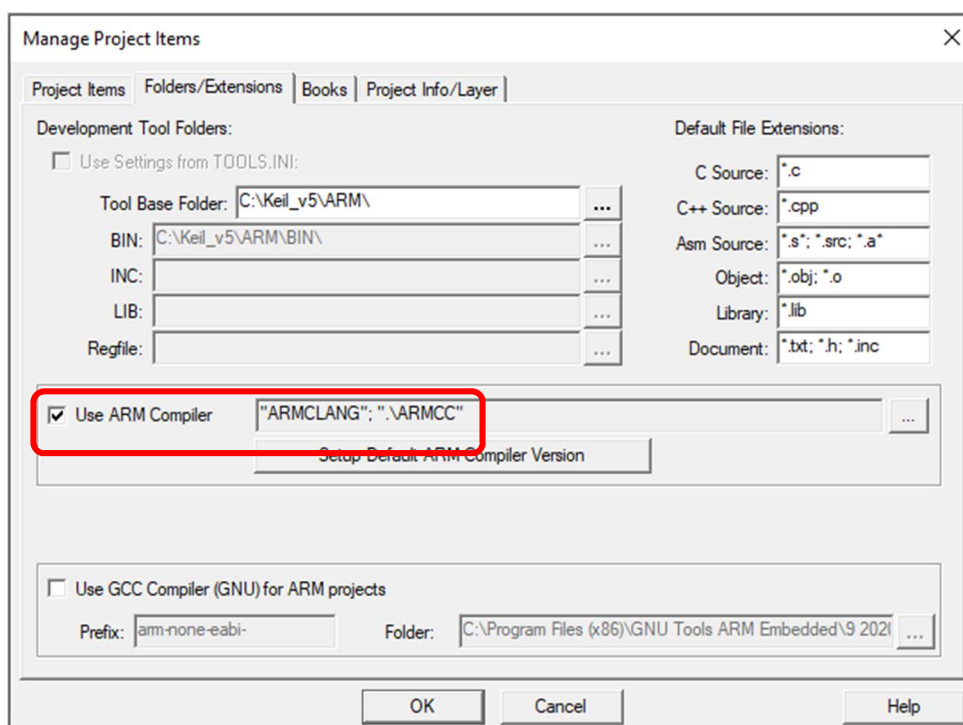


Figura IV.7. Ventana de configuración del compilador.

### IV.2.2. Configuración de opciones

Se puede renombrar la entrada *Target 1* como LPC1768 (haciendo clic sobre ella, tras estar previamente seleccionada). En este tutorial se cambiará el nombre a *LPC1768*.

A continuación, es necesario modificar algunas opciones del proyecto. Para ello se hace clic con el botón derecho del ratón sobre el texto *LPC1768* para que aparezca su menú contextual y se selecciona *Options for Target 'LPC1768'...* (Figura IV.4).

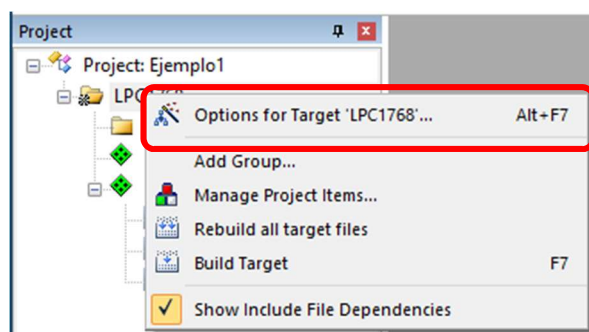


Figura IV.8. Menú contextual de LPC1768.

Una vez aparezca la ventana *Options for Target 'LPC1768'* se debe configurar, en primer lugar, la versión del compilador en la pestaña *Target*. Se debe seleccionar como *ARM Compiler* la opción *Use default compiler version 5* (Figura IV.9).

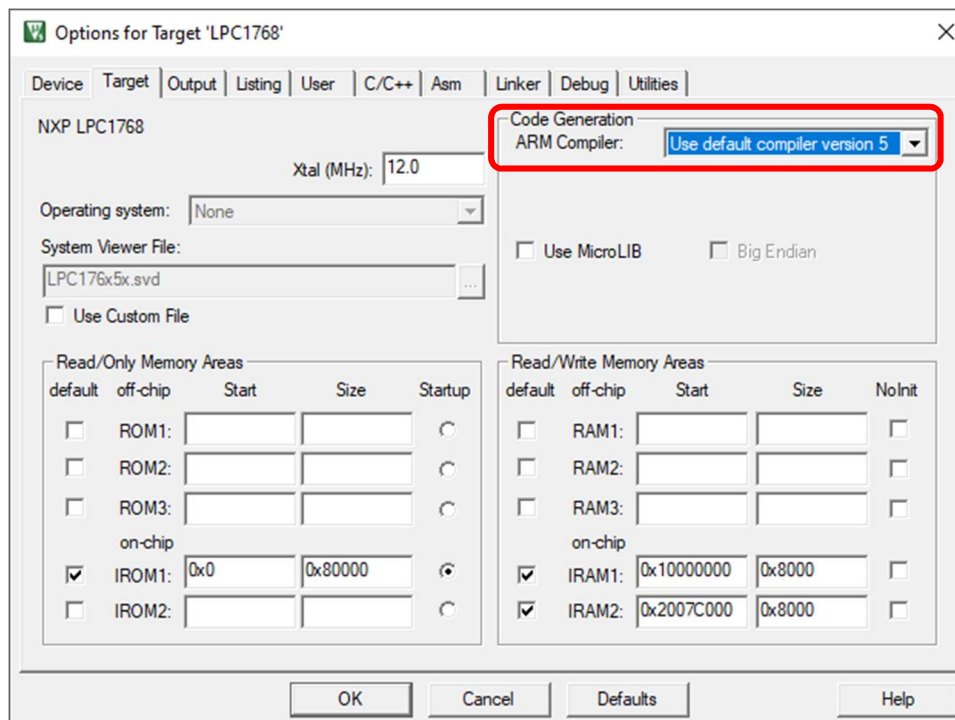


Figura IV.9. Opciones para la pestaña *Target*.

En la solapa *Debug* también es necesario realizar cambios. Se debe seleccionar que se use el simulador para probar el funcionamiento del programa y, además, se deben cambiar las librerías *Dialog DLL* que vienen por defecto para dejarlo como muestra la Figura IV.10.

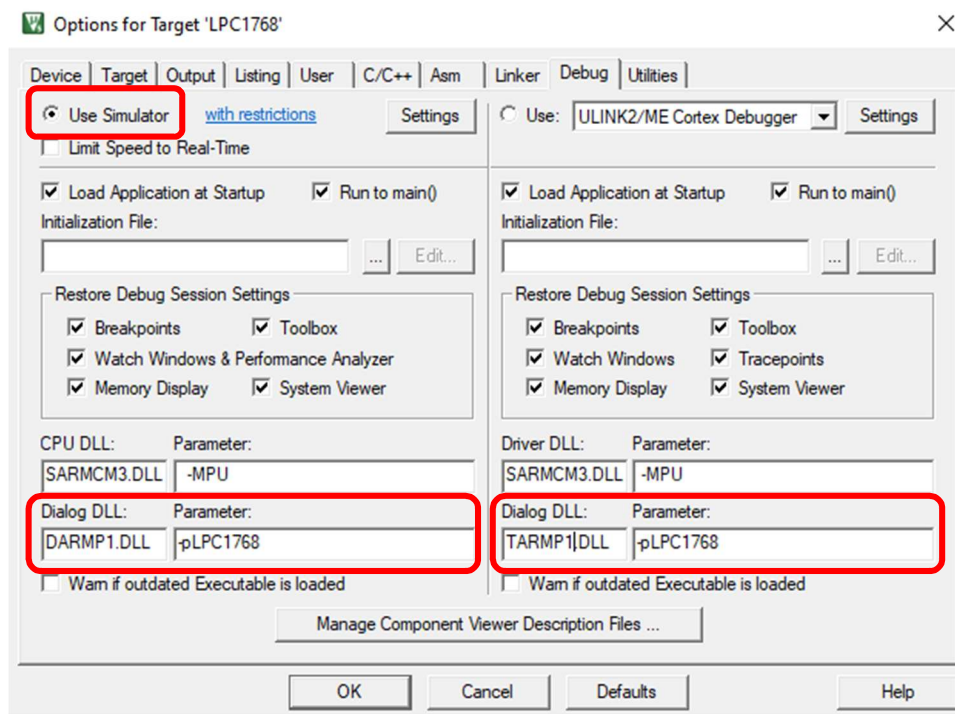


Figura IV.10. Opciones para la pestaña *Debug*.

Finalmente, en la solapa *Linker* es necesario activar una casilla para configurar el sistema de memoria, como se muestra en la Figura IV.11.

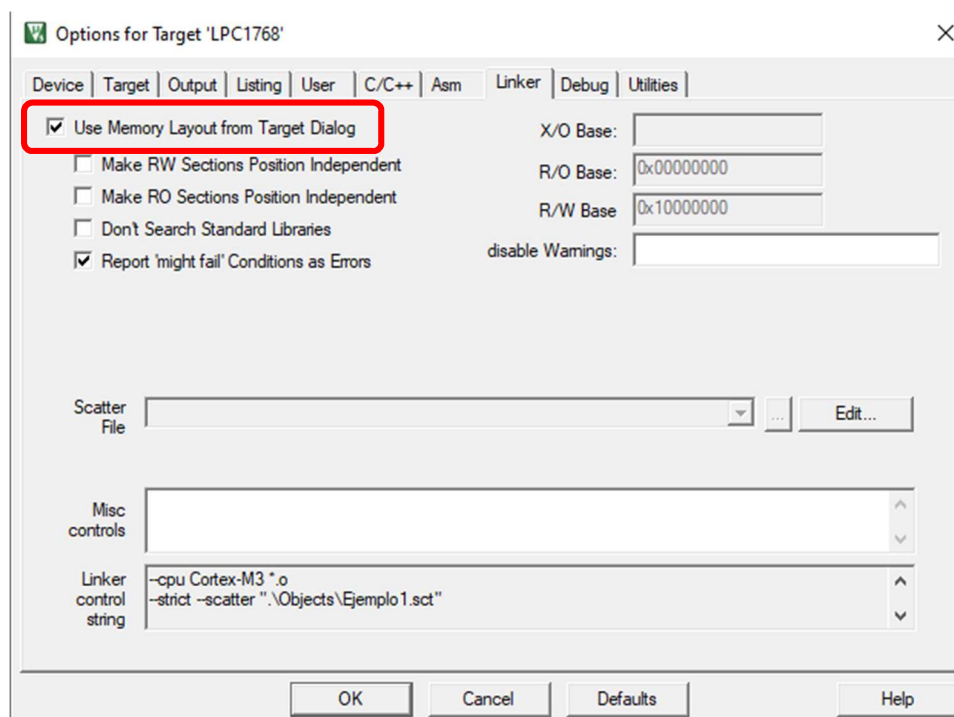


Figura IV.11. Opciones para la pestaña *Linker*.

### IV.2.3. Agregar ficheros al proyecto

Únicamente falta añadir el fichero o ficheros fuente que tienen el código de la aplicación. En este anexo se va a incluir el código mostrado en el Programa IV.1. Este código se debe escribir con un editor de texto (bloc de notas o entorno Keil) y se guarda en la carpeta del proyecto en un fichero con el nombre *Ejemplo1.c*.

```
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;
    }
}
```

Programa IV.1. Código ejemplo.



Una vez creado el fichero se añade al proyecto. Esta operación se puede hacer haciendo clic con el botón derecho en la carpeta *Source Group 1* de la ventana *Project* y seleccionando la opción *Add Existing Files to Group 'Source Group 1'...* (Figura IV.12).

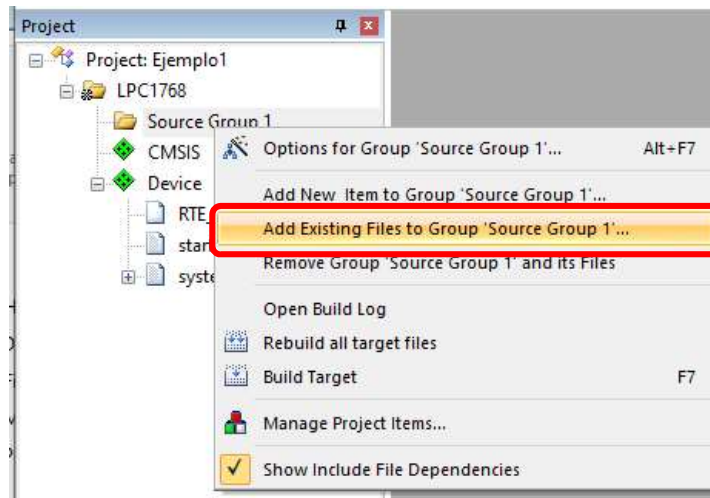


Figura IV.12. Añadir fichero fuente desde la ventana *Project*.

Se selecciona el fichero que se ha creado previamente y se hace clic en el botón *Add*, y luego en el botón *Close*. (Figura IV.13).

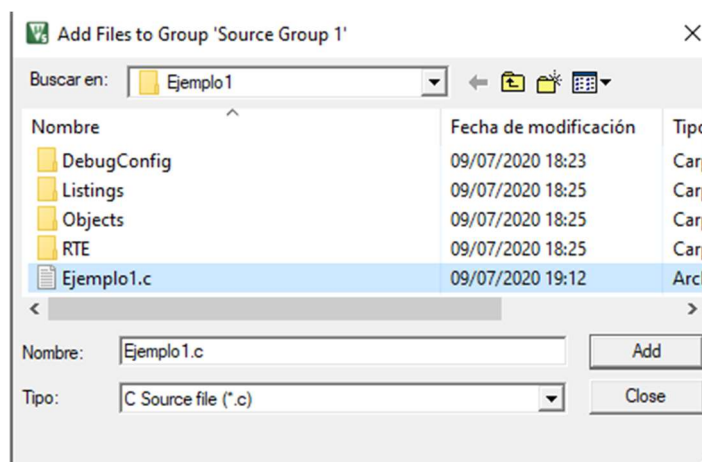




Figura IV.13. Selección del fichero fuente.

En la Figura IV.14 se muestra la estructura completa del proyecto ejemplo y el resultado de su compilación y enlazado mediante el comando Build (F7) o haciendo clic sobre cualquiera de los iconos  . Obsérvese la ventana inferior *Build Output* que indica si se ha producido algún error durante la compilación y también informa sobre el tamaño que ocupa el programa.



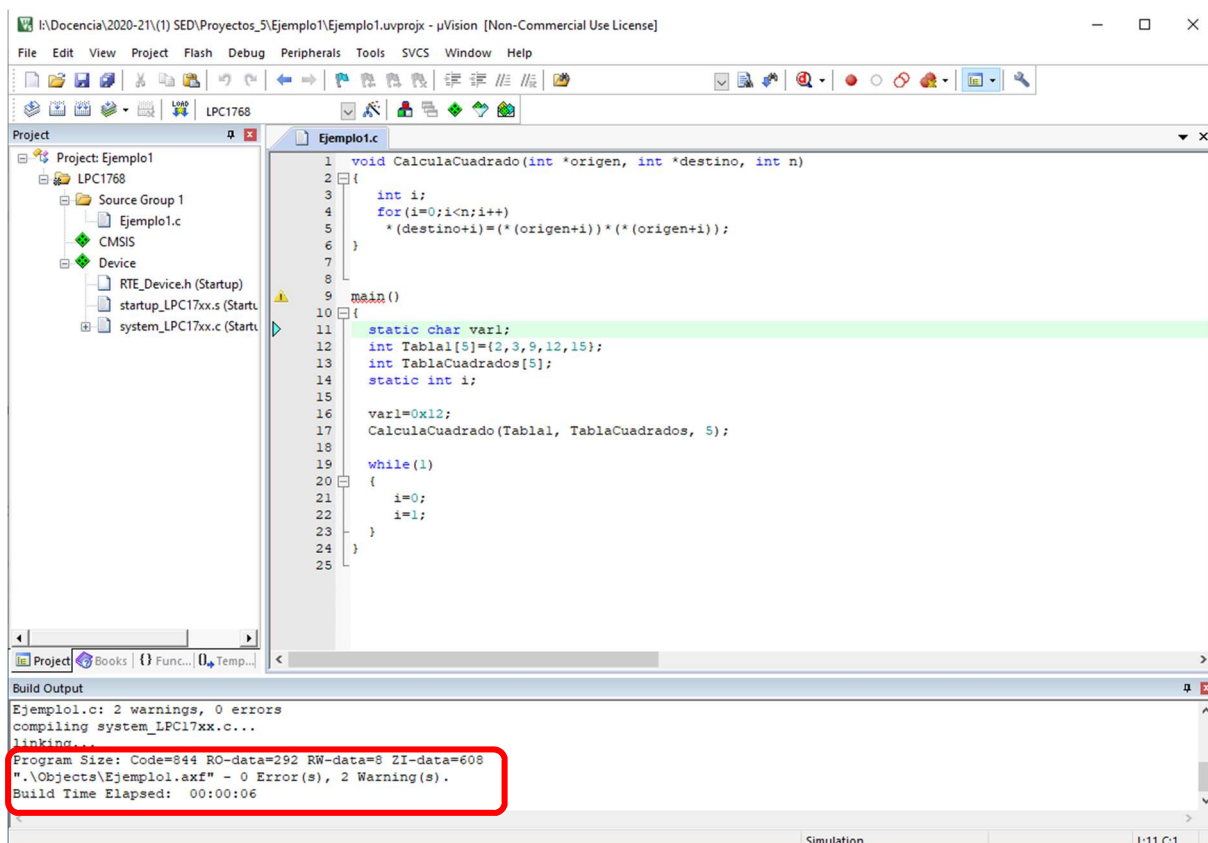









Figura IV.14. Compilado y “linkado” del proyecto (ZI-data: Zero Initialized Data, RO-data son constantes. Tamaño total de RAM = RW data + ZI data. Tamaño total de ROM = Code + RO data).

#### IV.2.4. Simulación

Una vez que se ha compilado el programa se tiene la posibilidad de ejecutar el programa en el simulador que incluye Keil  $\mu$ Vision®5 o de cargarlo en una placa de desarrollo externa para que se ejecute físicamente en el microcontrolador. De momento, se hará uso del simulador para ver algunas de sus interesantes posibilidades. Para ello, se arranca el simulador haciendo clic sobre el botón  (F5) y aparecerá la ventana de la Figura IV.15. Estos son algunos comandos que resultarán de utilidad:

- Se ejecuta el programa paso a paso haciendo un clic sobre  (F11) por cada instrucción.
- Si se desea que se ejecute con un simple clic toda una función, sin necesidad de ejecutar paso a paso cada una de las instrucciones que la componen, se debe hacer clic sobre  (F10).
- Para ejecutar todo el código hasta la línea donde hemos colocado el cursor, clic sobre el icono  (Ctrl.+F10).
- Para hacer reset y llevar la ejecución al comienzo del código se debe pulsar en el icono .
- Si lo que se desea 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 IV.15:

- 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 del programa.

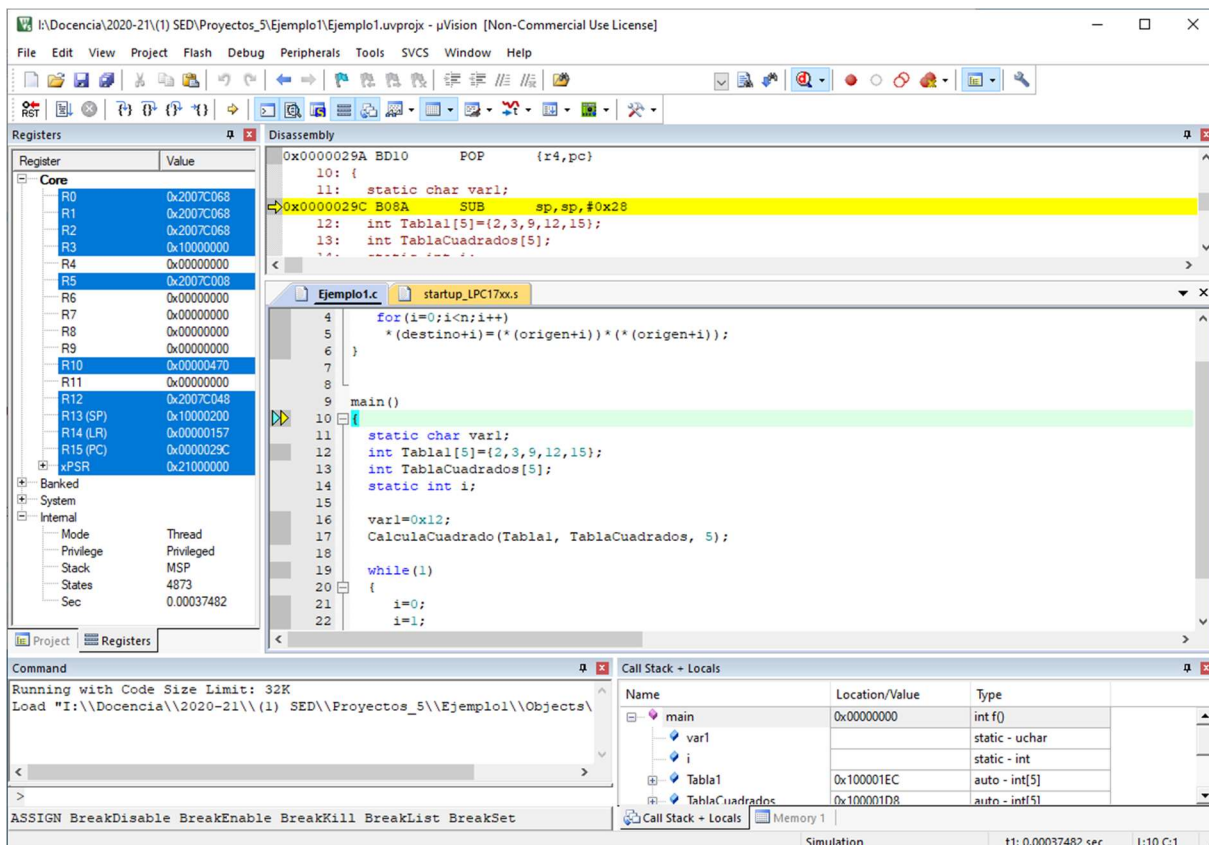


Figura IV.15. Ventana inicial del simulador.

En la ventana *Locals* también se puede comprobar que la variable *Tabla1* se ha almacenado a partir de la dirección de memoria 0x100001EC. Para mostrar el contenido de esta zona de memoria se hace clic sobre la pestaña *Memory 1* y se introduce en el campo *Address* el valor 0x100001EC. Se coloca el cursor delante de la línea `CalculaCuadrado(Tabla1, TablaCuadrados, 5)` y se ejecuta hasta la posición del cursor (Ctrl+F10). En este momento ya está inicializada la variable *Tabla1* y en la Figura IV.16 se puede ver el contenido de la zona de memoria donde se ha almacenado esta variable.

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 (dirección menor de las 4 afectadas). Se puede modificar a mano el valor de una posición de memoria haciendo doble clic 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).

Como ejercicio se recomienda, que observe dónde se almacena el array `TablaCuadrados[5]` y que compruebe cómo se actualiza mientras el programa se ejecuta paso a paso. Repita los mismos pasos 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` se puede hacer clic 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*.

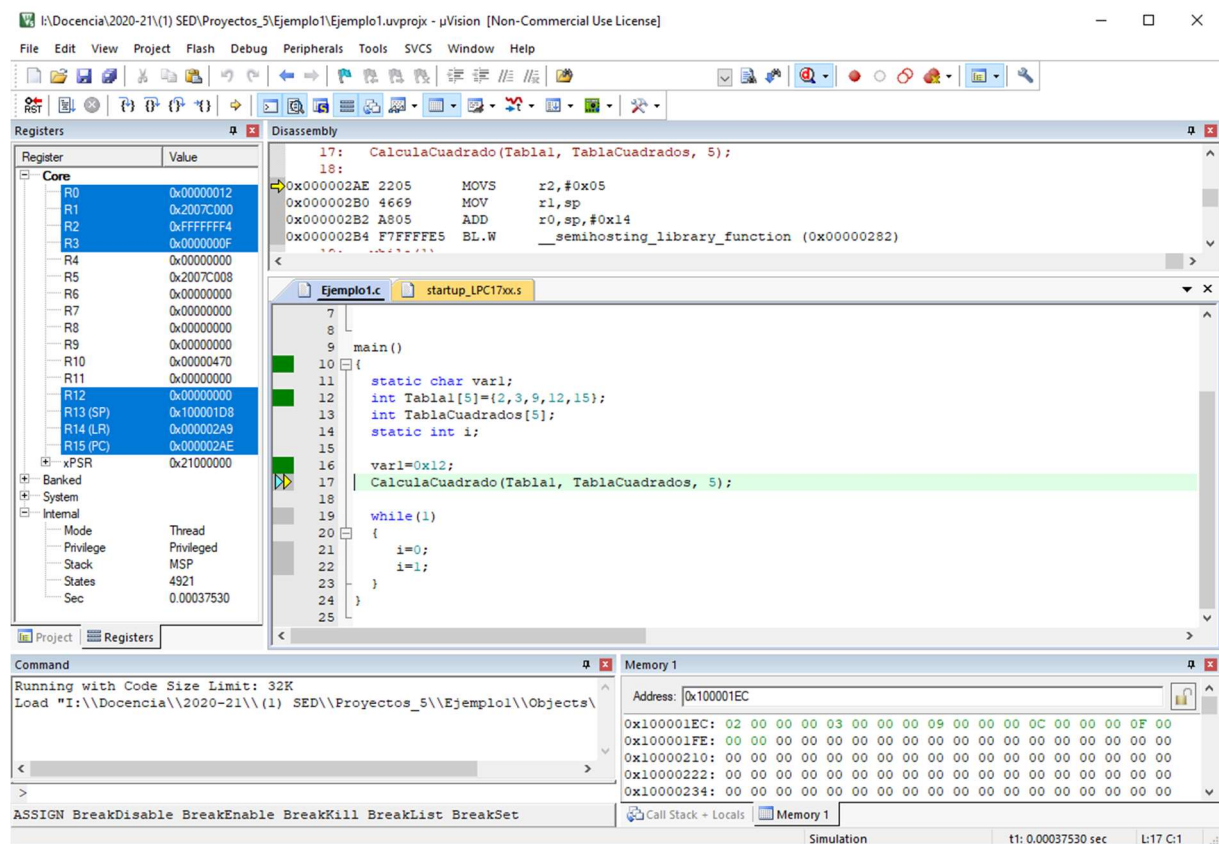


Figura IV.16. Contenido de la memoria donde se ha almacenado la variable `Tabla1`.