第6章 集成开发环境 IDE

6.1 综　述

在这一章中，将详细介绍 μ'nSP™ 集成开发环境。

μ'nSP™ 集成开发环境，它集程序的编辑、编译、链接、调试以及仿真等功能为一体。具有友好的交互界面、下拉菜单、快捷键和快速访问命令列表等，使人们的编程、调试工作更加方便且高效。此外，它的软件仿真功能可以在不连接仿真板的情况下模拟硬件的各项功能来调试程序。

IDE 的开发界面如图 6.1所示。本章将介绍 μ'nSP™开发环境的菜单、窗口界面以及项目的操作等，使有兴趣者对开发环境有一个总体了解，并能够动手实践。

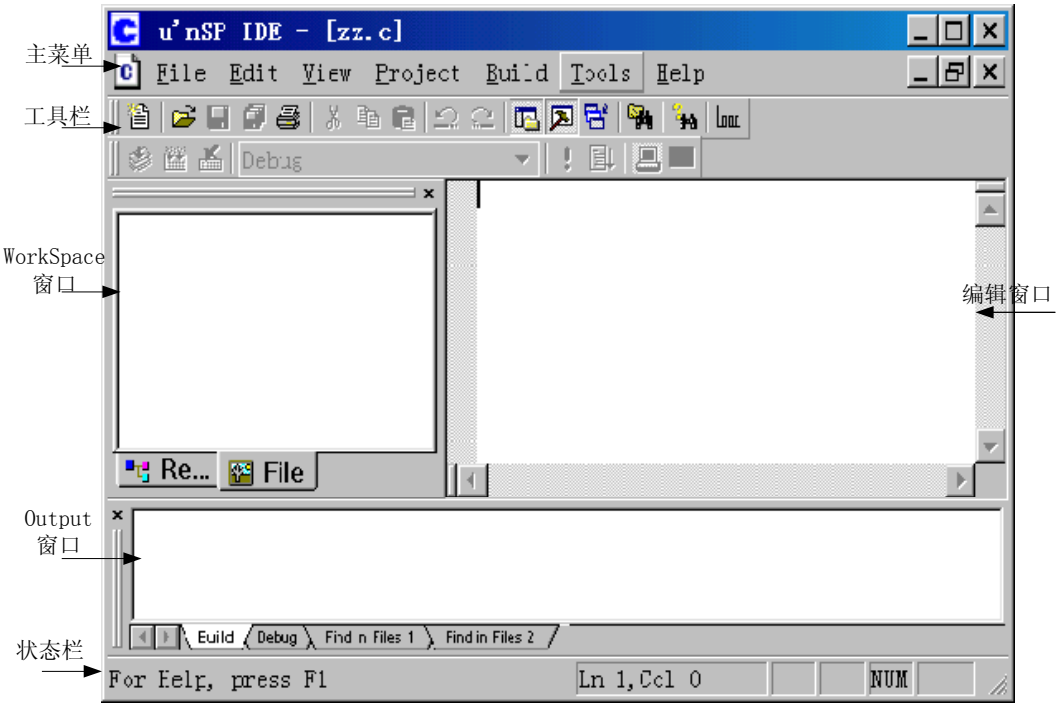


图6.1 μ'nSP™ IDE 桌面

6.2 菜　单

在标题栏的下面是集成环境的主菜单。菜单栏中的菜单命令提供了开发、调试和保存应用程序所需要的工具。μ'nSP™ IDE 菜单栏共有七项，即文件(File)、编辑(Edit)、视图(View)、项目(Project)、编辑(Build)、工具(Tools)和帮助(Help)。每个菜单项含有若干个菜单命令，执行不同的操作，用鼠标单击某个菜单项，即可打开该菜单，然后用鼠标单击菜单中的某一条就能执行相应的菜单命令。

菜单中的命令分为两种类型，一类是可以直接执行的命令，这类命令的后面没有任何信息（例如保存项目）；另一类在命令名后面带省略号（例如打开项目），需要通过打开对话框来执行。在用鼠标单击一条命令后，屏幕上将显示一个对话框，利用对话框可以执行各种有关的操作。在有些命令的后面还带有其它信息，例如：

打开项目　　Ctrl + O

其中 Ctrl + O 叫做“热键”。在菜单中，热键列在相应的菜单命令之后，与菜单命令具有相同的作用。使用热键方式，不必打开菜单就能执行相应的菜单命令。例如：按 Ctrl + O，可以立即执行“打开项目”命令。注意，只有部分菜单命令能通过热键执行。下面介绍菜单栏各项的内容及作用。

6.2.1 文件(File)

文件(File)的下拉菜单内容及功能如表 6.1:

表6.1 文件(File)的下拉菜单内容及功能

内　容	作　用	热　键
新建(New)	新建项目和各种文件	Ctrl + N
打开(Open)	打开项目或各种文件	Ctrl + O
关闭(Close)	关闭文件窗口	
打开项目(Open Project)	用来关闭当前的项目，装入新的项目。执行该命令后，将打开一个对话框，可以在该对话框中，输入要打开项目名称。	
保存项目(Save Project)	保存当前项目及其所有文件。	
关闭项目(Close Project)	关闭当前项目。	
下载程序(Load Program)	将程序下载到仿真板或本机内存中。	
保存(Save)	保存当前的文件	Ctrl + S
另存(Save As)	用于改变存盘文件的名称。执行该命令后，将弹出一个对话框，可以在这个对话框中输入存盘的文件名。	
全部保存(Save All)	保存目前所有的文件和项目	
打印设置(Print　Setup)	在执行该命令后，将显示标准的“打印设置”对话框，在该对话框中设置打印机、页面方向、页面大小、纸张来源以及其它打印选项。	
打印(Print……)	把窗体及代码在由 Windows 设定的打印机打印出来。	Ctrl + P
近期文件(Recent　File)	打开最近使用的 10 个文件,主要是方便开发者在最短的时间内找到并打开所需的文件。	

续

内　容	作　用	热　键
近期项目(Recent Project)	打开最近使用的 10 个项目,主要是方便开发者在最短的时间内找到并打开所需的项目。	
退出(Exit)	退出开发环境	

文件（File）的下拉菜单界面如图 6.2：

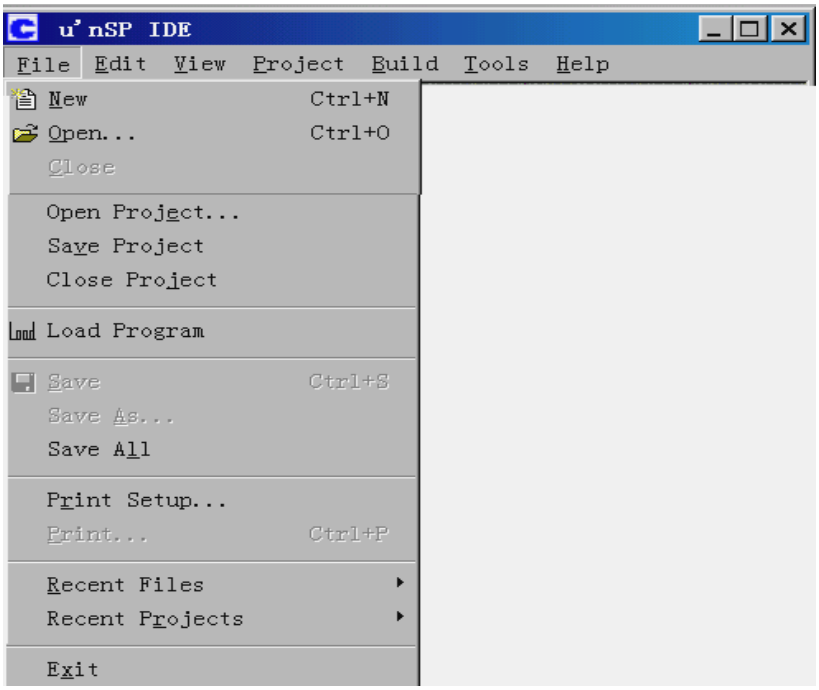


图6.2 文件下拉菜单界面

6.2.2 编辑(Edit)

编辑(Edit)的下拉菜单内容及功能如表 6.2:

表6.2 编辑(Edit)的下拉菜单内容及功能

内　容	作　用	热　键
撤消键入（Undo）	取消最近的编辑操作。	Ctrl + Z
重复键入（Redo）	恢复撤消键入之前的编辑内容。	Ctrl + U
剪切（Cut）	删除选中的文件内容或文件，可以复制。	Ctrl + X
复制（Copy）	拷贝选中的文件内容或文件	Ctrl + C
粘贴（Paste）	粘贴到指定的位置	Ctrl + V

内　容	作　用	热　键
删除（Delete）	删除选中的文件内容或文件	Del
全选（Select All）	选中所有的文件内容或文件	Ctrl + A
查找（Find…）	查找文件内容或文件	Ctrl + F
在指定文件内查找（Find in File）	在指定文件内查找文件内容或文件	
查找下一个（Find Next）	用来查找并选择在“查找”对话框的“查找内容”框中指定的文本的下一次出现位置	F3
查找前一个（Find Previous）	用来查找并选择在“查找”对话框的“查找内容”框中指定的文本的上一次出现位置	Ctrl + F3
替换（Replace…）	替换指定的文本，执行该命令后，将显示一个对话框，在对话框的两个栏内分别输入要查找的文本和替换文本，即可一个一个的替换或一次全部替换。	Ctrl + H
定位（Go to…）	定位到某一行或列	Ctrl + G
标记（Bookmark）	在指定的位置设置标记	Alt + F2
下一个标记（NextBookmark）	光标指到下一个标记处	F2
前一个标记（Previous）	光标指到前一个标记处	Ctrl + F2
清除所有标记（Clear All Bookmark）	清除文件内所有标记	Shift + F2
断点（Breakpoints）	设置光标所在处为断点	Alt + F9
外部编译器（External Editor）	目前基本不用	Ctrl + E

编辑(Edit)的下拉菜单界面如图 6.3：

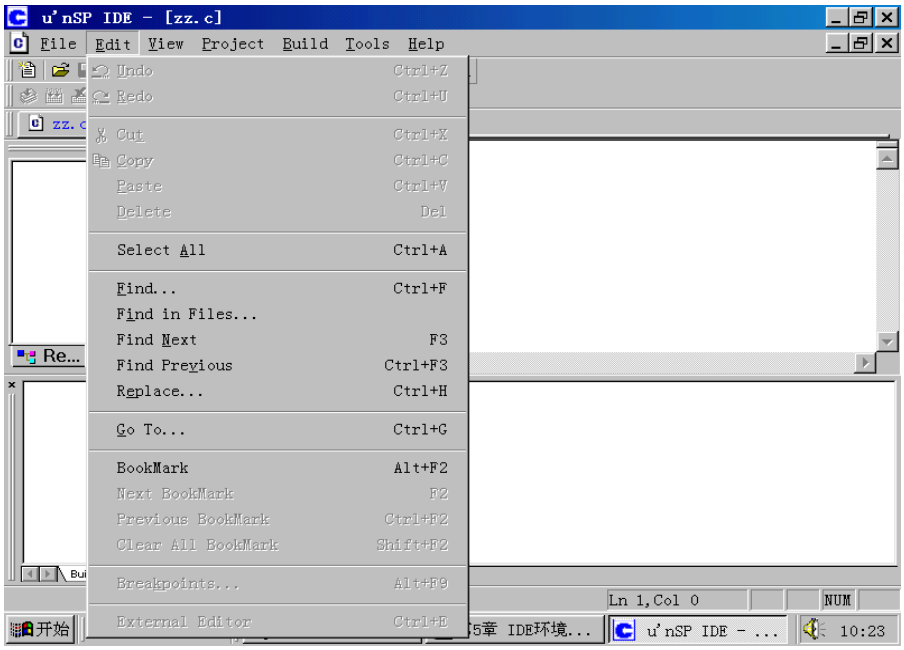


图6.3 编辑下拉菜单界面

6.2.3 视图(View)

视图(View)的下拉菜单内容及功能如表 6.3:

表6.3 视图(View)的下拉菜单内容及功能

内　容	作　用	热　键
全屏（Full Screen）	编辑窗口为全屏	
工作区（Workspace）	单击后，弹出 Workspace 窗口。	Alt + 0
输出（Output）	单击后，弹出 Output 窗口。	Alt + 1
命令（Command）	单击后，弹出 Command 窗口	Alt + 2
调试窗口 （Debug Windows）	调试时使用。其包括： 1）变量表 Watch 窗口； 2）寄存器 Register 窗口； 3）内存 Memory 窗口 4）反汇编窗口 Disassemble 窗口	 Alt + 3 Alt + 4 Alt + 5 Alt + 6
常用工具栏 （Main Toolbar）	包括：新建、打开、保存、全存、打印、剪切、复制、粘贴、查找、撤消等工具。	
编辑工具栏（Build Toolbar）	包括： 编译、编辑、停止编辑、运行、下载、本机仿真、连接仿真板调试。	
调试工具栏（Debug Toolbar）	包括：运行、下载、中断、停止调试、重新开始、单步执行、各调试窗口。	
文件标签栏（FileTabs Bar）	文件标签用于显示编辑窗口打开的文件名称。	
状态栏（Status Bar）	提示光标所在的行、列数。	
属性（Properties）	所选中的文件属性	

视图（View）的下拉菜单界面如图 6.4：

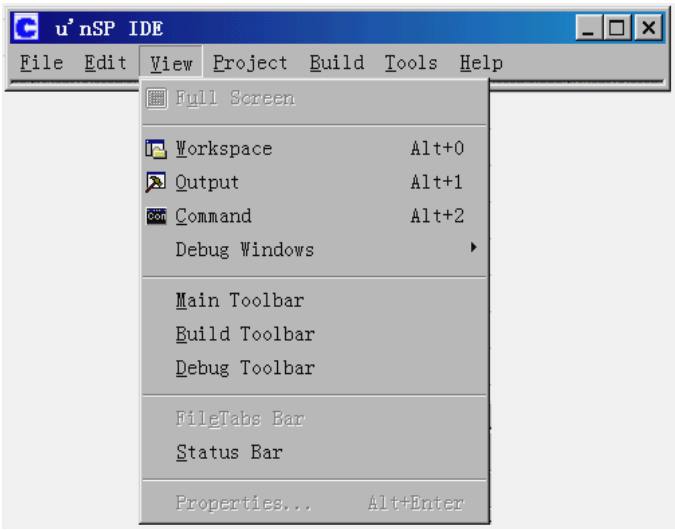


图6.4 视图下拉菜单界面

6.2.4 项目(Project)

项目(Project)的下拉菜单内容及功能如表 6.4:

表6.4 项目(Project)的下拉菜单内容及功能

内　容	作　用	热键
加到项目（Add to Project）	包括：向项目中加源文件和资源文件。	
项目选项设置(Setting…)	包括：General、Option、Link、Section、Hardware、Device 属性页设置。(后有描述)	Alt + F7

项目(Project)的下拉菜单界面图 6.5：

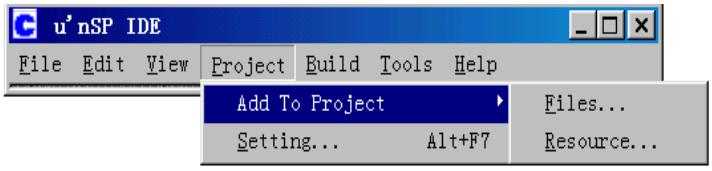


图6.5 项目下拉菜单界面

6.2.5 编译（Build）

编辑（Build）的下拉菜单内容及功能如表 6.5:

表6.5 编辑（Build）的下拉菜单内容及功能

内　容	作　用	热 键
编译（Compile）	编译目前文件	Ctrl + F7
编辑（Build）	编译后链接文件	F7
停止编辑（Stop Build）	停止编辑目前文件	Ctrl + Break
编辑所有文件（Build All）	编辑该项目中的所有文件	
清除（Clean）	清除刚编辑过的文件	
开始调试（Start Debug）	调试刚编辑过的文件包括：下载、单步调试等	
执行（Execute）	运行文件	
分析（Profile…）	详细分析软件执行效率	

编译（Build）的下拉菜单界面如图 6.6：

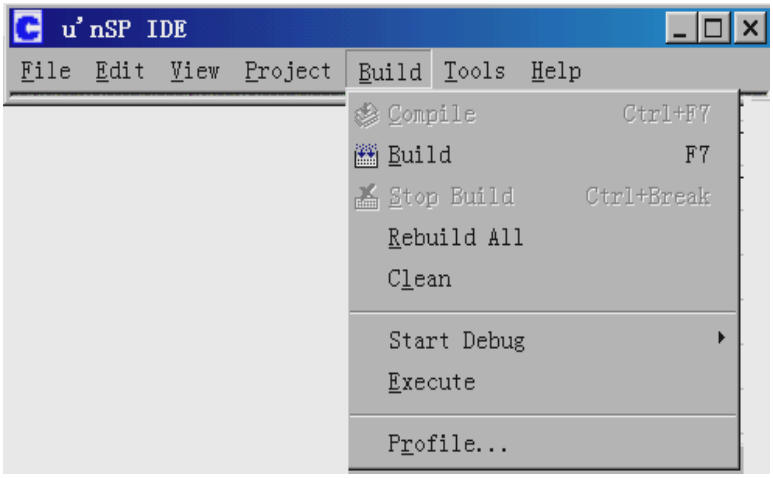


图6.6 编译下拉菜单界面

6.2.6 工具（Tools）

工具（Tools）的下拉菜单内容及功能如表 6.6

表6.6 工具（Tools）的下拉菜单内容及功能

内　容	作　用
选项（Option…）	包括：编辑窗口格式设置、库文件的路径设置
制作库文件（Lib Maker）	将所需的 Obj 文件转换成库文件，方便开发时用。

工具（Tools）的下拉菜单界面如图 6.7：

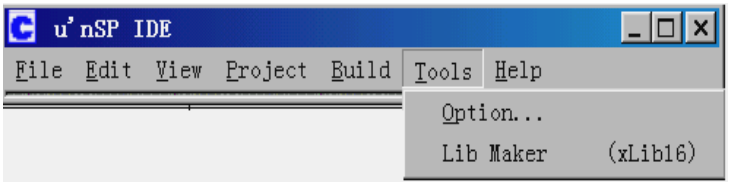


图6.7 工具下拉菜单界面

6.2.7 帮助（Help）

帮助（Help）的下拉菜单内容及功能如表 6.7:

表6.7 帮助（Help）的下拉菜单内容及功能

内　容	作　用
帮助主题（Help Topics	介绍 IDE 环境
关于 IDE（About IDE）	IDE 的版本号、开发公司、所占空间。

帮助（Help）的下拉菜单的界面如图 6.8：

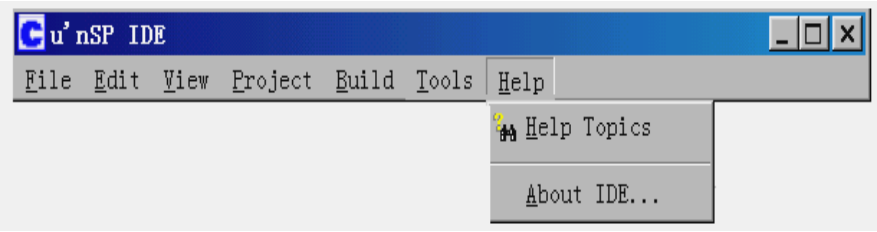


图6.8 帮助下拉菜单界面

6.2.8 调试（Debug）

在调试模式下，菜单栏中多出一个调试菜单。
调试的下拉菜单界面如图 6.9：

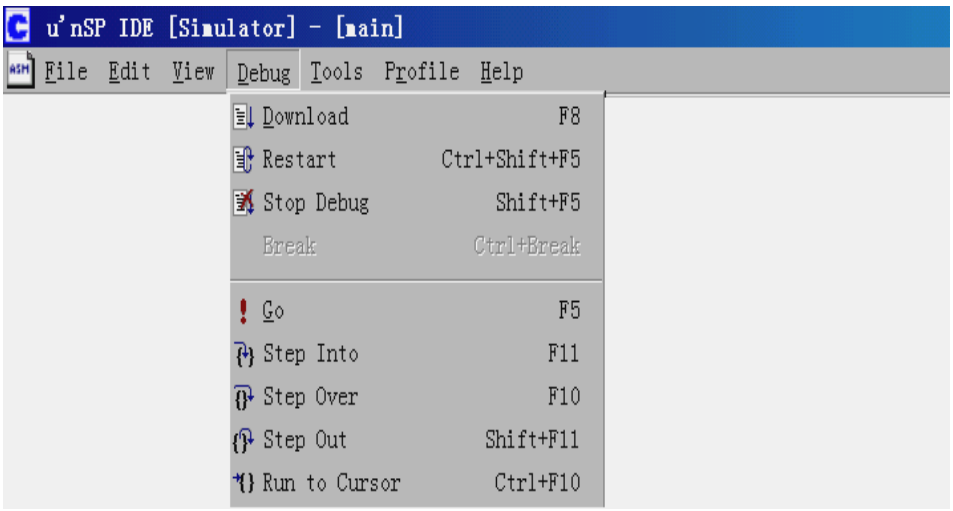


图6.9 调试下拉菜单界面

调试（Debug）的下拉菜单内容及功能如表 6.8:

表6.8 调试（Debug）的下拉菜单内容及功能

内　容	作　用	热键
下载(Download)	将程序文件编译连接生成可执行文件	F8
复位（重新开始）(Restart)	在调试模式下，重新运行程序。	Ctrl+Shift+F5
停止调试(Stop Debug)	退出调试模式。	Shift+F5
中断(Break)	停止程序运行	Ctrl+ Break
运行(Go)	在调试模式下,运行程序。	F5
单步进入(Step Into)	单步运行时，进入子程序。	F11
(Step Over)	单步运行时，不进入子程序。	F10
单步跳出(Step Out)	单步运行在子程序中时，跳出子程序。	Shift+F11
运行到光标处(Run to Cursor)	在调试模式下，程序全速运行到光标处停止。	Ctrl+ F10

6.3 工具栏

μ'nSP™ IDE 提供了 3 种工具栏，包括标准、编辑和调试。每种工具栏都有固定和浮动两种形式。把鼠标移到固定形式工具栏中没有图标的地方，按住左按钮，向下拖动鼠标，即可把工具栏变为浮动的；而双击浮动工具栏的标题条，则可变为固定工具栏。

固定形式的标准工具栏位于菜单栏的下面，它以图标的形式提供了部分常用菜单

命令的功能。只要用鼠标单击代表某个命令的图标按钮，就能直接执行相应的菜单命令。工具条中有 38 个图标，代表 38 种操作，如图 6.10所示。大多数图标都有与之等价的菜单命令。图 6.11到图 6.13是浮动形式的标准、编辑和调试工具栏。表 6.9列出了工具栏中各图标的作用。

图6.10　工具栏

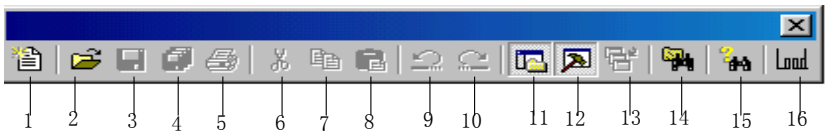


图6.11　标准工具栏

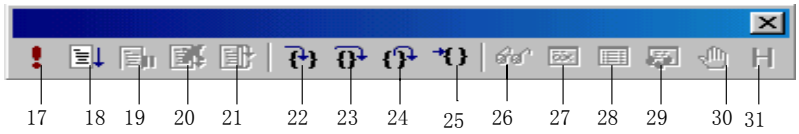


图6.12　调试工具栏

图6.13　编辑工具栏

表6.9　工具栏一览表

编号	名称	作用
1	新建	新建项目和文件，相当于 File 菜单中的 New 命令
2	打开	打开项目和文件，相当于 File 菜单中的 Open 命令
3	保存	保存文件，相当于 File 菜单中的 Save 命令
4	全存	保存所有文档，相当于 File 菜单中的 Save All 命令
5	打印	打印当前文件,相当于 File 菜单中的 Print 命令
6	剪切	删除并复制选中的文件内容或文件，相当于 Edit 菜单中的 Cut 命令
7	复制	复制选中的文件内容或文件,相当于 Edit 菜单中的 Copy 命令
8	粘贴	粘贴选中的文件内容或文件,相当于 Edit 菜单中的 Paste 命令
9	撤消键入	取消当前的操作

续

编号	名称	作用
10	重复键入	对撤消的反操作
11	Workspace 窗口	打开或关闭 Workspace 窗口,相当于 View 菜单中的 Workspace 命令
12	Output 窗口	打开或关闭 Output 窗口,相当于 View 菜单中的 Output 命令
13	窗体布局窗口	打开窗体布局窗口。
14	在文件中查找	打开“在文件中查找”对话框，相当于 File 菜单中的 Find in File 命令
15	帮助主题	打开“帮助主题”窗口，相当于 Help 菜单中的 Help Topics 命令。
16	打开可执行文件	打开可执行文件（.s37 或.tsk）
17	运行	在调试模式下,运行程序。相当于 Debug 菜单中的 Go 命令。
18	下载	下载可执行文件。相当于 Debug 菜单中的 Download 命令。
19	中断	停止正在运行程序。相当于 Debug 菜单中的 Break 命令。
20	停止调试	退出调试模式。相当于 Debug 菜单中的 Stop Debug 命令。
21	重新开始（复位）	在调试模式下，重新运行程序。
22	单步进入	单步运行时，进入子程序。相当于 Debug 菜单中的 Step Into 命令。
23	Step Over	单步运行时,不进入子程序。相当于 Debug 菜单中的 Step Over 命令。
24	单步跳出	单步运行在子程序中时，跳出子程序。相当于 Debug 菜单中的 Step Out 命令。
25	运行到光标处	在调试模式下，程序全速运行到光标处停止。相当于 Debug 菜单中的 Run to Cursor 命令。
26	变量表窗口	在调试模式下，打开变量表窗口。相当于 View 菜单中的 Watch 命令。
27	寄存器窗口	在调试模式下，打开寄存器窗口。相当于 View 菜单中的 Registers 命令。
28	内存窗口	在调试模式下，打开内存窗口。相当于 View 菜单中的 Memory 命令。
29	反汇编窗口	在调试模式下，打开反汇编窗口。相当于 View 菜单中的 Disassembly 命令。
30	断点	在调试模式下，打开设置断点的对话框。相当于 Edit 菜单中的 Breakpoints 命令。
31	历史缓冲区	在仿真模式下，打开历史缓冲区窗口。
32	编译	编译文件。相当于 Build 菜单中的 Compile 命令。
33	编辑	编辑文件。相当于 Build 菜单中的 Build 命令。
34	停止编辑	停止编辑文件。相当于 Build 菜单中的 Stop Build 命令。
35	运行	在调试模式下,运行程序。相当于 Debug 菜单中的 Go 命令。
36	下载	下载可执行文件。相当于 Debug 菜单中的 Download 命令。
37	本机调试（使用仿真器）	在本机上调试。
38	仿真板上调试（使用在线仿真）	结合仿真板调试。

6.4 窗　口

前面介绍了标题栏、菜单栏和工具栏，它们所在的窗口称为主窗口，实际上，除主窗口外，μ'nSP™ IDE 编程环境中还有其它一些窗口，如下：

1．Workspace 窗口
2．Edit 窗口
3．Output 窗口
4．Debug 窗口
　1）变量表 Watch 窗口
　2）寄存器 Register 窗口
　3）内存 Memory 窗口
　4）反汇编窗口 Disassemble 窗口
　5）历史缓冲区窗口
5．其它窗口
　1）令窗口
　2）转存窗口

这一节主要介绍这些窗口。

6.4.1 Workspace 窗口

在 Workspace 窗口中，含有建立一个应用程序所需要的文件清单。其中包括所有的与该项目相关资源文件（如语音数据等）和被编辑的程序文件。我们可用视窗标签来切换显示 File, Resource 两个视窗。

File 视窗主要用来显示源文件组和头文件组中所包含的所有文件。

Resource 视窗主要用来显示资源文件组中所包含的所有资源文件。

1．打开 Workspace 窗口方法

第一种方法：单击菜单栏 View/Workspace 菜单命令即可打开/关闭 Workspace 窗口。

第二种方法：单击标准工具栏中的 Toggle Workspace 按钮，也可打开/关闭 Workspace 窗口。

图 6.14为 Workspace 窗口界面：

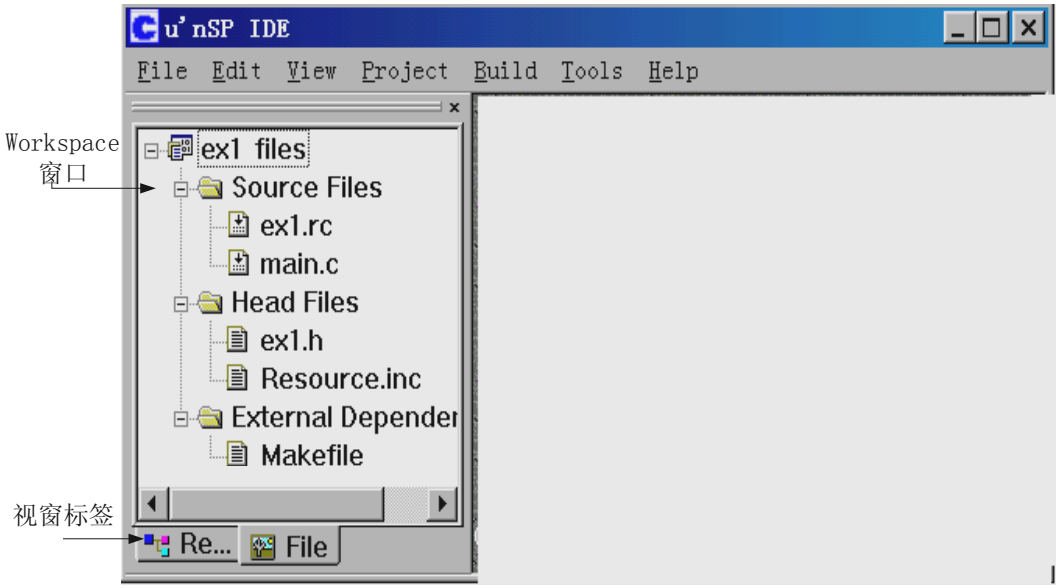


图6.14 Workspace 窗口界面

通过对 Workspace 窗口中 Resource 和 File 标签的点击可以切换 File 视窗和 Resource 视窗。

图 6.15是 Workspace 窗口下的 File 视窗界面：

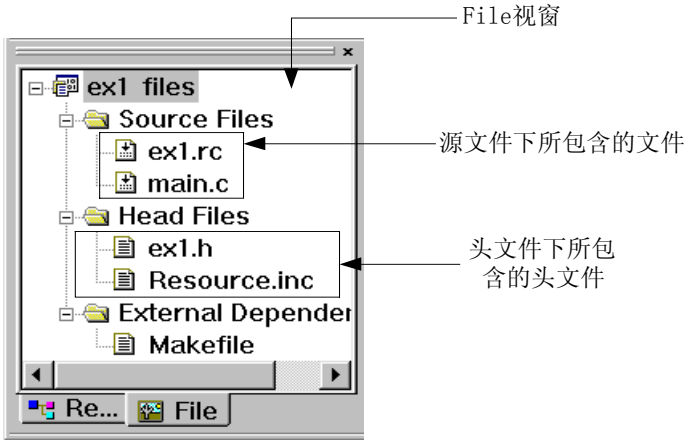


图6.15 Workspace 窗口 File 视窗界面

图 6.16是 Workspace 窗口下的 Resource 视窗界面：

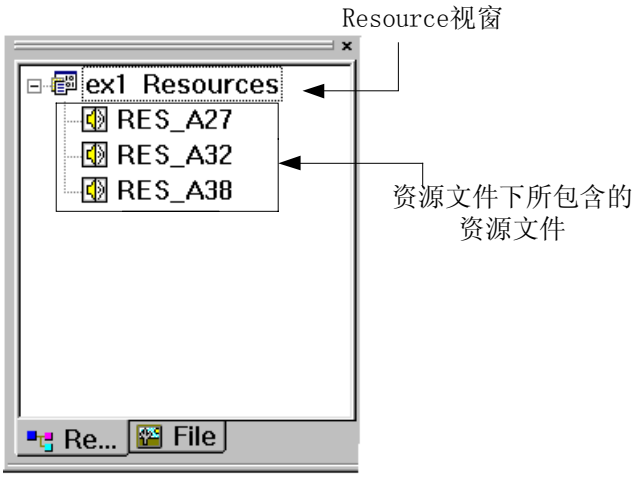


图6.16 Workspace 窗口 Resource 视窗界面

图 6.16 Workspace 窗口 Resource 视窗界面中的资源文件 RES_A32、RES_A38、RES_A27 为 A2000 格式的语音数据文件。

6.4.2 编辑窗口（Edit 窗口）

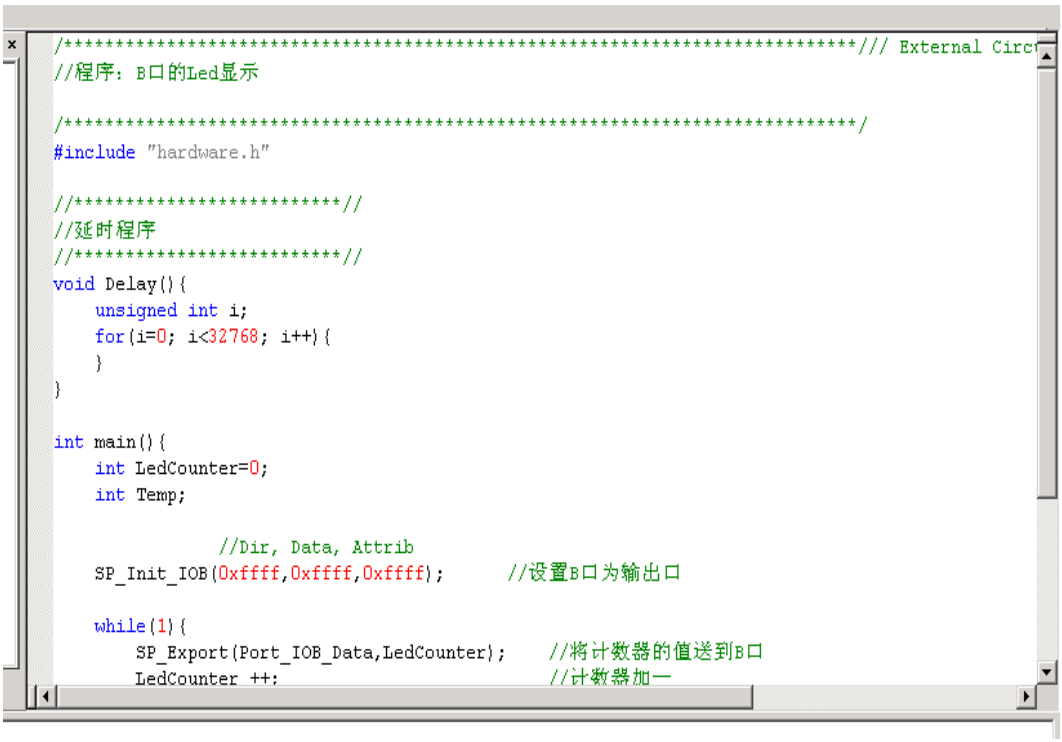
在前面介绍桌面时已经指出编辑窗口的位置，编辑窗口主要是用来键入程序文件和其它编辑文件的显示。

打开编辑窗口的方法：
新建任何一文件，即可打开编辑窗口。
例如：单击 File / New / Creat C File 打开该 C 文件的编辑窗口。

编辑窗口包括文本编辑器和二进制编辑器。

6.4.3 文本编辑器

文本编辑器用来编辑程序的。当您在项目中打开一个文件时，文件所有的内容都显示在文本编辑器中。
图 6.17就是文本编辑器的界面：



```
/*********************************************************************************/// External Circ
//程序：B口的Led显示

/*********************************************************************************/
#include "hardware.h"

//*************************//
//延时程序
//*************************//
void Delay(){
    unsigned int i;
    for(i=0; i<32768; i++){
    }
}

int main(){
    int LedCounter=0;
    int Temp;

                //Dir, Data, Attrib
    SP_Init_IOB(0xffff,0xffff,0xffff);      //设置B口为输出口

    while(1){
        SP_Export(Port_IOB_Data,LedCounter);    //将计数器的值送到B口
        LedCounter ++;                          //计数器加一
```

图6.17 文本编辑器的界面

打开文本文件的方法：

- 单击 File/Open,弹出 Open 对话框。选择一文件。
- 单击 File/Recent Files ,选择一近期文件立即打开。

6.4.4 二进制编辑器

二进制编辑器用来编辑项目中十六进制或 ASCII 格式的二进制代码的资源文件。如图 6.18为二进制编辑器界面。

打开二进制文件的步骤：

第一步：单击 File/Open,弹出 Open 对话框。

第二步：在 Open as 文本框中选择 Binary 。

第三步：选择一个二进制文件、打开。

编辑二进制编辑器：

- 单击选中将修改二进制文件内容，按数字键可以更改二进制文件内容。
- 保存修改后的内容。
- 在二进制编辑器中，有效键为 [↓ / ↑][← / →][page up][page down][Home/End][Contrl + Home][Contrl + End] 。

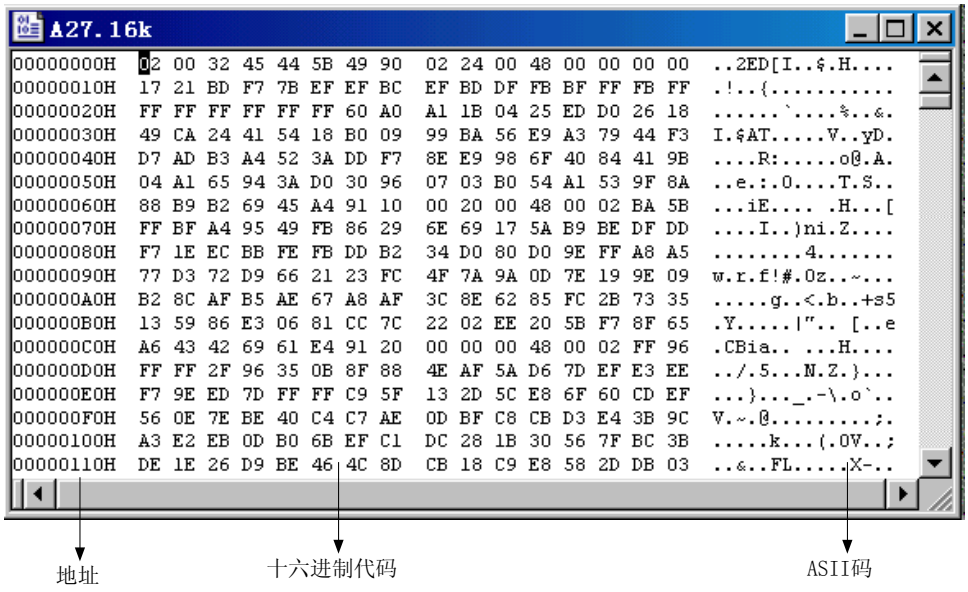


图6.18 二进制编辑器

6.4.5 输出窗口（Output 窗口）

输出窗口主要用来显示编辑、调试、查找的输出结果。

打开输出窗口的方法：

第一种方法：单击菜单栏 View / Output 菜单命令即可打开/关闭 Output 窗口。

第二种方法：单击标准工具栏中的 Toggle Output 按钮，也可打开/关闭 Output 窗口。

输出窗口界面图 6.19：

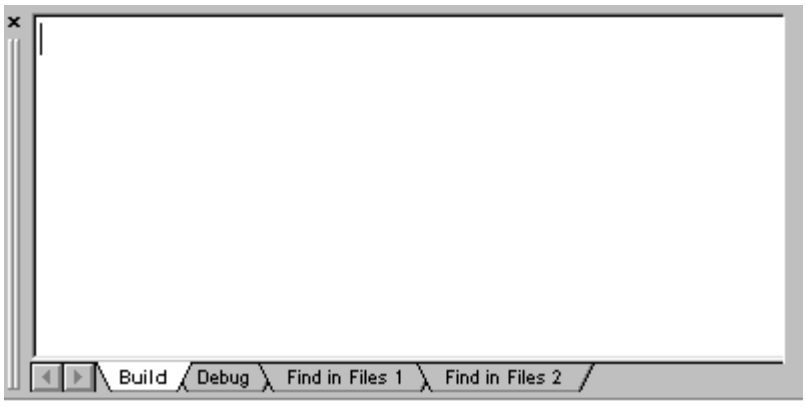


图6.19 输出窗口界面

输出窗口还可以细分为编辑输出窗口、调试输出窗口、查找输出窗口三种。这三种输出窗口可以通过输出窗口底部的标签切换。

6.4.6 编译输出窗口（Build）

编译后，在输出窗口显示出编译、连接的信息。在编译过程中，错误和警告信息也会被列出。当输出窗口无错误信息时，说明该程序已完全成功地编译。

举例： 编译 IDE 下 Example 中的 Ex2。
编译后的输出窗口编辑信息界面图 6.20：

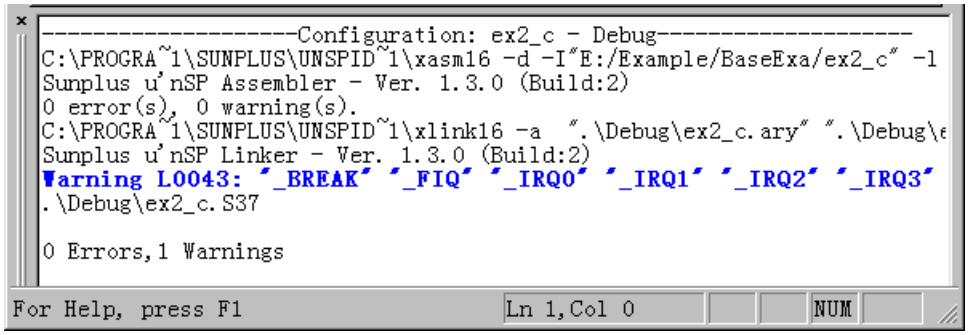


图6.20 输出窗口编辑信息界面

6.4.7 调试输出窗口（Debug）

在输出窗口显示调试信息，通常为：调试结束，调试过程采用无优化代码的方法。
举例： 编辑 IDE 下 Example 中的 Ex2。
编辑后的输出窗口调试信息界面图 6.21：

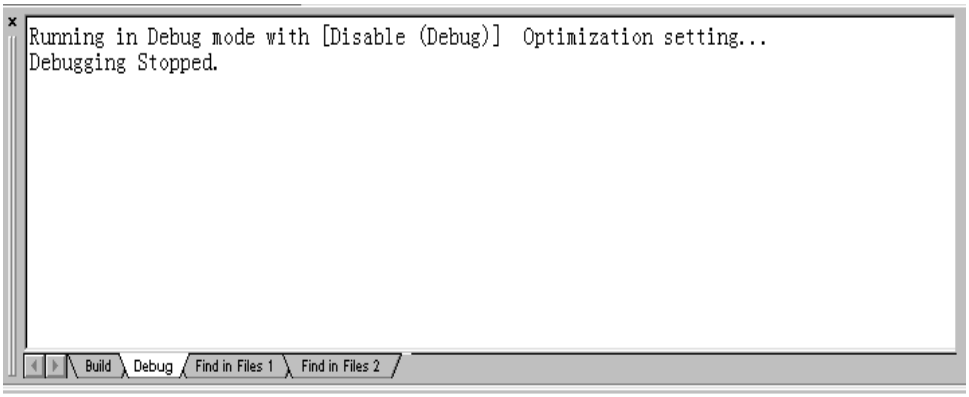


图6.21 输出窗口调试信息界面

6.4.8 查找输出窗口

显示在文件中查找文本的结果。

举例： 在 IDE 下 Example 中的 Ex2 查找单词‘**code**’。
查找后的输出窗口查找信息界面如图 6.22：

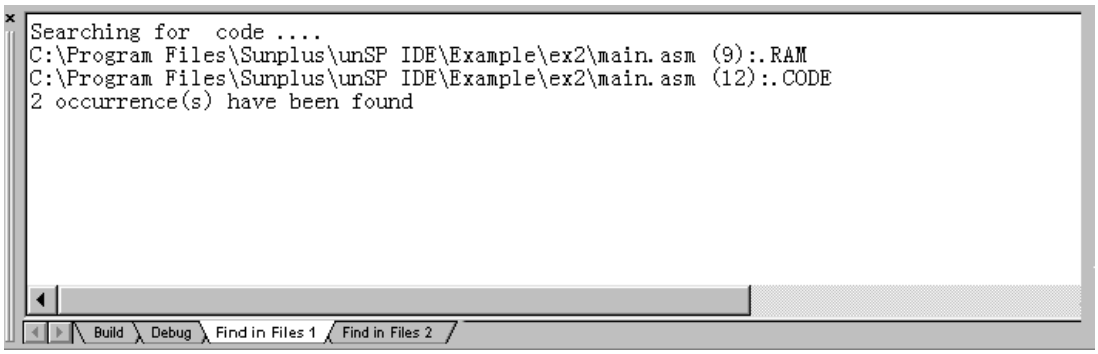


图6.22 输出窗口查找信息界面

6.4.9 调试窗口（Debug 窗口）

程序文件经过编译无错后，单击工具栏中的 Download 按钮，即可进入调试模式。所有的调试窗口均在调试模式下方可打开。

调试窗口主要用来显示有关的调试信息。在调试模式下，调试菜单显示在主菜单下。

调试窗口包括:

1）变量表 Watch 窗口
2）寄存器 Register 窗口
3）内存 Memory 窗口
4）反汇编窗口 Disassemble 窗口
5）历史缓冲区窗口

6.4.9.1 变量表 Watch 窗口

变量表 Watch 窗口用于输入并编辑变量，显示变量内容。

打开/关闭变量表窗口的方法：
第一种方法：单击菜单栏中 View/Watch 菜单命令。即可打开变量表窗口。
第二种方法：单击调试工具栏中的 Watch 按钮，即可打开变量表窗口。
第三种方法：通过热键 Alt+3 即可打开变量表窗口。
变量表 Watch 窗口界面如图 6.23：

图6.23 变量表 Watch 窗口界面

☆使用方法：双击变量名称处，可以出来一文本框，在文本框中写入变量名称，则相应的变量值和变量所在地址就可以显示出来。当要删除一变量时，选中该变量的所在的文本行，按 Del 键即可删除变量。或者单击右键选中删除命令也可以删除变量。
※注意：选中整行的内容方可删除变量。

6.4.9.2 寄存器 (Register) 窗口

寄存器 (Register) 窗口显示当前常用寄存器和特殊寄存器的内容。

打开/关闭寄存器窗口的方法：

第一种方法：单击菜单栏中 View/ Register 菜单命令。即可打开寄存器窗口。

第二种方法：单击调试工具栏中的 Register 按钮，即可打开寄存器窗口。

第三种方法：通过热键 Alt+4 即可打开寄存器窗口。

CPU 寄存器分通用型和专用型，其中通用型包括：R1～R4；专用型包括：SP、BP、SR、PC。

寄存器 Register 窗口界面见图 6.24：

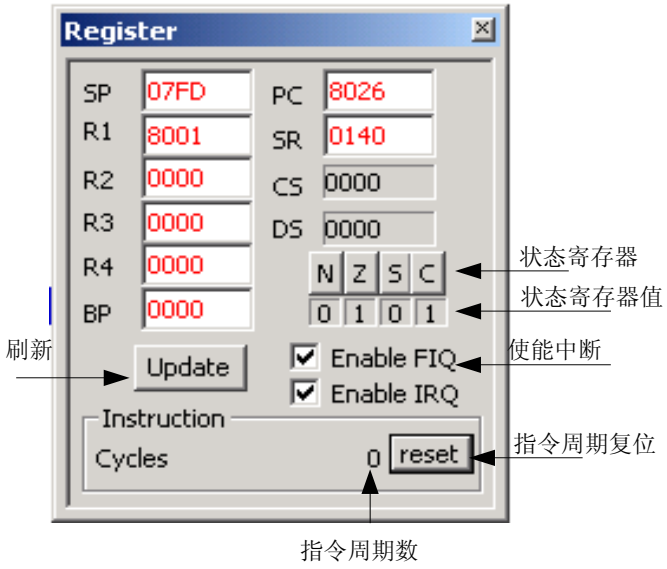


图6.24 寄存器(Register) 窗口界面

☆ 使用方法：单击寄存器的文本框即可以编辑该寄存器。
※注意：可以更改寄存器值。CS 的值最好不要轻易更改,否则会使调试程序出错。

6.4.9.3 内存 (Memory) 窗口

内存 (Memory) 窗口显示内存内容的。
打开/关闭内存窗口的方法：
第一种方法:单击菜单栏中 View/ Memory 菜单命令。即可打开内存窗口。
第二种方法：单击调试工具栏中的 Memory 按钮，即可打开内存窗口。
第三种方法：通过热键 Alt+5 即可打开内存窗口。
内存(Memory) 窗口界面见图 6.25：

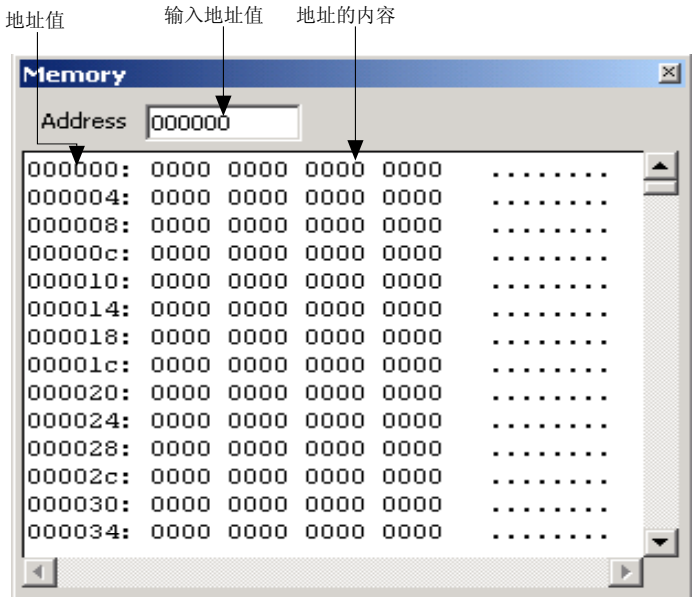


图6.25 内存 (Memory) 窗口界面

☆使用方法：在地址的文本框中可以直接写入要查找的地址值，回车后，内存窗口会自动到查找的地址处。

6.4.9.4 反汇编窗口 (Disassemble)

反汇编窗口 (Disassemble)显示反汇编内容。

打开/关闭反汇编窗口的方法：

第一种方法:单击菜单栏中 View/ Disassemble 菜单命令。即可打开反汇编窗口。

第二种方法：单击调试工具栏中的 Disassemble 按钮，即可打开反汇编窗口。

第三种方法：通过热键 Alt+6 即可打开反汇编窗口。

反汇编窗口 (Disassemble)界面见图 6.26：

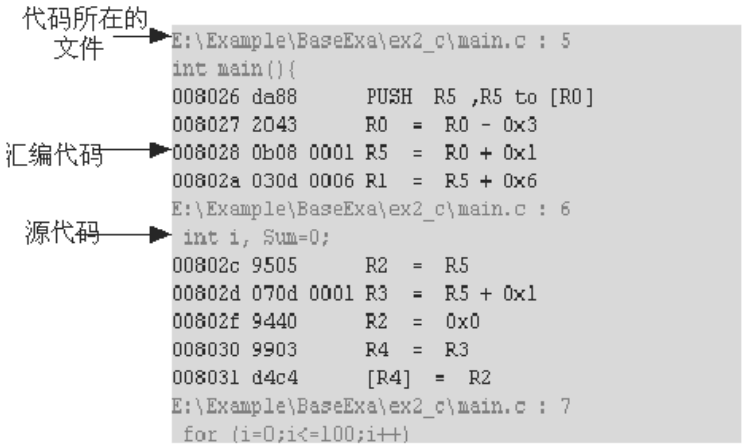


图6.26 反汇编窗口 (Disassemble)界面

6.4.9.5 历史缓冲区窗口

在仿真模式下，执行完程序后，被执行的指令、状态、内存内容将被存储到历史缓冲区中。

激活历史缓冲区的方法：

单击菜单栏 Project / Setting ，弹出 Setting 对话框,在 General 标签下，单击 reset 即可以激活 PC Trace Enable 。文件编译执行后，在调试环境下，单击[H]按钮，即可打开历史缓冲区窗口，被调试的程序的汇编码显示在历史缓冲区窗口内。如果您注意观察就会发现这时在项目文件夹中多了一个 .his 文件，即历史文件。

历史缓冲区窗口界面如图 6.27：

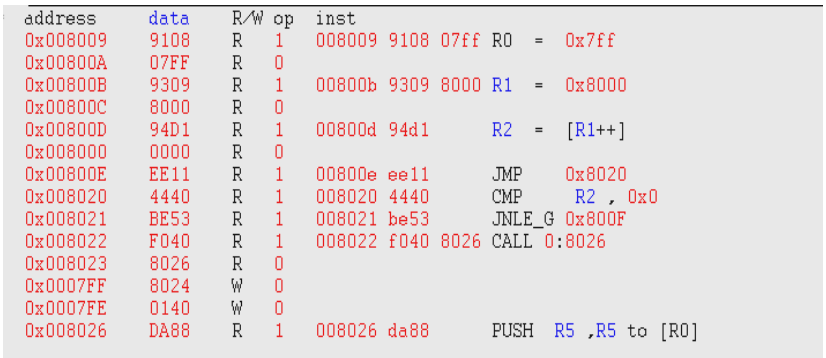


图6.27 历史缓冲区界面

历史缓冲区的界面与反汇编窗口内容很相似，但是反汇编界面比历史缓冲区界面多一项程序源代码，以方便开发者看反汇编程序时更清楚；而历史缓冲区界面比反汇编窗口界面多一项读写（R/W），这一项是方便开发者了解每一指令的具体操作。

6.4.10 其它窗口

6.4.10.1 命令窗口

单击 View 菜单下的 Command 命令，打开命令窗口，在该窗口列表框下面的文本输入框中键入帮助字符“**H**”并确认后，会在列表中列出 IDE 的所有命令及相应功能描述，图 6.28打开命令窗口界面：

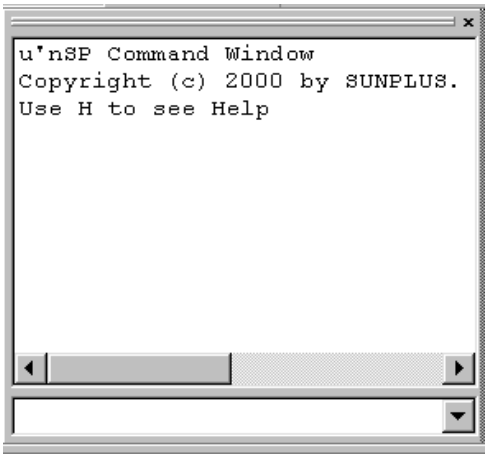


图6.28 命令窗口界面

在文本框中键入“H”后，列出命令及相应功能描述界面，如图 6.29

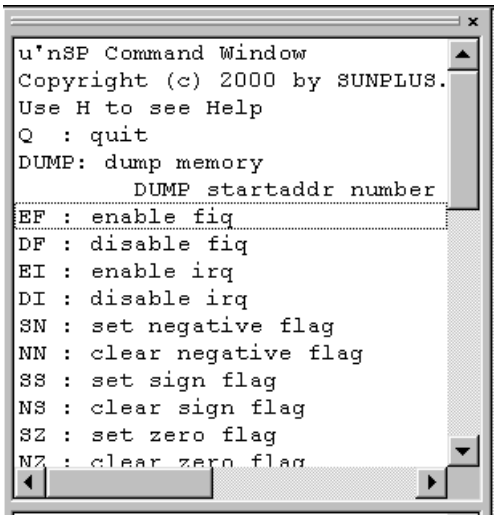


图6.29 文本框中键入“H”后界面

IDE 的命令及其功能描述见表 6.10：

表6.10 IDE 的命令及其功能

命令	功能描述	语法格式及语例
Q	退出μ'nSP™ IDE	
Dump	转储内存中的字数据	Dump　<起始地址>　<转储字数>. Dump　100 100　　//转储 0x100 ~ 0x1ff 中的字数据
EF	允许产生 FIQ 中断	
DF	禁止产生 FIQ 中断	
EI	允许产生 IRQ 中断	
DI	禁止产生 IRQ 中断	
SN	设置负标志	
NN	清除负标志	
SS	设置符号标志	
NS	清除符号标志	
SZ	设置零标志	
NZ	清除零标志	
SC	设置进位标志	
NC	清除进位标志	
X	复位(程序指针指向复位向量中的地址)	
RX	设定寄存器的值	Rx　<寄存器号>　<设定值>. Rx　3　abcd　　//将 R3 的值设为 0xabcd
O	设定内存单元中的值	O　<内存地址>　<设定值> O　7016　abcd　//将 0x7016 单元的值设为 0xabcd
F	设定内存区中的值	F　<内存起始地址>　<内存结束地址>　<设定值> F　100　1ff　1234　//将 0x100 ~ 0x1ff 单元填入 0x1234
BC	清除断点[1]	BC　<断点地址>　<断点标志>　<断点数据> BC　8000　8082　1234 //清除当向 0x28000 单元中写入数据 0x1234 时的条件断点
BP	设置断点	BP　<断点地址>　<断点标志>　<断点数据> BP　8000　8082　1234 //设置当向 0x28000 单元中写入数据 0x1234 时的条件断点
G	连续运行程序	
S	单步运行程序	
L	将二进制文件装入内存[2]	L <文件名> <起始地址> <结束地址> <内存的起始地址> L test.bin 100 1ff 8000//将 test.bin 文件中第 0x100~0x1ff 单元的//数据装入内存 0x8000 单元
RF	将内存中的数据内容转储到文件中	RF　<起始地址>　<内存单元个数>　<文件名> RF　100　100　test.bin //将 0x100 ~ 0x1ff 单元的内容转储至 test.bin 文件中
H	显示命令帮助信息	显示μ'n SP IDE 的所有命令及及其内容描述

命令的检索：用鼠标左键点中列表框中的某一命令，在 PC 机键盘上每敲入该命

令的头一个字符时会发现，列表框中当前命令的指向会在所有首字符同敲入字符的命令之间移动。据此功能可在列表框里列出的诸多命令中迅速检索到所需的命令。

命令的操作：按照列表框中列出的的命令格式在文本输入框中正确键入某命令字符并确认后，该命令便会被执行。

6.4.10.2 转存窗口

在调试模式下，单击 Tools /Dump Memory 即进入转存窗口。

该窗口用于存储指定地址范围的内容到指定的文件中。另外，它也可以将高字节和低字节分别指定的地址范围存储到两个文件中。

转存窗口界面见图 6.30：

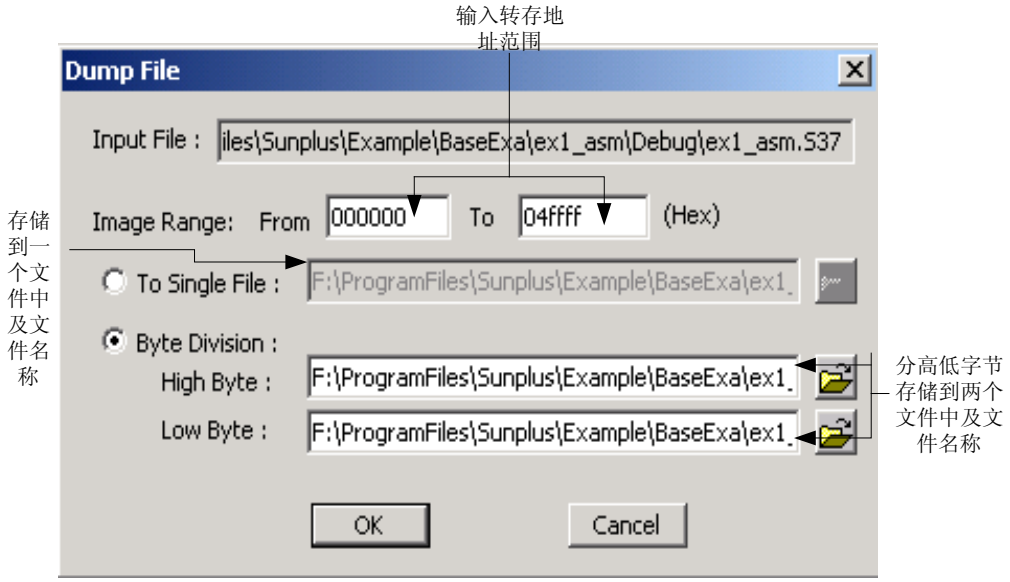


图6.30 转存窗口界面

项目的操作与使用

在 C、C++、VB 等语言中，广泛使用“Project”一词。在译成中文时有的译成“项目”，有的译成“工程”。在这里我们译成“项目”。

开发一个应用程序需要很多文件，这些文件需要规范管理，所以一整组的相关文件就构成了一个项目。项目是可以独立执行的程序单元。一个应用程序可以是一个单独的项目。在项目中，可以含有不同的元组和文件。准确一点，项目是指为用户调程建立起来的一个开发环境，提供用户程序及资源文档的编辑和管理，并提供各项环境要素的设置途径，最后将通过用户程序及库的编制（包括编译、汇编以及链接等）提供出一个良好的调试环境。下面详细介绍项目的各项操作及使用。

6.5 项目

6.5.1 建立项目

新建项目的方法步骤：

1. 用鼠标左键单击 File 下拉菜单 New 弹出 New 对话框，如图 6.31所示。
2. 在该窗口中选中 Project 标签并在 File 的文本框中键入项目的名称。在 Location 下的文本框中输入项目的存取路径或利用该文本框右端的浏览按钮制定项目的存储位置。
3. 用鼠标左键单击 New 对话框里的 OK 按钮,则项目建立完成。

新建项目的需求：

在做一个应用程序前，首先要建项目。

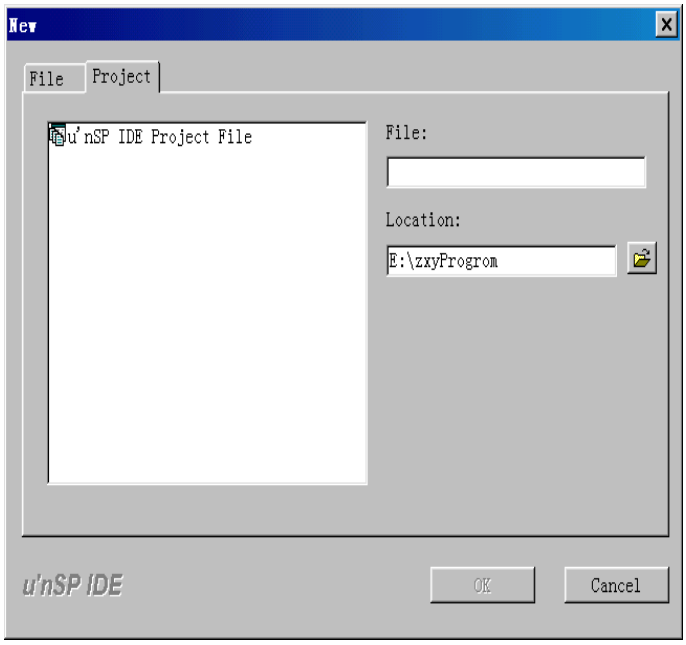


图6.31 新建项目/文件对话框

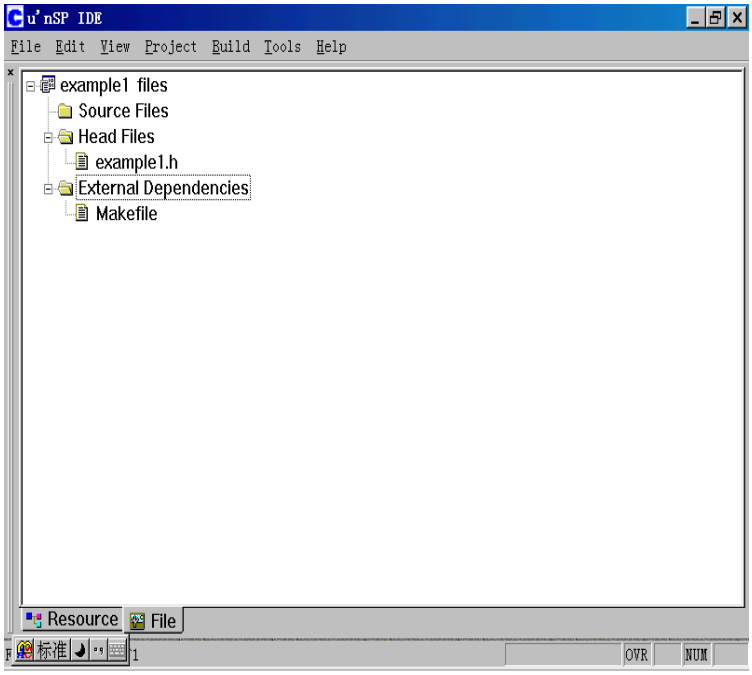


图6.32 新建项目后的 Workspace 窗口

例如：

项目名称： Example1

项目位置：E:\ZxyProgram\ Example1

新建项目后的 Workspace 窗口 结果: 生成了新项目 Example1。

6.5.2 在项目中新建 C 文件（.C）

新建 C 文件的方法:在新建项目下, 点击菜单 File 下拉菜单 New 弹出 New 对话框,如图 6.33。点击 μ'nSP™ IDE C File, 在 File 下的文本框内键入文件名称, OK。

新建 C 文件的需求：

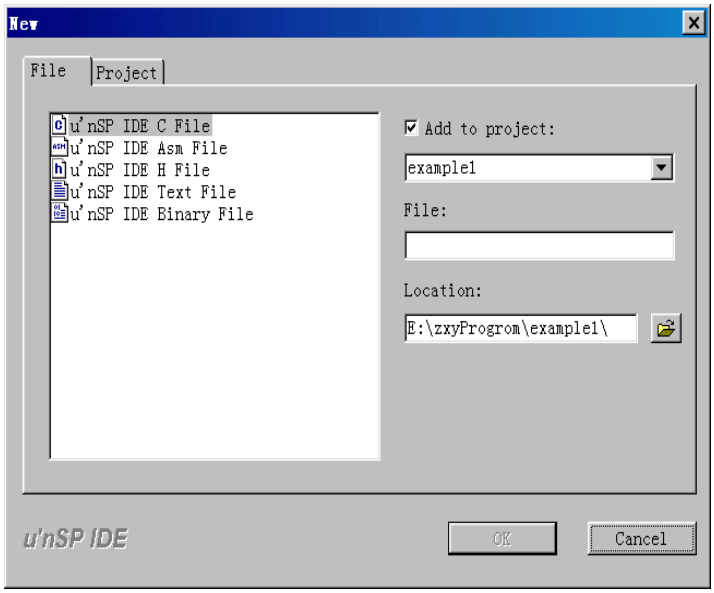


图6.33 新建文件/项目对话

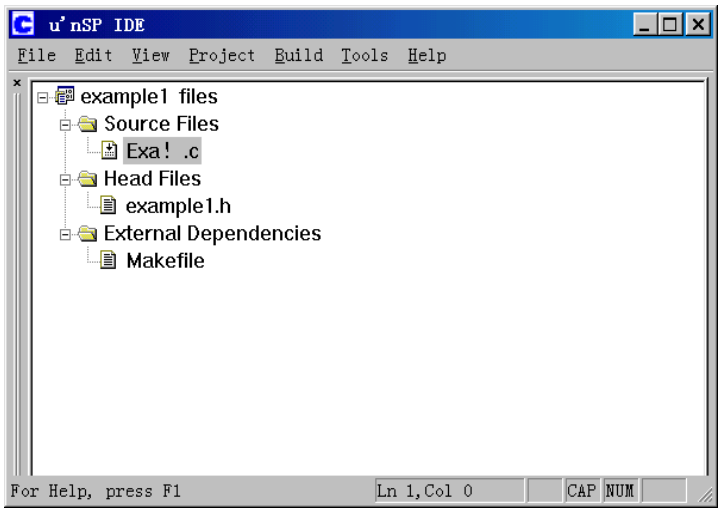


图6.34 新建 C 文件后的 Workspace 窗口

用 C 语言做程序时需要建立 C 文件类型。
例如：
文件名称： Exa1
文件位置:
E:\ZxyProgram\Example1\Exa1.c　　OK
新建 C 文件后的 Workspace 窗口 结果: Source File 下多出一个 Exa1.c 文件。

6.5.3 在项目中新建汇编文件(.asm)

新建汇编文件的方法:在新建项目下，点击菜单 File 下拉菜单 New 弹出新建文件/项目的对话框,如图 6.35。点击 μ'nSP™ IDE ASM File ,在 File 下的编辑框内写入文件名称,OK。

新建汇编文件需求：
用汇编语言做程序时需要建立汇编文件类型。
例如：
文件名称：　Exa1
文件位置: E:\ZxyProgram\ Example1\Exa1.ASM　　OK
新建汇编文件后的 Workspace 窗口 结果: Source File 下多出一个 Exa1.asm 文件。

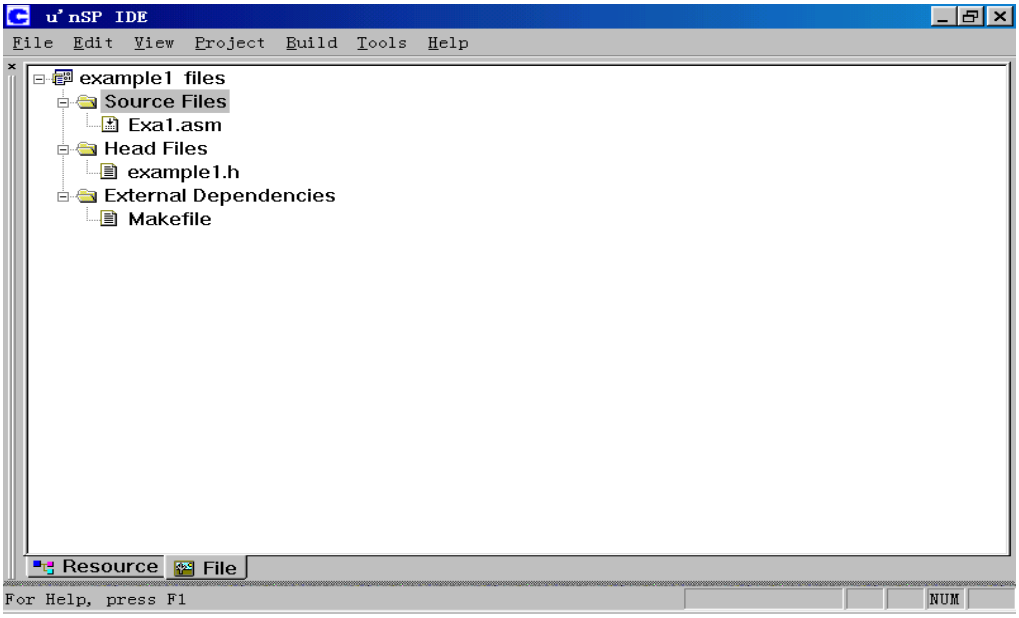


图6.35 新建汇编文件后的 Workspace 窗口

6.5.4 在项目中新建头文件 (.H)

新建头文件的方法:在新建项目下，点击菜单 File 下拉菜单 New 弹出新建文件/项目的对话框,如图 6.36。点击 μ'nSP™ IDE H File,在 File 下的编辑框内写入文件名称,OK。

新建头文件需求：
多个文件共享的文件可以建成头文件。
例如：
文件名称：　head
文件位置: E:\ZxyProgram\ Example1\　　OK

新建头文件后的 Workspace 窗口 结果: Head File 下多出一个 head.h 文件

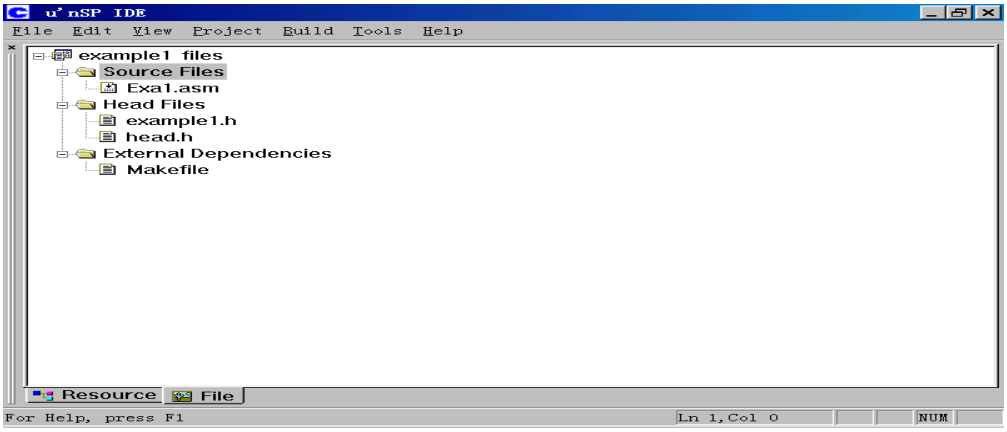


图6.36 新建头文件后的 Workspace 窗口

6.5.5 在项目中新建文本文件(.txt)

新建文本文件的方法:在新建项目下，点击菜单 File 下拉菜单 New 弹出新建文件/项目的对话框,如图 6.37。点击 μ'nSP™ IDE Text File,在 File 下的编辑框内写入文件名称,OK。

新建文本文件的需求：

对程序文件做文档说明时，可以建文本文件类型。

例如：

文件名称：text

文件位置: E:\ZxyProgram\ Example1\　　OK

新建文本文件后的 Workspace 窗口 结果: External Dependencies 下多出一个 text.txt 文件。

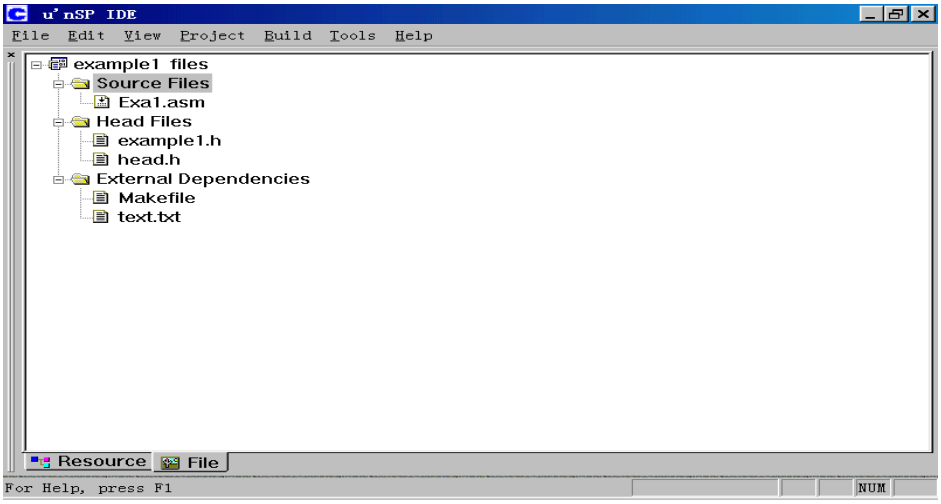


图6.37 新建文本文件后的 Workspace 窗口

6.5.6 在项目中新建二进制文件

新建二进制文件的方法:在新建项目下，点击菜单 File 下拉菜单 New 弹出新建文件/项目的对话框,如图 6.38。点击 μ'nSP™ IDE Binary File ,在 File 下的编辑框内写入文件名称,OK。

新建二进制文件的需求：
在做资源文件时，需建立二进制文件类型。
例如：
文件名称：bin
文件位置: E:\ZxyProgram\ Example1\　　OK
新建二进制文件后的 Workspace 窗口 结果: External Dependencies 下多出一个 bin 文件。

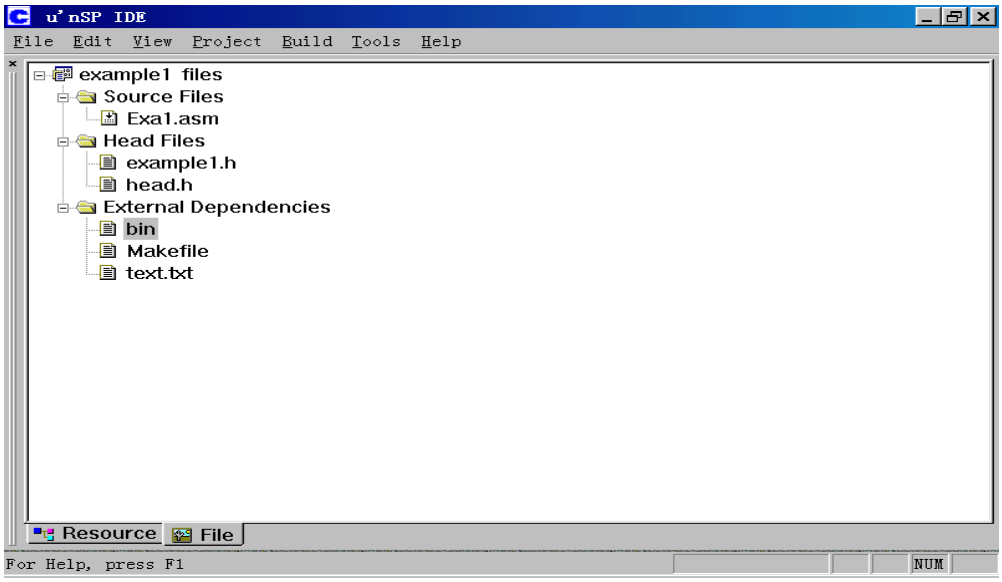


图6.38 新建二进制文件后的 Workspace 窗口

6.5.7 在项目中添加/删除文件

在项目中添加文件的方法：
第一种方法：通过 Project 菜单方法。
可通过菜单途径用鼠标左键单击 Project 菜单里 Add to Project 选项中的 Files 或 Resource 子项,激活 Add Files 对话框;
第二种方法：通过 Workspace 窗口。
（1）在 Workspace 窗口内，选中元组，单击右键弹出以下拉菜单。如图 6.39。所示。
（2）用鼠标左键单击 Add Files To Folder 选项，可激活 Add Files 对话框；如图 6.40所示。

（3）在文本框中键入将添加的文件，单击“打开”按钮。即将添加的文件加到所选的元组中。

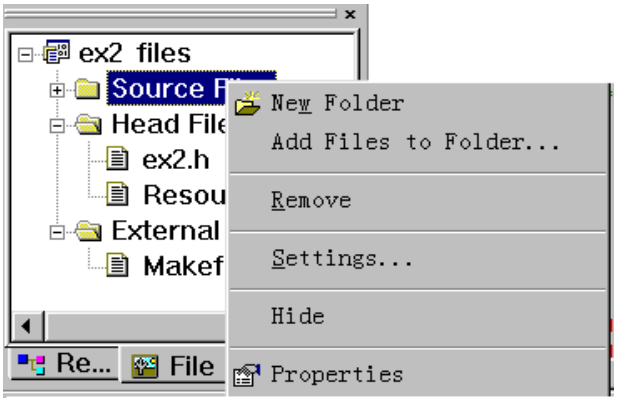


图6.39 添加文件下拉菜单界面

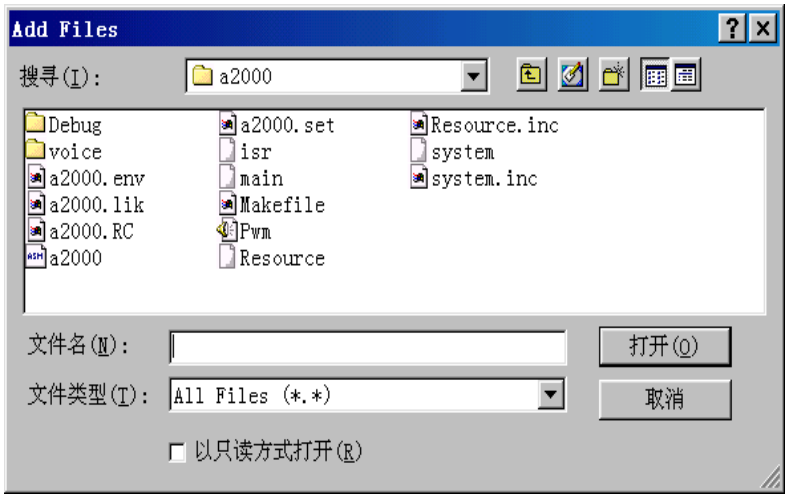


图6.40 添加文件对话框界面

删除文件步骤：

1．在 File 视窗或 Resource 视窗里选中元组中的某个文件。

2．单击鼠标右键，再弹出的下拉菜单如图 6.39所示。选中 Remove 选项，则该文件会从元组中被删除。

6.5.8 在项目中使用资源

当在项目里的资源元组中添加资源文件时，该资源文件的存储路径及名称会自动被记入项目中的.rc 文件中，并以 RES_*的缺省文件名格式被赋予一个新的文件名(此处’*’是指资源文件在其存储路径上的文件名);同时,添入的资源文件还会被安排一个文

件标识符 ID。

6.5.9 项目选项的设置

项目选项的设置是针对不同目标而对开发环境的各个要素进行的设置。其设置界面如图 6.41：

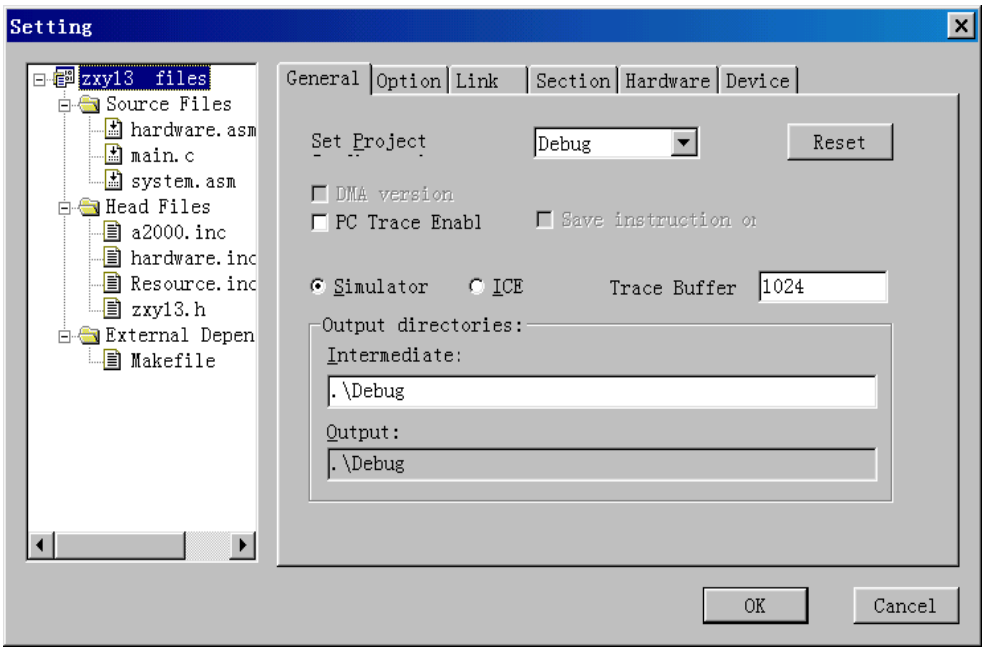


图6.41 项目选项设置界面

根据界面中的这些标签便会进入相应的属性页里进行项目的各项设置。

1.General 属性页

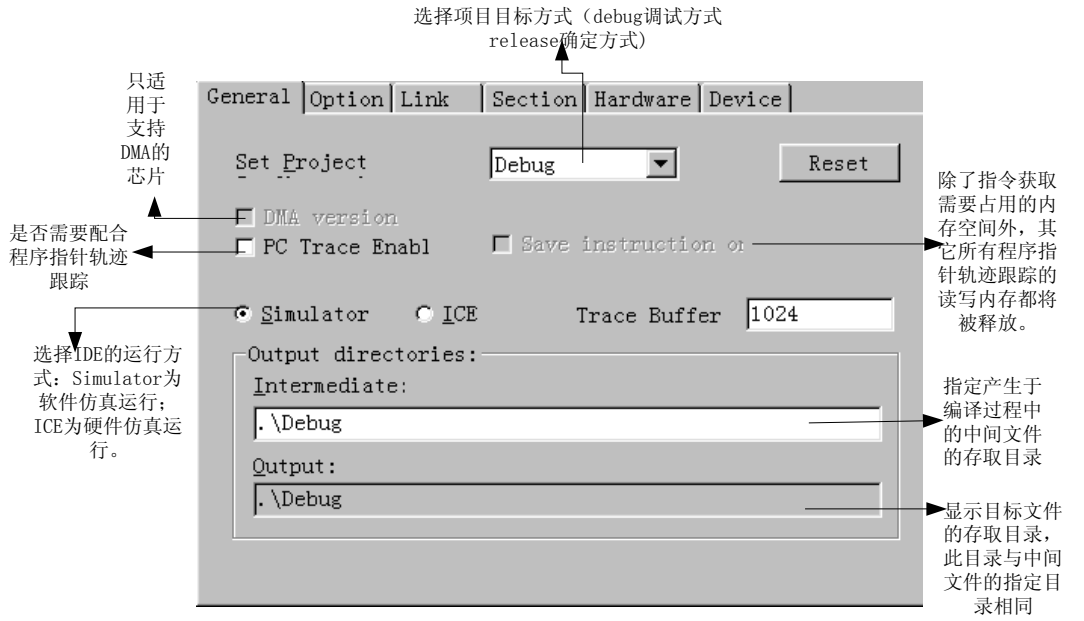


图6.42 General 属性页

2.Option 属性页

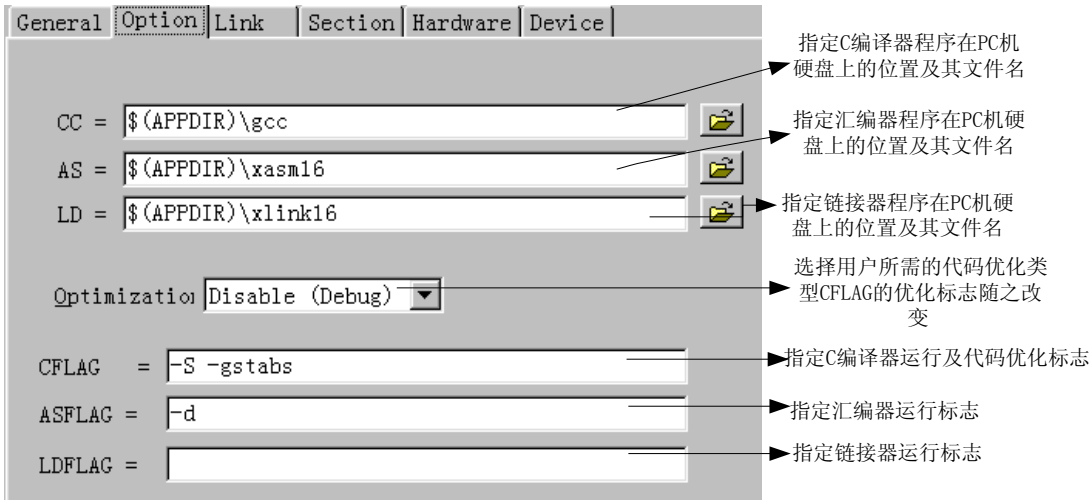


图6.43 Option 属性页

3．Link 属性页

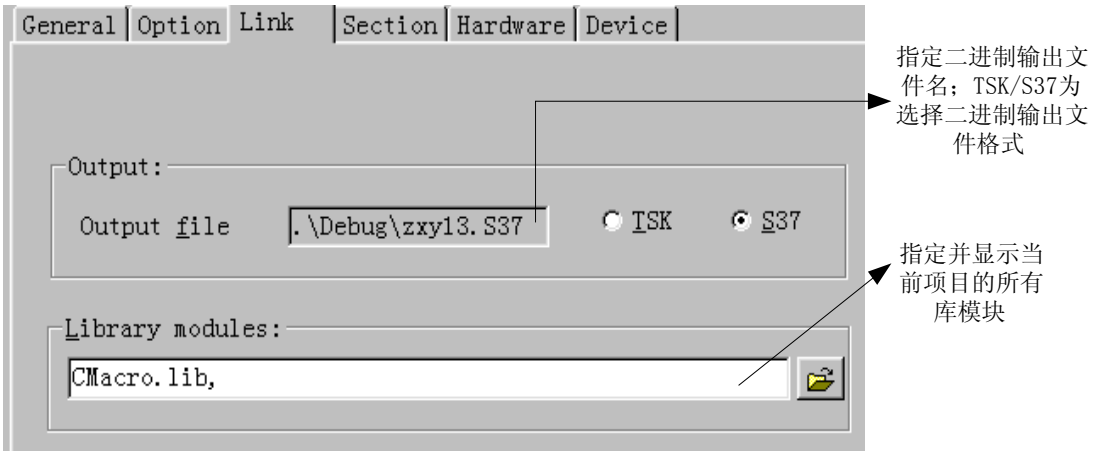


图6.44 Link 属性页

4．Section 属性页

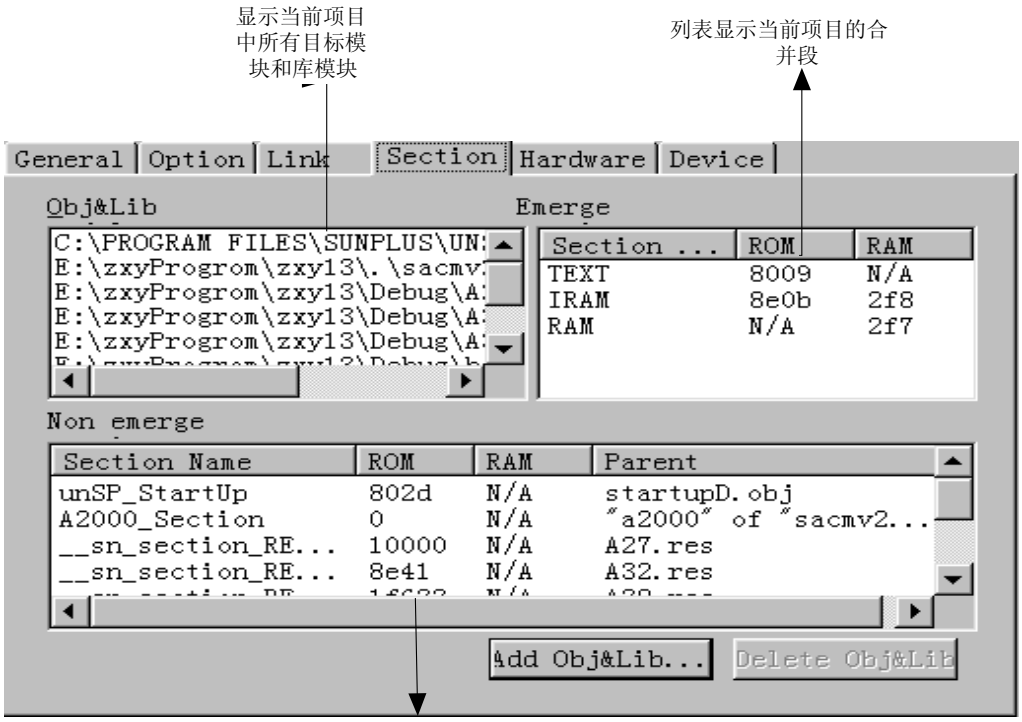


图6.45 Section 属性页

5．Hardware 属性页

图6.46 Hardware 属性页

6．Device 属性页

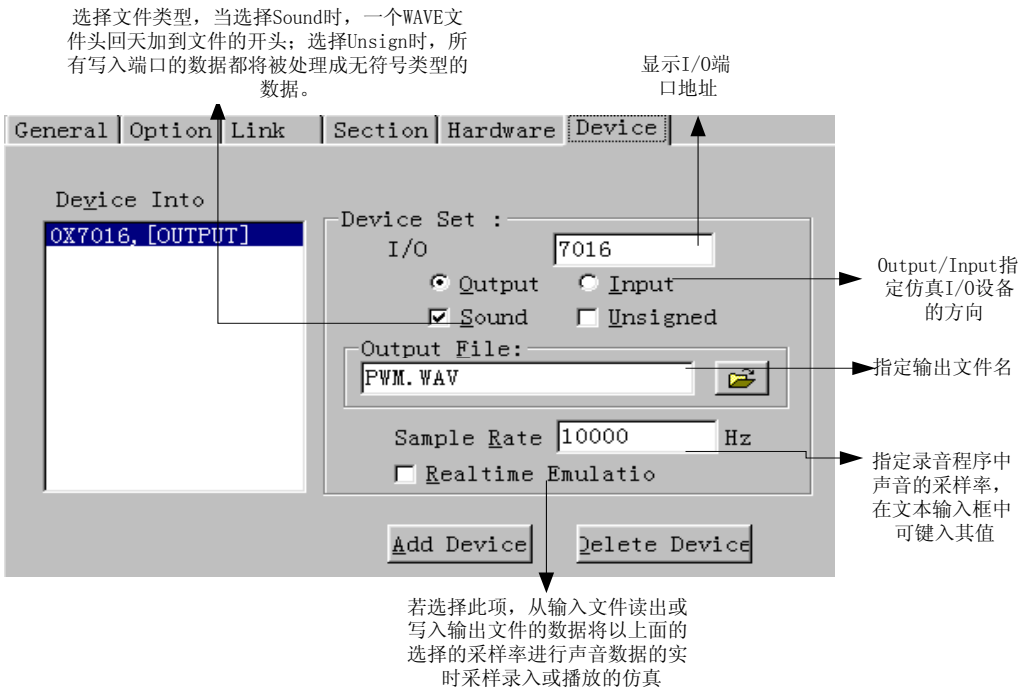


图6.47 Device 属性页

6.5.10 项目的编译

当项目中的文件编写结束后，要对项目中的程序进行编译，并将编译出来的二进制代码与库中的各个模块连接成一个完整的、地址统一的可执行目标文件和符号表文件，供用户调试使用，在这里要使用编译器、汇编器、链接器等工具。

项目编译的基本操作包括：

- Compile：对编辑窗口中当前文件进行编译
- Build ：编制当前的文件
- Rebuild All ：重新编制当前项目目标，将处理当前项目中的所有文件
- Stop Build ：终止当前项目目标编制

6.5.10.1 Compile/Build/Rebuild All/Stop Build 的方法

单击 Build 菜单弹出下拉菜单包括 Compile/Build/Rebuild All/Stop Build 命令或者在 Build 工具栏 中也可以找到这几个工具。

6.5.10.2 Compile/Build/Rebuild All/Stop Build 后的结果

编制过程中的一些操作信息将显示在输出窗口的 Build 视窗中，如图 6.48所示。

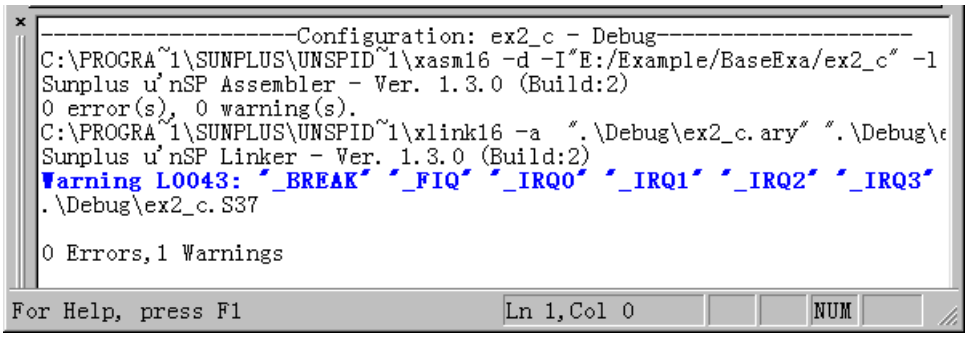


图6.48 编辑后输出窗口的 Build 视窗

6.6 代码剖视器（profiler）使用及功能

μ'nSP™ IDE 的代码剖视器是一个强有力的分析工具。通过应用此工具来剖析、优化程序代码。具体地，此工具具有以下一些功能：

- 提供代码优化的准确信息。对部分程序进行诸多重要因素的剖析，包括某段程序花费了多少个指令周期的执行时间，程序中的标号流等一些有助于提高

程序效率的信息；

- 检测并分析程序运行当中使用算法的有效性之高低；
- 检查用户程序的代码段是否面临处在系统测试程序区的危险。

6.6.1 激活 Profile 方法

在非调试情况下，用鼠标左键单击 Build 菜单的 Profile 选项，激活 Profile Configure 对话框。如图 6.49。

在调试情况下，直接点击菜单栏中的 Profile 菜单命令，即可激活 Profile Configure 对话框。如图 6.49。

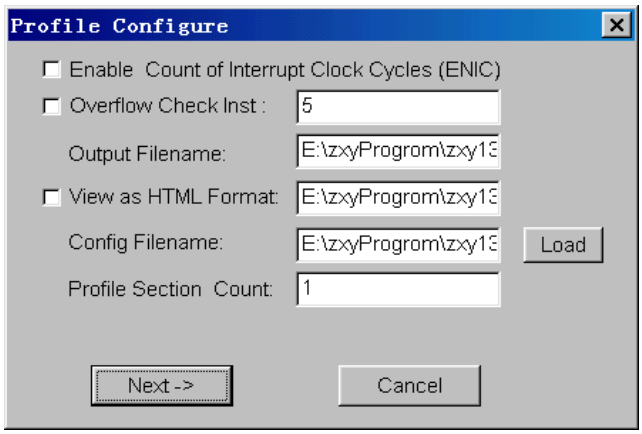


图6.49 Profile 对话框界面

Profile Configure 对话框 1 中设置选项及其内容如下表 6.11：

表6.11 Profile Configure 对话框 1 中设置选项及其内容

设置项	设置形式	设置内容描述
Enable Count of Interrupt Clock Cycles (ENIC)：	复选框	ENIC 选项是为在 IRQ 中断服务子程序中仍可连续续剖视代码所设。若要求剖视的代码段处于 IRQ 子程序中，则须选中此项
Overflow Check Inst：	文本输入框	设定当运算产生溢出时多少个指令内未检查溢出标志便产生警告信息
Output Filename：	文本输入框	指定容纳最终的剖视结果的文件名称
View as HTML Format	复选框及文本框	选择是否需以网页格式来查看剖视代码结果，若需要则应在文本框中指定网页格式的剖视结果文件名称
Config Filename：	文本输入框	指定存储配置参数的文件名称，用于每次调程开始时重新装入
Profile section count：	文本输入框	指定需要剖视程序段的段数

6.6.2 使用 Profile 步骤

第一步：根据对话框选项的介绍，设置对话框 1 的选项，单击 Next 按钮。

如：Profile section count 选项设为 1。

第二步：将出现对话框 2。如图 6.50，设置 Profile 程序的停止地址。如：8df2。单击 Next 按钮。

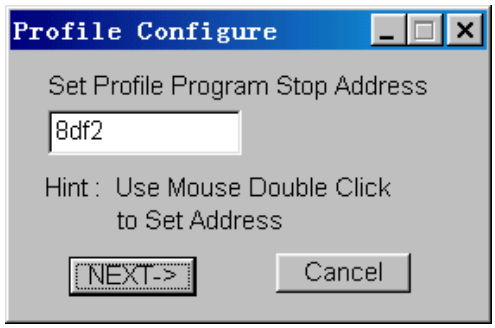


图6.50 Profile 对话框 2 界面

第三步：设置 Profile 第一部分的起始地址。如图 6.51，如：8deb。单击 Next 按钮。

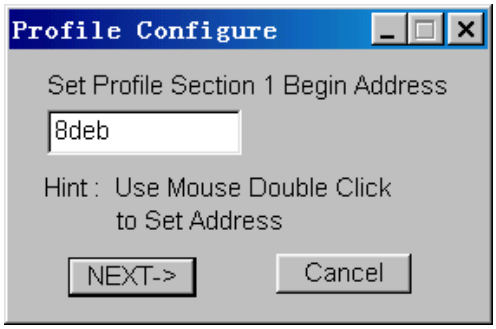


图6.51 Profile 对话框 3 界面

第四步：设置 Profile 第一部分的停止地址 。如图 6.52，如：8ded。单击 NEXT 按钮。

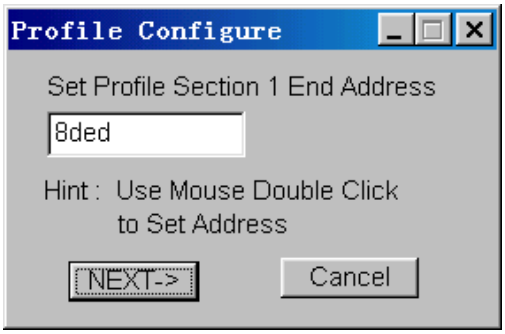


图6.52 Profile 对话框 4 界面

第五步： 如图 6.53 Profile 对话框 5，单击 Profile 按钮。开始 Profile 并弹出剖视结果窗口如图。

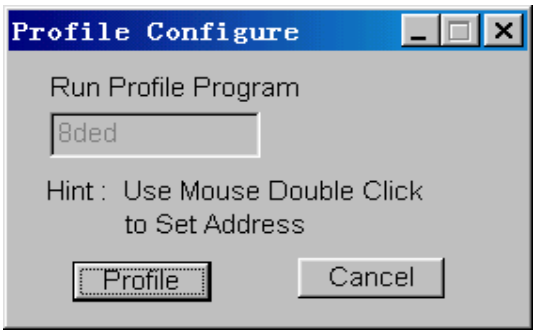


图6.53 Profile 对话框 5

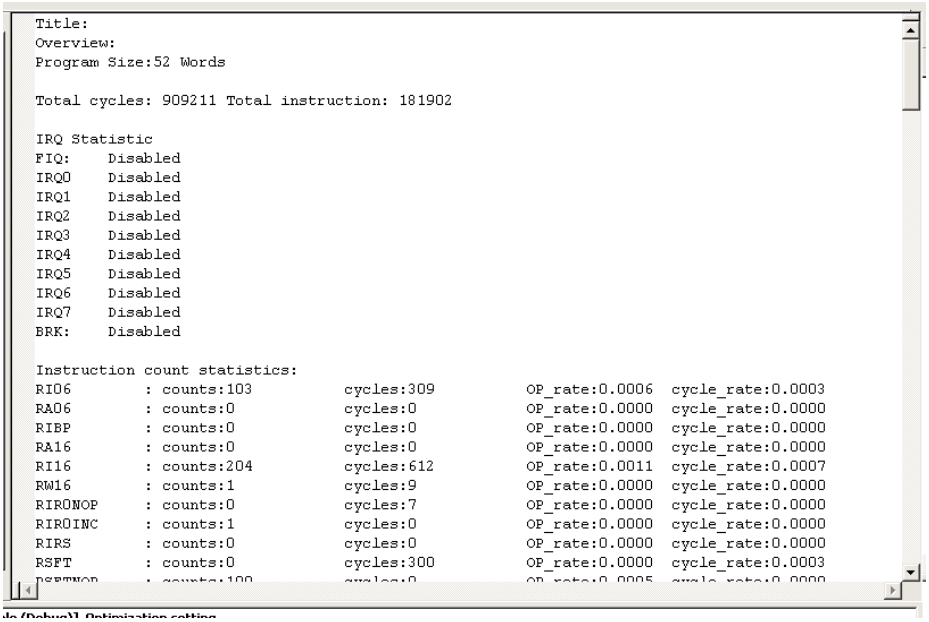


图6.54 显示剖视结果信息的窗口

第六步：停止剖视器操作并关闭剖视结果信息的窗口

方法一：MS-DOS 窗口下按下 PC 机键盘的任意键便可关闭该窗口且结束剖视器的操作。

方法二：单击窗口右上角（×），直接关闭剖视结果信息的窗口。

6.7 举例

程序6-1 简单 C 语言程序

范例：　从 1+2+…+100 计算结果

```
//*****************************************************************//
EX1
//*****************************************************************//
int main(){
        int i, Sum=0;
        for   (i=0;i<=100;i++)
              Sum = Sum + i;                    // Sum 为累加结果

        while(1){                               //程序死循环
                                                // 使用变量 Watch 窗口观察 Sum 的值
         }
}
//*****************************************************************//
```

方法步骤：

1. 新建项目, 项目名称 EX1。
2. 该项目下新建 C 文件, 文件名称 EX1。
3. 在 C 文件中键入范例源代码（图 6.55）
4. 保存项目
5. 编译（单击 Build/Compile 菜单命令）该程序，如0所示。检查是否有语法错误。如果无错，继续，否则，改错。
6. 编辑（单击 Build/ Build 菜单命令）该程序，如图 6.57所示。如果无错，继续，否则，改错。
7. 将程序下载到本机中进行调试。（单击编辑工具栏中 Download 命令按钮）。进入调试状态。如图 6.58所示。
8. 在调试状态下，打开寄存器、变量等调试窗口，单步执行，仔细观察寄存器和变量的变化。图 6.59为程序执行到死循环后，变量和寄存器结果。

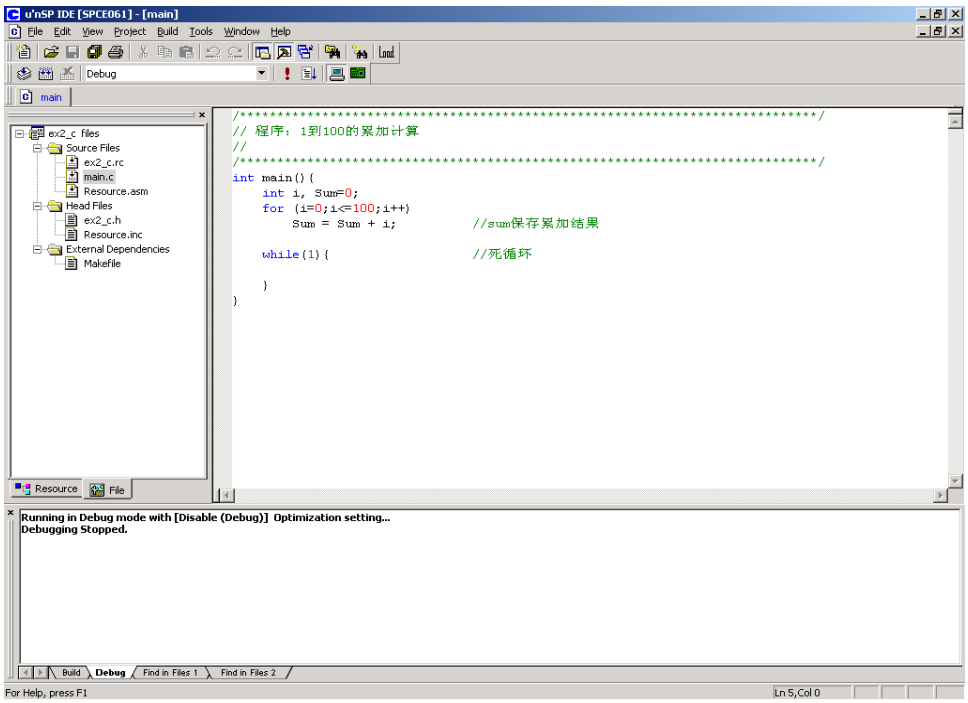


图6.55　C 文件范例程序界面

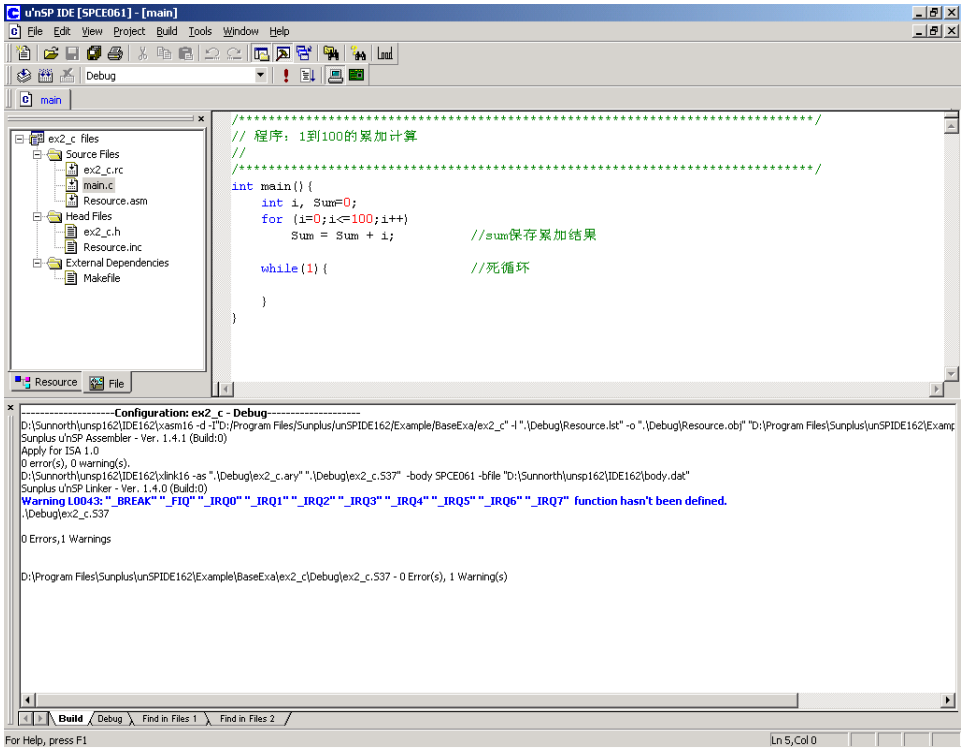


图6.56　C 文件程序编译后界面（注意输出窗口）

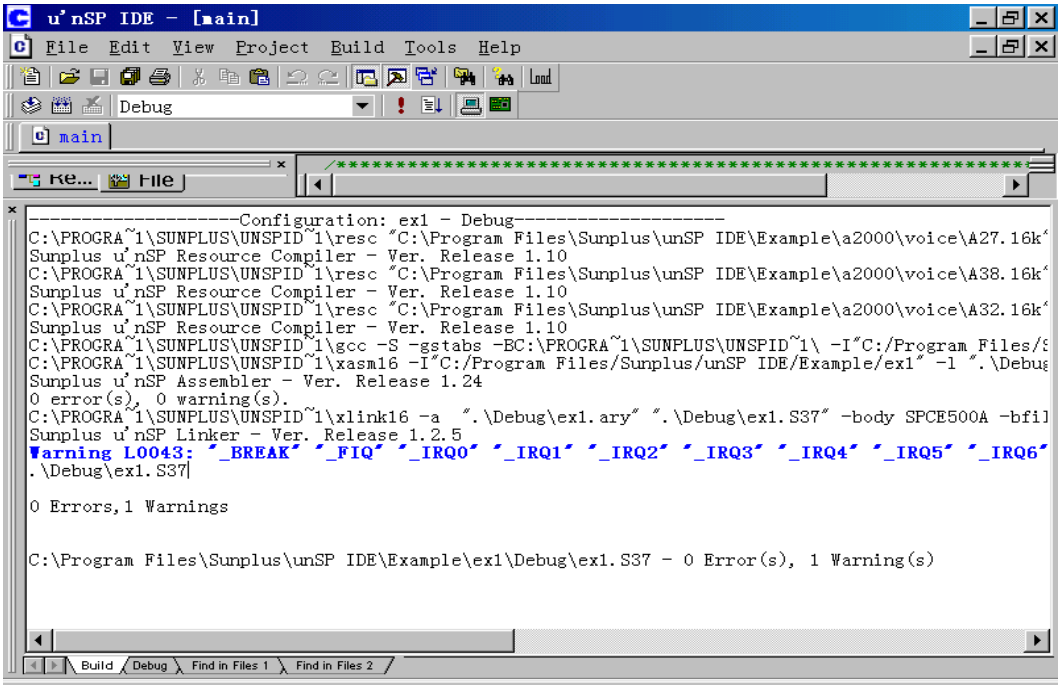


图6.57　C 文件程序编辑后输出窗口界面

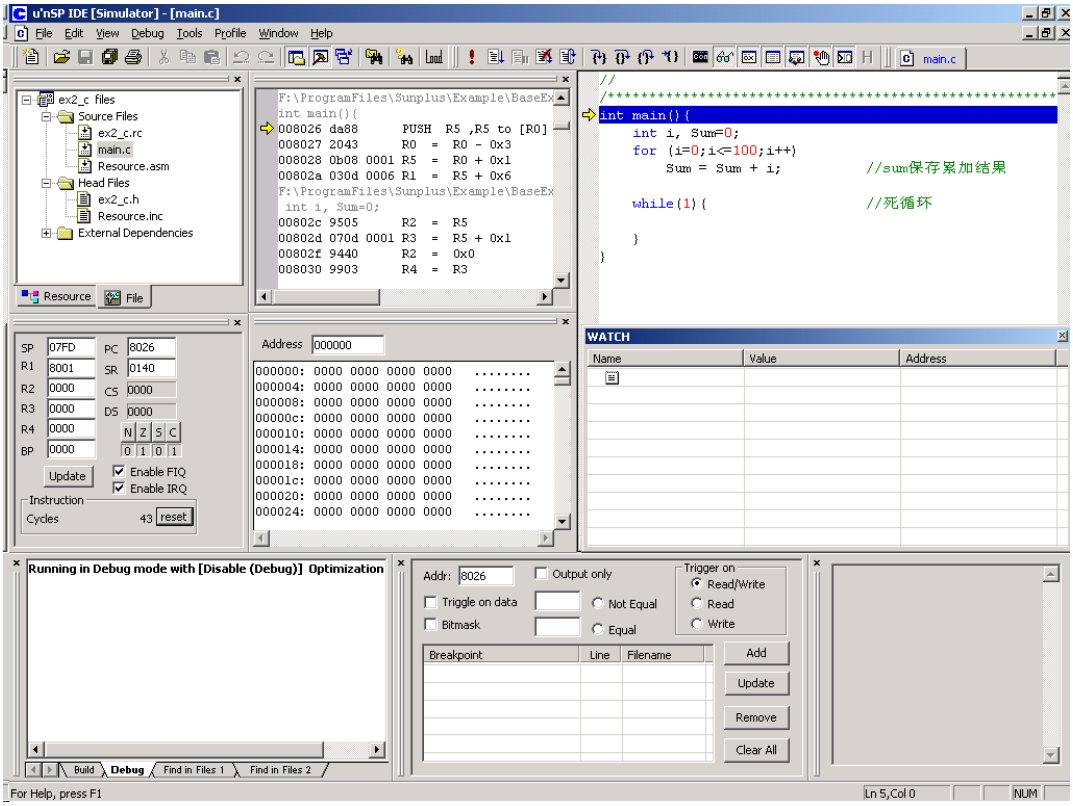


图6.58　C 文件程序 Download 后的调试界面

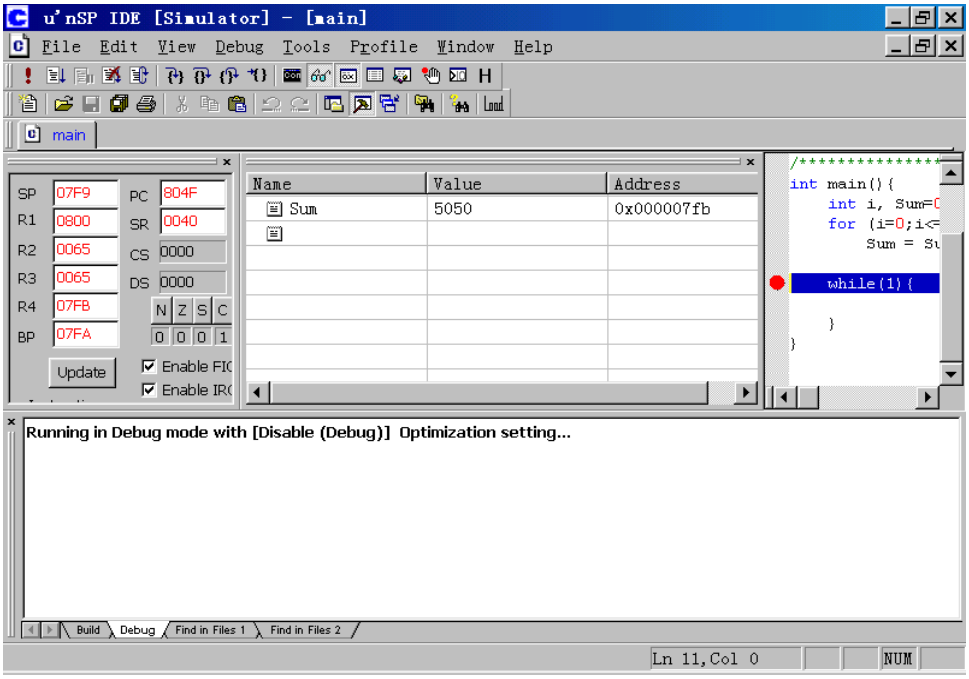


图6.59 调试状态下，打开寄存器、变量等调试窗口图

```
//*********************************************************************//
EX1        END
//*********************************************************************//
```

程序6-2 简单汇编语言程序

范例：　从 1+2+…+100 计算结果

```
//*********************************************************************//
EX2
//*********************************************************************//
.RAM                                    // 定义 RAM 段
.VAR    R_Sum;                          // 定义变量 R_Sum 保存累加结果

.CODE                                   //定义 CODE 段
.PUBLIC _main;                          // 对 MAIN 的声明
_main:
            R1 = 0x0001;                // r1=[1..100]
            R2 = 0x0000;                //
L_SumLoop:
            R2 += R1;                   // 累加值放到 R2 中
            R1 += 1;                    // 下一个被加数
            CMP   R1,100;               // 被加数是否为 100?
```

```
        JNA  L_SumLoop;                    // 如果 r1 <= 100 返回到 L_SumLoop
        [R_Sum] = R2;                      // 将最终累加结果保存到 R_Sum 中

L_ProgramEndLoop:                          // 程序死循环
        JMP     L_ProgramEndLoop;
//*************************************************************//
```

方法步骤：

1. 新建项目，项目名称 EX2。
2. 该项目下新建汇编文件，文件名称 EX2。
3. 在汇编文件中键入范例源代码（如图 6.60）
4. 保存项目
5. 编译调试该程序（其中编译调试步骤与例 1 相同，不再赘述）

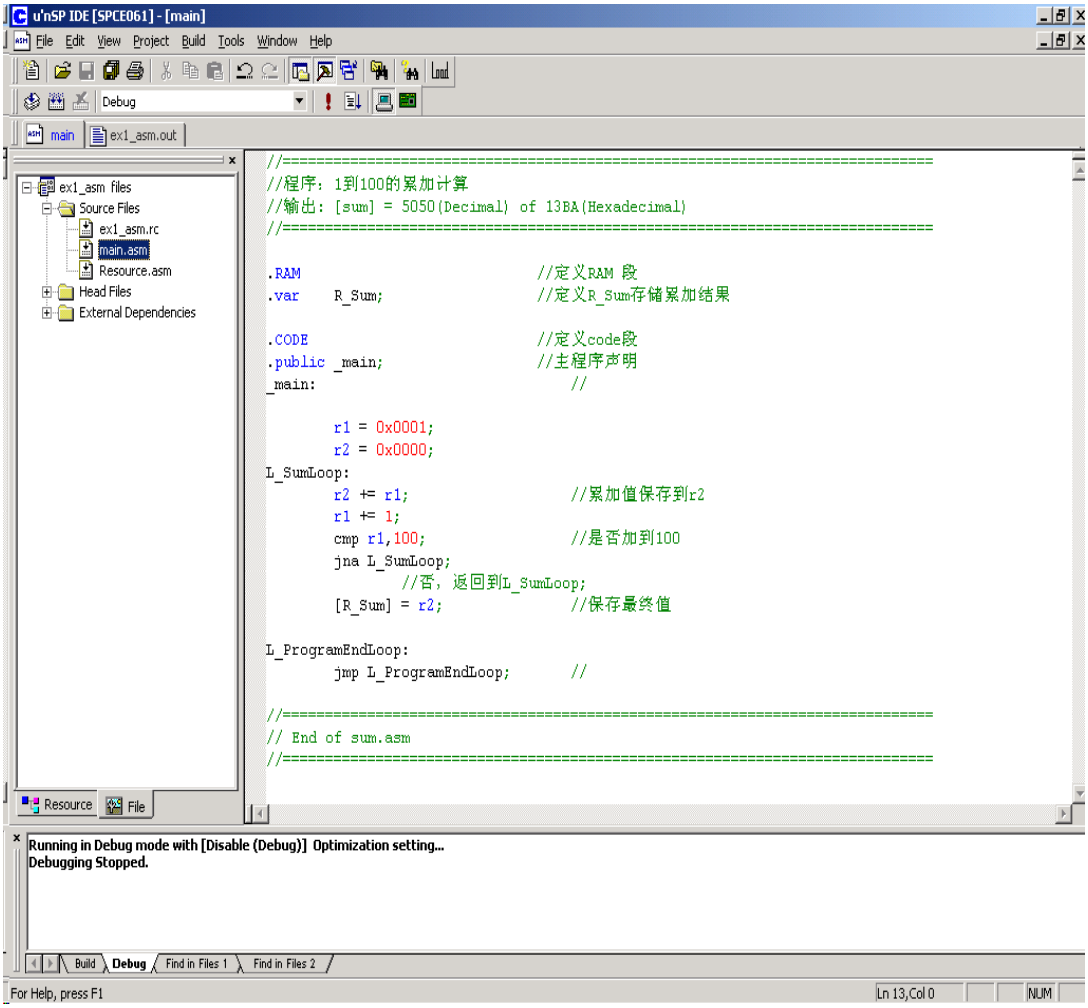


图6.60 汇编文件范例程序界面

```
//*************************************************************//
EX2         END
//*************************************************************//
```

程序6-3 简单 C 与 ASM 综合文件

范例：设置 A 口为输入，B 口为输出。

```
//*****************************************************************//
EX3
//*****************************************************************//
```

主程序为 C 文件，程序如下：

```
#include "hardware.h"

int main(){
 int Key;
                                          //Dir,   Data,   Attrib
     SP_Init_IOA(0x0000,0x0000,0x0000);   //设置 A 口为带下拉电阻的输入口
     SP_Init_IOB(0xffff,0x0000,0xffff);   //设置 B 口为带数据缓存器的低电平输出口

while(1){
     }
}
```

汇编程序：

```
.INCLUDE HARDWARE.INC                     //包含 HARDWARE.INC
.PUBLIC _SP_Init_IOB;                     //声明子函数_SP_Init_IOB
.PUBLIC _SP_Init_IOA;                     //声明子函数_SP_Init_IOA
.CODE
_SP_Init_IOA: .PROC                       //子程序的开始
          PUSH BP,BP TO [SP];             //将 BP 压栈
          BP = SP + 1;
          PUSH R1,R1 TO [SP];             //将 R1 压栈

          R1 = [BP+3];                    //取 A 口的方向向量
          [P_IOA_Dir] = R1;
          R1 = [BP+4];                    //取 A 口的数据向量
          [P_IOA_Data] = R1;
          R1 = [BP+5];                    //取 A 口的属性向量
          [P_IOA_Attrib] = R1;

          POP R1,R1 FROM [SP];            //出栈
          POP BP,BP FROM [SP];            //出栈
          RETF;
          .ENDP                           //子程序结束

////////////////////////////////////////////////////////////////
// SP_Inti_IOB
////////////////////////////////////////////////////////////////
```

```
_SP_Init_IOB: .PROC                          //子程序的开始
        PUSH BP,BP TO [SP];                  //将 BP 压栈
        BP = SP + 1;
        PUSH R1,R1 TO [SP];                  //将 R1 压栈
        R1 = [BP+3];                         //取 B 口的方向向量
        [P_IOB_Dir] = R1;
        R1 = [BP+4];                         //取 B 口的数据向量
        [P_IOB_Data] = R1;
        R1 = [BP+5];                         //取 B 口的属性向量
        [P_IOB_Attrib] = R1;
        POP R1,R1 FROM [SP];                 //出栈
        POP BP,BP FROM [SP];                 //出栈
        RETF;
        .ENDP                                //子程序结束
```

方法步骤：

(1)新建项目, 项目名称 EX3。

(2)该项目下新建汇编文件, 文件名称 EX3ASM。

(3)在汇编文件中键入范例汇编源代码（如图 6.61）。

(4)该项目下新建 C 文件， 文件名称 EX3.C。

(5)在 C 文件中键入范例 C 源代码（如图 6.62）。

(6)保存项目。

(7)编译调试该程序。

※注意：因为程序包含头文件，所以应指明头文件路径。

方法步骤：

1） 单击菜单栏中 Tools/Option,弹出 Option 对话框，对话框中有五个标签页。

2） 单击 Directories 标签。出现如图 6.63界面。

3） 在 Show Directories for 文本框中，选择 Include Files 并单击 Creat a new item 按钮。在窗口中会出现 Browse 按钮。

4） 单击 Browse 按钮选择包含的头文件(include 文件夹)，如 hardware.inc。指明头文件路径结束。

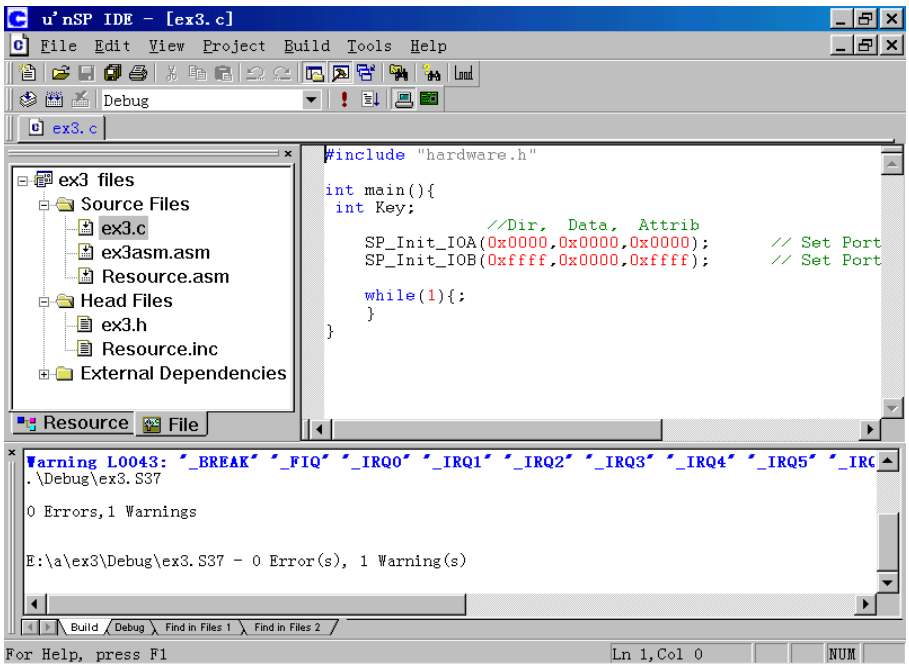


图6.61　EX3 C 文件源代码界面

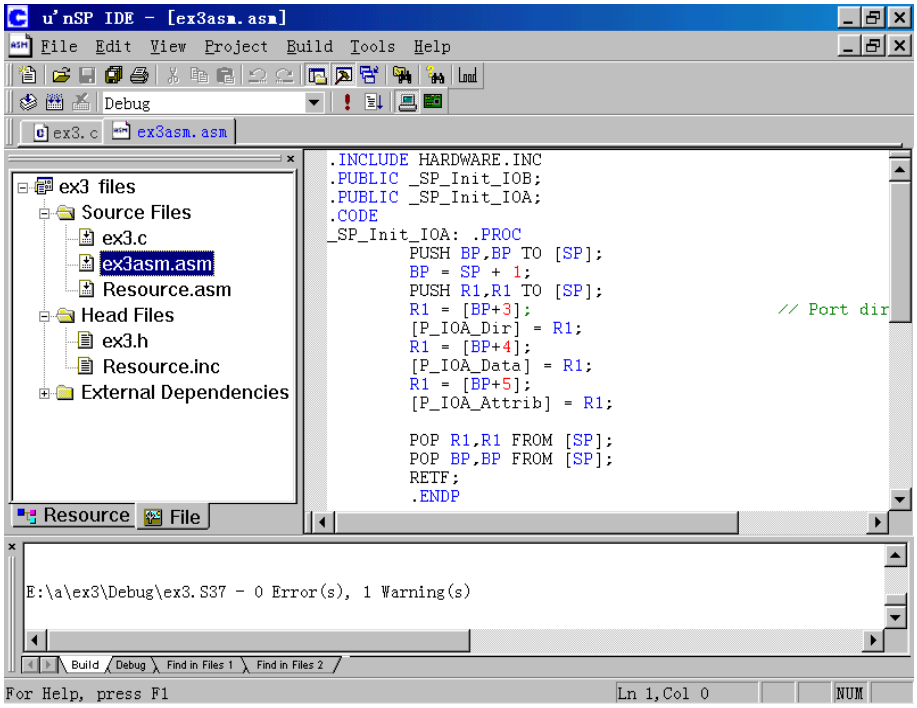


图6.62　EX3 汇编文件源代码界面

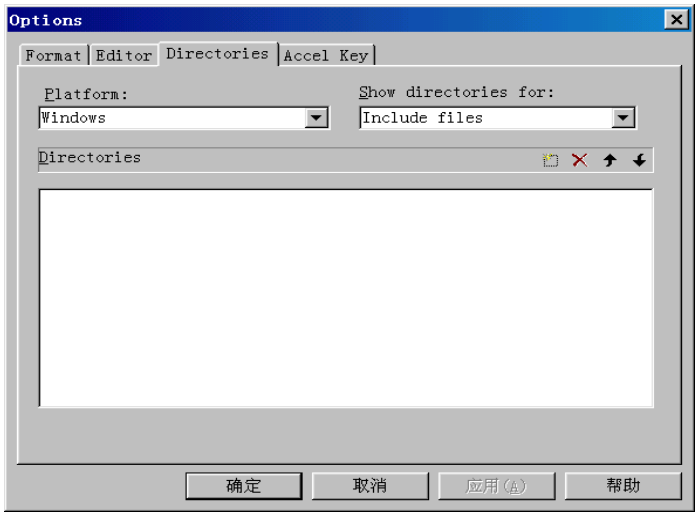


图6.63 Option 对话框

```
//*****************************************************************//
EX3        END
//*****************************************************************//
```

程序6-4 带有资源文件(.rc) C 与 ASM 文件

范例：自动放音实验

```
//*****************************************************************//
EX4
//*****************************************************************//
```

主程序为 C 文件，程序如下：

```
#include "a2000.h"                              //包括 A2000.h

#define SPEECH_1    0                           // SPEECH_1 = 0 表示最大语音索引号
#define DAC1    1                               // 第一通道
#define DAC2    2                               // 第二通道
#define Ramp_UpDn_Off 0                         // 禁止音量增/减调节
#define Ramp_Up_On    1                         // 仅允许音量增调节
#define Ramp_Dn_On    2                         // 仅允许音量减调节
#define Ramp_UpDn_On    3                       // 允许音量增减调节

main()
{
    SACM_A2000_Initial(1);                      // 设置为自动放音方式
    SACM_A2000_Play(SPEECH_1, DAC1+DAC2,Ramp_UpDn_On);
```

```
                                        // 播放资源的 SACM_A2000 格式语音和乐曲
    while(1)
    {
        SACM_A2000_ServiceLoop();       //从 SRAM 中获取语音数据，对其进行解码
                                        //等待中断服务程序将其送出 DAC 通道
    }
}
```

中断服务程序的 ASM 文件,程序如下:

```
.TEXT
.INCLUDE hardware.inc                   //包含头文件
.INCLUDE a2000.inc
.INCLUDE Resource.inc
.PUBLIC _FIQ;                           //声明_FIQ 中断
_FIQ:
    PUSH R1,R4 TO [SP];
    R1 = C_FIQ_TMA;
    TEST R1,[P_INT_Ctrl]
    JNZ L_FIQ_TimerA;                   //定时器 A 的中断入口

L_FIQ_PWM:                              //中断 PWM FIQ 入口
    R1 = C_FIQ_PWM;
    [P_INT_Clear] = R1;
    POP R1,R4 FROM [SP];

L_FIQ_TimerA:                           //中断子程序 FIQ_TimerA
    [P_INT_Clear] = R1;                 //清中断
    CALL F_FIQ_Service_SACM_A2000;      // SACM-A2000 定时器 A FIQ 解码
    POP R1,R4 FROM [SP];                //出栈
    RETI;

L_FIQ_TimerB:                           //中断子程序 FIQ_TimerB
    [P_INT_Clear] = R1;                 //清中断
    POP R1,R4 FROM [SP];                //出栈
    RETI
```

方法步骤：

(1)新建项目, 项目名称 EX4。
(2)该项目下新建汇编文件, 文件名称 SYSTEM.ASM。
(3)在汇编文件中键入范例汇编源代码（如图 6.64）。
(4)该项目下新建 C 文件， 文件名称 MAIN.C。
(5)在 C 文件中键入范例 C 源代码（如图 6.65）。
(6)在源文件组中添加 HARDWARE.ASM 文件
(7)在头文件组中添加 A2000.INC、HARDWARE.INC 和 RESOURCE.INC 头文件
(8)在资源文件视窗中，添加 RES_A27 、RES_A32 和 RES_A38 资源文件

(9)保存项目。

(10)编译调试该程序。

(11)下载到仿真板中

※注意：当编译时，会出现如下错误，

Error L0080: The external symbol "T_SACM_A2000_SpeechTable" has not a public definition.

这时在系统自动生成的 Resource.asm 文件中添加如下内容即可：

```
.PUBLIC T_SACM_A2000_SpeechTable;
T_SACM_A2000_SpeechTable:
.DW _RES_A27_SA;                                    //资源起始地址
.DW _RES_A32_SA
.DW _RES_A38_SA
```

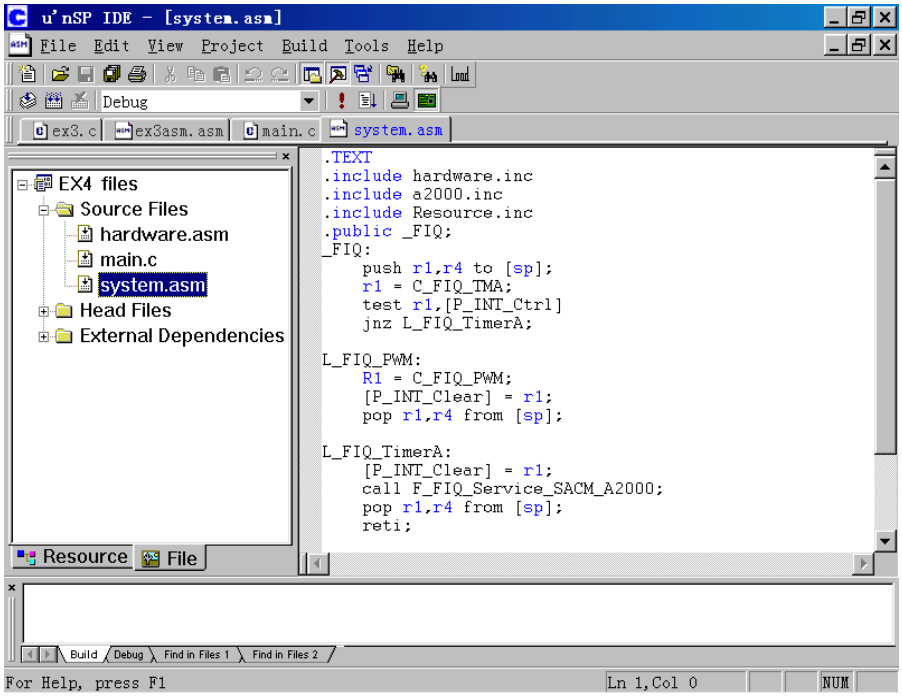


图6.64 EX4 汇编文件源代码界面

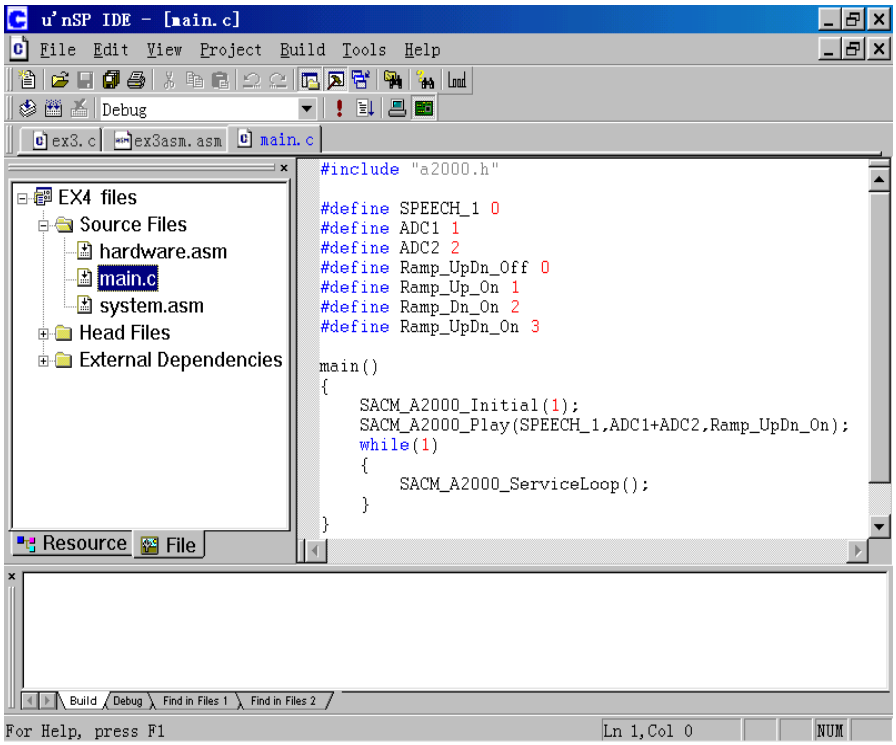


图6.65　EX4 C 文件源代码界面

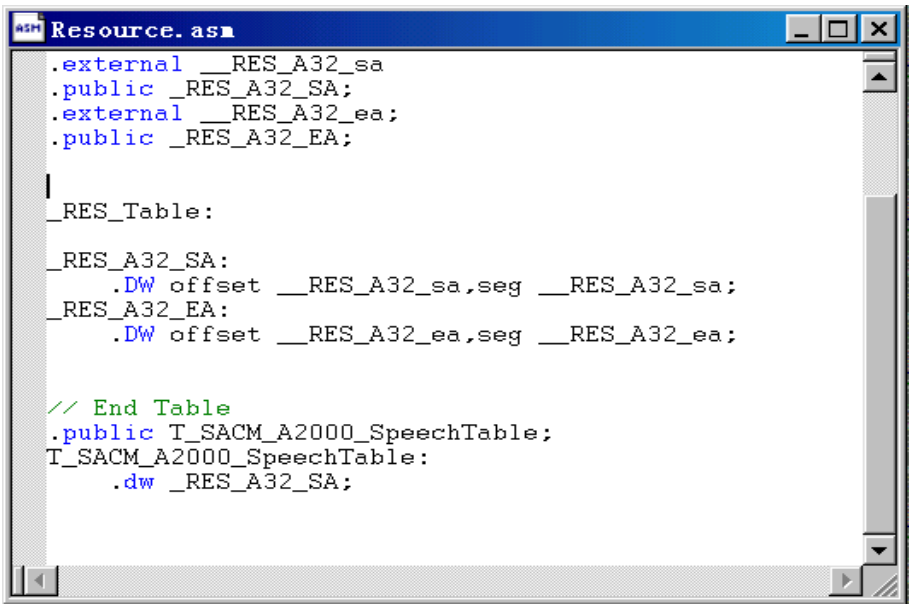


图6.66　EX4 RESOURCE .ASM　编译成功后的文件界面

```
//*************************************************************//
EX4       END
//*************************************************************//
```