PRONTUARIO DEL ENTORNO DE DESARROLLO KEIL µVision® 4 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)

Referencia del documento: P_KuV4_v1.0 Fecha: 11 de marzo de 2010	Sustituye a: P_KuV4_v0.0 Fecha:8 de marzo de 2010	Sustituido por: Fecha:

© 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

11 de marzo de 2010
Córdoba, España

ÍNDICE

PREFACIO

En este prontuario sólo se van a presentar las funciones básicas para poder afrontar el desarrollo y la depuración de código ensamblador para los microcontroladores de la familia MCS51.
En las figuras de los ejemplos comentados se podrá observar, frecuentemente, que en los menús aparece una serie de opciones de las que algunas ni se mencionan. Esto ha sido así por dos motivos: o bien porque sólo resultan útiles en casos muy particulares y excepcionales (normalmente en el desarrollo profesional de grandes aplicaciones, utilizándose en la programación el lenguaje C o mixto C y ensamblador) o bien porque su utilidad es secundaria y muy intuitiva de captar a poco que se interactúe con el entorno de desarrollo *µVision 4*© de *Keil*© 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– también por otros medios no explicitados. Queda a criterio del lector el curiosear por su cuenta en la ayuda en línea para advertir esto.

A pesar del carácter de prontuario de este documento, que por ello resulta forzosamente sintético, 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.

Si el lector estuviese manejando la versión electrónica de este documento, entonces aquellas imágenes cuyo texto no se aprecie bien y se desee verlo con detalle pueden aumentarse de tamaño actuando en el visor (por ejemplo, *Acrobat Reader*©) con la opción o herramienta de ampliar. La resolución de las imágenes es la misma que se tendría si se observasen en pantalla al ejecutar la aplicación *µVision 4*©.

Para finalizar, y bajo la premisa latina de que *errare humanum est*, es posible que a pesar de la atención prestada durante la revisión del documento pueda el lector encontrar algún tipo de error en este prontuario, sea de forma o de fondo. Si fuese éste el caso se agradecerá se ponga en conocimiento del autor para proceder a corregirlo en futuras versiones del documento. Las comunicaciones pueden dirigirse a la siguiente cuenta de correo electrónico:

a.caparros.aceyte@gmail.com

Igualmente son bienvenidas las sugerencias y comentarios que se estimen oportunos.

Córdoba, marzo de 2010

0. INTRODUCCIÓN

El entorno de desarrollo *µVision*© de *Keil*© es una herramienta de carácter profesional de enorme calidad que –junto con alguna que otra más– se ha convertido en un estándar de facto para el desarrollo de aplicaciones basadas en la familia de microcontroladores MCS51.
Aunque la filosofía de esta herramienta se ha venido manteniendo de versión a versión, en lo concerniente a la interfaz con el usuario *µVision 4*© presenta algunos rasgos que la distinguen con respecto a su precedente versión, *µVision 3*©. Por este motivo se comienza este prontuario comentando las peculiaridades del nuevo aspecto de la interfaz gráfica de usuario, así como la filosofía de trabajo y configuración de las diversas ventanas que componen el entorno.

El entorno *µVision 4* © de *Keil* © *pre*senta dos modos básicos de trabajo:

- **Modo *principal* o *de proyecto***
- **Modo *de depuración***

En el modo de proyecto la interfaz del entorno se presenta adecuada para la gestión de un proyecto de programación, con lo que conlleva de definición de sus características, tipo de procesador a utilizar, partes en las que se decide subdividir su desarrollo, definición de las técnicas de optimización en la compilación si se programa en lenguaje C, gestión de los diversos módulos de código fuente a la hora de construir la aplicación final, tipo de archivos o ficheros a generar al ensamblar (o compilar) y montar, documentación asociada al proyecto, etcétera, por sólo mencionar algunas de las cuestiones involucradas en la gestión de un proyecto.

Una vez finalizada la creación de un proyecto y ya disponible la aplicación final, resulta imprescindible realizar la depuración funcional del código desarrollado. Para ello *µVision 4* ofrece el modo de depuración, mediante el que se puede la llevar a cabo a través de la ejecución real del código volcándolo previamente en alguna placa de desarrollo o en la final que soportará en la realidad el código. *µVision 4* permite el enlace del entorno con este tipo de placas, bien sean también de *Keil*, bien de terceras partes o propias.
También es posible realizar la depuración sin un soporte físico que ejecute realmente el código. Para ello *µVision 4* integra un excelente simulador con el que se puede simular el funcionamiento del procesador incluyendo todos los periféricos que éste pueda incorporar.

0.1 Componentes del sistema de µVision

El entorno *µVision 4* engloba una serie de componentes mediante los que se pueden llevar a cabo los procesos inherentes al desarrollo y depuración de aplicaciones. En la figura siguiente se muestra un esquema de cómo se interrelacionan éstos. Los componentes son los siguientes:

- **El EDI de µVision**.
  El entorno de desarrollo integrado (EDI, o *IDE* en inglés) combina, entre otras características:
  - La gestión de proyectos
  - Un editor específicamente concebido para este tipo de aplicaciones y dotado con corrección interactiva de errores
  - Ajuste de opciones
  - Utilidades de construcción de la aplicación
  - Ayuda en línea.

- **C51 (Compilador) y A51 (Macroensamblador)**.
  Los archivos o ficheros fuente creados con el EDI de *µVision* se pasan al compilador C51 o al macroensamblador A51. Estos componentes los procesan y generan los ficheros objeto reubicables que serán procesados por el módulo montador.
  El compilador C51 de Keil es totalmente conforme con la especificación ANSI del lenguaje de programación C. Además, contempla características adicionales para la arquitectura MCS51.
  El macroensamblador admite el repertorio común completo del 8051 así como el de todos sus derivados, algunos de los cuales poseen un repertorio ampliado. Además, como tal macroensamblador, admite el uso de macros en la escritura del código.

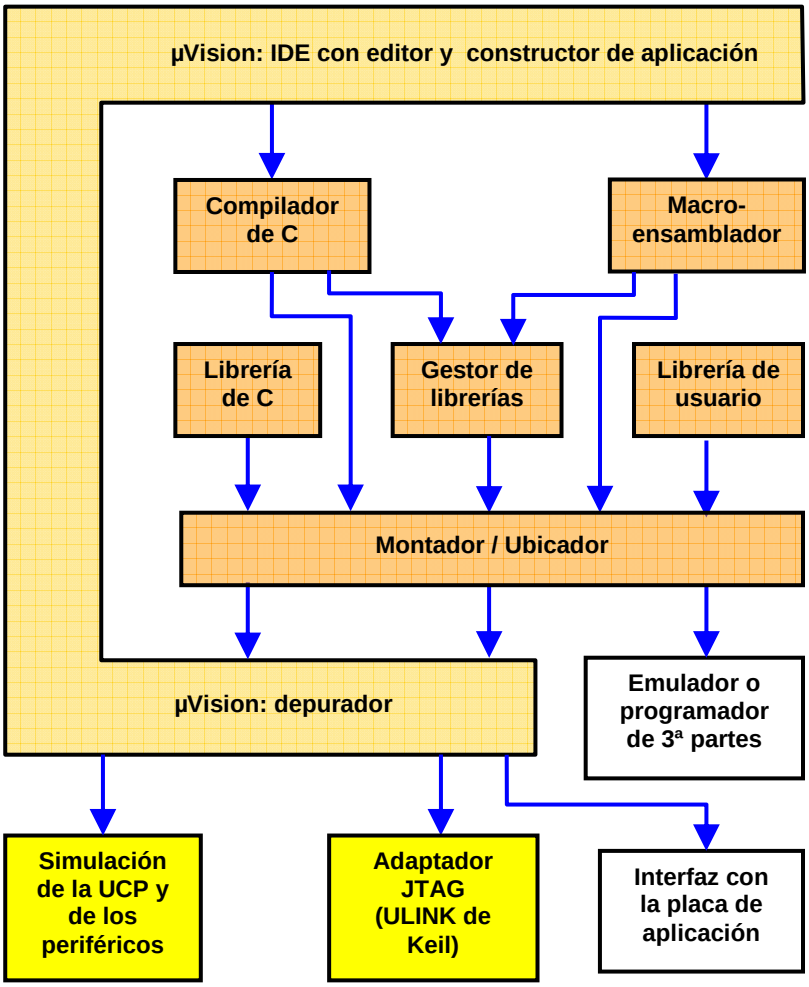


Fig. 0.1: Componentes del sistema *µVision* de *Keil*

- **LIB51: Gestor de librerías**.
  Este componente permite al usuario crear librerías de módulos objeto generados por el compilador y el macroensamblador. Las librerías son una colección de módulos objeto especialmente formateada, que constituye un conjunto de programas o subrutinas llamables desde el proyecto del usuario y que pueden ser tomadas selectivamente por el montador en el momento de generar la aplicación final. La ventaja de disponer de una colección de librerías de funciones es que el programador no tiene que diseñar por su cuenta aquellas subrutinas o funciones que se encuentren en ellas. El paquete de software *µVision* incluye una serie de librerías que ofrecen, por ejemplo, funciones de tipo matemático con aritmética de coma fija, de emulación de coma flotante, etcétera. A estas librerías el usuario puede añadirles sus propias funciones diseñadas ex profeso o crear otras librerías nuevas con el gestor de librerías que ofrece el entorno de desarrollo.

- **BL51: Montador / Ubicador**

  La misión de este componente es crear un archivo o fichero absoluto tipo ELF/DWARF a partir de los creados por el compilador o el macroensamblador así como, si es el caso, de los extraídos de las librerías por ser referidos por el programador al crear su código de aplicación. Un fichero objeto absoluto se caracteriza por no poseer código reubicable, sino que ha de ubicarse en posiciones concretas de la memoria. Este archivo absoluto generado por el montador se usa para alguna de las siguientes cuestiones:

  - Programar la ROM tipo FLASH de un microcontrolador, u otros tipos de memorias. Se usa el fichero en formato HEX generado por el montador.
  - Simulación o depuración en placa, mediante *µVision* en modo depuración.
  - Verificación real del programa mediante un emulador en-circuito.

  Todos los entornos de desarrollo incluyen una utilidad que suele denominarse ***Make*** y que permite generar de manera muy cómoda la aplicación final incluso cuando el desarrollo esté constituido por innumerables módulos de código fuente. Como se verá más adelante, esta utilidad hace un uso automático de los módulos compilador, ensamblador y montador selectivamente sobre los ficheros fuente que lo precisen. En el entorno *µVision* esta utilidad *make* recibe el nombre de ***Build*** (*construir).*

- **Depurador *µVision***

  Se trata de un depurador simbólico al nivel del código fuente, es decir, que el seguimiento de la ejecución y la interactuación se pueden hacer sobre el propio código fuente. Esto lo hace ideal para una depuración rápida, inteligible y fiable del código desarrollado. Incluye un veloz simulador capaz de simular un sistema completo basado en el 8051 o cualquier otro derivado de la familia, así como todos los periféricos existentes (previa selección del dispositivo en una lista de procesadores soportados) y hardware externo.
  También permite probar la aplicación desarrollada; entiéndase en la placa real sobre la que se ejecutará. Esto se puede realizar de distintas maneras:

  - Instalando el monitor MON51 en la placa y descargando el programa de aplicación usando la interfaz con MON51 integrada en *µVision*. MON51 forma parte del paquete *µVision*.

  - Usando la interfaz AGDI, para utilizar el depurador de *µVision* en la observación de los recursos del sistema real.

  En el caso de los derivados XC8xx de Infineon (antiguamente SIEMENS) y µPSD33xx/34xx de ST, el adaptador USB-JTAG ***ULINK*** de *µVision* permite usar la interfaz de depuración del entorno pero sólo para presentar los resultados de tal proceso e interactuar con él. La depuración propiamente dicha la lleva a cabo el sistema de depuración integrado en el propio microcontrolador. La interconexión entre *µVision* y la placa con el dispositivo se realiza mediante un aparato adaptador USB-JTAG. Dicho adaptador se conecta mediante un cable USB al computador en que se ejecuta *µVision*, y mediante una conexión JTAG a la placa con el microcontrolador.

1. ASPECTO DE LA APLICACIÓN

El aspecto que por defecto presenta *µVision 4* la primera vez que se entra en ella es el que se ve en la figura:

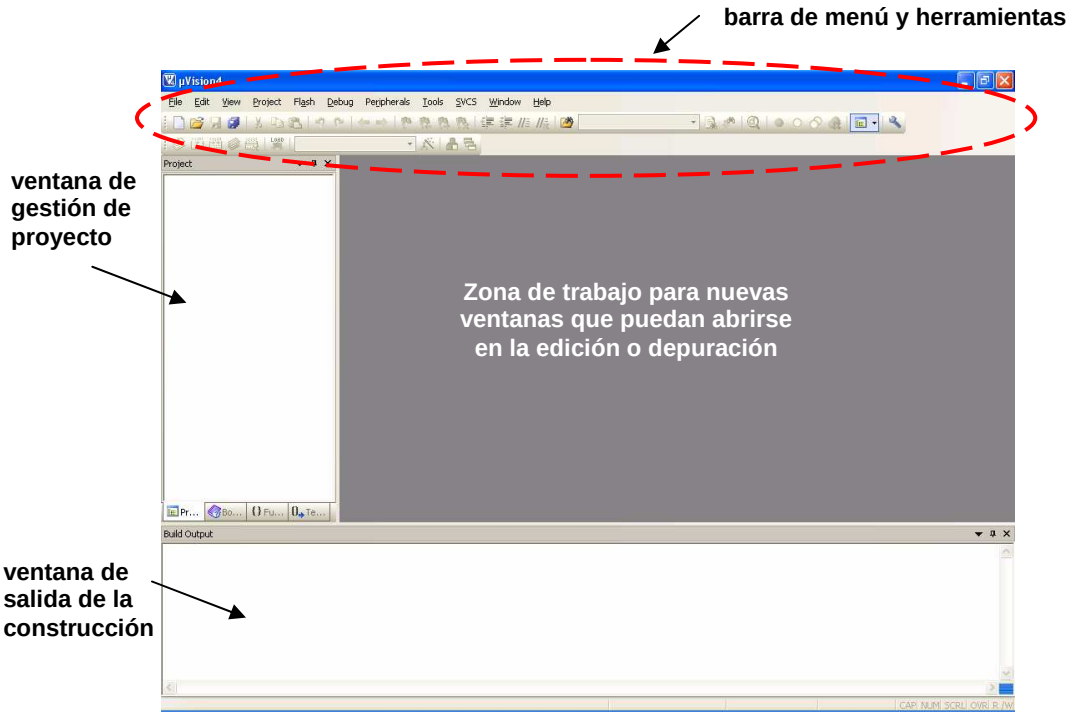


Fig. 1.1: Aspecto inicial de la aplicación

En la parte superior se encuentra la **barra de menú**, mediante la cual se puede acceder a todas las opciones de *µVision 4* de Keil, 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.

Fig. 1.2: la barra de menú y de herramientas

Es importante advertir que si ya se hubiese trabajado con algún proyecto y se hubiese modificado la apariencia y forma de las ventanas de la interfaz, entonces al entrar en *µVision 4* se mostrará la interfaz exactamente igual a como quedó al cerrase la aplicación la última vez en la que se trabajó con ella.

1.1. LA FILOSOFÍA DE LAS VENTANAS EN *KEIL µVision 4*

En el entorno de depuración *µVision 4* de *Keil* existen dos tipos de ventanas básicas:

- Las típicas del sistema operativo *Windows*©, manejables como tales.

- Las específicas de *µVision 4*, que aparecen o pueden aparecer dentro de una ventana *Windows*©. Por ejemplo, las dos mostradas en la primera figura: las de gestión del

proyecto y la salida de construcción, entre otras. En estas ventanas, en la esquina superior derecha, pueden aparecer los controles mostrados a continuación:

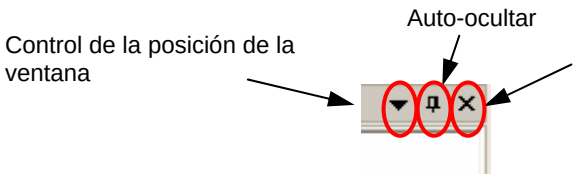


Fig. 1.3: Símbolos de control de una ventana

1.1.1. Control de la posición de la ventana

Al pinchar con el ratón en este símbolo se despliega el menú con los posibles tipos de posicionado de la ventana.

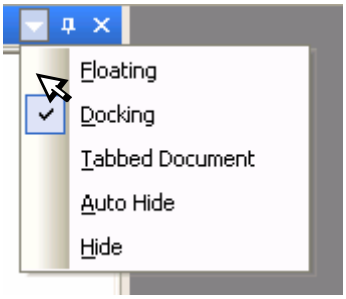


Fig. 1.4: Opciones de posición

- **Flotante** (*floating*): La ventana se independiza y aparece flotando. Con el ratón es posible redimensionarla, arrastrarla y ubicarla donde se desee, tal y como se haría con una de *Windows*. Una manera rápida de hacer una ventana flotante, partiendo de una situación acoplada, y viceversa, es con una doble pulsación izquierda del ratón en la barra con su descripción.

- **Acoplada** (*docking*): La ventana aparece acoplada solidariamente a otras dentro de la ventana *Windows* de *µVision4* en la que se encuentre. Es la opción por defecto.

- **Documento con pestaña** (*tabbed document*): La ventana aparece en la zona de trabajo, con una pestaña identificadora en la parte superior. Si más de una ventana se ha puesto de esta manera, las pestañas aparecen escalonadas, como en un archivador de fichas, y pinchando en cada pestaña se hace visible el documento (ventana) oportuno. Apuntando una pestaña y pulsando el botón derecho del ratón emergen las posibles opciones a realizar con dicha ventana.

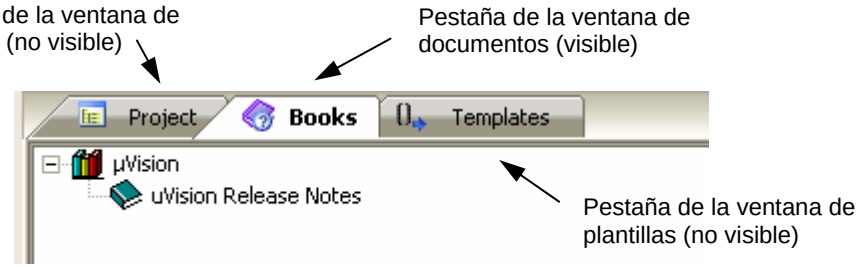


Fig. 1.5: Documentos con pestaña

Para volver a la organización por defecto es necesario hacer ***Windows→ Reset View to defaults*** en la barra de menú.

- **Auto-ocultable** (*Auto Hide*): Con esta opción la ventana se oculta de modo que aparece a la izquierda una pestaña vertical. Cuando el ratón se mueve y entra en esta pestaña (sin necesidad de pulsación alguna) entonces aparece la ventana, no desapareciendo hasta que el ratón sale de su área, en cuyo caso se colapsa.

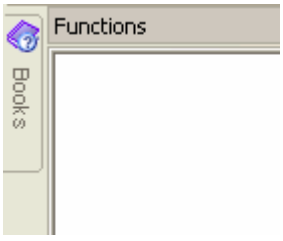


Fig. 1.6: Ventana auto-ocultable

Obsérvese en este ejemplo que la ventana de ***Documentos*** (*books*) se auto-oculta, mientras que la de *Funciones está visible (acoplada)*.
Si el ratón pasase por encima de la pestaña oculta, emergería la ventana tal y como puede verse en la figura que sigue.

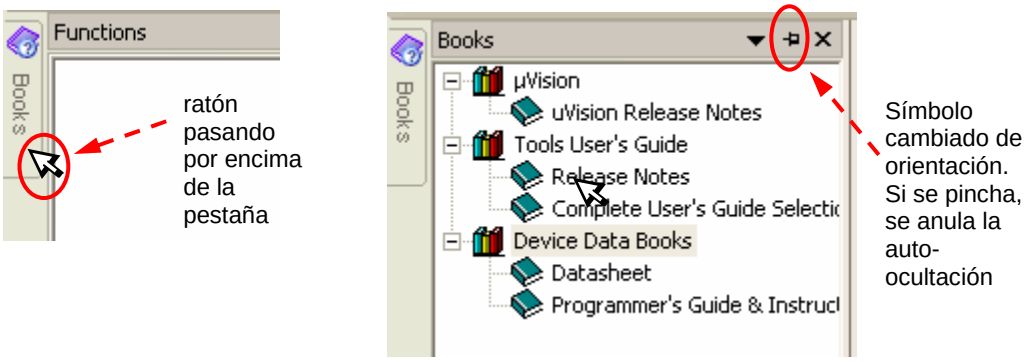


Fig. 1.7: a) Ventana auto-ocultada a punto de reaparecer b) Ventana reaparecida

Obsérvese cómo el ratón se hace pasar por encima de la pestaña. En ese momento emerge la ventana, con límite por la derecha el de la ventana *µVision 4* en la que se halle (la ocupa en su totalidad), y límite inferior el de la ventana de *Windows*© de la aplicación.
Al hacerse visible la ventana, como se aprecia en la figura superior, puede observarse que el icono ***Auto-ocultar*** se muestra con cambio de orientación (horizontal en lugar de vertical). Para colapsar y hacer desaparecer la ventana basta con ubicar el ratón fuera de sus límites.

- **Ocultar** (*Hide*): Cierra definitivamente la ventana. Por supuesto, puede volver a abrirse.

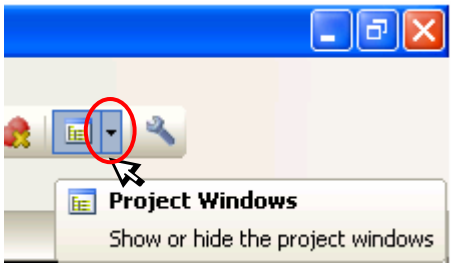
1.1.2. Auto-ocultar

El símbolo ***Auto-ocultar*** ofrece una vía rápida de conseguir esta característica para una ventana en lugar de tener que hacerlo vía símbolo de control de la ***Posición de la Ventana***.

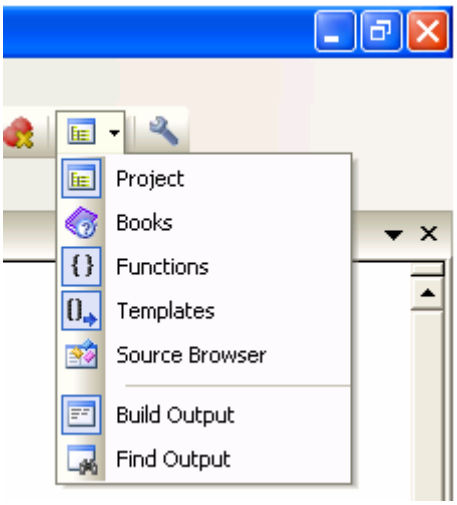
1.1.3. Cerrar

Se trata del típico símbolo para cerrar una ventana. Cuando se cierra una ventana se puede volver a abrirla (o cerrarla si estuviese abierta) de una de estas dos formas:

- Mediante la barra de menú, haciendo ***View → <Ventana_deseada>***.
- Mediante la barra de herramientas, con el icono de ***Ventana de Proyecto***.



a) Por barra de menú



a) Por barra de herramientas

Fig. 1.8: Apertura de la ventana de proyecto

Al pinchar en la zona del triángulo, se despliega un menú con las distintas ventanas y su estado (abierta o cerrada). Pinchando en una concreta se conmuta su estado.

1.2. REDISTRIBUCIÓN DE LA ZONA DE TRABAJO Y DEMÁS VENTANAS

En la zona de trabajo, ya se ha dicho que se pueden abrir diversas ventanas en el proceso de creación y trabajo con un proyecto, o en la fase de depuración funcional. Por defecto, en la zona de trabajo las ventanas aparecen en su forma de documentos con pestaña, ya comentada.

La zona de trabajo puede cambiarse de aspecto y de organización. Para ello es necesario actuar sobre alguna o algunas de las restantes ventanas, lo que origina un cambio conjunto de la distribución de **todas** las ventanas y zona de trabajo. Los procesos que se van a comentar se pueden aplicar sucesivamente con diferentes ventanas o pestañas una tras otra.
La manera de proceder es la siguiente:

- Primero, pínchese en la barra superior de la ventana que se desee reorganizar (o en la pestaña oportuna en el caso de una organización en pestañas y que sólo se desee reubicar esa pestaña en concreto y no todas las de la ventana). En el ejemplo, pestaña de ***Proyecto***.

- Manteniendo pulsado el botón izquierdo del ratón, arrástresela. En ese preciso momento aparecerán unos símbolos u operadores de organización, tal y como se aprecia en la figura.

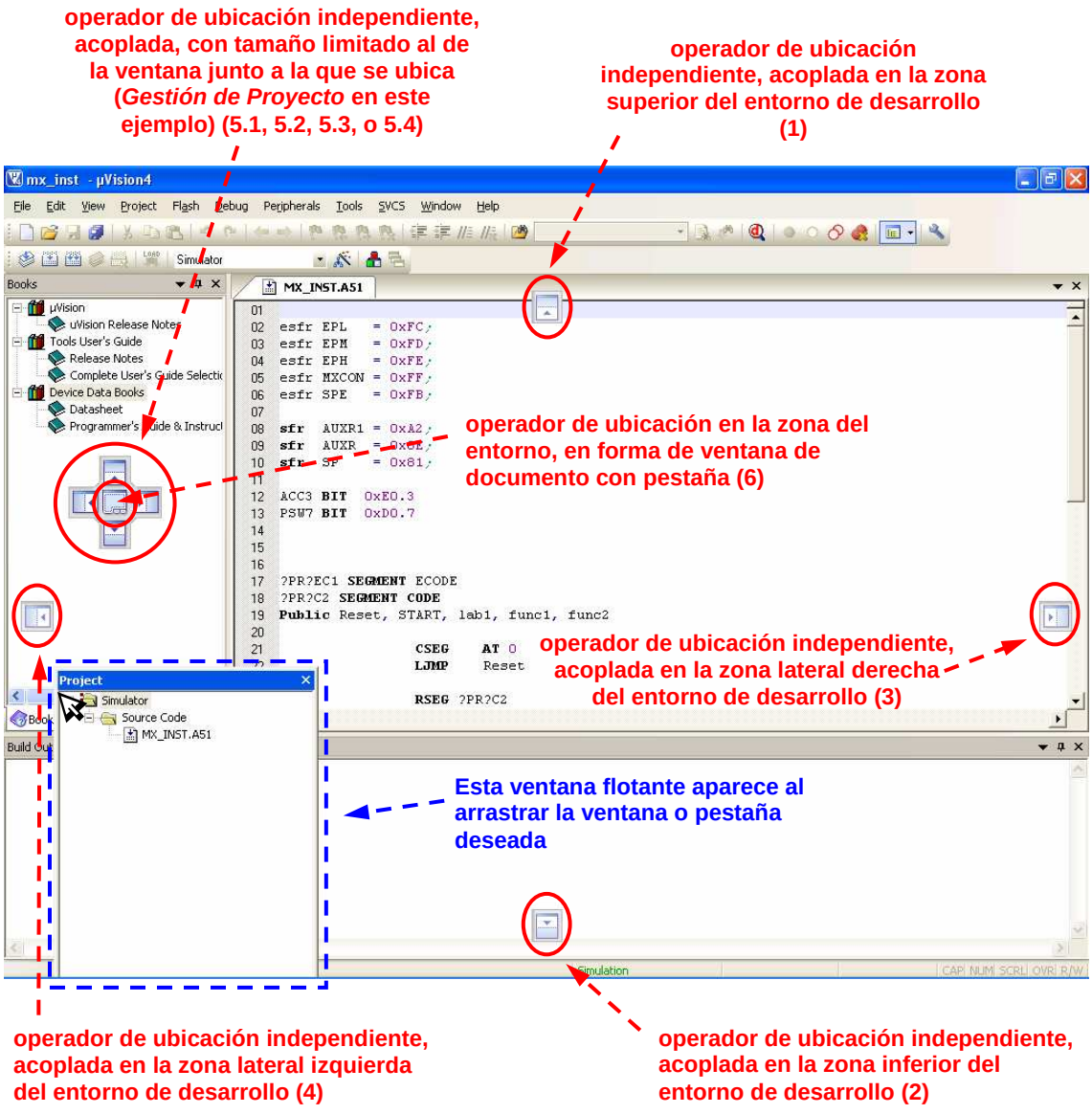


Fig. 1.9: Redistribución de ventanas mediante arrastre

- Dependiendo de cómo se quiera reorganizar el aspecto de la ventana o pestaña, se arrastrará hasta que el puntero del ratón se coloque sobre el símbolo de organización deseado. Existen las siguientes posibilidades:

  0. Dejarla flotante; para ello se deberá soltar la ventana sin que se haga sobre ninguno de los símbolos de organización.

  1. Ubicarla acoplada en la parte superior de la ventana general de *Windows*.

  2. Ubicarla acoplada en la parte inferior de la ventana general de *Windows*.

  3. Ubicarla acoplada en la parte lateral derecha de la ventana general de *Windows*.

  4. Ubicarla acoplada en la parte lateral izquierda de la ventana general de *Windows*.

  5.1 Ubicarla acoplada encima de la ventana local del entorno *µVision*.

  5.2 Ubicarla acoplada debajo de la ventana local del entorno *µVision*.

  5.3 Ubicarla acoplada a la derecha de la ventana local del entorno *µVision*.

  5.4 Ubicarla acoplada a la izquierda de la ventana local del entorno *µVision*.

  6. Ubicarla en una zona del entorno, en forma de ventana de documento con pestaña.

En los casos 1, 2, 3 y 4 la ventana se extiende todo el ancho de la ventana *Windows*, mientras que en los casos 5.n la ventana adopta un tamaño ajustado al de esa zona o ventana local.

- Cada vez que el puntero del ratón se coloca sobre uno de esos símbolos, se sombrea en la pantalla la zona donde se ubicará la ventana que se quiere reubicar y reorganizar. Esto puede verse en la figura siguiente:

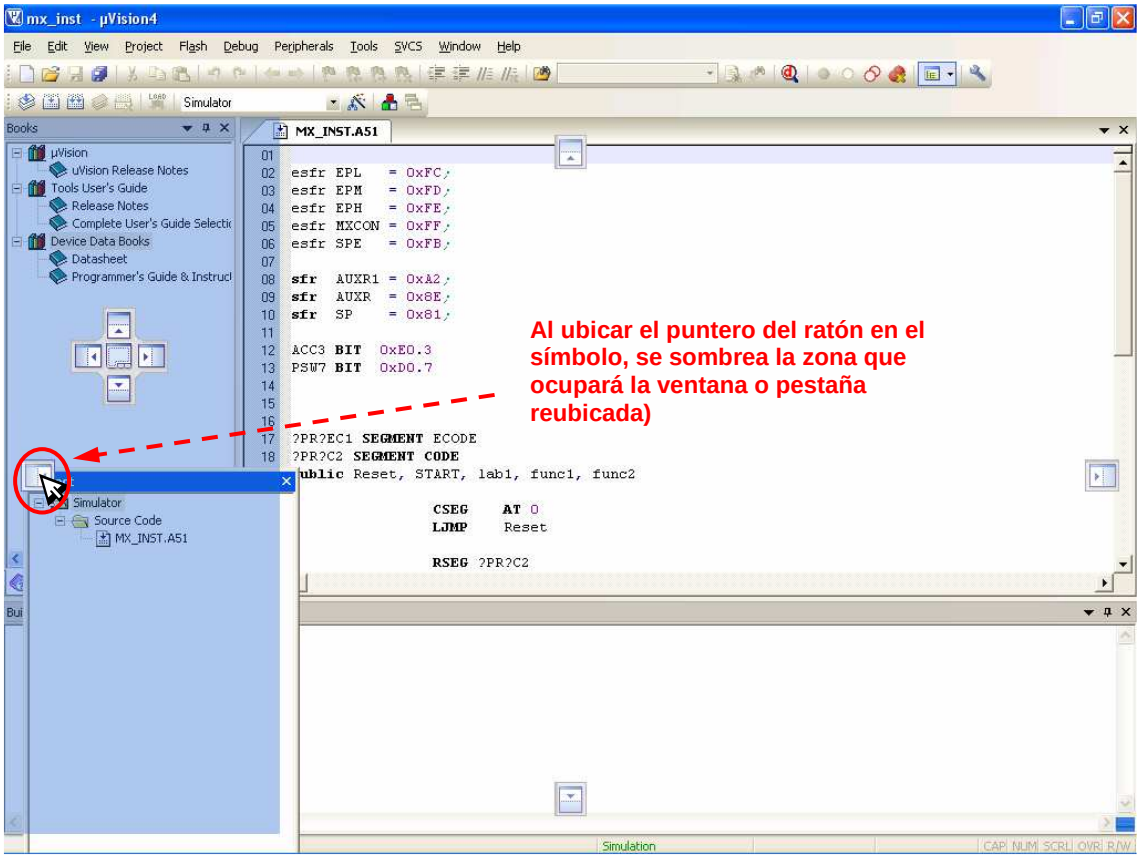


Fig. 1.10: Efecto del arrastre de una pestaña, ventana o zona del entorno: marcado de zona de ubicación

Si en el caso del ejemplo de la figura anterior se soltase el ratón, el efecto sería el mostrado en la figura que sigue:

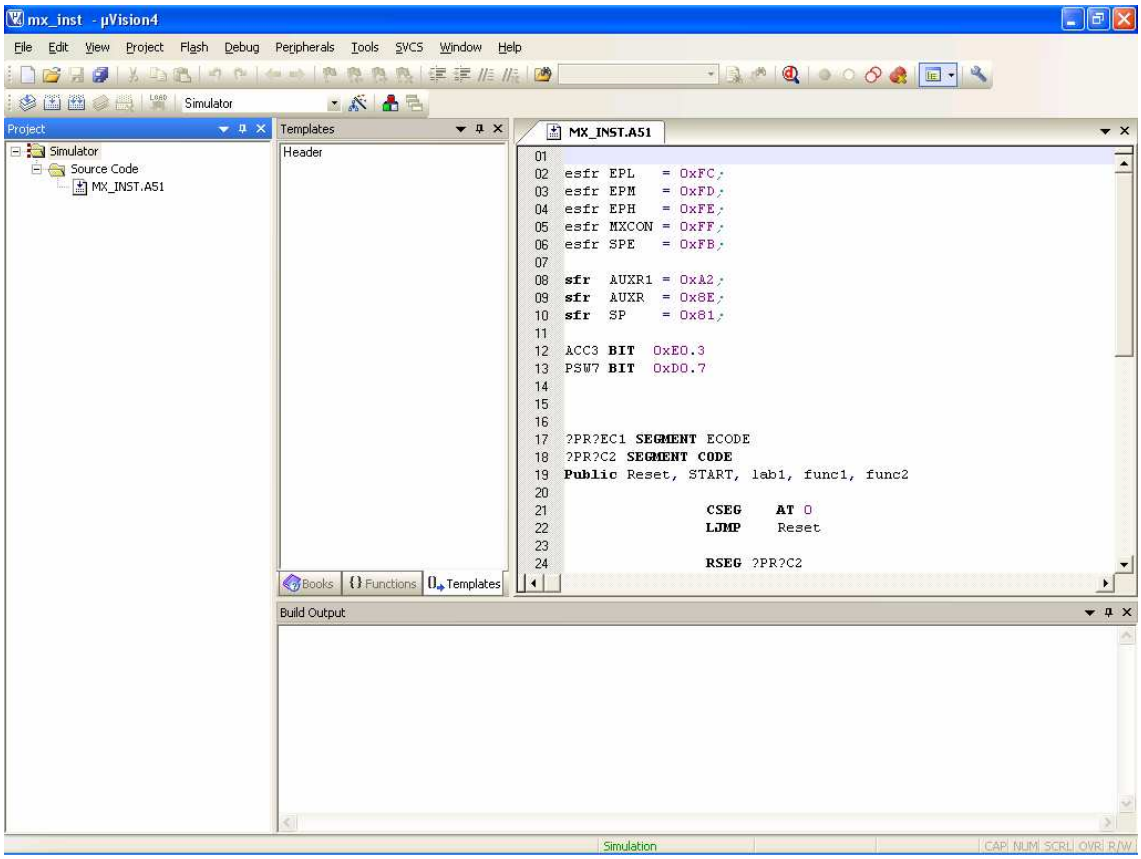


Fig. 1.11: Resultado de reubicación a la izquierda del entorno

A continuación, a modo de resumen visual, se presentan algunas otras posibilidades.

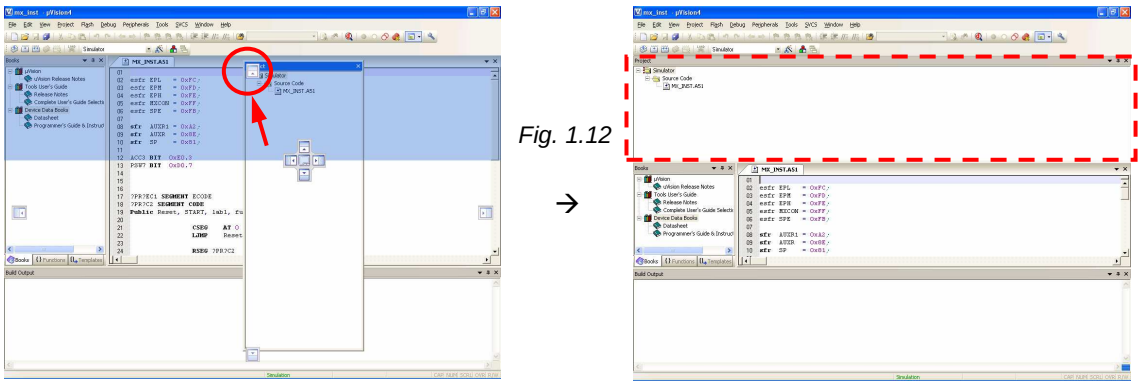


Fig. 1.12 →

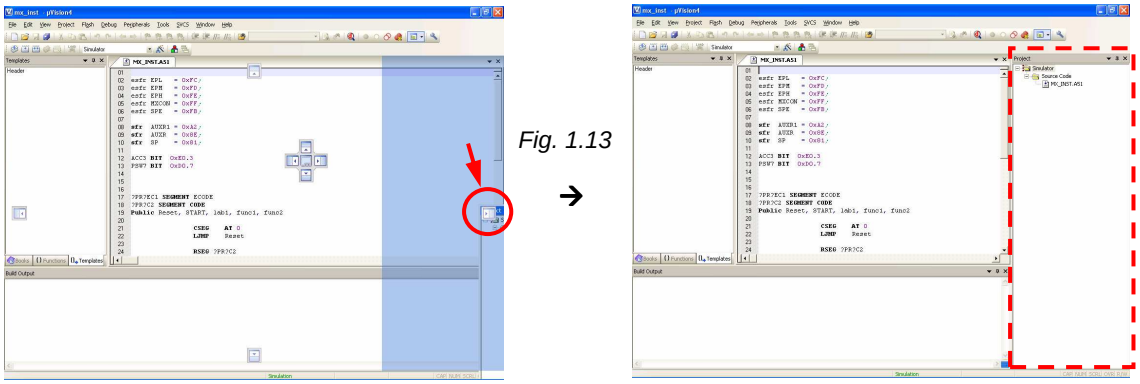


Fig. 1.13 →

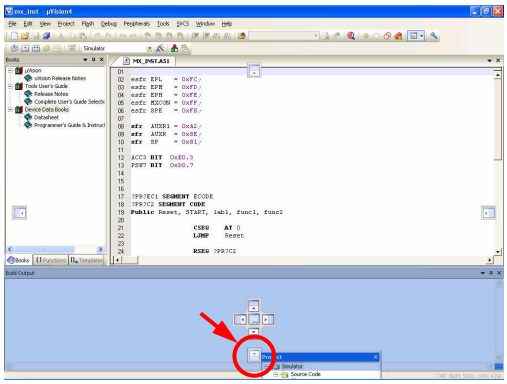
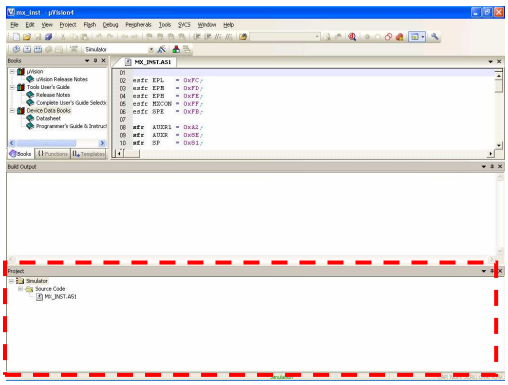


Fig. 1.14



Ejemplos de ubicación junto a la ventana de gestión del proyecto. Primero a su izquierda…

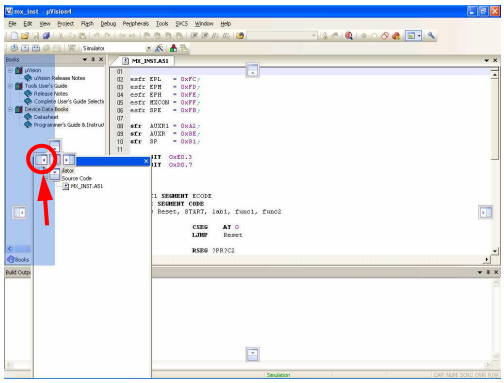
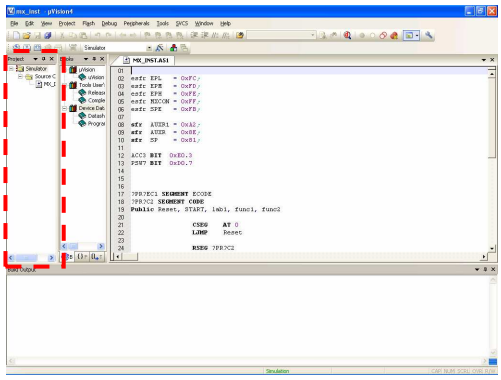


Fig. 1.15



… y luego arriba de ella:

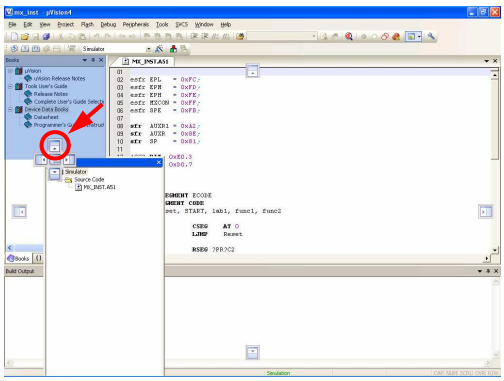
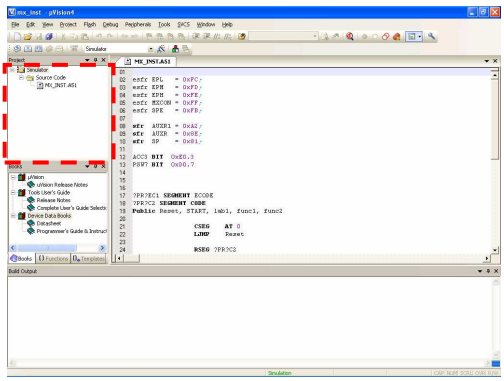


Fig. 1.16



Ahora, un ejemplo de ubicación junto a la zona de trabajo, a su izquierda:

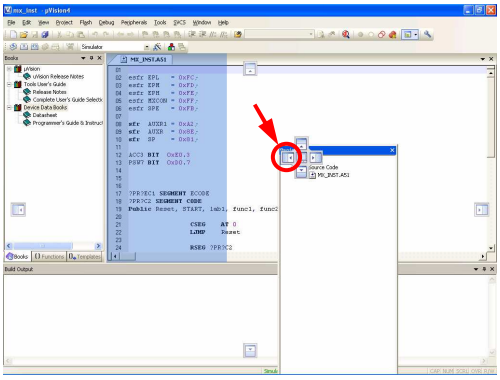
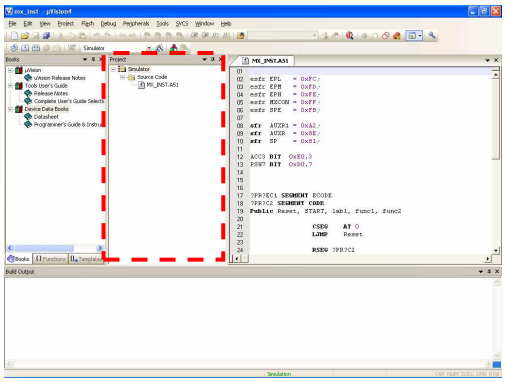


Fig. 1.17



Si lo que se desea es ubicar la ventana en la zona de trabajo, con forma de documento con pestaña, hay que arrastrar la ventana o pestaña hasta algún punto de la zona de trabajo. Al aparecer los símbolos de ubicación, apúntese con el puntero al centro de la cruz y suéltese. Automáticamente aparecerá la pestaña en la zona de trabajo. Téngase en cuenta que si lo que se arrastra es una ventana compuesta por varias pestañas entonces se trasladarán todas las pestañas.
Lo dicho anteriormente se ilustra en los siguientes ejemplos.

Ejemplo de traslado de la pestaña de proyecto a la zona de trabajo:

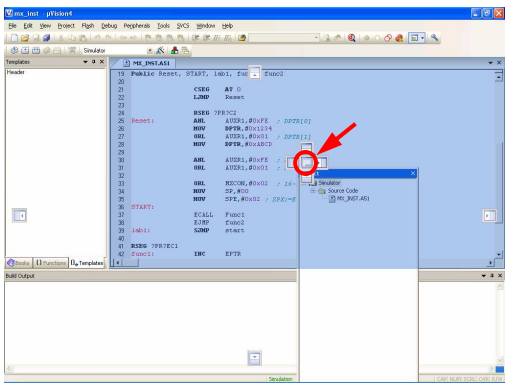
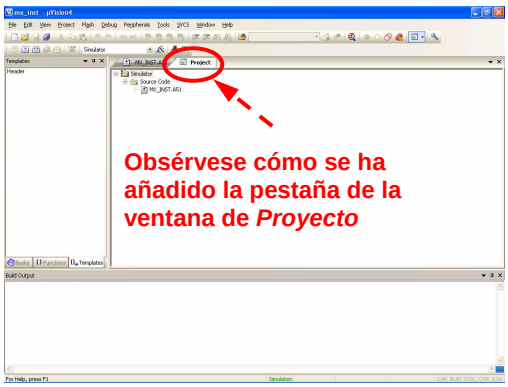


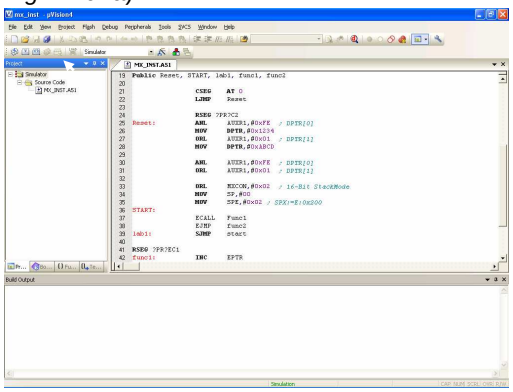
Fig. 1.18

→

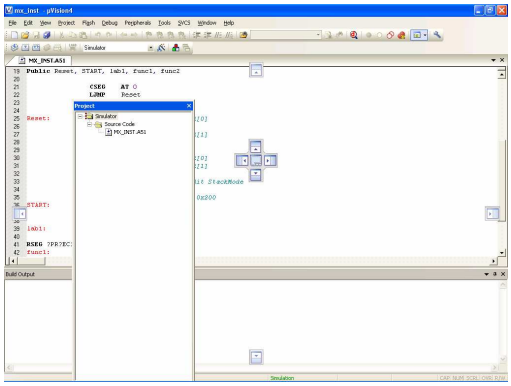


Ejemplo de traslado de todas las ventanas (pestañas) en la zona de gestión del proyecto a la zona de trabajo: Primero (*fig 1.19.a*) se arrastra desde la parte superior de la ventana, luego (*fig.1.19.b*) se mueve hasta algún punto de la zona de trabajo:

Fig. 1.19: a)

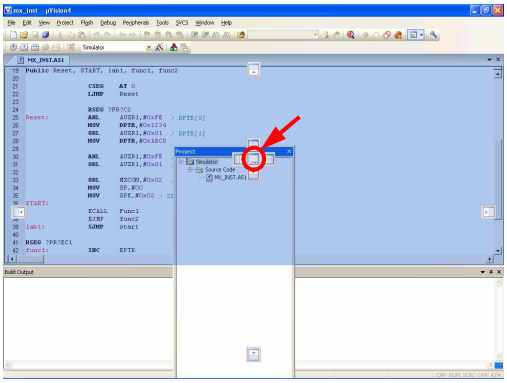


b)

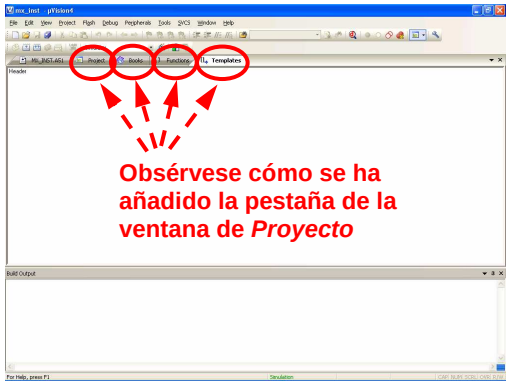


Como se ve en *fig.1.19.b*, desaparece la ventana de su ubicación original y se expande la zona de trabajo. Al aparecer los símbolos de ubicación, se sigue arrastrando hasta apuntar (*fig.1.19.c*) con el puntero del ratón el símbolo del centro de la cruz que representará, en este caso, la zona de trabajo. Entonces (*fig.1.19.d*) se suelta el ratón y automáticamente se ubican las ventanas en forma de documentos con pestañas:

c)



d)



2. CREACIÓN DE UN PROYECTO

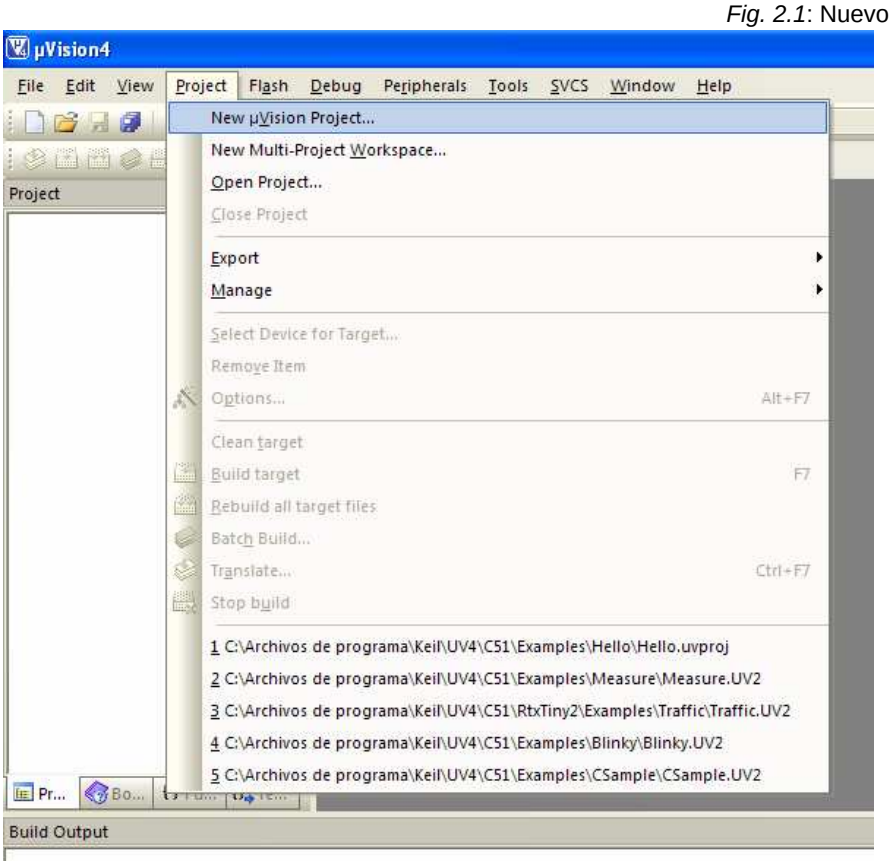
Para el trabajo con *µVision 4* de Keil, antes de hacer nada hay que crear un proyecto.

2.1. Creación de un proyecto.

Si no estuviese ya creado, para crearlo hágase lo siguiente:

Project→New µVision Project

Fig. 2.1: Nuevo proyecto



Si ya se hubiese creado el proyecto anteriormente, entonces habría que abrir el proyecto. Esto se puede hacer mediante la barra de menú vía ***Project→Open Project***. También, como alternativa, se puede abrir un proyecto ya existente seleccionándolo de entre los que aparezcan relacionados en la parte inferior del menú desplegable que se abre al seleccionar la opción ***Project*** en la barra de menú. Esto se puede advertir en la figura de arriba. Los proyectos que aparecen relacionados son los que se hayan abierto últimamente; es decir, los proyectos más recientes con los que se haya trabajado.

En la figura precedente también puede observarse que ciertas opciones no aparecen elegibles (están en gris claro) debido a que en este caso no existe ningún proyecto abierto. Si lo estuviese ya uno, entonces estas opciones serían aplicables a dicho proyecto. Por ejemplo, las funciones de *cerrar proyecto (Close Project)*, *seleccionar dispositivo (Select Device for Target)*, *suprimir elemento (Remove Item)*, *construir aplicación (Build target)*, etcétera.

Cuando se crea un proyecto nuevo, en la ventana que se abre désele nombre al proyecto y ubíqueselo en la carpeta que se desee (si no existiese, créese). Esta ventana es una típica ventana de *Windows* de apariencia y funcionalidad similar a la que aparece cuando con cualquier aplicación se va a abrir o guardar un fichero. A través de ella se puede navegar por la estructura de directorios o carpetas de las distintas unidades de almacenamiento que puedan existir, así como incluso crear nuevas carpetas o realizar las demás funciones posibles de trabajo con archivos o ficheros. Si se tienen conocimientos sólidos de la filosofía de trabajo con un sistema operativo basado en ventanas, como por ejemplo *Windows*, se sabrá a qué nos estamos refiriendo. En la figura que sigue puede observarse lo dicho:

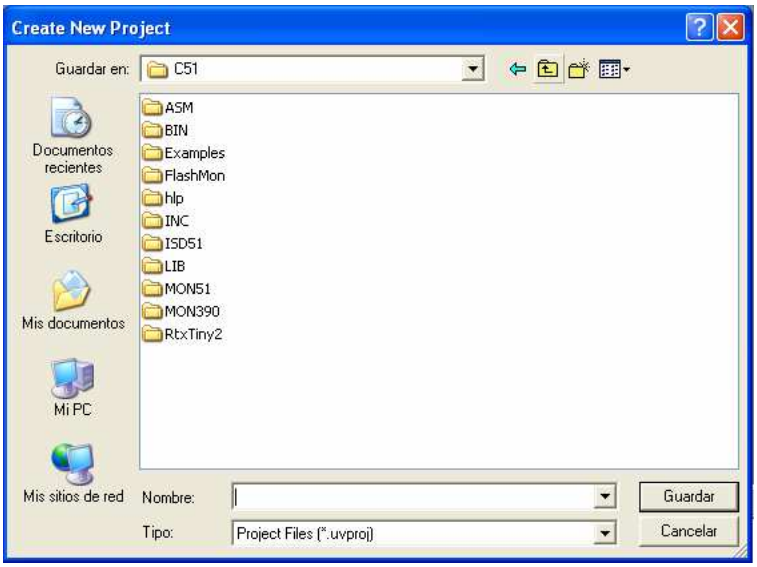


Fig. 2.2: Ventana para dar nombre y ubicar el proyecto

A continuación, se abrirá otra ventana en la que hay que seleccionar el procesador que se vaya a utilizar. Para ello se selecciona el fabricante pinchando en su nombre o en el símbolo **+** de la relación, ante lo cual se expande y aparecen los diferentes dispositivos existentes para dicho fabricante, tal y como puede observarse en la siguiente figura:

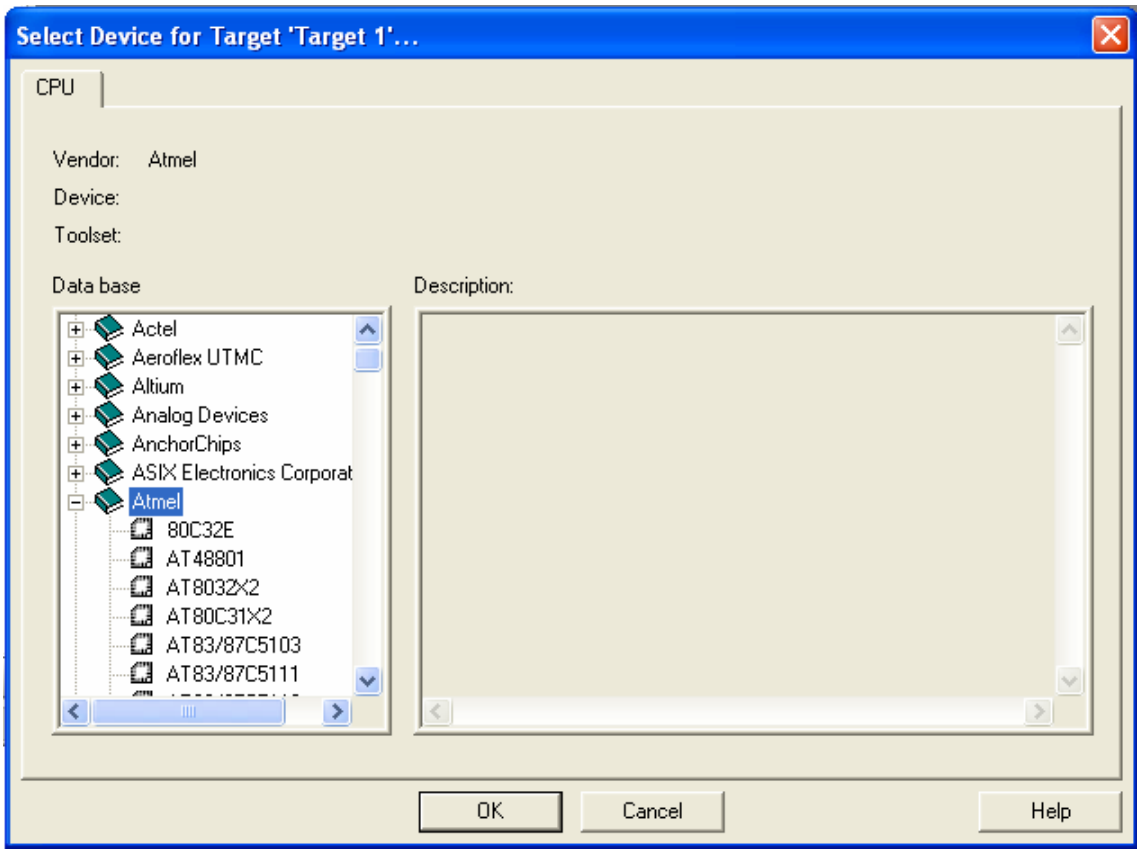


Fig. 2.3: Ventana para seleccionar el fabricante del dispositivo

De entre las diferentes referencias de microcontroladores, se busca y elige el deseado. En nuestro caso será la CPU de ATMEL, AT89S52, pero bien pudiera ser cualquier otro.

Esto puede verse en la siguiente figura:

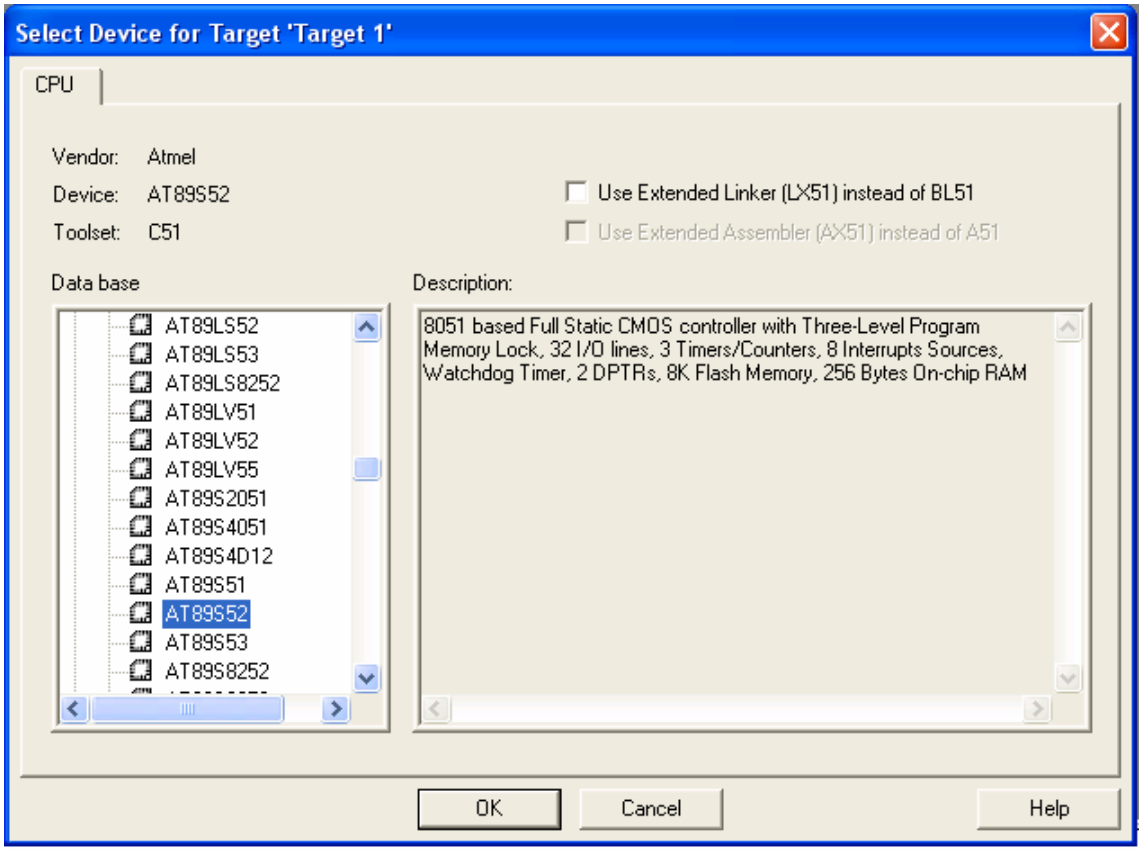


Fig. 2.4: Ventana para seleccionar el modelo de microcontrolador

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

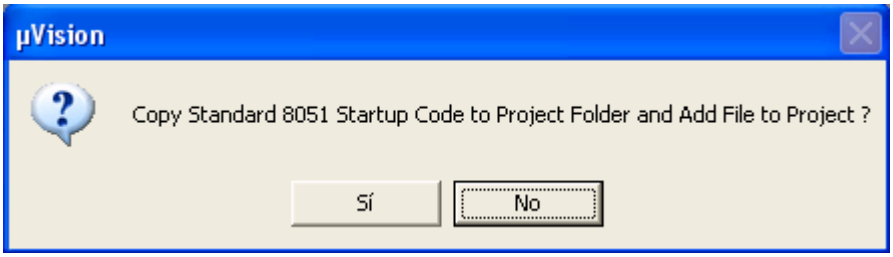


Fig. 2.5: Ventana para incluir o no una plantilla al proyecto

2.2. Apertura de un proyecto.

Una vez que se haya creado un proyecto, cada vez que se vaya a trabajar se empezará abriendo el proyecto, caso de no estarlo ya por defecto, haciendo en la barra de menú:

Project→Open Project

En la ventana que se abre (una típica de *Windows* para trabajar con ficheros), 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, así como otros documentos asociados.

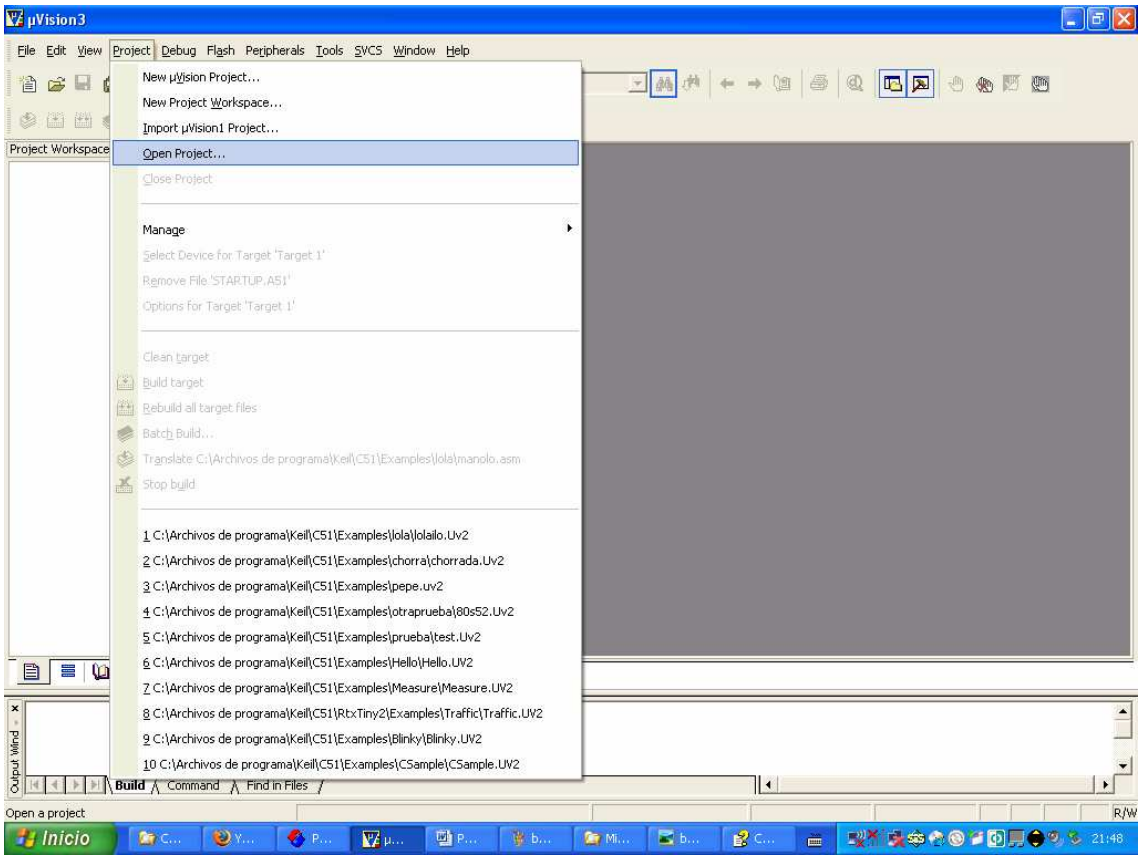


Fig. 2.6: Ventana con las opciones de proyecto

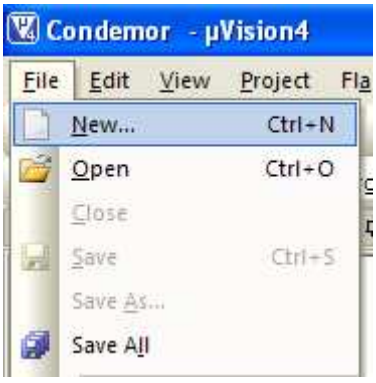
3 EDICIÓN DEL CÓDIGO FUENTE

3.1. Creación y guardado de un fichero fuente

En nuestro caso, sólo se crearán proyectos con un único fichero fuente.

Lo primero, si no se ha creado ninguno, será asociar uno al proyecto (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 o ficheros, repartiéndose el código fuente total entre ellos en lugar de meter todo el código en un gran y único fichero). Como extensión del fichero, utilícese preferiblemente ***ASM*** (por *assembler*) o ***A51*** (por *assembler 8051*), o incluso **S**, **SRC** (por *source* en ambos casos) *o* **A** (por *assembler*).



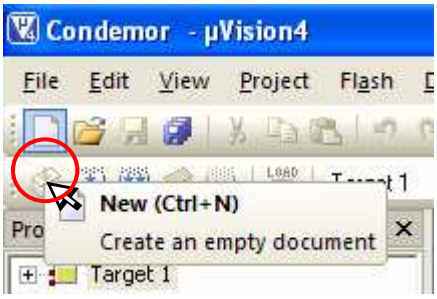
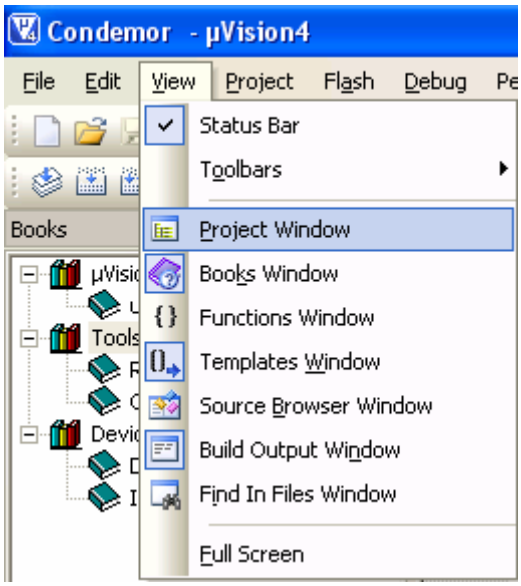


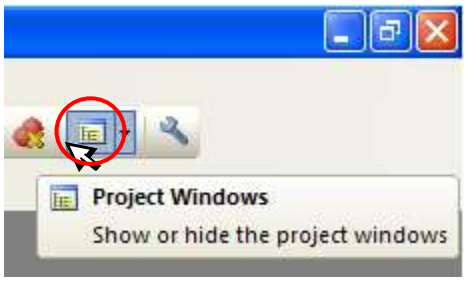
Fig 3.1: a) Creación de fichero vía barra de menú b) ídem vía barra de herramientas

3.2. Asociación de un fichero a un proyecto

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*.



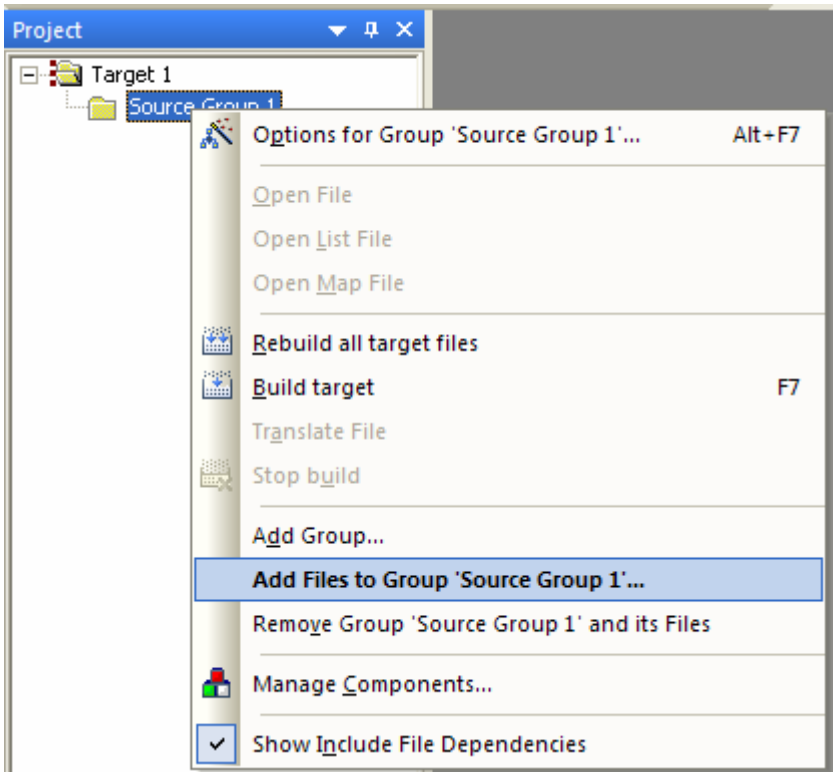
a) Apertura de la ventana de proyecto en barra de menú



b) Apertura de ventana de proyecto mediante la barra herramientas

c) Resultado de la apertura de la ventana del proyecto

Fig. 3.2: Modos de abrir un proyecto

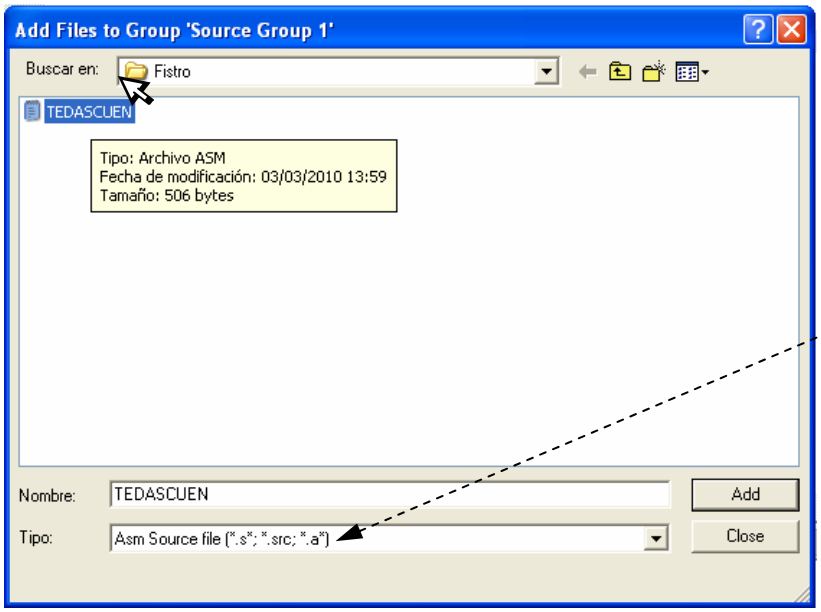


Una vez hecho lo anterior, pínchese en ***Source Group1*** y con el botón derecho del ratón se desplegará un submenú.

Selecciónese entonces la opción ***Add Files to Group…***

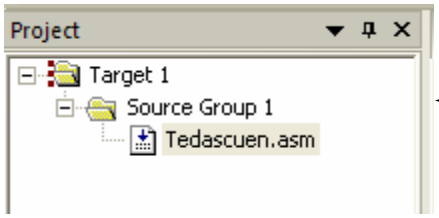
Fig. 3.3: Asociación de un fichero a un proyecto

En la ventana que se despliega, búsquese selecciónese el fichero deseado.



(si se programa en ensamblador, asegúrese de que la plantilla de búsqueda de ficheros es ***ASM source file*** o bien ***All files***).

Fig. 3.4: Asociación de un fichero a un proyecto



El fichero seleccionado queda, así, añadido al proyecto

Fig. 3.5: Resultado de la asociación

Para asociar un fichero a un proyecto no es necesario que esté escrito del todo; estando vacío también es posible asociarlo.

3.3. Apertura y edición de un módulo de código fuente

Una vez creado un proyecto y asociado(s) un(os) fichero(s), también 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 selección de ***Open*** en el menú desplegable). Otra manera es actuando sobre el icono de apertura de fichero, en 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).

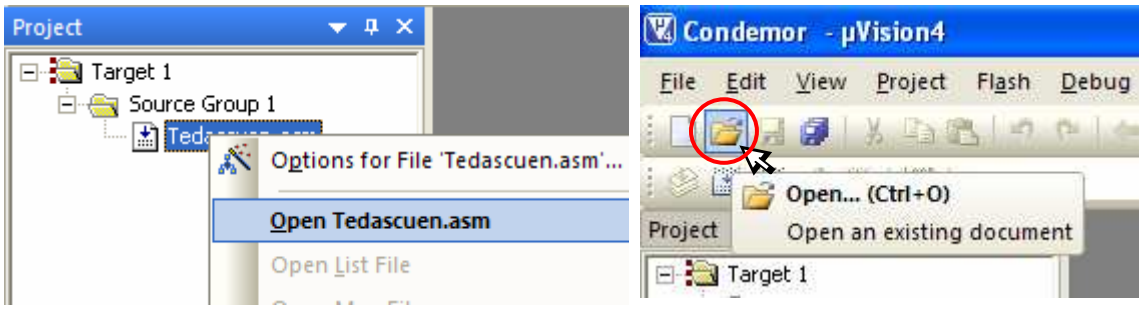


Fig. 3.6: a) Apertura mediante menú local b) Apertura en barra de herramientas

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

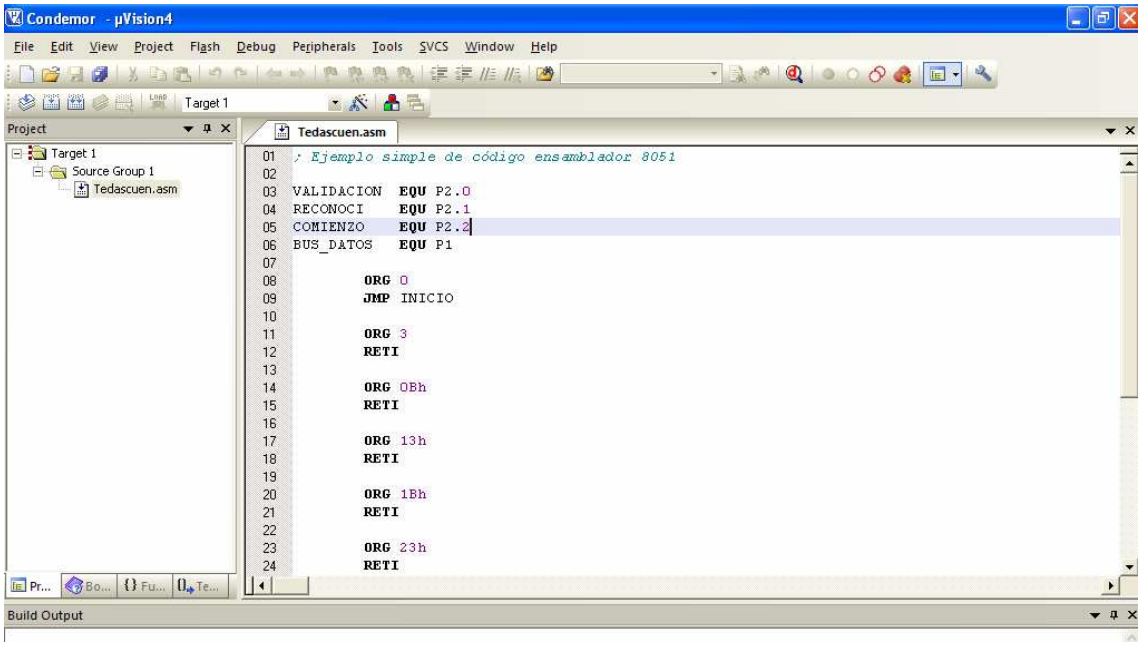


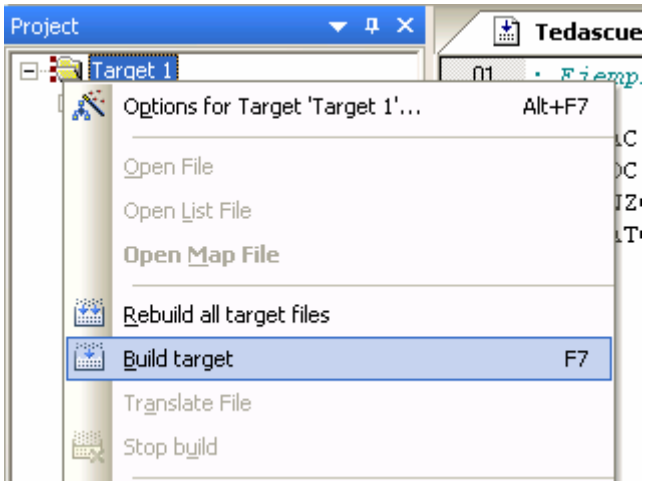
Fig. 3.7: Ventana de edición lista para ser usada

Para ello bastará ubicar el cursor en la ventana de edición, y comenzar a escribir. Esta ventana, por defecto, se ubica en la zona de trabajo, en forma de documento con pestaña.

4. CONSTRUCCIÓN DEL FICHERO EJECUTABLE FINAL

4.1. Ensamblaje y montaje de los módulos

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, púlsese con el botón derecho sobre ***Target 1*** en la ventana de proyecto. Selecciónese ***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 barra de herramientas.



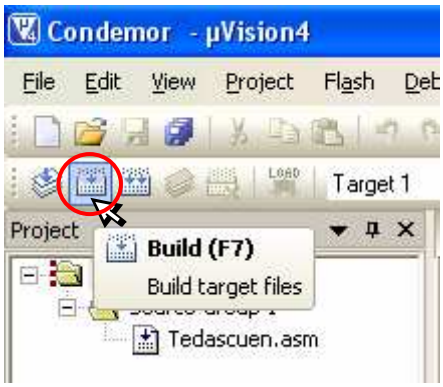


Fig. 4.1: a) Construcción de la aplicación final mediante menú local

b) Ídem mediante la barra de herramientas

4.2. Definición de las opciones de construcción de un proyecto

Si se tiene intención de programar la FLASH ROM de un microcontrolador, entonces 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 manera es hacerlo mediante el oportuno icono de la barra de herramientas.

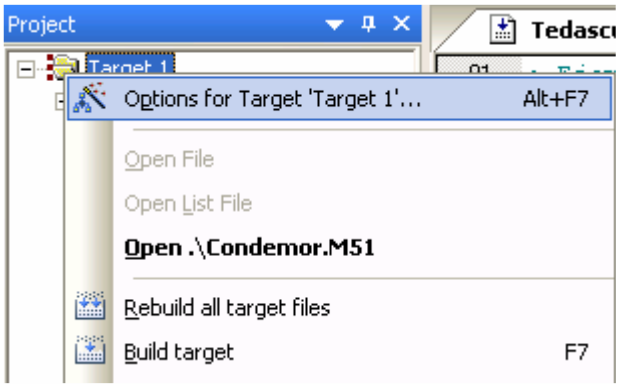


Fig. 4.2:
a) Selección de las opciones de salida en menú emergente

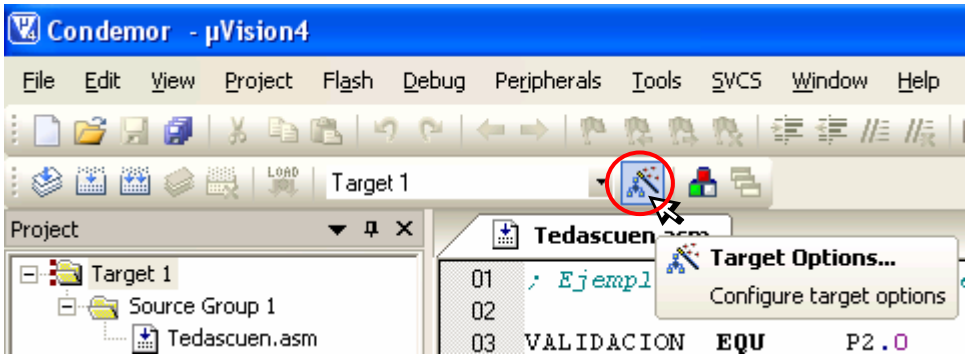


Fig. 4.2: b) Ídem mediante la barra de herramientas

Se abrirá una ventana, y en la pestaña **Output** se deberá activar la opción ***Create HEX file***.

También, en la pestaña ***Target*** póngase la frecuencia del cristal que se vaya a utilizar en el diseño y márquese la opción ***Use on-chip ROM***.
Lo primero resulta útil en ciertos casos de depuración en los que para una adecuada simulación se precisa una medición de tiempos no tanto en ciclos máquina sino en valor absoluto. Si no se supiese la frecuencia de trabajo entonces no sería posible llevar a cabo adecuadamente el análisis de la depuración.

Sobre la opción *Use on-chip ROM*, ésta se refiere a informar que en el diseño que se está llevando a cabo se usará la ROM interna de programa. De esta manera el depurador puede saber cómo comportarse. Téngase en cuenta que existe una técnica de desarrollo de código mediante conmutación de bancos (*bank switching*, en inglés), y esta técnica no es aplicable con la ROM interna de programa.

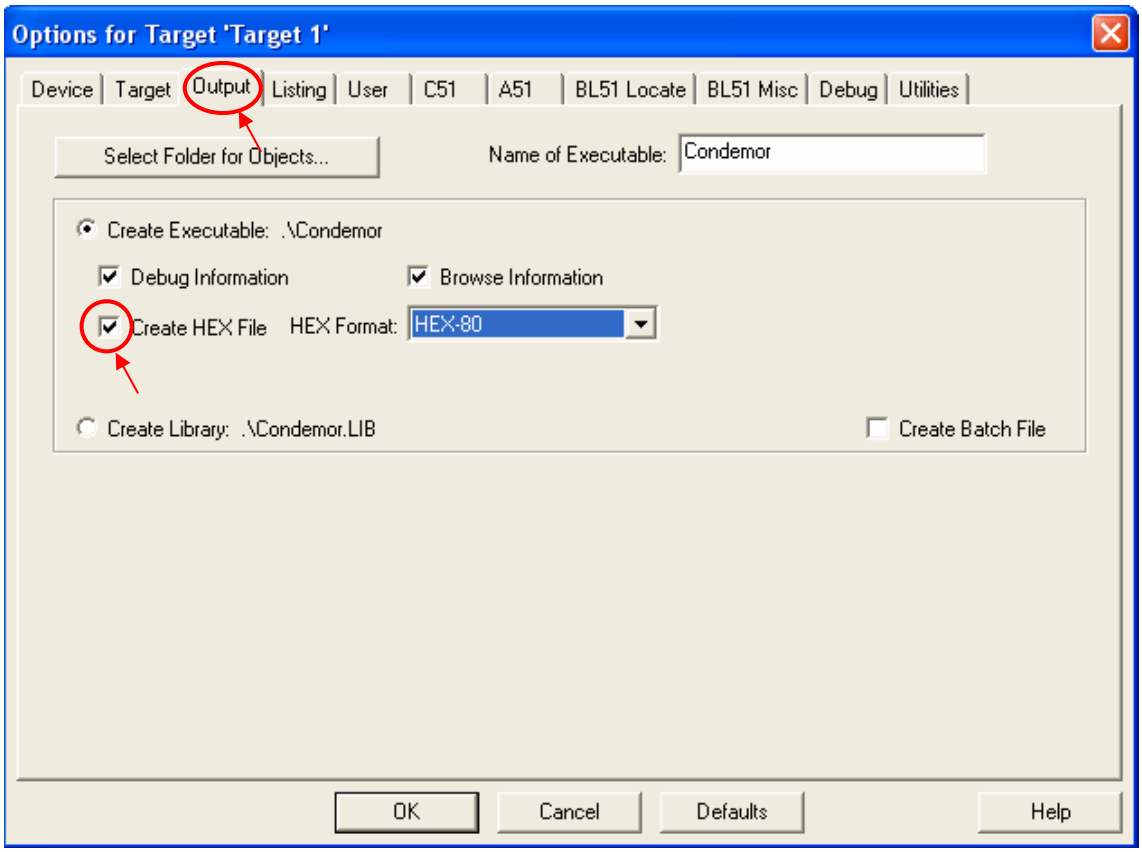


Fig. 4.3: Pestaña con las opciones de salida de un proyecto

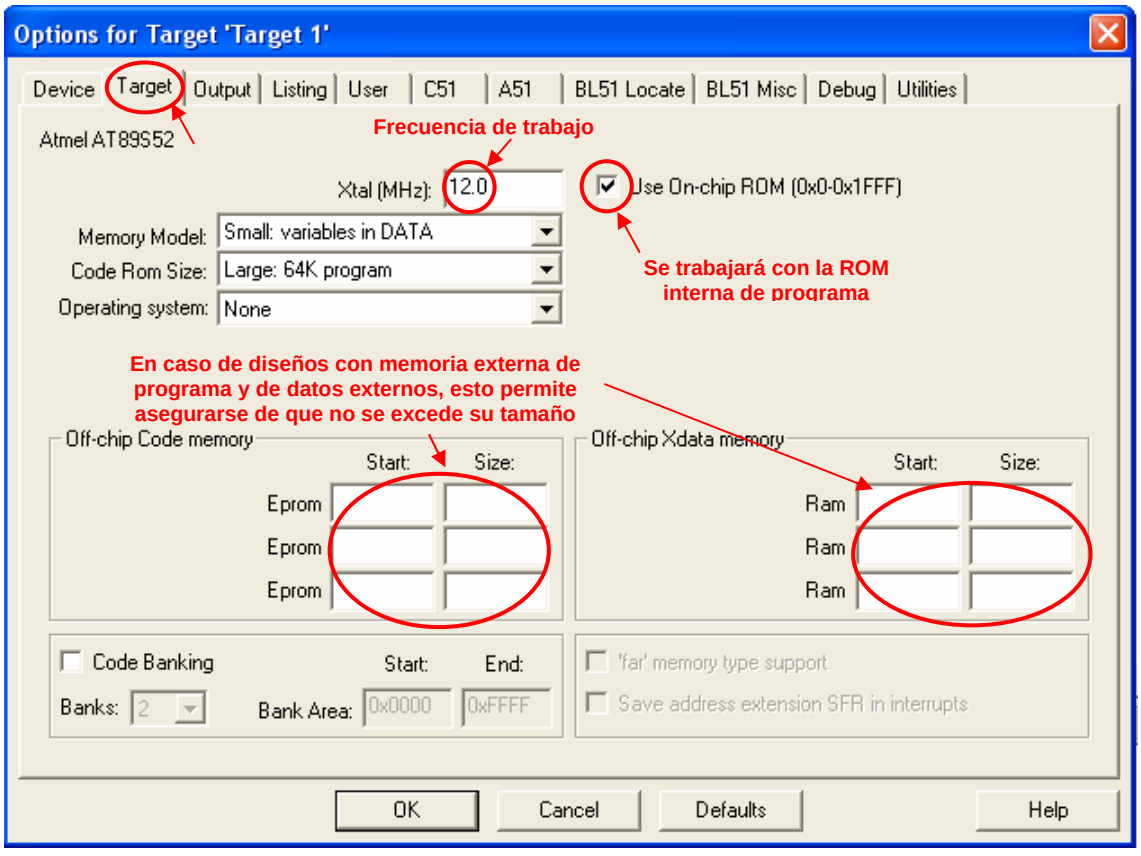


Fig. 4.4: Pestaña para la definición del sistema microcontrolador

4.3. Diferencia entre *Ensamblar*, *Construir* y *Reconstruir*

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. Lo que en *µVision* se denomina *build* en la generalidad de este tipo de herramientas se denomina *make*.

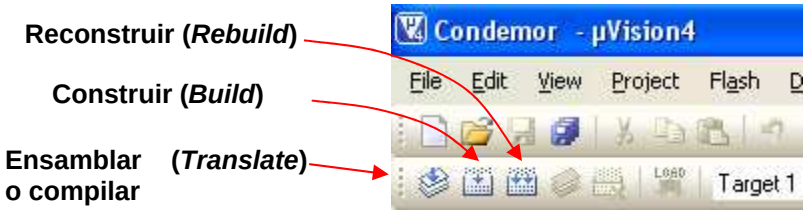


Fig. 4.5: Iconos de construcción, reconstrucción y ensamblado/compilado

La función ***Translate*** ensambla (o compila, caso de que se programe en C) exclusivamente el fichero activo, es decir, el archivo que se encuentre abierto en la ventana de edición en la zona de trabajo.

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 (*Build Output)*, ubicada normalmente en la zona inferior del entorno de desarrollo *Keil µVision 4*. Para ver estas incidencias es necesario que dicha ventana esté abierta, independientemente de su formato (ventana acoplada, flotante, auto-ocultable, etcétera). En la figura que sigue puede advertirse la información que resultaría del proceso de construcción del proyecto del ejemplo.

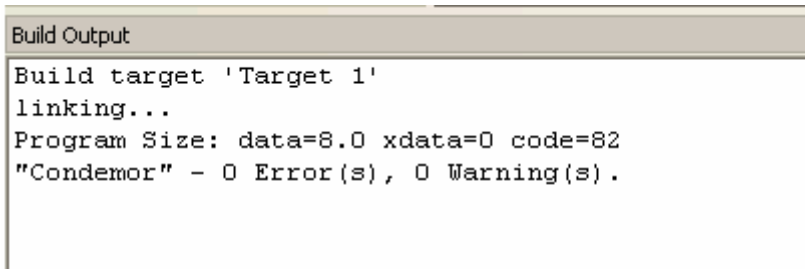


Fig. 4.6: Mensajes en la ventana de salida sobre el proceso de construcción

4.4. Gestión de errores en el ensamblado y montado

Caso de producirse errores, sean de edición o de montaje, 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 o entrará en la ventana de edición del módulo fuente en el que se encuentre ese error y el cursor se situará en la línea que lo produjo, de manera que así rápidamente el usuario podrá hacer la oportuna corrección.

En la figura que sigue se ilustra el caso de la existencia de dos errores en la escritura del ejemplo que se ha venido usando como proyecto. El primer error, en la línea 32 del código fuente, consiste en que se ha cometido el desliz de referir la etiqueta VALIDASION en lugar de VALIDACION, que es el nombre usado en su declaración en la línea 3. El segundo error, en la línea 34, es de tipo sintáctico y consiste en el uso de un mnemotécnico inexistente; se ha escrito SET en lugar de SETB.
En la siguiente figura se puede ver el efecto de hacer una doble pulsación izquierda en la línea de la ventana de salida donde se informa del error en la línea 32 del fichero. Obsérvese cómo

queda marcada la línea errónea en la ventana de edición, y el cursor se ubica al comienzo de ella. Así se podrá realizar rápidamente la oportuna corrección del error.

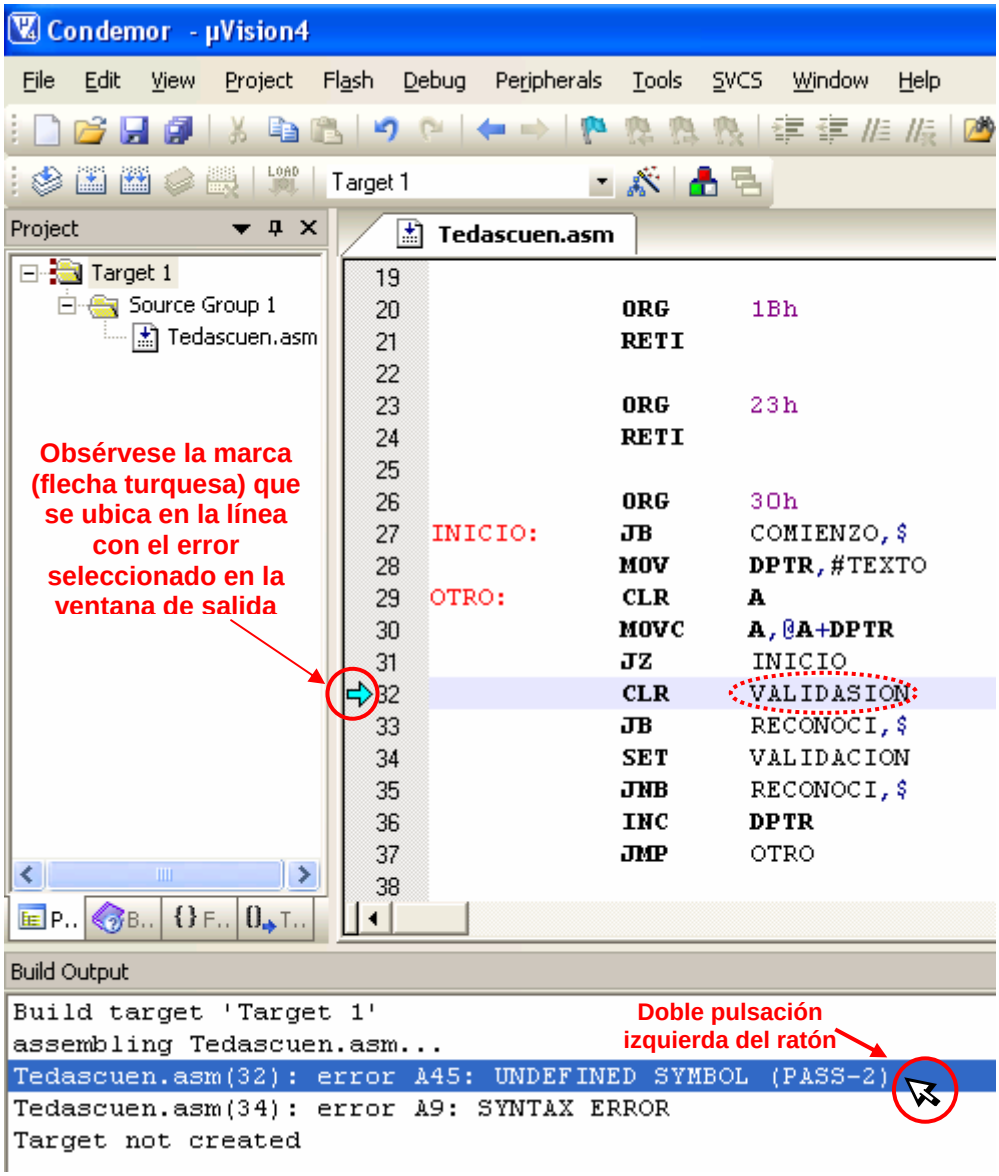


Fig. 4.7: Gestión de los errores desde la ventana de salida del proceso de construcción

4.5. Caso de programación en lenguaje C: opciones de compilación

Si se programa en lenguaje C –*Keil µVision* incluye un excelente compilador de C– entonces el consumo de memoria puede resultar superior al de una buena programación en lenguaje ensamblador. Para intentar minimizar esto, resulta imprescindible configurar adecuadamente las opciones de compilado en lo que se refiere a las técnicas de optimización a utilizar en la compilación del módulo fuente seleccionado. Esta cuestión no se tratará en este tutorial o prontuario, que se centra en el uso del lenguaje ensamblador. No obstante, los pasos serían los siguientes:

1º) Defínanse las opciones del fichero fuente escrito en C; para ello antes hay que seleccionar el archivo en el proyecto pinchándolo con el ratón en la ventana de Proyecto. Entonces 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:

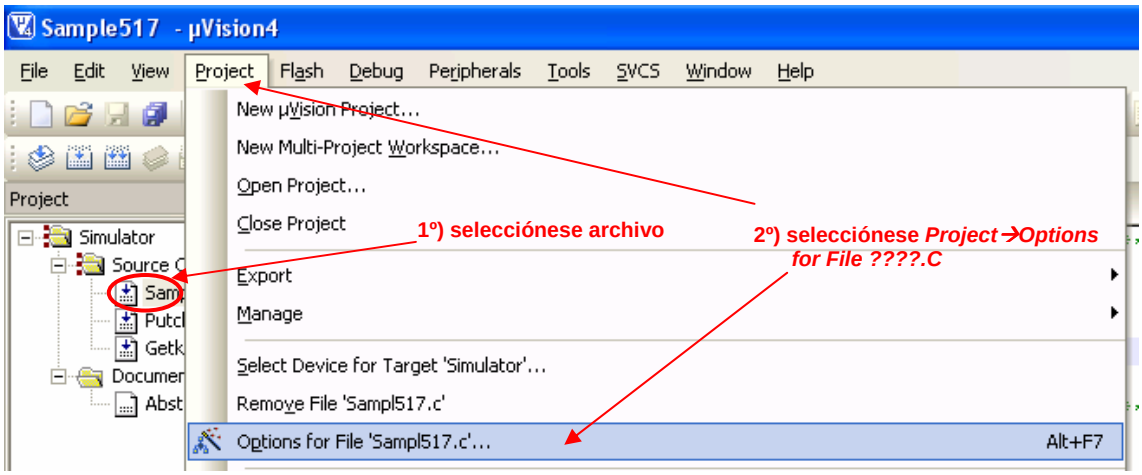


Fig. 4.8: a) Opciones de compilación vía barra de menú

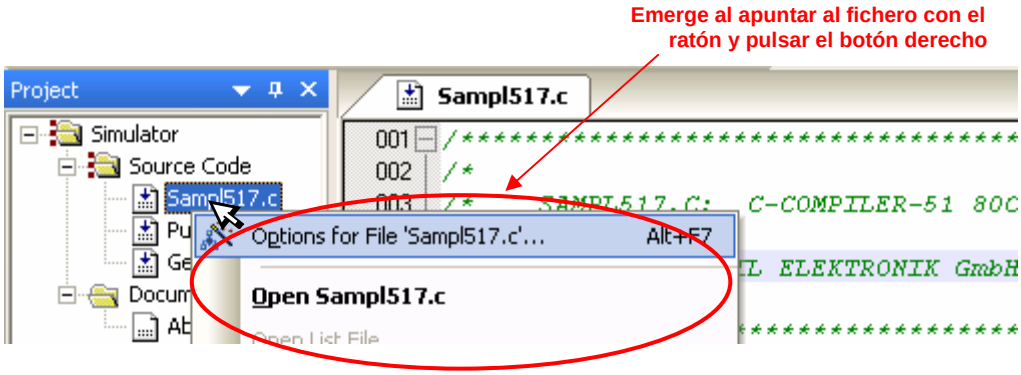


Fig. 4.8: b) Opciones de compilación vía ventana de proyecto

2º) Al hacer lo anterior, 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 en ensamblador. Esto es útil para poder ver cómo de eficiente es el código generado automáticamente por C51, y para afinarlo a mano si se estima oportuno. Esta opción, como puede suponerse, se suele utilizar no tanto para analizar el conjunto del código sino sólo el de aquellas porciones o subrutinas que se estimen críticas para conseguir el rendimiento buscado.
- Nivel de optimización del código, en lo referentes a las diversas técnicas de compilación a utilizar.
- Criterio principal: optimizar el tamaño o la velocidad del código. Téngase en cuenta que normalmente ambos criterios son antagónicos: a mayor velocidad menor compactación del tamaño del código y a mayor compactación del código menor velocidad de ejecución.

Esto puede verse ilustrado en las siguientes figuras.

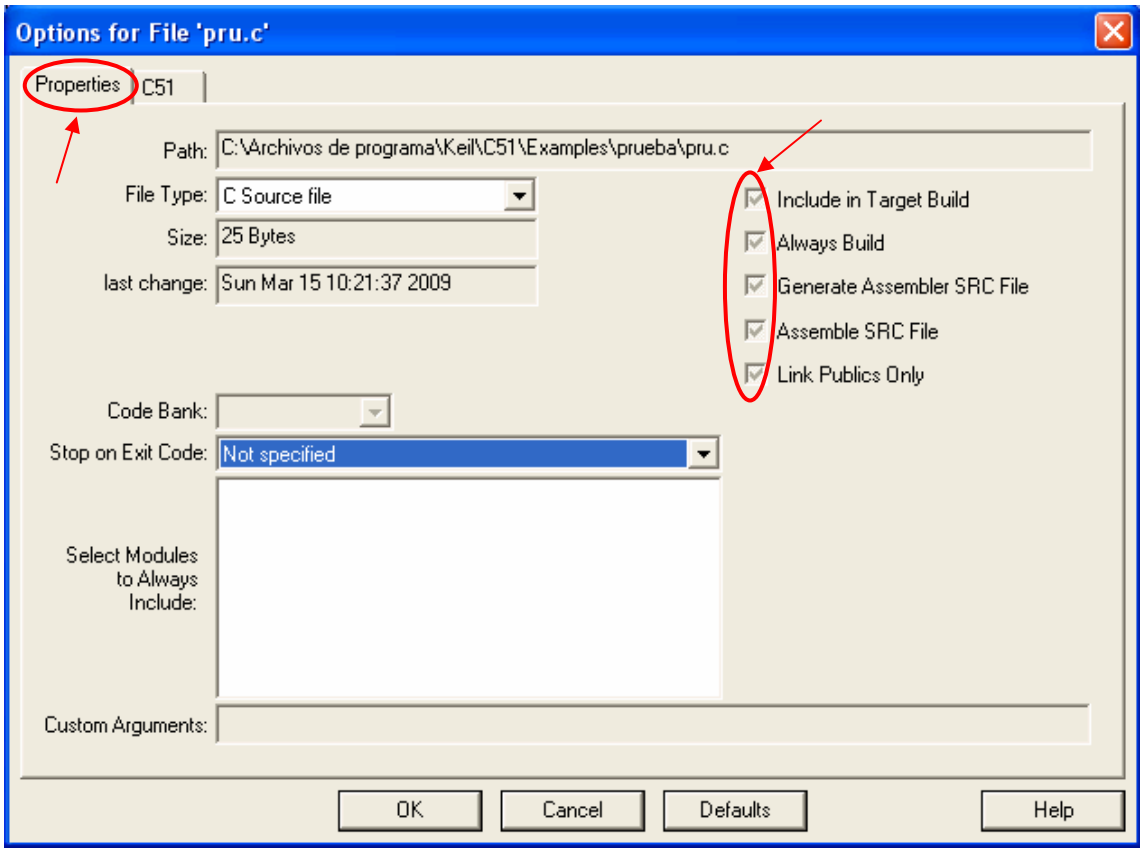


Fig. 4.9: Opciones de compilación: propiedades

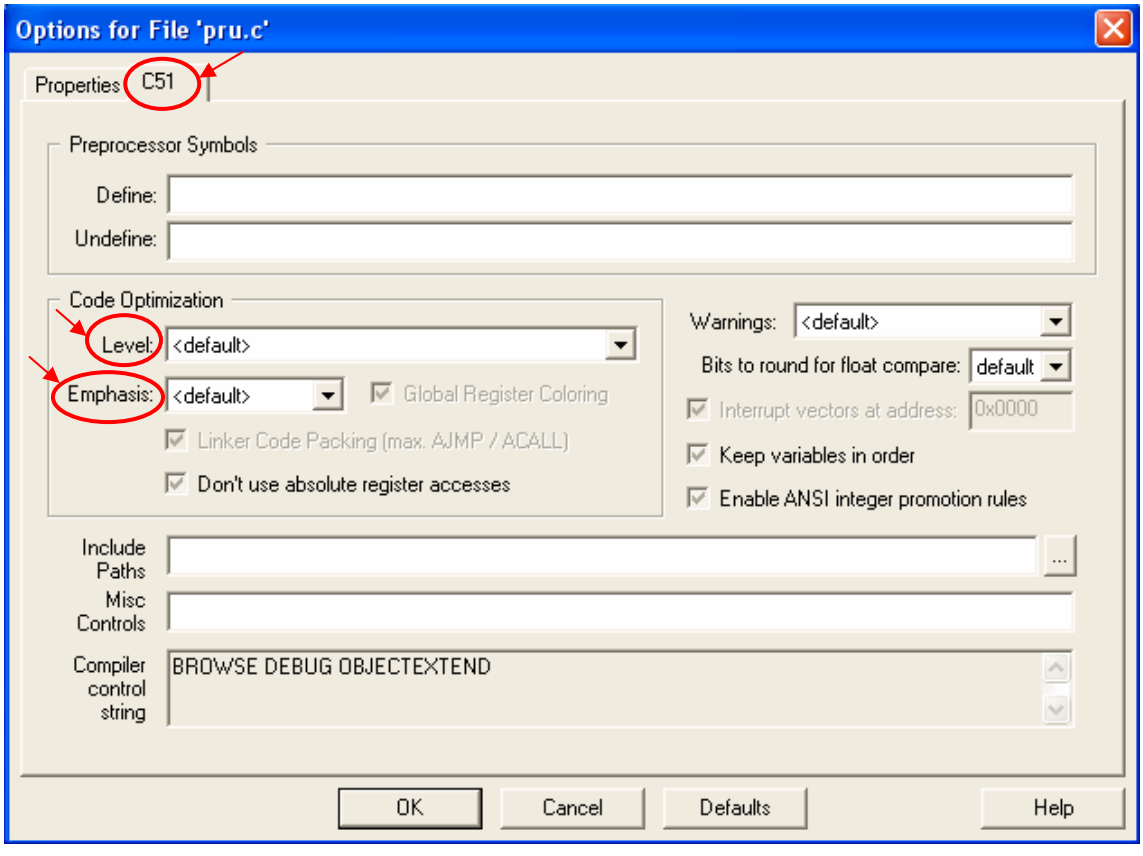


Fig. 4.10 Opciones de compilación: nivel y énfasis de optimización

Opciones del nivel de optimización del código:

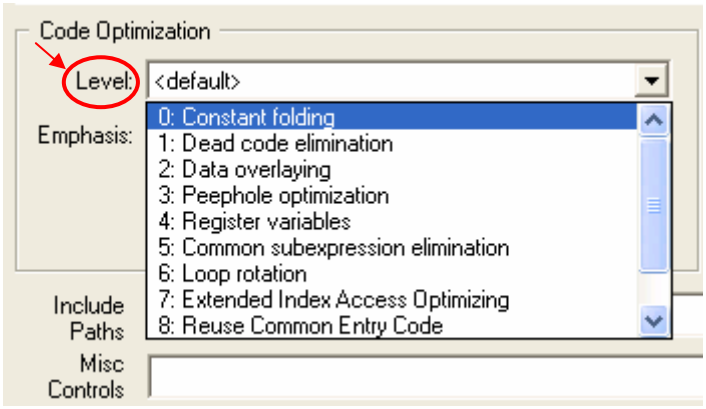


Fig. 4.11: Opciones de nivel de optimización

Opciones del énfasis en la optimización:

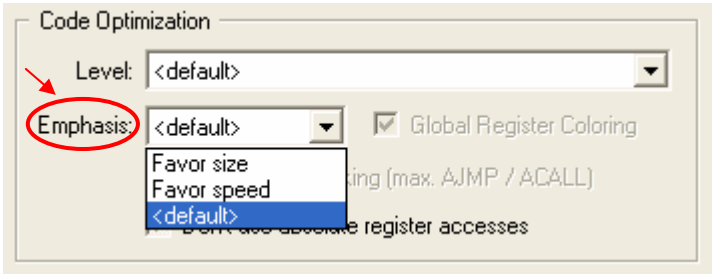


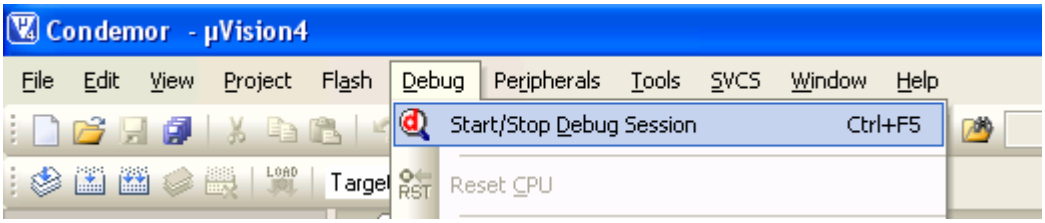
Fig. 4.12: Opciones de énfasis de optimización

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

5. DEPURACIÓN DEL CÓDIGO I: EL ENTORNO Y SU CONFIGURACIÓN

5.1. Inicio de una sesión de depuración

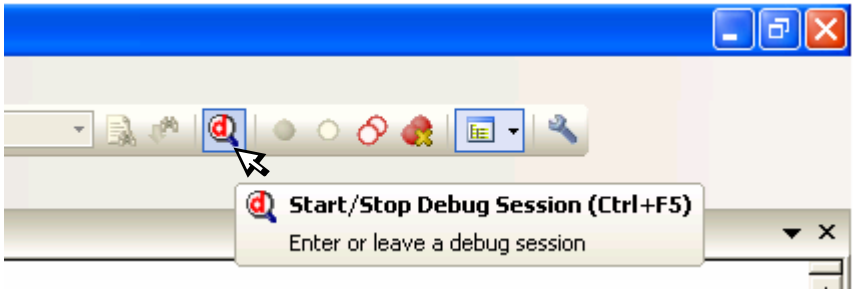
Selecciónese ***Debug*** en la barra de menú y en el desplegable selecciónese ***Start/Stop Debug Session***.



a) en barra menú

Fig. 5.1: Inicio del modo de depuración

b) en barra de herramientas



Se entra en la ventana de depuración automáticamente, salvo si se está trabajando con la versión demo de *µVision 4*, en cuyo caso entonces aparecerá una ventana en la que se advierte de que se está trabajando con una limitación de 2 kilo-octetos de código:

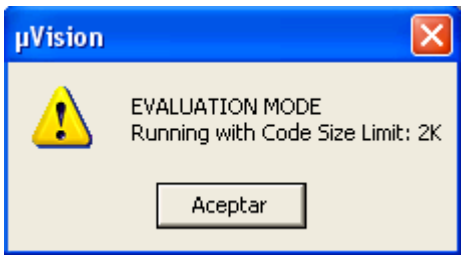


Fig. 5.2: Advertencia de limitación de tamaño para la versión demo

:

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 habilidad y soltura en su empleo.

5.2. Aspecto del entorno en modo depuración

El aspecto de la ventana de depuración mostrado en la figura siguiente es el que aparece por defecto la primera vez que se entra a depurar un proyecto. Si el usuario cambia la configuración y disposición de las ventanas o abre otras nuevas, al cerrar el proyecto al finalizar una sesión de trabajo entonces la próxima vez que abra el proyecto y se entre en el modo de depuración se recuperará la misma organización que quedó en la sesión precedente.

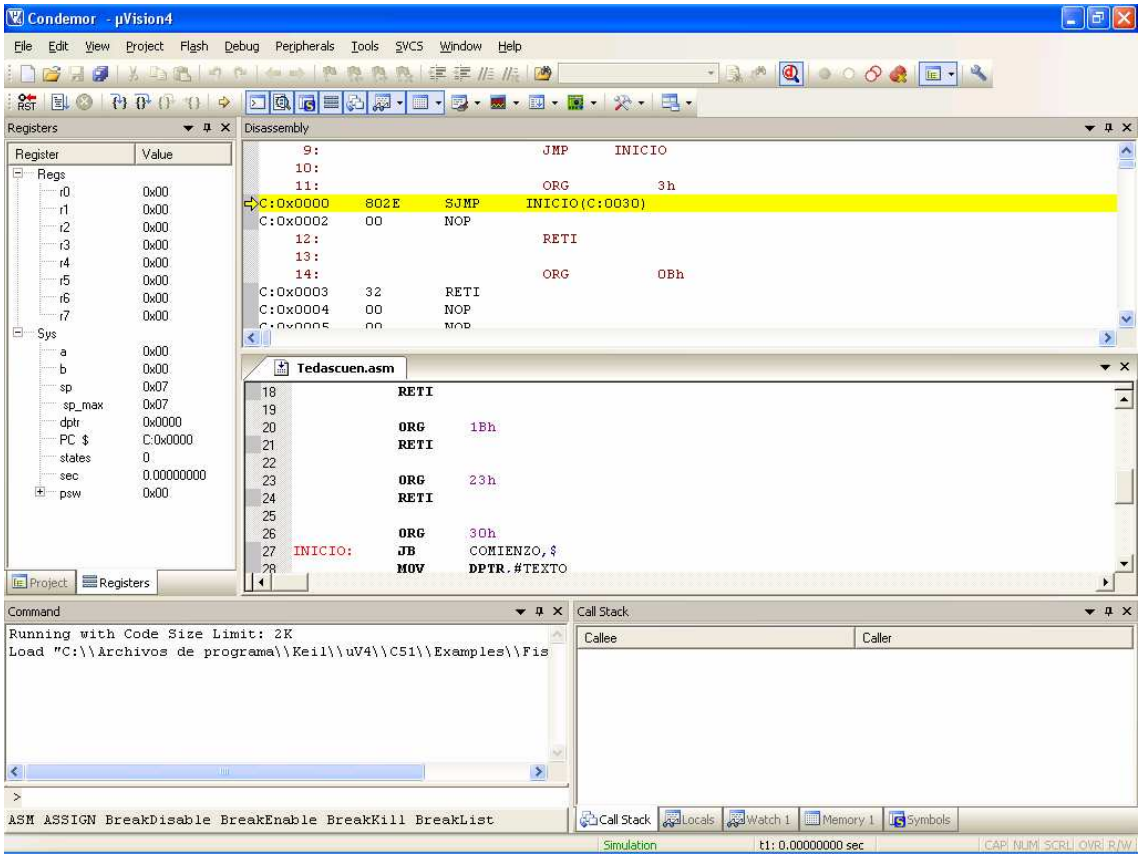


Fig. 5.3: Aspecto del entorno en modo depuración

Las partes que se pueden distinguir en ella son las siguientes:

Ventana de registros internos

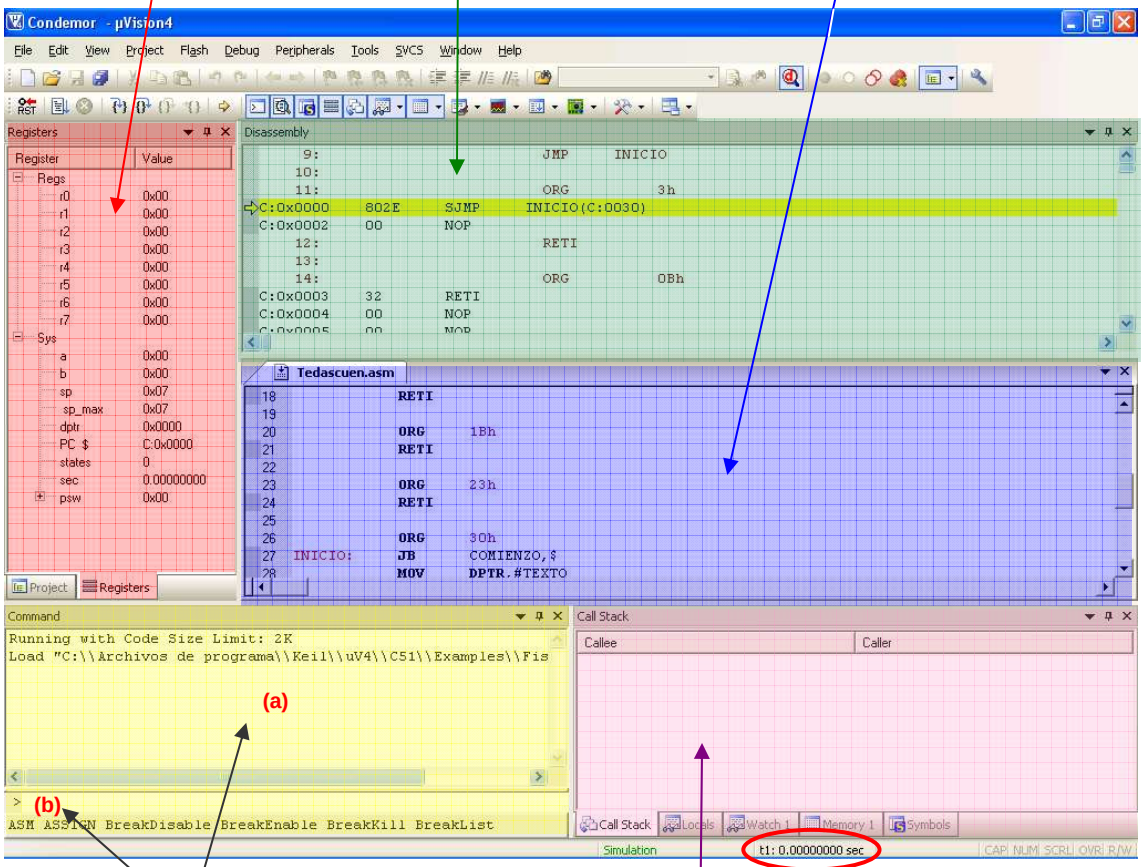
Obsérvese que existe también una pestaña para la ventana de proyecto

Ventana de desensamblado

Con ella se sigue y controla la ejecución

Ventana de edición

La misma que se tenía antes de entrar en *modo depuración* pero con algunas posibilidades adicionales para la depuración (por ejemplo, seguir y controlar la ejecución)



información sobre el tiempo transcurrido en la ejecución

Ventana de órdenes

En la parte superior (a) el sistema informa del resultado de los comandos dirigidos al depurado e introducidos en la zona inferior (b), que es la línea de comandos. La zona (a) también responde a varias de las acciones realizadas por la barra de menú o de herramientas

Zona de análisis

Comprende diversas pestañas útiles en el proceso de depuración, para inspeccionar los recursos del sistema

Fig. 5.4: Zonas del entorno en modo depuración

5.2.1. La ventana de registros internos

En esta ventana se muestran los registros internos del procesador. Se presentan agrupados; por un lado los registros Rn, y por otro los restantes significativos (acumulador A, registro B, puntero DPTR, puntero de pila, etcétera). El registro de estado, PSW, se presenta en forma de registro o en forma de banderas individuales.
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 respectivamente en el signo **+** ó **–** del ítem deseado.

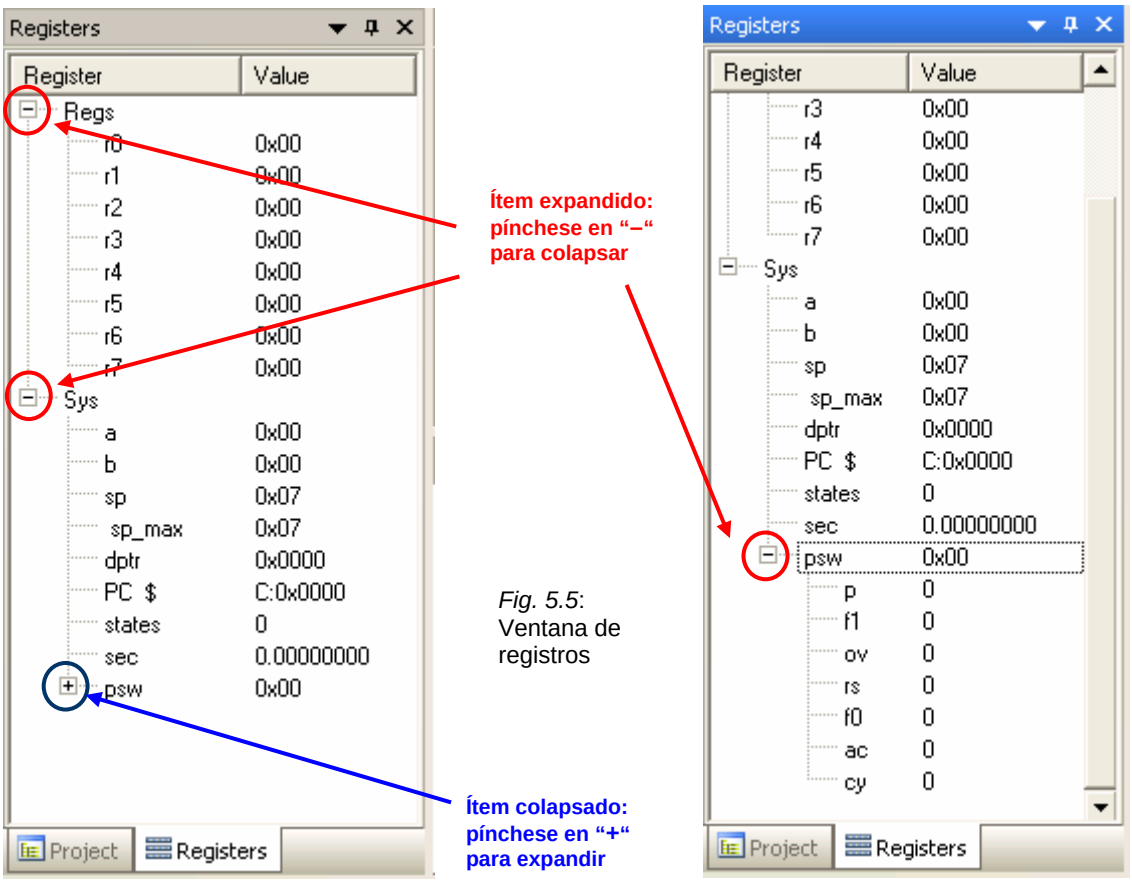


Fig. 5.5: Ventana de registros

Caso de que se hubiese cerrado la ventana de registros, para volverla a abrir hágase ***View→Project Window*** en la barra de menú o pínchese en el icono  .

5.2.2. La ventana de edición

La ventana de edición es la misma que aparecía en el entorno en la fase de creación y edición de un proyecto. Por tanto, se puede hacer exactamente lo mismo que entonces. No obstante no ha de perderse de vista el que se está en la fase de depuración y por tanto **los cambios editados no tendrán ningún efecto sobre el código depurado**. De hacerse algún cambio y pretender que éste tenga efecto, es necesario salir del *modo depuración* y regresar al modo inicial para proceder a ensamblar, construir o reconstruir el proyecto y sólo entonces reentrar en el *modo depuración*.

En el *modo depuración* la ventana de edición tiene una utilidad adicional. A su izquierda se puede observar, por un lado, una flecha amarilla que indica la próxima instrucción a ejecutarse (es decir, donde apunta el contador de programa); y por otro la parte del código fuente que es ejecutable (sombreado en gris oscuro).
Como se verá más adelante, sobre esta ventana se puede interactuar, como atajo, para realizar ciertos procesos en la depuración. Por ejemplo, poner puntos de ruptura.

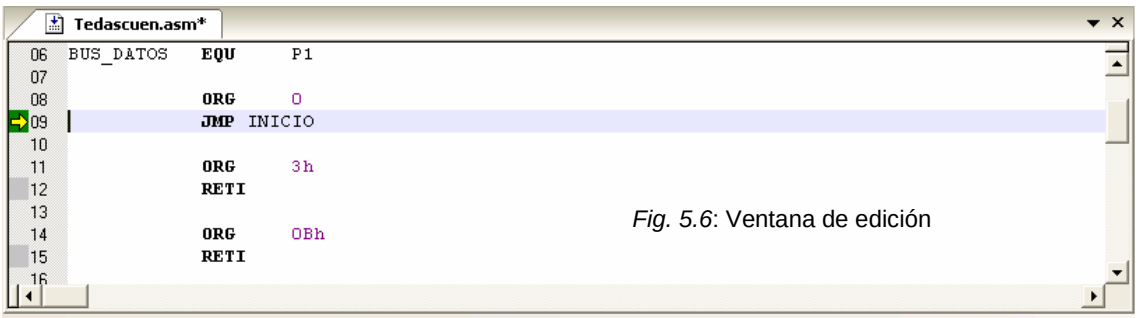


Fig. 5.6: Ventana de edición

5.2.3. La ventana de desensamblado

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 a partir de que se encuentra cada instrucción.

Igualmente, en el borde izquierdo se aprecia una flecha amarilla que significa lo mismo que la de la ventana de edición.

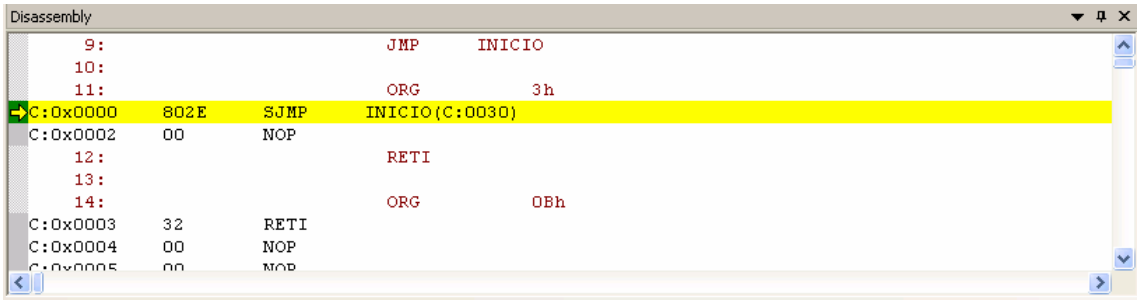
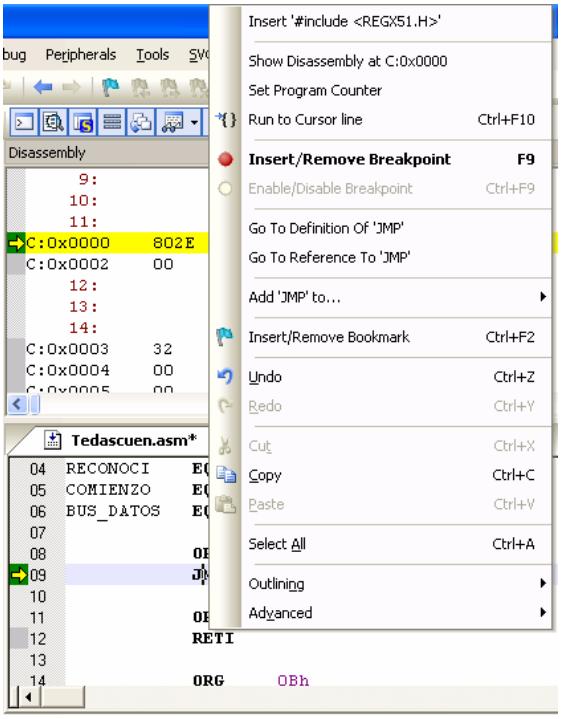


Fig. 5.7: Ventana de desensamblado

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 menú con todas las posibilidades de interactuación:



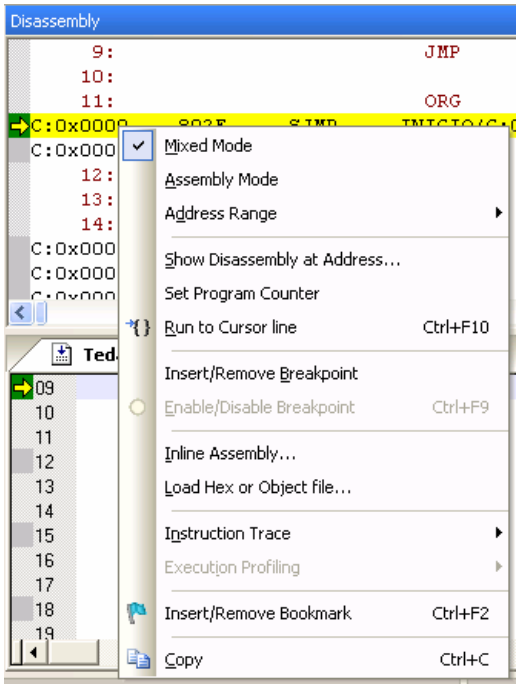


Fig. 5.8: a) Ventana emergente en la de edición b) ventana emergente en la de desensamblado

Las opciones fundamentales se comentarán más adelante, especialmente aquellas relacionadas con el proceso de depuración. Tan sólo nos detendremos en las opciones ***Mixed mode*** y ***Assembly Mode***.

Modos de visualización: mixta o sólo en ensamblador
El modo mixto (*mixed mode*) consiste en que en la ventana de desensamblado se presenta la información en formato tanto tal y como figura en el código fuente como, a continuación, en formato desensamblado de la memoria (dirección ocupada por la instrucción, su codificación binaria en formato hexadecimal, y la interpretación de tal patrón binario como una instrucción en formato mnemotécnico). En el modo ensamblador no se acompaña con la información en formato editado.

A título aclaratorio, obsérvese en el desplegable de la figura anterior cómo aparece marcada la opción ***Modo mixto*** *(Mixed Mode)*. Para observar la diferencia entre este modo y el ***Modo ensamblador*** *(Assembly* Mode*)* pueden verse las siguientes figuras:

Modo en ensamblador:

La ventana presenta los contenidos de la memoria interpretándolos también como instrucciones en formato mnemotécnico. Por este motivo difiere de cómo aparece en la ventana de edición

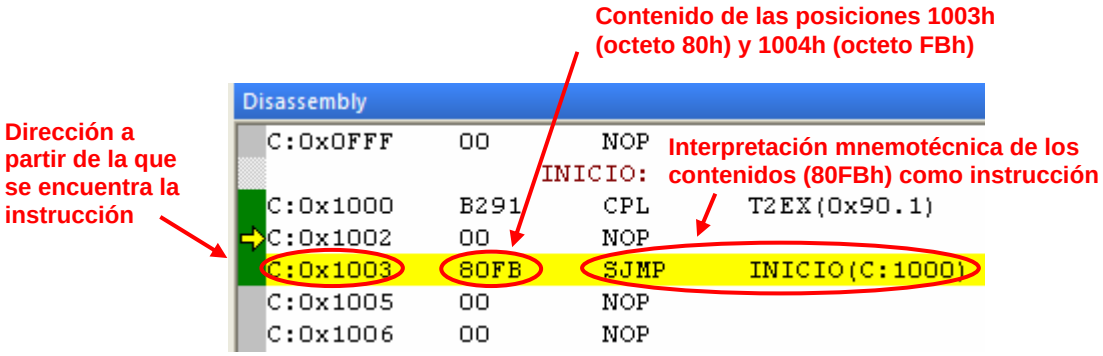


Fig. 5.9: Ventana de desensamblado en modo ensamblador

Modo mixto:

La ventana presenta cada línea tal y como se editó (aparece en color rojo), junto con su la interpretación de los contenidos como en el modo ensamblador.

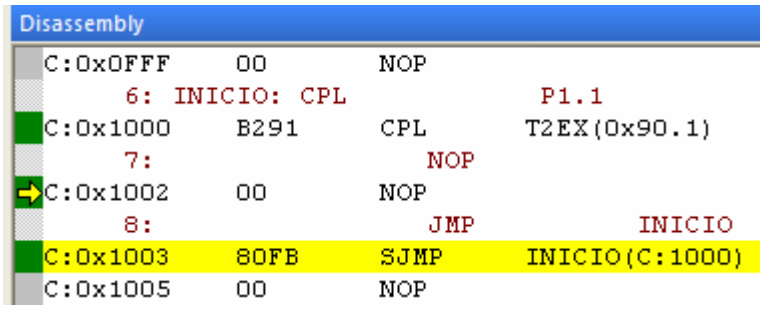


Fig. 5.10: Ventana de desensamblado en modo mixto

El modo mixto normalmente sólo resulta de utilidad cuando se programa en lenguaje C, dada las diferencias abismales entre una sentencia en este lenguaje y su equivalente en ensamblador como resultado de su compilación. Cuando se programa sólo en ensamblador el modo mixto resulta un engorro que no aporta gran cosa y sólo sirve para enmarañar la información mostrada en la ventana de desensamblado; por ello, es recomendable seleccionar la visualización en modo ensamblador exclusivamente.
En la figura del modo ensamblador puede verse que la instrucción **80FBh** se desensambla como ***SJMP INICIO(C:0x1000)*** en vez de ***SJMP 0x1003***. Esto pone de manifiesto cómo el depurador es de tipo simbólico al nivel de código fuente, pues tiene en cuenta los símbolos definidos al escribir el código fuente aunque se vea sólo en formato desensamblado.

5.2.4. Zona de análisis

Esta zona engloba una serie de ventanas que resultan de utilidad en el proceso de depuración. Mediante ellas se pueden inspeccionar ciertos recursos del sistema e interactuar con ellos. Las diferentes ventanas son las siguientes:

- **Ventana(s) de memoria**: En ellas se puede inspeccionar y modificar los contenidos de la memoria del sistema. Por defecto sólo existe una ventana, pero es posible abrir hasta cuatro. Típicamente para inspeccionar la RAM interna, la memoria de programa y la memoria externa de datos.

- **Ventana(s) de observación**: Está(n) concebida(s) para agrupar recursos dispersos que conviene tener juntos para facilitar y hacer más cómoda la depuración.

- **Ventana de símbolos**: En ella se presenta el valor de todos los símbolos utilizados, bien sean predefinidos (dirección equivalente de los nombres de los registros normales y de los SFR) o bien sean los definidos por el usuario al crear su código.

- **Ventana de variables locales**: En ella se presentan las variables definidas por el programador.

- **Ventana de la pila de llamadas** a subrutina, para saber la estructura de llamadas anidadas a subrutina.

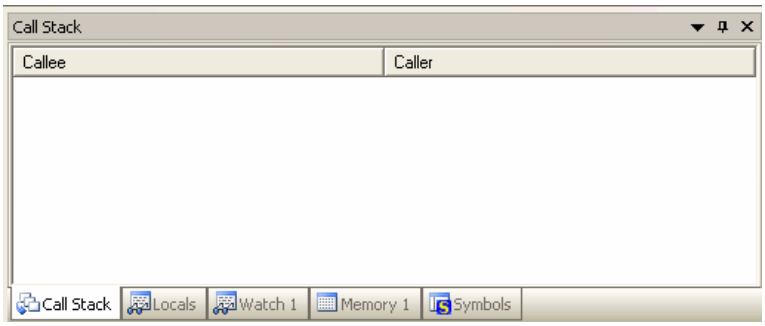


Fig. 5.11: Zona de análisis

5.2.5. Ventana de órdenes

Posee dos zonas; una amplia en la parte superior, en la que el entorno *µVision 4* informa del resultado de los procesos invocados por cualquier medio, y una pequeña en la parte inferior a través de la cual el usuario puede invocar en modo *consola* o *terminal* las distintas aplicaciones y funciones que conforman el sistema de desarrollo (por ejemplo, el módulo de ensamblado o de montaje). A través de esta ventana se pueden introducir comandos que no son posibles de otra manera o bien otros que pueden invocarse mediante la barra de menú o barra de herramientas pero que de este modo se pueden acompañar de parámetros o modificadores complejos.

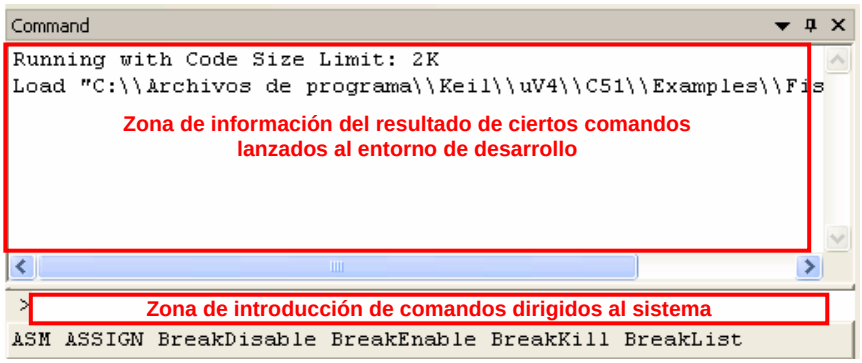


Fig. 5.12: Ventana de órdenes

5.3. PASOS PREVIOS A LA DEPURACIÓN DEL CÓDIGO ENSAMBLADOR

Como se acaba de ver, al entrar por primera vez en *modo depuración* se parte de una situación en la que se muestran ciertas ventanas. Según cada caso, puede resultar conveniente:

- Eliminar algunas de ellas por inútiles, y abrir otras que no lo están pero que se considera que se necesitarán.
- Definir la información que se desea analizar con estas ventanas.
- Reubicarlas, reformatearlas, redimensionarlas y, finalmente, definirlas y dotarlas de contenido.

Parte de estas cuestiones ya se han visto al comentar la filosofía de ventanas que posee el entorno *µVision 4*. Otras se comentarán a continuación.

5.3.1 Primero: abrir las ventanas no presentes

Existen varias posibilidades de hacerlo:

a) Vía barra de menú, con la opción ***View***.

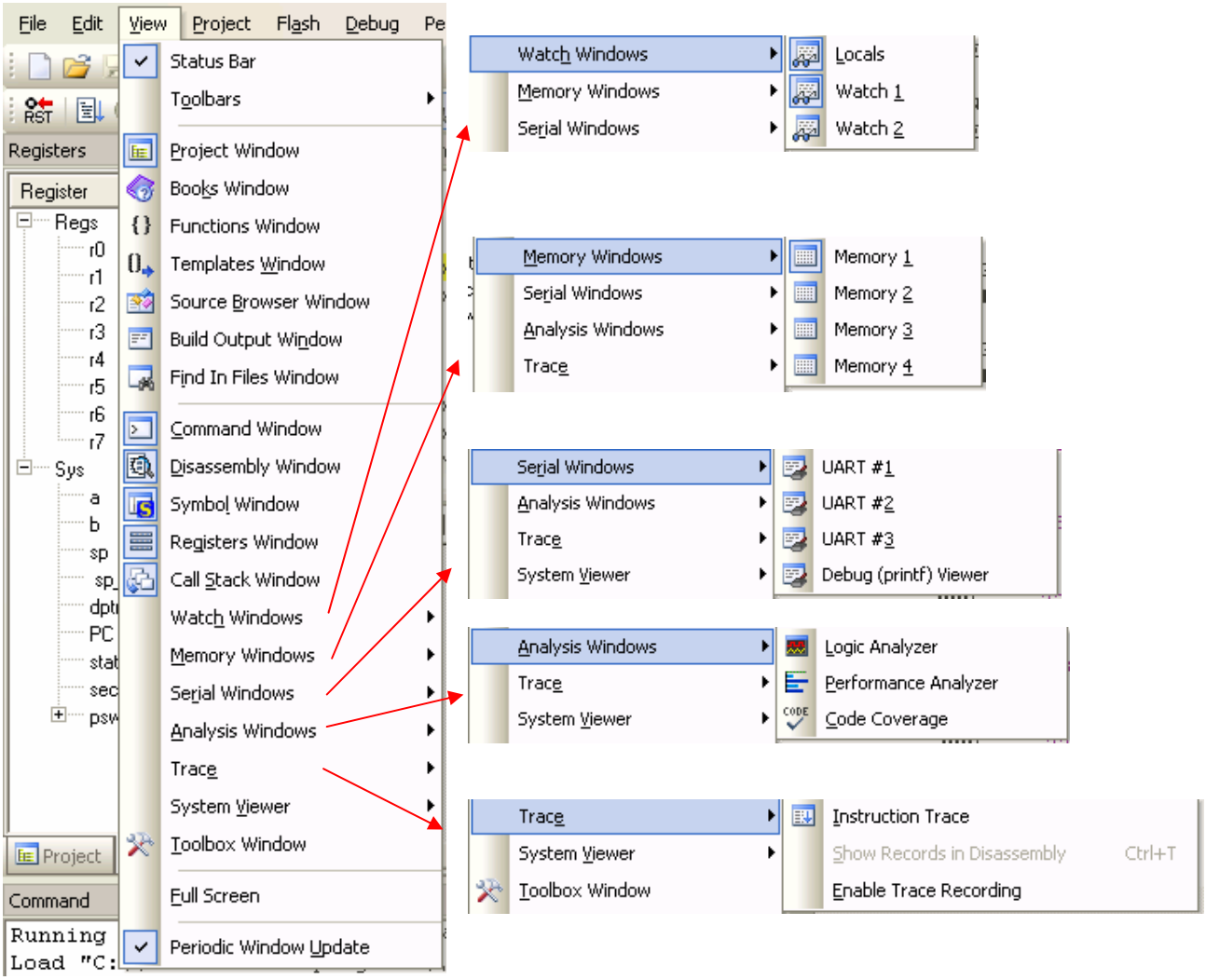


Fig. 5.13: Selección de las ventanas a abrir, vía opción *View* en la barra de menú

b) Vía barra de herramientas, para las ventanas usuales.

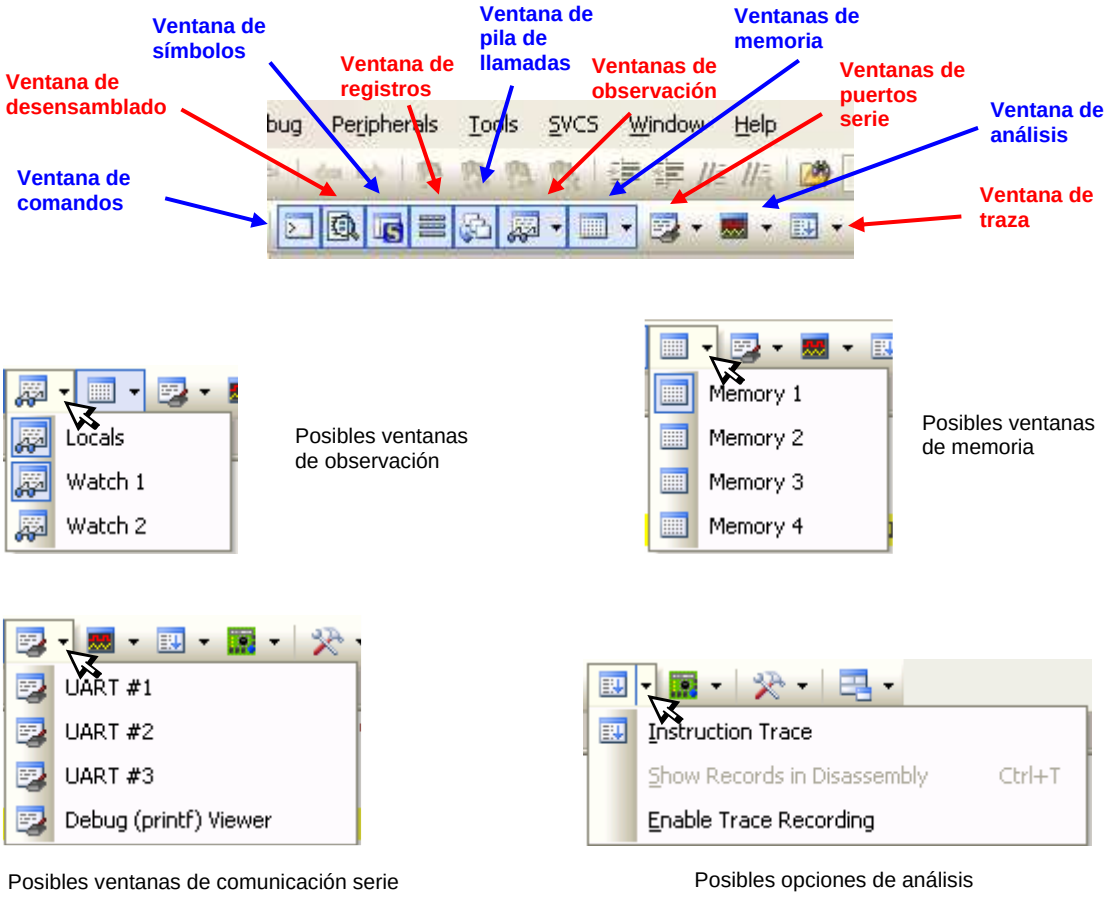
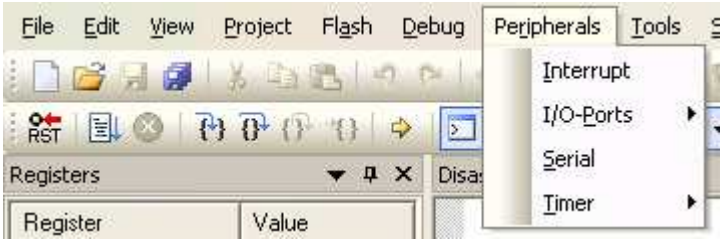
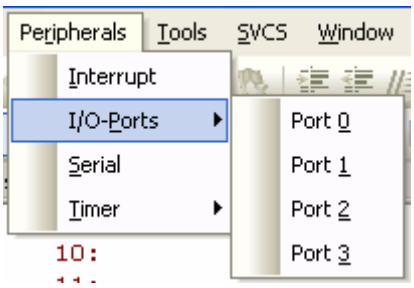


Fig. 5.14: Selección de las ventanas a abrir, vía barra de herramientas

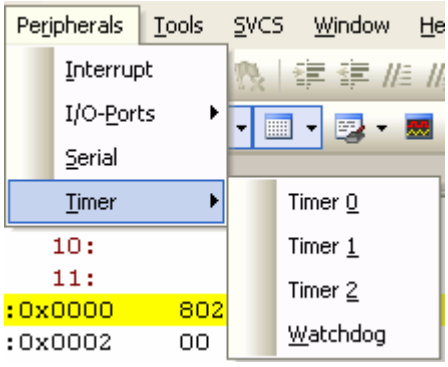
c) Si lo que se desea es abrir ventanas de los diversos periféricos, entonces en la barra de menú se selecciona la opción ***Peripherals*** y se selecciona el o los periféricos deseados.

Fig. 5.15:
a) Apertura de ventanas de periféricos





b)
Opciones de puertos



c)
Opciones de temporizadores

Las ventanas de los periféricos son del tipo propio de *Windows*, con la salvedad de que están bloqueadas y por tanto no se pueden redimensionar ni minimizar; tan sólo se pueden arrastrar y cerrar. En las siguientes figuras pueden verse las ventanas de algunos de los periféricos. Debe tenerse en cuenta que el tipo de periféricos, y por tanto sus ventanas, estarán en consonancia con el tipo de procesador que se haya definido al crear el proyecto.

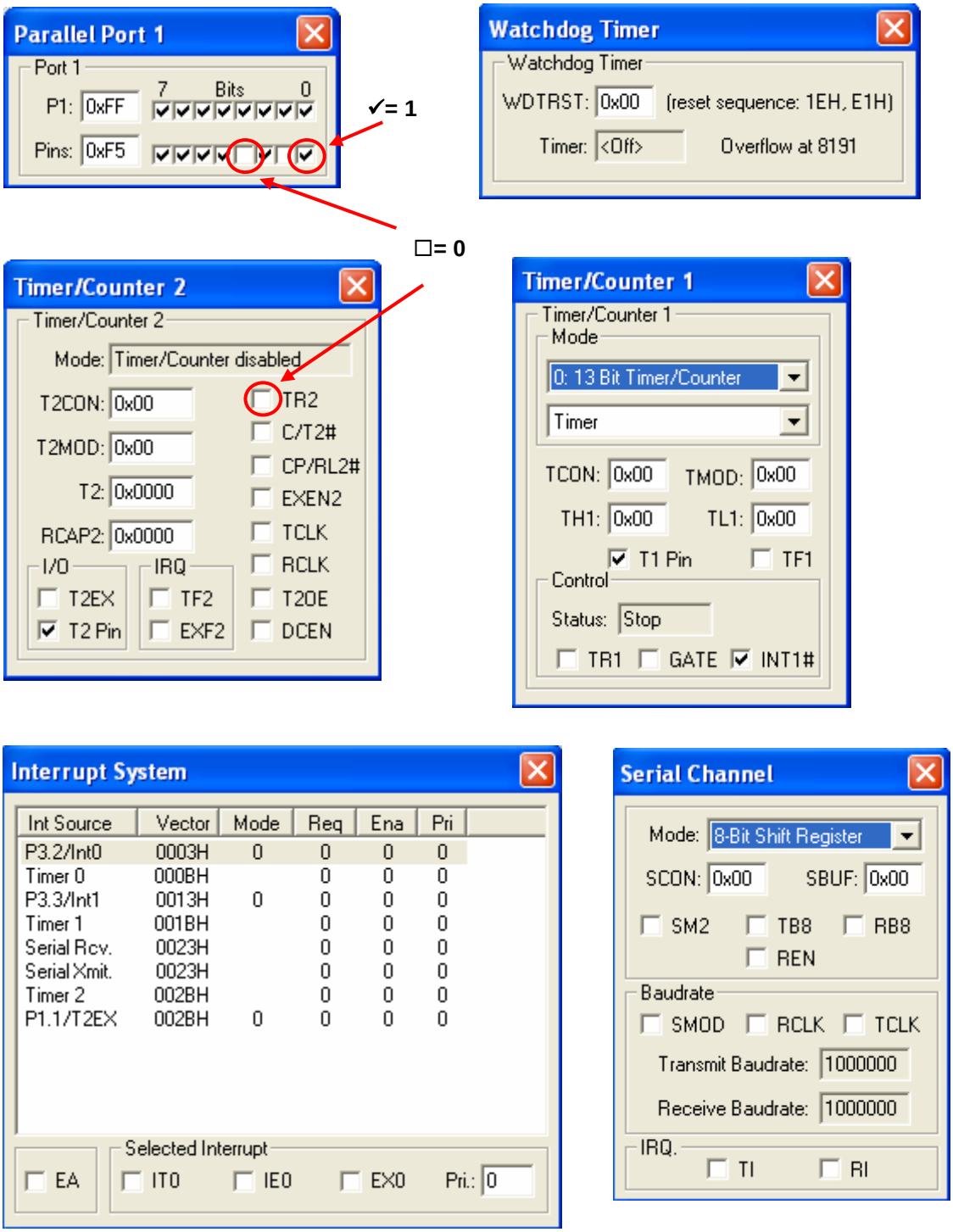


Fig. 5.16: Aspecto de diferentes ventanas de periféricos

En las ventanas puede verse que existen campos meramente informativos, así como otros con los que el usuario puede interactuar modificando sus valores, lo que implicará los oportunos cambios en el modo operativo del periférico afectado. En los campos de bit el símbolo ✓ representa un **1** lógico, y su ausencia un **0** lógico.

5.3.2. Segundo: reubicar, formatear y redimensionar las ventanas

Antes de empezar, conviene ubicar y formatear las ventanas en el lugar que se considere más adecuado. Aquí interviene frecuentemente el gusto personal del usuario, pues cada persona tiene sus preferencias subjetivas en cuanto a la configuración visual del entorno de trabajo. La manera de hacer esto ya se comentó al hablar de la filosofía de las ventanas en el entorno *µVision 4*.
Para el redimensionado, se puede actuar de diferentes formas en función del formato de cada ventana:

- **Si la ventana se encuentra acoplada o como documento con pestaña**, quiere decir que está en una zona unida solidariamente a otra. Recuérdese que las zonas son cinco: la de registros, la de desensamblado, la de edición, la de comandos y la de análisis. Existirá una separación entre esas zonas y al ubicar el ratón sobre esa zona fronteriza puede observarse que el símbolo de éste (puntero en forma de flecha) se muta en un símbolo indicador de frontera (distinto según se trate de una frontera vertical (−||−) o una horizontal (≒).

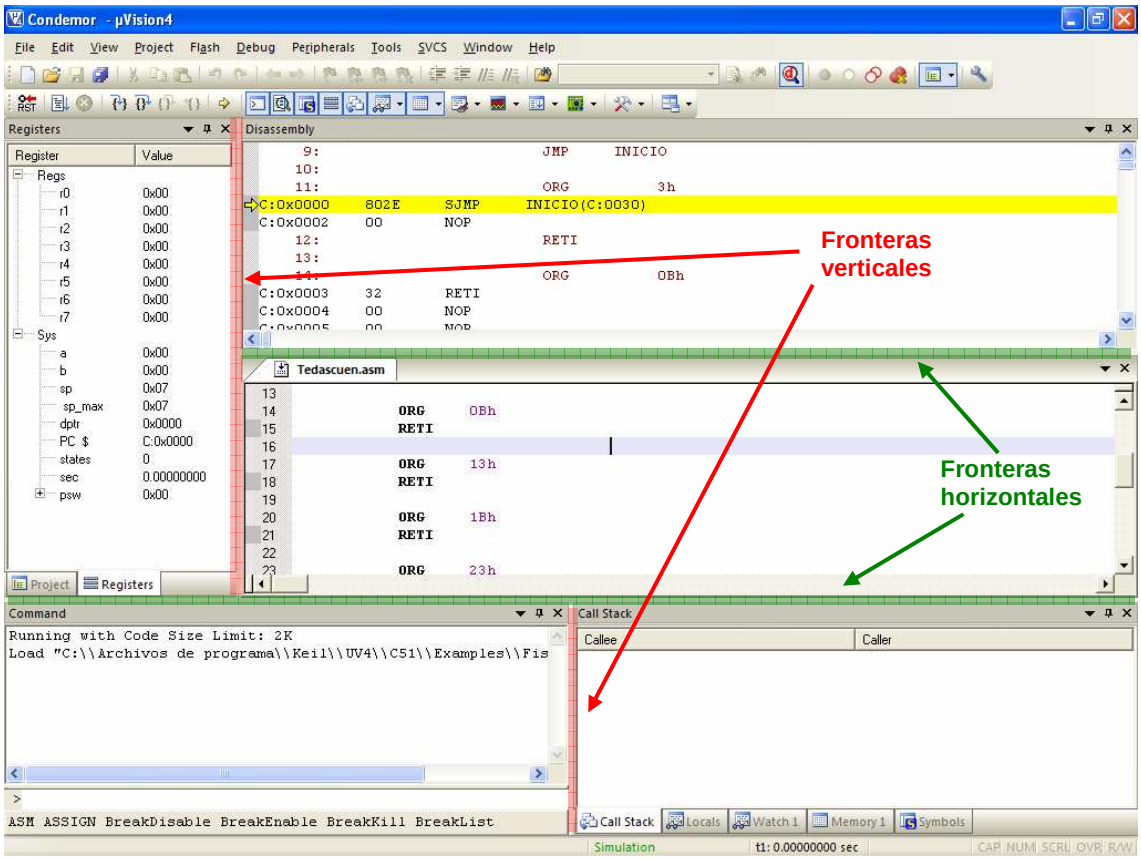


Fig. 5.17: Redimensionado actuando sobre las fronteras entre zonas acopladas

Por tanto, bastará con ubicar el puntero del ratón en una zona fronteriza hasta que el símbolo del ratón tome la forma del redimensionador de ventana. Para redimensionar se pulsará el botón izquierdo del ratón y se arrastrará hasta que se redimensione en la forma deseada. Es decir, se obra de manera similar a la típica 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 en *Excel*.

- **Si la ventana es flotante**, entonces se puede redimensionar de manera idéntica a como se haría con cualquier ventana tipo *Windows*©.

5.3.3. Tercero: definir y dar contenido a las ventanas

En la mayoría de las ventanas que se abran para el proceso de depuración se observa que aparecen vacías. Éste sería el caso de las de memoria y las de observación, por ejemplo. En otras, aparecen con información pero es preciso modificarla. Pues bien, a continuación se indicará cómo llevar a cabo esto en las ventanas más significativas y útiles.

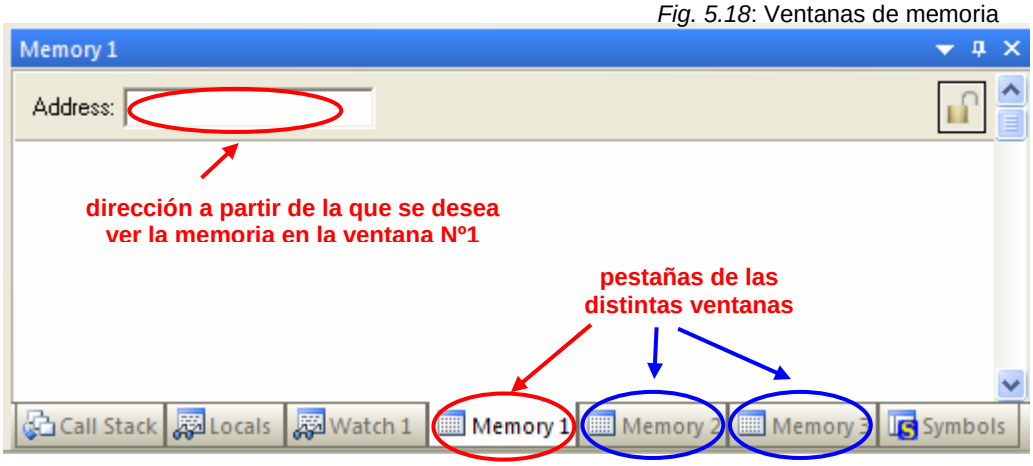
5.3.3.1. Contenidos en las ventanas de memoria:

Una vez que se hayan abierto tal y como se dijo en un epígrafe anterior (en la barra de menú haciendo ***View→Memory Windoa→Memory "n"***, o en la barra de herramientas sobre el icono de *ventanas de memoria*) entonces es necesario especificar lo siguiente:

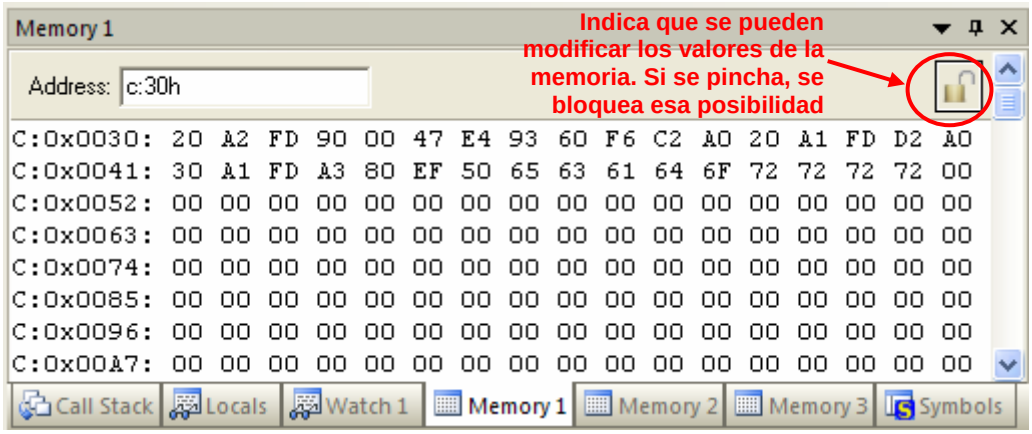
- Qué tipo de memoria se desea ver: RAM interna, memoria de programa o memoria externa de datos.
- A partir de qué dirección se desea verla.

En la figura que sigue puede verse el aspecto de la zona de análisis con tres ventanas de memoria vacías. Puede observarse que el campo de dirección (*address*) de la ventana 1 de memoria se encuentra sin definir.

Fig. 5.18: Ventanas de memoria



Si se quisiese definir la ventana nº1 para ver la memoria de programa a partir de la dirección 30h, se introduciría en el campo de dirección la expresión **C:30h** y se validaría con la tecla <*intro*> (tecla ↵). El resultado sería el que se muestra a continuación:



Como puede verse, en cada línea primero aparece la dirección de memoria a partir de la que se muestra el contenido, y entonces los contenidos (mostrados como valores en hexadecimal) de ésa y de las siguientes posiciones.

Dado que en un microcontrolador 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 de datos.

5.3.3.2. Contenidos en las ventanas de observación

Si no lo estuviesen, se abren haciendo ***View→Watch Window→Watch "n" / Locals*** en la barra de menú, o pulsando sobre el icono oportuno en la de herramientas y seleccionando la deseada de las tres posibles:

Fig. 5.19: Opciones en las ventanas de observación

Estas posibilidades son:

- **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**: ídem.
- **Pila de llamadas** a subrutinas: para saber la estructura de llamada concatenadas a subrutina.

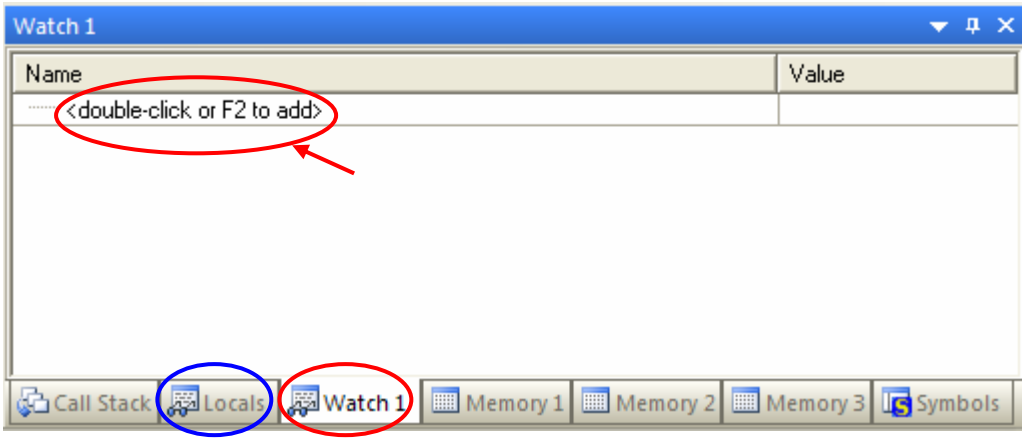


Fig. 5.20: Aspecto de una ventana de observación

La utilidad de la *ventana de observación*, como ya se ha dicho, es que permite agrupar en una sola ventana información parcial –pero fundamental para la depuración– que de otra manera estaría dispersa entre otras muy diversas ventanas: registros, memorias, pila, código y que por tanto resultaría muy incómodo de consultar en la depuración del código.

Cómo añadir un elemento a la ventana de observación

A una ventana de observación hay que dotarla de contenido. Para ello, se pulsa la tecla F2 (o se da una doble pulsación izquierda con el ratón) y en el campo que se abre se teclea el nombre del recurso. Al pulsar *<intro>* automáticamente se añade ese ítem a la lista de elementos en la ventana de observación y se muestra su valor.

Por ejemplo, si se quisiese observar el registro R2, el búfer serie SBUF, la etiqueta TEXTO y las señales VALIDACION y RECONOCI, y el bus BUS_DATOS usadas en el proyecto del ejemplo, se irían tecleando esos nombres, y al validar cada vez con <*intro*> se irían añadiendo a la lista y se mostraría su valor, tal y como se recoge en la imagen siguiente:

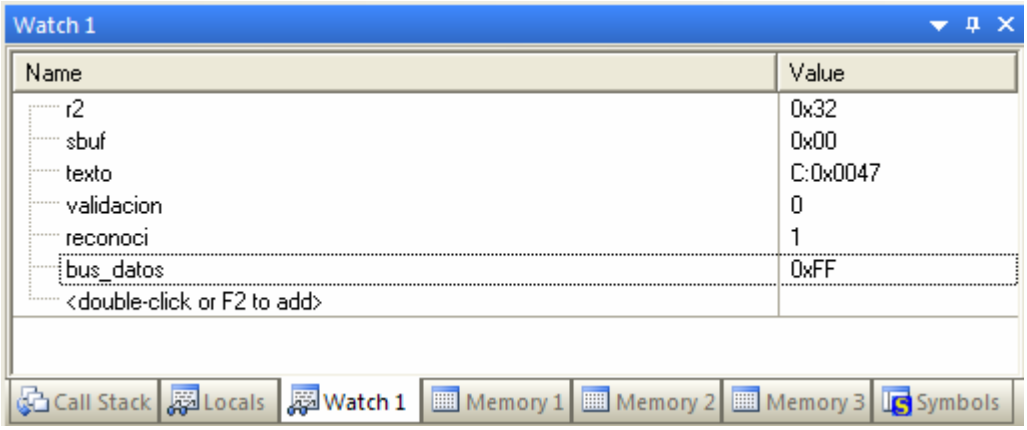


Fig. 5.21: Añadido de elementos a una ventana de observación

Obsérvese que para la etiqueta TEXTO se da su valor (dirección que representa) y no su contenido (que en el ejemplo del proyecto es 50h; es decir en la posición 0047h de la memoria de programa se almacena el octeto de valor 50h).

Cómo eliminar un elemento de la ventana de observación

Si se desea eliminar un ítem de la lista de la ventana de observación, primero hay que seleccionarlo con una doble pulsación izquierda del ratón (se pone con fondo azul), borrarlo (con la tecla de retroceso, ←, no la de *cursor izquierda*) y pulsar entonces <*intro*> (tecla ↵). Otra manera es apuntarlo con el ratón, pulsar el botón derecho y seleccionar en el desplegable la opción *Remove Watch <ítem>*.

Cómo cambiar el sistema numérico de presentación del valor de un elemento

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 que sigue.

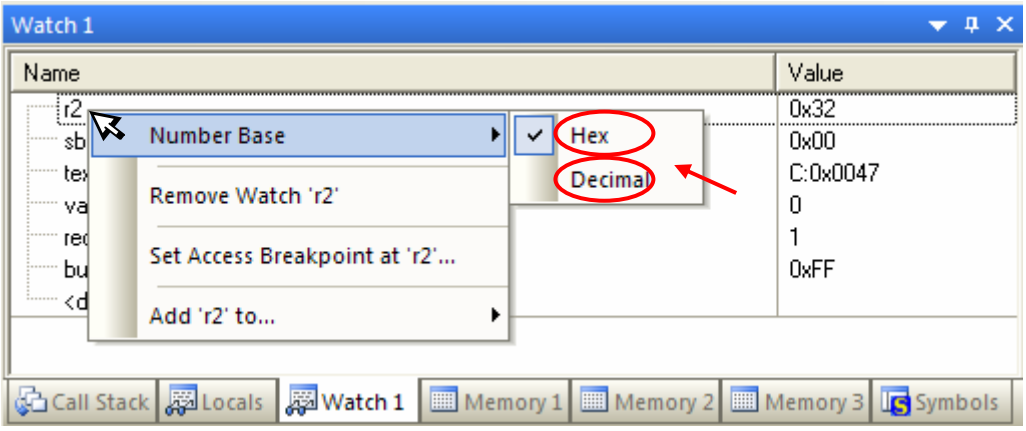


Fig. 5.22: Opciones de sistema de representación de un valor de observación

5.3.4. 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 apunta el elemento y con el botón derecho se despliega un menú específico con las posibles acciones a realizar con él y se selecciona la deseada, ***Modify Memory at*** en este ejemplo:

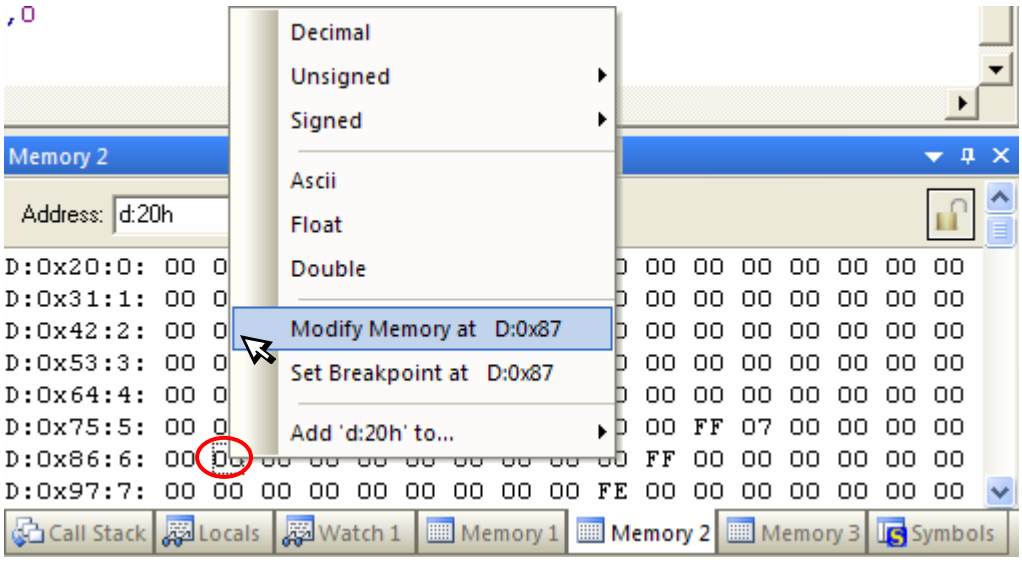


Fig. 5.23: Introducción de valores en una ventana de memoria

En esta figura se ve el menú emergente al abrir con el ratón la posición 87h de la RAM interna. También pueden verse las opciones de visualización de los valores en la ventana de memoria. Si no está seleccionado el modo ***Decimal***, se muestra en hexadecimal (salvo formato incompatible, como la coma flotante). Si los datos en memoria representasen palabras de dieciséis bits, sin signo, se elegiría verlos, por ejemplo, como ***unsigned int***:

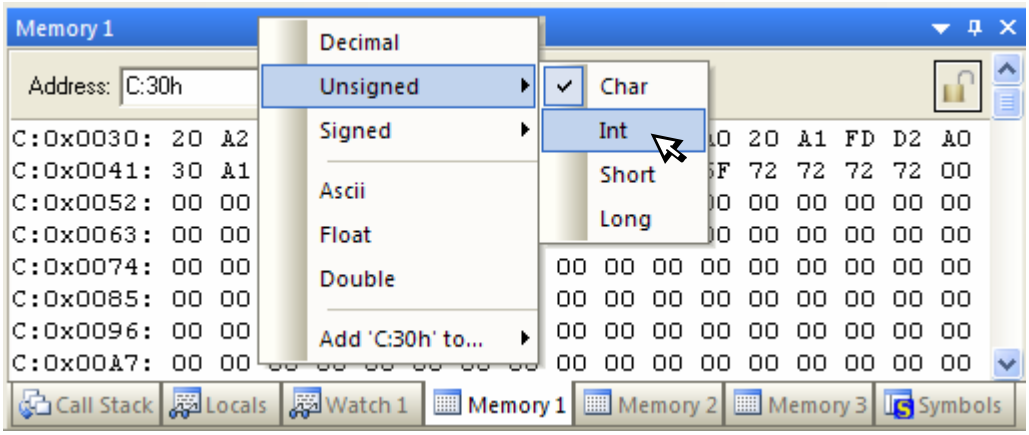


Fig. 5.24: Opciones de formato de visualización de valores

Si se observa la figura de arriba y se compara con la inmediatamente anterior a ella, puede observarse que no aparecen las opciones *Modify Memory at* ni *Set Breakpoint at*. El motivo es que se ha pinchado con el ratón en una zona intermedia entre dos valores, y no justo sobre uno de ellos.

El resultado de hacer lo anterior es lo siguiente:

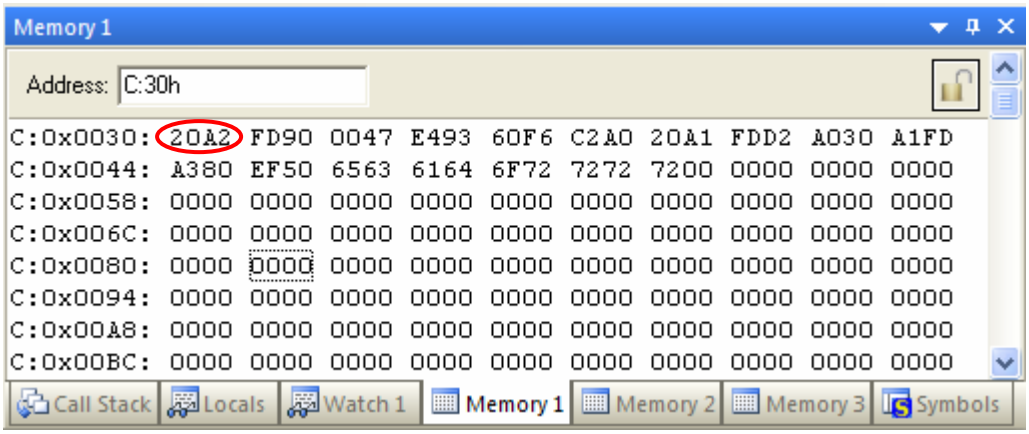


Fig. 5.25: Opciones de formato de visualización de valores enteros de 16 bits

Como puede verse en esta figura, ahora se juntan los valores de dos en dos octetos para tener una noción más clara de cómo se agrupan. No obstante, adviértase que la palabra de dieciséis bits sigue el criterio *octeto alto en memoria baja* y *octeto bajo en memoria alta*. Si el criterio fuese el contrario debería tenerse en cuenta pues, en el ejemplo, en lugar de ser el dato 20A2h sería el A220h

Al seleccionar la opción ***Modify Memory at***, es decir, modificar la memoria en la dirección especificada, 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 esa dada, se teclean los valores separados por comas. Por ejemplo:

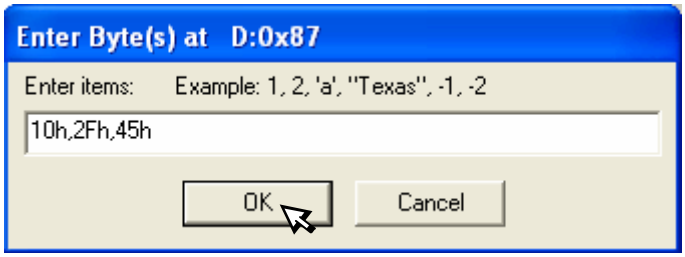
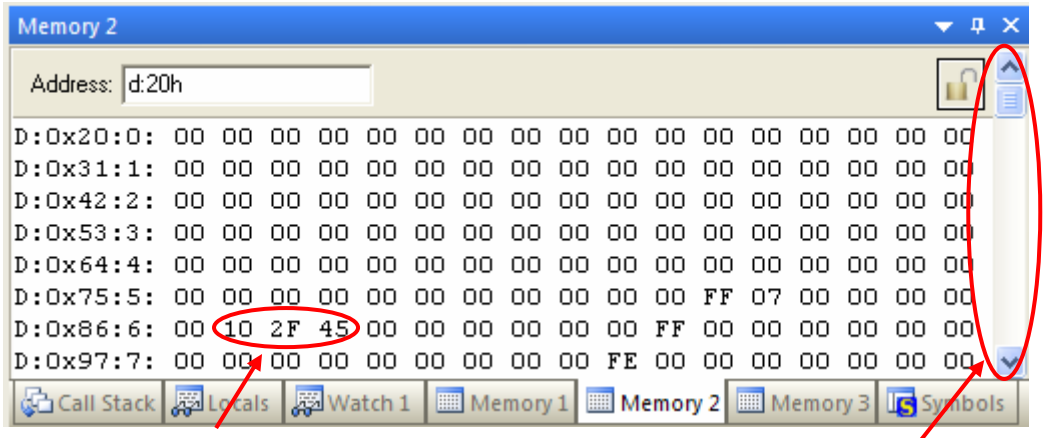


Fig. 5.26: Ventana de introducción de valores numéricos

Aquí se introducirán los valores 10h, 2Fh y 45h en las posiciones 87h, 88h y 89h de la RAM interna. Al validar los nuevos valores, en la ventana de memoria del ejemplo se visualizará lo que se muestra en la siguiente figura.
Si en una ventana no cupiese toda la información visualizable, se disponen de los típicos *ascensores* de una ventana Windows.

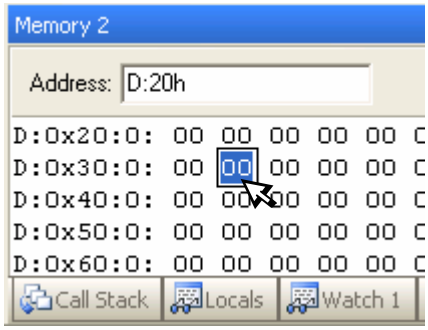


Nuevos valores introducidos

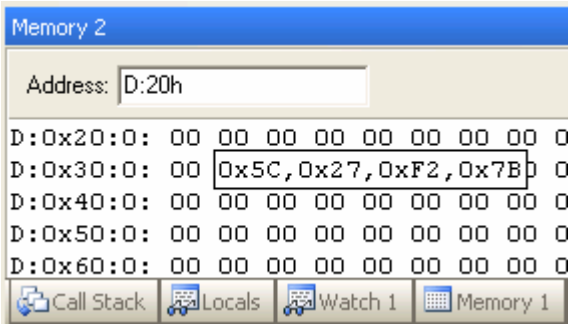
Fig. 5.27: Resultado de la introducción

Ascensor de desplazamiento vertical

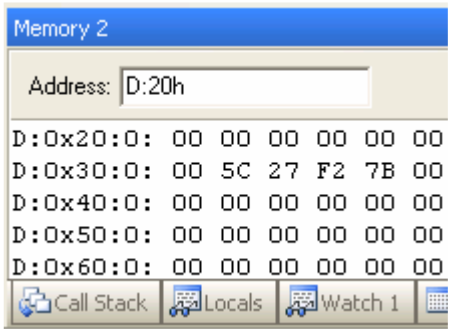
Otra manera de introducir valores es haciendo una doble pulsación izquierda del ratón sobre la posición que se desea modificar o a partir de la que se desea hacerlo (**a**). Se abrirá un campo numérico, y se introducirá el nuevo valor, o los valores sucesivos separados por comas (**b**).



a)



b)



c)

Fig. 5.28:
En estas figuras puede verse el proceso de introducción de los valores 5Ch, 27h, F2h y 7Bh a partir de la dirección 31h de la RAM interna (*figs. a* y *b*) así como el resultado final en la memoria al pulsar finalmente <intro> *(fig. c)*

En las ventanas de memoria, en la esquina superior derecha, pueden verse los símbolos de candado que permiten congelar o no la actualización de estas ventanas. En ocasiones resulta de utilidad el congelar el refresco cuando se está depurando código que accede frecuentemente a la memoria, puesto que la velocidad de simulación se ve afectada negativamente.

Fig. 5.29: Iconos de congelación del refresco de una ventana de memoria

Pulsando con el ratón sobre estos iconos se conmuta su estado.
No debe confundirse esta función de congelación o no de las ventanas de memoria (que sólo afecta a éstas) con la función ***View→Periodic Window Update*** , accesible vía barra de menú y que afecta a **todas** las ventanas del entorno.

5.3.4.1. Código de colores en las ventanas de memoria

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 RAM no inicializada 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

5.3.4.2. Formato numérico de los valores introducidos o mostrados

Como en toda herramienta de desarrollo, se admiten diversos sistemas de numeración a la hora de introducir 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.

Posibilidad de especificar cómodamente valores numéricos de códigos ASCII

Cuando se desea indicar el valor numérico de un carácter **ASCII**, 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 escribir **0x41** ó **41h**. **"Hola**" es equivalente a escribir **'H'**,**'o'**,**'l'**,**'a'** o a escribir **48h**, **4Fh**, **4Ch**, **41h**. Puede advertirse la gran ventaja y comodidad que supone la especificación de valores en ASCII frente a la numérica explícita.

Posibilidad de especificar con seguridad valores binarios de numerosos dígitos
Cuando se considera conveniente introducir un valor en binario pero éste es muy largo, es decir, consta de numerosos dígitos, entonces no es raro que inadvertidamente se cometa un error al omitir inadvertidamente introducir uno o repetir otro. Para evitar esto, se pueden introducir los bits en grupos, separando cada grupo con el carácter $. De esta manera es más fácil comprobar que lo introducido es correcto antes de validarlo.
Por ejemplo, si se hubiese decidido introducir en memoria el dato de dieciséis bits **1101000110101011y**, éste se podría introducir de la siguiente manera:

1101$**0001**$**1010**$**1011**y

Como puede verse, así resulta más sencillo introducir sin temor a equivocarse y no darse cuenta del error.
El número de dígitos de cada grupo de bits es a voluntad. Por ejemplo, el valor anterior podría introducirse como **11010001$10101011y** ó como **11$01000$110$101$011y**.

5.3.4.3. Fijación del sistema de numeración por defecto

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 con *Decimal* desactivado), entonces se puede entrar el valor 7E sin necesidad de explicitar 0x7E ó 7Eh.
Para fijar el sistema de numeración por defecto, se introduce en la ventana de órdenes el comando ***radix= <n>***, siendo ***<n>*** la base del sistema (10T ó 16T). Para ver la base por defecto actual, se introduce el comando ***radix***.
El sistema de numeración por defecto sólo es aplicable a la ventana de órdenes y a los valores introducidos por edición directa sobre un campo, no ante apertura de ventana.

ADVERTENCIA: µVision no verifica la corrección de las expresiones numéricas introducidas, por lo que siempre debe prestarse especial atención.

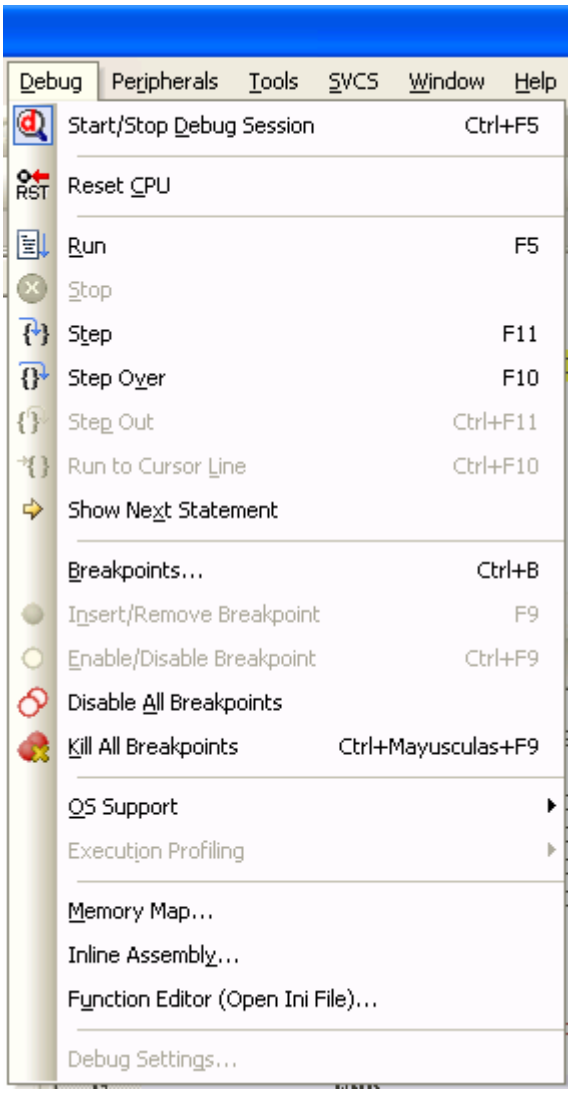
6. DEPURACIÓN DEL CÓDIGO II: COMANDOS PARA LA DEPURACIÓN

6.1. Opciones del menú de 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.

Habiendo manejado ya alguna herramienta de depuración se tiene la base para de manera intuitiva familiarizarse con las peculiaridades de *µVision*. En la siguiente figura pueden verse las distintas opciones:

Fig. 6.1: Opciones de *depuración*



- **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.
- **Inhibir todos los puntos de ruptura** (*Disable All Breakpoints*): Permite inhibir todas las rupturas que puedan existir.
- **Suprimir todos los puntos de ruptura** (*Kill All Breakpoints*): Permite suprimir definitivamente todas las rupturas.
- **Mapa de memoria** (*Memory Map*): Permite analizar o definir la memoria en uso; es decir, la que está efectivamente ocupada por código de programa así como la RAM interna (directa, indirecta y direccionable a bit) y externa de datos de que está dotado el sistema.
- **Ensamblador de línea** (*Inline Assembly*): Permite, en el *modo depuración*, modificar el código de programa. Sólo tiene cierta utilidad para cambios puntuales provisionales. Es recomendable no usar esta opción y en su lugar hacer los cambios en el *modo proyecto*.

6.2. 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:

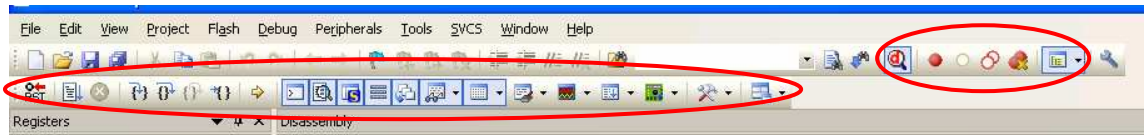


Fig. 6.2: Iconos de depuración en la barra de herramientas

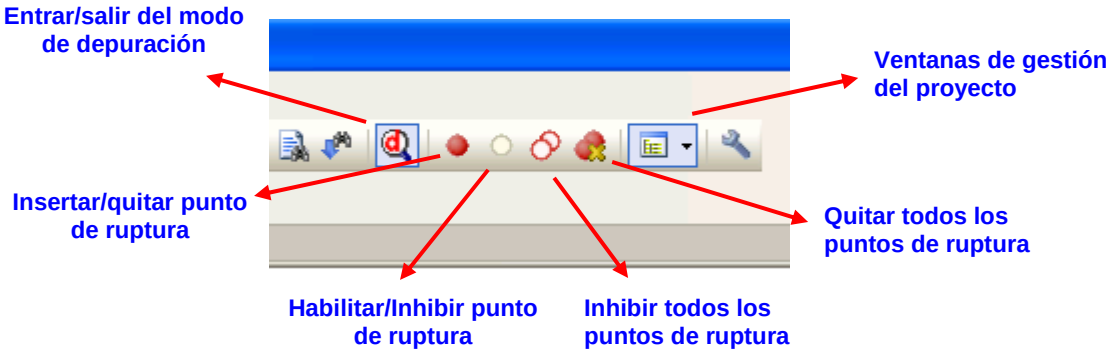


Fig. 6.3: Iconos para trabajar con puntos de ruptura

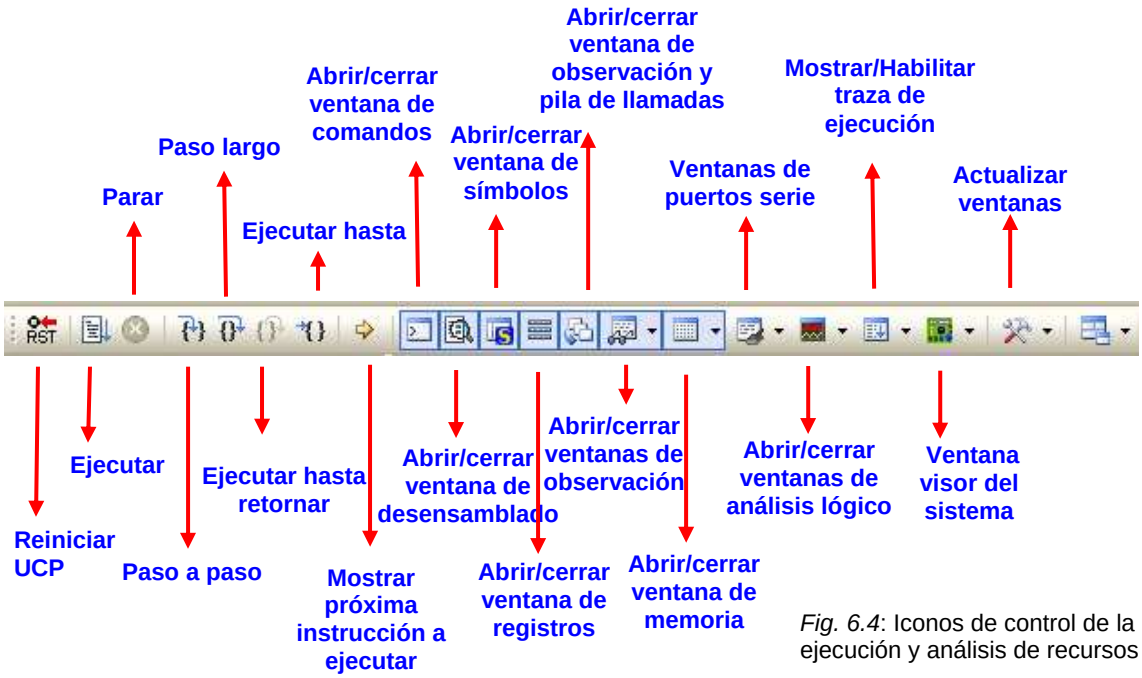
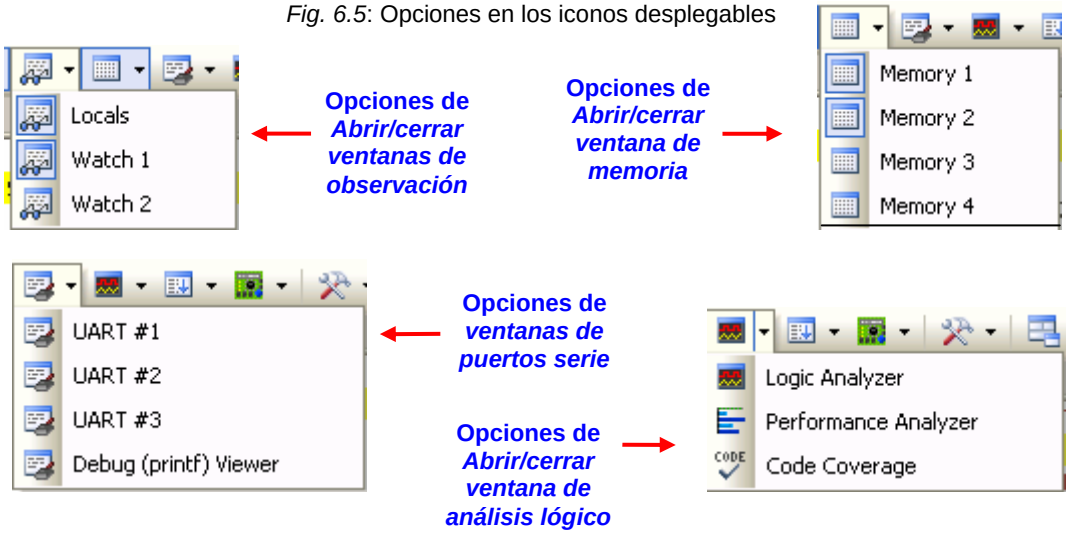


Fig. 6.4: Iconos de control de la ejecución y análisis de recursos

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.
La traza de ejecución se refiere a ver en una ventana la relación 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.

Fig. 6.5: Opciones en los iconos desplegables



6.2.1. Opciones avanzadas de depuración

Aunque algunas de las opciones que siguen requieren el dominio de conceptos y de la metodología de la depuración avanzada, se comentarán brevemente para que el lector se haga una idea somera de su utilidad:

Ventana del analizador lógico
Una vez abierta, su configuración permite visualizar las señales de interés de forma similar a como se verían con un analizador lógico conectado al sistema físico objeto de depuración. En ocasiones esta utilidad puede resultar muy interesante por la facilidad del análisis gráfico y de interrelación temporal entre señales que aporta. No es ni más ni menos que un analizador lógico virtual.

Ventana de análisis de rendimiento
Permite, tras configurar un perfil de ejecución, analizar en qué partes del código se consume más tiempo de ejecución y en cuales menos. Esto es extremadamente útil para, en caso de duda, tomar la decisión de qué partes del código merecen la pena ser optimizadas y cuáles no. Sería absurdo esforzarse en optimizar unas rutinas complejas que se van a ejecutar de tanto en tanto y que, en condiciones no críticas, tendrán un impacto irrelevante en el rendimiento global del sistema.

Cobertura o uso del código
Permite ver para cada subrutina qué tanto por ciento del código que lo compone ha llegado realmente a ejecutarse. Es útil para ver qué partes no han llegado a depurarse y también para identificar código muerto –esto es, que nunca llega a ejecutarse– y ver el impacto relativo de su tamaño con respecto al de la rutina.

Traza de instrucciones (o de ejecución)
Permite observar el rastro lineal de la ejecución de instrucciones por parte del procesador. Puede resultar útil en ciertos casos de depuración en los que interesa analizar un registro de qué es lo que ha hecho el procesador en cierto intervalo de tiempo. No obstante, es bastante farragoso de seguir si la traza de ejecución tiene una gran extensión. Con la práctica se llega a saber acotar razonablemente los intervalos temporales para los que se activa la traza de ejecución.

Fig. 6.6: Opciones en *traza de instrucción*

6.3. Control de la ejecución: *paso a paso* y *ejecución rápida hasta* una línea

Lo habitual es empezar la depuración ejecutando paso a paso. Sólo se ejecutarán de corrido aquellas porciones de código ya depuradas o que se presumen funcionalmente correctas.
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, cuando se ejecuta una instrucción ésta se marca de manera permanente con un código de color en la ventana de edición y en la de desensamblado. Como puede verse en la figura, aparece una **marca de color verde** a la izquierda de la línea, antes de la numeración de ésta; en caso contrario la marca es la **gris**.

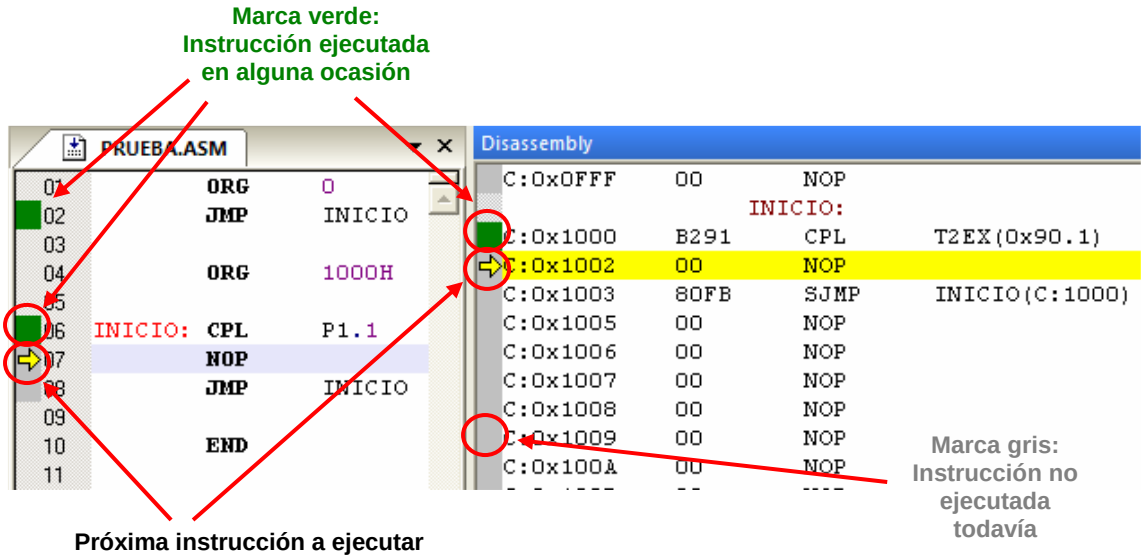


Fig. 6.7: Marcas de control en las ventanas de edición y desensamblado

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**.

La marca de **flecha azul turquesa** significa que con el ratón se ha seleccionado esa línea de código para hacer algo con ella. Para ello o se pincha la línea con el botón izquierdo del ratón o simplemente se apunta y se pulsa el botón derecho para seleccionar en el desplegable la operación a realizar. Esto se ilustra en la figura siguiente (en ella el desplegable se encuentra truncado, existiendo más opciones de las mostradas).

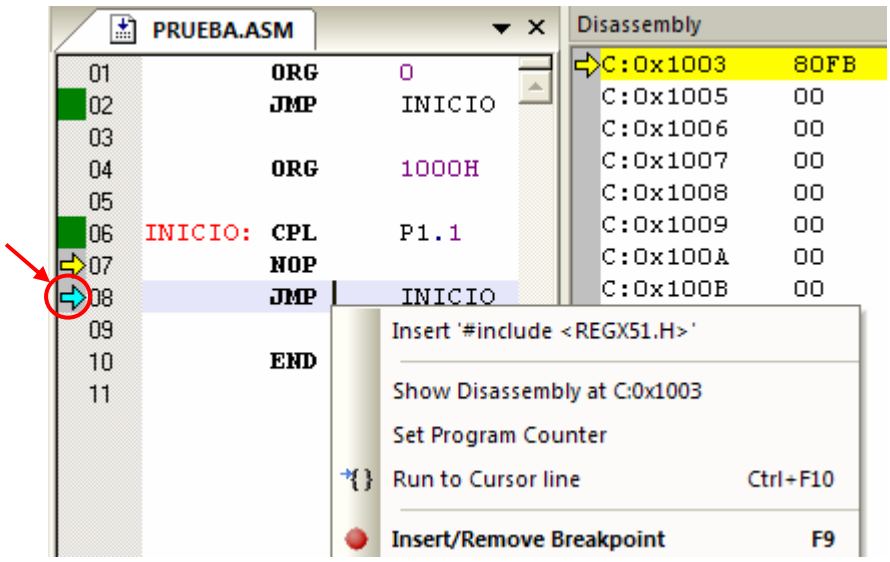


Fig. 6.8: Marca de línea seleccionada en las ventanas de edición y desensamblado

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 el menú de opciones y se selecciona ***Run to Cursor line***.
También se puede hacer con el icono de tal función en la barra de herramientas: 

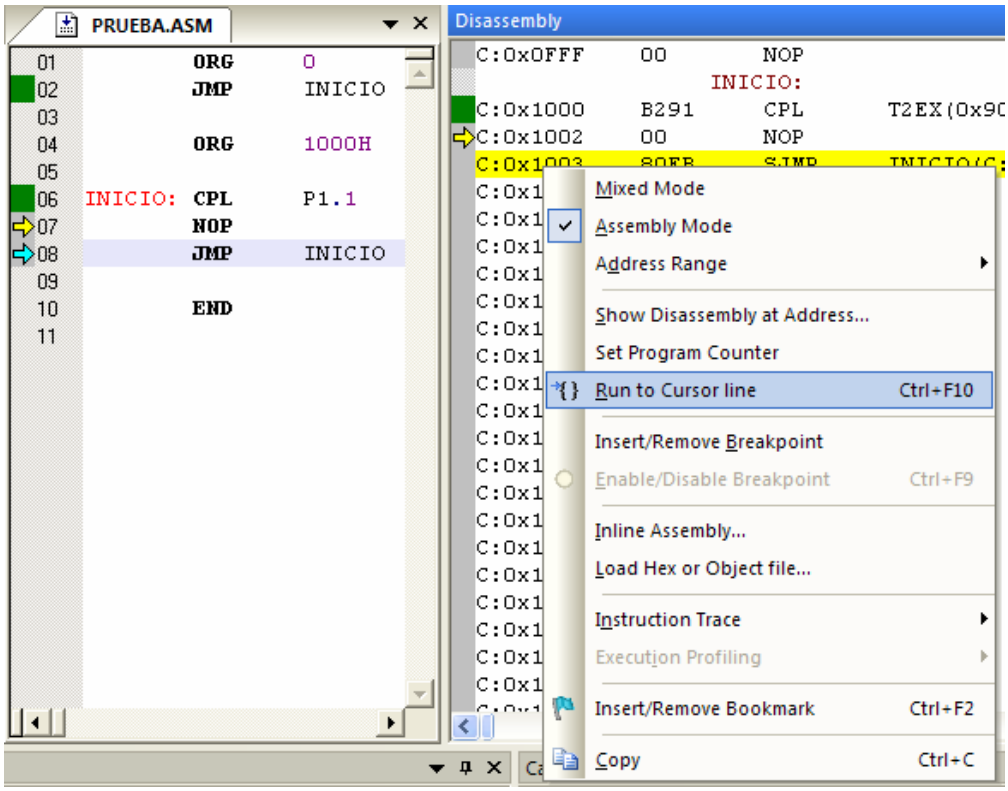


Fig. 6.9: Selección de opción de *ejecutar* hasta, en el desplegable

6.4. El trabajo con los puntos de ruptura

Un punto de ruptura es una *marca simbólica* que se coloca sobre una instrucción o un recurso del sistema (registro interno, registro de un periférico, posición de memoria), con el fin de que el depurador sepa cómo comportarse al ir a simular la ejecución de esa instrucción o si al ejecutarse una instrucción se acceda a ese recurso. Típicamente, se usan los puntos de ruptura para poder ejecutar a velocidad rápida el código del que se tiene la certeza de que funciona correctamente y poder entrar en modo paso a paso a partir del punto en que se tienen dudas del correcto funcionamiento.
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 observarse cuatro puntos de ruptura:

- Uno en la posición 1000h de la memoria de programa; se encuentra habilitada.
- Otro en 1003h de la memoria de programa, que también está habilitada.
- Otro en la posición 56h de la RAM interna, efectiva sólo cuando se escriba por sexta vez en esa posición
- Finalmente, otro 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 se ve, existe una gran flexibilidad a la hora de definir un punto de ruptura.

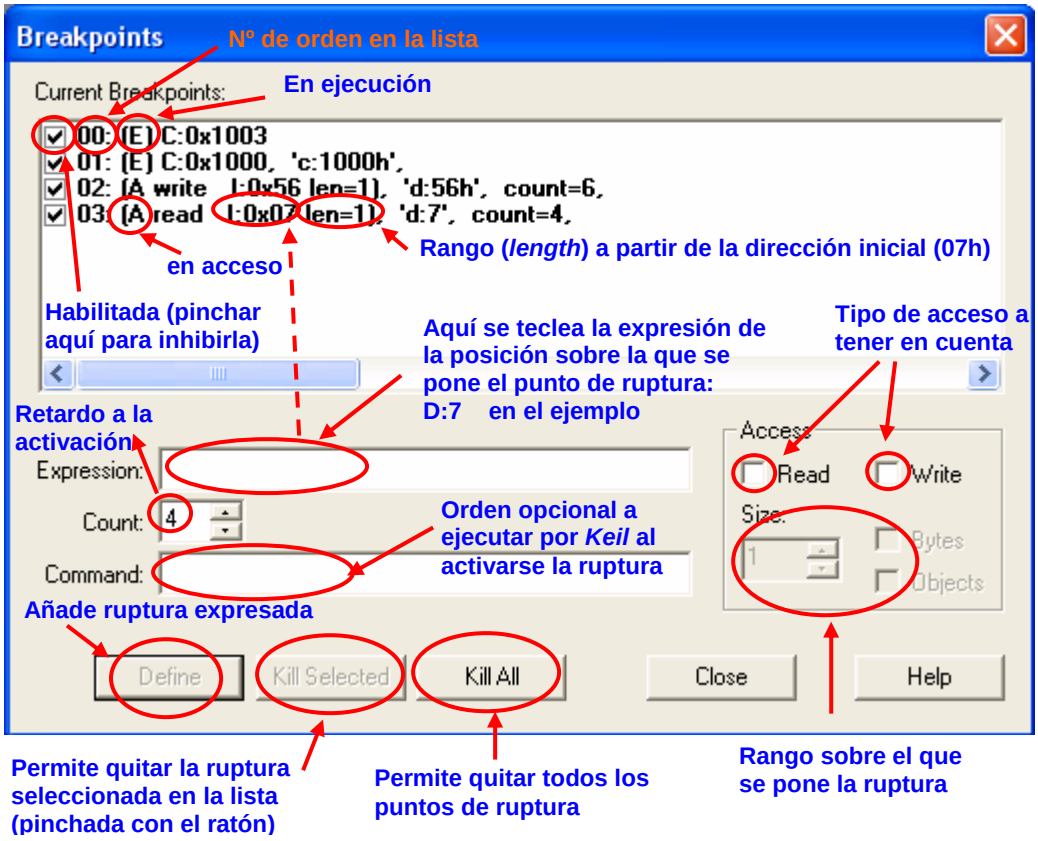


Fig. 6.10: Ventana de puntos de ruptura

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 sólo las hay sobre direcciones individuales).

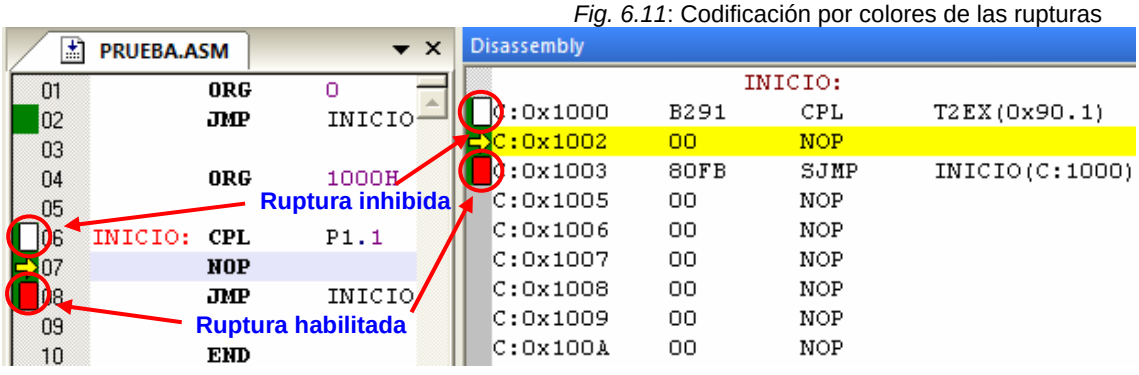
6.4.1. Cómo poner/quitar rápidamente una ruptura

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.

6.4.2. 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:

Fig. 6.11: Codificación por colores de las rupturas



6.5 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 en 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**
Paso largo:	**F10**
Ejecutar hasta retornar:	**CTRL+F11**
Ejecutar hasta:	**CTRL+F10**
Gestionar los puntos de ruptura:	**CTRL+B**
Poner / quitar punto de ruptura:	**F9**
Habilitar / Inhibir rupturas:	**CRTL+F9**
Quitar todas las rupturas:	**CTRL+Mays+F9**

7. FINALIZACIÓN DEL TRABAJO

Al finalizar la depuración, selecciónese ***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***.

No se puede cerrar un proyecto sin salir del modo depuración, salvo que se cierre el entorno *µVision*.

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

8. AYUDA EN LÍNEA

En la barra de menú existe una ayuda en línea que puede consultarse para profundizar o ver con mayor detalle los aspectos de manejo de *Keil uVision 4*.

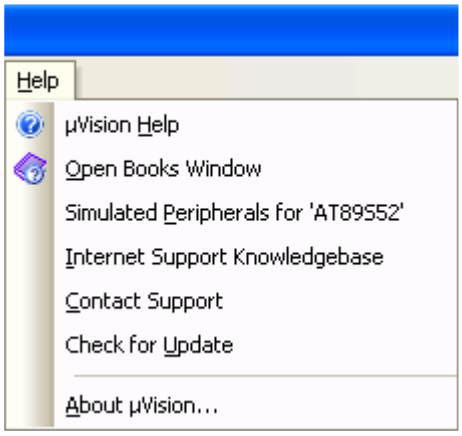


Fig. 8.1: Desplegable con opciones de ayuda

APÉNDICE 1

PASOS PARA CREAR UN PROYECTO, EDITAR EL CÓDIGO FUENTE Y DEPURARLO

1º) Crear el proyecto: **Si ya está creado, abrirlo:**

Project→New Project ***Project→Open Project***

El proyecto se ubicará en la carpeta oportuna (créese ésta si fuese necesario)

2º) Definir las opciones del proyecto, si no se ha hecho antes:

Target 1→Options for Target (con botón secundario del ratón sobre *Target 1)*

En la pestaña ***Target***:
- definir la frecuencia de trabajo (***XTAL (MHz***)
- marcar ***Use On-chip ROM***

En la pestaña ***Output***: ● marcar ***Create HEX file*** (si se va a programar la FLASH)

3º) Crear fichero fuente y añadirlo al proyecto

BARRA DE MENÚ	BARRA DE HERRAMIENTAS	ATAJO CON TECLADO
File→New		**Ctrl+N**

Al guardarlo, désele el nombre oportuno y ubíquese en la carpeta adecuada; será la del proyecto normalmente

4º) Editar el código fuente y guardar el fichero
Si no está abierto el fichero, abrirlo con:

BARRA DE MENÚ	BARRA DE HERRAMIENTAS	ATAJO CON TECLADO
File→Open		**Ctrl+O**

Ubicar el cursor en la ventan de edición, y escribir. Para guardar o cambiar su nombre:

BARRA DE MENÚ	BARRA DE HERRAMIENTAS	ATAJO CON TECLADO
File→Save		**Ctrl+S**
File→Save as…	-------	--------

5º Construir la aplicación: ensamblar y montar

BARRA DE MENÚ	BARRA DE HERRAMIENTAS	ATAJO CON TECLADO
Project→Build target		**F7**

6º) Si hay errores, iterar pasos 4º y 5º. Si no los hay, pasar al modo depuración

BARRA DE MENÚ	BARRA DE HERRAMIENTAS	ATAJO CON TECLADO
Project→Start/Stop Debug Session		**Ctrl+F5**

7º) Configurar el entorno conforme a las necesidades de depuración

- Cerrar ventanas innecesaria
- Abrir ventanas necesarias
- Reformatear y redimensionar las ventanas según se desee
- Definir y dotar de contenido las ventanas de memoria, de observación, etcétera

8º) Comenzar la depuración

Utilizar, según vaya conviniendo, las funciones de:

FUNCIÓN	BARRA DE MENÚ	BARRA DE HERRAMIENTAS	ATAJO CON TECLADO
Ejecución	***Debug→Run***		**F5**
Paso a paso	***Debug→Step***		**F11**
Paso largo	***Debug→Step Over***		**F10**
Ejecutar hasta	***Debug→Run to cursor line***		**Ctrl+F10**
Ejecutar hasta retornar	***Debug→Step Out***		**Ctrl+F11**
Puntos de ruptura	***Debug→Breakpoints***	-------	**Ctrl+B**
Poner / Quitar ruptura	***Debug→Insert/Remove Break.***		**F9**
Habili. / Inhibir ruptura	***Debug→Enable/Disable Break.***		**Ctrl+F9**
Inhibir todas las rupturas	***Debug→Disable All Breakpts.***		**--------**
Eliminar todas las rupturas	***Debug→Kill All Breakpoints***		**Ctrl+Mays+F9**

9º) Si hay errores funcionales, iterar pasos 4º, 5º, 6º, (7º) y 8º.

10º) Salir del modo depuración y cerrar el proyecto

ANOTACIONES:

ANOTACIONES:

ANOTACIONES:

ANOTACIONES:

ÍNDICE DE FIGURAS

5 DEPURACIÓN DEL CÓDIGO I: EL ENTORNO Y SU CONFIGURACIÓN

6 DEPURACIÓN DEL CÓDIGO II: COMANDOS PARA LA DEPURACIÓN

7 FINALIZACIÓN DEL TRABAJO

No hay figuras

8 AYUDA EN LÍNEA

APÉNDICE 1

No hay figuras

HISTORIAL DE REVISIONES

NOMBRE	VERSIÓN	FECHA	DESCRIPCIÓN
P_KuV4_v0.0	0.0	8/3/2010	Versión inicial
P_KuV4_v1.0	1.0	11/3/2010	Corrección de erratas y mejora de edición. Mejora de algunas figuras. Inserción de pies en las figuras. Añadido de apéndice 1, como resumen de pasos. Añadido de índice de figuras.

ÁREA DE ARQUITECTURA Y TECNOLOGÍA DE COMPUTADORES
UNIVERSIDAD DE CÓRDOBA
(ESPAÑA)