PRONTUARIO DEL ENTORNO DE DESARROLLO KEIL µVision® PARA LA FAMILIA DE MICROCONTROLADORES MCS51

ÁREA DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES
DEPTO. DE ARQUITECTURA DE COMPUTADORES,
ELECTRÓNICA Y TECNOLOGÍA ELECTRÓNICA
(UNIVERSIDAD DE CÓRDOBA)

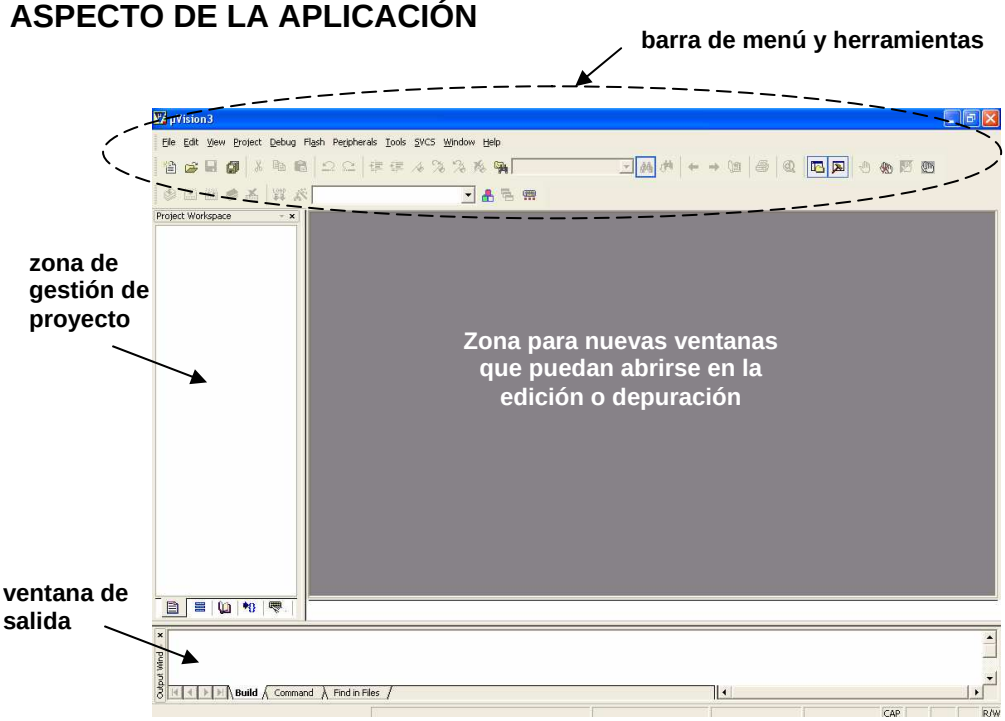
AUTOR:
Prof. Antonio Moreno Fernández-Caparrós
Depto. de Arquitectura de computadores, electrónica y tecnología electrónica
Área de Arquitectura y Tecnología de Computadores
Universidad de Córdoba

Córdoba, marzo de 2009
España

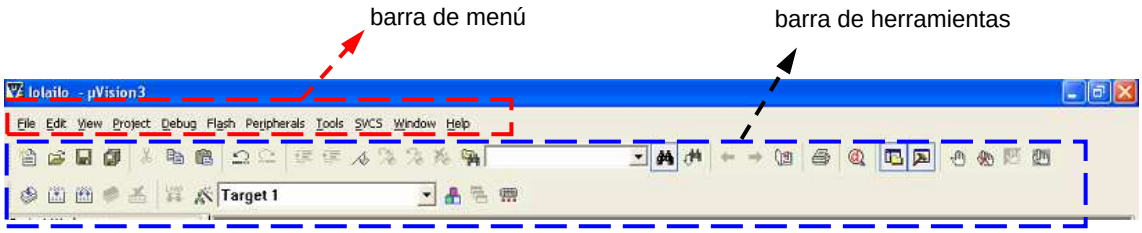
ÍNDICE

PASOS PARA TRABAJAR CON UN PROGRAMA EN KEIL µVision 3

ASPECTO DE LA APLICACIÓN



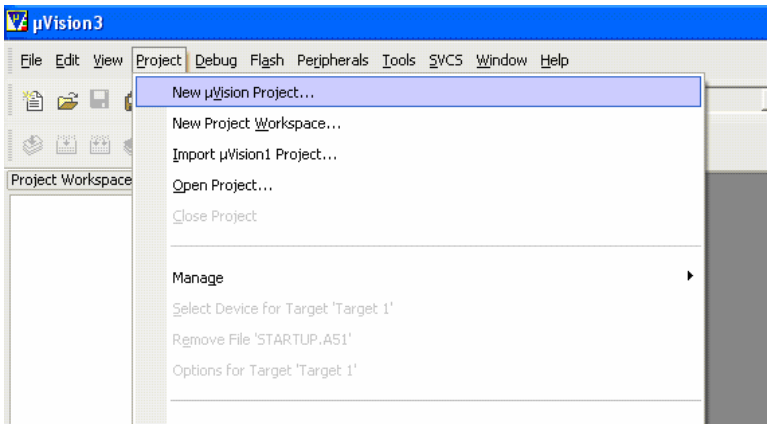
En la parte superior se encuentra la **barra de menú**, mediante la cual se puede acceder a todas las opciones de Keil µVision 3, y debajo de ella se encuentra la **barra de herramientas**, con los iconos de las funciones más usuales. Estos iconos suponen un atajo alternativo para realizar tales tareas.



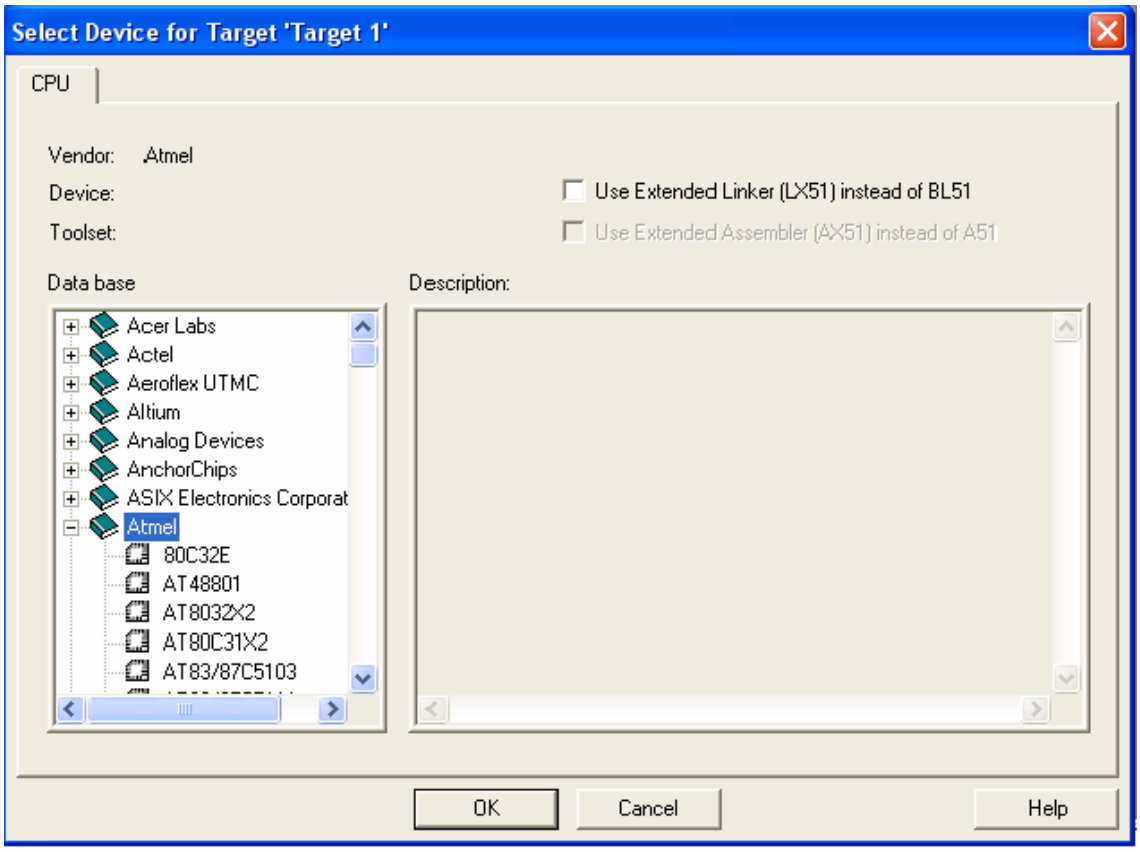
1º) CREACIÓN DE UN PROYECTO:

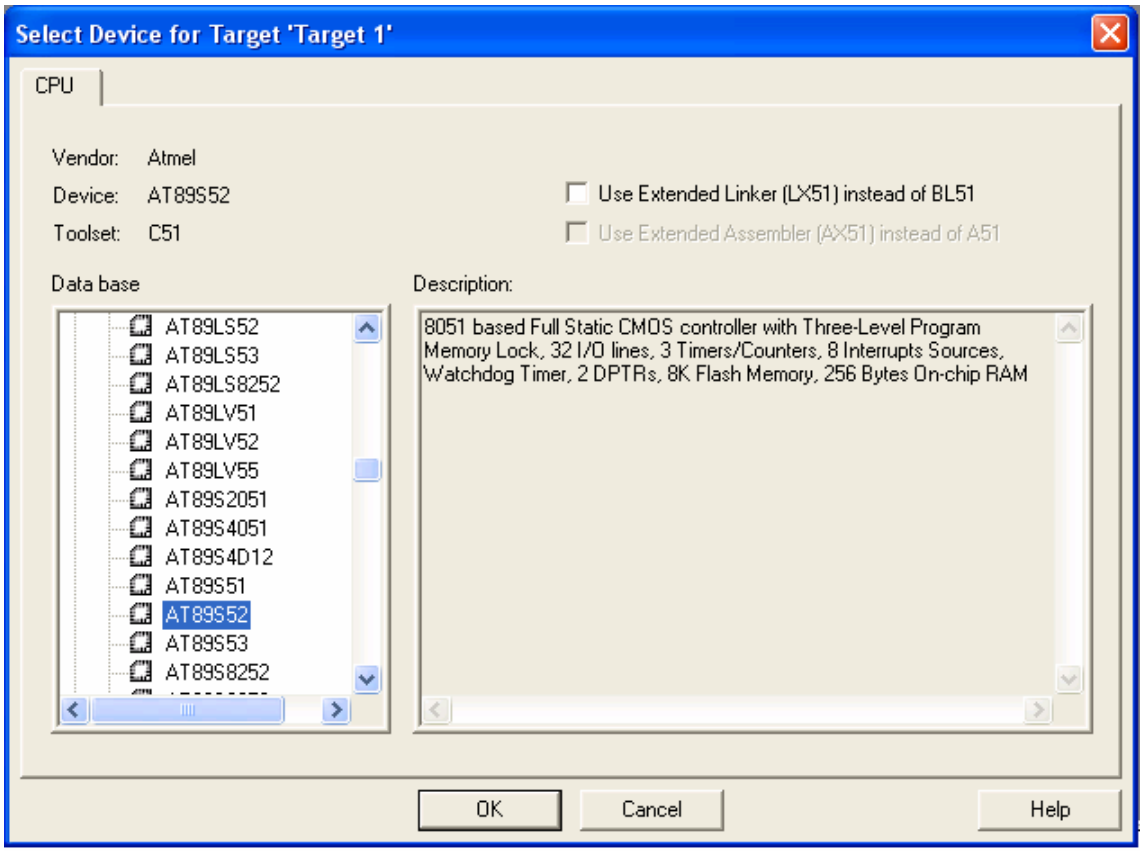
Lo primero, si no se ha hecho antes, es crear un proyecto. Para ello, hágase lo siguiente:

Project→New µVision Project

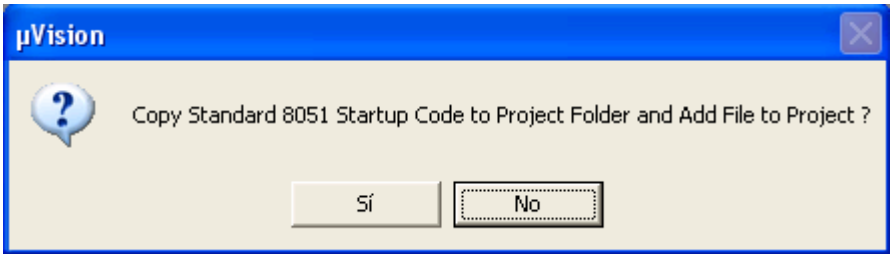


En la ventana que se abre, hay que dar nombre al proyecto y ubicarlo en la carpeta que se desee (si no existiese, créese). A continuación, se abrirá otra ventana en que hay que seleccionar el procesador que se vaya a utilizar: CPU de ATMEL, AT89S52 (o el que sea).



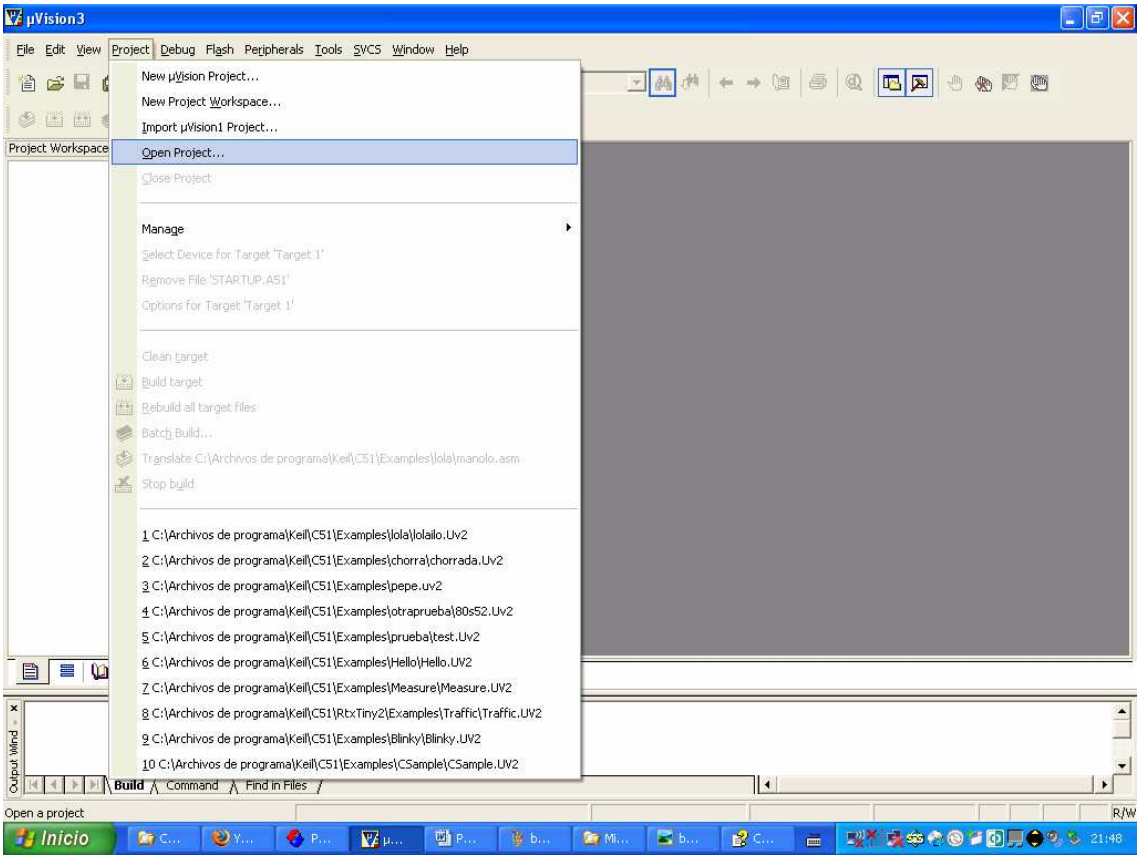


A continuación se pregunta si se desea copiar al proyecto un fichero con una plantilla para iniciar el código que se vaya a escribir. Responder ***Sí*** o ***No*** según se desee (inicialmente, ***NO***).



2º) APERTURA DE UN PROYECTO

Una vez que se haya creado un proyecto, cada vez que se vaya a trabajar se empezará abriendo (caso de no estarlo por defecto) el proyecto: ***Project→Open Project***

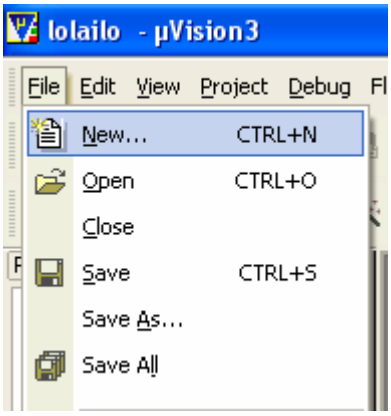


En la ventana que se abre, búsquese y selecciónese el proyecto. Inicialmente, y salvo que se haya dicho que sí a la pregunta de copiar al proyecto una plantilla de partida, el proyecto estará vacío. Un proyecto es una abstracción y representa el conjunto de ficheros de código fuente en que se haya estimado conveniente dividir la escritura de un programa. En nuestro caso, sólo se crearán proyectos con un único fichero fuente. Lo primero, si no se ha creado ninguno, será asociar al proyecto uno (también, en el transcurso de un desarrollo es posible ir añadiendo, si se estima conveniente, nuevos ficheros o módulos fuente).

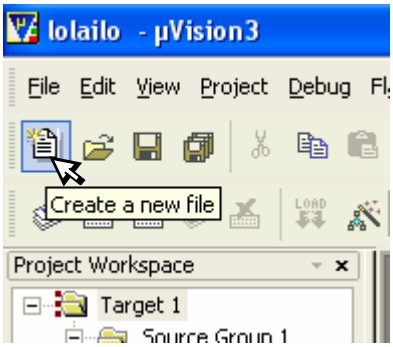
Para ello, lo primero es crear un fichero: ***File→New***

Se abrirá la ventana de edición y se podrá ya escribir el código. Debe guardarse el fichero abierto, haciendo ***File→Save as*** para dar nombre al fichero y guardarlo en donde se haya creado el proyecto. El nombre se le puede dar el que se desee, no teniendo por qué

coincidir con el del proyecto (cosa lógica si se piensa que un proyecto puede estar compuesto por varios módulos (ficheros) repartiendo el código fuente total entre ellos (en vez de meter todo el código en un gran y único fichero). Como extensión del fichero, poner ASM o A51 (o incluso S, SRC o A).

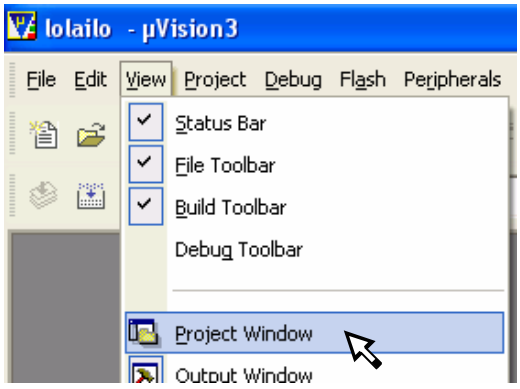


a) Creación de fichero vía barra de menú

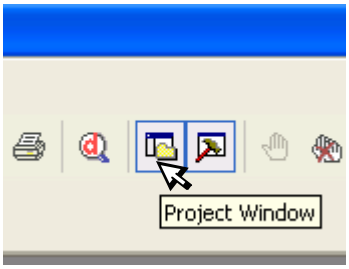


b) ídem vía barra de herramientas

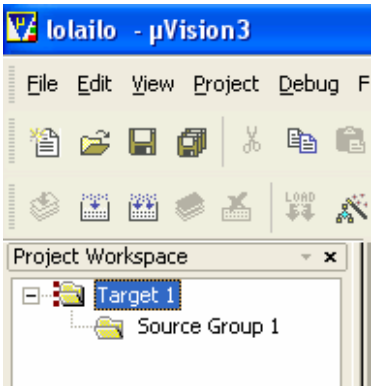
Es necesario agregar los ficheros fuente a un proyecto. Para ello, si no estuviese ya abierta, ábrase la ventana de proyecto en ***View→Project window***. Aparecerá la estructura que se le haya dado al proyecto. Cuando está vacío, cuelga de ***Target 1*** el *grupo1*.



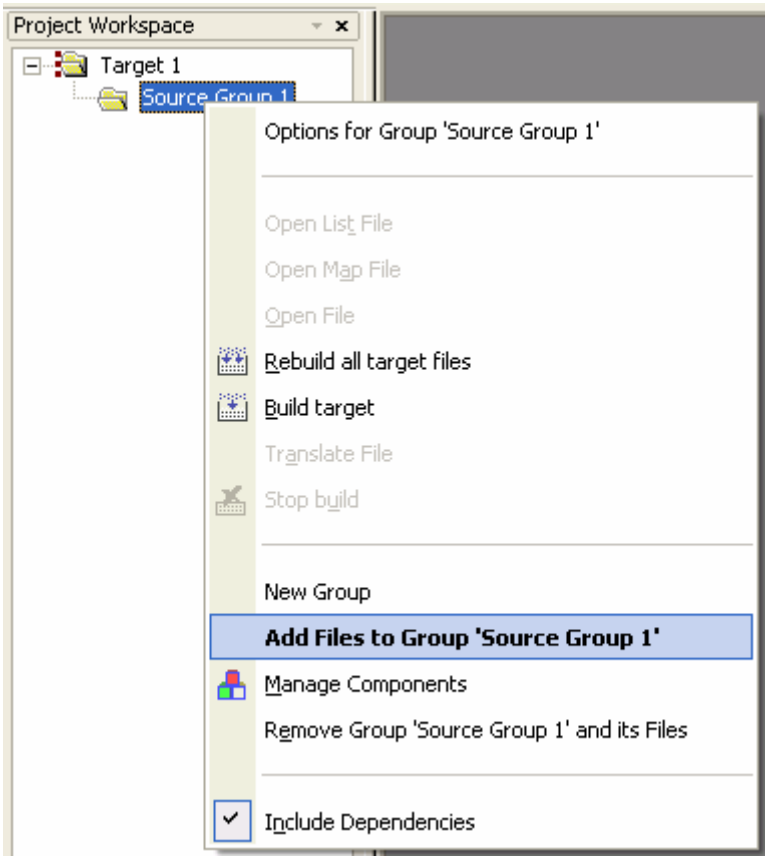
a) Apertura de ventana de proyecto en barra menú



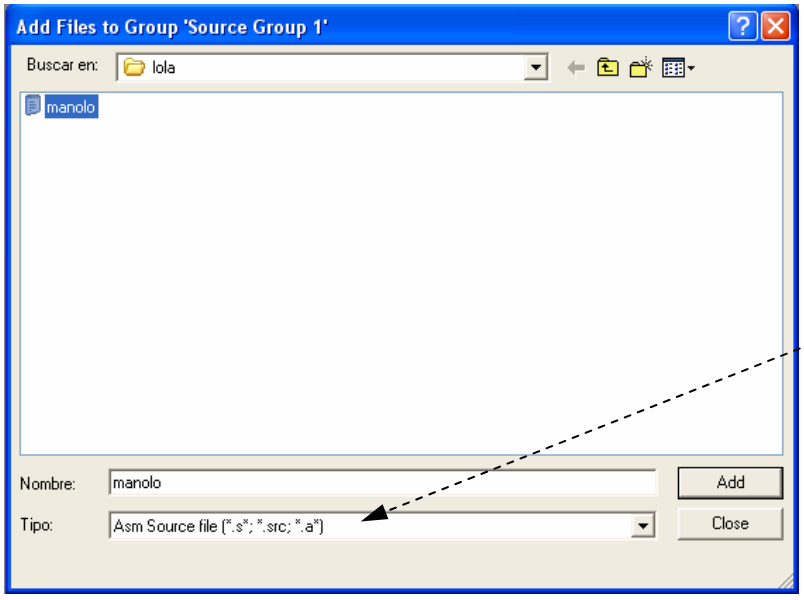
b) Apertura de ventana de proyecto mediante la barra herramientas



c) Resultado de apertura de la ventana del proyecto

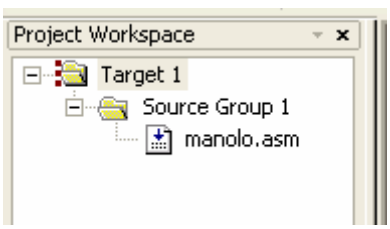


Pinchar en ***Source Group1*** y con el botón derecho del ratón se desplegará un submenú. Seleccionar la opción ***Add Files to Group***.



En la ventana que se despliega, buscar y seleccionar el fichero deseado.

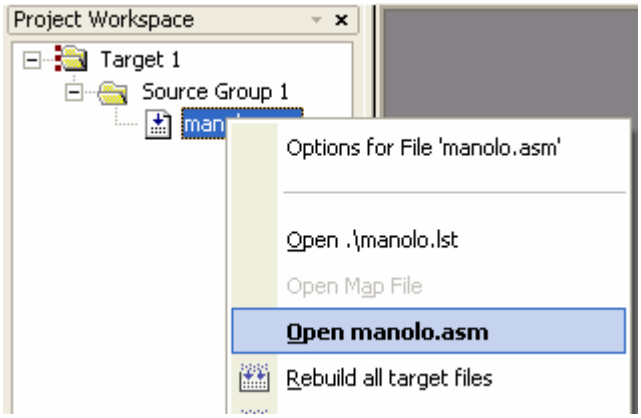
(hay que asegurarse de que la plantilla de búsqueda de ficheros es ***ASM source file*** o bien ***All files***). Para asociar un fichero a un proyecto no es necesario que esté escrito del todo; estando vacío también es posible asociarlo)



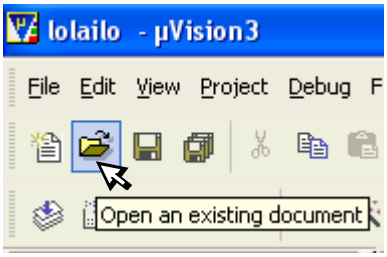
El fichero seleccionado queda, así, añadido al proyecto

3º) EDICIÓN DEL CÓDIGO FUENTE.

Una vez creado un proyecto y asociado(s) un(os) fichero(s), es posible editar uno de ellos abriéndolo. La manera más directa es, en la ventana de proyecto, pincharlo con el ratón (doble pulsación con el botón izquierdo, o botón derecho y seleccionar ***Open*** en el menú emergente). Otra manera es actuando sobre el icono de apertura de fichero, el la barra de herramientas, y buscando y seleccionándolo en la ventana subsiguiente (típica ventana de Windows de búsqueda y selección de un fichero en una estructura de carpetas).

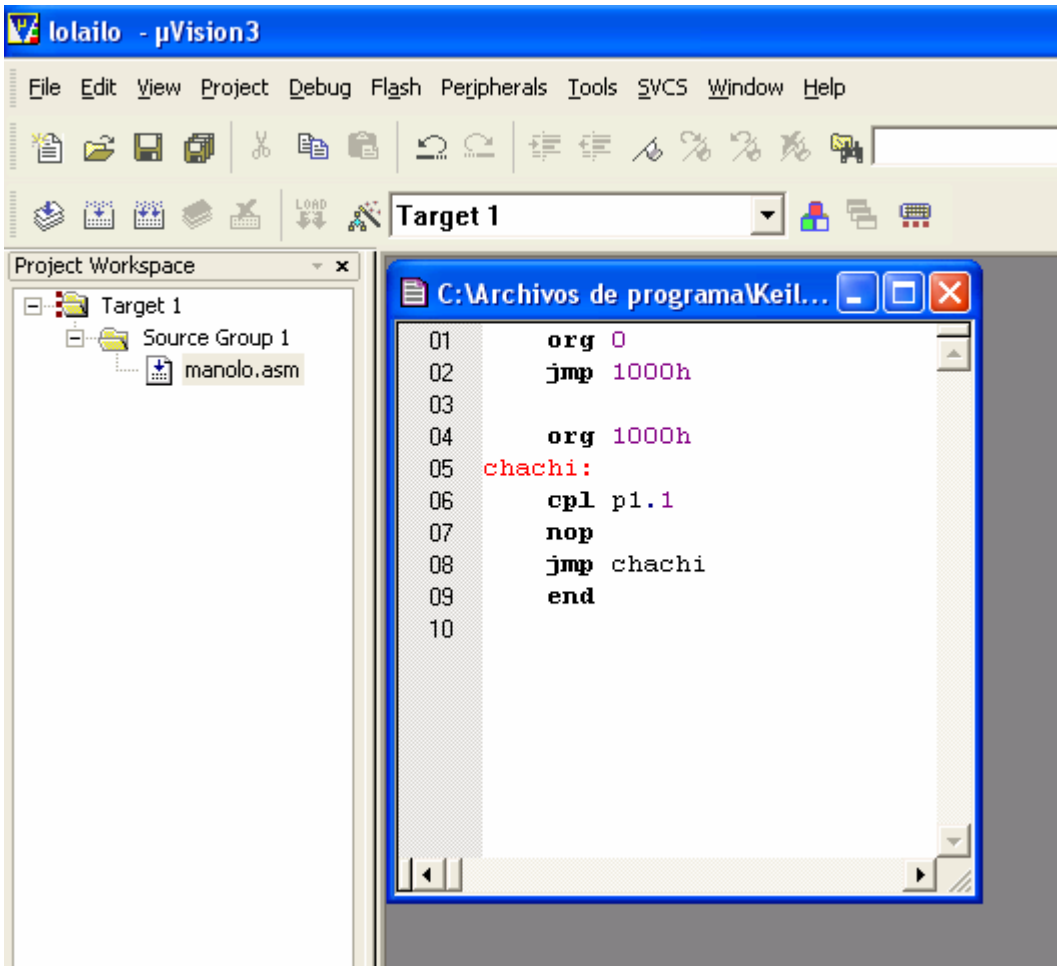


a) Apertura mediante menú local



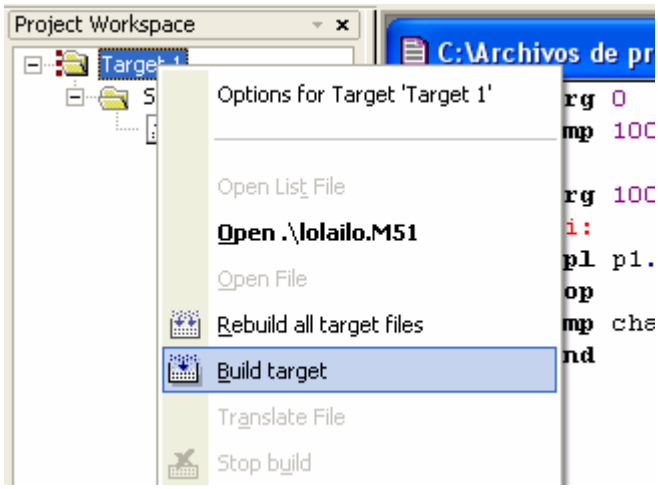
b) Apertura en barra de herramientas

Al abrir un fichero éste puede editarse en la ventana de edición que se abre:

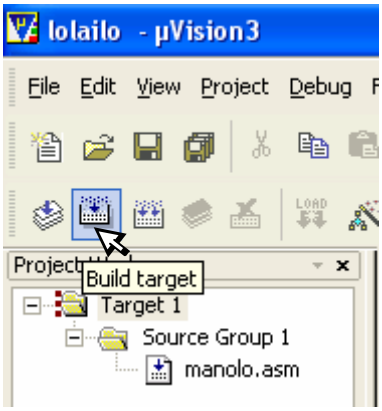


4º) CONSTRUCCIÓN DEL FICHERO EJECUTABLE FINAL (ENSAMBLADO Y MONTADO)

Una vez que se ha terminado de editar los ficheros que conforman un proyecto, hay que crear la aplicación final para poderla depurar. Para ello, pulsar con el botón derecho sobre ***Target 1*** en la ventana de proyecto. Seleccionar ***Build target***. En la ventana de salida se informará de las incidencias del proceso de ensamblado y de montaje de los diferentes módulos fuente que constituyan el proyecto. Otro camino es vía la barra de herramientas.



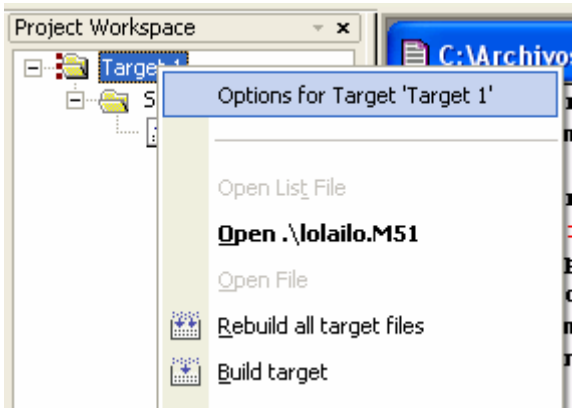
a) Construcción de la aplicación final mediante menú local



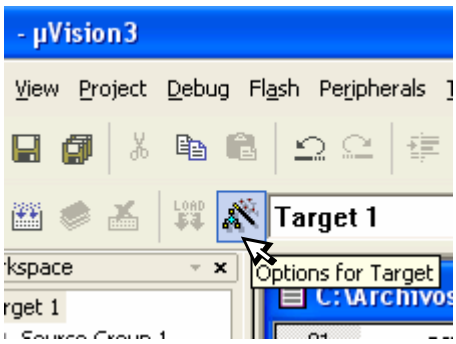
b) Ídem mediante la barra de herramientas

Si se tiene la intención de programar la FLASH ROM de un microcontrolador, es necesario asegurarse al hacer ***Build target*** de que se va a crear el fichero hexadecimal que necesitan los equipos de programación de MCUs. Para ello, hágase lo siguiente: ***Project→Options for target 'Target 1'***. Otra opción es hacerlo con la barra de herramientas.

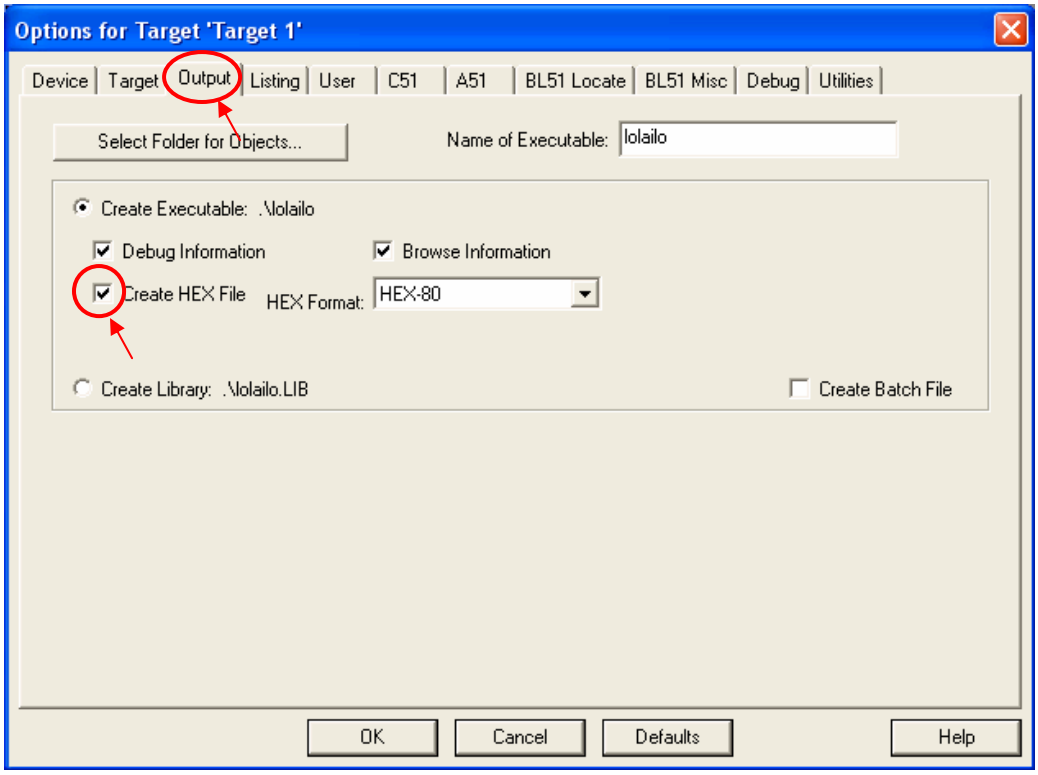
Se desplegará una ventana, y en la pestaña **Output** activar la opción ***Create HEX file***. También, en la pestaña ***Target*** póngase la frecuencia del cristal que se vaya a utilizar en el diseño (esto permite ciertas funciones en la depuración) y márquese la opción ***Use on-chip ROM***.

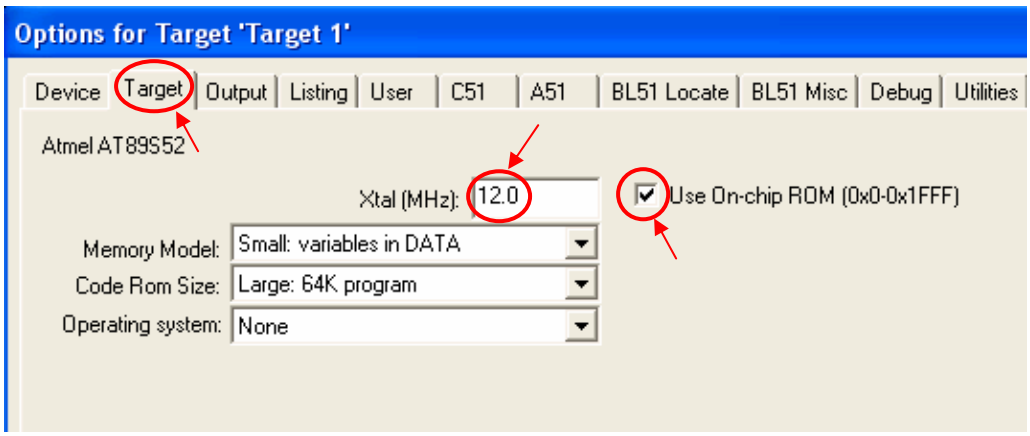


a) Selección de las opciones de salida mediante menú local

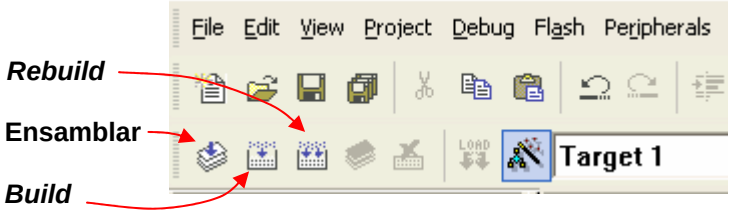


b) Ídem mediante la barra de herramientas



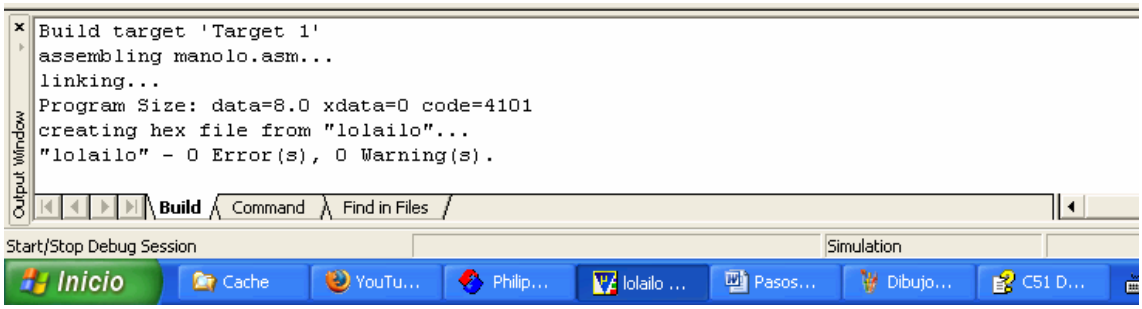


La diferencia entre ***Rebuild target*** y ***Build target*** es que la primera ensambla y enlaza todos los módulos, mientras que la segunda sólo ensambla aquellos módulos que hayan sido cambiados desde la última construcción (*build* o *rebuild*), acelerando así el proceso de actualización del proyecto, y a continuación enlaza todos los módulos objeto.

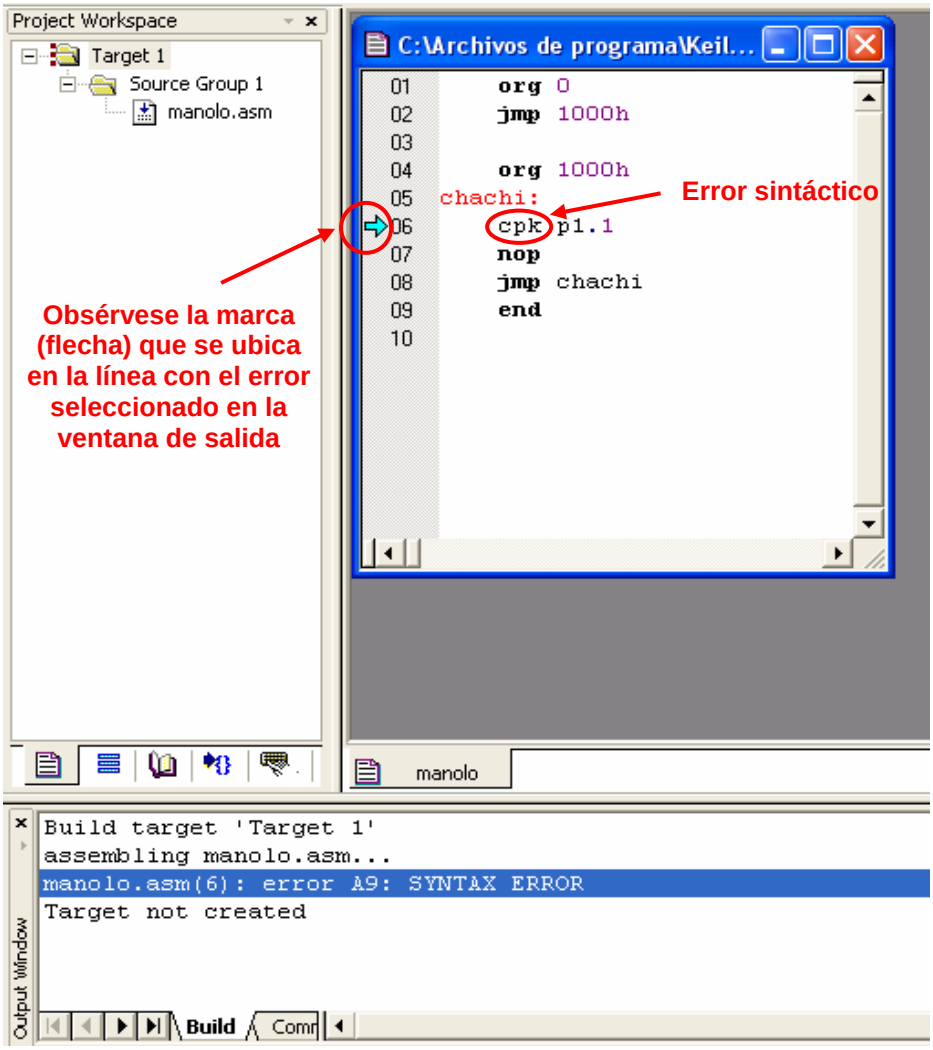


Las opciones *Build* y *Rebuild* ensamblan y montan. La opción de *ensamblar* sólo ensambla

Al ensamblar y montar el o los módulos fuente y objeto de un proyecto pueden o no producirse errores. Las incidencias sucedidas en este proceso de construcción de la aplicación se muestran en la ventana de salida, ubicada en la ventana inferior del entorno de desarrollo Keil µVision. Para ver estas incidencias, es necesario seleccionar la pestaña *Build*, tal y como se observa en la figura que sigue a continuación.



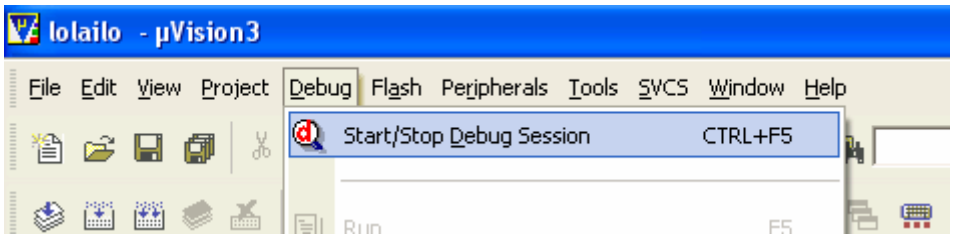
Caso de producirse errores (sean de edición o de montado), basta con una doble pulsación con el ratón en uno de los mensajes de error para que automáticamente se active la ventana oportuna. Por ejemplo, si el error es de edición, se abrirá la ventana de edición del módulo fuente en que se encuentre ese error y el cursor se situará en la línea en que se encuentre ese error, de manera que rápidamente el usuario podrá hacer la oportuna corrección.



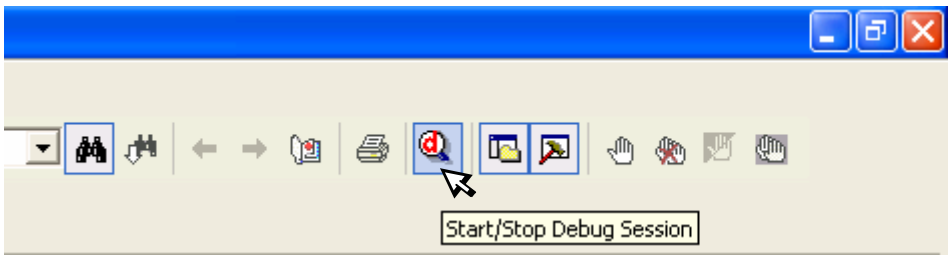
La ventana de salida también posee otras pestañas que permiten seleccionar otras subventanas. Una de ellas es la de órdenes (*Command*), mediante la cual se pueden invocar en *modo consola* las distintas aplicaciones y funciones que conforman el sistema Keil µVision (por ejemplo, el módulo de ensamblado, de montado, etc.). Para más detalle, véase el manual.

5º) DEPURACIÓN DEL CÓDIGO:

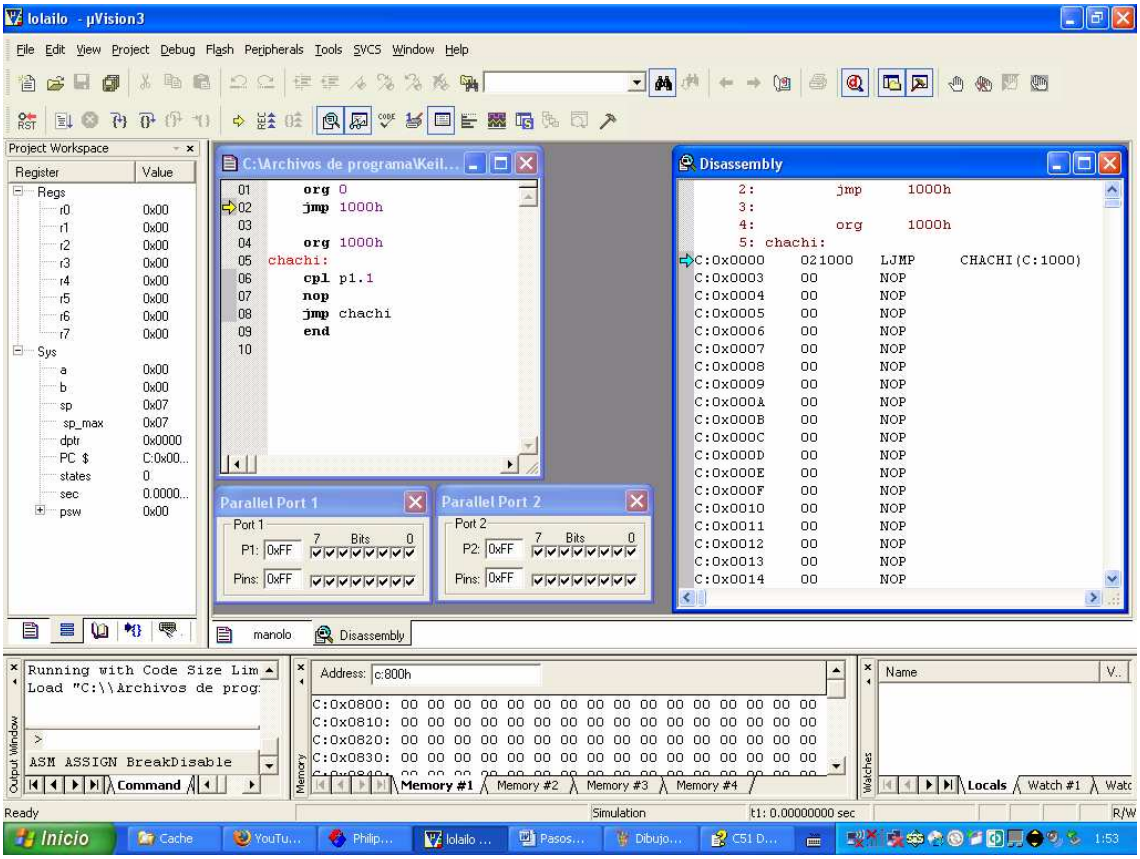
Selecciónese ***Debug*** y en el menú desplegable elíjase ***Start/Stop Debug Session***.



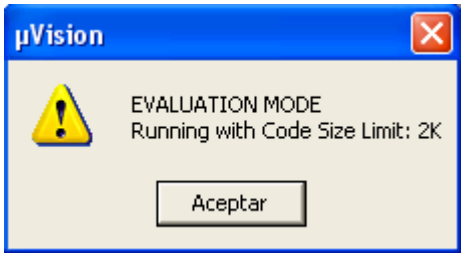
Esto también se puede hacer en la barra de herramientas:



Se entra en la ventana de depuración automáticamente:



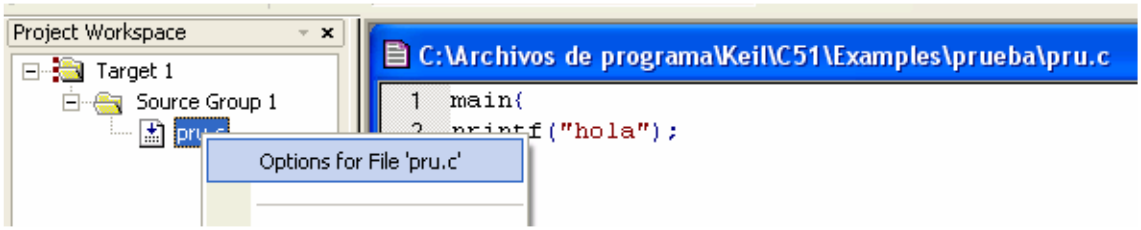
La primera vez que se entra en modo de depuración tan sólo se abre la ventana de código de la aplicación, además de la ventana de salida. En *modo depuración*, la ventana de salida es distinta a la del *modo edición*. En modo depuración las pestañas (y por tanto, las subventanas) son las específicas y necesarias para poder depurar. Por ejemplo, las ventanas de memoria sólo se pueden definir en la ventana de salida, y no como ventanas independientes en la zona central del entorno.
Si se está trabajando con la versión demo de µVision, entonces aparecerá una ventana en la que se advierte de que se está trabajando con una limitación de 2 kilo-octetos de código (este tamaño se refiere no al fichero fuente sino al binario ejecutable, *metible* en ROM):



Realmente, esta limitación de 2K resulta irrelevante en bastantes ocasiones, y lo es siempre en el ámbito didáctico. Con 2K de memoria de código se pueden desarrollar aplicaciones realmente complejas si se programa en lenguaje ensamblador; en este lenguaje se puede optimizar notablemente el tamaño del código si se tiene suficiente experiencia y soltura en su empleo.

Caso de trabajar en lenguaje C
Si se programa en lenguaje C –Keil µVision incluye un compilador de lenguaje C– entonces el consumo de memoria es o suele ser muy superior; para intentar minimizar esto, resulta imprescindible configurar adecuadamente las opciones de compilación en lo que se refiere a las técnicas de optimización utilizadas por el compilador. Esta cuestión no se tratará en este prontuario, que se centra en el uso del lenguaje ensamblador. No obstante, los pasos sería los siguientes:

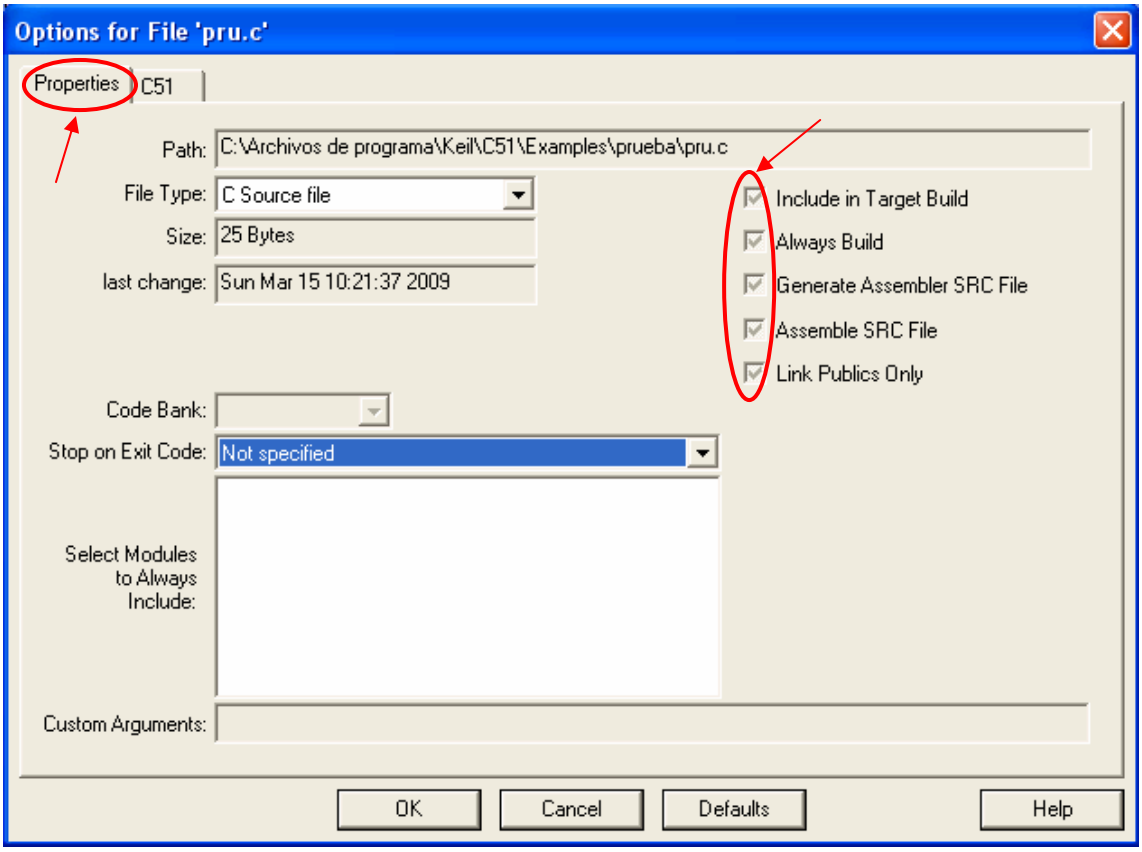
1º) Definir las opciones del fichero fuente escrito en C; para ello hágase ***Project→Options for file ????.C*** vía barra de menú o caminos alternativos, siendo *????.C* el nombre del fichero con el que se trabaja. Por ejemplo:

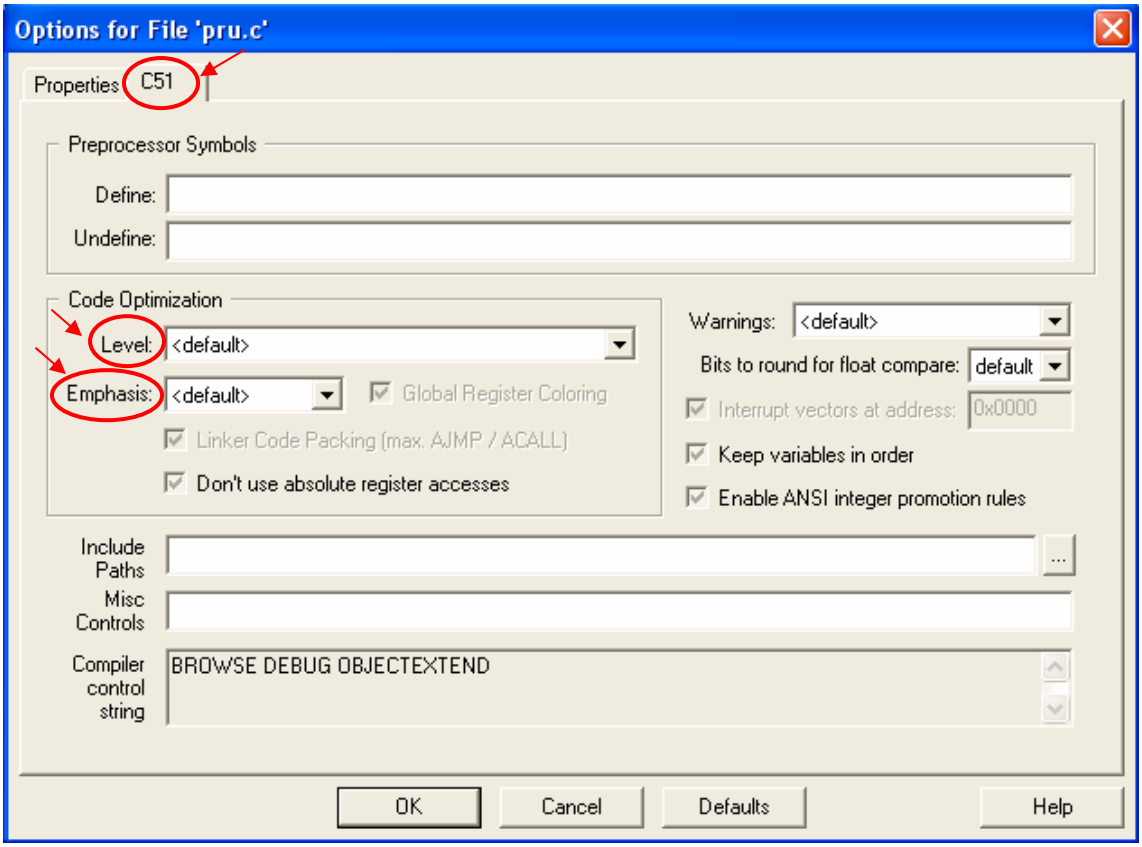


2º) Se abrirá una ventana de configuración de opciones de compilado. En esta ventana, actuando sobre las pestañas oportunas (***Properties*** o ***C51***, es decir, *Propiedades* y *Compilador C51*) se podrá seleccionar, entre otras cosas:

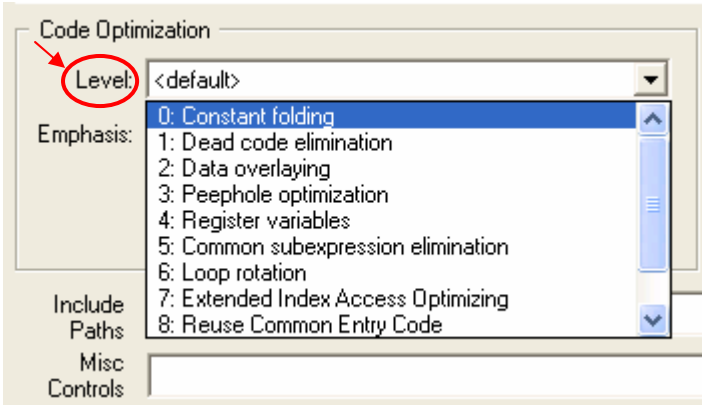
- Si ese módulo en C se incluye en la construcción del proyecto (compilado)
- Si, al compilar, se generará el fichero fuente en ensamblador. Esto es útil para poder ver cómo de eficiente es el código generado y para afinarlo a mano.
- Nivel de optimización del código
- Criterio principal utilizado: optimizar el tamaño o la velocidad del código

Esto puede verse en las siguientes figuras.

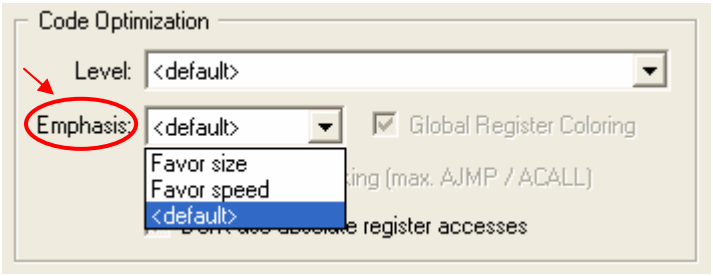




Opciones del nivel de optimización del código:



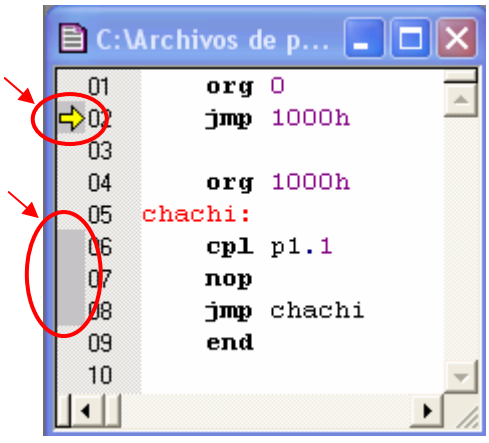
Opciones del énfasis en la optimización:



Para entender en qué consisten los niveles de optimización, y las implicaciones que tienen, consúltese el manual del compilador.

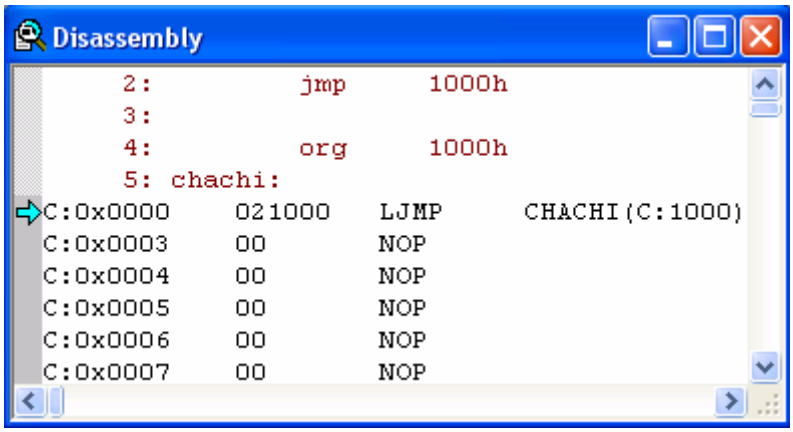
PREPARACIÓN PARA LA DEPURACIÓN DEL CÓDIGO ENSAMBLADOR

Antes de empezar a depurar conviene abrir las ventanas oportunas que ayuden durante el proceso de la depuración. Por ejemplo, las de memoria, pila, etcétera. Para ello en ***View*** selecciónese las ventanas oportunas.
En las figuras siguientes se pueden ver las ventanas de código editado (se abre siempre por defecto) y la de código desensamblado.



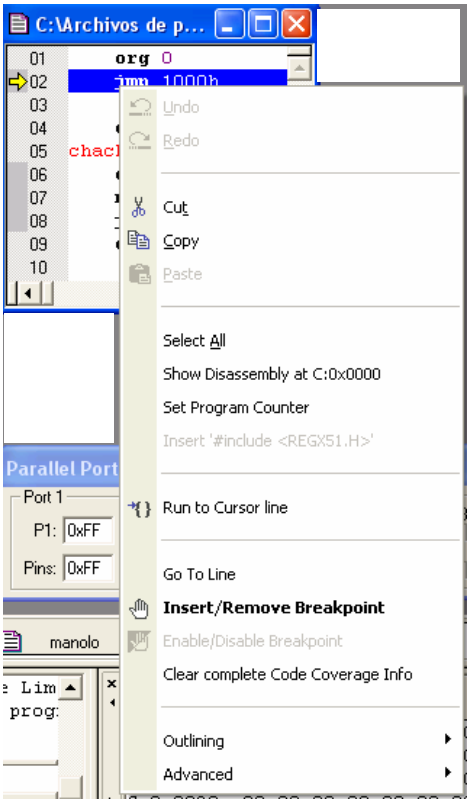
La ventana de código editado presenta la misma información que la de edición, pues de hecho es editable. Además, a la izquierda se puede observar, por un lado, una flecha amarilla que indica la próxima instrucción a ejecutarse (es decir, a dónde apunta el contador de programa); y por otro, la parte del código fuente que es ejecutable (sombreado en gris azulado).
Sobre esta ventana se puede interactuar, como atajo, para realizar ciertos procesos en la depuración. Por ejemplo, poner puntos de ruptura. Más adelante se hablará sobre cómo hacerlo

Fig. Ventana de código

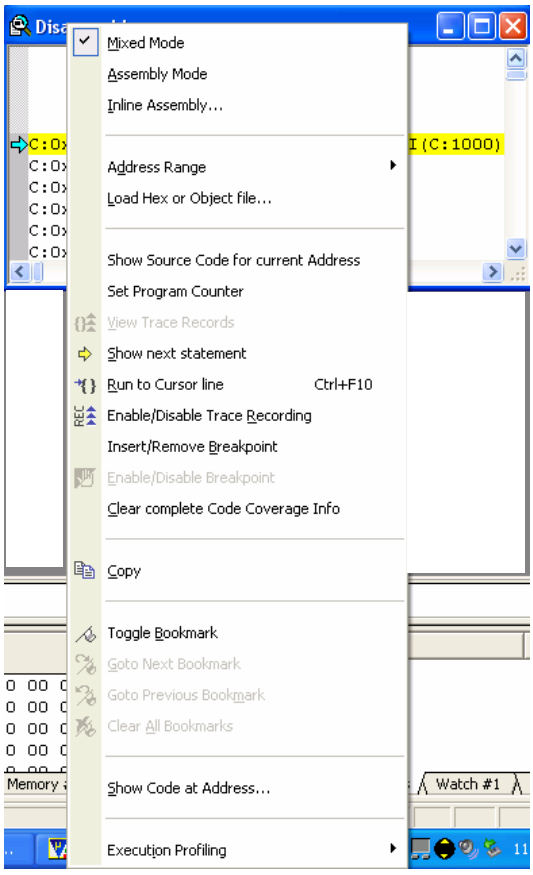


En la llamada ventana de desensamblado se presenta la misma información que en la de edición, pero además se muestra su codificación binaria (en equivalente hexadecimal), junto con la dirección de memoria en que se inicia cada instrucción.

Igualmente, en el borde izquierdo se aprecia una flecha azul turquesa que significa lo mismo que la amarilla de la de código: próxima instrucción a ejecutarse.
Pinchando sobre una línea de estas ventanas (de código o de desensamblado) se puede interactuar con el proceso de depuración. Y si se pincha y pulsa el botón derecho del ratón, emerge una ventana con todas posibilidades de interactuación:



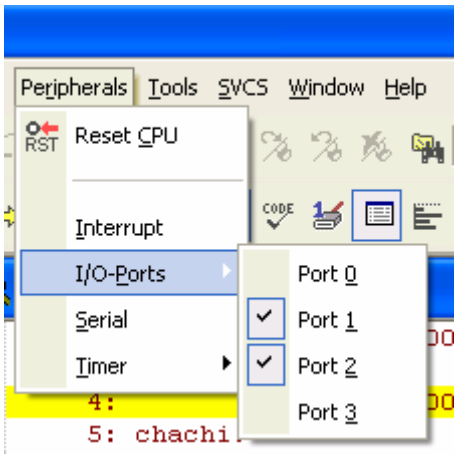
a) emergente en la de edición

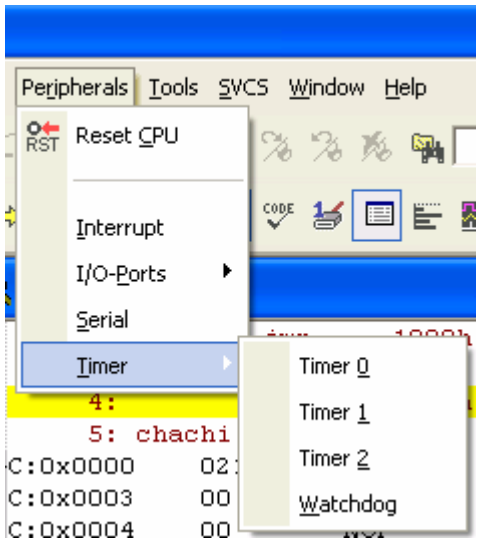


b) emergente en la de desensamblado

Como es lógico, aparecerán algunas opciones diferentes en cada ventana emergente, dada la distinta naturaleza de las ventanas de edición y de desensamblado (por ejemplo, en la de edición se podrán cortar y pegar líneas, cosa que no tiene sentido en la de desensamblado). No obstante, opciones básicas de depuración se pueden llevar a cabo sobre ambas.

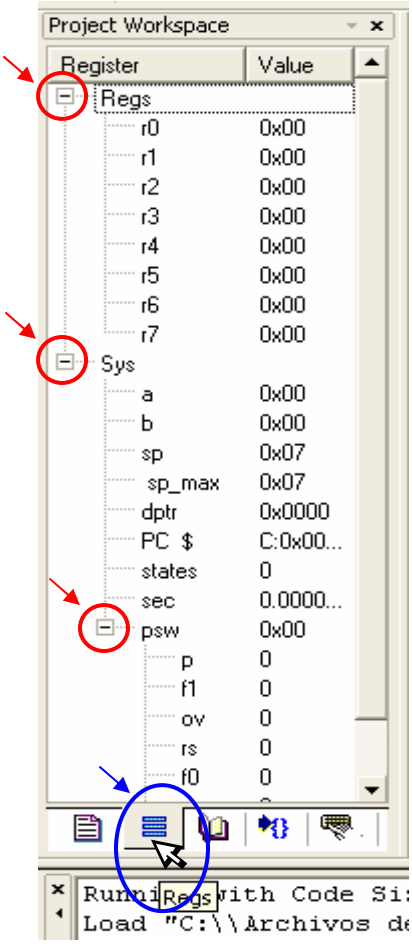
Para abrir otras ventanas para la depuración, lo mismo es aplicable a las ventanas de periféricos; para ello en ***Peripherals*** selecciónese en el desplegable las ventanas deseadas (interrupciones, puertos paralelos, puerto serie y temporizadores):





Ventana de registros (en ventana de proyecto):

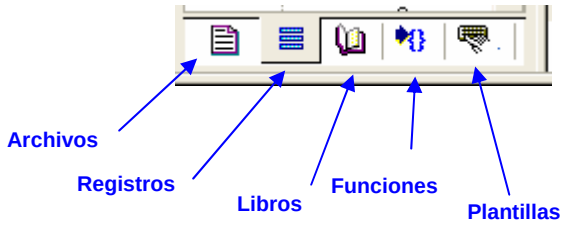
Para ver los registros característicos, selecciónese ***View→Project Window***. Aparecerán los registros Rn, acumulador A, registro B, puntero DPTR, SPW (registro de estado), etc. Obsérvese que la *ventana de proyecto* posee varias pestañas, por lo que puede conmutarse a voluntad la ventana mostrada.



Obsérvese la típica estructura arborescente de la información mostrada en la ventana de registros, por lo que a voluntad puede expandirse o colapsarse la información mostrada, pinchando en el signo **+** ó en el **–** del ítem deseado.

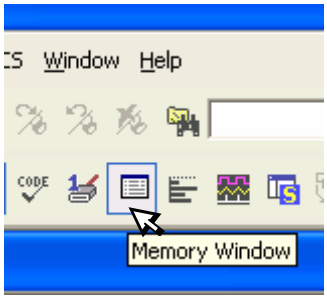
En la parte inferior de la ventana del proyecto pueden observarse las pestañas ya mencionadas. En la figura se encuentra seleccionada la de registros internos del microcontrolador. Si se desease ver otra ventana relacionada con el proyecto, bastaría con pulsar sobre la pestaña adecuada:

- **Archivos**: para ver la estructura dada al proyecto
- **Registros**: para ver los registros de trabajo
- **Libros**: para consultar los manuales y hojas técnicas
- **Funciones**
- **Plantillas**

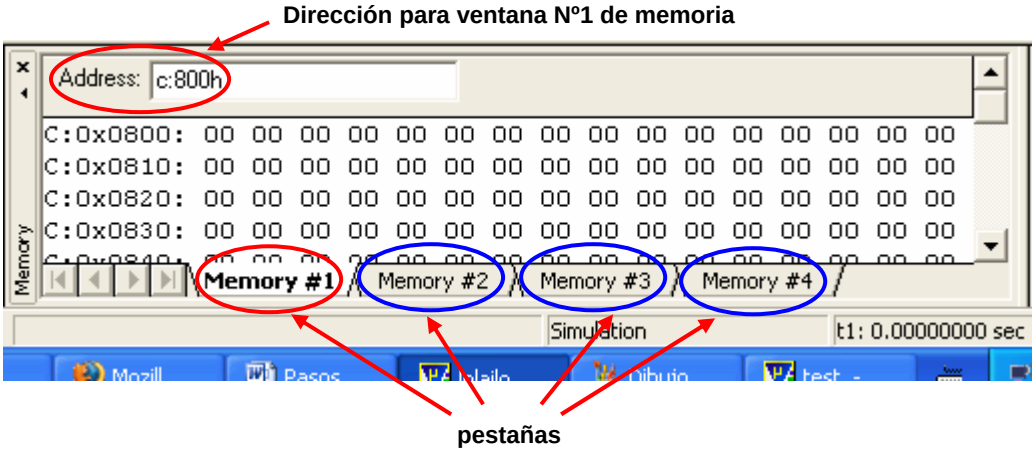


Ventanas de memoria:

Se abren haciendo ***View→Memory Window*** en la barra de menú, o pulsando sobre el icono oportuno en la de herramientas:



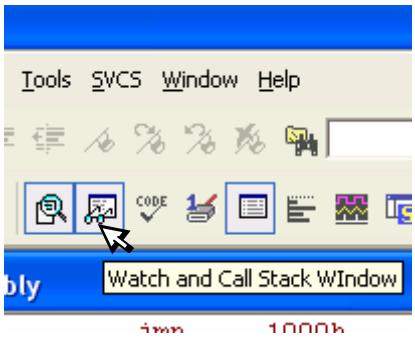
Al abrir la ventana de memoria, aparece una ventana con múltiples pestañas, una para cada ventana de memoria. Cada una de ellas tiene un campo de dirección (***address***) en el que se puede introducir un valor a partir del que se verá la memoria:



Dado que en un µC tipo 8051 existen varios tipos de memoria con idéntico valor de dirección, con un prefijo se indica el tipo de memoria que se desea ver. Para código **C:** (por ejemplo, C:1000h ó C:0x1000 para ver la memoria de código a partir de la dirección 1000h). **D:** para la RAM interna; **X:** para la memoria externa.

Ventana de observación

Se abre haciendo ***View→Watch & Call Stack Window*** en la barra de menú, o pulsando sobre el icono oportuno en la de herramientas:

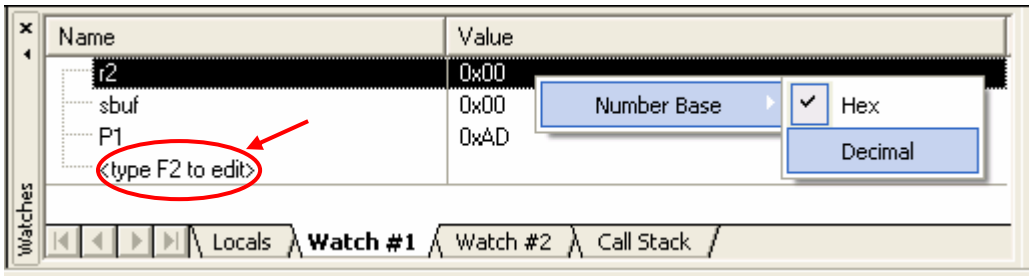


Esta ventana tiene varias pestañas:

- **Variables locales**: para ver las variables que se hayan declarado.
- **Observación Nº1**: para agrupar en una sola ventana todos los elementos (registros y variables) que se deseen. De esta manera se evita tener que mirar varias ventanas diferentes.
- **Observación Nº2**: una segunda ventana.
- **Pila de llamadas** a subrutinas: para saber la estructura de llamadas a subrutina concatenadas.

La utilidad de la *ventana de observación y pila de llamadas*, como se ha dicho, es que permite agrupar en una sola ventana información parcial –pero fundamental para la depuración– que de otra manera estaría repartida entre otras muy diversas ventanas: registros, memorias, pila o código.

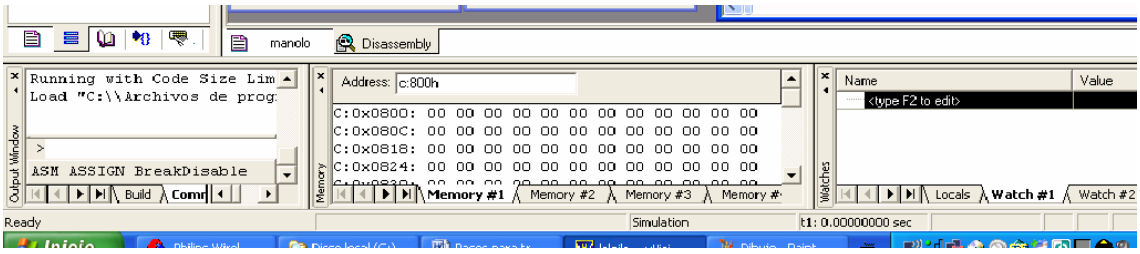
Una ventana de observación hay que dotarla de contenido. Para ello, se pulsa la tecla **F2** y en el campo que se abre se teclea el nombre del recurso. A pulsar <intro> automáticamente se añade ese elemento a la lista de elementos en la ventana de observación y se muestra su valor.



Una vez añadido un recurso a la ventana, es posible seleccionar el formato de presentación de su valor. Para ello, se selecciona el elemento (da igual hacerlo en el campo del nombre o en el del valor), y con el botón derecho del ratón emerge una subventana en la que se puede seleccionar el sistema de numeración en el que se mostrará el valor, tal y como puede verse en la imagen anterior.

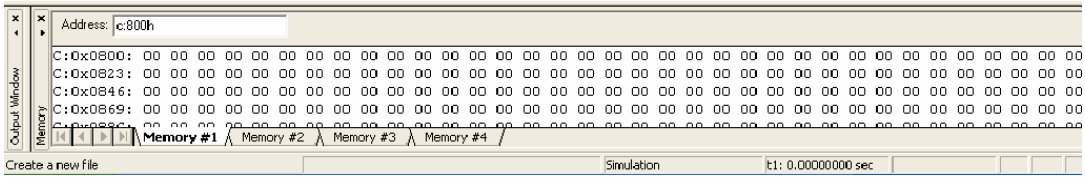
Dimensionado de las ventanas solidarias

Llamamos *ventanas solidarias* a las que aparecen en la parte inferior del depurador, y que no tienen la apariencia típica de toda ventana clásica de Windows. Por ejemplo, son estas ventanas las de salida, las de memoria, las de observación y pila de llamadas.

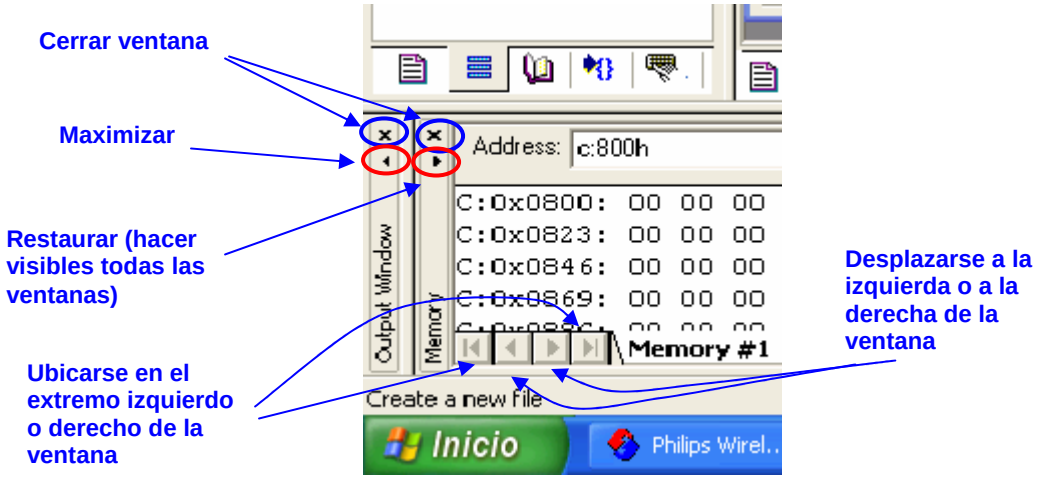


Estas ventanas pueden de alguna forma redimensionarse y *maximizarse*. El redimensionado se refiere al tamaño relativo dentro del entorno Keil µVision. Para ello bastará con ubicar el puntero del ratón en una zona fronteriza hasta que el símbolo del puntero del ratón adopte la forma del *redimesionador de campo*: ←||→ para el horizontal y el rotado 90º para el vertical. Para redimensionar, bastará con pulsar el botón izquierdo del ratón y arrastrarlo hasta que se redimensione en la forma deseada. Es decir, lo típico de una aplicación Windows en lo que respecta al redimensionado de campos dentro de una aplicación (redimensionado de campos, que no de ventanas), como es el caso de las celdas de una tabla en Microsoft Word o Excel.

Por otro lado, la *maximización* de este tipo de ventana implica la minimización de las restantes. En la figura que sigue puede verse el aspecto al maximizar la ventana de memoria:



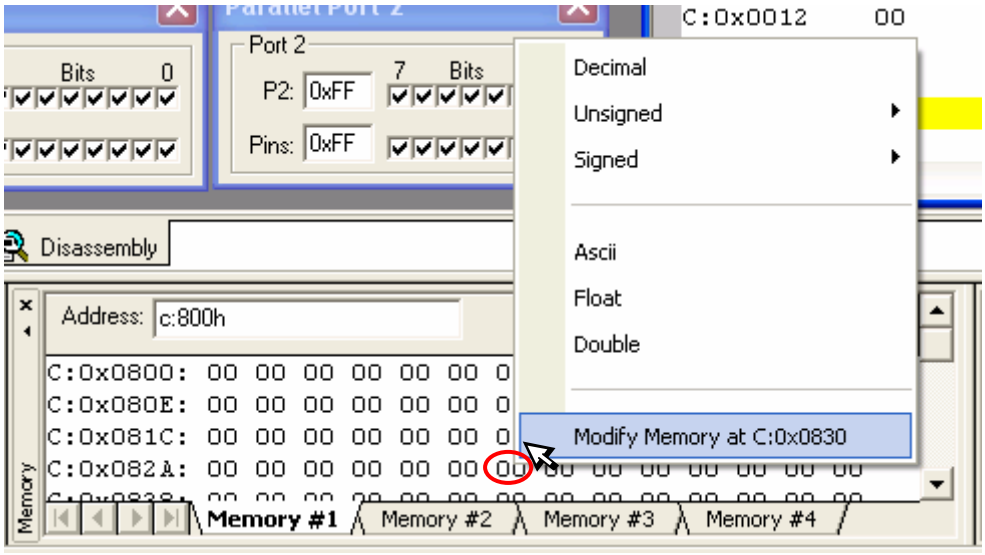
Si se observa la barra vertical a la izquierda de cada ventana, pueden verse unos símbolos significativos:



Igualmente, si la ventana no es lo suficientemente ancha, en la esquina inferior izquierda aparecen los típicos símbolos de las aplicaciones Windows mediante los que es posible desplazar la parte visible de la ventana. Todo esto puede verse en la figura de arriba

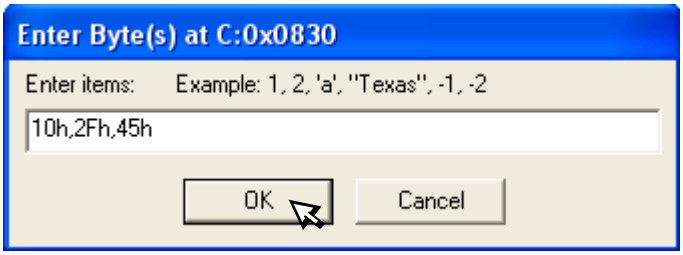
Introducción de valores numéricos en las ventanas

Hay ventanas en las que se muestra información sobre recursos del sistema: registros internos, posiciones de memoria, variables, etcétera. En este tipo de ventanas habitualmente es posible modificar al vuelo el valor de uno o varios de esos recursos. La manera de hacerlo es la típica e intuitiva de Windows: con el ratón se pincha el elemento y con el botón derecho se despliega un menú específico con las posibles acciones a realizar con ese elemento:

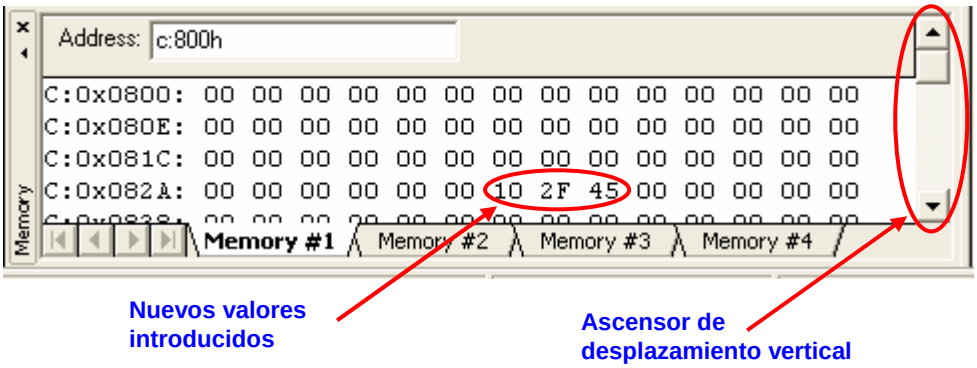


En esta imagen se ve la ventana emergente al abrir con el ratón la posición 0x0830. También pueden verse las opciones de presentación de los valores en la ventana de memoria. Si no está seleccionado el modo ***Decimal***, se muestra en hexadecimal (salvo formato incompatible).

Al seleccionar la opción (en este caso, modificar introduciendo un nuevo valor), se abre una ventana en la que se puede introducir un nuevo valor. Si lo que se quiere hacer es introducir varios valores en posiciones consecutivas a partir de una dada, se teclean los valores separados por comas. Por ejemplo:

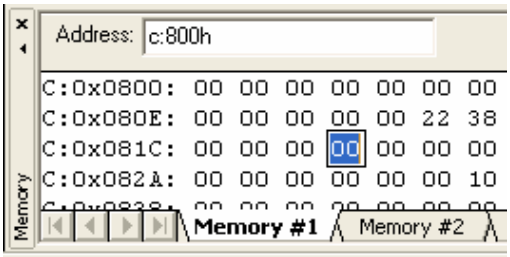


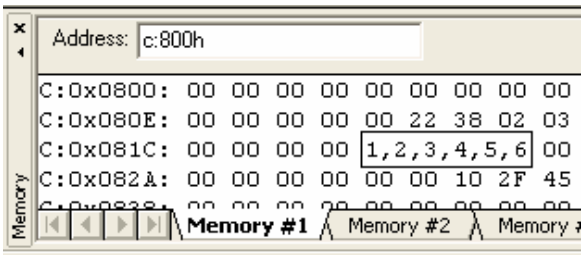
Aquí, en este ejemplo, se introducirán los valores 10h, 2Fh y 45h en las posiciones 0830h, 0831h y 0832h de la memoria de programa. Al validar los nuevos valores, en la ventana de memoria del ejemplo se visualizará lo que sigue:



Si en una ventana no cupiese toda la información visualizable, se disponen de los típicos *ascensores* de una ventana Windows.

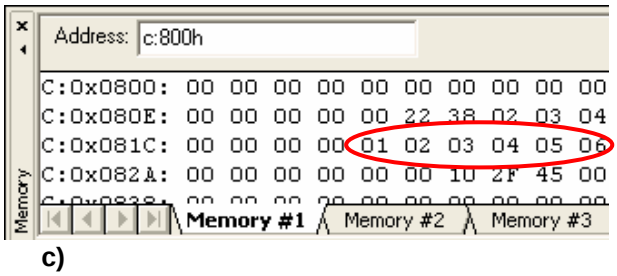
Otra manera de introducir valores es haciendo una doble pulsación sobre la posición que se desea modificar o a partir de la que se desea hacerlo (**a**). Se abrirá un campo numérico, y el usuario introducirá el nuevo valor, o los valores sucesivos separados por comas (**b**).





a) **b)**

En estas figuras puede verse el proceso de introducción de valores y el resultado final en la memoria al pulsarse finalmente <intro> (**c**).



c)

Código de colores

En la ventana de memoria los contenidos de las posiciones se muestran empleando un código de colores:

■	**Negro**	Indica memoria que es de programa o que no se usa en la aplicación
■	**Rojo**	Para datos CONST en FLASH o ROM que se han accedido al menos una vez
■	**Oro**	Indica memoria que se ha iniciado, pero que no se ha accedido todavía
■	**Verde**	Indica que la posición de memoria se ha accedido al menos una vez

Formato numérico de los valores introducidos o mostrados

Como toda herramienta de desarrollo, se admiten diversos sistemas de numeración a la hora de introducir o de mostrarse los valores de la memoria. La notación que emplea Keil µVision es la indicada en la siguiente tabla:

BASE	PREFIJO	SUFIJO	EJEMPLO
binario	No admitido	**Y** ó **y**	**101101y**
octal	No admitido	**Q**, **q**, **O** u **o**	**6721q** (1)
decimal	No admitido	**T** o ninguno (2)	**1743** ó **1741T**
hexadecimal	**0x** ó **0X**	**H o h**	**0xA75F** ó **0A75Fh**(3)

(1) En octal no se recomienda el uso de **Q**, **O** u **o** para evitar la confusión con el carácter 0 (cero)
(2) *Ninguno* sólo si el sistema utilizado por defecto es el decimal
(3) Obsérvese el uso de un **0** (cero) delante del dígito hexadecimal **A**, al ser un dígito tipo *carácter letra*.

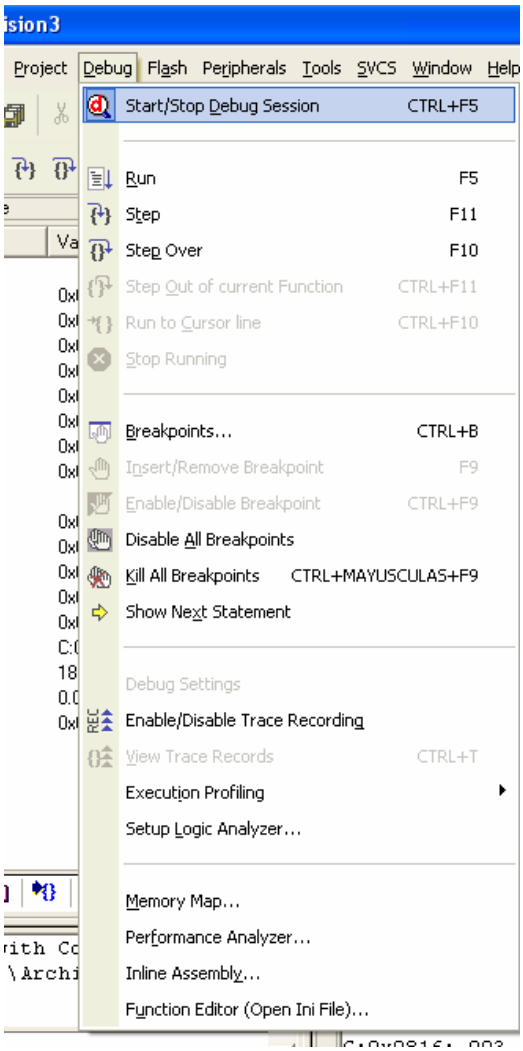
Cuando se desea indicar el valor numérico de un carácter **ASCII**, entonces es posible notarlo no en formato numérico sino en formato carácter. Si se trata de un solo carácter, se encerrará entre apóstrofos (**‘**). Si es una sucesión de caracteres, se encerrarán entre comillas (**“**).
Por ejemplo: **‘A’** es equivalente a haber escrito **0x41** ó **41h**. **“Hola**” será equivalente a haber escrito **‘H’**,**’o’**,**’l’**,**’a’** o a haber escrito **48h**, **4Fh**, **4Ch**, **41h**.

Cuando en la ventana de memoria se van a introducir valores, no es necesario utilizar prefijo ni sufijo alguno si el sistema a utilizar es el establecido por defecto. Por ejemplo, si en la ventana de memoria se ha seleccionado ver los valores en hexadecimal (*unsigned char* y *Decimal* desactivado), entonces se puede entrar el valor 7E sin necesidad de explicitar 0x7E ó 7Eh.

UTILIDADES PARA LA DEPURACIÓN

Para la depuración se tienen las utilidades usuales en cualquier herramienta de desarrollo: ejecución, paso a paso, puntos de ruptura, etcétera. Las diversas opciones se pueden ver en la opción ***Debug*** de la barra de menú.
Si se ha manejado ya alguna herramienta de depuración se tiene la base para de manera intuitiva familiarizarse con las peculiaridades de Keil µVision.

En la siguiente figura pueden verse las distintas opciones:



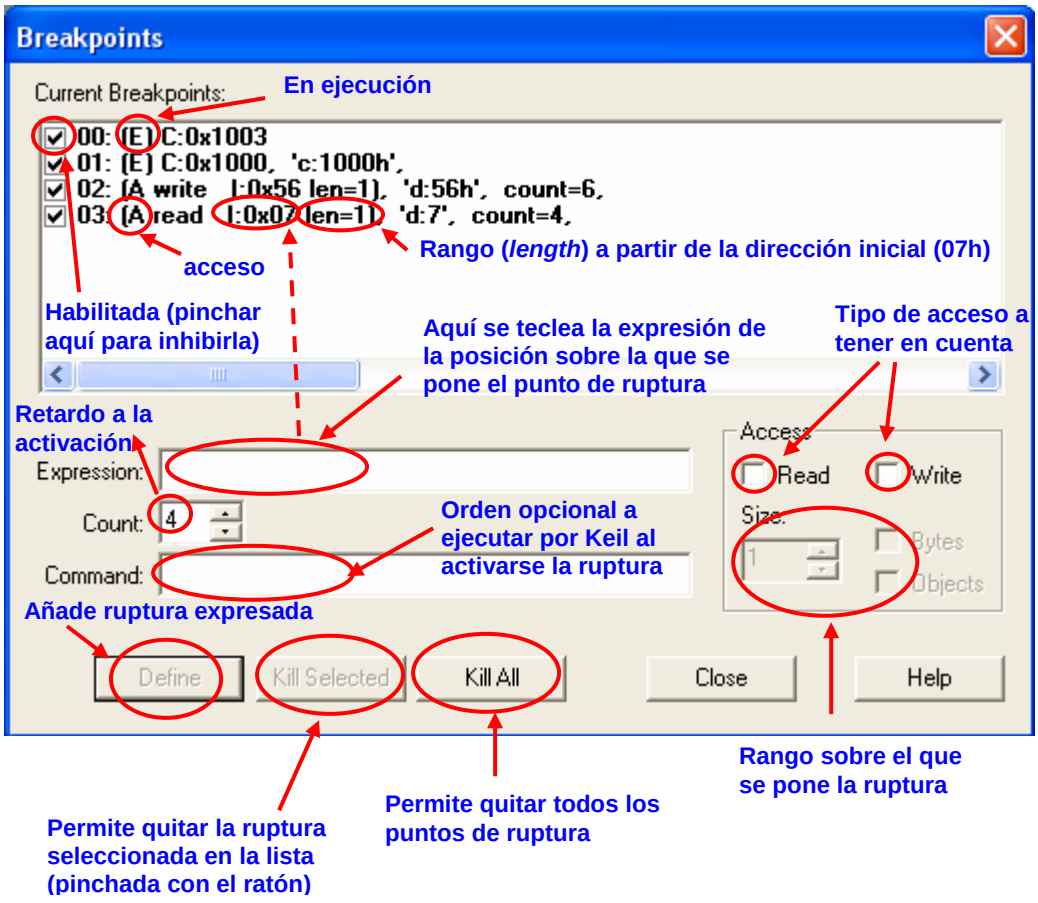
- **Ejecutar** (*Run*): Ejecuta A velocidad rápida el código
- **Paso a paso** (**Step**): Ejecuta sólo la instrucción a la que apunta el contador de programa.
- **Paso largo** (*Step Over*): Permite ejecutar una instrucción CALL sin entrar paso a paso en la subrutina (ésta se ejecuta rápido).
- **Ejecutar hasta retornar** (Step Out): Habiendo entrado en una subrutina, ejecuta rápido hasta llegar a un RET.
- **Ejecutar hasta** (*Run to Cursor line*). Si con el ratón se marca una línea de código, permite una ejecución rápida hasta alcanzarse esa línea.
- **Parar ejecución** (*Stop Running*): Si se ha dado la orden de ejecutar, permite detener la ejecución.
- **Puntos de ruptura** (*Breakpoints*)**:** Permite establecer o ver puntos de ruptura, asignando atributos.
- **Insertar/Quitar punto de ruptura** (*Insert/Remove Breakpoint*). Permite poner o quitar un punto de ruptura en la instrucción seleccionada con el ratón en la ventana de edición o en la de desensamblado.
- **Habilitar/Inhibir punto de ruptura** (*Enable/Disable Breakpoint*): Permite habilitar o inhibir el punto de ruptura seleccionado con el ratón en la ventana de edición o de desensamblado, pero no lo suprime.

Puntos de ruptura

Cuando se selecciona la opción *Puntos de Ruptura* emerge una ventana en la que se puede indicar en qué posición de memoria se pone el punto, si es de código, de datos, externa o SFR, si el acceso es de lectura, de escritura o ambos; si se le añade un factor de retardo a la activación, etcétera. Observando la figura que sigue puede intuirse cómo actuar. Si ya hubiese otros puntos de ruptura, aparecerán listados, así como sus atributos.
En esta figura pueden verse cuatro puntos de ruptura:

- Una en la posición 1000h de la memoria de programa, que se encuentra habilitada.
- Otra en 1003h de la memoria de programa, que también está habilitada.
- Otra en la posición 56h de la RAM interna, efectiva sólo cuando se escriba por sexta vez en esa posición
- Finalmente, otra en la posición 07h de la RAM interna (R7 del banco 0), efectiva sólo cuando se lea por cuarta vez (es decir, cuando R7 del banco 0 se use por cuarta vez)

Como puede verse, existe una gran flexibilidad a la hora de definir un punto de ruptura.



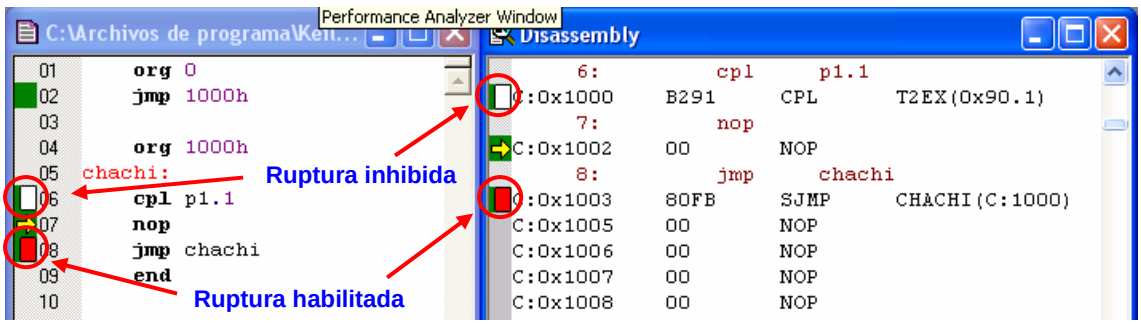
Como puede verse, es posible poner puntos de ruptura sobre un rango de posiciones consecutivas. Esto es un método más cómodo comparado con poner puntos individuales sobre cada posición del rango (en el ejemplo de la figura superior no se ha puesto rupturas sobre un rango sino sólo sobre direcciones individuales).

Poner/quitar rápidamente una ruptura

Una manera rápida de poner o quitar un punto de ruptura es con una doble pulsación del ratón sobre la instrucción en la que se quiere poner o quitar la ruptura. Esto se hará en cualquier punto de la línea en la ventana de desensamblado, o en el campo izquierdo (sombreado) de la línea en la ventana de edición.

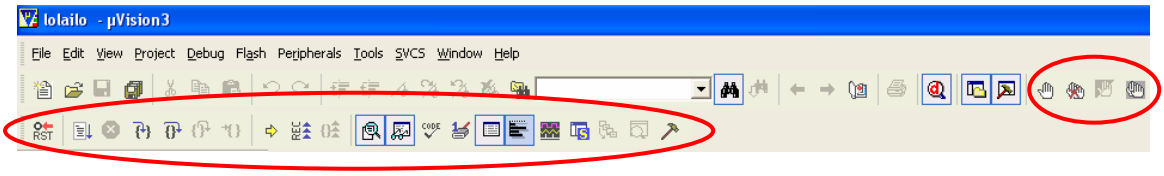
Codificación por colores de las rupturas

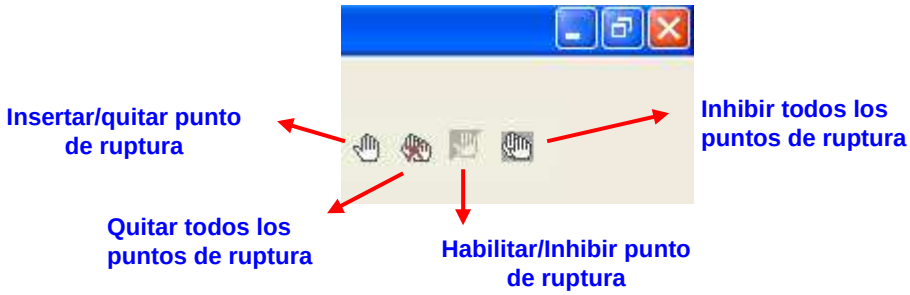
Al poner puntos de ruptura, en las ventanas de edición y de desensamblado se fijan unas marcas de colores que indican tal eventualidad. Una marca en rojo indica un punto de ruptura en la instrucción de esa línea. Si el punto de ruptura se hubiese inhibido temporalmente, entonces aparecerá la marca en color blanco. Esto puede apreciarse en la siguiente figura:

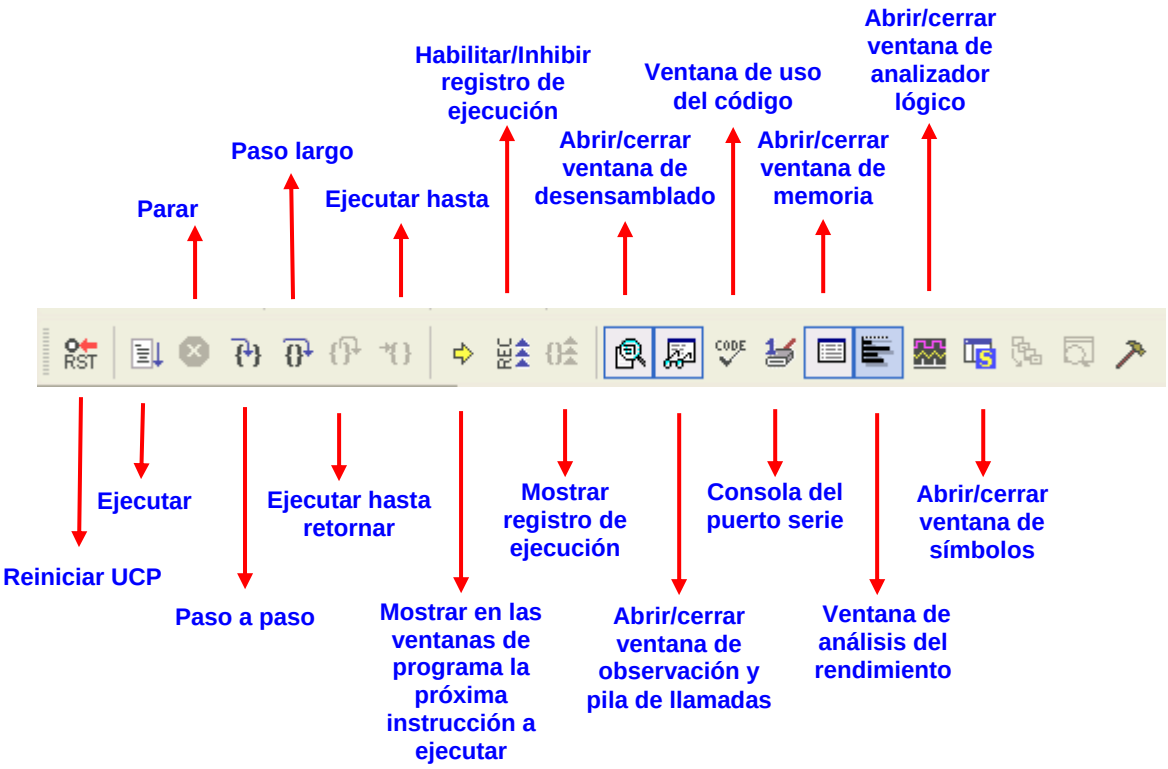


Control de la depuración mediante la barra de herramientas

Como se sabe, la barra de herramientas supone un atajo para los comandos más usuales, incluidos los de depuración. Se tienen los siguientes:







La herramienta de mostrar en las ventanas de edición y de desensamblado la próxima instrucción a ejecutar tiene su utilidad en proporcionar una vía rápida de volver a ver el código próximo a ejecutar si en esas ventanas nos hemos movido hacia delante o detrás mediante los ascensores de ventana.
El registro de ejecución se refiere a guardar un registro lineal de las instrucciones que se han ido ejecutando en el tiempo. Puede ser útil en ciertos casos de depuración, pero normalmente carece de utilidad.

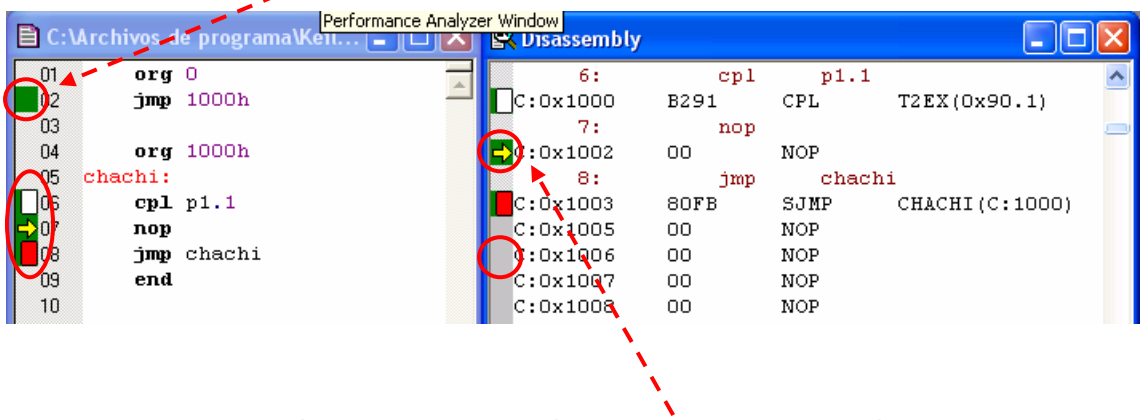
Atajos mediante el teclado.

Como se ha visto, muchos de los procesos de depuración se pueden llevar a cabo de manera más rápida actuando sobre los iconos de la barra de herramientas. No obstante, en algunos casos es todavía más rápido usar las teclas de función. Entre otras:

Ejecutar:	**F5**
Paso a paso:	**F11**
Ejecutar hasta retornar:	**F10**
Poner o quitar ruptura:	**F9**
Ejecutar hasta:	**CTRL+F10**

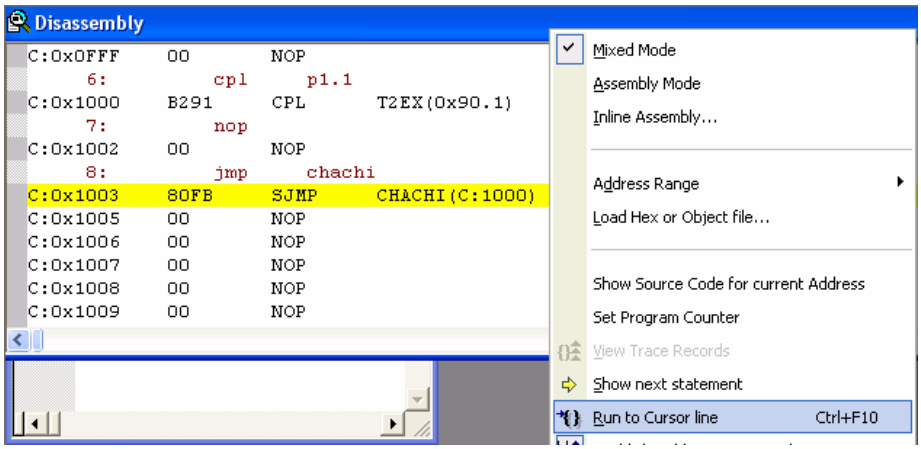
Control de la ejecución

Lo habitual es empezar la depuración ejecutando paso a paso. Sólo se ejecutará de corrido aquellas porciones de código ya depuradas.
La manera más rápida de hacerlo es con el icono de ***Paso a Paso*** en la barra de herramientas o, más aún, con la tecla **F11**. Sea la que sea la opción u opciones que se hayan elegido para la ejecución controlada del código, cada vez que se ejecuta una instrucción ésta se marca con un código de color en la ventana de edición y en la de desensamblado. Aparece una **marca de color verde** a la izquierda de la línea; en caso contrario la marca es la gris.



Igualmente, en la línea con la instrucción a ejecutar a continuación (es decir, a la que apunta el contador de programa) aparece una marca en forma de flecha amarilla.

Para ***Ejecutar Hasta*** antes hay que marcar la instrucción a la que se quiere llegar en la ejecución. Una manera de hacerlo es pinchándola con el ratón. Esto hace que se marque la línea en amarillo. Entonces o se teclea **CTRL+F10** o con el botón derecho del ratón se despliega la ventana de opciones y se selecciona ***Run to Cursor line***.



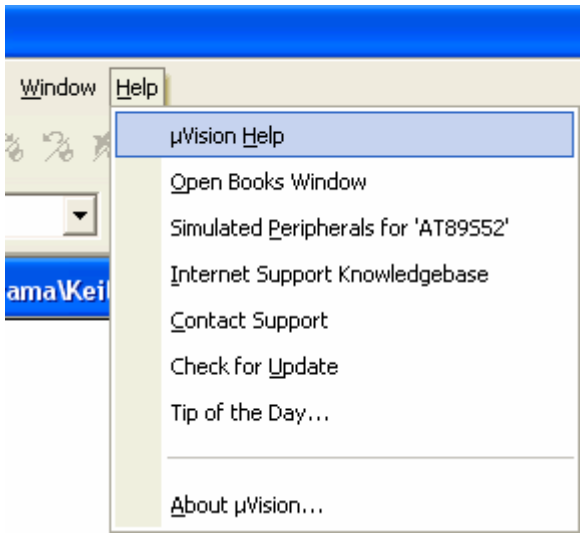
6º) FINALIZACIÓN DEL TRABAJO

Al finalizar la depuración, seleccionar ***Start/Stop Debug Session*** (en este caso, se cierra la sesión de depuración).
Al finalizar una sesión de trabajo con un proyecto, seleccionar ***Project→Close Project***.

Cuando más adelante se vuelva a abrir el proyecto, el entorno lo hará en las mismas condiciones en que se cerró.

7º) AYUDA EN LÍNEA

En la barra de menú existe una ayuda en línea que puede consultarse para profundizar o ver con detalle los aspectos de manejo de Keil uVision3.



COMENTARIO FINAL

En este prontuario sólo se han presentado las funciones básicas para poder afrontar la depuración de código ensamblador para los microcontroladores de la familia MCS51. En las figuras de los ejemplos comentados se pueden observar una serie de opciones que ni se han mencionado. Esto ha sido por dos motivos: o porque sólo resultan útiles en casos muy particulares y excepcionales (normalmente en el desarrollo profesional de grandes aplicaciones) o porque su utilidad es secundaria y muy intuitiva de captar a poco que se interactúe con el entorno Keil µVision y se tengan unos sólidos fundamentos de la filosofía de trabajo con herramientas que se ejecutan bajo el sistema operativo Windows.
Por otro lado, algunas de las funciones comentadas pueden realizarse –además de por el o los métodos indicados– por otros medios alternativos no explicitados. Queda al buen juicio del lector el curiosear por su cuenta con la herramienta y en la ayuda en línea para advertir esto.

Con lo aquí esbozado se tiene más que suficiente para hacer uso de los recursos que en el 99% de los casos se necesitarán en un trabajo de pequeña o mediana escala. Y, por supuesto, en el 100% de un trabajo estudiantil.

Caso de encontrarse algún tipo de error en este prontuario, sea de forma o de fondo, se agradecerá que ello se ponga en conocimiento del autor para proceder a corregirlo en futuras versiones de este documento. Las comunicaciones pueden dirigirse a la siguiente cuenta de correo electrónico: a.caparros.aceyte@gmail.com

NOTAS: