Keil 程序调试窗口

上一讲中我们学习了几种常用的程序调试方法，这一讲中将介绍 Keil 提供各种窗口如输出窗口、观察窗口、存储器窗口、反汇编窗口、串行窗口等的用途，以及这些窗口的使用方法，并通过实例介绍这些窗口在调试中的使用。

一、程序调试时的常用窗口

Keil 软件在调试程序时提供了多个窗口，主要包括输出窗口（Output Windows）、观察窗口（Watch&Call Statck Windows）、存储器窗口（Memory Window）、反汇编窗口（Dissambly Window）串行窗口（Serial Window）等。进入调试模式后，可以通过菜单 View 下的相应命令打开或关闭这些窗口。

图 1 是输出窗口、观察窗口和存储器窗口，各窗口的大小可以使用鼠标调整。进入调试程序后，输出窗口自动切换到 Command 页。该页用于输入调试命令和输出调试信息。对于初学者，可以暂不学习调试命令的使用方法。

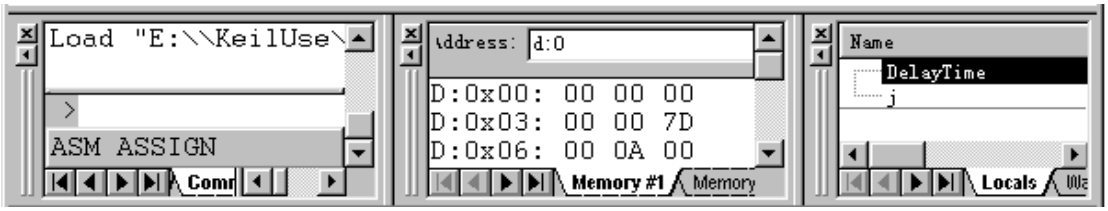

图 1 调试窗口（命令窗口、存储器窗口、观察窗口）

1、存储器窗口

存储器窗口中可以显示系统中各种内存中的值，通过在 Address 后的编缉框内输入“字母：数字”即可显示相应内存值，其中字母可以是 C、D、I、X，分别代表代码存储空间、直接寻址的片内存储空间、间接寻址的片内存储空间、扩展的外部 RAM 空间，数字代表想要查看的地址。例如输入 D：0 即可观察到地址 0 开始的片内 RAM 单元值、键入 C：0 即可显示从 0 开始的 ROM 单元中的值，即查看程序的二进制代码。该窗口的显示值可以以各种形式显示，如十进制、十六进制、字符型等，改变显示方式的方法是点鼠标右键，在弹出的快捷菜单中选择，该菜单用分隔条分成三部份，其中第一部份与第二部份的三个选项为同一级别，选中第一部份的任一选项，内容将以整数形式显示，而选中第二部份的 Ascii 项则将以字符型式显示，选中 Float 项将相邻四字节组成的浮点数形式显示、选中 Double 项则将相邻 8 字节组成双精度形式显示。第一部份又有多个选择项，其中 Decimal 项是一个开关，如果选中该项，则窗口中的值将以十进制的形式显示，否则按默认的十六进制方式显示。Unsigned 和 Signed 后分别有三个选项：Char、Int、Long，分别代表以单字节方式显示、将相邻双字节组成整型数方式

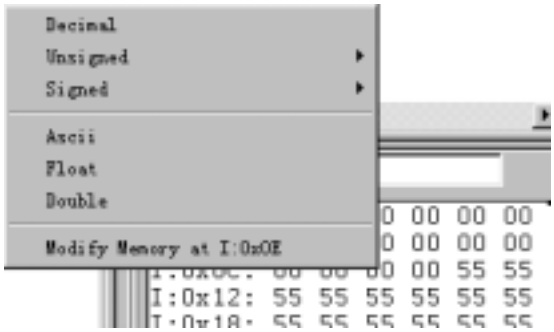

图 2 存储器数值各种方式显示选择

显示、将相邻四字节组成长整型方式显示，而 Unsigned 和 Signed 则分别代表无符号形式和有符号形式，究竟从哪一个单元开始的相邻单元则与你的设置有关，以整型为例，如果你输入的是 I:0，那么 00H 和 01H 单元的内容将会组成一个整型数，而如果你输入的是 I：1，01H 和 02H 单元的内容全组成一个整型数，以此类推。有关数据格式与 C 语言规定相同，请参考 C 语言书籍，默认以无符号单字节方式显示。第三部份的 Modify Memory at X:xx 用于更改鼠标处的内存单元值，选中该项即出现如图 3 所示的对话框，可以在对话框内输入要修改的内容。

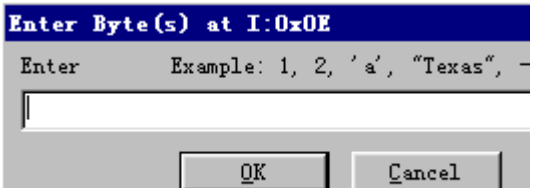

图 3 存储器的值的修改

2、工程窗口寄存器页

图 4 是工程窗口寄存器页的内容，寄存器页包括了当前的工作寄存器组和系统寄存器，系统寄存器组有一些是实际存在的寄存器如 A、B、DPTR、SP、PSW 等，有一些是实际中并不存在或虽然存在却不能对其操作的如 PC、Status 等。每当程序中执行到对某寄存器的操作时，该寄存器会以反色（蓝底白字）显示，用鼠标单击然后按下 F2 键，即可修改该值。

3、观察窗口

观察窗口是很重要的一个窗口，工程窗口中仅可以观察到工作寄存器和有限的寄存器如 A、B、DPTR 等，如果需要观察其它的寄存器的值或者在高级语言编程时需要直接观察变量，就要借助于观察窗口了。

其它窗口将在以下的实例中介绍。

一般情况下，我们仅在单步执行时才对变量的值的变化感兴趣，全速运行时，变量的值是不变的，只有在程序停下来之后，才会将这些值最新的变化反映出来，但是，在一些特殊场合下我们也可能需要在全速运行时观察变量的变化，此时可以点击 View->Periodic Window Updata（周期更新窗口），确认该项处于被选中状态，即可在全速运行时动态地观察有关值的变化。但是，选中该项，将会使程序模拟执行的速度变慢。

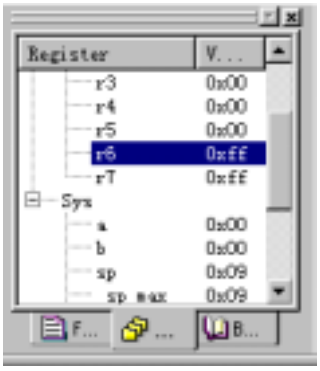

图 4 工程窗口寄存器页

二、各种窗口在程序调试中的用途

以下通过一个高级语言程序来说明这些窗口的使用。例 2：

```
#include "reg51.h"
sbit P1_0=P1^0;     //定义 P1.0
void mDelay(unsigned char DelayTime)
{     unsigned int j=0;
      for(;DelayTime>0;DelayTime--)
      {    for(j=0;j<125;j++)    {;}     }
}
void main()
{     unsigned int i;
      for(;;){    mDelay(10);    // 延 时   10 毫秒
                  i++;
                  if(i==10)
                  {       P1_0=!P1_0;
                          i=0;        }
      }   }
```

这个程序的工作过程是：不断调用延时程序，每次延时 10 毫秒，然后将变量 I 加 1，随后对变量 I 进行判断，如果 I 的值等于 10，那么将 P1.0 取反，并将 I 清 0，最终的执行效果

是 P1.0 每 0.1S 取反一次。

输入源程序并以 exam2.c 为文件名存盘，建立名为 exam2 的项目，将 exam2.c 加入项目，编译、连接后按 Ctrl+F5 进入调试，按 F10 单步执行。注意观察窗口，其中有一个标签页为 Locals，这一页会自动显示当前模块中的变量名及变量值。可以看到窗口中有名为 I 的变量，其值随着执行的次数而逐渐加大，如果在执行到 mDelay(10)行时按 F11 跟踪到 mDelay 函数内部，该窗口的变量自动变为 DelayTime 和 j。另外两个标签页 Watch #1 和 Watch #2 可以加入自定义的观察变量，点击“ type F2 to edit ”然后再按 F2 即可输入变量，试着在 Watch #1 中输入 I，观察它的变化。在程序较复杂，变量很多的场合，这两个自定义观察窗口可以筛选出我们自己感兴趣的变量加以观察。观察窗口中变量的值不仅可以观察，还可以修改，以该程序为例，I 须加 10 次才能到 10，为快速验证是否可以正确执行到 P1_0=!P1_0 行，点击 I 后面的值，再按 F2，该值即可修改，将 I 的值改到 9，再次按 F10 单步执行，即可以很快执行到 P1_0=!P1_0 程序行。该窗口显示的变量值可以以十进制或十六进制形式显示，方法是在显示窗口点右键，在快捷菜单中选择如图 5 所示。

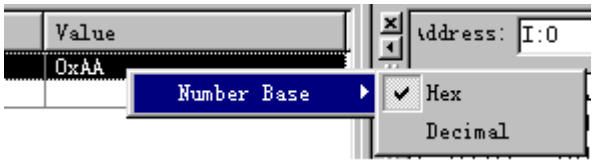

图 5 设定观察窗的显示方式

点击 View->Dissambly Window 可以打开反汇编窗口，该窗口可以显示反汇编后的代码、源程序和相应反汇编代码的混合代码，可以在该窗口进行在线汇编、利用该窗口跟踪已找行的代码、在该窗口按汇编代码的方式单步执行，这也是一个重要的窗口。打开反汇编窗口，点击鼠标右键，出现快捷菜单，如图 6 所示，其中 Mixed Mode 是以混合方式显示，Assembly Mode 是以反汇编码方式显示。

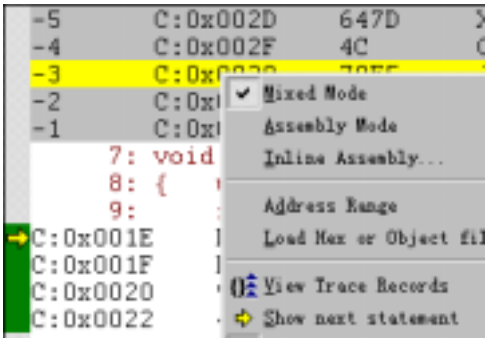

图 6 反汇编窗口

程序调试中常使用设置断点然后全速运行的方式，在断点处可以获得各变量值，但却无法知道程序到达断点以前究竟执行了哪些代码，而这往往是需要了解的，为此，Keil 提供了跟踪功能，在运行程序之前打开调试工具条上的允许跟踪代码开关，然后全速运行程序，当程序停止运行后，点击查看跟踪代码按钮，自动切换到反汇编窗口，如图 6 所示，其中前面标有“ - ”号的行就是中断以前执行的代码，可以按窗口边的上卷按钮向上翻查看代码执行记录。

利用工程窗口可以观察程序执行的时间，下面我们观察一下该例中延时程序的延时时间是否满足我们的要求，即是否确实延时 10 毫秒，展开工程窗口 Regs 页中的 Sys 目录树，其中的 Sec 项记录了从程序开始执行到当前程序流逝的秒数。点击 RST 按钮以复位程序，Sec 的值回零，按下 F10 键，程序窗口中的黄色箭头指向 mDelay(10)行，此时，记录下 Sec 值为 0.00038900，然后再按 F10 执行完该段程序，再次查看 Sec 的值为 0.01051200，两者相减大约是 0.01 秒，所以延时时间大致是正确的。读者可以试着将延时程序中的 unsigned int 改为 unsigned char 试试看时间是否仍正确。注意，使用这一功能的前提是在项目设置中正确设置晶振的数值。

Keil 提供了串行窗口，我们可以直接在串行窗口中键入字符，该字符虽不会被显示出来，但却能传递到仿真 CPU 中，如果仿真 CPU 通过串行口发送字符，那么这些字符会在串行窗口显示出来，用该窗口可以在没有硬件的情况下用键盘模拟串口通讯。下面通过一个例子说明 Keil 串行窗口的应用。该程序实现一个行编缉功能，每键入一个字母，会立即回显到窗

口中。编程的方法是通过检测 RI 是否等于 1 来判断串行口是否有字符输入，如果有字符输入，则将其送到 SBUF，这个字符就会在串行窗口中显示出来。其中 ser_init 是串行口初始化程序，要使用串行口，必须首先对串行口进行初始化。例 3：

```
    MOV     SP,#5FH    ;堆栈初始化
    CALL    SER_INIT ;串行口初始化
LOOP:
    JBC     RI,NEXT  ;如果串口接收到字
    符，转
    JMP     LOOP     ;否则等待接收字符
NEXT:
    MOV     A,SBUF    ;从 SBUF 中取字符
    MOV     SBUF,A    ;回送到发送 SBUF 中
SEND:
    JBC     TI,LOOP    ;发送完成，转 LOOP
    JMP     SEND       ;否则等待发送完
SER_INIT:              ;中断初始化
    MOV     SCON,#50H
    ORL     TMOD,#20H
    ORL     PCON,#80H
    MOV     TH1,#0FDH ;设定波特率
    SETB    TR1     ;定时器 1 开始运行
    SETB    REN     ;允许接收
    SETB    SM2
    RET
    END
```

输入源程序，并建立项目，正确编译、连接，进入调试后，全速运行，点击串行窗口 1 按钮，即在原源程序窗口位置出现一个空白窗口，击键，相应的字母就会出现在该窗口中。在窗口中击鼠标右键，出现一个弹出式菜单，选择“ Ascii Mode ”即以 Ascii 码的方式显示接收到的数据；选择“ Hex Mode ”以十六进制码方式显示接收到的数据；选择“ Clear Window ”可以清除窗口中显示的内容。

由于部份 CPU 具有双串口，故 Keil 提供了两个串行窗口，我们选用的 89C51 芯片只有一个串行口，所以 Serial 2 串行窗口不起作用。

小技巧：凡是鼠标单击然后按 F2 的地方都可以用鼠标连续单击两次（注意：不是双击）来替代。