PIC单片机I/O引脚适合直接驱动的是纯电阻性负载，如点亮发光二极管，通过限流电阻驱动三极管的基极等。I/O引脚一般不能直接用于驱动感性负载，如小型继电器。感性负载在关断时会产生极高的反向感应电压，这对I/O引脚是致命的，就算用户在外面加了续流二极管保护，遗憾的是二极管非理想元器件，它有一定的导通延时，无法抑制瞬间的高压，故驱动感性负载时在单片机和负载间要加三极管或集成芯片隔离驱动。I/O引脚也很少直接驱动场效应管，因为场效应管的栅极实际上是一个电容负载。若要控制场效应管的快速通断，必须让栅极电容快速充满电压或放完电压，这需要瞬间的大电流驱动，而这在PIC单片机的I/O引脚上是绝对提供不了的。驱动场效应管一般都用I/O引脚通过专门的驱动芯片实现。

5.1.5 端口的“读—修改—写”问题

在前面的5.1.3小节已经介绍了单片机读端口是直接读引脚上的电平，在引脚作为输出时理想情况下是输出什么状态就应该读到什么状态。但经常由于外围硬件电路设计的缺陷或因为特殊的引脚配置而不能读回与原输出相一致的结果。在对端口的单个引脚进行位设置时，这个问题就会集中暴露出来。

我们先来回顾一下PIC单片机的位操作指令，总共只有4条：2条是位判断指令BTFSC/BTFSS；另外2条是位清0“BCF”和置1“BSF”指令。关键的后两条位清0/置1指令在执行时分成3个步骤：

(1) 读——把整个寄存器的内容读到内部ALU单元；

(2) 修改——以实际读到的数据为基础，按指令对中间的某一个数据位清0或置1；

(3) 写——把修改后的值的整个字节写回到端口锁存器再输出。

由此可见，看似单单对某一个位的操作，实际上还牵涉到了同一个寄存器的其他数据位被读取和写回，这就是通常所说的“读—修改—写”过程，实际上对任何寄存器的任何操作都是这样的一个过程，不仅仅限于端口寄存器。对内部RAM寄存器而言，读出的数据一定是上一次写入的内容，所以无需太多关心这样一个过程。但对输出端口而言，由于存在外围电路的影响，读的又是当前引脚上的电平，所以读进来的数据完全有可能不是原先输出的逻辑状态。所以，我们在介绍端口时特别要强调“读—修改—写”这一问题。

我们以实际电路为例进一步加深理解端口的“读—修改—写”。如图5-2所示电路，设计要求PORTB的7/6/5这3个引脚输出高电平时分别驱动继电器吸合和发光二极管点亮。在初始化相关引脚为输出状态后，有如下的程序过程。

(1) bsf PORTB, 7

目的：RB7输出高电平，吸合继电器。此时有电流流经NPN三极管基极，三极管导通，

继电器能够吸合，似乎一切在按设计要求进行。但注意，由于三极管 BE 结的钳位作用，此时 RB7 上只有 0.7 V 的电压。

(2) bsf PORTB, 6

目的：RB6 输出高电平，点亮 LED1，且继电器保持吸合状态。但实际结果是继电器将因为这条指令的执行而被断开！在对 RB6 置 1 时，PORTB 整个端口的 8 位数据都被读回，此时因为 RB7 上只有 0.7 V 电压，所以读回时 RB7＝0，在处理完 RB6 后再整个字节写出去时，RB7 就真的变成 0 输出了，继电器将立即断开。另外，注意此引脚外围电容 C_1 为 PCB 长走线引入的分步电容，由于此电容的存在，RB6 上的电压将缓慢上升，当然这是相比指令执行的速度而言。

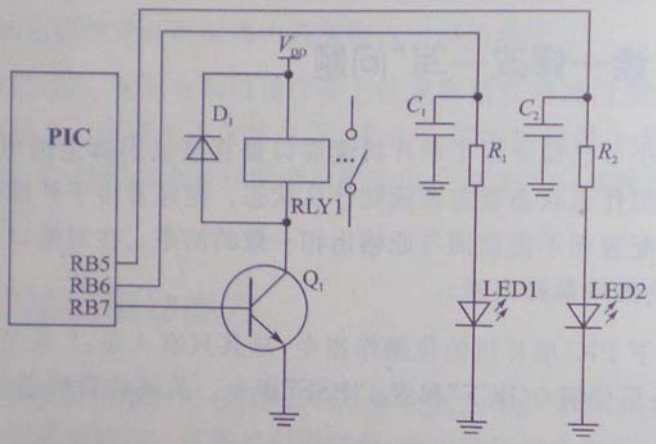

图 5－2 引发端口“读—修改—写”问题的不良设计

(3) bsf PORTB, 5

目的：RB5 输出高电平，点亮 LED2，维持前面的输出状态。但实际上继电器早就因为上一条指令的执行而断开了。再看 LED1 控制的问题，由于 PCB 走线分布电容 C_1 的存在，RB6 上的高电压建立需要一定的时间，如果现在这条指令紧接着上面第(2)条指令运行的话，整个端口被读回时 RB6 也是 0，故(2)和(3)这 2 条指令连续运行时将根本看不到 LED1 被点亮的情形。这样 3 条指令连续运行的最终结果是只有 LED2 被点亮，完全有悖设计初衷。究其根本原因就是因为输出的状态没有能够正确读回。

那么如何克服端口的这一“读—修改—写”问题？最根本的解决办法是硬件设计一定要合理可靠。把图 5－2 的不良设计稍加改正，如图 5－3 所示，RB7 通过限流电阻 R_3 再接到三极管基极，PCB 布线时 $R_1/R_2/R_3$ 这些电阻尽量靠近单片机，避免走线电容使引脚输出的电压滞后建立。这样基本上就可以消除输出结果不能立即读回的现象。

如果硬件设计已经固定，那么在软件中克服“读—修改—写”出现的问题有两个可选的

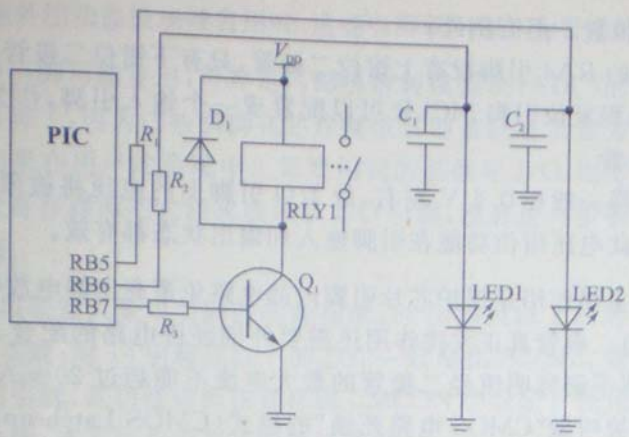

图 5-3 克服端口"读—修改—写"问题的改良设计

办法。一是对端口操作后加入足够长的延时，例如上面举例的第(2)条指令执行后，插入几条延时指令，再执行第(3)条指令，这样就可以看到 LED1 和 LED2 同时被点亮的结果。显而易见，对图 5-2 中 RB7 这样的电路设计缺陷单靠简单的延时是无法解决问题的。此时惟一的解决办法就是在软件中使用端口的"影子"寄存器，先在影子寄存器中完成所有的设定，然后整个字节一起复制到端口寄存器输出，如代码范例 5-1 所列。

例 5-1 使用影子寄存器解决"读—修改—写"问题

```
bsf     PORTB_SHADOW,7      ;影子寄存器第 7 位置 1
bsf     PORTB_SHADOW,6      ;影子寄存器第 6 位置 1
bsf     PORTB_SHADOW,5      ;影子寄存器第 5 位置 1
movf    PORTB_SHADOW,w
movwf   PORTB               ;复制影子寄存器到 PORTB
```

这样处理后，就算是图 5-2 这样的不良设计，也可以看到所希望的输出结果：继电器吸合，LED1 和 LED2 同时点亮。但这只是弥补硬件缺陷的不得已而为之，不是治本的办法。

在系统调试时，如果单步运行控制结果正常，全速运行失败，请首先检查是否存在"读—修改—写"的问题；如果对一个端口中的某引脚做一次操作时发现同一个端口内的其他引脚跟着变化，还是先请检查是否存在"读—修改—写"的问题，而不用着急怀疑仿真工具或芯片的质量。

5.1.6 引脚端口的 ESD 保护

几乎所有的 PIC 单片机 I/O 引脚内部都有钳位保护二极管，图 5-1 已经清楚地显示了