Arduino板上通过操作端口寄存器来进行控制

 sudo-wang 2019-02-27 20:45:43 5524 已收藏 15

分类专栏： Arduino

端口寄存器允许在Arduino板上更低级和更快地操纵微控制器的i / o引脚。Arduino板上使用的芯片（ATmega8和ATmega168）有三个端口：

B（数字引脚8到13）
C（模拟输入引脚）
D（数字引脚0到7）
每个端口由三个寄存器控制，这三个寄存器也是arduino语言中定义的变量。DDR寄存器确定引脚是INPUT还是OUTPUT。PORT寄存器控制引脚是高电平还是低电平，PIN寄存器通过pinMode（）读取设置为输入的INPUT引脚的状态。**ATmega8** 和 **ATmega168** 芯片的地图显示了端口。较新的Atmega328p芯片完全遵循Atmega168的引脚排列。

DDR和PORT寄存器可以写入和读取。PIN寄存器对应于输入状态，只能读取。

PORTD映射到Arduino数字引脚0到7

- DDRD - 端口D数据方向寄存器 - 读/写
- PORTD - 端口D数据寄存器 - 读/写
- PIND - 端口D输入引脚寄存器 - 只读

PORTB映射到Arduino数字引脚8到13两个高位（6和7）映射到晶体引脚，不可用

- DDRB - 端口B数据方向寄存器 - 读/写
- PORTB - 端口B数据寄存器 - 读/写
- PINB - 端口B输入引脚寄存器 - 只读

PORTC映射到Arduino模拟引脚0到5.引脚6和7只能在Arduino Mini上访问

- DDRC - 端口C数据方向寄存器 - 读/写
- PORTC - 端口C数据寄存器 - 读/写
- PINC - 端口C输入引脚寄存器 - 只读

这些寄存器的每一位对应一个引脚; 例如，DDRB，PORTB和PINB的低位指的是引脚PB0（数字引脚8）。有关Arduino引脚编号到端口和位的完整映射，请参见芯片图：ATmega8，ATmega168。（注意，端口的某些位可用于除i / o之外的其他位;请注意不要更改与它们对应的寄存器位的值。）

例子
参考上面的引脚图，PortD寄存器控制Arduino数字引脚0到7。

但是，您应该注意，引脚0和1用于串行通信以编程和调试Arduino，因此通常应避免更改这些引脚，除非需要串行输入或输出功能。请注意，这可能会干扰程序下载或调试。

DDRD是端口D的方向寄存器（Arduino数字引脚0-7）。该寄存器中的位控制PORTD中的引脚是否配置为输入或输出，例如：

```
DDRD = B11111110; //将Arduino引脚1至7设置为输出，将引脚0设置为输入
DDRD = DDRD | B11111100; //这样更安全，因为它将引脚2到7设置为输出
                         //不改变引脚0和1的值，即RX和TX
```

PORTD是输出状态的寄存器。例如;

```
PORTD = B10101000; // sets digital pins 7,5,3 HIGH
```

如果使用DDRD寄存器或pinMode（）将引脚设置为输出，则只能在这些引脚上看到5伏电压。

PIND是输入寄存器变量它将同时读取所有数字输入引脚。

为何使用端口操作？（来自Bitmath教程）

一般来说，做这类事并不是一个好主意。为什么不？原因如下：

代码对于调试和维护来说要困难得多，并且对于其他人来说理解起来要困难得多。处理器执行代码只需几微秒，但您可能需要几个小时才能弄清楚它为什么不能正常工作并修复它！你的时间很宝贵，对吗？但计算机的时间非常便宜，以你喂它的电费来衡量。通常以最明显的方式编写代码要好得多。
代码不太便携。如果使用digitalRead（）和digitalWrite（），编写将在所有Atmel微控制器上运行的代码要容易得多，而控制端口寄存器和端口寄存器在每种微控制器上都可以不同。
通过直接端口访问可以更容易地导致意外故障。注意线DDRD = B11111110; 上面提到必须将引脚0作为输入引脚。引脚0是串行端口上的接收线（RX）。通过将引脚0更改为输出引脚，很容易意外地导致串口停止工作！现在当你突然无法接收串行数据时会非常混乱，不是吗？
所以你可能会对自己说，很好，为什么我会想要使用这些东西呢？以下是直接端口访问的一些积极方面：

您可能需要能够非常快速地打开和关闭引脚，这意味着在几微秒内。如果你看一下lib / targets / arduino / wiring.c中的源代码，你会看到digitalRead（）和digitalWrite（）都是大约十几行代码，它们被编译成很多机器指令。每个机器指令需要一个16MHz的时钟周期，这可以累积在时间敏感的应用中。直接端口访问可以在更少的时钟周期内完成相同的工作。
有时您可能需要在同一时间设置多个输出引脚。调用digitalWrite（10，HIGH）; 接着是digitalWrite（11，HIGH）; 这将导致引脚10在引脚11之前几微秒变为高电平，这可能会混淆您连接的某些对时间敏感的外部数字电路。或者，您可以使用PORTB | = B1100将两个引脚设置为完全相同的时刻;
如果程序内存不足，可以使用这些技巧使代码更小。它需要更少的编译代码字节来同时通过端口寄存器同时写入一堆硬件引脚，而不是使用for循环分别设置每个引脚。在某些情况下，这可能会使您的程序适合闪存或不兼容！