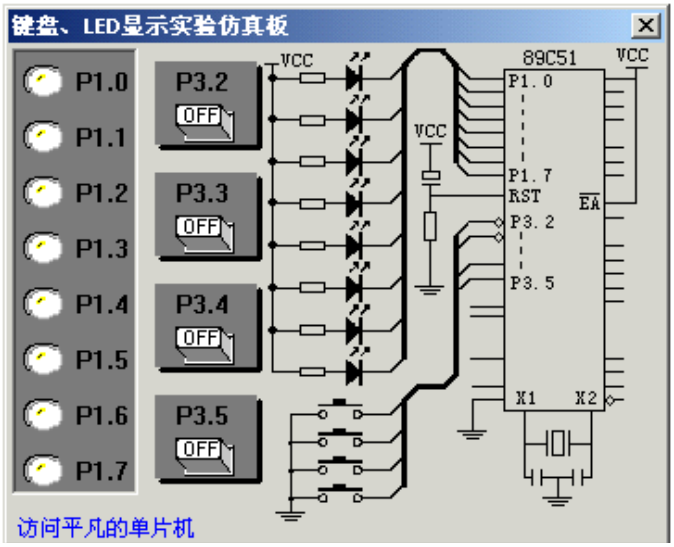
其于 Keil 的实验仿真板的使用

前面介绍了 Keil 软件的使用，从中我们可以看到 Keil 的强大功能，不过，对于初学者来说，还有些不直观，调试过程中看到的是一些数值，并没有看到这些数值所引起的外围电路的变化，例如数码管点亮、发光管发光等。为了让初学者更好地入门，笔者利用 Keil 提供的 AGSI 接口开发了两块仿真实验板。

这两块仿真板将枯燥无味的数字用形象的图形表达出来，可以使初学者在没有硬件时就能感受到真实的学习环境，降低单片机的入门门槛。图 1 是键盘、LED 显示实验仿真板的图，从图中可以看出，该板比较简单，有在 P1 口接有 8 个发光二极管，在 P3 口接有 4 个按钮，图的右边给出了原理图。



键盘、LED 显示实验仿真板

图 2 是另一个较为复杂的实验仿真板。在该板上有 8 个数码管，16 个按键（接成 4＊4 的矩阵式），另外还有 P1 口接的 8 个发光管，两个外部中断按钮，一个带有计数器的脉冲发生器等资源，显然，这块板可以完成更多的实验。

一、实验仿真板的安装

这两块仿真实验板实际上是两个 dll 文件，名称分别是 ledkey.dll 和 simboard.dll，安装时只要根据需要将这两个或某一个文件拷贝到 keil 软件的 c51\bin 文件夹中即可。

图 2 单片机实验仿真板

二、实验仿真板的使用

要使用仿真板，必须对工程进行设置，设置的方法是点击 Project->Option for Target ‘Target1’打开对话框，然后选中 Debug 标签页，在 Dialog :Parameter:后的编缉框中输入 -d 文件名。例如要用 ledkey.dll（即第一块仿真板）进行调试，就输入-dledkey，如图 3 所示，输入完毕后点击确定退出。编译、连接完成后按 CTRL+F5 进入调试，此时，点击菜单 Peripherals，即会多出一项“ 键盘 LED 仿真板（K）”，选中该项，即会出现如图 1 的界面，

同样，在设置时如果输入-dsimboard 则能够调出如图 2 的界面。

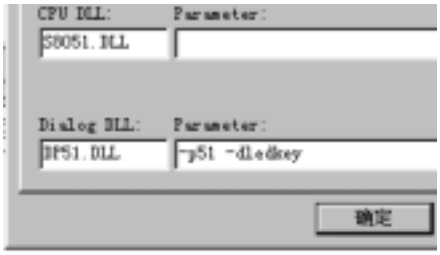


图 3 实验仿真板的设置

第一块仿真板的硬件电路很简单，电路图已在板上，第二块板实现的功能稍复杂，其键盘和数码显示管部份的电路原理图如图 4 所示。下表给出了常用字形码，读者也可以根据图中的接线自行写出其它如 A、B、C、D、E、F 等的字形码。 除了键盘和数码管以外，P1 口同样也接有 8 个发光二极管，连接方式与图 1 相同；脉冲发生器是接入 T0 即 P3.4 引脚。

0c0h	0f9h	0a4h	0b0h	99h	92h	82h	0f8h	80h	90h	0FFH
0	1	2	3	4	5	6	7	8	9	消隐

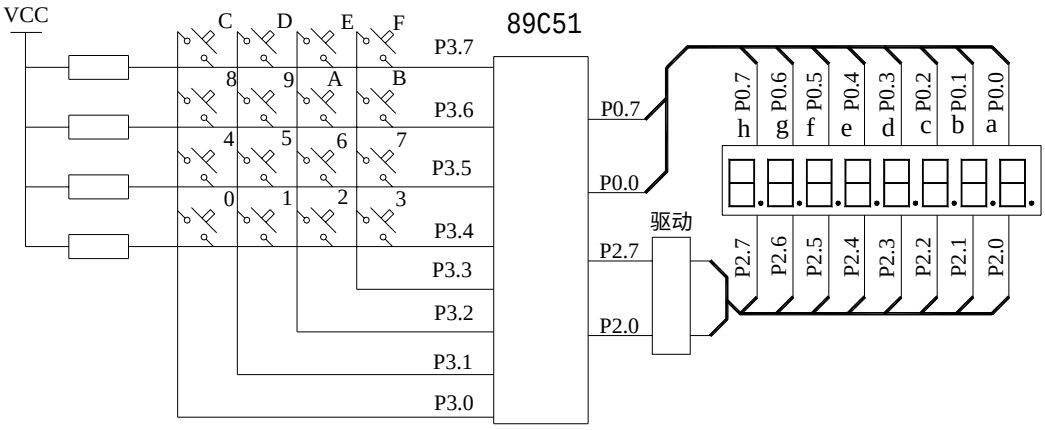


图 4 实验仿真板 2 数码管和键盘部份的电路图

三、实例调试

以下以一个稍复杂的程序为例，说明键盘、LED 显示实验仿真板的使用。该程序实现的是可控流水灯，接 P3.2 的键为开始键，按此键则灯开始流动（由上而下），接 P3.3 的键为停止键，按此键则停止流动，所有灯暗，接 P3.4 的键为向上键，按此键则灯由上向下流动，接 P3.5 的键为向下键，按此键则灯由下向上流动。

例 8：

```
UpDown      BIT  00H ;上下行标志
StartEnd    BIT  01H ;起动及停止标志
LAMPCODE    EQU      21H ;存放流动的数据代码
        ORG      0000H
        AJMP     MAIN
        ORG      30H
MAIN:
        MOV      SP,#5FH
        MOV      P1,#0FFH
        CLRUpDown        ;启动时处于向上的状态
```

```
        CLR StartEnd  ;启动时处于停止状态
        MOV     LAMPCODE,#01H;单灯流动的代码
LOOP:   ACALL   KEY       ;调用键盘程序
        JNB F0,LNEXT     ;如果无键按下,则继续
        ACALL   KEYPROC          ;否则调用键盘处理程序
LNEXT:  ACALL   LAMP    ;调用灯显示程序
        AJMP    LOOP          ;反复循环,主程序到此结束
;延时程序,键盘处理中调用
DELAY:  MOV     R7,#100
D1:  MOV     R6,#100
        DJNZ    R6,$
        DJNZ    R7,D1
        RET
KEYPROC:
        MOV     A,B       ;从 B 寄存器中获取键值
        JB   ACC.2,KeyStart     ;分析键的代码,某位被按下,则该位为 1
        JB   ACC.3,KeyOver
        JB   ACC.4,KeyUp
        JB   ACC.5,KeyDown
        AJMP    KEY_RET
KeyStart:
        SETB    StartEnd  ;第一个键按下后的处理
        AJMP    KEY_RET
KeyOver:
        CLR StartEnd  ;第二个键按下后的处理
        AJMP    KEY_RET
KeyUp:
        SETB    UpDown       ;第三个键按下后的处理
        AJMP    KEY_RET
KeyDown:
        CLR UpDown       ;第四个键按下后的处理
KEY_RET:
        RET
KEY:
        CLR F0   ;清 F0,表示无键按下。
        ORL     P3,#00111100B     ;将 P3 口的接有键的四位置 1
        MOV     A,P3      ;取 P3 的值
        ORL     A,#11000011B      ;将其余 4 位置 1
        CPL A    ;取反
        JZ   K_RET   ;如果为 0 则一定无键按下
        CALL    DELAY  ;否则延时去键抖
        ORL     P3,#00111100B
        MOV     A,P3
        ORL     A,#11000011B
```

```
        CPL A
        JZ   K_RET
        MOV     B,A ;确实有键按下,将键值存入 B 中
        SETB    F0  ;设置有键按下的标志
    ;以下的代码是可以被注释掉的，如果去掉注释，就具有判断键是否释放的功能，否则没有
    K_RET:  ;ORL    P3,#00111100B    ;此处循环等待键的释放
        ;MOV    A,P3
        ;ORL    A,#11000011B
        ;CPL        A
        ;JZ     K_RET1          ;读取的数据取反后为 0 说明键释放了
        ;AJMP   K_RET
        ;K_RET1:CALL    DELAY           ;消除后沿抖动
        RET
    D500MS:         ;流水灯的延迟时间
        MOV     R7,#255
    D51:    MOV     R6,#255
        DJNZ    R6,$
        DJNZ    R7,D51
        RET
    LAMP:
        JB   StartEnd,LampStart ;如果 StartEnd=1,则启动
        MOV     P1,#0FFH
        AJMP    LAMPRET    ;否则关闭所有显示,返回
    LampStart:
        JB   UpDown,LAMPUP      ;如果 UpDown=1,则向上流动
        MOV     A,LAMPCODE
        RL  A       ;实际就是左移位而已
        MOV     LAMPCODE,A
        MOV     P1,A
        LCALL   D500MS
        LCALL   D500MS
        AJMP    LAMPRET
    LAMPUP:
        MOV     A,LAMPCODE
        RR  A   ;向下流动实际就是右移
        MOV     LAMPCODE,A
        MOV     P1,A
        LCALL   D500MS
    LAMPRET:
        RET
        END
```

将程序输入并建立工程文件，设置工程文件，在 Debug 标签页中加入“ -dledkey ”，汇编、连接文件，按 Ctrl+F5 开始调试，打开仿真板，使用 F5 功能键全速运行，可以看到所

有灯均不亮，点击最上面的按钮，立即会看到灯流动起来了，点击第二个按键，灯将停止流动，再次点击第一个按钮，使灯流动起来，点击第三个按钮，可以发现灯流动的方向变了，点击第四个按钮，灯的流动方向又变回来了。如果没有出现所描述的现象，可以使用单步、过程单步等调试手段进行调试，在进行调试时实验仿真板会随时显示出当前的情况，是不是非常的直观和方便呢？

下面的一个例子是关于第二块实验仿真板的，演示点亮 8 位数码管。例 9：

```
        ORG     0000h
        JMP      MAIN
        ORG     30H
MAIN:
        MOV     SP,#5FH
        MOV     R1,#08H
        MOV     R0,#58H      ;显示缓冲区首地址
        MOV     A,#2
INIT:
        MOV     @R0,A        ;初始化显示缓冲区
        INC     A
        INC     R0
        DJNZ    R1,INIT      ;将 0-7 送显示缓冲区
LOOP:
        CALL    DISPLAY
        JMP     LOOP
;主程序到此结束
DISPLAY:
        MOV     R0,#7FH      ;列选择
        MOV     R7,#08H      ;共有 8 个字符
        MOV     R1,#58H      ;显示缓冲区首地址
AGAIN:
        MOV     A,@R1
        MOV     DPTR,#DISPTABLE
        MOVC    A,@A+DPTR
        MOV     P0,A
        MOV     P2,R0
        MOV     A,R0
        RR      A
        MOV     R0,A
        INC     R1
        DJNZ    R7,AGAIN
        RET
DISPTABLE:   DB 0c0h,0f9h,0a4h,0b0h,99h,92h,82h,0f8h,80h,90h,0FFH     ;字形码表
END
```

这一程序内部 RAM 中 58H 到 5FH 被当成是显示缓冲区，主程序中用 2-9 填充该显示区，然后调用显示程序显示 2-9。这里是用了最简单的逐位显示的方式编写的显示程序。

最后介绍一个小小技巧，将鼠标移入按钮区域，按下左键，按钮显示被按下，不要放开

鼠标左键，将光标移出按钮区域，松开左键，可以看到，按钮仍处于按下状态，利用这一功能，在需要 I/O 口长期处于低电平时，你就不必一直用手按着鼠标的左键啦。