第3章 单片机程序的编写与下载——Keil的应用和通信板的制作

要使单片机按我们的意愿工作，就要给它编程序。单片机内部有一个空间，是专门用来放程序的，叫程序存储器。怎样编写程序并让单片机“理解”？又怎样把程序下载到它的存储器里？这是本章实验所要讨论的问题。

3.1 单片机的“语言”

3.1.1 二进制机器语言及十六进制表达

和计算机一样，单片机使用的语言是机器语言，也就是用“0”和“1”这两个“字母”来表达一切的语言。这显然是世界上使用符号最少的语言。在机器里它常以“0”和“1”构成的8位字符串为基本单位（字节）来表示事物，如用01000001来表示字母A（ASCII码，属机器代码）。可以看出，这种二进制字符串书写起来很长且不方便，于是我们用十六进制字符来对它进行缩写。刚好可以把4个二进制字符，按二进制与十六进制的数制转换关系缩写成一个十六进制的字符，这样8位二进制字符串表示的一个字节就可以用2位十六进制数的字符来表示。例如前面提到的字母A，用十六进制字符表示的机器码就是41H（H表示它左侧的数字41是十六进制字符）。现在几乎所有的机器码都用十六进制字符来表达，但不要忘记，它们只是“0”和“1”的机器语言的缩写。

3.1.2 指令的物理实在

在单片机里，指令也是“0”和“1”字符串构成的机器码。但它跟表达字母A之类的代码不一样，它们是使机器产生动作的符号，每一位都与一系列逻辑运算电路相关，其他的二进制数及代码只是它处理的对象。指令与硬件电路息息相关。

单片机生产厂家用“0”和“1”设计机器指令，同时又要构造出对应的数字逻辑电路解释执行系统。所以，每一种类型的单片机都有它自己的指令系统，我们只能使用，不能创造。也就是说，我们编程用的每一条指

令，都必须在指令表中找得到，这一点往往被初学者所忽略。

指令的处理对象是数据，所以指令一般由操作码(Opcode)和操作数(Operand)构成。

3.1.3 助记符和汇编

用“0”和“1”构成的机器指令，单片机能用它的逻辑电路来“理解”，我们却看不懂。必须用一些我们可以理解的助记符(Mnemonic)对这些机器指令中的操作码进行代换。被代换之后形成的指令叫符号指令，由它们构成的语言叫汇编语言，用汇编语言写出的程序叫汇编语言程序(源程序)。之所以叫汇编(Assembly)，是因为这些程序里的符号指令，在被下载到机器里之前，必须用翻译工具把它汇合编译之后还原成机器码。这个翻译工具也是程序，不过，它不是单片机程序而是计算机程序，叫汇编程序。

助记符是表达指令功能的英文缩写，可根据英文字母来记忆并理解其功能。

51系列单片机的汇编程序有Proteus的ASEM51(位于Proteus程序文件的\TOOLS文件夹下)，Keil的A51(位于Keil程序文件的\C51\BIN文件夹下)。

3.1.4 修改机器码的实验

在第2章里，我们利用Proteus软件，做了加载文件TWOBALL.HEX到单片机里，进行仿真运行的实验，见2.7.1节。当时的实验结果是：两个低电平在P1口双双来回移动。在这一节里，我们要把两个低电平改在P0口移动，要做到这一点，我们需要改动这个以十六进制表达的Intel格式的机器码文件TWOBALL.HEX。这种文件格式是Intel公司提出的、带程序存储区信息、按地址排列、以字节为单位、十六进制数字表示的数据信息格式，常用来保存单片机或其他处理器的目标程序代码，一般的下载器或编程器都支持这种格式。

1. 修改 TWOBALL.HEX 机器码

复制TWOBALL.HEX文件，并用Windows附件里的记事本工具打开它，如下所示。

```
:03000000020100FA
:10010000743F7B06F5907800790070A023126031B54
:10011000BB00F17B06F5907800790070A0231262346
:100120001BBB00F12100180000B800FA190000B94B
:0901300000F41A0000BA00EE22EE
:00000001FF
```

以上每一行的前8位数分别记录这一行代码的字节数(2位数)、起始地址(4位数)、类型(2位数)，末尾的2位数表示校验和，中间部分才是程序代码。显然，我们看不懂这种代码。

研究此程序运行时内部寄存器的变化情况，我们发现P1(P1口锁存器)和Acc(A的寄存器表示)的值始终相等，如2.7.1节实验截图所示。这说明P1的数据是由A送来的，对应的符号指令是“MOV P1，A”。查单片机特殊功能寄存器地址表，得到P1的内存地址是90H，查指令表，这类符号指令“MOV direct，A”(direct直接指内存地址)的代码是(F5)

(direct),把 direct 用 90 代替,得出这条指令的机器码是 F590。我们在上面的代码中发现有两处 F590,拟对它们进行修改。

把原来 P1 口输出的数改为 P0 口输出,可尝试把符号指令"MOV P1, A"改成"MOV P0, A"来实现。由于 P0 口的地址是 80H,"MOV P0, A"这条指令的机器码就是 F580,用它代换上面的 F590,修改后的文件代码如下。

```
:03000000020100FA
:10010000743F7B06F580780079007A023126031B64
:10011000BB00F17B06F580780079007A0231262356
:100120001BBB00F12100180000B800FA190000B94B
:0901300000F41A0000BA00EE22EE
:00000001FF
```

修改内容涉及第 2 行和第 3 行,这两行末尾的两位校验和也做了相应的调整。修改完成后,重新保存文件。注意,如果要改名另存为新文件,须保证文件后缀名是. HEX。有时,记事本工具会自动加上它自己的文件后缀名. txt,如 P0-TWOBALL. HEX. txt。如果出现这种情况,请用重命名的方法,把后面的尾巴. txt 去掉。如果看不到后缀名,请单击资源管理器中的菜单"工具"→"文件夹选项"→"查看"选项卡,找到"☑隐藏已知文件类型的扩展名",把其前面小方框(复选框)中的勾单击去掉,变成"□隐藏已知文件类型的扩展名",再单击"确定"按钮退出。

2. 运行代码

按 2.7.1 节 Proteus 仿真实验的方法,重新加载文件,运行并查看寄存器,如图 3-1 所示。

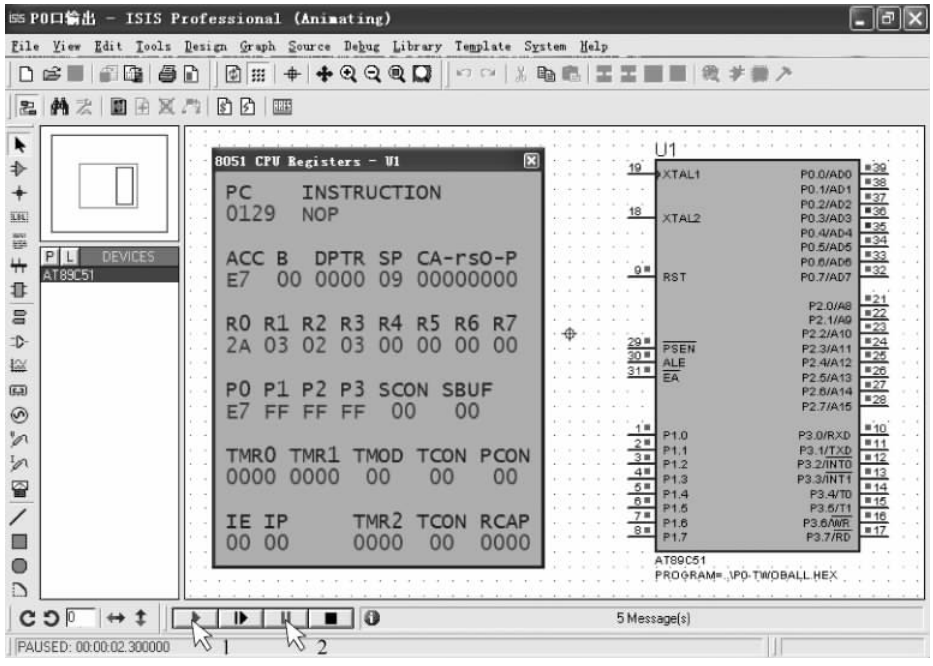

图 3-1 由 P1 口改为 P0 口输出 TWOBALL. HEX 的运行情况

可以看出，两个低电平（即所谓的“TWOBALL”）在P0口移动，同时，P0锁存器的数据发生变化，而P1口则处于初始的FF状态，没有变化。

以上我们通过研究机器码，达到修改程序运行情况的目的，这也是一些“红客”和“黑客”经常使用的手法。在单片机应用领域中，也存在程序被读出、利用和修改等问题，单片机程序的加密也是许多厂家研究的课题。

3.1.5 扩展练习

（1）请按上节方法，把TWOBALL.HEX的P1口输出改为P2口输出，并观察实验结果。

提示：P1口地址是90H，P2口地址是A0H，修改之后，第2行校验和改为44，第3行校验和改为36。

（2）请在记事本中，输入如下字符（字母一定要大写）。

```
:04000000B29080FC3E
:00000001FF
```

存盘生成扩展名为.hex文件，再加载在Proteus的单片机中，仿真运行并观察结果。

3.2 Keil编程实验

在3.1节里我们了解了由机器码构成的程序，这种程序能直接下载到机器里运行，但编写起来却很困难。为了降低难度，我们用汇编语言来编程。通过这种方式给单片机下命令，需要经过以下几个步骤。

（1）用汇编语言编写源程序。

（2）用汇编程序对源程序进行汇编，翻译成机器码。

（3）把翻译成机器码的文件下载到单片机芯片里。

前两个步骤可以借助工具软件来完成，这方面的软件很多，常用的是Keil软件。第3个步骤，由单片机厂家提供的下载软件完成，这种软件可以从单片机制造商官网获得。需由我们自己完成的事就是要解决计算机与单片机的数据传递问题。

下面我们用Keil来学习编程。

3.2.1 建立Keil工程项目文件

Keil μVision4（以下简称Keil）是集编辑、汇编或编译、仿真等于一体的集成开发软件，它支持众多不同公司生产的MCS51架构的芯片，支持PLM、汇编和C语言的程序设计，不论是学习还是开发51系列的单片机，它都是一个很好的工具。从μVision2到μVision3到现在的μVision4，产品不断升级，但基本功能都一样，对于初学者来说，选择其中任一版本都能满足需要。

学习单片机编程，从汇编开始，可以很好地熟悉了解硬件资源及工作原理，为更有效地使用 C 语言进行编程打下基础。本书用 Keil 来汇编、调试汇编语言程序。

Keil 软件把编程看成一个工程项目，以满足我们在软件开发过程中不断添加文件的需要。使用时应创建工程项目文件，并把源程序文件加入工程项目中才能被汇编。

安装 Keil 软件并运行 Uv4.exe 文件，出现界面如图 3-2 所示。

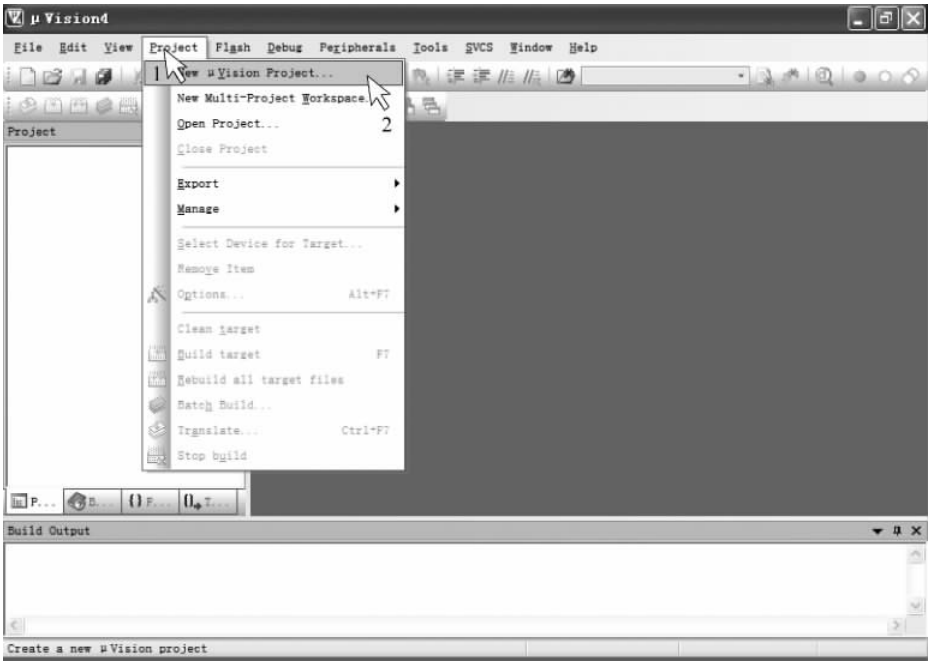

图 3-2 Keil 建立工程项目文件的界面

单击 Project 菜单，选择弹出下拉菜单中的 New μVision Project，此时屏幕中央弹出了一个 Create New Project（创建新项目）的对话框，如图 3-3 所示。

在“文件名”中输入项目名称，这里我们输入的是“e1”，单击“保存”后，程序为文件添加扩展名.uvproj，这是 Keil μVision4 的项目文件扩展名（Keil μVision3 的是.uv2），生成一个 e1.uvproj 文件，它被保存在对话窗口上面列出的文件夹下。同时，弹出一个选择单片机芯片的对话框，如图 3-4 所示。

在 Data base 选项区域中，双击 Atmel 或单击其前面的“＋”号，展开 Atmel 公司生产的芯片目录，再移动滚动条，选择常用的 AT89C51 或 AT89S51（STC89C51 属于这个系列），对话框右边有芯片的简单介绍。单击 OK 按钮后，弹出如图 3-5 所示的对话框。

单击“否”按钮，左边 Project Workspace（项目栏）出现了 Target1（目标 1）文件夹，项目文件正式建立起来，但此时 Source Group 1（源文件群 1）文件夹还是空的，因为还没有程序文件被加入。

图 3-3 建立项目文件对话窗口

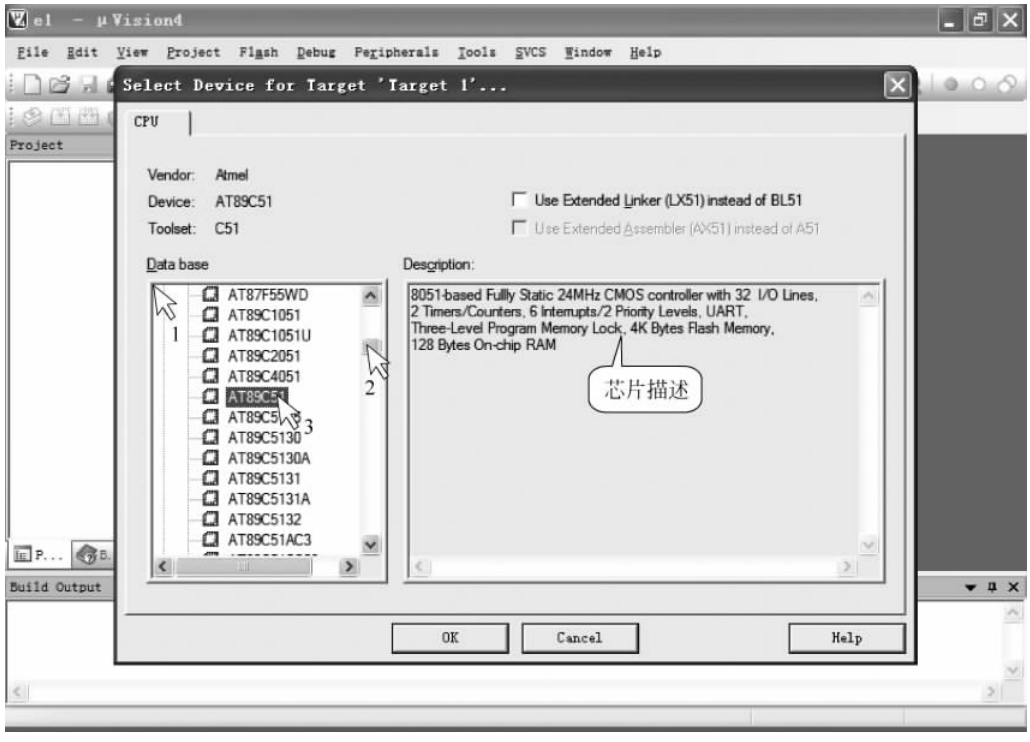

图 3-4 单片机芯片选择

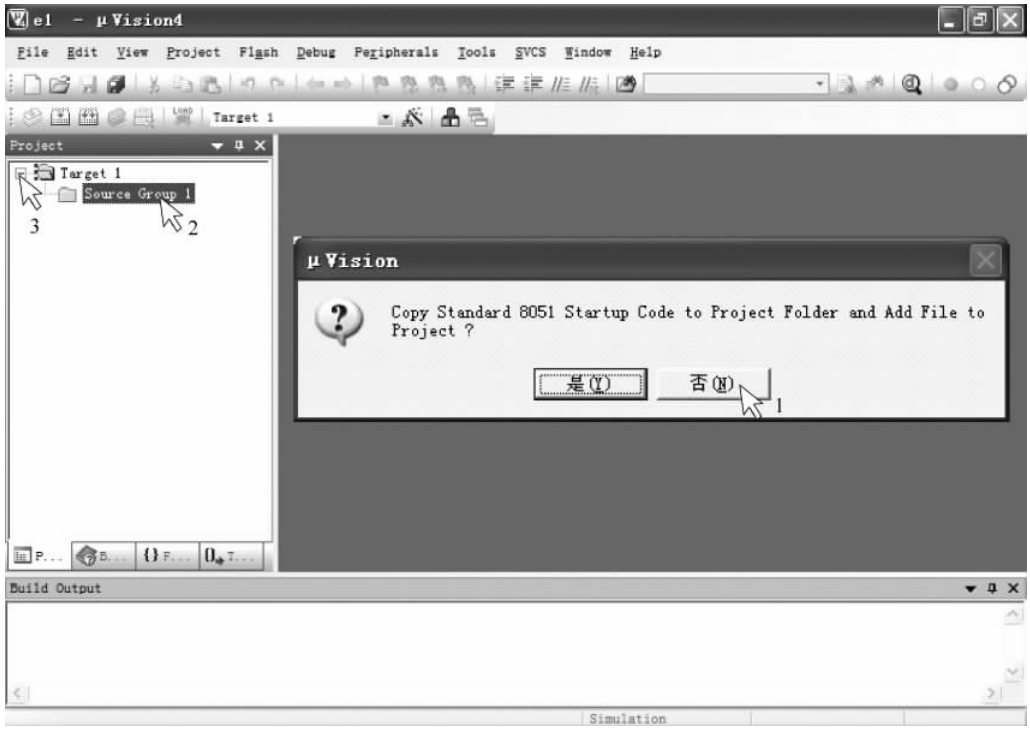

图 3-5 选择芯片后弹出的对话框

3.2.2 新建源程序文件并把它加入项目

1. 新建源程序文件

图 3-5 中，中间的灰色大窗口，是建立程序文件(源文件)的位置。

单击图 3-6 中的“新建文件”按钮，新建源程序文件。也可以通过菜单 File→New 或快捷键 Ctrl+N 来实现，在原来的灰色窗口中出现了一个新的文字编辑窗口，系统自动命名为 Text1，它还不是我们自己的文件。单击“保存”按钮，在弹出的对话框里选择要保存源程序的文件夹，并为源文件命名(扩展名一定要为.asm)。单击保存后，原来的 Text1 文件窗口就变成我们自己命名的文件编辑窗口，如 e1-1.asm 文件窗口，可在里面编写源程序。此时，新建的源文件还没有加入项目文件中的 Source Group 1 中。

2. 把源文件加入项目

要对源程序进行汇编，应先把它加入项目。步骤如图 3-7 所示。

在屏幕左边的 Source Group 1 文件夹图标上右击(注意是右击)弹出菜单，这里可以进行在项目中添加或移除文件等操作。单击 Add File to Group‘Source Group 1’，弹出文件窗口。Keil 默认的文件类型是 C 语言文件，这里需单击文件类型下拉列表，选择 Asm Source file(汇编源文件) 文件类型。如果上面显示的文件夹里有.asm 文件，会自动显示出

图 3-6 建立源程序文件

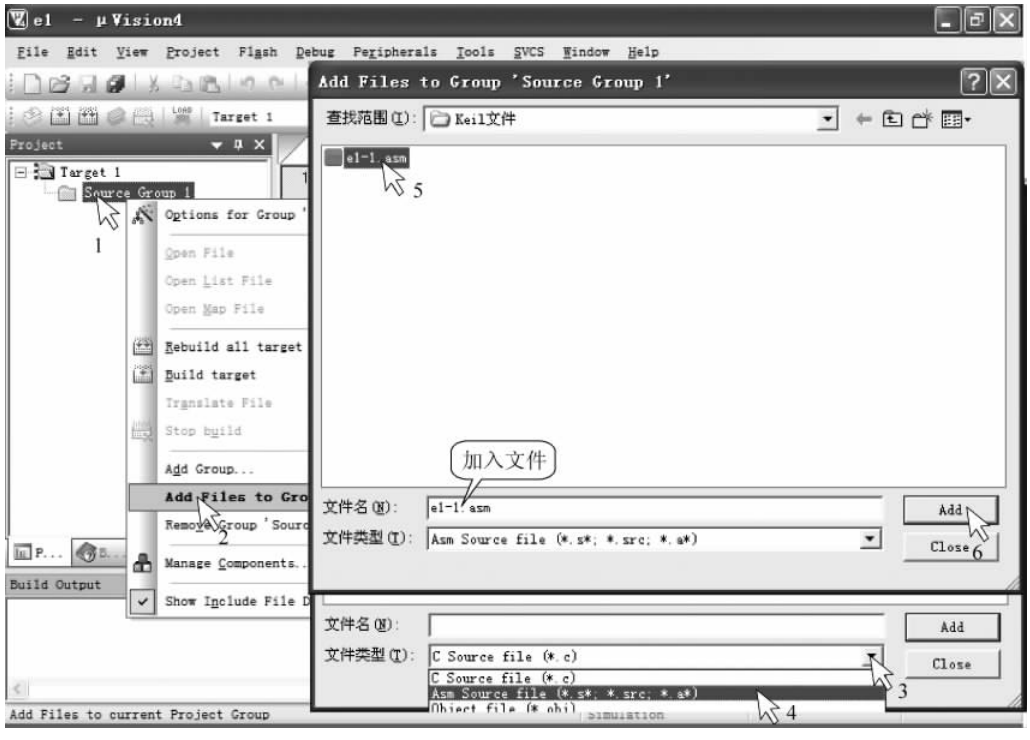

图 3-7 把源程序文件加入项目

来，单击它，就被加在文件名的框里，单击 Add 按钮，关闭文件窗，它就被加到项目中。这时在 Source Group 1 文件夹图标左边出现了一个小“＋”号，说明文件群组中有文件，单击它就可以展开查看。

现在我们可以编辑、调试源程序了。

3.2.3 P 口输出编程实验

1. 任务和方法

我们的任务是把 P1 口的 P1.0 置为低电平，有多种实现方法。

```
mov p1,#11111110b          ;置 P1.0 为 0
mov p1,#0feh               ;置 P1.0 为 0
clr p1.0                   ;置 P1.0 为 0
```

前两句直接传一个数给 P1 口锁存器，使它的第 0 位为 0，首部 # 号标记这个数是一个纯数（立即数），尾部 b 表示它是一个二进制数；h 表示十六进制数，fe 前加个 0，表示它后面的字母 f 是数值。第 3 条指令则是直接对 P1.0 置 0。

2. 编程

在源文件编辑窗口中输入程序，如图 3-8 所示。注意，指令“,”及注释符“;”都必须在纯英文状态下书写。org（开始）和 end（结束）是伪指令，是告诉翻译者（汇编软件）的，并不会被翻译成机器码。

图 3-8 编程和汇编

如果已对源文件进行了保存，在输入指令时，助记符的正确或错误，立刻就可在屏幕上反映出来，正确的助记符会自动地用加黑的方式显示，如图 3-8 所示。除此之外，还有其他色彩显示方式，用于识别指令的各组成部分。

对中文注释，前面一定要打英文的“;”，否则，一些注释不显示，在排错时很难找出此类错误。

图 3-8 所示程序的功能是使 P1.0 产生 0 和 1 的变化，以观察前面提到的三条指令使 P1.0 置 0 的作用。

3. 汇编

程序编完后，单击 按钮进行汇编。如果下面的 Build Output 信息框里出现 0 Error (零错误)，表示汇编成功。接着可单击 按钮进行仿真调试，如图 3-9 所示。

4. 单步调试

图 3-9 是由单击 按钮打开的调试窗口。初次打开时，窗口下方会出现一些默认打开的小窗口，可单击各窗口右上角的“×”，暂时关闭它们。

单击 按钮进行单步调试，但是，现在看不到 P1 口的变化，还需要打开 P1 窗口。

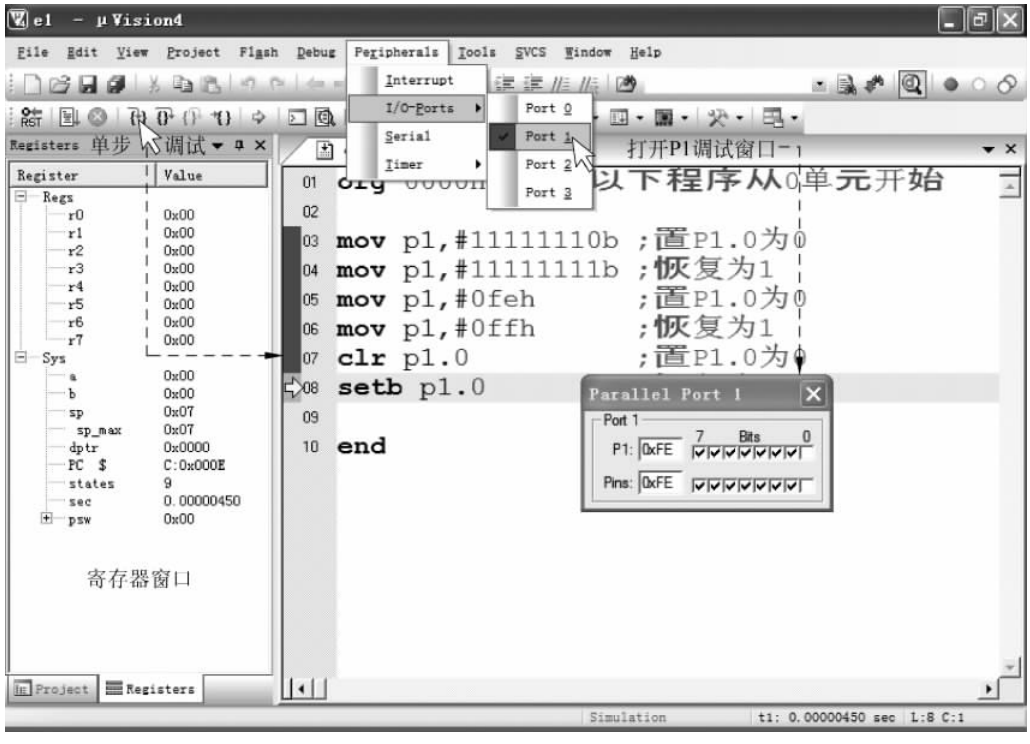

图 3-9 调试窗口

单击菜单 Peripherals(外围部件)→I/O- Ports(输入输出口)→Port 1，就打开了 P1 口的调试窗口 Parallel Port 1，如图 3-9 所示。