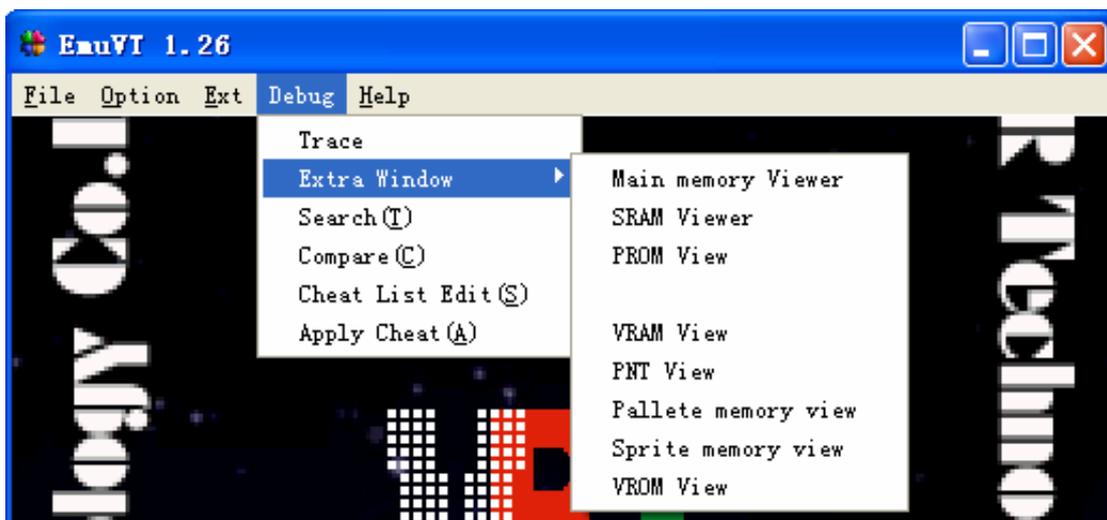


Debug Windows users note:

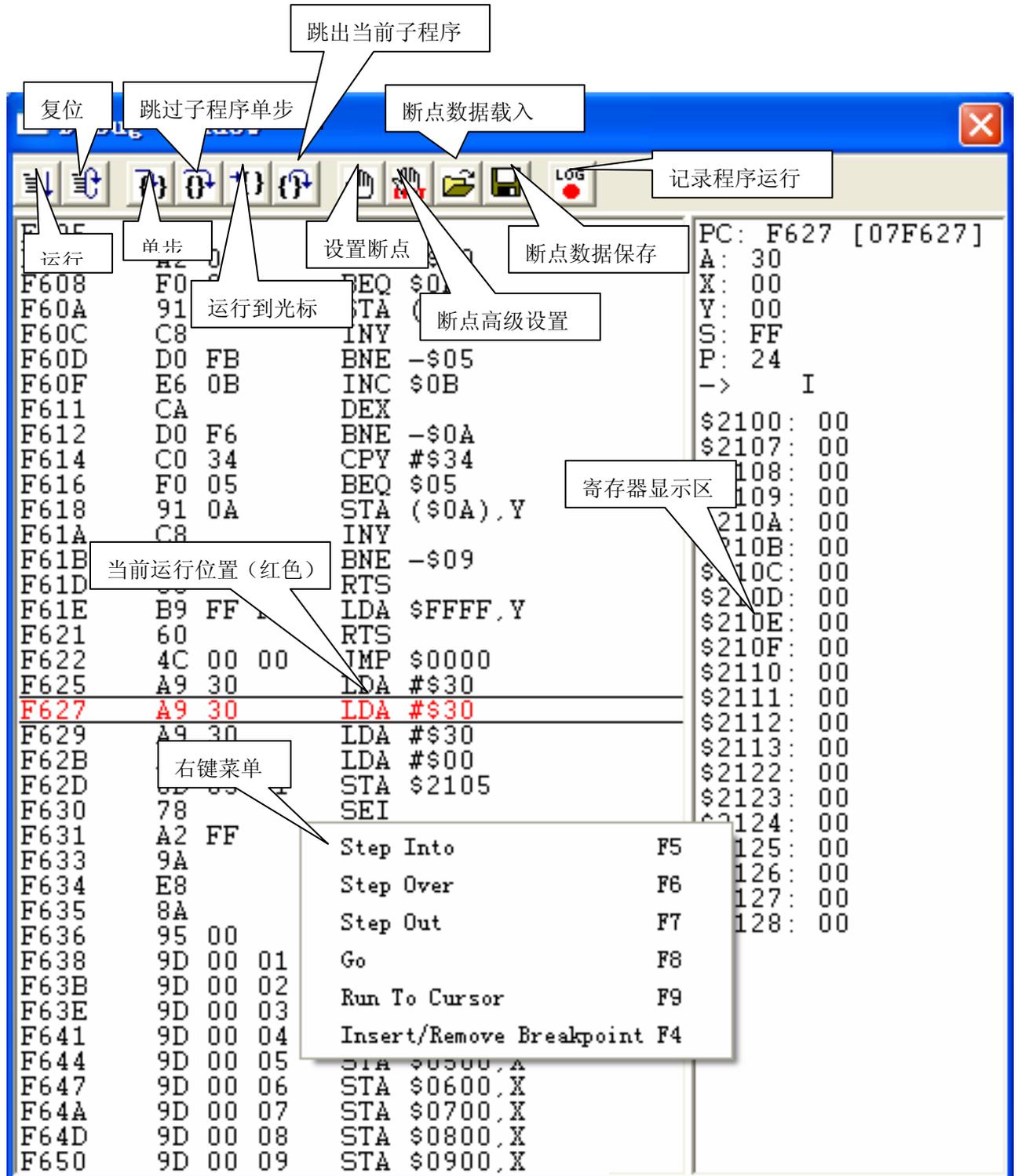
设置完之后, Trace Set 窗口可以关闭也可以不关闭, 点运行那个按钮 (Debug Windows 左起第一个按钮) 才能跟踪, 或者点复位那个按钮 (Debug Windows 左起第一个按钮) 然后点运行那个按钮 (Debug Windows 左起第二个按钮)。在 Debug 的过程中不要把 Debug Windows 关闭, 否则不能跟踪。Trace Set 窗口中的“Enable”选项只是显示这个 break point 是否使能,而且是反的,没勾的时候是使能, 有勾的时候是禁止。

Debug 菜单



| | |
|-----------------|---------|
| Trace | 跟踪调试窗口 |
| Extra Window | 内存查看器 |
| Search | 主内存搜索查找 |
| Compare | 主内存比较 |
| Cheat List Edit | 金手指设置器 |
| Apply Cheat | 金手指使能 |

Trace 跟踪调试窗口



The screenshot shows the Trace window of the VT1682 Emulator. The window title is "Trace 跟踪调试窗口". The interface includes a toolbar with various icons and a main display area showing assembly code and register values.

Annotations:

- 复位 (Reset)
- 跳出当前子程序 (Exit current subprogram)
- 跳过子程序单步 (Skip subprogram single step)
- 断点数据载入 (Load breakpoint data)
- 记录程序运行 (Record program execution)
- 运行 (Run)
- 单步 (Single step)
- 设置断点 (Set breakpoint)
- 断点数据保存 (Save breakpoint data)
- 运行到光标 (Run to cursor)
- 断点高级设置 (Advanced breakpoint settings)
- 寄存器显示区 (Register display area)
- 当前运行位置 (红色) (Current execution position (red))
- 右键菜单 (Right-click menu)

Assembly Code Display:

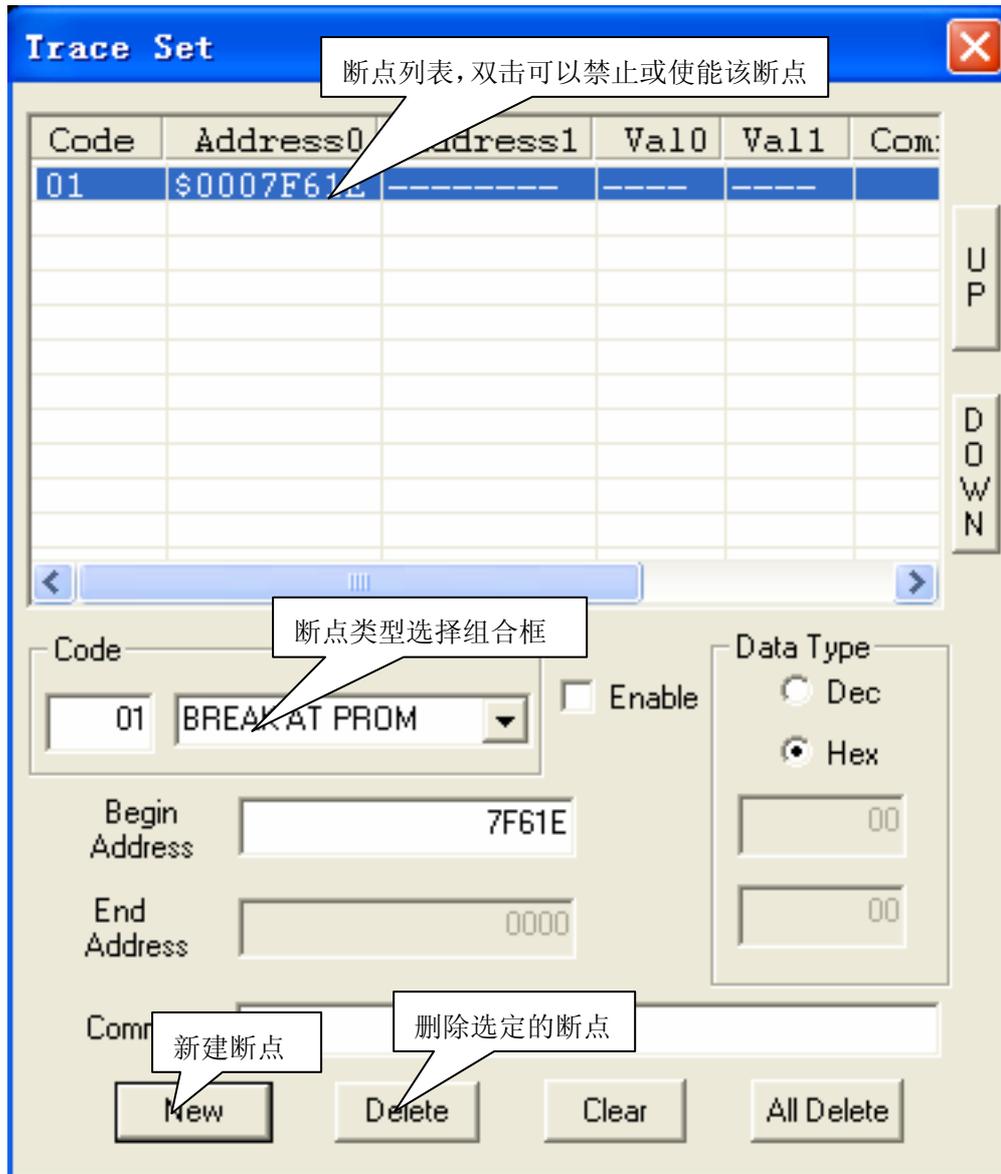
```

PC: F627 [07F627]
A: 30
X: 00
Y: 00
S: FF
P: 24
->      I
$2100: 00
$2107: 00
$2108: 00
$2109: 00
$210A: 00
$210B: 00
$210C: 00
$210D: 00
$210E: 00
$210F: 00
$2110: 00
$2111: 00
$2112: 00
$2113: 00
$2122: 00
$2123: 00
$2124: 00
$2125: 00
$2126: 00
$2127: 00
$2128: 00
F608 F0 00 00 BEQ $00,X
F60A 91 00 00 STA ($00),Y
F60C C8 00 00 INY
F60D D0 FB BNE -$05
F60F E6 0B INC $0B
F611 CA 00 00 DEX
F612 D0 F6 BNE -$0A
F614 C0 34 CPY #$34
F616 F0 05 BEQ $05
F618 91 0A STA ($0A),Y
F61A C8 00 00 INY
F61B D0 09 BNE -$09
F61D 00 00 00 RTS
F61E B9 FF LDA $FFFF,Y
F621 60 00 00 RTS
F622 4C 00 00 JMP $0000
F625 A9 30 LDA #$30
F627 A9 30 LDA #$30
F629 A9 30 LDA #$30
F62B 00 00 00 LDA #$00
F62D 00 00 00 STA $2105
F630 78 00 00 SEI
F631 A2 FF LDA $2105,X
F633 9A 00 00 STA $210A,X
F634 E8 00 00 STA $2108,X
F635 8A 00 00 STA $210A,X
F636 95 00 00 STA $2105,X
F638 9D 00 01 STA $2105,X
F63B 9D 00 02 STA $2105,X
F63E 9D 00 03 STA $2105,X
F641 9D 00 04 STA $2105,X
F644 9D 00 05 STA $2105,X
F647 9D 00 06 STA $0600,X
F64A 9D 00 07 STA $0700,X
F64D 9D 00 08 STA $0800,X
F650 9D 00 09 STA $0900,X
  
```

Right-click menu:

- Step Into F5
- Step Over F6
- Step Out F7
- Go F8
- Run To Cursor F9
- Insert/Remove Breakpoint F4

按下“断点高级设置”：



断点类型：

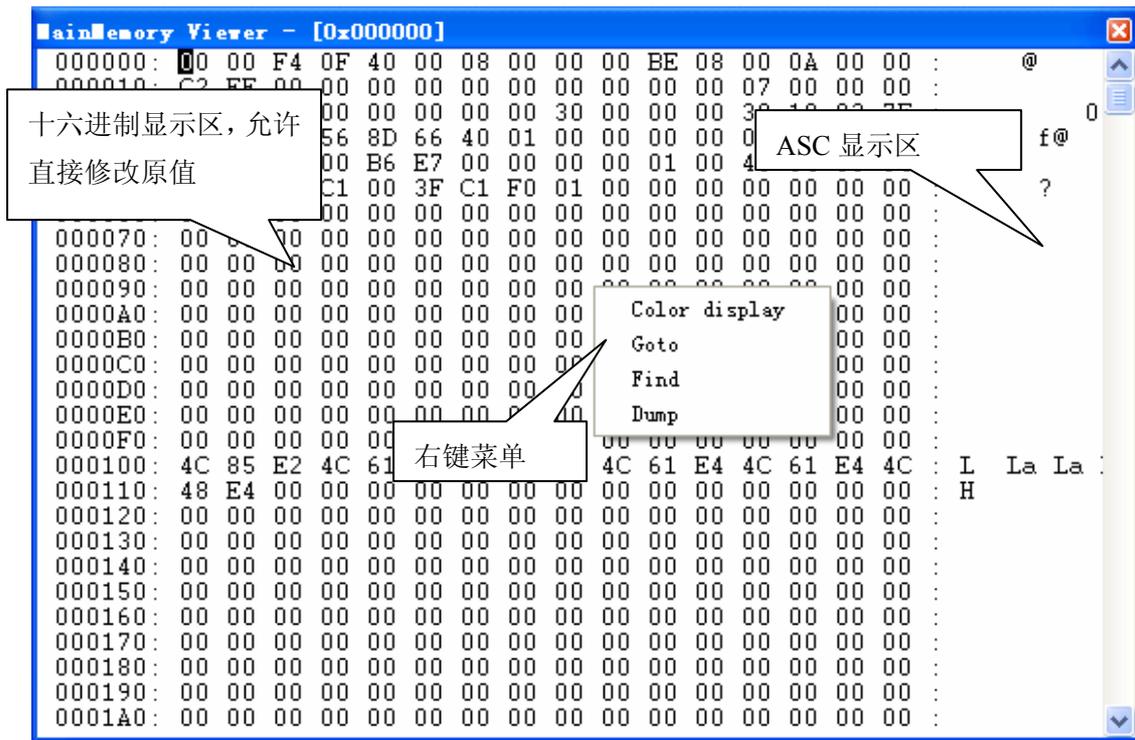
| | |
|-------------------|--------------------|
| BREAK AT PROM | 运行到物理地址中断 |
| BREAK AT ADDRESS | 运行到逻辑地址中断 |
| BREAK IN ADDRESS | 运行到逻辑地址范围中断 |
| BREAK IN PROM | 运行到物理地址范围中断 |
| BREAK AT WRITE | 写指定地址（寄存器）中断 |
| BREAK IN WRITE | 写指定地址（寄存器）范围中断 |
| BREAK AT WRITE AT | 以指定值写指定地址（寄存器）中断 |
| BREAK IN WRITE AT | 以指定值写指定地址（寄存器）范围中断 |
| BREAK AT WRITE IN | 以指定范围值写指定地址（寄存器）中断 |

| | |
|-------------------|-----------------------|
| BREAK IN WRITE IN | 以指定范围值写指定地址（寄存器）范围中断 |
| BREAK AT READ | 读指定地址（寄存器）中断 |
| BREAK IN READ | 读指定地址（寄存器）范围中断 |
| BREAK AT READ AT | 读指定地址（寄存器）得到指定值中断 |
| BREAK IN READ AT | 读指定地址（寄存器）范围得到指定值中断 |
| BREAK AT READ IN | 读指定地址（寄存器）得到指定范围值中断 |
| BREAK IN READ IN | 读指定地址（寄存器）范围得到指定范围值中断 |
| EXT IRQ BREAK | 设定扩展 IRQ 中断断点 |
| NMI BREAK | 设定 NMI 中断断点 |
| IRQ BREAK | 设定 IRQ 中断断点 |
| SCPU IRQ BREAK | 设定 SCPU IRQ 中断断点 |
| TIMER IRQ BREAK | 设定 TIMER IRQ 中断断点 |

举例：如果欲设置这样一个断点，当对\$200 到\$20F 范围有写操作，且写入的值为\$00 到\$7F 之间的任意值时。首先我们点 New 按钮，然后在组合框选择“BREAK IN WRITE IN”，代码是 25，在 Begin Address 中输入 200，在 End Address 中输入 210，选择 Hex 模式，在 Date 靠上的一个输入框输入 00，靠下一个输入框输入 80，这样就设置好了一个断点。

Extra Window 内存查看器

| | |
|------------------------|---------------|
| Main memory Viewer | 主内存查看器 |
| SRAM Viewer | 扩充内存查看器 |
| PROM Viewer | 程序空间查看器 |
| VRAM Viewer | PPU RAM 内存查看器 |
| PNT Viewer | PPU PNT 内存查看器 |
| Pallette memory Viewer | 调色板内存查看器 |
| Sprite memory Viewer | 卡通内存查看器 |
| VROM Viewer | PPU 点阵区查看 |



右键菜单说明:

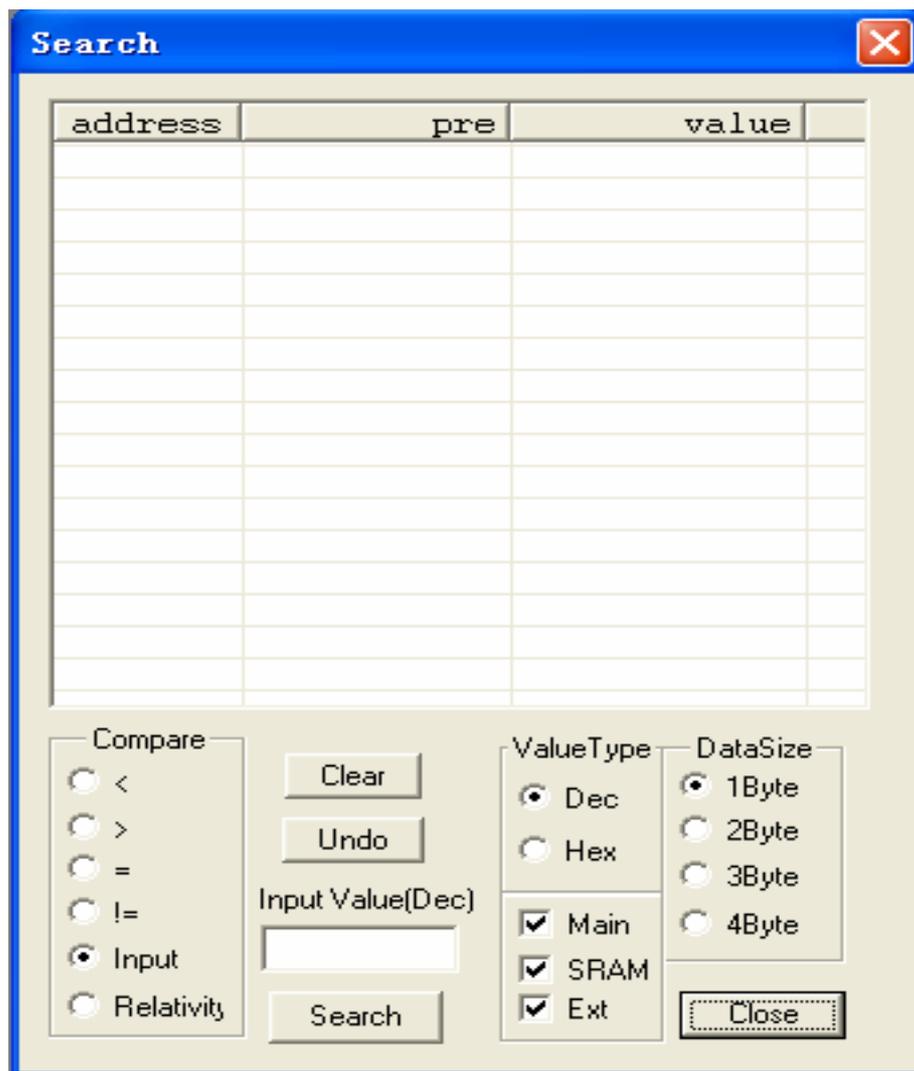
Color Display 以特殊颜色标示特殊的内容 ÷

Goto 跳到指定的内存偏移地址

Find 在当前内存区域查找内容

Dump 将当前内存区域内容以二进制或文本方式存到指定文件

Search 主内存搜索查找



这个功能类似于“金山游侠”或“FPE”之类的游戏工具，用来搜索找出特定内存单元的含义，比如做一些游戏的 HACK，我们需要找出游戏中“命”的数量是存在哪个内存单元，以便我们做出不死“命”的游戏版本来。