

## 第八十节： 单片机 IO 口驱动 LED。

### 【80.1 不再依赖第 11 节模板程序。】

前面大量的章节主要是讲 C 语言本身的基础知识，因此每次的练习例程都要依赖第 11 节的模板程序。从本节开始，正式进入到单片机主题，如果没有特殊说明，以后的练习程序就不再需要依赖第 11 节模板程序，可以脱离模板单飞了。

### 【80.2 寄存器。】

寄存器是跨越在软件与硬件之间的桥梁，单片机的 C 语言想控制单片机引脚输出 0V 或者 5V 的物理电压，本质就是通过往寄存器里填数字，往哪个寄存器填数字，填什么样的数字，对应的引脚就输出什么样的电压。至于“为什么往寄存器填数字就会在引脚上输出对应的电压”这个问题，对于我们“应用级”工程师来说是一个黑匣子。我们写软件的最底层就是操作到“寄存器”这个层面，至于“寄存器与物理电压之间是如何关联如何实现”的这个问题，其实是“芯片级”半导体工程师所研究的事，因为单片机本身其实就是一个成品，我们从“芯片级”半导体工程师那里拿到这个成品，这个成品的说明书告诉了我们该成品的每个寄存器的作用，我们只能在这个基础上去做更上层的应用。该说明书其实就是大家通常所说的芯片的 datasheet。

寄存器在单片机 C 语言层面，是一个全局变量，是一个具备特定名字的全局变量，是一个被系统征用的全局变量。寄存器的名字就像古代皇帝的名字，所有普通老百姓的变量名字都要“避尊者讳”，不能跟寄存器的名字重名，否则 C 编译器就编译不通过。

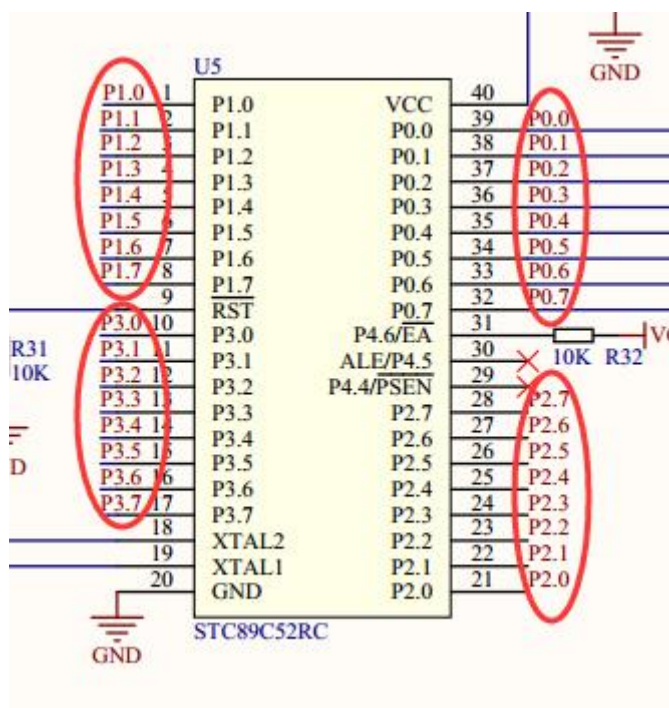


图 80.2.1 单片机的 32 个 IO 口引脚

本教程用的 STC89C52 单片机 IO 口寄存器有 4 个，分别是 P0, P1, P2, P3 这 4 个寄存器，每个寄存器都是一个 8 位的全局变量，每一位代表控制一个单片机的 IO 口引脚，因此，该单片机一共有 32 个（4 乘以 8）IO 口引脚，每个引脚都是可以单独控制的（俗称位操作）。往该位填入 0，对应的引脚就输出 0V 的物理电

压。往该位填入 1，对应的引脚就输出 5V 的物理电压。

### 【80.3 C 语言操作 I/O 口寄存器。】

C 语言操作单片机 I/O 口寄存器，以便在对应的引脚上输出对应的物理电压，有两种方式。一种是并口的方式，另外一种是指操作的方式。并口方式，一次操作 8 个位（8 个引脚），往往用在并口数据总线上。位操作方式，一次操作 1 个位（1 个引脚），该方式因为单独控制到某个引脚，所以应用更加灵活广泛。

并口方式。并口方式的时候，可以直接对 P0, P1, P2, P3 这 4 个寄存器赋值，就像对一个 unsigned char 的全局变量赋值一样。比如：

```
#include "REG52.H"
void main()
{
    P0=0xF0; //直接对 P0 赋值 0xF0，意味着 P0 口的 8 个引脚，高 4 位全部输出 5V，低 4 位全部输出 0V。
    while(1)
    {
    }
}
```

“P0=0xF0”这行代码，把十六进制的 0xF0 分解成二进制 11110000 来理解，P0.7，P0.6，P0.5，P0.4 这 4 个引脚分别输出 5V 物理电压，而 P0.3，P0.2，P0.1，P0.0 这 4 个引脚分别输出 0V 物理电压。

位操作方式。并口方式因为一次操作就绑定了 8 个引脚，是非常不方便的，因此，位操作就显得特别灵活实用，你可以直接操作 P0，P1，P2，P3 这 4 组引脚中（共 32 个）的某 1 个引脚，而不影响其它引脚的状态。比如，P1.4 引脚是属于 P1 组的 8 个引脚中的某 1 个引脚，如果想直接位操作 P1.4 引脚，要用到特定的关键词 sbit 和符号“^”这个组合，sbit 和符号“^”的组合类似宏定义，使用方式如下。

```
#include "REG52.H"
sbit P1_4=P1^4; //利用 sbit 和符号“^”的组合，把变量名字 P1_4 与 P1.4 引脚关联起来
void main()
{
    P1_4=0; //P1.4 引脚输出 0V 物理电压，而不影响其它 P1 口引脚的状态。
    while(1)
    {
    }
}
```

### 【80.4 点亮 LED。】

LED 灯要有电流通过，才会发光。要有电流通过，必须要有电压的“正压差”，“压差”可以用水压来比喻。

比如在 2 楼的水，对于 1 楼来说，它就有“正压差”（2 减去 1 等于“正 1”），因此只要构成回路（有水管），2 楼的水是可以往 1 楼流动的。

比如在 2 楼的水，对于 3 楼来说，它虽然有压差，但是有的只是“负压差”（2 减去 3 等于“负 1”），因此哪怕构成回路（有水管），2 楼的水也是不可以往 3 楼流动的。

比如在 2 楼的水，对于同楼层的 2 楼来说，它的压差是 0 压差（2 减去 2 等于“0 压差”），因此哪怕构成回路（有水管），2 楼的水也是不可以在 2 楼之间流动的。

上面三个比喻很关键，精髓在于是否有“正压差”。要点亮一个 LED 灯，并不是说你单片机引脚直接输出一个 5V 的物理电压就能点亮的，还要看它构成的整个 LED 灯回路，也就是实际的电路图是什么样的。在本教程的原理图中，我们点亮 LED 灯是采样“灌入式”的电路，也就是单片机输出 5V 电压的时候 LED 灯是熄灭的，而输出 0V 物理电压时 LED 灯反而是被点亮的。如下两个图：

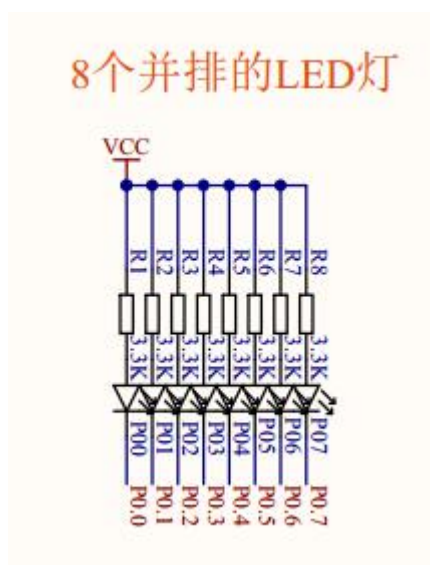


图 80. 4. 1 灌入式驱动 8 个 LED

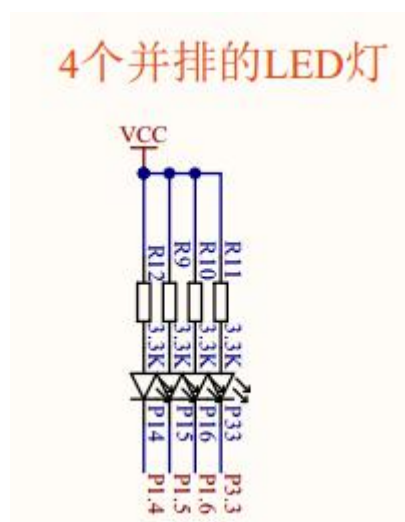


图 80. 4. 2 灌入式驱动 4 个 LED

现在根据这原理图，编写一个并口和位操作的练习例子，直接把程序烧录进开发板，就可以看到对应的 LED 灯的状态。

```
#include "REG52.H"
sbit P1_4=P1^4; //利用 sbit 和符号“^”的组合，把变量名字 P1_4 与 P1.4 引脚关联起来
void main()
{
    P0=0xF0; //直接对 P0 赋值 0xF0，意味着 P0 口的 8 个引脚，高 4 位全部输出 5V，低 4 位全部输出 0V。
    P1_4=0; //P1.4 引脚输出 0V 物理电压，而不影响其它 P1 口引脚的状态。
    while(1)
    {
    }
}
```

现象分析：

“P0=0xF0”直接对 P0 赋值 0xF0，意味着 P0 口的 8 个引脚，高 4 位全部输出 5V（LED 灯反而灭），低 4 位全部输出 0V（LED 灯反而被点亮）。

“P1\_4=0”P1.4 引脚输出 0V 物理电压（LED 灯反而被点亮）。