第一课,了解单片机及单片机的控制原理和 DX516 的用法，控制一个 LED 灯的亮和灭

本章学习内容：

单片机基本原理，如何使用 DX516 仿真器，如何编程点亮和灭掉一个 LED 灯，如何进入 KEILC51uV 调试环境，如何使用单步，断点，全速，停止的调试方法

聂小猛 2006 年 6 月

单片机现在是越来越普及了，学习单片机的热潮也一阵阵赶来，许多人因为工作需要或者个人兴趣需要学习单片机。可以说，掌握了单片机开发，就多了一个饭碗。

51 单片机已经有 30 多年的历史了，在中国，高校的单片机课程大多数都是 51，而 51 经过这么多年的发展，也增长了许多的系列，功能上有了许多改进，也扩展出了不少分支。而国内书店的单片机专架上，也大多数都是 51 系列。可以预见，51 单片机在市场上只会越来越多，功能只会越来越丰富，在可以预见的数十年内是不可能会消失的。

作为一个初学者，如何单片机入门？需要那些知识和设备呢？知识上，其实不需要多少东西，会简单的 C 语言，知道 51 单片机的基本结构就可以了。一般的大学毕业生都可以快速入门，自学过这 2 门课程的高中生也够条件。

就算你没有学过单片机课程，只掌握了 C 语言的皮毛，通过本系列的教程，您也会逐渐的进入单片机的大门。当然在学习的过程中，您还是必须多去研读单片机书籍，了解他们的基本结构及工作方式。

下面以 51 为例来了解一下单片机是什么东西，控制原理又是什么？

在数字电路中，电压信号只有两种情况，高电平和低电平，用数字来记录就是 1 和 0。单片机内部的 CPU，寄存器，总线等等结构都是通过 1 和 0 两种信号来运作的，数据也是以 1 或者 0 来保存的。单片机的输入输出管脚，也就是 IO 口，也是只输出或识别 1 和 0 两种信号，也就是高电平和低电平。当单片机输出一个或一组电平信号到 IO 口后，外部的设备就可以读到这些信号，并进行相应操作，这就是单片机对外部的控制。当外部一个或一组电平信号送到单片机的 IO 口时，单片机也可以读到这些信号，并进行分析操作，这就是单片机对外部设备信号的读取。当然实际的操作中，这些信号可能十分复杂，必须严格地按照规定的时间顺序（时序）输入输出。每种设备也都规定了自己的时序，只要都严格遵守，就可以控制任何设备，做出只要你想象得出的任何事情。

您可能会再问，我如何让单片机去控制和分析外部设备呢？答案是程序，您可以编写相关的程序，并且把他们烧写到单片机内部的程序空间，单片机在上电时，就会一步一步按照您写的程序去执行指令，做您想做的事情。

在 51 标准芯片中，有 32 个输入输出 IO，分为 4 组，每组 8 个，分别为 P0 口，P1 口，P2 口，P3 口。P1 口的 8 条脚就用 P1.0 至 P1.7 表示，其余类似。51 就是用这 32 个口来完成所有外部操作的。对于 51 的内部结构，如果您已经了解，那是最好；如果不懂，也可以先放下，在完成了本教程开始的几个章节之后，您就会大有兴趣，自己去寻找资料阅读了。当然，如果您希望成为一个优秀的单片机开发程序员，还是必须熟悉单片机的内部结构及工作原理，切不可偷懒！

在这一章，您将用程序去控制一个 LED 发光管的亮和灭。你应该知道，LED 发光管在通过一定电流时亮，不通电就灭。为了不让 LED 通过太大的电流把它烧坏，我们还要串上限流电阻。51 的 IO 是弱上拉的方式，在输出高电平时，只能输出几十微安的电流到地，而在输出低电平时，VCC 电源可以输入几十毫安的电流到 IO。一般 LED 需要 10 毫安左右电流点亮，我们就将 LED 接在电源 VCC 和 IO 口之间，中间串上电阻，当 IO 输出低电平时，灯就亮了，反之，灯就灭了。我们在这个程序里要控制的是 P1.0。请参考一下我们将要使用的试验板的电路图，这个试验板是在购买 dx516 仿真器是赠送的。

图 1，试验电路图

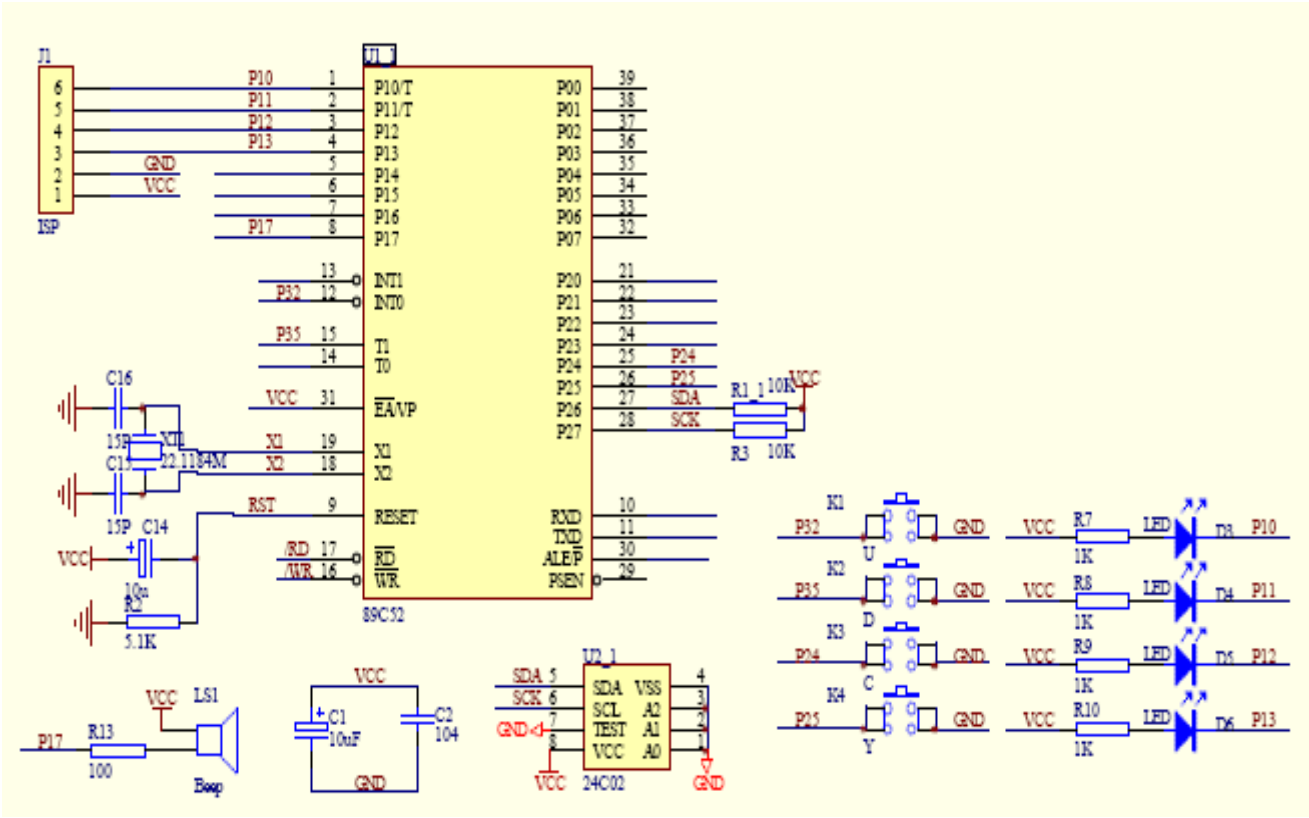
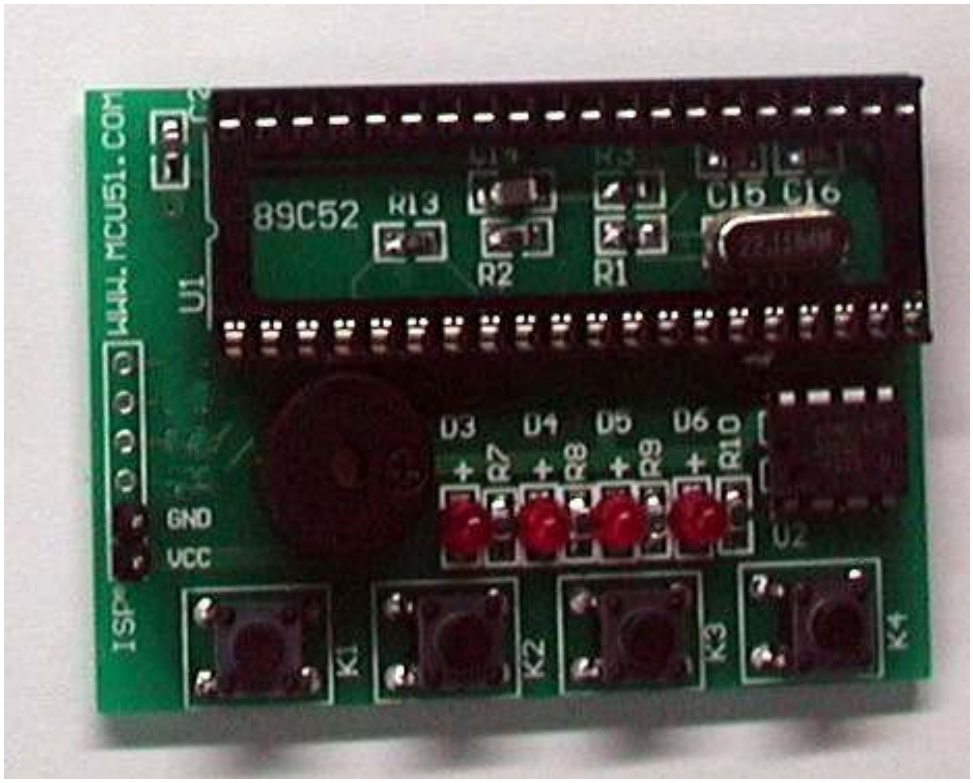


图 2：试验板外观图



下面介绍一下仿真器和仿真环境。

在实际的单片机学习和开发中，你可以用仿真器模拟一个 CPU 芯片，让它按照您编写的程序工作，并且进行调试，一步步排除程序的 bug，使程序正常工作。程序工作正常后，您就可以用烧写器将您编写的程序烧入购买来的单片机芯片中，让它自己去运行了。

要使用仿真器，还得有一个编译调试的环境，这个环境是在计算机上运行的，我们就在计算机上编写和调试程序，计算机和仿真器有连接，仿真器中的各种数据和程序，都可以从计算机上观察到，并可以观察变量，写入变量的值，单步调试程序，在程序中设置断点调试，全速运行，停止程序运行，等等操作。

我们使用世界上目前最先进的 keilC51 编译调试环境，仿真器使用大虾电子网（ http://www.daxia.com ）设计的 DX516 专业版仿真器，这个仿真器功能齐全，性价比最佳，是学习开发的好工具！

您可以在此页 http://www.daxia.com/product/dx58/ 的资料下载栏目里下载到 keilc51 相关的中文说明资料，这些资料详细地说明了如何使用 C51 编程和如何使用 keil uV2 环境调试，请在本章试验完成或者试验过程中，如果遇到不懂的地方，一定要抽时间阅读！

您应该也可以在上面的网页中找到下载破解版本的 keilc51 的办法。中国法律规定，在学习和研究工作中使用有版权的软件是可以的，但是，如果您开发产品时，建议您还是去购买一个正版的软件。

下面是 DX516 仿真器的使用介绍：

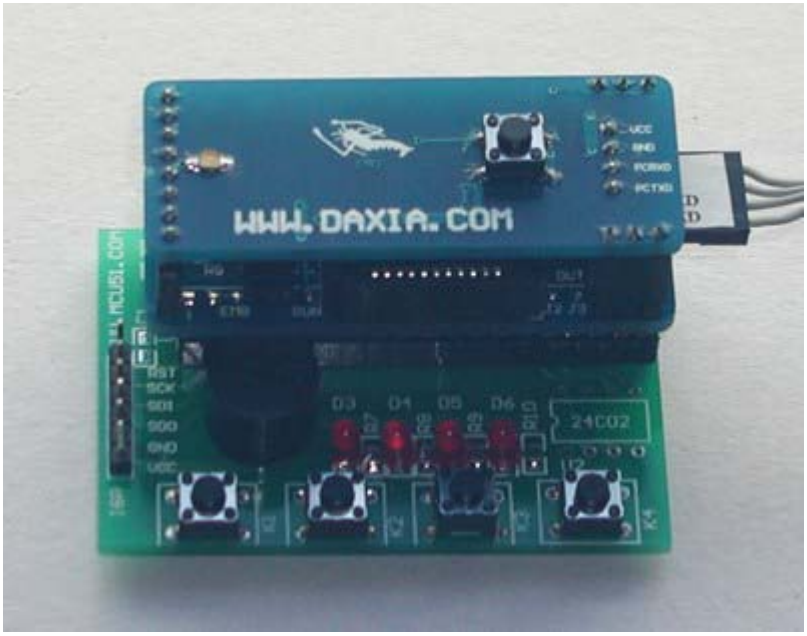
1。安装

将仿真器和试验板按图 3 组装好，串口线按照正确方向插入仿真器，另一端和电脑串口连接，请尽量使用计算机的硬串口。

仿真器底座左边的跳线，请放在 EMB 这边，以进入仿真状态。如果放在 RUN 这边，将会进入脱机运行状态。

晶振选择跳线请放在 IN 这边，以使用仿真器内部晶振，内部晶振更加可靠。如果放在 OUT 这边，则会使用外部的用户板晶振。

图 3 仿真器插在试验板上



2。电源

因为用户板使用电流不大，可以使用 usb 取电，usb 最大电流可以提供 500mA，将 usb 取电板插入电脑的 usb 口中。（实际应用中，如果用户板使用电流超过 100mA，我们就建议使用外部电源）

3。启动

在仿真器上电，或者按一下仿真器上面的按钮时，仿真器会发出“嘀－”，表示仿真器正常启动。同时仿真器上面的灯闪烁一次，表示进入正常仿真状态。

4。仿真设置

第一个设置：

C51 用户请在您的代码的 main()函数前面，加上一句：

```
char code dx516[3] _at_ 0x003b;
```

如果以上设置你没有做，在装载过程中，仿真器会发出“嘀嘀嘀” 的三声短声报警，这时的仿真结果将可能不正确。

在我们的例程中，这句话已经加入了。这句话并不会影响程序的工作，可以一直保留。

第二个设置：

请在硬件仿真设置选项中，选择 serial interrupt,在前面打勾。

如果以上设置你没有做，在装载过程中，仿真器会发出“嘀－” 的一声长声报警，这时的仿真结果将可能不正确。

其余设置：

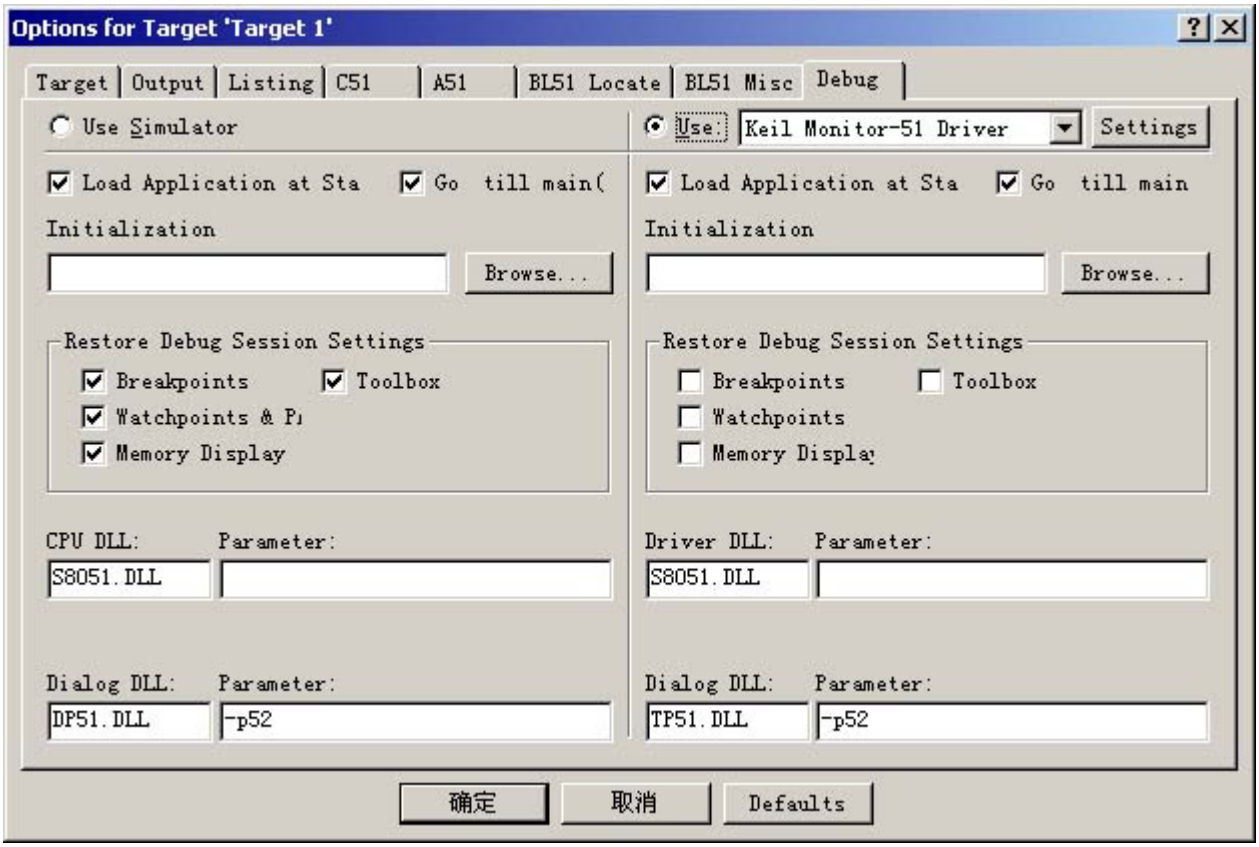
请选择 use keil Monitor-51 Driver ，这样才会使用硬件仿真

请选择 load Application at start ，在启动时直接装载程序

请选择 Go till main ，装载后直接运行到 main 函数

请在硬件仿真设置选项中，选择 115200bps 波特率，所有 cache 都可以不选，或者只选 cache code。同时请选择正确的串口号。

图 4 仿真设置



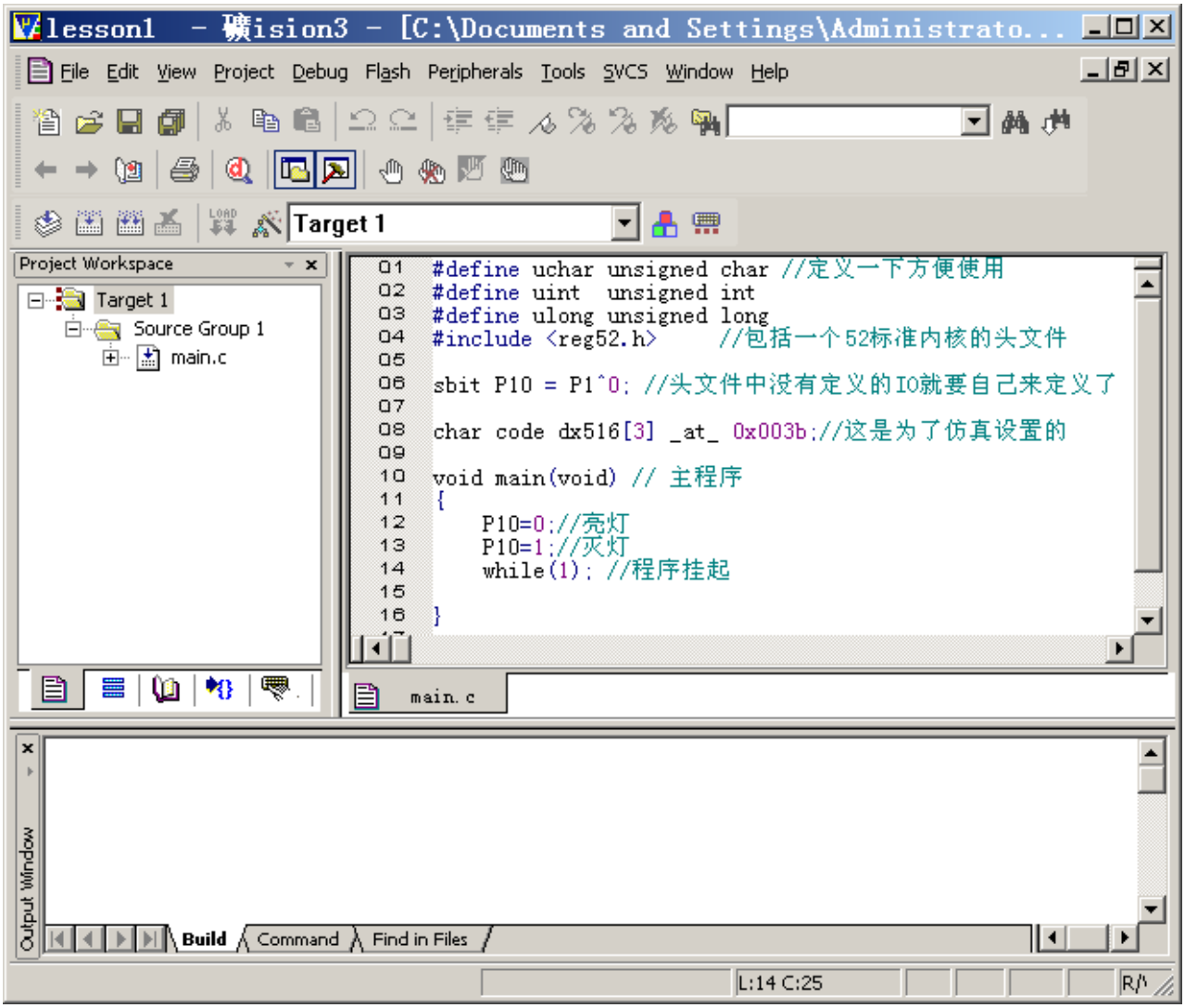
好了，现在可以开始做试验了，我们打开已经建立好的工程和编写好的程序试验。顺便还会学习一下程序调试的技巧。至于如何建立一个新工程，请参考 C51 的帮助文件，或者自己摸索一下，WWW.DAXIA.COM 的 DX516

专栏里也有“一步步教你如何第一次做…”的文章可以学习。

请双击 lessoncode01 目录下的 lesson1.uv2，打开后界面如下：

图 6：程序界面

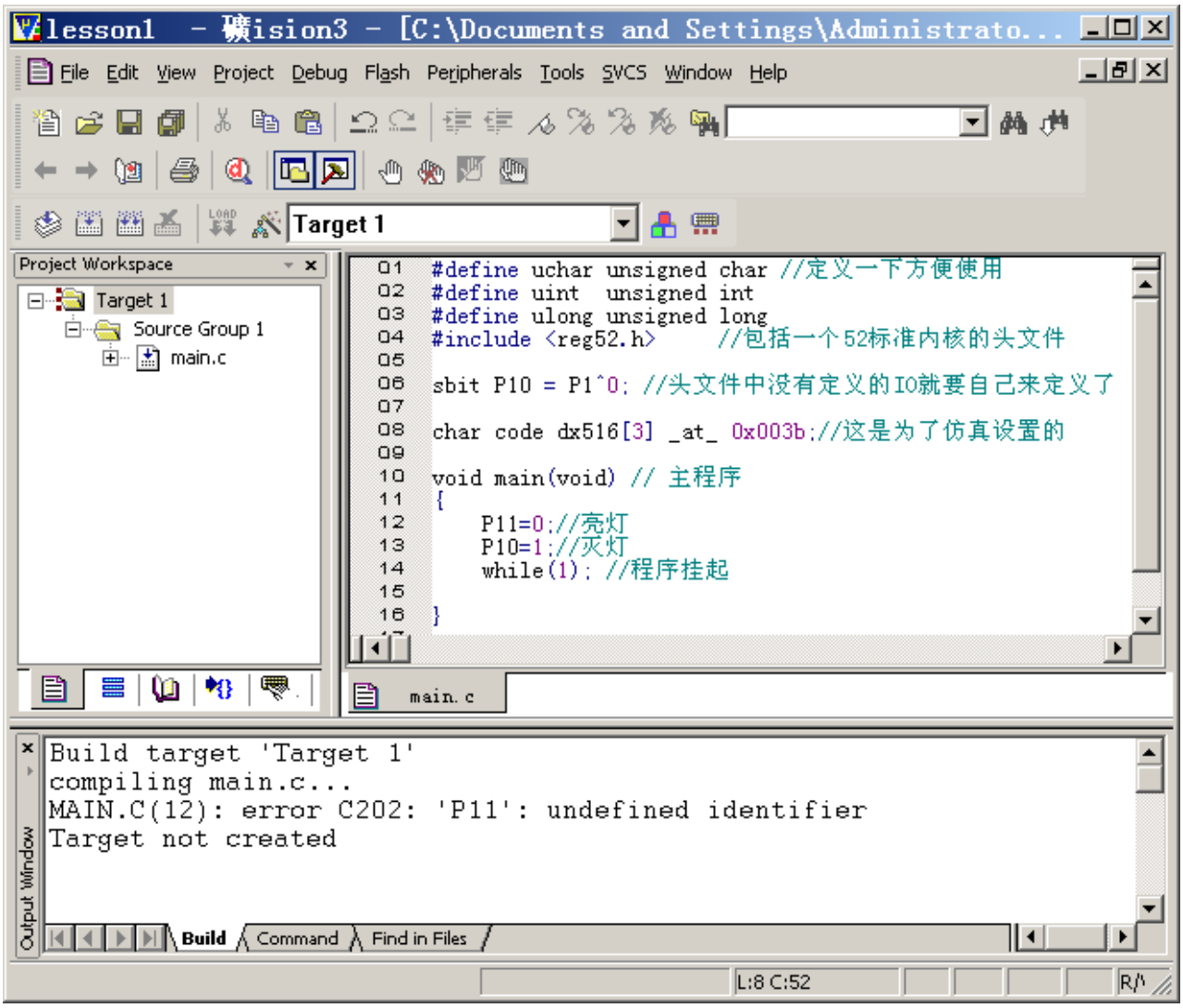
这个界面是 uV3 的，和 uV2 是一样用的。



点一下上图第三排第 2 或者第 3 个按钮（您的编译器按钮位置不一定在那个位置，自己找找），就可以看到编译结果了。上面显示是 0errrs,0warnings，这是最佳的编译结果，如果有 error，则无法进行下一步仿真，如果有 warning，一定要尽量消除，确实无法消除的，也要确认不会对程序造成影响，才进行下一步的仿真。

在编译结果中，我们还可以看到有 data，xdata，code 等用了多少字节的报告，要注意您的单片机中是否有这么多的资源，如果不够，将来烧片运行时就可能出现问题。比如 AT89C51 的程序空间是 4K，xdata 如果没有外扩就是 0 个，data 是 128 个。超出这些范围，程序就不能在 AT89c51 中运行。不同的芯片有不同的容量，如 SST89E516RD 就有 64K 程序，内部 768 字节 XDATA，还有 256 个字节的 data。我们的例程中肯定都考虑了这些了，肯定不会超出，因为 DX516 仿真器是和 SST89E516RD 有同样的容量的，将来自己开发时就要注意了。

下面我们故意把第 9 行的 P10 写成 P11，点编译，因为没有预先定义 P11，所以就报告错误了，如下图：

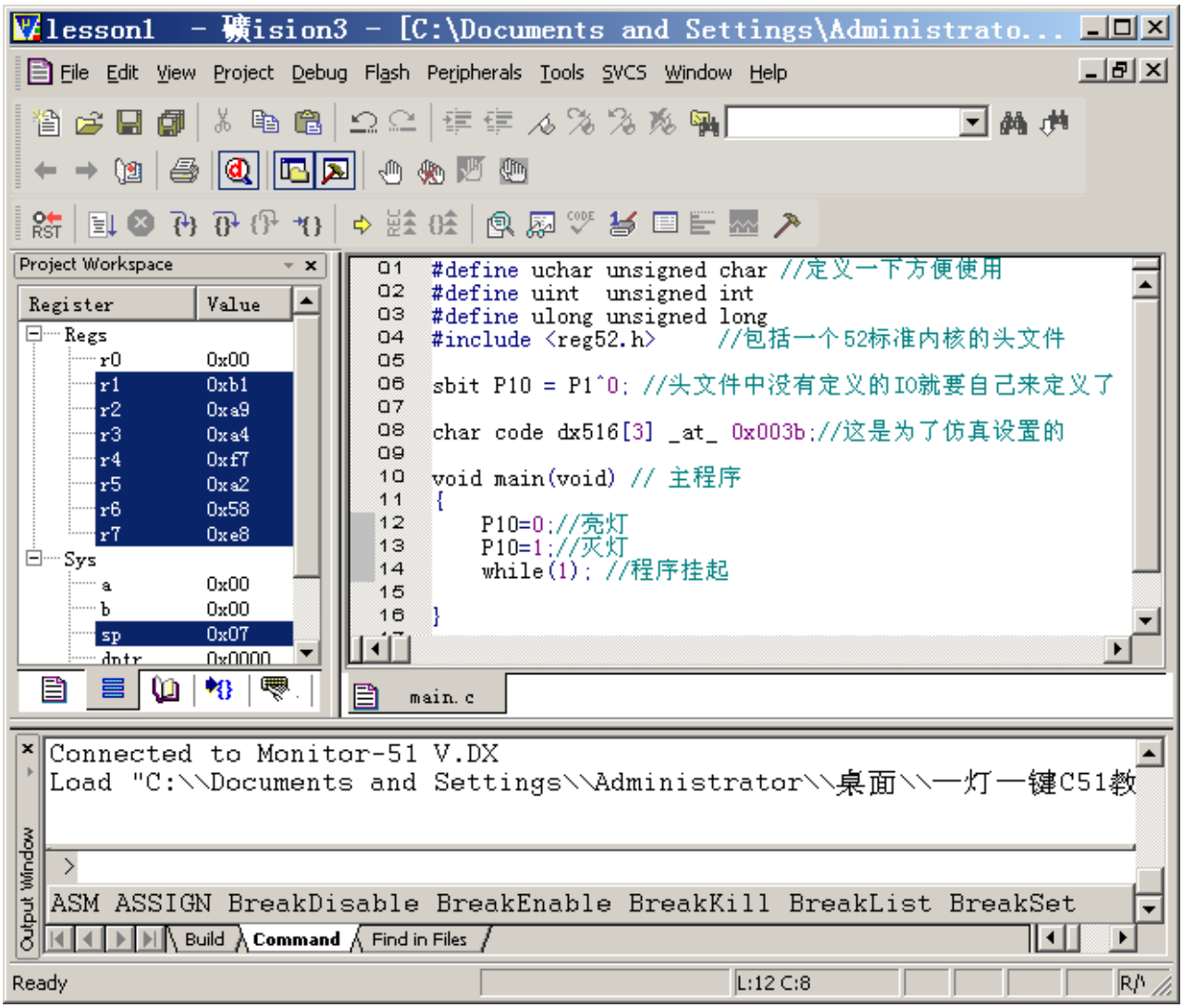


双击一下错误报告的那一行，窗口就也会跳到这一行，方便您进行修改。好了，现在请把错误改回去，再编译一次，出现报告正确了以后，下面开始仿真了。

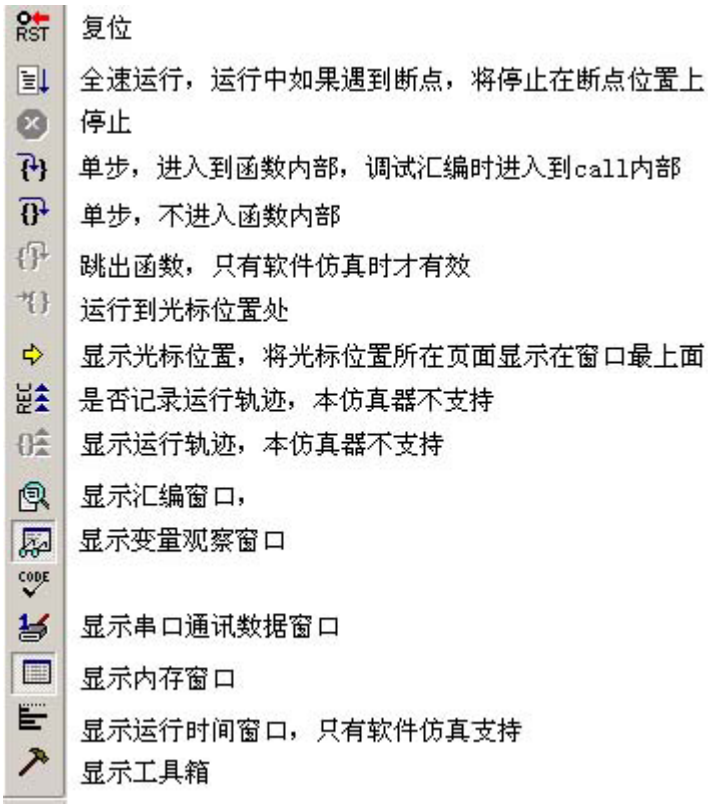
点一下第二行第 5 个一个放大镜里面一个 d 字母的按钮，就可以进入仿真了，仿真器要事先连接好哟。进入仿真后要退出仿真环境也是点这个按钮。注意，等会如果程序在正在全速运行时，仿真环境是不能直接退出的，得先点停止运行后，再点仿真按钮才可以退出。

点进入仿真按钮，程序开始装载，PC 自动运行到了 main（）停下，并指向了 main（）函数的第一行。

下面是进入了仿真环境的截图：



再顺便把调试界面上的按钮介绍一下：
图 5：按钮说明



进入仿真窗口后，如果出现的不是前面的源代码窗口，而是夹有反汇编代码的窗口，直接关掉这个窗口就会恢复到代码窗口。下次进入也会直接进入到源代码窗口。

现在先试验单步，点单步（两个单步都可以，一般点单步跨过）。可以看到灯亮了。PC 指针也指向了下一个程序行。

图：照片，灯亮

再点一下单步，PC 又走下一步，灯灭了。

再点一次，PC 走到挂起的程序行了，继续点仍然在这一行。这句指令其实就是使程序不断地跳到自己这一行，别的什么也不做。一般称作程序挂起。

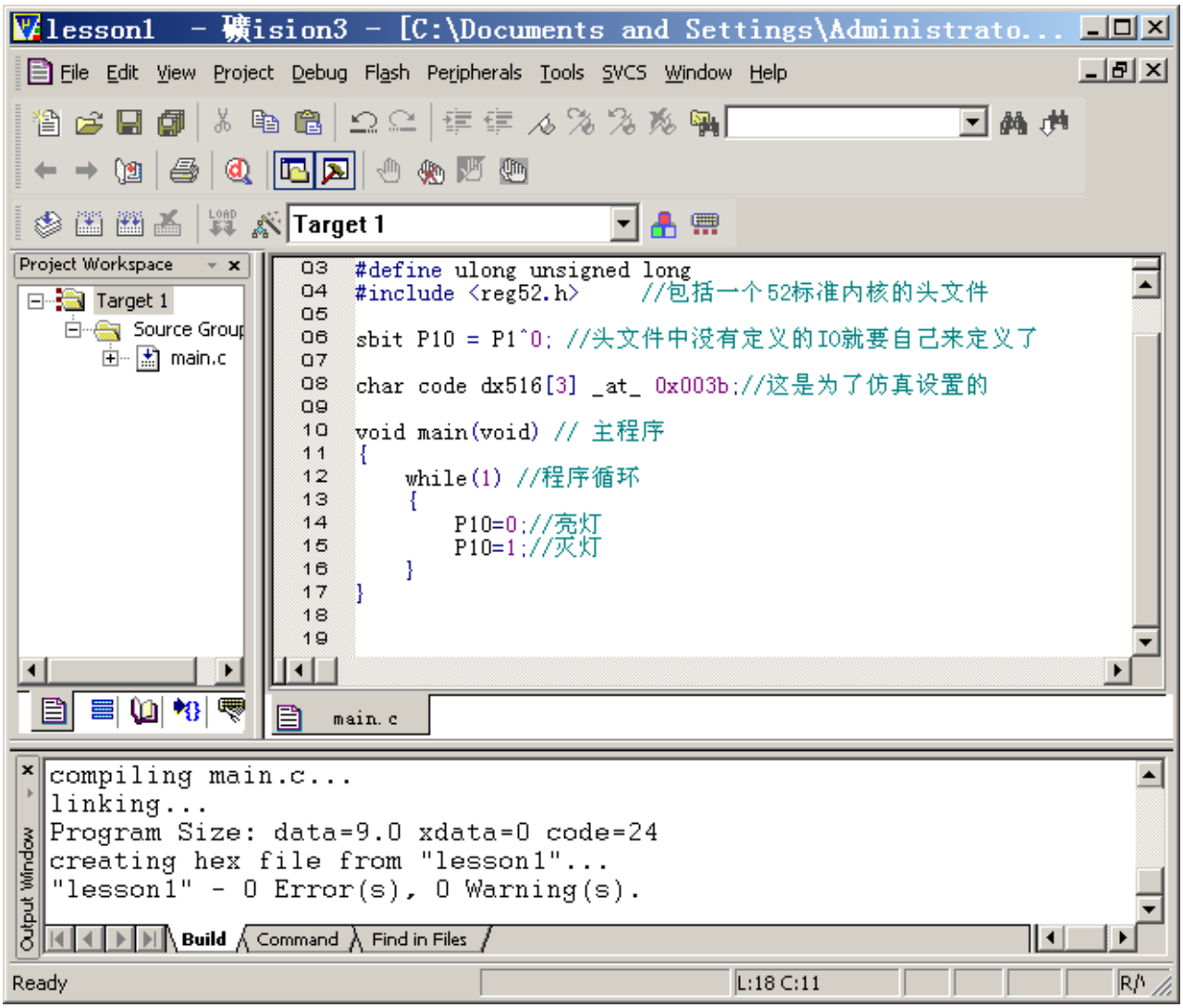
一般的实际应用中的程序是不会挂起的，一般是在 main 函数里做一个大循环，程序如下：

```
void main(void)//  主程序
{
    while(1)
    {
        P11=0;//亮灯
        P10=1;//灭灯
     }
}
```

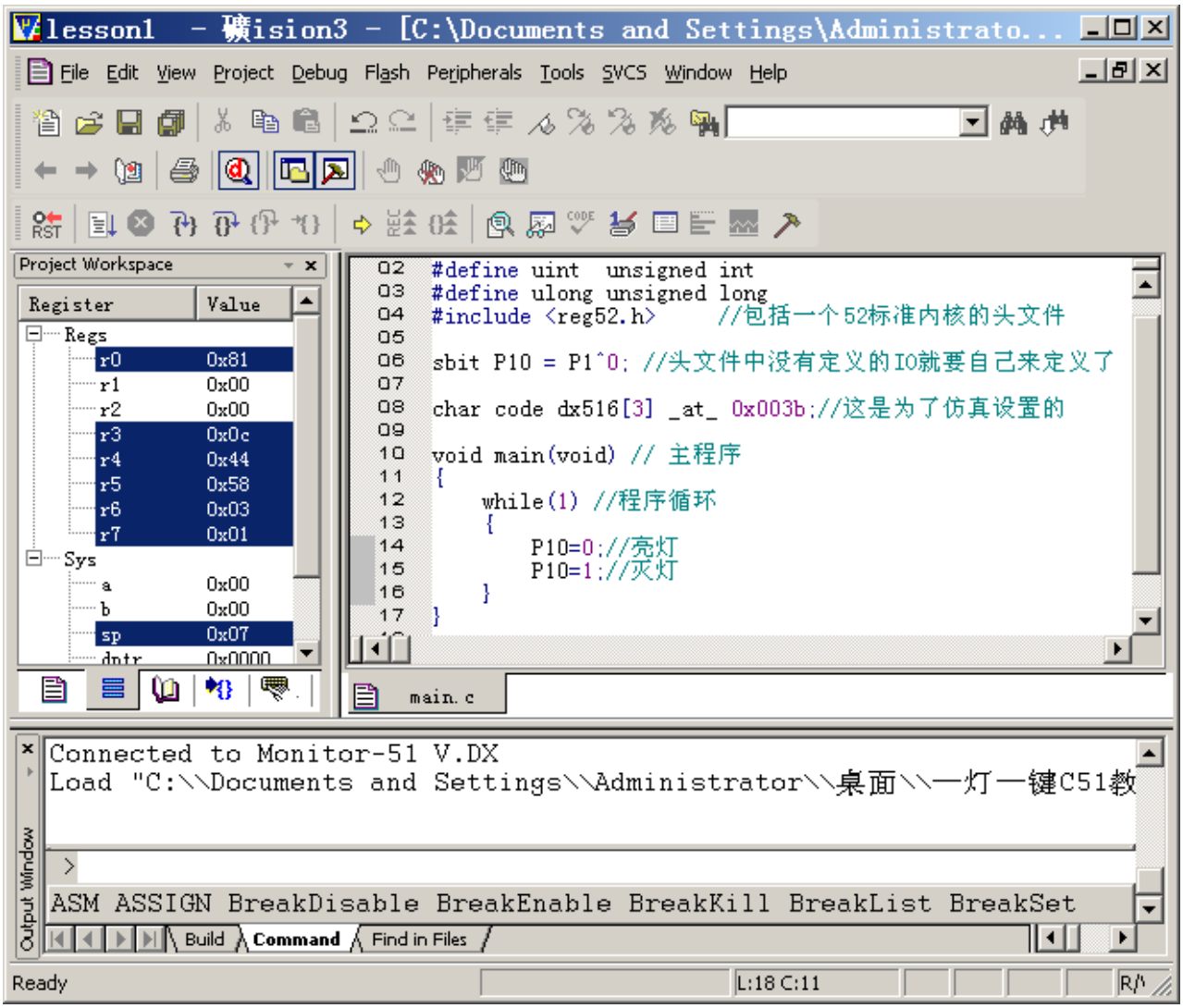
请将 main 函数程序改为上面的代码，我们下一步将试验断点的操作。

编译后结果如下：

图：


lesson1 - 礦ision3 - [C:\Documents and Settings\Administrato...
File Edit View Project Debug Flash Peripherals Tools SVCS Window Help
Target 1
Project Workspace
Target 1
Source Group
main.c
#define ulong unsigned long
#include <reg52.h> //包括一个52标准内核的头文件
sbit P10 = P1^0; //头文件中没有定义的IO就要自己来定义了
char code dx516[3] _at_ 0x003b;//这是为了仿真设置的
void main(void) // 主程序
{
while(1) //程序循环
{
P10=0;//亮灯
P10=1;//灭灯
}
}
main.c
compiling main.c...
linking...
Program Size: data=9.0 xdata=0 code=24
creating hex file from "lesson1"...
"lesson1" - 0 Error(s), 0 Warning(s).
Output Window
Build
Command
Find in Files
Ready
L:18 C:11

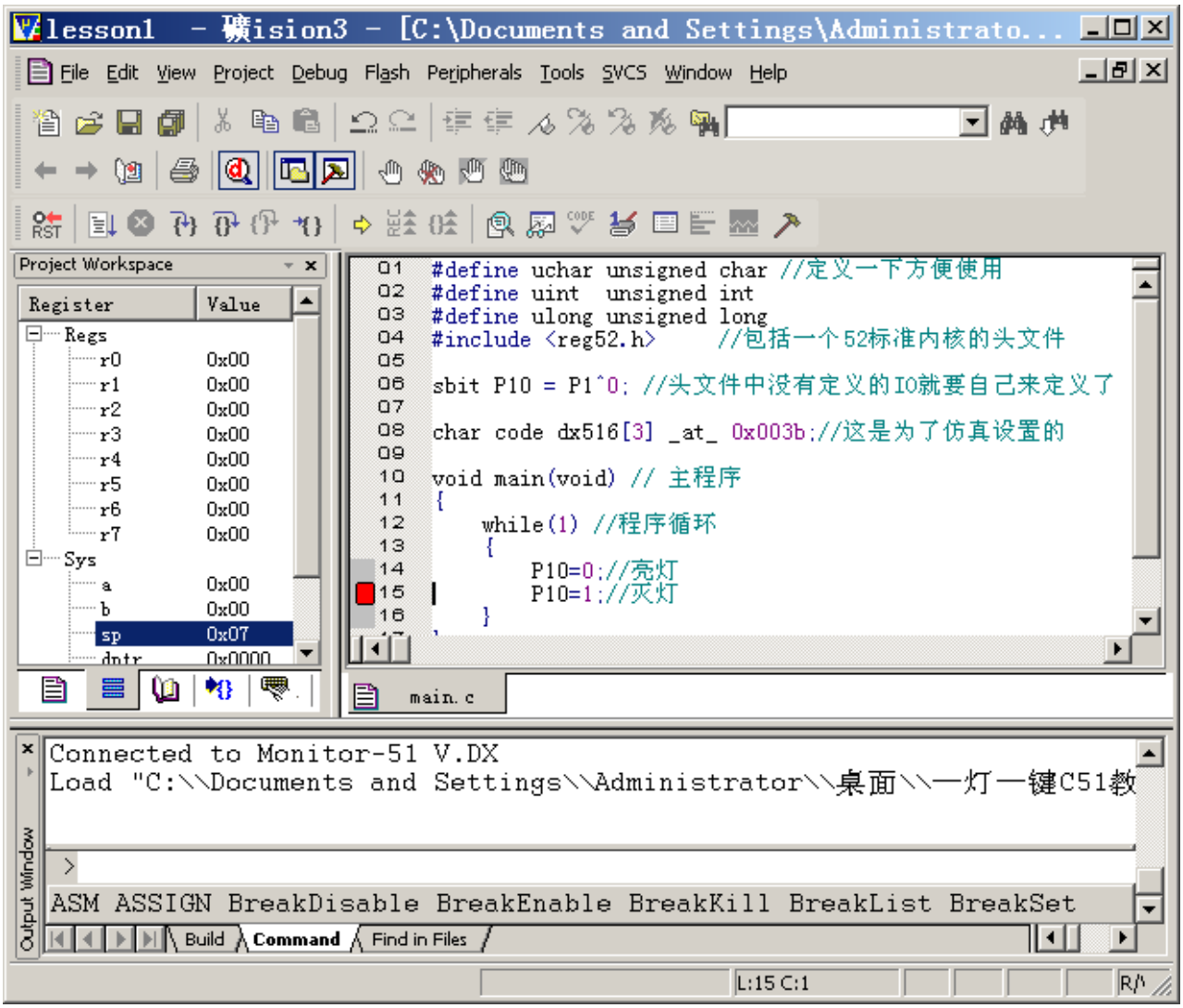
进入仿真后
图：



可以看到下面的提示窗口中显示：“ connected to Monitor-51 V.DX ”,后面的 V.DX 就是已经连接到大虾仿真器的提示了。V.DX 是大虾仿真器特有的标识。

在第 15 行双击一下，可以看到程序行左边出现了一个红方块，这就是设置断点，再双击一次，断点就取消了。如果程序在全速运行的过程中遇到断点，就会自动停下来给你分析。注意在进入仿真后，并且程序是停止状态时，才可以设置或者取消断点。

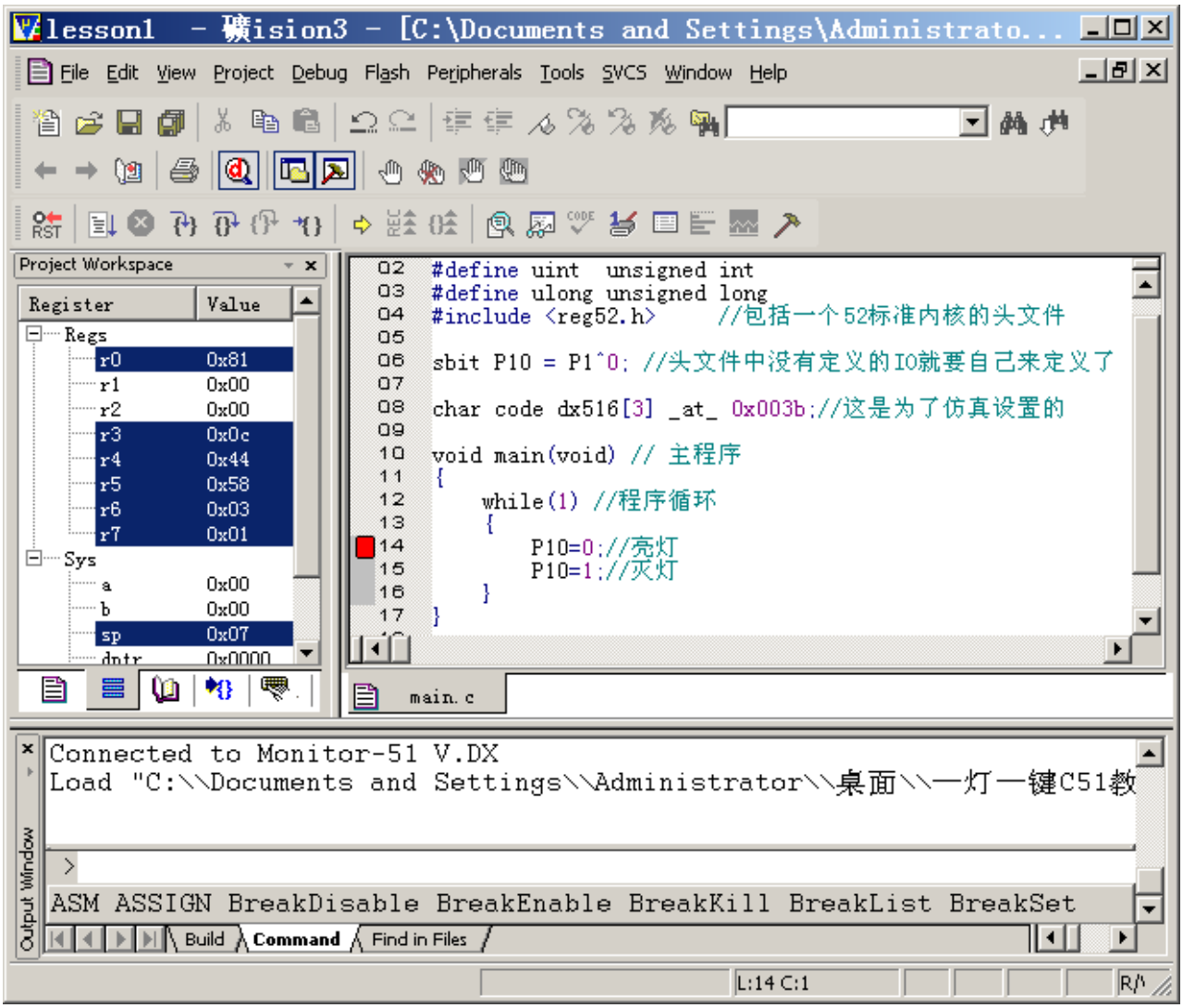
图：设置了断点



现在点全速运行，可以看到程序在断点处停了下来，并且由于前一句指令刚刚执行了点灯，所以这时灯是亮着的。

现在在第 14 行设置断点，并且取消上一个断点。

图：设置了另一个断点



现在点全速运行，可以看到程序在断点处停了下来，并且由于刚刚执行了灭灯，灯是灭着的。

好，现在试验全速运行和停止。
把断点取消，再点全速运行，可以看到灯是亮着的，但是不是很亮，这是由于程序是循环的，亮灭交替进行，亮的时间并不是全部的时间。
现在点停止，可以看到程序停止了，重复几次进行全速和停止，可以发现每次停止的地方不一定是同一位置。

这一课就先结束了，我们学习了如何点灯及一些基本的编译和调试操作，下一课将学习如何使 LED 闪烁，和更多的调试方法，和如何查看运行状态和设置内部寄存器的值。

课后作业：
改为第 2 个 LED 灯（P1.1）做完本章的试验。